AGENDA
SQL VIEWS AND IMIS BUSINESS OBJECTS

What are we talking about?
Why is this stuff useful?
Language and terminology
Today's working environment

The basics
Creating a view in SQL
Creating a business object using your view

Getting fancy
Doing more in SQL views
Enhancing your business objects
Practical application examples
Getting help
WHAT ARE WE TALKING ABOUT?

WHY IS THIS STUFF USEFUL?

• The conference program says
  – The core Business Objects are very powerful and yet, most customers need much more
  – iMIS is becoming IQA driven and IQAs require Business Objects AND Business Objects require SQL Views (and tables)

• Making it easy for your users

• Elegant presentation to your constituents

WHAT ARE WE TALKING ABOUT?

WHY BUILD VIEWS AND BUSINESS OBJECTS?

• Because joins are hard (for some people)
• Because joins are irritating if you have to set them up all the time
• Because even some of the common things people want require reasonably in-depth knowledge of business objects

• Because using functions is not intuitive to everyone
• Because using functions to build SQL expressions in IQAs is fiddly

• Because you can make the database technicalities hidden to your users and let them get on at the level they are comfortable
WHAT ARE WE TALKING ABOUT?
LANGUAGE AND TERMINOLOGY

• SQL - Structured Query Language
• Table - Where data is stored
• Field - Data structure for a piece of information in a table
• View - A dynamic table
• Business object - Similar to a view, but effectively made available within the iMIS environment as a source for IQA
• Function - Reusable SQL logic that hides the steps and complexity from other code
• SQL Server Management Studio (SSMS) - The tool we use to manage a Microsoft SQL database

WHAT ARE WE TALKING ABOUT?
LANGUAGE AND TERMINOLOGY

• An example SQL select statement

```sql
SELECT Name.ID, 
    Name.FULL_NAME, 
    Name.PAID_THRU, 
    Name_Fin.RENEWED_THRU, 
    Name_Fin.RENEW_MONTHS
FROM Name INNER JOIN Name_Fin
ON Name.ID = Name_Fin.ID
WHERE Name.STATUS = 'A'
AND Name.MEMBER_TYPE IN ('MEM', 'AFF', 'STU')
ORDER BY Name.LAST_FIRST
```
WHAT ARE WE TALKING ABOUT?

LANGUAGE AND TERMINOLOGY

• SELECT is the equivalent of IQA display

```sql
SELECT
    Name.ID,
    Name.FULL_NAME,
    Name.PAID_THRU,
    Name_Fin.RENEWED_THRU,
    Name_Fin.RENEW_MONTHS
```

Query Display Columns

- Only display unique results

<table>
<thead>
<tr>
<th>Display</th>
<th>Property</th>
<th>Function</th>
<th>Alias</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️ NetContact.IMS_ID</td>
<td>None</td>
<td>ID</td>
<td>-part-as/ID/PED</td>
<td></td>
</tr>
<tr>
<td>✔️ NetContact.Major Key</td>
<td>None</td>
<td>None</td>
<td>Anonym</td>
<td></td>
</tr>
<tr>
<td>✔️ NetContact.FullName</td>
<td>None</td>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

WHAT ARE WE TALKING ABOUT?

LANGUAGE AND TERMINOLOGY

• FROM is the equivalent of IQA sources

```sql
FROM Name INNER JOIN Name_Fin
ON Name.ID = Name_Fin.ID
```

Query Sources

<table>
<thead>
<tr>
<th>Source</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetContact</td>
<td>Business Object</td>
<td>Equals</td>
</tr>
<tr>
<td>CoMEmail_Name_IMS</td>
<td>Business Object</td>
<td></td>
</tr>
</tbody>
</table>

Relations

- Custom (When NetContact.Company.Id = CoMEmail_Name_IMS.Id)
• WHERE is the equivalent of IQA filters

```sql
WHERE Name.STATUS = 'A'
AND Name.MEMBER_TYPE IN ('MEM', 'AFF', 'STU')
```

• ORDER BY is the equivalent of IQA sorting

```sql
ORDER BY Name.LAST_FIRST
```

• [You don’t actually use ORDER BY in views]
WHAT ARE WE TALKING ABOUT?
TODAY'S WORKING ENVIRONMENT

- iMIS 20.2 standard demo database
- iMIS staff site
- SQL Server Management Studio
- iMIS Desktop

THE BASICS
CREATING A VIEW IN SQL

- Access SQL Server Management Studio

[Image of SQL Server Management Studio interface]
THE BASICS
CREATING A VIEW IN SQL

• Log in

THE BASICS
CREATING A VIEW IN SQL

• Object Explorer and Object Explorer Details
THE BASICS
CREATING A VIEW IN SQL

• Use the view menu to get access to what you cannot see

• Multiple databases on a shared SQL server
THE BASICS
CREATING A VIEW IN SQL

• Expand the database in Object Explorer

• Create a new view with a menu (right) click on the Views folder
THE BASICS
CREATING A VIEW IN SQL

• You will be invited to add one or more tables
• For our exercise, add Name and Name_Fin then close the window

THE BASICS
CREATING A VIEW IN SQL

• You have access to a graphical tool for building your view
• Alternatively you can write the code directly
THE BASICS
CREATING A VIEW IN SQL

- Select the fields you want in your view by ticking them
  - From Name
    - ID
    - FULL_NAME
    - LAST_FIRST
    - PAID_THRU
  - From Name_Fin
    - RENEWED_THRU
    - RENEW_MONTHS

<table>
<thead>
<tr>
<th>Column</th>
<th>Alias</th>
<th>Title</th>
<th>Output</th>
<th>Sort Type</th>
<th>Sort Order</th>
<th>Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FULL_NAME</td>
<td>Name</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAST_FIRST</td>
<td>Name</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAID_THRU</td>
<td>Name</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RENEW_MON</td>
<td>Name_fin</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RENEW_MON</td>
<td>Name_fin</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

THE BASICS
CREATING A VIEW IN SQL

- Execute your SQL to test it
- Save your view
- Naming your view
  - Prefix it with vw_
  - Some people like to prefix with zvw_ to sort all their views to the bottom
  - It is good to include your organisation acronym or abbreviation
  - Consultants may use their own organisation acronym
THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• Switch to the staff site and sign in

![Sign In Form]

• Go to RiSE> Business Object Designer

![Business Object Designer Interface]
THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• Select New > Design Business Definition

THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• Name your business object and create it
Go to the Database tab and click Add

In the table list, scroll to select your view or start typing its name
THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• Select the view and click OK

THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• Click to Add All as Properties
THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• At the bottom right of the window, Save

• At the top right of the window, Publish

• Check the results

• At the bottom right of each window, Close

THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• Now create an IQA to use your business object
THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• Select your business object as a source

THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• Change your view to all columns and tick to display all
THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

- Run your IQA
- Don’t forget to save it

GETTING FANCY
USING SQL FUNCTIONS IN VIEWS

- Now you want to start using a basic query window in SSMS
  - Copy the SELECT statement from your view definition
  - Create a new query
  - Paste in your statement
  - It is easier to type your functions in here and then paste back into the view definition
GETTING FANCY
USING SQL FUNCTIONS IN VIEWS

• What can you do with functions?
  – Change data types
    • Datetime to character
    • Number/currency to character
    • Character to number
  – Manipulating character fields
    • Concatenating (joining)
    • Replacing text
    • Taking part of the field
    • Length and finding text
  – Calculations
    • With dates
    • With numbers
  – Display
    • Show the correct number of decimals
    • Show part of a datetime only

GETTING FANCY
USING SQL FUNCTIONS IN VIEWS

• Change data type: datetime to character
  – Use the CONVERT function

Syntax for CONVERT:
CONVERT ( data_type [ ( length ) ] , expression [ , style ] )

– Examples
  • CONVERT(varchar(10), Name.DATE_ADDED, 103)
    – 31/10/2015
  • CONVERT(varchar(11), Name.BIRTH_DATE, 106)
    – 16 Nov 1987
  • CONVERT(varchar(8), Name.LAST_UPDATED,108)
    – 10:45:06
GETTING FANCY
USING SQL FUNCTIONS IN VIEWS

• Change data type: number/currency to character
  – Use the CAST function
    
    Syntax for CAST:
    
    CAST ( expression AS data_type [ ( length ) ] )
    
    – Examples
    • CAST(Name_Fin.RENEW_MONTHS AS varchar(5))
      – 0, 3, 6, 12
    • CAST(Trans.AMOUNT AS varchar(20))
      – 122.73, -300.00

GETTING FANCY
USING SQL FUNCTIONS IN VIEWS

• Change data type: character to number
  – Use the CAST function
  – Example
    • CAST(Meet_Master.UF_1 AS INT)
      – 20000, 40000
  – You may need to trap for errors using the ISNUMERIC function
  – Example
    • CASE WHEN ISNUMERIC(UF_1) = 1 THEN CAST(UF_1 AS INT) ELSE 0 END
Manipulating character fields: concatenating
  – Use + in the same way you would use & in Excel
  – Example
    • PREFIX+FIRST_NAME+LAST_NAME
      – MrConradHenley-Calvert
  – Better example
    • PREFIX + ' ' + FIRST_NAME + ' ' + LAST_NAME
      – Mr Conrad Henley-Calvert

Manipulating character fields: replacing text
  – Use the REPLACE function

REPLACE ( string_expression , string_pattern , string_replacement )

– Example
  • REPLACE(DESCRIPTION, 'CPD', 'Continuous Professional Development')
**GETTING FANCY**

**USING SQL FUNCTIONS IN VIEWS**

- Manipulating character fields: taking part of the field
  - Use the LEFT, RIGHT and SUBSTRING functions

    \[
    \text{SUBSTRING} \left( \text{expression}, \text{start}, \text{length} \right)
    \]

    - Examples
      - `LEFT(DESCRIPTION, 5)`
      - `RIGHT(DESCRIPTION, 7)`
      - `SUBSTRING(DESCRIPTION, 5, 3)`

- Manipulating character fields: length and finding text
  - Use LEN to find length of a field
    - Example
      - `LEN(DESCRIPTION)`
  - Use CHARINDEX to find if text exists within a field
    - If the specified text is not found, 0 is returned
    - Examples
      - `CHARINDEX('Annual', DESCRIPTION)`
      - `CHARINDEX('Annual', DESCRIPTION, 4)`
Calculations: with dates
- Use the DATEADD function to add or subtract intervals from dates
  \[ \text{DATEADD}(\text{datepart}, \text{number}, \text{date}) \]
  - Examples
    - \text{DATEADD}(\text{yyyy}, 18, \text{BIRTH\_DATE})
    - \text{DATEADD}(\text{mm}, 3, \text{JOIN\_DATE})

- Use the DATEDIFF function to return the time between two dates
  It is specifically the number of boundaries crossed
  \[ \text{DATEDIFF}(\text{datepart}, \text{startdate}, \text{enddate}) \]
  - Example
    - \text{DATEDIFF}(\text{yyyy}, \text{BIRTH\_DATE}, \text{JOIN\_DATE})

Calculations: with numbers
- + - * /
- \text{ABS}() for the absolute value
  - \text{ABS}(45) is 45, \text{ABS}(-34) is 34
- \text{ROUND}()
  - \text{ROUND}(3.14) is 3, \text{ROUND}(3.5) is 4, \text{ROUND}(3.75) is 4, \text{ROUND}(4) is 4
- \text{CEILING}()
  - \text{CEILING}(3.14) is 4, \text{CEILING}(3.5) is 4, \text{CEILING}(3.75) is 4, \text{CEILING}(4) is 4
- \text{FLOOR}()
  - \text{FLOOR}(3.14) is 3, \text{FLOOR}(3.5) is 3, \text{FLOOR}(3.75) is 3, \text{FLOOR}(4) is 4
**GETTING FANCY**

**USING SQL FUNCTIONS IN VIEWS**

- **Display:** show the correct number of decimals
  - Use the CAST function to change the data type
    - Example
      - `CAST(QUANTITY as INT)`
  - Use the ROUND function with arithmetic to specify decimal places
    - Example for two decimal places
      - `ROUND(UNITS * 100) / 100`

**GETTING FANCY**

**USING SQL FUNCTIONS IN VIEWS**

- **Display:** show part of a datetime only
  - Use the DATEPART function to extract part of a date
    - Examples
      - `DATEPART/yyyy, TRANSACTION_DATE`
      - `DATEPART/mm, TRANSACTION_DATE`
      - `DATEPART/dd, TRANSACTION_DATE`
  - Use CONVERT to change the data type
    - Examples
      - `CONVERT(date, DATE_ADDED)`
      - `CONVERT(time, DATE_ADDED)`
  - And working with the current date?
    - Use `GETDATE()`
GETTING FANCY
USING CASE EXPRESSIONS IN VIEWS

• Conditions

  Simple CASE expression:
  CASE input_expression
      WHEN when_expression THEN result_expression [ ... ]
      [ ELSE else_result_expression ]
  END
  Searched CASE expression:
  CASE
      WHEN Boolean_expression THEN result_expression [ ... ]
      [ ELSE else_result_expression ]
  END

GETTING FANCY
USING CASE EXPRESSIONS IN VIEWS

• Simple CASE expression

  CASE RENEW MONTHS
      WHEN 0 THEN 'Annual'
      WHEN 3 THEN 'Quarterly'
      WHEN 6 THEN 'Half Yearly'
      WHEN 12 THEN 'Annual'
      ELSE 'Check Value of Renew Months'
  END
GETTING FANCY
USING CASE EXPRESSIONS IN VIEWS

- Searched CASE expression

```sql
CASE
  WHEN MEMBER_TYPE = 'MEM' and PAID_THRU > GETDATE()
    THEN 'Financial Member'
  WHEN MEMBER_TYPE = 'MEM'
    THEN 'Unfinancial Member'
  WHEN MEMBER_TYPE = 'STU'
    THEN 'Student Member'
  WHEN MEMBER_RECORD = 1
    THEN 'Other Member'
  ELSE 'Non Member'
END
```

---

GETTING FANCY
JOINS AND FILTERS IN BUSINESS OBJECTS

- Business objects can use views or tables
- Business objects can use more than one view or table
  - You need to define a join
  - Look at the definition of CsContact as an example

```
CsContact
```

```
<table>
<thead>
<tr>
<th>Tables</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (Primary)</td>
<td></td>
</tr>
<tr>
<td>Name_Fin</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Joins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name_Fin.ID = Name.ID</td>
</tr>
</tbody>
</table>
```
• Business objects can use a filter so that they only display a subset of the data in the table(s)
  – Look at the definition of CsLetters as an example

GETTING FANCY
JOINS AND FILTERS IN BUSINESS OBJECTS

GETTING FANCY
ORDERING PROPERTIES

• Clicking “Add All as Properties” adds all fields to your business object in alphabetic order

• You can select and add fields individually using “Add as Properties” to have control over the order

• Once you have added a property you have to delete it to be able to reorder it by adding it to the bottom of the list
GETTING FANCY
MAKING PROPERTIES DEFAULT

- On the Properties tab, each property has its own tabs for
  - Definition
  - Display
  - Values

- The definition tab allows you to specify that a property will automatically be included “default” on the display of an IQA

GETTING FANCY
FIELD PROMPTS AND DISPLAY MASKS

- You don’t have to accept the field prompt that is assigned by default

- You can change the display mask for the field, especially for dates
• You can create a custom value list for your field in the business object.

GETTING FANCY
FIELD VALUES

• The best way to do this is with a query object (IQ).

GETTING FANCY
FIELD VALUES
GETTING FANCY
FIELD VALUES

• There is a set of predefined value lists in
  – RiSE> Intelligent Query Architect
    • Common> Queries> Value Lists

• You can create your own value list IQA from scratch
• For general lookup/validation, you can copy an existing one, such as CategoryList

• All General lookup/validation ‘tables’ are in fact held in one database table called Gen_Tables, made available to IQA in a business object called CsGeneralLookupTables

• To get the values for your chosen table, you need to filter on the field TABLE_NAME, which shows in the business object as Table Name
• Create a value list for the validation table NEWSTYPE

![General lookup/validation](image)

• This is the validation table for Communication.NewsType

![Field values](image)

• In RiSE> Intelligent Query Architect> Common> Queries> Value Lists
  – Select and Edit the IQA CategoryList
  – Change the Table Name filter to be NEWSTYPE instead of CATEGORY

![Document Browser](image)

  – Save As a new IQA
Now let's add the table and field to our view

In SSMS, find the view, menu (right) click it and click Design

Menu (right) click in the tables area and click Add Table

Add the Communication table and close the Add Table window

Add NewsType to the selected columns by ticking it and save the view
Now select and edit your business object
Go to the Database tab add the new field as a property

Save your business object
GETTING FANCY
FIELD VALUES

- Choose query object, browse to your IQA (Common> Queries> Value Lists), select it and click OK
- Set Data Value = Code and Display Value = Description

- Save and Publish your business object

GETTING FANCY
FIELD VALUES

- Now go to your IQA
- On the filters tab, add News Type and see how you can select values from the drop down list
GETTING FANCY
PRACTICAL APPLICATION: ACTIVITY TYPES

- Make activities easy for your staff to include in IQAs
  - Prepare by creating a value list for each validation table for the activity
  - Create a business object
  - Add a filter to the business object for the activity type
  - Add ID, SEQN, TRANSACTION_DATE (the activity date)
  - Add only the fields you have labelled as part of the activity type definition, in a logical order
  - Save
  - For each field
    - Tick if you want it automatically included in queries
    - Set the prompt to match your activity type definition
    - If it is a date, consider changing the display mask to short date
    - If appropriate, link to a value list
  - Save and Publish

GETTING FANCY
PRACTICAL APPLICATION: EXTENDED EVENT INFORMATION

- Provide a drop down list for event selection
- Bring additional information about events through for IQAs

- Create a view based on the Meet_Master table that contains
  - MEETING (the event code)
  - A field combining event date and name information
  - Fields to sort your list by
- Create a business object using this view
- Create a value list IQA using this business object as a source
- Create another business object using Meet_Master
  - Add MEETING (the event code) and use the list above as a value list
  - Add any of MUF_1 to MUF_10 that you use and edit the prompts
- Use this business object as a source along with CsEvent
GETTING FANCY
PRACTICAL APPLICATION: IQAS FOR MEMBER AND PUBLIC USE

- If you publish an IQA on your website
  - The content item must have security that allows the target audience to access it
  - The IQA must have security that allows the target audience to access it
  - Every value list used in every business object used as a source for the IQA must also have security that allows the target audience to access it

- So, if you are basing a “find a member” search on an IQA, you will need to check the value lists used for all business objects, both standard and ones that you have created.

GETTING FANCY
PRACTICAL APPLICATION: MULTI-SELECT FIELDS

- These are a pain to provide a user-friendly way to analyse, especially if you want to publish reports prompting for runtime parameters

- Create a view that expands the multi-select field

```sql
SELECT TOP (100) PERCENT dbo.Name_C0_Profile.Id,
       dbo.Gen_Tables.CODE AS DIVISION
FROM
       dbo.Name_C0_Profile CROSS JOIN
       dbo.Gen_Tables
WHERE
       (dbo.Gen_Tables.TABLE_NAME = 'DIVISION')
AND
       (CHARINDEX(dbo.Gen_Tables.CODE, dbo.Name_C0_Profile.DIVISIONS) > 0)
UNION
SELECT Id, '' AS DIVISION
FROM
       dbo.Name_C0_Profile
WHERE
       DIVISIONS = ''
```
GETTING FANCY
PRACTICAL APPLICATION: MULTI-SELECT FIELDS

• Create a value list for your validation field
• Create a business object using the view
• Use the value list as the query object for the expanded field’s values

GETTING FANCY
PRACTICAL APPLICATION: MULTI-SELECT FIELDS

• Add this expansion source to your IQA linked by ID
• Use the expanded field in the filters, ticked to allow multiple values

• Include the original multi-select field in the display, but not the expanded field
• Set the display to only include unique results

• If you select three expanded values, your results will be any record that has at least one of those values in the multi-select field
GETTING FANCY
PRACTICAL APPLICATION: MULTI-SELECT FIELDS

• If you want descriptions rather than codes in your results, you can expand your multi-select field to full descriptions using a user-defined function.

• Another recent suggestion on the NiUG Listserv for dealing with this was to use the view to parse out the multi-select field into one field per possible value (using CASE statements and CHARINDEX).

GETTING FANCY
THE AUTO-GENERATE UTILITY AND ONGOING MANAGEMENT

• In the staff site, under Settings> Utilities> User-defined tables

  ![Auto-Generate Business Objects from User-Defined Tables](image)

  **Build Results**

• You can do a one-off generation of business objects for all your user-defined tables.
• It doesn’t do the work of creating and setting value lists.
• Once you have made manual changes to the auto-created objects, don’t run it again or you will lose your changes.
GETTING FANCY

GETTING HELP

• Go to the web
• You want references for Transact-SQL, which is the Microsoft-specific implementation of SQL

<table>
<thead>
<tr>
<th>SQL functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Functions - W3Schools [link]</td>
</tr>
<tr>
<td>SQL aggregate functions return a single value, calculated from values in a column.</td>
</tr>
<tr>
<td>Useful aggregate functions: AVG() - Returns the average value. COUNT() - Returns the number of rows. FIRST() - Returns the first value.</td>
</tr>
<tr>
<td>Built-in Functions (Transact-SQL) - MSDN - Microsoft [link]</td>
</tr>
</tbody>
</table>

AND FINALLY

• Don’t double up on sessions (unless you want to)

• My conference presentation on SQL Views and Functions is a cut-down, non-hands-on version of this training session
WITH THANKS TO OUR SPONSORS
NI UG DISCOVERY CONFERENCE