SQL SELECT extensions in Orixa

There are a small number of additions and extensions to the SQL SELECT syntax which are used in Orixa. These are intended to make it easier for your App to interact with the database.

When an extension is added, if that SQL is run in your App the app will request input from the user, and return this input into the SQL statement. Extensions are usually added to WHERE clauses, and used to limit the number of records returned to the user, so the user sees records that are useful to them.

The extensions allow specialized keywords to be added to a SQL SELECT, which result in variables to be pulled into the SQL SELECT from your App. Using an extension involves adding the keyword(s) to the SQL enclosed in square brackets. If you see square brackets used in any of the SQL of your App's system-tables, it is an Orixa SQL extension.

There are also "lazy extensions" which are even shorter pieces of text that can be added to your SQL to prompt for user input. These should be avoided if possible, as they are ambiguous, but do also work.

List of Orixa SQL Extensions

Add any of these elements to a SQL Statement to create an entry point for your App to request a date from the user.

Note that the Extension can be used at multiple points in the SQL statement, but the user will only be prompted for 1 entry, and this will be reused at each place that the extension is present in the statement.

Orixa SQL extensions are used most widely in the **Searches** and **Resources** system-tables. Reviewing the content of this table shows examples of how Orixa SQL extensions work. The SQL linked to Resources will run when the user clicks on an "Action" somwhere in the Orixa App. Searches will run when the user shows a Grid of data in an Orixa App.

A help-topic on the Searches system-table is here: The Searches System Table

A help-topic on the Resources system-table is here: Using the "Resources" system-table to create reports, charts, data-cubes and dashboards

1. **[MinDate] [MaxDate]** The user will be prompted to enter 1 or 2 dates.

2. **[CurrentUser]** The ID of the current user will be returned into the SQL, silently.

3. **[STR <prompttext>]** The user will be prompted to enter text, <prompttext> will be added as the label for the text entry, so give a context for their entry. The App returns the text the user has entered into the SQL statement. Note this means the SQL-coder has to add SQL to cope with the fact that a varchar-variable is being returned. An example of how to use this is shown below.

4. **[BO <tablename>]** The user will be prompted to pick a record from the data-table <tablename> the list of elements to pick will be created using the ListScript from the BusinessObjects record for <tablename>. The App returns the **ID** of this record into the SQL. Note this extension should be used with care, if a data-table contains millions of records it is not sensible to use this extension, in such cases use "[LU <resourcename>]" as described below, and write a resource to limit the returned records to a reasonable number.

5. **[LU <resourcename>]** The user will be prompted to pick a record from a list created by running SQL in a Resources record with name <resourcename>. The framework will return an "ID" value. The system will look for <resourcename> in the Resources system table for SQL which will be used to generate the list. Note that the <resourcename> record must be formatted correctly, with a "name" field and "ID" field for this to work. Incorrectly formatted Resources will result in an error.

6. **[LUTypes <fieldname>, <tablename>]** The user will be prompted to pick one "type" for records in a <tablename>. The App returns the ID of the selected record into the SQL. For example if the system has a "Products" table and a "ProductsType" field, the user will be prompted to select 1 ProductType.

7. **[LUStatus <tablename>]** The user will be prompted to pick one "status" for records in <tablename>. The App returns the ID of this Status system-table record.

8. **[LUSTR <resourcename>]** The user will be prompted to pick a record from a list created by running SQL in a Resources record with name <resourcename>. The system will look for <resourcename> in the Resources system table for SQL which will be used to generate the list. This SQL must include a "Name" field. The user will be shown the list of "Names" and when they pick one **the chosen text** will be returned

8. to the SQL.

9. **[BOOL <prompttext>]** The user will be shown a tickbox, labelled with <prompttext>. If they tick it the boolean value "true" will be returned into the SQL, if they leave it blank the boolean value false will be returned.

10. **[NUM <prompttext>]** The user asked to enter a number, labelled with <prompttext>, The data-entry field will only allow valid entry of numbers. The App will return this number into the SQL statement. Orixa expects the numbers to be whole numbers.

11. **"Lazy extension" %s** The user will be prompted to enter some text, which will be added to the SQL Statement replacing the %s characters. Note that because the percent sign can be used in other parts of a SQL SELECT Statement, this extension can only be used in certain situations. [STR <sometext>] should probably always be used instead, The "Lazy Extension" is maintained for backward-compatibility purposes.

12. **"Lazy extension" %d** The App will try to add an ID into the SQL. If the currently open window is associated with a BusinessObject, the App will either substutite the ID of the currently open record or open the same list that is opened for the extension [BO <tablename>]. For the same reasons as for Lazy extension %s, use of %d is discouraged.

SQL Extension example: [MinDate] and [MaxDate]

This is perhaps the simplest and most obvious use of a SQL extension in Orixa, and gives a clear example of why these extensions exist and what they are useful for.
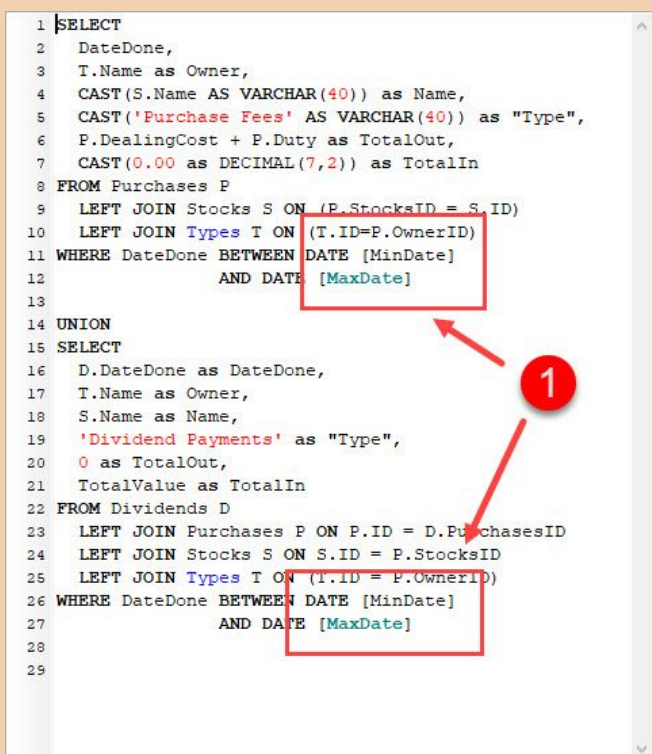
Often you will write SQL to generate a report, grid or chart, but want the user to be able to choose for themselves the date or date-range of data that is returned. If the SQL was fixed you would need to write different scripts to return one year or one month's worth of data.

By adding the [MinDate] and [MaxDate] extensions, one script can be written and each time it is run the user can select any date range they wish.

SQL Select MaxDate MinDate

Example SQL:

```
SELECT
DateDone,
T.Name as Owner,
CAST(S.Name AS VARCHAR(40)) as Name,
CAST('Purchase Fees' AS VARCHAR(40)) as
"Type",
P.DealingCost + P.Duty as TotalOut,
CAST(0.00 as DECIMAL(7,2)) as TotalIn
FROM Purchases P
LEFT JOIN Stocks S ON (P.StocksID = S.ID)
LEFT JOIN Types T ON (T.ID=P.OwnerID)
WHERE DateDone BETWEEN DATE [MinDate] AND
DATE [MaxDate]

UNION
SELECT
D.DateDone as DateDone,
T.Name as Owner,
S.Name as Name,
'Dividend Payments' as "Type",
0 as TotalOut,
TotalValue as TotalIn
FROM Dividends D
LEFT JOIN Purchases P ON P.ID = D.PurchasesID
LEFT JOIN Stocks S ON S.ID = P.StocksID
LEFT JOIN Types T ON (T.ID = P.OwnerID)
WHERE DateDone BETWEEN DATE [MinDate] AND
DATE [MaxDate]
```

MinDate-MaxDate Date Picker Form

When the above SQL is run, Orixa will dynamically generate a form to gather data from the user, as shown on the left. For every SQL extension that you add to the SQL, Orixa will add a user data-entry element to the "picker form".

These user data-entry elements are configured to allow correct data selection. For example, if the extension is for a Date, then the user will be prompted to pick a valid date, and not type just any text.

Setting a date range for any SQL SELECT

- The Orixa will simply "drop in" the dates selected by the user into the SQL SELECT. This means that you must add the DATE or AS DATE keyword to allow the database to interpret the value.

- It is perfectly acceptable to use either [MinDate] or [MaxDate] both do not have to be used together.

- They can be used as often as required, multiple instances of either will be substituted with the same value at each point in the SQL statment.

Examples using other SQL Extensions and how the resulting Searches records appear in the User Interface

Search called "By Date Created or Edited "

Image on the left shows a list of searches for a Business Object

The user has selected one. It's SQL is as follows:

```
WHERE DateCreated >= CAST([MinDate] as
TIMESTAMP)
OR ID IN
( SELECT
  LinkID
  FROM EditHistory
  WHERE DateCreated >= CAST([MinDate] as
       TIMESTAMP)
  AND LinkTable = 'Resources'
)
```
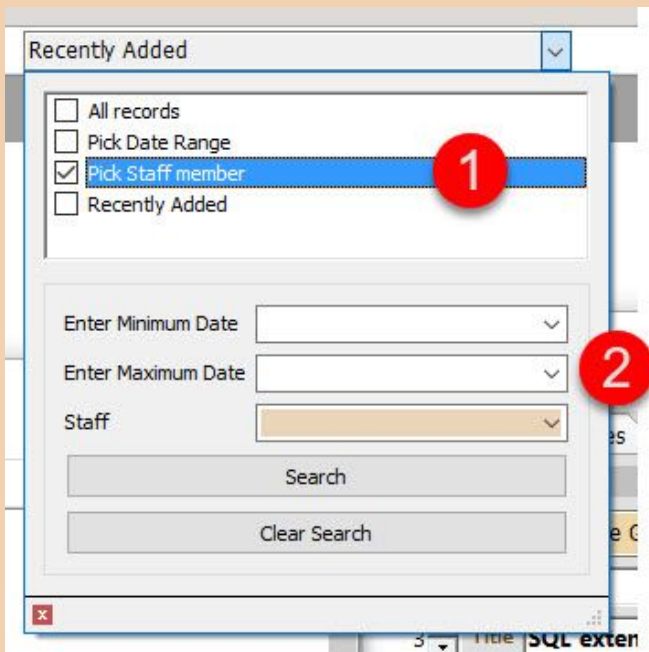
1. User clicks on "Pick Staff Member" item in search list

2. Orixa App constructs a "search picker containing fields to allow data entry based on the SQL extensions present in the data.

SQL Used to create this UI

```
WHERE AuthorID = [BO Staff]
AND DateCreated BETWEEN CAST([MinDate] AS
TIMESTAMP)
AND CAST([MaxDate] AS TIMESTAMP)
```
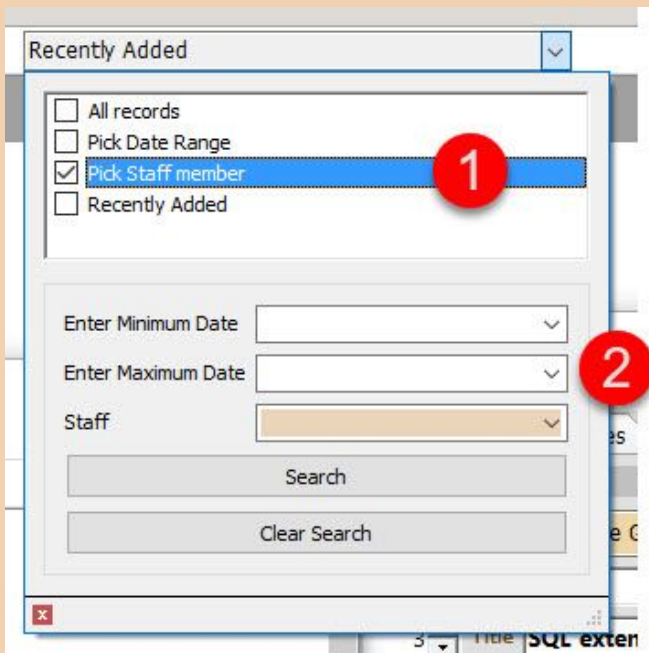
SQL Extensions Searches Window

SQL Extensions Searches Window using "Types"

Example using the [LUTypes ...] extension

A drop-down list is created in the App. It will contain all the records for entered in the Types table for Linktable = <tablename> and Linkfield = <linkfield> in the SQL

SQL Used to create this UI

```
WHERE ProductsTypeID = [LUTypes
ProductsTypeID, Products]
```

Multiple uses of a single extension

In the SQL below, the extension [STR Email] is used 3 times. However the user is only prompted to "Enter text" once.

Orixa creates one text prompt for each extension, so if [STR Name] and [STR Description] had been added 2 prompts would have been created.

SQL Used to create this UI

```
WHERE (UPPER(WhoToList) LIKE UPPER('%[STR Email]%')
OR UPPER(EmailCCList) LIKE UPPER('%[STR Email]%') OR UPPER(WhoFrom) LIKE UPPER('%[STR Email]%'))
AND DateSent BETWEEN DATE [MinDate] AND DATE [MaxDate]
```