

Appendix A - Basic Outline and Use of Structured Query Language (SQL)

This is an introduction to the syntax and applied use of SQL with specific reference to the ORMGP database. SQL allows users to access (and manipulate) data from relational databases (or more formally known as a 'Relational Database Management System' - RDBMS) in a fairly straightforward manner using a language-natural syntax without (with the exception of advanced queries) having to know computer programming. Almost all cases can be considered to use 'standard' SQL (also referred to as 'ANSI' SQL) that should be 'mostly' applicable across a number of relational database software (e.g. Microsoft SQL Server, Microsoft Access, Oracle, MySQL, SQLITE, etc ...). Note that there are slight differences in the supported syntax between these packages. In addition, syntax described below may also be applicable across multiple SQL statement types (instead of just within the example given).

Note that all SQL syntax will be shown in capitals. Comments and placeholders will be shown in lowercase. Single quotes (i.e. 'example') are used where text is being examined - note that in some software, double quotes (i.e. "example") would be used instead.

A.1 Select Statement

Select Statement - Overview

SELECT is the most common statement used within a relational database, allowing the user to 'select' specific information from an existing table (or tables). The general syntax is of the form

```
SELECT <comma-delimited list of fields> FROM <a single table name>
```

Examining the contents of the D_LOCATION table would require

```
SELECT * FROM D_LOCATION
```

which would return all the fields and all the rows within the D_LOCATION table. Note the use of the '*' wildcard, this allows the user to allow selection of all existing fields without having to list the fields individually, as

```
SELECT  
LOC_ID,  
LOC_NAME,  
LOC_NAME_ALT1,  
LOC_TYPE_CODE,  
OWN_ID,  
...  
FROM  
D_LOCATION
```

where the '...' indicates the remainder of the fields (this is not valid SQL syntax, just a placeholder for descriptive purposes). Note that the fields are comma-delimited and are

listed across multiple lines - line spacing (and additional blank space on lines themselves) does not affect the query and can be used as a formatting tool (enabling ease of viewing).

You likely would not want to examine all rows from a particular table at once. If you were assembling a query (i.e. putting a complicated query together based upon succeeding steps) you could choose to examine only the first few records returned using the TOP keyword.

```
SELECT TOP 1000 LOC_ID,LOC_NAME FROM D_LOCATION
```

This extracts the location identifier and associated location name from the D_LOCATION table but limits the number of rows returned to the first 1000. The order the rows are returned should not be relied upon - the means for specifying a particular order will be outlined in a subsequent section.

In many cases, the user is looking for particular information. For example, if you wanted to see the names associated with a particular location in the database, given a LOC_ID of '-828830263', then the query

```
SELECT * FROM D_LOCATION WHERE LOC_ID = -828830263
```

would return all fields but only a single row from the D_LOCATION table - the row matching the provided location identifier. Other operators, instead of using '=' (equal to) as the test operator, can also be used including

- Not equal (!=)
- Less than (<)
- Greater than (>)
- Less than or equal to (<=)
- Greater than or equal to (>=)

So a user could determine all those boreholes (from the D_BOREHOLE table) with a BH_BOTTOM_DEPTH of 20m by

```
SELECT * FROM D_BOREHOLE WHERE BH_BOTTOM_DEPTH > 20
```

The name fields (e.g. LOC_NAME, LOC_NAME_ALT1, etc ...) can also be used to extract information about a particular location (stored in D_LOCATION). If only a name was known

```
SELECT * FROM D_LOCATION WHERE LOC_NAME='Port Perry OW 05/3'
```

would pull the same information as using the LOC_ID (above) of '-828830263'. However, searching through text fields is different from numeric in that rarely do the text fields match the exact text the user enters. Instead of an exact match, the user - when looking for location names containing certain text - can use the LIKE keyword

```
SELECT * FROM D_LOCATION WHERE LOC_NAME LIKE '%UGAIS%'
```

This extracts all rows (and fields) of information where the location name contains, anywhere within its name, the 'UGAIS' key. Note the use of the '%' signs (in Microsoft Access and other packages, a '*' can be used instead) as part of the quoted text - these are text wildcards specifying that any number of characters can be in front of the 'UGAIS' text and that any number of characters can take place after it. This includes the case of no characters before and/or after.

Multiple checks can be made (after WHERE) against the table. This is accomplished through additional boolean (or logical) operators.

```
SELECT
*
FROM D_LOCATION
WHERE
LOC_NAME LIKE '%UGAIS%' AND LOC_ORIGINAL_NAME LIKE 'TR%
```

In this case, we're specifying two checks - one against the LOC_NAME field, another against the LOC_ORIGINAL_NAME field. For the latter, notice that the '%' character only appears after the 'TR', indicating this text must be placed at the beginning of the name. The boolean operator used is AND - both checks must be true, i.e. 'UGAIS' must appear in LOC_NAME and 'TR' must be at the beginning of the LOC_ORIGINAL_NAME. Another boolean operation is OR, as in

```
SELECT
*
FROM D_LOCATION
WHERE
LOC_NAME LIKE '%UGAIS%' OR LOC_NAME LIKE '%OGS'
```

where the rows returned from D_LOCATION depend on whether the 'UGAIS' or the 'OGS' text appear as part of LOC_NAME. To return to numeric comparisons, if we wanted to find all boreholes from D_BOREHOLE with BH_BOTTOM_DEPTH of greater than 20m but less than or equal to 40m the following could be used

```
SELECT
*
FROM D_BOREHOLE
WHERE
BH_BOTTOM_DEPTH > 20 AND BH_BOTTOM_DEPTH <= 40
```

More complicated queries can be built by combining multiple checks on values within a table, for example the following

```
SELECT
*
FROM
D_LOCATION
WHERE
(LOC_COORD_EASTING >= 620000 AND
LOC_COORD_EASTING <=640000)
```

```

AND
(LOC_COORD_NORTHING >= 4831000 AND
LOC_COORD_NORTHING <= 4841000)
AND
(LOC_NAME LIKE '%UGAIS%' OR LOC_NAME LIKE '%OGS%')

```

where we are searching for all locations within a designated spatial area (using coordinates; i.e. LOC_COORD_EASTING and LOC_COORD_NORTHING) as well as extracting only those locations which include 'UGAIS' or 'OGS' in their names.

Select Statement - IN condition

Multiple OR statements can be used to find all locations given a particular characteristic, like any location that could have geological information

```

SELECT LOC_ID,LOC_NAME,LOC_NAME_ALT1,LOC_TYPE_CODE
FROM D_LOCATION
WHERE
LOC_TYPE_CODE=1
OR LOC_TYPE_CODE=7
OR LOC_TYPE_CODE=11
OR LOC_TYPE_CODE=17
OR LOC_TYPE_CODE=18
OR LOC_TYPE_CODE=19

```

where the LOC_TYPE_CODES (refer to R_LOC_TYPE_CODE) are

- Well or Borehole (1)
- Testpit (7)
- Outcrop (11)
- Geological Section (17)
- Oil and Gas Well (18)
- Bedrock Outcrop (19)

However, this statement can be simplified by using the IN statement which allows the possible values the user is checking against to be listed, as in

```

SELECT LOC_ID,LOC_NAME,LOC_NAME_ALT1,LOC_TYPE_CODE
FROM D_LOCATION
WHERE
LOC_TYPE_CODE IN (1,7,11,17,18,19)

```

which is shorter and much easier to understand.

The IN condition also allows us to combine SELECT queries (instead of using, for example, JOINS - as described below). The first SELECT query (actually occurring at the end of the 'full' query) would come up with a list of single values

```

SELECT LOC_ID FROM D_BOREHOLE

```

Here we're getting a list of all LOC_ID (i.e. locations) that are found in the D_BOREHOLE table. This is approximately equivalent to using the LOC_TYPE_CODE in the previous example. Now that we have a list of LOC_ID's, we can pull their names and type codes from D_LOCATION except, this time, we're using the IN keyword to include the first SELECT statement

```
SELECT LOC_ID,LOC_NAME,LOC_NAME_ALT1,LOC_TYPE_CODE
FROM D_LOCATION
WHERE
LOC_ID IN
(
  SELECT LOC_ID FROM D_BOREHOLE
)
```

This should return equivalent results to that of using the LOC_TYPE_CODE earlier. Again note that, for this type of use, the first SELECT statement (i.e. the one in brackets) should only return a list of a single value (otherwise an SQL error will be returned).

Select Statement - BETWEEN condition

The BETWEEN condition can be used for comparisons for a range of values. Thus, to simplify a previous example

```
SELECT
*
FROM
D_LOCATION
WHERE
LOC_COORD_EASTING BETWEEN 620000 AND 640000
AND
LOC_COORD_NORTHING BETWEEN 4831000 AND 4841000
AND
(LOC_NAME LIKE '%UGAIS%' OR LOC_NAME LIKE '%OGS%')
```

with similar results.

Select Statement - Joins

In a relational database, only certain information on a particular object/entity would be contained in a single table (e.g. D_LOCATION containing basic locational information including name, coordinates, location type, etc ...). Additional information would then be stored in additional tables (dependent upon the particular location). For example, borehole locations would have information stored in D_BOREHOLE, including depths and elevations of the top and bottom of the hole, the start and ending dates of drilling, etc ... A borehole location might also have a screen - that information would be stored in other tables including D_INTERVAL (the interval name and type), D_INTERVAL_MONITOR (the top and bottom of the screen), etc ... The information in the tables would be linked (i.e. related) by keys (i.e. primary keys where each row in the table can be distinguished between based upon the key, each being unique). Examples of primary keys in the database include LOC_ID (found in D_LOCATION,

D_BOREHOLE, etc ...) and INT_ID (found in D_INTERVAL, D_INTERVAL_MONITOR, etc ...). Means of combining information from multiple tables, based upon these primary keys, are called Joins. We'll examine two types - inner joins and outer joins

Select Statement - Inner Joins

All joins use the primary key to relate tables. Inner joins require that this key exists in both tables being referenced. For example, if we wanted the location information (found in D_LOCATION) and borehole elevation and depth for any borehole (in D_BOREHOLE) with depths greater than 20m, our query would look like

```
SELECT
dloc.LOC_ID,
LOC_NAME,
LOC_NAME_ALT1,
LOC_NAME_ORIGINAL,
BH_GND_ELEV,
BH_BOTTOM_DEPTH
FROM
D_BOREHOLE AS dbore
INNER JOIN
D_LOCATION AS dloc
ON
dbore.LOC_ID = dloc.LOC_ID
```

where we're 'joining' our two tables (using the INNER JOIN keywords) based upon the presence of LOC_ID (which is our primary key) in each table (note that in the table we're relating, it's referred to as the foreign key). We've also introduced another concept here, the AS keyword. As some of the fields we're using occur in both tables (namely, in this case, LOC_ID) we should specify which one we're actually referencing/using in the query - the AS keyword allows us to use an alias to reference the table (and also fields, but this is not shown here) using a different name. Here, D_BOREHOLE is aliased as 'dbore' and D_LOCATION is aliased as 'dloc'. This allows us to be clear regarding the source of the fields as well as shortening the names required in the query itself (using 'Dot Notation', i.e. having a 'period' between the name of the table and the name of the field).

Multiple tables can be joined at the same time. If we wanted to include interval data as part of our, above, query then we could use

```
SELECT
dloc.LOC_ID,
dloc.LOC_NAME,
dloc.LOC_NAME_ALT1,
dloc.LOC_ORIGINAL_NAME,
dbore.BH_GND_ELEV,
dbore.BH_BOTTOM_DEPTH,
dint.INT_ID,
dint.INT_NAME
```

```

FROM
D_BOREHOLE AS dbore
INNER JOIN
D_LOCATION AS dloc ON dbore.LOC_ID = dloc.LOC_ID
INNER JOIN
D_INTERVAL AS dint ON dloc.LOC_ID = dint.LOC_ID

```

Note that we've now specified exactly (using the <table>.<field> syntax) which field is coming from which table.

Select Statement - Outer Joins

Outer joins are used when we're unsure whether the key will appear in one of the tables. Outer joins are different in that they're specified using a LEFT or RIGHT keyword - this stresses which of the two tables (i.e. the left or right table in the query) is the one we're using as the base (i.e. which has the key). Using our previous example, if we wanted to include the bedrock elevation for each borehole, knowing that some (most?) boreholes will not actually have bedrock values we would

```

SELECT
dloc.LOC_ID,
dloc.LOC_NAME,
dloc.LOC_NAME_ALT1,
dloc.LOC_ORIGINAL_NAME,
dbore.BH_GND_ELEV,
dbore.BH_BOTTOM_DEPTH,
dint.INT_ID,
dint.INT_NAME,
vbed.BEDROCK_ELEV
FROM
D_BOREHOLE AS dbore
INNER JOIN
D_LOCATION AS dloc ON dbore.LOC_ID = dloc.LOC_ID
INNER JOIN
D_INTERVAL AS dint ON dloc.LOC_ID = dint.LOC_ID
LEFT OUTER JOIN
V_GEN_BOREHOLES_BEDROCK as vbed ON vbed.LOC_ID = dbore.LOC_ID

```

Select Statement - Calculations (Arithmetic Operations) and Mathematical Functions

In many cases, a user would wish to combine values in some sort of calculation and return the result of that calculation as part of the query. In the previous query, for example, instead of returning just the borehole bottom depth (i.e. BH_BOTTOM_DEPTH) we may wish to calculate the elevation at the bottom of the borehole as well as the depth at which the bedrock surface is encountered. In which case we would change the above to

```

SELECT
dloc.LOC_ID,
dloc.LOC_NAME,
dloc.LOC_NAME_ALT1,

```

```

dloc.LOC_ORIGINAL_NAME,
dbore.BH_GND_ELEV,
dbore.BH_BOTTOM_DEPTH,
(dbore.BH_GND_ELEV - dbore.BH_BOTTOM_DEPTH) as BH_BOT_ELEV,
(dbore.BH_GND_ELEV - vbed.BH_BEDROCK_ELEV) as BED_DEPTH_M,
dint.INT_ID,
dint.INT_NAME
FROM
D_BOREHOLE AS dbore
INNER JOIN
D_LOCATION AS dloc ON dbore.LOC_ID = dloc.LOC_ID
INNER JOIN
D_INTERVAL AS dint ON dloc.LOC_ID = dint.LOC_ID
LEFT OUTER JOIN
V_GEN_BOREHOLES_BEDROCK as vbed ON vbed.LOC_ID = dbore.LOC_ID

```

Notice the use of the AS keyword here to assign the name BED_DEPTH_M and BH_BOT_ELEV to the result of the calculations; if not specified, an automatically generated column name would be assigned instead.

Standard arithmetic operations include

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Modulo (%)

The results of 'Modulo' (for those unfamiliar with the operation) would be the (integer) remainder after a division operation. (For example: '3 % 2' would return a value of 1; '2 % 2' would return a value of 0.)

In addition there are a number of non-standard (i.e. not necessarily available in all relational database software) mathematical functions that can be useful.

- ABS(x)
 - Returns the absolute value of (x)
- SIGN(x)
 - Returns the sign of (x) (i.e. one of -1, 0 or 1)
- MOD(x,y)
 - Modulo; returns the (integer) remainder of x divided by y (same as 'x % y')
- FLOOR(x)
 - Returns the largest value less than or equal to (x)
- CEILING(x)
 - Returns the smallest integer greater than or equal to (x); also CEIL(x)
- POWER(x,y)
 - Returns the value of x raised to the power of y

- ROUND(x)
 - Returns the value of x rounded to the nearest integer; users must be careful when using this function as various ROUND(x) implementations (across software platforms) may calculate this value differently
- SQRT(x)
 - Returns the square-root value of (x)

Other functions, including trigonometric, are also available in the various relational database software (but are not described here).

Select Statement – CASE condition

The CASE conditional statement can be used when multiple values are present and the user wants to modify the result based upon the particular value. For example, if we were performing a check on the calculation of depth (in metres) for boreholes based upon their OUOM values there is an issue – the units of the OUOM values can vary; we’ll need to account for this, as follows

```
SELECT
dbore.LOC_ID
,dbore.BH_BOTTOM_DEPTH
,CASE
WHEN dbore.BH_BOTTOM_UNIT_OUOM LIKE 'mbgs' THEN dbore.BH_BOTTOM_OUOM
WHEN dbore.BH_BOTTOM_UNIT_OUOM LIKE 'masl' THEN
(dbore.BH_GND_ELEV – dbore.BH_BOTTOM_OUOM)
WHEN dbore.BH_BOTTOM_UNIT_OUOM LIKE 'fbgs' THEN
(dbore.BH_BOTTOM_OUOM * 0.3048)
ELSE -9999
END AS CHK_BH_BOTTOM_DEPTH
FROM
D_BOREHOLE as dbore
```

This returns a single value, calculated in certain instances, in ‘mbgs’ (i.e. metres below ground surface) to compare against BH_BOTTOM_DEPTH; the returned value is assigned the column name CHK_BOTTOM_DEPTH. Note that the default, if none of the conditions are met, is a particular value (i.e. ‘-9999’) that can be used to check for conditions the user has not specified.

Select Statement - ORDER BY

Information returned from a query can, at times, be more useful when it is ordered in some way. This can be the case in particular when working with time-stamped data (i.e. data with an associated date-time). If we, for example, wanted to examine all manual water levels from the borehole ‘Port Perry OW 05/3’ (LOC_ID -828830263) we would first need to determine the interval associated with the location (remember, water level - temporal - information is linked against a particular screen/interval)

```
SELECT INT_ID,INT_NAME,INT_NAME_ALT1
FROM D_INTERVAL WHERE D_LOCATION=-828830263
```

This would return the interval identifier (a numeric) and the names associated with the particular location we're examining. However, we can skip this step by including the determination of the INT_ID as part of the water level query

```
SELECT
*
FROM
D_INTERVAL_TEMPORAL_2 AS dit2
INNER JOIN
D_INTERVAL AS dint
ON
dit2.INT_ID=dint.INT_ID
WHERE
dint.LOC_ID=-828830263
```

where we've only specified the location identifier. Alternately, the name of the location could be used to extract the particular borehole (and compared against, for example, LOC_NAME). This query, though, extracts all information linked against this location (and interval). We need to limit what is returned to only that in which we are interested

```
SELECT
dit2.INT_ID
,dit2.RD_DATE
,dit2.RD_VALUE
,ruc.UNIT_DESCRIPTION
FROM
D_INTERVAL_TEMPORAL_2 AS dit2
INNER JOIN
D_INTERVAL AS dint
ON
dit2.INT_ID=dint.INT_ID
INNER JOIN
R_UNIT_CODE as ruc
on
dit2.UNIT_CODE=ruc.UNIT_CODE
WHERE
dint.LOC_ID=-828830263
and dit2.RD_NAME_CODE=628
```

We have not specified that we're only interested in water level manual readings (RD_NAME_CODE 628; refer to R_RD_NAME_CODE). Note also that the UNIT_DESCRIPTION from R_UNIT_CODE has been included as a check that all values returned (RD_VALUE) are in consistent units (in this case, all values should be in 'masl'). The information returned would be in some random order making it difficult to look for trends - we'll now (finally) order the information by date (RD_DATE)

```
SELECT
dit2.INT_ID
,dit2.RD_DATE
,dit2.RD_VALUE
,ruc.UNIT_DESCRIPTION
```

```

FROM
D_INTERVAL_TEMPORAL_2 AS dit2
INNER JOIN
D_INTERVAL AS dint
ON
dit2.INT_ID=dint.INT_ID
INNER JOIN
R_UNIT_CODE as ruc
on
dit2.UNIT_CODE=ruc.UNIT_CODE
WHERE
dint.LOC_ID=-828830263
and dit2.RD_NAME_CODE=628
ORDER BY
dit2.RD_DATE

```

By default, the returned information would be in ascending order (i.e. from the earliest date to the latest). If we wished to reverse the order, we would append DESC after the final 'dit2.RD_DATE' reference (i.e. at the end of the query).

Select Statement - Aggregate Functions and GROUP BY

If we needed to examine the average, the minimum/maximum or the total number of recorded values of the previous water level manuals we would make use of an aggregate function. This requires, in addition, a GROUP BY statement as part of the query. The GROUP BY allows a list of values to be created based upon a particular field. These functions include

AVG (the average of the specified field)
COUNT (the number of rows satisfying the GROUP BY condition)
MAX (the largest value found in the specified field)
MIN (the smallest value found in the specified field)
SUM (the total of all values found in the specified field)

These are used as following

```

SELECT
dit2.INT_ID
,AVG(dit2.RD_VALUE) as AVG_WL
FROM
D_INTERVAL_TEMPORAL_2 AS dit2
INNER JOIN
D_INTERVAL AS dint
ON
dit2.INT_ID=dint.INT_ID
INNER JOIN
R_UNIT_CODE as ruc
on
dit2.UNIT_CODE=ruc.UNIT_CODE
WHERE
dint.LOC_ID=-828830263
and dit2.RD_NAME_CODE=628

```

```
GROUP BY
dit2.INT_ID
```

Here we're grouping the returned information by the INT_ID - this means that any values related to a particular INT_ID will become part of a (virtual) list. From this list, we then calculate the average value using the AVG keyword. The result would be one value/row returned for each INT_ID. We can perform multiple aggregate functions at once

```
SELECT
dit2.INT_ID
,AVG(dit2.RD_VALUE) AS AVG_WL
,MIN(dit2.RD_VALUE) AS MIN_WL
,MAX(dit2.RD_VALUE) AS MAX_WL
,COUNT(dit2.RD_VALUE) AS NUM_WL
FROM
D_INTERVAL_TEMPORAL_2 AS dit2
INNER JOIN
D_INTERVAL AS dint
ON
dit2.INT_ID=dint.INT_ID
INNER JOIN
R_UNIT_CODE AS ruc
ON
dit2.UNIT_CODE=ruc.UNIT_CODE
WHERE
dint.LOC_ID=-828830263
AND dit2.RD_NAME_CODE=628
GROUP BY
dit2.INT_ID
```

This query returns the average, minimum, maximum and number of manual water levels associated with this particular location (and interval).

A.2 UNION and UNION ALL

Allows combinations of SELECT statements to be 'appended' together resulting in a single output set. The number of fields, the order and the field types must match between the statements. Generally, the names of the fields should match as well. Note that UNION will remove duplicate records between (and possibly within) the statements while UNION ALL will retain all records. Thus, a statement like the following

```
SELECT
dit1a.SAM_ID
,dit1a.SAM_SAMPLE_NAME
,dit1a.SAM_SAMPLE_DATE
,dit2.RD_NAME_CODE
,dit2.RD_VALUE
,dit2.UNIT_CODE
,NULL AS RD_MDL
,NULL AS RD_VALUE_QUALIFIER
,NULL AS RD_UNCERTAINTY
FROM
```

```

D_INTERVAL_TEMPORAL_2 AS dit2
INNER JOIN
D_INTERVAL_TEMPORAL_1A AS dit1a
ON
dit2.INT_ID=dit1a.INT_ID AND dit2.RD_DATE=dit1a.SAM_SAMPLE_DATE
WHERE
dit2.RD_TYPE_CODE=77

UNION ALL

SELECT
dit1a.SAM_ID
,dit1a.SAM_SAMPLE_NAME
,dit1a.SAM_SAMPLE_DATE
,dit1b.RD_NAME_CODE
,dit1b.RD_VALUE
,dit1b.UNIT_CODE
,dit1b.RD_MDL
,dit1b.RD_VALUE_QUALIFIER
,dit1b.RD_UNCERTAINTY
FROM
D_INTERVAL_TEMPORAL_1B AS dit1b
INNER JOIN
D_INTERVAL_TEMPORAL_1A AS dit1a
ON
dit1b.SAM_ID=dit1a.SAM_ID

```

would combine the parameter results from both D_INTERVAL_TEMPORAL_1A/1B and D_INTERVAL_TEMPORAL_2 that were taken at the same date-time for the same interval. Note that, as D_INTERVAL_TEMPORAL_2 does not have uncertainty and mean-detection limit fields, NULL values are used as placeholders. Also, the RD_TYPE_CODE of '77' indicates that the parameter was taken in the field even though it is otherwise generally assumed to be a 'laboratory' parameter.

Appendix B - Soil Classification Systems and Translation of Geologic Layers

B.1 Unified Soil Classification System

The most common soil classification system used in geotechnical engineering studies is the Unified Soil Classification System (USCS) (ASTM D2488-93). Details of the USCS system are provided in the attached table (Table XXX). The first step in the USCS system is to determine, on the basis of grain size, whether the soil is fine-grained (silts and clays) or coarse-grained (sands and gravels). If 50% of the soil were to pass through a 200 mesh sieve, the soil would be described as fine-grained. Similarly, if more than 50% of the soil were to be retained on a 200 mesh sieve, the soil would be described as coarse-grained.

The difficulty with using the USCS is that soil classified as coarse-grained (i.e. more than 50% retained on a 200 mesh sieve) may still contain considerable clay and may both look like a clay-rich soil and also behave as a plastic or cohesive soil. Plasticity refers to the ability of a material to be deformed rapidly without cracking or crumbling and then to maintain that deformed state after the deforming force is released (e.g. molding clay into a shape). For this reason, describing the soil according to the USCS is quite difficult to do accurately based on field observations alone. It is more appropriate where laboratory testing of grain size (sieves and hydrometers) and physical properties (plastic and liquid limits) is completed. This is not the case for most of the borehole logs that are likely to be encountered in the ORMGP.

As outlined in Table XXX, the USCS has a two-letter capitalized group of symbol designations (GW, GP, etc.) that is not used in other systems. Recognizing these symbols is important in identifying that the USCS system is being used and therefore translating the descriptions appropriately. Be aware that there is a sedimentological lithofacies classification system, which is described below, that uses two letters symbols as well; that system, however, only has the first letter capitalized.

COARSE-GRAINED SOILS More than half the material (by weight) is individual grains visible to the naked eye	GRAVELLY SOILS More than half of coarse fraction is larger than 4.75 mm		CLEAN GRAVELS Will not leave a stain on a wet palm		Substantial amounts of all grain particle sizes		GW
			DIRTY GRAVELS Will leave a stain on a wet palm		Predominantly one size or range of sizes with some intermediate sizes missing		GP
					Non-plastic fines (to identify, see ML below)		GM
			SANDY SOILS More than half of coarse fraction is smaller than 4.75 mm		CLEAN SANDS Will not leave a stain on a wet palm		Wide range in grain size and substantial amounts of all grain particle sizes.
	Predominantly one size or a range of sizes with some intermediate sizes missing						SP
	DIRTY SANDS Will leave a stain on a wet palm		Non-plastic fines (to identify, see ML below)		SM		
Plastic fines (to identify, see CL below)			SC				
FINE-GRAINED SOILS More than half the material (by weight) is individual grains not visible to the naked eye (<0.074 mm)	Ribbon	Liquid Limit	Dry Crushing Strength	Dilatancy Reaction	Toughness	Stickiness	
	None	<50	None to Slight	Rapid	Low	None	ML
	Weak	<50	Medium to High	None to Very Slow	Medium to High	Medium	CL
	Strong	>50	Slight to Medium	Slow to None	Medium	Low	MH
	Very Strong	>50	High to Very High	None	High	Very High	CH
HIGHLY ORGANIC SOILS	Readily identified by colour, odour, spongy feel and frequently by fibrous texture						OL OH Pt

Translating the USCS into Material Codes

If the USCS system is used in a borehole log to be translated into our database, the following key points should be considered:

- Clean gravels (GW and GP) have little if any fine-grained material (silt and clay). Therefore, silt and clay would not be placed in any of the Material 1, 2 or 3 columns.
- Dirty gravels (GM or GC) by contrast contain fine-grained materials. GM essentially refers to gravel with silt and GC refers to gravel with clay. Therefore, gravel should be placed in the Material 1 column in both cases and silt should be placed in Material 2 for GM, whereas clay should be placed in Material 2 for GC.
- The same rationale as described above for clean and dirty gravels applies to clean sand (SW, SP) and dirty sand (SM, SC).
- Often times, more than one symbol is used to describe a soil that contains many constituents. For example SW/GW would describe a soil containing a wide range of sand and gravel grain sizes. In this case, sand should be placed in Material 1 and gravel should be placed in the Material 2 column.
- Fine-grained soils are described as either M – silt or C – clay. The second letter refers to the results of the liquid limit test; those with a $LL > 50\%$ are H, those below are L. For clays, this roughly translates in practical sense to ‘highly plastic’ (H) or ‘low plasticity’ (L), although the same logic does not hold true for silts, since they are never highly plastic by nature, despite sometimes being classified as MH. Regardless, it is probably easiest for our purposes to just use the first letter to determine whether it is a clay (C) or silt (H) for the material coding.

B.2 Golder Associates Classification System (GACS)

Hydrogeologists in southern Ontario often prepare borehole logs based on field observations alone or with a limited number of grain size analysis. As a result, they often use a practical soil classification system that has been previously described by Golder Associates (Golder) as the Golder Associates Classification System (GACS) (GAL, 1992), although other consultants undoubtedly use their own variation.

The first step in the GACS is to determine, on the basis of performance, whether the soil behaves as a plastic (cohesive) or a non-plastic (cohesionless) material. This can be accomplished in the field by varying tests, such as determining if a thread of the soil can be rolled with varying moisture content. If the soil behaves plastically, then it is described as a clay (either a clay, silty clay or clayey silt). If the soil behaves as a non-plastic material, it is then described according to the USCS for coarse-grained materials. This distinction is made from the USCS because a soil can start to behave as a plastic soil at less than 10% volume clay content but it would be difficult, based on field observations alone to determine whether the soil had 10% clay content or 60% clay content. Furthermore, plastic soils, regardless of clay content, are more likely to comprise aquitards than aquifers. Cohesionless soils do not exhibit plastic behavior at any water content. By contrast, cohesive soils exhibit plastic behavior as defined by the Atterberg limit test over a reasonably wide range of molding water contents. Plasticity is controlled by the presence and type of clay mineral within the soil.

In the case of plastic soils, the predominant soil type is determined based on the degree of plasticity, which can be estimated in the field by a number of field identification tests (e.g. rolling a thread). The predominant soil type are often capitalized (SILTY CLAY) and these would be put in the Material 1 and 2 columns. Note that in the case of this example (SILTY CLAY), silt is the second major constituent after clay, and would be placed in the Material 2 column. Where there is greater than 30% by weight of a coarse-grained constituent, the modifier with is used – for example, SILTY CLAY with sand. In this case clay would be placed in the Material 1 column, silt in the Material 2 column and sand in the Material 3 column.

For non-plastic soils, major soil constituents are those that constitute 30% by weight or more of the soil. Often times, consultant’s logs will have the major constituents capitalized in a soil description. When a single constituent dominates (typically >50%) by weight, that constituent is named (e.g. SAND). This constituent would therefore be described in the Material 1 field. When two constituents dominate (say one 50% and one 30%), both constituents are commonly capitalized in the consultant’s log (e.g. SAND and GRAVEL). In this case, the most abundant (if identified) would be placed in the Material 1 column and the other would be placed in the Material 2 column.

The GACS also uses modifiers ‘and’ and ‘with’ to describe soils where two constituents dominate. The modifier ‘and’ is used to connect cohesionless constituents (e.g. SAND and GRAVEL). The modifier ‘with’ is used to connect coarse-grained constituents to cohesive soils (e.g. CLAYEY SILT with SAND). In the last example, silt would be noted as Material 1, clay as Material 2 and sand as Material 3.

<i>Percent by Weight</i>	<i>Modifier</i>	<i>Example</i>
0-5	Trace	Trace Sand
5-12	Little	Little Sand
12-20	Some	Some Sand
20-30	(ey) or (y)	Sandy

Minor soil constituents should be placed into the Material 3 column unless there is only one predominant constituent, in which case the minor constituent should be placed as Material 2.

Note that the modifiers are not directly applicable to clay mineral content because as little as 8 – 12% clay mineral by weight can make a soil behave as a plastic material. Minor amounts of clay in an essentially cohesionless soil should be indicated as ‘trace clay’

B.3 Comment on Tills and Diamicts

It is important to realize that the USCS does not enable the identification of tills or diamicts. In the GACS or similar system, sometime tills are identified in brackets after the main constituents (e.g. CLAYEY SILT (till), trace gravel). By using the word till or diamict, the person who logged the soil implies that the soil is massive with a poorly

sorted mixture of fine and coarse-grained constituents. Use of the word till implies an interpretation of depositional setting (usually subglacial), whereas the use of the word diamict implies no associated interpreted depositional setting. It is also important to realize that, in Southern Ontario, many tills can have the same constituents as glaciolacustrine deposits so that if the soil has been interpreted as a till, it is imperative that this interpretation be translated into our database.

Identification of tills in one of the key challenges of our work as recognizing tills greatly aids establishing the stratigraphy. Therefore, borehole logs in which the tills have been correctly identified are invaluable. Because of this, if a log identifies a soil as a till, till should always be placed in the Material 3 column. This may result in having to leave out one of the minor soil constituents but it is more important to have till in the database rather than a minor soil constituent. For example, clay silt (TILL), trace gravel, might be either translated to CLAY (material 1), SILT (material 2), TILL (material 3). Alternatively, gravel could be placed in the Material 2 column and silt could be left out.

Tills / diamicts are considered in sedimentological classifications systems, described below.

B.4 Lithofacies Classification Systems

A lithofacies classification system was developed by fluvial sedimentologists. We may occasionally see a borehole log using this system – typically from someone involved in academic research / GSC etc.

The original lithofacies system (Miall, 1978) employed 20 standard lithofacies types, each had assigned code letters. The codes are in two parts: the first is a capital letter G, S or F, representing gravel, sand or fines, respectively. The second part of the code consists of one or two letters to describe the most characteristic internal feature (structure) (not necessarily constituent). A brief overview of these lithofacies codes is provided in Table XXX.

Facies Code	Lithofacies	Sedimentary Structure
Gms	Massive, matrix-supported (ms) gravel (G)	None
Gm	Massive (m) or crudely bedded gravel (G)	Horizontal bedding or imbrication
Gt	Gravel (G), stratified	Trough (t) cross-beds
Gp	Gravel (G), stratified	Planar (p) cross-beds
St	Sand (S), medium to very coarse, maybe pebbly, stratified	Trough (t) cross-beds
Sp	Sand (S), medium to very coarse, maybe pebbly, stratified	Planar (p) cross-beds
Sr	Sand (S), very fine to coarse	Ripples (r)
Sh	Sand (S), very fine to coarse	Horizontal (h) lamination
Sl	Sand (S), fine	Low-angle (l) cross-beds
Se	Scours (S), erosional (e)	
Sc	Sand (S), fine to coarse	Shallow scour (s) fills
Sse, She, Spe	Analogous to Ss, Sh and Sp	Equivalent eolian (e) deposits
Fl	Fines (F) – fine sand, silt, mud (clay)	Fine lamination (l)
Fsc	Fines (F) –silt, mud (clay)	Laminated to massive (silt, clay)
Fcf	Fines (F) – mud	Massive with freshwater mollusks (clay, freshwater)
Fm	Fines (F) – mud, silt	Massive (m), desiccation cracks
Fr	Fines (F) – silt, mud	Rootlet (r) traces
C	Coal (C), carbonaceous mud	

If this coding system is encountered

- Check to see if there is any accompanying description that may provide details regarding the constituent grain sizes. For example, the fines (F) may be further described as either silt or clay.
- Use Material 1 2 and possibly 3 to put in translated interpreted grain sizes, with Material 1 being used for the most predominant grain size.
- Any textural symbols that are of relevance should be put in the Material 3 column. For example, in our project, we are frequently trying to determine whether fine-grained silts and clays are glaciolacustrine / channel fill in origin (possibly Oak Ridges Moraine equivalent) or a till (e.g. Newmarket Till). If the fine-grained material is laminated (Fl), it is more likely to be of lacustrine or channel fill origin as opposed to a true lodgement till in origin, so this information would go in the Material 3 column. Our lithology symbol library in the database only has an option of ‘layered’ that would be appropriate in this case.

Quaternary Geology Lithofacies Classification System

The lithofacies classification system discussed above was adapted and modified by Quaternary geologists to enable description and interpretation of Quaternary (i.e. glacial) facies and depositional environments. The classification system was modified into a four part lithofacies code. The code designator D was introduced for diamict. In Southern Ontario the overburden deposits are Quaternary deposits, therefore, this is the system we

are more likely to encounter in a local academic / scientific borehole log. This system is summarized in Table XXX.

<i>Facies Code</i>	<i>Description</i>
Diamict D	
Dm	Matrix supported diamict
Dc	Clast supported diamict
D-m	Massive diamict
D-s	Stratified diamict
D-g	Graded diamict
Genetic Interpretation ()	
D--(r)	Diamict - residimented
D--(c)	Diamict - current reworked
D--(s)	Diamict - sheared
Sand S	
Sr	Rippled sand
St	Trough cross-bedded sand
Sh	Horizontal laminated sand
Sm	Massive sand
Fine-Grained (Mud) F	
Fl	Fines, laminated
Fm	Fines, massive
F-d	Fines with dropstones (pebbles, gravels, etc...)

It should be noted that more than one secondary code can be used. A few examples of this are: Dmm – matrix-supported, massive, diamict; Dmg, matrix supported, graded etc.

If this coding system is encountered, the approach is similar to that used for the previous lithofacies classification system

- Check to see if there is any accompanying description that may provide details regarding the constituent grain sizes. For example, the fines (F) may be further described as either silt or clay.
- Use Material 1 2 and possibly 3 to put in translated interpreted grain sizes, with Material 1 being used for the most predominant grain size.
- Use of the diamict designation (D) should be considered carefully because it be a till, possibly one of the regionally extensive till sheets that we are attempting to map in our project (e.g. Newmarket Till) or it could be some other type of diamict (debris flow etc.) that would not be as regionally significant and which should not be confused with a till. If, based on its stratigraphic position or other evidence,

you think it is a till then put TILL in the Material 3 column, otherwise, leave it as a diamicton. The available list of lithologies to choose from in Sitefx includes both till and diamicton.

- If fine-grained materials are identified as layered (F1), the dropdown choice of 'layered' should be put in the Material 3 code in SiteFX. This will help identify this unit as unlikely to be a till (although not conclusively).

Appendix C - Baseflow Estimation

The ORMGP database no longer contains estimates of groundwater discharge from total streamflow at gauging stations. Instead this information is generated on-the-fly, using a variety of techniques, online through the ORMGP website.

Appendix D - External Database Sources

The following sources have been used for adding of information into the database. This is not to be taken as an exhaustive list. Approximate dates are indicated (multiple dates are possible).

Environment Canada

Date February 2004
Name Climate Data
Source Sandy Radecki - Burlington

Description Climate data up to the end of 2003 has been added to the database. Included is the following data, provided on a daily average basis.

1. Precipitation
2. Rain
3. Snow on ground
4. Maximum, minimum and average temperature
5. Solar radiation where available

Details Approximately 560 climate stations are listed in the database, 517 operated by Environment Canada, and the remaining 43 operated by CA and other agencies. Locations are shown in the following figure. Climate stations are identified in the database using a LOC_TYPE_CODE of 9 in the D_LOCATION table. A standard ORMGP query is provided in the database to display climate stations– this query is called ORMGP – Climate Stations ALL

Future Data Stream flow data from Environment Canada extends to December 2000, and requires updating. Past requests have been referred to their web site, which currently does not accommodate the bulk downloads required by the database. Improvements to the web site are expected, but unfortunately it has been difficult to access the required data.

Geological Survey of Canada Data

Date August 2004
Name Outcrops
Source Dave Sharpe's ORM database, August 2004 version from Charles Logan

Description Outcrops mapped by the GSC and the OGS were compiled into a database by the Terrain Science Group to complement their well log data for the ORM. Outcrop data basically looks like well log data, with an X, Y and Z positions and a geological sequence. The geological sequence can range from several centimeters to many meters in the case of large exposed faces. Geological descriptions have, for the most part, been mapped into the MAT1,2,3 and 4 descriptions used in the database.

Details Approximately 30,000 outcrops have been added to the database, covering an area from xx to yy. Locations are shown in the following figure. Outcrops are identified in the database using a LOC_TYPE_CODE of 11 in the D_LOCATION table. A standard ORMGP query is provided in the database to display outcrops – this query is called ORMGP – Outcrops ALL

Future Data The GSC located a number of non MOE wells for the database. Some will already exist in the ORMGP database (MTO geotechnical wells for bridge abutments); however, the remaining wells must be identified and transferred into ORMGP.

MNR Surface Water Station Coordinates

Date June 2004

Name Updated UTM coordinates for MNR and EC stations

Source MNR

Description The original Environment Canada coordinates were initially provided in digital lat long format with limited accuracy, resulting in stations not plotting on their assigned water course. This has been updated using MNR GPS data.

????? SHOULD THIS BE REMOVED ?????

MOE PTTW Database

Publically available format. As of 2014-08-01 (approximately).

MOE South Simcoe Groundwater Study data files

Date June 2004

Name Dixon Hydrogeology database for MOE GW study

Source Dixon Hydrogeology

Description DHL released their updated project database for use in the ORMGP database. Of value were several hundred new wells installed as monitoring wells by DHL during previous projects, and updated UTM coordinates for about 6000 MOE wells across Simcoe County

MOE WWDB (Water Well Database)

Multiple import periods, including (approximately):

- 2002
- 2006
- 2010
- 2013-08-01

Scarborough Report

120 Wells in digital form from a competing/preceding project with the city of Scarborough (see Eyles and Doughty, 1996).

UGAIS Wells

Date June 2004

Name Geotechnical borehole data for urban areas

Source GSC - MNR

Description UGAIS (Urban Geology Automated Information System) is a compilation of largely urban boreholes for major municipalities across Canada prepared by the GSC in the mid seventies. Wells from Hamilton to Oshawa were added to the ORMGP database. These wells are largely geotechnical investigation test holes, offering descriptive and accurate soil descriptions and blow counts. Soil descriptions have been mapped into the standard look up fields of the D_GEOLOGY_LAYER table in the database. The geological descriptions provided are based on the soils recovered in split spoon samples, and the depths provided were the depths of the split spoon sample.

Details Approximately 33,200 geotechnical boreholes, with depths ranging from 0.1 m (soil sample) to in excess of 100 m were converted from ASCII format and appended to the database. These boreholes are identified in the database using a LOC_TYPE_CODE of 1 and DATA_SOURCE = 'ugais' in the D_LOCATION table. A standard ORMGP query is provided in the database called ORMGP – Wells UGAIS. The wells have been assigned as position certainty code of 3 (10 to 30 m), although as geotechnical borings, it is expected the accuracy may be greater, but this cannot be confirmed. Only about 7,040 of the 33,200 include water level information.

Future Data

In several cases, the geological description for the wells included water level information. Many of these water levels remain in the geological description field, although often hidden by carriage return characters. Further work is necessary to parse the water level data into the water level table. It is noted that the original ASCII files included Liquid Limit and Plastic Limit values; this data could not be confidently converted owing to missing depth readings.

Appendix E – YPDT-CAMC (ORMGP) Database Timeline

Database Release 1 (First Released 2003)

20040629 ('YPDT Database Meeting - June 29, 2004' Memo)

Comments

- Report library to be kept as separate database; reports should have LOC_ID's assigned
- ID's will be random
- Report database should have columns for tracking info (e.g. does it have boreholes, geophysics, etc ...)
- SiteFX should be developed to access reports in C:\OAKRIDGES directory through selection

July Database Release

- Some P1 wells have changed location (up to 4km)
- Errors in locations (coordinates): Nottawasaga, Scugog, York
- Elevation code of '1' should never be updated (these wells do not match the DEM)
- Implement QA code for elevation; keep the difference between the DEM and original elevation
- Outline database policy in manual/user guide
- UGAIS geology - split spoon samples only; first three columns would be Materials 1-3 and column 4 will be the sample interval; blow counts, etc ... will also be incorporated; water levels in feet - to be fixed
- Screens are to be assigned to hydrostratigraphy

Water Level Discussion

- Create a temporary water level inventory table for each well - provides a synthesis of water level information (e.g. maximum, minimum, average, last)
- Table will have all locations (possible more than once; e.g. water level and water level for pumping rates)
- Skip chemistry (for now)

Water Level Structure

- Existing structure is York region (logger's produce 100000 records per year)
- Access time rather than amount of data introduces difficulties
- Archive data that isn't being used?
- Use multiple linked tables segregated on a time interval (say five years)
- Using BLOB's for water level data - is it too much of a black box? (i.e. will it be accessible?)
- All temporal data tables to be treated as a single table within SiteFX
- Policy needed on how to switch to archive

- Validating of data
- Link GSC database for internal QA
- Update County and Township codes for all wells
- York wants to (continue to) use Microsoft SQL Server
 - Keep all water levels in one SQL Server table; install the Microsoft Database Engine so that Microsoft Access can read/process the SQL Server table
 - Use BLOB's to reduce the size of the tables

***Database Release 2
(First Released 2004)***

20050117 ('YPDT Database Meeting' Memo)

Comments

- Additional GSC database components to be added to the database
- Modification of records should only be accomplished by a single person
- A running record of changes should be kept
- QA/QC procedures need to be in place (for acceptance of data into the database)
 - Track changes form to be returned with database submissions
 - No trading of databases (between partners)
 - All changes to be approved by overseer (one for each of: West; Central; and East)
 - Each synchronization requires SiteFX document outlining changes and including explanations
- Reconciliation is currently too long and painful (import/export through SiteFX?)
- Agency flag table should be in the database and populated - include a buffer around the agency
- Location names must be standardized
- When adding information to the database:
 - The data source must be populated (master source and then details concerning the source)
 - The location table must be tidied up to reflect at least one and maybe two levels of water (the primary watershed flowing into Lake Ontario or Lake Simcoe and the secondary watershed that may or may not be populated)
 - Location QA code must be populated; standard methodology must be documented
 - All locations must have a DEM elevation and wells have a bottom depth and elevation
 - A linkage from a location to a report must be established; links between log information and scanned original
- Next database reconciliation August 1, 2005
- Proposed Import/Export tool (in SiteFX); incorporate into February training course

20050322 ('YPDT Database Meeting' memo)

Comments

- INT_GEOLOGY in D_INTERVAL table will be used to manually assign screen to a geologic unit (anyone can fill this column); this can override the automatic updated table if a head map is produced
- Addition of D_LOCATION_HYDROSTRAT; automatically populated; includes LOC_ID, HYDROSTRAT_UNIT, TOP_ELEV, BOT_ELEV
- DEM will always be the top of RAC; each unit will be represented in each BH (RAC, HKT Aquitard, ORAC, NT Aquitard, TAC, SD Aquitard, SAC, WBAC, Bedrock Aquitard, Channel silts, Channel sands)
- Naming conventions - four aquifer complexes: SAC; TAC; ORAC; RAC (Recent surficial stuff); longer term - each of these can have a modifier that describes a more local aquifer setting (e.g. we can have ORAC-Caledon or ORAC-Brougham, etc ...)
- An IBOUNDS file should be set up to reflect existence of each unit (once geology finalized); used to clip head maps in each aquifer; should be created automatically
- Get new wells from Azimuth, Jagger-Hims, Golder
- Review bedrock surface (process and/or methodology; include outcrop)
- Produce series of equations to calculated vertical gradients between two aquifers (only where both aquifers present)

20060303 through 20060426 ('New Database Notes' Memo)

Comments

- Import db table D_LIBRARY; reconcile lookup tables with existing; remove/archive D_DOC_BH, D_DOC_DOCUMENT and D_DOC_ASSOC
- D_INTERVAL_EQUIPMENT; check and dump
- Should specific capacity be assigned an interval property - is it more a function of the well (especially bedrock wells); leave for now (change in next database?)
- D_INTERVAL_PUMP; remove
- D_INTERVAL_SOIL; replace with D_INTERVAL_HYDROMETER and D_INTERVAL_SPLITSPOON?; review, take soils data out of database and redefine tables
- D_HYDROMETER and D_HYDROMETER_READINGS; remove
- D_INTERVAL_TEMPORAL_BASEFLOW_EC_ZIP; contains blob's; keep ADV and L tables as summary of contents; include sample queries for de-blobbing/access; all EC_ZIP tables combined into single table
D_INTERVAL_TEMPORAL_TYPE_ENVCAN
- D_INTERVAL_TEMPORAL_BULK; contains flow rates from spot flow locations; information should be moved into spotflow tables - SWFLOW_EC and SWFLOW_OTHER coded with station type; restructure all temporal data table into 4 tables, only
- D_DATA_SOURCE is used to describe datasets; linked to DATA_ID fields
- D_INTERVAL_TEMPORAL_CLIMATE; empty, remove

- D_INTERVAL_MONITOR; many intervals with ‘good’ top- and bottom-depths, calculated elevations -10000; these are from ‘nodata’ markers of -99999; SiteFX needs to recalculate elevations using DEM as base; many intervals with screens greater than 10m (some greater than 50m); no action at present; review interval types
- D_INTERVAL_PROPERTY; leave as is - use RD_NAME_CODE to map _OUOM to reading code; handled through SiteFX
- D_INTERVAL_TEMPORAL_DATA; move all water levels to D_INTERVAL_TEMPORAL_WL; include soil and water quality data?
- D_INTERVAL_TEMPORAL_DATA_DETAILS should be linked to D_INTERVAL_TEMPORAL_DATA; should these be relabelled to chemistry data?
- D_INTERVAL_TEMPORAL_LOGGER_ZIP; empty, remove and add logger files to _WL_ZIP
- D_INTERVAL_TEMPORAL_PROD; no field descriptions; York tracking?; remove and extract WL and pumping data;
- D_INTERVAL_TEMPORAL_PUMP; remove, data extracted
- D_INTERVAL_TEMPORAL_PUMP_ADV; remove, data extracted
- D_INTERVAL_TEMPORAL_SWFLOW; empty, remove
- D_INTERVAL_TEMPORAL_SWFLOW_ADV; empty, remove
- D_INTERVAL_TEMPORAL_SWFLOW_BASEFLOW; where did this information go; review
- D_INTERVAL_TEMPORAL_WL; reconcile consistent name of tables (table classes: WL, WL_ADV, WL_Summary, WL_ZIP, WL_EC_ZIP); repeat for _SWFLOW; latter must contain all partner spotflows; tables keeping should have consistent fields in consistent order (some of these temporal tables have extra fields)
- Repeat previous structure with: CLIMATE; WQ; PUMP; remove BULK
- Incorporate all PGMN data, municipal data, MOE data into 4 (previously mentioned) temporal tables; do not have a temporal table per partner agency per data type; queries should separate the data, not the tables
- Correct Bathurst water levels (check codes)
- Merge York water level data; some records in fasl; information out of date?
- INTERVAL_TEMPORAL_WL_HISTORICAL; mix of water levels from various well sources here; clean up and move to 4 temporal tables
- _OUOM has special meaning throughout database; cannot be changed (used by SiteFX?)
- D_LOCATION
 - added a CA code, keep
 - LOC_USER_ID, remove this field
 - UTM_ZONE_OUOM, left as is, not changed to UTM_ZONE as suggested
 - LOC_ACTIVE vs LOC_STATUS; move the active code field (as they both pertain to the same information)

- correct LOC_COUNTY_CODE for locations where it is missing; same for LOC_TOWNSHIP_CODE
- LOC_BASIN1 to 11; remove; populate Watershed1 field using Viewlog
- remove WATERSHED_DEV field
- no sources identified for a number of locations; add D_DATA_SOURCE table and reference
- review climate station locations (counts are off)
- add location (and interval) confidentiality codes and look-up table
- D_LOCATION_PURPOSE separate from LOC_USE_PRIMARY_CODE and LOC_USE_SECONDARY_CODE, the latter being populated by MOE codes
- LOC_CORR; unknown MOE flag
- change name in LOC_NAME_ALT1 to 'Owner name withheld by MOE' for new wells added (from MOE)
- D_LOCATION_ACTIVITY; used to track changes if required; not standardized
- A number of locations do not have an AVI code in the AVI table; re-run analysis?
- LOC_PURPOSE; review to include all locations (not just boreholes?)
- D_LOCATION_QA
 - some locations do not have QA codes; this needs to be fixed (catch it in SiteFX)
 - include revised date (for elevation update) in QA_ELEV_METHOD
 - rename QA_ELEV_CONFIDENCE_CODE?
 - change coordinate confidence codes for Environment Canada stations (elevations should be blank); same for surface water stations
 - All outcrops assigned a QA_ELEV_CONFIDENCE_CODE of '2' (from DEM); keep or review
 - QA_ELEVATION_CONFIDENCE_CODE not assigned for locations from MOE WWDB 2004
 - under R_LOC_DATA_SOURCE_CODE, change 'suspected' to 'inspected'; should this table have a 'QA' in its name
 - a large number of records do not have a QA_DATA_SOURCE_CODE; update before next synchronization
 - a large number of records do not have a QA_DRILLER_ACCURACY_CODE; update or drop?
 - same issue with QA_PUMPING_CODE and QA_WL_STATIC fields
 - shouldn't the relationship between this table and D_LOCATION be 1:1
- D_LOG; should be hidden, not necessary to distribute; remove?
- D_LOGGER_CALIBRATION_READINGS, D_LOGGER_IMPORT_ERROR, D_LOGGER_IMPORT_HISTORY, D_LOGGER_NAME; all used for importing of data from York; keep
- D_OWNER; make sure wells are being linked to this table
- D_PUMPTEST and D_PUMPTEST_STEP; has any new MOE data been incorporated?

- D_SURFACE_WATER; add the data from the CRA study here; why is there a special SW_ID; parallels D_BOREHOLE
- D_VERSION; references required SiteFX version (regarding structure changes)
- D_WATERUSE_*; exclude permits outside area?; change to D_PTTW_* instead (next release?)
- D_BOREHOLE; many locations do not have a BH_GND_ELEV_DEM value
- Include RD_NAME_CODES from CLOCA
- D_CLIMATE; GND_ELEV - update?; start and end dates?
- D_CRITERIA; data deleted from table
- D_DATABASE_DEVELOPMENT; remove
- D_GEOLOGY_FEATURE; add information from new MOE WWDB
- D_GEOLOGY_LAYER; remove GEOL_UNIT_CODE if table of formation tops added
- D_INTERVAL; what is INT_REGULATORY_CODE; what is DWS# - drinking water system (stems from work being carried out at York Region)
- Procedure when new coordinates replace existing coordinates?
- A location is not required to have an interval
- Should make plain relationship between R_ and D_ tables before next database release
- R_BH_CONSULTANTS; leave in database
- Change BH_GND_ELEV_QA_CODE to R_LOC_ELEV_QA_CODE
- R_BH_STATUS; info comes from the MOE; leave for later
- R_CONV_CLASS; for dealing with elevation, depth and temporal data conversions
- R_DOC_FORMAT_CODE, R_DOC_GROUP_CODE and R_DOC_TYPE_CODE cannot be removed at present due to SiteFX relationships
- D_GEOLOGY_FEATURE; water found descriptions cleaned up; remove geology descriptions
- R_GEOL_LAYERTYPE_CODE; legacy table; determine if we can remove it
- Material code tables should be made consistent; this is a possible future problem as users can add values to one (or more) of the tables without checks in place
- R_GEOL_SUBCLASS_CODE; originally intended to track different geology interpretations
- R_PUMP_TYPE_CODE; removed

20060627 ('Database Meeting - June 27, 2006' Memo)

Comments

- What other problems are there to be fixed
- Do we need to adopt a Policy Manual regarding changes in the future, for example
 - Changes to database - do we accept changes or only do them ourselves
 - New locations - should these only be accepted through SiteFX
 - How often do we update Environment Canada data; what is the most efficient way of getting this information

- Do the baseflow procedures/calculations need to be re-run every time (i.e. at each new database release) or should this be fixed; should this information be contained in it's own table if it's to remain static
- What other big databases are to be incorporated
- Picks table - should it be part of the database; how do we streamline the reconciliation process
- How is the master database to be managed; where is it to reside

***Database Release 3
(First Released 2006, July)***

20060830 ('Database Meeting Notes' Memo)

Comments upon '2006 Database Release'

- Synchronization
 - Extended time for database synchronization
 - Source datasets inadequate
 - Unrealistic re: assumptions about 'automation'
 - Poor QA/QC on incoming datasets (overlapping datasets, subsets of data, multiple sharing/processing)
 - Original '2005' synchronization did not go well (problems with GSC and other import problems)
 - Lookup tables should auto-increment
- New and Updated Datasets
 - Poor/inconsistent instructions on the level of detail required to capture the information
 - Many changes/evolution since the original MOE update (difficult to replicate the import/translation process)
 - Comprehensive inventory analysis on new datasets was not done
- Correction and Revision of Data
- Database Check, Normalization and Translation
 - Data inconsistencies (e.g. watershed codes, municipalities, usage, etc...); application of lookup tables
 - Incomplete/uneven classification (e.g. all wells need a purpose and QA code)
- Software and Query Compatibility
- Pitfalls or Challenge Tasks
 - Clients require full database access (i.e. not just through tools)
 - Synchronizing data between agencies
 - Adding new locations (most visible problem)
 - Database complexity (e.g. blobs, multiple related tables, overlapping/unclear storage, evolving MOE database)
- Review of User Needs
 - User groups: Conservation Authorities; Municipalities

- Mandate - what data should be captured (only groundwater data)?
- Integration with user databases
- Evolving user needs; no longer a regional tool, move to more detailed studies and analysis
- Data Directions and Data Needs
 - Future database needs: modelling, municipal monitoring, permit allocation
 - Future data directions: MOE and MNR collection plans; who is the integrator of the datasets?; what datasets will become the definitive data source; how do we handle corrections and updates
 - Golden spikes?
- Database Scope and Objectives
 - Evaluate needs, data directions and mandate to determine: what to include/exclude (clear goals/objectives); what is the CAMC mandate for database storage; what is the mandate for source water protection
 - Confidentiality
 - Internal data (e.g. York pumping data)
- Technical Issues
 - Platform (SQL Server)
 - Interface/Tools (SiteFX, web)
 - Dataset refresh (MOE WWDB, climate, HYDAT, PTTW)
 - New datasets (evaluation, translation and merging)
 - New Synchronizations
 - Option A (not an option); repeat of '2006' synchronization process; one monumental update
 - Option B (not an option); database synchronization - EarthFX runs synchronization algorithms, child database synchronized un-reviewed; data is either accepted or rejected by algorithms
 - Option C; SiteFX update tool; tool handles submittal and checking of data changes; DB synchronization through user-specified intervals; datasets held in review by CAMC (location comparison, spatial tools); lookup tables and data locked; users flag (some) data as 'do not submit'; good web based user feedback
 - Option D; Full time database administrator; strategic planning
- Staffing and Management Issues
 - Scientific/directions team (define goals, objectives, etc ...)
 - Database Administrator; coordinate uploads, updates, delivery, etc ...
 - QA/QC review team
 - Programming/Tools team

20060830 (*'Database Release Notes' Memo*)

Comments

- Late release of database: scope of database grown considerably; data errors had crept in; event based update process does not work
- Quality Control Issues

- Cannot rely on junior staff; need knowledge of database, data formats, applications
- Database structure complex (more SiteFX data entry rules?)
- Update/Addition process; numerous steps and decisions made
- Central database; central server into which all data is loaded; simple terminal services client system
- Need scheduled update process; well defined tasks and deliverables; long term planning and scheduling
- Data Streams
 - PGMN integration and check
 - York Region catch-up
 - Integration with Peel and Durham
 - Water Quality
 - Report mining
 - Internal QA coding and cleanup
 - ARCHydro integration?
 - MNR SW connection?
- Evolving Needs
 - Transient
 - Site scale analysis/refinement
 - Confidentiality
- Responsibility
 - Review and identify data gaps
 - Rules, guidelines, checks, documentation, training manual update, data entry rules, data QA queries, visual checks of the data
 - Tools; web entry tools, web based datasets, air photos and other web based data; processing
 - Political issues and commitment; firm commitment and dedication of staff for training and support

20061106 ('Database Meeting - November 6, 2006' Memo)

Comments

- Protocol required regarding making changes to database
- Proposed to treat solid samples as intervals; we may change the number and type of intervals and add appropriate tables to hold results; could this be accommodated in existing database structure
- Blow counts; do we want this information to be displayed on log guides or is it a geologic feature or an interval attribute
- SiteFX does not allow 'bedrock encountered' to be selected
- Area of gauge station to be added for surface water types; add whether a surface gauge is natural or regulated (the latter being an Environment Canada site)
- Incorporate LOC_STUDY approach of REG into the database

- Master database remains July 2006 (the issues surrounding [the use of the] DEM left hanging)
- FTP access (for changes?) setup on Halton's site; remote-access restricted to 5pm to 7am
- Change Justification - new table
 - LOCID
 - ChangeDate
 - ChangeReason
 - ChangeComment
 - ChangeAuthor
 - ChangeJustification (duplicate of reason?)
 - ChangeTable (maybe)
 - ChangeField (maybe)

20061129 ('2006 Database Synchronization - Summary of Activities' Memo)

The 2006 database synchronization involved several key additions and changes to the database. Broad outlines of the changes made to the database between the 2004 and 2006 versions include

- Expansion of the number of records in the recorded dataset
- Inclusion of Report Library data and reference tables
- Addition and population (through Viewlog) of D_FORMATION_SCREEN_ASSIGNMENT
- Addition of D_DATA_FILE_SOURCE for tracking of data files
- Expansion of D_SURFACE_WATER to include all surface water stations (not just the Hydat stations); this includes CA's and GSC stations
- Addition of D_AGENCY table (for associated partner agencies with particular locations)
- Addition of D_GEOPHYSICAL_* tables for handling of geophysical data
- Addition of R_GEOL_UNIT_CODE table to standardize geology
- Addition of R_GEOL_FORMATION_CODE; to assist in understanding the formation nomenclature used by the Ontario Geological Survey as well as by the Oil and Gas Industry; it is expected that this table will replace R_GEOL_UNIT_CODE in next database version
- Addition of R_INT_REGULATORY_CODE; used to track the DWS or CofA numbers that a well might be tied to; associated with field in D_INTERVAL
- Tables dropped
 - D_INTERVAL_EQUIPMENT
 - D_INTERVAL_PUMP
 - D_HYDROMETER
 - D_HYDROMETER_READINGS
 - D_DOC_ASSOCIATION
 - D_DOC_BOREHOLE

- D_DOC_DOCUMENT
- D_INTERVAL_SURFACE_WATER
- D_SPLITSPoon
- D_WATERUSE_MUNICIPAL
- D_WATERUSE_CAPTUREZONE
- D_OPERATOR
- D_PROPERTY_ENVRISK
- D_PROPERTY_ENVRISK_EQUIPMENT
- D_PROPERTY_ENVRISK_MATERIAL
- D_PROPERTY_ENVRISK_WELL
- D_LAS_HEADER
- D_LAS_DATA
- D_OWNER_BAK
- L_PTTW_CITYTOWNSHIP_LIST
- L_PTTW_DISTRICT_LIST
- L_PTTW_UPPER_TIER_LIST
- L_SW_SUMMARY
- L_TEMPORAL_DATA_STATS
- L_WELL_WATERUSE_LINK
- L_WL_SUMMARY
- R_RD_METHOD
- R_WATER_TYPE_CODE
- R_MOE_PTTW_DATUM_CODE
- R_PTTW_UTM_ACCURACY
- R_WATERUSE_DISCHARGE
- R_WATERUSE_POND_TYPE
- R_WATERUSE_PUMPING_ROUTINE
- R_WATERUSE_WELL_TYPE
- R_NAICS_CODE
- R_CITYTOWNSHIP_LIST_ONT
- R_BH_CONSULTANTS
- R_MOE_TEST_METHOD_CODE
- R_MOE_CODE_MUN
- R_MOE_WATERUSE_CODE
- R_MUNICIPALITY_TYPE_CODE
- R_DATA_CORRECTION_SOURCE
- PTTW information is now found in the D_PTTW_* tables
- D_INTERVAL_PROPERTY can now be populated through SiteFX; includes the specific capacity values for the new MOE wells
- D_INTERVAL_TEMPORAL_SUMMARY provides a summary of the different types of temporal data in the database (counts)
- New temporal data should now go into D_INTERVAL_TEMPORAL_7
- BLOB's (Binary Large Objects) have been added to the database to make more efficient use of memory; this applies to D_INTERVAL_TEMPORAL_4 (data-

logger information) and D_INTERVAL_TEMPORAL_5 (Environment Canada and Hydat information); queries have been included in the database to search or extract information from these tables

- D_INTERVAL_TEMPORAL_6 provides baseflow separation results
- D_INTERVAL_MONITOR has been corrected for erroneous values
- D_LOCATION has had the following fields removed
 - LOC_BASIN fields
 - LOC_USER_ID
 - LOC_ACTIVE
 - WATERSHED_DEV
- R_PURPOSE_PRIMARY and _SECONDARY have been updated to include Oil and Gas wells as well as climate and surface water monitoring stations
- PGMN wells have been flagged with the unique Primary and Secondary purpose values 'Engineering' (3) and 'PGMN Monitoring Well' (57)
- D_INTERVAL has had the new field D_INT_REGULATORY_CODE added to store information regarding (for example) 'Drinking Water System Number'
- R_BH_DRILLER_CODE; additional drillers have been added (with their MOE code)
- LOC_QA codes (for location and elevation) were reviewed; in general, the following applies
 - Outcrops/Bedrock Outcrops/Geological Sections
 - LOC_QA_CODE; generally assigned a value of '4' (coordinates taken by GSC staff from 1:50000 mapping)
 - ELEV_QA_CODE; assigned a value of '10' (elevation taken from the 10m DEM obtained from MNR)
 - Climate Stations
 - LOC_QA_CODE; generally assigned a value of '7' (coordinates provided by Environment Canada); this will be changed to '5' (to be considered reasonably located)
 - ELEV_QA_CODE; assigned a value of '5' (no elevations assigned or required for these locations); this should be changed in a future database version
 - Surface Water Locations
 - LOC_QA_CODE; generally assigned a value of '2', '3', or '4' (coordinates provided by the CA's or the GSC)
 - ELEV_QA_CODE; assigned a value of '5'; however, the elevations in the D_SURFACE_WATER_TABLE are from the DEM; this should change in a future database version
 - PTTW
 - LOC_QA_CODE; generally assigned a value of '4' (coordinates taken by GSC staff from 1:50000 map)
 - ELEV_QA_CODE; ELEV_QA_CODE; assigned a value of '10' (elevation taken from the 10m DEM obtained from MNR)
 - Seismic Stations

- LOC_QA_CODE; assigned a value of '3' (coordinates provided by GSC)
- ELEV_QA_CODE; assigned a value of '3'; no elevations are currently found in the database
- Oil and Gas Wells
 - LOC_QA_CODE; assigned a value of '4' (coordinates provided by MNR)
 - ELEV_QA_CODE; assigned a value of '10' (elevation taken from the 10m DEM obtained from the MNR)
- UGAIS Wells
 - LOC_QA_CODE; assigned a value of '3' or '4' (coordinates provided from an OGS database)
 - ELEV_QA_CODE; assigned a value of '10' (elevation taken from the 10m DEM obtained from the MNR)
- MOE Wells
 - LOC_QA_CODE; original QA code from the MOE was retained
 - ELEV_QA_CODE; assigned a value of '10' (elevation taken from the 10m DEM obtained from the MNR)

20061222+

Incorporation of additional MOE boreholes/locations, primarily from the Crowe Region CA (refer to Appendix G for details).

20080110 (*'Notes For New 200[8] Database Release' Memo*)

The following should be examined prior to the next release of the database (in 2008)

- Review solid versus liquid sample handling
- D_INTERVAL_MONITOR; elevations not updated
- R_INTERVAL_TYPE_CODES; review
- D_INTERVAL; populate interval end dates (e.g. MOE wells should have the same start and end dates)
- York monitoring wells missing a DATA_SOURCE
- D_SURFACE_WATER; populate drainage areas for spotflow stations; differentiate between a surface gauge and spotflow location
- Reconcile D_PUMPTEST and D_PUMPTEST_STEP
- Documents - do we have link between the digital document and the database
- D_LOCATION_ACTIVITY does not match with R_ACTIVITY_CODE
- D_INTERVAL_TEMPORAL_3; most records do not have a reading type code
- D_INTERVAL_TEMPORAL_4; no relationship with D_INTERVAL
- D_INTERVAL_TEMPORAL_5; no relationship with D_INTERVAL
- D_INTERVAL_TEMPORAL_2; many locations where static values mixed between masl and fasl - SiteFX should make these consistent (converted)
- Some wells do not have a borehole record; some surface water stations have a borehole record; some outcrops do not have a borehole record

- D_INT_TYPE_CODE has not been assigned to all intervals - do so
- D_BOREHOLE_CONSTRUCTION; standardize the units used
- Check bedrock indicator (and lack thereof) in many cases
- Not consistent between D_INTERVAL_MONITOR (flowing tag) and D_PUMPTEST (FLOWING_RATE); correct
- Correct D_INTERVAL_TEMPORAL_2 records with no RD_VALUE
- Check surface water spot stations with no spot flow value
- Fix dates (some up to 2029) in D_INTERVAL_TEMPORAL_SUMMARY
- D_INTERVAL_TEMPORAL_2; pumping rates and flows not converted to 'system' rates
- Check DATA_ID usage and purpose in D_INTERVAL_TEMPORAL_1A/1B
- Check locations with no COUNTY_CODE and TOWNSHIP_CODE
- Add R_CONFIDENTIALITY_CODE with
 - 1 - Location information is open; no restrictions
 - 2 - Location information restricted to partner agencies; not Province; not Public
 - 3 - Location information restricted to agencies with jurisdiction over location
 - 4 - Location information restricted to CVC
 - 5 - Location information restricted to TRCA
 - etc ...
- Test pits currently only have a primary purpose of 'Government Mapping/Research'; review and change
- SiteFX needs adjustment to require a Coordinate QA code upon entry of new data
- Many locations have no LOC_COORD_QA code
- What data is attached to GSC seismic locations? Return to 2002-2003 usage
- PTTW locations have been assigned an elevation QA code; no elevation associated with them, change to reflect
- Many MOE wells indicated to have coordinates from the MNR in D_LOCATION_QA; change to come from the MOE
- Bring back the MOE elevation data to the respective _OUOM field and add a column for the MOE elevation reliability code in the QA table
- QA_Driller_Accuracy_Code, QA_Pumping_Code and QA_Water_Level_Static need to be updated for new MOE wells
- What is the purpose of the D_LOGGER tables (they are empty)
- D_PUMPTEST; some records that do not have a data source (or it is erroneous)
- D_SURFACE_WATER; move SW_GROUND_ELEV_SOURCE to D_LOCATION_QA; populate the drainage area for spot flow locations
- D_CLIMATE; add a start and ending year; include two new fields to be updated each time new data is entered
- D_CRITERIA; can this table be dropped (it is empty)
- Check boreholes - many have a DEM ground elevation of less than 75m
- Add note to the look-up fields in each table (i.e. Table-Properties) designating which R_* table it references

- R_BH_CONSULTANTS; was removed; do we want to keep it
- Merge or remove old document system (e.g. R_DOC_FORMAT_CODE, R_DOC_GROUP_CODE, etc...)
- D_GEOL_FEATURE; some descriptions of geology should be moved to D_GEOL_LAYER
- Add an interval type for a surface water flow gauge with logger and a second for surface water (only?)
- Correct wells with a data source of '7' to 'CLOCA'
- Develop a method for tracking soil/rock samples from boreholes
- Check some MOE wells that are based on new coordinates
- GRCA-PGMN and Lake Simcoe files were not incorporated due to poor quality; check and prepare for next time
- Reading name codes need to be tidied up - many duplicates and group codes not used appropriately
- Elevations in D_BOREHOLE table need to be repopulated (the _OUOM field)
- R_CONFIDENTIALITY table still needs to be updated
- D_CLIMATE; DEM elevation _OUOM fields should be empty
- QA code for climate stations should be changed to '5' - assumed to be reasonably located
- Surface water stations have no elevation received from Environment Canada; db should reflect this
- QA_DATA_SOURCE_CODE not equivalent to that in LOC_DATA_SOURCE_CODE; review
- QA_DRILLER_ACCURACY_CODE not updated; remove
- D_LOCATION_QA; update or delete fields
- Elevations of water levels in D_INTERVAL_TEMPORAL_6 are not being converted properly; correct
- Add a formation number column to D_GEOL_LAYER; GEOL_UNIT_CODE and SUBCLASS_CODE can probably be dropped
- BH_STATUS_CODE used haphazardly for non-MOE wells; review
- R_DOC_FORMAT_CODE can be dropped (tied to old document tables)
- R_EQ_* tables related to deleted equipment tables; remove
- R_ECDATA has been removed; re-think or explain deletion
- R_FEATURE_CODE descriptions revised for 'Water Found - Fresh' and etc ...; the description 'Water Found' can be removed from D_GEOL_FEATURE
- R_GEOL_CLASS_CODE; what is this table for
- D_GEOL_LAYER_TYPE code should be built into Material 1-4 lists (where they have been used and where they apply)
- R_GEOL_MAT*; these tables should be identical
- R_GEOL_ORGANIC; remove unless used
- GSC seismic locations are present in the db but have no data tied to them; review

***Database Release 4
(Unreleased)***

20080717 ('Management of Temporal Data' Memo)

The following direction or additions to the YPDT-CAMC database should be implemented

- De-blobbing the temporal files (when moving from MS Access to MS SQL Server)
- Inclusion of min-, max- and avg-daily water levels; partner agencies to maintain any higher-resolution datasets (or on a case-by-case basis)
- Specifying recommended update times (bi-yearly)
- Inclusion of water quality information (municipal pumping and other borehole sampling)
- Suggestion that water quality data from private domestic wells be NOT included [in the database]
- Inclusion of Provincial Water Quality Data and other miscellaneous datasets
- Inclusion of pumping data from the regional municipality's production wells (at a minimum, the daily pumping rate)
- Suggestion that short-term pumping information NOT be included
- Inclusion of streamflow data from both Conservation Authorities and the Provincial/Federal Hydat network

20081202 ('Agenda Items for Upcoming Database Meeting' Memo)

The 2006 database synchronization involved several key additions and changes to the database. The following broadly outlines the changes that were made to the database between the 2004 and the 2006 versions. (See also the information under the date '20061129', above).

Comments

- Temporal Data; Both of the Environment Canada Datasets have been updated: i) Climate data now extends to between February and April, 2005 (depending on the individual station); ii) Hydat data goes until December 31, 2004.
- Report Library; The D_Doc table and 12 associated R_Doc tables that have been added to the database. These tables describe the reports that have been scanned and are available on the website.
- D_FORMATION_SCREEN_ASSIGNMENT; this table is populated from Viewlog using the current geological layers. For every location, the table is populated with the top of each layer. The screened interval is also assigned to the aquifer in which it is interpreted to be situated. Note that in the D_Interval table there is a field (Int_Manual_Screen_Geology) that can be used to record the formation that the screen is set in outside of the Viewlog interpretation – ideally the two should match when the surfaces are updated.
- D_DATA_FILE_SOURCE; this is a new table that tracks data files that have been incorporated into the project – it is used to track files that have temporal data.

- D_SURFACE_WATER; This table previously only had the Hydat stations as well as the CRA spot flow stations (520 records). The table has been expanded to track all of the Surface Water stations including those from partner agency CAs as well as the GSC stations. The number of records has increased to 4,585 records. Most of the spot flow locations have not had the drainage area populated – this will be done for the next update.
- D_AGENCY; This table flags all locations as being linked to a particular agency. The table is set up to include wells within a ~5 km buffer of each agency. Any location can have more than one flag indicating that it is linked to more than one agency.
- D_GEOPHYSICAL_*; D_Geophysical_Log_Databin; Field_Details; Litho_Descriptions; Location_Details – these four tables have been added to deal with the borehole geophysics data collected across the area over the past few years.
- R_GEOL_UNIT_CODE; This table has been populated with both bedrock and overburden formation names used in the study area – the table currently links to the Geol_Unit_Code field in the D_Geol_Layer table and is to be filled out manually if a geologist wishes to assign a geological formation to a particular interval within a well.
- R_GEOL_FORMATION_CODE; This table was constructed to assist in understanding the formation nomenclature used by the Ontario Geological Survey as well as by the Oil and Gas industry. Certain Formation names are not standard between the two datasets. In the next database release this table will replace the R_Geol_Unit_Code table and will link to a revised field in the D_Geol_Layer table.
- R_INT_REGULATORY_CODE; This table has been added by Earthfx to possibly track the DWS # or CofA # that a well might be tied to - we'll see how and if it is used before determining whether to keep it or not. It is tied to a field in the D_Interval table.

Comments Regarding Elevations

There is a concern that we haven't been keeping a good track on how the elevation is assigned to a well and how it might change over time as the database is updated and changed. The problem is of concern from two different perspectives:

- In particular this is a significant issue with respect to making picks on the geology – if the elevation changes after picks are made then the picks must be adjusted to reflect the new elevation. It is recommended that the picks table be altered to add a new field and that Viewlog be adjusted to write the ground elevation at the time of picking to the picks table.
- If we have drawn – and circulated - cross-sections in the past and the DEM subsequently changes – then wells that might have previously been located at the bottom of a valley might now be on the shoulder or even at the top of the valley. Given the resolution and better DEM maps that have come along this is likely not to be a significant issue into the future.

The current three Elevation Fields in the D_BOREHOLE Table must be used more rigorously and effectively. Depending on capabilities of selecting the 'Best' Elevation on the fly we might add another field called BH_BEST_GND_ELEV (or to keep the 'Best' as BH_GND_ELEV).

For MOE Wells:

- The BH_GND_ELEV_OUOM field is to house the original MOE well elevation in feet or metres. We have to go back to the original MOE datasets and roll back this field to match the MOE value - we don't ever want DEM elevations in this field.
- The BH_GND_ELEV field is to house a surveyed elevation if one is available – in the case of MOE wells – where the MOE code is 1 the MOE value can be copied to this field. Should this field be renamed to BH_GND_SURV_ELEV? If there is no surveyed elevation this field will remain blank - we don't ever want DEM elevations in this field.
- The BH_DEM_GND_ELEV is to house the DEM elevation from the 10 m MNR DEM as provided to us a few years back. This field can be overwritten every time the database or DEM is updated.

For other wells:

- The BH_GND_ELEV_OUOM field is to house the original well elevation in feet or metres as derived from the borehole record or scanned report. For UGAIS, MTO, Oil and Gas wells etc. whatever elevation was provided will go into this field in its original units (we will not, at present, go back to the original datasets to check this). We'll leave it blank if the original is gone – we don't ever want DEM elevations in this field.
- The BH_GND_SURV_ELEV field is to house a surveyed elevation if one is available – if the elevation is noted to be a surveyed elevation on the borehole record then the elevation can be copied into this field. If there is uncertainty as to how the elevation was derived then this field will remain blank - we don't ever want DEM elevations in this field.
- The BH_DEM_GND_ELEV is to house the DEM elevation from the 10 m MNR DEM as provided to us a few years back. This field can be overwritten every time the database is updated.

For consistency I would like these three ELEV fields to be named in a consistent fashion

- BH_GND_ELEV_ORIG (rather than OUOM)
- BH_GND_ELEV_SURV
- BH_GND_ELEV_DEM

If we keep the 'BEST' field – then it could become BH_GND_ELEV (this then becomes the best elevation to use. This is what Sitefx uses to calculate geology, screens, etc)

All reference to elevations (and coordinates) should be removed from the surface water and climate tables.

The fields QA_ELEV_SOURCE, QA_ELEV_METHOD and QA_ELEV_COMMENT need to be reviewed and corrected.

Elevation Reliability Coding - ELEV_OUOM

Currently the MOE elevation reliability code has been overwritten in most cases and is lost. We need to go back and repopulate the MOE reliability code into the D_LOCATION_QA table. We should rename the field from QA_ELEV_CONFIDENCE_CODE to QA_ELEV_ORIG_CONFIDENCE_CODE – this field should then be repopulated with the original MOE value. In conjunction with this, the R_QA_ELEV_CONFIDENCE_CODE table should be re-named to R_QA_ELEV_ORIG_CONFIDENCE_CODE and expanded to account for the ability now for folks to estimate “original” elevations off of digital maps etc.

Given a an elevation confidence code of ‘0’, the following could be added

- Code 20 = Elevation obtained from BH log (unknown accuracy)
- Code 21 = Elevation obtained from BH log (<1 m accuracy)
- Code 22 = Elevation obtained from BH log (noted as surveyed <5 m accuracy)
- Code 23 – Elevation obtained from GPS (unknown accuracy)
- Code 24 - Elevation obtained from GPS (<5 m accuracy)
- Code 22 = Elevation estimated from Viewlog (~5 – 10 m accuracy)
- Code 23 – Elevation estimated from other GIS mapping (~5 -10 m accuracy)

We can add whatever else we come up against – the main thing is that this field addresses the accuracy of the ORIGINAL Elevation estimate – so [it contains] MOE plus extra stuff. Although typically we do not need to make “original” elevation estimates anymore since the DEM populated elevation would be the general default.

Elevation Reliability Coding - SURF_ELEV

Given that we are now going to have the Second elevation field reflect the Surveyed Elevations (BH_GND_ELEV_SURV) we must have another field in the D_LOCATION_QA table called - QA_ELEV_SURV_CONFIDENCE_CODE – this will link to a new R_ table which we can call R_QA_ELEV_SURV_CONFIDENCE_CODE

The codes would then be

- Code 1 – Consultant Report – Survey
- Code 2 = Golder Survey
- Code 3 = PJB BH Log – BH Drilled in bottom of Pit
- Code 4 = RG – TS Survey
- Code 5 = CRA TS Survey
- Code 6 = Mount Albert EA Study Survey
- Code 7 = MMM Survey (Murray Gomer)

- Code 8 = Driller Survey – from BH log
- Code 9 = King City Water Supply – Survey
- Code 10 = IWA Survey

This could be generalized or specific but this table would only relate to the Surveyed Elevations. This table will capture much of what is currently found in two fields in the D_LOCATION_QA Table, namely the QA_COORD_SOURCE and the QA_COORD_METHOD fields. Both of these fields are largely tied to various consultant surveys that we know about – so they would vanish from the D_LOCATION_QA Table.

Elevation Reliability Coding - DEM_ELEV

We also will do the same thing with the BH_GND_ELEV_DEM field. There will be a new field in the D_LOCATION_QA Table that will be called QA_ELEV_DEM_CONFIDENCE_CODE. This field will link to a new R table which we can call R_QA_ELEV_DEM_CONFIDENCE_CODE.

The codes would then look like (each time the elevations are replaced by a new DEM, a new code would be introduced)

- Code 1 = Ver 1 DEM – 30 m received from MNR 2001 (used between 2001 & 2004)
- Code 2 = Ver 2 DEM – 10 m received from MNR 2004 (used between 2004 & 2007)
- Code 3 = City of Toronto DEM – 50 m due to inaccurate MNR DEM (Used between 2001 and 2003)
- Code 4 = 1 m DEM provided by City of Toronto from Ortho photography – (used between 2008 and 2012)

Comments Regarding Geology

The geological descriptions (including both formation and comments fields in various tables) in the database have become confused and the purpose of some of the existing tables remains unclear. One issue we should resolve is the issue of Mat1/2/3/4 codes - we had decided early on to stick with the MOE's protocol of simple geological coding (Mat 1; Mat 2; Mat 3 - adding Mat 4). More complicated geological descriptions were to be entered into the "Full Description" memo field. The City of Toronto wells were granted an exception since they were imported in from a database – but these should be fixed now. Another problem is that similar information is being entered into different fields.

Proposed

- Go through the Material 1, Mat 2, Mat 3 and Mat 4 fields in the D_Geol_Layer table and look for ways to remove the more entangled descriptions (generally those with codes of 1000 or greater)
- Delete the R_Geol_Subclass_Code table (see D_GEOLOGY_LAYER below)
- Combine the R_Geol_Unit_Code table into the R_Geol_Formation_Code table - these should be merged into one table. In order to merge them we would have to add the few overburden units from Geol_Unit_Code to the Geol_Formation_Table (even though they are not proper “Formations”: Channel – clay; Channel – silt; Channel – sand; Channel – gravel; Iroquois; Oak Ridges; and maybe undifferentiated Paleocene Bedrock)
- Within the R_Geol_Formation_Code table – we should change the name of the Geol_Unit_Code to Geol_Formation_Number – the additional “Formations” that will come from the Geol_Unit_Code table will need to receive a Formation_Number.
- D_GEOLOGY_LAYER
 - remove the GEOL_SUBCLASS_FIELD (two classes, not clear the source of each)
 - change GEOL_UNIT_CODE to GEOL_FORMATION_CODE
 - for Oil & Gas records, the GEOL_TOP_ELEV has been populated but the GEOL_BOTTOM_ELEV has not
 - GEOL_NAME redundant (as it should be in R_GEOL_FORMATION_CODE); check and remove
 - GEOL_COMMENT contains many geological descriptions; review (should this information be elsewhere)
 - should a LAYER_NUMBER be re-implemented
 - DATA_SOURCE should be dropped (should be contained in D_DATA_SOURCE)
 - Material layer type ‘5’ (Bedrock) for many UGAIS wells do not have bedrock descriptions - review and correct
 - the GEOL_MAT_GSC_CODE needs to be populated
 - DATA_FILE; as DATA_SOURCE (above)
 - MAP_UNIT_NUMBER is only partially populated; if keeping, populate
- R_GEOL_CLASS_CODE; table should be dropped
- D_GEOPHYSICAL_*; additional records to be added from DGI
- should a new location type (for geophysical locations) be added OR should locations be flagged for geophysical information
- D_GEOLOGY_FEATURE; move feature codes to D_GEOLOGY_LAYER (should contain, mostly, water found information)
- all material tables should be made identical
- D_INTERVAL_SOIL and D_BOREHOLE_SAMPLE contain similar information; consolidate

Comments Regarding Screens

In D_FORMATION_SCREEN_ASSIGNMENT, do we want to keep the screen assignment (GEOL_UNIT_CODE) within this table or move to an interval table. If moved, then this becomes the D_FORMATION_TOP table only holding the geology from the gridded surfaces. Fields should be added to reflect surface version. Fields should be included for updated geological units.

Why are there differences (in counts) between D_INTERVAL and D_INTERVAL_MONITOR (checked - OK).

For D_INTERVAL_MONITOR

- the screened interval is captured within the Mon_Comment field – however this field again hasn't been used very effectively – there are a few formations in here and there are lots of comments on the sand pack extents - in addition, there [are a] number of records that are not tagged with how the screen was assigned
- there are a number of records with no top and no bottom depths or elevations recorded, these should be dropped; other have *_OUOM value but these have not been converted to elevations
- MON_TOP_OUOM; a number of records do not have a top recorded; these may or may not have a MON_BOT_OUOM as well

Comments - Miscellaneous Tables

- DATA_SOURCE and DATA_FILE
 - numerous tables have both of these fields, incompletely populated; many record similar information; review which tables should keep these fields (and how they are used)
- D_LOCATION_ACTIVITY
 - need to be more specific regarding what 'part' of the location is being updated
 - ACTIVITY_CODE; not generally populated; remove?
 - ACT_FLAG; what is this for?
- D_LOCATION_DEM; what is the purpose of this table
- D_LOCATION_QA
 - QA_COORD_CONFIDENCE_CODE_UPDATE; an empty boolean field; review how this is to be used
 - QA_COORD_SOURCE; this is not used in a systematic way - a methodology needs to be developed (possibly populated through SiteFX); should be a connection to D_LOCATION_ACTIVITY
 - QA_COORD_METHOD, QA_COORD_COMMENT and QA_COORD_SOURCE fields contain much duplication; standardize (e.g. through the use of R_UTM_COORD_METHOD_CODE)
 - QA_DATA_SOURCE_CODE; what is the value of this field and where did the information populating it come from; remove
 - QA_WL_STATIC_FIELD; what is this field used for; remove (information should be put somewhere else)

- D_LOG; unused since 2004; remove
- D_OWNER; remove unused fields (e.g. OWN_ADD_LOT, OWN_ADD_CON, OWN_ADD_COUNTY, etc ...)
- D_PUMPTTEST and D_PUMPTTEST_STEP; should these be combined
- D_SURFACE_WATER; the elevation and coordinate fields should be removed (the information should be duplicated in D_LOCATION)
- D_CLIMATE; as D_SURFACE_WATER, above
- D_VERSION; what is this table used for
- R_RD_NAME_CODE; review and remove equivalent names (move these to R_READING_NAME_ALIAS)

20081209 ('Minutes & Actions from Dec 9 Meeting - December 11, 2008' Memo)

Comments

- YPDT-CAMC report library and website to be run against a single (Master) database (i.e. no duplicate databases)
- One-way synchronization to be enabled first (two-way subsequently)
- Hidden S_* tables track SiteFX changes
- DATA_SOURCE and DATA_FILE should be dropped from all tables with the exception of D_DATA_SOURCE
- Location fields removed from D_CLIMATE and D_SURFACEWATER (duplicates of the fields in D_LOCATION)
- D_INTERVAL_PROPERTY; field names should be changed to be more specific and understandable
- D_DOCUMENT; only two _ID codes will be carried
- D_PUMPTTEST and D_PUMPTTEST_STEP; initially proposed to be made a single table; this idea has been discarded - keep them separate
- Incorporate tables to incorporate 'Grouping' concept; D_GROUP, D_GROUP_OBJECT, etc ...; should LOC_STUDY or LOC_AREA be incorporated as a series of groups? (with the idea that these fields would eventually be dropped)
- D_LOCATION_DEM; to be removed (replaced by D_LOCATION_ELEV)
- R_GEOL_FORMATION; information to be moved to R_GEOL_UNIT then dropped
- R_GEOL_SUBCLASS; leave for now
- GEOL_NAME to be renamed GEOL_LOCAL_NAME (to record a local name that may be of interest)
- Elevations should be included in 'Picks' table at time of picking; review of how elevations are handled in the various tables in the database and through SiteFX

Database Release 5 (First Released 2012-06-15)

200908 +

Initial evaluation of conversion between MS Access and MSSQL2008.

200910 +

Expansion of the D_DOCUMENT table and addition of supporting table to expand the capability of the database with regard to the report library. Inclusion of REG library.

200911 +

Testing of Viewlog with initial MSSQL2008 version. Correction and simplification of geologic material types.

200912 +

Re-evaluation of RD_NAME_CODE table begins (including updates). CLOCA interval data incorporated. Correction of pumptest data. Incorporation of interval data in MSSQL2008 database (i.e. un-blobbed).

201001 +

Incorporation of MTO wells started.

201002 +

Beginning of process of conversion from MS Access to MSSQL2008. Testing begins of replication process (over VPN) with between Downsvew and EarthFX. Bedrock wells - evaluation and correction. Screen lengths check and correction.

201004 +

Setup and evaluation of replication process begins. Interval and group types evaluated.

201005 +

Replication testing begins. Timeout problems due to size of interval-temporal tables.

201006 +

Use of Groups and LOC_STUDY formalized. Initial problems with 'identity' fields using distributed replication. Incorporation of 'Scarborough Report' data. Re-evaluation of D_DOCUMENT (and supporting) tables.

20100610,23,30 (*'Database Meeting with Kelsy - June 10, 23 and 30' Memo*)

Comments

- Assume that wells have been added within study geographic area (including SWP extensions) plus a 10km buffer
- Review of R_LOC_TYPE (removal of duplicates)
- All MOE wells will have a LOC_TYPE_CODE of '1'; all MOE wells will have an INT_TYPE_CODE of '15' (Screen)
- Wells with a COUNTY_CODE of '0' or '99' modified to '72' (Not specified)
- Wells with a TOWNSHIP_CODE of '69713' changed to '1500' (Not specified)
- MOE only uses '1' and '3' LOC_STATUS_CODE; add '13' (Alteration) and '14' (Other status); '0' code dropped

- Can the description of columns/tables be reincorporated from Microsoft Access version of database
- R_LOC_MOE_USE_1ST_CODE changes to match MOE WWDB (with the exception of 'Cooling & A/C' versus 'Cooling or A/C')
- R_LOC_MODE_USE_2ND_CODE made identical
- R_DRILL_METHOD_CODE
 - 'Direct Push' equivalent to 'Driving'
 - 'Digging' equivalent to 'Hand Auger'
 - 'Other' changed to 'Unknown'
 - 'Sonic' equivalent to 'Rotosonic'
 - 'Auger' changed to 'Power Auger'
- R_BH_DRILLER_CODE; to be repopulated; delete all BH_DRILLER_DESCRIPTION (for the ones 'Not Classified'), repopulated with 'MOE Driller No. XXX'
- R_MAT_CODE_* tables; '1115' and '27' changed to 'Other'; remove 'Cobbles' and 'Bedrock' (duplicate entries; '27' subsequently removed)
- R_GEOL_MAT_GSC_CODE no longer included with MOE WWDB
- GEOL_MOE_LAYER to be added (back) into 'D_GEOLOGY_LAYER' (for old wells in addition, if possible)
- R_FEATURE_CODE; need to determine codes '8' and '9' from MOE; D_GEOLOGY_FEATURE to be populated
- R_CON_SUBTYPE_CODE; should change back to MOE default but ... for now, all subtype codes in MOE database changed to match YPDT-CAMC
- Cleaned up R_CON_TYPE_CODE and R_CON_SUBTYPE_CODE
- D_DOCUMENT; DOC_USER_ID dropped; DOC_FORMAT_CODE should only be defined by two values - 'Digital' and 'Digital and Hard Copy' (only reports YPDT-CAMC has possession of should be included in the Report Library); R_DOC_GROUP_CODE removed;
- To deal with missing wells - all remaining MOE water well records will be translated into YPDT-CAMC format (and include all MOE wells in our database)
- R_CON_TYPE_CODE; 'Sand' changed to 'Annulus - Sand Pack'; 'Seal' changed to 'Annulus - Seal'
- For pump tests (from MOE), is GPM implied to be Imperial?
- D_PUMPTTEST; duplicates to be removed; day/month flipped (on second entry into the database)
- PUMPTTEST_METHOD_CODE and PUMPTTEST_TYPE_CODE moved from D_PUMPTTEST_STEP to D_PUMPTTEST
- R_PUMPTTEST_TYPE_CODE; 'Slug' and 'Packer' added; now consist of 'Constant Rate', 'Variable Rate', 'Slug' and 'Packer'; no recovery test (water levels flagged as recovery water levels as necessary)
- R_WATER_CLARITY_CODE; added as a look-up table

201007 +

Delivery of initial MSSQL2008 version(s) and testing begins. Querytimeout parameters modified to correct issues with interval-temporal tables. Trusted versus untrusted

connection problems with SiteFX/Viewlog. Corrected with new versions (of each). Replication through FTP is enabled.

20100728 ('Database Meeting with Kelsy - June 10, 23 and 30' Memo)

Comments

- Downsviiew and EarthFX servers replicating against each other (test database)
- Field/table descriptions must be added manually to new SQL Server database
- For replication, should R_* tables be uni-directional only (i.e. from Master to Partners; changes at Downsviiew only); exception would be R_GROUP_* tables
- SiteFX constantly asks for password when accessing SQL Server through non-Windows Authentication (SQL Server login/passwords)
- Grouping functions work in SiteFX
- With respect to missing wells (e.g. Barrie), take all wells from all Counties covered by YPDT-CAMC project and add them to the database
- Review location status versus borehole status

20100826 ('Database Meeting - August 26, 2010' Memo)

Discussed and approved database backup procedure using 'Simple' database model. York reading name codes found to not (necessarily) be equivalent to YPDT-CAMC codes (SiteFX adds new codes whenever reading name not present). Now corrected. Proposed adding of last modified user and date to all tables (for tracking purposes). 'Identity' field problems with primary keys (with regard to replication). Decision made to keep.

Data source and DATA_ID issues

- DATA_DESCRIPTION poorly populated
- Many tables have a text-based DATA_ID field (instead of numeric); corrected
- Any data incorporated is given a new DATA_ID, even if related to a previous DATA_ID - is this a potential problem

Twelve baseflow methods (calculations) now included in the D_INTERVAL_TEMPORAL_2 table. All bad pumping test data have been removed. Target for first version SQL Server version of 'Master' database slated for 20100903.

201009 +

First test of the replication process with CLOCA (over VPN). Inclusion of SYS_TEMP* fields in every table.

20100914

First working (Master) version within MSSQL2008 resulting from conversion of the original Microsoft Access database. All subsequent changes are to this version only.

201010 +

Problems with D_GEOPHYSICAL* tables (blobbed data; no primary keys). Temporal data incorporated into two interval-temporal tables. Methodology for and assignment of

geologic unit to screen interval begins. Calculation of 'mbgs' for all intervals incorporated.

201011 +

RD_NAME_CODE (along with RD_NAME_ALIAS) re-evaluations continue. Assessment of required tables for SiteFX and streamlining of table information begins. Inclusion of surficial geologic unit. Bedrock depth calculation moved to a view. Additional fields included in interval-temporal 1A/B for method tracking and uncertainty.

20101112 (*'Database Meeting - November 12, 2010' Memo*)

Comments

- Viewlog and connections to SQL Server database problems
- Duplicate location corrections (including temporal data)
- R_PURPOSE_SECONDARY_CODE; Remove primary purpose fields; add 'Not Viable - Abandoned' and 'Other - Research'
- Check of 'Reported Screens' in YPDT-CAMC database against MOE WWDB
- Check for 'Open Hole' in overburden wells
- Make sure primary and secondary purpose codes assigned correctly as part of the interval type designation
- Review MOE well records and re-assign to one of the following interval types
 - Reported Screen
 - Reported Open Hole
 - Overburden Well - 1m Screen Assigned
 - Assumed Open Hole (Screen assigned from bottom of casing - bottom of hole)
 - Assumed Open Hole (Screen assigned from top of bedrock - bottom of hole)
- For non-MOE wells, screens need to be investigated for interval category to be assigned

20101119

Comparison between YPDT-CAMC database and MOE WWDB, for wells where there is only one screen reported there is some discrepancy between OUOM fields; to be overwritten with MOE data

201012 +

First phase of table removal and data transfer (in general to D_INTERVAL_TEMPORAL_* tables). Evaluation of necessary 'Views' begins. Versioning by date recommended and approved. Added 'last modified' tracking columns for both date and user to most tables. Addition of SYS_LAST_MODIFIED and SYS_LAST_MODIFIED_BY. Correction of some MOE wells, changing 'Early Warning Well' to 'Supply Well' - re-examining records for reported screens. Removal of duplicates and transfer of temporal data for various Ballantrae, Nobleton and Alton wells.

201101 +

R_RD_NAME_CODE, R_RD_TYPE_CODE and R_READING_GROUP_CODE tables modified. Evaluate usage of D_BOREHOLE_SAMPLE and D_INTERVAL_SOIL. Sample data included from 'Scarborough Report' (Eyles and Doughty, 1996).

20110112 ('Database Meeting Notes - January 12, 2011' Memo)

Comments

- Interval type conversion
 - Interval type conversion nearly complete (30000 wells left without new code when running conversion rule test)
 - Proposed rule to ignore 'Open Hole' record in overburden wells; if no screen reported, given 1ft screen (at bottom)
 - 'Abandoned' MOE wells should have no interval applied as a default screen
- BH/Screen Diameter
 - No (metric) conversions should take place with regard to diameters (e.g. inches converted to m's); correct
- D_BOREHOLE_SAMPLE
 - Data should be incorporated into D_INTERVAL_TEMPORAL_1A/1B and D_INTERVAL_SOIL
- UNIT_CODE
 - Check of unit codes necessary for SiteFX
- Analytical Method Table
 - Incorporation of analytical methods to be included as a look-up table but not included into SiteFX; these codes should be added to D_INTERVAL_TEMPORAL_* and D_INTERVAL_PROPERTY as necessary
- Data Logger Files and YPDT data; determine 'best' methods by which to incorporate data (e.g. straight from logger; through analyzed spreadsheet)

201102 +

Presentation of replication methodology (including distribution of partner databases from the 'Master') at Peel. Incorporation of DEM information (multiple sources). Proposed replication testing at Peel. Review diameter in borehole table and correct any problems. Check and removal of duplicate interval-temporal data.

201103 +

Second phase of table removal and data transfer. Evaluation of fields within tables required by SiteFX. Review of requirements necessary for evaluation of replication methodology at Peel – overview documentation prepared and sent. Geologic screen assignments revisited. Duplicates in geology-layers checked and removed. Review dates and fix errors.

20110302 ('Database Meeting Notes - March 2, 2011' Memo)

Comments

- Elevations
 - DEM elevations to now be stored in D_LOCATION_ELEV; will house 'Master DEM' field for unsurveyed locations and will then be used to populate BH_GND_ELEV, SW_GND_ELEV and CL_GND_ELEV (in associated tables)
 - Removal of these fields from any table other than D_LOCATION_ELEV shelved, currently, as this would require many changes in SiteFX
- Interval Types
 - 'Screen Information Omitted' requires checking
 - Some 'Reported Screen' intervals with no top or bottom _OUOM values; generally from having multiple screens within a single interval (from MOE); should be deleted; note, different diameters and slots sizes recorded - need to be re-examined
- Bedrock Elevation to be calculated on demand; proposed changing 'Material 1' code of 'Bedrock' to 'Boulder' - used to flag false bedrock indicators
- D_GEOLOGY_LAYER; Re-examine removal of GEOL_CONSISTENCY_CODE, GEOL_MOISTURE_CODE, GEOL_TEXTURE_CODE, GEOL_ORGANIC_CODE (need to figure out a way to save the data from the Toronto BHs that use them)
- NULL Fields; need to generate a list of required SiteFX fields; looking at removal of unnecessary/unused fields
- R_UNIT_CODE; clean up table
- D_BOREHOLE_SAMPLE; has missing records - need to re-import
- Incorporation of analytical methods in D_INTERVAL_TEMPORAL_* tables

201104 +

Removal of D_BOREHOLE_SAMPLE and incorporation of data into D_INTERVAL, D_INTERVAL_SOIL and the D_INTERVAL_TEMPORAL_* tables. Evaluation of QA confidence codes and inclusion of out-of-area codes. Duplicate intervals and geology removed. Example consultant view prepared. Top and bottom depths of intervals calculated and included in interval-monitor table.

201105 +

Geologic unit codes updated and assigned. Include model geologic unit assignments to intervals. Example databases for each agency prepared and distributed (for testing only). Halton and York database checks and comparison with master database. Check and remove invalid DATA_ID's. Checked PGMN locations.

201106 +

Addition of manual formation assignments to formation assignment table. Trust relationship problems between workstations and primary server (MSSQL2008). Incorporated old MOE water level data. Changed structure of D_LOCATION_ELEV table.

201107 +

Start incorporation of historical chemistry data. Water level imports. Halton and York initial database fixes (before import). Various reading units reviewed (e.g. water levels) and modified.

201108 +

Addition of reading name aliases to include all interval data. Methodology for incorporation of logger data developed. Geologic and 'consultant' views added. Corrected problems with cascaded deletes (from within SiteFX). Start of Halton temporal data import. Correction of document table errors.

201109 +

Initial review of to-date views. Amalgamation/correction of 'flows' reading name codes. Addition/update of reference elevations. Added views for use with Viewlog.

201110 +

Corrections to isotope data. Removal of bedrock fields from D_BOREHOLE table (now calculated on-the-fly using V_General_BHs_Bedrock).

201111 +

Addition of new tables to match latest SiteFX version(s). Initial training session (at Peel) using the MSSQL2008 database. Invalid coordinates 'nulled' and assigned appropriate QA code. Correction of D_BOREHOLE table due to missing BH_ID's. Removal of invalid PICKS information. Initial setup of replication-over-web begins. Corrected pump information with regard to conversion between liters and gallons. Comprehensive review of required views. Moved specific capacity information to D_INTERVAL_TEMPORAL_2.

201112 +

Amalgamation (and correction) of all PICKS information.

201201 +

Simplification of all water level codes. Database manual preparation started. Correction of interval-temporal lab data.

201202 +

Simplification of all pumping codes. General View's (i.e. non-expert) setup. Evaluation of coordinate QA's versus elevation QA's. Removal of lab info from the interval-temporal tables with no associated data. Removal of duplicate records in D_INTERVAL_TEMPORAL_2.

20120224

Version of database standardized to this date (unreleased).

201203 +

D_AGENCY abandoned and functionality moved to D_LOCATION_AGENCY (with change in structure). Use of type codes for tracking logger (and other) manufacturers implemented. Removed duplicates found imported interval-temporal tables. Applied type code methodology to interval-temporal tables. Corrected depths using D_BOREHOLE_CONSTRUCTION and D_GEOLOGY_LAYER.

201204 +

Views for pumping and chemistry updated (returning additional fields). Bibliographic view implemented. Further correction of chemistry data, interval-temporal 1A/B tables (invalid number of SAM_ID's). Re-evaluation of chemistry and water level R_RD_NAME_CODES. Unit codes modified to correct for conversion problems. Standardized to 'mbref' for all depth units. Populated all 'null' OUOM fields in the temporal tables.

201205 +

Replication-over-web setup and trouble-shooting (for all partners). 'Identity' field problems return.

20120528

Addition of views to OAK_SUP for listing technical information on all tables in the master database.

201206 +

Replication-over-web setup issues continue.

20120612

Check re-seeding of ident fields in various tables.

20120615

Version of database standardized to this date.

20120703+

Initial evaluation of D_INT_FORM_ASSIGN for duplicate intervals (i.e. duplicate top- and bottom-depths for a single INT_ID).

20120727+

Correction of D_GEOL_LAYER for Peel Region (spatial extent). Evaluation of methodology (in SQL) for determining shallow water table surface.

20120808

Initial evaluation of determining sand and gravel thicknesses as an indicator of aquifer availability.

20120809

Addition of V_WL_Average_lt_20m to OAK_SUP (for determination of shallow water table surface).

20120815

Correction of TRCA PGMN water levels (where no water level is recorded).

20120816

Initial evaluation of master-partner database checking (in SQL).

20120821

Corrected top- and bottom-depths of geologic layers in D_GEOL_LAYER.

20120823

Moved incorrectly located static water levels from D_INT_TEMP_1A/1B to D_INT_TEMP_2. Creation of LOC_ID QA/QC checks by EV (first draft).

20120827

Updated/corrected location elevations in D_LOC_ELEV and D_BOREHOLE.

20120907

Updated/corrected D_INT_REF_ELEV.

201210+

Updated OAK_SUP with V_SG_* views (for determining sand and gravel thicknesses for less-than and greater-than-or-equal-too 20m depths). Updated OAK_CHECKS with V_D_INT_FM_ASSIGN_* for checking whether an existing model has been applied against an interval and well as evaluating the top- and bottom-depths of screens (for correctness/conversion). Added V_D_DOCUMENT_..., V_D_INTERVAL_... and V_D_LOCATION_... to be used for checking (when importing) existing _ID's.

20121016+

Reviewed D_INT_MON. A number of records are present (i.e. multiple monitor tops and bottoms) tied to a single INT_ID - this needs to be corrected. Likely tied back to MOE WWDB nested well configurations (checked against two intervals - this turned out to be the case). These should be changed to multiple locations each with a single interval.

20121024

Updated all valid intervals with the geologic models in D_INT_FM_ASSIGN - added the interpreted information for the Durham Model (2007). Re-calculated the ASSIGNED_UNIT field (to accommodate the additional model in the assignment logic; see Section 2.4.1).

20121118

Updated descriptions to all tables (D_ and R_) in database.

20121120

Added missing locations and elevations to D_LOC_ELEV.

20121128

Correction of ground elevations in D_BOREHOLE. Examination and initial correction of D_INT_REF_ELEV reference elevations. Added descriptions to all views in database.

20121130 (Database Meeting)

???? NOTES ?????

20121211+

Correction of chemistry data and sample information in D_INT_TMP_1A and D_INT_TMP_1B.

20121212

Updated D_INTERVAL setting all INT_TYPE_CODEs of 118 to 29 ('Soil' to 'Sample'). Changed 'Sample' to 'Soil or Rock' (29) and dropped 'Soil' (118). Added values to 'R_REC_STATUS_CODE' to tag information that is not to be used in certain instances. Anything less than 100 is kept. Started work on D_INT_REF_ELEV removing all intervals whose INT_TYPE_CODEs are not related to having a reference elevation. Added missing intervals whose INT_TYPE_CODEs do relate (these would include type codes: 18,19,20,21,22,27,28,101,102,112). Removal of duplicate chemistry readings.

20121214

After discussion, decided to (in this version of the database) use INT_TYPE_ALT_CODE in R_INT_TYPE_CODE as a field for grouping similar interval 'types' together, for use in simplifying queries. As this is a text field, these groups must be specified as text strings. Currently only the grouping 'Screen' has been applied. The usage of this field can be applied to other reference tables as well (as needed). In subsequent version of the database, this may be replaced with a function similar to the R_RD_NAME_CODE and R_READING_GROUP_CODE tables.

20121217

Finished working on updating D_INT_REF_ELEV; updated all possible NULL REF_ELEVs with 'ASSIGNED_ELEV + 0.75m' values (~100000 updated; ~20000 remaining NULL values with either no ASSIGNED_ELEV or no BH_DRILL_END_DATE). Removal of duplicate intervals (i.e. intervals formed from having multiple screens within a single borehole).

20130109

Corrected D_BOREHOLE null bottom elevations (when available, assigned either the minimum bottom elevation of geology - from D_GEOLOGY_LAYER - for borehole OR minimum bottom elevation of construction element - from D_BOREHOLE_CONSTRUCTION - for borehole).

20130110+

Correcting D_GEOLOGY_LAYER for null GEOL_BOT_OUOM values begins (mainly a problem for Oil & Gas wells).

20130122

Updated D_BOREHOLE, modifying the BH_BOTTOM_ELEV, BH_BOTTOM_DEPTH, BH_BOTTOM_OUOM and BH_BOTTOM_UNIT_OUOM fields. As many of the boreholes present in this table (~22000) had NULL or invalid bottom elevations (or depths), these were corrected (previously) based upon construction or geology details (D_BOREHOLE_CONSTRUCTION and D_GEOLOGY_LAYER, respectively). For the remainder, any screen present in the D_INTERVAL_MONITOR table was used to extract a bottom depth (plus 0.3m) then converted to an elevation. If the borehole did not have a screen, the BH_BOTTOM_ELEV was given the same value as the BH_GND_ELEV and the remainder of the _BOTTOM fields assigned NULL values.

20130214+

Updated D_LOCATION_QA; all those locations with a NULL QA_COORD_CONFIDENCE_CODE have had a '9' (i.e. 'Unknown') assigned. York has started comparing records of their municipal wells - some problems that need to be corrected.

20130218

Addition of 'V_D_LOCATION_Geometry' to OAK_SUP (for spatial objects in the 'Master' database). Addition of the BH_BEDROCK_ELEV column to D_BOREHOLE - this is to be used instead of the dynamic-search implemented in 'V_General_BHs_Bedrock' (for speed considerations).

20130306

Addition of the field MON_WL_ELEV_AVG to D_INTERVAL_MONITOR. Contains the average water-level at the interval across the whole temporal record available.

20130311+

Initial examination of monthly water levels as compared to the DEM show invalid water level elevations (far above surface). Start of the correction of elevation and water level values for locations.

20130313+

Many borehole depths (seemingly) invalid (not including 20130122, above). Start of re-examination and population of borehole depths (in D_BOREHOLE).

20130314

Added 'Alternate' ('6'; Class Code 'Lithology', '1') to R_GEOL_SUBCLASS_CODE. This is used to differentiate between geologic interpretations for the same location/borehole.

20130319

Corrected stratigraphic top-and-bottom problems for various MOE bedrock wells (previously missing top- or bottom- depth/elevations). Updated values copied into D_BOREHOLE (BH_BEDROCK_ELEV).

20130320

Added 'Original (Invalid)' ('7'; Class Code 'Lithology', '1') to R_GEOL_SUBCLASS_CODE. This is used to tag invalid geologic interpretations (usually problem units from the MOE, mainly lack of elevation/depth data). The 'Original' subclass description is now changed to 'Original (or Corrected)'.

20130401+

Start of process for addition of tagged locations for each partner agency into D_LOCATION_AGENCY. Methodology now incorporates data from the OAK_SPATIAL accessory database.

20130405

Added INT_NAME_MAP to D_INTERVAL. This is to be used for plotting of intervals in plan (i.e. overhead) view and matches LOC_NAME_MAP in D_LOCATION. These names for intervals/locations can be no longer than ten characters in length.

20130407

Added soil intervals – those with tops and bottoms – to the D_INTERVAL_FORMATION_ASSIGNMENT table and applied the CORE Model (2004) interpretation. These are to be incorporated along with the screen intervals in future updates.

20130408+

Started correcting intervals in both D_INT_MON and D_INT_SOIL creating top- and/or bottom-depths and elevations as appropriate (or available). These are subsequently to be used in populating D_INT_FM_ASSIGN.

20130419

Alternate PICKS2 created with indexes and keys setup using the method described in Appendix H (dated 20130419b). This is to be used for the foreseeable future to avoid the pick-delay problems with the original PICKS table. This is to be fixed in the next version of the database.

20130501

Dropped RD_NAME_CODE of '630' ('Water Level – MOE Well Record') from R_RD_NAME_CODE and R_READING_NAME_ALIAS. Use the appropriate R_RD_TYPE_CODE to access MOE water levels.

Updated R_GEOL_UNIT_CODE, standardizing the YPDT-CAMC model layer names. Added an AQUIFER tag (a NULL, '0' or '1' value; '1' indicating an aquifer unit) to be used when assigning formations in D_INTERVAL_FORMATION_ASSIGNMENT

(allowing the 'aquifer' layer to be assigned preferentially if the top and bottom of the screen do not match).

20130506

Added ~130000 rows to D_LOCATION_ALIAS – these are BORE_HOLE_IDS (from the MOE WWDB) that were otherwise missing. As WELL_ID is not a primary key, we used WELL_ID, EAST83, NORTH83 and ELEVATION to match against LOC_IDS in D_LOCATION. This has been included as a comment if any checking is required. Note that there still a number of BORE_HOLE_IDS not matched to LOC_IDS in the database as these four fields were not matched exactly. This still needs to be corrected.

ELEV_ORIGINAL is now populated in D_LOCATION_ELEV based on these new aliases (i.e. linked back to the MOE WWDB 201304 cut).

20130510

Corrected formations – mainly related to bedrock elevations – for another set of boreholes (mostly MOE). Update D_BOREHOLE with new elevations.

20130604

Updated D_LOCATION_GEOM; start of updating D_LOCATION_AGENCY based upon temporary CHK_* views (as tables). Re-initialized York and LSRCA databases.

20130607

Completed updating D_LOCATION_AGENCY.

201307

Start work on the next version of the database (tentatively referenced as OAK_20130701_MASTER or v20130701). This includes: complete SQL scripts for creation of all tables and their associated keys, indexes and triggers; evaluate methods (including functions and stored procedures) to implement partner-specific ranges of values without resort to identity fields (this involves the user of the bigint type; problems using this are to be examined); re-evaluate the replication method including the presence of SYS_TEMP* fields (which must be included for access by SiteFX but should not be replicated) and SiteFX specific S_* tables.

201308

Initial evaluation of the CLOCA partner database for import of differences into the master db. Problems encountered between conflicting primary keys – partner db's and the master db can end up with the same primary keys in some tables. One table uses incremental values (an S_* table) which guarantees this. The development of the official QA/QC procedures/methods is necessary.

20130826

In D_LOCATION, where LOC_COORD_OUOM_CODE is null, copied the LOC_COORD_EASTING and LOC_COORD_NORTHING values (if non-null) into their respective *_OUOM fields and assigned a value of '4' (i.e. UTMz17 NAD83) to the

LOC_COORD_OUOM_CODE field. Various coordinate corrections (and QA_COORD_CONFIDENCE_CODE corrections) during this process.

20130927

Corrected layer info in D_GEOLOGY_LAYER (examining units and NULL values) and updated D_BOREHOLE as required.

20130930

Corrected layer info in D_GEOLOGY_LAYER (examining units and NULL values) and updated D_BOREHOLE as required.

20131004

Corrected chemistry parameters in D_INTERVAL_TEMP_1B and added aliases (as necessary and appropriate) into R_READING_NAME_ALIAS.

20131007

Focussed look at the CLOCA partner database for inclusion within the master db. Note that this is to be used as a test case for the QA/QC methods/procedures (and is the continuation of an earlier test).

20131011

CLOCA partner database information has been added to the master db.

20131018a

Correction of the location of 'Conductivity' (temporal) values, moving them from D_INTERVAL_TEMPORAL_1A/1B to D_INTERVAL_TEMPORAL_2 (note that it is possible that values can be present in the 1A/1B tables – the analysis would have to be lab based only).

20131018b

Addition of 'Active (Monitoring)' and 'Inactive (Monitoring)' codes to the R_LOC_STATUS_CODE table (to differentiate from the MOE WWDB understanding of 'Active' and 'Inactive').

20131022 and 23

Updated D_LOCATION coordinates (translated as necessary) prior to examining elevations. QA code '117' is beginning to be used to tag those locations outside of the YPDT-CAMC buffered area.

Re-examined elevations in D_BOREHOLE and D_LOCATION_ELEV. Each of BH_GND_ELEV, BH_GND_ELEV_OUOM and BH_DEM_GND_ELEV should now match (for those locations with a valid QA, i.e. not '117') that of the ASSIGNED_ELEV in D_LOCATION_ELEV. The BH_GND_ELEV_OUOM values have been copied/converted to ELEV_ORIGINAL (if not already present). The BH_GND_ELEV_UNIT_OUOM has been standardized to 'masl' (the BH_GND_ELEV_OUOM value was converted as necessary).

20131202

Evaluated possible problem with 'V_General_Consultant_Hydrogeology' – turns out that multiple instances of an INT_ID in D_INTERVAL_FORMATION_ASSIGNMENT results in the doubling (or other multiplier) of the number of water levels associated with the interval. Note that this is not an error with a view but with the population of the the formation assignment table – there should only be one row/tuple for each INT_ID. The supplementary check database view 'CHK_D_INT_FM_ASSIGN_Multiple_INT_IDs' should be run periodically to correct this. Note also that these 'blank' rows (i.e. the additional INT_IDs) have been added in many cases by the 'NT Authority' or 'YKREGION\SQLAgentAdmin' user – this is not mapped to an actual user (so where are these runs coming from? Is this a SiteFX issue? - supposedly not).

20131209

Fixed various data constraint issues (i.e. information currently in the database not correctly matching 'new' constraints) when developing methodology and testing conversion/translation of data between database and spreadsheet formats. This included (but were not necessarily limited to) the tables D_BOREHOLE, D_DOCUMENT, D_GEOLOGY_LAYER, D_GROUP_LOCATION, D_INTERVAL_REF_ELEV, D_LOCATION_ALIAS and D_LOCATION_PURPOSE.

20131211

Updated D_INTERVAL_TEMPORAL_2; all rows with no values, no OUOM values and no comments were removed. This should be consistently applied in the future.

20140115

Deleted ~1100 records from D_INTERVAL_FORMATION_ASSIGNMENT that contained repeating blank/empty rows for various INT_IDs. There are multiple SYS_LAST_MODIFIED_BY users (no SYS_USER_STAMP) from both Y-C and the partner agencies – is there some automatic processing running? Monitor these occurrences.

20140117

Modified V_Random_ID_Creator to randomize ID's between specified values (actually a lower value and range). A view needs to be created for each user/partner (according to the uniqueIDrange as found in S_USER for SiteFX). Currently V_Random_ID_Creator_MD and V_Random_ID_Creator_REG has been created.

20140206

Started in 20130923, the methodology for the MOE WWDB updates to be included as part of the YPDT-CAMC database are near-completion. The final steps concerning the inclusion of boreholes with no geology remain. Notes regarding possible issues and corrections applied (as needed) are shown here.

Notes on 2013 MOE Well Import File

- D_Location - the field Loc_Start Date - is typically empty for MOE wells - usually the only MOE date gets stored in the Drill_End_Date field in D_Borehole.
 - NOT MODIFIED
- D_Location - Loc_Name_Alt1 - should read "MOE Well 2013 - Name Withheld by MOE' - this will make it consistent with the other MOE wells that Kelsy brought in in 2010.
 - MODIFIED
- D_Location - If I do a Group By query on D_Location using Master_Loc_ID and Loc_ID - I can't see any of the famous MOE multi wellls - did we decide to give each BH its own LocID? Even then though we should have them linked through the Master Loc_ID (which would be the Loc_ID of the Well that has the MOE Geology) - right?.
 - CHECK
- D_Location - I think Sitefx needs the wells to be assigned a Site_ID of 1 (for YPDT-CAMC).
 - MODIFIED
- D_Location - the tblWWR table contains the MOE data source code - did none of the new wells have a code to bring in? should go to Loc_Data_Source in the D_Location table.
 - MODIFIED (18 bhs)
- D_Location - We should also assign the Loc_Confidentiality_Code of 1 - for accessible to all - for these wells.
 - MODIFIED
- D_Location_Alias - looks OK
 - NOT MODIFIED
- D_Location_Purpose - This is a big one - The MOE Use 1st and MOE USE 2nd have to be properly translated to our Loc_Use_Primary and Loc_Use_Secondary Codes - they are not the same - see attached table below.
 - MODIFIED
- D_Location_QA - What about the Elev_Codes in the Loc_QA table? In the Loc_QA table - we don't have two columns for Elev Code (as we do for the UTM Code) - so we still have a number of old MOE Codes in here that are no longer relevant. Should we add another field in the table for Elev_Reliability_Code_OUOM? then the Elev_Code would be switched to either a 10 or a 1. As an aside - do you know where the QA_DATA_SOURCE_CODE field in the QA table links to? Should the QA_ELEV_SOURCE field in this table be updated to MNR DEM ver 2 - for most wells? (i.e. remove the "MNR DEM Ver1" terminology from here). All new wells should get a code of QA_Elev_Confidence_Code of 10 right? With appropriate comment in the QA_ELEV_SOURCE Field. Also for the UTM Codes we should probably populate the QA_COORD_METHOD from the "Location Method" field in the tblBore_Hole table. (as long as the code is not p1 through p9).
 - MODIFIED (some notes not addressed – external to import procedure)
- D_Location_Elev - looks ok
 - NOT MODIFIED

- D_Borehole - we still need all of the Elev fields to be populated so that they appear in Sitefx. I think we can make them all the same for the new BHs.
 - MODIFIED; elevs to be added at a later step
- D_Borehole - the BH_Bottom (Depth, OUOM and Unit_OUM) fields need to be filled in - I think we need to take the deepest of either the geology or the BH_Construction depths
 - MODIFIED; elevs and depths to be added at a later step
- D_Borehole - Let's fill in BH Dip to be 90 and BH_Azimuth to 0
 - MODIFIED
- D_Borehole - can we populate the BH_Bedrock_Elev field?
 - NOT MODIFIED; this is a process external to import procedure
- D_Borehole - we should populate the MOE_BH_Geology_Class field - the data is found in tblBore_Hole (CODEOB field) - and is linked to the code_Well_Type Table
 - MODIFIED
- D_Borehole_Construction - looks good
 - NOT MODIFIED
- D_Geology_Feature - it looks to me like Sitefx doesn't touch this table - I think we should just do the conversions and populate the "Final" fields and bypass the OUOM fields.
 - NOT MODIFIED; as more than MOE data can be included here, there should be a separate procedure for dealing with these values (depths)
- D_Geol_Layer - looks good - As an aside - do you know what the Geol_LayerType_Code field links to?
 - NOT MODIFIED; GEOL_LAYERTYPE_CODE reference table?
- D_Interval - why do we have 125 Int_Type = 28 (Screen Information Omitted)? Why aren't these changed to 19 (1ft above bottom of BH)? Can't tell if its because they have no bottom depth since that field isn't populated in D_Borehole
 - NOT MODIFIED; no depth data associated with these intervals/locations (neither geology, construction or screen details)
- D_Interval - I also wanted to confirm that the ones you coded with an Int_type of 21 - were indeed bedrock holes
 - NOT MODIFIED; bedrock determination (elevation) is through a separate process not related to the import procedure; this should be an external check (i.e. evaluating all INT_TYPE_CODE tagged intervals with value '21')
- D_Interval - we should populate the Int_Confidentiality_Code with a 1 - indicating share with everyone
 - MODIFIED
- D_Interval - I think that all of the existing Intervals have a "-1" in the Int_Active field - I have never used this field and I don't think it will ever be used - but maybe we should be consistent and put a "-1" in here?
 - NOT MODIFIED; INT_ACTIVE is a bit field - '-1' indicates a 'true' value for Microsoft products; should automatically be converted (as it is not '0') on import
- D_Interval_Monitor - We have 8,492 Intervals that we are pulling in and 8,583 records in D_Interval_Monitor - are all of these situations where there are more than

one pipe at a location? or are there some intervals that have a multi-part screen and if so have we handled them OK? There is a field in D_Interval called Int_More_1_Part - should we fill that in if there are two or more parts to a screen?

- NOT MODIFIED; this check was originally made (see Section G.10.15) – for duplicates and multiple static water levels assigned to a single INT_ID; re-examined manually (with no results modified); note that overlapping screens (and, elsewhere, geologic layers) need to be investigated external to this import process – as such, those checks are not made here (manually, 1 overlapping screen found)
- D_Interval_Monitor - looks like you have introduced a coding for the Mon_Screen_Material field - is that correct (its written out in existing table)? Do you have a corresponding "R_" table?
 - MODIFIED; no, this is a direct copy from the MOE data; is the MOE using the codes from _code_casing_material?; the assumption has been made that this is the case and the relevant code has been updated
- D_Interval_Monitor - what is the "rnum field for? just temporary?
 - NOT MODIFIED; all rnum fields are for internal (before import) use only
- D_Interval_Monitor - How about populating the Mon_Top_Depth_M and Mon_Bot_Depth_M while we do the import?
 - NOT MODIFIED; this is a separate process outside of the MOE import/conversion
- D_Interval_Monitor - is it easier to infill the Mon_WL_Elev_Avg for these MOE wells now - just use the static WL?
 - NOT MODIFIED; this is a separate process outside of the MOE import/conversion
- D_Int_Ref_Elev - looks OK except for numbers - there are 8,492 records in here - but we have 8,583 records in D_Int_Monitor - the extra pipes should also be assigned a Ref_Elev - right? or are there two (or more) part screens - see above?
 - NOT MODIFIED; these are all multi-part screens; one ref elevation per INT_ID
- D_Int_Temporal_2 - I see a bunch of WL in here where the date is "null" and others where it has been defaulted as discussed to July 1, 1867 - what is the difference between the two sets of WLs?
 - NOT MODIFIED; the default dates only apply to pumping/recovery data (where the time interval is of interest); are we imposing dates on static water levels?; these are for wells with no completion or starting date
- D_Int_Temporal_2 - There are quite a few dates between 2002 and 1951 - are you sure that these are correct? I would have suspected that all of the wells we are pulling in would be drilled between 2006/2007 and 2012/2013 - it is possible that MOE added a few old wells in the meantime - but just want to make sure.
 - NOT MODIFIED; there are 67 locations/intervals with dates prior to 2007-01-01; this matches to 670 values in the D_INT_TEMP_2 table
- D_Int_Temporal_2 - I think that you have brought in too many WL duplicates to this table - e.g if you look at IntID -2128160263 o -2111097950 - for both of these there are 27 WLs tied to the Int - all are the same on the same 1867 date/time - can you see

what the issue is with this? Don't know if its just the 1867 ones or if there are others - just check IntID -2132667689 - and it also has 11 duplicate WLs (as a guide we should probably have about 8,441 x 5 or about 40,000 WLs in the D_Temporal 2 Table at the most - since many of the MOE wells have 0 or only 1 water level - the existing table has 58,239 measurements).

- NOT MODIFIED; these are pumping/recovery levels and static water levels
- D_Int_Temporal_2 - I only see 3,415 static water levels (Rd_Name_Code = 628) for the 8,441 wells that we are bringing in - are you sure there aren't more static WLs - this seems low?
 - NOT MODIFIED; checked but no additional static water levels located; this assumes that these levels are stored in 'TblPump_Test'
- D_Int_Temporal_2 - There are quite a few WLs where the Reading Value is less than 75mASL - I see that the OUOMs are quite ridiculous - lets see how many remain when the duplicates are removed from the table - but maybe we can check to see if the OUOMs are elevations instead of depths?
 - CHECK; but ... this may be a check to be performed within the db itself as, is likely, there will be similar errors
- D_Int_Temporal_2 - don't know what Rec_Status Code is for but it looks like it is typically populated with a 1 - should we do similar?
 - MODIFIED
- D_Pumptest - the field "Flowing_Rate_IGPM" - should only be populated if the monitor is flowing naturally - it doesn't record the same value as the Rec_Pump_Rate_IGPM - so it should be largely blank except for a few values when the monitor is flowing. MOE might have this screwed up - I think Albert and I went through and fixed records in the DB a while ago -
 - If the Monitor Flowing field is -1 - then we assume it is really flowing and we can leave the value alone - regardless of whether the Rec_Pump_Rate and the Flowing_rate are the same - some are really high though) (77 instances where they are the same; 1 instance where Flowing>Rec; and 4 instances where the Rec>Flowing);
 - If there is no Monitor_Flowing Flag and the Rec_Pumping Rate is the same as the Flowing Rate - then we assume it is not flowing and we delete the Flowing_Rate_IGPM value (2359 instances).
 - If there is no Monitor_Flowing Flag and if there is a flowing rate and no Rec_Pump_rate - then we can assume it is flowing and add a -1 to the Monitor_Flowing Flag (2 instances).
 - If there is no Monitor_Flowing Flag and if the Flowing Rate is less than the Rec_Pumping Rate - then we add a -1 to the Monitor Flowing field and assume it is really flowing (17 instances).
 - If there is no Monitor_Flowing Flag and if the Flowing Rate is greater than the Rec_Pumping Rate - then we assume there is an error and the monitor is not flowing - delete Flowing_Rate value (1 instance).
 - MODIFIED
- D_Pumptest - only 125 of 3038 records have a PumpTest_Method_Code - seems low - is that right?

- NOT MODIFIED; the remainder have been given a '0' value ('Unknown'; translated to NULL)
- D_Pumptest_Step - I think the import is messed up in here - there should only be multiple PumpTest IDs in here if the test was done at a variable pumping rate - if the rate is constant - as is the case for most/all of the MOE wells - then they should only appear once in here. There are many cases where they appearing multiple times - up to 13 times - I think in our existing table we only have 2 pumping tests where the pumping rate was changed during the test. We should have about 3038 records in this table - or less if no pumping rate is specified.
 - MODIFIED; this has been modified to only record the varying pumping rates (and their time intervals) for any particular PUMP_TEST_ID (which will now result in, basically, a single record for each identifier); this reduced to ~2700 records
- D_Pumptest_Step - Typically we have started the pumping test at 12:00 AM on the day it occurred - and then based on how many water levels were provided (usually four at 15, 30, 45 and 60 minutes - we stop the test at 1:00 AM - I see many of your test starting at 12:03 or so and ending 1 minute later - don't know how you populated these times - but they should be dictated by the WLS provided.
 - NOT MODIFIED; these have been examined and found to be correct
- D_Pumptest_Step - are the fields "mum", "testlevel", "testlevel_uom", and "testtype" just temporary?
 - NOT MODIFIED; all lowercase fields are only temporary holders (of data) and are not for final import

20140605-06

Started in 20130923, the methodology for the MOE WWDB updates to be included as part of the YPDT-CAMC database are complete (refer to Section G.10 for methodology). This new information has now been added to the database. Notes regarding possible issues and corrections applied (as needed) are shown here.

Notes on MOE 2013 Well Import

D-Location – 16525 Wells coming into DB

Min LocID = 240000035

Max LocID = 241680698

D_BOREHOLE TABLE

- Ground Elevation - Max = 529.6 mASL
- Ground Elevation - Min = 67.4 mASL – this well is in Lake Ontario (checked) (10 wells (all have same WellID) - with zero elevation) BATHYMETRY SUBSTITUTED
- Reduce the # of decimal places in the BH_GND_ELEV; BH_GND_ELEV_OUOM, BH_DEM_GND_ELEV, BH_BOTTOM_ELEV, BH_BOTTOM_DEPTH and BH_BOTTOM_DEPTH_OUOM to 2 places SEPARATE PROCEDURE

- 5,148 BHs have no specified depth; 3 have a depth of <0; 4 have a depth of 0 DEPTH <0 FIXED
- Max Depth = 1167.5? – 2nd largest is 445 m before that (once errors are fixed – the greatest depth should be 402 m)
- Min Depth = .1524
- 835 BHs with no drill_end_date (ok)
- Min Drill_End_Date = 06/07/1951
- Max Min Drill_End_Date = 22/03/2012
- Many wells with no BH_Geol_Class (ok)
- A few MOE errors that I caught – we need to fix either before or after import.....
 - MOE ERROR - LocID 240087270 – Change Depth from 1167.5 m to $1167.5 \times .3048 = 355.85$ m (note the geology only goes to 333 m – I think MOE messed up units. FIXED
 - I can't figure where you got the depth of 445 m for LocID 240097354 – the deepest element I can find for this well is the plug at 6.75 m – just maybe check to see that this isn't a more systemic error. FROM BH_CONS; FIXED
 - MOE ERROR - I think the depth of 395 m is incorrect for LocID 240001642 – both the Casing and the Fm go to 30.5 m – I think that is more reasonable GEOL_LAY CORR
 - MOE ERROR – LocID 240938223 – Fm table goes to 255 ft – I think the MOE units are wrong and should be feet – therefore depth = $266 \times .3048 = 81.08$ GEOL_LAY CORR
 - I can't figure where you got the depth of 250 m for LocID 240416215 – the deepest element I can find for this well is the Geology at 258 ft - therefore depth = $258 \times .3048 = 78.64$ m – just maybe check to see that this isn't a more systemic error. GEOL_LAY CORR
 - MOE Error – LocID 240500279 – everything is in ft except the Plug – likely an error – depth should be $180 \times .3048 = 54.86$ m GEOL_LAY CORR
 - LocID 240183435 – Plug From indicates 55 ft – depth should be $55 \times .3048 = 16.76$ m (look for Plug From for a depth if none available – LocID – 240183511 (11 ft); LocID 240183453 – (17 ft) DONE
- CHECK FOR ADDITIONAL TOP OF PLUG NO BOTTOM FOR NULL DEPTHS (DONE 20140527)

D_BOREHOLE_CONSTRUCTION TABLE

- Looks like top and bottom were exchanged if the top of the construction element was deeper than the bottom – good

- Some errors in BH Diameter – some too big – some too small – looks like mostly an issue of inappropriate units (Max Diam = 6125 inch) – should likely be 6.25 inch – also several 625 inch – all should be 6.25 inch – typically bored wells are 3 ft in diameter (36 inch (91.44 cm) MODIFIED
- Smallest BH Diameter = 0.0041 cm (a few 0 diameter) – should be evaluated once in DB and appropriate changes made NOT MODIFIED
- Only 8 Con subtypes used (10, 16, 21, 23, 24, 25, 31, 32) – seals (31) and casings only – no sand packs

D_GEOLOGY_FEATURE_CODE TABLE

- No codes 7 in table (Iron Water found)
- Mostly fresh (Code =1 – 3002 records) or untested (Code = 8 – 1722 records)
- Shallowest water found = 0.2 m / deepest water found = 550 ft
- 10,224 Water found records (732 Well Ids have more than 1 Water Found Record)

D_GEOL_LAYER TABLE

- There are 28 layers where the top is below the bottom – I didn't check in more detail – but wonder if it makes sense to switch the Top and Bottom fields so that they are proper. THIS IS NOT STRAIGHTFORWARD - THERE ARE MANY CASES WHERE THE DECIMAL PLACE HAS BEEN MISPLACED AND ETC FIXED
- If a layer has no top and no bottom specified – should we even bring it into this table? Don't know why there are all these blanks? There are 1664 records (layers) in the table where the top and bottom are null and Mat 1 = 0 – should we just delete these?
- There are around 90 cases where the Mat 1 assignment is 0 – but there are regular values assigned to Mat 2 or Mat 3 – wonder if we should do the same thing you did when the Mat 1 code was empty – move the assignments from Mat2 &/or Mat 3 up to Mat 1? MOVED MAT2 TO MAT1; MAT3 DIDN'T SEEM REASONABLE

D_INTERVAL TABLE

- Table looks good
- there are 16,525 Intervals coming into the DB –
 - Int_Type Code =
 - 21 – Assumed Open Hole (Bot. of Casing to Bot. of BH) (28 Ints);
 - 18 - Reported Screen (6,061 Ints);
 - 19 – Overburden – Assumed (1 ft Screen above Bot. of Hole) (2,989 Ints);

- 22 – Assumed Open Hole (Top of Bedrock to Bot. of Hole) (2,299 Ints);
 - 28 – Screen Information Omitted (5,148 Ints)
 - ADDED 123 – TOP INFO ONLY
- There are equal number of BHs and Intervals – therefore no Loc contains more than one screen – right? AN INTERVAL CAN HAVE MORE THAN ONE SCREEN IN A SINGLE PIPE (133)

D_INTERVAL_MONITOR TABLE

- 133 Int IDs have more than 1 record in this table – we should look for those cases where it appears to be two pipes in one BH (e.g. the diameter of plastic pipes changes – from 2 inch to 1 inch) – other cases are simply a change in slot size and that is OK. IN SOME CASES (3 OUT OF 3), THE DEPTH VALUES EXACTLY MATCH SUCH THAT THE SCREENS APPEAR CONTIGUOUS – SHOULD ALL OF THESE 133 BE EXAMINED AND SEPARATED
- MOE Error - D_Interval_Monitor – there are 208 cases where the Mon Top OUOM is greater than the Mon Bot OUOM – how about we switch them so that the top is always less than the bottom (unless units are m(ft)asl) FIXED
- D_Interval_Monitor – why don't we make the “Mon_Screen_Slot” a number field instead of Text? Also in our DB – I guess we should check with Sitefx – it might need text. NOT AN ISSUE WILL BE CONVERTED UPON IMPORT (IS REAL IN MASTER DB)

D_INTERVAL_REF_ELEV TABLE

- Can we reduce the number of decimal places to 2 in the Ref_Elev and OUOM fields WILL BE RECTIFIED ON IMPORT AUTOMATICALLY (CHANGE IN TYPE)
- For the 10 Ints where we have no Ground Surface (I think they are in Lake Ont – can we remove the Ref_Elev of 0.75 mASL – just make null FIXED FOR BATHYMETRY

D_LOCATION

- There are 347 locations that have more than 1 Well tied to them (linked using Master Loc_ID)
- Well-ID 7140211 – has 258 wells tied to it; Loc ID 7140275 has 234 wells tied to it – this is totally unreasonable (should make a note to Tim at MOE) – when I searched in the MOE DB to see if these numbers were correct – I wind up with 246 wells tied to 7140275 and 272 wells tied to 7140211 – did we make a mistake bringing these into our format – are we missing some of the BHs tied to these multi-0location well ids? DIFFERENCES IN COUNTS DUE TO INVALID COORDINATES; NOTE DEPTHS COME FROM SPECIFIED MOE DEPTH

- Why are there 10 LocIDs with no elevation? No coordinates or no DEM?
BATHYMETRY, NOW NONE

D_LOCATION_ELEV DONE

- looks good – none of the imported wells has an ElevRC of 1 – therefore no elevations to preserve.

D_LOCATION_QA

- There are 1,125 wells where the Loc_Coord_Confidence_Code (i.e. the one you assigned) is lower than the Loc_Coord_Confidence_Code_Orig – some are flagged with “MOE 201304 Import - Coordinate Conversion (possible error)“ – but not all of them. Wonder why these wells had the codes changed and what the possible import error might be? IN SOME CASES THE UTM COORDINATES WERE CORRECTED (IF POSSIBLE); IN OTHER CASES, THE UTM COORDINATES WERE INVALID SO A GEOCODING METHODOLOGY WAS IMPOSED – THESE ARE TAGGED WITH A SEPARATE CODE AS THEY WILL NOT BE AS ACCURATE (IF CORRECT) AS SPECIFIED COORDINATES
- Were any of the Wells in Zone 18 and converted to Zone 17 equivalents? Do we retain that in the UTM Zone – or somewhere? Just curious – the UTM OUOM match the UTM Final – so are we losing this info and do we care? THE ORIGINAL COORDINATES, WHETHER Z18 OR Z17 HAVE BEEN MAINTAINED

20140606

As part of the MOE WWDB import (201304 version), various corrections for other locations were applied using standard procedures/methodology as developed for the water well database.

D_INTERVAL_MONITOR rows were found with no bottom elevations/depths; these were found to correspond to zero thickness bedrock wells. In these cases, ‘0.3m’ screen intervals were imposed above the bottom of the well.

201406

Started assembling information for creating a new version of the database; both this document and Appendix H (‘Current Problems’) will be examined for necessary changes. The schema created by SiteFX will be reviewed and both additions and modifications will be included as necessary.

20140716

An SSMA_Timestamp should be incorporated into each table – they help avoid errors when Access updates records in tables linked to SQL Server. Review ‘Supporting Concurrency Checks’ from Microsoft documents. In particular, ‘The timestamp column increases automatically every time a new value is assigned to any column in the table. Access automatically detects when a table contains this type of column and uses it in the WHERE clause of all UPDATE and DELETE statements affecting that table’ (Stack Overflow, 2012).

20140723

At the moment, ranges of values include NULL, ‘-1’ through ‘2’. The ‘-1’ values will be changed to ‘1’ and ‘0’ will be changed to NULL. What is ‘2’ indicating?

201408

Started implementing the new version of the database. Corrections will be made on-the-fly with regard to errors while copying information between the old and the new versions.

20140908

Updated R_LOC_WATERSHED1_CODE removing multiple ‘Rouge River’ indicators.

20140909

D_INTERVAL_REF_ELEV has NULL values for REF_ELEV_START_DATE – this has been corrected (assigned the ‘holder’ value ‘1867-07-01’). A note should be made to populate this field upon data entry (this should be a constraint in the new version of the database).

In D_LOCATION_QA, what is QA_COORD_CONFIDENCE_CODE_UPDATE (a ‘bit’ field) being used for? Should this be removed (likely, yes).

For D_LOCATION_VULNERABILITY:

Should this table be re-evaluated?

‘AVI_Feb2003_Final 3 tier’ column name should be changed.

‘Water Table Sept 04 from model’ column name should be changed.

Added R_GEOL_LAYERTYPE_CODE into OAK_20120615_MASTER (non-replicating) and populated manually from the original Microsoft Access database. The ‘0’ value was not included (no description was tagged to GEOL_LAYERTYPE_CODE ‘0’). This value (i.e. ‘0’) was removed (assigned a NULL value) from D_GEOLOGY_LAYER (the GEOL_LAYERTYPE_CODE field).

20140910

For D_INTERVAL_FORMATION_ASSIGNMENT:

SYS_RECORD_ID has been removed as both a key and a field – there should be only a single row per INT_ID. Refer to Section 2.4.1 for the reasoning behind this.

Do we want to re-think the setup of this table? Instead of adding additional columns for each model, should we simplify the table and add a look-up table for model codes? An example would be:

INT_ID,ASSIGN_CODE,ASSIGN_DATE,GEOL_UNIT_CODE,MODEL_CODE,
VDIST

Where ASSIGN_CODE (a look-up table) would consist of (for example):

Top Layer
Bottom Layer
Next Layer
Previous Layer
Assigned Unit
Override Unit
Manual Unit

And MODEL_CODE (a look-up table) would consist of (for example):

Core Model 2004
Regional Model 2004
Extended Core Model 2006
Durham Model 2007
East Model 2010

Note that VDIST would only be populated for ‘Next ...’ and ‘Previous ...’ records.

There are various problems with this table (invalid GEOL_UNIT_CODE; invalid INT_ID) that should be addressed.

20141017

D_INTERVAL_TEMPORAL_1B and D_INTERVAL_TEMPORAL_2 have had their SYS_RECORD_ID's substituted for ranges of values that have been assigned to PEEL, CLOCA, YORK and NVCA. This is to, temporarily, avoid errors when incorporating temporal data in the database where the primary keys conflict. This arises from each of the partner agencies having a subset of the total database – keys can exist in the master that do not exist in their copy (and will conflict during replication). Note that some sort of fix should be applied to all tables with identifiers.

20141022

D_INTERVAL_MONITOR is missing screened intervals (as specified in D_INTERVAL); this includes INT_TYPE_CODE's of: 18, 19, 21, 22 and 27. Each is assigned tops and bottoms as appropriate. See write-up in Section G.23.

20141110

D_LOCATION_PURPOSE is setup in a manner such that multiple purposes can be assigned to a particular location – this can take the form of multiple ‘rows’ of information. Upon examination, it is found that there has been locations with more than two purposes assigned them (i.e. other than a single row with PURPOSE_PRIMARY_CODE and PURPOSE_SECONDARY_CODE). There is an end date indicator (PURPOSE_DATE_END) but this is not being used. In this case the multiples are repeating purposes (i.e. duplicates) for that particular location – these duplicates have now been removed. However, more than two purposes can be assigned to any particular location, still (see the PTTW dataset). If this is to continue, an indicator field – PRIMACY – is suggested which will tag the rows in the order that they should be considered.

20141111

Corrected location issue with regard to the MOE WWDB 201304 import – geocoding issues placed a subset of wells (~900) in the likely incorrect location. This should continually be checked (using D_LOCATION_QA) when working with these locations. QA_COORD_SOURCE and QA_COORD_COMMENT contain the relative details regarding the coordinate assignment.

20141112

Re-populated the contents of D_LOCATION_GEOM so as to incorporate changes in the D_LOCATION table (including the update of the MOE 201304 imported boreholes).

Examined resulting table (through plotting). Corrected locations with invalid coordinates or assigned the invalid tag of ‘117’ in D_LOCATION_QA.

It was noticed, during these coordinate corrections, that at times a 6-digit number actually conforms to a lat/long designation in ‘ddmmss’ format. This is primarily found for Environment Canada climate stations and surface water sites. This was then checked against other locations in D_LOCATION with coordinates modified as appropriate. If addresses were found, an attempt at geocoding was performed.

20141201

Values within the temporal tables are found to have RD_NAME_CODEs that do not have a READING_GROUP_CODE assigned. These have been updated as appropriate

(either by modifying the RD_NAME_CODE and RD_NAME_OUOM or by applying a READING_GROUP_CODE to the RD_NAME_CODE). Note that in the case of temperatures, some seem to exceed that which would be appropriate to groundwater but are associated with a 'reported screen'.

Users are assigning parameters to invalid temporal tables (e.g. herbicides in DIT2; water levels in DIT1B). This is likely due to the default SiteFX setup. RD_NAME_CODES have been evaluated and re-assigned as necessary.

20141215

Reset invalid 'Temperature' records, reassigning them to the specific RD_NAME_CODE for their interval type (either a barometric logger or water level/temperature logger).

Corrected invalid DIT2 and DIT1B rows to appropriate table.

201502+

Start if review of 'new' database format (views and tables).

20150224

D_BOREHOLE and D_LOCATION_ELEV elevations are once again out-of-sync. Some of these are tagged as 'surveyed' but still have their elevations changed – once this tag is in place, no elevation changes should ever again be made. Corrected all of the changes (based on the QA_COORD_CONFIDENCE_CODE).

Updated all elevations (BH_GND_ELEV, BH_GND_ELEV_OUOM and BH_DEM_GND_ELEV) setting them to equivalent values (namely the ASSIGNED_ELEV); assigned BH_GND_ELEV_UNIT_OUOM to 'masl'. Where the location did not have an ELEV_ORIGINAL, the BH_GND_ELEV_OUOM was used to populate the field (converting to 'masl' as appropriate).

Updated/corrected all BH_BEDROCK_ELEV where the elevation has changed (subsequent to the correction of elevations, above).

Removed 'soil' intervals (1) and 'surface water spot stage elevation' intervals (27) from D_INTERVAL_MONITOR. Removed duplicates from D_INTERVAL_MONITOR. Starting correcting of top- and bottom-screen intervals based upon various assumptions (e.g. '0.3' metres above bottom of hole for INT_TYPE_CODE '19'). Started re-examining incorporation of various screens into D_INTERVAL_MONITOR based upon various assumptions (e.g. as in the preceding sentence).

20150225

Incorporated 'new' intervals into D_INTERVAL_MONITOR with INT_TYPE_CODE 21 where:

INT_TYPE_CODE 21 - Assumed Open Hole (Bot. of Casing to Bot. of Hole)
these are assumed bedrock wells (i.e. they have casing that extends to approx top of bedrock)

Boreholes with a difference between the bottom of the hole and the bottom of casing greater than 0.1m were incorporated as is. The bottom-of-casing is considered to be the top-of-bedrock (and an update was made in D_BOREHOLE).

20150227

Finished updating/modifying INT_TYPE_CODE '21'; reassigned those (to '123') that were not appropriate. Approximately 4000 do not have elevations due to invalid coordinates. Update required some evaluation of BH_BEDROCK_ELEV and changes in D_GEOLOGY_LAYER; for the latter, a review of 'fbgs' OUOMs (which tend to be integer values) and 'mbgs' OUOMs (which tend to be real values) should be evaluated. It appears that some current 'mbgs' should be re-tagged as 'fbgs'.

20150303

Updating INT_TYPE_CODE '28' (Screen Information Omitted). UGAIS wells (DATA_ID '2076806159', i.e. DATA_DESCRIPTION 'MNDM - Ontario Geological Survey (UGAIS Wells)') are assigned INT_TYPE_CODE '22' (Assumed open hole, top of bedrock to bottom of hole) where the BH_BEDROCK_ELEV is greater than 0.3m above the BH_BOTTOM_ELEV. Note that only a subset of these records contain actual water level readings.

20150305

Updating INT_TYPE_CODE '28' (Screen Information Omitted). Changed to INT_TYPE_CODE '22' or '19' based upon the presence of a BH_BEDROCK_ELEV. Deleted those records with no valid bottom depth of any kind (the remainder were from the MOE WWDB 201304 import). Twenty-four of these remain. There are >8000 INT_TYPE_CODEs '28' remaining.

20150306

Update D_BOREHOLE_CONSTRUCTION; corrected CON_TOP_OUOM and CON_BOT_OUOM based upon casing information. Units are converted from 'inch' and 'cm' to 'mbgs'. Where CON_UNIT_OUOM is NULL, numeric 'integer' depths are assigned 'fbgs'; numeric 'real' (i.e. with a decimal) depths are assigned 'mbgs'. Corrected (added) depths based upon construction details.

20150313

Summary review of 'new' database views (and related tables).

DB REVIEW – March 13, 2015

Views

1. V_Consultant_Document – OK
2. V_Consultant_General
 - Well Depth is missing
 - Change Surface_Elev to Ground Elev
 - Change Interval_Monitor_Num – to Number_Of_Screens
 - Change BH_Diameter_M to BH_Diameter_cm – (or whatever the correct unit is – it is not metres)
 - Given that the Number/StartDate/EndDate of the Water Levels and Water Quality info is already in the “V_Consultant_Hydrogeology” View and the intent is to provide all of these together – I think we can remove these fields from this table.
 - Wonder about the Several “Water_Found” fields at the end of the table – with no depth provided I don’t know if this is all that useful to include in the view
3. V_Consultant_Geology – OK
4. V_Consultant_Hydrogeology
 - We should add “Screened Formation” to this view
 - What is the difference between Pumptest_Start and Pump_Start_Date (and Pumptest_End vs Pump_End_Date
 - Similarly – doesn’t Pumptest_Date_MDY relect the Pumptest_Start – they look the same – except maybe the time?
 - Does the Field “Pump_Readings_Num” record the number of water levels during the pump test? Or a change in pumping rate? Maybe change to “Test_WL_Readings_Num”?
 - How come the “Pumptest_Rate_IGPM” is in whole numbers and the “Pumptest_Rec_Rate_IGPM” has decimal places – check the numbers to make sure that we didn’t convert Gal to L or vice versa – make decimal places consistent between two fields. No decimal places unless we converted from L to gal – then maybe 2 decimals.
 - Wonder if we should add “Daily_Pumping_Volume_Num” (and start/end dates) to provide indication of the data availability for the main pumping wells.
5. V_Consultant_Pick
 - This View doesn’t return any records.
6. V_General

- Purpose_Primary and Purpose_Secondary (Fix spelling) – do not appear to pull the data correctly – they are all null – should pull description from the correct R tables;
- Move “QA_Coord_Code” – so that is right beside “Coord_Confidence” field;
- Move “Status_Code” so that it is located beside the “Status” field;
- Move “Type_Code” so that it is located right beside the “Type” field – also change name from “Type” to “Location_Type”
- What does “Access_Code” refer to? Do we need it here?
- I see some of the 2006 PTTW records in here – did you create this DB before we deleted and replaced them?

7. V_General_Borehole

- Since all records in View are from the D_Borehole - we can probably remove the “Type” field (are there any outcrops in this view? Maybe we remove them from here.
- Change “Surface_Elev” to “Ground_Elev”
- Change “Bottom_Depth” to “Bottom_Depth_m”
- Change “Data_MDY” to “Date_Drilled_MDY”
- Change “Interval_Monitor_Num” – to “Number_Of_Screens”
- Change “Screen_Geol_Unit” to “Screened_Geol_Unit”
- What does “Screen_Geol_Unit_Elev” point to? Top/Middle/bottom of geol Unit? Top of screen? Bottom of Screen?
- I don’t think we would need “Status_Code”, “Access_Code” or “Type_Code” in this View

8. V_General_Borehole_Bedrock

- Do we need this View anymore? It’s the same as previous except the Bedrock Field is never null. If we keep it – then we might want to change name to “Bedrock_Reached” _ notice there are lots of wells in the table where the screen is in an overburden unit and not in the bedrock.

9. V_General_Borehole_Outcrop

- Change name to V_General_Outcrop
- I think the Alternative_Name and the Study fields are both Null for all Outcrops – maybe we can remove them
- Remove “Type” field?
- Change name of “Bottom_Depth” to Bottom_Depth_m)
- Remove “Borehole_Diameter_cm” – not applicable
- Do any of the outcrops have a date associated with them? If not then remove – if yes then let’s change the name from “Date_MDY” to “Date_Logged_MDY”

- Remove fields “Interval_Monitor_Num”; “Interval_Soil_Num”; “Screen_Geol_Unit”; “Screen_Geol_Unit_Elev”; “Status_Code”; “Access_Code” and “Type_Code” all are not applicable and/or blank/consistent.
- Maybe we could add a Field “Num_of_Geol_Layers” – and record the number of records for each outcrop in the D_Geol_Layer table.

10. V_General_Chemistry

- What is in this view? Does it include everything from all of the following Chemistry Views? If yes – should we change name to “V_General_Chemistry_All”
- Change “Name” to “Loc_Name”
- What does “Formation” refer to? Change to “Screened_Formation”.
- I don’t think anyone would care about “Bedrock_Elevation” in here – delete field.
- Change “Group_Code” to the description – or add description and leave the Code.
- What does Int_Access_Code refer to? Remove?
- What does Int_Active_Code – refer to? Is it updated/used? Remove?
- This view also returns all of the soil grain size analyses and other lab data that pertains to soils – which I guess is why we called it “Lab” before. Don’t know if we should change name or change what the view returns (Int_Type <> “Soil or Rock”) – we should discuss.

11. V_General_Chemistry_Bacteriologicals

- Same points as for V_General_Chemistry

12. V_General_Chemistry_Extractables

- Same points as for V_General_Chemistry
- Maybe we should add the “Qualifier” Field so that all of the “<” signs pop up

13. V_General_Chemistry_Herbicides_Pesticides

- Same points as for V_General_Chemistry
- Maybe we should add the “Qualifier” Field so that all of the “<” signs pop up

14. V_General_Chemistry_Ions

- Same points as for V_General_Chemistry

15. V_General_Chemistry_Isotopes

- Same points as for V_General_Chemistry

16. V_General_Chemistry_Metals

- Same points as for V_General_Chemistry
- Maybe we should add the “Qualifier” Field so that all of the “<” signs pop up

17. V_General_Chemistry_Organics

- Same points as for V_General_Chemistry
- Maybe we should add the “Qualifier” Field so that all of the “<” signs pop up

18. V_General_Chemistry_SVOCs

- Same points as for V_General_Chemistry

- Maybe we should add the “Qualifier” Field so that all of the “<” signs pop up
19. V_General_Chemistry_VOCs
 - Same points as for V_General_Chemistry
 - Maybe we should add the “Qualifier” Field so that all of the “<” signs pop up
 20. V_General_Document
 - Remove “Study” field – not used for documents
 21. V_General_Document_Bibliography – OK
 22. V_General_Field
 - What does this View Return? All Field Data? Change Name to “General_Field_All”?
 - Change “Name” to “Loc_Name”
 - Change “Parameter_Type” to “Measurement_Descriptor”
 - Remove “Bedrock_Elev” – not relevant to this View
 - Remove “Int_Access_Code” and “Int_Active_Code”
 23. V_General_Field_Metereological
 - Change “Name” to “Loc_Alternative_Name” – the AltName field has the Environment Canada Name – so its easier for folks to quickly know the Climate Station location
 - Maybe remove the “Int_Type” and “Int_Type_Code” – since all of these should be climate station intervals?
 - What about adding RD_NAME_CODE to this View – someone might want to query out snowpack or something and the code might be handy (rather than text);
 - Change “Parameter_Type” to “Measurement_Descriptor”
 - Isn’t the “Group_Code” all 22 for the metereological data? If it doesn’t change then maybe we can delete this field.
 - Remove “Int_Access_Code” and “Int_Active_Code”
 24. V_General_Field_Stream_Flow
 - I didn’t check all – but if the Int_Alt_Name always has the Env Canada Long Name – then its fine – however if it doesn’t then maybe we have to add Loc_Alternative_Name to show this.
 - Maybe remove the “Int_Type” and “Int_Type_Code” – since all of these should be surface water stations?
 - Isn’t the “Group_Code” all 25 for the stream flow data? If it doesn’t change then maybe we can delete this field.
 - Change “Parameter_Type” to “Measurement_Descriptor”
 - Remove “Int_Access_Code” and “Int_Active_Code”
 25. V_General_Field_Summary

- So does this summarize the Count and start/end dates for each Parameter for each Interval from Int 2? Looks like Climate and Streamflow Ints are omitted. Maybe change name to reflect this – “V_General_Field_Summary_BHs”
 - Remove “Bedrock_Elev” – not relevant to this View
 - Wonder if a “Value_Avg” would be a useful addition to this table?
 - Remove “Int_Access_Code” and “Int_Active_Code”
26. V_General_Geology
- Change “Name” to “Loc_Name”
 - Maybe add “Layer_Thickness”?
 - Don’t know if we need UTM’s and QA code in here – don’t think that plotting would occur very often from this View
 - Remove Type_Code; Access_Code; and Status_Code – not needed
 - For those wells that have it – it might be nice to add in the “GSC_Code”
27. V_General_Geology_Deepest_Nonrock
- Don’t know if we need UTM’s and QA code in here – don’t think that plotting would occur very often from this View
 - Remove Type_Code; Access_Code; and Status_Code – not needed
28. V_General_Geology_Outcrop
- If we separate Outcrops – then should we rename V_General_Geology – to “V_General_Geology_BHs”?
 - Same points as above
29. V_General_Hydrogeology
- This view only pulls BHs right? Therefore the Type and Type_Code are likely not needed
 - Access_Code not needed
 - Same as for “V_Consultant_Hydrogeology” - Rectify difference between Pump_Readings and Pumping_Test – is one showing the daily pumping rates? If yes – let’s try find a better name – how about “Daily Pumped Volume_Num” (too long?)
 - We should add “Screened Formation” to this view
 - Is “Name” – the “Int Name” or the “Loc_Name”? – clarify
 - We need to check the field “PUMPTEST_REC_RATE_IGPM” – the values do not appear to be whole numbers – even when the pumping rate is a whole number – did we do a conversion here that we didn’t mean to?
30. V_General_MOE_Report
- What is the field “WellNumbers”? – mostly contains ND(ND)ND?
 - Does “Water” indicate “Water_Found”? Maybe change name?
31. V_General_MOE_Well
- We need to add a field for MOE_Well_ID (Loc Original Name)

- I don't think that Primary and Secondary Use are being populated correctly – they are all null for the first 1000 records
- “Access Code” and “Type Code” and “Type” are probably not needed

32. V_General_Pick

- View doesn't return any records

33. V_General_Station_Climate

- How about changing name to “V_General_Field_Summary_Meteorological” – similar to #25
- We should add Int_ID and Maybe IntName – so if someone wants to quickly find the info the Int_ID is right here.
- Type; Type_Code; and Access_Code are likely not needed
- There are quite a few stations that have no Precip or Temp readings – is that correct? No data for them – wonder if we should delete them?

34. V_General_Station_SurfaceWater

- How about changing name to “V_General_Field_Summary_SurfaceWater” – similar to #25
- What about adding in the Max and Min Water Level so that the Staff Gauges get summarized
- Any chance that for the staff gauges we could have the min/max/avg/start date/end date and number of WL readings - I guess this might mean we need two sets of readings here (one for WLev and one for streamflow)
- I think the status code should also be in text (i.e. “Active” vs “Not Active”)

35. V_General_Water_Level_Best

- I don't like the name of this view – cannot tell what is returned?
- If we keep this View – how about changing WL_Source to WL_Type (i.e. static vs logger vs other?)

36. “General”

- What do you think of removing “General” from all of the Views – I don't think it provides any info that is useful. If you/Rick want to go for only “pure” views that have codes only or no change in field names – then why not preface those with a “YPDT” - it looks to me that most of your stuff/Rick's stuff is tucked away in your linked DBs anyway. Let's discuss again.

37. “Names”

- In general you tend to use Name – in many of the views – and this is tied to LocName – I think the Loc_Name_Alt1 is a much more informative name in general – for spot flows and MOE wells – the AltName1 is better. Just thinking about what I have said below here – that we should combine “Consultant Views” with the General Vviews” to provide one – we are not supposed to give out “Owner” – so wondering if Loc_Name_Alt1” is different enough from “owner”

that I can plead it wasn't intentional or if we should have one separate view for consultants – where we drop the Alt_1 name. We should discuss when you read this.

38. Sitefx_SearchLoc_YPDT_Custom_view

- Provides the info for one template format that we need to keep – there might be others – so this view should be transferred to the new DB

39. Overburden Wells

- To get overburden wells – we would filter General_BHs for “Bedrock_Elev is null” – correct? [Which way are we approaching this – users that know exactly what they're doing OR general users?]

40. Consultant PTTW

- We formerly had a View Consultant PTTW – we should likely make a new one with our revised PTTW tables. [Revised tables have not, thus far, been approved.]

41. V_Consultant_TemporalData

- We also had a V_Consultant_TemporalData – the intent of this view was to pull info from Interval_Temporal 2 – where a flag in D_Location (SysTemp1 or SysTemp2) was inserted. The consultants were to be given the first views and based on the info – largely in Consultant_Hydrogeology – they would be able to see if they needed any of the temporal data.

42. Climate station views

- The Views for the Climate Stations are missing – We should restore V_General_Climate_Annual_Prec_Summary – this provides the number of readings in any given year as well as the Avg/Max/Min for the year – it looks like to me the Avg is only calculated when the number of readings in any given year reaches close to 365 (maybe above 360) – but I don't know if the annual avg precip is all that valueable – we can discuss – but maybe we dump the Avg and just keep the max and min? I also see some stations where the number of readings in a year exceeds 365 – can we check to see why that is? (See point below)

43. V_General_Climate_Annual_Temp_Summary

- Restore - this view is similar to the above – except with Temp. Again some years have more than 365 readings (e.g. Toronto City – 2004 has 397 readings – I suppose that if they took more than one measurement per day this could happen – don't know how it would affect the Avg calculation though – if it is an issue – then maybe we can dump the Avg. (See point below)

44. General_Climate_Data_Range

- The current view takes too long to run (>11 minutes) – but I thought you were going to rejig things to have some of the key info temporarily stored in Summary Tables? Off-hand I don't see any tables that house summarized climate data – but I do see the View YPDT_Sys_Summary_Temp_Air that basically appears to do

the same as the old view – but only for Temp. I see YPDT_Sys_Summary_Precip does the same for precip. These views are infinitely faster – so they must be relying on a built table somewhere. Wonder if we want to have them in the same view and pull it out of the “Sys_View” pile and put it up in general?

45. The View “General Climate Stations” View

- Can be restored – its pretty basic – but at least should provide the status for all of the climate stations (active vs non-active). – ok I see it below with new name.
- How about for Climate Stations/Meteorological – what if we have the following views:
 - V_General_Station_Climate_All_Data (currently named “Field_Meteorological”)
 - V_General_Station_Climate_Summary (currently named “Station_Climate”)
 - V_General_Station_Climate_Annual_Temp_Summary
 - V_General_Station_Climate_Annual_Precip_Summary
 - Same with the Streamflow stations
 - V_General_Station_Streamflow_All_Data (currently named “Field_Stream_Flow”)
 - V_General_Station_Streamflow_Summary (currently named “Station_Surface_Water”)
 - V_General_Station_Streamflow_Annual_Flow_Summary
- We should discuss whether “Station” comes before or after “Streamflow” and “Climate”

46. V_General_Surface_Water_Station_Spotflow

- I think we should bring this back so that folks remember we have data and that they should add info to DB

47. V_General_Documents

- Similar to the Consultant View but has slightly more info – maybe we can discuss and fold into only one View – V_General_Documents_(Consultant)?

48. V_General_Field

- Does this view only contain data from the BHs (i.e. not meteorological stations nor surface water stations?) – if yes maybe we change name to V_General_Field_Borehole? Then we would have V_General_Field_ 1) BH; 2) Meteorological; and 3) Stream Flow as well as the V_General_Field_Summary – which includes all three of above.

49. V_General_Geology

- We can probably look at adding a couple of fields to V_Consultant_Geology (maybe i) Ground Elev and ii) Bottom Elev (of the well – not the layer) and iii) Formation (Interpreted)) and merging the two views V_General_Geology_(Consultant) – Don’t know how many layers would have a populated Material 4 – but we may want to include it in the view so that partners etc. would know that it is available;

50. V_General_Hydrogeology vs V_Consultant_Hydrogeology
 - Maybe combine to one view - there are a few fields in the General one that aren't in the Consultant one that I would think we should add (i) Interpreted Formation (screen top); ii) Interpreted Formation (screen bottom); iii) First Measured WL (mASL); iv) Most recent measured WL (mASL); v) Drawdown During Pumpstest (m); v) Completion date; vi) screen type)
51. V_General_Outcrops
 - Maybe bring back– just so folks are reminded that they are in the DB – just found it with new name – don't know if I like calling it Borehole?
52. V_General_Locations
 - Similar to above – maybe we combine this with the “Consultant_General” View - re-name to V_General_Locations_(Consultant)
53. V_General_Permits
 - We should probably develop a new View that works with the new PTTW table(s) AND COMBINE WITH Consultant view – re-name to V_General_PTTW_(Consultant)
54. V_General_Pick
 - Let's add the “s” back – “V_General_Picks”
55. LOC_TYPE
 - Wonder if we should add the Loc_Type to the chemistry views – so that folks can readily screen out SW vs GW
56. General_Chemistry
 - Maybe change the name of General_Chemistry to General_Chemistry_All
56. Locs/Ints with Chemistry
 - How about a view (a group by) that pulls out the Locations/Intervals that have chemistry data associated with them? Maybe called V_Chemistry_Present? – I don't think we need to necessarily pull the chemistry values – but maybe the number of “dates” that appear in Int1a/b – and the first/last date.
57. Temporal_BHs_Discharge_Production
 - I would like to bring back the Temporal_BHs_Discharge_Production” view – should focus on the "Production - Pumped Volume (Total Daily)" Rd_Name Code.
58. Viewlog views
 - For the Viewlog views (VL) – I think I would want to keep the names the same (especially for VL_Log_Header) since everyone's Viewlog project would be looking for this view to show wells in cross section. We can talk specifically about these – I think the ones you omitted are ok to leave out for now.
59. General_Temporal_Chemistry_General_Water_Soil_Rock
 - We might want to bring this one back – Waterfront Toronto is thinking of asking us to put in 15,000 BHs into the DB - all having soil chemistry results.

20150317-19

Reassigned INT_TYPE_CODES '27' (i.e. 'Reported Open Hole - derived from bh construction') to one of '21' and '22' based upon the presence of casing and a bedrock elevation. Started corrections of D_INTERVAL_MONITOR with invalid or duplicate rows for particular INT_IDS. Started corrections of D_GEOLOGY_LAYER with possibly invalid units.

20150320

Updated D_LOCATION_ELEV for new locations (~800).

20150407-08

With regard to the new database version: Reworked D_INTERVAL_FORMATION_ASSIGNMENT view(s); added the number of geologic layers to the summary tables; incorporated the SITEFX_SEARCHLOC view; added the V_GENERAL_WATER_LEVEL_OTHER view; added the V_GENERAL_HYDROGEOLOGY_BEDROCK view; added the V_GENERAL_WATER_FOUND view; updated various other views to reflect underlying changes.

20150410

Exchanged LOC_NAME and LOC_NAME_ALT1 for the climate stations. Added the spot flow view(s) to the new database version.

20150413

Modified V_CONSULTANT_HYDROGEOLOGY in new version of database. Add V_SUMMARY_CHEMISTRY_SAMPLES and V_GENERAL_FIELD_PUMPING_PRODUCTION. Examined duplicate INT_IDS and combined data to single INT_ID.

20150415+

Corrected D_GEOLOGY_LAYER based on duplicate '0' depth values (and other errors).

20150420

Worked on PICKS table (trying to decrease access/update speed).

20150507-12

Incorporated UGAIS grainsize (with corrections) data previously included as part of the GEOL_DESCRIPTION within D_GEOLOGY_LAYER. Additional soil intervals were created (in some cases) in some instances (where they previously did not exist).

20150515+

Start of the water level (DIT2) corrections as part of a shallow water table calculation across the entire area. The first phase is the examination of those values that greatly exceed '20m' depth (at which depth the shallow wells are arbitrarily defined). Various modifications are made including (for example): depths such as '203m' for boreholes that are only '4m' deep are assigned a value of '2.03m' instead; units are converted from (some combination of) 'ft' to 'm' (or vice-versa) as appropriate based upon the depth of the well or depth of the screen.

Note that these modifications have/may affect borehole details especially with regard to depths and including modification of REF_ELEV from D_INTERVAL_REF_ELEV as appropriate.

There is a real issue with regard to multiple instances of elevations throughout the various tables (e.g. D_BOREHOLE, D_LOCATION_ELEV, D_INTERVAL_REF_ELEV, D_INTERVAL_MONITOR, and etc...). Mismatched elevations occur between these tables – an effort should be made to reduce these to single value with all others changed to depths (instead). This is an issue with SiteFX, however.

In addition, correction of the water levels themselves (as found in D_INTERVAL_TEMPORAL_2) has-been/is-being performed. This includes adjustment of 'masl' units to depth units (based upon, usually, the original elevation) – a similar problem, as described above, was found.

20150529

Evaluated R_BH_DRILLER_CODE to swap BH_DRILLER_DESCRIPTION and BH_DRILLER_DESCRIPTION_LONG; not possible as there are duplicates in the latter.

201506+

Worked with TRCA on Portlands data (extraction and addition of). Evaluated the presence of boreholes in the lake for this area (UGAIS wells; they were drilled through in the lake itself and is not an locational error). Determined that Toronto uses a modified NAD27 datum – modified Portlands locations based upon this.

20150623

Reset all Y/N fields in D_DOCUMENT; 'N' is changed to a NULL value and 'Y' changed to a '1' value for consistency.

20150814

Change BH_BEDROCK_ELEV values to four decimal places. This should be standardized across all relevant (i.e. REAL types) columns (e.g. elevations and depths).

201509+

Major push to finish database structure (including views) for the updated master database release. Note that if replication is to be discontinued, the auto-increment (i.e. IDENT) columns can be reinstated (from the programming – maximum determination – currently imposed). Remember, IDENT fields are very finicky with regard to multiple users and replication.

20150904

R_GEOL_MAT*_CODE out of sync (mainly MAT3 and MAT4). Corrected these and D_GEOLOGY_LAYER.

20150908

Applied a GEOL_SUBCLASS_CODE of '7' ('Original (Invalid)') to those records in D_GEOL_LAYER where the GEOL_TOP_OUOM and GEOL_BOT_OUOM have NULL values. This is code (i.e. '7') should be used to weed out invalid geology layers in the various views (as required; e.g. V_GENERAL_MOE_REPORT).

20150923

Corrected D_GEOLOGY_LAYER for those wells in the 'Yonge St. Aquifer' study. Mainly, these have zeroed elevation layers; in some cases, geology for a single layer is spread over multiple rows.

20151005

Incorporation of Permite-to-take-Water database (from 20141003) using the newly developed structure.

20151103

Changed water level data in D_INTERVAL_TEMPORAL_2 from 628/629 to 70899 when the RD_VALUE is NULL due to: frozen; dry; other ... The RD_TYPE_CODE is adjusted to match as well.

201606-08

A tracking mechanism is implemented for both coordinates (D_LOCATION_COORD_HIST) and elevations (D_LOCATION_ELEV_HIST).

20160831

A new version of the database is created – this incorporates all changes as originally implemented in the OAK_20120615_VIEWS database. All views and tables are now found entirely in a single database.

20170115

Secondary version number change (now 20160831.20170115). This is tied to the update of the master database to match the current SiteFX schema base. Comment added to D_VERSION_CURRENT was 'Matched to subset of SiteFX db v16082201'.

20170524

Secondary version number change (now 20160831.20170524). All meteorological data (i.e. from climate stations) have now been moved into the D_INTERVAL_TEMPORAL_3 table; updates to the various views are ongoing. Comment added to D_VERSION_CURRENT was 'Climate data moved to DIT3'.

Appendix F - Accessory ORMGP Databases

As of ORMGP Database version 20160831, all supplementary database tables and views have been incorporated in OAK_20160831_MASTER. Refer to the 'YPDT-CAMC Database Manual Version 5 (Dated Version 20120615)' manual for previous accessory databases.

Appendix G - Procedures (or Methodology for Various Tasks)

The following procedures are outlined:

- G.1 Formation Assignment and Associated Calculations (Automated)
- G.2 Update of D_LOCATION_GEOM (Automated)
- G.3 Report Library - Addition
- G.4 Ground Elevation Assignment
- G.5 Update of Bedrock Wells
- G.6 ~~Addition to/Population of D_LOCATION_AGENCY~~
- G.7 Update of Bedrock Elevation (Automated)
- G.8 ~~Assignment of MOE Elevations as Original Elevations~~
- G.9 Correction of Datalogger Information (DESCRIPTION NEEDED)
- G.10 Import of MOE Water Well Database
- G.11 Correction of D_GEOLOGY_LAYER – Missing Depths and Units
- G.12 ~~Creation of the TRAINING database (a subset of the MASTER database)~~
- G.13 ~~Synchronizing non-replicating databases (e.g. CLOCA)~~
- G.14 Population of coordinates (REVIEW)
- G.15 Synchronize elevations between D_BOREHOLE and D_LOCATION_ELEV (REVIEW)
- G.16 ~~Check D_INTERVAL_FORMATION_ASSIGNMENT for Invalid (Null) Rows~~
- G.17 Correction of elevations (D_BOREHOLE and D_LOCATION_ELEV) (REVIEW)
- G.18 ~~Extracting LOC_IDs for the Training database~~
- G.19 Addition of INT_ID to D_INTERVAL_FORMATION_ASSIGNMENT
- G.20 Calculate and Incorporate Specific Capacity
- G.21 Perform QA/QC Check Against OAK_20120615_MASTER Backup
- G.22 Incorporation of the MOE Permit-To-Take-Water database
- G.23 Population of D_INTERVAL_MONITOR (Top and Bottom)
- G.24 Update D_INTERVAL_MONITOR Depths
- G.25 Correction of Water Levels and Associated Data
- G.26 Correction or Update of Borehole Coordinates (MOE WWDB)
- G.27 York Database – Incorporation of Temporal Data
- G.28 Updating Elevations in D_* tables
- G.29 Update MOE BORE_HOLE_ID (D_LOCATION_ALIAS)
- G.30 Update Locations from MOE WWDB
- G.31 Incorporate D_LOCATION_COORD_HIST and D_LOCATION_ELEV_HIST Records in D_LOCATION_SPATIAL_HIST
- G.32 Automated Scripts (Listing and Calling Order)
- G.33 Update of D_AREA_GEOM

G.1 Formation Assignment and Associated Calculations (Automated)

Tables

- D_INTERVAL_FORM_ASSIGN
- D_INTERVAL_FORM_ASSIGN_FINAL

Views (incomplete – example only)

- V_SYS_CHK_DIFA_CM2004_REMOVE
- V_SYS_CHK_DIFA_CM2004_ADD
- V_SYS_DIFA_GL_MOD_CM2004
- V_SYS_CHK_DIFA_GL_OAKRIDGES_CM2004_THICK
- V_SYS_CHK_DIFA_GL_THORNCLIFFE_CM2004_THICK
- V_SYS_CHK_DIFA_GL_SCARBOROUGH_CM2004_THICK
- V_SYS_DIFAF_ASSIGN
- V_SYS_PUMP_MOE_TRANS

Estimated Recurrence Time: Weekly

A methodology has been adopted to automatically update the formation assignment (for each INT_ID), superceding the previous manual instructions. This allows for correction within a short time period if either of the coordinates or elevation of a location has been modified. These sequence of events are implemented externally to the master database (ORMGPDB) but use the listed tables and views to accomplish much of the process (an exception being the determination of the various geologic surface elevations, refer to the following for details).

Overall control of the population and update of D_INTERVAL_FORM_ASSIGN (DIFA) D_INTERVAL_FORM_ASSIGN_FINAL (DIFAF) is controlled by

d_int_form_ass.bat

(as described in Section G.32) with a section of code defined for each of the geologic models being evaluated. Currently (as of 2021-02-11) this includes the CM2004, WB2018 and YT32011 models. Using CM2004 as an example of the geologic model being examined, the order of processing is

- Removal of intervals from DIFA (V_SYS_CHK_DIFA_CM2004_REMOVE)
- Addition of intervals to DIFA (V_SYS_CHK_DIFA_CM2004_ADD)
- Creation of a temporary table (TMP_CM2004_INT) with the CM2004 geologic formation(s) identified for each affected INT_ID (V_SYS_DIFA_GL_MOD_CM2004); note that this relies upon a call to the Giant System command 'assign_layer' to interface the ORMGPDB with the grid files associated with this geologic model
- Update of DIFA based upon the contents of TMP_CM2004 where any of the current contents (of DIFA) deviate from the contents of the temporary table
- Removal of the temporary table

- Assign the unit associated with the interval for this particular model (in DIFA; refer to Section 2.4.5 for details)
- Determine the thickness of the specified aquifer (for CM2004, this includes: Oak Ridges Moraine Complex, Channel Sands, Thorncliffe, Scarborough) for each affected INT_ID and store the result in a temporary table (in order: TMP_CM2004_ORAC, TMP_CM2004_CHANSA, TMP_CM2004_THORN and TMP_CM2004_SCAR); the V_SYS_CHK_DIFA_GL_* views are used here; note that this relies upon the Giant System command 'grid_examine'
- Update the THICKNESS_M field in DIFA for each aquifer using the temporary tables (TMP_CM2004_ORAC, TMP_CM2004_CHANSA, TMP_CM2004_THORN and TMP_CM2004_SCAR) as a source
- Remove these temporary tables

Each geologic model is processed in similar manner.

We now have sufficient information to calculate Specific Capacity, Transmissivity and Hydraulic Conductivity and store the results in DIFA. Refer to Section 2.4.5 for details on how this is accomplished (and the assumptions made for these calculations). This uses V_SYS_PUMP_MOE_TRANS as a source called repeatedly by the Giant System command 'data_batch'.

At this point, the DIFAF is processed, including

- Removal of those intervals not found in DIFA (and have a NULL OVERRIDE_UNIT and MANUAL_UNIT)
- Addition of those intervals found in DIFA that are not currently in DIFAF
- Update the ASSIGNED_UNIT field based upon the models present in DIFA (V_SYS_DIFAF_ASSIGN); refer to Section 2.4.5 for details regarding the choice of the final assigned geologic unit for a particular INT_ID

G.2 Update of D_LOCATION_GEOM (Automated)

Tables

- D_LOCATION
- D_LOCATION_GEOM
- D_LOCATION_QA
- D_LOCATION_SPATIAL
- D_LOCATION_SPATIAL_HIST

Views

- V_SYS_CHK_LOC_GEOM_ADD
- V_SYS_CHK_LOC_GEOM_CHANGE
- V_SYS_CHK_LOC_GEOM_REMOVE

- V_SYS_CHK_LOC_GEOM_WKB_UPDATE
- V_SYS_LOC_GEOMETRY
- V_SYS_LOC_GEOMETRY_WKB

Estimated Recurrence Time: Weekly

This process has been automated, to a certain extent, such that a weekly check of the D_LOCATION_GEOM in comparison to D_LOCATION is made. This is controlled by

d_loc_geom.bat

(as described in Section G.32) and includes

- Removal of those LOC_IDs not found in D_LOCATION (V_SYS_CHK_LOC_GEOM_REMOVE)
- Determination if the coordinates in D_LOCATION have changed from those stored in D_LOCATION_GEOM (as a geometry; V_SYS_CHK_LOC_GEOM_CHANGE); if there has been a change, the COORD_CHECK field is updated (incremented by a value of ‘1’ if non-NULL)
- Addition of those LOC_IDs present in D_LOCATION but absent from D_LOCATION_GEOM (V_SYS_CHK_LOC_GEOM_ADD); these are calculated using V_SYS_LOC_GEOMETRY
- Update of NULL GEOM fields (V_SYS_LOC_GEOMETRY)
- Update of GEOM_WKB (V_SYS_CHK_LOC_GEOM_WKB_UPDATE); these are calculated using V_SYS_LOC_GEOMETRY_WKB

If a non-NULL COORD_CHECK

The view specified (i.e. V_D_LOCATION_Geometry) uses the coordinates found in D_LOCATION to create spatial geometries (i.e. points) for each location with valid coordinates (i.e. not having a QA_COORD_CONFIDENCE_CODE of ‘117’). Using a ‘SELECT ... INTO ...’ statement, these spatial objects are stored in the D_LOCATION_GEOM table (when re-creating the table; otherwise an ‘INSERT ...’ is used). Note that this (latter) table will be replaced every time this procedure is applied.

A ‘shortcut’ is provided by ‘CHK_D_LOC_GEOM_Missing_LOC_ID’ (found in OAK_CHECKS). This view access each of the D_LOCATION, D_LOCATION_QA and V_D_LOCATION_Geometry tables/views and returns those LOC_IDs (and their geometries) which are not found in D_LOCATION_GEOM. An ‘INSERT ...’ can then be used. For example

```
insert into [OAK_SPATIAL].dbo.D_LOCATION_GEOM
(LOC_ID,GEOM)
SELECT [LOC_ID]
,[GEOM]
FROM [OAK_CHECKS].[dbo].[CHK_D_LOC_GEOM_Missing_LOC_ID]
```

A check for non-populated LOC_COORD_EASTING and LOC_COORD_NORTHING values (based upon the last time SiteFX was correctly run) can be accomplished through the following query.

```
SELECT dloc.[LOC_ID]
,[LOC_NAME]
,[LOC_NAME_ALT1]
,[LOC_TYPE_CODE]
,[OWN_ID]
,[LOC_ORIGINAL_NAME]
,[LOC_MASTER_LOC_ID]
,[LOC_AREA]
,[LOC_STUDY]
,[LOC_COORD_EASTING]
,[LOC_COORD_NORTHING]
,[LOC_COORD_EASTING_OUOM]
,[LOC_COORD_NORTHING_OUOM]
,[LOC_COORD_OUOM_CODE]
FROM [OAK_20120615_MASTER].[dbo].[D_LOCATION] as dloc
inner join [OAK_20120615_MASTER].dbo.D_LOCATION_QA as dlqa
on dloc.LOC_ID=dlqa.LOC_ID
where
dlqa.QA_COORD_CONFIDENCE_CODE<>117
and LOC_COORD_EASTING_OUOM is not null
and LOC_COORD_NORTHING_OUOM is not null
and LOC_COORD_EASTING is null
and LOC_COORD_NORTHING is null
```

D_LOCATION can then be updated by

```
update [OAK_20120615_MASTER].[dbo].[D_LOCATION]
set
LOC_COORD_EASTING=LOC_COORD_EASTING_OUOM
,LOC_COORD_NORTHING=LOC_COORD_NORTHING_OUOM
FROM [OAK_20120615_MASTER].[dbo].[D_LOCATION] as dloc
inner join [OAK_20120615_MASTER].dbo.D_LOCATION_QA as dlqa
on dloc.LOC_ID=dlqa.LOC_ID
where
dlqa.QA_COORD_CONFIDENCE_CODE<>117
and LOC_COORD_EASTING_OUOM is not null
and LOC_COORD_NORTHING_OUOM is not null
and LOC_COORD_EASTING is null
and LOC_COORD_NORTHING is null
```

G.3 Report Library - Incorporation of Reports

Tables

- D_DOCUMENT
- D_LOCATION
- D_LOCATION_QA

Views

- V_SYS_DOC_REPLIB_ENTRY

Estimated Recurrence Time: As necessary

A form (through Microsoft Access) is available so as to standardize the entry of new reports into a format consistent with the database. This information is stored, with appropriate fields, as a single table in an Access database; modification of this data is necessary before import is possible.

A temporary key (e.g. 'rkey') must be assigned to the table; this will be eventually replaced by a LOC_ID and DOC_ID during import; these fields may need to be added as well. DOC_FOLDER_ID should be checked for duplication. As fields within this single table can be found in each of D_DOCUMENT, D_LOCATION and D_LOCATION_QA it is duplicated and the unnecessary fields (from each table) are deleted.

For D_LOCATION:

- Add each of the LOC_COORD_EASTING and LOC_COORD_NORTHING and copy the values (performing a conversion as necessary) from LOC_COORD_EATING_OUOM and LOC_COORD_NORTHING_OUOM.
- Change DOC_FOLDER_ID to LOC_NAME.
- Add a SITE_ID column and populate with a value of '1'.
- All text fields should be ANSI rather than UNICODE.
- Add a LOC_TYPE_CODE column and populate it with a value of '25' (i.e. 'Document').
- All cells that will, eventually, be NULL are assigned the value '-9999' to distinguish them from valid data.
- Each column is examined for invalid or empty cells. Each column should be sorted in ascending and descending order to make note, as necessary, that the column contains 'some' valid data.

For D_LOCATION_QA:

- Add the QA_COORD_CONFIDENCE_CODE field. By default, unless a spatial uncertainty value is stated within the report, a value of '5' (i.e. 'Margin of Error: 100m-300m') is assigned to QA_COORD_CONFIDENCE_CODE. Those locations without coordinates are assigned a value of '117' (i.e. 'Invalid ...') instead.

For D_DOCUMENT:

- It is assumed that for each '*_CODE' field, all necessary 'codes' are available (or have been entered) into the appropriate 'R_DOC_*' table.
- All cells that will, eventually, be NULL are assigned the value '-9999' to distinguish them from valid data.
- Each column is examined for invalid or empty cells. Each column should be sorted in ascending and descending order to make note, as necessary, that the column contains 'some' valid data.

Each table is then imported into a temporary database in Microsoft SQL (e.g. 'temphold'). All cells assigned a '-9999' placeholder value are converted to NULL values.

Populate the LOC_ID field in D_LOCATION using the (example) script:

```
update [tempfold],[dbo],[D_LOCATION_20150311]
set
LOC_ID=t3.LOC_ID
from
[tempfold],[dbo],[D_LOCATION_20150311] as t1
inner join
(
select
t2.LOC_ID
,ROW_NUMBER() over (order by t2.LOC_ID) as rkey
from
(
select
top 1000
v.NEW_ID as LOC_ID
from
[OAK_SUP].dbo.V_Random_ID_Creator_MD as v
where
v.NEW_ID not in
(select LOC_ID from [OAK_20120615_MASTER].dbo.d_location)
) as t2
) as t3
on t1.rkey=t3.rkey
```

Populate the LOC_ID field in D_LOCATION_QA using the (example) script:

```
update [tempfold],[dbo],[D_LOCATION_QA_20150311]
set
LOC_ID=dloc.LOC_ID
FROM [tempfold],[dbo],[D_LOCATION_QA_20150311] as dlqa
inner join [tempfold].dbo.[d_location_20150311] as dloc
on dlqa.rkey=dloc.rkey
```

Populate the LOC_ID field in D_DOCUMENT using the (example) script:

```
update [tempfold],[dbo],[D_DOCUMENT_20150311]
set
LOC_ID=dloc.LOC_ID
FROM
[tempfold],[dbo],[D_DOCUMENT_20150311] as ddoc
inner join [tempfold].dbo.[D_LOCATION_20150311] as dloc
on ddoc.rkey=dloc.rkey
```

Populate the DOC_ID field in D_DOCUMENT using the (example) script:

```
update [tempfold],[dbo],[D_DOCUMENT_20150311]
set
DOC_ID=t3.DOC_ID
from
[tempfold],[dbo],[D_DOCUMENT_20150311] as t1
inner join
(
select
t2.DOC_ID
,ROW_NUMBER() over (order by t2.DOC_ID) as rkey
from
(
select
top 1000
v.NEW_ID as DOC_ID
from
[OAK_SUP].dbo.V_Random_ID_Creator_MD as v
```

```

where
v.NEW_ID not in
(select DOC_ID from [OAK_20120615_MASTER].dbo.d_document)
) as t2
) as t3
on t1.rkey=t3.rkey

```

Note that in each of the 'update' example scripts, the 'top 1000' should be modified to a value greater than the number of locations/documents being entered (in this case, '764' documents were being added). This controls the number of random identifiers to return from 'V_Random_ID_Creater_MD'.

Insert the completed (example) D_LOCATION_20150311 table into D_LOCATION (note that 'rkey' is not being inserted):

```

insert into [OAK_20120615_MASTER].dbo.D_LOCATION
(
[LOC_NAME]
,[LOC_NAME_ALT1]
,[LOC_COORD_EASTING_OUOM]
,[LOC_COORD_NORTHING_OUOM]
,[LOC_COORD_OUOM_CODE]
,[OWN_ID]
,[LOC_ID]
,[LOC_COORD_EASTING]
,[LOC_COORD_NORTHING]
,[SITE_ID]
,[LOC_STUDY]
)
SELECT
[LOC_NAME]
,[LOC_NAME_ALT1]
,[LOC_COORD_EASTING_OUOM]
,[LOC_COORD_NORTHING_OUOM]
,[LOC_COORD_OUOM_CODE]
,[OWN_ID]
,[LOC_ID]
,[LOC_COORD_EASTING]
,[LOC_COORD_NORTHING]
,[SITE_ID]
,[LOC_STUDY]
FROM
[temphold].[dbo].[D_LOCATION_20150311]

```

Insert the completed (example) D_LOCATION_QA_20150311 table into D_LOCATION_QA (note that the 'rkey' is not being inserted):

```

insert into [OAK_20120615_MASTER].dbo.D_LOCATION_QA
(
[LOC_ID]
,[QA_COORD_CONFIDENCE_CODE]
)
SELECT
[LOC_ID]
,[QA_COORD_CONFIDENCE_CODE]
FROM
[temphold].[dbo].[D_LOCATION_QA_20150311]

```

Insert the completed (example) D_DOCUMENT_20150311 table into D_DOCUMENT (note that the 'rkey' is not being inserted):

```

insert into [OAK_20120615_MASTER].dbo.D_DOCUMENT
(
[DOC_FOLDER_ID]
,[DOC_AUTHOR]
,[DOC_YEAR]
,[DOC_MONTH]
,[DOC_DAY]
,[DOC_PAGE_RANGE]
,[DOC_AUTHOR_AGENCY_CODE]
,[DOC_TYPE_CODE]
,[DOC_FORMAT_CODE]
,[DOC_LOCATION_CODE]
,[DOC_DESCRIPTION]
,[DOC_TOPIC_CODE]
,[DOC_CLIENT_AGENCY_CODE]
,[DOC_YN_LOCATION_MAP]
,[DOC_YN_CROSS_SECTION]
,[DOC_YN_BH_LOG]
,[DOC_YN_GEOPHYSICS]
,[DOC_YN_PUMP_TEST]
,[DOC_YN_MODELLING]
,[DOC_YN_CHEMISTRY]
,[DOC_YN_WATERLEVEL]
,[DOC_YN_DIGITAL_DATA]
,[DOC_VOLUME_OTHER]
,[DOC_YN_DRAFT]
,[DOC_YN_PARTIAL]
,[LOC_ID]
,[DOC_ID]
)
SELECT
[DOC_FOLDER_ID]
,[DOC_AUTHOR]
,[DOC_YEAR]
,[DOC_MONTH]
,[DOC_DAY]
,[DOC_PAGE_RANGE]
,[DOC_AUTHOR_AGENCY_CODE]
,[DOC_TYPE_CODE]
,[DOC_FORMAT_CODE]
,[DOC_LOCATION_CODE]
,[DOC_DESCRIPTION]
,[DOC_TOPIC_CODE]
,[DOC_CLIENT_AGENCY_CODE]
,[DOC_YN_LOCATION_MAP]
,[DOC_YN_CROSS_SECTION]
,[DOC_YN_BH_LOG]
,[DOC_YN_GEOPHYSICS]
,[DOC_YN_PUMP_TEST]
,[DOC_YN_MODELLING]
,[DOC_YN_CHEMISTRY]
,[DOC_YN_WATERLEVEL]
,[DOC_YN_DIGITAL_DATA]
,[DOC_VOLUME_OTHER]
,[DOC_YN_DRAFT]
,[DOC_YN_PARTIAL]
,[LOC_ID]
,[DOC_ID]
FROM
[temphold].[dbo].[D_DOCUMENT_20150311]

```

These locations will automatically be added to D_LOCATION_GEOM on a weekly basis (refer to Appendix G.2 and G.32 for details).

The document records currently found in the ORMGP database (i.e. in D_DOCUMENT and associated) can be added to the Microsoft Access database (for reference by any

Partner Agency) through V_SYS_DOC_REPLIB_ENTRY. This view assembles the requisite fields that match the format found in the report entry database.

G.4 Ground Elevation Assignment

Tables

- D_BOREHOLE
- D_LOCATION
- D_LOCATION_GEOM
- D_LOCATION_QA
- D_LOCATION_SPATIAL (DLS)
- D_LOCATION_SPATIAL_HIST (DLSH)
- R_LOC_ELEV_CODE (RLEC)
- R_QA_ELEV_CONFIDENCE_CODE (RQECC)

Views

- V_SYS_LOC_COORDS

Estimated Recurrence Time: Weekly

The ground elevation for locations within the database are stored in D_BOREHOLE (BH_GND_ELEV) and D_LOCATION_SPATIAL_HIST (LOC_ELEV). These values should (optimally) match. However, for historical reasons, the BH_GND_ELEV can be updated without capturing the change in D_LOCATION_SPATIAL_HIST. The latter allows a tight connection to be made between the coordinates of the location and its ground elevation and should be updated for any change of information related to coordinates or elevation data (e.g. QA_COORD_CONFIDENCE_CODE or QA_ELEV_CONFIDENCE_CODE from D_LOCATION_QA). A new record is created in this table for each change (or changes) – D_LOCATION_SPATIAL records reflect the ‘current’ coordinates and elevation for any particular location. These values can be accessed (easily) using V_SYS_LOC_COORDS. Further details can be found in Section 2.4.2.

All locations with valid coordinates (as found in D_LOCATION) should have a record in D_LOCATION_GEOM. (Note that a QA_COORD_CONFIDENCE_CODE of ‘117’ indicates invalid coordinates in D_LOCATION_QA.) If a location has any coordinates or elevation associated with it, this will be recorded in DLSH (and DLS).

An elevation can be entered directly as part of an import process for any particular location. This would be considered an ‘Original’ elevation (i.e. LOC_ELEV_CODE ‘2’ in R_LOC_ELEV_CODE) and is generally not kept as the ground surface elevation unless a QA_ELEV_CONFIDENCE_CODE (or QA_ELEV_CODE in DLSH) of ‘1’ has been applied (i.e. a ‘Surveyed’ location). Note that another exception, this time using a code of ‘11’ (i.e. an ‘External change has been applied – do not modify’), can also be

used to 'lock' an elevation value. This, though, is usually applied against locations that have already been imported (and not to new locations).

Each location should have a Digital Elevation Model (DEM) value associated with it (whether it is being used or not, as in the case of a surveyed location). Within the ORMGP study area this will be drawn from the Ministry of Natural Resources (2005) DEM (DEM_MNR_v2; a 10-metre resolution dataset; a LOC_ELEV_CODE of '3' in R_LOC_ELEV_CODE). Outside the ORMGP study area this will be drawn from the Jarvis et al (2008) Shuttle Radar Topography Mission DEM (DEM_SRTM_v41; a 90-metre resolution dataset; a LOC_ELEV_CODE of '5' in R_LOC_ELEV_CODE). In some cases, both DEM values will be stored (though in more recent data imports, this practice has been abandoned).

When populating the elevation fields, the location coordinates and LOC_ID are exported to an external GIS (the coordinates are found in either of D_LOCATION or DLS/DLSH – these should match). Alternatively, the LOC_ID and geometry from D_LOCATION_GEOM can be accessed by an external GIS (the geometry must match the coordinates found in the other tables). These locations are compared against the DEM_MNR_v2 and/or DEM_SRTM_v41 grids and the associated elevation recorded (by LOC_ID) as new attribute in a temporary table accessible to the master database.

For example, the command

```
gnat grid_examine -grass -grid=DEM_MNR_v2_10m_BATHY_25km_buffer
-hdrs=x,y,loc_id -method=locnrfw
-i=odbc,cloca_temphold,<user>,<pass>,elev_upd_20201013
-o=odbc,cloca_temphold,<user>,<pass>,elev_mnr_20201013
```

could be used in a pre-configured Grass GIS environment to determine and store the appropriate DEM values linked to a specific LOC_ID.

This table can then be used to update the associated LOC_ELEV field (as found in DLSH) using, for example

```
update d_location_spatial_hist
set
loc_elev_code= 3
,loc_elev_date= '2020-10-13'
,loc_elev= m.value
,loc_elev_unit_code= 6
,loc_elev_ouom= m.value
,loc_elev_unit_ouom= 'masl'
,qa_elev_code= 10
from d_location_spatial_hist as dls
inner join d_location_spatial as dls
on dls.spat_id=dls.spat_id
inner join temphold.dbo.elev_mnr_20201013 as m
on dls.loc_id=m.loc_id
where
m.value <> -9999
```

Note that this presupposes that a new record containing the current coordinates has been added to DLSH and referenced in DLS (refer to Appendix G.2 for details). Here we are incorporating DEM_MNR_v2 values (as indicated by a LOC_ELEV_CODE of '3').

In some cases, a NULL value could be determined as part of this process (this is indicated with a value of '-9999'). This is checked as part of the update query and usually occurs for those locations that lie outside the ORMGP study area. When this occurs, a DEM_SRTM_v41 value would be determined and incorporated.

This would directly be available as the 'current' coordinates and elevation as found through V_SYS_LOC_COORDS. The D_BOREHOLE values, however, would still need to be updated. This is accomplished by

```
update d_borehole
set
bh_gnd_elev= delev.assigned_elev
,bh_gnd_elev_ouom= delev.assigned_elev
,bh_gnd_elev_unit_ouom= 'masl'
,bh_dem_gnd_elev= delev.assigned_elev
,sys_temp1= delev.sys_temp1
,sys_temp2= delev.sys_temp2
from d_borehole as dbore
inner join v_sys_loc_coords as delev
on dbore.loc_id=delev.loc_id
cross join (
select valf as _elevcorr from s_constant where const_name='SYS_ELEV_CORR_RANGE'
) as sc
where
delev.sys_temp1= '20201013z'
and not( dbore.bh_gnd_elev between (delev.assigned_elev - sc._elevcorr) and (delev.assigned_elev + sc._elevcorr) )
```

Here, each of BH_GND_ELEV, BH_GND_ELEV_OUOM and BH_DEM_GND_ELEV are updated with the same value. The use of the SYS_TEMP1 and SYS_TEMP2 fields in the query aid in updating affected elevations in other tables (refer to Appendix G.28).

G.5 Update of Bedrock Wells (Intervals)

Tables

- D_BOREHOLE
- D_BOREHOLE_CONSTRUCTION
- D_GEOLOGY_LAYER

Views

- V_SYS_BH_BEDROCK_ELEV

Estimated Recurrence Time: Periodic (as needed)

These may be classified as 'Open Holes – Bedrock' within the ORMGP database.

For those boreholes that reach bedrock (especially with regard to MOE WWDB imports), additional information from D_BOREHOLE_CONSTRUCTION can be used for determining bedrock elevation and (if possible) correcting formation interval problems. An example is used for describing the methodology.

	FORMATION_ID	BORE_HOLE_ID	WELL_ID	LAYER	COLOR	MAT1	MAT2	MAT3	FORMATION_TOP_DEPTH
1	2251657	1001701129	7108812	1	2	28	NULL	77	0
2	2251698	1001701129	7108812	2	2	05	12	85	11
3	2251699	1001701129	7108812	3	NULL	34	NULL	NULL	23
4	2251700	1001701129	7108812	4	2	05	12	NULL	NULL
5	2251701	1001701129	7108812	5	NULL	34	NULL	NULL	46
6	2251702	1001701129	7108812	6	2	05	28	74	NULL
7	2251703	1001701129	7108812	7	NULL	34	NULL	NULL	56
8	2251704	1001701129	7108812	8	2	05	NULL	73	NULL
9	2251705	1001701129	7108812	9	NULL	34	NULL	NULL	91
10	2251706	1001701129	7108812	10	2	05	11	74	NULL
11	2251707	1001701129	7108812	11	NULL	34	NULL	NULL	137
12	2251708	1001701129	7108812	12	2	15	NULL	73	NULL

FORMATION_END_DEPTH	FORMATION_END_DEPTH_UOM
11	ft
23	ft
NULL	ft
46	ft
NULL	ft
56	ft
NULL	ft
91	ft
NULL	ft
137	ft
NULL	ft
151	ft

Here, as shown, is the source data from a sample MOE well (release 2010; the ‘TbIFormation’ table) showing the recorded formations for the particular location. Note the missing, overlapping or (possibly) invalid layers. From D_BOREHOLE_CONSTRUCTION

	BH_ID	CON_SUBTYPE_CODE	CON_TOP_ELEV	CON_BOT_ELEV	CON_TOP_OUOM	CON_BOT_OUOM	CON_UNIT_OUOM
1	725001967	24	NULL	NULL	138	151	ft
2	725001967	21	NULL	NULL	-2	138	ft
3	725001967	31	NULL	NULL	0	20	ft

CON_COMMENT	CON_DIAMETER	CON_DIAMETER_OUOM	CON_DIAMETER_UNIT_OUOM
NULL	NULL	6.125	inch
NULL	NULL	6.25	inch
NULL	NULL	NULL	NULL

it can be seen (or interpreted) that the casing (CON_SUBTYPE_CODE ‘21’) ends at ‘138’ ‘ft’ (CON_BOT_OUOM and CON_UNIT_OUOM, respectively) with an open hole (CON_SUBTYPE_CODE ‘24’) immediately below. As the formation information records the presence of bedrock (‘15’ in MAT1), it can be interpreted that the bedrock depth should occur around the ‘138’ foot-mark. Notice that a formation interval ends at a depth ‘137’ feet (FORMATION_END_DEPTH) – we can use this, then, as the top-of-bedrock mark and correct the bedrock layer, extending to the bottom of the borehole at ‘151’ feet (again in FORMATION_END_DEPTH). This leads, as a starting point, to the correction of the entire formation column, as shown (from D_GEOLOGY_LAYER).

	LOC_ID	GEOLOGICAL_TOP_ELEV	GEOLOGICAL_BOT_ELEV	GEOLOGICAL_TOP_OUOM	GEOLOGICAL_BOT_OUOM	GEOLOGICAL_UN
1	372767949	260.3589	257.0061	0	11	fbgs
2	372767949	257.0061	253.3485	11	23	fbgs
3	372767949	253.3485	246.3381	23	46	fbgs
4	372767949	246.3381	243.2901	46	56	fbgs
5	372767949	243.2901	232.6221	56	91	fbgs
6	372767949	232.6221	218.6013	91	137	fbgs
7	372767949	218.6013	214.3341	137	151	fbgs

GEOLOGICAL_UNIT_OUOM	GEOLOGICAL_MAT1_CODE	GEOLOGICAL_MAT2_CODE	GEOLOGICAL_MAT3_CODE	GEOLOGICAL_COMMENT
fbgs	28	NULL	77	ft changed to fbgs
fbgs	5	12	85	ft changed to fbgs
fbgs	34	5	12	ft changed to fbgs; till indic
fbgs	5	28	74	ft changed to fbgs; till indic
fbgs	34	5	73	ft changed to fbgs; till indic
fbgs	5	11	74	ft changed to fbgs; till indic
fbgs	15	NULL	73	ft changed to fbgs; interval

GEOLOGICAL_COMMENT	SYS_RECORD_ID	GEOLOGICAL_SUBCLASS_CODE
ft changed to fbgs	1815037829	5
ft changed to fbgs	-2033119134	5
ft changed to fbgs; till indicated; ...	-1552435872	5
ft changed to fbgs; till indicated; ...	-125232498	5
ft changed to fbgs; till indicated; ...	-185730196	5
ft changed to fbgs; till indicated; ...	1589346170	5
ft changed to fbgs; interval deter...	-1489429960	5

Note that comments have been included to explain (and track) the modifications from the original source data. Here it appears that an additional material 'Till' (GEOLOGICAL_MAT?_CODE '34') has been included for describing some of the layers; this has been incorporated either as a material code or in the comments (dependent upon the existing number of material codes used). In this way, the number of original layers are reduced in D_GEOLOGY_LAYER recording instead the 'interpreted' layers. The GEOLOGICAL_SUBCLASS_CODE of '5' indicates that the layer is original or has been corrected. In the case where the formation cannot be corrected (for example, if the neither of the tops or bottoms are provided) a code of '7' (i.e. 'Invalid') is instead applied.

At this point, the bedrock elevation can be extracted from D_GEOLOGY_LAYER (a value of '218.6013') and applied to D_BOREHOLE (in the BH_BEDROCK_ELEV field).

BH_ID	LOC_ID	BH_GND_ELEV	BH_GND_ELEV_OUOM	BH_GND_ELEV_UNIT_OUOM	BH_DEM_GND_ELEV	BH_BO	
1	725001967	372767949	260.3589	260.3589	masl	260.3589	214.334

BH_BOTTOM_ELEV	BH_BOTTOM_DEPTH	BH_BOTTOM_OUOM	BH_BOTTOM_UNIT_OUOM	BH_BEDROCK_ELEV
214.3341	46.0248	151	fbgs	218.6013

In the case that there is a single bedrock unit that has neither top nor bottom specified and all other layers have a top and bottom (or most), the default thickness of 0.3m (1ft for non-metric locations) is applied.

Note that V_SYS_BH_BEDROCK_ELEV is used for determining and checking bedrock elevations in reference to the D_GEOLOGY_LAYER. This process is automated and described in Appendix G.7 and G.32.

G.6 Addition to/Population of D_LOCATION_AGENCY

The D_LOCATION_AGENCY table has been replaced by a series of views by which to determine those locations that fall within a specific Partner Agency. Refer to the V_SYS_AGENCY_* views for details.

G.7 Update of Bedrock Elevation (Automated)

Tables

- D_BOREHOLE
- D_GEOLOGY_LAYER

Views

- V_SYS_BH_BEDROCK_ELEV

Estimated Recurrence Time: Weekly

V_SYS_BH_BEDROCK_ELEV checks for locations/boreholes that intersect bedrock and returns the BH_BEDROCK_ELEV. This determination is made by examining D_GEOLOGY_LAYER for those locations with a GEOL_MAT1_CODE that corresponds to a 'ROCK' designation (i.e. having a value of '1') in R_GEOL_MAT1_CODE.

The bedrock elevation to be assigned to BH_BEDROCK_ELEV in D_BOREHOLE is the value contained in 'new_BH_BEDROCK_ELEV. This can be compared against BH_BEDROCK_ELEV.

The value of BH_BEDROCK_ELEV in D_BOREHOLE can be updated by

```
update d_borehole
set
bh_bedrock_elev=v.new_bh_bedrock_elev
from
d_borehole as dbore
left outer join v_sys_bh_bedrock_elev as v
on dbore.loc_id=v.loc_id
where
(dbore.bh_bedrock_elev is null and v.new_bh_bedrock_elev is not null)
or (dbore.bh_bedrock_elev is not null and v.new_bh_bedrock_elev is null)
```

or dbore.bh_bedrock_elev<>v.new_bh_bedrock_elev

Note that this process has been automated – refer to Appendix G.32 for details.

G.8 Assignment of MOE Elevations as Original Elevations

Refer to Appendix G.10 for details regarding the incorporation of original elevations as part of the MOE WWDB import process.

G.9 Correction of Datalogger Information

Tables

Estimated Recurrence Time: Before data import

?????Description?????

G.10 Import of MOE Water Well Database

Tables (OAK_20160831_MASTER compatible)

- D_BOREHOLE
- D_BOREHOLE_CONSTRUCTION
- D_DATA_SOURCE
- D_GEOLOGY_FEATURE
- D_GEOLOGY_LAYER
- D_INTERVAL
- D_INTERVAL_MONITOR
- D_INTERVAL_REF_ELEV
- D_INTERVAL_TEMPORAL_2
- D_LOCATION
- D_LOCATION_ALIAS
- D_LOCATION_ELEV
- D_LOCATION_PURPOSE
- D_LOCATION_QA
- D_PUMPTEST
- D_PUMPTEST_STEP

Tables (MOE Water Well Database, MOE_20170905)

- TblBore_Hole
- TblCasing
- TblFormation
- TblHole
- TblMethod_Construction
- TblPipe

- TblPlug
- TblPump_Test
- TblPump_Test_Detail
- TblScreen
- TblWater
- TblWWR

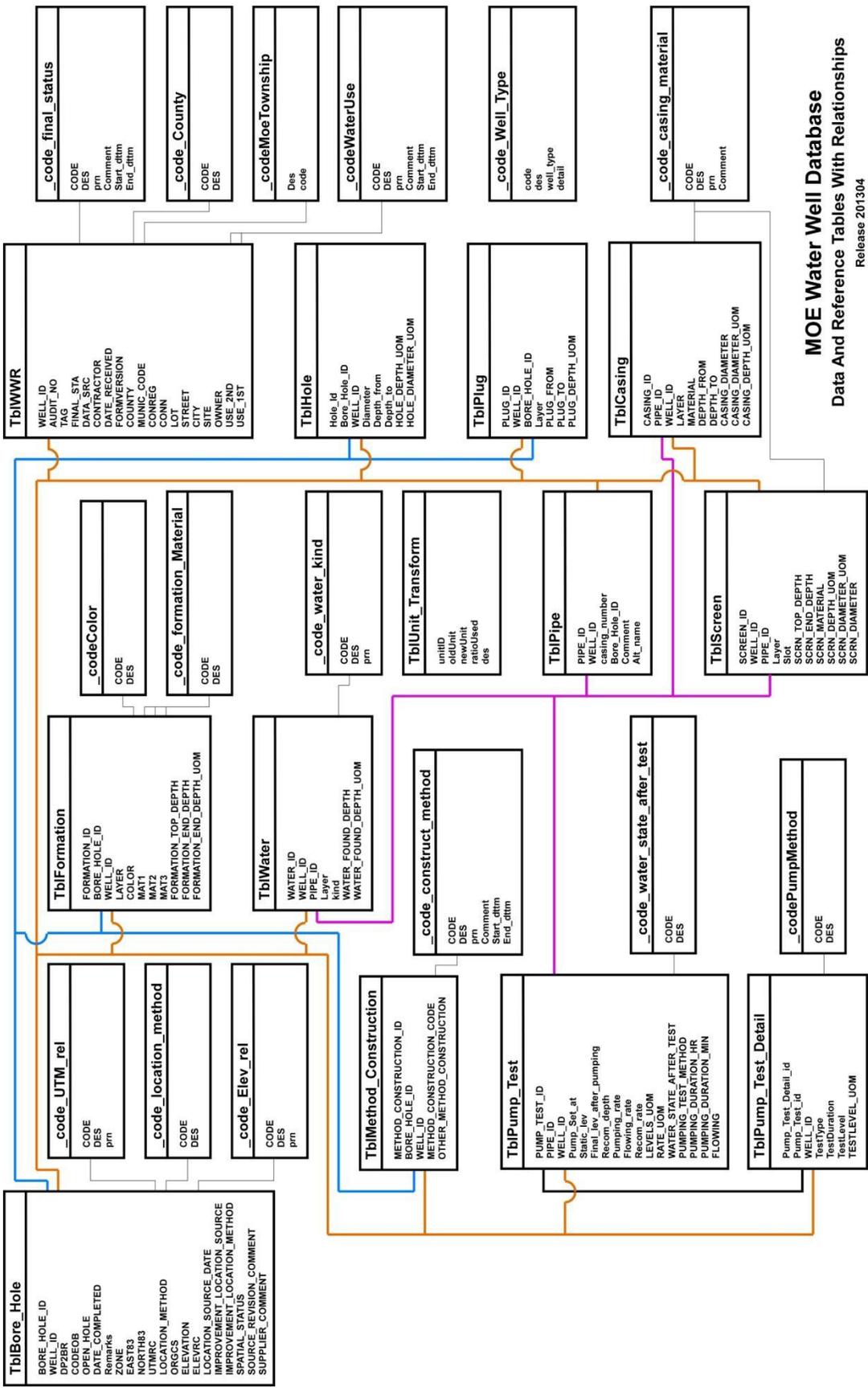
Estimated Recurrence Time: As necessary (bi-monthly?)

This section describes the methodology, procedures and scripts necessary to convert the MOE Water Well Database (WWDB) to the format necessary for import into the YPDT-CAMC Database (YCDB). The details provided are particular to the structure and state of the WWDB as of 2017-09, the date at which the updated database was made available to the YPDT-CAMC (additional notes may be included for characteristics/changes in newer versions of the WWDB). This section presupposes that the WWDB has already been incorporated as an SQL Server database. (Note that this can be accomplished through the use of the ‘SQL Server Import and Export Data’ tool; the 32-bit version is recommended in this case.)

Note that only the data tables are listed above. In addition, various look-up tables from both databases are used as part of the conversion process. The process itself is destructive in that modifications are made as necessary to the source tables. As such, a backup of the WWDB should be created before proceeding.

SQL code is available both in this document and also available as an ‘.sql’ file (found within the ‘Appendix – G – Procedures – scripts’ directory; the file names are of the form ‘<G section>_<script number>_<script name>.sql’). Note that a pair of scripts is present for each step; one for viewing the information, the other for applying/creating the information. The separate file may contain additional code (usually described herein but not listed) that can be used as an aid in the data import process.

The relationships between the MOE WWDB tables (as of 20170905) is shown.



MOE Water Well Database
Data And Reference Tables With Relationships
 Release 201304

TblBore_Hole

BORE_HOLE_ID
WELL_ID
DPZBR
CODEOB
OPEN_HOLE
DATE_COMPLETED
Remarks
EXSTR3
NORTH83
UTMRC
LOCATION_METHOD
ORGS
ELEVATION
LOCATION_SOURCE_DATE
IMPROVEMENT_LOCATION_SOURCE
SPATIAL_STATUS
SOURCE_REVISION_COMMENT
SUPPLIER_COMMENT

_code_UTM_rel

CODE
DES
prn

_code_location_method

CODE
DES

_code_Elev_rel

CODE
DES
prn

TblFormation

FORMATION_ID
BORE_HOLE_ID
WELL_ID
LAYER
LOR
MAT1
MAT2
MAT3
FORMATION_TOP_DEPTH
FORMATION_END_DEPTH
FORMATION_END_DEPTH_UOM

TblWater

WATER_ID
WELL_ID
PIPE_ID
Layer
WATER_FOUND_DEPTH
WATER_FOUND_DEPTH_UOM

_code_Color

CODE
DES

_code_formation_Material

CODE
DES

TblIWWR

WELL_ID
AUDT_NO
TAG
FINAL_STA
CONTRACTOR
DATE_RECEIVED
OWNER
COUNTY
MUNIC_CODE
CONFRG
CONN
LOT
STREET
SITE
OWNER
USE_2ND
USE_1ST

_code_Final_Status

CODE
DES
prn
Comment
Start_dttm
End_dttm

_code_County

CODE
DES

_codeMoeTownship

Des
code

_code_water_kind

CODE
DES
prn

TblUnit_Transform

unitID
oldunit
newunit
ratioUsed
des

TblHole

Hole_ID
Bore_Hole_ID
WELL_ID
Diameter
Depth_from
Depth_to
DEPTH_UOM
HOLE_DIAMETER_UOM

_code_Water_Use

CODE
DES
prn
Comment
Start_dttm
End_dttm

_code_construct_method

CODE
DES
prn
Comment
Start_dttm
End_dttm

TblPipe

PIPE_ID
WELL_ID
Number
Bore_Hole_ID
Comment
Alt_name

TblPlug

PLUG_ID
WELL_ID
WELL_HOLE_ID
PLUG_FROM
PLUG_TO
PLUG_DEPTH_UOM

_code_Well_Type

code
des
well_type
detail

_code_water_state_after_test

CODE
DES

_code_Pump_Method

CODE
DES

TblScreen

SCREEN_ID
WELL_ID
PIPE_ID
Layer
Slot
SCRN_TOP_DEPTH
SCRN_BOTTOM_DEPTH
SCRN_MATERIAL
SCRN_DEPTH_UOM
SCRN_DIAMETER_UOM
SCRN_DIAMETER

TblCasing

CASING_ID
PIPE_ID
WELL_ID
LAYER
MATERIAL
DEPTH_FROM
DEPTH_TO
CASING_DIAMETER
CASING_DEPTH_UOM

_code_casing_material

CODE
DES
prn
Comment

TblPump_Test

PUMP_TEST_ID
PIPE_ID
WELL_ID
Pump_Set_at
Static_lev
Final_lev_after_pumping
Recomm_rate
Flowing_rate
Flowing_rate
Recomm_rate
LEVELS_UOM
RATE_UOM
WATER_STATE_AFTER_TEST
PUMPING_TEST_METHOD
PUMPING_DURATION_MAX
PUMPING_DURATION_MIN
FLOWING

TblPump_Test_Detail

Pump_Test_Detail_id
Pump_Test_id
WELL_ID
TestType
TestDuration
TestLevel
TESTLEVEL_UOM

G.10.1 Determination of WELL_IDs to be imported

The WELL_ID is the primary key by which missing boreholes (from the YCDB) are determined. Note that this is not the primary key used to distinguish between boreholes in the WWDB – the BORE_HOLE_ID (variously identified) is used instead, where multiple BORE_HOLE_IDs are associated with a single WELL_ID. The WELL_ID is stored as LOC_ORIGINAL_NAME in D_LOCATION for many locations; both the WELL_ID and the BORE_HOLE_ID are stored as aliases in D_LOCATION_ALIAS, linked by the LOC_ID. Inclusion of both identifiers allows back-checking of information (to the WWDB).

Approximately 768,401 WELL_IDs are present in 'TblBore_Hole'. In addition to the data already present in the YCDB, we're also using a NULL value in any of ZONE, EAST83 or NORTH83 to remove a particular WELL_ID from consideration. The row-counter is present in the case we need to create an on-the-fly relationship to other tables.

Create the look-up table.

```
select
ZONE
,count(*) as [rowcount]
,case
    when ZONE=15 then 7
    when ZONE=16 then 8
    when ZONE=17 then 4
    when ZONE=18 then 5
    else null        -- the remainder of the zones should be invalid
end
as [LOC_COORD_OUOM_CODE]
FROM
MOE_20170905.[dbo].[TblBore_Hole] as tbh
group by
ZONE
```

Script: G_10_01_01_ZONE.sql

Extract the WELL_IDs.

```
select
distinct(missing.WELL_ID)
--,ROW_NUMBER() over (order by missing.WELL_ID) as [rnum]
,cast(null as int) as well_id_rnum
from
(
select
    tbh.WELL_ID
    ,v.LOC_ID
    ,tbh.ZONE
    ,tbh.EAST83
    ,tbh.NORTH83
from
    MOE_20170905.dbo.TblBore_Hole as tbh
left outer join OAK_20160831_MASTER.dbo.V_SYS_MOE_LOCATIONS as v
on cast(tbh.WELL_ID as int)=v.MOE_WELL_ID
) as missing
inner join MOE_20170905.dbo.YC_20170905_LOC_COORD_OUOM_CODE as yccode
on missing.ZONE=yccode.ZONE
where
```

missing.LOC_ID is null
and yccode.ZONE is not null
and missing.EAST83 is not null
and missing.NORTH83 is not null

Script: G_10_01_02_WELL_ID.sql

Note that the SQL 'collate' (found in previous versions) has been removed. Instead, V_SYS_MOE_LOCATIONS is being substituted for D_LOCATION (to determine what WELL_IDs have been loaded into the database).

Get the coordinates for each of these WELL_IDs, using BORE_HOLE_ID as the primary key. We'll use this to delimit the locations to only the YPDT-CAMC study area.

```
select
bh.BORE_HOLE_ID
,bh.WELL_ID
,bh.ZONE
,bh.EAST83
,bh.NORTH83
from
MOE_20170905.dbo.TblBore_Hole as bh
inner join MOE_20170905.dbo.YC_20170905_WELL_ID_AVAIL as avail
on bh.WELL_ID=avail.WELL_ID
```

Script: G_10_01_03_COORDS.sql

These locations should be exported to a GIS and, by UTM Zone, checked as to whether they are within the study area (check for invalid zones and match against the WELL_IDs for correction; locations with obvious coordinate errors are dropped). Note that we are actually using the YPDT-CAMC study area with a 25km buffer zone. Zones 15 and 16 lie outside the YPDT-CAMC area and are not considered; NULL values are no-longer evaluated as if lying within Zone 17 and are excluded (see G_10_01_02_WELL_ID; these should be evaluated separately using county or address information). Two new fields should be added into this table – EAST83_ORIG and NORTH83_ORG – and these should be populated with the original coordinate values only if a conversion is necessary (i.e. if the location lies within Zone 18) while EAST83 and NORTH83 should contain only Zone 17 coordinates.

Those locations that lie within the YPDT-CAMC area should be loaded into the new table YC_20170905_BORE_HOLE_ID_COORDS_YC within the MOE_20180530 database.

Get the list of WELL_IDs that are going to be added to the YCDB.

```
select
WELL_ID
,cast(null as int) as well_id_num
into MOE_20170905.dbo.YC_20170905_YPDT_WELL_ID
from
MOE_20170905.dbo.YC_20170905_BORE_HOLE_ID_COORDS_YC as y
group by
WELL_ID
```

Script: G_10_01_04_YPDT_WELL_ID.sql

In the previous database (20170905) there were no instances of multiple boreholes assigned to a single well identifier. In the current version (20180530) this is not the case – likely due to the incorporation of data within the YPDT-CAMC study area using a 25km buffer.

G.10.2 Determination of BORE_HOLE_ID characteristics

We'll use the BORE_HOLE_IDs from YC_20170905_BORE_HOLE_ID_COORDS_YC (the table created and imported from an external GIS) against which to characterize the locations. First, we'll determine which BORE_HOLE_ID has geology information (from 'TblFormation'). A row-count has been added in order to create on-the-fly relationships as necessary. In addition, a series of NULL fields are introduced that will be populated at a later stage in the process.

```
select
y.BORE_HOLE_ID
,y.WELL_ID
,ROW_NUMBER() over (order by y.BORE_HOLE_ID) as [bore_hold_id_rnum]
,.cast(null as int) as [LOC_ID]
,.cast(null as int) as [BH_ID]
,.convert(int,1) as [LOC_MASTER_LOC_ID]
,.convert(float,null) as [FM_MAX_DEPTH]
,.convert(varchar(50),null) as [FM_MAX_DEPTH_UNITS]
,.convert(float,null) as [MOE_MAX_DEPTH]
,.convert(varchar(50),null) as [MOE_MAX_DEPTH_UNITS]
,.convert(float,null) as [CON_MAX_DEPTH]
,.convert(varchar(50),null) as [CON_MAX_DEPTH_UNITS]
,.convert(float,null) as [MAX_DEPTH_M]
,.cast(null as int) as NOFORMATION
into MOE_20170905.dbo.YC_20170905_BH_ID
from
MOE_20170905.dbo.YC_20170905_BORE_HOLE_ID_COORDS_YC as y
```

Script: G_10_02_01_BORE_HOLE_ID.sql

As we're using the BORE_HOLE_ID as the base key for data conversion, we don't need to check for repeating WELL_IDs (this differs from using the WELL_ID in the previous import scheme as the base key).

Note the use of a '1' value for the LOC_MASTER_LOC_ID field – this is to indicate that these boreholes are the 'master' boreholes in the case of multi-borehole locations (i.e. those locations for which the MOE has allowed a single site geology description to be submitted where multiple boreholes were drilled). We'll update this field based on the absence/presence of formation information. First check that there are not multiple boreholes, by WELL_ID, that contain formation information.

```
select
t2.WELL_ID
,t2.rcount
from
(
select
y.WELL_ID
```

```

,count(*) as rcount
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
inner join
(
select
f.BORE_HOLE_ID
,count(*) as rcount
from
MOE_20170905.dbo.tblFormation as f
inner join MOE_20170905.dbo.YC_20170905_BH_ID as y
on f.BORE_HOLE_ID=y.BORE_HOLE_ID
group by
f.bore_hole_id
) as t
on y.BORE_HOLE_ID=t.BORE_HOLE_ID
group by
y.WELL_ID
) as t2
where
t2.rcount>1

```

This should return no rows if there is only a single borehole for each WELL_ID that contains formation information (no rows are returned for MOE_20170905). All of these boreholes (i.e. with formation information) can now be considered ‘master’ locations - we’ll temporarily assign the BORE_HOLE_ID to the LOC_MASTER_LOC_ID to specify this.

```

update MOE_20170905.dbo.YC_20170905_BH_ID
set
LOC_MASTER_LOC_ID=y.BORE_HOLE_ID
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
inner join
(
select
f.BORE_HOLE_ID
,count(*) as rcount
from
MOE_20170905.dbo.tblFormation as f
inner join MOE_20170905.dbo.YC_20170905_BH_ID as y
on f.BORE_HOLE_ID=y.BORE_HOLE_ID
group by
f.bore_hole_id
) as t
on
y.BORE_HOLE_ID=t.BORE_HOLE_ID

```

There are considered to be ‘master’ locations. Assign the appropriate BORE_HOLE_ID to the remainder (i.e. the BORE_HOLE_ID and LOC_MASTER_LOC_ID will not match in this case); various methodologies for doing so follows.

```

select
t2.WELL_ID
,t2.rcount
from
(
select
t.WELL_ID
,count(*) as rcount
from
(
select
y.WELL_ID

```

```

,y.BORE_HOLE_ID
,y.LOC_MASTER_LOC_ID
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
where
LOC_MASTER_LOC_ID<>1
) as t
group by
t.WELL_ID
) as t2
where
t2.rcount>1

```

If there are no rows returned (as in the case of 20170905), there are no cluster/grouped wells with formation information. These need to be further evaluated in a subsequent step. We'll change the '1' values to a 'null' for subsequent examination.

```

update MOE_20170905.dbo.YC_20170905_BH_ID
set
LOC_MASTER_LOC_ID=null
where
LOC_MASTER_LOC_ID=1

```

Script: G_10_02_02_MASTER_LOCATIONS.sql

There are 5,661 rows without a declared master location – we will now try and assign them. First evaluate those that consist of a single line, only.

Now that we have assigned a set of master locations (with formation information), check if the null LOC_MASTER_LOC_IDS are relatable.

```

select
t2.WELL_ID
,t3.WELL_ID
from
(
select
t.WELL_ID
,t.rcount
from
(
select
y.WELL_ID
,count(*) as rcount
from
MOE_20180530.dbo.YC_20180530_BH_ID as y
where
LOC_MASTER_LOC_ID is null
group by
y.WELL_ID
) as t
where
t.rcount=1
) as t2
inner join
(
select
y.WELL_ID
from
MOE_20180530.dbo.YC_20180530_BH_ID as y
where
y.BORE_HOLE_ID=y.LOC_MASTER_LOC_ID
group by

```

```

y.WELL_ID
) as t3
on
t2.WELL_ID=t3.WELL_ID

```

If this returns any rows, check and assign the appropriate BORE_HOLE_ID as the LOC_MASTER_LOC_ID for each row (remember that these won't have any formation information directly).

```

update MOE_20180530.dbo.YC_20180530_BH_ID
set
LOC_MASTER_LOC_ID= t.LOC_MASTER_LOC_ID
,NOFORMATION=1
from
MOE_20180530.dbo.YC_20180530_BH_ID as y
inner join
(
select
y.WELL_ID
,y.BORE_HOLE_ID
,y.LOC_MASTER_LOC_ID
from
MOE_20180530.dbo.YC_20180530_BH_ID as y
where
y.loc_master_loc_id=y.bore_hole_ID
) as t
on y.well_id=t.well_id
where
y.loc_master_loc_id is null

```

Now evaluate those that consist of a single line, only (these have no formation information as well).

```

update MOE_20170905.dbo.YC_20170905_BH_ID
set
LOC_MASTER_LOC_ID=y.BORE_HOLE_ID
,NOFORMATION=1
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
inner join
(
select
t.WELL_ID
,t.rcount
from
(
select
y.WELL_ID
,count(*) as rcount
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
where
LOC_MASTER_LOC_ID is null
group by
y.WELL_ID
) as t
where
t.rcount=1
) as t
on y.WELL_ID=t.WELL_ID

```

Script: G_10_02_03_MASTER_LOCATIONS_SINGLE.sql

For the remainder, those WELL_IDs with no formation info but having multiple BORE_HOLE_IDs, assign the smallest BORE_HOLE_ID as the master identifier (this can be changed at a later date, as necessary; note that this was unnecessary for 20170905 but was required for 20180530).

```
update MOE_20170905.dbo.YC_20170905_BH_ID
set
LOC_MASTER_LOC_ID=t.BORE_HOLE_ID
,NOFORMATION=1
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
inner join
(
select
y.WELL_ID
,min(BORE_HOLE_ID) as BORE_HOLE_ID
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
where
y.LOC_MASTER_LOC_ID is null
group by
y.WELL_ID
) as t
on y.WELL_ID=t.WELL_ID
```

Script: G_10_02_04_MASTER_LOCATIONS_MULTIPLE.sql

All LOC_MASTER_LOC_IDs should now have been assigned.

Previously, the LOC_MASTER_LOC_IDs were assigned (for non-formation locations) in the following manner:

- If only one of the boreholes has a depth associated with it (from FM_D_BOREHOLE), that borehole is designated the master.
- If only one of the boreholes has water levels associated with it (either from pumping or measure statics), that borehole is designated the master.
- If only one of the boreholes has any construction details, that borehole is designated the master.
- If none of the boreholes have any further details, the deepest or first LOC_ID is designated the master.

G.10.3 Introduction of a LOC_ID (disabled)

The contents of this section has been moved to later in the process.

G.10.4 Creation of formation D_LOCATION (FM_D_LOCATION)

For creating the import D_LOCATION table, a series of look-up tables should be made equating WWDB values to YCDB values.

Convert the MOE final status code to the YPDT-CAMC status code.

```
select
wvr.FINAL_STA
```

```

.case
  when wwr.FINAL_STA is null then 4 -- Unknown to Unknown; this should not be used as we're converting nulls to 0
  when wwr.FINAL_STA='0' then 4 -- Unknown to Unknown
  when wwr.FINAL_STA='1' then 1 -- Water Supply to Active
  when wwr.FINAL_STA='2' then 1 -- Observation to Active
  when wwr.FINAL_STA='3' then 1 -- Test Hole to Active
  when wwr.FINAL_STA='4' then 1 -- Recharge to Active
  when wwr.FINAL_STA='5' then 3 -- Abandoned Supply to Abandoned
  when wwr.FINAL_STA='6' then 3 -- Abandoned Quality to Abandoned
  when wwr.FINAL_STA='7' then 3 -- Unfinished to Abandoned
  when wwr.FINAL_STA='8' then 4 -- Not A Well to Unknown
  when wwr.FINAL_STA='9' then 1 -- Dewatering to Active
  when wwr.FINAL_STA='A' then 3 -- Abandoned-Other to Abandoned
  when wwr.FINAL_STA='B' then 4 -- Replacement Well to Unknown
  when wwr.FINAL_STA='C' then 4 -- Alteration to Unknown
  when wwr.FINAL_STA='D' then 4 -- Other Status to Unknown
  when wwr.FINAL_STA='E' then 17 -- Monitoring and Test Hole to MOE Monitoring and Test Hole
  when wwr.FINAL_STA='F' then 18 -- Abandoned Monitoring and Test Hole to MOE Abandoned Monitoring and Test Hole
  else -9999
end
as [LOC_STATUS_CODE]
.case
  when wwr.FINAL_STA is null then 4 -- Unknown to Unknown; this should not be used as we're converting nulls to 0
  when wwr.FINAL_STA='0' then null -- Unknown to Unknown
  when wwr.FINAL_STA='1' then 1 -- Water Supply to Active
  when wwr.FINAL_STA='2' then 2 -- Observation to Active
  when wwr.FINAL_STA='3' then 3 -- Test Hole to Active
  when wwr.FINAL_STA='4' then 4 -- Recharge to Active
  when wwr.FINAL_STA='5' then 5 -- Abandoned Supply to Abandoned
  when wwr.FINAL_STA='6' then 6 -- Abandoned Quality to Abandoned
  when wwr.FINAL_STA='7' then 7 -- Unfinished to Abandoned
  when wwr.FINAL_STA='8' then 8 -- Not A Well to Unknown
  when wwr.FINAL_STA='9' then 9 -- Dewatering to Active
  when wwr.FINAL_STA='A' then 10 -- Abandoned-Other to Abandoned
  when wwr.FINAL_STA='B' then 11 -- Replacement Well to Unknown
  when wwr.FINAL_STA='C' then 12 -- Alteration to Unknown
  when wwr.FINAL_STA='D' then 13 -- Other Status to Unknown
  when wwr.FINAL_STA='E' then 14 -- Abandoned Monitoring and Test Hole to MOE Abandoned Monitoring and Test Hole
  when wwr.FINAL_STA='F' then 15 -- Abandoned Monitoring and Test Hole to MOE Abandoned Monitoring and Test Hole
  else -9999
end
as [BH_STATUS_CODE]
into
MOE_20170905.dbo.YC_20170905_FINAL_STATUS
from
MOE_20170905.dbo.TblWWR as wwr
group by
FINAL_STA
order by
FINAL_STA

```

For null values in 'TblWWR', assign them a '0' value to match the look-up table.

```

update MOE_20170905.[dbo].[TblWWR]
set
FINAL_STA=0
where
FINAL_STA is null

```

Script: G_10_04_01_FINAL_STATUS.sql

The look-up table for the MOE first and second use should not contain any NULLs (as they cannot be used as a key); change these values to '0'.

```

update MOE_20170905.dbo.TblWWR

```

```

set
USE_1ST=0
where
USE_1ST is null

update MOE_20170905.dbo.TblWWR
set
USE_2ND=0
where
USE_2ND is null

```

Create the MOE use look-up table (both the USE_1ST and USE_2ND are examined; the SQL code would be otherwise equivalent after the change in the field name).

```

select
wvr.USE_1ST
,case
  when wvr.USE_1ST is null then 0 -- Unknown
  when wvr.USE_1ST='0' then 0 -- Unknown
  when wvr.USE_1ST='A' then 10 -- OTHER
  when wvr.USE_1ST='B' then 11 -- TEST HOLE
  when wvr.USE_1ST='C' then 12 -- DEWATERING
  when wvr.USE_1ST='D' then 13 -- MONITORING
  when wvr.USE_1ST='E' then 14 -- Monitoring and Test Hole
  else wvr.USE_1ST
end
as [MOE_USE]
into
MOE_20170905.dbo.YC_20170905_MOE_USE
from
MOE_20170905.dbo.TblWWR as wvr
group by
USE_1ST
order by
USE_1ST

```

Script: G_10_04_02_MOE_USE.sql

We'll now extract D_LOCATION equivalent information from the WWDB tables. Note that starting with the 20160531 database, the BORE_HOLE_ID is being used to differentiate between wells (instead of the WELL_ID as previously applied); in this case, the WELL_ID may be applied as the LOC_ORIGINAL_NAME in multiple cases. The LOC_MASTER_LOC_ID should/can be used to determine the 'master' LOC_ID - i.e. the WELL_ID/BORE_HOLE_ID combination linked to the geology at a particular location. In most cases, the LOC_ID and LOC_MASTER_LOC_ID will be same value; for those locations where a MOE 'multiple-borehole' submittal was done (i.e. multiple locations tied to single description of the geology) the LOC_ID will differ from the LOC_MASTER_LOC_ID. In addition, a new DATA_ID must be specified (and included in D_DATA_SOURCE) for each MOE import date. For example, the DATA_ID of '519' refers to 'MOE WWR Database – 201700905'.

```

select
y.BORE_HOLE_ID as LOC_ID
,cast(bh.WELL_ID as varchar(255)) as [LOC_NAME]
,cast('MOE Well 20210119 - Name Witheld by MOE' as varchar(255)) as [LOC_NAME_ALT1]
,cast(1 as int) as [LOC_TYPE_CODE]
,cast(bh.WELL_ID as varchar(255)) as [LOC_ORIGINAL_NAME]
,y.[LOC_MASTER_LOC_ID]
,ycoords.[EAST83] as [LOC_COORD_EASTING]
,ycoords.[NORTH83] as [LOC_COORD_NORTHING]
,ycoords.east83_orig as LOC_COORD_EASTING_OUOM

```

```

,yccoords.north83_orig as LOC_COORD_NORTHING_OUOM
,case
when yccoords.zone_orig= 17 then 4
else 5
end as LOC_COORD_OUOM_CODE
,cast(wwr.COUNTY as int) as [LOC_COUNTY_CODE]
,cast(wwr.MUNIC_CODE as int) as [LOC_TOWNSHIP_CODE]
,ycfs.LOC_STATUS_CODE
,bh.DATE_COMPLETED as [LOC_START_DATE]
,cast(1 as int) as [SITE_ID]
,cast(1 as int) as [LOC_CONFIDENTIALITY_CODE]
,cast(wwr.DATA_SRC as int) as [LOC_DATA_SOURCE_CODE]
,cast(wwr.CONN as varchar(50)) as [LOC_CON]
,cast(wwr.LOT as varchar(50)) as [LOC_LOT]
,cast(RTRIM(LTRIM(wwr.STREET)) as varchar(255)) as [LOC_ADDRESS_INFO1]
,cast(RTRIM(LTRIM(wwr.CITY)) as varchar(255)) as [LOC_ADDRESS_CTY]
,ycmoeu1.MOE_USE as [LOC_MOE_USE_1ST_CODE]
,ycmoeu2.MOE_USE as [LOC_MOE_USE_2ND_CODE]
,523 as DATA_ID
into MOE_20210119.dbo.M_D_LOCATION
from
MOE_20210119.dbo.YC_20210119_BH_ID as y
inner join MOE_20210119.dbo.TblBore_Hole as bh
on y.BORE_HOLE_ID=bh.BORE_HOLE_ID
left outer join MOE_20210119.dbo.TblWWR as wwr
on bh.WELL_ID=wwr.WELL_ID
left outer join MOE_20210119.dbo.YC_20210119_LOC_COORD_OUOM_CODE as yccode
on bh.ZONE=yccode.ZONE
left outer join MOE_20210119.dbo.YC_20210119_FINAL_STATUS as ycfs
on wwr.FINAL_STA=ycfs.FINAL_STA
inner join MOE_20210119.dbo.YC_20210119_MOE_USE as ycmoeu1
on wwr.USE_1ST=ycmoeu1.USE_1ST
inner join MOE_20210119.dbo.YC_20210119_MOE_USE as ycmoeu2
on wwr.USE_2ND=ycmoeu2.USE_1ST
inner join MOE_20210119.dbo.YC_20210119_BORE_HOLE_ID_COORDS_YC as yccoords
on y.BORE_HOLE_ID=yccoords.BORE_HOLE_ID

```

Check the LOC_ADDRESS field for zero-length values (and null them).

```

update MOE_20170905.dbo.M_D_LOCATION
set
LOC_ADDRESS_INFO1=null
from
MOE_20170905.dbo.M_D_LOCATION as m
where
len(LOC_ADDRESS_INFO1)=0

```

Script: G_10_04_03_M_D_LOCATION.sql

G.10.5 Determining elevations

We'll now use the source of YC_20180530_BORE_HOLE_ID_COORDS_YC (i.e. the external GIS) to determine the elevation of each location from the relevant DEMs. We are not including any automated geocoding here (as we did in the previous conversion instructions) as this introduced a number of erroneous results.

The resultant table should be called YC_20160531_BORE_HOLE_ID_ELEVS and contain the following fields.

- BORE_HOLE_ID
- DEM_MNR

- DEM_SRTM

For those locations falling (just) outside of the study area, their MNR elevations will be tagged with a '-9999' value. Refer to the following script (not shown) for details.

Script: G_10_05_ELEVATIONS.sql

G.10.6 Location aliases

Multiple aliases for each location are present within the WWDB. These include the BORE_HOLE_ID (where the WELL_ID is found in LOC_ORIGINAL_NAME), the WELL_ID, the 'MOE Tag Number' and the 'MOE Audit Number'.

Assemble the alias information for BORE_HOLE_ID.

```
select
y.BORE_HOLE_ID as LOC_ID
,cast(y.BORE_HOLE_ID as varchar(255)) as LOC_NAME_ALIAS
,cast(3 as int) as [LOC_ALIAS_TYPE_CODE]
,cast(null as int) as SYS_RECORD_ID
into MOE_20170905.dbo.M_D_LOCATION_ALIAS
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
```

Add the information for the 'MOE Tag Number'.

```
insert into MOE_20170905.dbo.M_D_LOCATION_ALIAS
(LOC_ID,LOC_NAME_ALIAS,LOC_ALIAS_TYPE_CODE)
select
y.BORE_HOLE_ID as LOC_ID
,cast(m.TAG as varchar(255)) as LOC_NAME_ALIAS
,cast(1 as int) as [LOC_ALIAS_TYPE_CODE]
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
inner join MOE_20170905.dbo.TblWWR as m
on y.WELL_ID=m.WELL_ID
where
m.TAG is not null
```

Add the information for the 'MOE Audit Number'.

```
insert into MOE_20170905.dbo.M_D_LOCATION_ALIAS
(LOC_ID,LOC_NAME_ALIAS,LOC_ALIAS_TYPE_CODE)
select
y.BORE_HOLE_ID as LOC_ID
,cast(m.AUDIT_NO as varchar(255)) as LOC_NAME_ALIAS
,cast(2 as int) as [LOC_ALIAS_TYPE_CODE]
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
inner join MOE_20170905.dbo.TblWWR as m
on y.WELL_ID=m.WELL_ID
where
m.AUDIT_NO is not null
```

Add the information for the WELL_ID.

```
insert into MOE_20170905.dbo.M_D_LOCATION_ALIAS
(LOC_ID,LOC_NAME_ALIAS,LOC_ALIAS_TYPE_CODE)
select
```

```

y.LOC_ID
,cast(y.LOC_ORIGINAL_NAME as varchar(255)) as LOC_NAME_ALIAS
,cast(4 as int) as [LOC_ALIAS_TYPE_CODE]
from
MOE_20170905.dbo.M_D_LOCATION as y

```

Script: G_10_06_01_FM_D_LOC_ALIAS.sql

G.10.7 Location QA

The QA codes between the WWDB and YCDB match with the exceptions that a NULL value from the WWDB needs to be changed to a value of ‘9’ for the YCDB. Default the QA_ELEV_CONFIDENCE_CODE to ‘10’ (DEM) unless the ELEVRC (from the MOE) has a value of ‘1’.

```

select
y.BORE_HOLE_ID as LOC_ID
,case
    when m.UTMRC is null then 9
    else m.UTMRC
end
as [QA_COORD_CONFIDENCE_CODE]
,case
    when m.UTMRC is null then 9
    else m.UTMRC
end
as [QA_COORD_CONFIDENCE_CODE_ORIG]
,m.[LOCATION_METHOD] as [QA_COORD_METHOD]
,case
    when m.ELEVRC=1 then 1
    else 10
end as [QA_ELEV_CONFIDENCE_CODE]
,m.ELEVRC as [QA_ELEV_CONFIDENCE_CODE_ORIG]
into MOE_20170905.dbo.M_D_LOCATION_QA
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
inner join MOE_20170905.dbo.TblBore_Hole as m
on y.BORE_HOLE_ID=m.BORE_HOLE_ID

```

Script: G_10_07_01_M_D_LOC_QA.sql

G.10.8 Location elevation

The DEM elevations using the coordinates in the M_D_LOCATION table have been determined externally to this database-translation scheme and are present in YC_20160531_BORE_HOLE_ID_ELEVS (as found in Section G.10.5). The MOE elevation is being pulled from ‘TblBore_Hole – ELEVATION’ and is assumed to be expressed in ‘masl’. Note that if the QA_COORD_CONFIDENCE CODE for a location has a value of ‘1’, the final assigned elevation would be this. Otherwise, the value from ‘MNR DEM v2’ (i.e. using the field ‘dem_mnr’) or ‘SRTM DEM v4.1’ (where the location appears outside of the ORMGP study area; i.e. using the field ‘dem_srtm’)

We’ll populate M_D_LOCATION_SPATIAL_HIST with the original elevation (from the MOE) first, where present.

```

select

```

```

y.BORE_HOLE_ID as LOC_ID
,cast(4 as int) as LOC_COORD_HIST_CODE
,cast('2021-01-19' as datetime) as LOC_COORD_DATE
,ycoord.east83 as X
,ycoord.north83 as Y
,cast(26917 as int) as EPSG_CODE
,ycoord.east83_orig as X_OUOM
,ycoord.north83_orig as Y_OUOM
,cast(( case when ycoord.zone_orig=17 then 26917 else 26918 end) as int) as EPSG_CODE_OUOM
,dlqa.qa_coord_confidence_code as QA_COORD_CODE
,cast(523 as int) as LOC_COORD_DATA_ID
,cast(( case when m.location_method is not null and len(m.location_method)>0 then m.location_method else null end) as
varchar(255)) as LOC_COORD_METHOD
,cast(( case when m.elevation is not null then 2 else null end) as int) as LOC_ELEV_CODE
,cast(( case when m.elevation is not null then '2021-01-19' else null end) as datetime) as LOC_ELEV_DATE
,cast(m.ELEVATION as float) as LOC_ELEV
,cast(( case when m.elevation is not null then 6 else null end) as int) as LOC_ELEV_UNIT_CODE
,cast(m.ELEVATION as float) as LOC_ELEV_OUOM
,cast(( case when m.elevation is not null then 'masl' else null end) as varchar(50)) as LOC_ELEV_UNIT_OUOM
,cast(null as int) as QA_ELEV_CODE
,cast(( case when m.elevation is not null then 523 else null end) as int) as LOC_ELEV_DATA_ID
,cast(( case when m.elevrc is not null and len(m.elevrc)>0 then m.elevrc else null end) as varchar(255)) as
LOC_ELEV_COMMENT
into MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST
from
MOE_20210119.dbo.YC_20210119_BH_ID as y
inner join MOE_20210119.dbo.TblBore_Hole as m
on y.BORE_HOLE_ID=m.BORE_HOLE_ID
inner join MOE_20210119.dbo.YC_20210119_BORE_HOLE_ID_COORDS_YC as ycoord
on y.bore_hole_id=ycoord.bore_hole_id
inner join MOE_20210119.dbo.M_D_LOCATION_QA as dlqa
on y.bore_hole_id=dlqa.loc_id

```

Then the MNR DEM (v2) or SRTM DEM (v4.1); the latter is used if the first is tagged with a '-9999' value.

```

insert into MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST
(
LOC_ID
,LOC_COORD_HIST_CODE
,LOC_COORD_DATE
,X
,Y
,EPG_CODE
,X_OUOM
,Y_OUOM
,EPG_CODE_OUOM
,QA_COORD_CODE
,LOC_COORD_DATA_ID
,LOC_COORD_METHOD
,LOC_ELEV_CODE
,LOC_ELEV_DATE
,LOC_ELEV
,LOC_ELEV_UNIT_CODE
,LOC_ELEV_OUOM
,LOC_ELEV_UNIT_OUOM
,QA_ELEV_CODE
,LOC_ELEV_DATA_ID
,LOC_ELEV_COMMENT
)
select
y.BORE_HOLE_ID as LOC_ID
,cast(4 as int) as LOC_COORD_HIST_CODE
,cast('2021-01-19' as datetime) as LOC_COORD_DATE
,ycoord.east83 as X
,ycoord.north83 as Y
,cast(26917 as int) as EPSG_CODE
,ycoord.east83_orig as X_OUOM

```

```

.ycoord.north83_orig as Y_OUOM
.cast( ( case when ycoord.zone_orig=17 then 26917 else 26918 end ) as int ) as EPSG_CODE_OUOM
.dlqa.qa_coord_confidence_code as QA_COORD_CODE
.cast( 523 as int ) as LOC_COORD_DATA_ID
.cast( ( case when m.location_method is not null and len(m.location_method)>0 then m.location_method else null end ) as
varchar(255) ) as LOC_COORD_METHOD
.cast( ( case when ye.dem_mnr=-9999 then 5 else 3 end ) as int ) as LOC_ELEV_CODE
.cast( '2021-01-19' as datetime ) as LOC_ELEV_DATE
.cast( ( case when ye.dem_mnr=-9999 then ye.dem_srtm else ye.dem_mnr end ) as float ) as LOC_ELEV
.cast( 6 as int ) as LOC_ELEV_UNIT_CODE
.cast( ( case when ye.dem_mnr=-9999 then ye.dem_srtm else ye.dem_mnr end ) as float ) as LOC_ELEV_OUOM
.cast( 'masl' as varchar(50) ) as LOC_ELEV_UNIT_OUOM
.cast( 10 as int ) as QA_ELEV_CODE
.cast( null as int ) as LOC_ELEV_DATA_ID
.cast( null as varchar(255) ) as LOC_ELEV_COMMENT
.cast( null as varchar(255) ) as LOC_ELEV_COMMENT
from
MOE_20210119.dbo.YC_20210119_BH_ID as y
inner join MOE_20210119.dbo.YC_20210119_BORE_HOLE_ID_ELEVS as ye
on y.bore_hole_id=ye.bore_hole_id
inner join MOE_20210119.dbo.TblBore_Hole as m
on y.BORE_HOLE_ID=m.BORE_HOLE_ID
inner join MOE_20210119.dbo.YC_20210119_BORE_HOLE_ID_COORDS_YC as ycoord
on y.bore_hole_id=ycoord.bore_hole_id
inner join MOE_20210119.dbo.M_D_LOCATION_QA as dlqa
on y.bore_hole_id=dlqa.loc_id

```

Script: G_10_08_01_M_D_LOC_ELEV.sql

Note that the corresponding table – D_LOCATION_SPATIAL – will be created at a later step (after the D_LOCATION_SPATIAL_HIST has been imported; the required SPAT_ID will be automatically generated at that time).

G.10.09 Borehole information

As part of the process, a driller identifier (BH_DRILLER_CODE) must be present in R_BH_DRILLER_CODE. Potentially, the WWDB could add new driller identifiers – these will need to be accounted for before the M_D_BOREHOLE table is created.

Determine if there are any drillers not listed in R_BH_DRILLER_CODE

```

select
m.CONTRACTOR
from
MOE_20170905.dbo.TblWWR as m
where
m.CONTRACTOR collate database_default
not in
(
select bh_driller_alt_code collate database_default
from OAK_20160831_MASTER.dbo.R_BH_DRILLER_CODE
)

```

Script: G_10_09_01_DRILLER_CODE.sql

If any CONTRACTOR codes/names are returned, these need to be added into the R_BH_DRILLER_CODE (note that there were no new driller codes found for this version of the WWDB).

A comparison table is required for matching the MOE drill method codes to the YPDT-CAMC drill method codes. In some cases, the codes are not specified (or not specified correctly). As such, there some effort required for manipulating the MOE drill method codes so that they will match directly (in addition, the text field OTHER_METHOD_CONSTRUCTION in the WWDB is accessed). Refer to the following script reference.

Script: G_10_09_02_MOD_DRILL_METH.sql

We can now map the MOE (modified) drill methods to the YPDT-CAMC drill method codes on a case-by-case (by BORE_HOLE_ID) basis.

```
select
y.BORE_HOLE_ID
,m.METHOD_CONSTRUCTION_CODE
,case
  when m.METHOD_CONSTRUCTION_CODE is null then 0
  when m.METHOD_CONSTRUCTION_CODE = '0' then 0      -- Unknown
  when m.METHOD_CONSTRUCTION_CODE = '1' then 1      -- Cable-tool
  when m.METHOD_CONSTRUCTION_CODE = '2' then 2      -- Rotary Conventional
  when m.METHOD_CONSTRUCTION_CODE = '3' then 3      -- Rotary Reverse
  when m.METHOD_CONSTRUCTION_CODE = '4' then 4      -- Rotary Air
  when m.METHOD_CONSTRUCTION_CODE = '5' then 5      -- Air Percussion
  when m.METHOD_CONSTRUCTION_CODE = '6' then 6      -- Boring
  when m.METHOD_CONSTRUCTION_CODE = '7' then 7      -- Diamond
  when m.METHOD_CONSTRUCTION_CODE = '8' then 8      -- Jetting
  when m.METHOD_CONSTRUCTION_CODE = '9' then 9      -- Driving
  when m.METHOD_CONSTRUCTION_CODE = 'A' then 10     -- Hand auger digging
  when m.METHOD_CONSTRUCTION_CODE = 'B' then 0      -- Other Method
  when m.METHOD_CONSTRUCTION_CODE = 'C' then 0      -- TBD
  when m.METHOD_CONSTRUCTION_CODE = 'D' then 9      -- Direct Push
  when m.METHOD_CONSTRUCTION_CODE = 'E' then 12     -- Auger (solid stem auger)
  when m.METHOD_CONSTRUCTION_CODE = 'F' then 13     -- Hollow stem auger
  when m.METHOD_CONSTRUCTION_CODE = 'G' then 12     -- Solid stem auger
  when m.METHOD_CONSTRUCTION_CODE = 'H' then 16     -- Geoprobe
  when m.METHOD_CONSTRUCTION_CODE = 'I' then 43     -- Sonic
  when m.METHOD_CONSTRUCTION_CODE = 'Y' then 18     -- ADDED: a YC code only - Pionjar
  when m.METHOD_CONSTRUCTION_CODE = 'Z' then 46     -- ADDED: a YC code only - Rotary (dual)
  else 0
end
as BH_DRILL_METHOD_CODE
into MOE_20180530.dbo.YC_20180530_DRILL_CODE
from
MOE_20180530.dbo.YC_20180530_BH_ID as y
inner join [MOE_20160531].[dbo].[TblMethod_Construction] as m
on y.BORE_HOLE_ID= m.BORE_HOLE_ID
```

Script: G_10_09_03_DRILL_METH.sql

We'll also need to check that a BH_DRILLER_CODE exists for every specified MOE driller. Insert any that are missing into R_BH_DRILLER_CODE. (Note that if we're not working against the master database, we'll need to store these values in M_R_BH_DRILLER_CODE to be incorporated later; this code is not shown here.)

```
insert into [OAK_20160831_MASTER].dbo.R_BH_DRILLER_CODE
(BH_DRILLER_DESCRIPTION,BH_DRILLER_DESCRIPTION_LONG,BH_DRILLER_ALT_CODE)
select
t.BH_DRILLER_DESCRIPTION
,t.BH_DRILLER_DESCRIPTION_LONG
,t.BH_DRILLER_ALT_CODE
```

```

from
(
select
cast('MOE Driller No. ' + cast(moewwr.CONTRACTOR as varchar(255)) as varchar(255)) as BH_DRILLER_DESCRIPTION
,cast('MOE Driller No. ' + cast(moewwr.CONTRACTOR as varchar(255)) as varchar(255)) as
BH_DRILLER_DESCRIPTION_LONG
,cast(moewwr.CONTRACTOR as varchar(255)) as BH_DRILLER_ALT_CODE
from
MOE_20210119.dbo.YC_20210119_BH_ID as y
left outer join MOE_20210119.dbo.TblWWR as moewwr
on y.WELL_ID=moewwr.WELL_ID
left outer join
OAK_20160831_MASTER.dbo.R_BH_DRILLER_CODE as rbdc
on moewwr.CONTRACTOR collate database_default=rbdc.BH_DRILLER_ALT_CODE collate database_default
where
rbdc.BH_DRILLER_CODE is null
) as t
group by
t.BH_DRILLER_DESCRIPTION,t.BH_DRILLER_DESCRIPTION_LONG,t.BH_DRILLER_ALT_CODE

```

Script: G_10_09_04_DRILLER.sql

The M_D_BOREHOLE table can now be assembled. Note that we're introducing a placeholder (i.e. BH_BOTTOM_OUOM and BH_BOTTOM_UNIT_OUOM) for the borehole depths until we determine the maximum depth (of the borehole) from various methods.

For the 20200721 import, it was found that there a number of drill methods specified for the same borehole (by BORE_HOLE_ID). These additional procedures were translated into text and added to the BH_COMMENT field. This methodology should be standardized before the next import (there was only an ad-hoc development here; refer to the following script file for details).

```

select
y.BORE_HOLE_ID as BH_ID
,y.BORE_HOLE_ID as LOC_ID
,cast(null as float) as BH_GND_ELEV
,cast(null as float) as BH_GND_ELEV_OUOM
,cast('masl' as varchar(50)) as BH_GND_ELEV_UNIT_OUOM
,cast(null as float) as BH_DEM_GND_ELEV
,cast(null as float) as BH_BOTTOM_ELEV
,cast(null as float) as BH_BOTTOM_DEPTH
,cast(null as float) as BH_BOTTOM_OUOM
,cast('mbgs' as varchar(50)) as BH_BOTTOM_UNIT_OUOM
,ydc.BH_DRILL_METHOD_CODE
,rbdc.BH_DRILLER_CODE as [BH_DRILLER_CODE]
,moebh.DATE_COMPLETED as [BH_DRILL_END_DATE]
,ycfs.BH_STATUS_CODE
,cast(90 as int) as [BH_DIP]
,cast(0 as int) as [BH_AZIMUTH]
,cast(moeh.max_diameter as float) as BH_DIAMETER_OUOM
,cast(moeh.HOLE_DIAMETER_UOM as varchar(50)) as BH_DIAMETER_UNIT_OUOM
,cast(moecwt.[des] as varchar(255)) as MOE_BH_GEOLOGY_CLASS
,rtrim( cast( ydc.BH_COMMENT as varchar(255) ) ) as BH_COMMENT
,row_number() over (order by y.bore_hole_id) as rkey
into MOE_20210119.dbo.M_D_BOREHOLE
from
MOE_20210119.dbo.YC_20210119_BH_ID as y
left outer join MOE_20210119.dbo.YC_20210119_DRILL_CODE as ydc
on y.BORE_HOLE_ID=ydc.BORE_HOLE_ID
left outer join
(
select

```

```

moeh.Bore_Hole_ID
,MAX(moeh.Diameter) as [max_diameter]
,moeh.HOLE_DIAMETER_UOM
from
MOE_20210119.dbo.TblHole as moeh
where
moeh.diameter is not null
group by
Bore_Hole_ID,moeh.HOLE_DIAMETER_UOM
) as moeh
on y.BORE_HOLE_ID=moeh.Bore_Hole_ID
left outer join MOE_20210119.dbo.TblWWR as moewwr
on y.WELL_ID=moewwr.WELL_ID
left outer join MOE_20210119.dbo.YC_20210119_FINAL_STATUS as ycfs
on moewwr.FINAL_STA=ycfs.FINAL_STA
inner join MOE_20210119.dbo.TblBore_Hole as moebh
on y.BORE_HOLE_ID=moebh.BORE_HOLE_ID
left outer join MOE_20210119.dbo._code_Well_Type as moecwt
on moebh.CODEOB=moecwt.code
left outer join OAK_20160831_MASTER.dbo.R_BH_DRILLER_CODE as rbdc
on moewwr.CONTRACTOR collate database_default=rbdc.BH_DRILLER_ALT_CODE collate database_default

```

In order to account for any multiple drill methods used for a particular location we'll first create a view.

```

create view V_DBORE_DRILL_CODES as
select
dbore.rkey
,t.min_rkey
,t.max_rkey
,t.rcount
from
(
select
loc_id
,min(rkey) as min_rkey
,max(rkey) as max_rkey
,count(*) as rcount
from
MOE_20210119.dbo.M_D_BOREHOLE
group by
loc_id
) as t
inner join MOE_20210119.dbo.M_D_BOREHOLE as dbore
on t.loc_id=dbore.loc_id
where
t.rcount>1
and dbore.rkey=t.max_rkey

```

We can then use this as a base to create a series of temporary tables containing the additional drilling methods.

```

select
dbore.BH_ID
,dbore.LOC_ID
,dbore.BH_DRILL_METHOD_CODE
,dbore.rkey
--into MOE_20210119.dbo.YC_20210119_DBORE_DRILL_CODES_1
--into MOE_20210119.dbo.YC_20210119_DBORE_DRILL_CODES_2
--into MOE_20210119.dbo.YC_20210119_DBORE_DRILL_CODES_3
--into MOE_20210119.dbo.YC_20210119_DBORE_DRILL_CODES_4
from
MOE_20210119.dbo.M_D_BOREHOLE as dbore
inner join MOE_20210119.dbo.V_DBORE_DRILL_CODES as v
on dbore.rkey=v.rkey

```

After each table is created, we'll remove the particular record from M_D_BOREHOLE (until it matches the number of rows in M_D_LOCATION).

```
delete from MOE_20210119.dbo.M_D_BOREHOLE
where rkey in
(
select
rkey
from
--MOE_20210119.dbo.YC_20210119_DBORE_DRILL_CODES_1
--MOE_20210119.dbo.YC_20210119_DBORE_DRILL_CODES_2
--MOE_20210119.dbo.YC_20210119_DBORE_DRILL_CODES_3
--MOE_20210119.dbo.YC_20210119_DBORE_DRILL_CODES_4
)
```

These additional tables can then be used to generated the text describing these methods which can be added to the BH_COMMENT field.

```
update MOE_20210119.dbo.M_D_BOREHOLE
set
BH_COMMENT=
case
when bh_comment is null then t.to_add
else bh_comment + ';' + t.to_add
end
from
MOE_20210119.dbo.M_D_BOREHOLE as moe
inner join
(
select
dbore.LOC_ID
,'Addn drill methods: ' + r1.bh_drill_method_description +
case
when d2.loc_id is not null then
case
when d3.loc_id is not null then
';' + r2.bh_drill_method_description + ';' + r3.bh_drill_method_description
else ';' + r2.bh_drill_method_description
end
end
else "
end as to_add
from
MOE_20210119.dbo.M_D_BOREHOLE as dbore
inner join MOE_20210119.dbo.YC_20210119_DBORE_DRILL_CODES_1 as d1
on dbore.loc_id=d1.loc_id
inner join OAK_20160831_MASTER.dbo.R_BH_DRILL_METHOD_CODE as r1
on d1.bh_drill_method_code=r1.bh_drill_method_code
left outer join MOE_20210119.dbo.YC_20210119_DBORE_DRILL_CODES_2 as d2
on dbore.loc_id=d2.loc_id
left outer join OAK_20160831_MASTER.dbo.R_BH_DRILL_METHOD_CODE as r2
on d2.bh_drill_method_code=r2.bh_drill_method_code
left outer join MOE_20210119.dbo.YC_20210119_DBORE_DRILL_CODES_3 as d3
on dbore.loc_id=d3.loc_id
left outer join OAK_20160831_MASTER.dbo.R_BH_DRILL_METHOD_CODE as r3
on d3.bh_drill_method_code=r3.bh_drill_method_code
) as t
on moe.loc_id=t.loc_id
```

Note that these scripts will be modified depending upon the maximum number of drilling methods specified for any particular location.

Script: G_10_09_05_M_D_BOREHOLE.sql

G.10.10 Borehole construction information

A temporary borehole construction table is first setup allowing the various construction elements to be incorporated. Once this is done, a random SYS_RECORD_ID can be added to the table (using a row-count relationship).

The possible construction elements includes both casing and plug information. As the casing material list is limited, a 'CASE ... END' statement will be used instead of a look-up table. Assemble the casing information (at least one column must have information before the row is included).

```
select
-- note that we're using BORE_HOLE_ID as a temporary BH_ID
y.BORE_HOLE_ID as BH_ID
.case
when moec.MATERIAL is null then 10 -- unknown
when moec.MATERIAL = 1 then 21 -- steel casing
when moec.MATERIAL = 2 then 16 -- galvanized
when moec.MATERIAL = 3 then 23 -- concrete
when moec.MATERIAL = 4 then 24 -- open hole
when moec.MATERIAL = 5 then 25 -- plastic
when moec.MATERIAL = 6 then 32 -- fibreglass
when moec.MATERIAL = 7 then 10 -- unknown/other
when moec.MATERIAL = 8 then 52 -- stainless steel
else -9999
end
as [CON_SUBTYPE_CODE]
.moec.DEPH_FROM as [CON_TOP_OUOM]
.moec.DEPH_TO as [CON_BOT_OUOM]
.moec.CASING_DEPTH_UOM as [CON_UNIT_OUOM]
.moec.CASING_DIAMETER as [CON_DIAMETER_OUOM]
.moec.CASING_DIAMETER_UOM as [CON_DIAMETER_UNIT_OUOM]
,convert(varchar(255),null) as CON_COMMENT
from
MOE_20200721.dbo.YC_20200721_BH_ID as y
inner join MOE_20200721.dbo.TblPipe as moep
on y.BORE_HOLE_ID=moep.Bore_Hole_ID
inner join MOE_20200721.dbo.TblCasing as moec
on moep.PIPED_ID=moec.PIPED_ID
where
not
(
moec.DEPH_TO is null
and moec.CASING_DEPTH_UOM is null
and moec.CASING_DIAMETER is null
and moec.CASING_DIAMETER_UOM is null
)
```

Assemble the plug information.

```
insert into MOE_20170905.dbo.YC_20170905_DBHCONS
(BH_ID,CON_SUBTYPE_CODE,CON_TOP_OUOM,CON_BOT_OUOM,CON_UNIT_OUOM)
select
ycb.BORE_HOLE_ID as BH_ID
,31 as CON_SUBTYPE_CODE
.moep.PLUG_FROM as [CON_TOP_OUOM]
.moep.PLUG_TO as [CON_BOT_OUOM]
.moep.PLUG_DEPTH_UOM as [CON_UNIT_OUOM]
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
inner join MOE_20170905.dbo.TblPlug as moep
on ycb.BORE_HOLE_ID=moep.BORE_HOLE_ID
```

```

where
not
(
moep.PLUG_FROM is null
and moep.PLUG_TO is null
and moep.PLUG_DEPTH_UOM is null
)

```

The CON_TOP_OUOM should always be less than the CON_BOT_OUOM values. Check this and correct as necessary (not shown).

Script: G_10_10_01_DBHCONS.sql

A series of SYS_RECORD_IDs are now created for the records in the YC_20160531_DBHCONS table (remember that SYS_RECORD_ID is used to create a primary key for a table but cannot be used to create relationships between tables). There will be one SYS_RECORD_ID for each row. Here, though, we'll just implement a counter (which will function as a temporary value). When importing into the master database, we'll replace this counter value with an actual random value.

```

SELECT
[BH_ID]
,[CON_SUBTYPE_CODE]
,[CON_TOP_OUOM]
,[CON_BOT_OUOM]
,[CON_UNIT_OUOM]
,[CON_DIAMETER_OUOM]
,[CON_DIAMETER_UNIT_OUOM]
,[CON_COMMENT]
,ROW_NUMBER() over (order by y.BH_ID) as SYS_RECORD_ID
into MOE_20170905.dbo.M_D_BOREHOLE_CONSTRUCTION
FROM
MOE_20170905.[dbo].[YC_20170905_DBHCONS] as y

```

Script: G_10_10_02_M_D_BORE_CONS.sql

G.10.11 Location primary and secondary purposes

The recorded MOE usage codes (i.e. LOC_MOE_USE_1ST_CODE and LOC_MOE_USE_2ND_CODE as found in the M_D_LOCATION table) are used, in conjunction (in some cases) with the BH_STATUS_CODE to assign primary and secondary usages to each borehole. Refer to the 'Purposes – Assigned' table and SQL code for assignment logic.

Due to their excessive length, this code is only found in the following script file. Note that the script(s) may need to be run more than once in the case that updates to the assignment tables (above) need to be made (not shown).

Script: G_10_11_01_M_D_LOC_PURP.sql

G.10.12 Geology feature information (water found)

The D_GEOLOGY_FEATURE table (for information from the WWDB) only contains the ‘Water Found’ values (including the kind and depths; the rows without depths are still incorporated as placeholders, for later editing). Create the table and include a temporary count for the SYS_RECORD_ID value.

```
select
y.BORE_HOLE_ID as LOC_ID
,case
when moew.kind is null or moew.kind=0 then null -- not specified
else moew.kind -- matches YC codes
end
as [FEATURE_CODE]
,'Water Found' as [FEATURE_DESCRIPTION]
,moew.WATER_FOUND_DEPTH as [FEATURE_TOP_OUOM]
,moew.WATER_FOUND_DEPTH_UOM as [FEATURE_UNIT_OUOM]
,ROW_NUMBER() over (order by y.LOC_ID) as [SYS_RECORD_ID]
into MOE_20170905.dbo.M_D_GEOLOGY_FEATURE
from
MOE_20170905.dbo.YC_20170905_BH_ID as y
inner join MOE_20170905.dbo.TblPipe as moep
on y.BORE_HOLE_ID=moep.Bore_Hole_ID
inner join MOE_20170905.[dbo].[TblWater] as moew
on moep.PIPE_ID=moew.PIPE_ID
```

Script: G_10_12_01_M_D_GEOL_FEAT.sql

G.10.13 Geology

The geology layer look-up codes are equivalent between the WWDB and YCDB with the exception of code ‘27’ (i.e. ‘Other’). This is to be matched to code ‘0’ (i.e. ‘Unknown’) for each of the material fields (MAT1 through MAT3). The process is not described here – refer to the following script.

Script: G_10_13_01_GL_OTHER.sql

All depth units from the WWDB must be in either ‘ft’ or ‘m’; in some cases, ‘inch’ or ‘cm’ is used instead. These should be changed – the process is not described here – refer to the following script.

Script: G_10_13_02_GL_UNITS.sql

The geology table (i.e. equivalent to D_GEOLOGY_LAYER) can now be created. Note that a few modifications are made.

- If all material codes are NULL, MAT1 is assigned a ‘0’
- If MAT1 is null but MAT2 is not, make MAT1=MAT2 then make MAT2 NULL
- If MAT1 and MAT2 is null but MAT3 is not, make MAT1=MAT3 and NULL MAT2 and MAT3
- If any of these conditions are met, an appropriate comment is added, otherwise NULL

Review the script itself for details.

```

select
-- note that we're substituting the BORE_HOLE_ID for LOC_ID
ycb.BORE_HOLE_ID as LOC_ID
.case
  when moef.COLOR is null or moef.COLOR=0 then null
  else moef.COLOR
end
as [GEOL_MAT_COLOUR_CODE]
.moef.FORMATION_TOP_DEPTH as GEOL_TOP_OUOM
.moef.FORMATION_END_DEPTH as GEOL_BOT_OUOM
.moef.FORMATION_END_DEPTH_UOM as GEOL_UNIT_OUOM
.moef.LAYER as GEOL_MOE_LAYER
.case
-- check if all mat fields are null
  when moef.MAT1 is null and moef.MAT2 is null and moef.MAT3 is null then 0
-- if mat1 is null but mat2 is not, make mat1=mat2
  when moef.MAT1 is null and moef.MAT2 is not null then moef.MAT2
-- if mat1 and mat2 is null but mat3 is not, make mat1=mat3
  when moef.MAT1 is null and moef.MAT2 is null and moef.MAT3 is not null then moef.MAT3
  else moef.MAT1
end as GEOL_MAT1_CODE
.case
-- if all mat fields are null, leave mat2 as is
-- if mat1 is null, mat2 has been moved to mat1, return null
-- we won't move mat3 to mat2
  when moef.MAT1 is null and moef.MAT2 is not null then null
  else moef.MAT2
end as GEOL_MAT2_CODE
.case
-- if mat1 and mat2 is null but mat3 is not, make mat1=mat3
  when moef.MAT1 is null and moef.MAT2 is null and moef.MAT3 is not null then null
  else moef.MAT3
end as GEOL_MAT3_CODE
-- include some comments if we've messed about with the mat codes
.case
  when moef.MAT1 is null and moef.MAT2 is null and moef.MAT3 is null then 'No material, assigned unknown'
  when moef.MAT1 is null and moef.MAT2 is not null then 'No mat1, assigned mat2 to mat1'
  when moef.MAT1 is null and moef.MAT2 is null and moef.MAT3 is not null then 'No mat1 or mat2, assigned mat3 to mat1'
  else null
end as GEOL_COMMENT
.ROW_NUMBER() over (order by ycb.LOC_ID) as SYS_RECORD_ID
into MOE_20170905.dbo.M_D_GEOLOGY_LAYER
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
inner join MOE_20170905.dbo.TblFormation as moef
on ycb.BORE_HOLE_ID=moef.BORE_HOLE_ID

```

Script: G_10_13_03_M_D_GEOL_LAY.sql

G.10.14 Determine maximum depth

Maximum depth for each borehole needs to be determined – this information is found in the D_BOREHOLE table as well as being used for creating the D_INTERVAL_MONITOR table. The maximum depths themselves can come from any of formation depth, reported MOE depth or construction depths. These fields in YC_20130923_BHID are populated and then the final, maximum depth is determined.

We'll first determine the maximum depth from construction details. We need to make sure that the depths are in 'm' or 'ft' only.

```

select
ycb.LOC_ID
,ycb.BH_ID

```

```

,ycm.*
from
[MOE_20170905].dbo.YC_20170905_BH_ID as ycb
inner join [MOE_20170905].dbo.M_D_BOREHOLE_CONSTRUCTION as ycm
on ycb.BORE_HOLE_ID=ycm.BH_ID
where
not(ycm.CON_UNIT_OUOM in ('m','ft'))

```

The common cases are those depths specified using ‘inch’ or ‘cm’ – convert these to the appropriate units (only inches found for MOE_20170905; script not shown).

Script: G_10_14_01_CONS_UNITS.sql

We can now calculate the maximum depth based upon the construction details.

```

update MOE_20170905.dbo.YC_20170905_BH_ID
set
CON_MAX_DEPTH=yccon.CON_MAX_DEPTH
,CON_MAX_DEPTH_UNITS=yccon.CON_MAX_DEPTH_UNITS
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
inner join
(
select
ycm.BH_ID
,max(ycm.CON_BOT_OUOM) as [CON_MAX_DEPTH]
,ycm.CON_UNIT_OUOM as CON_MAX_DEPTH_UNITS
,COUNT(*) as rcount
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
inner join MOE_20170905.dbo.M_D_BOREHOLE_CONSTRUCTION as ycm
on ycb.BORE_HOLE_ID=ycm.BH_ID
group by
ycm.BH_ID,ycm.CON_UNIT_OUOM
) as yccon
on
ycb.BORE_HOLE_ID=yccon.BH_ID

```

Script: G_10_14_02_CONS_DEPTH.sql

We’ll now look at the maximum depth as reported by the MOE. Similar to ‘G_10_14_01_CONS_UNITS.sql’, above, check that the units are ‘m’ or ‘ft’ and correct those that are not (only the check is included here, refer to the script for correction details; no corrections were necessary in MOE_20170905).

```

select
ycb.LOC_ID
,ycb.BH_ID
,moeh.*
from
MOE_20170905.dbo.TblHole as moeh
inner join MOE_20170905.dbo.YC_20170905_BH_ID as ycb
on moeh.Bore_Hole_ID=ycb.BORE_HOLE_ID
where
moeh.Depth_to is not null
and not(moeh.HOLE_DEPTH_UOM in ('m','ft'))

```

Script: G_10_14_03_MOE_UNITS.sql

Determine the maximum depth as reported by the MOE (and update YC_20170905_BH_ID). Make sure to check the record counts in the case that multiple units are used in specifying the depths (refer to the script-file itself as this latter check is not shown here).

```
update MOE_20170905.dbo.YC_20170905_BH_ID
set
MOE_MAX_DEPTH=moe_depth.MOE_MAX_DEPTH
,MOE_MAX_DEPTH_UNITS=moe_depth.MOE_MAX_DEPTH_UNITS
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycbh
inner join
(
select
moech.Bore_Hole_ID
,max(moech.Depth_to) as MOE_MAX_DEPTH
,moech.HOLE_DEPTH_UOM as MOE_MAX_DEPTH_UNITS
from
MOE_20170905.dbo.TblHole as moech
inner join MOE_20170905.dbo.YC_20170905_BH_ID as ycb
on moech.Bore_Hole_ID=ycb.BORE_HOLE_ID
where
moech.Depth_to is not null
group by
moech.Bore_Hole_ID,moech.HOLE_DEPTH_UOM
) as moe_depth
on ycbh.BORE_HOLE_ID=moe_depth.Bore_Hole_ID
```

Script: G_10_14_04_MOE_DEPTH.sql

Determine the maximum depth of the borehole based upon the geology information (and update YC_20170905_BH_ID). Make sure to check the record counts in the case that multiple units are used in specifying the depths (refer to the script-file itself as this latter check is not shown here).

```
update MOE_20170905.dbo.YC_20170905_BH_ID
set
fm_max_depth=fm_depth.FM_MAX_DEPTH
,fm_max_depth_units=fm_depth.FM_MAX_DEPTH_UNITS
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
inner join
(
select
BORE_HOLE_ID
,max(FORMATION_END_DEPTH) as [FM_MAX_DEPTH]
,FORMATION_END_DEPTH_UOM as [FM_MAX_DEPTH_UNITS]
from
MOE_20170905.[dbo].[TblFormation]
where
BORE_HOLE_ID
in
(
select
BORE_HOLE_ID
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
)
and FORMATION_END_DEPTH is not null
group by
BORE_HOLE_ID,FORMATION_END_DEPTH_UOM
) as fm_depth
on ycb.BORE_HOLE_ID=fm_depth.BORE_HOLE_ID
```

Script: G_10_14_05_FM_DEPTH.sql

Now that the possible depths have been determined, convert all of them to metres (i.e. 'm'). This value is used to populate MAX_DEPTH_M in YC_20170905_BH_ID. Note that the check and update scripts are not shown (but can be found in the script-file itself).

Script: G_10_14_06_DEPTH_M.sql

Determine the MAX_DEPTH_M from the other depths.

```
update MOE_20170905.dbo.YC_20170905_BH_ID
set
MAX_DEPTH_M=md.MAX_DEPTH_M
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
inner join
(
select
depths.BORE_HOLE_ID
,max(depths.adepth) as [MAX_DEPTH_M]
from
(
select
ycb.BORE_HOLE_ID
,ycb.FM_MAX_DEPTH as adepth
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
where ycb.FM_MAX_DEPTH is not null
)
)
union
(
select
ycb.BORE_HOLE_ID
,ycb.MOE_MAX_DEPTH as adepth
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
where ycb.MOE_MAX_DEPTH is not null
)
)
union
(
select
ycb.BORE_HOLE_ID
,ycb.CON_MAX_DEPTH as adepth
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
where ycb.CON_MAX_DEPTH is not null
)
) as depths
group by
depths.BORE_HOLE_ID
) as md
on ycb.BORE_HOLE_ID=md.BORE_HOLE_ID
```

(For MOE_20170905, 13150 rows were updated.) For those locations with NULL depths (4035), examine alternate options for determining an approximate depth. This includes examination of FM_D_GEOLOGY_FEATURE, 'TblScreen', 'TblPump_Test' and 'TblWater' (not shown; examine script for details). In this case, very few additional depths were determined (refer to script for details) leaving 4024 without depth data.

Script: G_10_14_07_MAX_DEPTH.sql

We can now populate the depths and elevations in D_BOREHOLE.

```
update MOE_20210119.dbo.M_D_BOREHOLE
set
BH_GND_ELEV=delev.LOC_ELEV
,BH_GND_ELEV_OUOM=delev.LOC_ELEV
,BH_DEM_GND_ELEV=delev.LOC_ELEV
,BH_BOTTOM_ELEV=(delev.LOC_ELEV-ycb.MAX_DEPTH_M)
,BH_BOTTOM_DEPTH=ycb.MAX_DEPTH_M
,BH_BOTTOM_OUOM=ycb.MAX_DEPTH_M
from
MOE_20210119.dbo.M_D_BOREHOLE as dbore
inner join MOE_20210119.dbo.YC_20210119_BH_ID as ycb
on dbore.LOC_ID=ycb.BORE_HOLE_ID
inner join
(
select
dlsh.LOC_ID
,LOC_ELEV
from
MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST as dlsh
where
dlsh.LOC_ELEV_CODE=3
-- Only load the SRTM elev if no MNR elev
union
select
dlsh.LOC_ID
,LOC_ELEV
from
MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST as dlsh
where
dlsh.LOC_ELEV_CODE=5
) as delev
on dbore.LOC_ID=delev.LOC_ID
```

Script: G_10_14_08_D_BORE_Update

Note that we're assuming in the script that only a DEM value will be used to populate the assigned elevation (as found in BH_GND_ELEV). This will need to be adjusted for any MOE location with a QA_COORD_CONFIDENCE_CODE of '1' (i.e. a surveyed location).

G.10.15 Screen information

Before creation of the (compatible) D_INTERVAL_MONITOR table, we need to determine the interval types present (i.e. as represented in R_INT_TYPE_CODE). A look-up table translating the WWDB screen slot size (a text field) to the YCDB screen slot size (a numeric field) must be created first.

```
select
moes.Slot as MOE_SLOT
,convert(varchar(50),moes.Slot) as YC_SLOT
,cast(null as float) as CONV_YC_SLOT
into MOE_20170905.dbo.YC_20170905_MOE_SLOT
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
inner join MOE_20170905.dbo.TblPipe as moep
on ycb.BORE_HOLE_ID=moep.Bore_Hole_ID
inner join MOE_20170905.dbo.TblScreen as moes
```

```

on moep.PIPE_ID=moes.PIPE_ID
where
moes.SCRN_TOP_DEPTH is not null
and moes.SCRN_END_DEPTH is not null
group by
moes.Slot

```

Note that this look-up table will likely need to be modified. The WWDB MOE_SLOT information is in a 'text' form while the YCDB slot format is 'float' – the YC_20170905_MOE_SLOT table should have its YC_SLOT field modified until all values can be converted quickly. A test script is available to check user edits (a non-failure indicates the conversion between databases is correct).

```

select
ycm.MOE_SLOT
,ycm.YC_SLOT
,cast(YC_SLOT as float) as CONV_YC_SLOT
from
MOE_20170905.dbo.YC_20170905_MOE_SLOT as ycm

```

Script: G_10_15_01_SLOT_SIZE.sql

We can now extract the reported screens from the WWDB (i.e. using 'TblScreen', where both the bottom and top depths are present). Substituting the 'inner join' with the 'left outer join' as indicated will allow a check of the 1-to-1 relationship between the BORE_HOLE_ID and the IDs present in 'TblPipe' and 'TblScreen' (if the counts are not the same between runs, there are multiple screens in some of the boreholes). Detail is available in the script file. We're storing the information in a temporary table until all interval types have been added.

```

select
ycb.BORE_HOLE_ID as TMP_LOC_ID
,18 as tmp_INT_TYPE_CODE
,moeslot.CONV_YC_SLOT as MON_SCREEN_SLOT
,moes.SCRN_MATERIAL as MON_SCREEN_MATERIAL
,moes.SCRN_DIAMETER as MON_DIAMETER_OUOM
,moes.SCRN_DIAMETER_UOM as MON_DIAMETER_UNIT_OUOM
,moes.SCRN_TOP_DEPTH as MON_TOP_OUOM
,moes.SCRN_END_DEPTH as MON_BOT_OUOM
,moes.SCRN_DEPTH_UOM as MON_UNIT_OUOM
,cast(null as varchar(255)) as MON_COMMENT
into MOE_20170905.dbo.YC_20170905_DINTMON
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
inner join
--left outer join
MOE_20170905.dbo.TblPipe as moep
on ycb.BORE_HOLE_ID=moep.Bore_Hole_ID
inner join
--left outer join
MOE_20170905.dbo.TblScreen as moes
on moep.PIPE_ID=moes.PIPE_ID
left outer join
MOE_20170905.dbo.YC_20170905_MOE_SLOT as moeslot
on moes.Slot=moeslot.MOE_SLOT
where
moes.SCRN_TOP_DEPTH is not null
or moes.SCRN_END_DEPTH is not null

```

Script: G_10_15_02_SCR_REPORT.sql

In this case, we're using a temporary identifier (TMP_LOC_ID) until an actual INT_ID is to be created (in the case we'll need multiple INT_IDs for each LOC_ID). Note that we're 'parking' the interval type (i.e. TMP_INT_TYPE_CODE) here temporarily until the D_INTERVAL table is created.

We'll now find those wells classified as 'open hole' – the screen here is considered to be from the bottom of any casing down to the bottom of the hole. (These intervals should not already be in the YC_20170905_DINTMON as there is no reported screen; remember, we've loaded the reported screens into this temporary table.)

```
select
ycb.BORE_HOLE_ID
,ycdim.TMP_LOC_ID
,moebh.*
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
inner join MOE_20170905.dbo.TblBore_Hole as moebh
on ycb.BORE_HOLE_ID=moebh.BORE_HOLE_ID
inner join MOE_20170905.dbo.YC_20170905_DINTMON as ycdim
on ycb.BORE_HOLE_ID=ycdim.TMP_LOC_ID
where
moebh.OPEN_HOLE='Y'
```

This is provided as a check and should return no screens/intervals (if the previous procedures worked).

We can now determine how many 'open hole' wells have casing. This is a check only.

```
select
ycb.BORE_HOLE_ID
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
inner join MOE_20170905.dbo.TblBore_Hole as moebh
on ycb.BORE_HOLE_ID=moebh.BORE_HOLE_ID
left outer join MOE_20170905.dbo.M_D_BOREHOLE_CONSTRUCTION as ycbc
on ycb.BORE_HOLE_ID=ycbc.BH_ID
where
moebh.OPEN_HOLE='Y'
-- this a list of casing info
and ycbc.CON_SUBTYPE_CODE in (9,16,21,23,25,32)
```

If any rows are returned, we can then add these as well screen intervals where the interval type is classified as 'open hole, bottom-of-casing to bottom-of-hole'. If the casing depths return a NULL value, the maximum construction depth from YC_20180530_BH_ID is substituted. In all cases when populating the temporary interval monitor table (i.e. YC_20170905_DINTMON), we're not checking against NULL tops or zero-length screens. This is to be considered at a later step. For MOE_20170905, no rows were added from this process.

```
insert into [MOE_20180530].dbo.YC_20180530_DINTMON
(TMP_LOC_ID,TMP_INT_TYPE_CODE,MON_TOP_OUOM,MON_BOT_OUOM,MON_UNIT_OUOM,MON_COMMENT)
select
ycb.BORE_HOLE_ID as TMP_LOC_ID
,21 as TMP_INT_TYPE_CODE
--,cd.CASING_DEPTH as MON_TOP_OUOM
,case
```

```

when cd.CASING_DEPTH is not null then cd.CASING_DEPTH
else ycb.CON_MAX_DEPTH
end as MON_TOP_OUOM
,ycb.MAX_DEPTH_M as MON_BOT_OUOM
,'m' as MON_UNIT_OUOM
--, (ycb.MAX_DEPTH_M - cd.CASING_DEPTH) as DIFF_OPEN_HOLE
,'open hole; bottom-of-casing to bottom-of-hole' as MON_COMMENT
from
[MOE_20180530].dbo.YC_20180530_BH_ID as ycb
inner join [MOE_20180530].dbo.TblBore_Hole as moebh
on ycb.BORE_HOLE_ID=moebh.BORE_HOLE_ID
left outer join
(
select
cd.BH_ID
,max(CON_BOT_OUOM) as [CASING_DEPTH]
from
(
select
ycbc.BH_ID
,case
when ycbc.CON_UNIT_OUOM='ft' then ycbc.CON_BOT_OUOM*0.3048
else ycbc.CON_BOT_OUOM
end
as CON_BOT_OUOM
from
[MOE_20180530].dbo.M_D_BOREHOLE_CONSTRUCTION as ycbc
where
ycbc.CON_SUBTYPE_CODE in (9,16,21,23,25,32)
) as cd
group by
cd.BH_ID
) as cd
on
ycb.BH_ID=cd.BH_ID
where
moebh.OPEN_HOLE='Y'
--and (ycb.MAX_DEPTH_M - cd.CASING_DEPTH) > 0

```

Script: G_10_15_03_SCR_OPENHOLE.sql

We would now examine the number of screen intervals that are reported as ‘open hole’ but do not have any casing information associated with them. In addition, we’re only interested in locations that fall within the YPDT-CAMC area. The SQL statements and explanation are thus only found in the external script. No rows were returned from the MOE_20170905 database. From the MOE_20180530 database, 18 records were returned – these were handled, above (Section G.10.15.03), by applying the maximum construction depth.

Script: G_10_15_04_SCR_OPENHOLE_NC.sql

We need to determine, now, how many locations still require a screen interval to be assigned and the particular ‘assumed’ type to apply.

Currently, how many locations have screens assigned?

```

select
COUNT(*) as [Distinct_LOC_ID]
from
(
select

```

```

ydim.TMP_LOC_ID
,COUNT(*) as recount
from
MOE_20170905.dbo.YC_20170905_DINTMON ydim
group by
ydim.TMP_LOC_ID
) as test

```

How many locations/intervals (using a one-to-one relationship for these MOE boreholes) remain to be assigned.

```

select
COUNT(*) as [To_Assign_tmp_LOC_IDs]
from
MOE_20170905.dbo.YC_20170905_BH_ID as ybc
where
ybc.BORE_HOLE_ID
not in
(
select distinct(tmp_LOC_ID) from MOE_20170905.dbo.YC_20170905_DINTMON
)

```

Script: G_10_15_05_SCR_NUMBER.sql

Various assumptions are made for applying the screen/interval types to the remainder of the boreholes.

We'll now check for the existence of bedrock within the hole. We can then assign the interval type 'Assumed Open Hole (Top of Bedrock to Bottom of Hole)'. Bedrock types are defined in R_GEOL_MAT1_CODE as having a ROCK value of '1' (i.e. a 'True' value). Add these to the YC_20170905_DINTMON table.

```

insert into MOE_20170905.dbo.YC_20170905_DINTMON
(tmp_LOC_ID,tmp_INT_TYPE_CODE,MON_TOP_OUOM,MON_BOT_OUOM,MON_UNIT_OUOM,MON_COMMENT)
select
bi.LOC_ID as tmp_LOC_ID
,22 as tmp_INT_TYPE_CODE
,bi.MON_TOP_OUOM
,case
when ybc.MAX_DEPTH_M>bi.MON_BOT_OUOM then ybc.MAX_DEPTH_M
else bi.MON_BOT_OUOM
end
as MON_BOT_OUOM
,'m' as MON_UNIT_OUOM
,'bedrock, no valid casing; open hole, top-of-bedrock to bottom-of-hole' as MON_COMMENT
from
(
select
fmdgl.LOC_ID
,case
when fmdgl.GEOL_UNIT_OUOM='ft' then MIN(fmdgl.GEOL_TOP_OUOM)*0.3048
else MIN(fmdgl.GEOL_TOP_OUOM)
end as MON_TOP_OUOM
,case
when fmdgl.GEOL_UNIT_OUOM='ft' then MAX(fmdgl.GEOL_BOT_OUOM)*0.3048
else MAX(fmdgl.GEOL_BOT_OUOM)
end as MON_BOT_OUOM
from
MOE_20170905.dbo.M_D_GEOLOGY_LAYER as fmdgl
inner join OAK_20160831_MASTER.dbo.R_GEOL_MAT1_CODE as rgmc
on fmdgl.GEOL_MAT1_CODE=rgmc.GEOL_MAT1_CODE
where

```

```

fmdgl.LOC_ID
in
(
select
ybc.BORE_HOLE_ID as LOC_ID
from
MOE_20170905.dbo.YC_20170905_BH_ID as ybc
where
ybc.BORE_HOLE_ID
not in
( select tmp_LOC_ID from MOE_20170905.dbo.YC_20170905_DINTMON )
)
and rgmc.GEOL_MAT1_ROCK=1
group by
fmdgl.LOC_ID,fmdgl.GEOL_UNIT_OUOM
) as bi
inner join MOE_20170905.dbo.YC_20170905_BH_ID as ybc
on bi.LOC_ID=ybc.BORE_HOLE_ID

```

Script: G_10_15_06_SCR_BEDROCK.sql

We could use the scripts from G_10_15_05_SCR_NUMBER to determine how many locations now require intervals to be assigned (as a check). For this remainder we will use the interval type ‘Overburden – Assumed (1ft above bottom of hole)’ (i.e. a value of ‘19’ as found in R_INT_TYPE_CODE; when we’re assigning a screen interval in meters, though, a ‘0.3’ metre value is used instead). Note that for these screens/intervals, we’re imposing the metre as the depth unit (instead of checking to see how the various depth units are applied for any particular location). Now, if the number of rows returned here doesn’t match the number of rows expected, check the records that do not have a valid maximum depth (to make up the difference).

The number of rows returned here should make up the remainder (otherwise some troubleshooting will be required). Add the overburden screens/intervals.

```

insert into MOE_20170905.dbo.YC_20170905_DINTMON
(tmp_LOC_ID,tmp_INT_TYPE_CODE,MON_TOP_OUOM,MON_BOT_OUOM,MON_UNIT_OUOM,MON_COMMENT)
select
ybc.BORE_HOLE_ID as tmp_LOC_ID
,19 as tmp_INT_TYPE_CODE
,(ybc.MAX_DEPTH_M-0.3) as MON_TOP_OUOM
,ybc.MAX_DEPTH_M as MON_BOT_OUOM
,'m' as MON_UNIT_OUOM
,'overburden; assumed screen 0.3m above bottom' as MON_COMMENT
from
MOE_20170905.dbo.YC_20170905_BH_ID as ybc
where
ybc.BORE_HOLE_ID not in
( select tmp_LOC_ID from MOE_20170905.dbo.YC_20170905_DINTMON )
and ybc.MAX_DEPTH_M is not null

```

Script: G_10_15_07_SCR_OVERBUR.sql

The remaining non-assigned locations (i.e. without an assigned interval) are considered to have no valid information for creation of a screen. These will be given an interval type of ‘Screen Information Omitted’ (a value of ‘28’).

Note that these are to be added to YC_20170905_DINTMON temporarily only (as placeholders). They will be used for populating FM_D_INTERVAL but will not be transferred to FM_D_INTERVAL_MONITOR.

```
insert into MOE_20170905.dbo.YC_20170905_DINTMON
(TMP_LOC_ID,TMP_INT_TYPE_CODE,MON_COMMENT)
select
ycb.BORE_HOLE_ID as tmp_LOC_ID
,28 as tmp_INT_TYPE_CODE
,'no valid screen determined' as MON_COMMENT
from
MOE_20170905.dbo.YC_20170905_BH_ID as ycb
where
ycb.BORE_HOLE_ID
not in
( select tmp_LOC_ID from MOE_20170905.dbo.YC_20170905_DINTMON )
and ycb.MAX_DEPTH_M is null
```

Script: G_10_15_08_SCR_NODATA.sql

Make sure that the MON_TOP_OUOM is above (smaller than) MON_BOT_OUOM.

```
update MOE_20170905.dbo.YC_20170905_DINTMON
set
MON_TOP_OUOM=MON_BOT_OUOM
,MON_BOT_OUOM=MON_TOP_OUOM
where
MON_BOT_OUOM<MON_TOP_OUOM
```

We can now copy the information from YC_20170905_DINTMON to M_D_INTERVAL_MONITOR. We should first check the number of records in the first table against the number of valid locations from YC_20170905_BH_ID (not shown, see the script file, below, for details). In MOE_20160531, a value of 28,575 versus 28,188 was determined; there must be some locations with multiple intervals (in MOE_20170905, no duplicates were found). Check that there are no duplicates and then check that there is only a single static water level reading per BORE_HOLE_ID/PIPE_ID (from the original MOE WWDB information). Only the final SQL statement is shown (refer to the script file for additional details).

```
select
t3.BORE_HOLE_ID
,t3.rcount
from
(
select
t2.BORE_HOLE_ID
,COUNT(*) as rcount
from
(
--*****
SELECT
moept.[PIPE_ID]
,t1.BORE_HOLE_ID
,[WELL_ID]
,[Static_lev]
,[LEVELS_UOM]
FROM
MOE_20170905.[dbo].[TblPump_Test] as moept
inner join
(
```

```

select
moetp.PIPe_ID
,ycb.BORE_HOLE_ID
from
MOE_20170905.dbo.[YC_20170905_BH_ID] as ycb
inner join MOE_20170905.dbo.TblPipe as moetp
on ycb.BORE_HOLE_ID=moetp.Bore_Hole_ID
) as t1
on moetp.PIPe_ID=t1.PIPe_ID
where
moetp.Static_lev is not null
_*****
) as t2
group by
t2.BORE_HOLE_ID
) as t3
where
t3.rcount>1

```

No locations were duplicated. Check for duplicate (or null) values for screen info.

```

select
test.TMP_LOC_ID
,ycd.*
from
(
select
TMP_LOC_ID
,COUNT(*) as rcount
from
MOE_20170905.dbo.YC_20170905_DINTMON as ycd
group by
TMP_LOC_ID,MON_SCREEN_SLOT,MON_SCREEN_MATERIAL,MON_DIAMETER_OUOM,
MON_DIAMETER_UNIT_OUOM,MON_TOP_OUOM,MON_BOT_OUOM
) as test
inner join
MOE_20170905.dbo.YC_20170905_DINTMON as ycd
on test.TMP_LOC_ID=ycd.TMP_LOC_ID
where
test.rcount>1
order by
test.TMP_LOC_ID

```

There are 0 here for MOE_20170905 (in MOE_20160531 there were 20); any found , here, would need to be corrected after assigning a SYS_RECORD_ID.

We'll copy the YC_20170905_DINTMON to M_D_INTERVAL_MONITOR adding a blank INT_ID column and a counter SYS_RECORD_ID column.

Check that the number of rows from the source table matches the row number count in the output (not shown), then create the M_D_INTERVAL_MONITOR table.

```

select
TMP_LOC_ID as INT_ID
,dim.tmp_INT_TYPE_CODE
,dim.MON_SCREEN_SLOT
,cast(moeccm.DES as varchar(255)) as MON_SCREEN_MATERIAL
,dim.MON_DIAMETER_OUOM
,dim.MON_DIAMETER_UNIT_OUOM
,dim.MON_TOP_OUOM
,dim.MON_BOT_OUOM
,dim.MON_UNIT_OUOM
,dim.MON_COMMENT

```

```
.cast(null as int) as MON_FLOWING
,row_number() over (order by dim.TMP_LOC_ID) as SYS_RECORD_ID
into MOE_20170905.dbo.M_D_INTERVAL_MONITOR
from
MOE_20170905.dbo.YC_20170905_DINTMON as dim
left outer join MOE_20170905.dbo._code_casing_material as moeccm
on dim.MON_SCREEN_MATERIAL=moeccm.CODE
```

Note that we're copying TMP_LOC_ID to INT_ID (temporarily) and using a row counter for SYS_RECORD_ID. Delete any duplicates (that we noticed earlier; this occurred in MOE_20160531).

```
delete from MOE_20160531.dbo.M_D_INTERVAL_MONITOR
where
SYS_RECORD_ID in
(
select
dim.SYS_RECORD_ID
from
MOE_20160531.dbo.M_D_INTERVAL_MONITOR as dim
inner join
(
select
t.INT_ID
,t.keep_SRI
from
(
select
INT_ID
,COUNT(*) as rcount
,min(SYS_RECORD_ID) as keep_SRI
from
[MOE_20160531].dbo.M_D_INTERVAL_MONITOR as dim
group by
INT_ID,MON_SCREEN_SLOT,MON_SCREEN_MATERIAL,MON_DIAMETER_OUOM,MON_DIAMETER_UNIT_OUOM,MON_TOP_OUOM,MON_BOT_OUOM
) as t
where t.rcount>1
) as t2
on dim.INT_ID=t2.INT_ID
where
dim.SYS_RECORD_ID<>t2.keep_SRI
```

Make sure that we're catching all the locations/intervals.

```
select
b.*
from
MOE_20170905.dbo.YC_20170905_BH_ID as b
where
BORE_HOLE_ID
not in
(
select
INT_ID
from
MOE_20170905.dbo.M_D_INTERVAL_MONITOR
group by
INT_ID
)
```

If any are missing, re-introduce them from YC_20170905_DINTMON (not shown).

Script: G_10_15_09_SCR_INT_ID_SRI.sql

G.10.16 Screen information – D_INTERVAL

We can now create the actual intervals for M_D_INTERVAL using the information in M_D_LOCATION and M_D_INTERVAL_MONITOR. Note that the LOC_ORIGINAL_NAME (i.e. WELL_ID) is being assigned to INT_NAME and the BORE_HOLE_ID (from YC_20160531_BH_ID) to INT_NAME_ALT2.

```
select
dim.INT_ID
,ycb.BORE_HOLE_ID as LOC_ID
,dloc.LOC_ORIGINAL_NAME as INT_NAME
,ycb.BORE_HOLE_ID as INT_NAME_ALT2
,dim.tmp_INT_TYPE_CODE as INT_TYPE_CODE
,case
when dloc.LOC_START_DATE is not null then dloc.LOC_START_DATE
else cast('1867-07-01' as datetime)
end as INT_START_DATE
,cast(1 as int) as INT_CONFIDENTIALITY_CODE
,cast(1 as int) as INT_ACTIVE
,cast(519 as int) as [DATA_ID]
into MOE_20170905.dbo.M_D_INTERVAL
from
(
select
dim.INT_ID
,dim.INT_ID as LOC_ID
,dim.tmp_INT_TYPE_CODE
from
MOE_20170905.dbo.M_D_INTERVAL_MONITOR as dim
group by
dim.INT_ID,dim.tmp_INT_TYPE_CODE
) as dim
inner join MOE_20170905.dbo.YC_20170905_BH_ID as ycb
on dim.LOC_ID=ycb.BORE_HOLE_ID
inner join MOE_20170905.dbo.M_D_LOCATION as dloc
on ycb.BORE_HOLE_ID=dloc.LOC_ID
```

Script: G_10_16_01_M_D_INTERVAL.sql

Make sure that there isn't an error here regarding interval identifiers – there should be no duplicates (check not shown).

G.10.17 Interval reference elevation

We can now build the M_D_INTERVAL_REF_ELEV table using the M_D_INTERVAL and M_D_LOCATION_ELEV tables (we'll use a row count for populating the SYS_RECORD_ID). Note that we're assuming a '0.75' metre stickup (assigned to REF_STICK_UP; previously this was assigned to REF_POINT) for all MOE WWDB wells.

```
SELECT
dint.[INT_ID]
,case
when dint.INT_START_DATE is not null then dint.INT_START_DATE
else cast('1867-07-01' as datetime)
end
as REF_ELEV_START_DATE
,cast('0.75' as varchar(50)) as REF_STICK_UP
```

```

,(delev.LOC_ELEV+0.75) as [REF_ELEV]
,(delev.LOC_ELEV+0.75) as [REF_ELEV_OUOM]
,cast('mas' as varchar(50)) as REF_ELEV_UNIT_OUOM
,cast('ASSIGNED_ELEV + 0.75m' as varchar(255)) as REF_COMMENT
,row_number() over (order by dint.INT_ID) as SYS_RECORD_ID
,convert(int,523) as [DATA_ID]
into MOE_20210119.dbo.M_D_INTERVAL_REF_ELEV
FROM
MOE_20210119.[dbo].[M_D_INTERVAL] as dint
inner join
(
select
dlsh.LOC_ID
,LOC_ELEV
from
MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST as dlsh
where
dlsh.LOC_ELEV_CODE=3
-- Only load the SRTM elev if no MNR elev
union
select
dlsh.LOC_ID
,LOC_ELEV
from
MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST as dlsh
where
dlsh.LOC_ELEV_CODE=5
) as delev
on dint.loc_id=delev.LOC_ID
where
delev.LOC_ELEV is not null

```

Script: G_10_17_01_M_D_INT_REF_ELEV.sql

G.10.18 Interval Temporal Data – Static Water Levels

Static water levels, from the MOE WWDB, are located within the ‘TblPump_Test’ table (as found in the field ‘Static_lev’). We’ll use PIPE_ID to access this information. Note that a check should be made, first, to determine whether multiple static water levels are associated with any particular location (not shown; refer to script itself for details; single static water levels, only, were found in MOE_20170905).

```

select
dint.INT_ID
,cast(0 as int) as RD_TYPE_CODE
,cast(628 as int) as RD_NAME_CODE
,dint.INT_START_DATE as [RD_DATE]
,cast(
case
when moept.LEVELS_UOM like 'ft' then delev.LOC_ELEV-(0.3048*moept.Static_lev)
else delev.LOC_ELEV-moept.Static_lev
end
as float
) as [RD_VALUE]
,cast(6 as int) as UNIT_CODE
,cast('Water Level - Manual - Static' as varchar(255)) as RD_NAME_OUOM
,cast(moept.Static_lev as float) as [RD_VALUE_OUOM]
,cast(moept.LEVELS_UOM as varchar(50)) as [RD_UNIT_OUOM]
,cast(1 as int) as [REC_STATUS_CODE]
,cast(null as varchar(255)) as RD_COMMENT
,cast(523 as int) as DATA_ID
,ROW_NUMBER() over (order by dint.INT_ID) as SYS_RECORD_ID
into MOE_20210119.dbo.M_D_INTERVAL_TEMPORAL_2
from

```

```

MOE_20210119.dbo.TblPipe as moetp
inner join MOE_20210119.dbo.YC_20210119_BH_ID as ycb
on moetp.Bore_Hole_ID=ycb.BORE_HOLE_ID
inner join MOE_20210119.dbo.TblPump_Test as moept
on moetp.PIPE_ID=moept.PIPE_ID
inner join MOE_20210119.dbo.M_D_LOCATION as dloc
on ycb.BORE_HOLE_ID=dloc.LOC_ID
inner join MOE_20210119.dbo.M_D_INTERVAL as dint
on dloc.LOC_ID=dint.LOC_ID
inner join
(
select
dsh.LOC_ID
,LOC_ELEV
from
MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST as dsh
where
dsh.LOC_ELEV_CODE=3
-- Only load the SRTM elev if no MNR elev
union
select
dsh.LOC_ID
,LOC_ELEV
from
MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST as dsh
where
dsh.LOC_ELEV_CODE=5
) as delev
on dloc.LOC_ID=delev.LOC_ID
where
moetp.Static_lev is not null
and delev.LOC_ELEV is not null

```

Script: G_10_18_01_STATIC_WLS.sql

This table (i.e. M_D_INTERVAL_TEMPORAL_2) can be checked so that duplicate INT_IDs are found and removed. In the case of MOE_20160531, three wells had multiple, equivalent static water levels associated with them – test extra rows can now be removed using the varying SYS_RECORD_ID values (not shown; refer to script for details). No duplicates INT_IDs were found in MOE_20170905

G.10.19 Pumping Data

As a first step, we can find those boreholes that are marked as ‘flowing’ (note that we’re not checking whether any other values are present, just if the well has been tagged).

```

select
COUNT(*) as rcount
from
MOE_20170905.dbo.M_D_INTERVAL_MONITOR as dim
where
dim.INT_ID
in
(
select
dint.INT_ID
from
MOE_20170905.dbo.TblPipe as moetp
inner join MOE_20170905.dbo.YC_20170905_BH_ID as ycb
on moetp.Bore_Hole_ID=ycb.BORE_HOLE_ID
inner join MOE_20170905.dbo.TblPump_Test as moept
on moetp.PIPE_ID=moept.PIPE_ID
inner join MOE_20170905.dbo.M_D_LOCATION as dloc

```

```

on ycb.BORE_HOLE_ID=dloc.LOC_ID
inner join MOE_20170905.dbo.M_D_INTERVAL as dint
on dloc.LOC_ID=dint.LOC_ID
where
moept.FLOWING like 'Y'
)

```

If the count returned is larger than zero we can then update those intervals in `M_D_INTERVAL_MONITOR` (note that the 'count(*)' could be replaced with just a '*' to examine the rows returned) with a 'is flowing' status.

```

update MOE_20170905.dbo.M_D_INTERVAL_MONITOR
set
MON_FLOWING=1
where
INT_ID
in
(
select
dint.INT_ID
from
MOE_20170905.dbo.TblPipe as moept
inner join MOE_20170905.dbo.YC_20170905_BH_ID as ycb
on moept.Bore_Hole_ID=ycb.BORE_HOLE_ID
inner join MOE_20170905.dbo.TblPump_Test as moept
on moept.PIPE_ID=moept.PIPE_ID
inner join MOE_20170905.dbo.M_D_LOCATION as dloc
on ycb.BORE_HOLE_ID=dloc.LOC_ID
inner join MOE_20170905.dbo.M_D_INTERVAL as dint
on dloc.LOC_ID=dint.LOC_ID
where
moept.FLOWING like 'Y'
)

```

When selecting these (flowing locations) subsequently, use a 'is not null' check against the `MON_FLOWING` field.

We'll now extract those pump tests that have values in any of the 'Recom_rate' or 'Flowing_rate' columns or if there are step records (i.e. a stepped pump test) in 'TblPump_Test_Detail'. We're assuming that all 'GPM' rates are actually 'IGPM'. The `PUMPING_TEST_METHOD` is being converted from MOE codes to YPDT-CAMC codes as well.

```

select
moept.PUMP_TEST_ID
,dint.INT_ID
,dint.INT_START_DATE as [PUMPTEST_DATE]
,cast(ycb.WELL_ID as varchar(20)) as PUMPTEST_NAME
,cast(
case
when moept.LEVELS_UOM='ft' then moept.Recom_depth*0.3048
else moept.Recom_depth
end as float) as [REC_PUMP_DEPTH_METERS]
,cast(
case
when moept.RATE_UOM='LPM' then moept.Recom_rate/4.55
else moept.Recom_rate
end as float) as [REC_PUMP_RATE_IGPM]
,cast(
case
when moept.RATE_UOM='LPM' then moept.Flowing_rate/4.55
else moept.Flowing_rate

```

```

end as float) as [FLOWING_RATE_IGPM]
.cast(519 as int) as [DATA_ID]
.cast(
case
when moept.PUMPING_TEST_METHOD is null then null
when moept.PUMPING_TEST_METHOD=0 then null
when moept.PUMPING_TEST_METHOD=1 then 1
when moept.PUMPING_TEST_METHOD=2 then 2
when moept.PUMPING_TEST_METHOD=3 then 4
when moept.PUMPING_TEST_METHOD=4 then 8
when moept.PUMPING_TEST_METHOD=5 then 9
else 10
end as int) as [PUMPTEST_METHOD_CODE]
.cast(1 as int) as [PUMPTEST_TYPE_CODE] -- this is a constant rate indicator
.cast(
case
when moept.WATER_STATE_AFTER_TEST is null then 0
else moept.WATER_STATE_AFTER_TEST
end as int) as [WATER_CLARITY_CODE]
.ROW_NUMBER() over (order by dint.INT_ID) as rnum
into MOE_20170905.dbo.M_D_PUMPTEST
from
MOE_20170905.dbo.TblPipe as moept
inner join MOE_20170905.dbo.YC_20170905_BH_ID as ycb
on moept.Bore_Hole_ID=ycb.BORE_HOLE_ID
inner join MOE_20170905.dbo.TblPump_Test as moept
on moept.PIPE_ID=moept.PIPE_ID
inner join MOE_20170905.dbo.M_D_LOCATION as dloc
on ycb.BORE_HOLE_ID=dloc.LOC_ID
inner join MOE_20170905.dbo.M_D_INTERVAL as dint
on dloc.LOC_ID=dint.LOC_ID
where
(
moept.Recom_depth is not null
or moept.Recom_rate is not null
or moept.Flowing_rate is not null
or moept.PUMP_TEST_ID in (select PUMP_TEST_ID from MOE_20170905.dbo.TblPump_Test_Detail)
)

```

We can now modify (as an attempt to correct perceived errors) the FLOWING_RATE_IGPM field/values which should only be populated when the borehole is actually flowing. The guidelines for doing so are:

- If MON_FLOWING (populated in M_D_INTERVAL_MONITOR) is populated then the well is assumed to be actually flowing; the value is not modified (this was determined in a previous statement)
- If MON_FLOWING is not tagged and the REC_PUMP_RATE_IGPM is equal to the FLOWING_RATE_IGPM then the well is not flowing; NULL the latter field
- If MON_FLOWING is not tagged and no REC_PUMP_RATE_IGPM but there is a FLOWING_RATE_IGPM, the well is flowing; tag MON_FLOWING
- If MON_FLOWING is not tagged and FLOWING_RATE_IGPM is less than REC_PUMP_RATE_IGPM, the well is flowing; tag MON_FLOWING
- If MON_FLOWING is not tagged and FLOWING_RATE_IGPM is more than REC_PUMP_RATE_IGPM, this is an error and the well is not flowing; null the former field

Check and clear FLOWING_RATE_IGPM if it equals REC_PUMP_RATE_IGPM.

```
update MOE_20170905.dbo.M_D_PUMPTEST
```

```

set
FLOWING_RATE_IGPM=null
from
MOE_20170905.dbo.M_D_PUMPTEST as dpump
where
dpump.INT_ID
not in
(
select
dim.INT_ID
from
MOE_20170905.dbo.M_D_INTERVAL_MONITOR as dim
where
dim.MON_FLOWING is not null
)
and dpump.REC_PUMP_RATE_IGPM=dpump.FLOWING_RATE_IGPM

```

Check and assign a tag to MON_FLOWING (in FM_D_INTERVAL_MONITOR) if a value is present in FLOWING_RATE_IGPM but no REC_PUMP_RATE_IGPM is given.

```

update MOE_20170905.dbo.M_D_INTERVAL_MONITOR
set
MON_FLOWING=1
from
MOE_20170905.dbo.M_D_INTERVAL_MONITOR as dim
where
dim.INT_ID
in
(
select
dpump.INT_ID
from
MOE_20170905.dbo.M_D_PUMPTEST as dpump
where
dpump.INT_ID
in
(
select
dim.INT_ID
from
MOE_20170905.dbo.M_D_INTERVAL_MONITOR as dim
where
dim.MON_FLOWING is null
)
and dpump.REC_PUMP_RATE_IGPM is null
and dpump.FLOWING_RATE_IGPM is not null
)

```

Check and assign a tag to MON_FLOWING if both REC_PUMP_RATE_IGPM and FLOWING_RATE_IGPM are present and the latter is less than the former.

```

update MOE_20170905.dbo.M_D_INTERVAL_MONITOR
set
MON_FLOWING=1
from
MOE_20170905.dbo.M_D_INTERVAL_MONITOR as dim
where
dim.INT_ID
in
(
select
dpump.INT_ID
from
MOE_20170905.dbo.M_D_PUMPTEST as dpump

```

```

where
dpump.INT_ID
in
(
select
dim.INT_ID
from
MOE_20170905.dbo.M_D_INTERVAL_MONITOR as dim
where
dim.MON_FLOWING is null
)
and dpump.FLOWING_RATE_IGPM<dpump.REC_PUMP_RATE_IGPM
)

```

If FLOWING_RATE_IGPM is greater than REC_PUMP_RATE_IGPM and MON_FLOWING is not tagged, NULL the FLOWING_RATE_IGPM field (this is considered an error).

```

update MOE_20170905.dbo.M_D_PUMPTEST
set
FLOWING_RATE_IGPM=null
from
MOE_20170905.dbo.M_D_PUMPTEST as dpump
where
dpump.INT_ID
in
(
select
dpump.INT_ID
from
MOE_20170905.dbo.M_D_PUMPTEST as dpump
where
dpump.INT_ID
in
(
select
dim.INT_ID
from
MOE_20170905.dbo.M_D_INTERVAL_MONITOR as dim
where
dim.MON_FLOWING is null
)
)
and dpump.FLOWING_RATE_IGPM>dpump.REC_PUMP_RATE_IGPM
)

```

Script: G_10_19_01_PUMPING.sql

Before proceeding with a determination of information to be included from 'TblPump_Test_Detail' into M_D_PUMPTEST_STEP (i.e. the actual 'stepped' data), we can use PUMP_TEST_ID as a pointer for determining which intervals need to be assigned together (i.e. joined into a single INT_ID).

As of 20131104, we are not extracting the 'single step' held within the 'TblPump_Test' table. It was found that this duplicated the information held within the 'TblPump_Test_Detail'. (Instructions for doing so, however, are still found in the preceding script file under 'IMPORTANT 20131104: NO LONGER USED'.)

A single PUMP_TEST_ID should be assigned for any valid interval – when multiple INT_IDs are associated with a PUMP_TEST_ID it provides indication that the intervals (i.e. screens) are successive. We'll use this to create a reference table.

```
select
t3.pump_test_id
,t3.int_id_new
,dpump.int_id as int_id_old
into [MOE_20160531].dbo.YC_20160531_PUMP_INT
from
(
select
dpump.pump_test_id
,max(dpump.int_id) as int_id_new
from
(
select
t1.pump_test_id
,t1.rcount
from
(
select
pump_test_id
,COUNT(*) as rcount
from
[MOE_20160531].[dbo].[M_D_PUMPTEST]
group by
pump_test_id
) as t1
where
t1.rcount>1
) as t2
inner join
[MOE_20160531].[dbo].[M_D_PUMPTEST] as dpump
on
t2.pump_test_id=dpump.pump_test_id
group by
dpump.pump_test_id
) as t3
inner join
[MOE_20160531].[dbo].[M_D_PUMPTEST] as dpump
on
t3.pump_test_id=dpump.pump_test_id
```

If the table is empty, we don't need to proceed with the remainder of the scripts in this section (MOE_20160531 had 0 records returned; MOE_20170905 also had 0 records returned).

The remainder of this section was unnecessary for the 20160531 and 20170905 MOE WWDB import.

Here, INT_ID_NEW is the INT_ID (of those available for the 'matched' intervals) to be used for reassignment while INT_ID_OLD are all the original INT_IDs that are to be affected. FM_D_INTERVAL_MONITOR can be checked (not shown) to see how these intervals match. Note that depths should not overlap but other details will likely vary.

We can now reassign those intervals in FM_D_INTERVAL_MONITOR to the appropriate INT_ID (remember that SYS_RECORD_ID remains to differentiate the rows).

```

update [MOE_201304].dbo.FM_D_INTERVAL_MONITOR
set
INT_ID=yepi.int_id_new
from
[MOE_201304].dbo.YC_20130923_PUMP_INT as yepi
inner join
[mo_201304].dbo.FM_D_INTERVAL_MONITOR as dim
on
yepi.int_id_old=dim.INT_ID

```

These intervals can now be removed from each of FM_D_INTERVAL, FM_D_INTERVAL_REF_ELEV, D_INTERVAL_TEMP_2 and D_PUMPTEST (only an example, using FM_D_INTERVAL, is shown; refer to the script file for full details).

```

select
dint.INT_ID
from
[MOE_201304].dbo.FM_D_INTERVAL as dint
where
dint.INT_ID
in
(
select
yepi.int_id_old
from
[MOE_201304].dbo.yc_20130923_pump_int as yepi
where
yepi.int_id_old
not in
(
select int_id_new from [MOE_201304].dbo.yc_20130923_pump_int
)
)

```

```

delete from [MOE_201304].dbo.FM_D_INTERVAL
where
INT_ID
in
(
select
yepi.int_id_old
from
[MOE_201304].dbo.yc_20130923_pump_int as yepi
where
yepi.int_id_old
not in
(
select int_id_new from [MOE_201304].dbo.yc_20130923_pump_int
)
)

```

Script: G_10_19_02_PUMP_INT_CORRECT.sql

Some of the pump tests found in the M_D_PUMPTEST table have NULL dates (which are dependent upon a valid date for the MOE well). In order to facilitate the use of the pumptest data, we'll assign a 'tag' date – a holder that allows users to evaluate the data which is dependent upon the time within the pumptest at which a level was measured/recorded (rather than an absolute date). The 'tag' date chosen is '1867-07-01'; note that these dates, again, only apply to the pump test and the resultant data (not to the location as a whole; this would affect D_PUMPTEST, D_PUMPTEST_STEP and

D_INTERVAL_TEMPORAL 2). A check (after modification) would be a search for dates less than (i.e. before) '1868-01-01'. The scripts themselves (for this entire process) are not shown here.

We can now check whether any step information is to be found in 'TblPump_Test_Detail' for the particular locations we are extracting. Duration here is in minutes, so we're doing the conversion using the DATEADD() function (note that DATEADD() does not allow non-integer values; all times must be specified using whole minutes/seconds). All times start at midnight (i.e. 00:00; <hours:minutes>) by default.

We'll need to evaluate this in two steps. Here we'll extract all durations except the first (as we need to reference the two sets of information at the same time – namely the current and the previous; this removes the first duration from examination).

```

select
curr.Pump_Test_id
,cast(moept.Pumping_rate as float) as PUMP_RATE
,cast(moept.RATE_UOM as varchar(50)) as PUMP_RATE_UNITS
,cast(moept.Pumping_rate as float) as PUMP_RATE_OUOM
,cast(moept.RATE_UOM as varchar(50)) as PUMP_RATE_UNITS_OUOM
,dateadd(minute,curr.TestDuration,dpump.PUMPTEST_DATE) as [PUMP_START]
,dateadd(minute,prev.TestDuration,dpump.PUMPTEST_DATE) as [PUMP_END]
,cast(519 as int) as DATA_ID
,null as [SYS_RECORD_ID]
,null as [rnum]
,prev.TestType as [testtype]
,prev.TestLevel as [testlevel]
,prev.TESTLEVEL_UOM as [testlevel_uom]
into MOE_20170905.dbo.YC_20170905_PUMP_STEP
from
(
    select
    moeptd.Pump_Test_id
    ,moeptd.TestDuration
    ,ROW_NUMBER() over (order by moeptd.Pump_Test_id,moeptd.TestDuration) as rnum
    from
    MOE_20170905.dbo.TblPump_Test_Detail as moeptd
    where
    moeptd.TestType='D'
) as [curr]
inner join
(
    select
    moeptd.Pump_Test_id
    ,moeptd.TestDuration
    ,moeptd.TestLevel
    ,moeptd.TestType
    ,moeptd.TESTLEVEL_UOM
    ,ROW_NUMBER() over (order by moeptd.Pump_Test_id,moeptd.TestDuration) as rnum
    from
    MOE_20170905.dbo.TblPump_Test_Detail as moeptd
    where
    moeptd.TestType='D'
) as [prev]
on
curr.Pump_Test_id=prev.Pump_Test_id and curr.rnum=(prev.rnum-1)
inner join MOE_20170905.dbo.TblPump_Test as moept
on curr.Pump_Test_id=moept.PUMP_TEST_ID
inner join MOE_20170905.dbo.M_D_PUMPTEST as dpump
on curr.Pump_Test_id=dpump.pump_test_id
inner join MOE_20170905.dbo.TblPipe as moep
on moept.PIPE_ID=moep.PIPE_ID
inner join MOE_20170905.dbo.YC_20170905_BH_ID as ycb

```

```

on moep.Bore_Hole_ID=ycb.BORE_HOLE_ID
inner join MOE_20170905.dbo.M_D_LOCATION as dloc
on ycb.BORE_HOLE_ID=dloc.LOC_ID
order by
curr.Pump_Test_id,PUMP_START

```

We can now create the first duration by extracting the first record from each 'Pump_Test_ID'.

```

insert into MOE_20170905.dbo.YC_20170905_PUMP_STEP
(
[PUMP_TEST_ID]
,[PUMP_RATE]
,[PUMP_RATE_UNITS]
,[PUMP_RATE_OUOM]
,[PUMP_RATE_UNITS_OUOM]
,[PUMP_START]
,[PUMP_END]
,[DATA_ID]
,[SYS_RECORD_ID]
,[rnum]
,[testtype]
,[testlevel]
,[testlevel_uom]
)
select
curr.Pump_Test_id
,cast(moept.Pumping_rate as float) as PUMP_RATE
,cast(moept.RATE_UOM as varchar(50)) as PUMP_RATE_UNITS
,cast(moept.Pumping_rate as float) as PUMP_RATE_OUOM
,cast(moept.RATE_UOM as varchar(50)) as PUMP_RATE_UNITS_OUOM
,dateadd(minute,0,dpump.PUMPTEST_DATE) as [PUMP_START]
,dateadd(minute,curr.TestDuration,dpump.PUMPTEST_DATE) as [PUMP_END]
,cast(519 as int) as DATA_ID
,cast(null as int) as [SYS_RECORD_ID]
,cast(null as int) as [rnum]
,moepd.TestType as [testtype]
,moepd.TestLevel as [testlevel]
,moepd.TESTLEVEL_UOM as [testlevel_uom]
from
(
select
moepd.Pump_Test_id
,min(moepd.TestDuration) as TestDuration
from
MOE_20170905.dbo.TblPump_Test_Detail as moepd
where
moepd.TestType='D'
group by
moepd.Pump_Test_id
) as curr
inner join MOE_20170905.dbo.TblPump_Test as moept
on curr.Pump_Test_id=moept.PUMP_TEST_ID
inner join MOE_20170905.dbo.TblPump_Test_Detail as moepd
on curr.Pump_Test_id=moepd.Pump_Test_id and curr.TestDuration=moepd.TestDuration
inner join MOE_20170905.dbo.M_D_PUMPTEST as dpump
on curr.Pump_Test_id=dpump.pump_test_id
inner join MOE_20170905.dbo.TblPipe as moep
on moept.PIPE_ID=moep.PIPE_ID
inner join MOE_20170905.dbo.YC_20170905_BH_ID as ycb
on moep.Bore_Hole_ID=ycb.BORE_HOLE_ID
inner join MOE_20170905.dbo.M_D_LOCATION as dloc
on ycb.BORE_HOLE_ID=dloc.LOC_ID
where
moepd.TestType='D'
order by
curr.Pump_Test_id,PUMP_START

```

Script: G_10_19_03_PUMP_STEP.sql

So far, we've only been extracting pumping-step data records for drawdown ('D') data – we must also extract the recovery information (i.e. 'R') as well. Here we don't need to access the pumping information other than the final pumping time (adding it to the start of the recovery test). The data is stored in the pump test table as a temporary measure before import into M_D_INTERVAL_TEMPORAL_2. This data will not be imported into the actual (final) D_PUMPTEST_STEP table itself.

```
insert into MOE_20170905.dbo.YC_20170905_PUMP_STEP
(
[PUMP_TEST_ID]
,[PUMP_END]
,[DATA_ID]
,[SYS_RECORD_ID]
,[rnum]
,[testtype]
,[testlevel]
,[testlevel_uom]
)
select
curr.Pump_Test_id
,dateadd(minute,curr.TestDuration,mt.enddate) as [PUMP_END]
,cast(519 as int) as DATA_ID
,null as [SYS_RECORD_ID]
,null as [rnum]
,curr.TestType as [testtype]
,curr.TestLevel as [testlevel]
,curr.TESTLEVEL_UOM as [testlevel_uom]
from
(
select
moeptd.Pump_Test_id
,moeptd.TestDuration
,moeptd.TestType
,moeptd.TestLevel
,moeptd.TESTLEVEL_UOM
from
MOE_20170905.dbo.TblPump_Test_Detail as moeptd
where
moeptd.TestType='R'
) as [curr]
inner join
(
select
ycps.Pump_Test_id
,max(ycps.PUMP_END) as enddate
from
MOE_20170905.dbo.YC_20170905_PUMP_STEP as ycps
where
ycps.TestType='D'
group by
ycps.Pump_Test_id
) as mt
on
curr.Pump_Test_id=mt.Pump_Test_id
inner join MOE_20170905.dbo.TblPump_Test as moept
on curr.Pump_Test_id=moept.PUMP_TEST_ID
inner join MOE_20170905.dbo.M_D_PUMPTEST as dpump
on curr.Pump_Test_id=dpump.pump_test_id
inner join MOE_20170905.dbo.TblPipe as moep
on moept.PIPE_ID=moep.PIPE_ID
inner join MOE_20170905.dbo.YC_20170905_BH_ID as ycb
on moep.Bore_Hole_ID=ycb.BORE_HOLE_ID
```

```

inner join MOE_20170905.dbo.M_D_LOCATION as dloc
on ycb.BORE_HOLE_ID=dloc.LOC_ID
order by
curr.Pump_Test_id,PUMP_END

```

Script: G_10_19_04_PUMP_STEP_R.sql

We can now copy across the appropriate data to the M_D_PUMPTEST_STEP table. Note that we're grouping the pumping rate data (based on PUMP_RATE, PUMP_RATE_UNITS, PUMP_RATE_OUOM and PUMP_RATE_UNITS_OUOM) as only variable pumping rates (and their pumping – time – interval) are to be stored in M_D_PUMPTEST_STEP.

```

select
t1.PUMP_TEST_ID
,t1.PUMP_RATE
,t1.PUMP_RATE_UNITS
,t1.PUMP_RATE_OUOM
,t1.PUMP_RATE_UNITS_OUOM
,t1.PUMP_START
,t1.PUMP_END
,.cast(519 as int) as DATA_ID
,t1.SYS_RECORD_ID
into MOE_20170905.dbo.M_D_PUMPTEST_STEP
from
(
SELECT
[PUMP_TEST_ID]
,[PUMP_RATE]
,[PUMP_RATE_UNITS]
,[PUMP_RATE_OUOM]
,[PUMP_RATE_UNITS_OUOM]
,.min([PUMP_START]) as [PUMP_START]
,.max([PUMP_END]) as [PUMP_END]
,.ROW_NUMBER() over (order by PUMP_TEST_ID) as SYS_RECORD_ID
FROM MOE_20170905.[dbo].[YC_20170905_PUMP_STEP]
where
testtype='D'
group by
Pump_Test_id,PUMP_RATE,PUMP_RATE_UNITS,PUMP_RATE_OUOM,PUMP_RATE_UNITS_OUOM
) as t1

```

Script: G_10_19_05_PUMP_STEP_FINAL.sql

Now we can use YC_20170905_PUMP_STEP (the temporary table) as a basis for importing the temporal pumping data into M_D_INTERVAL_TEMPORAL_2 (all of the records from the former should be added to the latter). Note that we're accessing M_D_INTERVAL_TEMPORAL_2 to determine the total number of records currently present; we'll start a new count (for the SYS_RECORD_ID) from that point.

```

insert into MOE_20210119.dbo.M_D_INTERVAL_TEMPORAL_2
(
INT_ID
,RD_TYPE_CODE
,RD_NAME_CODE
,rd_date
,RD_VALUE
,UNIT_CODE
,RD_NAME_OUOM
,RD_VALUE_OUOM
,RD_UNIT_OUOM

```

```

.REC_STATUS_CODE
.RD_COMMENT
.DATA_ID
.SYS_RECORD_ID
)
select
dp.INT_ID
.case
when dps.testtype='D' then cast(65 as int) -- i.e. moe pumping level
else cast(64 as int) -- i.e. moe recovery level
end as [RD_TYPE_CODE]
.cast(70899 as int) as [RD_NAME_CODE] -- i.e. Water Level - Manual - Other
.dps.PUMP_END as [RD_DATE]
.case
when dps.testlevel_uom='m' then delev.loc_elev-dps.testlevel
else delev.loc_elev-(dps.testlevel*0.3048)
end as [RD_VALUE]
.cast(6 as int) as [UNIT_CODE]
.cast('Water Level - Manual - Other' as varchar(100)) as RD_NAME_OUOM
.cast(dps.testlevel as float) as RD_VALUE_OUOM
.cast(dps.testlevel_uom as varchar(50)) as RD_UNIT_OUOM
.cast(1 as int) as REC_STATUS_CODE
.case
when dps.testtype='D' then cast('Pumping - Drawdown' as varchar(255))
else cast('Pumping - Recovery' as varchar(255))
end as [RD_COMMENT]
.dps.DATA_ID as [DATA_ID]
--***** add the number to the following row
,3779 + ROW_NUMBER() over (order by dps.PUMP_END) as SYS_RECORD_ID
from
MOE_20210119.[dbo].YC_20210119_PUMP_STEP as dps
inner join MOE_20210119.dbo.M_D_PUMPTTEST as dp
on dps.PUMP_TEST_ID=dp.PUMP_TEST_ID
inner join MOE_20210119.dbo.M_D_INTERVAL as dint
on dp.INT_ID=dint.INT_ID
inner join
(
select
dlsh.LOC_ID
,LOC_ELEV
from
MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST as dlsh
where
dlsh.LOC_ELEV_CODE=3
-- Only load the SRTM elev if no MNR elev
union
select
dlsh.LOC_ID
,LOC_ELEV
from
MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST as dlsh
where
dlsh.LOC_ELEV_CODE=5
) as delev
on dint.LOC_ID=delev.LOC_ID
where
dps.testtype is not null
and delev.LOC_ELEV is not null
order by
dp.INT_ID,RD_DATE

```

Script: G_10_19_06_PUMP_TEMP_2.sql

G.10.20 Population of remaining LOC_MASTER_LOC_ID

For the MOE WWDB, version as of 20160531, the LOC_MASTER_LOC_ID was populated in an earlier step (Section G.10.2). The scripts used in the previous import have been maintained but were not used.

G.10.21 Reassignment of all *_ID keys

The M_* tables should now be at the point where they can be (almost) directly added into the master database. The final step is to modify all *_ID keys so that there be no conflicts with existing keys in each of the tables in the master database. A separate (series of) identifier look-up tables will be created for this purpose.

We'll first add the appropriate DATA_ID to D_DATA_SOURCE (in OAK_20160831_MASTER) to hold the identifier for this data.

```
insert into [OAK_20160831_MASTER].[dbo].[D_DATA_SOURCE]
(DATA_ID,DATA_TYPE,DATA_DESCRIPTION,DATA_COMMENT)
values
(519,'Well Data','MOE WWR Database - 20170905','Refer to Appendix G.10 in database manual for methodology')
```

Determine how many LOC_IDs are present - we'll use this number to create a new series of LOC_ID's (check, first, that there are the same number of BH_IDs as LOC_IDs in the M_D_BOREHOLE and M_D_LOCATION tables as we'll use the same lookup table to populate both; not shown here).

```
select
t1.BORE_HOLE_ID
,t2.NEW_ID as new_LOC_ID
,t1.BH_ID
,cast(null as int) as new_BH_ID
,ROW_NUMBER() over (order by t1.BORE_HOLE_ID) as rnum
into MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID
from
(
select
dloc.BORE_HOLE_ID
,dloc.BH_ID
,ROW_NUMBER() over (order by BORE_HOLE_ID) as rnum
from
MOE_20170905.dbo.YC_20170905_BH_ID as dloc
) as t1
inner join
(
select
top 25000
v.NEW_ID
,ROW_NUMBER() over (order by NEW_ID) as rnum
from
OAK_20160831_MASTER.dbo.V_SYS_RANDOM_ID_001 as v
where
v.NEW_ID
not in
(
select LOC_ID from OAK_20160831_MASTER.dbo.D_LOCATION
)
) as t2
on
t1.rnum=t2.rnum
```

These LOC_IDs don't already exist as part of the master database. Now create a series of new BH_IDs.

```
update MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID
set
new_BH_ID=t2.new_BH_ID
from
MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID as y
inner join
(
select
t1.new_BH_ID
,ROW_NUMBER() over (order by t1.new_BH_ID) as rnum
from
(
select
top 25000
v.NEW_ID as new_BH_ID
from
OAK_20160831_MASTER.dbo.V_SYS_RANDOM_ID_001 as v
where
v.NEW_ID
not in
(
select BH_ID from OAK_20160831_MASTER.dbo.D_BOREHOLE
)
) as t1
) as t2
on y.rnum=t2.rnum
```

Script: G_10_21_01_new_LOC_ID.sql

Create a series of new INT_IDs in roughly the same manner as the above.

```
select
t1.INT_ID
,t2.NEW_ID as [new_INT_ID]
,ROW_NUMBER() over (order by t1.INT_ID) as rnum
into MOE_20170905.dbo.YC_20170905_new_INT_ID
from
(
select
dint.INT_ID
,ROW_NUMBER() over (order by INT_ID) as rnum
from
MOE_20170905.dbo.M_D_INTERVAL as dint
) as t1
inner join
(
select
top 25000
v.NEW_ID
,ROW_NUMBER() over (order by NEW_ID) as rnum
from
OAK_20160831_MASTER.dbo.V_SYS_RANDOM_ID_001 as v
where
v.NEW_ID
not in
(
select INT_ID from OAK_20160831_MASTER.dbo.D_INTERVAL
)
) as t2
on
t1.rnum=t2.rnum
```

We'll have to do the same with PUMP_TEST_ID.

```
select
t1.PUMP_TEST_ID
,t2.NEW_ID as [new_PUMP_TEST_ID]
,ROW_NUMBER() over (order by t1.PUMP_TEST_ID) as rnum
into MOE_20170905.dbo.YC_20170905_new_PUMP_TEST_ID
from
(
select
dpump.PUMP_TEST_ID
,ROW_NUMBER() over (order by PUMP_TEST_ID) as rnum
from
MOE_20170905.dbo.M_D_PUMPTEST as dpump
) as t1
inner join
(
select
top 5000
v.NEW_ID
,ROW_NUMBER() over (order by NEW_ID) as rnum
from
OAK_20160831_MASTER.dbo.V_SYS_RANDOM_ID_001 as v
where
v.NEW_ID
not in
(
select PUMP_TEST_ID from OAK_20160831_MASTER.dbo.D_PUMPTEST
)
) as t2
on
t1.rnum=t2.rnum
```

As of the MOE 20200721 database, the D_LOCATION_ELEV and D_LOCATION_ELEV_HIST functionality (as well as D_LOCATION_COORD_HIST) have been replaced by D_LOCATION_SPATIAL and D_LOCATION_SPATIAL_HIST. These will be populated (along with their associated identifiers) in a subsequent step.

Script: G_10_21_02_new_INT_ID.sql

The remainder of the identifiers only occur within single tables and, as such, no relationship tables need to be built (i.e. SYS_RECORD_ID and similar). We can now update the identifiers within the M_* tables in preparation for final import.

Update M_D_BOREHOLE.

```
update MOE_20170905.dbo.M_D_BOREHOLE
set
LOC_ID=y.new_LOC_ID
,BH_ID=y.new_BH_ID
from
MOE_20170905.dbo.M_D_BOREHOLE as dbore
inner join MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID as y
on dbore.LOC_ID=y.BORE_HOLE_ID
```

Update M_D_BOREHOLE_CONSTRUCTION.

```
update MOE_20170905.dbo.M_D_BOREHOLE_CONSTRUCTION
set
BH_ID=y.new_BH_ID
```

```

from
MOE_20170905.dbo.M_D_BOREHOLE_CONSTRUCTION as dbc
inner join MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID as y
on dbc.BH_ID=y.BORE_HOLE_ID

```

Also, we'll need to update the SYS_RECORD_ID for M_D_BOREHOLE_CONSTRUCTION (don't forget to check how many records there are).

```

update MOE_20170905.dbo.M_D_BOREHOLE_CONSTRUCTION
set
SYS_RECORD_ID=t2.new_SRI
from
MOE_20170905.dbo.M_D_BOREHOLE_CONSTRUCTION as b
inner join
(
select
t.new_SRI
,ROW_NUMBER() over (order by t.new_SRI) as SYS_RECORD_ID
from
(
select
top 50000
v.NEW_ID as new_SRI
from
OAK_20160831_MASTER.dbo.V_SYS_RANDOM_ID_001 as v
where
v.NEW_ID
not in
(
select SYS_RECORD_ID from OAK_20160831_MASTER.dbo.D_BOREHOLE_CONSTRUCTION
)
) as t
) as t2
on b.SYS_RECORD_ID=t2.SYS_RECORD_ID

```

Update M_D_GEOLOGY_FEATURE.

```

update MOE_20170905.dbo.M_D_GEOLOGY_FEATURE
set
LOC_ID=ycl.new_LOC_ID
from
MOE_20170905.dbo.M_D_GEOLOGY_FEATURE as dgf
inner join MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID as ycl
on dgf.LOC_ID=ycl.BORE_HOLE_ID

```

As well as the SYS_RECORD_ID in M_D_GEOLOGY_FEATURE.

```

update MOE_20170905.dbo.M_D_GEOLOGY_FEATURE
set
SYS_RECORD_ID=t2.NEW_ID
from
MOE_20170905.dbo.M_D_GEOLOGY_FEATURE as dgf
inner join
(
select
dbc.SYS_RECORD_ID
,ROW_NUMBER() over (order by SYS_RECORD_ID) as rnum
from
MOE_20170905.dbo.M_D_GEOLOGY_FEATURE as dbc
) as t1
on
dgf.SYS_RECORD_ID=t1.SYS_RECORD_ID
inner join

```

```

(
select
top 20000
vr.NEW_ID
,ROW_NUMBER() over (order by NEW_ID) as rnum
from
OAK_20160831_MASTER.dbo.V_SYS_RANDOM_ID_001 as vr
where
vr.NEW_ID
not in
(
select FEATURE_ID from [OAK_20160831_MASTER].dbo.D_GEOLOGY_FEATURE
)
) as t2
on
t1.rnum=t2.rnum

```

Update M_D_GEOLOGY_LAYER.

```

update MOE_20170905.dbo.M_D_GEOLOGY_LAYER
set
LOC_ID=ycl.new_LOC_ID
from
MOE_20170905.dbo.M_D_GEOLOGY_LAYER as dgl
inner join MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID as ycl
on dgl.LOC_ID=ycl.BORE_HOLE_ID

```

Also modify the SYS_RECORD_ID in M_D_GEOLOGY_LAYER.

```

update MOE_20170905.dbo.M_D_GEOLOGY_LAYER
set
SYS_RECORD_ID=t2.NEW_ID
from
MOE_20170905.dbo.M_D_GEOLOGY_LAYER as dgl
inner join
(
select
dbc.SYS_RECORD_ID
,ROW_NUMBER() over (order by SYS_RECORD_ID) as rnum
from
MOE_20170905.dbo.M_D_GEOLOGY_LAYER as dbc
) as t1
on
dgl.SYS_RECORD_ID=t1.SYS_RECORD_ID
inner join
(
select
top 40000
vr.NEW_ID
,ROW_NUMBER() over (order by NEW_ID) as rnum
from
OAK_20160831_MASTER.dbo.V_SYS_RANDOM_ID_001 as vr
where
vr.NEW_ID
not in
(
select GEOL_ID from OAK_20160831_MASTER.dbo.D_GEOLOGY_LAYER
)
) as t2
on
t1.rnum=t2.rnum

```

Update M_D_LOCATION.

```

update MOE_20170905.dbo.M_D_LOCATION

```

```

set
LOC_ID=ycl.new_LOC_ID
from
MOE_20170905.dbo.M_D_LOCATION as dloc
inner join MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID as ycl
on dloc.LOC_ID=ycl.BORE_HOLE_ID

```

We also need to update LOC_MASTER_LOC_ID in M_D_LOCATION.

```

update MOE_20170905.dbo.M_D_LOCATION
set
LOC_MASTER_LOC_ID=ycl.new_LOC_ID
from
MOE_20170905.dbo.M_D_LOCATION as dloc
inner join MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID as ycl
on dloc.LOC_MASTER_LOC_ID=ycl.BORE_HOLE_ID

```

Update M_D_LOCATION_ALIAS (a non-null value to SYS_RECORD_ID to fulfill constraint requirements).

```

update MOE_20170905.dbo.M_D_LOCATION_ALIAS
set
LOC_ID=ycl.new_LOC_ID
,SYS_RECORD_ID=ycl.new_LOC_ID
from
MOE_20170905.dbo.M_D_LOCATION_ALIAS as dla
inner join MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID as ycl
on dla.LOC_ID=ycl.BORE_HOLE_ID

```

The update of D_LOCATION_ELEV and D_LOCATION_ELEV_HIST are no longer necessary – their content is now found in D_LOCATION_SPATIAL and D_LOCATION_SPATIAL_HIST (as of MOE database version 20200721). The latter table can now be updated.

```

update MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST
set
LOC_ID=ycl.new_LOC_ID
from
MOE_20210119.dbo.M_D_LOCATION_SPATIAL_HIST as dlsh
inner join MOE_20210119.dbo.YC_20210119_new_LOC_ID_BH_ID as ycl
on dlsh.LOC_ID=ycl.BORE_HOLE_ID

```

Update M_D_LOCATION_PURPOSE.

```

update MOE_20170905.dbo.M_D_LOCATION_PURPOSE
set
LOC_ID=ycl.new_LOC_ID
from
MOE_20170905.dbo.M_D_LOCATION_PURPOSE as dpurp
inner join MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID as ycl
on dpurp.LOC_ID=ycl.BORE_HOLE_ID

```

Update M_D_LOCATION_QA.

```

update MOE_20170905.dbo.M_D_LOCATION_QA
set
LOC_ID=ycl.new_LOC_ID
from
MOE_20170905.dbo.M_D_LOCATION_QA as dlq
inner join MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID as ycl

```

on dlq.LOC_ID=ycl.BORE_HOLE_ID

Update M_D_INTERVAL. We need to modify the LOC_ID first.

```
update MOE_20170905.dbo.M_D_INTERVAL
set
LOC_ID=ycl.new_LOC_ID
from
MOE_20170905.dbo.M_D_INTERVAL as dint
inner join MOE_20170905.dbo.YC_20170905_new_LOC_ID_BH_ID as ycl
on dint.LOC_ID=ycl.BORE_HOLE_IDD
```

Then the INT_ID.

```
update MOE_20170905.dbo.M_D_INTERVAL
set
INT_ID=yqi.new_INT_ID
from
MOE_20170905.dbo.M_D_INTERVAL as dint
inner join MOE_20170905.dbo.YC_20170905_new_INT_ID as yqi
on dint.INT_ID=yqi.INT_ID
```

Update FM_D_INTERVAL_MONITOR (remember that there may be more than one row of information tied to a particular INT_ID).

```
update MOE_20170905.dbo.M_D_INTERVAL_MONITOR
set
INT_ID=yqi.new_INT_ID
from
MOE_20170905.dbo.M_D_INTERVAL_MONITOR as dmon
inner join MOE_20170905.dbo.YC_20170905_new_INT_ID as yqi
on dmon.INT_ID=yqi.INT_ID
```

Also modify the SYS_RECORD_ID in M_D_INTERVAL_MONITOR.

```
update MOE_20170905.dbo.M_D_INTERVAL_MONITOR
set
SYS_RECORD_ID=t2.new_SRI
from
MOE_20170905.dbo.M_D_INTERVAL_MONITOR as dim
inner join
(
select
t1.new_SRI
,ROW_NUMBER() over (order by t1.new_SRI) as SYS_RECORD_ID
from
(
select
top 20000
vr.NEW_ID as new_SRI
from
OAK_20160831_MASTER.dbo.V_SYS_RANDOM_ID_001 as vr
where
vr.NEW_ID
not in
(
select MON_ID from OAK_20160831_MASTER.dbo.D_INTERVAL_MONITOR
)
) as t1
) as t2
on dim.SYS_RECORD_ID=t2.SYS_RECORD_ID
```

Update M_D_INTERVAL_REF_ELEV.

```
update MOE_20170905.dbo.M_D_INTERVAL_REF_ELEV
set
INT_ID=yqi.new_INT_ID
from
MOE_20170905.dbo.M_D_INTERVAL_REF_ELEV as dref
inner join MOE_20170905.dbo.YC_20170905_new_INT_ID as yqi
on dref.INT_ID=yqi.INT_ID
```

Also modify the SYS_RECORD_ID in M_D_INTERVAL_REF_ELEV.

```
update MOE_20170905.dbo.M_D_INTERVAL_REF_ELEV
set
SYS_RECORD_ID=t2.new_SRI
from
MOE_20170905.dbo.M_D_INTERVAL_REF_ELEV as dim
inner join
(
select
t1.new_SRI
,ROW_NUMBER() over (order by t1.new_SRI) as SYS_RECORD_ID
from
(
select
top 20000
vr.NEW_ID as new_SRI
from
OAK_20160831_MASTER.dbo.V_SYS_RANDOM_ID_001 as vr
where
vr.NEW_ID
not in
(
select SYS_RECORD_ID from OAK_20160831_MASTER.dbo.D_INTERVAL_REF_ELEV
)
) as t1
) as t2
on dim.SYS_RECORD_ID=t2.SYS_RECORD_ID
```

Update M_D_INTERVAL_TEMPORAL_2.

```
update MOE_20170905.dbo.M_D_INTERVAL_TEMPORAL_2
set
INT_ID=yqi.new_INT_ID
from
MOE_20170905.dbo.M_D_INTERVAL_TEMPORAL_2 as dit2
inner join MOE_20170905.dbo.YC_20170905_new_INT_ID as yqi
on dit2.INT_ID=yqi.INT_ID
```

Also modify the SYS_RECORD_ID in M_D_INTERVAL_TEMPORAL_2.

```
update MOE_20170905.dbo.M_D_INTERVAL_TEMPORAL_2
set
SYS_RECORD_ID=t2.new_SRI
from
MOE_20170905.dbo.M_D_INTERVAL_TEMPORAL_2 as dim
inner join
(
select
t1.new_SRI
,ROW_NUMBER() over (order by t1.new_SRI) as SYS_RECORD_ID
from
(
select
```

```

top 70000
vr.NEW_ID as new_SRI
from
OAK_20160831_MASTER.dbo.V_SYS_RANDOM_ID_001 as vr
where
vr.NEW_ID
not in
(
select SYS_RECORD_ID from OAK_20160831_MASTER.dbo.D_INTERVAL_TEMPORAL_2
)
) as t1
) as t2
on dim.SYS_RECORD_ID=t2.SYS_RECORD_ID

```

Update M_D_PUMPTEST. We'll need to first modify the INT_ID.

```

update MOE_20170905.dbo.M_D_PUMPTEST
set
INT_ID=yqi.new_INT_ID
from
MOE_20170905.dbo.M_D_PUMPTEST as dpump
inner join MOE_20170905.dbo.YC_20170905_new_INT_ID as yqi
on dpump.INT_ID=yqi.INT_ID

```

And then modify the PUMP_TEST_ID.

```

update [MOE_20160531].dbo.M_D_PUMPTEST
set
PUMP_TEST_ID=yqp.new_PUMP_TEST_ID
from
[MOE_20160531].dbo.M_D_PUMPTEST as dpump
inner join [MOE_20160531].dbo.YC_20160531_new_PUMP_TEST_ID as yqp
on dpump.PUMP_TEST_ID=yqp.PUMP_TEST_ID

```

Update FM_D_PUMPTEST_STEP. We'll need to first modify the PUMP_TEST_ID.

```

update MOE_20170905.dbo.M_D_PUMPTEST_STEP
set
PUMP_TEST_ID=yqp.new_PUMP_TEST_ID
from
MOE_20170905.dbo.M_D_PUMPTEST_STEP as dps
inner join MOE_20170905.dbo.YC_20170905_new_PUMP_TEST_ID as yqp
on dps.PUMP_TEST_ID=yqp.PUMP_TEST_ID

```

And then modify the SYS_RECORD_ID.

```

update MOE_20170905.dbo.M_D_PUMPTEST_STEP
set
SYS_RECORD_ID=t2.new_SRI
from
MOE_20170905.dbo.M_D_PUMPTEST_STEP as dps
inner join
(
select
t1.new_SRI
,ROW_NUMBER() over (order by t1.new_SRI) as SYS_RECORD_ID
from
(
select
top 4000
vr.NEW_ID as new_SRI
from
OAK_20160831_MASTER.dbo.V_SYS_RANDOM_ID_001 as vr
where

```

```

vr.NEW_ID
not in
(
select SYS_RECORD_ID from OAK_20160831_MASTER.dbo.D_PUMPTEST_STEP
)
) as t1
) as t2
on dps.SYS_RECORD_ID=t2.SYS_RECORD_ID

```

Script: G_10_21_03_ASSIGN_IDS.sql

This completes the identifier update for the MOE WWDB database. These tables can be directly inserted into the master database. However, a few additional checks should be made (see following).

G.10.22 Additional Checks

Additional checks are made to either correct or update various fields from the tables. Note that for modification of depths, each of M_D_BOREHOLE, M_D_BOREHOLE_CONSTRUCTION and M_D_INTERVAL_MONITOR may need to be updated if any of the depths are changed.

The checks should be made on a table-by-table basis. Note that the order of import corresponds to the following list of tables.

D_LOCATION

Before incorporation, some basic checks should be performed to avoid errors during insertion. These include

- Fields such as LOC_ADDRESS_INFO1, LOC_CON cannot carry 0-length values (i.e. blank fields; different from NULL fields); these should be converted to NULL values
- LOC_COUNTY_CODE, if unknown (a value of '99' or other invalid value) should be assigned a NULL value
- LOC_TOWNSHIP_CODE, if unknown (a value of '9999' or other invalid value) should be assigned a NULL value
- Remove all the extraneous records (the table could be backed up first); these are the rows with a NULL LOC_COORD_EASTING value

D_BOREHOLE

Check for negative depths – this could result from negative values being specified in the borehole construction table (correct this as well).

No additional checks. However, if the table M_R_BH_DRILLER_CODE is present it must be added (and the BH_DRILLER_CODEs updated, as necessary) before this table (i.e. D_BOREHOLE) is updated with the new locations.

D_BOREHOLE_CONSTRUCTION

Check for non-zero/non-null construction details for top depths (i.e. CON_TOP_OUOM) for estimating depths where no other bottom depths (either through construction of formation details) are available. Mark these for subsequent review of the MOE PDF sheets. These may be changed to '123' ('Assumed Open Hole (Top of info only)') at some point in the future.

Check for invalid diameters (CON_DIAMETER_OUOM; the units are generally 'inch') and adjust as appropriate.

D_GEOLOGY_FEATURE

Check if the feature has a negative depth.

D_GEOLOGY_LAYER

Check the top- versus bottom-depths (i.e. top must be less than bottom; GEOL_TOP_OUOM less than GEOL_BOT_OUOM) as well as any negative values.

Check whether top-depths are null and bottom-depths not null (as well as the reverse).

Check for a material code of '0' (i.e. 'Unknown') in GEOL_MAT1_CODE (in table M_D_GEOLOGY_LAYER) where a GEOL_MAT2_CODE is non-null (and then reassign; i.e. GEOL_MAT1_CODE=GEOL_MAT2_CODE); do not move GEOL_MAT3_CODE to either GEOL_MAT2_CODE or GEOL_MAT1_CODE.

D_LOCATION_ALIAS

No additional checks.

D_LOCATION_PURPOSE

Remove any duplicate purposes (i.e. the combination of primary and secondary purposes). Add a (populated) SYS_RECORD_ID field.

D_LOCATION_QA

No additional checks (note that there is no QA_ELEV_CONFIDENCE_CODE_ORIG in D_LOCATION_QA).

D_INTERVAL

No additional checks (note that this appears to take some time to insert).

D_INTERVAL_MONITOR

Check for MON_TOP_OUOM below MON_BOT_OUOM in M_D_INTERVAL_MONITOR and correct.

Check the bottom against the borehole depth.

Check for MON_TOP_OUOM equal to MON_BOT_OUOM; artificially modify these screen lengths to 0.3m/1ft (modify the top; note that the bottom may have been changed in previous imports). This keeps the BH_BOTTOM_DEPTH consistent.

D_INTERVAL_REF_ELEV

No additional checks.

D_INTERVAL_TEMPORAL_2

No additional checks.

D_PUMPTEST

No additional checks.

D_PUMPTEST_STEP

No additional checks.

Script: G_10_22_01_CHECKS_VAR.sql

G.10.23 Table incorporation

From the previous section (G.10.22), there should be no duplicate identifiers in the tables. Each identifier should be checked if some time elapses between the creation and insertion of the data – the following script can be used as a check of the various * _IDs.

Script: G_10_23_01_CHECK_IDS.sql

Use the following script for importing the transformed MOE WWDB tables into the master database (not shown).

Script: G_10_23_02_INSERT.sql

G.10.24 Subsequent procedures

Once the translated MOE WWDB information has been incorporated as part of the master database, some additional procedures need to be performed in order to update certain other tables.

The M_D_LOCATION_SPATIAL_HIST (which replaces the use of D_LOCATION_COORD_HIST and D_LOCATION_ELEV_HIST) should now be added to D_LOCATION_SPATIAL_HIST (in the master database). These new locations should also be added into D_LOCATION_SPATIAL (using the newly generated SPAT_ID)

```
insert into OAK_20160831_MASTER.dbo.D_LOCATION_SPATIAL
(
LOC_ID
,SPAT_ID
,DATA_ID
,SYS_TEMP1
,SYS_TEMP2
)
select
dlsh.LOC_ID
,dlsh.SPAT_ID
,523 as DATA_ID
,'20210119a' as SYS_TEMP1
,20210119 as SYS_TEMP2
from
(
select
d1.LOC_ID
,d1.SPAT_ID
from
oak_20160831_master.dbo.D_LOCATION_SPATIAL_HIST as d1
where
d1.LOC_ELEV_CODE=3
and d1.loc_coord_data_id= 523
-- Only load the SRTM elev if no MNR elev
union
select
d2.LOC_ID
,d2.SPAT_ID
from
oak_20160831_master.dbo.D_LOCATION_SPATIAL_HIST as d2
where
d2.LOC_ELEV_CODE=5
and d2.loc_coord_data_id=523
) as dlsh
```

Script: G_10_24_01_DLOCSPAT.sql

Other procedures include

- The various elevations and conversion from *_OUOM values need to be undertaken (likely through SiteFX); this may entail changing ‘m’ and ‘ft’ to ‘mbgs’ and ‘fbgs’ for various *_OUOM fields
- Determine whether a bedrock elevation should be included (see G.7; automated)
- Add the new INT_ID’s to the D_INTERVAL_FORMATION_ASSIGNMENT table (see G.19; automated)
- Apply formation assignments (see G.1; automated)
- Include the new locations in the spatial table D_LOCATION_GEOM (automated)
- Modify the views that pull MOE wells using the DATA_ID; these include
 - V_GEN_MOE_WELL
 - V_SYS_MOE_DATA_ID

- Check the *_OUOM coordinates for changes to scientific notation during the conversion (and correct)
- Remove INT_TYPE_CODE '28' (i.e. no valid screen determined) records from the D_INTERVAL_MONITOR table

Some of these procedures are encapsulated in the attached script.

Script: G_10_24_02_PROCEDURES.sql

G.10.25 Decommissioned Wells (Abandonment)

Pertinent to the MOE_2017095 database, a new field – ABANDONMENT_REC - has been added to 'tblWWR' to indicate that the borehole/well is a decommissioning record (rather than a 'new' borehole). As such, a 'Y' (or 'not null') in this field should be used to change the imported MOE boreholes from a LOC_TYPE_CODE of '1' (i.e. 'Well or Borehole') to '27' (i.e. 'Decommissioned Well'). The code for doing so takes the following form.

```
update D_LOCATION
set
LOC_TYPE_CODE=27
from
OAK_20160831_MASTER.dbo.D_LOCATION as d
inner join OAK_20160831_MASTER.dbo.V_SYS_MOE_LOCATIONS as v
on d.loc_id=v.loc_id
inner join MOE_20170905.dbo.tblWWR as m
on v.MOE_WELL_ID=cast(m.WELL_ID as int)
where
m.ABANDONMENT_REC is not null
and d.LOC_TYPE_CODE=1
```

As a check, the LOC_STATUS_CODE of these locations could be changed to a value of '22' (i.e. 'MOE Abandonment Record') leaving the LOC_TYPE_CODE as a '1' (i.e. a 'Well or Borehole'). This would allow the locations to be included in cross-section and evaluated with regard to the surrounding geology. Note that the LOC_STATUS_CODE can be reverted to the original value from the MOE database as necessary. This was done for the MOE 20200721 database (instead of applying a LOC_TYPE_CODE of '27').

Script: G_10_25_01_DECOM.sql

In the future, this methodology could be incorporated at an earlier stage in the MOE WWDB import procedure. When these scripts were run for the first time (i.e. for MOE_20170905) the decommissioning tag would have also applied to boreholes from an earlier MOE WWDB import.

G.10.26 Update of MOE WWDB Coordinates

In the case that the MOE has updated the coordinates of previously loaded boreholes, a check should be made and any new coordinates added to the D_LOCATION_COORD_HIST table. Refer to Appendix G.26 for additional details.

G.11 Correction of D_GEOLOGY_LAYER – Missing Depths or Units

Tables

- D_GEOLOGY_LAYER
- D_BOREHOLE_CONSTRUCTION
- D_BOREHOLE

Estimated Recurrence Time: As necessary

When the original units of measure are missing, units are assigned based upon: presence in succeeding layers; presence in D_BOREHOLE_CONSTRUCTION; presence in the original database (e.g. through checking of the original MOE import database); examination of the values themselves (where whole numbers are considered to be ‘fbgs’ and real/decimal numbers considered to be ‘mbgs’; note that this particular point is assumed across multiple table-field checks).

When the original depths are missing, the deepest unit is determined by examination of D_BOREHOLE_CONSTRUCTION and BH_BOTTOM_DEPTH (in D_BOREHOLE) when present. All other layers are taken to be equal intervals down-hole. Comments detailing this process will be present in GEOL_COMMENT (e.g. ‘imposed depth’). In addition, the GEOL_SUBCLASS_CODE will be tagged with the value of ‘5’ (i.e. ‘Original (Corrected)'). Note that a NULL value in GEOL_SUBCLASS_CODE is assumed to be original/non-modified (i.e. it hasn’t been compared against the original hard copy form).

If no depths are found, the GEOL_SUBCLASS_CODE is given a value of ‘7’ (i.e. ‘Original (Invalid)').

If the top depth of the last unit is present but the bottom depth is not, it is assumed that the driller stopped at an (hard) interface. This layer is extended with a false bottom of ‘1ft’ (or ‘0.3m’ as appropriate).

If the material codes matches to a surficial material (e.g. topsoil, fill, ...), a depth of ‘1ft’ (or ‘0.3m’ as appropriate) is assumed. The subsequent layer depths (and elevations) should be correct as well.

In some cases, one (or more) layers have a measurement unit that does not correspond to the remainder of the layers. Here, the values modified to match the most likely unit and the depth values are modified appropriately (e.g. whole numbers are assumed, by default, to be ‘fbgs’). The GEOL_SUBCLASS_CODE would be assigned a value of ‘5’.

Where the layers are missing one of top or bottom depths, preceding or succeeding layer depths are used – the difference is split amongst the layers (e.g. a two layer borehole with a depth, of one layer, of '30.5' metres with no other depths indicated; here, a top depth would be applied of '15.25' to the second layer while the first would have a '0' top and a '15.25' bottom).

Where a layer is not contributing any additional information (i.e. no colour or materials) and duplicates depths of other layers, this layer is removed and the remaining layers are marked 'Simplified' in the GEOL_COMMENT field.

G.12 Creation of the TRAINING Database (a Subset of the MASTER Database)

This section has been removed. Previously, the Training database would be created by making a backup of the ORMGP master database and subsequently deleting those records that lie outside the specified Training study area (this was defined by a temporary table containing LOC_IDs that are to be used).

Refer to Appendix G.????? for details on creating a Partner database (the same methodology could be used to create a Training database).

G.13 Synchronizing Non-Replicating Databases

There is no longer any non-replicating, separate databases – the ORMGP database is centralized and now accessed through the ORMGP web interface at

<https://www.oakridgeswater.ca>

or through the Citrix XenDesktop machines. In both cases, these are (in general) read-only connections; the XenDesktop connections do allow changes/updates to be made for specific users.

For information on incorporation of database information in a similar format to the ORMGP database, refer to Appendix G.27.

G.14 Population of coordinates

Tables

- D_LOCATION
- D_LOCATION_QA
- R_LOC_COORD_OUOM_CODE

Views (OAK_CHECK)

- CHK_D_LOC_Coords_Not_Assigned

Estimated Recurrence Time: As necessary

Coordinates are found in D_LOCATION within the fields LOC_COORD_EASTING and LOC_COORD_NORTHING. These are in UTMz17-NAD83 projection and datum (the system default). If not populated (i.e. they are NULL) the original coordinates are found in LOC_COORD_EASTING_OUOM and LOC_COORD_NORTHING_OUOM with the original projection/datum referenced as a reference code (to R_LOC_COORD_OUOM_CODE).

The view 'CHK_D_LOC_Coords_Not_Assigned' extracts the *_OUOM coordinates from D_LOCATION where the QA code is not '117'. It also includes the projection and datum information. If NULL or invalid values are represented here by the coordinate columns, the QA code should be immediately changed in D_LOCATION_QA to '117' (storing the original value in QA_COORD_CONFIDENCE_CODE_ORIG; a comment can be included, in QA_COORD_COMMENT, as well).

If the coordinates are not in system units (i.e. UTMz17, NAD83) they should be exported and converted (remember to keep the LOC_ID as part of the export to relate-back the converted coordinates). Otherwise, the coordinates can be copied directly to LOC_COORD_EASTING and LOC_COORD_NORTHING.

G.15 Synchronize elevations between D_BOREHOLE and D_LOCATION_ELEVATION

Tables (OAK_20120615_MASTER)

- D_BOREHOLE
- D_LOCATION_ELEV
- D_LOCATION_QA

Views (OAK_CHECKS)

- CHK_D_LOC_ELEV_D_BOREHOLE

Estimated Recurrence Time: 1 month

The values found in BH_GND_ELEV, BH_GND_ELEV_OUOM and BH_DEM_GND_ELEV (in D_BOREHOLE) should all match that of ASSIGNED_ELEV (in D_LOCATION_ELEV). If the QA code has a value of '1' (note that there are two QA confidence codes – one for coordinates, QA_COORD_CONFIDENCE_CODE, and a second for elevations, QA_ELEV_CONFIDENCE_CODE; a '1' in either implies either a survey-grade coordinate in the horizontal or a survey-grade elevation in the vertical dimension), the location has been surveyed. Refer to Section 2.4 for details regarding the assignment of elevations (i.e. what values end up in ASSIGNED_ELEV and the appropriate columns in D_BOREHOLE).

Though SiteFX uses the elevations contained within D_BOREHOLE for calculation, all elevations should be stored initially in D_LOCATION_ELEV and only the ASSIGNED_ELEV should be copied across to the D_BOREHOLE table.

The view CHK_D_LOC_ELEV_D_BOREHOLE returns those LOC_IDs which have differing values between the D_LOCATION_ELEV (ASSIGNED_ELEV) and the D_BOREHOLE (BH_GND_ELEV) table. These need to be evaluated by the user to determine the appropriate values to use/apply for each of the particular elevation fields available. Example code (also checking against the QA_COORD_CONFIDENCE_CODE) would be

```
SELECT
chk.[LOC_ID]
,[ASSIGNED_ELEV]
,[SURVEYED_ELEV]
,[DEM_MNR_10m_v2]
,[ELEV_ORIGINAL]
,[BH_GND_ELEV]
,[BH_GND_ELEV_OUOM]
,[BH_GND_ELEV_UNIT_OUOM]
,[BH_BOTTOM_ELEV]
,BH_BOTTOM_DEPTH
,dlqa.QA_COORD_CONFIDENCE_CODE
,dlqa.QA_ELEV_CONFIDENCE_CODE
FROM [OAK_CHECKS].[dbo].[CHK_D_LOC_ELEV_D_BOREHOLE] as chk
inner join
[OAK_20120615_MASTER].[dbo].D_LOCATION_QA as dlqa
on
chk.LOC_ID=dlqa.LOC_ID
where
dlqa.QA_COORD_CONFIDENCE_CODE != 1
order by
chk.loc_id
```

If a location has been surveyed and no SURVEYED_ELEV is present in D_LOCATION_ELEV, the BH_GND_ELEV_OUOM (converted to 'masl' as necessary) value will be assigned to SURVEYED_ELEV (and, subsequently, to the ASSIGNED_ELEV and all the borehole elevation fields). In almost all other cases, the BH_GND_ELEV_OUOM (converted) will be stored in ELEV_ORIGINAL and the DEM_MNR_10m_v2 value will be assigned to ASSIGNED_ELEV (and to all the appropriate D_BOREHOLE fields). Do not forget, the BH_GND_ELEV_UNIT_OUOM field needs to be corrected (or changed) to read 'masl' and that the BH_BOTTOM_ELEV field needs to be updated. So, for non-surveyed locations, D_BOREHOLE can be updated directly.

```
select
dbore.LOC_ID
,dbore.BH_GND_ELEV
,chk.ASSIGNED_ELEV
,chk.ELEV_ORIGINAL
,dbore.BH_BOTTOM_DEPTH
from
[OAK_20120615_MASTER].[dbo].D_BOREHOLE as dbore
inner join
[OAK_CHECKS].[dbo].CHK_D_LOC_ELEV_D_BOREHOLE as chk
on
dbore.LOC_ID=chk.LOC_ID
```

```

inner join
[OAK_20120615_MASTER].dbo.D_LOCATION_QA as dlqa
on
dbore.LOC_ID=dlqa.LOC_ID
where
dlqa.QA_COORD_CONFIDENCE_CODE != 1

update [OAK_20120615_MASTER].dbo.D_BOREHOLE
set
BH_GND_ELEV=chk.ASSIGNED_ELEV
,BH_GND_ELEV_OUOM=chk.ASSIGNED_ELEV
,BH_DEM_GND_ELEV=chk.ASSIGNED_ELEV
,BH_BOTTOM_ELEV=chk.ASSIGNED_ELEV-dbore.BH_BOTTOM_DEPTH
from
[OAK_20120615_MASTER].dbo.D_BOREHOLE as dbore
inner join
[OAK_CHECKS].dbo.CHK_D_LOC_ELEV_D_BOREHOLE as chk
on
dbore.LOC_ID=chk.LOC_ID
inner join
[OAK_20120615_MASTER].dbo.D_LOCATION_QA as dlqa
on
dbore.LOC_ID=dlqa.LOC_ID
where
dlqa.QA_COORD_CONFIDENCE_CODE != 1

```

If a SURVEYED_ELEV is present in D_LOCATION_ELEV and a different elevation is present in the D_BOREHOLE field BH_GND_ELEV_ORIG (translated values don't signify), it is considered that the surveyed elevation has been updated (unless there is, seemingly, a loss in accuracy; an example of this may be a change in the value recorded from 6 significant digits down to 5 significant digits). As such, SURVEYED_ELEV is assigned this value and all other fields adjusted accordingly.

The BH_GND_ELEV, if a change has been made, can be set to NULL to enable a re-conversion of all appropriate elevations for this location. Remember, all ground elevations in D_BOREHOLE should be set to the same value (the ASSIGNED_ELEV). Also, NULLs do not show up in comparisons (i.e. these rows will not be pulled with a CHK_D_LOC_ELEV_D_BOREHOLE run).

G.16 Check D_INTERVAL_FORMATION_ASSIGNMENT for Invalid (Null) Rows

This applied to the second version of the D_INTERVAL_FORMATION_ASSIGNMENT table which has been superseded by D_INTERVAL_FORM_ASSIGN and D_INTERVAL_FORM_ASSIGN_FINAL.

G.17 Correction of elevations (D_BOREHOLE and D_LOCATION_ELEV)

Tables (OAK_20120615_MASTER)

- D_BOREHOLE
- D_LOCATION_ELEV
- D_LOCATION_QA

Views (OAK_CHECKS)

- CHK_D_LOC_ELEV_D_BOREHOLE
- CHK_D_LOC_ELEV_D_BOREHOLE_NOT_ASSIGNED
- CHK_D_LOC_ELEV_MISSING
- CHK_D_LOC_ELEV_NOT_ASSIGNED
- CHK_D_LOC_ELEV_SURV_NULL

Estimated Recurrent Time: 1 month

At times, elevation corrections are made (generally through SiteFX) that update elevations in D_BOREHOLE that do not get passed onto D_LOCATION_ELEV. In particular, the latter table should catch any locations whose status has been designated as 'surveyed' (i.e. QA_ELEV_CONFIDENCE_CODE of '1').

CHK_D_LOC_ELEV_SURV_NULL checks whether a surveyed elevation exists for any location with a QA_ELEV_CONFIDENCE_CODE of '1'. This view returns elevations from D_BOREHOLE and D_LOCATION_ELEV to be used as a check if no SURVEYED_ELEV exists. In most cases, the BH_GND_ELEV_OUOM should be used to update SURVEYED_ELEV as BH_GND_ELEV should match that of ASSIGNED_ELEV (if the latter do not match, the BH_GND_ELEV* should match; if these do not, the original documents from which the elevations are drawn should be referenced). Correct these elevations as necessary (i.e. in D_LOCATION_ELEV).

CHK_D_LOC_ELEV_D_BOREHOLE_NOT_ASSIGNED should be run to compare the D_LOCATION_ELEV and D_BOREHOLE tables. Elevations should be updated as appropriate (with regard to the QA_* codes). Note that, in the absence of surveyed elevations, the current value in BH_GND_ELEV should be copied into ELEV_ORIGINAL if the latter is NULL. Update all depths in D_BOREHOLE before changing the elevations.

CHK_D_LOC_ELEV_D_BOREHOLE should be run subsequently (this compares the values/differences between the D_BOREHOLE and D_LOCATION_ELEV tables). BH_GND_ELEV_OUOM (and *_UNIT) should be updated appropriately with both BH_GND_ELEV and BH_BOTTOM_ELEV values deleted. The SiteFX update routine should then be run.

A complete example follows. Firstly, include/add the missing LOC_IDs into D_LOCATION_ELEV:

```
insert into [OAK_20120615_MASTER].dbo.D_LOCATION_ELEV
(LOC_ID)
SELECT
[LOC_ID]
FROM
[OAK_CHECKS].[dbo].[CHK_D_LOC_ELEV_Missing]
```

Then update the 'DEM_MNR_10m_v2' and 'DEM_SRTM_90m_v41' fields based upon the externally derived elevations for the missing locations:

```

update [OAK_20120615_MASTER].dbo.D_LOCATION_ELEV
set
DEM_MNR_10m_v2=leu.MNR
,DEM_SRTM_90m_v41=leu.SRTM
FROM
[OAK_20120615_MASTER].dbo.D_LOCATION_ELEV as delev
inner join [tempHold].[dbo].[LOC_ELEV_UPDATE_20150320] as leu
on delev.loc_id=leu.loc_id

```

Check for surveyed values and modify the SURVEYED_ELEV field as appropriate:

```

SELECT
[LOC_ID]
,[ASSIGNED_ELEV]
,[SURVEYED_ELEV]
,[DEM_MNR_10m_v2]
,[BH_GND_ELEV]
,[BH_GND_ELEV_OUOM]
,[BH_GND_ELEV_UNIT_OUOM]
FROM [OAK_CHECKS].[dbo].[CHK_D_LOC_ELEV_SURV_NULL]

```

And then:

```

update [OAK_20120615_MASTER].dbo.D_LOCATION_ELEV
set
SURVEYED_ELEV=t1.BH_GND_ELEV
from
[OAK_20120615_MASTER].dbo.D_LOCATION_ELEV as delev
inner join
(
select
chk.LOC_ID
,chk.BH_GND_ELEV
from
[OAK_CHECKS].[dbo].[CHK_D_LOC_ELEV_D_BOREHOLE_NOT_ASSIGNED] as chk
where
QA_COORD_CONFIDENCE_CODE=1 or QA_ELEV_CONFIDENCE_CODE=1
) as t1
on delev.LOC_ID=t1.LOC_ID

```

Assign those elevations, as appropriate (i.e. using the MNR or SRTM DEM) for those locations that are not boreholes (and, thus, do not exist in the D_BOREHOLE table):

```

update [OAK_20120615_MASTER].dbo.D_LOCATION_ELEV
set
assigned_elev=
case
when delev.DEM_MNR_10m_v2 is not null then delev.DEM_MNR_10m_v2
else delev.DEM_SRTM_90m_v41
end
from
[OAK_20120615_MASTER].dbo.D_LOCATION_ELEV as delev
inner join [OAK_CHECKS].[dbo].[CHK_D_LOC_ELEV_D_BOREHOLE_NOT_ASSIGNED] as chk
on delev.loc_id=chk.loc_id
where
chk.bh_gnd_elev is null

```

Update the ELEV_ORIGINAL field based upon the existing ground elevation in D_BOREHOLE:

```

update [OAK_20120615_MASTER].dbo.D_location_elev

```

```

set
elev_original=dbore.BH_GND_ELEV
from
[OAK_20120615_MASTER].dbo.D_LOCATION_ELEV as delev
inner join [OAK_20120615_MASTER].dbo.D_BOREHOLE as dbore
on delev.loc_id=dbore.loc_id
inner join [OAK_CHECKS].[dbo].[CHK_D_LOC_ELEV_D_BOREHOLE_NOT_ASSIGNED] as chk
on dbore.loc_id=chk.loc_id

```

Update the elevations and depth in D_BOREHOLE (note that the elevation to use is dependent upon whether there is a SURVEYED, MNR or SRTM value):

```

update [OAK_20120615_MASTER].dbo.D_BOREHOLE
set
bh_bottom_ouom=bh_bottom_depth
,BH_BOTTOM_UNIT_OUOM='mbgs'
,bh_bottom_elev=null
,BH_GND_ELEV=null
,bh_gnd_elev_ouom=chk.DEM_MNR_10m_v2
,BH_DEM_GND_ELEV=chk.DEM_MNR_10m_v2
from
[OAK_20120615_MASTER].dbo.D_BOREHOLE as dbore
inner join [OAK_CHECKS].[dbo].[CHK_D_LOC_ELEV_D_BOREHOLE_NOT_ASSIGNED] as chk
on dbore.loc_id=chk.loc_id

```

Assign the appropriate value to ASSIGNED_ELEV:

```

update [OAK_20120615_MASTER].dbo.D_LOCATION_ELEV
set
ASSIGNED_ELEV=
case
when delev.SURVEYED_ELEV is not null then delev.SURVEYED_ELEV
when delev.DEM_MNR_10m_v2 is not null then delev.DEM_MNR_10m_v2
else delev.DEM_SRTM_90m_v41
end
from
[OAK_20120615_MASTER].dbo.D_LOCATION_ELEV as delev
inner join [OAK_CHECKS].[dbo].[CHK_D_LOC_ELEV_D_BOREHOLE_NOT_ASSIGNED] as chk
on delev.loc_id=chk.loc_id

```

The D_BOREHOLE table can (now) be updated based upon the differences found between the two tables (note the correction of the BH_BOTTOM_ELEV):

```

update [OAK_20120615_MASTER].dbo.D_BOREHOLE
set
BH_GND_ELEV=chk.ASSIGNED_ELEV
,BH_GND_ELEV_OUOM=chk.ASSIGNED_ELEV
,BH_GND_ELEV_UNIT_OUOM='masl'
,BH_DEM_GND_ELEV=chk.ASSIGNED_ELEV
,BH_BOTTOM_ELEV=chk.ASSIGNED_ELEV-dbore.BH_BOTTOM_DEPTH
from
[OAK_20120615_MASTER].dbo.D_BOREHOLE as dbore
inner join [OAK_CHECKS].[dbo].[CHK_D_LOC_ELEV_D_BOREHOLE] as chk
on dbore.loc_id=chk.loc_id

```

Periodically, a check should be made that 'new' boreholes (or other location types) have been included in the D_LOCATION_ELEV table. In this case, the D_LOC_ELEV_MISSING view will return those LOC_IDs that do not appear in latter table. These can be incorporated directly using the query:

```

insert into [OAK_20120615_MASTER].dbo.D_LOCATION_ELEV

```

```
(LOC_ID)
SELECT
[LOC_ID]
FROM [OAK_CHECKS].[dbo].[CHK_D_LOC_ELEV_Missing]
```

The CHK_D_LOC_ELEV_NOT_ASSIGNED view returns the coordinates of those LOC_IDs that do not have an ASSIGNED_ELEV (or a DEM value assigned) in the D_LOCATION_ELEV table – this is used to directly import those locations into a GIS so as to determine the elevations (from the MNR and SRTM DEMs).

Inserting and populating new LOC_IDs is described further in Section G.4.

G.18 Extracting LOC_IDs for the Training Database

This was required when the Partner databases were being replicated between MAIN and MAIN\PARTNER and we were accessing geometry information on the former and extracting it to the latter. This relied on instructions found in Section 3.3.4 as well as the use of ‘pass-through’ queries through the ‘openquery()’ function.

G.19 Addition of INT_ID to D_INTERVAL_FORMATION_ASSIGNMENT

Tables

- D_INTERVAL_MONITOR
- D_INTERVAL_FORMATION_ASSIGNMENT

Views (OAK_CHECK)

- CHK_D_INT_FM_ASSIGN_Mon_Missing
- CHK_D_INT_FM_ASSIGN_Mon_Missing_Count
- CHK_D_INT_FM_ASSIGN_Multiple_INT_IDs
- CHK_D_INT_FM_ASSIGN_Blank_INT_IDs
- CHK_D_INT_FM_ASSIGN_Multiple_Blank_INT_IDs
- CHK_D_INT_MON_TOP_BOT

Estimated Recurrence Time: 1 month

In D_INTERVAL_MONITOR, any one INT_ID can occur multiple times (if multiple screens are present within the borehole within a single ‘pipe’) as SYS_RECORD_ID is the primary key. As such, the minimum and maximum depth for all of these records (i.e. all SYS_RECORD_ID keys tied to a particular INT_ID) must be used when adding the INT_ID to D_INTERVAL_FORMATION_ASSIGNMENT (DIFA; which should only have a single row for each disparate INT_ID). (The views CHK_D_INT_FM_ASSIGN_Multiple_INT_IDs and CHK_D_INT_FM_ASSIGN_Blank_INT_IDs can be used for cleaning-up the table, as necessary.)

The fields MON_TOP_DEPTH_M and MON_BOT_DEPTH_M should be populated (along with associated fields) as outlined in Section G.1 (above). Use CHK_D_INT_MON_TOP_BOT to check this.

Use CHK_D_INT_FM_ASSIGN_Multiple_Blank_INT_IDs to check for duplicate INT_ID's that have been incorporated in the D_INTERVAL_FORMATION_ASSIGNMENT table but are empty. These can be deleted.

```
select
difa.SYS_RECORD_ID
,chk.ID_COUNT
from
[OAK_20120615_MASTER].dbo.D_INTERVAL_FORMATION_ASSIGNMENT as difa
inner join
[OAK_CHECKS].[dbo].[CHK_D_INT_FM_ASSIGN_Multiple_Blank_INT_IDs] as chk
on
difa.INT_ID=chk.INT_ID
```

```
delete from [OAK_20120615_MASTER].dbo.D_INTERVAL_FORMATION_ASSIGNMENT
where
SYS_RECORD_ID
in
(
select
difa.SYS_RECORD_ID
from
[OAK_20120615_MASTER].dbo.D_INTERVAL_FORMATION_ASSIGNMENT as difa
inner join
[OAK_CHECKS].[dbo].[CHK_D_INT_FM_ASSIGN_Multiple_Blank_INT_IDs] as chk
on
difa.INT_ID=chk.INT_ID
```

Use CHK_D_INT_FM_ASSIGN_Mon_Missing to determine those INT_IDs currently missing from the DIFA table – note that this command groups together common INT_IDs and selects the minimum top and maximum bottom of all the screens found in D_INTERVAL_MONITOR (any rows with NULL top or bottom depths will be dropped). These (number of screens) should be checked (in advance) to see how many correspond to the particular INT_ID; this is accomplished using CHK_D_INT_FM_ASSIGN_Mon_Missing_Count where the 'rcount' field returns a value greater than '1' where multiple screens are associated with a borehole.

These INT_IDs need to be inserted into the table along with a SYS_RECORD_ID, the latter functioning as the primary key for the table (do not forget to get a count of the number of rows being inserted first – this count takes the place of the '500' value, below. This is accomplished by, for example, the following code:

```
insert into [OAK_20120615_MASTER].dbo.D_INTERVAL_FORMATION_ASSIGNMENT
(INT_ID,SYS_RECORD_ID)
select
test.INT_ID
,vr.NEW_ID as SYS_RECORD_ID
from
(
select
INT_ID
,ROW_NUMBER () over (order by INT_ID) as num
```

```

from
[OAK_CHECKS].[dbo].[CHK_D_INT_FM_ASSIGN_Mon_Missing]
) as test
inner join
(
select
top 500
vr.NEW_ID
,ROW_NUMBER() over (order by NEW_ID) as rnum
from
[OAK_SUP].[dbo].[V_Random_ID_Creator_MD] as vr
where
vr.NEW_ID not in
(
select
SYS_RECORD_ID
from
[OAK_20120615_MASTER].[dbo].[D_INTERVAL_FORMATION_ASSIGNMENT]
)
) as vr
on
test.rnum=vr.rnum

```

Notice that we're matching the number of random identifiers (i.e. the SYS_RECORD_ID values) to the approximate number of values returned by the 'test' table (in the example).

The methodology outlined in Section G.1 can now be followed to populate the values in the table.

G.20 Calculate and Incorporate Specific Capacity

Tables

- D_INTERVAL_TEMPORAL_2
- D_PUMPTEST
- D_PUMPTEST_STEP

Views (OAK_SUP)

- V_YPDT_Calculate_Specific_Capacity
- V_YPDT_Determine_Specific_Capacity

Estimated Recurrence Time: various (generally upon data import)

Specific capacity can be calculated when, for a particular interval, each of the D_PUMPTEST (PUMPTEST_DATE), D_PUMPTEST_STEP (PUMP_RATE) and D_INTERVAL_TEMPORAL_2 (RD_DATE; RD_NAME_CODE – '628' for static water level, '70889' for 'Water Level – Manual – Other'; and RD_TYPE_CODE – '65' for 'WL – MOE Well Record – Pumping') tables have been populated with pumping rates as well as static and pumping water levels for a particular date (broken down by year-month-day; multiple pumping tests can exist).

'V_YPDT_Calculate_Specific_Capacity' is used to calculate specific capacity for all intervals where the pertinent information available for an interval. It also allows checks to be made with regard to the 'depth_m' (which should not be '0'), sc_lpmpm (i.e. specific

capacity in 'litres per minute per metre'; this should not be null) and the 'pump_rate_count' (which should only have a value of '1') fields.

'V_YPDT_Determine_Specific_Capacity', using the 'V_YPDT_Calculate_Specific_Capacity' view as a base returns only those calculated records which do not already exist within the database. All records with NULL (or other problematic) fields have been removed (refer to the previous paragraph). Field names have been adjusted to correspond to that expected by D_INTERVAL_TEMPORAL_2. For import/insertion into this latter table, a SYS_RECORD_ID needs to be associated with each row.

The number of rows needs to be determined, first.

```
select
COUNT(*)
from
[OAK_SUP].dbo.V_YPDT_Determine_Specific_Capacity
```

This number is then incorporated (rounded up; in this case 3318 is rounded to 4000) as part of the SYS_RECORD_ID creation.

```
select
t1.[INT_ID]
,t1.[RD_NAME_CODE]
,t1.[RD_DATE]
,t1.[RD_VALUE]
,t1.[UNIT_CODE]
,t1.[RD_NAME_OUOM]
,t1.[RD_VALUE_OUOM]
,t1.[RD_UNIT_OUOM]
,t2.SYS_RECORD_ID
from
(
select
v.[INT_ID]
,v.[RD_NAME_CODE]
,v.[RD_DATE]
,v.[RD_VALUE]
,v.[UNIT_CODE]
,v.[RD_NAME_OUOM]
,v.[RD_VALUE_OUOM]
,v.[RD_UNIT_OUOM]
,ROW_NUMBER() over (order by v.INT_ID) as rnum
from
[OAK_SUP].dbo.V_YPDT_Determine_Specific_Capacity as v
) as t1
inner join
(
select
top 4000
vr.NEW_ID as SYS_RECORD_ID
,ROW_NUMBER() over (order by vr.NEW_ID) as rnum
from
[OAK_SUP].dbo.V_Random_ID_Creator_MD as vr
where
vr.NEW_ID
not in
(
select SYS_RECORD_ID from [OAK_20120615_MASTER].dbo.D_INTERVAL_TEMPORAL_2
)
) as t2
```

on
t1.rnum=t2.rnum

Note that D_INTERVAL_TEMPORAL_2 is examined so that no duplication of the SYS_RECORD_ID is created. These can then be inserted straight into the temporal table.

G.21 Perform QA/QC Check against OAK_20120615_MASTER Backup

Tables (OAK_20120615_MASTER)

- All

Tables (OAK_CHK_20140104)

- All

Estimated Recurrence Time: 3 months

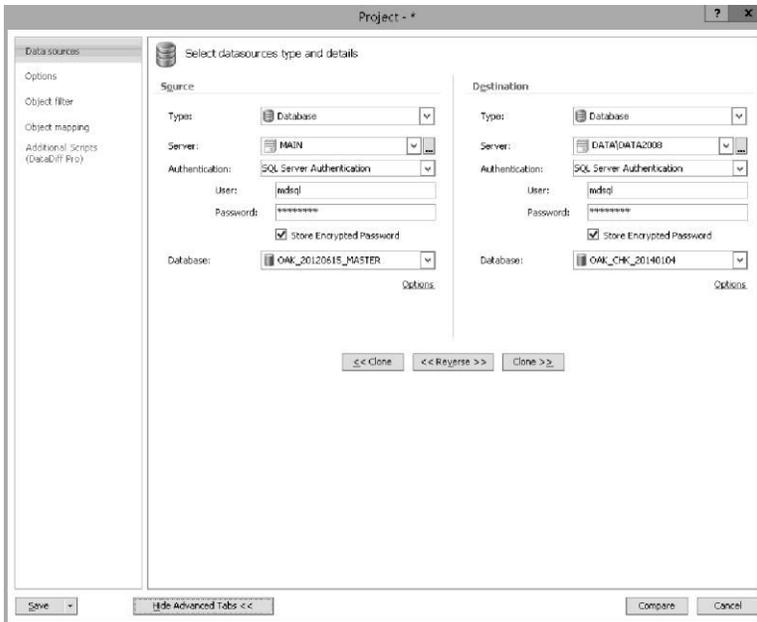
The first part of this procedure outlines the configuration process for ApexSQL; this must be done, once, to setup the tables to be examined – this should be saved as a ‘Project’. Subsequent comparisons (against the same destination database) use the same project but required that the destination database name (corresponding to the OAK_CHK_<date> database we’re checking against) be adjusted to match the ‘new’ name.

The second part of this procedure outlines the actual comparison process.

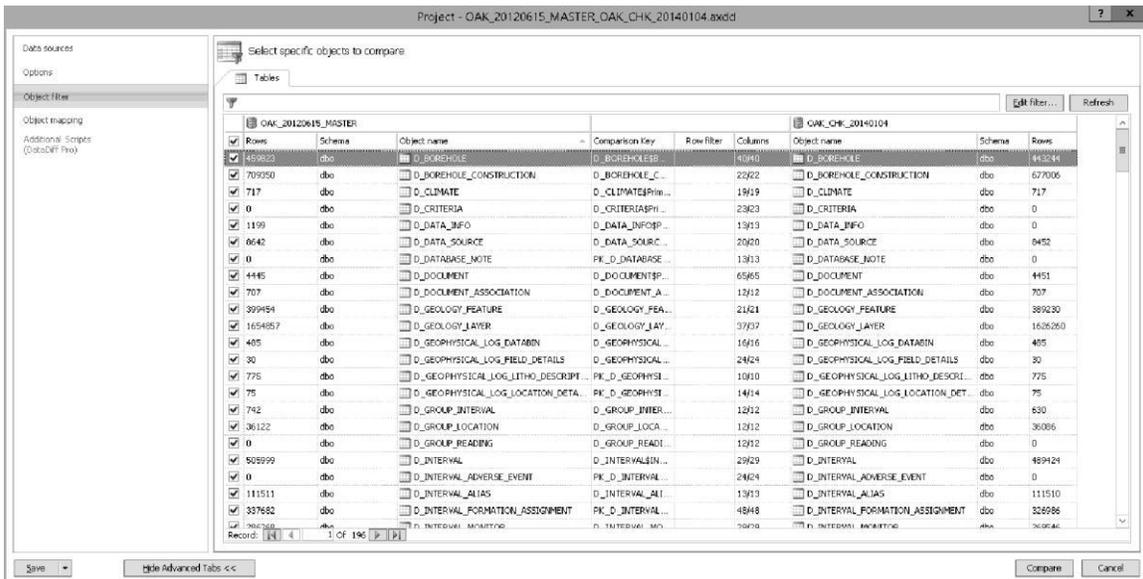
In both cases, a backup of the OAK_20120615_MASTER database must be loaded into the DATA\DATA2008 SQL Server instance (on DATA) by: creating an empty database (don’t forget that the data and log files are located in two different locations) – use the name OAK_CHK_<date> where <date> corresponds to the backup date of the database; select ‘OAK_CHK_20140104 - <right-click> - Tasks – Restore – Database’ and use the ‘.bak’ file (i.e. the master database backup) as the ‘Device’ from which to backup (replace the existing database and reassign the location of the data and log files).

Configuration

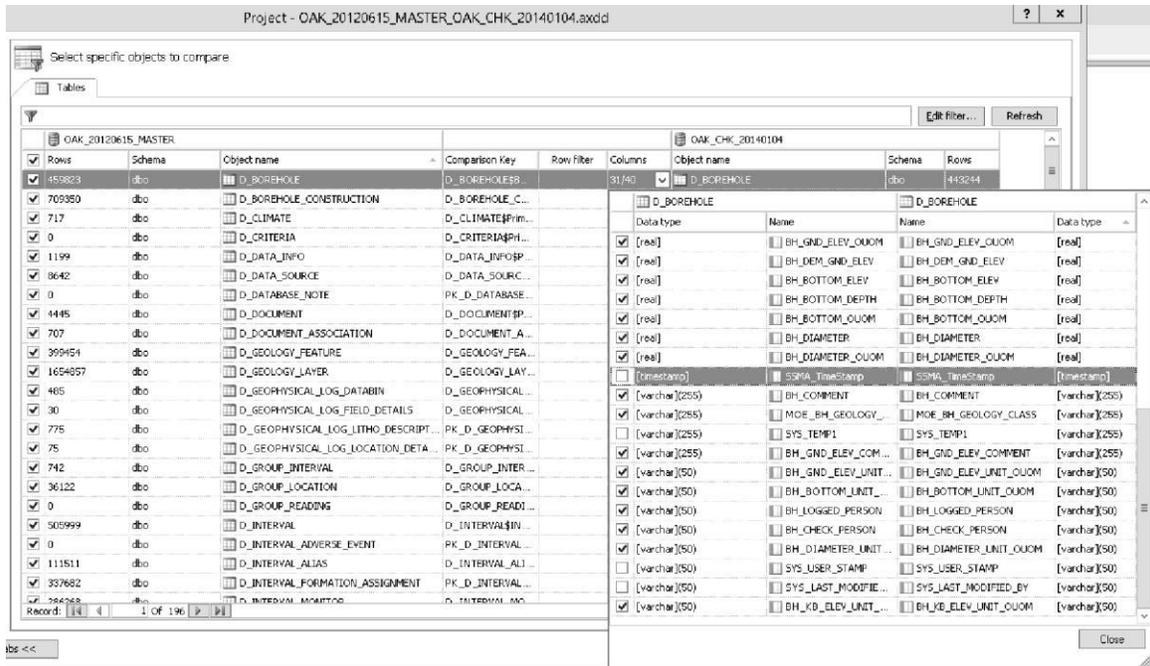
A new ‘Project’ is created for comparing the databases and tables to be examined. For the databases, the source will correspond to OAK_20120615_MASTER while the destination will be OAK_CHK_20140104 (note that the actual date string should be changed depending upon the backup of the master database used). The ‘Advanced Tabs’ should be available so the individual tables and fields can be specified.



Select the 'Object Filter' option; this will load the metadata from each database and perform table matching (which should match).



For each table, remove the 'SYS_*' fields from comparison (as well as any 'SSMA_Timestamp' fields); this is accomplished by selecting the 'Columns' value for each table name/'Object Name') – this does not include the SYS_RECORD_ID field (which should remain checked).



The following tables can be removed from examination (as they should be empty or not useful for comparison between databases).

- D_CRITERIA
- D_DATABASE_NOTE
- D_GROUP_READING
- D_INTERVAL_ADVERSE_EVENT
- D_ITEMPORAL_2_LOG
- D_LOCATION_DEPTH_DATA
- D_LOCATION_INFO
- D_LOCATION_INFO_DETAIL
- D_LOGGER_CALIBRATION_READINGS
- R_ADVERSE_COMMENT_CODE
- R_ADVERSE_TYPE_CODE
- R_GROUP_READING_CODE
- R_GROUP_READING_TYPE_CODE
- R_INT_ALIAS_TYPE_CODE
- R_INT_REGULATORY_CODE
- R_INT_SAMPLE_MATRIX_DESC
- R_INT_SAMPLE_TYPE_DESC
- R_INT_SAMPLE_USER_FILTER_DESC
- R_LOC_CORR_WATER_CODE
- R_LOC_INFO_CODE
- R_LOC_TIER1_CODE
- R_LOC_TIER2_CODE
- R_LOC_TIER3_CODE

- S_ARCHIVE
- S_CHANGE_HISTORY
- S_DATA_SEARCH_INTERVALS
- S_GLOBALSAVED_*
- S_GROUP_SEARCH
- S_IMPORT_MAPPINGS
- S_LOGFILE
- S_RPTSETTINGS*
- S_SAM_*
- S_SiteData*
- S_SRV_*
- S_TBL*
- S_USER_SETDB
- Variables_Standards
- VarStandsTemp
- Z_OLD_R_GEOL_FORMATION_CODE

This table list should be re-evaluated periodically.

Save the 'Project' file to 'C:\NTFS_MNTS\SOURCEDISKS\replication\apexsql_projects' with the filename specifying the source and destination databases (note that this destination also corresponds to the scripts and source material necessary for the 'Replication' process). The file in this case would be 'OAK_20120615_MASTER_OAK_CHK_20140104.axdd'. There should be a project file for each destination database examined.

Comparison

Once the project is loaded (or created and saved), the 'Compare' button (within the 'Project' window) can be selected.

G.22 Incorporation of the MOE Permit-To-Take-Water database (as of 2014-10)

Tables

- D_LOCATION
- D_LOCATION_PURPOSE
- D_LOCATION_QA
- D_PTTW_PERMIT
- D_PTTW_SOURCE
- R_PTTW_CLIENT_CODE
- R_PTTW_WATER_SOURCE

Estimated Recurrence Time:

The following fields are found within the MOE Permit-To-Take-Water (PTTW) database:

- PERMITNO
- CLIENTNAME
- PURPOSECAT
- SPURPOSE
- EXPIRYDATE
- ISSUEDDATE
- RENEWDATE
- OLDCTYTWN
- P_LOT
- P_CON
- P_MUNICIP
- P_UPPERT
- P_LOWERT
- SURFGRND
- SOURCEID
- EASTING
- NORTHING
- UTMZONE
- MAXL_DAY
- DAYS_YEAR
- HRS_DAYMAX
- L_MINUTE
- AMENDED_BY
- EXPIRED_BY
- PERMIT_END
- ACTIVE
- LATITUDE
- LONGITUDE

Explanations for conversion or mapping of these fields are described below (descriptions are from the 'Field_Description' sheet of the imported database/excel file).

PERMITNO

Permit number for water taking.

Mapped to D_PTTW_PERMIT.PTTW_PERMIT_NUMBER

CLIENTNAME

Name of client requesting a permit.

Convert to numeric D_PTTW_PERMIT.PTTW_CLIENT_CODE; based upon contents of R_PTTW_CLIENT_CODE (linked by PTTW_CLIENTNAME; note that this table is created by finding the distinct CLIENTNAMEs from the input dataset).

PURPOSECAT

Major taking category.

Convert to D_LOCATION_PURPOSE.PURPOSE_PRIMARY_CODE; based upon contents of R_PURPOSE_PRIMARY_CODE (the description of the PURPOSECAT is checked against those found in R_PURPOSE_PRIMARY_CODE). Check that the purpose is already defined within R_PURPOSE_PRIMARY_CODE (and add as necessary).

SPURPOSE

Specific purpose.

Convert to D_LOCATION_PURPOSE.PURPOSE_SECONDARY_CODE; based upon contents of R_PURPOSE_SECONDARY_CODE (the description of SPURPOSE is checked against those found in R_PURPOSE_SECONDARY_CODE). Check that the purpose is already defined (and add as necessary)..

In the case of 'Municipal', check PURPOSECAT to differentiate between 'Exploration', 'Monitoring' and 'Supply' (they will most likely be 'Municipal Supply').

EXPIRYDATE

Expiry date of permit.

Mapped to D_LOCATION.LOC_END_DATE as well as D_PTTW_PERMIT.PTTW_EXPIRYDATE. Any invalid dates (usually empty in the original) are assigned the tag '1867-07-01'. Any invalid years (usually greater than '2050') are assigned the year '1867'.

ISSUEDDATE

Date the permit was issued

Mapped to D_LOCATION.LOC_START_DATE. Refer to EXPIRYDATE (above) regarding invalid dates.

RENEWDATE

Date that a legacy permit was renewed (legacy permits are those that were issued before 2005).

Mapped to D_PTTW_PERMIT.PTTW_RENEWDATE. Note that this is not a date (rather, a numeric).

OLDCTYTWN

Former township name related to lot/concession.

Mapped to D_PTTW_SOURCE.PTTW_OLDCTYTWN. Note that in most cases, the value here seems to match P_MUNICIP (if declared) from the import database.

P_LOT

Lot where permit is located.

This column contains information in a ‘free-form’ format; it can include LOT numbers, CONCESSION numbers and general addresses. As such, a straight numeric (or readily identified lot number) maps to D_LOCATION.LOC_LOT; the contents of P_LOT (regardless of numeric or address info) is copied to D_LOCATION.LOC_ADDRESS_INFO1. All ‘NA’ values (only – no other text is present) should be removed.

An attempt to handle concessions and lots specified internally (e.g. as part of the address) will be made using search routines. For lot’s, the following code can be used:

```
select
  [LOT and CON].[rnum]
,regexp([LOT and CON].[P_LOT],"(.*)(Lot ?[0-9][0-9]?)(.*)","$2") as [test]
from
  [LOT and CON]
where
  [P_LOT] likex ".*Lot ?[0-9][0-9]?.*"
```

For concessions, the following codes can be used:

```
select
  [LOT and CON].[rnum]
,regexp([LOT and CON].[P_LOT],"(.*)(Concession ?[0-9][0-9]?)(.*)","$2") as [regexp_CONCESSION]
from
  [LOT and CON]
where
  [P_LOT] likex ".*Concession ?[0-9][0-9]?.*"
```

Note that multiple iterations are necessary as, for example, ‘Lot’ could also be specified as ‘Lt’ (and other combinations) and ‘Concession’ could also be specified as ‘Conc’ (and other combinations). Both the P_LOT and P_CONC columns need to be examined.

P_CON

Concession where permit is located.

The information contained here is similar to that found in P_LOT and will be handled in much the same way. Note that a straight numeric (or readily identified concession number) will map to D_LOCATION.LOC_CON while address information will map to LOC_ADDRESS_INFO2. Refer to ‘P_LOT’ above for details.

P_MUNICIP

Current municipality where permit is located.

Map to D_LOCATION.LOC_TOWNSHIP_CODE. Also can be used to populate D_LOCATION.LOC_COUNTY_CODE. Note that modifications to R_LOC_TOWNSHIP_COE.LOC_TOWNSHIP_ABBR have been made to perform 'easier' matches to text strings from this field (and used to populate LOC_COUNTY_CODE, only).

P_UPPERT

Upper tier (county/district/regional municipality) where permit is located.

Mapped to D_PTTW_SOURCE.PTTW_P_UPPERT.

P_LOWERT

Lower tier (county/district/regional municipality) where permit is located.

Mapped to D_PTTW_SOURCE.PTTW_LOWERT.

SURFGRND

Origin of water taking: Surface, Ground or Both

Indicates the source of water, one of: surface, groundwater or both. Change to lookup code PTTW_WATER_SOURCE_CODE (table R_PTTW_WATER_SOURCE_CODE). Mapped to D_PTTW_PERMIT.PTTW_WATER_SOURCE_CODE.

SOURCEID

Description of exact location of water taking.

Mapped to D_PTTW_SOURCE.PTTW_SOURCEID.

EASTING

Easting coordinates in UTM.

Only used if LATITUDE/LONGITUDE is not defined. The assumed datum is NAD83. Coordinates are translated to longitude. If blank, a QA_COORD_CONFIDENCE_CODE of '117' will be applied. Refer to

Mapped (when used) to D_LOCATION.LOC_COORD_EASTING_OUOM.

NORTHING

Northing coordinates in UTM.

Refer to EASTING (above) for use. Coordinates are translated to latitude.

Mapped (when used) to D_LOCATION.LOC_COORD_NORTHING_OUOM.

UTMZONE

UTM Zone (one of 15, 16, 17 or 18).

Only used if LATITUDE/LONGITUDE is not defined. If blank, a value of NAD83Z17 is assumed. Used in conjunction with EASTING and NORTHING to assign latitude and longitude.

MAXL_DAY

Maximum litres allowed per day.

Mapped to D_PTTW_PERMIT.PTTW_MAX_L_DAY.

DAYS_YEAR

Number of days that water taking is allowed.

Mapped to D_PTTW_PERMIT.PTTW_MAX_DAYS_YEAR.

HRS_DAYMAX

Maximum water taking in hours per day.

Mapped to D_PTTW_PERMIT.PTTW_MAX_HRS_DAY.

L_MINUTE

Allowable taking per minute (in litres).

Map to D_PTTW_PERMIT.PTTW_MAX_L_MINUTE.

AMENDED_BY

Pervious permit number.

Mapped to D_PTTW_PERMIT.PTTW_AMENDED_BY.

EXPIRED_BY

Previous permit number.

Mapped to D_PTTW_PERMIT.PTTW_EXPIRED_BY.

PERMIT_END

Date at which the water permit ends.

Refer to EXPIRYDATE (above) regarding invalid dates.

Mapped to D_PTTW_PERMIT.PTTW_EXPIRYDATE.

ACTIVE

Indicates whether the permit is active or expired (yes/no) based on PERMIT_END and the date that data was extracted from the source database.

Mapped to D_LOCATION.LOC_ACTIVE (a 0/1 field).

LATITUDE

The latitude coordinates.

Mapped to LOC_COORD_NORTHING_OUOM. Where this value is not defined, NORTHING is used instead. A NAD83 datum is assumed.

LONGITUDE

The longitude coordinates.

Mapped to LOC_COORD_EASTING_OUOM. Where this value is not defined, EASTING is used instead. A NAD83 datum is assumed.

Primary Key – Input Database

Each row from the input database is tagged with a separate numeric value (using an ‘Arithmetic Series’ with a step of ‘1’). This allows each row to be identified with a ‘primary key’ (similar to a database-table).

Additional Information

In addition, the following information is incorporated:

D_LOCATION

Coordinates

The coordinates (both UTM and Latitude/Longitude) are assumed to have a NAD83 datum. Any UTMZONE with a value of ‘0’ is assumed to have a value of ‘17’ (i.e. Zone 17). Latitude and Longitude are used to convert to UTMZ17 coordinates (the former are used as OUOM values). If only UTM coordinates are available, those are used as a substitute. If neither Latitude/Longitude or UTM coordinates are available, the QA_COORD_CONFIDENCE_CODE is assigned a value of ‘117’ and the coordinates and zone are given a NULL value.

Lots and Concessions

Any empty P_LOT or P_CON is considered undefined. Any P_LOT or P_CON with a value of ‘NA’ (or similar) or ‘0’ is also considered undefined. These are removed from consideration (made NULL).

D_LOCATION_QA

Each PTTW location is assigned a QA_COORD_CONFIDENCE_CODE of '5' (with the ...)

R_LOC_TOWNSHIP_CODE

The LOC_TOWNSHIP_ABBR field has been updated (currently empty in R_LOC_TOWNSHIP_CODE) to reflect easier-to-match strings for the PTTW free-form database (and can be used for other import data). This comparison has been done through a combination of R_LOC_COUNTY_CODE and R_LOC_TOWNSHIP_CODE where both have a LOC_COUNTY_CODE field (the information we are trying to determine for the specific PTTW). Note that the PTTW location LOC_COUNTY_CODE can only be populated in this manner (not the LOC_TOWNSHIP_CODE).

Duplicates

A check against duplicate records should be made. This can be performed against the source database and/or the populated (i.e. for import) tables. An SQL 'GROUP BY' check – where all columns in the source database are specified – is used to see what columns are 'exactly' the same. The results can be examined and any offending rows removed; those that are 'approximately' the same are grouped together into one record with appropriate notes made in PTTW_SOURCE_COMMENT (as found in D_PTTW_SOURCE) and/or PTTW_PERMIT_COMMENT (as found in D_PTTW_PERMIT) of this change. The number of rows 'grouped' together due to duplication is indicated by the PTTW_GROUPED field in D_PTTW_SOURCE (where the number indicated here represents the 'total' number of inputs rows, including this one); this field should be NULL otherwise.

D_LOCATION is then examined (using all fields) to determine duplicate 'locations' (i.e. the same coordinates and information with the same LOC_NAME and etc...). The minimum value for the 'rkey' (i.e. the key that we're using to originally number the rows from the input file; remember that this was a straight 'Arithmetic Series' with a step of '1') is determined as well – this is the 'row' that we'll keep as the 'main' location. All other of these rows will be removed from the D_LOCATION table (but will remain in the D_PTTW_PERMIT and D_PTTW_SOURCE tables) as well as the D_LOCATION_QA table. For D_LOCATION_PURPOSE, the purpose codes will be examined and any duplication (i.e. the same primary key and the same PURPOSE_PRIMARY_CODE and PURPOSE_SECONDARY_CODE) will be removed. Note that there should be the same or more rows in D_LOCATION_PURPOSE than in D_LOCATION (unless non-assigned purposes are removed).

Master Database Incorporation

At this point, all tables will be linked together with a non-unique (to the master database) identifier. A new unique identifier (depending upon the table) should be assigned based

upon existing values within the master database (e.g. assignment of new LOC_IDS or SYS_RECORD_IDS). Each table should be updated with these numeric values. All non-values (usually signified with a '-9999' place holder) should be assigned a NULL. The input information can now be added to the master database.

G.23 Population of D_INTERVAL_MONITOR (Top and Bottom)

Tables

- D_INTERVAL_MONITOR
- D_LOCATION
- D_BOREHOLE
- D_BOREHOLE_CONSTRUCTION
- D_INTERVAL
- R_INT_TYPE_CODE

Views

- None required (see query statements, below)

Estimated Recurrence Time: various (generally upon data import)

D_INTERVAL_MONITOR contains various screen information – of primary importance (here) are the depths/elevations of the interval/screen top and bottom. Note that there should only be a single INT_ID row – no duplicate INT_IDS should be present.

A variety of screen ‘types’ are specified (as found in R_INT_TYPE_CODE; each screen type is given a ‘Screen’ tag in the INT_TYPE_ALT_CODE field). They are

- 18 – Reported Screen
- 19 – Assumed Screen (Overburden Well 1ft Screen Assigned)
- 20 – Reported Open Hole
- 21 – Assumed Open Hole (Bottom of Casing to Bottom of Hole)
- 22 – Assumed Open Hole (Top of Bedrock to Bottom of Hole)
- 27 – Reported Open Hole (Derived from Borehole Construction)
- 28 – Screen Information Omitted
- 123 – Assumed Screen (‘Top of ...’ Information Only)

Each of these are described below. Note that some of these types are only used when there is a lack of information and various assumptions must be made. In the case where there is no depths (of any kind) reported, no screen is assigned.

Reported Screen (18)

The top- and bottom-depth of the screen has been reported as part of the dataset. This is used directly.

Assumed Screen (Overburden Well 1ft Screen Assigned) (19)

Reported Open Hole (20)

Assumed Open Hole (Bottom of Casing to Bottom of Hole) (21)

Casing information has been specified as part of the borehole construction information and bedrock has been intersected - no screen has been listed. In this case, the screen interval is taken as the bottom of the casing to the bottom of the hole.

Assumed Open Hole (Top of Bedrock to Bottom of Hole) (22)

If this is a bedrock well (i.e. the borehole intersects bedrock), the screen interval is taken from the top of bedrock to the bottom of the hole.

Reported Open Hole (Derived from Borehole Construction) (27)

Screen Information Omitted (28)

No information is available (either reported or derived from construction details) to populate the top- and bottom-depths of the screen interval. No record will appear in the D_INTERVAL_MONITOR table.

Assumed Screen ('Top of ...' Information Only) (123)

From borehole construction information, only a 'Top of ...' value is available (e.g. a top of plug depth has been specified but no depth). In this case, a 1ft (0.3m) screen interval is assigned below this depth.

Split – No depths, no screen

Split – Reported Screen – use top and bottom (18)

Split – Overburden vs Bedrock (own analysis; MOE check)

Overburden – screen assigned to 1ft above bottom of well (19)

Split – Bedrock wells; casing vs non-casing

Casing – bottom of casing to bottom of well

No Casing – top of bedrock to bottom of well

G.24 Update D_INTERVAL_MONITOR Depths (M)

Tables

- D_INTERVAL_MONITOR

Views (OAK_CHECKS)

- CHK_D_INT_MON_DEPTHS_M

Estimated Recurrence Time: 1 Month

The fields MON_TOP_DEPTH_M and MON_BOT_DEPTH_M are not automatically calculated upon data import. As such, the CHK_D_INT_MON_DEPTHS_M view allow easy calculation of the depths in metres from the OUOM fields; each of ‘mbgs’, ‘fbgs’ and ‘masl’ are supported. Errors or units other than those listed result in ‘-9999’ tag returned. This value should be checked before updating D_INTERVAL_MONITOR. For example:

```
select
chk.*
from
[OAK_CHECKS].dbo.CHK_D_INT_MON_DEPTHS_M as chk
where
chk.MON_TOP_DEPTH_M=-9999
```

Any rows returned should be corrected before finalizing (note that the D_LOCATION_QA table is accessed for checking against non-valid locations).

The actual table can now be updated:

```
update [OAK_20120615_MASTER].dbo.D_INTERVAL_MONITOR
set
MON_TOP_DEPTH_M=chk.MON_TOP_DEPTH_M
,MON_BOT_DEPTH_M=chk.MON_BOT_DEPTH_M
from
[OAK_20120615_MASTER].dbo.D_INTERVAL_MONITOR as dim
inner join [OAK_CHECKS].dbo.CHK_D_INT_MON_DEPTHS_M as chk
on dim.SYS_RECORD_ID=chk.SYS_RECORD_ID
```

Note that only those rows with NULL MON_TOP_DEPTH_M and MON_BOT_DEPTH_M as well as non-NULL MON_TOP_OUOM and MON_BOT_OUOM fields will be examined.

G.25 Correction of Water Levels and Associated Data

Tables

- D_BOREHOLE
- D_BOREHOLE_CONSTRUCTION
- D_INTERVAL_MONITOR
- D_INTERVAL_REF_ELEV
- D_INTERVAL_TEMPORAL_2
- D_LOCATION_ELEVATION

Views (OAK_CHECK)

- CHK_ELEVS_AND_DEPTHS

Views (OAK_SUP)

- V_YPDT_ELEVS_AND_DEPTHS

Estimated Recurrence Time: As necessary

As part of the process for creation of regional water level surfaces (whether shallow or deep), checks and corrections – as necessary – should be made with regard to the water level data as contained within D_INTERVAL_TEMPORAL_2. In most cases, invalid water levels arise out of calculated elevations based upon an invalid or old reference elevation (as found in D_INTERVAL_REF_ELEV). These water levels can be corrected through a modification of the existing reference elevations (either or both of REF_ELEV and REF_ELEV_OUOM; note that REF_POINT should be updated to include the value of the stick-up), removal of the calculated values populating RD_VALUE (in D_INTERVAL_TEMPORAL_2) and re-running the data-calculation scheme within SiteFX.

However, other errors can be found.

Water Levels as Metres-Above-Sea-Level (masl)

When existing RD_VALUE_OUOMs are in units of 'masl', the REF_ELEV is not used. In this case, the existing REF_ELEV should be used to change these values to depths (as appropriate; note that in some cases, the ELEV_ORIGINAL in D_LOCATION_ELEV may need to be used, instead). The SiteFX routine should then be run to re-calculate the correct RD_VALUES.

In some cases, no original value or reference elevation seems to apply to these water levels (i.e. the application of the above correction cannot be performed; the original elevation to which these water levels were referenced has been lost). Where these values are 'very-incorrect' such that they impact the water level surface calculation being performed, a mid-point depth (based upon the borehole depth and the variation in the water levels) is assigned instead. The original values are recorded in RD_COMMENT with an explanation.

Invalid or Incorrect BH_BOTTOM_DEPTH

In these cases, the BH_BOTTOM_DEPTH indicates that the well depth is above that of the recorded water levels. (The case of a 'deep' well with shallow water levels is not usually considered as invalid and must be examined/corrected using an alternative scheme.) Here, the view V_YPDT_ELEVS_AND_DEPTHS is used to check whether the calculated BH_BOTTOM_DEPTH is valid in comparison with construction detail depths (D_BOREHOLE_CONSTRUCTION), interval monitor depths

(D_INTERVAL_MONITOR), geologic layer depths (D_GEOLOGY_LAYER) and, finally, water level depths (D_INTERVAL_TEMPORAL_2). Preferably, the deepest (largest depth) recorded amongst the first three parameters is used to correct the BH_BOTTOM_DEPTH (and related information; e.g. BH_BOTTOM_ELEV). In cases where no other depth is present, a depth is assigned based upon the water level depth – usually a '0.3' metre value is added dependent upon the number of water levels recorded. If more than a single value is found, this value should be added to the largest (in all cases thus far, only a single water level has been found).

The view CHK_YPDT_ELEVS_AND_DEPTHS can be used, generally, to find those boreholes whose currently assigned BH_BOTTOM_DEPTH is not within '+/- 1' metre of the deepest (i.e. the maximum) of the various depths being examined (the '1' metre value is arbitrary and may be adjusted).

G.26 Correction or update of borehole coordinates (in comparison with an updated MOE Water Well Database)

Tables

- D_LOCATION
- D_LOCATION_SPATIAL
- D_LOCATION_SPATIAL_HIST
- D_LOCATION_GEOM
- D_LOCATION_QA

Estimated Recurrence Time: After MOE Water Well Database update (Appendix G.10)

Corrections to coordinates within the database are made under certain conditions or assumptions. As an aid (or check) to the process, the centroid of all Township polygons were determined. All MOE borehole locations with an invalid coordinate pair (identified with a QA code of '117') but with an identifiable Township code were given a new QA code ('118') and assigned the coordinates of the centroid (i.e. to LOC_COORD_EASTING and LOC_COORD_NORTHING). Note that this check was initially performed using the '2016-05-31' release of the MOE Water Well Database.

Any change in the coordinates of a location has a tremendous impact upon a number of tables (listed above). A new elevation will need to be determined (Appendix G.4) and incorporated into D_LOCATION_SPATIAL_HIST (matched to the updated coordinates). This will directly impact elevation values in multiple tables that will need to be updated (refer to Appendix G.XXX to do so).

The DATA_ID of the particular MOE WWDB import should be included as part of the record added to D_LOCATION_SPATIAL_HIST in order to track update occurrences. These records should be tagged with a LOC_COORD_HIST_CODE of '5' ('MOE WWIS Update').

Coordinate Check 00 – MOE Zone 17 Coordinates

The coordinates for MOE locations in Zone 17 and Zone 18 (encompassing the ORMGP study area) should be converted to consistent Zone 17 coordinates for easy comparison.

```
select
v.loc_id
,c.east83
,c.north83
,c.east83_orig
,c.north83_orig
,c.zone_orig
,m.utmrc
,m.location_method
,m.improvement_location_source
,m.improvement_location_method
,m.elevation
,m.elevrc
,row_number() over (order by v.loc_id) as rkey
into moe_20210119.dbo.YC_20210119_BORE_HOLE_ID_COORDS_UPD
from
oak_20160831_master.dbo.v_sys_moe_locations as v
inner join moe_20210119.dbo.YC_20210119_BORE_HOLE_ID_COORDS_YC_SRC as c
on v.moe_bore_hole_id=c.bore_hole_id
--on v.moe_well_id=c.well_id
inner join oak_20160831_master.dbo.d_location as d
on v.loc_id=d.loc_id
inner join moe_20210119.dbo.tblbore_hole as m
on v.moe_bore_hole_id=m.bore_hole_id
where
d.loc_type_code= 1
group by
v.loc_id,c.east83,c.north83,c.east83_orig,c.north83_orig,c.zone_orig
,m.utmrc,m.location_method,m.improvement_location_source,m.improvement_location_method
,m.elevation,m.elevrc
```

Note that there may be duplicate LOC_IDs (based upon grouping of BORE_HOLE_IDs to one location). As such, this should be checked and corrected (for the MOE 20200721 import, there were five duplicate LOC_ID rows; the coordinates were within one metre of each other; one of each duplicate was removed manually).

Coordinate Check 01 – MOE Coordinates – Township (QA 118)

In the case that the MOE have updated the coordinates for a borehole, compare all QA '118' locations against the MOE coordinates. We're checking to see if the centroid coordinates (i.e. the assigned MOE Township code) lies within 25km (chosen as a default) of the MOE assigned coordinates. Note that this was applied against both Zone 17 and 18 coordinates (the latter converted to Zone 17) as found within the master database.

```
select
dloc.LOC_ID
,dloc.LOC_COORD_EASTING
,dloc.LOC_COORD_NORTHING
,cast( 5 as int ) as LOC_COORD_HIST_CODE
,cast( '2021-01-19' as datetime ) as LOC_COORD_DATE
,m.east83 as X
,m.north83 as Y
,cast( 26917 as int ) as EPSG_CODE
,m.east83_orig as X_OUOM
```

```

.m.north83_orig as Y_OUOM
.case
when m.zone_orig=18 then 26918
else 26917
end as EPSG_CODE_OUOM
--,m.utmrc as QA_COORD_CODE
.case
when m.utmrc is not null then cast(m.utmrc as int)
else null
end as QA_COORD_CODE
.cast(
.case
when m.location_method is not null then m.location_method + '; '
else ''
end
+
.case
when m.improvement_location_method is not null then m.improvement_location_method + '; '
else ''
end
+
.case
when m.improvement_location_source is not null then m.improvement_location_source + '; '
else ''
end
as varchar(255)) as LOC_COORD_METHOD
.cast('Updated from QA_COORD_CODE [118]' as varchar(255) ) as LOC_COORD_COMMENT
.cast( 523 as int ) as LOC_COORD_DATA_ID
.case
when m.elevation is not null then cast( 2 as int )
else null
end as LOC_ELEV_CODE
.case
when m.elevation is not null then cast( '2021-01-19' as datetime )
else null
end as LOC_ELEV_DATE
.m.elevation as LOC_ELEV
.case
when m.elevation is not null then cast( 6 as int )
else null
end as LOC_ELEV_UNIT_CODE
--,m.elevrc as QA_ELEV_CODE
.case
when len(m.elevrc)>0 then cast(m.elevrc as int)
else null
end as QA_ELEV_CODE
.cast('20210202a' as varchar(255) ) as SYS_TEMP1
.cast( 20210202 as int ) as SYS_TEMP2
from
moe_20210119.dbo.yc_20210119_bore_hole_id_coords_upd as m
inner join oak_20160831_master.dbo.d_location as dloc
on m.loc_id=dloc.loc_id
inner join oak_20160831_master.dbo.d_location_qa as dlqa
on dloc.loc_id=dlqa.loc_id
where
-- make sure to not include the current DATA_ID
( dloc.data_id is null or dloc.data_id<>523 )
and dlqa.qa_coord_confidence_code= 118
and dloc.loc_coord_easting between (m.east83 - 25000) and (m.east83 + 25000)
and dloc.loc_coord_northing between (m.north83 - 25000) and (m.north83 + 25000)

```

For MOE 20200721, this returned 240 locations – these were inserted as new rows in the coordinate tracking table D_LOCATION_SPATIAL_HIST.

Coordinate Check 02 – MOE Coordinates – Invalid (QA 117)

In the case that the MOE has updated its coordinates, compare the QA '117' location coordinates against the MOE coordinates. Here we're looking to see if the original (invalid) database coordinates differ from those in the recent MOE database. We'll need to check, firstly, that these have not already been captured in the database.

```
select
m.*
from
moe_20200721.dbo.yc_20200721_bore_hole_id_coords_upd_z17 as m
inner join d_location_qa as dlqa
on m.loc_id=dlqa.loc_id
inner join
(
select
loc_id
,x
,y
from
d_location_spatial_hist
where
x is not null and y is not null
group by
loc_id,x,y
) as t
on m.loc_id=t.loc_id and m.east83_final=t.x and m.north83_final=t.y
where
dlqa.qa_coord_confidence_code= 117
```

Any records returned should be marked as 'present' (i.e. they've already been added).

For new records, an allowance of a +/- 1m error in the coordinate values (for the comparison) is implemented. Note that this was applied against both Zone 17 and 18 coordinates (the latter was converted to Zone 17 in a previous step).

```
select
dloc.LOC_ID
,dloc.LOC_COORD_EASTING
,dloc.LOC_COORD_NORTHING
,cast( 5 as int ) as LOC_COORD_HIST_CODE
,cast( '2021-01-19' as datetime ) as LOC_COORD_DATE
,m.east83 as X
,m.north83 as Y
,cast( 26917 as int ) as EPSG_CODE
,m.east83_orig as X_OUOM
,m.north83_orig as Y_OUOM
,case
when m.zone_orig=18 then 26918
else 26917
end as EPSG_CODE_OUOM
--,m.utmrc as QA_COORD_CODE
,case
when m.utmrc is not null then cast(m.utmrc as int)
else null
end as QA_COORD_CODE
,cast(
case
when m.location_method is not null then m.location_method + ';'
else ''
end
+
case
when m.improvement_location_method is not null then m.improvement_location_method + ';'
else ''
end
```

```

+
case
when m.improvement_location_source is not null then m.improvement_location_source + ';'
else ''
end
as varchar(255) as LOC_COORD_METHOD
,cast('Updated from QA_COORD_CODE [117]' as varchar(255) ) as LOC_COORD_COMMENT
,cast( 523 as int ) as LOC_COORD_DATA_ID
.case
when m.elevation is not null then cast( 2 as int )
else null
end as LOC_ELEV_CODE
.case
when m.elevation is not null then cast( '2021-01-19' as datetime )
else null
end as LOC_ELEV_DATE
,m.elevation as LOC_ELEV
.case
when m.elevation is not null then cast( 6 as int )
else null
end as LOC_ELEV_UNIT_CODE
--,m.elevrc as QA_ELEV_CODE
.case
when len(m.elevrc)>0 then cast(m.elevrc as int)
else null
end as QA_ELEV_CODE
,cast( '20210202b' as varchar(255) ) as SYS_TEMP1
,cast( 20210202 as int ) as SYS_TEMP2
from
moe_20210119.dbo.yc_20210119_bore_hole_id_coords_upd as m
inner join oak_20160831_master.dbo.d_location as dloc
on m.loc_id=dloc.loc_id
inner join oak_20160831_master.dbo.d_location_qa as dlqa
on dloc.loc_id=dlqa.loc_id
where
dlqa.qa_coord_confidence_code= 117
and m.present is null
and
(
not( dloc.loc_coord_easting between (m.east83 - 1) and (m.east83 + 1) )
or not( dloc.loc_coord_northing between (m.north83 - 1) and (m.north83 + 1) )
)
)

```

This returned 1 location for the MOE 20200721 import – these were added
D_LOCATION_SPATIAL_HIST.

Coordinate Check 03 – MOE Coordinates – Valid coordinates (non-QA 117 and 118)

In the case that the MOE has updated it's coordinates, a comparison of the MOE coordinates is made with non-QA '117' and '118' location coordinates. Here we're looking to see if the current location coordinates differ from those in the recent MOE database. An allowance of a +/- 1m error in the coordinate values (for the comparison) is implemented. As of the v20190509 MOE database only original UTM Zones 17 and 18 (as found in the MOE database) are examined.

```

select
dloc.LOC_ID
,dloc.LOC_COORD_EASTING
,dloc.LOC_COORD_NORTHING
,cast( 5 as int ) as LOC_COORD_HIST_CODE
,cast( '2021-01-19' as datetime ) as LOC_COORD_DATE
,m.east83 as X
,m.north83 as Y

```

```

.cast( 26917 as int ) as EPSG_CODE
.m.east83_orig as X_OUOM
.m.north83_orig as Y_OUOM
.case
when m.zone_orig=18 then 26918
else 26917
end as EPSG_CODE_OUOM
--,m.utmrc as QA_COORD_CODE
.case
when m.utmrc is not null then cast(m.utmrc as int)
else null
end as QA_COORD_CODE
.cast(
case
when m.location_method is not null then m.location_method + ','
else ''
end
+
case
when m.improvement_location_method is not null then m.improvement_location_method + ','
else ''
end
+
case
when m.improvement_location_source is not null then m.improvement_location_source + ','
else ''
end
as varchar(255)) as LOC_COORD_METHOD
.cast( 'Updated from various QA_COORD_CODEs (CHECK03)' as varchar(255) ) as LOC_COORD_COMMENT
.cast( 523 as int ) as LOC_COORD_DATA_ID
.case
when m.elevation is not null then cast( 2 as int )
else null
end as LOC_ELEV_CODE
.case
when m.elevation is not null then cast( '2021-01-19' as datetime )
else null
end as LOC_ELEV_DATE
.m.elevation as LOC_ELEV
.case
when m.elevation is not null then cast( 6 as int )
else null
end as LOC_ELEV_UNIT_CODE
--,m.elevrc as QA_ELEV_CODE
.case
when len(m.elevrc)>0 then cast(m.elevrc as int)
else null
end as QA_ELEV_CODE
.cast( '20210202c' as varchar(255) ) as SYS_TEMP1
.cast( 20210202 as int ) as SYS_TEMP2
from
moe_20210119.dbo.yc_20210119_bore_hole_id_coords_upd as m
inner join oak_20160831_master.dbo.d_location as dloc
on m.loc_id=dloc.loc_id
inner join oak_20160831_master.dbo.d_location_qa as dlqa
on dloc.loc_id=dlqa.loc_id
where
(dloc.data_id is null or dloc.data_id<>523 )
and dlqa.qa_coord_confidence_code not in ( 117, 118 )
and m.present is null
and
(
not( dloc.loc_coord_easting between (m.east83 - 1) and (m.east83 + 1) )
or not( dloc.loc_coord_northing between (m.north83 - 1) and (m.north83 + 1) )
)
and m.east83 is not null and m.north83 is not null
and m.east83 between ( 100000 ) and ( 1000000 )
and m.north83 between ( 1000000 ) and ( 6000000 )
and dloc.loc_id not in
(

```

```

select
distinct(loc_id)
from
oak_20160831_master.dbo.d_location_spatial_hist
where
loc_coord_hist_code=5
)

```

In addition, cut-off checks are used to exclude obviously bad coordinates (note the ranges specified for EAST83_FINAL and NORTH83_FINAL). The LOC_COORD_HIST_CODE value of '5' ('MOE WWIS Update') and the DATA_ID can be used to determine what coordinates were incorporated from what version of the MOE database (refer to D_DATA_SOURCE through V_SYS_MOE_DATA_ID to determine the version dates).

Coordinate Check 04 – MOE Coordinates – Geocoded Locations

The previous instructions for this section no longer apply – the 'gcode' locational method (from the MOE database) will be caught in 'Coordinate Check 03', above.

Coordinate Check 05 – MOE Boreholes – Manual Check

For those locations with a QA of '117' or '118', examine the original MOE report and determine for each borehole the location coordinates using a GIS or Google Maps (for example). As the latter returns latitude/longitude values (WGS84), convert these to UTM Zone 17 (NAD83) and store both sets of values.

Coordinate Check 06 – MOE Coordinates – Invalid (QA 117 or 118) Update

For those locations with a QA of '117' or '118', examine those MOE coordinates where they differ from the database *_OUOM coordinates and where the UTMRC code is not '9' (generally, these locations will have coordinates that have been tagged as invalid; for example, having values of '99999' or similar). Note the use of the LOC_COORD_HIST_CODE and DATA_ID to limit the examination to a particular MOE database import.

```

select
dloc.loc_id
,dlsh.loc_coord_hist_code
,dlsh.loc_coord_date
,dlsh.x
,dlsh.y
,dlsh.epsg_code
,dlsh.x_ouom
,dlsh.y_ouom
,dlsh.epsg_code_ouom
,dlsh.qa_coord_code
,dlsh.loc_coord_data_id
,dlsh.loc_coord_method
,dlsh.loc_coord_comment
,3 as loc_elev_code
,cast('2021-01-19' as datetime) as loc_elev_date
,6 as loc_elev_unit_code
,10 as qa_elev_code
,'20210202d' as sys_temp1

```

```

,20210202 as sys_temp2
from
oak_20160831_master.dbo.d_location as dloc
inner join oak_20160831_master.dbo.d_location_qa as dlqa
on dloc.loc_id=dlqa.loc_id
inner join oak_20160831_master.dbo.d_location_spatial_hist as dlsh
on dloc.loc_id=dlsh.loc_id
where
dlqa.qa_coord_confidence_code in ( 117, 118 )
and dlsh.loc_coord_data_id= 523
and ( dlsh.qa_elev_code is null or dlsh.qa_elev_code<>1 )

```

Insert these into D_LOCATION_SPATIAL_HIST and populate the elevations (note that the fields, above, are set to reflect the DEM that is presently being used – MNR version 2, 2005, at 10m resolution). Once the elevations have been populated, D_LOCATION_SPATIAL should be updated to point to these records.

Coordinate Check 07 – MOE Coordinates – QA improvement Update

For those locations whose QA codes, as found in the MOE database, are better than that currently specified, the new MOE coordinates are substituted. We'll first check against a current QA_COORD_CODE of 9 (this value should be indicative of those locations that have not been manually reviewed by the ORMGP group).

```

select
dlsh.loc_id
,dlsh.loc_coord_hist_code
,dlsh.loc_coord_date
,dlsh.x
,dlsh.y
,dlsh.epsg_code
,dlsh.x_ouom
,dlsh.y_ouom
,dlsh.epsg_code_ouom
,dlsh.qa_coord_code
,dlsh.loc_coord_method
,dlsh.loc_coord_comment + ' Updated from QA_COORD_CODE 9 (CHECK07) as loc_coord_comment
,dlsh.loc_coord_data_id
,.cast( 3 as int ) as loc_elev_code
,.cast( '2021-01-19' as datetime ) as loc_elev_date
,.cast( 6 as int ) as loc_elev_unit_code
,.cast( 10 as int ) as qa_elev_code
,'20210202e' as sys_temp1
,20210202 as sys_temp2
from
oak_20160831_master.dbo.d_location_spatial_hist as dlsh
inner join oak_20160831_master.dbo.v_sys_loc_coords as v
on dlsh.loc_id=v.loc_id
where
dlsh.sys_temp1= '20210119b'
and dlsh.qa_coord_code < v.qa_coord_code
and v.qa_coord_code= 9

```

These should be inserted into D_LOCATION_SPATIAL_HIST and updated with an elevation (from the current DEM). Make these the current coordinates in D_LOCATION_SPATIAL. Make a note of those with invalid elevations (i.e. outside of the DEM area) and assign a QA_COORD_CODE of '117'.

For the remainder of the QA_COORD_CODES, check if they've been manually modified based upon the LOC_COORD_HIST_CODE, the SYS_USER_STAMP and any of the comment fields. If indicated in the latter fields, update the LOC_COORD_HIST_CODE of the record to '8' (i.e. 'Manual – Other') and provide a comment in LOC_COORD_COMMENT.

```
select
dlsh2.*
--distinct( dlsh2.loc_coord_comment )
from
d_location_spatial_hist as dlsh
inner join v_sys_loc_coords as v
on dlsh.loc_id=v.loc_id
inner join d_location_spatial_hist as dlsh2
on v.spat_id=dlsh2.spat_id
where
dlsh.sys_temp1= '20200806b'
and dlsh.qa_coord_code < v.qa_coord_code
and v.qa_coord_code <> 9
and dlsh2.loc_coord_hist_code < 6
```

Find those that appear to be updates and not otherwise modified in the database.

```
select
dlsh.loc_id
,dlsh.loc_coord_hist_code
,dlsh.loc_coord_date
,dlsh.x
,dlsh.y
,dlsh.epsg_code
,dlsh.x_ouom
,dlsh.y_ouom
,dlsh.epsg_code_ouom
,dlsh.qa_coord_code
,dlsh.loc_coord_method
,dlsh.loc_coord_comment + '; Updated from QA_COORD_CODE ' + cast( v.qa_coord_code as varchar(255) ) + ' (CHECK07)' as
loc_coord_comment
,dlsh.loc_coord_data_id
,.cast( 3 as int ) as loc_elev_code
,.cast( '2021-01-19' as datetime ) as loc_elev_date
,.cast( 6 as int ) as loc_elev_unit_code
,.cast( 10 as int ) as qa_elev_code
,'20210202f' as sys_temp1
,'20210202' as sys_temp2
from
oak_20160831_master.dbo.d_location_spatial_hist as dlsh
inner join oak_20160831_master.dbo.v_sys_loc_coords as v
on dlsh.loc_id=v.loc_id
inner join oak_20160831_master.dbo.d_location_spatial_hist as dlsh2
on v.spat_id=dlsh2.spat_id
where
dlsh.sys_temp1= '20210202b'
and dlsh.qa_coord_code < v.qa_coord_code
and v.qa_coord_code <> 9
and dlsh2.loc_coord_hist_code < 6
```

Insert these into D_LOCATION_SPATIAL_HIST, update their elevations and update D_LOCATION_SPATIAL to reflect these changes.

For all of these checks and updates, be sure to update the appropriate fields in D_LOCATION, D_LOCATION_QA and D_LOCATION_GEOM (for the latter, specify

a COORD_CHECK of '10001' to indicate the coordinates and elevation have been modified; this is a key used as a reminder to update all elevation-affected fields).

G.27 York Database – Incorporation of Temporal Data

Tables

- D_INTERVAL_ALIAS
- D_INTERVAL_TEMPORAL_1A
- D_INTERVAL_TEMPORAL_1B
- D_INTERVAL_TEMPORAL_2

Estimated Recurrence Time: Based upon availability

In the absence of the SiteFX (EarthFX) merge tool, a methodology was developed and applied to incorporate temporal data found within the York database (YKDB) into the 'master' database (MADB). Between the two databases, common locations and intervals have differing LOC_IDs and INT_IDs. This has been examined (on dates leading up to, for example, each of '2015-09-30', '2015-10-02' and '2017-11-16') and the results incorporated into the D_INTERVAL_ALIAS table of the MADB (i.e. the INT_NAME_ALIAS will be the INT_ID of the interval within the YKDB) and given an INT_ALIAS_TYPE_CODE of '1' ('Matching INT_ID in the York database') or '2' ('Matching INT_ID in the York database – Not used'). These INT_IDs are then used (in the former case) as a basis for review of new temporal data.

Tables containing records to be copied into the MADB (matching, approximately, the format of D_INTERVAL_TEMPORAL_1A, D_INTERVAL_TEMPORAL_1B and D_INTERVAL_TEMPORAL_2) are created within the YKDB and then copied across to the 'temphold' database (TMPDB) for subsequent incorporation. These are created (within the YKDB) through the use of a series of views which extract information from the various temporal tables as found in the YKDB. All imports are provided with a distinctive DATA_ID (in the MADB D_DATA_SOURCE table) linking the imported information to a specific version/date of the YKDB.

These series of views created within the copy of the YKDB include, for D_INTERVAL_TEMPORAL_1A:

```
create view v_tr_dit1a as
select
t.int_id as ypdt_int_id
,t.york_int_id
,t2.ypdt_sam_id
,dit1a.sam_id as sam_id
,dit1a.sam_sample_name
,t2.ypdt_sam_sample_name
,dit1a.sam_sample_date
,dit1a.sam_analysis_date
,dit1a.sam_lab_sample_id
,t2.ypdt_sam_lab_sample_id
,dit1a.sam_lab_job_number
,t2.ypdt_sam_lab_job_number
```

```

.dit1a.sam_internal_id
.dit1a.sam_sample_name_ouom
.dit1a.sam_type_code
.t2.ypdt_sam_type_code
.dit1a.sam_sample_date_ouom
.dit1a.sam_comment
.dit1a.sam_data_file
.dit1a.sys_time_stamp
.dit1a.data_id
,row_number() over (order by dit1a.sam_id) as rkey
from
yorkdb_20210503.dbo.d_interval_temporal_1a as dit1a
inner join
-- this is the york-to-ypdt int_id conversion
(
select
dia.int_id
,cast(dia.int_name_alias as int) as york_int_id
from
oak_20160831_master.dbo.d_interval_alias as dia
where dia.int_alias_type_code=1
) as t
on dit1a.int_id=t.york_int_id
left outer join
-- this is the current contents of the master db
(
select
dit1a.int_id
,dit1a.sam_id as ypdt_sam_id
,dit1a.sam_sample_date
,dit1a.sam_sample_name as ypdt_sam_sample_name
,dit1a.sam_lab_sample_id as ypdt_sam_lab_sample_id
,dit1a.sam_lab_job_number as ypdt_sam_lab_job_number
,dit1a.sam_type_code as ypdt_sam_type_code
from
oak_20160831_master.dbo.d_interval_temporal_1a as dit1a
inner join oak_20160831_master.dbo.d_interval_alias as dia
on dit1a.int_id=dia.int_id
where
dia.int_alias_type_code=1
) as t2
on t.int_id=t2.int_id and t2.sam_sample_date=dit1a.sam_sample_date and dit1a.sam_type_code=t2.ypdt_sam_type_code

```

Note that this extracts 'all' data from the YKDB; those not present in the MADB will have a NULL YPDT_SAM_ID.

For D_INTERVAL_TEMPORAL_1B, there is an issue matching the RD_NAME_CODEs between the databases. As such, a view was created comparing the RD_NAME_OUOM to RD_NAME_DESCRIPTION (the latter is found in R_RD_NAME_CODE).

```

create view v_tr_rd_name_code as
select
t.rd_name_ouom
,case
when rmc.rd_name_code is not null then rmc.rd_name_code
when rma.rd_name_code is not null then rma.rd_name_code
else -9999
end as ypdt_rd_name_code
,case
when rmc.rd_name_code is not null then rmc.rd_name_description
when rma.rd_name_code is not null then rma.reading_name_alias
else '-9999'
end as ypdt_rd_name_description
from

```

```

(
select
distinct( dit1b.rd_name_ouom ) as rd_name_ouom
from
yorkdb_20210503.dbo.d_interval_temporal_1b as dit1b
inner join yorkdb_20210503.dbo.v_tr_dit1a as v
on dit1b.sam_id=v.sam_id
) as t
left outer join oak_20160831_master.dbo.r_rd_name_code as rmc
on t.rd_name_ouom like rmc.rd_name_description
left outer join oak_20160831_master.dbo.r_reading_name_alias as rra
on t.rd_name_ouom like rra.reading_name_alias

```

Unmatched names can either be added to R_READING_ALIAS (where the parameter already exists) or R_RD_NAME_CODE (if it does not); the tag '-9999' is used to signify this. Similarly, the UNIT_CODE is also examined.

```

create view v_tr_unit_code as
select
t.rd_unit_ouom
,t.rcount
,case
when ruc.unit_code is not null then ruc.unit_code
else
case
when t.rd_unit_ouom = 'pH' then 32
when t.rd_unit_ouom = 'mV' then 106
end
end as ypdt_unit_code
,case
when ruc.unit_code is not null then ruc.unit_description
else
case
when t.rd_unit_ouom = 'pH' then 'pH Units'
when t.rd_unit_ouom = 'mV' then 'millivolts'
end
end as ypdt_unit_description
from
(
select
rd_unit_ouom
,count(*) as rcount
from
yorkdb_20210503.dbo.d_interval_temporal_1b as d1b
inner join yorkdb_20210503.dbo.v_tr_dit1a as v
on d1b.sam_id=v.sam_id
group by
rd_unit_ouom
) as t
left outer join oak_20160831_master.dbo.r_unit_code as ruc
on t.rd_unit_ouom = ruc.unit_description

```

The view for extracting information from D_INTERVAL_TEMPORAL_1B can now be created.

```

create view v_tr_dit1b as
select
d1b.sam_id
,v.ypdt_sam_id
,d1b.rd_value_qualifier
,d1b.rd_value
,vmc.ypdt_rd_name_code
,vuc.ypdt_unit_code
,d1b.rd_md1
,d1b.rd_rd1
,vmc.ypdt_rd_name_description

```

```

.d1b.rd_value_ouom
,vuc.ypdt_unit_description
.d1b.rd_md1_ouom
.d1b.rd_rdl_ouom
.d1b.rd_comment
.d1b.sys_record_id
.cast(null as integer) as ypdt_sys_record_id
.d1b.sys_time_stamp
.row_number() over (order by d1b.sys_record_id) as rkey
from
yorkdb_20210503.dbo.d_interval_temporal_1b as d1b
inner join yorkdb_20210503.dbo.v_tr_dit1a as v
on d1b.sam_id=v.sam_id
inner join yorkdb_20210503.dbo.v_tr_rd_name_code as vrnc
on d1b.rd_name_ouom=vrnc.rd_name_ouom
inner join yorkdb_20210503.dbo.v_tr_unit_code as vuc
on d1b.rd_unit_ouom=vuc.rd_unit_ouom

```

Four equivalent D_INTERVAL_TEMPORAL_2 tables reside in the YKDB (as of 20210503); a view to extract information must be created for each.

These include D_INTERVAL_TEMPORAL_2, D_INTERVAL_TEMPORAL_EXT, D_INTERVAL_TEMPORAL_YORK and D_INTVL_TEMP2). The basic structure for each are the same (the corresponding names are: V_TR_DIT2; V_TR_DITEXT; V_TR_DITYORK; and V_TR_DITVL2).

```

create view v_tr_dit2 as
select
t.int_id as ypdt_int_id
,t.york_int_id
,d.rd_date
.case
  when d.rd_name_code=707 then 67
  when d.rd_name_code=711 then 85
  else null
end as rd_type_code
.case
  when d.rd_name_code=163 then 71037
  when d.rd_name_code=369 then 70871
  when d.rd_name_code=699 then 447
  when d.rd_name_code=700 then 629
  when d.rd_name_code=701 then 611
  when d.rd_name_code=702 then 612
  when d.rd_name_code=703 then 628
  when d.rd_name_code=706 then 70899
  when d.rd_name_code=707 then 70899
  when d.rd_name_code=711 then 629
  else null
end as rd_name_code
.case
  when d.unit_code=45 then d.rd_value*0.001
  else d.rd_value
end as rd_value
.case
  when unit_code=45 then 74
  when unit_code=6 then 6
  when unit_code=3 then 3
  else null
end as unit_code
,d.rd_name_ouom
,d.rd_value_ouom
,d.rd_unit_ouom
,d.sys_record_id
.cast(null as integer) as ypdt_sys_record_id
--,row_number() over (order by d2.sys_record_id) as rkey

```

```

from
yorkdb_20210503.dbo.d_interval_temporal_2 as d
inner join
(
select
dia.int_id
.cast(dia.int_name_alias as int) as york_int_id
from
oak_20160831_master.dbo.d_interval_alias as dia
where dia.int_alias_type_code=1
) as t
on d.int_id=t.york_int_id
where
d.rd_name_code in (163,369,699,700,701,702,703,706,707)

```

Note that the RD_NAME_CODE and UNIT_CODE conversions are taking place entirely within the view itself. The RD_TYPE_CODE field is unused in the YKDB but must be specified for the MADB to distinguish pump-on and pump-off conditions.

Due to the number of records (see below), the output from the views were written to a temporary database before determination of their new identifiers (to be used within the MADB) and final insertion.

For combining the three views for D_INTERVAL_TEMPORAL_2, an INTO statement and two insertions were used (with an increment for the row number in RKEY for each of the subsequent sources of information). Note that in addition to checking whether there is 'new' data to be loaded, differences in 'existing' data (i.e. present in both the YKDB and MADB for the particular interval) are being included (to be incorporated as an update process). For the latter, a range of values is used for comparison purposes.

D_INTERVAL_TEMPORAL_2

```

select
v.ypdt_int_id
,v.york_int_id
,v.sys_record_id
,v.ypdt_sys_record_id as ormgp_add_sys_record_id
,d.sys_record_id as ormgp_cur_sys_record_id
,v.rd_date
,v.rd_type_code
,v.rd_name_code
,v.rd_value
,d.rd_value as ormgp_cur_rd_value
,v.unit_code
,v.rd_name_ouom
,v.rd_value_ouom
,v.rd_unit_ouom
,row_number() over (order by v.sys_record_id) as rkey
into yorkdb_20210503.dbo.dit2
from
yorkdb_20210503.dbo.v_tr_dit2 as v
left outer join oak_20160831_master.dbo.d_interval_temporal_2 as d
on v.ypdt_int_id=d.int_id and v.rd_date=d.rd_date and v.rd_name_code=d.rd_name_code
where
v.rd_value is not null
and ( d.sys_record_id is null or (d.sys_record_id is not null and not( v.rd_value between (d.rd_value-0.001) and (d.rd_value+0.001) ) )
)

```

D_INTERVAL_TEMPORAL_EXT

```

insert into yorkdb_20210503.dbo.dit2
(
ypdt_int_id
,york_int_id
,sys_record_id
,ormgp_add_sys_record_id
,ormgp_cur_sys_record_id
,rd_date
,rd_type_code
,rd_name_code
,rd_value
,ormgp_cur_rd_value
,unit_code
,rd_name_ouom
,rd_value_ouom
,rd_unit_ouom
,rkey
)
select
v.ypdt_int_id
,v.york_int_id
,v.sys_record_id
,v.ypdt_sys_record_id as ormgp_add_sys_record_id
,d.sys_record_id as ormgp_cur_sys_record_id
,v.rd_date
,v.rd_type_code
,v.rd_name_code
,v.rd_value
,d.rd_value as ormgp_cur_rd_value
,v.unit_code
,v.rd_name_ouom
,v.rd_value_ouom
,v.rd_unit_ouom
,( row_number() over (order by v.sys_record_id) ) + 9645129 as rkey
from
yorkdb_20210503.dbo.v_tr_ditext as v
left outer join oak_20160831_master.dbo.d_interval_temporal_2 as d
on v.ypdt_int_id=d.int_id and v.rd_date=d.rd_date and v.rd_name_code=d.rd_name_code
where
v.rd_value is not null
and ( d.sys_record_id is null or (d.sys_record_id is not null and not( v.rd_value between (d.rd_value-0.001) and (d.rd_value+0.001) ) )
)

```

Note the modification of the ‘rkey’ field – this is to allow each row to be identified as a separate record.

D_INTERVAL_TEMPORAL_YORK

```

insert into yorkdb_20210503.dbo.dit2
(
ypdt_int_id
,york_int_id
,sys_record_id
,ormgp_add_sys_record_id
,ormgp_cur_sys_record_id
,rd_date
,rd_type_code
,rd_name_code
,rd_value
,ormgp_cur_rd_value
,unit_code
,rd_name_ouom
,rd_value_ouom
,rd_unit_ouom
,rkey
)

```

```

select
v.ypdt_int_id
,v.york_int_id
,v.sys_record_id
,v.ypdt_sys_record_id as ormgp_add_sys_record_id
,d.sys_record_id as ormgp_cur_sys_record_id
,v.rd_date
,v.rd_type_code
,v.rd_name_code
,v.rd_value
,d.rd_value as ormgp_cur_rd_value
,v.unit_code
,v.rd_name_ouom
,v.rd_value_ouom
,v.rd_unit_ouom
,( row_number() over (order by v.sys_record_id ) + 9664584 as rkey
from
yorkdb_20210503.dbo.v_tr_dityork as v
left outer join oak_20160831_master.dbo.d_interval_temporal_2 as d
on v.ypdt_int_id=d.int_id and v.rd_date=d.rd_date and v.rd_name_code=d.rd_name_code
where
v.rd_value is not null
and ( d.sys_record_id is null or (d.sys_record_id is not null and not( v.rd_value between (d.rd_value-0.001) and (d.rd_value+0.001) ) )
)

```

And D_INTVL_TEMP2

```

insert into yordb_20210503.dbo.dit2
(
ypdt_int_id
,york_int_id
,sys_record_id
,ormgp_add_sys_record_id
,ormgp_cur_sys_record_id
,rd_date
,rd_type_code
,rd_name_code
,rd_value
,ormgp_cur_rd_value
,unit_code
,rd_name_ouom
,rd_value_ouom
,rd_unit_ouom
,rkey
)
select
v.ypdt_int_id
,v.york_int_id
,v.sys_record_id
,v.ypdt_sys_record_id as ormgp_add_sys_record_id
,d.sys_record_id as ormgp_cur_sys_record_id
,v.rd_date
,v.rd_type_code
,v.rd_name_code
,v.rd_value
,d.rd_value as ormgp_cur_rd_value
,v.unit_code
,v.rd_name_ouom
,v.rd_value_ouom
,v.rd_unit_ouom
,( row_number() over (order by v.sys_record_id ) + 9664584 as rkey
from
yorkdb_20210503.dbo.v_tr_ditv12 as v
left outer join oak_20160831_master.dbo.d_interval_temporal_2 as d
on v.ypdt_int_id=d.int_id and v.rd_date=d.rd_date and v.rd_name_code=d.rd_name_code
where
v.rd_value is not null

```

and (d.sys_record_id is null or (d.sys_record_id is not null and not(v.rd_value between (d.rd_value-0.001) and (d.rd_value+0.001))))

The D_INTERVAL_TEMPORAL_1A/_1B tables are created using INTO statements.

D_INTERVAL_TEMPORAL_1A

```
SELECT
ypdt_int_id,
york_int_id,
ypdt_sam_id,
sam_id,
sam_sample_name,
ypdt_sam_sample_name,
sam_sample_date,
sam_analysis_date,
sam_lab_sample_id,
ypdt_sam_lab_sample_id,
sam_lab_job_number,
ypdt_sam_lab_job_number,
sam_internal_id,
sam_sample_name_ouom,
sam_type_code,
ypdt_sam_type_code,
sam_sample_date_ouom,
sam_comment,
sam_data_file,
sys_time_stamp,
data_id,
rkey
into yorkdb_20210503.dbo.dit1a
FROM
yorkdb_20210503.dbo.v_tr_dit1a
where
ypdt_sam_id is null
```

D_INTERVAL_TEMPORAL_1B

```
SELECT
sam_id,
ypdt_sam_id,
rd_value_qualifier,
rd_value,
ypdt_rd_name_code,
ypdt_unit_code,
rd_mdl,
rd_rdl,
ypdt_rd_name_description,
rd_value_ouom,
ypdt_unit_description,
rd_mdl_ouom,
rd_rdl_ouom,
rd_comment,
sys_record_id,
ypdt_sys_record_id,
sys_time_stamp,
rkey
into yorkdb_20210503.dbo.dit1b
FROM
yorkdb_20210503.dbo.v_tr_dit1b
where
ypdt_sam_id is null
```

The row counts for the 20171124 YKDB import were as follows:

D_INTERVAL_TEMPORAL_1A – 4865
D_INTERVAL_TEMPORAL_1B – 51119
D_INTERVAL_TEMPORAL_2 – 15871104

The row counts for the 20181008 YKDB import were as follows:

D_INTERVAL_TEMPORAL_1A – 679
D_INTERVAL_TEMPORAL_1B – 8451
D_INTERVAL_TEMPORAL_2 (update) – 3891217
D_INTERVAL_TEMPORAL_2 (new) – 7435230

These tables can now be processed for new identifiers and then incorporated in the MADB.

G.28 Updating elevations in D_* tables

Tables

- D_BOREHOLE
- D_BOREHOLE_CONSTRUCTION
- D_GEOLOGY_FORMATION
- D_GEOLOGY_LAYER
- D_PICK
- D_INTERVAL_REFERENCE_ELEVATION
- D_INTERVAL_MONITOR
- D_INTERVAL_SOIL
- D_INTERVAL_TEMPORAL_2

Views

- V_SYS_CHK_CORR_ELEV_DBOR
- V_SYS_CHK_CORR_ELEV_DBC
- V_SYS_CHK_CORR_ELEV_DGF
- V_SYS_CHK_CORR_ELEV_DGL
- V_SYS_CHK_CORR_ELEV_DPICK
- V_SYS_CHK_CORR_ELEV_DIRE
- V_SYS_CHK_CORR_ELEV_DIM
- V_SYS_CHK_CORR_ELEV_DIS
- V_SYS_CHK_CORR_ELEV_D2

Estimated Recurrence Time: Weekly (or at time of elevation update)

For correction of elevations the value of BH_GND_ELEV in D_BOREHOLE must be populated and correct (a separate process; refer to Appendix G.4 for details).

Each of the views, above, are run in turn.

V_SYS_CHK_CORR_ELEV_DBOR (D_BOREHOLE)

Calculates NEW_BH_BOTTOM_ELEV and NEW_BH_BOTTOM_DEPTH; returns a record if the former does not match the original BH_BOTTOM_ELEV.

V_SYS_CHK_CORR_ELEV_DBC (D_BOREHOLE_CONSTRUCTION)

Calculates NEW_CON_TOP_ELEV and NEW_CON_BOT_ELEV; returns a record if either does not match the original (and respective) CON_TOP_ELEV or CON_BOT_ELEV.

V_SYS_CHK_CORR_ELEV_DGF (D_GEOLOGY_FORMATION)

Calculates NEW_FEATURE_TOP_ELEV and NEW_FEATURE_BOT_ELEV; returns a record if either does not match the original (and respective) FEATURE_TOP_ELEV or FEATURE_BOT_ELEV.

V_SYS_CHK_CORR_ELEV_DGL (D_GEOLOGY_LAYER)

Calculates NEW_GEOL_TOP_ELEV and NEW_GEOL_BOT_ELEV; returns a record if either does not match the original (and respective) GEOL_TOP_ELEV or GEOL_BOT_ELEV.

V_SYS_CHK_CORR_ELEV_DPICK (D_PICK)

The difference between the ground elevation in D_PICK (GND_ELEV) and D_BOREHOLE (BH_GND_ELEV) is compared and stored in BE_PE_CMP_M if they do not match. The recalculated pick top elevation is stored in BE_NEW_TOP_ELEV. If BE_PE_CMP_M is not null then the ground elevation has changed and the pick should be updated. Note that allow rows are returned (this is different than most of the other views here).

V_SYS_CHK_CORR_ELEV_DIRE (D_INTERVAL_REFERENCE_ELEVATION)

This uses V_SYS_INT_REF_ELEV_RANGE as a source which calculates a new reference elevation REF_ELEV_BE_CMP (based upon BH_GND_ELEV and REF_STICK_UP). This value is compared against the current REF_ELEV and stored in BE_RE_CMP_M - if this value is not null then the ground elevation has changed and the record should be updated. Only records matching this case will be returned by the calling view.

V_SYS_CHK_CORR_ELEV_DIM (D_INTERVAL_MONITOR)

Calculates NEW_MON_TOP_ELEV, NEW_MON_BOT_ELEV, NEW_MON_TOP_DEPTH_M and NEW_MON_BOT_DEPTH_M; returns a record if either of the calculated elevations do not match (respectively) MON_TOP_ELEV or MON_BOT_ELEV.

V_SYS_CHK_CORR_ELEV_DIS (D_INTERVAL_SOIL)

Calculates NEW_SOIL_TOP_ELEV, NEW_SOIL_BOT_ELEV, NEW_SOIL_TOP_M and NEW_SOIL_BOT_M; returns a record if either of the calculated elevations do not match (respectively) SOIL_TOP_ELEV or SOIL_BOT_ELEV.

V_SYS_CHK_CORR_ELEV_D2 (D_INTERVAL_TEMPORAL_2)

Calculates NEW_RD_VALUE for any record that has a UNIT_CODE of '6' (i.e. 'masl'); returns a record if this does not match the current RD_VALUE

G.29 Update MOE BORE_HOLE_ID (D_LOCATION_ALIAS)

Tables

- D_LOCATION_ALIAS
- tblBore_Hole (MOE WWDB; MOEDB)

Views

- V_SYS_MOE_LOCATIONS

Estimated Recurrence Time: After MOE WWDB import.

Upon import of the latest version of the MOE WWDB (refer to Section G.10), a check should be made of any new mapping between a BORE_HOLE_ID (in the MOE database; MOEDB) and a LOC_ID (in the ORMGP database; ORMGPDB). Note that this will only apply in the case of a 1:1 relationship a single WELL_ID matching a single BORE_HOLE_ID in the MOEDB; in the ORMGPDB, a single WELL_ID maps to a single LOC_ID.

This is first tested using the following script

```
select
v.loc_id
,t3.well_id
,t3.bore_hole_id
,v.moe_bore_hole_id
--count(*) as rcount
from
(
select
cast(t2.well_id as int) as well_id
,b2.bore_hole_id
from
(
```

```

select
t.well_id
from
(
select
b.well_id
,count(*) as bhi_num
from
moe_20190509.dbo.tblbore_hole as b
group by
well_id
) as t
where
t.bhi_num= 1
) as t2
inner join moe_20190509.dbo.tblbore_hole as b2
on t2.well_id=b2.well_id
) as t3
inner join oak_20160831_master.dbo.v_sys_moe_locations as v
on t3.well_id=v.moe_well_id
where
v.moe_bore_hole_id is null

```

Any NULL values occurring in the MOE_BORE_HOLE_ID field (or any rows returned) indicate that the BORE_HOLE_ID needs to be incorporated in the D_LOCATION_ALIAS table, as follows

```

insert into d_location_alias
(
loc_id, loc_name_alias, loc_alias_type_code, data_id, sys_temp1, sys_temp2
)
select
v.loc_id
,cast(t3.bore_hole_id as varchar(255)) as loc_name_alias
,3 as loc_alias_type_code
,521 as data_id
,'20190530a' as sys_temp1
,20190530 as sys_temp2
from
(
select
cast(t2.well_id as int) as well_id
,b2.bore_hole_id
from
(
select
t.well_id
from
(
select
b.well_id
,count(*) as bhi_num
from
moe_20190509.dbo.tblbore_hole as b
group by
well_id
) as t
where
t.bhi_num= 1
) as t2
inner join moe_20190509.dbo.tblbore_hole as b2
on t2.well_id=b2.well_id
) as t3
inner join oak_20160831_master.dbo.v_sys_moe_locations as v
on t3.well_id=v.moe_well_id
where
v.moe_bore_hole_id is null

```

This process should be undertaken before examining locations for new data from the latest MOEDB. Note the inclusion of the DATA_ID (which should correspond to the MOEDB under examination) and SYS_TEMP1/SYS_TEMP2 (making them available for any subsequent checks).

G.30 Update Locations from MOE WWDB

Tables (OAK_20160831_MASTER compatible)

- D_BOREHOLE
- D_BOREHOLE_CONSTRUCTION
- D_DATA_SOURCE
- D_GEOLOGY_FEATURE
- D_GEOLOGY_LAYER
- D_INTERVAL
- D_INTERVAL_MONITOR
- D_INTERVAL_REF_ELEV
- D_INTERVAL_TEMPORAL_2
- D_LOCATION
- D_LOCATION_ALIAS
- D_LOCATION_ELEV
- D_LOCATION_PURPOSE
- D_LOCATION_QA
- D_PUMPTEST
- D_PUMPTEST_STEP

Tables (MOE Water Well Database, MOE_20190509)

- TblBore_Hole
- TblCasing
- TblFormation
- TblHole
- TblMethod_Construction
- TblPipe
- TblPlug
- TblPump_Test
- TblPump_Test_Detail
- TblScreen
- TblWater
- TblWWR

Estimated Recurrence Time: After MOE WWDB import.

After the MOE WWDB (MOEDB) has been imported, a check should be made as to whether any locations currently present in the master database (ORMGPDB) have been

updated in the MOEDB. We'll determine this on a table-by-table basis. Many of the methods and queries used herein are adapted from Section G.10.

G.30.1 D_GEOLOGY_LAYER

Here we are looking at those locations that do not have any records present in D_GEOLOGY_LAYER. We cannot rely upon differences in record counts between this table and [tblFormation] (in the MOEDB) as geologic layer corrections are made in the ORMGPDB. We will create a table to be used as a reference to those LOC_IDs for which we will extract information from [tblFormation].

```
select
t.*
,t2.rcount_moe
--count(*) as rcount
into moe_20190509.dbo ORMGP_20190509_upd_DGL
from
(
select
dbore.loc_id
,v.moe_well_id
,v.moe_bore_hole_id
,dgl.rcount
,dloc.loc_type_code
from
oak_20160831_master.dbo.d_borehole as dbore
inner join oak_20160831_master.dbo.d_location as dloc
on dbore.loc_id=dloc.loc_id
inner join oak_20160831_master.dbo.d_location_qa as dlqa
on dbore.loc_id=dlqa.loc_id
inner join oak_20160831_master.dbo.v_sys_agency_ypdt as yc
on dbore.loc_id=yc.loc_id
left outer join
(
select
loc_id
,count(*) as rcount
from
oak_20160831_master.dbo.d_geology_layer
group by
loc_id
) as dgl
on dbore.loc_id=dgl.loc_id
inner join oak_20160831_master.dbo.v_sys_moe_locations as v
on dbore.loc_id=v.loc_id
where
dgl.rcount is null
and dloc.loc_type_code=1
and dlqa.qa_coord_confidence_code<>117
) as t
inner join
(
select
m.bore_hole_id
,count(*) as rcount_moe
from
moe_20190509.dbo.tblformation as m
where
m.mat1 is not null
and cast(m.mat1 as int)<>0
group by
bore_hole_id
) as t2
on t.moe_bore_hole_id=t2.bore_hole_id
```

Script: G_30_01_01_DGL_BORE_HOLE_ID.sql

Now that we have a list of locations which require updating, make sure that their material codes in the MOEDB are not specified as 'Other' (i.e. '27'); make them 'Unknown' (i.e. '0') instead. This needs to be accomplished for each of the three material fields, only the first is shown here.

```
update MOE_20190509.dbo.TblFormation
set
MAT1=0
from
MOE_20190509.dbo.TblFormation as mf
inner join ORMGP_20190509_upd_DGL as od
on mf.bore_hole_id=od.moe_bore_hole_id
where
mf.MAT1=27
```

Script: G_30_01_02_DGL_MATS.sql

Check that the depth units are one of 'ft' or 'm' – modify others as necessary (not shown).

```
select
od.LOC_ID
,mf.*
from
MOE_20190509.dbo.TblFormation as mf
inner join ORMGP_20190509_upd_DGL as od
on mf.bore_hole_id=od.moe_bore_hole_id
where
not(mf.FORMATION_END_DEPTH_UOM in ('ft','m'))
order by
od.loc_id,mf.formation_top_depth
```

Script: G_30_01_03_DGL_UNITS.sql

We can now create the temporary file containing the new MOEDB information, formatted to match D_GEOLOGY_LAYER. Note that we are including values for DATA_ID, SYS_TEMP1 and SYS_TEMP2 for subsequent tracking purposes.

```
select
od.LOC_ID
,cast( null as int ) as GEOL_ID
,cast( case
when moef.COLOR is null or moef.COLOR=0 then null
else moef.COLOR
end as int ) as [GEOL_MAT_COLOUR_CODE]
,case
when moef.FORMATION_END_DEPTH_UOM='m' then ( dbore.bh_gnd_elev - moef.formation_top_depth )
else ( dbore.bh_gnd_elev - ( moef.formation_top_depth * 0.3048 ) )
end as GEOL_TOP_ELEV
,case
when moef.FORMATION_END_DEPTH_UOM='m' then ( dbore.bh_gnd_elev - moef.formation_end_depth )
else ( dbore.bh_gnd_elev - ( moef.formation_end_depth * 0.3048 ) )
end as GEOL_BOT_ELEV
,moef.FORMATION_TOP_DEPTH as GEOL_TOP_OUOM
,moef.FORMATION_END_DEPTH as GEOL_BOT_OUOM
,cast(moef.FORMATION_END_DEPTH_UOM as varchar(20)) as GEOL_UNIT_OUOM
```

```

.moef.LAYER as GEOL_MOE_LAYER
.cast( case
-- check if all mat fields are null
when moef.MAT1 is null and moef.MAT2 is null and moef.MAT3 is null then 0
-- if mat1 is null but mat2 is not, make mat1=mat2
when moef.MAT1 is null and moef.MAT2 is not null then moef.MAT2
-- if mat1 and mat2 is null but mat3 is not, make mat1=mat3
when moef.MAT1 is null and moef.MAT2 is null and moef.MAT3 is not null then moef.MAT3
else moef.MAT1
end as int) as GEOL_MAT1_CODE
.cast( case
-- if all mat fields are null, leave mat2 as is
-- if mat1 is null, mat2 has been moved to mat1, return null
-- we won't move mat3 to mat2
when moef.MAT1 is null and moef.MAT2 is not null then null
else moef.MAT2
end as int) as GEOL_MAT2_CODE
.cast( case
-- if mat1 and mat2 is null but mat3 is not, make mat1=mat3
when moef.MAT1 is null and moef.MAT2 is null and moef.MAT3 is not null then null
else moef.MAT3
end as int) as GEOL_MAT3_CODE
-- include some comments if we've messed about with the mat codes
.cast( case
when moef.MAT1 is null and moef.MAT2 is null and moef.MAT3 is null then 'No material, assigned unknown'
when moef.MAT1 is null and moef.MAT2 is not null then 'No mat1, assigned mat2 to mat1'
when moef.MAT1 is null and moef.MAT2 is null and moef.MAT3 is not null then 'No mat1 or mat2, assigned mat3 to mat1'
else null
end as varchar(255)) as GEOL_COMMENT
.cast( 521 as int ) as DATA_ID
,'20190531a' as SYS_TEMP1
,20190531 as SYS_TEMP2
.row_number() over (order by od.loc_id) as rkey
into O_D_GEOLOGY_LAYER
from
ORMGP_20190509_upd_DGL as od
inner join MOE_20190509.dbo.TblFormation as moef
on od.moe_bore_hole_id=moef.bore_hole_id
inner join oak_20160831_master.dbo.d_borehole as dbore
on od.loc_id=dbore.loc_id

```

We will now create the new GEOL_IDs necessary for import.

```

update moe_20190509.dbo.o_d_geology_layer
set
geol_id=t2.geol_id
from
moe_20190509.dbo.o_d_geology_layer as od
inner join
(
select
t.geol_id
,row_number() over (order by t.geol_id) as rkey
from
(
select
top 5000
v.new_id as geol_id
from
oak_20160831_master.dbo.v_sys_random_id_bulk_001 as v
where
v.new_id not in
(select geol_id from oak_20160831_master.dbo.d_geology_layer )
) as t
) as t2
on od.rkey=t2.rkey

```

And, finally, insert the new MOEDB information.

```
insert into oak_20160831_master.dbo.d_geology_layer
(
[LOC_ID],
[GEOL_ID],
[GEOL_MAT_COLOUR_CODE],
[GEOL_TOP_ELEV],
[GEOL_BOT_ELEV],
[GEOL_TOP_OUOM],
[GEOL_BOT_OUOM],
[GEOL_UNIT_OUOM],
[GEOL_MOE_LAYER],
[GEOL_MAT1_CODE],
[GEOL_MAT2_CODE],
[GEOL_MAT3_CODE],
[GEOL_COMMENT],
[DATA_ID],
[SYS_TEMP1],
[SYS_TEMP2]
)
select
[LOC_ID],
[GEOL_ID],
[GEOL_MAT_COLOUR_CODE],
[GEOL_TOP_ELEV],
[GEOL_BOT_ELEV],
[GEOL_TOP_OUOM],
[GEOL_BOT_OUOM],
[GEOL_UNIT_OUOM],
[GEOL_MOE_LAYER],
[GEOL_MAT1_CODE],
[GEOL_MAT2_CODE],
[GEOL_MAT3_CODE],
[GEOL_COMMENT],
[DATA_ID],
[SYS_TEMP1],
[SYS_TEMP2]
from
moe_20190509.dbo.o_d_geology_layer
```

Script: G_30_01_04_DGL_ADD.sql

G.30.2 D_GEOLOGY_FEATURE

Here we are looking at those locations that do not have any records present in D_GEOLOGY_FEATURE. Similar to the process for D_GEOLOGY_LAYER, we shouldn't rely upon differences in record counts between the MOEDB and ORMDB. We'll (again) create a table to be used as a reference to those LOC_IDs for which we will extract information.

```
select
t.*
,t2.rcount_moe
--count(*) as rcount
into moe_20190509_final.dbo.ORMGP_20190509_upd_DGF
from
(
select
dbore.loc_id
,v.moe_well_id
,v.moe_bore_hole_id
,dgl.rcount
```

```

from
oak_20160831_master.dbo.d_borehole as dbore
inner join oak_20160831_master.dbo.d_location as dloc
on dbore.loc_id=dloc.loc_id
inner join oak_20160831_master.dbo.d_location_qa as dlqa
on dbore.loc_id=dlqa.loc_id
inner join oak_20160831_master.dbo.v_sys_agency_ypdt as yc
on dbore.loc_id=yc.loc_id
left outer join
(
select
loc_id
,count(*) as rcount
from
oak_20160831_master.dbo.d_geology_feature
group by
loc_id
) as dgl
on dbore.loc_id=dgl.loc_id
inner join oak_20160831_master.dbo.v_sys_moe_locations as v
on dbore.loc_id=v.loc_id
where
dgl.rcount is null
and dloc.loc_type_code=1
and dlqa.qa_coord_confidence_code<>117
and v.moe_bore_hole_id is not null
) as t
inner join
(
select
moep.bore_hole_id
,count(*) as rcount_moe
from
MOE_20190509.dbo.TblPipe as moep
inner join MOE_20190509.[dbo].[TblWater] as moew
on moep.PIPE_ID=moew.PIPE_ID
group by
moep.bore_hole_id
) as t2
on t.moe_bore_hole_id=t2.bore_hole_id

```

Script: G_30_02_01_DGF_BORE_HOLE_ID.sql

We can then create the base table that will be used for import into the ORMDB.

```

select
y.LOC_ID
,cast(null as int) as FEATURE_ID
,case
when moew.kind is null or moew.kind=0 then null -- not specified
else moew.kind -- matches YC codes
end
as [FEATURE_CODE]
,'Water Found' as [FEATURE_DESCRIPTION]
,moew.WATER_FOUND_DEPTH as [FEATURE_TOP_OUOM]
,moew.WATER_FOUND_DEPTH_UOM as [FEATURE_UNIT_OUOM]
,cast(521 as int) as DATA_ID
,ROW_NUMBER() over (order by y.LOC_ID) as rkey
into MOE_20190509.dbo.O_D_GEOLOGY_FEATURE
from
MOE_20190509.dbo.ORMGP_20190509_upd_DGF as y
inner join MOE_20190509.dbo.TblPipe as moep
on y.moe_bore_hole_id=moep.Bore_Hole_ID
inner join MOE_20190509.[dbo].[TblWater] as moew
on moep.PIPE_ID=moew.PIPE_ID

```

Update the FEATURE_ID field.

```
update moe_20190509.dbo.o_d_geology_feature
set
feature_id=t.feature_id
from
moe_20190509.dbo.o_d_geology_feature as od
inner join
(
select
t.feature_id
,row_number() over (order by t.feature_id) as rkey
from
(
select
top 10000
v.new_id as feature_id
from
oak_20160831_master.dbo.v_sys_random_id_bulk_001 as v
where
v.new_id not in
( select feature_id from oak_20160831_master.dbo.d_geology_feature )
) as t
) as t2
on od.rkey=t2.rkey
```

And, finally, insert the records into D_GEOLOGY_FEATURE.

```
insert into oak_20160831_master.dbo.d_geology_feature
(
[LOC_ID],
[FEATURE_ID],
[FEATURE_CODE],
[FEATURE_DESCRIPTION],
[FEATURE_TOP_OUOM],
[FEATURE_UNIT_OUOM],
[DATA_ID],
SYS_TEMP1,
SYS_TEMP2
)
select
[LOC_ID],
[FEATURE_ID],
[FEATURE_CODE],
[FEATURE_DESCRIPTION],
[FEATURE_TOP_OUOM],
[FEATURE_UNIT_OUOM],
[DATA_ID],
cast( '20190509a' as varchar(255) ) as SYS_TEMP1,
cast( 20190509 as int ) as SYS_TEMP2
from
moe_20190509.dbo.o_d_geology_feature
```

Script: G_30_02_02_DGF_ADD.sql

G.30.3 D_BOREHOLE_CONSTRUCTION

We will now examine those MOE locations present in the ORMDB that do not have any borehole construction details. Determine these locations and create a temporary reference table (for speed/access purposes). Index the resultant table-fields (i.e. the identifier fields) to enhance access. Note that there may be instances of multiple

BORE_HOLE_IDs present for a single LOC_ID. These seem to be decommissioned or alteration records. Refer also to Section G.10.10 for additional details.

```

select
dbore.loc_id
,v.moe_bore_hole_id
--into moe_20190509_final.dbo ORMGP_20190509_base_DBHCONS
from
oak_20160831_master.dbo.d_borehole as dbore
inner join oak_20160831_master.dbo.d_location as dloc
on dbore.loc_id=dloc.loc_id
inner join oak_20160831_master.dbo.d_location_qa as dlqa
on dbore.loc_id=dlqa.loc_id
inner join oak_20160831_master.dbo.v_sys_agency_ypdt as yc
on dbore.loc_id=yc.loc_id
left outer join
(
select
loc_id
,count(*) as rcount
from
oak_20160831_master.dbo.d_borehole_construction as dbc
inner join oak_20160831_master.dbo.d_borehole as dbore
on dbc.bh_id=dbore.bh_id
group by
dbore.loc_id
) as d
on dbore.loc_id=d.loc_id
inner join oak_20160831_master.dbo.v_sys_moe_locations as v
on dbore.loc_id=v.loc_id
where
d.rcount is null
and dloc.loc_type_code=1
and dlqa.qa_coord_confidence_code<>117
and v.moe_bore_hole_id is not null
group by
dbore.loc_id,v.moe_bore_hole_id

```

Assemble any casing information for these locations.

```

select
-- note that we're using BORE_HOLE_ID as a temporary BH_ID
y.LOC_ID
,y.moe_BORE_HOLE_ID as BH_ID
,case
when moec.MATERIAL is null then 10 -- unknown
when moec.MATERIAL = 1 then 21 -- steel casing
when moec.MATERIAL = 2 then 16 -- galvanized
when moec.MATERIAL = 3 then 23 -- concrete
when moec.MATERIAL = 4 then 24 -- open hole
when moec.MATERIAL = 5 then 25 -- plastic
when moec.MATERIAL = 6 then 32 -- fibreglass
end
as [CON_SUBTYPE_CODE]
,moec.DEPTH_FROM as [CON_TOP_OUOM]
,moec.DEPTH_TO as [CON_BOT_OUOM]
,moec.CASING_DEPTH_UOM as [CON_UNIT_OUOM]
,moec.CASING_DIAMETER as [CON_DIAMETER_OUOM]
,moec.CASING_DIAMETER_UOM as [CON_DIAMETER_UNIT_OUOM]
,convert(varchar(255),null) as CON_COMMENT
--into MOE_20190509_final.dbo ORMGP_20190509_upd_DBHCONS
from
MOE_20190509_final.dbo ORMGP_20190509_base_DBHCONS as y
inner join MOE_20190509_final.dbo.TblPipe as moep
on y.moe_BORE_HOLE_ID=moep.Bore_Hole_ID
inner join MOE_20190509_final.dbo.TblCasing as moec
on moep.PIPE_ID=moec.PIPE_ID

```

```

where
not
(
moec.DEPTH_TO is null
and moec.CASING_DEPTH_UOM is null
and moec.CASING_DIAMETER is null
and moec.CASING_DIAMETER_UOM is null
)

```

Add any plug information.

```

insert into MOE_20190509_final.dbo.ORMGP_20190509_upd_DBHCONS
(LOC_ID,BH_ID,CON_SUBTYPE_CODE,CON_TOP_OUOM,CON_BOT_OUOM,CON_UNIT_OUOM)
select
-- note that we're using BORE_HOLE_ID as a temporary BH_ID
y.LOC_ID
,y.moe_BORE_HOLE_ID as BH_ID
,31 as CON_SUBTYPE_CODE
,moep.PLUG_FROM as [CON_TOP_OUOM]
,moep.PLUG_TO as [CON_BOT_OUOM]
,moep.PLUG_DEPTH_UOM as [CON_UNIT_OUOM]
from
MOE_20190509_final.dbo.ORMGP_20190509_base_DBHCONS as y
inner join MOE_20190509_final.dbo.TbPlug as moep
on y.moe_BORE_HOLE_ID=moep.BORE_HOLE_ID
where
not
(
moep.PLUG_FROM is null
and moep.PLUG_TO is null
and moep.PLUG_DEPTH_UOM is null
)

```

Swap the top and bottom depths if the former is larger than the second (not shown). Create the final, formatted table including the updated DATA_ID and a set of tags for SYS_TEMP1 and SYS_TEMP2 (for reference). Pull the actual BH_ID for the location from the ORMDB. Note that we're introducing a counter for later assignment of random identifiers.

```

SELECT
d.[BH_ID]
,[CON_SUBTYPE_CODE]
,[CON_TOP_OUOM]
,[CON_BOT_OUOM]
,[CON_UNIT_OUOM]
,[CON_DIAMETER_OUOM]
,[CON_DIAMETER_UNIT_OUOM]
,[CON_COMMENT]
,cast( 521 as int ) as DATA_ID
,cast(null as int) as SYS_RECORD_ID
,cast( '20190509a' as varchar(255) ) as SYS_TEMP1
,cast( 20190509 as int ) as SYS_TEMP2
,ROW_NUMBER() over (order by y.BH_ID) as rkey
,y.bh_id as moe_bore_hole_id
--into moe_20190509_final.dbo.O_D_BOREHOLE_CONSTRUCTION
FROM
MOE_20190509_final.[dbo].ORMGP_20190509_upd_DBHCONS as y
inner join oak_20160831_master.dbo.d_borehole as d
on y.loc_id=d.loc_id

```

Script: G_30_03_01_DBHCONS.sql

Update the SYS_RECORD_ID and insert the final information into D_BOREHOLE_CONSTRUCTION in the ORMDB (not shown).

G.30.4 Depths

In order to process the screen information (as found in the next section, G.30.5) for some of the screen types, we'll need to determine the depth of the borehole in question. Similar to the methodology in Section G.10, we'll look at tables for each of: formation; water found; borehole construction; MOE assigned. From these, we'll determine the maximum depth.

First, we'll assemble all those locations in the ORMDB that currently have no depth associated with them as found in D_BOREHOLE. This is irregardless of whether they have updated depths in the MOEDB.

```
select
dbore.loc_id
,v.moe_bore_hole_id
,cast(null as float) as fm_max_depth_m
,cast(null as float) as dgf_max_depth_m
,cast(null as float) as casing_max_depth_m
,cast(null as float) as dbc_max_depth_m
,cast(null as float) as moe_max_depth_m
,cast(null as float) as max_depth_m
into moe_20200721.dbo.ORMGP_20200721_upd_DEPTH
from
oak_20160831_master.dbo.d_borehole as dbore
inner join oak_20160831_master.dbo.d_location as dloc
on dbore.loc_id=dloc.loc_id
inner join oak_20160831_master.dbo.d_location_qa as dlqa
on dbore.loc_id=dlqa.loc_id
inner join oak_20160831_master.dbo.v_sys_agency_ypdt as yc
on dbore.loc_id=yc.loc_id
inner join oak_20160831_master.dbo.v_sys_moe_locations as v
on dbore.loc_id=v.loc_id
where
dbore.bh_bottom_ouom is null
and dloc.loc_type_code=1
and dlqa.qa_coord_confidence_code<>117
and v.moe_bore_hole_id is not null
group by
dbore.loc_id,v.moe_bore_hole_id
```

Note that we're creating a series of NULL fields which we will subsequently populate. Now, determine the maximum formation depth.

```
update moe_20200721.dbo.ormgp_20200721_upd_depth
set
fm_max_depth_m= t2.fm_max_depth_m
from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm2
inner join
(
select
t.moe_bore_hole_id
,max(t.fm_end_depth) as fm_max_depth_m
from
(
select
orm.moe_bore_hole_id
```

```

.case
when moe.[FORMATION_END_DEPTH_UOM]='ft' then moe.[FORMATION_END_DEPTH]*0.3048
else moe.[FORMATION_END_DEPTH]
end as fm_end_depth
from
moe_20200721.dbo.ORMGP_20200721_upd_DEPTH as orm
inner join moe_20200721.dbo.tblformation as moe
on orm.moe_bore_hole_id=moe.bore_hole_id
) as t
group by
t.moe_bore_hole_id
) as t2
on orm2.moe_bore_hole_id=t2.moe_bore_hole_id

```

Check the casing for inverted top- and bottom-depths.

```

update moe_20200721.dbo.tblcasing
set
depth_from=depth_to
,depth_to=depth_from
where
depth_from>depth_to

```

Then process the various tables for construction details/depths.

```

update moe_20200721.dbo.ormgp_20200721_upd_depth
set
dbc_max_depth_m= t2.dbc_max_depth_m
from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm
inner join
(
select
t.moe_bore_hole_id
,max(t.dbc_depth) as dbc_max_depth_m
from
(
select
orm.moe_bore_hole_id
.case
when mcase.casing_depth_uom = 'ft' then mcase.depth_to * 0.3048
else mcase.depth_to
end as dbc_depth
from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm
inner join moe_20200721.dbo.tblbore_hole as mbore
on orm.moe_bore_hole_id=mbore.bore_hole_id
inner join moe_20200721.dbo.tblcasing as mcase
on mbore.well_id=mcase.well_id
union all
select
orm.moe_bore_hole_id
.case
when mplug.plug_depth_uom = 'ft' then mplug.plug_to * 0.3048
else mplug.plug_to
end as dbc_depth
from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm
inner join moe_20200721.dbo.tblplug as mplug
on orm.moe_bore_hole_id=mplug.bore_hole_id
union all
select
orm.moe_bore_hole_id
.case
when mscr.scrn_depth_uom = 'ft' then mscr.scrn_end_depth * 0.3048
else mscr.scrn_end_depth
end as dbc_depth

```

```

from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm
inner join moe_20200721.dbo.tblbore_hole as mbore
on orm.moe_bore_hole_id=mbore.bore_hole_id
inner join moe_20200721.dbo.tblscreen as mscre
on mbore.well_id=mscre.well_id
union all
select
orm.moe_bore_hole_id
,case
when mpump.levels_uom = 'ft' then mpump.recom_depth * 0.3048
else mpump.recom_depth
end as dbc_depth
from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm
inner join moe_20200721.dbo.tblbore_hole as mbore
on orm.moe_bore_hole_id=mbore.bore_hole_id
inner join moe_20200721.dbo.tblpump_test as mpump
on mbore.well_id=mpump.well_id
) as t
group by
t.moe_bore_hole_id
) as t2
on orm.moe_bore_hole_id=t2.moe_bore_hole_id

```

We'll repeat part of this query in order to process the casing depths separately.

```

update moe_20200721.dbo.ormgp_20200721_upd_depth
set
casing_max_depth_m= t2.casing_max_depth_m
from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm
inner join
(
select
t.moe_bore_hole_id
,max(t.dbc_depth) as casing_max_depth_m
from
(
select
orm.moe_bore_hole_id
,case
when mcase.casing_depth_uom = 'ft' then mcase.depth_to * 0.3048
else mcase.depth_to
end as dbc_depth
from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm
inner join moe_20200721.dbo.tblbore_hole as mbore
on orm.moe_bore_hole_id=mbore.bore_hole_id
inner join moe_20200721.dbo.tblcasing as mcase
on mbore.well_id=mcase.well_id
) as t
group by
t.moe_bore_hole_id
) as t2
on orm.moe_bore_hole_id=t2.moe_bore_hole_id

```

Populate the geology features (i.e. water found) depths.

```

update moe_20200721.dbo.ormgp_20200721_upd_depth
set
dgr_max_depth_m= t2.dgr_max_depth_m
from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm
inner join
(
select

```

```

t.moe_bore_hole_id
,max(t.dgf_depth) as dgf_max_depth_m
from
(
select
orm.moe_bore_hole_id
.case
when mwater.water_found_depth_uom = 'ft' then mwater.water_found_depth * 0.3048
else mwater.water_found_depth
end as dgf_depth
from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm
inner join moe_20200721.dbo.tblbore_hole as mbore
on orm.moe_bore_hole_id=mbore.bore_hole_id
inner join moe_20200721.dbo.tblwater as mwater
on mbore.well_id=mwater.well_id
) as t
group by
t.moe_bore_hole_id
) as t2
on orm.moe_bore_hole_id=t2.moe_bore_hole_id

```

And then, finally, the maximum depth as specified by the MOE.

```

update moe_20190509_final.dbo.ormgp_20190509_upd_depth
set
moe_max_depth_m= t2.moe_max_depth_m
from
moe_20190509_final.dbo.ormgp_20190509_upd_depth as orm
inner join
(
select
t.moe_bore_hole_id
,max(t.moe_depth) as moe_max_depth_m
from
(
select
orm.moe_bore_hole_id
.case
when mhole.hole_depth_uom = 'ft' then mhole.depth_to * 0.3048
else mhole.depth_to
end as moe_depth
from
moe_20190509_final.dbo.ormgp_20190509_upd_depth as orm
inner join moe_20190509_final.dbo.tblhole as mhole
on orm.moe_bore_hole_id=mhole.bore_hole_id
) as t
group by
t.moe_bore_hole_id
) as t2
on orm.moe_bore_hole_id=t2.moe_bore_hole_id

```

We can now determine the maximum depth based upon the values of these four fields.

```

update moe_20200721.dbo.ormgp_20200721_upd_depth
set
max_depth_m= t.max_depth_m
from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm
inner join
(
select
orm.moe_bore_hole_id
.(
select max(v) from
(
values (fm_max_depth_m),(dgf_max_depth_m),(dbc_max_depth_m),(moe_max_depth_m)

```

```

) as value(v)
) as max_depth_m
from
moe_20200721.dbo.ormgp_20200721_upd_depth as orm
) as t
on orm.moe_bore_hole_id=t.moe_bore_hole_id

```

Script: G_30_04_01_DEPTHS.sql

This can now be used as a reference (value) in the following section.

G.30.5 D_INTERVAL and Associated

We will now examine those MOE locations present in the ORMDB that do not currently have screens identified. Similar to Section G.10, the methodology will be adapted based upon the screen type (e.g. reported, open hole, etc...) as each of these require slightly differing steps.

We'll determine those locations already present in the ORMDB and create a temporary table – this is for speed purposes. The alternative, on-the-fly analysis, takes a significant time. Note that the resulting table will subsequently be indexed by hand (i.e. the identifier fields), again for speed purposes. We're including locations/screens that are already present in the ORMDB D_INTERVAL table which have a screen type of 'Screen Information Omitted' (INT_TYPE_CODE 28; these were imported from an earlier MOEDB but did not have screen information at that time). A non-null INT_ID identifies these locations.

```

select
t.*
,dint.int_id
into moe_20200721.dbo.ORMGP_20200721_base_DINT
from
(
select
dbore.loc_id
,v.moe_bore_hole_id
from
oak_20160831_master.dbo.d_borehole as dbore
inner join oak_20160831_master.dbo.d_location as dloc
on dbore.loc_id=dloc.loc_id
inner join oak_20160831_master.dbo.d_location_qa as dlqa
on dbore.loc_id=dlqa.loc_id
inner join oak_20160831_master.dbo.v_sys_agency_ypdt as yc
on dbore.loc_id=yc.loc_id
inner join oak_20160831_master.dbo.v_sys_moe_locations as v
on dbore.loc_id=v.loc_id
left outer join oak_20160831_master.dbo.d_interval as dint
on dbore.loc_id=dint.loc_id
where
(dint.int_id is null or dint.int_type_code=28)
and dloc.loc_type_code=1
and dlqa.qa_coord_confidence_code<>117
and v.moe_bore_hole_id is not null
group by
dbore.loc_id,v.moe_bore_hole_id
) as t
left outer join oak_20160831_master.dbo.d_interval as dint
on t.loc_id=dint.loc_id

```

Now that we have a listing of the locations and their associated BORE_HOLE_ID, determine which of these actually has a reported screen. We'll create the initial compatible table from which we'll generate D_INTERVAL and D_INTERVAL_MONITOR.

```
select
d.LOC_ID
,d.INT_ID
,d.moe_bore_hole_id as TMP_INT_ID
,18 as tmp_INT_TYPE_CODE
,moeslot.CONV_YC_SLOT as MON_SCREEN_SLOT
,moes.SCRN_MATERIAL as MON_SCREEN_MATERIAL
,moes.SCRN_DIAMETER as MON_DIAMETER_OUOM
,moes.SCRN_DIAMETER_UOM as MON_DIAMETER_UNIT_OUOM
,moes.SCRN_TOP_DEPTH as MON_TOP_OUOM
,moes.SCRN_END_DEPTH as MON_BOT_OUOM
,moes.SCRN_DEPTH_UOM as MON_UNIT_OUOM
,cast(null as varchar(255)) as MON_COMMENT
into moe_20200721.dbo.ORMGP_20200721_upd_DINTMON
from
moe_20200721.dbo.ORMGP_20200721_base_DINT as d
inner join
MOE_20200721.dbo.TblPipe as moep
on d.moe_bore_hole_id=moep.Bore_Hole_ID
inner join
MOE_20200721.dbo.TblScreen as moes
on moep.PIPE_ID=moes.PIPE_ID
left outer join
MOE_20200721.dbo.YC_20200721_MOE_SLOT as moeslot
on moes.Slot=moeslot.MOE_SLOT
where
moes.SCRN_TOP_DEPTH is not null
or moes.SCRN_END_DEPTH is not null
```

Script: G_30_05_01_DINT_BORE_HOLE_ID.sql

We'll now check whether we have a non-reported screen, in this case an OPEN_HOLE (as declared in the MOEDB; bottom-of-casing to bottom-of-hole). Add these to the temporary table ORMGP_20190509_upd_DINTMON.

```
insert into moe_20200721.dbo.ormgp_20200721_upd_dintmon
(
loc_id,tmp_int_id,tmp_int_type_code,mon_top_ouom,mon_bot_ouom,mon_unit_ouom,mon_comment
)
select
orm.LOC_ID
,orm.moe_bore_hole_id as tmp_INT_ID
,cast( 21 as int ) as tmp_INT_TYPE_CODE
,od.casing_max_depth_m as MON_TOP_OUOM
,od.max_depth_m as MON_BOT_OUOM
,cast('m' as varchar(50)) as MON_UNIT_OUOM
,cast('open hole; bottom-of-casing to bottom-of-hole' as varchar(255)) as MON_COMMENT
from
moe_20200721.dbo.ormgp_20200721_base_dint as orm
inner join moe_20200721.dbo.ormgp_20200721_upd_depth as od
on orm.moe_bore_hole_id=od.moe_bore_hole_id
inner join moe_20200721.dbo.tblbore_hole as moebh
on orm.moe_bore_hole_id=moebh.bore_hole_id
where
moebh.OPEN_HOLE='Y'
and orm.moe_bore_hole_id not in
( select tmp_int_id from moe_20200721.dbo.ormgp_20200721_upd_dintmon )
```

If there are bedrock boreholes (and they haven't previously been assigned a screen), we can make the assumption that the screen is from the bottom-of-casing (penetrating to bedrock) to the bottom of the hole. Add these (note that this query can take some time to complete).

```

insert into moe_20200721.dbo.ormgp_20200721_upd_dintmon
(
loc_id, int_id, tmp_INT_ID, tmp_int_type_code, mon_top_ouom, mon_bot_ouom, mon_unit_ouom, mon_comment
)
select
orm.LOC_ID
,orm.INT_ID
,orm.moe_bore_hole_id as tmp_INT_ID
,cast( 22 as int ) as tmp_INT_TYPE_CODE
,t2.geol_top_m as MON_TOP_OUOM
,od.max_depth_m as MON_BOT_OUOM
,'m' as MON_UNIT_OUOM
,cast( 'bedrock, no valid casing; open hole, top-of-bedrock to bottom-of-hole' as varchar(255) ) as MON_COMMENT
from
moe_20200721.dbo.ormgp_20200721_base_dint as orm
inner join moe_20200721.dbo.ormgp_20200721_upd_depth as od
on orm.moe_bore_hole_id=od.moe_bore_hole_id
inner join
(
select
t.moe_bore_hole_id
,min(t.geol_top_m) as geol_top_m
,max(t.geol_bot_m) as geol_bot_m
from
(
select
orm.moe_bore_hole_id
,case
when mform.formation_end_depth_uom = 'ft' then mform.formation_top_depth * 0.3048
else mform.formation_top_depth
end as geol_top_m
,case
when mform.formation_end_depth_uom = 'ft' then mform.formation_end_depth * 0.3048
else mform.formation_end_depth
end as geol_bot_m
from
moe_20200721.dbo.ormgp_20200721_base_dint as orm
inner join moe_20200721.dbo.tblformation as mform
on orm.moe_bore_hole_id=mform.bore_hole_id
inner join oak_20160831_master.dbo.r_geol_mat1_code as rgmc
on cast( mform.mat1 as int )=rgmc.geol_mat1_code
where
rgmc.geol_mat1_rock=1
) as t
group by
t.moe_bore_hole_id
) as t2
on orm.moe_bore_hole_id=t2.moe_bore_hole_id
where
orm.moe_bore_hole_id not in
( select tmp_int_id from moe_20200721.dbo.ormgp_20200721_upd_dintmon )

```

The remainder of the (possible) screens should be considered overburden boreholes. As such, we'll assume a '0.3m' screen above the bottom-of-hole. Only those with valid depths are considered.

```

insert into moe_20200721.dbo.ormgp_20200721_upd_dintmon
(
loc_id, int_id, tmp_int_id, tmp_int_type_code, mon_top_ouom, mon_bot_ouom, mon_unit_ouom, mon_comment
)

```

```

select
orm.LOC_ID
,orm.INT_ID
,orm.moe_bore_hole_id as tmp_INT_ID
,19 as tmp_INT_TYPE_CODE
,(od.max_depth_m - 0.3 ) as MON_TOP_OUOM
.od.max_depth_m as MON_BOT_OUOM
.cast( 'm' as varchar(50) ) as MON_UNIT_OUOM
.cast( 'overburden; assumed screen, 0.3m above bottom-of-hole' as varchar(255) ) as MON_COMMENT
from
moe_20200721.dbo.ormgp_20200721_base_dint as orm
inner join moe_20200721.dbo.ormgp_20200721_upd_depth as od
on orm.moe_bore_hole_id=od.moe_bore_hole_id
where
od.max_depth_m is not null
and orm.moe_bore_hole_id not in
( select tmp_int_id from moe_20200721.dbo.ormgp_20200721_upd_dintmon )

```

Script: G_30_05_02_DINT_OTHER.sql

Make sure the ORMGP_20190509_upd_DINTMON has an indexed key (this may not be necessary – try running the ‘select’ statement before creating a new table to test the run time). We can now create the D_INTERVAL and D_INTERVAL_MONITOR tables, the latter table should be assembled first (update the DATA_ID, SYS_TEMP1 and SYS_TEMP2 fields).

```

select
dim.LOC_ID
,dim.tmp_INT_ID
,dim.tmp_INT_TYPE_CODE
,dim.INT_ID
,dim.MON_SCREEN_SLOT
.cast(moeccm.DES as varchar(255)) as MON_SCREEN_MATERIAL
,dim.MON_DIAMETER_OUOM
,dim.MON_DIAMETER_UNIT_OUOM
,dim.MON_TOP_OUOM
,dim.MON_BOT_OUOM
,dim.MON_UNIT_OUOM
,dim.MON_COMMENT
.cast(null as int) as MON_FLOWING
.cast( null as int ) as MON_ID
.cast( 522 as int ) as DATA_ID
.cast( '20200810a' as varchar(255) ) as SYS_TEMP1
.cast( 20200810 as int ) as SYS_TEMP2
,row_number() over (order by dim.loc_id) as rkey
into moe_20200721.dbo.O_D_INTERVAL_MONITOR
from
MOE_20200721.dbo.ORMGP_20200721_upd_DINTMON as dim
left outer join MOE_20200721.dbo_code_casing_material as moeccm
on dim.MON_SCREEN_MATERIAL=moeccm.CODE

```

Assemble the D_INTERVAL table. Note that we’re using an assigned date of ‘1867-07-01’ for those missing INT_START_DATES. We’re also using using a ‘group by’ statement to avoid creating duplicate interval/screen records.

```

select
t.LOC_ID
,t.INT_ID
,t.tmp_INT_ID
.cast( v.moe_well_id as varchar(255) ) as INT_NAME
.cast( t.tmp_int_id as varchar(255) ) as INT_NAME_ALT1
,t.tmp_INT_TYPE_CODE as INT_TYPE_CODE
,case

```

```

when moe.date_completed is not null then moe.date_completed
else cast('1867-07-01' as datetime)
end as INT_START_DATE
,cast(1 as int) as INT_CONFIDENTIALITY_CODE
,cast(1 as int) as INT_ACTIVE
,cast( case
when t.int_id is not null then 1
else null
end as int) as int_exists
,cast(522 as int) as [DATA_ID]
,cast('20200721a' as varchar(255) ) as SYS_TEMP1
,cast( 20200721 as int ) as SYS_TEMP2
,row_number() over (order by t.int_id) as rkey
into moe_20200721.dbo.O_D_INTERVAL
from
(
select
dim.LOC_ID
,dim.INT_ID
,dim.tmp_INT_ID
,dim.tmp_INT_TYPE_CODE
from
moe_20200721.dbo.O_D_INTERVAL_MONITOR as dim
group by
dim.loc_id,dim.tmp_int_id,dim.int_id,dim.tmp_int_type_code
) as t
inner join oak_20160831_master.dbo.v_sys_moe_locations as v
on t.tmp_int_id=v.moe_bore_hole_id
inner join moe_20200721.dbo.tblbore_hole as moe
on t.tmp_int_id=moe.bore_hole_id

```

Perform a check to make sure there are not any duplicate records in D_INTERVAL.
Correct/delete those rows (not shown).

Script: G_30_05_03_DINT_DIM.sql

We can now update those INT_IDs that already exist in the ORMDB as well as add any new intervals. We'll update the INT_IDs first.

```

update oak_20160831_master.dbo.d_interval
set
int_name= d.int_name
,int_name_alt1= d.int_name_alt1
,int_type_code= d.int_type_code
,int_start_date= d.int_start_date
,int_confidentiality_code= d.int_confidentiality_code
,int_active= d.int_active
,data_id= d.data_id
,sys_temp1= d.sys_temp1
,sys_temp2= d.sys_temp2
from
oak_20160831_master.dbo.d_interval as dint
inner join moe_20210119.dbo.o_d_interval as d
on dint.int_id=d.int_id
where
d.int_exists=1
and
(
( d.int_start_date <> dint.int_start_date )
or
( d.int_type_code <> dint.int_type_code )
or
( d.int_name_alt1 <> dint.int_name_alt1 )
)

```

We'll need to create some random INT_IDs for the remainder (i.e. these intervals are not already present in the ORMDB).

```
update moe_20200721.dbo.o_d_interval
set
int_id= t2.int_id
from
moe_20200721.dbo.o_d_interval as dint
inner join
(
select
t.int_id
,row_number() over (order by t.int_id) as rkey
from
(
select
top 5200
v.new_id as int_id
from
oak_20160831_master.dbo.v_sys_random_id_bulk_001 as v
where
v.new_id not in
( select int_id from oak_20160831_master.dbo.d_interval )
) as t
) as t2
on dint.rkey=t2.rkey
where
dint.int_exists is null
```

Insert them into D_INTERVAL.

```
insert into oak_20160831_master.dbo.d_interval
(
[LOC_ID],
[INT_ID],
[INT_NAME],
[INT_NAME_ALT1],
[INT_TYPE_CODE],
[INT_START_DATE],
[INT_CONFIDENTIALITY_CODE],
[INT_ACTIVE],
[DATA_ID],
[SYS_TEMP1],
[SYS_TEMP2]
)
select
[LOC_ID],
[INT_ID],
[INT_NAME],
[INT_NAME_ALT1],
[INT_TYPE_CODE],
[INT_START_DATE],
[INT_CONFIDENTIALITY_CODE],
[INT_ACTIVE],
[DATA_ID],
[SYS_TEMP1],
[SYS_TEMP2]
from
moe_20200721.dbo.o_d_interval as d
where
d.int_exists is null
```

Now that the intervals are updated or created, we can update O_D_INTERVAL_MONITOR and then insert into D_INTERVAL_MONITOR. First we'll associate the INT_ID.

```

update moe_20200721.dbo.o_d_interval_monitor
set
int_id= dint.int_id
from
moe_20200721.dbo.o_d_interval_monitor as dim
inner join moe_20200721.dbo.o_d_interval as dint
on dim.tmp_int_id=dint.tmp_int_id
where
dint.int_exists is null

```

And then populate the MON_ID.

```

update moe_20200721.dbo.o_d_interval_monitor
set
mon_id= t2.mon_id
from
moe_20200721.dbo.o_d_interval_monitor as d
inner join
(
select
t.mon_id
,row_number() over (order by t.mon_id) as rkey
from
(
select
top 5200
v.new_id as mon_id
from
oak_20160831_master.dbo.v_sys_random_id_bulk_001 as v
where
v.new_id not in
( select mon_id from oak_20160831_master.dbo.d_interval_monitor )
) as t
) as t2
on d.id=t2.rkey

```

And, finally, add the new records into D_INTERVAL_MONITOR.

```

insert into oak_20160831_master.dbo.d_interval_monitor
(
[INT_ID],
[MON_SCREEN_SLOT],
[MON_SCREEN_MATERIAL],
[MON_DIAMETER_OUOM],
[MON_DIAMETER_UNIT_OUOM],
[MON_TOP_OUOM],
[MON_BOT_OUOM],
[MON_UNIT_OUOM],
[MON_COMMENT],
[MON_FLOWING],
[MON_ID],
[DATA_ID],
[SYS_TEMP1],
[SYS_TEMP2]
)
select
[INT_ID],
[MON_SCREEN_SLOT],
[MON_SCREEN_MATERIAL],
[MON_DIAMETER_OUOM],
[MON_DIAMETER_UNIT_OUOM],
[MON_TOP_OUOM],
[MON_BOT_OUOM],
[MON_UNIT_OUOM],
[MON_COMMENT],

```

```

[MON_FLOWING],
[MON_ID],
[DATA_ID],
[SYS_TEMP1],
[SYS_TEMP2]
from
moe_20200721.dbo.o_d_interval_monitor as odim
where
odim.int_id not in
( select distinct(int_id) from oak_20160831_master.dbo.d_interval_monitor )

```

Script: G_30_05_04_DINT_DIM_ADD.sql

Note that the MON_FLOWING field may need to be updated in a future step.

G.30.6 D_INTERVAL_REF_ELEV

For these new or updated intervals, we'll check and correct (or add to) the intervals in the D_INTERVAL_REF_ELEV table. We should first make sure that an elevation has been assigned in the D_BOREHOLE table (as we're actually calculating a reference elevation, herein, we'll need the ground elevation at the location). Note that we're using the temporary D_INTERVAL table to determine which intervals need to be updated.

```

-- create view V_MOE_20200721_UPD_ELEVS as
select
dbore.loc_id
,dloc.loc_coord_easting as x
,dloc.loc_coord_northing as y
from
moe_20200721.dbo.o_d_interval as dint
inner join oak_20160831_master.dbo.d_borehole as dbore
on dint.loc_id=dbore.loc_id
inner join oak_20160831_master.dbo.d_location as dloc
on dbore.loc_id=dloc.loc_id
left outer join oak_20160831_master.dbo.d_interval_ref_elev as dire
on dint.int_id=dire.int_id
where
( dire.int_id is null or dire.ref_elev is null )
and dbore.bh_gnd_elev is null

```

We'll create a view from this, determine the elevations and populate the D_BOREHOLE table (not shown). Now we can create the temporary D_INTERVAL_REF_ELEV table.

```

select
dint.INT_ID
,dire.SYS_RECORD_ID
,dint.int_start_date as REF_ELEV_START_DATE
,( dbore.bh_gnd_elev + 0.75 ) as REF_ELEV
,( dbore.bh_gnd_elev + 0.75 ) as REF_ELEV_OUOM
,cast('masl' as varchar(50)) as REF_ELEV_UNIT_OUOM
,0.75 as REF_STICK_UP
,cast('0.75' as varchar(50)) as REF_POINT
,cast( 522 as int ) as DATA_ID
,cast( '20200810a' as varchar(255) ) as SYS_TEMP1
,cast( 20200810 as int ) as SYS_TEMP2
,case
when dire.int_id is not null then 1
else null
end as int_exists
,row_number() over (order by dint.int_id) as rkey
into moe_20200721.dbo.O_D_INTERVAL_REF_ELEV

```

```

from
moe_20200721.dbo.o_d_interval as dint
inner join oak_20160831_master.dbo.d_borehole as dbore
on dint.loc_id=dbore.loc_id
left outer join oak_20160831_master.dbo.d_interval_ref_elev as dire
on dint.int_id=dire.int_id
where
dire.int_id is null
or dire.ref_elev is null

```

And then populate the NULL SYS_RECORD_IDS.

```

update moe_20200721.dbo.o_d_interval_ref_elev
set
sys_record_id= t2.sri
from
moe_20200721.dbo.o_d_interval_ref_elev as dire
inner join
(
select
t.sri
,row_number() over (order by t.sri) as rkey
from
(
select
top 1500
v.new_id as sri
from
oak_20160831_master.dbo.v_sys_random_id_bulk_001 as v
where
v.new_id not in
( select sys_record_id from oak_20160831_master.dbo.d_interval_ref_elev )
) as t
) as t2
on dire.rkey=t2.rkey
where
dire.int_exists is null

```

We can update the fields for those INT_IDS (actually through SYS_RECORD_ID) that already exist in the ORMDB.

```

update oak_20160831_master.dbo.d_interval_ref_elev
set
ref_elev_start_date= m.ref_elev_start_date
,ref_elev= m.ref_elev
,ref_elev_ouom= m.ref_elev_ouom
,ref_elev_unit_ouom= m.ref_elev_unit_ouom
,ref_stick_up= m.ref_stick_up
,ref_point= m.ref_point
,data_id= m.data_id
,sys_temp1= m.sys_temp1
,sys_temp2= m.sys_temp2
from
oak_20160831_master.dbo.d_interval_ref_elev as dire
inner join moe_20200721.dbo.o_d_interval_ref_elev as m
on dire.sys_record_id=m.sys_record_id
where
m.int_exists is not null

```

And we can insert the remainder.

```

insert into oak_20160831_master.dbo.d_interval_ref_elev
(
[INT_ID],
[SYS_RECORD_ID],

```

```

[REF_ELEV_START_DATE],
[REF_ELEV],
[REF_ELEV_OUOM],
[REF_ELEV_UNIT_OUOM],
[REF_STICK_UP],
[REF_POINT],
[DATA_ID],
[SYS_TEMP1],
[SYS_TEMP2]
)
select
[INT_ID],
[SYS_RECORD_ID],
[REF_ELEV_START_DATE],
[REF_ELEV],
[REF_ELEV_OUOM],
[REF_ELEV_UNIT_OUOM],
[REF_STICK_UP],
[REF_POINT],
[DATA_ID],
[SYS_TEMP1],
[SYS_TEMP2]
from
moe_20200721.dbo.o_d_interval_ref_elev as m
where
m.int_exists is null

```

Script: G_30_06_01_DIRE.sql

G.30.7 D_INTERVAL_TEMPORAL_2 – Static Waterlevels

We will not rely upon the interval list generated in Section G.30.5, previously. This was tested (i.e. limited to the interval list) and only returns a subset of the missing values. Instead, we'll build the values to be incorporated from the complete list of MOEDB related boreholes.

```

select
dint.int_id
,v.moe_bore_hole_id
into moe_20200721.dbo.ORMGP_20200721_upd_DIT2_628
from
oak_20160831_master.dbo.v_sys_moe_locations as v
inner join oak_20160831_master.dbo.v_sys_agency_ypdt as y
on v.loc_id=y.loc_id
inner join oak_20160831_master.dbo.d_interval as dint
on v.loc_id=dint.loc_id
left outer join
(
select
int_id
from
oak_20160831_master.dbo.d_interval_temporal_2
where
rd_type_code=0
and rd_name_code=628
group by
int_id
) as d2
on dint.int_id=d2.int_id
inner join moe_20200721.dbo.tblpipe as moetp
on v.moe_bore_hole_id=moetp.bore_hole_id
inner join moe_20200721.dbo.tblpump_test as moept
on moetp.pipe_id=moept.pipe_id
where
d2.int_id is null

```

```

and moept.static_lev is not null
group by
dint.int_id,v.moe_bore_hole_id

```

We can now use this list to generate the D_INTERVAL_TEMPORAL_2 compatible table containing the static water levels.

```

select
dbore.LOC_ID
,dint.INT_ID
,.cast(0 as int) as RD_TYPE_CODE
,.cast(628 as int) as RD_NAME_CODE
,.case
when dint.int_start_date is null then cast('1867-07-01' as datetime )
else dint.INT_START_DATE
end as [RD_DATE]
,.cast(
case
when moept.LEVELS_UOM like 'ft' then dbore.bh_gnd_elev-(0.3048*moept.Static_lev)
else dbore.bh_gnd_elev-moept.Static_lev
end
as float
) as[RD_VALUE]
,.cast(6 as int) as UNIT_CODE
,.cast('Water Level - Manual - Static' as varchar(255)) as RD_NAME_OUOM
,.cast(moept.Static_lev as float) as [RD_VALUE_OUOM]
,.cast(moept.LEVELS_UOM as varchar(50)) as [RD_UNIT_OUOM]
,.cast(1 as int) as [REC_STATUS_CODE]
,.cast(null as varchar(255)) as RD_COMMENT
,.cast(522 as int) as DATA_ID
,.cast(null as int) as SYS_RECORD_ID
,.ROW_NUMBER() over (order by dint.INT_ID) as rkey
into moe_20200721.dbo.O_D_INTERVAL_TEMPORAL_2_628
from
moe_20200721.dbo.ormgp_20200721_upd_dit2_628 as d2
inner join moe_20200721.dbo.tblpipe as moetp
on d2.moe_bore_hole_id=moetp.bore_hole_id
inner join moe_20200721.dbo.tblpump_test as moept
on moetp.pipe_id=moept.pipe_id
inner join oak_20160831_master.dbo.d_interval as dint
on d2.int_id=dint.int_id
inner join oak_20160831_master.dbo.d_borehole as dbore
on dint.loc_id=dbore.loc_id

```

Check this table (i.e. O_D_INTERVAL_TEMPORAL_2_628) for NULL RD_VALUE fields – these do not have a valid BH_GND_ELEV value in D_BOREHOLE. Fix this issue (not shown) then re-create the table (using the above script). We will now populate the SYS_RECORD_ID field.

```

update moe_20200721.dbo.o_d_interval_temporal_2_628
set
sys_record_id= t2.sri
from
moe_20200721.dbo.o_d_interval_temporal_2_628 as d
inner join
(
select
t.sri
,.row_number() over (order by t.sri) as rkey
from
(
select
top 3000
v.new_id as sri
from

```

```

oak_20160831_master.dbo.v_sys_random_id_bulk_001 as v
where
v.new_id not in
( select sys_record_id from oak_20160831_master.dbo.d_interval_temporal_2 )
) as t
) as t2
on d.rkey=t2.rkey

```

And insert the values into D_INTERVAL_TEMPORAL_2.

```

insert into oak_20160831_master.dbo.d_interval_temporal_2
(
[INT_ID],
[RD_TYPE_CODE],
[RD_NAME_CODE],
[RD_DATE],
[RD_VALUE],
[UNIT_CODE],
[RD_NAME_OUOM],
[RD_VALUE_OUOM],
[RD_UNIT_OUOM],
[REC_STATUS_CODE],
[RD_COMMENT],
[DATA_ID],
SYS_TEMP1,
SYS_TEMP2,
[SYS_RECORD_ID]
)
select
[INT_ID],
[RD_TYPE_CODE],
[RD_NAME_CODE],
[RD_DATE],
[RD_VALUE],
[UNIT_CODE],
[RD_NAME_OUOM],
[RD_VALUE_OUOM],
[RD_UNIT_OUOM],
[REC_STATUS_CODE],
[RD_COMMENT],
[DATA_ID],
cast( '20200810b' as varchar(255) ) as SYS_TEMP1,
cast( 20200810 as int ) as SYS_TEMP2,
[SYS_RECORD_ID]
from
moe_20200721.dbo.o_d_interval_temporal_2_628

```

Script: G_30_07_01_DIT2_628.sql

G.30.8 D_PUMPTEST and related

We should first check and update the status of flowing wells in the ORMDB as tagged in the MOEDB (this information is found in the 'tblpump_test' table in the latter).

```

update oak_20160831_master.dbo.d_interval_monitor
set
mon_flowings= 1
,data_id= case
when dim.data_id is null then 522
else dim.data_id
end
.mon_comment= case
when dim.data_id is null then 'MON_FLOWING update from DATA_ID 522'
else dim.mon_comment + ' ; MON_FLOWING update from DATA_ID 522'

```

```

end
,sys_temp1= cast( '20200810c' as varchar(255) )
,sys_temp2= cast( 20200810 as int )
from
oak_20160831_master.dbo.d_interval_monitor as dim
inner join oak_20160831_master.dbo.d_interval as dint
on dim.int_id=dint.int_id
inner join oak_20160831_master.dbo.v_sys_moe_locations as v
on dint.loc_id=v.loc_id
inner join oak_20160831_master.dbo.v_sys_agency_ypdt as y
on v.loc_id=y.loc_id
inner join moe_20200721.dbo.tblpipe as moetp
on v.moe_bore_hole_id=moetp.bore_hole_id
inner join moe_20200721.dbo.tblpump_test as moept
on moetp.pipe_id=moept.pipe_id
where
moetp.FLOWING like 'Y'
and dim.mon_flowng is null
and dint.int_type_code in
( select int_type_code from oak_20160831_master.dbo.v_sys_int_type_code_screen )

```

We can now assemble those INT_IDs (and related) that require pumping information to be added.

```

select
dint.int_id
,v.moe_well_id
,v.moe_bore_hole_id
,moetp.pipe_id
into moe_20200721.dbo.ORMGP_20200721_upd_DPUMP
from
oak_20160831_master.dbo.v_sys_moe_locations as v
inner join oak_20160831_master.dbo.v_sys_agency_ypdt as y
on v.loc_id=y.loc_id
inner join oak_20160831_master.dbo.d_interval as dint
on v.loc_id=dint.loc_id
left outer join
(
select
int_id
from
oak_20160831_master.dbo.d_pumptest
group by
int_id
) as d
on dint.int_id=d.int_id
inner join moe_20200721.dbo.tblpipe as moetp
on v.moe_bore_hole_id=moetp.bore_hole_id
inner join moe_20200721.dbo.tblpump_test as moept
on moetp.pipe_id=moept.pipe_id
where
d.int_id is null
-- there are non-screen intervals, make sure not to include them
and dint.int_type_code in ( select int_type_code from oak_20160831_master.dbo.v_sys_int_type_code_screen )
and (
moetp.Recom_depth is not null
or moetp.Recom_rate is not null
or moetp.Flowing_rate is not null
or moept.PUMP_TEST_ID in
(select PUMP_TEST_ID from moe_20200721.dbo.tblpump_test_detail)
)

```

And then assemble the temporary O_D_PUMPTEST table.

```

select
d.INT_ID

```

```

.cast( null as int ) as PUMP_TEST_ID
.moept.pump_test_id as moe_pump_test_id
.case
when dint.int_start_date is not null then dint.int_start_date
else cast( '1867-07-01' as datetime )
end as PUMPTEST_DATE
.cast(d.moe_well_id as varchar(20)) as PUMPTEST_NAME
.cast(
case
when moept.LEVELS_UOM='ft' then moept.Recom_depth*0.3048
else moept.Recom_depth
end as float) as [REC_PUMP_DEPTH_METERS]
.cast(
case
when moept.RATE_UOM='LPM' then moept.Recom_rate/4.55
else moept.Recom_rate
end as float) as [REC_PUMP_RATE_IGPM]
.cast(
case
when moept.RATE_UOM='LPM' then moept.Flowing_rate/4.55
else moept.Flowing_rate
end as float) as [FLOWING_RATE_IGPM]
.cast(522 as int) as [DATA_ID]
.cast(
case
when moept.PUMPING_TEST_METHOD is null then null
when moept.PUMPING_TEST_METHOD=0 then null
when moept.PUMPING_TEST_METHOD=1 then 1
when moept.PUMPING_TEST_METHOD=2 then 2
when moept.PUMPING_TEST_METHOD=3 then 4
when moept.PUMPING_TEST_METHOD=4 then 8
when moept.PUMPING_TEST_METHOD=5 then 9
else 10
end as int) as [PUMPTEST_METHOD_CODE]
.cast(1 as int) as [PUMPTEST_TYPE_CODE] -- this is a constant rate indicator
.cast(
case
when moept.WATER_STATE_AFTER_TEST is null then 0
else moept.WATER_STATE_AFTER_TEST
end as int) as [WATER_CLARITY_CODE]
.ROW_NUMBER() over (order by dint.INT_ID) as rkey
into moe_20200721.dbo.O_D_PUMPTEST
from
moe_20200721.dbo.TblPump_Test as moept
inner join moe_20200721.dbo.orgp_20200721_upd_dpump as d
on moept.pipe_id=d.pipe_id
inner join oak_20160831_master.dbo.d_interval as dint
on d.int_id=dint.int_id

```

Now, corresponding to the methodology in G.10.19, we'll adjust the MON_FLOWING tag in D_INTERVAL_MONITOR (as well as related fields in various other tables). We will also modify the FLOWING_RATE_IGPM (as necessary). First, do we have a null MON_FLOWING while REC_PUMP_RATE_IGPM equals FLOWING_RATE_IGPM – if so, make FLOWING_RATE_IGPM null.

```

update moe_20200721.dbo.o_d_pumptest
set
flowing_rate_igpm=null
from
moe_20200721.dbo.o_d_pumptest as d
where
d.INT_ID
not in
(
select
dim.INT_ID

```

```

from
oak_20160831_master.dbo.d_interval_monitor as dim
where
dim.MON_FLOWING is not null
)
and d.REC_PUMP_RATE_IGPM=d.FLOWING_RATE_IGPM

```

Next, there is no REC_PUMP_RATE_IGPM but there is a FLOWING_RATE_IGPM – tag MON_FLOWING.

```

update oak_20160831_master.dbo.d_interval_monitor
set
mon_flowings=1
from
oak_20160831_master.dbo.d_interval_monitor as dim
where
dim.INT_ID
in
(
select
d.INT_ID
from
moe_20200721.dbo.o_d_pumptest as d
where
d.INT_ID
in
(
select
d2.INT_ID
from
oak_20160831_master.dbo.d_interval_monitor as d2
where
d2.MON_FLOWING is null
)
)
and d.REC_PUMP_RATE_IGPM is null
and d.FLOWING_RATE_IGPM is not null
)

```

If the FLOWING_RATE_IGPM is less than the REC_PUMP_RATE_IGPM and MON_FLOWING is not tagged, tag MON_FLOWING.

```

update oak_20160831_master.dbo.d_interval_monitor
set
mon_flowings=1
from
oak_20160831_master.dbo.d_interval_monitor as dim
where
dim.INT_ID
in
(
select
d.INT_ID
from
moe_20200721.dbo.o_d_pumptest as d
where
d.INT_ID
in
(
select
d2.INT_ID
from
oak_20160831_master.dbo.d_interval_monitor as d2
where
d2.MON_FLOWING is null
)
)
and d.REC_PUMP_RATE_IGPM is null

```

```
and d.FLOWING_RATE_IGPM is not null
)
```

If the FLOWING_RATE_IGPM is greater than REC_PUMP_RATE_IGPM and MON_FLOWING is not tagged, null the FLOWING_RATE_IGPM (this is considered an error).

```
update moe_20200721.dbo.o_d_pumptest
set
flowing_rate_igpm=null
from
moe_20200721.dbo.o_d_pumptest as dpump
where
dpump.INT_ID
in
(
select
d.INT_ID
from
moe_20200721.dbo.o_d_pumptest as d
where
d.INT_ID
in
(
select
d2.INT_ID
from
oak_20160831_master.dbo.d_interval_monitor as d2
where
d2.MON_FLOWING is null
)
)
and dpump.FLOWING_RATE_IGPM>dpump.REC_PUMP_RATE_IGPM
)
```

Script: G_30_08_01_DPUMP.sql

We can now build the pumping drawdown (i.e. ‘D’) and recovery (i.e. ‘R’) information for the particular pump tests (as determined in the previous step). First the drawdown information.

```
select
curr.Pump_Test_id as moe_pump_test_id
,cast(moept.Pumping_rate as float) as PUMP_RATE
,cast(moept.RATE_UOM as varchar(50)) as PUMP_RATE_UNITS
,cast(moept.Pumping_rate as float) as PUMP_RATE_OUOM
,cast(moept.RATE_UOM as varchar(50)) as PUMP_RATE_UNITS_OUOM
,dateadd(minute,curr.TestDuration,dpump.PUMPTEST_DATE) as [PUMP_START]
,dateadd(minute,prev.TestDuration,dpump.PUMPTEST_DATE) as [PUMP_END]
,cast(521 as int) as DATA_ID
,null as [SYS_RECORD_ID]
,null as [rkey]
,prev.TestType as [testtype]
,prev.TestLevel as [testlevel]
,prev.TESTLEVEL_UOM as [testlevel_uom]
into MOE_20200721.dbo.ORMGP_20200721_upd_DPUMPSTEP
from
(
select
moeptd.Pump_Test_id
,moeptd.TestDuration
,ROW_NUMBER() over (order by moeptd.Pump_Test_id,moeptd.TestDuration) as rnum
from
MOE_20200721.dbo.TblPump_Test_Detail as moeptd
where
```

```

        moeptd.TestType='D'
    ) as [curr]
inner join
(
    select
    moeptd.Pump_Test_id
    ,moeptd.TestDuration
    ,moeptd.TestLevel
    ,moeptd.TestType
    ,moeptd.TESTLEVEL_UOM
    ,ROW_NUMBER() over (order by moeptd.Pump_Test_id,moeptd.TestDuration) as rnum
    from
    MOE_20200721.dbo.TblPump_Test_Detail as moeptd
    where
    moeptd.TestType='D'
) as [prev]
on curr.Pump_Test_id=prev.Pump_Test_id and curr.rnum=(prev.rnum-1)
inner join MOE_20200721.dbo.TblPump_Test as moept
on curr.Pump_Test_id=moept.PUMP_TEST_ID
inner join moe_20200721.dbo.o_d_pumptest as dpump
on curr.pump_test_id=dpump.moe_pump_test_id
order by
curr.Pump_Test_id,PUMP_START

```

Now we can create the first time step – add to the previous table.

```

insert into MOE_20200721.dbo.ORMGP_20200721_upd_DPUMPSTEP
(
    moe_pump_test_id
    ,[PUMP_RATE]
    ,[PUMP_RATE_UNITS]
    ,[PUMP_RATE_OUOM]
    ,[PUMP_RATE_UNITS_OUOM]
    ,[PUMP_START]
    ,[PUMP_END]
    ,[DATA_ID]
    ,[SYS_RECORD_ID]
    ,[rnum]
    ,[testtype]
    ,[testlevel]
    ,[testlevel_uom]
)
select
curr.Pump_Test_id as moe_pump_test_id
,cast(moept.Pumping_rate as float) as PUMP_RATE
,cast(moept.RATE_UOM as varchar(50)) as PUMP_RATE_UNITS
,cast(moept.Pumping_rate as float) as PUMP_RATE_OUOM
,cast(moept.RATE_UOM as varchar(50)) as PUMP_RATE_UNITS_OUOM
,dateadd(minute,0,dpump.PUMPTEST_DATE) as [PUMP_START]
,dateadd(minute,curr.TestDuration,dpump.PUMPTEST_DATE) as [PUMP_END]
,cast(521 as int) as DATA_ID
,cast(null as int) as [SYS_RECORD_ID]
,cast(null as int) as [rkey]
,moeptd.TestType as [testtype]
,moeptd.TestLevel as [testlevel]
,moeptd.TESTLEVEL_UOM as [testlevel_uom]
from
(
    select
    moeptd.Pump_Test_id
    ,min(moeptd.TestDuration) as TestDuration
    from
        MOE_20200721.dbo.TblPump_Test_Detail as moeptd
    where
    moeptd.TestType='D'
    group by
    moeptd.Pump_Test_id
) as curr

```

```

inner join MOE_20200721.dbo.TblPump_Test as moept
on curr.Pump_Test_id=moept.PUMP_TEST_ID
inner join MOE_20200721.dbo.TblPump_Test_Detail as moeptd
on curr.Pump_Test_id=moeptd.Pump_Test_id and curr.TestDuration=moeptd.TestDuration
inner join moe_20200721.dbo.o_d_pumptest as dpump
on curr.pump_test_id=dpump.moe_pump_test_id
where
moeptd.TestType='D'
order by
curr.Pump_Test_id,PUMP_START

```

Now add the pumping recovery information (based upon the maximum datetime for the particular PUMP_TEST_ID).

```

insert into moe_20200721.dbo.ormgp_20200721_upd_dpumpstep
(
moe_pump_test_id
,pump_end
,data_id
,sys_record_id
,rnum
,testtype
,testlevel
,testlevel_uom
)
select
curr.Pump_Test_id as moe_pump_test_id
,dateadd(minute,curr.TestDuration,mt.enddate) as [PUMP_END]
,.cast(522 as int) as DATA_ID
,.null as [SYS_RECORD_ID]
,.null as [rnum]
,.curr.TestType as [testtype]
,.curr.TestLevel as [testlevel]
,.curr.TESTLEVEL_UOM as [testlevel_uom]
from
(
select
moeptd.Pump_Test_id
,moeptd.TestDuration
,moeptd.TestType
,moeptd.TestLevel
,moeptd.TESTLEVEL_UOM
from
MOE_20200721.dbo.TblPump_Test_Detail as moeptd
where
moeptd.TestType='R'
) as [curr]
inner join
(
select
ycps.moe_Pump_Test_id
,max(ycps.PUMP_END) as enddate
from
MOE_20200721.dbo.ORMGP_20200721_upd_dpumpstep as ycps
where
ycps.TestType='D'
group by
ycps.moe_Pump_Test_id
) as mt
on curr.Pump_Test_id=mt.moe_Pump_Test_id
inner join MOE_20200721.dbo.TblPump_Test as moept
on curr.Pump_Test_id=moept.PUMP_TEST_ID
inner join MOE_20200721.dbo.o_D_PUMPTEST as dpump
on curr.Pump_Test_id=dpump.pump_test_id
order by
curr.Pump_Test_id,PUMP_END

```

Create the O_D_PUMPTEST_STEP table based upon a grouping of the grouping of pumping rate and units. Note that only the drawdown information is being used.

```

select
cast( null as int ) as PUMP_TEST_ID
,t1.moe_PUMP_TEST_ID
,t1.PUMP_RATE
,t1.PUMP_RATE_UNITS
,t1.PUMP_RATE_OUOM
,t1.PUMP_RATE_UNITS_OUOM
,t1.PUMP_START
,t1.PUMP_END
,cast(522 as int) as DATA_ID
,t1.SYS_RECORD_ID
,t1.rkey
into MOE_20200721.dbo.O_D_PUMPTEST_STEP
from
(
SELECT
[moe_PUMP_TEST_ID]
,[PUMP_RATE]
,[PUMP_RATE_UNITS]
,[PUMP_RATE_OUOM]
,[PUMP_RATE_UNITS_OUOM]
,.[min]([PUMP_START]) as [PUMP_START]
,.[max]([PUMP_END]) as [PUMP_END]
,cast( null as int ) as SYS_RECORD_ID
,ROW_NUMBER() over (order by moe_PUMP_TEST_ID) as rkey
FROM MOE_20200721.[dbo].ormgp_20200721_upd_dpumpstep
where
testtype='D'
group by
moe_Pump_Test_id,PUMP_RATE,PUMP_RATE_UNITS,PUMP_RATE_OUOM,PUMP_RATE_UNITS_OUOM
) as t1

```

We can now add the pumping and recovery water levels into a compatible D_INTERVAL_TEMPORAL_2 table. Add an index to this table.

```

select
dp.INT_ID
,case
when dps.testtype='D' then cast(65 as int) -- i.e. moe pumping level
else cast(64 as int) -- i.e. moe recovery level
end as [RD_TYPE_CODE]
,cast(70899 as int) as [RD_NAME_CODE] -- i.e. Water Level - Manual - Other
,dps.PUMP_END as [RD_DATE]
,case
when dps.testlevel_uom='m' then dbore.bh_gnd_elev - dps.testlevel
else dbore.bh_gnd_elev - (dps.testlevel*0.3048)
end as [RD_VALUE]
,cast(6 as int) as [UNIT_CODE]
,cast('Water Level - Manual - Other' as varchar(100)) as RD_NAME_OUOM
,cast(dps.testlevel as float) as RD_VALUE_OUOM
,cast(dps.testlevel_uom as varchar(50)) as RD_UNIT_OUOM
,cast(1 as int) as REC_STATUS_CODE
,cast(
case
when dps.testtype='D' then 'Pumping - Drawdown'
else 'Pumping - Recovery'
end as varchar(255) ) as [RD_COMMENT]
,dps.DATA_ID as [DATA_ID]
,dps.SYS_RECORD_ID
,dps.rkey
into moe_20200721.dbo.O_D_INTERVAL_TEMPORAL_2_70899
from
MOE_20200721.[dbo].ORMGP_20200721_upd_DPUMPSTEP as dps

```

```

inner join MOE_20200721.dbo.O_D_PUMPTTEST as dp
on dps.moe_PUMP_TEST_ID=dp.moe_PUMP_TEST_ID
inner join oak_20160831_master.dbo.d_interval as dint
on dp.int_id=dint.int_id
inner join oak_20160831_master.dbo.d_borehole as dbore
on dint.loc_id=dbore.loc_id
where
dps.testtype is not null
and dbore.bh_gnd_elev is not null
order by
dp.INT_ID,RD_DATE

```

Finally populate the SYS_RECORD_ID field.

```

update moe_20200721.dbo.o_d_interval_temporal_2_70899
set
sys_record_id= t2.sri
from
moe_20200721.dbo.o_d_interval_temporal_2_70899 as d
inner join
(
select
t.sri
,row_number() over (order by t.sri) as rkey
from
(
select
top 15000
v.new_id as sri
from
oak_20160831_master.dbo.v_sys_random_id_bulk_001 as v
where
v.new_id not in
(select sys_record_id from oak_20160831_master.dbo.d_interval_temporal_2 )
) as t
) as t2
on d.rkey=t2.rkey

```

And then insert into the D_INTERVAL_TEMPORAL_2 table.

```

insert into oak_20160831_master.dbo.d_interval_temporal_2
(
[INT_ID],
[RD_TYPE_CODE],
[RD_NAME_CODE],
[RD_DATE],
[RD_VALUE],
[UNIT_CODE],
[RD_NAME_OUOM],
[RD_VALUE_OUOM],
[RD_UNIT_OUOM],
[REC_STATUS_CODE],
[RD_COMMENT],
[DATA_ID],
[SYS_RECORD_ID],
SYS_TEMP1,
SYS_TEMP2
)
select
[INT_ID],
[RD_TYPE_CODE],
[RD_NAME_CODE],
[RD_DATE],
[RD_VALUE],
[UNIT_CODE],
[RD_NAME_OUOM],
[RD_VALUE_OUOM],

```

```

[RD_UNIT_OUOM],
[REC_STATUS_CODE],
[RD_COMMENT],
[DATA_ID],
[SYS_RECORD_ID],
cast( '20200811a' as varchar(255) ) as SYS_TEMP1,
cast( 20200811 as int ) as SYS_TEMP2
from
moe_20200721.dbo.o_d_interval_temporal_2_70899

```

Script: G_30_08_02_DPUMPSTEP.sql

We'll now incorporate the information from the pumping tables into the ORMDB.
Create the PUMP_TEST_ID values for O_D_PUMPTEST.

```

update moe_20200721.dbo.o_d_pumptest
set
pump_test_id= t2.pump_test_id
from
moe_20200721.dbo.o_d_pumptest as d
inner join
(
select
t.pump_test_id
,row_number() over (order by t.pump_test_id) as rkey
from
(
select
top 2000
v.new_id as pump_test_id
from
oak_20160831_master.dbo.v_sys_random_id_bulk_001 as v
where
v.new_id not in
( select pump_test_id from oak_20160831_master.dbo.d_pumptest )
) as t
) as t2
on d.rkey=t2.rkey

```

Insert these values into D_PUMPTEST.

```

insert into oak_20160831_master.dbo.d_pumptest
(
[INT_ID],
[PUMP_TEST_ID],
[PUMPTEST_DATE],
[PUMPTEST_NAME],
[REC_PUMP_DEPTH_METERS],
[REC_PUMP_RATE_IGPM],
[FLOWING_RATE_IGPM],
[DATA_ID],
[PUMPTEST_METHOD_CODE],
[PUMPTEST_TYPE_CODE],
[WATER_CLARITY_CODE],
SYS_TEMP1,
SYS_TEMP2
)
select
[INT_ID],
[PUMP_TEST_ID],
[PUMPTEST_DATE],
[PUMPTEST_NAME],
[REC_PUMP_DEPTH_METERS],
[REC_PUMP_RATE_IGPM],
[FLOWING_RATE_IGPM],

```

```

[DATA_ID],
[PUMPTEST_METHOD_CODE],
[PUMPTEST_TYPE_CODE],
[WATER_CLARITY_CODE],
cast( '20200811b' as varchar(255) ) as SYS_TEMP1,
cast( 20200811 as int ) as SYS_TEMP2
from
moe_20200721.dbo.o_d_pumptest

```

Now, update the PUMP_TEST_ID field in O_D_PUMPTEST_STEP.

```

update moe_20200721.dbo.o_d_pumptest_step
set
pump_test_id= dp.pump_test_id
from
moe_20200721.dbo.o_d_pumptest_Step as dps
inner join moe_20200721.dbo.o_d_pumptest as dp
on dps.moe_pump_test_id=dp.moe_pump_test_id

```

And then populate the SYS_RECORD_ID field.

```

update moe_20200721.dbo.o_d_pumptest_step
set
sys_record_id= t.sri
from
moe_20200721.dbo.o_d_pumptest_step as d
inner join
(
select
t.sri
,row_number() over (order by t.sri) as rkey
from
(
select
top 1600
v.new_id as sri
from
oak_20160831_master.dbo.v_sys_random_id_bulk_001 as v
where
v.new_id not in
( select sys_record_id from oak_20160831_master.dbo.d_pumptest_step )
) as t
) as t2
on d.rkey=t2.rkey

```

Finally, insert this information into D_PUMPTEST_STEP.

```

insert into oak_20160831_master.dbo.d_pumptest_step
(
[PUMP_TEST_ID],
[PUMP_RATE],
[PUMP_RATE_UNITS],
[PUMP_RATE_OUOM],
[PUMP_RATE_UNITS_OUOM],
[PUMP_START],
[PUMP_END],
[DATA_ID],
[SYS_RECORD_ID],
SYS_TEMP1,
SYS_TEMP2
)
select
[PUMP_TEST_ID],
[PUMP_RATE],
[PUMP_RATE_UNITS],

```

```

[PUMP_RATE_OUOM],
[PUMP_RATE_UNITS_OUOM],
[PUMP_START],
[PUMP_END],
[DATA_ID],
[SYS_RECORD_ID],
cast( '20200811b' as varchar(255) ) as SYS_TEMP1,
cast( 20200811 as int ) as SYS_TEMP2
from
moe_20200721.dbo.o_d_pumptest_step

```

Script: G_30_08_03_DPID.sql

G.31 Incorporate D_LOCATION_COORD_HIST and D_LOCATION_ELEV_HIST records into D_LOCATION_SPATIAL_HIST

Tables

- D_LOCATION
- D_LOCATION_COORD_HIST
- D_LOCATION_ELEV
- D_LOCATION_ELEV_HIST
- D_LOCATION_QA
- D_LOCATION_SPATIAL
- D_LOCATION_SPATIAL_HIST

Estimated Recurrence Time: Single database update due to methodology change.

The tables D_LOCATION_COORD_HIST and D_LOCATION_ELEV_HIST functionality is to be combined in a single table – D_LOCATION_SPATIAL_HIST – to more closely link coordinates and elevations. The original methodology, using the two disparate tables, was found to be inadequate with regard to timely (or, at all) tracking of changes in one or the other of location or elevation without the values becoming disassociated (i.e. one value no longer related to the other). If this information is found in a single table, it is thought, that can be updated with a new record resulting from any change the likelihood of the location and the elevation being unrelated should be lessened.

Current coordinates and elevation, which should be recorded in the appropriate tables, is added into the D_LOCATION_SPATIAL_HIST table. Note that the conversion of LOC_COORD_OUOM_CODE from D_LOCATION (to the appropriate EPSG code) occurs within the query. Where assumptions are made concerning this conversion (as originally found in R_LOC_COORD_OUOM_CODE), these are recorded in the EPSG_CODE_COMMENT field. All pertinent comment (or text) fields from the various tables are appended into the coordinate or elevation method/comment fields in D_LOCATION_SPATIAL_HIST (zero-length text fields are converted to NULL values – not shown here). Differing dates from the variety of tables is examined with the most relevant date used to populate each of the date fields in the new table. Note that the choice of information used should be consistent through each of the following steps.

```

select
dloc.loc_id
.case
when dlch.loc_coord_hist_code is not null then dlch.loc_coord_hist_code
else 1
end as loc_coord_hist_code
.case
when dlch.sys_time_stamp is not null then dlch.sys_time_stamp
when dloc.loc_coord_hist_code is not null then dloc.loc_coord_hist_code
else getdate()
end as loc_coord_date
.dloc.loc_coord_easting as x
.dloc.loc_coord_northing as y
.cast(26917 as int) as epsg_code
.case
-- the original coordinates in these cases are invalid
when dloc.loc_coord_ouom_code in ( 25, 29, 31, 33, 35 ) then null
else dloc.loc_coord_easting_ouom
end as x_ouom
.case
-- the original coordinates in these cases are invalid
when dloc.loc_coord_ouom_code in ( 25, 29, 31, 33, 35 ) then null
else dloc.loc_coord_northing_ouom
end as y_ouom
.case
when dloc.loc_coord_ouom_code= 1 then null
when dloc.loc_coord_ouom_code= 2 then 26717
when dloc.loc_coord_ouom_code= 3 then 26718
when dloc.loc_coord_ouom_code= 4 then 26917
when dloc.loc_coord_ouom_code= 5 then 26918
when dloc.loc_coord_ouom_code= 6 then 26910
when dloc.loc_coord_ouom_code= 7 then 26915
when dloc.loc_coord_ouom_code= 8 then 26916
when dloc.loc_coord_ouom_code= 9 then 26919
when dloc.loc_coord_ouom_code= 10 then 26917
when dloc.loc_coord_ouom_code= 11 then 26717
when dloc.loc_coord_ouom_code= 12 then 26717
when dloc.loc_coord_ouom_code= 13 then 32190
when dloc.loc_coord_ouom_code= 14 then 7991
when dloc.loc_coord_ouom_code= 15 then 32190
when dloc.loc_coord_ouom_code= 16 then 32196
when dloc.loc_coord_ouom_code= 17 then 26917
when dloc.loc_coord_ouom_code= 18 then 4267
when dloc.loc_coord_ouom_code= 19 then 4269
when dloc.loc_coord_ouom_code= 20 then 4269
when dloc.loc_coord_ouom_code= 21 then 4269
when dloc.loc_coord_ouom_code= 22 then 26917
when dloc.loc_coord_ouom_code= 23 then 26918
when dloc.loc_coord_ouom_code= 24 then 32617
when dloc.loc_coord_ouom_code= 25 then null
when dloc.loc_coord_ouom_code= 26 then 26917
when dloc.loc_coord_ouom_code= 27 then 26917
when dloc.loc_coord_ouom_code= 29 then null
when dloc.loc_coord_ouom_code= 31 then null
when dloc.loc_coord_ouom_code= 33 then null
when dloc.loc_coord_ouom_code= 35 then null
when dloc.loc_coord_ouom_code= 36 then 26917
else null
end as epsg_code_orig
.case
when dloc.loc_coord_ouom_code= 10 then 'UTM NAD83 specified; Z17 assumed'
when dloc.loc_coord_ouom_code= 11 then 'UTM NAD27 specified; Z17 assumed'
when dloc.loc_coord_ouom_code= 12 then 'UTM Z17 specified; NAD27 assumed'
when dloc.loc_coord_ouom_code= 13 then 'MTM specified; Z10 and NAD83 assumed'
when dloc.loc_coord_ouom_code= 20 then 'Lat/Long, Z17 specified; NAD83 assumed'
when dloc.loc_coord_ouom_code= 21 then 'Lat/Long specified; NAD83 assumed'
when dloc.loc_coord_ouom_code= 22 then 'Z17 specified; UTM and NAD83 assumed'
when dloc.loc_coord_ouom_code= 23 then 'Z18 specified; UTM and NAD83 assumed'

```

```

when dloc.loc_coord_ouom_code= 26 then 'EPSG_CODE 26911 specified; 26917 assumed'
when dloc.loc_coord_ouom_code= 27 then 'EPSG_CODE 26902 specified; 26917 assumed'
when dloc.loc_coord_ouom_code= 36 then 'UTM Z17 and NAD83 assumed'
else null
end as epsg_code_comment
,dlqa.qa_coord_confidence_code as qa_coord_code
.case
when dlqa.qa_coord_method is null then "
else rtrim(dlqa.qa_coord_method) + ';'
end
+
case
when dlch.coord_hist_loc_method is null then "
else rtrim(dlch.coord_hist_loc_method) + ';'
end
as loc_coord_method
.case
when dloc.loc_coord_comment is null then "
else rtrim(dloc.loc_coord_comment) + ';'
end
+
-- dlch.coord_hist_comment
case
when dlch.coord_hist_comment is null then "
else rtrim(dlch.coord_hist_comment) + ';'
end
+
-- dlqa.qa_coord_source
case
when dlqa.qa_coord_source is null then "
else rtrim(dlqa.qa_coord_source) + ';'
end
+
-- dlqa.qa_coord_comment
case
when dlqa.qa_coord_comment is null then "
else rtrim(dlqa.qa_coord_comment) + ';'
end
as loc_coord_comment
,dlch.data_id as loc_coord_data_id
.case
when dleh.loc_elev_code is not null then dleh.loc_elev_code
else null
end as loc_elev_code
.case
when dleh.loc_elev_hist_date is not null then dleh.loc_elev_hist_date
when dleh.sys_time_stamp is not null then dleh.sys_time_stamp
else cast( null as datetime )
end as loc_elev_date
,delev.assigned_elev as loc_elev
,6 as loc_elev_unit_code
,delev.assigned_elev as loc_elev_ouom
,'mas!' as loc_elev_unit_ouom
.case
when dlqa.qa_elev_confidence_code is not null then dlqa.qa_elev_confidence_code
when dleh.qa_elev_confidence_code is not null then dleh.qa_elev_confidence_code
when delev.assigned_elev is not null then 10
else null
end as qa_elev_code
.case
when dlqa.qa_elev_method is null then "
else rtrim(dlqa.qa_elev_method) + ';'
end
as loc_elev_method
.case
when dlqa.qa_elev_source is null then "
else rtrim(dlqa.qa_elev_source) + ';'
end
+
case

```

```

when dlqa.qa_elev_comment is null then "
else rtrim(dlqa.qa_elev_comment) + ' '; '
end
+
case
when dleh.elev_hist_comment is null then "
else rtrim(dleh.elev_hist_comment) + ' '; '
end
as loc_elev_comment
,dleh.data_id as loc_elev_data_id
,.cast('20200714a' as varchar(255)) as sys_temp1
,.cast(20200714 as int) as sys_temp2
--,null as sys_last_modified_by
.case
when dleh.sys_last_modified>dloc.sys_last_modified then dleh.sys_last_modified_by
else dloc.sys_last_modified_by
end as sys_last_modified_by
--,null as sys_last_modified
.case
when dleh.sys_last_modified>dloc.sys_last_modified then dleh.sys_last_modified
else dloc.sys_last_modified
end as sys_last_modified
--,null as sys_user_stamp
.case
--when dlch.sys_user_stamp is not null then dlch.sys_user_stamp
when dloc.sys_last_modified_by is not null then dloc.sys_last_modified_by
else null
end as sys_user_stamp
--,null as sys_time_stamp
.case
when dlch.sys_time_stamp is not null then dlch.sys_time_stamp
when dloc.sys_last_modified is not null then dloc.sys_last_modified
else null
end as sys_time_stamp
from
d_location as dloc
left outer join d_location_elev as delev
on dloc.loc_id=delev.loc_id
left outer join d_location_elev_hist as dleh
on delev.loc_elev_id=dleh.loc_elev_id
left outer join d_location_coord_hist as dlch
on dloc.loc_id=dlch.loc_id
left outer join d_location_qa as dlqa
on dloc.loc_id=dlqa.loc_id
where
dlch.current_coord= 1

```

The resultant information is inserted into D_LOCATION_SPATIAL_HIST (not shown). As these are the most current coordinates, the SPAT_ID from these records are used to update D_LOCATION_SPATIAL which is used to flag the ‘current’ coordinates for a particular location.

```

insert into d_location_spatial
( loc_id, spat_id, spatial_comment, sys_temp1, sys_temp2 )
select
loc_id
,spat_id
,'Initial assignment 20200714'
,'20200714a' as sys_temp1
,20200714 as sys_temp2
from
d_location_spatial_hist

```

During this process, it was noted that some locations had been added to which the coordinates and/or elevations had not been captured. Include these.

```

select
dloc.loc_id
,l as loc_coord_hist_code
,case
when dloc.sys_last_modified is not null then dloc.sys_last_modified
else getdate()
end as loc_coord_date
,dloc.loc_coord_easting as x
,dloc.loc_coord_northing as y
,.26917 as epsg_code
,dloc.loc_coord_easting_ouom as x_ouom
,dloc.loc_coord_northing_ouom as y_ouom
,case
when dloc.loc_coord_ouom_code= 4 then 26917
when dloc.loc_coord_ouom_code= 12 then 26717
when dloc.loc_coord_ouom_code= 21 then 4269
else null
end as epsg_code_orig
,case
when dloc.loc_coord_ouom_code= 12 then 'UTM Z17 specified; NAD27 assumed'
when dloc.loc_coord_ouom_code= 21 then 'Lat/Long specified; NAD83 assumed'
else null
end as epsg_code_comment
,case
when dlqa.qa_coord_confidence_code is not null then dlqa.qa_coord_confidence_code
else 5
end
as qa_coord_code
,case
when dlqa.qa_coord_method is null then ''
else rtrim(dlqa.qa_coord_method) + ';'
end
as loc_coord_method
,case
when dloc.loc_coord_comment is null then ''
else rtrim(dloc.loc_coord_comment) + ';'
end
+
-- dlqa.qa_coord_source
case
when dlqa.qa_coord_source is null then ''
else rtrim(dlqa.qa_coord_source) + ';'
end
+
-- dlqa.qa_coord_comment
case
when dlqa.qa_coord_comment is null then ''
else rtrim(dlqa.qa_coord_comment) + ';'
end
as loc_coord_comment
,case
when dleh.loc_elev_code is not null then dleh.loc_elev_code
else null
end as loc_elev_code
,case
when dleh.loc_elev_hist_date is not null then dleh.loc_elev_hist_date
when dleh.sys_time_stamp is not null then dleh.sys_time_stamp
else cast( null as datetime )
end as loc_elev_date
,delev.assigned_elev as loc_elev
,.6 as loc_elev_unit_code
,delev.assigned_elev as loc_elev_ouom
,'masl' as loc_elev_unit_ouom
,case
when dlqa.qa_elev_confidence_code is not null then dlqa.qa_elev_confidence_code
when dleh.qa_elev_confidence_code is not null then dleh.qa_elev_confidence_code
when delev.assigned_elev is not null then 10
else null

```

```

end as qa_elev_code
.case
when dlqa.qa_elev_method is null then "
else rtrim(dlqa.qa_elev_method) + ';'
end
as loc_elev_method
.case
when dlqa.qa_elev_source is null then "
else rtrim(dlqa.qa_elev_source) + ';'
end
+
.case
when dlqa.qa_elev_comment is null then "
else rtrim(dlqa.qa_elev_comment) + ';'
end
+
.case
when dleh.elev_hist_comment is null then "
else rtrim(dleh.elev_hist_comment) + ';'
end
as loc_elev_comment
.cast('20200714b' as varchar(255)) as sys_temp1
.cast(20200714 as int) as sys_temp2
.case
when dleh.sys_last_modified>dloc.sys_last_modified then dleh.sys_last_modified_by
else dloc.sys_last_modified_by
end as sys_last_modified_by
--,null as sys_last_modified
.case
when dleh.sys_last_modified>dloc.sys_last_modified then dleh.sys_last_modified
else dloc.sys_last_modified
end as sys_last_modified
--,null as sys_user_stamp
.case
when dloc.sys_last_modified_by is not null then dloc.sys_last_modified_by
when dloc.sys_user_stamp is not null then dloc.sys_user_stamp
else null
end as sys_user_stamp
.case
when dloc.sys_last_modified is not null then dloc.sys_last_modified
when dloc.sys_time_stamp is not null then dloc.sys_time_stamp
else null
end as sys_time_stamp
from
d_location as dloc
left outer join d_location_qa as dlqa
on dloc.loc_id=dlqa.loc_id
left outer join d_location_elev as delev
on dloc.loc_id=delev.loc_id
left outer join d_location_elev_hist as dleh
on delev.loc_elev_id=dleh.loc_elev_id
left outer join d_location_spatial as dls
on dloc.loc_id=dls.loc_id
where
dls.loc_id is null
and dloc.loc_coord_easting is not null
and dloc.loc_coord_northing is not null

```

Note that only a subset of the LOC_COORD_OUOM_CODE values were processed in this case.

Now that we've processed the 'current' coordinates, process those coordinates that have an associated elevation. This only uses D_LOCATION_COORD_HIST and D_LOCATION_ELEV_HIST as a source.

```
select
```

```

dlch.loc_id
.case
when dlch.loc_coord_hist_code is not null then dlch.loc_coord_hist_code
else 1
end as loc_coord_hist_code
.case
when dlch.sys_time_stamp is not null then dlch.sys_time_stamp
else getdate()
end as loc_coord_date
,dlch.x_utmz17nad83 as x
,dlch.y_utmz17nad83 as y
,.cast(26917 as int) as epsg_code
.case
-- the original coordinates in these cases are invalid
when dlch.loc_coord_ouom_code in ( 25, 29, 31, 33, 35 ) then null
else dlch.x_ouom
end as x_ouom
.case
-- the original coordinates in these cases are invalid
when dlch.loc_coord_ouom_code in ( 25, 29, 31, 33, 35 ) then null
else dlch.y_ouom
end as y_ouom
.case
when dlch.loc_coord_ouom_code= 1 then null
when dlch.loc_coord_ouom_code= 2 then 26717
when dlch.loc_coord_ouom_code= 3 then 26718
when dlch.loc_coord_ouom_code= 4 then 26917
when dlch.loc_coord_ouom_code= 5 then 26918
when dlch.loc_coord_ouom_code= 6 then 26910
when dlch.loc_coord_ouom_code= 7 then 26915
when dlch.loc_coord_ouom_code= 8 then 26916
when dlch.loc_coord_ouom_code= 9 then 26919
when dlch.loc_coord_ouom_code= 10 then 26917
when dlch.loc_coord_ouom_code= 11 then 26717
when dlch.loc_coord_ouom_code= 12 then 26717
when dlch.loc_coord_ouom_code= 13 then 32190
when dlch.loc_coord_ouom_code= 14 then 7991
when dlch.loc_coord_ouom_code= 15 then 32190
when dlch.loc_coord_ouom_code= 16 then 32196
when dlch.loc_coord_ouom_code= 17 then 26917
when dlch.loc_coord_ouom_code= 18 then 4267
when dlch.loc_coord_ouom_code= 19 then 4269
when dlch.loc_coord_ouom_code= 20 then 4269
when dlch.loc_coord_ouom_code= 21 then 4269
when dlch.loc_coord_ouom_code= 22 then 26917
when dlch.loc_coord_ouom_code= 23 then 26918
when dlch.loc_coord_ouom_code= 24 then 32617
when dlch.loc_coord_ouom_code= 25 then null
when dlch.loc_coord_ouom_code= 26 then 26917
when dlch.loc_coord_ouom_code= 27 then 26917
when dlch.loc_coord_ouom_code= 29 then null
when dlch.loc_coord_ouom_code= 31 then null
when dlch.loc_coord_ouom_code= 33 then null
when dlch.loc_coord_ouom_code= 35 then null
when dlch.loc_coord_ouom_code= 36 then 26917
else null
end as epsg_code_orig
.case
when dlch.loc_coord_ouom_code= 10 then 'UTM NAD83 specified; Z17 assumed'
when dlch.loc_coord_ouom_code= 11 then 'UTM NAD27 specified; Z17 assumed'
when dlch.loc_coord_ouom_code= 12 then 'UTM Z17 specified; NAD27 assumed'
when dlch.loc_coord_ouom_code= 13 then 'MTM specified; Z10 and NAD83 assumed'
when dlch.loc_coord_ouom_code= 20 then 'Lat/Long, Z17 specified; NAD83 assumed'
when dlch.loc_coord_ouom_code= 21 then 'Lat/Long specified; NAD83 assumed'
when dlch.loc_coord_ouom_code= 22 then 'Z17 specified; UTM and NAD83 assumed'
when dlch.loc_coord_ouom_code= 23 then 'Z18 specified; UTM and NAD83 assumed'
when dlch.loc_coord_ouom_code= 26 then 'EPSG_CODE 26911 specified; 26917 assumed'
when dlch.loc_coord_ouom_code= 27 then 'EPSG_CODE 26902 specified; 26917 assumed'
when dlch.loc_coord_ouom_code= 36 then 'UTM Z17 and NAD83 assumed'
else null

```

```

end as epsg_code_comment
.dlch.qa_coord_confidence_code as qa_coord_code
.dlch.coord_hist_loc_method as loc_coord_method
.dlch.coord_hist_comment as loc_coord_comment
.dlch.data_id as loc_coord_data_id
.dleh.loc_elev_code
.case
when dleh.loc_elev_hist_date is not null then dleh.loc_elev_hist_date
else dleh.sys_time_stamp
end as loc_elev_date
.dleh.loc_elev_masl as loc_elev
,6 as loc_elev_unit_code
.dleh.loc_elev_masl as loc_elev_ouom
,'masl' as loc_elev_unit_ouom
,null as loc_elev_method
,rtrim(dleh.elev_hist_comment) as loc_elev_comment
.dleh.data_id as loc_elev_data_id
.cast('20200714c' as varchar(255)) as sys_temp1
.cast(20200714 as int) as sys_temp2
.case
when dleh.sys_last_modified>dlch.sys_last_modified then dleh.sys_last_modified_by
else dlch.sys_last_modified_by
end as sys_last_modified_by
.case
when dleh.sys_last_modified>dlch.sys_last_modified then dleh.sys_last_modified
else dlch.sys_last_modified
end as sys_last_modified
.case
when dlch.sys_user_stamp is not null then dlch.sys_user_stamp
when dleh.sys_user_stamp is not null then dleh.sys_user_stamp
else null
end as sys_user_stamp
.case
when dlch.sys_time_stamp is not null then dlch.sys_time_stamp
when dleh.sys_time_stamp is not null then dleh.sys_time_stamp
else null
end as sys_time_stamp
from
d_location_coord_hist as dlch
inner join d_location_elev_hist as dleh
on dlch.loc_elev_id=dleh.loc_elev_id
where
dlch.current_coord is null

```

We can now look at the records from each of the D_LOCATION_COORD_HIST and D_LOCATION_ELEV_HIST that are unrelated (through a LOC_ELEV_ID); each of these will be handled separately. Examining the former, there appears to be less than 500 records with a subset of information available.

```

select
loc_id
,loc_coord_hist_code
,sys_time_stamp as loc_coord_date
,x_utmz17nad83 as x
,y_utmz17nad83 as y
,26917 as epsg_code
,x_ouom
,y_ouom
.case
when dlch.loc_coord_ouom_code= 3 then 26718
when dlch.loc_coord_ouom_code= 4 then 26917
when dlch.loc_coord_ouom_code= 5 then 26918
when dlch.loc_coord_ouom_code= 12 then 26717
when dlch.loc_coord_ouom_code= 17 then 26917
when dlch.loc_coord_ouom_code= 21 then 4269
when dlch.loc_coord_ouom_code= 22 then 26917
when dlch.loc_coord_ouom_code= 24 then 32617

```

```

when dlch.loc_coord_ouom_code= 36 then 26917
else null
end as epsg_code_orig
,qa_coord_confidence_code as qa_coord_code
,'No associated elevation specified' as loc_elev_comment
,'20200714d' as sys_temp1
,20200714 as sys_temp2
,sys_last_modified
,sys_last_modified_by
,sys_time_stamp
,sys_user_stamp
from
d_location_coord_hist as dlch
where
dlch.current_coord is null
and dlch.loc_elev_id is null

```

The elevation history table has many more records (due, mainly, to the disparate DEM surfaces referenced).

```

select
loc_id
,'No associated coordinates specified' as loc_coord_comment
,loc_elev_code
,case
when loc_elev_hist_date is not null then loc_elev_hist_date
else sys_time_stamp
end as loc_elev_date
,loc_elev_masl as loc_elev
,6 as loc_elev_unit_code
,loc_elev_masl as loc_elev_ouom
,'masl' as loc_elev_unit_ouom
,qa_elev_confidence_code as qa_elev_code
,elev_hist_comment as loc_elev_comment
,'20200714e' as sys_temp1
,20200714 as sys_temp2
,sys_last_modified
,sys_last_modified_by
,sys_time_stamp
,sys_user_stamp
from
d_location_elev_hist as dleh
left outer join
(
select
delev.loc_elev_id as p_loc_elev_id
from
d_location_elev as delev
union
select
dlch.loc_elev_id as p_loc_elev_id
from
d_location_coord_hist as dlch
where
dlch.loc_elev_id is not null
) as t
on dleh.loc_elev_id=t.p_loc_elev_id
where
t.p_loc_elev_id is null

```

Now for those locations that only have a single coordinate, we can associate those coordinates with these newly added elevations (that lack coordinates). This leaves approximately 50,000 records without related coordinates or elevations.

```

update d_location_spatial_hist

```

```

set
loc_coord_hist_code= t2.loc_coord_hist_code
loc_coord_date= t2.loc_coord_date
,x= t2.x
,y= t2.y
,epsg_code= t2.epsg_code
,x_ouom= t2.x_ouom
,y_ouom= t2.y_ouom
,epsg_code_orig= t2.epsg_code_orig
,epsg_code_comment= t2.epsg_code_comment
,qa_coord_code= t2.qa_coord_code
,loc_coord_method= t2.loc_coord_method
,loc_coord_comment= t2.loc_coord_comment
,sys_temp1= t2.sys_temp1
from
d_location_spatial_hist as dlsh2
inner join
(
select
dlsh.loc_id
,dlsh.spat_id
,dlch.loc_coord_hist_code
,dlch.sys_time_stamp as loc_coord_date
,dlch.x_utmz17nad83 as x
,dlch.y_utmz17nad83 as y
,26917 as epsg_code
,dlch.x_ouom
,dlch.y_ouom
,case
when dlch.loc_coord_ouom_code= 1 then null
when dlch.loc_coord_ouom_code= 2 then 26717
when dlch.loc_coord_ouom_code= 3 then 26718
when dlch.loc_coord_ouom_code= 4 then 26917
when dlch.loc_coord_ouom_code= 5 then 26918
when dlch.loc_coord_ouom_code= 6 then 26910
when dlch.loc_coord_ouom_code= 7 then 26915
when dlch.loc_coord_ouom_code= 8 then 26916
when dlch.loc_coord_ouom_code= 9 then 26919
when dlch.loc_coord_ouom_code= 10 then 26917
when dlch.loc_coord_ouom_code= 11 then 26717
when dlch.loc_coord_ouom_code= 12 then 26717
when dlch.loc_coord_ouom_code= 13 then 32190
when dlch.loc_coord_ouom_code= 14 then 7991
when dlch.loc_coord_ouom_code= 15 then 32190
when dlch.loc_coord_ouom_code= 16 then 32196
when dlch.loc_coord_ouom_code= 17 then 26917
when dlch.loc_coord_ouom_code= 18 then 4267
when dlch.loc_coord_ouom_code= 19 then 4269
when dlch.loc_coord_ouom_code= 20 then 4269
when dlch.loc_coord_ouom_code= 21 then 4269
when dlch.loc_coord_ouom_code= 22 then 26917
when dlch.loc_coord_ouom_code= 23 then 26918
when dlch.loc_coord_ouom_code= 24 then 32617
when dlch.loc_coord_ouom_code= 25 then null
when dlch.loc_coord_ouom_code= 26 then 26917
when dlch.loc_coord_ouom_code= 27 then 26917
when dlch.loc_coord_ouom_code= 29 then null
when dlch.loc_coord_ouom_code= 31 then null
when dlch.loc_coord_ouom_code= 33 then null
when dlch.loc_coord_ouom_code= 35 then null
when dlch.loc_coord_ouom_code= 36 then 26917
else null
end as epsg_code_orig
,case
when dlch.loc_coord_ouom_code= 10 then 'UTM NAD83 specified; Z17 assumed'
when dlch.loc_coord_ouom_code= 11 then 'UTM NAD27 specified; Z17 assumed'
when dlch.loc_coord_ouom_code= 12 then 'UTM Z17 specified; NAD27 assumed'
when dlch.loc_coord_ouom_code= 13 then 'MTM specified; Z10 and NAD83 assumed'
when dlch.loc_coord_ouom_code= 20 then 'Lat/Long, Z17 specified; NAD83 assumed'
when dlch.loc_coord_ouom_code= 21 then 'Lat/Long specified; NAD83 assumed'

```

```

when dlch.loc_coord_ouom_code= 22 then 'Z17 specified; UTM and NAD83 assumed'
when dlch.loc_coord_ouom_code= 23 then 'Z18 specified; UTM and NAD83 assumed'
when dlch.loc_coord_ouom_code= 26 then 'EPSG_CODE 26911 specified; 26917 assumed'
when dlch.loc_coord_ouom_code= 27 then 'EPSG_CODE 26902 specified; 26917 assumed'
when dlch.loc_coord_ouom_code= 36 then 'UTM Z17 and NAD83 assumed'
else null
end as epsg_code_comment
,dlch.qa_coord_confidence_code as qa_coord_code
,dlch.coord_hist_loc_method as loc_coord_method
,dlch.coord_hist_comment as loc_coord_comment
,'20200714f' as sys_temp1
from
d_location_spatial_hist as dlsh
inner join d_location_coord_hist as dlch
on dlsh.loc_id=dlch.loc_id
inner join
(
select
loc_id
,count(*) as rcount
from
d_location_coord_hist
group by
loc_id
) as t
on dlch.loc_id=t.loc_id
where
t.rcount=1
and dlsh.sys_temp1='20200714e'
and dlsh.x is null and dlsh.y is null
) as t2
on dlsh2.spat_id=t2.spat_id

```

G.32 Automated Scripts (Listing and Calling Order)

Tables

- D_BOREHOLE
- D_INTERVAL_FORM_ASSIGN
- D_INTERVAL_FORM_ASSIGN_FINAL
- D_INTERVAL_MONITOR
- D_INTERVAL_SUMMARY
- D_INTERVAL_TEMPORAL_2
- D_LOCATION_GEOM
- D_LOCATION_SUMMARY
- D_VERSION_STATUS
- W_GENERAL
- W_GENERAL_DOCUMENT
- W_GENERAL_GW_LEVEL
- W_GENERAL_LOC_MET
- W_GENERAL_LOC_SW
- W_GENERAL_OTHER
- W_GENERAL_SCREEN
- W_GEOLOGY_LAYER

Recurrence Time: Weekly

A variety of automated updates are performed against the database on a weekly basis (beginning on Saturday and ending on Sunday; note that the start times, listed subsequently, are relative to midnight Saturday.) These can be broken into the following general groups (as listed in processing order)

- Update D_INTERVAL_FORM_ASSIGN (and related)
- Update D_LOCATION_SUMMARY (location by area)
- Update D_INTERVAL_SUMMARY
- Update D_LOCATION_SUMMARY
- Update D_LOCATION_GEOM
- Update D_VERSION_STATUS
- Miscellaneous updates
- Update all W_GENERAL_* tables
- Database backup (and restore)
- Miscellaneous daily updates

G.32.1 Update D_INTERVAL_FORM_ASSIGN (and related)

Main Script: d_int_form_ass.bat

Starting Time: -05:30

The following scripts are called (listed in order of processing)

- Removal of those INT_IDs not present in CM2004 (rem_from_difa_cm2004.bat)
- Add INT_IDs present in CM2004 (add_to_difa_cm2004.bat)
- Create a temporary table containing CM2004 information (tmp_cm2004_int.bat)
- Update D_INTERVAL_FORM_ASSIGN for the CM2004 model and remove the temporary table (upd_difa_cm2004.bat)

This is repeated for each geologic model being evaluated (currently WB2018 and YT32011). This will have a similar naming structure as listed above.

The ASSIGNED_UNIT is then populated for each geologic model (upd_difa_au.bat).

The thicknesses for each aquifer unit for each geologic model is then determined for each interval/location using the following scripts

- Get the CM2004 ORAC thickness (tmp_gl_cm2004_orac.bat)
- Update the THICKNESS_M field for the applicable records (upd_difa_cm2004_orac.bat)
- Get the CM2004 Thorncliffe thickness (tmp_gl_cm2004_thorn.bat)

- Update the THICKNESS_M field for the applicable records (upd_difa_cm2004_thorn.bat)
- Get the CM2004 Scarborough thickness (tmp_gl_cm2004_scar.bat)
- Update the THICKNESS_M field for the applicable records (upd_difa_cm2004_scar.bat)
- Get the CM2004 Channel Sand thickness (tmp_gl_cm2004_chansa.bat)
- Update the THICKNESS_M field for the applicable records (upd_difa_cm2004_chansa.bat)

This is repeated for each geologic model and their associated aquifers (WB2018: ORAC, Thorncliffe and Scarborough; YT32011: ORAC, Thorncliffe, Scarborough and Channel Sands). These will have a similar naming scheme to the above.

The values of specific capacity (SC_LPMM), transmissivity (T) and hydraulic conductivity (K) can then be calculated (upd_difa_calc_tk.bat). The associated fields will be updated.

The D_INTERVAL_FORM_ASSIGN_FINAL table can be updated, including

- Removal of INT_IDs (for those not found in D_INTERVAL_FORM_ASSIGN and will NULL values for OVERRIDE_UNIT and MANUAL_UNIT; rem_from_difa_f.bat)
- Add missing INT_IDs (i.e. present in D_INTERVAL_FORM_ASSIGN and absent from D_INTERVAL_FORM_ASSIGN_FINAL; add_to_difa_f.bat)
- Update the ASSIGNED_UNIT field (upd_au_difa_f.bat)

G.32.2 Update D_LOCATION_SUMMARY (location by area)

Main Script: d_loc_summary_area.bat

Starting Time: 00:00

Each partner conservation area and region is examined and the associated records have their CA_AREA_ID, REG_AREA_ID and SWP_AREA_ID fields updated in D_LOCATION_SUMMARY. This is determined using the following scripts (listed in order of processing)

- CLOCA (cloca_area_d_loc_sum.bat)
- CVC (cvc_area_d_loc_sum.bat)
- Durham Region (durham_area_d_loc_sum.bat)
- GRCA (grca_area_d_loc_sum.bat)
- Halton Region (halton_area_d_loc_sum.bat)
- KCA (kca_area_d_loc_sum.bat)
- LSRCA (lsrca_area_d_loc_sum.bat)
- LTRCA (ltrca_area_d_loc_sum.bat)
- NVCA (nvca_area_d_loc_sum.bat)

- ORVA (orca_area_d_loc_sum.bat)
- Peel Region (peel_area_d_loc_sum.bat)
- SWP CTC (swp_ctc_area_d_loc_sum.bat)
- SWP LS (swp_ls_area_d_loc_sum.bat)
- SWP Trent (swp_trent_area_d_loc_sum.bat)
- City of Toronto (Toronto_area_d_loc_sum.bat)
- TRCA (trca_area_d_loc_sum.bat)
- York Region (York_area_d_loc_sum.bat)

G.32.3 Update D_INTERVAL_SUMMARY

Main Script: d_int_sum.bat

Starting Time: 02:00

Records and fields found in D_INTERVAL_SUMMARY (DIS) are now updated (in order of processing)

- Add missing INT_IDs (add_to_d_int_sum.bat)
- Remove INT_IDs not found in D_INTERVAL (rem_from_d_int_sum.bat)
- Update water levels (update_d_int_sum_wl.bat)
- Update manual water levels (update_d_int_sum_wl_man.bat)
- Update logger water levels (update_d_int_sum_wl_log.bat)
- Update average water levels (update_d_int_sum_wl_avg.bat)
- Update water quality (update_d_int_sum_wa.bat)
- Update water quality samples (update_d_int_sum_wq_samp.bat)
- Update precipitation (update_d_int_sum_precip.bat)
- Update pumping readings (update_d_int_sum_pump.bat)
- Update pumping daily volumnes (update_d_int_sum_pump_daily_vol.bat)
- Update streamflow readings (update_d_int_sum_sflow.bat)
- Update specific capacity (update_d_int_sum_spec_cap.bat)
- Update air temperature readings (update_d_int_sum_temp_air.bat)

G.32.4 Update D_LOCATION_SUMMARY

Main Script: d_loc.sum.bat

Starting Time: 02:30

Records and fields found in D_LOCATION_SUMMARY (DLS) are now updated (in order of processing)

- Add missing LOC_IDs (add_to_d_loc_sum.bat)
- Remove LOC_IDs not found in D_LOCATION (rem_from_d_loc_sum.bat)

- Update deepest screen top elevation
(update_d_loc_sum_deep_scr_top.bat)
- Update number of geologic layers
(update_d_loc_sum_geol_lay_num.bat)
- Update number of monitors/screens present
(update_d_loc_sum_mon_num.bat)
- Update total number of water levels (update_d_loc_sum_wl_total.bat)
- Update total number of water quality readings
(update_d_loc_sum_wa_total.bat)
- Update total number of water quality samples
(update_d_loc_sum_wq_total_samp.bat)
- Update total number of precipitation readings
(update_d_loc_sum_precip_total.bat)
- Update total number of pump readings
(update_d_loc_sum_pump_total.bat)
- Update total number of streamflow readings
(update_d_loc_sum_sflow_total.bat)
- Update minimum, maximum and average streamflow readings
(update_d_loc_sum_sflow_avgminmax.bat)
- Update soil readings (update_d_loc_sum_soil.bat)
- Update air temperature (update_d_loc_sum_temp_air.bat)

G.32.5 Update D_LOCATION_GEOM

Main Script: d_loc_geom.bat

Starting Time: 03:00

Records and fields found in D_LOCATION_GEOM (DLG) are now updated (in order of processing)

- Remove LOC_IDs missing from D_LOCATION
(rem_from_d_loc_geom.bat)
- Check current coordinates (in D_LOCATION) against the current calculated geometry (tag them as necessary; coord_check_d_loc_geom.bat)
- Add any new LOC_IDs (add_to_d_loc_geom.bat)
- Update any blank GEOM fields (upd_d_loc_geom.bat)
- Update any blank GEOM_WKB fields (update_d_loc_geom_wkb.bat)

G.32.6 Miscellaneous updates

Main Script: various.bat

Starting Time: 03:30

A variety of checks and updates are performed here (in a number of tables, in order of processing)

- Update locations in D_BOREHOLE that no longer have a bedrock formation (rem_bed_elev.bat)
- Update locations in D_BOREHOLE that have a bedrock formation (upd_bed_elev.bat)
- Add new LOC_IDs (including coordinates and elevations) to D_LOCATION_SPATIAL_HIST (add_to_d_loc_spat_hist.bat)
- Update D_LOCATION_SPAT adding newly assigned coordinates and elevations (add_to_d_loc_spat.bat)
- Update the LOC_ACTIVE field for PTTW locations (upd_pttw_active.bat)
- Update the LOC_STATUS_CODE for climate stations (upd_climate_active.bat)
- Update the LOC_STATUS_CODE for spotflow locations (upd_sw_spotflow_active.bat)
- Update the LOC_STATUS_CODE for streamflow gauges (upd_sw_gauge_active.bat)
- Update MON_TOP_DEPTH_M and MON_BOT_DEPTH_M in D_INTERVAL_MONITOR (upd_dim_depths.bat)
- Update the reference elevations and stick-ups in D_INTERVAL_REF_ELEV based upon a change in REF_POINT (upd_dire_stick_up.bat)

G.32.7 Update all W_GENERAL_* tables

Main Script: w_gen_all.bat

Starting Time: 04:00

The contents of all the W_GENERAL_* tables are removed and the tables re-populated (in order of processing)

- Delete all rows in W_GENERAL (WG; rem_from_w_gen.bat)
- Add all rows to WG (add_to_w_gen.bat)
- Delete all rows in W_GENERAL_DOCUMENT (WGC; rem_from_w_gen_doc.bat)
- Add all rows to WGC (add_to_w_gen_doc.bat)
- Delete all rows in W_GENERAL_OTHER (WGO; rem_from_w_gen_other.bat)
- Add all rows to WGO (add_to_w_gen_other.bat)
- Modify STATUS of PTTW locations in WGO (update_w_gen_other_pttw_active.bat)
- Delete all rows in W_GENERAL_SCREEN (WGS; rem_from_w_gen_scr.bat)
- Add all rows to WGS (add_to_w_gen_scr.bat)

- Delete all rows in W_GENERAL_GW_LEVEL (WGGL; rem_from_w_gen_gw_level.bat)
- Add daily logger water levels to WGGL (add_to_w_gen_gw_level_log.bat)
- Add all manual water levels to WGGL (add_to_w_gen_gw_level_man.bat)
- Delete all rows in W_GENERAL_LOC_MET (WGLM; rem_from_w_gen_loc_met.bat)
- Add all rows to WGLM (add_to_w_gen_loc_met.bat)
- Delete all rows in W_GENERAL_LOC_SW (WGLS; rem_from_w_gen_loc_sw.bat)
- Add all rows to WGLS (add_to_w_gen_loc_sw.bat)
- Add URLs for non-MOE borehole PDFs (in W_GENERAL and W_GENERAL_SCREEN; upd_w_gen_nonmoe_bh_pdf.bat)
- Update those locations that can be used for Piper plots (using the field WQ_CB_PIPER; upd_wq_cb_piper.bat)
- Update SPEC_CAP_LPMM in W_GENERAL_SCREEN (upd_spec_cap_lpmm.bat)
- Delete all rows from W_GEOLOGY_LAYER (WGL; rem_from_w_geol_layer.bat)
- Add all rows to WGL (add_to_w_geol_layer.bat)
- Create a temporary table containing shallow water levels (wls_create_tmp.bat)
- Update SHALLOW_WL_MASL and SHALLOW_WL_DEPTH_M in W_GENERAL; drop the temporary table (wls_update.bat)

G.32.8 Update D_VERSION_STATUS

Main Script: d_var_stat.bat

Starting Time: 5:00

Records are added to the D_VERSION_STATUS table capturing the status of the database. This includes:

- Adding location type counts (add_loc_type_counts.bat)
- Adding interval type counts (add_int_type_counts.bat)
- Adding reading group type counts (add_group_type_counts.bat)

G.32.9 Database backup (and restore)

Main Script: see below

Starting Time: 12:00

Each of the following are processed (in turn) to backup the master database and make available the OAK_20160831_WEEKLY database. This includes

- Update the OAK_20160831_WEEKLY database from the backup of OAK_20160831_MASTER (db_restore.bat) [12:00]
- Move the OAK_20160831_MASTER backup file to the B: drive on SQLSERVER2k16 (i.e. the temporary backup location; db_backup.bat) [14:00]
- Modify the permissions of various users to allow full access to OAK_20160831_WEEKLY (db_restore_permissions.bat) [14:55]
- Split the backup of the master database to allow easy transfer off-site (db_backup_split.bat) [15:30; now disabled]
- Move the backup file of the master database to B:\backup (i.e. the final backup location; db_backup_final.bat)

G.32.10 Miscellaneous daily updates

Main script: various_daily.bat

Starting time (daily): 22:00 (10:00pm)

Each of the following is processed, in turn:

- Update those records in D_INTERVAL_TEMPORAL_2 that have 'cmap' or 'map' as their original RD_UNIT_OUOM value; converts these to 'mbref'

G.33 Update of D_AREA_GEOM

Tables

- D_AREA_GEOM

Views

- V_SYS_AREA_GEOM_WKB

Estimated Recurrence Time: As required.

The D_AREA_GEOM contains the boundaries (as polygons) for the various partner agencies, the ORMGP study area as well as the extent of geologic and hydrogeologic models. Many of these include a buffered region as well. This information is stored in two columns: GEOM and GEOM_WKB. The former holds the default 'geometry' type for Microsoft SQL Server while the latter has been stored in the 'Well Known Binary' format.

The GEOM field is populated through any external GIS software that supports the native geometry format. The GEOM_WKB is then populated using V_SYS_AREA_GEOM_WKB (using GEOM as a source).

Projections can be checked through

```
select
area_id
,geom.STSrid
From
d_area_geom
```

which returns the EPSG projection numeric for each record/object. This should conform to the code '26917' (UTM Zone 17, NAD83). If these have been incorrectly assigned (from the external source), the specific records can be updated using

```
update d_area_geom
set
geom.STSrid= 26917
where
area_id= 76
```

The example here is updating the ORMGP Boundary 5km buffer layer.

Purposes - Assigned

BH STATUS CODE	BH STATUS DESCRIPTION	MOC 1ST USE	MOC 1ST USE DESCRIPTION	MOC 2ND USE	MOC 2ND USE DESCRIPTION	Primary Purpose	Primary Purpose - Description	Secondary Purpose	Secondary Purpose - Description	Note
		0		0		11	Unknown	71	Unknown	Added 20210126
1	WATER SUPPLY	0		0		10	Water Supply	33	Other Water Supply	Added 20210126
2	OBSERVATION WELLS	0		0		3	Engineering	51	Monitoring Well	Added 20210126
3	TEST HOLE	0		0		3	Engineering	30	Other Miscellaneous	Added 20210126
5	ABANDONED-SUPPLY	0		0		10	Water Supply	33	Other Water Supply	Added 20210126
6	ABANDONED-QUALITY	0		0		10	Water Supply	33	Other Water Supply	Added 20210126
9	DEWATERING	0		0		4	Dewatering	27	Other Dewatering	Added 20210126
10	ABANDONED-OTHER	0		0		11	Unknown	71	Unknown	Added 20210126
11	REPLACEMENT WELL	0		0		11	Unknown	71	Unknown	Added 20210126
12	ALTERATION	0		0		11	Unknown	71	Unknown	Added 20210126
13	OTHER STATUS	0		0		11	Unknown	71	Unknown	Added 20210126
14	Monitoring and Test Hole	0		0		3	Engineering	51	Monitoring Well	Added 20210126
15	Abandoned Monitoring and Test Hole	0		0		3	Engineering	51	Monitoring Well	Added 20210126
1	WATER SUPPLY	0	1	DOMESTIC	10	Water Supply	33	Other Water Supply	Added 20210126	
2	OBSERVATION WELLS	0	1	DOMESTIC	3	Engineering	51	Monitoring Well	Added 20210126	
3	TEST HOLE	0	1	DOMESTIC	11	Unknown	71	Unknown	Added 20210126	
10	ABANDONED-OTHER	0	1	OTHER	11	Unknown	71	Unknown	Added 20210126	
2	OBSERVATION WELLS	0	10	TEST HOLE	3	Engineering	51	Monitoring Well	Added 20210126	
3	TEST HOLE	0	10	TEST HOLE	3	Engineering	51	Monitoring Well	Added 20210126	
2	OBSERVATION WELLS	0	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20210126	
3	TEST HOLE	0	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20210126	
10	ABANDONED-OTHER	0	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20210126	
1	WATER SUPPLY	1	DOMESTIC	0		10	Water Supply	49	Domestic	Added 20210121
2	OBSERVATION WELLS	1	DOMESTIC	0		10	Water Supply	49	Domestic	Added 20210121
3	TEST HOLE	1	DOMESTIC	0		3	Engineering	51	Monitoring Well	Added 20210120
4	RECHARGE WELL	1	DOMESTIC	0		3	Engineering	47	Geotest Testhold	Added 20210121
5	ABANDONED-SUPPLY	1	DOMESTIC	0		10	Water Supply	33	Other Water Supply	Added 20210121
6	ABANDONED-QUALITY	1	DOMESTIC	0		3	Engineering	51	Monitoring Well	Added 20210121
9	DEWATERING	1	DOMESTIC	0		4	Dewatering	27	Other Dewatering	Added 20210121
10	ABANDONED-OTHER	1	DOMESTIC	0		10	Water Supply	49	Domestic	Added 20210121
11	REPLACEMENT WELL	1	DOMESTIC	0		10	Water Supply	49	Domestic	Added 20210121
12	ALTERATION	1	DOMESTIC	0		3	Engineering	49	Domestic	Added 20210121
13	OTHER STATUS	1	DOMESTIC	0		10	Water Supply	49	Domestic	Added 20210121
1	WATER SUPPLY	1	DOMESTIC	1	DOMESTIC	10	Water Supply	49	Domestic	
11	REPLACEMENT WELL	1	DOMESTIC	2	LIVESTOCK	10	Water Supply	49	Domestic	
2	OBSERVATION WELLS	1	DOMESTIC	3	INDUSTRIAL	3	Engineering	28	Other Industrial	
15	Abandoned Monitoring and Test Hole	1	DOMESTIC	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	Added 20170911
1	WATER SUPPLY	1	DOMESTIC	5	COMMERCIAL	10	Water Supply	33	Other Water Supply	Added 20210121
4	RECHARGE WELL	1	DOMESTIC	5	COMMERCIAL	3	Engineering	25	Other Commercial	Added 20210121
11	REPLACEMENT WELL	1	DOMESTIC	5	COMMERCIAL	10	Water Supply	49	Domestic	Added 20210121
12	ALTERATION	1	DOMESTIC	6	COMMERCIAL	10	Water Supply	25	Other Commercial	Added 20170911
1	WATER SUPPLY	1	DOMESTIC	6	MUNICIPAL	10	Water Supply	22	Municipal	
3	TEST HOLE	1	DOMESTIC	7	PUBLIC SUPPLY	10	Water Supply	33	Other Water Supply	
5	ABANDONED-SUPPLY	1	DOMESTIC	7	PUBLIC SUPPLY	10	Water Supply	33	Other Water Supply	
11	REPLACEMENT WELL	1	DOMESTIC	7	PUBLIC SUPPLY	10	Water Supply	33	Other Water Supply	Added 20200731
4	RECHARGE WELL	1	DOMESTIC	7	PUBLIC SUPPLY	10	Water Supply	33	Other Water Supply	Added 20210121
1	WATER SUPPLY	1	DOMESTIC	8	COOLING OR A/C	3	Engineering	49	Domestic	
13	OTHER STATUS	1	DOMESTIC	9	NOT USED	10	Water Supply	33	Other Water Supply	Added 20180530
9	DEWATERING	1	DOMESTIC	10	OTHER	4	Dewatering	27	Other Dewatering	Added 20190509
2	OBSERVATION WELLS	1	DOMESTIC	11	TEST HOLE	3	Engineering	51	Monitoring Well	Added 20210121
3	TEST HOLE	1	DOMESTIC	11	TEST HOLE	3	Engineering	51	Monitoring Well	Added 20210121
1	WATER SUPPLY	1	DOMESTIC	12	DEWATERING	10	Water Supply	27	Other Dewatering	Added 20180530
9	DEWATERING	1	DOMESTIC	12	DEWATERING	4	Dewatering	27	Other Dewatering	Added 20180530
2	OBSERVATION WELLS	1	DOMESTIC	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20210121
3	TEST HOLE	1	DOMESTIC	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20210121
9	DEWATERING	1	DOMESTIC	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20210121
1	WATER SUPPLY	1	LIVESTOCK	0		1	Agriculture	50	Stock	Added 20210121
4	RECHARGE WELL	2	LIVESTOCK	0		1	Agriculture	50	Stock	Added 20210121
5	ABANDONED-SUPPLY	2	LIVESTOCK	0		1	Agriculture	50	Stock	Added 20210121
6	ABANDONED-QUALITY	2	LIVESTOCK	0		1	Agriculture	50	Stock	Added 20210121
10	ABANDONED-OTHER	2	LIVESTOCK	0		1	Agriculture	50	Stock	Added 20210121
11	REPLACEMENT WELL	2	LIVESTOCK	0		1	Agriculture	50	Stock	Added 20210121
12	ALTERATION	2	LIVESTOCK	0		1	Agriculture	50	Stock	Added 20210121
13	OTHER STATUS	2	LIVESTOCK	1	DOMESTIC	1	Agriculture	50	Stock	
1	WATER SUPPLY	2	LIVESTOCK	3	IRRIGATION	1	Agriculture	50	Stock	Added 20170911
11	REPLACEMENT WELL	2	LIVESTOCK	4	INDUSTRIAL	1	Agriculture	50	Stock	
12	ALTERATION	2	LIVESTOCK	4	INDUSTRIAL	1	Agriculture	50	Stock	
1	WATER SUPPLY	2	LIVESTOCK	5	COMMERCIAL	1	Agriculture	50	Stock	
5	ABANDONED-SUPPLY	2	LIVESTOCK	9	NOT USED	1	Agriculture	50	Stock	Added 20170911
10	ABANDONED-OTHER	2	LIVESTOCK	9	NOT USED	1	Agriculture	50	Stock	Added 20170911
1	WATER SUPPLY	3	IRRIGATION	0		1	Agriculture	24	Other Agricultural	Added 20210121
2	OBSERVATION WELLS	3	IRRIGATION	0		1	Agriculture	24	Other Agricultural	Added 20210121
11	REPLACEMENT WELL	3	IRRIGATION	0		1	Agriculture	24	Other Agricultural	Added 20210121
12	ALTERATION	3	IRRIGATION	0		1	Agriculture	24	Other Agricultural	Added 20210121
1	WATER SUPPLY	3	IRRIGATION	4	INDUSTRIAL	3	Engineering	83	Recharge Well	Added 20210121
5	ABANDONED-SUPPLY	3	IRRIGATION	4	INDUSTRIAL	1	Agriculture	24	Other Agricultural	Added 20210121
10	ABANDONED-OTHER	3	IRRIGATION	4	INDUSTRIAL	1	Agriculture	24	Other Agricultural	Added 20180530
11	REPLACEMENT WELL	3	IRRIGATION	4	INDUSTRIAL	1	Agriculture	24	Other Agricultural	Added 20210121
12	ALTERATION	3	IRRIGATION	4	INDUSTRIAL	1	Agriculture	24	Other Agricultural	Added 20200731
1	WATER SUPPLY	3	IRRIGATION	5	COMMERCIAL	1	Agriculture	25	Other Commercial	Added 20180530
1	WATER SUPPLY	3	IRRIGATION	6	MUNICIPAL	10	Water Supply	33	Other Water Supply	Added 20210121
1	WATER SUPPLY	3	NOT USED	6	MUNICIPAL	10	Water Supply	77	Not Used	Added 20170911
3	TEST HOLE	3	NOT USED	6	MUNICIPAL	10	Water Supply	59	Municipal Exploration	Added 20170911
1	WATER SUPPLY	3	IRRIGATION	11	TEST HOLE	3	Engineering	33	Other Water Supply	
3	TEST HOLE	3	IRRIGATION	12	TEST HOLE	3	Engineering	33	Other Water Supply	Added 20200731
1	WATER SUPPLY	4	INDUSTRIAL	0		5	Industrial	28	Other Industrial	Added 20210121
1	WATER SUPPLY	4	INDUSTRIAL	0		10	Water Supply	28	Other Industrial	Added 20210121
5	ABANDONED-SUPPLY	4	INDUSTRIAL	0		10	Water Supply	28	Other Industrial	Added 20210121
10	ABANDONED-OTHER	4	INDUSTRIAL	1	DOMESTIC	10	Water Supply	33	Other Water Supply	Added 20210121
5	ABANDONED-SUPPLY	4	INDUSTRIAL	2	LIVESTOCK	10	Water Supply	33	Other Water Supply	Added 20210121
14	Monitoring and Test Hole	4	INDUSTRIAL	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	Added 20210121
15	Abandoned Monitoring and Test Hole	4	INDUSTRIAL	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	Added 20210121
1	WATER SUPPLY	4	INDUSTRIAL	5	COMMERCIAL	10	Water Supply	33	Other Water Supply	
5	ABANDONED-SUPPLY	4	INDUSTRIAL	5	COMMERCIAL	10	Water Supply	33	Other Water Supply	
2	OBSERVATION WELLS	4	INDUSTRIAL	6	MUNICIPAL	10	Water Supply	51	Monitoring Well	Added 20170911
9	DEWATERING	4	INDUSTRIAL	6	MUNICIPAL	4	Dewatering	27	Other Dewatering	
1	WATER SUPPLY	4	INDUSTRIAL	7	PUBLIC SUPPLY	5	Industrial	33	Other Water Supply	Added 20180530
1	WATER SUPPLY	4	INDUSTRIAL	7	PUBLIC SUPPLY	10	Water Supply	28	Other Industrial	Added 20190509
13	OTHER STATUS	4	INDUSTRIAL	9	NOT USED	10	Water Supply	28	Other Industrial	Added 20170911
13	OTHER STATUS	4	INDUSTRIAL	9	NOT USED	5	Industrial	28	Other Industrial	Added 20210121
1	WATER SUPPLY	4	INDUSTRIAL	10	OTHER	5	Industrial	28	Other Industrial	Added 20170911
2	OBSERVATION WELLS	4	INDUSTRIAL	11	TEST HOLE	10	Water Supply	28	Other Industrial	Added 20210121
2	OBSERVATION WELLS	4	INDUSTRIAL	11	TEST HOLE	5	Industrial	51	Monitoring Well	Added 20190509
9	DEWATERING	4	INDUSTRIAL	12	DEWATERING	3	Engineering	27	Other Dewatering	
9	DEWATERING	4	INDUSTRIAL	12	DEWATERING	4	Dewatering	27	Other Dewatering	
10	ABANDONED-OTHER	4	INDUSTRIAL	12	DEWATERING	4	Dewatering	27	Other Dewatering	
5	ABANDONED-SUPPLY	4	INDUSTRIAL	13	MONITORING	5	Industrial	33	Other Water Supply	Added 20190509
10	ABANDONED-OTHER	4	INDUSTRIAL	13	MONITORING	5	Industrial	51	Monitoring Well	
14	Monitoring and Test Hole	4	INDUSTRIAL	13	MONITORING	5	Industrial	51	Monitoring Well	Added 20180530
15	Abandoned Monitoring and Test Hole	4	INDUSTRIAL	13	MONITORING	5	Industrial	51	Monitoring Well	Added 20190509
1	WATER SUPPLY	5	COMMERCIAL	0		10	Water Supply	25	Other Commercial	Added 20210121
2	OBSERVATION WELLS	5	COMMERCIAL	3	OTHER	3	Engineering	25	Other Commercial	Added 20210121
5	ABANDONED-SUPPLY	5	COMMERCIAL	0		10	Water Supply	25	Other Commercial	Added 20210121
6	ABANDONED-QUALITY	5	COMMERCIAL	0		3	Engineering	25	Other Commercial	Added 20210121
9	DEWATERING	5	COMMERCIAL	0		4	Dewatering	25	Other Commercial	Added 20210121
10	ABANDONED-OTHER	5	COMMERCIAL	0		2	Commercial	25	Other Commercial	Added 20210121
11	REPLACEMENT WELL	5	COMMERCIAL	0		2	Commercial	25	Other Commercial	Added 20210121
12	ALTERATION	5	COMMERCIAL	0		2	Commercial	25	Other Commercial	Added 20210121
1	WATER SUPPLY	5	COMMERCIAL	1	DOMESTIC	10	Water Supply	25	Other Commercial	Added 20170911
2	OBSERVATION WELLS	5	COMMERCIAL	3	IRRIGATION	1	Agriculture	25	Other Commercial	Added 20190509
3	TEST HOLE	5	COMMERCIAL	4	INDUSTRIAL	3	Engineering	25	Other Commercial	
4	RECHARGE WELL	5	COMMERCIAL	4	INDUSTRIAL	3	Engineering	25	Other Commercial	
11	REPLACEMENT WELL	5	COMMERCIAL	4	INDUSTRIAL	10	Water Supply	25	Other Commercial	
12	ALTERATION	5	COMMERCIAL	4	INDUSTRIAL	10	Water Supply	25	Other Commercial	Added 20180530
13	OTHER STATUS	5	COMMERCIAL	4	INDUSTRIAL	10	Water Supply	25	Other Commercial	
5	ABANDONED-SUPPLY	5	COMMERCIAL	9	NOT USED	10	Water Supply	25	Other Commercial	Added 20170911

10	ABANDONED-OTHER	5	COMMERCIAL	9	NOT USED	2	Commercial	25	Other Commercial	
1	WATER SUPPLY	5	COMMERCIAL	10	OTHER	10	Water Supply	25	Other Commercial	Added 20210121
10	ABANDONED-OTHER	5	COMMERCIAL	10	OTHER	10	Water Supply	25	Other Commercial	Added 20210121
1	WATER SUPPLY	5	COMMERCIAL	11	TEST HOLE	3	Engineering	51	Monitoring Well	Added 20170911
2	OBSERVATION WELLS	5	COMMERCIAL	11	TEST HOLE	3	Engineering	33	Other Water Supply	Added 20190509
2	OBSERVATION WELLS	5	COMMERCIAL	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20190509
1	WATER SUPPLY	6	MUNICIPAL	0	0	10	Water Supply	33	Other Water Supply	Added 20210121
2	OBSERVATION WELLS	6	MUNICIPAL	0	0	10	Water Supply	33	Other Water Supply	Added 20210121
3	TEST HOLE	6	MUNICIPAL	0	0	3	Engineering	58	Municipal Monitor	Added 20210121
4	RECHARGE WELL	6	MUNICIPAL	0	0	10	Water Supply	33	Other Water Supply	Added 20210121
9	DEWATERING	6	MUNICIPAL	0	0	4	Dewatering	33	Other Water Supply	Added 20210121
10	ABANDONED-OTHER	6	MUNICIPAL	0	0	10	Water Supply	33	Other Water Supply	Added 20210121
12	ALTERATION	6	MUNICIPAL	0	0	10	Water Supply	33	Other Water Supply	Added 20210121
1	WATER SUPPLY	6	MUNICIPAL	1	DOMESTIC	10	Water Supply	33	Other Water Supply	Added 20170911
4	RECHARGE WELL	6	MUNICIPAL	1	DOMESTIC	10	Water Supply	33	Other Water Supply	Added 20210121
13	OTHER STATUS	6	MUNICIPAL	4	INDUSTRIAL	10	Water Supply	22	Municipal Supply	
14	Monitoring and Test Hole	6	MUNICIPAL	4	INDUSTRIAL	10	Water Supply	58	Municipal Monitor	Added 20180530
9	DEWATERING	6	MUNICIPAL	8	COOLING OR A/C	4	Dewatering	27	Other Dewatering	
2	OBSERVATION WELLS	6	MUNICIPAL	9	NOT USED	10	Water Supply	58	Municipal Monitor	Added 20170911
1	WATER SUPPLY	6	MUNICIPAL	10	OTHER	10	Water Supply	22	Municipal Supply	
1	WATER SUPPLY	6	MUNICIPAL	11	TEST HOLE	10	Water Supply	50	Municipal Exploration	Added 20170911
14	Monitoring and Test Hole	6	MUNICIPAL	11	TEST HOLE	10	Water Supply	50	Municipal Exploration	Added 20190509
2	OBSERVATION WELLS	6	MUNICIPAL	12	DEWATERING	3	Engineering	58	Municipal Monitor	
9	DEWATERING	6	MUNICIPAL	12	DEWATERING	3	Dewatering	27	Other Dewatering	
10	ABANDONED-OTHER	6	MUNICIPAL	12	DEWATERING	10	Water Supply	27	Other Dewatering	
10	ABANDONED-OTHER	6	MUNICIPAL	13	MONITORING	10	Water Supply	58	Municipal Monitor	
14	Monitoring and Test Hole	6	MUNICIPAL	13	MONITORING	10	Water Supply	58	Municipal Monitor	Added 20170911
14	Monitoring and Test Hole	6	MUNICIPAL	14	Monitoring and Test Hole	10	Water Supply	58	Municipal Monitor	Added 20210121
1	WATER SUPPLY	7	PUBLIC SUPPLY	0	0	10	Water Supply	22	Municipal Supply	Added 20210122
5	ABANDONED-SUPPLY	7	PUBLIC SUPPLY	0	0	10	Water Supply	33	Other Water Supply	Added 20210122
10	ABANDONED-OTHER	7	PUBLIC SUPPLY	0	0	10	Water Supply	33	Other Water Supply	Added 20210122
11	REPLACEMENT WELL	7	PUBLIC SUPPLY	0	0	10	Water Supply	33	Other Water Supply	Added 20210121
11	REPLACEMENT WELL	7	PUBLIC SUPPLY	1	DOMESTIC	10	Water Supply	33	Other Water Supply	
13	OTHER STATUS	7	PUBLIC SUPPLY	1	DOMESTIC	10	Water Supply	33	Other Water Supply	Added 20210122
1	WATER SUPPLY	7	PUBLIC SUPPLY	3	IRRIGATION	10	Water Supply	24	Other Agricultural	Added 20170911
3	TEST HOLE	7	PUBLIC SUPPLY	4	INDUSTRIAL	10	Water Supply	28	Other Industrial	Added 20170911
4	RECHARGE WELL	7	PUBLIC SUPPLY	4	INDUSTRIAL	10	Water Supply	28	Other Industrial	Added 20180530
11	REPLACEMENT WELL	7	PUBLIC SUPPLY	4	INDUSTRIAL	10	Water Supply	28	Other Industrial	
13	OTHER STATUS	7	PUBLIC SUPPLY	4	INDUSTRIAL	10	Water Supply	28	Other Industrial	
1	WATER SUPPLY	7	PUBLIC SUPPLY	5	COMMERCIAL	10	Water Supply	25	Other Commercial	
10	ABANDONED-OTHER	7	PUBLIC SUPPLY	5	COMMERCIAL	10	Water Supply	25	Other Commercial	Added 20170911
3	TEST HOLE	7	PUBLIC SUPPLY	6	MUNICIPAL	10	Water Supply	59	Municipal Exploration	Added 20170911
10	ABANDONED-OTHER	7	PUBLIC SUPPLY	6	MUNICIPAL	10	Water Supply	22	Municipal Supply	Added 20190509
15	Abandoned Monitoring and Test Hole	7	PUBLIC SUPPLY	6	MUNICIPAL	3	Engineering	22	Municipal Supply	
2	OBSERVATION WELLS	7	PUBLIC SUPPLY	11	TEST HOLE	3	Engineering	51	Monitoring Well	Added 20210122
10	ABANDONED-OTHER	7	PUBLIC SUPPLY	13	MONITORING	10	Water Supply	24	Other Agricultural	Added 20170911
1	WATER SUPPLY	8	COOLING OR A/C	0	0	10	Water Supply	33	Other Water Supply	Added 20210122
4	RECHARGE WELL	8	COOLING OR A/C	0	0	10	Water Supply	33	Other Water Supply	Added 20210122
1	WATER SUPPLY	8	COOLING OR A/C	4	INDUSTRIAL	10	Water Supply	83	Recharge Well	Added 20210122
5	ABANDONED-SUPPLY	8	COOLING OR A/C	4	INDUSTRIAL	3	Engineering	28	Other Industrial	Added 20170911
2	OBSERVATION WELLS	8	COOLING OR A/C	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20180530
14	Monitoring and Test Hole	8	COOLING OR A/C	13	MONITORING	3	Engineering	51	Monitoring Well	
1	WATER SUPPLY	9	NOT USED	0	0	11	Unknown	71	Unknown	
1	WATER SUPPLY	9	NOT USED	0	0	10	Water Supply	33	Other Water Supply	Added 20210126
2	OBSERVATION WELLS	9	NOT USED	0	0	3	Engineering	51	Monitoring Well	Added 20210126
3	TEST HOLE	9	NOT USED	0	0	3	Engineering	51	Monitoring Well	Added 20210126
5	ABANDONED-SUPPLY	9	NOT USED	0	0	10	Water Supply	33	Other Water Supply	Added 20210126
6	ABANDONED-QUALITY	9	NOT USED	0	0	10	Water Supply	33	Other Water Supply	Added 20210126
10	ABANDONED-OTHER	9	NOT USED	0	0	10	Water Supply	33	Other Water Supply	Added 20210126
13	OTHER STATUS	9	NOT USED	0	0	11	Unknown	71	Unknown	Added 20210122
14	Monitoring and Test Hole	9	NOT USED	0	0	3	Engineering	51	Monitoring Well	Added 20210126
15	Abandoned Monitoring and Test Hole	9	NOT USED	0	0	3	Engineering	51	Monitoring Well	Added 20210126
10	ABANDONED-OTHER	9	NOT USED	1	DOMESTIC	10	Water Supply	33	Other Water Supply	Added 20210126
4	RECHARGE WELL	9	NOT USED	4	INDUSTRIAL	3	Engineering	83	Recharge Well	Added 20180530
11	REPLACEMENT WELL	9	NOT USED	4	INDUSTRIAL	5	Industrial	29	Other Industrial	
14	Monitoring and Test Hole	9	NOT USED	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	Added 20210126
15	Abandoned Monitoring and Test Hole	9	NOT USED	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	Added 20200731
2	OBSERVATION WELLS	9	NOT USED	10	OTHER	3	Engineering	51	Monitoring Well	
13	OTHER STATUS	9	NOT USED	10	OTHER	11	Unknown	71	Unknown	Added 20180530
1	WATER SUPPLY	10	OTHER	0	0	11	Unknown	71	Unknown	Added 20210126
1	WATER SUPPLY	10	OTHER	0	0	10	Water Supply	33	Other Water Supply	Added 20210126
2	OBSERVATION WELLS	10	OTHER	0	0	3	Engineering	51	Monitoring Well	Added 20210126
3	TEST HOLE	10	OTHER	0	0	3	Engineering	51	Monitoring Well	Added 20210126
4	RECHARGE WELL	10	OTHER	0	0	3	Engineering	83	Recharge Well	Added 20210126
5	ABANDONED-SUPPLY	10	OTHER	0	0	10	Water Supply	33	Other Water Supply	Added 20210126
6	ABANDONED-QUALITY	10	OTHER	0	0	10	Water Supply	33	Other Water Supply	Added 20210126
10	ABANDONED-OTHER	10	OTHER	0	0	10	Water Supply	33	Other Water Supply	Added 20210126
11	REPLACEMENT WELL	10	OTHER	0	0	10	Water Supply	30	Other - Miscellaneous	Added 20210126
12	ALTERATION	10	OTHER	0	0	11	Unknown	71	Unknown	Added 20210126
13	OTHER STATUS	10	OTHER	0	0	11	Unknown	71	Unknown	Added 20210126
14	Monitoring and Test Hole	10	OTHER	0	0	3	Engineering	51	Monitoring Well	Added 20210126
15	Abandoned Monitoring and Test Hole	10	OTHER	0	0	3	Engineering	51	Monitoring Well	Added 20210126
11	REPLACEMENT WELL	10	OTHER	4	INDUSTRIAL	3	Engineering	28	Other Industrial	Added 20180530
14	Monitoring and Test Hole	10	OTHER	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	Added 20180530
10	ABANDONED-OTHER	10	OTHER	9	NOT USED	10	Water Supply	33	Other Water Supply	Added 20170911
10	ABANDONED-OTHER	10	OTHER	10	OTHER	10	Water Supply	77	Not Used	
1	WATER SUPPLY	10	OTHER	11	TEST HOLE	10	Water Supply	33	Other Water Supply	Added 20170911
3	TEST HOLE	10	OTHER	11	TEST HOLE	3	Engineering	51	Monitoring Well	Added 20180530
14	Monitoring and Test Hole	10	OTHER	11	TEST HOLE	3	Engineering	51	Monitoring Well	
2	OBSERVATION WELLS	10	OTHER	13	MONITORING	3	Engineering	51	Monitoring Well	
6	ABANDONED-QUALITY	10	OTHER	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20180530
13	OTHER STATUS	10	OTHER	13	MONITORING	3	Engineering	51	Monitoring Well	
14	Monitoring and Test Hole	10	OTHER	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20210126
1	WATER SUPPLY	11	TEST HOLE	0	0	10	Water Supply	33	Other Water Supply	Added 20210126
2	OBSERVATION WELLS	11	TEST HOLE	0	0	3	Engineering	33	Monitoring Well	Added 20210126
3	TEST HOLE	11	TEST HOLE	0	0	3	Engineering	33	Monitoring Well	Added 20210126
5	ABANDONED-SUPPLY	11	TEST HOLE	0	0	10	Water Supply	33	Other Water Supply	Added 20210126
6	ABANDONED-QUALITY	11	TEST HOLE	0	0	10	Water Supply	33	Other Water Supply	Added 20210126
9	DEWATERING	11	TEST HOLE	0	0	3	Engineering	27	Other Dewatering	Added 20210126
10	ABANDONED-OTHER	11	TEST HOLE	0	0	3	Engineering	51	Monitoring Well	Added 20210126
12	ALTERATION	11	TEST HOLE	0	0	3	Engineering	51	Monitoring Well	Added 20170911
13	OTHER STATUS	11	TEST HOLE	0	0	3	Engineering	51	Monitoring Well	Added 20210126
14	Monitoring and Test Hole	11	TEST HOLE	0	0	3	Engineering	51	Monitoring Well	Added 20210126
15	Abandoned Monitoring and Test Hole	11	TEST HOLE	0	0	3	Engineering	51	Monitoring Well	Added 20210126
1	WATER SUPPLY	11	TEST HOLE	3	IRRIGATION	10	Water Supply	33	Other Water Supply	Added 20180530
9	DEWATERING	11	TEST HOLE	4	INDUSTRIAL	4	Dewatering	47	Geotech Testhole	
13	OTHER STATUS	11	TEST HOLE	4	INDUSTRIAL	3	Engineering	47	Geotech Testhole	
14	Monitoring and Test Hole	11	TEST HOLE	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
15	Abandoned Monitoring and Test Hole	11	TEST HOLE	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	Added 20170911
12	ALTERATION	11	TEST HOLE	6	COMMERCIAL	3	Engineering	25	Other - Commercial	Added 20210126
1	WATER SUPPLY	11	TEST HOLE	6	MUNICIPAL	10	Water Supply	59	Municipal Exploration	Added 20200731
2	OBSERVATION WELLS	11	TEST HOLE	6	MUNICIPAL	10	Water Supply	59	Municipal Exploration	Added 20170911
5	ABANDONED-SUPPLY	11	TEST HOLE	6	MUNICIPAL	10	Water Supply	59	Municipal Exploration	Added 20200721
10	ABANDONED-OTHER	11	TEST HOLE	6	MUNICIPAL	10	Water Supply	59	Municipal Exploration	
14	Monitoring and Test Hole	11	TEST HOLE	6	MUNICIPAL	3	Engineering	59	Municipal Exploration	Added 20210126
5	ABANDONED-SUPPLY	11	TEST HOLE	9	NOT USED	10	Water Supply	33	Other Water Supply	Added 20210126
6	ABANDONED-QUALITY	11	TEST HOLE	9	NOT USED	10	Water Supply	33	Other Water Supply	Added 20200731
10	ABANDONED-OTHER	11	TEST HOLE	9	NOT USED	3	Engineering	51	Monitoring Well	Added 20170911
14	Monitoring and Test Hole	11	TEST HOLE	9	NOT USED	3	Engineering	51	Monitoring Well	Added 20210126
15	Abandoned Monitoring and Test Hole	11	TEST HOLE	9	NOT USED	3	Engineering	51	Monitoring Well	Added 20170911
10	ABANDONED-OTHER	11	TEST HOLE	10	OTHER	3	Engineering	51	Monitoring Well	Added 20210126
10	ABANDONED-OTHER	11	TEST HOLE	11	TEST HOLE	3	Engineering	51	Monitoring Well	Added 20170911
2	OBSERVATION WELLS	11	TEST HOLE	12	DEWATERING	4	Dewatering	51	Monitoring Well	Added 20180530
9	DEWATERING	11	TEST HOLE	12	DEWATERING	4	Dewatering	27	Other Dewatering	
10	ABANDONED-OTHER	11	TEST HOLE	12	DEWATERING	4	Dewatering	27	Other Dewatering	
11	REPLACEMENT WELL	11	TEST HOLE	12	DEWATERING	4	Dewatering	27	Other Dewatering	Added 20170911
14	Monitoring and Test Hole	11	TEST HOLE	12	DEWATERING	4	Dewatering	51	Monitoring Well	
1	WATER SUPPLY	11	TEST HOLE	13	MONITORING	10	Water Supply	51	Monitoring Well	
8	NOT A WELL	11	TEST HOLE	13	MONITORING	3	Engineering	51	Monitoring Well	
11	REPLACEMENT WELL	11	TEST HOLE	13	MONITORING	3	Engineering	51	Monitoring Well	
14	Monitoring and Test Hole	11	TEST HOLE	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20210126
15	Abandoned Monitoring and Test Hole	11	TEST HOLE	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20210126
12	DEWATERING	12	DEWATERING	0	0	4	Dewatering	27	Other Dewatering	Added 20210126

2	OBSERVATION WELLS	12	DEWATERING	0		3	Engineering	27	Other Dewatering	Added 20210126
3	TEST HOLE	12	DEWATERING	0		3	Engineering	27	Other Dewatering	Added 20210126
9	DEWATERING	12	DEWATERING	0		4	Dewatering	27	Other Dewatering	Added 20210126
10	ABANDONED-OTHER	12	DEWATERING	0		4	Dewatering	27	Other Dewatering	Added 20210126
10	ABANDONED-OTHER	12	DEWATERING	2	LIVESTOCK	4	Dewatering	24	Other Agricultural	
3	TEST HOLE	12	DEWATERING	4	INDUSTRIAL	4	Dewatering	27	Other Dewatering	Added 20180530
14	Monitoring and Test Hole	12	DEWATERING	4	INDUSTRIAL	4	Dewatering	51	Monitoring Well	
1	WATER SUPPLY	12	DEWATERING	5	COMMERCIAL	4	Dewatering	33	Other Water Supply	Added 20200731
9	DEWATERING	12	DEWATERING	5	COMMERCIAL	4	Dewatering	25	Other - Commercial	Added 20210126
9	DEWATERING	12	DEWATERING	10	OTHER	4	Dewatering	27	Other Dewatering	
9	DEWATERING	12	DEWATERING	11	TEST HOLE	4	Dewatering	27	Other Dewatering	
2	OBSERVATION WELLS	12	DEWATERING	13	MONITORING	4	Dewatering	51	Monitoring Well	
3	TEST HOLE	12	DEWATERING	13	MONITORING	4	Dewatering	51	Monitoring Well	Added 20210126
10	ABANDONED-OTHER	12	DEWATERING	13	MONITORING	4	Dewatering	51	Monitoring Well	
14	Monitoring and Test Hole	12	DEWATERING	13	MONITORING	4	Dewatering	51	Monitoring Well	
		13	MONITORING	0		3	Engineering	51	Monitoring Well	Added 20210126
1	WATER SUPPLY	13	MONITORING	0		10	Water Supply	51	Monitoring Well	Added 20210126
2	OBSERVATION WELLS	13	MONITORING	0		3	Engineering	51	Monitoring Well	Added 20210126
3	TEST HOLE	13	MONITORING	0		3	Engineering	51	Monitoring Well	Added 20210126
5	ABANDONED-SUPPLY	13	MONITORING	0		10	Water Supply	51	Monitoring Well	Added 20210126
6	ABANDONED-QUALITY	13	MONITORING	0		10	Water Supply	51	Monitoring Well	Added 20210126
9	DEWATERING	13	MONITORING	0		4	Dewatering	51	Monitoring Well	Added 20210126
10	ABANDONED-OTHER	13	MONITORING	0		3	Engineering	51	Monitoring Well	Added 20210126
11	REPLACEMENT WELL	13	MONITORING	0		3	Engineering	51	Monitoring Well	Added 20210126
12	ALTERATION	13	MONITORING	0		3	Engineering	51	Monitoring Well	Added 20210126
13	OTHER STATUS	13	MONITORING	0		3	Engineering	51	Monitoring Well	Added 20210126
14	Monitoring and Test Hole	13	MONITORING	0		3	Engineering	51	Monitoring Well	Added 20210126
15	Abandoned Monitoring and Test Hole	13	MONITORING	0		3	Engineering	51	Monitoring Well	Added 20210126
1	WATER SUPPLY	13	MONITORING	4	INDUSTRIAL	10	Water Supply	33	Other Water Supply	
4	RECHARGE WELL	13	MONITORING	4	INDUSTRIAL	3	Engineering	83	Recharge Well	
11	REPLACEMENT WELL	13	MONITORING	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	Added 20170911
12	ALTERATION	13	MONITORING	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
14	Monitoring and Test Hole	13	MONITORING	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
15	Abandoned Monitoring and Test Hole	13	MONITORING	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
10	ABANDONED-OTHER	13	MONITORING	9	NOT USED	3	Engineering	51	Monitoring Well	Added 20180530
2	OBSERVATION WELLS	13	MONITORING	9	NOT USED	3	Engineering	51	Monitoring Well	Added 20180530
10	ABANDONED-OTHER	13	MONITORING	10	OTHER	3	Engineering	51	Monitoring Well	Added 20200731
10	ABANDONED-OTHER	13	MONITORING	10	OTHER	3	Engineering	51	Monitoring Well	Added 20200731
13	OTHER STATUS	13	MONITORING	10	OTHER	3	Engineering	51	Monitoring Well	
13	OTHER STATUS	13	MONITORING	11	TEST HOLE	3	Engineering	51	Monitoring Well	Added 20190509
14	Monitoring and Test Hole	13	MONITORING	11	TEST HOLE	3	Engineering	51	Monitoring Well	
15	Abandoned Monitoring and Test Hole	13	MONITORING	11	TEST HOLE	3	Engineering	51	Monitoring Well	Added 20210126
9	DEWATERING	13	MONITORING	12	DEWATERING	4	Dewatering	51	Monitoring Well	Added 20180530
14	Monitoring and Test Hole	13	MONITORING	12	DEWATERING	4	Dewatering	51	Monitoring Well	
2	OBSERVATION WELLS	13	MONITORING	13	MONITORING	3	Engineering	51	Monitoring Well	
3	TEST HOLE	13	MONITORING	13	MONITORING	3	Engineering	51	Monitoring Well	
14	Monitoring and Test Hole	13	MONITORING	13	MONITORING	3	Engineering	51	Monitoring Well	Added 20170911
		14	Monitoring and Test Hole	0		3	Engineering	51	Monitoring Well	Added 20210126
1	WATER SUPPLY	14	Monitoring and Test Hole	0		10	Water Supply	51	Monitoring Well	Added 20210126
2	OBSERVATION WELLS	14	Monitoring and Test Hole	0		3	Engineering	51	Monitoring Well	Added 20210126
3	TEST HOLE	14	Monitoring and Test Hole	0		3	Engineering	51	Monitoring Well	Added 20210126
5	ABANDONED-SUPPLY	14	Monitoring and Test Hole	0		10	Water Supply	51	Monitoring Well	Added 20210126
6	ABANDONED-QUALITY	14	Monitoring and Test Hole	0		10	Water Supply	51	Monitoring Well	Added 20210126
9	DEWATERING	14	Monitoring and Test Hole	0		4	Dewatering	51	Monitoring Well	Added 20210126
10	ABANDONED-OTHER	14	Monitoring and Test Hole	0		3	Engineering	51	Monitoring Well	Added 20210126
13	OTHER STATUS	14	Monitoring and Test Hole	0		3	Engineering	51	Monitoring Well	Added 20210126
14	Monitoring and Test Hole	14	Monitoring and Test Hole	0		3	Engineering	51	Monitoring Well	Added 20210126
15	Abandoned Monitoring and Test Hole	14	Monitoring and Test Hole	0		3	Engineering	51	Monitoring Well	Added 20210126
14	Monitoring and Test Hole	14	Monitoring and Test Hole	3	IRRIGATION	1	Agricultural	51	Monitoring Well	Added 20210126
1	WATER SUPPLY	14	Monitoring and Test Hole	4	INDUSTRIAL	3	Engineering	33	Other Water Supply	
2	OBSERVATION WELLS	14	Monitoring and Test Hole	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
3	TEST HOLE	14	Monitoring and Test Hole	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
4	RECHARGE WELL	14	Monitoring and Test Hole	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
5	ABANDONED-SUPPLY	14	Monitoring and Test Hole	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
6	ABANDONED-QUALITY	14	Monitoring and Test Hole	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
9	DEWATERING	14	Monitoring and Test Hole	4	INDUSTRIAL	4	Dewatering	51	Monitoring Well	Added 20170911
10	ABANDONED-OTHER	14	Monitoring and Test Hole	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
13	OTHER STATUS	14	Monitoring and Test Hole	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
14	Monitoring and Test Hole	14	Monitoring and Test Hole	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
14	Monitoring and Test Hole	14	Monitoring and Test Hole	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
15	Abandoned Monitoring and Test Hole	14	Monitoring and Test Hole	4	INDUSTRIAL	3	Engineering	51	Monitoring Well	
14	Monitoring and Test Hole	14	Monitoring and Test Hole	10	OTHER	3	Engineering	51	Monitoring Well	Added 20210126
14	Monitoring and Test Hole	14	Monitoring and Test Hole	12	DEWATERING	4	Dewatering	51	Monitoring Well	Added 20210126
10	ABANDONED-OTHER	14	Monitoring and Test Hole	12	DEWATERING	4	Dewatering	51	Monitoring Well	Added 20210126

Appendix H - Current Problems (To Be Corrected) or Suggestions

20121022

Duplicate reports and report records found in the Report Library (directories) as well as D_LOCATION and D_DOCUMENT. In some (other) cases, duplicate reports have been deleted from the RL but not removed from D_DOCUMENT/D_LOCATION. Some files have been loaded into incorrect DOC_FOLDER_ID directories.

20121030

Some Environment Canada (precipitation only) data has been loaded twice. RD_NAME_OUOM's are: Precipitation - Day; Day Precip. Otherwise all other fields are duplicated. Refer to Section 3.2.2 ('Total Monthly Precipitation, Average Temperature and Water Level Data') where this problem was first noted. This problem is down to the repetition of, for example, 'Day Precip' and 'Precipitation - Day' being converted to the same RD_NAME_CODE - the values are then duplicated. The same occurs for the equivalent fields of 'Day Snow' and 'Day Rain'.

Borehole loggers, recording groundwater temperature data, have been incorrectly tagged with RD_NAME_CODE '369' ('Temperature (Air)'). These should instead be tagged with code '70871' ('Temperature (Water) - Logger'). Refer to Section 3.2.2 (as above) where this problem was first noted. There is also actual 'air' temperatures tagged against a well/borehole interval.

20121101

Some values in D_INTERVAL_TEMPORAL_2 have been tagged with a 'no value' key of '-999'. Should these records be removed entirely or converted to NULL.

20121203

Some borehole locations have had (at some previous time - early 2011?) a second 'false' interval (1ft at bottom of hole) added erroneously.

20130218

D_LOCATION_AGENCY currently stores locations with the tag 'LTCRA'; this should instead be 'LTRCA'. Also, the area applied to locations with this tag should, more appropriately, have the tag 'TRENTSWP' with the 'LTRCA' tag applied to the smaller area.

20130322

The application of GEOL_SUBCLASS_CODE needs to be reviewed. At present, most information in D_GEOLOGY_LAYER has a NULL value applied. Review changes in R_GEOL_SUBCLASS_CODE (i.e. incorporation of 'Invalid' and 'Alternate' entries).

20130408

A review of how useful many of the views are in relation to their long-run times. Should these be replaced with 'summary' tables containing (monthly?) updateable values for easy reference?

20130419a

Regarding the PICKS table – currently, active picking is working (after modification) against the PICKS2 table. The following changes need to be implemented against PICKS including:

Task 1

Drop the field named by rowguid (note that this is not possible for a replicated table – evaluate in the next database version)

This field appears to prevent table updates. (SQL Uniqueidentifier field types are also usually not recommended.)

Dynamically adding a unique identifier is handled in changes listed below.

Task 2

Set the Primary Key for fields Loc_id and Formation

This is used to ensure that you cannot make two picks with the same loc_id and formation in one borehole.

Task 3

Add default constraints for UNIQUEID and PICKDATE

UNIQUEID can be used instead of rowguid for uniqueness identification. The default Pickdate is set to the current date.

Task 4

Create a Nonclustered Index for Loc_id (for delete performance)

This is needed to improve the delete performance

Notes and SQL statements for this process are to be found in the directory 'v:\db\sql_queries\20130415_picks' on MDM6500. These have been applied to the PICKS2 table.

In addition, the original GND_ELEV trigger and the user name ('suser_sname()') function call have been reapplied without (much) change in the picking process (i.e. with regard to the time required for creating/deleting pick records).

20130419b

With regard to SiteFX – the replication should not examine changes to the SYS_TEMP1 or SYS_TEMP2 fields as SiteFX uses this to temporarily 'tag' individual rows without

otherwise changing the actual data. This tends to apply to entire tables at any one time. The replication process views this as an actual change for a row and transfers all the information for that row (and, thus, the entire table) between the publisher and subscribers. It has been suggested to add a SYS_TEMP3 field which can be used to tag rows by the YPDT-CAMC group and partner agencies but is not otherwise recognized by SiteFX.

In addition, the SYS_LAST_MODIFIED and SYS_LAST_MODIFIED_BY fields should be adjusted in the same manner, to look only for changes in data not modification of the temporary fields.

20130807

Look into locking the UTM coordinates for the partners (i.e. disabling the ability to modify the coordinates of a location at the partner agency end; this is to prevent unintentional modification when examining locations in Viewlog).

20130807

The table R_GEOL_LAYERTYPE_CODE, originally found in the Access database, needs to be added back into the Master YPDT-CAMC database.

20130927

Geology top- and bottom-depths have been found to cross; this appears to be from the MOE imports. Some areas have been fixed (e.g. Peel and South Simcoe/OGS Wells) but should be consistently looked at (see for example Section G.5 in the Appendix).

20130927b

In some cases, the bottom depth of the well has not been included. If other (valid) information is present (e.g. material codes), it is assumed that the driller stopped at an obstruction (of some sort) with no-to-little advance in depth, recording only the top depth. A false value of 1ft (or 0.3m) is to be added in these cases.

20130927c

Mixing of cm's and m's has occurred. This needs to be checked as borehole depth values are invalid when this occurs.

20130927d

The MOE has changed its original-units-of-measure between releases for some boreholes. That is, previous wells listed as m's depth are now considered to be in ft. A global check for this should be (as used in previous cases): if the values are integer only, consider them ft; otherwise they are m's.

20131016

The D_INTERVAL_TEMPORAL_1B table should, in the next version of the database, have a primary key setup for both the SAM_ID and the RD_NAME_CODE so as to avoid duplicate information for any particular sample. Note that this may be difficult as RD_NAME_CODE is created subsequent to the import (the RD_NAME_OUOM field is

the one being populated, leading to possible errors as multiple text names can be used for any one parameter; see R_READING_NAME_ALIAS table for examples).

20131023

Should another QA_COORD_CONFIDENCE_CODE be made available to indicated those locations whose coordinates are valid but which lie outside of the YPDT-CAMC buffered area and within Ontario. This would be applied as a substitute for the QA code of '117' (which was originally meant to tag wells with invalid coordinates).

20131206a

All fields with varchar(max) should be converted to varchar(255). This includes

D_GEOLOGY_LAYER

Others

20131206b

Update D_LOCATION_ELEV, adding a ELEV_COMMENT field.

20131209

SYS_USER_STAMP in D_INTERVAL_TEMPORAL_1B is not automatically being populated. Correct this and check that this occurs in other tables.

20131211

D_CLIMATE should contain no data. There currently is nothing contained within it that is not found in other tables. The table likely needs to remain as SiteFX requires it. The same can likely be said of D_SURFACEWATER, though the drainage area calculations are stored here (can they be moved elsewhere?).

20140115

Currently, the INT_NUMBER column is unused in D_INTERVAL_MONITOR. Should this be populated to avoid the use of on-the-fly calculations of the total number of screens and the screen number for any particular interval? (See also V_YPDT_SYS_SCREEN_NUM and V_YPDT_SYS_SCREEN_NUM_TOTAL in the OAK_20120615_UPDATE database.)

20140115b

SYS_USER_STAMP also (see 20131209) does not have a default value. To be corrected.

20140116

When setting replication, specify that all documents are to be replicated (not limited by being found within a partner agency boundary).

20140507

In the D_INTERVAL_MONITOR table, obvious changes in characteristics (e.g. change in diameter) should necessitate a separate INT_ID. That is, an examination of existing

INT_IDs should be carried out where multiple elevations are specified in the monitor table. These should be broken into multiple INT_IDs where possible.

20140604

D_LOCATION_PURPOSE should not have a SYS_RECORD_ID field as the primary key. Note that an exception to this (which would result in keeping the PK as is) is if we're recording 'all' purposes applied to that location (and then, these should be dated).

20140723a

D_INTERVAL_MONITOR – should this be changed to a 'bit' type from 'real'? At the moment, ranges of values include NULL, '-1' through '2'. The '-1' values will be changed to '1' and '0' will be changed to NULL. What is '2' indicating?

20140723b

D_PUMPTEST_STEP – all PUMP_RATE values should be standardized to a single unit (IGPM?).

20140723c

D_INTERVAL_TEMPORAL_2 – Specific capacity only has a single value for all intervals in this table. Should we locate this elsewhere as a single column (maybe D_INTERVAL)?

20140728

Some pumptest dates and corresponding water level dates do not match – there seems to have been a misapplication of MDY versus DMY import schemes.

20140728b

Should all 'unspecified' values in D_DOCUMENT be changed to strictly NULL values (the latter is easier to pick out when reviewing returned results)?

20140728c

Add a view that only pulls LOC_IDs that were originally from the MOE database(s).

20140815

Should DOC_ID in D_LOCATION_PROPERTY be a foreign key to D_DOCUMENT.

20140815b

Indices (on tables) should be renamed such that the indexed field is part of the index name (partially corrected – currently a mixture of field names or the table name for which the field is a foreign key).

20140818

Review D_LOCATION_VULNERABILITY – update the names of these fields. Do we need to repopulate/add information or locations? Should this table be removed?

20140818b

Should all tables with LOC_ID (for example; applies to any _ID field) have their index changed to 'UNIQUE' when the identifier is a foreign key and only a single row should appear for each _ID field. (An example of this is found in D_SURFACEWATER.)

20140819

The watershed fields in D_LOCATION do not reference the watershed tables – should this be corrected.

20140819b

In R_LOGGER_TYPE_READING, should the RD_NAME_CODE refer to the R_RD_NAME_CODE table.

20140820

What is S_DATA_SEARCH_INTERVALS used for? Not added by SiteFX (in the latest edition) and seemingly not modified since 2004.

20140820b

What is S_LOGFILE used for? Not added by SiteFX (in the latest edition) and seemingly not modified since 2010. This applies to (with the exception of the dates) each of: S_RPTSETTINGS; S_RPTSETTINGSPARA; S_RPTSETTINGSWELL.

20140826

Are R_EQ_GROUP/TYPE_CODE any use? What is 'Profile' used for in R_GEOL_MAT1_CODE?

20140826b

Should S_CHANGE_HISTORY be copied between database versions?

20140827

From D_BOREHOLE:

Should MOE_BH_GEOLOGY_CLASS be dropped?

What is BH_KB_ELEV, BH_KB_ELEV_OUOM and BH_KB_ELEV_UNIT_OUOM for (currently blank)?

Should D_DATABASE_NOTE be copied (one entry – Mezmure)?

What is INT_MORE_1_PART for in D_INTERVAL (currently empty)?

Add indexes to D_INTERVAL_FORMATION_ASSIGNMENT. Do we need add additional models to this table?

D_INTERVAL_MONITOR – has a MON_ID and SYS_RECORD_ID; which should we use as the primary key (currently the latter)?

D_INTERVAL_REF_ELEV – REF_POINT should be adjusted (after the conversion to the new database) to reflect information as expected by SiteFX.

For D_LOCATION_QA:

Should we remove QA_PUMPING_CODE

Should we remove QA_WL_STATIC.

20140908

Location ID's are not being removed from the D_LOCATION_AGENCY table when, likely, they are being removed from D_LOCATION.

20141008

There is an issue with partners importing data into their versions of the database and synchronizing against the master.

Even though they are restricted (hopefully) to a designated range of values, they are working with a subset of the database; rows that are present in the master are not available in their version. As such, their import software can create identifiers that will conflict with identifiers in the master during synchronization. This is currently happening with the NVCA data.

20141015

In the new version of the database, the DATA_ID field (found in most tables) should be NULL by default instead of populating with a random number.

20141124

When populating the SiteFX Access version of the database (i.e. using a blank database created by SiteFX and inserting into the empty tables) at times (the latest example was against D_INTERVAL_TEMPORAL_1B) a row cannot be inserted. Instead it returns the error:

[Microsoft][ODBC Microsoft Access Driver] Error evaluating CHECK constraint. (SQL-HY000) at ...

When the table is cleared and the process restarted, the population completes without further error. Upon investigation, no data-related reason could be seen for this problem to occur. Note that this has happened multiple times before but was taken as a data issue.

20141201

Users are inputting 'Temperature' records without further selecting the appropriate temperature reading name based upon the interval type (e.g. climate stations should be meteorological, screens should be logger related and spot flows should be field related). Care should be taken when inputting these values.

In addition, parameters have been assigned to invalid temporal tables (e.g. herbicides to DIT2; water levels to DIT1B). This is likely due to the default SiteFX import function.

20150811

OWN_NAME in D_OWNER should be changed to type varchar(255).
OWN_COMMENT should also be changed to varchar(255).

Appendix I – Training Setup

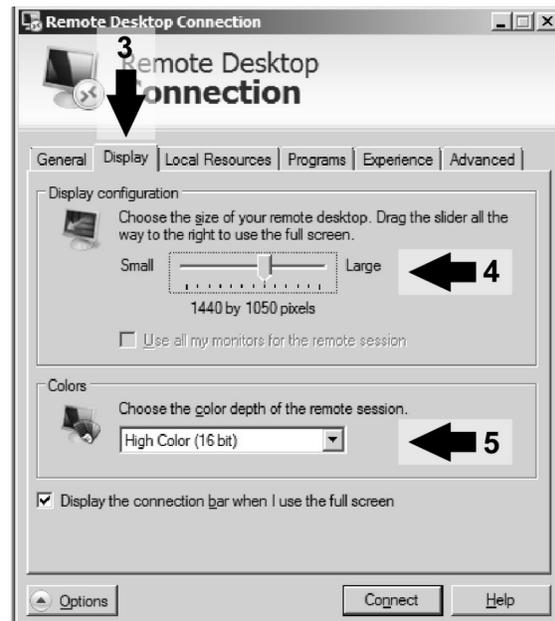
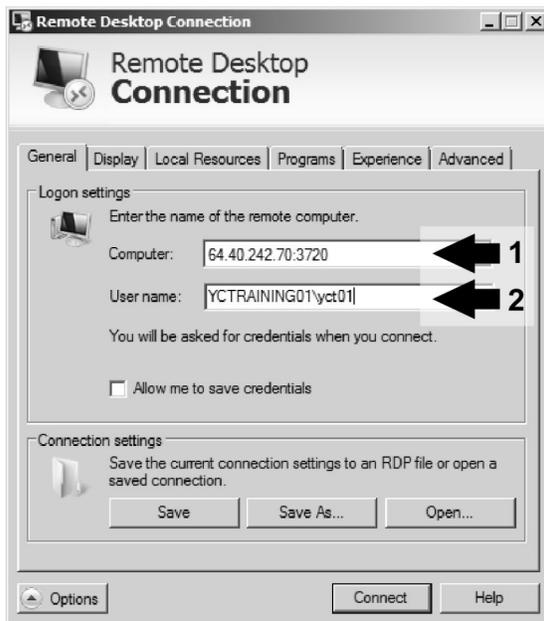
Overview

‘Training’ computers are made available during ORMGP training sessions but are otherwise not accessible outside of the Downsview office. These are a series of ‘Virtual Machines’ (VM) running off a dedicated server – each VM having the software necessary for the varied training exercises available. These include: Microsoft Office 2010; EarthFX SiteFX; EarthFX Viewlog (demo mode only); and Microsoft SQL Management Studio. No databases are installed; instead, connections are made to those databases residing on the MAINPARTNER SQL Server (limited permissions are granted for these connections).

Connection

All connections to these VM are made through the ‘Remote Desktop’ software found on all computers running later versions of Microsoft Windows (i.e. subsequent to Windows 2000). This software can be accessed through ‘Start – Accessories – Remote Desktop Connection’ (or, alternately, through ‘Start – All Programs – Accessories – Remote Desktop Connection’). (Note that shortcuts/links to this software may be available to users on their local computers. Variations in access to this software through the task bar are possible.)

Once the software is started, users can configure the settings necessary for the connection. The initial setup is found under ‘Options – General’.



Here the user should configure the IP address (i.e. the ‘web’ address; under ‘Computer’, indicated by ‘Point 1’) of the VM – all addresses for these VM’s will be similar, for example

64.40.242.70:3720

for the first training VM and

64.40.242.70:3731

for the final/last training VM. Note that the only changes between these addresses occur in the final 4-digit number (after the colon; i.e. 3720-3731). The login name used for each VM also varies. These, corresponding to the previously described ‘addresses’ would then be

YCTRaining01\yct01

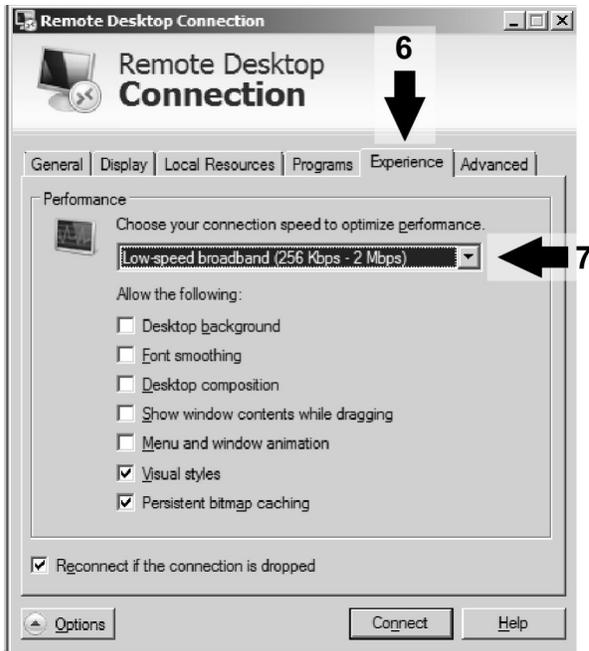
and

YCTRaining10\yct12

(as indicated by ‘Point 2’ in the figure, above). A complete listing of VM’s, their addresses and login characteristics are provided, following - below, in this section of the document. Note that the complete name is required (e.g. ‘YCTRaining01\yct01’; where ‘01’ varies depending upon the VM the user is accessing) for logging onto these machines.

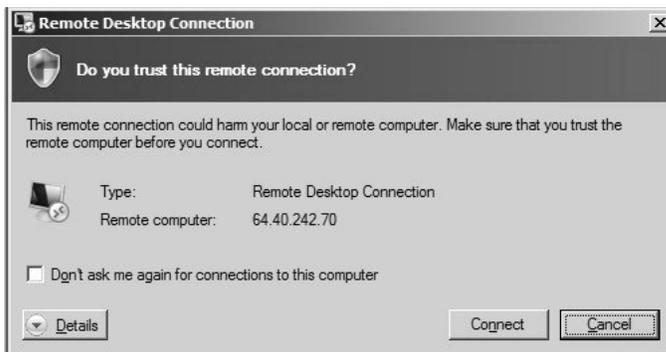
In order to minimize the amount of information being transferred from the Downsvew office, the ‘Display’ options of the ‘Remote Desktop’ should be adjusted. These are found under the ‘Display’ tab (see ‘Point 3’). The ‘Display Configuration’ (see ‘Point 4’) should be set to a maximum of ‘1440 x 1050’ pixels and with a ‘16bit’ colour scheme (see ‘Point 5’). This should provide a level-of-detail high enough for most purposes.

In addition, the connection speed should be adjusted, as shown

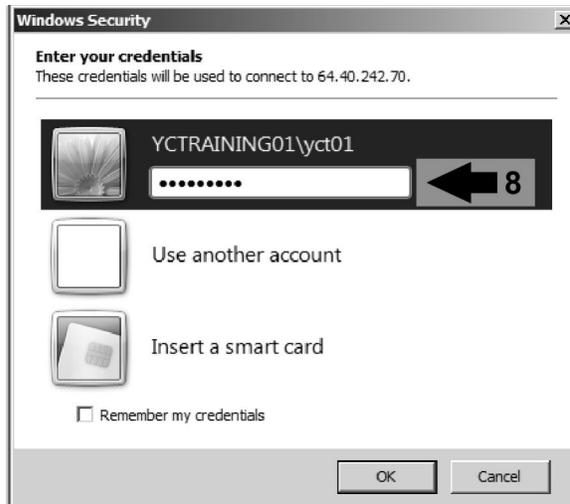


This reduces the speed to that of ‘Low speed broadband ...’, to (again) minimize the exchange of information across the connection. (Note that it may, depending upon the number of users connecting at one time, be necessary to drop the speed further to ‘Modem ...’.)

Selecting ‘Connect’ will connect the user to the remote VM (if enabled). The user may be prompted with a ‘Do you trust this remote connection’ dialog box, select ‘Connect’ again.



The user will then be prompted for the password for the specified login-name.



Enter the provided password (see 'Point 8'). The user may be prompted with a 'The identity of the remote computer ...' warning.



Connect to the VM despite the certificate error by selecting 'Yes'. After connecting, a standard Microsoft Windows 7 screen will be displayed. The user can then interact with the remote VM as they would a regular computer.

Virtual Machine Accounts

The following accounts are available for users accessing the training VM's (note that passwords are not provided here; these will be available during any training session itself).

Computer Name	Computer Address	User Account	Downsview Address
YCTRaining01	64.40.242.70:3720	yct01	192.168.2.30:3389
YCTRaining02	64.40.242.70:3721	yct02	192.168.2.31:3389
YCTRaining03	64.40.242.70:3722	yct03	192.168.2.32:3389
YCTRaining04	64.40.242.70:3723	yct04	192.168.2.33:3389
YCTRaining05	64.40.242.70:3724	yct05	192.168.2.34:3389
YCTRaining06	64.40.242.70:3725	yct06	192.168.2.35:3389
YCTRaining07	64.40.242.70:3726	yct07	192.168.2.36:3389
YCTRaining08	64.40.242.70:3727	yct08	192.168.2.37:3389
YCTRaining09	64.40.242.70:3728	yct09	192.168.2.38:3389
YCTRaining10	64.40.242.70:3729	yct10	192.168.2.39:3389
YCTRaining11	64.40.242.70:3730	yct11	192.168.2.40:3389
YCTRaining12	64.40.242.70:3731	yct12	192.168.2.31:3389

Each account allows access to any of the partner databases located on the SQL Server at MAIN\PARTNER. All logins use the combination of

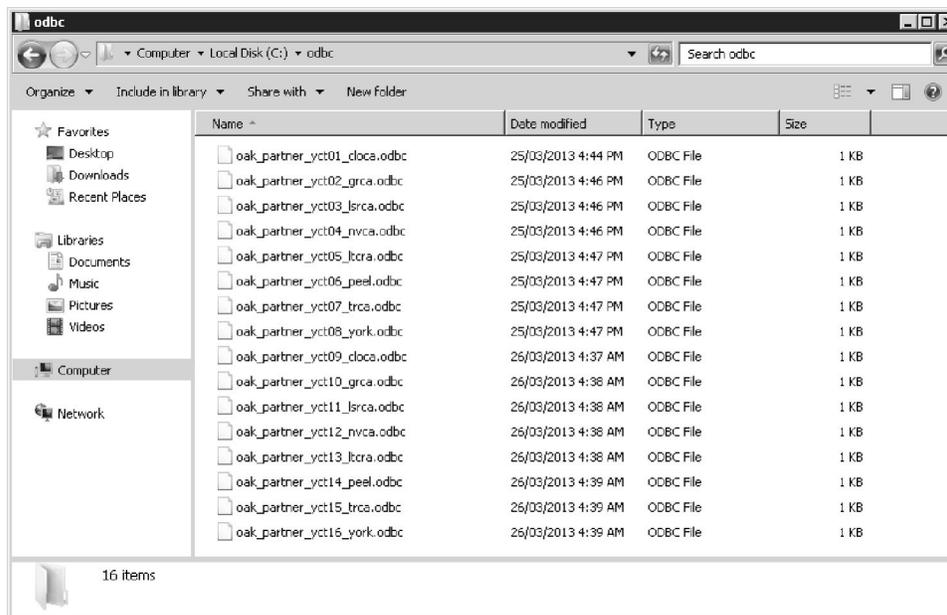
<computer name>\<user account>

For example, using the first VM, the login would be

YCTRaining01\yct01

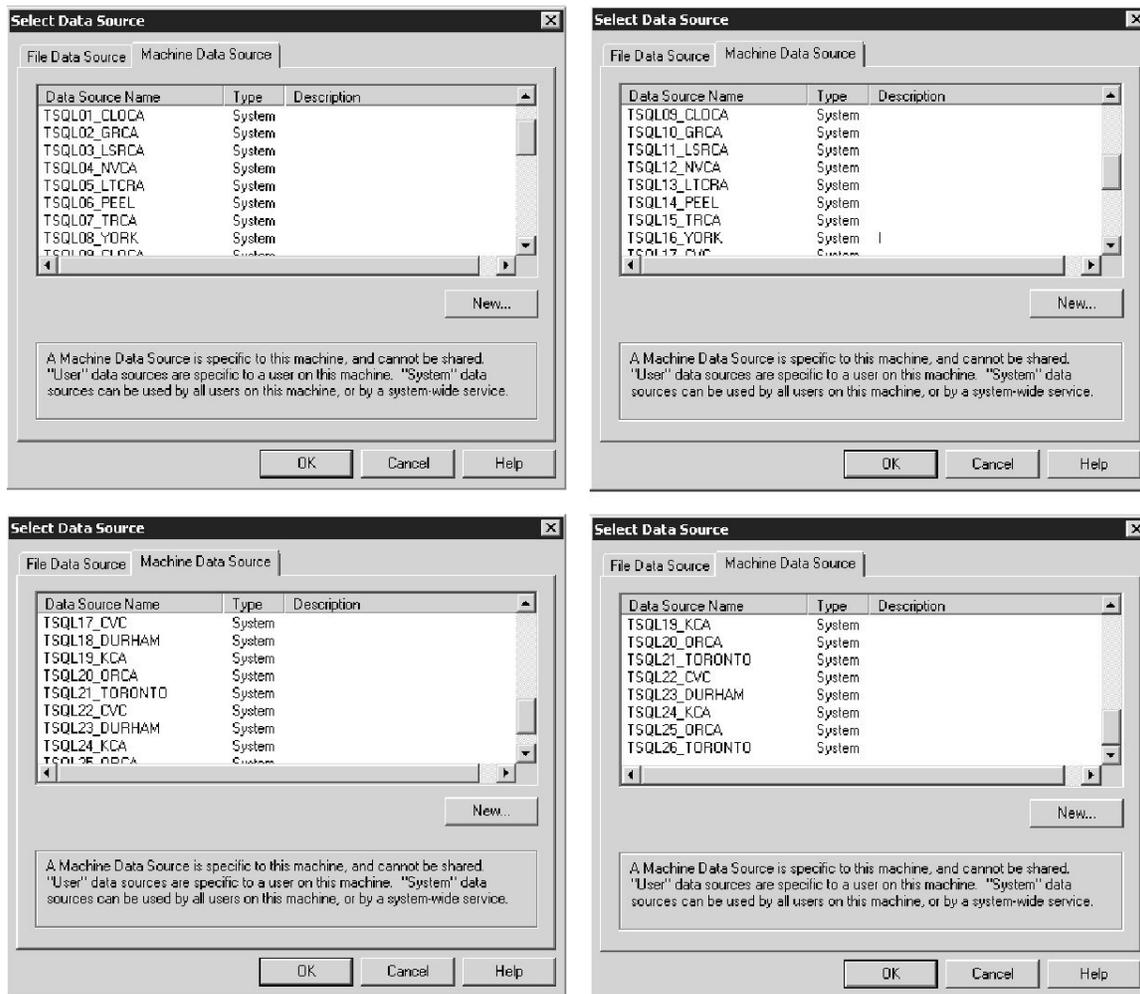
Database Connections

Two sets of connections (other than the direct-native connection found in Microsoft SQL Management Studio) are available for accessing a partner database. An ODBC file (i.e. ‘*.odbc’) is available under the ‘C:\ODBC’ directory on the VM.



These are used by EarthFX SiteFX and Viewlog for accessing a specific database. Two ODBC connections (with associated login and password information) are created for each partner database (examine the name of each file – they indicate both the login name as well as the database being accessed). Note that these are guest accounts only.

In addition, for use within Microsoft Access (for the linking of tables), ‘Machine Data Source’ ODBC connections are available.



Again, two accounts per partner database. Note that these contain the same account information as found in the ‘*.odbc’ files, above. (In addition, found under the ‘C:\DSN’ directory, are the equivalent ‘File DSN’ ODBC data sources to the above listing.) A summary of the login account information (for Microsoft Access and EarthFX’s SiteFX and Viewlog) is provided, below.

Primary Account	Secondary Account	Partner Database
tsql01	tsql09	CLOCA
tsql02	tsql10	GRCA
tsql03	tsql11	LSRCA
tsql04	tsql12	NVCA
tsql05	tsql13	LTCRA
tsql06	tsql14	PEEL
tsql07	tsql15	TRCA
tsql08	tsql16	YORK
tsql17	tsql22	CVC
tsql18	tsql23	DURHAM
tsql19	tsql24	KCA
tsql20	tsql25	ORCA
tsql21	tsql26	TORONTO

Note that all the accounts could be given the general permissions required to access any of the partner agencies databases (if, for example, a training session is being held exclusively at a single partner agency) but any access through the ‘Machine Data Source’ is limited to the configured connection (i.e. Microsoft SQL Management Studio would be able to access any database using any of these accounts). For (more) advanced users, this means that each of these ‘partner’ accounts have had a ‘User Mapping’ specified (i.e. ‘have been mapped to’) for each partner database. For connection properties, the following memberships have been enabled (for these accounts): ‘db_datareader’; ‘db_denydatawriter’; ‘public’ (this is accessible through ‘Security – Logins - <login name> - <right-click> - Properties – User Mapping’).

Training Database (OAK_20120615_TRAINING)

This is a database setup explicitly for training sessions, in particular when users are working within SiteFX (which requires read/write access; this is not available for the partner agency databases in a training session). The database itself is a subset of OAK_20120615_MASTER, incorporating locations within a specified buffer (10km) of the Oak Ridges Moraine and with a QA_COORD_CONFIDENCE_CODE less than or equal to ‘8’ – this reduces the database size and overhead required by the SQL Server instance. There is no connection between this training database and either the partner or master databases – any changes made here are abandoned after every training session. All ODBC files (for SiteFX) are found in the directory ‘C:\ODBC\OAK_20120615_TRAINING’. In addition, ODBC DSN files are available in the directory ‘C:\ODBC\OAK_20120615_TRAINING.DSN’.

All SQL Server accounts (i.e. tsql01 through tsql26) have ‘db_owner’ access to this database.

Troubleshooting

Valid user cannot login using ‘Remote Desktop’

Make sure that the user has been added to the list of users who have permission to connect to the virtual machine using Remote Desktop. Under 'Windows 7', this is accomplished through 'Computers - <right-click> - Properties – Remote Settings – Select Users'. Note that this only needs to be applied to 'training' users, 'Administrators' already have this permission enabled.

No ODBC links available under ODBC Data Source Administrator

Make sure that the links were created using the 'odbcad32' executable found under 'C:\Windows\SysWOW64' when using the 32-bit version of Microsoft Access. The 64-bit version is found under 'C:\Windows\System32'. All VMs are currently running the 32-bit version.

Use the DSN files (under 'C:\DSN'), instead, to temporarily bypass this problem.

Training Computers – VM Creation and Configuration Checklist

The following instructions are only for setting up the virtual machines (VM) on the Downsvie server by an administrator. Each virtual machine is a copy of the 'main' setup (actually the first training machine – 'Windows7-64bit-Training-01') modified to match a training machine number.

1. In 'vSphere Client', select the 'vSphere Server' (this may be an IPv4 number) and then the 'Configuration Tab'. Open/browse each of the 'Datastore_Original' (use/create the directory 'Training_Machines_02-12') and 'Datastore1' (open the 'Windows7-64bit-Training-01' directory) datastores. Make sure that the 'Windows7-64bit-Training-01' VM is shut down.
2. Make a new directory in 'Training_Machines_02-12', naming it 'Windows7-64bit-Training-02'.
3. Select within 'Windows7-64bit-Training-01' the files 'Windows764bitSp1_Master.vmx' and 'Windows764bitSp1_Master.vmdk'; right-click on one of these and select 'Copy'.
4. In the directory 'Windows7-64bit-Training-02', right-click the empty window and select 'Paste'. This procedure copies the VM files from the 'master' directory to a 'secondary' directory where the 'new' VM will be found.
5. After copying, right-click on the '.vmx' file and select 'Add to Inventory'. Use the name 'Windows7-64bit-Training-02'. This creates the new VM training machine. Turn the VM on.
6. Repeat steps 2 through 5 for each training machine '03' through '12'.
7. Login (through the 'Console') to 'Windows7-64bit-Training-02' as 'Administrator'.
8. Select 'Control Panel – Network and Internet – Network Connections – Local Area Connection - <right-click> - Properties – Internet Protocol Version 4 ... - Properties' and change the 'IP Address' to '192.168.2.31' (note that these addresses can be found in the appropriate table, above; the address can be checked using the 'ipconfig' command on the 'cmdline').

9. Select 'Start – Computer - <right-click> - Properties – Advanced system settings – Computer Name – Change – Computer name' and change the computer name to 'YCTRaining02'. This will necessitate the restart of the VM – do so.
10. Login to the VM using the 'yct02' user and right-click on the desktop and select 'Personalize'. Change the desktop to 'Windows Classic'. Right-click on the taskbar and de-select 'Lock the taskbar' and select 'Properties – Use small icons' and 'Auto-hide the taskbar'.
11. Make sure that the particular user (e.g. 'yct02') has 'Remote Login' privileges. This is found under 'Computer - <right-click> - Properties – Remote Settings – Remote – Select Users'.
12. Repeat steps 7 through 11 for each training machine '03' through '12' (using the appropriate names and IP addresses).

Note that, all virtual machines should have their 'Windows Update' function set to NOT automatically update; only the first training machine should be updated (and then copies made). In addition, only update the anti-virus information on the first training machine.

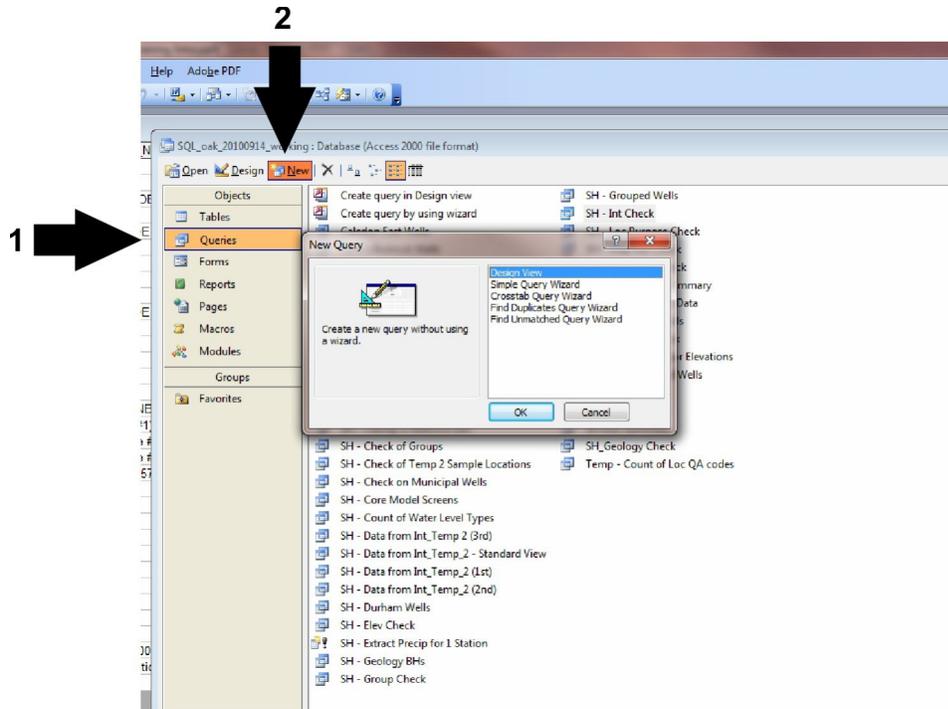
Appendix J Training Exercises

Section J.1 Training Exercises (Easy)

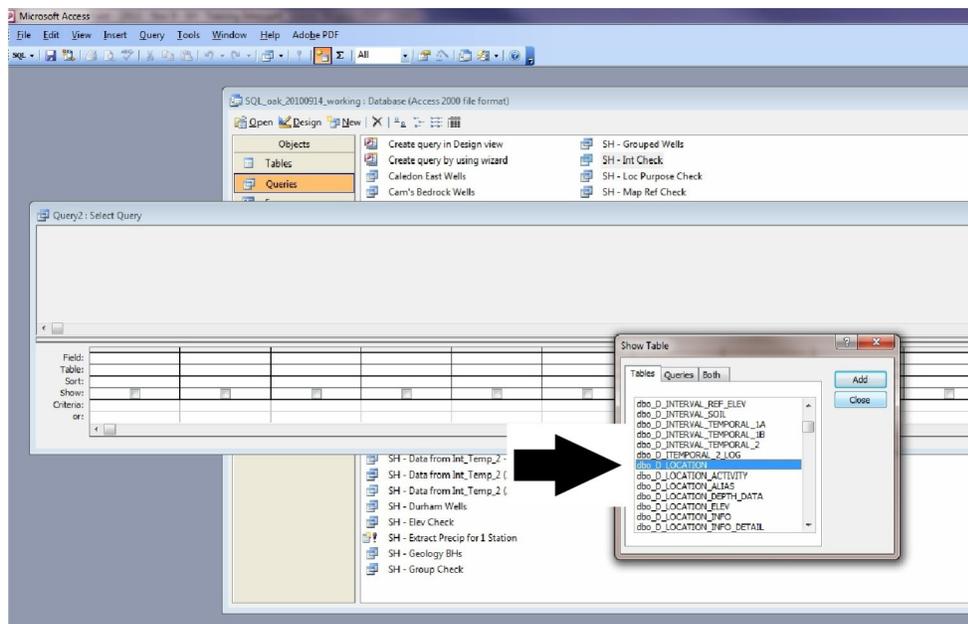
Example 1a - Wells Associated with Nobleton (Microsoft Access 2003)

As a first example, we will look for all wells that have been associated with Nobleton.

1. Select the 'Queries' option under 'Objects' and then select 'New'



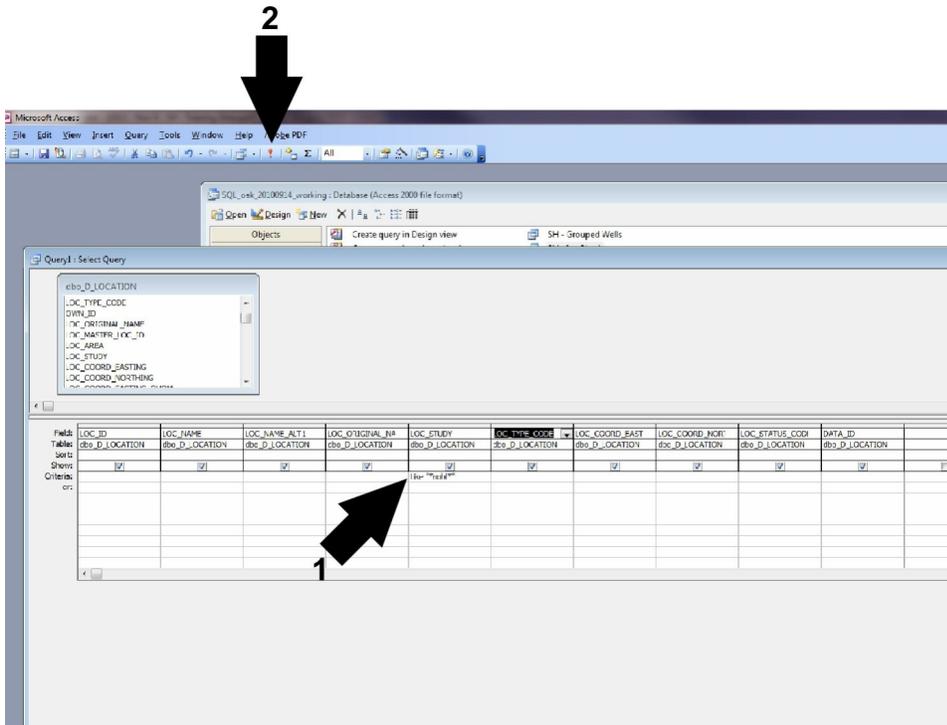
2. 'Design View' is automatically selected (in the upper left corner). Add the 'D_LOCATION' table by double-clicking in the 'Show Table' dialog box (or single-clicking and selecting 'Add')



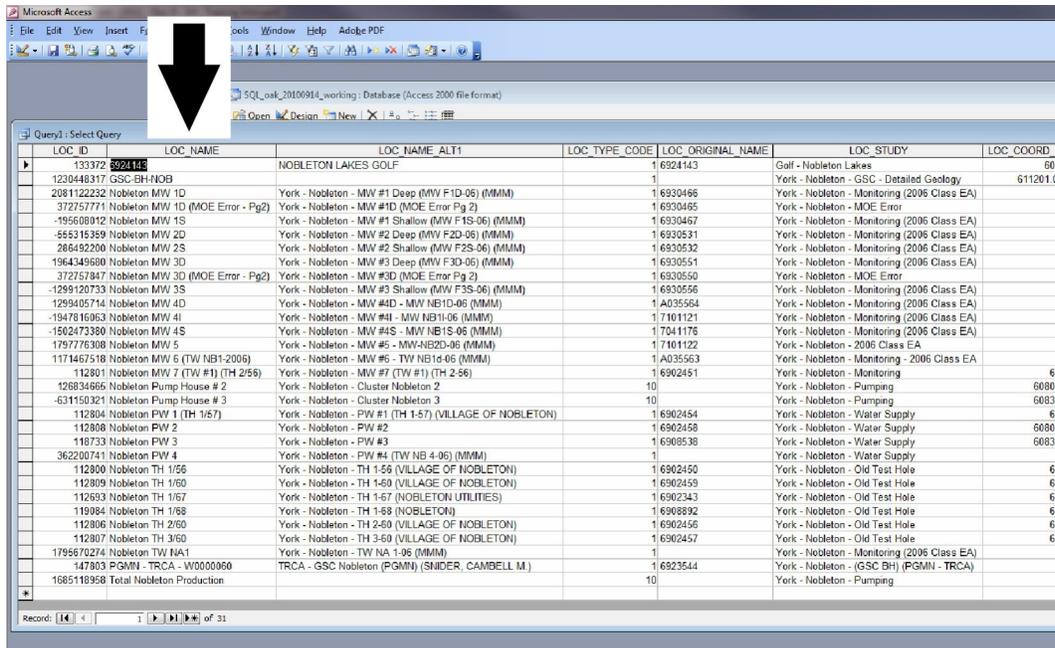
3. Add each of the following fields/columns into they query by double-clicking on them (or clicking-dragging) so they appear in the query box:

LOC_ID	LOC_STUDY
LOC_NAME	LOC_COORD_EASTING
LOC_NAME_ALT1	LOC_COORD_NORTHING
LOC_TYPE_CODE	LOC_STATUS_CODE
LOC_ORIGINAL_NAME	DATA_ID

4. Filter the query under the LOC_STUDY field/column by entering 'like "*Noble*"' under 'Criteria'. Select the red exclamation mark (the 'Run' button) to run the query



5. A number of records should be returned. They can be sorted by selecting 'LOC_NAME - <right-click> - Sort Ascending' (note that some of the field/column names have not been shown)



Note the following:

- The Nobleton Golf Course well also appears in the query
- MW indicates monitoring wells
- PW indicates pumping wells

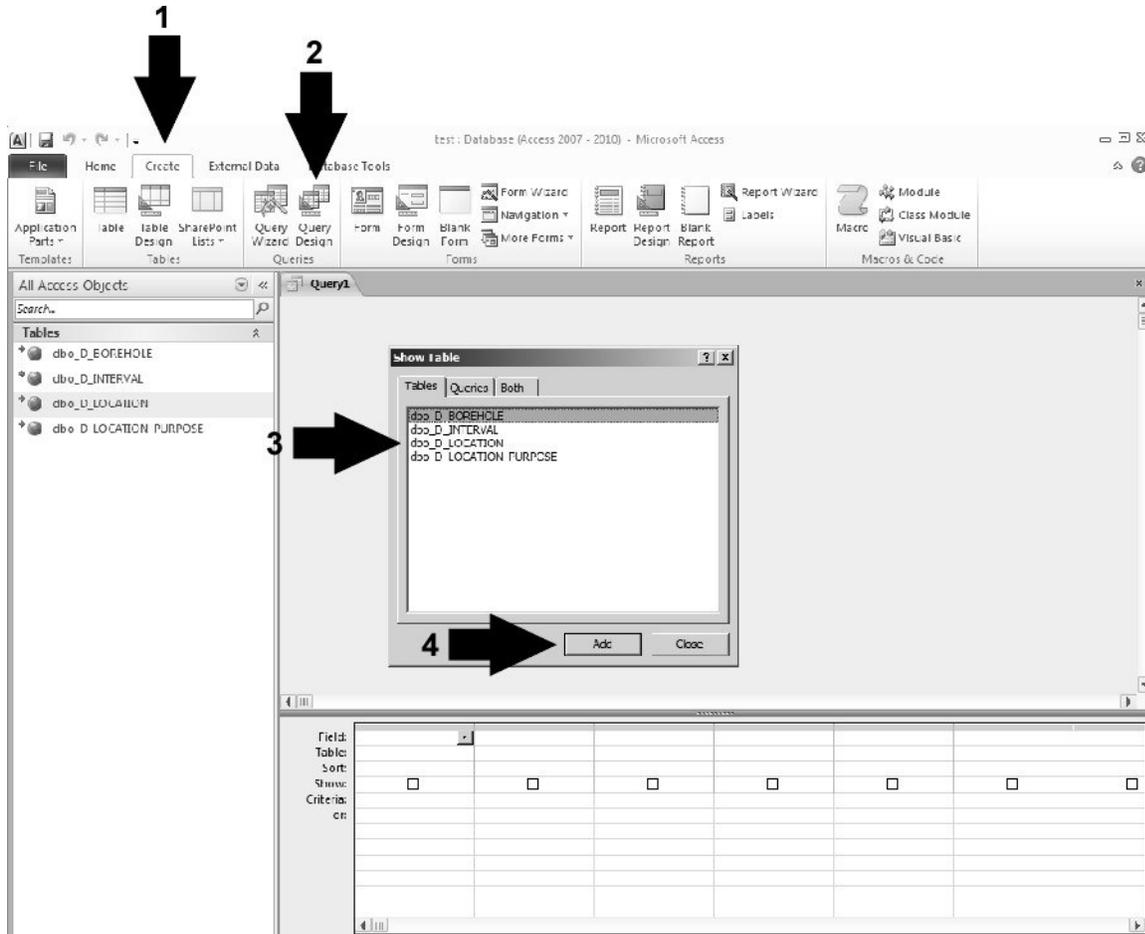
- Some locations are pumping stations (they have a LOC_TYPE of '10')
- There is no MOE number for some of the wells; an ATag number is used if known

Example 1b – Wells Associated with Nobleton (Microsoft Access 2010)

This example mirrors the previous (1a) with screenshots from Microsoft Access 2010.

Select the 'Queries' option under 'Objects' and then select 'New'

1. The 'Query Design' option is now found under 'Create – Query Design'

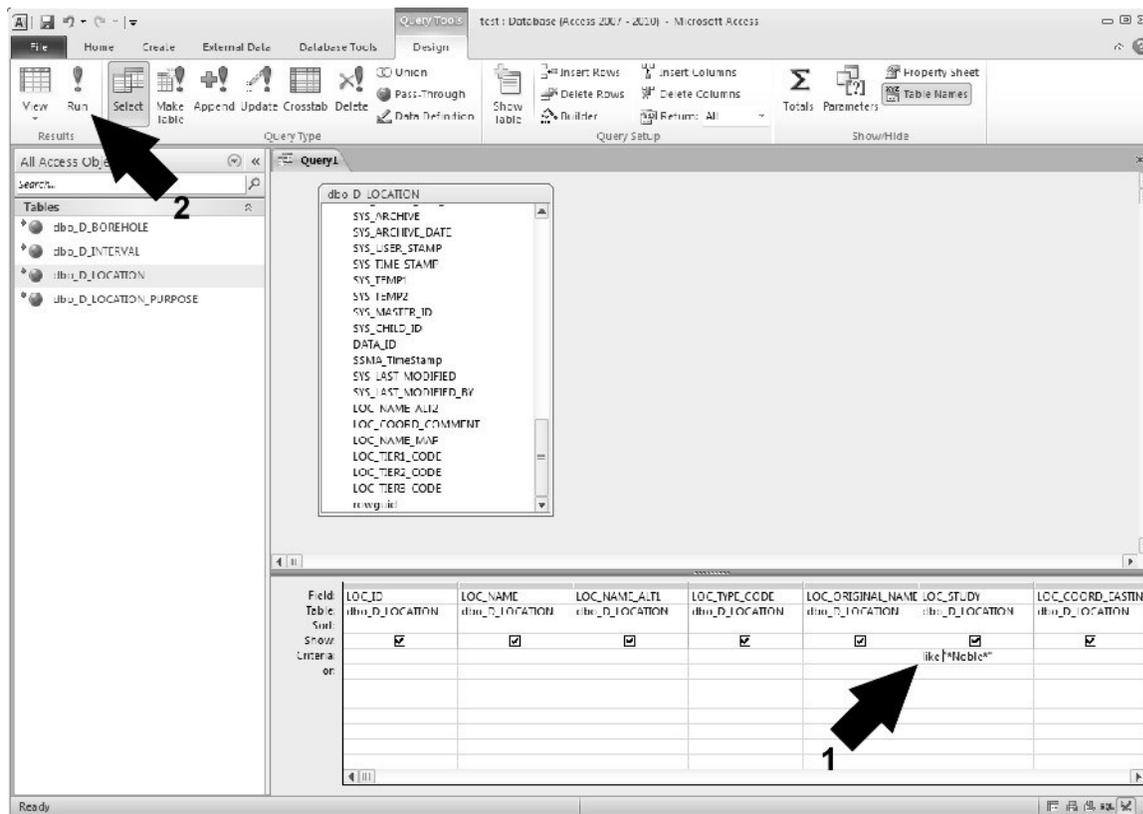


2. Add the 'D_LOCATION' table by double-clicking in the 'Show Table' dialog box (or single-clicking and selecting 'Add')
3. Add each of the following fields/columns into the query by double-clicking on them (or clicking-dragging) so they appear in the query box:

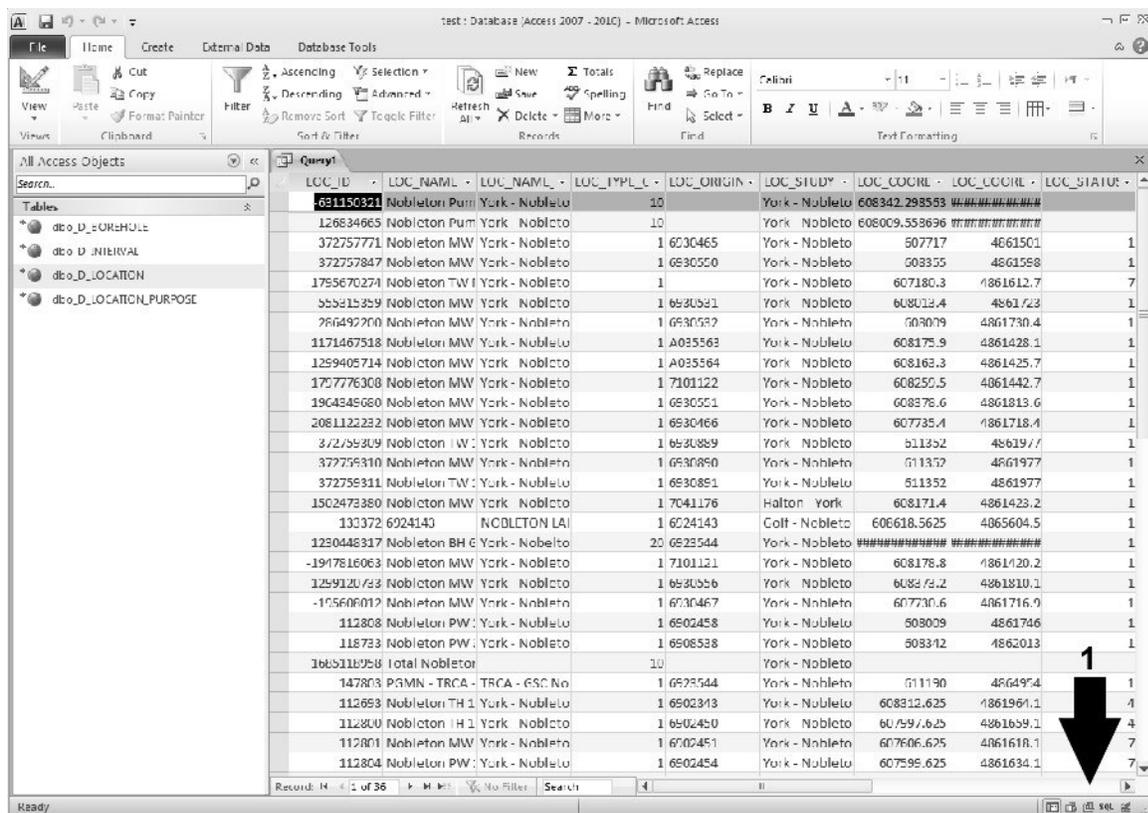
LOC_ID
 LOC_NAME
 LOC_NAME_ALT1
 LOC_TYPE_CODE
 LOC_ORIGINAL_NAME

LOC_STUDY
 LOC_COORD_EASTING
 LOC_COORD_NORTHING
 LOC_STATUS_CODE
 DATA_ID

- Filter the query under the LOC_STUDY field/column by entering 'like "*Noble*"' under 'Criteria'. Select the red exclamation mark (the 'Run' button) to run the query



- A number of records should be returned. They can be sorted by selecting 'LOC_NAME - <right-click> - Sort Ascending' (note that some of the field/column names have not been shown)



- Note that the method by which to switch between ‘Design View’, ‘SQL View’ and ‘Datasheet View’ has been moved to the lower-right of the Microsoft Access interface.

Example 1c - Wells Associated with Nobleton (Microsoft SQL Management Studio - SQL)

A similar design procedure can be accomplished in Microsoft SQL Management Studio through selection of the database (e.g. OAK_20120615_CLOCA) then selecting the ‘New Query’ button. Once the blank query is available (make sure that the cursor is within the query window) select ‘Query - Design Query in Editor’. A similar series of options as described above in ‘Example 1b’ will be presented. Once the query is finished being built/edited, the user will be returned to the query window where it can then be run. While the Access version of the query remains as part of the users (Access) database, the query built in Management Studio must be saved as a separate file (with an ‘.sql’ extension). The latter defaults to working within the SQL language as opposed to the graphic representation found in Access.

An SQL command that accomplishes the task in ‘Example 1a’ would be of the form:

```
SELECT
[LOC_ID]
,[LOC_NAME]
,[LOC_NAME_ALT1]
,[LOC_TYPE_CODE]
,[LOC_ORIGINAL_NAME]
,[LOC_STUDY]
,[LOC_COORD_EASTING]
,[LOC_COORD_NORTHING]
```

```

],[LOC_STATUS_CODE]
],[DATA_ID]
FROM [OAK_20120615_CLOCA].[dbo].[D_LOCATION]
WHERE
[LOC_STUDY] like '%noble%'
ORDER BY
[LOC_NAME]

```

This would return the same result set found in 'Example 1a'. There are a few differences between the two software packages related to formatting: the use of single quotes as opposed to double-quotes (after 'like'); the use of percent signs (%) in place of stars (*) for designating wild-card characters. Also, the '[OAK_20120615].dbo.' is unnecessary if the user is only dealing with one database.

A similar SQL command can be viewed within Access if the user selects 'View - SQL View' while in the 'Design' mode of the query (instead of in the results).

Example 2 - Bedrock Elevations using Views (Microsoft Access 2003/2010)

In this case, we'll use a 'View' within the database from which we'll extract certain pieces of information. View's in Microsoft SQL Server are used to hide the complexity of linked/joined tables from the user and to provide easier access to information through (for example) the minimization of searching/linking through tables.

Make sure the the view 'V_General_BHs_Bedrock' has been linked into Access (following the instructions of Section 3.1.1 and 'Example 1a/1b', above). Note that Access does not distinguish between tables and views, treating the latter as a table.

1. Select the 'Queries' option under 'Objects' and then select 'New'
2. 'Design View' is automatically selected (in the upper left corner). Add the 'V_General_BHs_Bedrock' view by double-clicking in the 'Show Table' dialog box (or single-clicking and selecting 'Add')
3. Add the following columns to the query (by double-clicking or by clicking/dragging):

Location ID	Bottom Elevation (masl)
Original (MOE) Name	Bedrock Elevation (masl)
Ground Elevation (masl)	

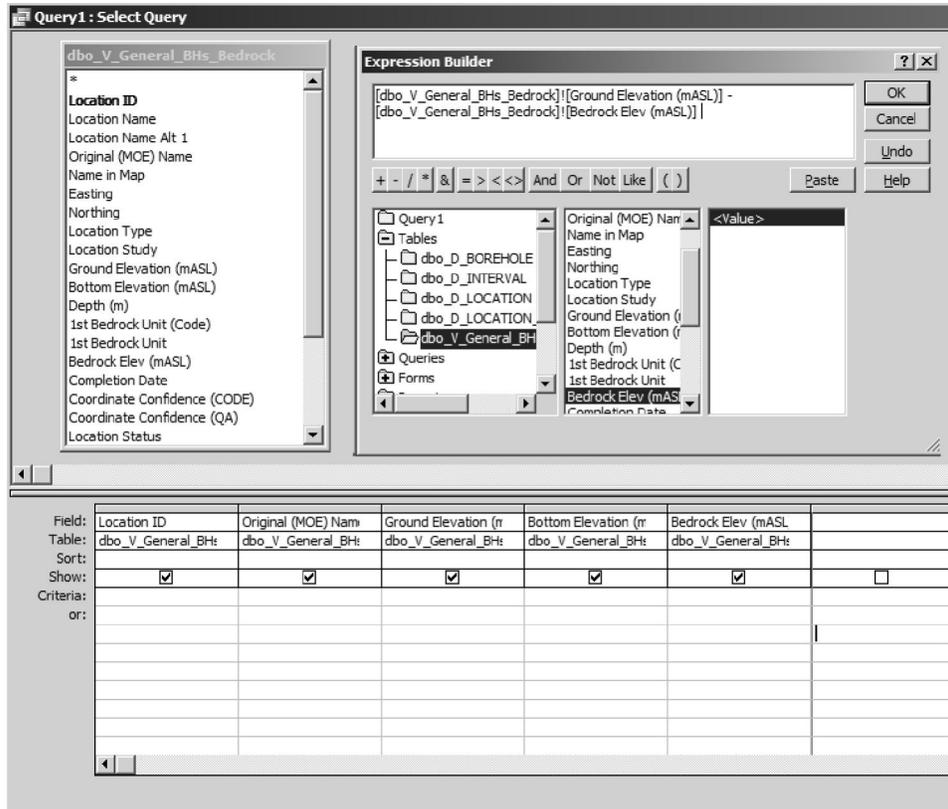
4. Run the query and note the values returned. The values themselves are found in various data tables including

- D_LOCATION
- D_BOREHOLE
- D_GEOLOGY_LAYER

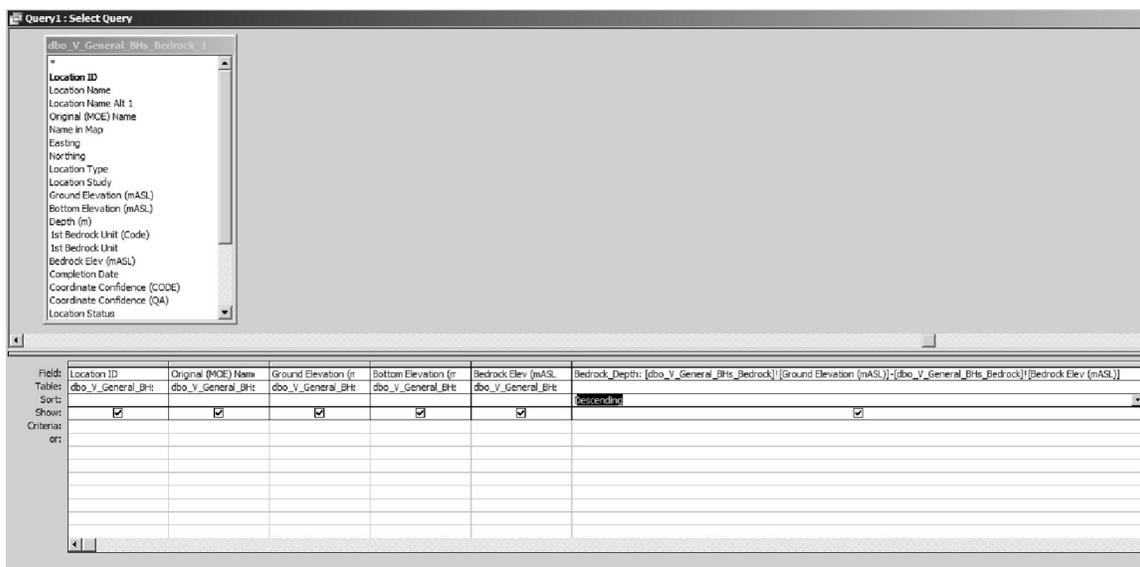
but the complexity of 'joining' the tables together based on their location is hidden within the view itself. The definition of 'bedrock' itself is present (and accessed on-the-fly) from another table - R_GEOL_MAT1_CODE - avoiding the classification of each layer in a borehole as bedrock (or not).

5. Now the bedrock elevation doesn't tell us (directly) at what depth bedrock was encountered.

Instead, we need to calculate the bedrock depth based upon the ground and bedrock elevations. To do so, return to the 'Design View' and right-click in a new/blank 'Field'; select 'Build' from the options. The 'Expression Builder' dialog box comes up - this allows you to combine fields from the tables you've included in your query. Use the options and subtract the bedrock elevation from the ground elevation.



6. Select 'Okay' when finished and change the 'Expr1:' name to 'Bedrock_Depth'. Re-run the query. Note that, now, you have new information - namely the depth at which bedrock was encountered.



7. If we, in addition, wanted to see the highest 100 depths at which bedrock was encountered, we can select a 'Sort' for the 'Depth' field - select 'Descending. To access only the top 100 (deepest), exchange the 'Design View' for the 'SQL View' - this is the actual SQL command that is run to extract the data from the view. Modify it by adding 'top 10' after the 'select' keyword - rerun the query and note that only 10 records are returned.

```

Query1 : Select Query
SELECT TOP 100 dbo_V_General_BHs_Bedrock.[Location ID], dbo_V_General_BHs_Bedrock.[Original (MOE) Name],
dbo_V_General_BHs_Bedrock.[Ground Elevation (mASL)], dbo_V_General_BHs_Bedrock.[Bottom Elevation (mASL)],
dbo_V_General_BHs_Bedrock.[Bedrock Elev (mASL)], [dbo_V_General_BHs_Bedrock].[Ground Elevation (mASL)]-
[dbo_V_General_BHs_Bedrock].[Bedrock Elev (mASL)] AS Bedrock_Depth
FROM dbo_V_General_BHs_Bedrock, dbo_V_General_BHs_Bedrock AS dbo_V_General_BHs_Bedrock_1
ORDER BY [dbo_V_General_BHs_Bedrock].[Ground Elevation (mASL)]-[dbo_V_General_BHs_Bedrock].[Bedrock Elev (mASL)];

```

8. The results are shown below

	Location ID	Original (MOE) Name	Ground Elevation (mASL)	Bottom Elevation (mASL)	Bedrock_Depth
1	-1118533832	Caledon Natural Gas Fields (Dover Oil Co.) - J. Gr	275.9568	-166.0432	860
2	-112853822	Premier Development No. 1 - L. Curran, Ancaster 3	214	-647.7	573
3	-1258472979	Imperial Oil, Glanford - 9 - III	224	-553.8	563.6
4	-608068431	W.P. Carr No.6, Ancaster - 29 - III	227	-537.7	555.3
5	372761166	7042932	279.7987	-257.2013	534
6	1569487970	NULL	280.1348	-247.1933	527.118
7	1981035710	Canadian Essex No.4, Beverly - 32 - VI	269	-466.2	524.6
8	2086131602	Imperial Oil, Saltfleet - 12 - IV	194	-545.1	523
9	1974572156	IMPERIAL 834, West Flamborough - 1 - III	249	-478.6	519.4
10	-1635710586	Okonta Oil No.3, Beverly - 29 - IV	249	-479.5	514.5
11	-1191446336	O'Konta Oil No. 2 - Lottridge No. 2, Beverly - 30	257	-509.6	512.1
12	2099809518	O'Konta Oil No. 4 - M. Lottridge No. 3, Beverly -	253	-479.1	509.6
13	-1190544554	O'Konta Oil No. 1 - Lottridge No. 1, Beverly - 30	249	-482.5	509
14	-495978137	Canadian Essex No.2, Beverly - 34 - V	251	-475.6	502.9
15	-1871347715	B.P. Exploration, West Flamborough - 2 - VI	251	-459.2	500.5
16	10721574	Canadian Essex Oil No. 1 - W. Coverdale No. 1, Eas	265	-412.9	461.8
17	-353462991	Suburban Gas Co. - T. Seynuck No.1, Nassagaweya -	334.7498	-945.4501	442
18	-380424672	Canadian Dutch Oil Company - Shelburne Well No. 2,	491.69	-533.31	418.5
19	1466844171	Tony Seynuck No. 2 - T. Seynuck No. 2 - (Anthony G	335.3098	-308.3903	415.75
20	1981453559	Imperial Oil, Saltfleet - 11 - I	84	-527.1	395.6
21	149510	6918868	272.87	150.95	389
22	372761921	7044918	212.0985	-247.9015	374
23	931092490	Page Oil & Gas Company - Page Farm, Vaughan - 33	186.3849	-179.4151	365.2
24	-1844435850	Page No.1, Vaughan - 34 - I	197.6524	-168.1476	365.2
25	225284046	Anthony Gas and Oil Exploration No. 9 - Acton No.	264.1783	-319.8217	360.9
26	372752778	5740381	261.3156	-108.6844	335
27	939361304	Star Chemical No.1, Nelson - 6 - III	108	-417.8	328
28	372750213	1917716	227.1997	-122.8003	328
29	333448329	#1, Chinguacousy - 14 - VIW	249.9101	-505.0899	327.7
30	1147189842	F. A. Ogletree - Mrs. J.A. Macklin No. 1, Nelson	105	-425.4	323.09
31	2040760924	Anthony Gas & Oil Expl. No. 32 - Homby No. 2 - J.	208.0326	-306.1674	319.7
32	372754467	1917444	273.1079	-33.89209	306
33	372754964	1918382	256.4188	-108.5812	305
34	372747452	1706249	427.9778	67.97778	302
35	372761916	7044913	218.2487	-171.7513	301
36	1417597	2516703	319	-181	301
37	372760509	7041192	223.0925	-93.90753	296.5
38	372761241	7043054	294.2565	-25.7435	290
39	1415935	6510300	365	73	285

Example 3 - Locate all Municipal Wells (using SQL - Access or Management Studio)

To locate all municipal pumping wells, the query would draw from the D_LOCATION table and the D_LOCATION_PURPOSE table, using criteria of 10 (water supply) for the primary water use, and a criteria of 22 (municipal) for the secondary water use. We also need to extract only boreholes (and not anything else) - this corresponds to a LOC_TYPE_CODE of '1'. Note that each water use has an end

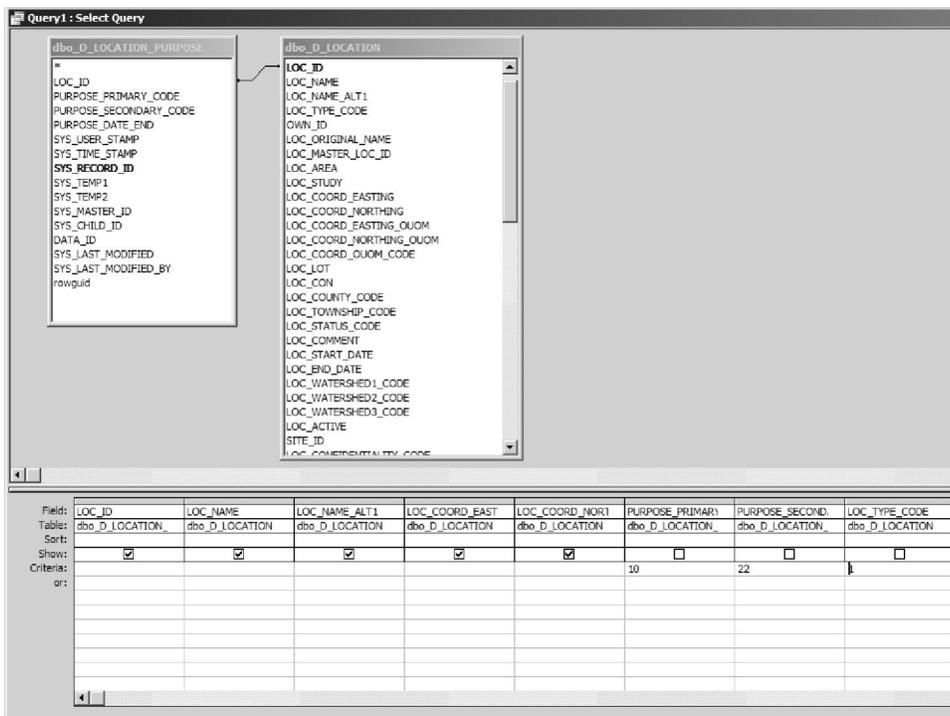
date, allowing chronological tracking of changing well uses with time. The SQL code would then be:

```

SELECT
  DLP.LOC_ID
  ,DLOC.LOC_NAME
  ,DLOC.LOC_NAME_ALT1
  ,DLOC.LOC_COORD_EASTING
  ,DLOC.LOC_COORD_NORTHING
FROM
  D_LOCATION_PURPOSE AS DLP
INNER JOIN
  D_LOCATION AS DLOC
ON
  DLP.LOC_ID=DLOC.LOC_ID
WHERE
  PURPOSE_PRIMARY_CODE=10
AND PURPOSE_SECONDARY_CODE=22
AND LOC_TYPE_CODE=1
  
```

Note here that we're using 'aliases' of table names to shorten the SQL code required (e.g. DLP for D_LOCATION_PURPOSE). Refer to Appendix A for a description of the application of SQL with regards to the ORMGP database.

The equivalent query, created in 'Access - Design View', would be similar to



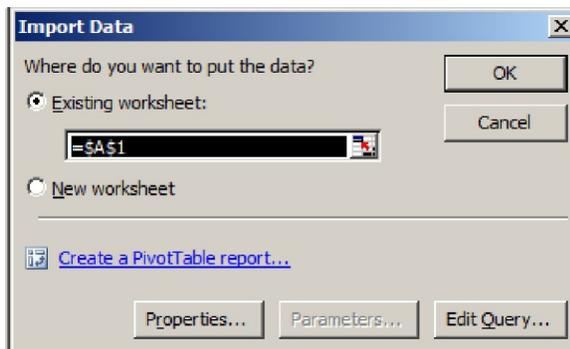
Example 4a - Plot Water Levels (using Microsoft Excel 2003)

It has been the case that datasets have been stored across multiple files (typically Access or Excel files)

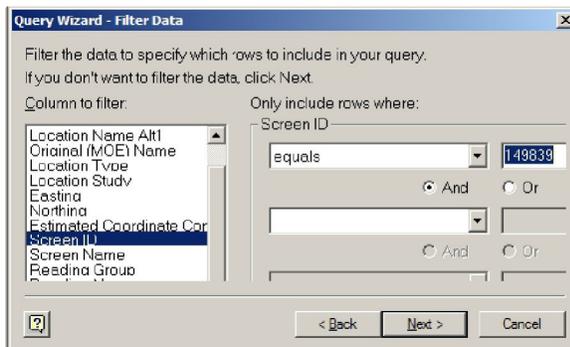
with information being copied-pasted as required for creating a particular one-off dataset. It is more advantageous to centralize this information (and thereby reducing possible errors during the transferral process) and access it as necessary. This allows information updates to be rolled into analysis without lengthy modification/reformatting procedures. When the user revisits the spreadsheet at some point in the future, a refresh of the dataset will cause any new information to be included automatically (refer also the comments at the end of this example).

One primary example of this is the plotting, using Excel, of water level data. This example assumes that the user has already reviewed Section 3.1.1 (General Database Access) and created the necessary files/odbc-links enabling Excel to communicate with the database. (Note that the reference version of Excel is 2003. This example should apply to more recent versions with only slightly different paths for accessing the Excel functions.)

- Start Excel and select 'Data - Import External Data - Import Data'
- Select the previously created database DSN file (refer to Section 3.1.1 for this information)
- Select the V_Temporal_Data view (Excel will treat this as a table; it may be referenced as dbo.V_Temporal_Data)
- Add the data to the existing worksheet



- Select 'Edit Query' then 'Next'
- Select 'Column To Filter - Screen ID' and add (equal) the INT_ID 149839 (note that this is the ORMGP borehole 294-4 and may not be present in all partner database's; select an appropriate INT_ID if necessary)

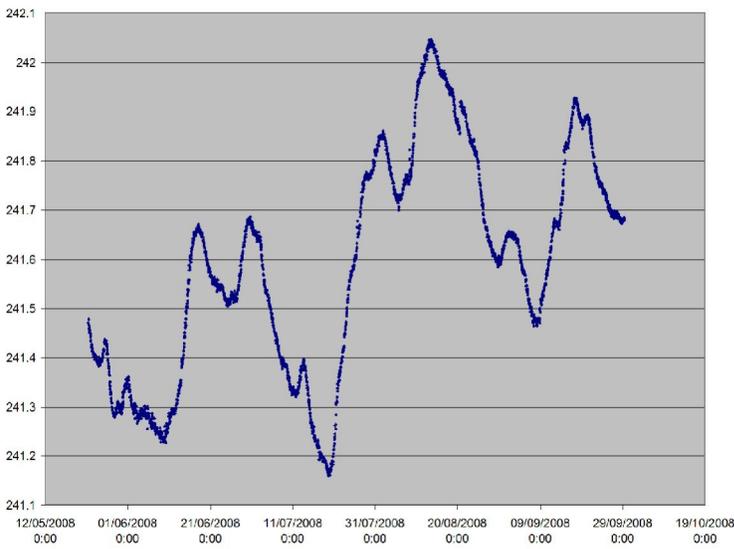


- Select 'Column To Filter - Reading Name' and make it equal to 'Water Level - Logger (Compensated BP)' (note that the text must be exact to match against; refer to R_READING_NAME_CODE if necessary)

- Select 'Next', then 'Sort by - Date'
- Select 'Next', then 'Return Data to Microsoft Excel'

The screenshot shows a Microsoft Excel spreadsheet with a large table of data. The columns include 'Location', 'Station', 'Reading Name', and 'Value'. The data appears to be a time series of readings from various stations, with values ranging from approximately 241.1 to 242.1. The x-axis represents time, with labels for dates like 12/05/2008, 01/06/2008, 21/06/2008, 11/07/2008, 31/07/2008, 20/08/2008, 09/09/2008, 29/09/2008, and 19/10/2008.

- This returns the data set to Excel (the current spreadsheet) where it can, subsequently, be plotted (the description for plotting in Excel is not provided)



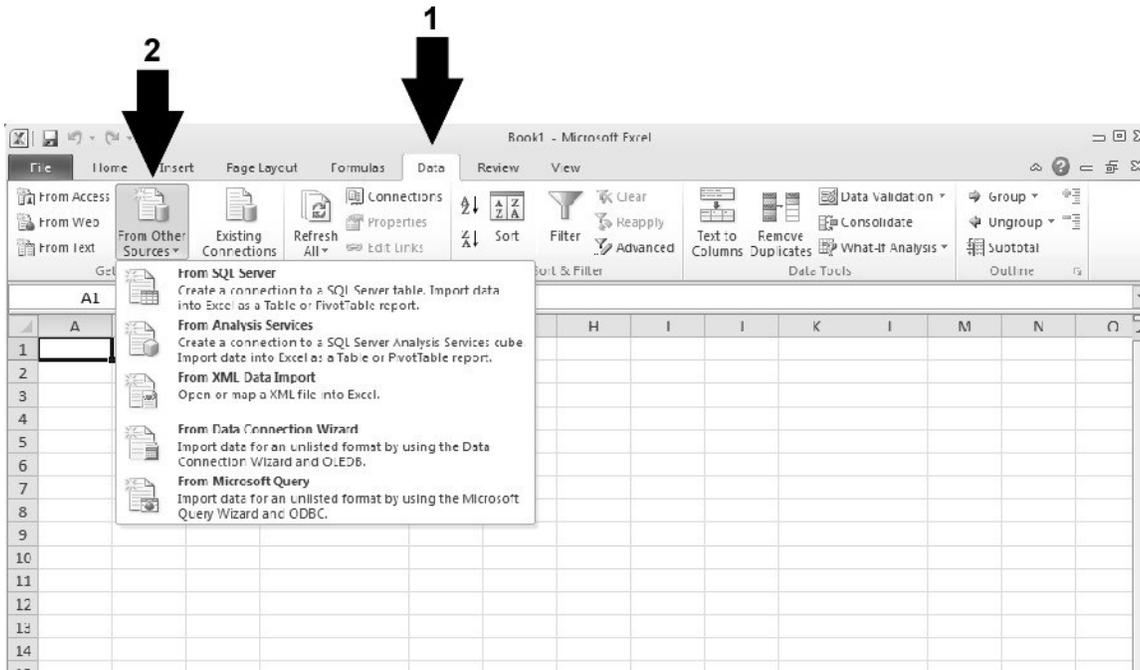
A few comments regarding this process

- Depending upon the version of Excel being used, it may be limited to a maximum number of rows (commonly ~32000 or ~65000); if so, the query should be modified (for example, by setting a 10 yr interval on the data, then creating multiple imported dataset to plot against)
- The user can drop unneeded columns from this process thereby reducing the resultant file size
- Plotting of the data requires the user to specify the starting and ending row numbers; however, when (or if) the data is refreshed (so as to automatically update the plot), more (i.e. updated) information may come into the spreadsheet than was originally available when the plot was created; as such, the user will find it necessary to modify the graphs to account for this

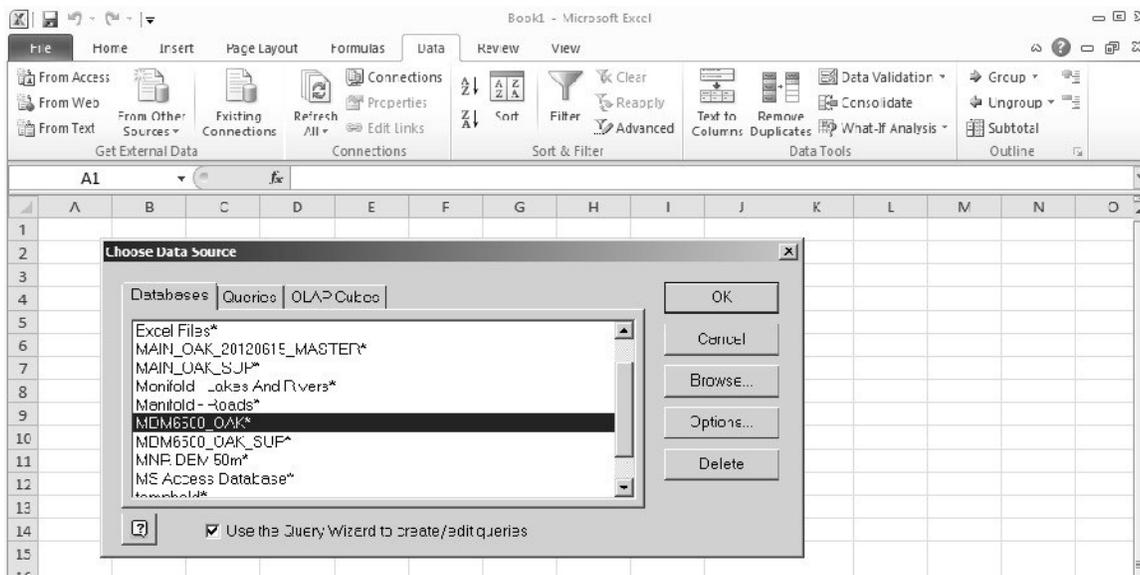
additional data (by reselecting the extents of the rows/columns)

Example 4b - Plot Water Levels (using Microsoft Excel 2010)

The method for plotting water levels in the updated version of Microsoft Excel is, basically, equivalent to that described in Example 4a (above). Access to an external data source is enabled through the 'Data – Other Sources' tab.

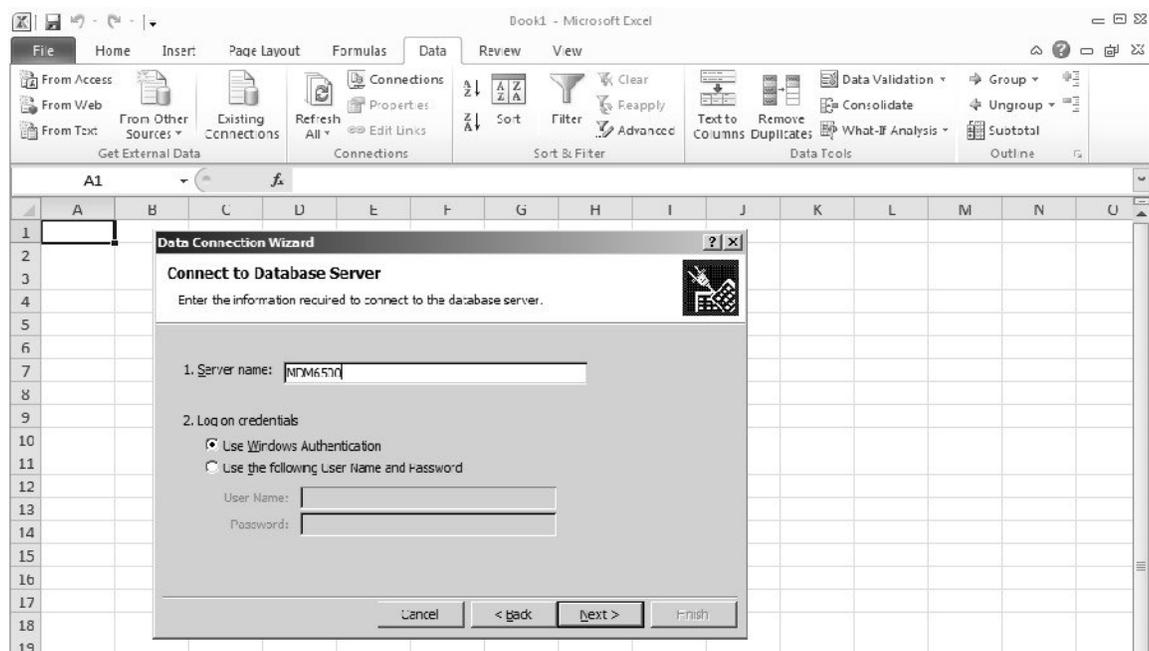


Selecting 'From Microsoft Query' allows the user to select the (previously enabled) database source (through, for example, the 'ODBC Administrator' program).

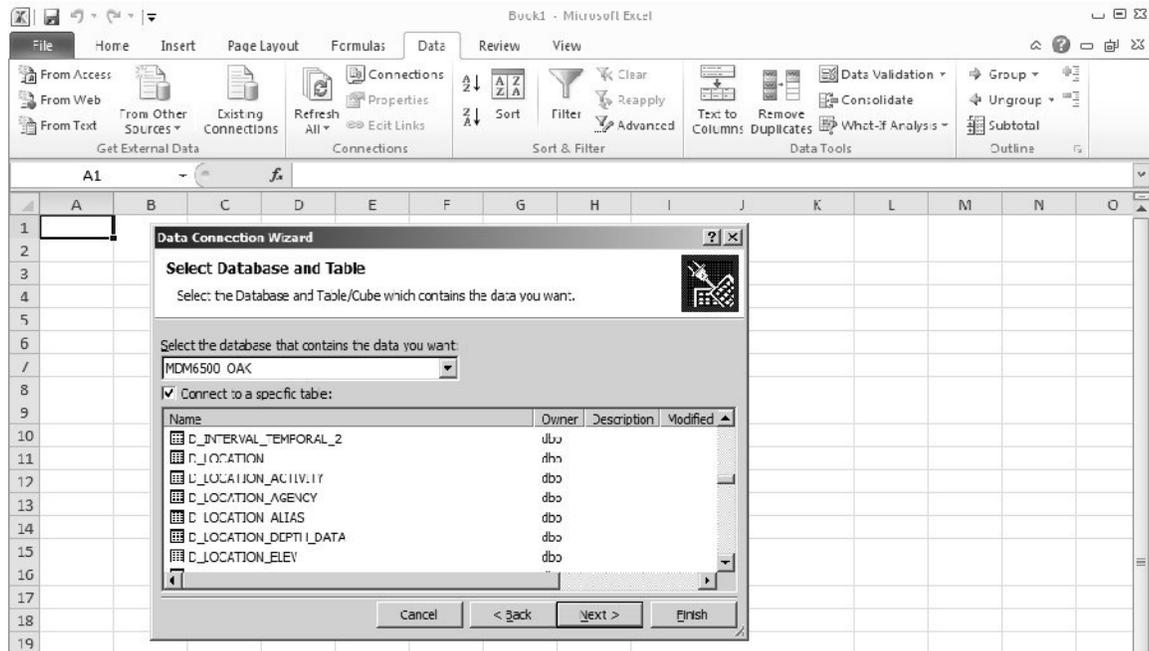


Here the database chosen would be 'MDM6500_OAK'; selecting 'OK' would take the user to the 'Query Designer' window (outlined previously).

Selecting, instead, the 'From SQL Server' (i.e. a direct connection) allows the user to specify the SQL Server (name) holding the ORMGP partner database.



And then select the database and table to import into Excel.



However, note that many of the tables present in the database will have an exceedingly large number of rows (and, potentially, columns) and the entire table would be imported. Using an alternate method of connecting to the database, allowing the reduction of the number of rows returned through Excel's 'Query Designer' is recommended.

Example 5 - Flow Gauge Locations (using SQL - Access or Management Studio)

Flow gauges, as they would record what would be considered as ‘temporal’ data, are assigned a particular type in the R_INT_TYPE_CODE table (i.e. ‘Surface Water Flow Gauge’ has a value of ‘4’). As such, the D_INTERVAL table is the first place to start determining the location of flow gauges. The SQL command

```
SELECT
*
FROM
[OAK_20120615_CLOCA].dbo.D_INTERVAL
WHERE
INT_TYPE_CODE=4
```

would return all rows where an ‘interval’ is identified as a flow gauge.

INT_ID	LOC_ID	INT_NAME	INT_NAME_ALT1	INT_NAME_ALT2	INT_TYPE_CODE	INT_START_DATE	INT_END_DATE	INT_COMMENT	INT_AAS_LOSSES	INT_COORDINATE_CODE	INT_REGISTRY_CODE	INT_NAME_SORT	CHGY_CLASS_CODE
1	268862874	268862874	268862874	268862874	NULL	NULL	NULL	NULL	0	1	NULL	NULL	NULL
2	268862874	268862874	268862874	268862874	NULL	NULL	NULL	NULL	0	1	NULL	NULL	NULL
3	192221902	192221902	192221902	192221902	NULL	NULL	NULL	NULL	0	1	NULL	NULL	NULL
4	192221902	192221902	192221902	192221902	NULL	2008-01-01 00:00:00	2008-09-26 00:00:00	NULL	0	1	NULL	NULL	NULL
5	192222020	192222020	192222020	192222020	NULL	NULL	NULL	NULL	0	1	NULL	NULL	NULL
6	154547557	154547557	154547557	154547557	NULL	NULL	NULL	NULL	0	1	NULL	NULL	NULL
7	142454781	142454781	142454781	142454781	NULL	NULL	NULL	NULL	0	1	NULL	NULL	NULL
8	142454781	142454781	142454781	142454781	NULL	NULL	NULL	NULL	0	1	NULL	NULL	NULL
9	142454781	142454781	142454781	142454781	NULL	2008-01-01 00:00:00	2008-11-02 00:00:00	NULL	0	1	NULL	NULL	NULL
10	152070985	152070985	152070985	152070985	NULL	NULL	NULL	NULL	0	1	NULL	NULL	NULL
11	152070985	152070985	152070985	152070985	NULL	2008-01-01 00:00:00	2008-11-02 00:00:00	NULL	0	1	NULL	NULL	NULL
12	170268601	170268601	170268601	170268601	NULL	NULL	NULL	NULL	0	1	NULL	NULL	NULL
13	170268601	170268601	170268601	170268601	NULL	NULL	NULL	NULL	0	1	NULL	NULL	NULL
14	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
15	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
16	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
17	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
18	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
19	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
20	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
21	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
22	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
23	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
24	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
25	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
26	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
27	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
28	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
29	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
30	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
31	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
32	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
33	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
34	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
35	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
36	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
37	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
38	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
39	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
40	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
41	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
42	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
43	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
44	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
45	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
46	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
47	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
48	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
49	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL
50	170268601	170268601	170268601	170268601	NULL	2008-01-01 00:00:00	2008-09-16 00:00:00	NULL	0	1	NULL	NULL	NULL

All columns (here, specified using ‘*’) would be returned. In reality, we only need the LOC_ID from this table in order to access the related information from the D_LOCATION table. We’ll now perform a ‘relational join’ between D_INTERVAL and D_LOCATION to extract the coordinates and names of the flow gauge stations.

```
SELECT
dint.LOC_ID
,dint.INT_ID
,dloc.LOC_NAME
,dloc.LOC_NAME_ALT1
,dloc.LOC_COORD_EASTING
,dloc.LOC_COORD_NORTHING
FROM
D_INTERVAL AS dint
INNER JOIN
D_LOCATION AS dloc
ON
```

```
dint.LOC_ID=dloc.LOC_ID
WHERE
dint.INT_TYPE_CODE=4
```

Note that we're using 'aliases' (with the AS keyword) to shorten the names of the D_LOCATION (i.e. dloc) and D_INTERVAL (i.e. dint) tables. This would then return the requisite information.

	LOC_ID	INT_ID	LOC_NAME	LOC_NAME_ALT1	LOC_COORD_EASTING	LOC_COORD_NORTHING
1	-681349540	-681349540	02EB018	LAKE MUSKOKA AT BEAUMARIS	618558.805746067	4991891.61240716
2	-751189504	-751189504	02HA023	REDHILL CREEK AT ALBION FALLS	595821.748779528	4783761.73530917
3	-688683937	-688683937	02HA014	REDHILL CREEK AT HAMILTON	598681.903692539	4787320.10244678
4	-398655850	-398655850	02HA006	TWENTY MILE CREEK AT BALLS FALLS	631487.122402047	4776920.71969026
5	-485176429	-485176429	02HB021	ANCASTER CREEK AT ANCASTER	583322.76050046	4786990.88623147
6	1102621431	-1580206228	CVC Credit River (Glen Williams)	NULL	596496	4835410
7	-2058621974	-2058621974	02EB023	GO HOME LAKE NEAR POTTERS LANDING	592345.67791415	4982074.90580965
8	-2056054271	-2056054271	02HA027	WALKER CREEK AT ST. CATHARINES	643900.623406679	4786306.85875421
9	-1932741597	-1932741597	02GB001	GRAND RIVER AT BRANTFORD	559598.847605316	4775820.2875623
10	2147456354	-1586154519	02HD023	MACKIE CREEK NEAR HAMPTON	686049	4872144
11	-1547476857	-1547476857	02HJ004	CAVAN CREEK NEAR KENDRY	707945.948058147	4900407.07610831
12	-1425475814	-1425475814	02HB017	LAKE ONTARIO AT BURLINGTON	597919.507918987	4794806.84273866
13	-1400474601	-1400474601	02HB007	SPENCER CREEK AT DUNDAS	584064.847077934	4790825.8203145
14	2147456355	-1395337935	02HG003	BLACKSTOCK CREEK NEAR BLACKSTOCK	673683	4888789
15	-1320318983	-1320318983	02HB027	FOURTEEN MILE CREEK AT OAKVILLE	604987.525565366	4808397.01559827
16	-1274372922	-1274372922	02HC051	CENTREVILLE CREEK NEAR ALBION	593566.138556224	4864141.35563044
17	-1193839091	-1193839091	02HL007	MOIRA RIVER NEAR TWEED	792656.657918244	4932735.81492047
18	-1118703211	-1118703211	02HC054	LYNDE CREEK AT BROOKLIN	663668.62736316	4869360.54987151
19	2147456357	-1002607453	02HJ007	BAXTER CREEK AT MILLBROOK	704690	4875073
20	-978532688	-978532688	02HD019	COBOURG BROOK AT COBOURG	725826.165714991	4871838.27081322
21	2147456342	-856029220	02ED030	SILVER CREEK AT ORILLIA	622758	4944866
22	2147456339	-75007904	02EC019	BLACK RIVER NEAR VANDOUGHNET	653754	4983968
23	-681577507	-681577507	02HA016	THREE MILE CREEK AT MOUNT HOPE	588619.910025777	4779838.83659807

The equivalent query, created in 'Access - Design View', would be similar to

The screenshot shows the 'Query1: Select Query' design view in Microsoft Access. It features two tables: 'dbo_D_INTERVAL' and 'dbo_D_LOCATION'. The 'Criteria' row for the 'INT_TYPE_CODE' field in the 'dbo_D_LOCATION' table is set to '4'. The design grid is as follows:

Field:	INT_ID	LOC_ID	LOC_NAME	LOC_NAME_ALT1	LOC_COORD_EAST	LOC_COORD_NORT	INT_TYPE_CODE
Table:	dbo_D_INTERVAL	dbo_D_INTERVAL	dbo_D_LOCATION	dbo_D_LOCATION	dbo_D_LOCATION	dbo_D_LOCATION	dbo_D_LOCATION
Sort:							
Show:	<input checked="" type="checkbox"/>						
Criteria:							4
or:							

Example 6 – Bedrock wells with 6 inch casing (using SQL – Access or Management Studio)

This question is somewhat complicated in that multiple tables need to be accessed to produce the result. These include

Tables

- D_LOCATION
- D_BOREHOLE
- D_BOREHOLE_CONSTRUCTION

Views

- V_General_BHs_Bedrock

Fields

- Location ID (from V_General_BHs_Bedrock)
- LOC_ID
- BH_ID
- CON_DIAMETER (from D_BOREHOLE_CONSTRUCTION)
- CON_DIAMETER_OUOM (from D_BOREHOLE_CONSTRUCTION)
- CON_DIAMETER_UNIT_OUOM (from D_BOREHOLE_CONSTRUCTION)
- BH_BEDROCK_ELEV (from D_BOREHOLE; an alternate method)

Any wells found in 'V_General_BHs_Bedrock' are, by design, wells in bedrock - the remainder of the required information is found elsewhere. In this case, the 'V_General_BHs_Bedrock' view and D_LOCATION table need to be linked (based on 'Location ID' and LOC_ID). This would allow the user to then link to D_BOREHOLE (also based on LOC_ID) in order to determine the BH_ID. This last piece of information allows the user to link to D_BOREHOLE_CONSTRUCTION (based on BH_ID). Finally, the content of CON_DIAMETER can be examined looking, initially for a value of '6'. Note that there is no conversion when copying the borehole diameter from CON_DIAMETER_OUOM to CON_DIAMETER. A whole number in CON_DIAMETER, for example '6', likely means '6 inches' - CON_DIAMETER_UNIT_OUOM could also be examined as a check. However, in some cases during data entry, a '6 inch' value could (potentially) have been converted to a width in cm's or m's; for completeness, those values (15.2 and 0.152) should be checked as well.

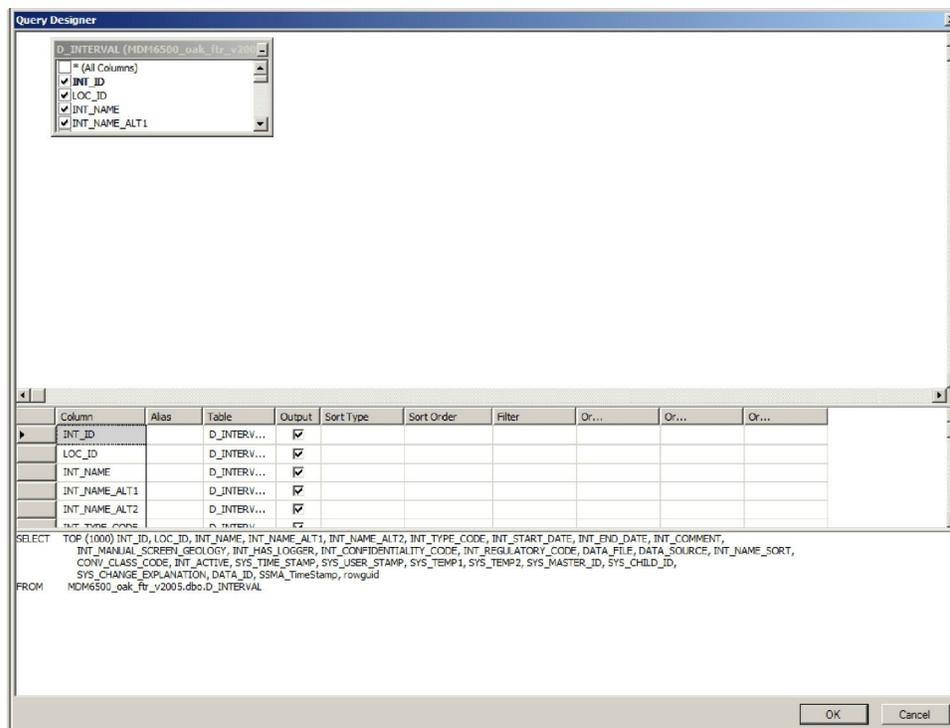
Note that D_BOREHOLE now contains a BH_BEDROCK_ELEV field. When non-NULL, this indicates that the particular borehole has reached bedrock. This field can be used instead of involving the view 'V_General_BHs_Bedrock'.

Section J.2 Training Exercises (Moderate)

Example 1a - Assembling Interval Data (Microsoft SQL Management Studio)

Using Microsoft SQL Server Management Studio (MSSQLSM) we'll assemble all pertinent information for a series of intervals in the database. We'll use 'Query Designer' to build the query.

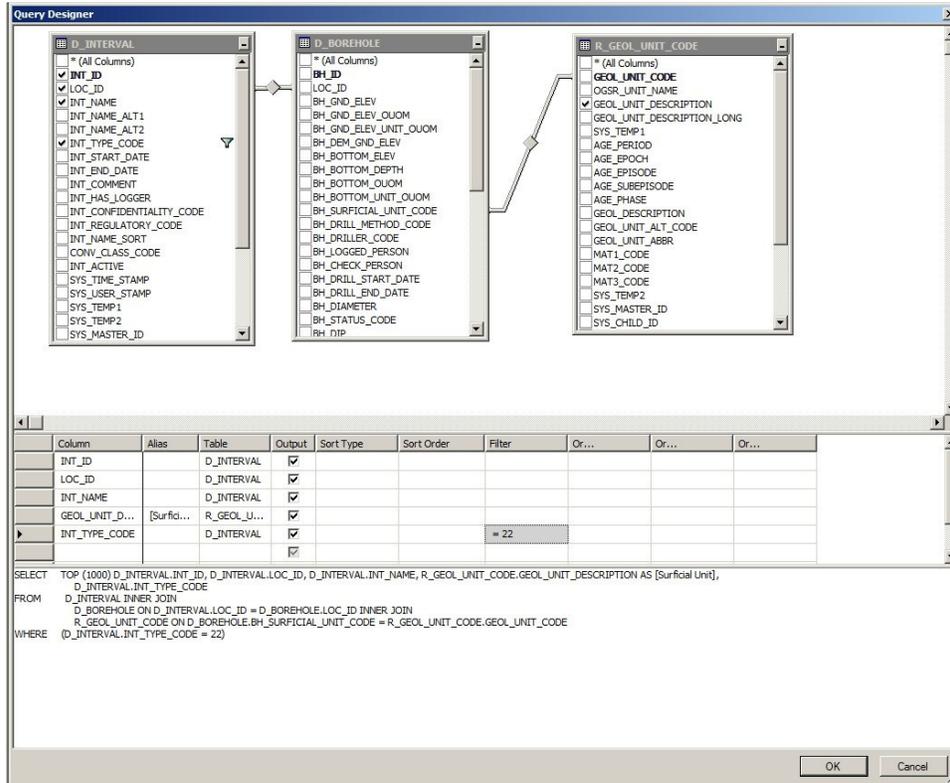
1. Make sure that the partner database is selected then select the first 1000 records from the D_INTERVAL table (this is an option under 'D_INTERVAL - <right-click> - Select Top 1000 Rows').
2. Select (highlight) the resultant SQL code and choose 'Query - Design Query in Editor'



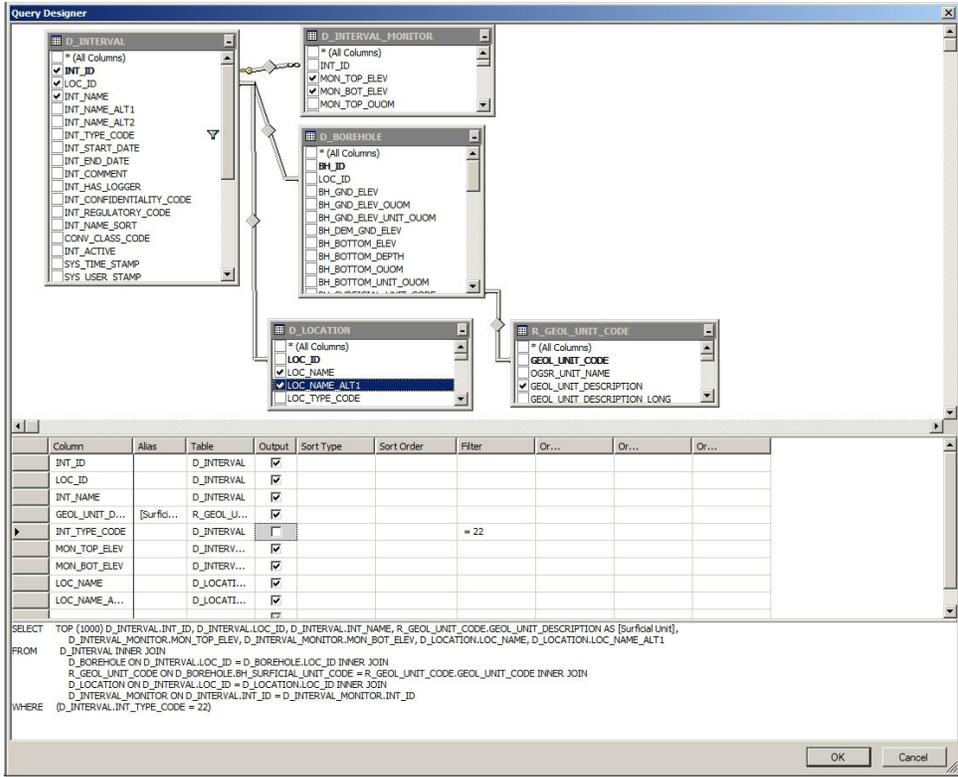
3. Deselect (for Output) all fields with the exception of LOC_ID, INT_ID, INT_NAME and INT_TYPE_CODE. Add '=22' to the 'Filter' column for INT_TYPE_CODE (this limits the result of the query to only those intervals of type 'Reported Screen' - this information is found in the table R_INT_TYPE_CODE). Deselect the INT_TYPE_CODE for 'Output'.

4. Add the tables D_BOREHOLE and R_GEOL_UNIT_CODE (in the query window '<right-click> - Add Table - <double-click on table(s)>'). Join the D_INTERVAL and D_BOREHOLE tables based upon the LOC_ID (select 'D_INTERVAL - LOC_ID' and drag to the 'D_BOREHOLE - LOC_ID'). Join the D_BOREHOLE and R_GEOL_UNIT_CODE tables based upon the BH_SURFICIAL_UNIT_CODE and the

GEOL_UNIT_CODE. Add the GEOL_UNIT_DESCRIPTION to the list of 'Output' columns and change its 'Alias' to 'Surficial Unit'.



4. Add the D_LOCATION and D_INTERVAL_MONITOR tables. Join them to D_INTERVAL based on LOC_ID and INT_ID respectively (unless they're already joined). Add LOC_NAME, MON_TOP_ELEV and MON_BOT_ELEV to the output fields.



5. Add tables D_INTERVAL_FORMATION_ASSIGNMENT and a second R_GEOL_UNIT_CODE. Join the former to D_INTERVAL using INT_ID and break the join on the second R_GEOL_UNIT_CODE and join it to D_INTERVAL_FORMATION_ASSIGNMENT using ASSIGNED_UNIT and GEOL_UNIT_CODE. Add a second GEOL_UNIT_DESCRIPTION from this table and make its 'Alias' 'Interval Unit'.

Query Designer

Column

Column	Alias	Table	Output	Sort Type	Sort Order	Filter	Or...	Or...	Or...
INT_ID		D_INTERVAL	<input checked="" type="checkbox"/>						
LOC_ID		D_INTERVAL	<input checked="" type="checkbox"/>						
INT_NAME		D_INTERVAL	<input checked="" type="checkbox"/>						
GEOL_UNIT_D...	[Surfci...	R_GEOLOGICAL...	<input checked="" type="checkbox"/>						
MON_TOP_ELEV		D_INTERVAL...	<input checked="" type="checkbox"/>						
MON_BOT_ELEV		D_INTERVAL...	<input checked="" type="checkbox"/>						
LOC_NAME		D_LOCATION...	<input checked="" type="checkbox"/>						
GEOL_UNIT_D...	[Interv...	R_GEOLOGICAL...	<input checked="" type="checkbox"/>						
INT_TYPE_CODE		D_INTERVAL	<input checked="" type="checkbox"/>			= 22			

```

SELECT TOP (1000) D_INTERVAL.INT_ID, D_INTERVAL.LOC_ID, D_INTERVAL.INT_NAME, R_GEOLOGICAL_UNIT_CODE.GEOL_UNIT_DESCRIPTION AS [Surficial Unit],
D_INTERVAL_MONITOR.MON_TOP_ELEV, D_INTERVAL_MONITOR.MON_BOT_ELEV, D_LOCATION.LOC_NAME,
R_GEOLOGICAL_UNIT_CODE_1.GEOL_UNIT_DESCRIPTION AS [Interval Unit]
FROM
D_INTERVAL INNER JOIN
D_BOREHOLE ON D_INTERVAL.LOC_ID = D_BOREHOLE.LOC_ID INNER JOIN
R_GEOLOGICAL_UNIT_CODE ON D_BOREHOLE.BH_SURFICIAL_UNIT_CODE = R_GEOLOGICAL_UNIT_CODE.GEOL_UNIT_CODE INNER JOIN
D_LOCATION ON D_INTERVAL.LOC_ID = D_LOCATION.LOC_ID INNER JOIN
D_INTERVAL_MONITOR ON D_INTERVAL.INT_ID = D_INTERVAL_MONITOR.INT_ID INNER JOIN
D_INTERVAL_FORMATION_ASSIGNMENT ON D_INTERVAL.INT_ID = D_INTERVAL_FORMATION_ASSIGNMENT.INT_ID INNER JOIN
R_GEOLOGICAL_UNIT_CODE AS R_GEOLOGICAL_UNIT_CODE_1 ON
  
```

OK Cancel

6. Add the table D_LOCATION_ELEV and join to the D_LOCATION table using LOC_ID. Add the ASSIGNED_ELEV to the 'Output' fields. We'll also calculate the depths of the 'Reported Screen' - in the 'Column' field, type in 'ASSIGNED_ELEV - MON_TOP_ELEV' and give it an 'Alias' of MON_TOP_DEPTH; do the same for MON_BOT_ELEV and give it an 'Alias' of MON_BOT_DEPTH.

Query Designer

Column

Column	Alias	Table	Output	Sort Type	Sort Order	Filter	Or...	Or...	Or...
GEOL_UNIT_DESCRIPTION	[Interval Unit]	R_GEOLOGICAL...	<input checked="" type="checkbox"/>						
ASSIGNED_ELEV		D_LOCATION...	<input checked="" type="checkbox"/>						
D_LOCATION_ELEV.ASSIGNED...	MON_TOP_DEPTH	D_LOCATION...	<input checked="" type="checkbox"/>						
D_LOCATION_ELEV.ASSIGNED...	MON_BOT_DEPTH	D_LOCATION...	<input checked="" type="checkbox"/>						
INT_TYPE_CODE		D_INTERVAL	<input checked="" type="checkbox"/>			= 22			

```

SELECT TOP (1000) D_INTERVAL.INT_ID, D_INTERVAL.LOC_ID, D_INTERVAL.INT_NAME, R_GEOLOGICAL_UNIT_CODE.GEOL_UNIT_DESCRIPTION AS [Surficial Unit],
D_INTERVAL_MONITOR.MON_TOP_ELEV, D_INTERVAL_MONITOR.MON_BOT_ELEV, D_LOCATION.LOC_NAME,
R_GEOLOGICAL_UNIT_CODE_1.GEOL_UNIT_DESCRIPTION AS [Interval Unit], D_LOCATION_ELEV.ASSIGNED_ELEV,
D_LOCATION_ELEV.ASSIGNED_ELEV - D_INTERVAL_MONITOR.MON_TOP_ELEV AS MON_TOP_DEPTH,
D_LOCATION_ELEV.ASSIGNED_ELEV - D_INTERVAL_MONITOR.MON_BOT_ELEV AS MON_BOT_DEPTH
FROM
D_INTERVAL INNER JOIN
D_BOREHOLE ON D_INTERVAL.LOC_ID = D_BOREHOLE.LOC_ID INNER JOIN
R_GEOLOGICAL_UNIT_CODE ON D_BOREHOLE.BH_SURFICIAL_UNIT_CODE = R_GEOLOGICAL_UNIT_CODE.GEOL_UNIT_CODE INNER JOIN
D_LOCATION ON D_INTERVAL.LOC_ID = D_LOCATION.LOC_ID INNER JOIN
D_INTERVAL_MONITOR ON D_INTERVAL.INT_ID = D_INTERVAL_MONITOR.INT_ID INNER JOIN
D_INTERVAL_FORMATION_ASSIGNMENT ON D_INTERVAL.INT_ID = D_INTERVAL_FORMATION_ASSIGNMENT.INT_ID INNER JOIN
R_GEOLOGICAL_UNIT_CODE AS R_GEOLOGICAL_UNIT_CODE_1 ON
D_INTERVAL_FORMATION_ASSIGNMENT.ASSIGNED_UNIT = R_GEOLOGICAL_UNIT_CODE_1.GEOL_UNIT_CODE INNER JOIN
D_LOCATION_ELEV ON D_INTERVAL.LOC_ID = D_LOCATION_ELEV.LOC_ID
WHERE
D_INTERVAL.INT_TYPE_CODE = 22
  
```

OK Cancel

7. Exit out of 'Query Designer' and run/execute the query. Note that, as both INT_ID and LOC_ID are present in the results, we could subsequently relate additional tables based on those keys (say, for example, by limiting the search to those locations tagged using the SYS_TEMP2 field). An example of the output is provided though results at each partner agency will likely be different:

INT_ID	LOC_ID	INT_NAME	Surficial Unit	MON_TOP_ELEV	MON_BOT_ELEV	LOC_NAME	Interval Unit	ASSIGNED_ELEV	MON_TOP_DEPTH	MON_BOT_DEPTH
1	-2138734108	912441011	5115572	254.2896	236.916	5115572	YPDT - Late Stage Glaciolacustrine-Glaciofluvial	256.728	2.4384	19.812
2	-2123267813	1510189772	4509331	175.7196	123.962	4509331	YPDT - Late Stage Glaciolacustrine-Glaciofluvial	176.158	2.4384	44.196
3	-2129353910	72995242	4513347	72.37827	70.54947	4513347	YPDT - Bedrock - Undifferentiated	76.64547	4.267197	6.096001
4	-2128129128	1268018580	5116348	201.8704	197.6032	5116348	YPDT - Lower Newmarket	216.196	14.32651	18.5928
5	-2121440985	620891056	2800799	190.1535	180.0951	2800799	YPDT - Madokaw/Oak Ridges (MIS/ORAC)	204.4791	14.32651	24.384
6	-2119244851	-1123345472	4509238	176.9855	176.6811	4509238	Sunnybrook Till	204.3323	27.7368	28.6512
7	-2101419374	-1619384938	2808229	208.2784	199.741	2808229	YPDT - Bedrock - Undifferentiated	222	13.72459	22.289
8	-2089088314	835176707	4509574	124.7978	123.2738	4509574	Sunnybrook Till	205.8746	81.0768	82.6008
9	-2095676111	2002300290	2709989	397.8417	295.8337	2709989	YPDT - Late Stage Glaciolacustrine-Glaciofluvial	399.4557	1.523887	103.632
10	-2093330795	468964867	5112520	271.218	250.7964	5112520	YPDT - Late Stage Glaciolacustrine-Glaciofluvial	271.9228	0.3048996	20.72639
11	-2090488809	1294350984	5115178	236.2549	204.8804	5115178	YPDT - Late Stage Glaciolacustrine-Glaciofluvial	236.9596	0.3047943	31.6992
12	-2079894399	-1909852188	5116317	180.0567	179.7519	5116317	YPDT - Bedrock - Undifferentiated	195.9063	15.84959	16.1544
13	-2078661113	-435998317	5112489	68.5	68.4	5112489	YPDT - Bedrock - Undifferentiated	102.6027	34.10268	34.20267
14	-2073161031	-1020875181	2804005	170.8551	168.4167	2804005	Halton Till	178	7.144897	9.583298
15	-2065749215	883121204	2905389	319.0413	317.8221	2905389	YPDT - Late Stage Glaciolacustrine-Glaciofluvial	336.7197	17.67841	18.89761
16	-2061141451	1794339063	5117025	240.2775	238.1439	5117025	YPDT - Bedrock - Undifferentiated	242.7159	2.4384	4.572006
17	-2055958063	81774285	2801005	220.5259	214.4299	2801005	YPDT - Late Stage Glaciolacustrine-Glaciofluvial	226.6219	6.095993	12.192
18	-2041464216	-905245459	4510642	180.3776	179.768	4510642	Thorncliffe Formation	208.1144	27.7368	28.34641
19	-2019149641	2033232943	2803673	247.0854	240.9894	2803673	YPDT - Madokaw/Oak Ridges (MIS/ORAC)	249	1.914597	8.01059
20	-1999960968	1565746506	2800190	142.1944	138.232	2800190	YPDT - Bedrock - Undifferentiated	150	7.805603	11.76801
21	-1994462936	86871844	6806481	233.8484	233.5438	6806481	Newmarket Till/Northern till	255	21.1516	21.45641
22	-1992507379	-1308899032	5116395	261.8485	235.7405	5116395	YPDT - Bedrock - Undifferentiated	261.8485	0	25.908
23	-1933194578	-505284647	2802052	222.1675	218.5399	2802052	YPDT - Bedrock - Undifferentiated	240	17.8325	21.4901
24	-1929240483	1839083334	2804486	219.2169	219.2928	2804486	Halton Till	226.9128	6.095993	7.819996
25	-1963616591	1441402268	2911914	137.4889	166.9885	2911914	YPDT - Bedrock - Undifferentiated	143.5649	6.095993	36.576
26	-1963035580	1175489673	2802564	187.0859	182.8167	2802564	YPDT - Late Stage Glaciolacustrine-Glaciofluvial	192.2675	5.161595	9.448907
27	-1948083007	1182824075	4509379	116.4689	115.8602	4509379	YPDT - Madokaw/Oak Ridges (MIS/ORAC)	118.9082	2.4384	3.047997
28	-1947745172	-2071240254	2803956	152.0528	147.176	2803956	Thorncliffe Formation	175	22.9472	27.82401
29	-1932522378	-1951591833	5114904	221.7732	209.9715	5114904	YPDT - Bedrock - Undifferentiated	238.2324	16.4952	29.2608
30	-1928529796	854339677	5115135	196.1953	182.7441	5115135	YPDT - Late Stage Glaciolacustrine-Glaciofluvial	197.9841	1.828736	15.24001
31	-19263176166	1351948525	4506559	198.625	185.5186	4506559	YPDT - Inter Newmarket Sediment/Lower	198.9298	0.3047943	13.41119

Example 1b - Assembling Interval Data (Microsoft SQL Management Studio - Views)

The view V_General_Hydrogeology reproduces automatically some of the steps described in the previous example. We'll again use MSSQLMS but, this time, using the view instead.

1. Make sure that the partner database is selected then select the first 1000 records from the V_General_Hydrogeology view. Note that a query can be stopped at any time by selecting the red 'Cancel Executing Query' button'.
2. Select (highlight) the resultant SQL code and choose 'Query - Design Query in Editor'.
3. Deselect all fields/columns except for: 'Location ID', 'Screen (Int) ID', 'Loc Name', 'Screen (Int) Name', 'Ground Elevation (masl)', 'Screen Top (masl)', 'Screen Bottom

(masl)', 'Interpreted Formation (Screen Top)' and 'Interpreted Formation (Screen Bottom)'.

4. Add the D_BOREHOLE and R_GEOL_UNIT_CODE tables and link based upon the LOC_ID/'Location ID' and BH_SURFICIAL_UNIT_CODE/GEOL_UNIT_CODE (as described above). Include the GEOL_UNIT_DESCRIPTION (give it an 'Alias' of 'Surficial Unit') in the output.

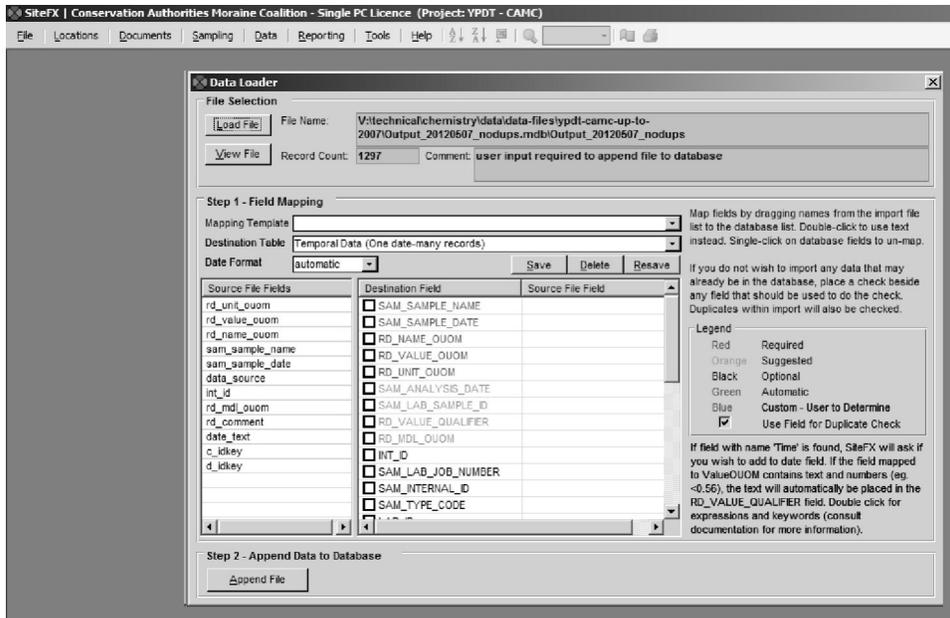
5. Include the calculations for depth (described in the previous example, step 6) but subtract 'Screen Top (masl)' from 'Ground Elevation (mASL)', give it an 'Alias' of 'Screen Top Depth'; subtract 'Screen Bottom (masl)' from 'Ground Elevation (mASL)', give it an 'Alias' of 'Screen Bottom Depth'. As the field names are somewhat long, select the second field (e.g. 'Screen Top...'), copy it, then select the first field (i.e. 'Ground Elevation ...'), add a minus (-) sign then paste the second field.

6. Exit out of 'Query Designer' and run/execute the query. The results from this query will mimic that of the previous example without having to load multiple tables - the view itself does that internally. However, the view is also carrying out some calculations behind the scenes such that it may actually run slower than one that is hand-built.

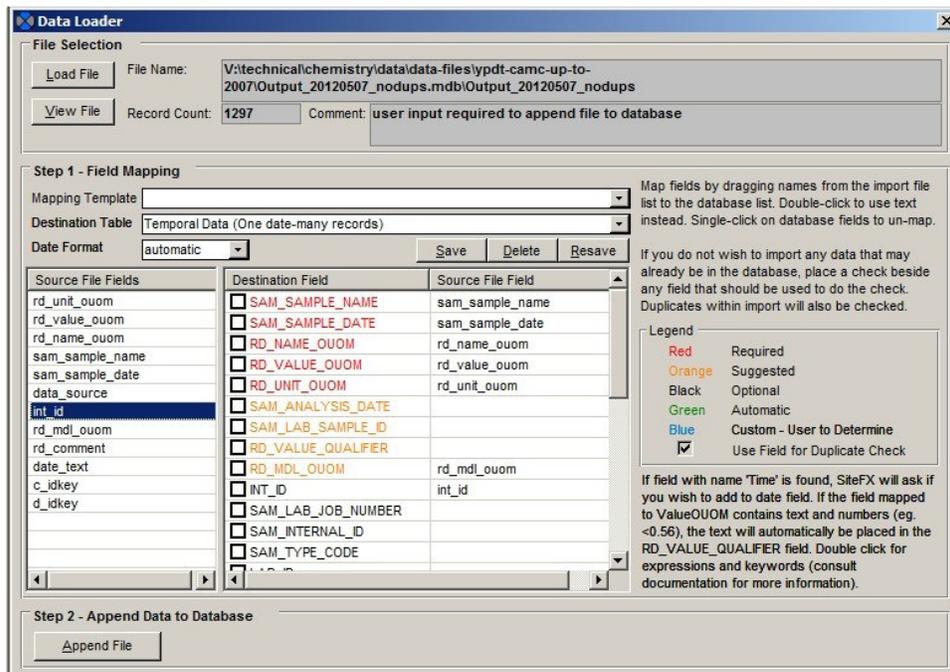
Example 2 - Importing Chemistry Data (SiteFX)

Refer to Section 2.3.4 for details regarding the formatting of laboratory/chemistry data for import into the ORMGP database. As two tables need to be populated (i.e. D_INTERVAL_TEMPORAL_1A/1B) with this information (the first with details regarding the sample - e.g. sample date, interval sampled against, laboratory details, etc ... - while the second will contain the actual results of the analysis), data import is best accomplished using the SiteFX tool (though it can be accomplished, albeit with slightly more difficulty, using only SQL commands).

When in SiteFX, the data import option is found under 'Data - Import Data - From Files' and selecting 'Load Data' from the resultant dialog box. The program then checks the file (in this case, an Access '.mdb' file; alternate file formats can be used) for data contents. Import columns/fields are listed as well as the number of rows present in the file itself.



The 'Destination Table' should be 'Temporal (one date, many records)'. When importing water levels, alternatively, the selection would then be 'D_INTERVAL_TEMPORAL_2 (one date, one record)'. The 'date' in both these cases actually corresponds to a date-time combination. The user would then 'map' the import columns/fields to the matching columns/fields within the database itself - SiteFX lists both the required and suggested fields that should be populated. In the example, the import fields have been named to match the names of the destination fields.



Selecting 'Append File' will load the information into the database - SiteFX returns a success or failure upon completion. In addition to populating the temporal tables, the program adds an entry into D_DATA_SOURCE that can be used for tracking - the sample itself and all sample analyses can be identified with the appropriate DATA_ID. However, SiteFX does not populate the DATA_DESCRIPTION field - this should be done by the user. The result is shown.

	DATA_ID	DATA_TYPE	DATA_DESCRIPTION	DATA_COMMENT	DATA_FILENAME	DATA_STATUS
1	-2147483386	NULL	YPDT-CAMC Chemistry	NULL	V:\technical\chemistry\data\data-files\ypdt-camc...	1297 of 1297 re
	DATA_STATUS	DATA_ADDED_METHOD	DATA_ADDED_USER	DATA_ADDED_DATE	DATA_RECORD_COUNT	
..	1297 of 1297 records appended.	SiteFX Data Loader	mikedoughty	2012-05-07 14:07:03.000	1297	

When temporal data is imported into the database through SiteFX, the program keeps a record of the transaction and its details. If at some point in the future it's determined that the information was erroneous, SiteFX can back-out the data (i.e. remove it from the database) with little effort. This would not be case to remove the information by hand.

Example 3 - Total Monthly Precipitation, Average Temperature and Water Level Data (Microsoft SQL Management Studio or Microsoft Access - SQL)

In order to simplify information the average of available results is taken. For time-step analysis (e.g. daily or monthly time-steps, as used in recharge analysis) this information needs to be averaged based upon the specified time interval (dependent upon data availability). This is accomplished through a conversion of the date-time field RD_DATE in D_INTERVAL_TEMPORAL_2 and a grouping of matching dates.

For water levels this would consist of an SQL statement such as

```

SELECT
Vgl.[Location ID],
Vgl.[Location Name],
Vgl.[Location Name Alt 1],
Vgl.[Original (MOE) Name],
Vgl.[Location Study],
i.INT_NAME AS [Screen Name],
wla.[Reading Date],
wla.[WL - Monthly Avg (masl)],
wla.[Record Count]
FRM
dbo.V_General_Locations AS Vgl
INNER JOIN
dbo.D_INTERVAL AS i ON i.LOC_ID=Vgl.[Location ID]
INNER JOIN
(
    SELECT
    i2.INT_ID
    ,CONVERT(varchar(7),i2.RD_DATE,121) AS [Reading Date]
    ,AVG(i2.RD_VALUE) AS [WL - Monthly Avg (masl)]
    ,COUNT(i2.RD_VALUE) AS [Record Count]
    FROM
    dbo.D_INTERVAL_TEMPORAL_2 as i2
    WHERE
    i2.RD_NAME_CODE IN (628,629)
    AND i2.UNIT_CODE=6
    GROUP BY
    i2.INT_ID,CONVERT(varchar(7),i2.RD_DATE,121)
) AS wla
ON i.INT_ID=wla.INT_ID

```

SECTION 1

SECTION 2

Where ‘Section 2’ is extracting and averaging the water level information based on the interval (i.e. the INT_ID) and ‘Section 1’ is extracting the associated information related to the interval itself (i.e. it’s location and interval name, etc ...).

For ‘Section 2’, we’re returning the interval identifier, the date (in the form ‘yyyy-mm’), the average water level (called ‘WL - Monthly Avg (masl)’) and the total number of values (a count; called ‘Reading Count’) used to calculate the average.

- CONVERT(varchar(7),i2.RD_DATE,121) is converting the date, pulled from the D_INTERVAL_TEMPORAL_2 table (which is being aliased/renamed as ‘i2’)
 - RD_DATE, from SQL Server, is returned (based upon the ‘121’ code) in the ‘yyyy-mm-dd hh:mm’ format; the use of the ‘varchar(7)’ type-code allows us to extract the first 7 characters from the date, reducing the result to ‘yyyy-mm’; making the reading a ‘monthly’ value
- GROUP BY i2.INT_ID,CONVERT(varchar(7),i2.RD_DATE,121)
 - As we’re trying to come up with monthly values for any particular interval, we subset the returned information based upon the INT_ID and the converted (now monthly) date
- AVG(i2.RD_VALUE)
 - The actual value as found in the temporal table; as we’re grouping the information (based upon INT_ID and the monthly date), this actually calculates the average of all values that correspond to a particular INT_ID and particular year-month; an ‘aggregate’ function
- COUNT(i2.RD_VALUE)

- The total number of records associated with the INT_ID and year-month (that we're grouping); an 'aggregate' function
- WHERE i2.RD_NAME_CODE IN (628,629) AND i2.UNIT_CODE=6
 - We're only interested in certain water levels, namely '628' (' Water Level - Manual - Static') and '629' (' Water Level - Logger (Compensated & Corrected)' - note that these values are found in R_RD_NAME_CODE
 - In addition, we only want to include those values that have been converted to 'masl' (corresponding to a 'UNIT_CODE' of '6'; as found in R_UNIT_CODE)
- AS wla
 - As the statement (i.e. 'Section 2') is bracketed, we need to 'alias' it so that it can be referred to in 'Section 1'

For 'Section 1', we're combining the results returned from 'Section 2' (referenced as 'wla') with information concerning the interval and location associated with it. This is accomplished through 'joining' 'Section 2' with that general location view (i.e. V_General_Locations; 'aliased' here as 'vgl'). Refer to Appendix A for details regarding the use of the 'SELECT' and 'INNER JOIN' SQL statements.

Example results (the first 10 returned - these will likely not be the same for any particular user) would then be

Location ID	Location Name	Location Name Alt 1	Original (MOE) Name	Location Study	Screen Name	Reading Date	W.L. Daily Avg (masl)	Record Count	
1	149708	Richmond Hill MMDW-PW East	MAJOR MAC DEWATERING WELL EAST	MMDW-PW East	York - Richmond Hill - Dewatering (manual W/L)	MMDW-PW East	2000-10	214.545837402344	1
2	149542	Newmarket MW 4 (TW 3/59)	York - Newmarket - MW #4 (Fogers Park - ROG) (E.G...	6300693	York - Newmarket - Monitoring	ROG	2003-08	210.603690969659	750
3	40980	PGMN - TRCA - W0000010	TRCA - Clarendon Deep MOE OW #333A (PGMN)	4606547	PGMN - TRCA (MOE - 1974 to 1980 OW Network)	PGMN - TRCA - W0000010	2002-06	177.070418781704	720
4	2005199501	6706177	KIRKPATRICK ROSS	6706177	NULL	6706177	1976-08	413.004028320313	1
5	129941	Queensville MW 2 (TW 8/98)	York - Queensville - MW #2 - TW 8-88	6920215	York - Queensville - Monitoring	Queensville MW 2	2006-10	209.96672692741	747
6	149707	Cawfield May HP	NULL	Cawfield May HP	York - manual W/L	Cawfield May HP	2005-02	306.279888778297	1
7	134331	Vandorf MW 2D	York - Vandorf - MW #2D (TW 3-89)	6825224	York - Vandorf - Monitoring	Vandorf MW 2D (3/98)	2004-04	251.722354198386	4321
8	134329	Vandorf MW 2i	York - Vandorf - MW #2i (TW 3-89)	6825222	York - Vandorf - Monitoring	Vandorf MW 2i (2/98)	2000-12	256.065623564655	745
9	149653	Queensville MW 12 (Old MW 5)	York - Queensville - MW #12 (MW #5)	MW15	York - Queensville - Monitoring	MW12D	2001-11	233.811001586914	5
10	149657	Richmond Hill MW 2	York - Richmond Hill - MW #2 (TW 1-75)	6913010	York - Richmond Hill - Monitoring (Oak Ridges)	OAK1-TW	1992-07	268.846801757813	2

For temperature, the RD_NAME_CODE used is '369' (' Temperature (Air)'; water temperatures use a different code) and the UNIT_CODE would then be '3' ('C'; degrees celsius). These would then be used to extract the RD_VALUE from D_INTERVAL_TEMPORAL_2. Otherwise the statement is similar.

```

SELECT
Vgl.[Location ID],
Vgl.[Location Name],
Vgl.[Location Name Alt 1],
Vgl.[Original (MOE) Name],
Vgl.[Location Study],
i.INT_NAME AS [Screen Name],
wla.[Reading Date],
wla.[Temp - Monthly Avg (degC)],
wla.[Record Count]
FROM
dbo.V_General_Locations AS Vgl
INNER JOIN
dbo.D_INTERVAL AS i ON i.LOC_ID=Vgl.[Location ID]
INNER JOIN
(
    SELECT
    i2.INT_ID
    ,CONVERT(varchar(7),i2.RD_DATE,121) AS [Reading Date]
    ,AVG(i2.RD_VALUE) AS [Temp - Monthly Avg (degC)]
    ,COUNT(i2.RD_VALUE) AS [Record Count]
    FROM
    dbo.D_INTERVAL_TEMPORAL_2 AS i2
    WHERE
    i2.RD_NAME_CODE IN (369)
    AND i2.UNIT_CODE=3
    GROUP BY
    i2.INT_ID,CONVERT(varchar(7),i2.RD_DATE,121)
) AS wla
ON i.INT_ID=wla.INT_ID

```

SECTION 1

SECTION 2

With the (example) results

!!!! THIS NEEDS TO BE FIXED - GW TEMPS !!!!!

	Location ID	Location Name	Location Name Alt 1	Original (MOE) Name	Location Study	Screen Name	Reading Date	Temp - Monthly Avg (mm)	Record Count
1	-91487520	Mount Albert MW 4 (MP-1)	York - Mount Albert - MW #4 (Mt. Albert Creek Pl...	MP-1 Piez	York - Mount Albert - Monitoring	MW #4 (Intacpiez)	2009-05	8.46805781318295	1488
2	-91487520	Mount Albert MW 4 (MP-1)	York - Mount Albert - MW #4 (Mt. Albert Creek Pl...	MP-1 Piez	York - Mount Albert - Monitoring	MW #4 (Intacpiez)	2009-02	3.04850444587923	1344
3	-91487520	Mount Albert MW 4 (MP-1)	York - Mount Albert - MW #4 (Mt. Albert Creek Pl...	MP-1 Piez	York - Mount Albert - Monitoring	MW #4 (Intacpiez)	2007-08	14.9891599391096	1488
4	-91487520	Mount Albert MW 4 (MP-1)	York - Mount Albert - MW #4 (Mt. Albert Creek Pl...	MP-1 Piez	York - Mount Albert - Monitoring	MW #4 (Intacpiez)	2007-02	3.8462872063475	1344
5	-91487520	Mount Albert MW 4 (MP-1)	York - Mount Albert - MW #4 (Mt. Albert Creek Pl...	MP-1 Piez	York - Mount Albert - Monitoring	MW #4 (Intacpiez)	2006-09	14.2670555538601	1440
6	-91487520	Mount Albert MW 4 (MP-1)	York - Mount Albert - MW #4 (Mt. Albert Creek Pl...	MP-1 Piez	York - Mount Albert - Monitoring	MW #4 (Intacpiez)	2006-06	11.4815069874128	1440
7	-91487520	Mount Albert MW 4 (MP-1)	York - Mount Albert - MW #4 (Mt. Albert Creek Pl...	MP-1 Piez	York - Mount Albert - Monitoring	MW #4 (Intacpiez)	2007-12	6.2215906098385	1490
8	-91487520	Mount Albert MW 4 (MP-1)	York - Mount Albert - MW #4 (Mt. Albert Creek Pl...	MP-1 Piez	York - Mount Albert - Monitoring	MW #4 (Intacpiez)	2006-05	8.23670026499738	1488
9	-91487520	Mount Albert MW 4 (MP-1)	York - Mount Albert - MW #4 (Mt. Albert Creek Pl...	MP-1 Piez	York - Mount Albert - Monitoring	MW #4 (Intacpiez)	2006-08	15.4405107933968	1488
10	-91487520	Mount Albert MW 4 (MP-1)	York - Mount Albert - MW #4 (Mt. Albert Creek Pl...	MP-1 Piez	York - Mount Albert - Monitoring	MW #4 (Intacpiez)	2006-02	4.26730794785575	1497
11	-91487520	Mount Albert MW 4 (MP-1)	York - Mount Albert - MW #4 (Mt. Albert Creek Pl...	MP-1 Piez	York - Mount Albert - Monitoring	MW #4 (Intacpiez)	2008-01	5.10459005768581	1488

Precipitation, the last parameter we're determining monthly values for, uses an RD_NAME_CODE of '551' ('Precipitation - Day') with a UNIT_CODE of '21' ('mm'). Instead of determining the average, though, here we're looking at the total precipitation for each month. This would then be of the form

```

SELECT
Vgl.[Location ID],
Vgl.[Location Name],
Vgl.[Location Name Alt 1],
Vgl.[Original (MOE) Name],
Vgl.[Location Study],
i.INT_NAME AS [Screen Name],
wla.[Reading Date],
wla.[Precip - Monthly Total (mm)],
wla.[Record Count]
FROM
dbo.V_General_Locations AS Vgl
INNER JOIN
dbo.D_INTERVAL AS i ON i.LOC_ID=Vgl.[Location ID]
INNER JOIN
(
    SELECT
    i2.INT_ID
    ,CONVERT(varchar(7),i2.RD_DATE,121) AS [Reading Date]
    ,SUM(i2.RD_VALUE) AS [Precip - Monthly Total (mm)]
    ,COUNT(i2.RD_VALUE) AS [Record Count]
    FROM
    dbo.D_INTERVAL_TEMPORAL_2 AS i2
    WHERE
    i2.RD_NAME_CODE IN (551)
    AND i2.UNIT_CODE=21
    GROUP BY
    i2.INT_ID,CONVERT(varchar(7),i2.RD_DATE,121)
) AS wla
ON i.INT_ID=wla.INT_ID

```

SECTION 1

SECTION 2

where you can see the AVG keyword has been modified to SUM. This allows a total to be calculated of all the grouped RD_VALUE's. Example results would then be

!!!! THIS NEEDS TO BE LOOKED AT - DATA LOADED TWICE? !!!!!

	Location ID	Location Name	Location Name Alt 1	Original (MOE) Name	Location Study	Screen Name	Reading Date	Precip - Monthly Total (mm)	Record Count
1	-865791875	6168100	SONYA SUNDANCE MEADOWS	6168100	Env Canada - Climate Station (2002)	6168100	2002-11	77.4000003486872	30
2	-865791875	6168100	SONYA SUNDANCE MEADOWS	6168100	Env Canada - Climate Station (2002)	6168100	2005-12	194.400000602007	62
3	-865791875	6168100	SONYA SUNDANCE MEADOWS	6168100	Env Canada - Climate Station (2002)	6168100	2004-08	50.0000002086163	31
4	-865791875	6168100	SONYA SUNDANCE MEADOWS	6168100	Env Canada - Climate Station (2002)	6168100	2006-12	127.600000113249	62
5	-865791875	6168100	SONYA SUNDANCE MEADOWS	6168100	Env Canada - Climate Station (2002)	6168100	2005-03	52.3999998967512	62
6	-865791875	6168100	SONYA SUNDANCE MEADOWS	6168100	Env Canada - Climate Station (2002)	6168100	2004-03	66.4000008404255	31
7	-865791875	6168100	SONYA SUNDANCE MEADOWS	6168100	Env Canada - Climate Station (2002)	6168100	2003-11	241.599997550249	60
8	-865791875	6168100	SONYA SUNDANCE MEADOWS	6168100	Env Canada - Climate Station (2002)	6168100	2003-12	62.6000010967255	31
9	-865791875	6168100	SONYA SUNDANCE MEADOWS	6168100	Env Canada - Climate Station (2002)	6168100	2001-09	87.2000014334917	30
10	-865791875	6168100	SONYA SUNDANCE MEADOWS	6168100	Env Canada - Climate Station (2002)	6168100	2003-01	68.4000006020069	31
11	-865791875	6168100	SONYA SUNDANCE MEADOWS	6168100	Env Canada - Climate Station (2002)	6168100	2001-08	43.6000006049871	31

Section J.3 Training Exercises (Difficult)

Example 1 - Determining Intervals Below Newmarket Till (Microsoft SQL Management Studio or Microsoft Access - SQL)

```

select
  dloc.LOC_ID
  ,dloc.LOC_NAME
  ,dloc.LOC_COORD_EASTING
  ,dloc.LOC_COORD_NORTHING
  ,dmon.MON_TOP_ELEV
  ,dmon.MON_BOT_ELEV
  ,dmon.MON_TOP_DEPTH_M
  ,dmon.MON_BOT_DEPTH_M
  ,delev.ASSIGNED_ELEV
from
  [OAK_20120615_YORK].dbo.d_location as dloc
inner join
  [OAK_20120615_YORK].dbo.D_LOCATION_QA as dlqa
on
  dloc.LOC_ID=dlqa.LOC_ID
inner join
  [OAK_20120615_YORK].dbo.D_INTERVAL as dint
on
  dloc.LOC_ID=dint.LOC_ID
inner join
  [OAK_20120615_YORK].dbo.D_INTERVAL_MONITOR as dmon
on
  dint.INT_ID=dmon.INT_ID
inner join
  [OAK_20120615_YORK].dbo.D_LOCATION_ELEV as delev
on
  dloc.LOC_ID=delev.LOC_ID
where
  dlqa.QA_COORD_CONFIDENCE_CODE<10 -- note that we're using possibly erroneous locations here
and dloc.loc_id
in
(

```

SECTION 1

```

select
  distinct(loc_id)
from
  [OAK_20120615_YORK].dbo.D_INTERVAL
where
  int_id
in
(

```

SECTION 2

```

select
  INT_ID
from
  [OAK_20120615_YORK].dbo.D_INTERVAL_FORMATION_ASSIGNMENT
where
  ASSIGNED_UNIT is not null
and ASSIGNED_UNIT in ( 5, -- YPDT - Inter Newmarket
                       6, -- YPDT - Lower Newmarket
                       7, -- YPDT - Bedrock Undiff
                       8, -- YPDT - Bedrock Shale
                       9, -- YPDT - Bedrock Limestone
                       10, -- YPDT - Channel - Clay
                       11, -- YPDT - Channel - Silt
                       12, -- YPDT - Channel - Sand
                       13, -- YPDT - Channel - Gravel
                       56, -- York Till
                       58, -- Don Fm
                       59, -- Scarborough Fm
                       61, -- Sunnybrook Till
                       63, -- Sunnybrook Drift
                       65, -- Thorncliffe Fm
                       66, -- Lower Thorncliffe Fm
                       68, -- Middle Thorncliffe Fm
                       70, -- Upper Thorncliffe Fm
                       150, -- Lower Sediment Upper Unit
                       151 -- Lower Sediment Lower Unit

```

SECTION 3

```
)
```

This example comprises the extraction of location- and interval-specific information for those intervals that are located below the Newmarket Till. The SQL statement (above)

has been divided into three sections, each performing a particular task (note that the Section 2 and 3 are bracketed). When creating a query accomplishing a complicated task, it is better to work in simple steps. The order in which each section is run is actually the reverse of what would be expected: Section 3 is run first the results of which are fed into Section 2; Section 2 is run second with the results feeding Section 1; Section 1 is run last.

Section 3

Notice that only a single field is being returned from this section, namely the INT_ID. This is the case when subqueries are present within a query - only a limited (in some cases only one) number of columns should be returned. The query here is looking for intervals (to be identified by the INT_ID) which are found below the Newmarket Till. This is accomplished by comparing the ASSIGNED_UNIT code against a list of codes corresponding to stratigraphic units found below the Newmarket Till (refer to R_GEOL_UNIT_CODE for additional code-description information). As an aid, the unit description is included as part of the query as comments (i.e. the text occurs after a double-dash, '--'). Note also that we're not including any unassigned intervals - we first check that ASSIGNED_UNIT is not null before comparing it against the unit codes.

Section 2

Only a single field is being returned from this section. In this case, the LOC_ID which is going to be the base against which will extract the information necessary for analysis. As more than one interval can be assigned to each location, we need to remove duplicated LOC_ID's found when converting the INT_ID's (returned from Section 3) the LOC_ID's in this section. The DISTINCT keyword does that, only returning one instance of any LOC_ID.

Section 1

The final section now determines and returns the information based on the LOC_ID's from Section 2 of the query. Here we're pulling the coordinates (LOC_COORD_EASTING, LOC_COORD_NORTHING; by system default, these are UTMZ17 and NAD83 coordinates), the name, the assigned elevation (ASSIGNED_ELEV) and the top and bottom elevation and depths of the borehole screen. Note that these fields are from multiple tables, linked by the LOC_ID (returned from Section 2). There are a number of JOIN's made to link the necessary information together.

One additional note on this section, we're currently using the QA coordinate code to delimit our returned results to an error of +/- 10km. This should be changed to reflect the accuracy of the analysis (usually a QA code of less than 6; refer to R_QA_COORD_CONFIDENCE_CODE for details).

GIS Analysis (Non-Specific) - Introduction

The results of this query can then be fed into a GIS for further analysis. Here we're interested in determining those wells (i.e. locations) that

- Fall within specified polygons (e.g. WHPA's)
- The distance and depth to production wells. (for those that do fall within the specified polygons)

Note that this analysis is non-specific for a GIS, only describing in general terms the means by which to perform any particular step (or reach a certain endpoint). If the user is unfamiliar with the GIS package they are using, a keyword search on the topics (or analysis type) described, following, would get them started on the methodology required for their particular package. Expert GIS users can safely ignore the following outlines.

GIS Analysis - Points And Areas

The defined polygons (called WHPA_Poly), the municipal wells (referred to as BHS_MUNIC) and the boreholes found to be below the Northern Till (referred to as BHS_BELOW_NT) need to be incorporated in the GIS package. The latter projection has been previously described - make sure that the projections of each layer match.

GIS Analysis - Topology Overlay

The user needs to determine a 'contained-within' relationship between BHS_BELOW_NT and WHPA. This is commonly referred to as a 'Topology Overlay' where attributes can be (or are) copied between overlapping objects (or are otherwise tagged) with the meaning that they encompass some (or all) of the same spatial area. Once it has been determined which points in BHS_BELOW_NT are found within a WHPA_Poly, the remainder can be removed.

Note that this operation can also be accomplished through a 'Select Touching' or (more correctly) 'Select Contained Within' operation. The operation may also be known as an 'Identity Overlay'.

GIS Analysis - Nearest Neighbour

Using the BHS_BELOW_NT and BHS_MUNIC layers, the user can carry out a 'Nearest Neighbour' analysis which determines which municipal well is closest to the boreholes below the Northern Till. In general, a straight line segment is drawn between the two locations and tagged with an (below NT) identifier and a distance - alternatively, the distance and (municipal) identifier can be added directly to the BHS_BELOW_NT table.

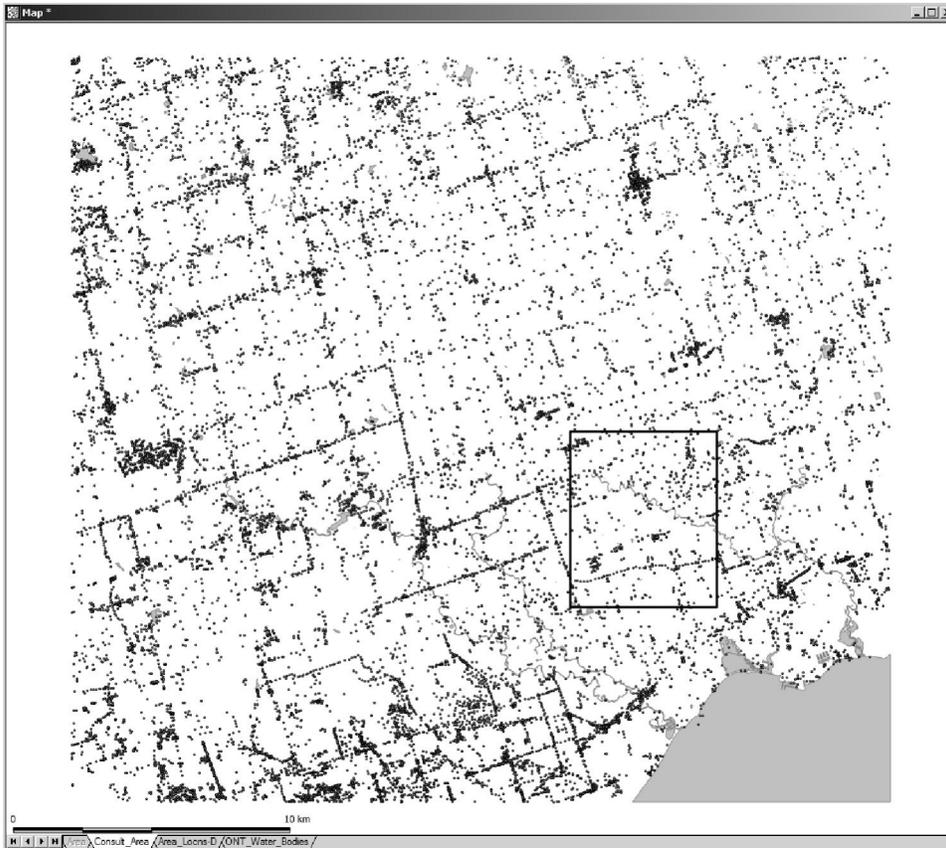
GIS_Analysis - Depths

Once there is a relationship between the BHS_BELOW_NT and BHS_MUNIC layers, a straight subtraction can occur between the top screen elevations for both boreholes. This calculates the vertical offsets between the screens.

If a new layer was created in the 'Nearest Neighbour' analysis, the objects in that layer need to be associated with both the BHS_BELOW_NT and BHS_MUNIC layers before calculation (as we'll need the elevations from both). This may be accomplished through a 'Spatial Overlay', where attributes are transferred between each other based upon their spatial location - in some cases, the fields need to be present in both tables before the analysis.

Example 2 - Extracting Consultant Information By Area (Microsoft SQL Management Studio with Spatial SQL or GIS)

There are various methods by which to extract information from a designated area. This generally arises when the user wishes to delimit the information provided to a third-party through specification of a spatial boundary. Given the following example area



where the points indicate locations found within the ORMGP database, how do you determine what locations are found within the black boundary-polygon/project-area. As the boundary is rectangular, the determination of what locations lie within is fairly straightforward. Provided with (or determining) that the minimum-maximum coordinates are:

- Minimum Easting - 629481

- Maximum Easting - 659107
- Minimum Northing - 4848108
- Maximum Northing - 4875111

An SQL query to extract the LOC_ID's that fall within the area (as a first step before extracting the remainder of the desired information) would be

```
SELECT
LOC_ID
FROM
D_LOCATION
WHERE
LOC_COORD_EASTING BETWEEN 629481 AND 659107
AND LOC_COORD_NORTHING BETWEEN 4848108 AND 4875111
```

However, if the boundary was not rectangular, this query would not be appropriate. Instead, for example, the user would use an external package (for example, Viewlog or a GIS package - ArcMap, etc ...) to plot the locations from D_LOCATION (using the LOC_COORD_EASTING and LOC_COORD_NORTHING fields). Then, using the boundary-polygon, a 'Contained Within' or other similar 'Overlay' analysis would be run, extracting the LOC_ID's that are found within the boundary. These could then be 'marked' - that is, a value placed in either of the SYS_TEMP1 or SYS_TEMP2 fields in D_LOCATION indicating that it lies within the boundary-polygon. If the value placed in the SYS_TEMP2 field was '20121030', then the query to extract the location identifier would be

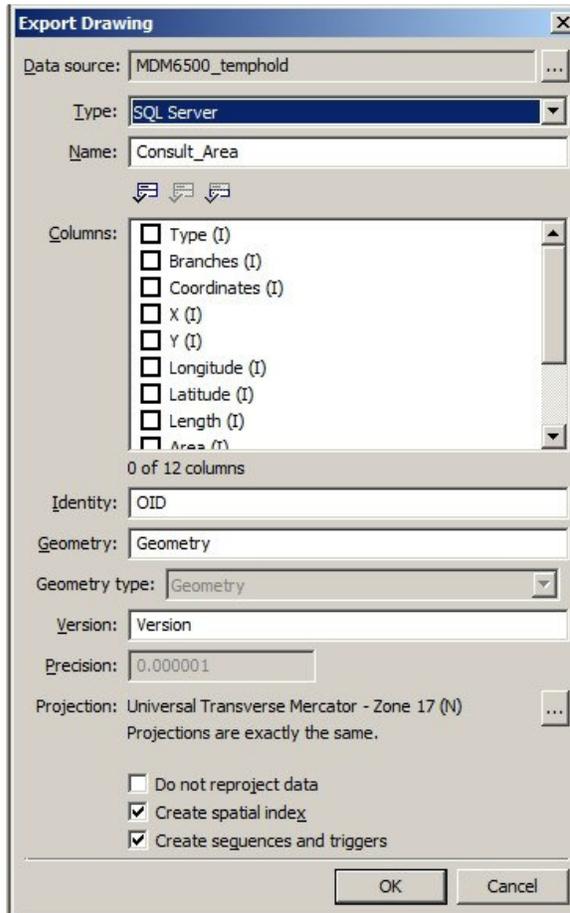
```
SELECT
LOC_ID
FROM
D_LOCATION
WHERE
SYS_TEMP2=20121020
```

This would duplicate the results from the previous query (i.e. using, directly, the coordinates of the boundary-polygon).

An alternative method would allow the user to query the SQL Server (i.e. the ORMGP) database directly. This requires loading the boundary-polygon directly into a supplementary SQL Server database - such a database holds temporary tables, views or other objects that the user requires to be compared/run-against the ORMGP database. This avoids inclusion of any information in the ORMGP database that is not considered 'official' (refer to Appendix F for examples of accessory/supplementary databases and their objects currently in use at the ORMGP).

Here is an example of loading the boundary-polygon ('Consult_Area') into a temporary database ('MDM6500_temphold') using Manifold GIS. (Most GIS packages should be

able to load spatial information into SQL Server. See Nielson (2008) for a free non-GIS software which will also load spatial files into SQL Server.)

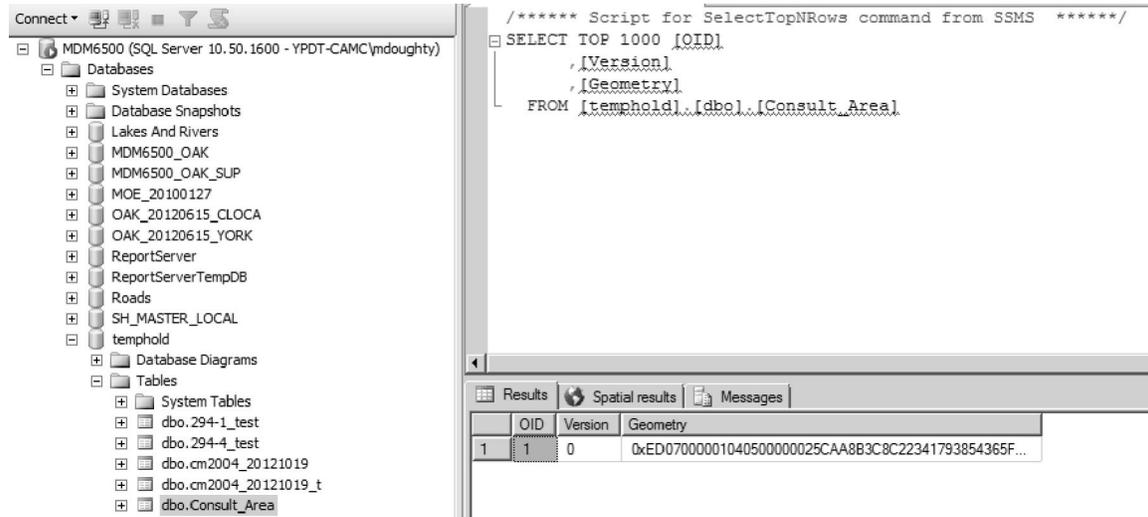


Notice that

- The 'Data source' is 'MDM6500_temphold'; this is the name of the database into which we're exporting the polygon
- The 'Type' is designated as 'SQL Server'; this software has the capability of working with spatial objects and has had its SQL syntax expanded to include spatial operations
- 'Identity' is a new (created) field/column to be used as a primary key (here, the default name is 'OID')
- 'Geometry' allows specification of 'Geometry' or 'Geographic'; the former is for Cartesian coordinates (e.g. UTM coordinates) while the latter is for Latitude-Longitude coordinates.
- A spatial index (checked under 'Create spatial index') must be created to allow SQL Server to interact with the object
- The projection must match that of a supported projection in SQL Server; these are identified by the European Petroleum Survey Group (EPSG) standard (now the OGP Geomatics Committee); in our case, we should be using code '2029' which matches to 'Universal Transverse Mercator - Zone 17 (N), WGS84' (note that the

WGS84 datum corresponds to the NAD83 datum used by the ORMGP database, with minor differences)

This is directly loaded into the 'MDM6500_temphold' database and looks, to all appearances, as a regular table.



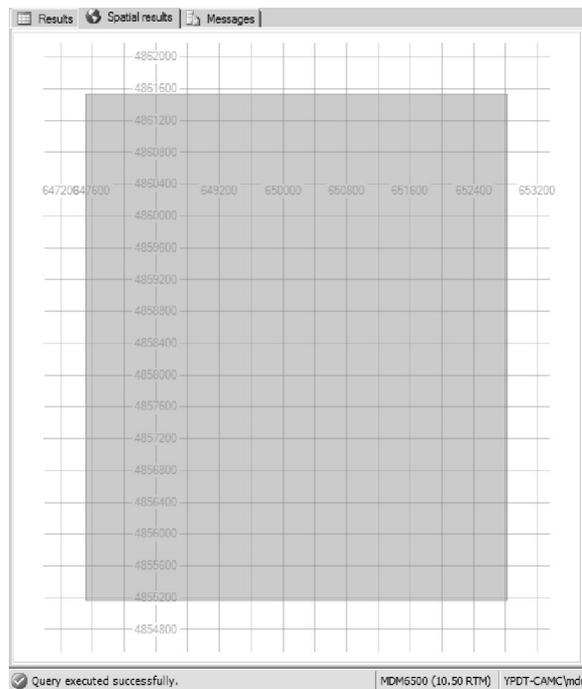
The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Server Enterprise Explorer' tree is expanded to show the 'MDM6500' server, then 'Databases', and finally 'tempohold' > 'Tables'. The 'dbo.Consult_Area' table is selected. The main window displays a query window with the following SQL script:

```
/****** Script for SelectTopNRows command from SSMS *****/  
SELECT TOP 1000 [OID],  
               [Version],  
               [Geometry]  
FROM [tempohold].[dbo].[Consult_Area]
```

Below the query window, the 'Results' tab is active, showing a grid with the following data:

	OID	Version	Geometry
1	1	0	0xED07000001040500000025CAA8B3C8C22341793854365F...

Notice the presence of a 'Geometry' column - this contains the spatial information attributed to the original boundary-polygon, now part of the 'Consult_Area' table/object. If we examine the 'Spatial results' tab, we can see a representation of the object.



A grid has also been plotted showing the coordinates (which should match those described earlier). A limited capability for zooming/navigating across a spatial dataset is provided (though it's not useful or necessary in this instance).

In order to compare the locations from D_LOCATION against that of 'Consult_Area' (i.e. the polygon), we need to convert the points (i.e. the coordinates) to a 'Geometry' type then run an 'Intersect' or 'Contains' operation. This is accomplished in a single step using the following SQL statement

```
select
coords.LOC_ID
from
(
    select
    dloc.LOC_ID
    ,geometry::STGeomFromText('POINT(' +
        CAST(dloc.loc_coord_easting as varchar(20)) +
        ' ' +
        CAST(dloc.loc_coord_northing as varchar(20)) +
        '),2029) as [geom]
    from
    OAK_20120615_MASTER.dbo.D_LOCATION as dloc
    where
    dloc.LOC_COORD_EASTING is not null
) as coords,[tempfold].dbo.Consult_Area as ca
where
coords.geom.STIntersects(ca.Geometry)=1
```

SECTION 1

SECTION 2

Section 2

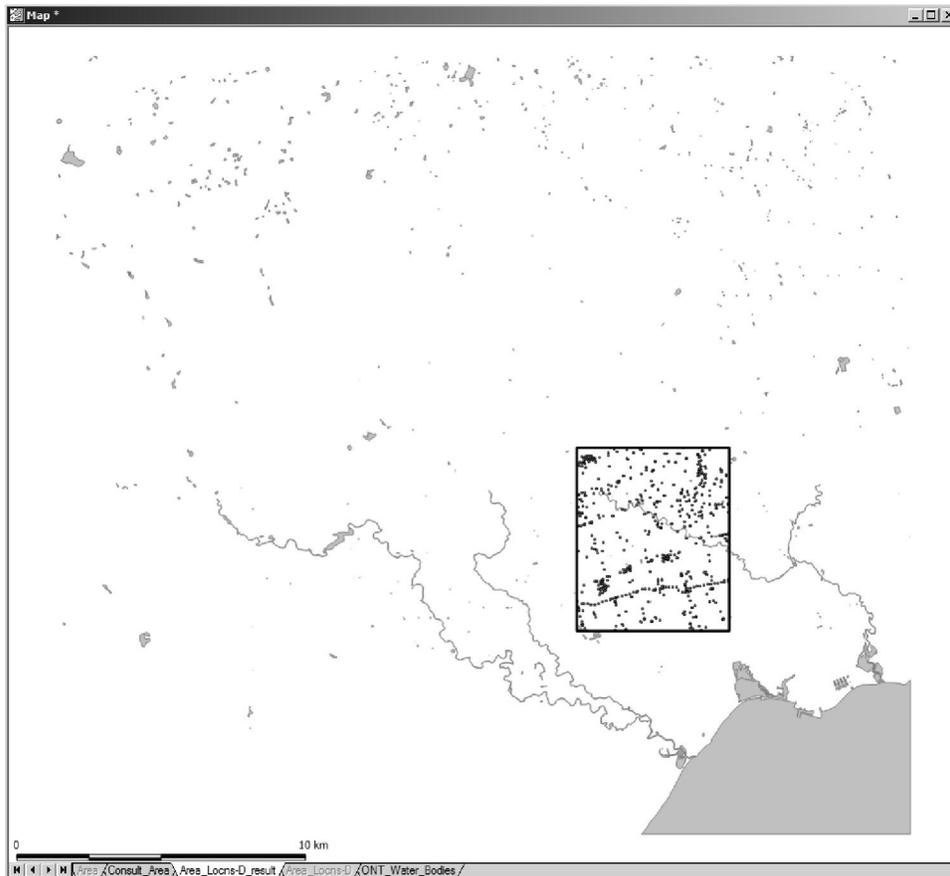
This section is copying the LOC_ID, LOC_COORD_EASTING and LOC_COORD_NORTHING values for each location in D_LOCATION and converting the latter two to a 'Geometry' object - a spatial object against which we'll be comparing the 'Consult_Area' polygon. The 'geometry::STGeomFromText' keyword is a built-in function of SQL Server ; this does the actual conversion. It requires a text string as input; that is being set using the 'CAST' statements (along with the '+' which, in this instance, is concatenating the individual text strings together into a single text string; note that there is a single ' ' shown - this is a space surrounded by single brackets). The output from this section would be a 'Geometry' object linked to a LOC_ID.

Section 1

This section does the actual comparison using the 'coords.geom.STIntersects' operation. This returns a 'True' (or 1) value when the geometries of the two objects (i.e. 'coords.geom' and 'ca.Geometry') cross (or intersect). When this value is 'True', we're returning the LOC_ID. The output from this script, then, is a single column containing the overlapping LOC_ID's to 'Consult_Area'.

	LOC_ID
1	-2128466099
2	-2112569424
3	-2106542756
4	-2106141606
5	-2100774601
6	-2092427643
7	-2086990595
8	-2080891040
9	-2077291638
10	-2062974353
11	-2055651595
12	-2043924525
13	-2036639488

This could then be used as input into another query pulling, for example, all location or borehole information corresponding to the 'Consult_Area' polygon. The selected locations are shown below (in comparison to the first location distribution figure).



Appendix K – Alternate Software Instructions

This section has not been included (this page is a place-holder only).

Appendix L – Database Reports