# SQL Views and Functions

**Conrad Henley-Calvert, Millpost Technologies**

## Agenda

**Introduction**
- This session
- Why build views
- Language and terminology

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

**Getting fancy**
- Using SQL functions in views
- Practical application examples
- Getting help
This presentation is a cut-down version of my training session from yesterday on SQL Views and Business Objects.

It will cover the basics of using a view in a business object.

The primary focus is creating views and using functions, so for more in-depth guidance on the business object side of things, check the slides from the training session.

**WHAT ARE WE TALKING ABOUT?**

**THIS SESSION**

- 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

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

WHAT ARE WE TALKING ABOUT?
LANGUAGE AND TERMINOLOGY

• **FROM** is the equivalent of IQA sources

```
FROM Name INNER JOIN Name_Fin
ON Name.ID = Name_Fin.ID
```
WHAT ARE WE TALKING ABOUT?
LANGUAGE AND TERMINOLOGY

WHERE is the equivalent of IQA filters

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

• ORDER BY is the equivalent of IQA sorting

```
ORDER BY Name.LAST_FIRST
```

• [You don’t actually use ORDER BY in views]
THE BASICS
CREATING A VIEW IN SQL

- Access SQL Server Management Studio

[Image of SQL Server Management Studio]

THE BASICS
CREATING A VIEW IN SQL

- Log in

[Image of Connect to Server dialog box]
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
THE BASICS
CREATING A VIEW IN SQL

• Multiple databases on a shared SQL server

THE BASICS
CREATING A VIEW IN SQL

• Expand the database in Object Explorer
THE BASICS
CREATING A VIEW IN SQL

• 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
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

SELECT * FROM

dbo.Name INNER JOIN

dbo.Name_Demo ON dbo.Name.ID = dbo.Name_Demo.ID

THE BASICS
CREATING A VIEW IN SQL

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

• Sign into the staff site
THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• Go to RiSE> Business Object Designer

![Diagram showing RiSE interface with Business Object Designer highlighted.]

THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• Select New> Design Business Definition

![Diagram showing RiSE interface with Design Business Definition highlighted.]

12
THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• Name your business object and create it

![Business Object Designer](image1)

• Go to the Database tab and click Add

![Database Tab](image2)
THE BASICS
CREATING A BUSINESS OBJECT USING YOUR VIEW

• 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

• 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
• 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

• 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

GETTING FANCY
USING SQL FUNCTIONS IN VIEWS

• 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 , stringReplacement )
```

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

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

```
SUBSTRING ( expression , start , length )
```

– 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
  - DATEADD(datepart, number, date)

- Examples
  - DATEADD(yyyy, 18, BIRTH_DATE)
  - DATEADD(mm, 3, JOIN_DATE)

- Use the DATEDIFF function to return the time between two dates
  - It is specifically the number of boundaries crossed
  - Example
    - DATEDIFF(yyyy, BIRTH_DATE, JOIN_DATE)
<table>
<thead>
<tr>
<th>GETTING FANCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>USING SQL FUNCTIONS IN VIEWS</td>
</tr>
</tbody>
</table>

- **Calculations: with numbers**
  - $+$, $-$, $\times$, $/$
  - $\text{ABS()}$ 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

---

<table>
<thead>
<tr>
<th>GETTING FANCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>USING SQL FUNCTIONS IN VIEWS</td>
</tr>
</tbody>
</table>

- **Display: show the correct number of decimals**
  - Use the $\text{CAST}$ function to change the data type
    - Example
      - $\text{CAST(QUANTITY as INT)}$
  - Use the $\text{ROUND}$ function with arithmetic to specify decimal places
    - Example for two decimal places
      - $\text{ROUND(UNITS * 100) / 100}$
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 [ ... n ]
  [ ELSE else_result_expression ]
END

Searched CASE expression:
CASE
  WHEN Boolean_expression THEN result_expression [ ... n ]
  [ ELSE else_result_expression ]
END
GETTING FANCY
USING CASE EXPRESSIONS IN VIEWS

• Simple CASE expression

```sql
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
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: 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_CI_Profile.ID,
       dbo.Gen_Tables.CODE AS DIVISION
FROM    dbo.Name_CI_Profile CROSS JOIN
dbo.Gen_Tables
WHERE   (dbo.Gen_Tables.TABLE_NAME = 'DIVISION')
AND (CHARINDEX(dbo.Gen_Tables.CODE, dbo.Name_CI_Profile.Divisions) > 0)
UNION
SELECT  ID, '' AS DIVISION
FROM    dbo.Name_CI_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

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