***Make sure you're looking at the correct SQL Server IC database in SSMS!***

What are USR\_ tables?

What are T\_ and t\_ tables?

**dbo.Lithostrat**

Not null columns:

well\_id int,

data\_type int,

top\_depth float,

symbol\_id int,

pk\_index PK int,

Others to investigate:

top\_boundary, base\_boundary (refers to dbo.BOUNDARIES, e.g. bound\_id 0 is "No boundary".

f\_interpid, interpretercreator, created (refers to dbo.T\_INTERPRETERS)

well\_id data\_type top\_depth symbol\_id pk\_index top\_age base\_age f\_interpid creator modifier abr source attr top\_boundary base\_depth base\_boundary legend interpreter created modified obsno mindepth maxdepth remark geofeature dipangle dipazimuth age owconf owqual owkind owbaseconf owbasequal owbasekind

Will dbo.INTERVALCOLUMNS need to be populated for each interval data type created, defining display styles etc?

What about dbo.tablenames

**dbo.PROJECTS**

pk\_index project\_id title client jobno code notes Units Map datum utmzone TVD\_datum OWTranslation f\_fieldname defchronodatatype deftstprops deftwtdata deffaultsdatatype defchronointerpid WellGroupField WellGroupFieldIsUserDefined WellOrderField WellOrderFieldIsUserDefined WellPatternTable WellPatternTableLayerField WellPatternTablePolygonField RPMWellTypeField DefaultRPMTemplates DefaultSummaryCharts DefaultWellstickTemplates f\_dynamic f\_WellQueryId

pk\_index int e.g. 1,

project\_id int e.g. 113,

WellGroupFieldIsUserDefined bit,

WellOrderFieldsIsUserDefined bit,

f\_dynamic bit,

f\_WellQueryId int.

Not necessary in database, but IC requires you to enter a project "Title".

Auto selects "Type: Static", "display units: metres", "Datum: MSL", "Geodatum: ED50", "Grid system: ED50 / UTM zone 30N",

Display tab "Group wells by: [No Grouping]", "Order wells by: Name"

**dbo.PROJECTWELLS** (which wells in which projects?)

pk\_index 11 PK int not null,

well\_id FK 2 int not null,

project\_id 2 FK int not null.

Needs to link with dbo.WELLS and dbo.PROJECTS?

**dbo.WELLS**

pk\_index, well\_id, units, created, creator, modified, modifier, project, rte, sea\_bed, rig\_elevation, datum, terminal\_depth, spud\_date, completion\_date, quadrant, sub\_block, kelly, symbol\_id, client, utmzone, code, name, field, location, country, basin, name1, name2, strat\_schemes, grnd\_elev, f\_block, grid\_x, grid\_y, latitude, longtitude, geodatum, facility, discovery\_name, seismic\_line, intent, f\_ipid, f\_licenceNumber, f\_api, f\_comment, f\_province, f\_county, f\_state, f\_section, f\_Township, f\_range, f\_uwi

**dbo.tablenames**

pk\_index 1 (note only pk\_index is not null in this table)

tabletype (can be I, P, D...)

tablename DATA\_Chronostrat

tabledescription Chronostratigraphy

hascores 0

fileprefix - can be I, A, W, L (ALSO D, B, M?)

interp 2 (why 2??)

pseudonames (can be blank, Log data, Facies, Lithostratigraphy etc)

samplebased 0

f\_tableid (can be 5, 7, 9, 10, unique for each)

**Reference other tables?**

tabletype not referenced elsewhere...

fileprefix doesn't seem to be used elsewhere (most often its the first letter of the tabledescription after DATA\_ but not always)

interp,

f\_tableid is referenced from elsewhere, but this seems to be the original.

**Creating new data data tables (USR\_)**

Note that when I create a new TestDataTable (and TestDataType\_int and TestDataType\_point) there are created under USR\_TestDataTable and

There are seeminly old data tables in the sql database that don't exist in the data manager anymore (e.g. USR\_Petronas\_Text, USR\_Petronas\_Data)

What differentiates 'manage individually' from 'manage as collection'?

"Tops" (DATA\_Tops) is managed individually.

"Petrographic Int Txt" (USR\_Petrographic\_Int001) is managed as a collection.

Answer: dbo.tablenames.samplebased is 1 for collection or 0 for individual.

**dbo.INTERVALCOLUMNS**

Appears to link data\_type to descrption/name (e.g. 112 is Formation)

Use to link data to table number (e.g. data\_type column of DATA\_Lithostrat table)

Note that this appears to be the source of the "style" column (I, H, S, C, etc.) which is used to DATA\_Lithostrat, etc. ***Search for I, H, S values elsewhere!***

pk\_index, data\_type, dest\_table, description, title, graphic, zonal, boundaries, legend, dic\_driven, col\_width, back\_colour, brush\_style, hatch\_style, int\_colour, row\_height, table\_header, horiz\_justify, vert\_justify, orientation, fontname, fontsize, fontweight, fontitalic, fontunderline, fontcolour, plotwith, style, maximum, minimum, units, horiz\_grid, chartstyle, plotsymbol, labelpoints, mergeevents, eventalign, sbugs\_igdtype, sbugs\_igdplotpos, allowoverlap, alttable, altfield, plotassequence, owinterp, ordering, wellcore, surface, f\_order, isdepthage, f\_style, f\_coreshiftid

**Tabulate for re-use:**

SELECT data\_type, description, dest\_table, graphic, zonal, boundaries, legend, dic-driven FROM INTERVALCOLUMNS

SELECT tabletype, tablename, tabledescription, f\_tableid FROM tablenames

**COMBINE INTO ONE SQL QUERY**

**Saved to** C:\Users\LBHAWZ\OneDrive - LR\Documents\SQL Server Management Studio\IC\_Combine tablenames and INTERVALCOLUMNS.sql

SELECT

ta.tabletype, ta.tablename, ta.tabledescription, ta.f\_tableid,

ty.dest\_table, ty.title, ty.data\_type, ty.graphic, ty.zonal, ty.boundaries, ty.legend, ty.dic\_driven

FROM dbo.tablenames AS ta

JOIN dbo.INTERVALCOLUMNS AS ty

ON ta.f\_tableid = ty.dest\_table

WHERE ta.tabledescription = 'Lithostratigraphy'

ORDER BY ta.tablename, ty.title;

-- Trying to find source of I, H, S, etc.

-- Note above results don't return "style" other than "I".

-- All distinct entries under "style": I, H, S, C, E, SC, SA, IMAGEFILE, ENGDATA

-- E corresponds to Data Type style 'Point - Occurrence Event' (though the SP doesn't return anything for occurrence - where is this stuff?)

-- H corresponds to point-value

--

SP "SearchAllTables" tells me that ENGDATA exists in:

INTERVALCOLUMNS.style

LASTVALUES.parm (column value is 'INT TAB LAST WIDTH DATA\_ENGDATA')

tablenames.pseudonames (column value is 'DATA\_EngData')

tablenames.tablename (column value is 'DATA\_EngData')

What tells a data type which IC data table it is in? dest\_table, e.g. 100? Can't find dest\_table column in any table other than INTERVALCOLUMNS. I think **f\_tableid = dest\_table** (means destination table??).

When writing any data (e.g. to DATA\_Lithostrat), need to check data\_type against INTERVALCOLUMNS.

To create a new Data Type in IC, I'd need to ensure the Data Table exists in dbo.tablenames and then populate dbo.INTERVALCOLUMNS accordingly.

Note that pick Data Types (e.g. Tops) are also stored in INTERVALCOLUMNS but are tagged with zonal: 0 instead of 1.

And Legend is 1 when a text string should be entered (like for Formation) but 0 for values like point-value.

**dbo.INTERVALDATA - weird one!**

This table seems to be the odd one out!

Contains data all tagged data\_type 5582 and 5583, which I can't find in INTERVALCOLUMNS.  Legends appear as 'J110.1 - J100' etc.

The IC Data Manager shows a Data Table named 'Stratigraphic Interp, with the DB Table: INTERVAL DATA (manage individually).

Within it, only a table called Source Rock is populated.

Also, this query shows a dictionary (dict\_id 176) called "Source intervals" refers to data\_type 5583:

SELECT \* FROM [IC DEMODB\_2018Dec\_SQL].[dbo].[SYMBOLDICTIONARY] WHERE data\_type LIKE '%5583%'.

How did the user manage to create a new table without the DATA\_ prefix, and why was it named INTERVALDATA? Very confusing!

**dbo.Tops?**

**dbo.DATA\_Lithostrat**

well\_id data\_type top\_depth symbol\_id pk\_index top\_age base\_age owconf owqual owkind owbaseconf owbasequal owbasekind f\_interpid creator modifier abr source attr top\_boundary base\_depth base\_boundary legend interpreter created modified obsno mindepth maxdepth remark geofeature dipangle dipazimuth age

\*\*\*I need to know which well\_id, data\_type, symbol\_id, f\_interpid to use\*\*\*

Strat file has well\_id, so thats ok.

data\_type comes from dbo.INTERVALCOLUMNS.

symbol\_id comes from?

f\_interpid comes from?

All not null columns: well\_id, data\_type, top\_depth, symbol\_id, pk\_index.

Odd that base\_depth and legend allow nulls.

well\_id needs to refer to dbo.WELLS.

data\_type signals Group (110), Formation and Members?

Auto-populated from Data Browser:

data\_type signals Group (110), Formation and Members?

symbol\_id 0

f\_interpid 0

creator 1

modifier 1

source Table Edit (what is import?)

attr {"ZoneColour":-1,"ZoneColourIsIpAuto":true,"EventSymbolId":0,"IsLocked":false,"OriginalZoneIndex":0}

top\_boundary 0

created

modified

obsno 0

mindepth 0

maxdepth 0

dip angle 0

dip azimuth 0

age 0

Where is "extended geological info" stored (Geologic Feature, Remark)

**Columns in NPD Stratigraphy > Wellbores data file:**

Wellbore name

Top depth [m]

Bottom depth [m]

Lithostrat. unit

Level (use to link with data\_type)

NPDID lithostrat. unit (USE THIS!)

Wellbore completion date - IGNORE

NPDID wellbore - IGNORE

Date updated - IGNORE

Date sync NPD - IGNORE

***Note that source column in DATA\_Lithostrat is nvarchar(255)  - change seismic\_line from 64 to 255?***

**Core data**

"Data Table: Core - Conventional, Data Type: Core Interval" is not suitable for "NPD Stratigraphy > Cores data" as it doesn't have depths.

Could use "Core length [m]" as "Recovered (%)", then get Top and Base Depths from the Lithostrat table, but not sure this represents recovered core.

Wellbore name

Core length [m]

Lithostrat. unit

Level

Wellbore completion date - IGNORE

NPDID lithostrat. unit - IGNORE

NPDID wellbore - IGNORE

Date sync NPD - IGNORE

**Better to use Wellbore > Table View > With > Cores.**

**Columns:**

Wellbore 1/2-1

Core sample number 1

Core sample - top depth 10208 ft (some ft, some m)

Core sample -  bottom depth 10208.4 ft

Core sample depth - uom (unit of measure, use to guide conversion)

Total core sample length [m] 56.2 (ALL IN METRES?)

Number of cores samples 8 - IGNORE

Cores available for sampling? YES - IGNORE

NPDID wellbore - IGNORE

Date updated - IGNORE

Date sync NPD - IGNORE

**dbo.DATA\_Core**

Same as dbo.DATA\_Lithostrat except for:

data\_type 5280

abr 34 (stores "Recovered (%)")

legend 1 (stores "Reported Core Number")

---

**Core Photos**

Wellbore > Table View > With > Core Photos

NPD columns:

Wellbore

Core sample number

Core photo title 10208-10228ft (separate into 3 columns).

Core photo URL (can I download these and save to disk?)

NPDID wellbore - IGNORE

Date updated - IGNORE

Thin sections (point - comment?)

CO2 (interval - collection???)

Oil samples (interval - collection???)

Drill stem depths  (interval - collection???)

Casing and leak-off tests (point - collection AND casing data type?)

Drilling mud (point - collection)

Documents (add to References?)

**Norwegian production data available at:**

Diskos Reports - Wellbore production/injection report - Monthly by wellbore (highest resolution available)

https://portal.diskos.cgg.com/prod-report-module/

**CODE**

**Updating**

update = [TABLE].update().\

where([TABLE].c.name = 'Todd').\

[values(email='newemail@example.com')](mailto:values(email='newemail@example.com'))

**Deleting specific rows**

import delete, select

#Select statement counts the records in the table, to ensure deletion of correct number of records.

**stmt = select([**

**func.count(table.columns.id)])**

#Execute the select statement and use the scalar() fetch method to save the record count

**connection.execute(stmt).scalar()**

#Option 1 - delete all records from a table so as to reload data

**delete\_stmt = delete(table)**

#Option 2 - build delete statement that targets the correct table

#Delete() takes the table we are loading data into as the argument

#Where clause chooses which rows to delete, targets id column for example

**delete\_stmt = delete(table).where(**

**table.columns.id == 3)**

**Substitute value 3 with a python variable name?**

#Execute the delete statement

**result\_proxy = connection.execute(delete\_stmt)**

#Print affected row count

**result\_proxy.rowcount**

#Print results of the executing statement to verify there are no rows

**print(connection.execute(select\_stmt).fetchall())**

**After running "df\_explo.to\_sql('WELLS', engine, if\_exists='append', index = False)" i get all these errors:**

ProgrammingError: (pyodbc.ProgrammingError) ('42S22', "--- Invalid column name 'Alternate 1'. (207) (SQLExecDirectW);

--- Invalid column name 'UWI number'. (207);

--- Invalid column name 'Block'. (207);

--- Invalid column name 'Operator'. (207);

--- Invalid column name 'Licence number'. (207);

--- Invalid column name 'Intent - planned'. (207);

 --- Invalid column name 'Well status'. (207);

 --- Invalid column name 'Well content'. (207);

 --- Invalid column name 'Status'. (207);

--- Invalid column name 'Type'. (207);

--- Invalid column name 'Subsea'. (207);

--- Invalid column name 'SPUD date'. (207);

--- Invalid column name 'Completion date'. (207);

--- Invalid column name 'Drill permit'. (207); --- Invalid column name 'Discovery name'. (207); --- Invalid column name 'Discovery wellbore'. (207); --- Invalid column name 'Bottom hole temperature [°C], [Seismic line'. (207); --- Invalid column name 'Maximum inclination [°], [KBE'. (207); --- Invalid column name 'Final vertical depth (TVD) [m RKB], [Terminal depth'. (207); --- Invalid column name 'Water depth'. (207); --- Invalid column name 'Kick off point [m RKB], [Oldest penetrated age'. (207); --- Invalid column name 'Oldest penetrated formation'. (207); --- Invalid column name 'Drilling facility type'. (207); --- Invalid column name 'Drilling facility category'. (207); --- Invalid column name 'Licensing activity awarded in'. (207); --- Invalid column name 'Multilateral'. (207); --- Invalid column name 'Entry year'. (207); --- Invalid column name 'Completed year'. (207); --- Invalid column name 'Reclassified from/to wellbore'. (207); --- Invalid column name 'Reentry activity'. (207); --- Invalid column name 'Plot symbol'. (207); --- Invalid column name '1st level with HC formation'. (207); --- Invalid column name '1st level with HC age'. (207); --- Invalid column name '2nd level with HC formation'. (207); --- Invalid column name '2nd level with HC age'. (207); --- Invalid column name '3rd level with HC formation'. (207); --- Invalid column name '3rd level with HC age'. (207); --- Invalid column name 'Drilling days'. (207); --- Invalid column name 'Reentry'. (207); --- Invalid column name 'Longitude'. (207); --- Invalid column name 'Surface X'. (207); --- Invalid column name 'Surface Y'. (207); --- Invalid column name 'UTM zone'. (207); --- Invalid column name 'Grid system'. (207); --- Invalid column name 'DISKOS Well Type'. (207); --- Invalid column name 'DISKOS Wellbore Parent'. (207); --- Invalid column name 'Publication date'. (207); --- Invalid column name 'Release date'. (207); --- Invalid column name 'NPDID wellbore'. (207); --- Invalid column name 'NPDID discovery'. (207); --- Invalid column name 'NPDID field'. (207); --- Invalid column name 'NPDID drilling facility'. (207); --- Invalid column name 'NPDID wellbore reclassified from'. (207); --- Invalid column name 'NPDID production licence drilled in'. (207); --- Invalid column name 'Date main level updated'. (207); --- Invalid column name 'Date all updated'. (207); --- Invalid column name 'Date sync NPD'. (207); --- There are fewer columns in the INSERT statement than values specified in the VALUES clause. The number of values in the VALUES clause must match the number of columns specified in the INSERT statement. (110); --- Statement(s) could not be prepared. (8180)")

**USEFUL QUERY TO FIND OTHER TABLES WITH SIMILAR COLUMN NAME**

E.g. like the data\_type column in DATA\_Lithostrat table.

SELECT      c.name  AS 'dest\_table'

            ,t.name AS 'INTERVALCOLUMNS'

FROM        sys.columns c

JOIN        sys.tables  t   ON c.object\_id = t.object\_id

WHERE       c.name LIKE 'dest%'

ORDER BY 'INTERVALCOLUMNS'

Useful Stored Procedure to "Search entire SQL Server database and all tables for a specified string"

[https://hanssens.com/code/search-entire-sql-server-database-and-all-tables-for-a-specified-string/](https://hanssens.com/code/search-entire-sql-server-database-and-all-tables-for-a-specified-string)

Create it and run "exec SearchAllTables 'gummy bears';