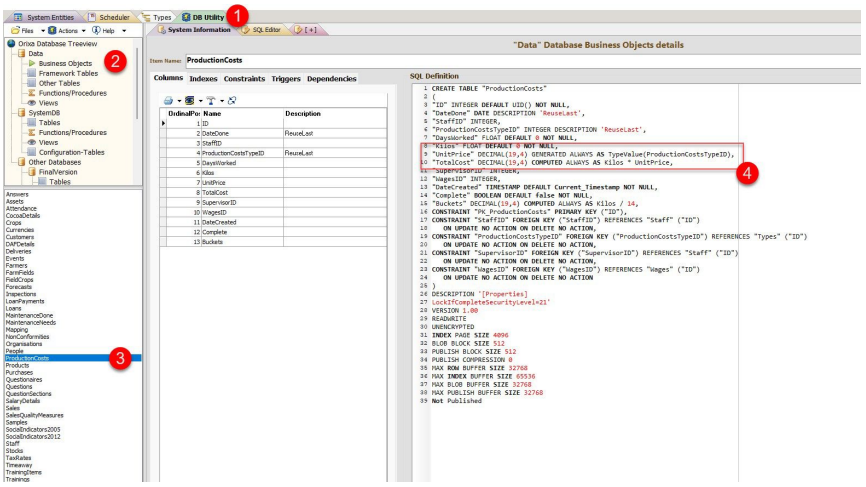
Using the Description field of a Types record to Generate a value

Where records contain a lot of repeated values, reusable values can be stored in a look-up field in the Types framework-table.
In this way, simply choosing the value for a "TypeID" field will automatically pull through the value stores in the "Description" field of the Types framework-table.
This allows both number and longer text data to be stored in the "Types" framework-table and pulled into a Business-Object table reducing repetative keying of data by staff.
Care needs to be taken with this pattern of data-design, as GENERATED columns will automatically recompute when the data in a record changes. Therefore, if the Types value is updated, the linked field will also update.

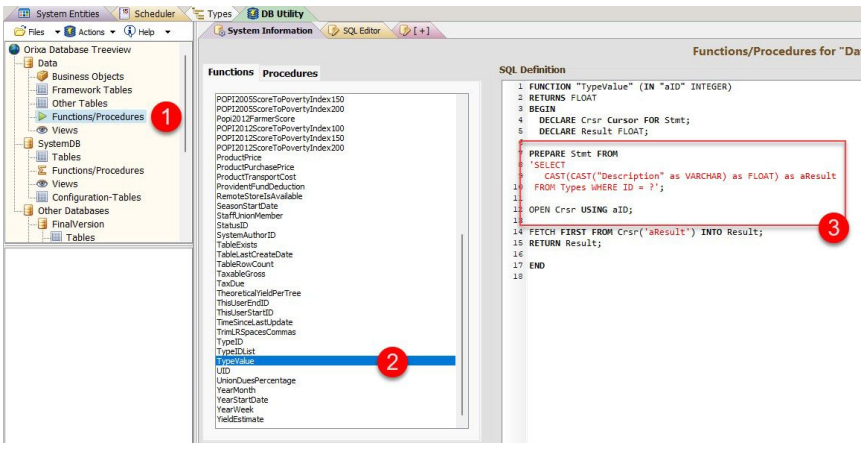
How to implement this in Your App



BusinessObject Using "TypeValue" function

1. In the DB Utility in your App
2. Click on the "BusinessObjects" heading.
3. Pick the ProductionCosts data-table.
4. View how the "UnitPrice" field is generated: By accessing a value using the "TypeValue" function. The SQL is: "UnitPrice" DECIMAL(19,4) GENERATED ALWAYS AS TypeValue(ProductionCostsTypeID)

How do I see what happens in the "TypeValue" Function?



Reviewing the "TypeValue" function

1. Click on the "Functions/Procedures" heading.
2. Find the "TypeValue" Function.
3. The SQL Definition will show

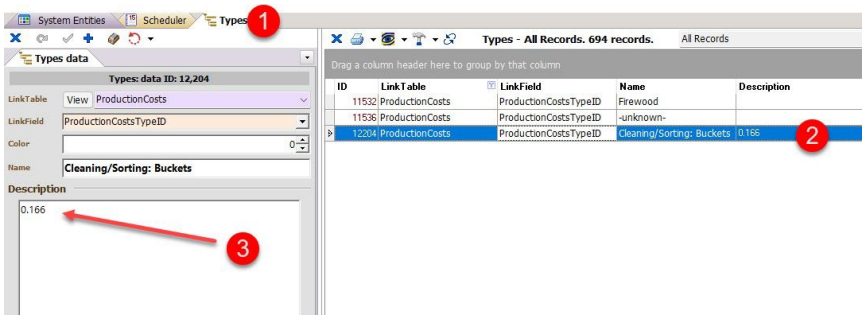
Notes about this SQL Definition

```
FUNCTION "TypeValue" (IN "aID" INTEGER)
RETURNS FLOAT
BEGIN
  DECLARE Crsr Cursor FOR Stmt;
  DECLARE Result FLOAT;
PREPARE Stmt FROM
  'SELECT
   CAST(CAST("Description" as VARCHAR) as FLOAT) as aResult
   FROM Types WHERE ID = ?';
OPEN Crsr USING aID;
FETCH FIRST FROM Crsr('aResult') INTO Result;
RETURN Result;
END
```

- The function is pretty simple, it just finds the record in the Types framework-table identified by the "aID" parameter passed into the function, and passes this back to the data-table to use in the "UnitPrice" field.
- It uses the CAST keyword. This is because the Description field is a CLOB long-character field, which must be returned as a FLOAT (number) field.
- Note that if the Description field cannot be CAST as a number (for example if text is stored in the field) an error will be generated.
- Note that if the Description field is **blank** NULL will be returned. If it is important that the number field is "0" rather than NULL, rewrite the Function adding an "IF aResult = NULL THEN SET Result = 0" clause.
- If you wanted to use this technique to return a piece of text stored in the Description field, you would need to add a second **Function**. This function would not need to use the CAST keyword.
- It is important to understand that the GENERATION will occur whenever the record is updated, even if the TypeID field is not changed. If you want to protect the value stored in the UnitPrice field so that it does not update in such cases you can add a "Complete" boolean field to the data-table, and test it's value in the GENERATION process.

Updating the Value Used in the Generation

This can be done for multiple values by opening the "Types" framework table.

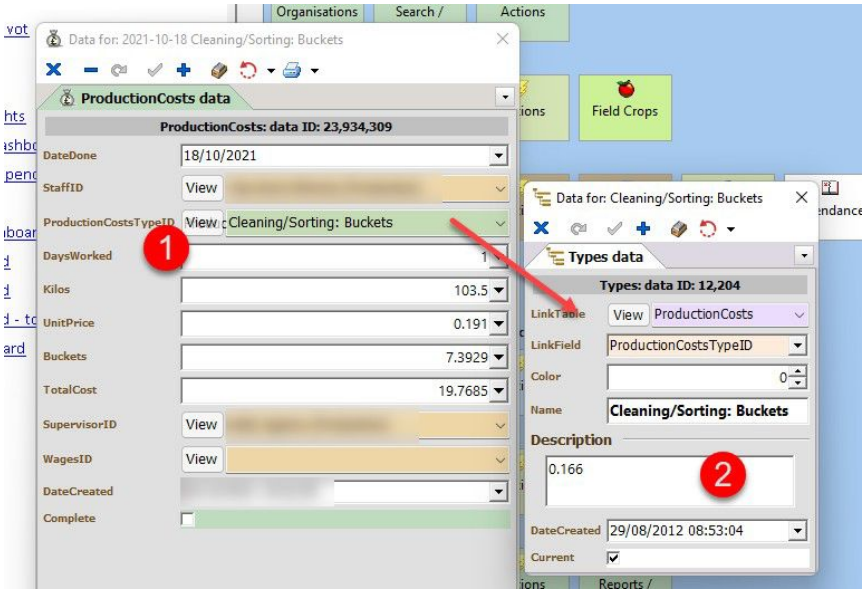


Updating a Types / Description lookup value

To update the Lookup value undertake the following steps

1. Open the Types data-table.
2. Find the row you wish to update
3. The value can be updated directly.

You can also update the value directly from a record in your App by accessing the Types record, via the "View" button as shown below



Editing The Types Record directly from the BusinessObject

1. In the correct Edit Window, find a record with the TypeID value you want to update, and click the "View" button.
2. The Types record will open and you can update the value.

What if I have entered some values wrongly and I want to update many records?

If a large number of records have been added to your system, without realizing that the UnitPrice should have been raised take the following steps:

1. Update the value to the correct amount, as shown above.
2. Find the range or ID's or the range of dates for which you need to change the values, and run the SQL shown below.

Range of IDs

```
UPDATE ProductionCosts
SET DateCreated = DateCreated + INTERVAL '1' SECOND
WHERE ID BETWEEN [enter low id]
         AND [enter high id]
```

Range of dates

```
UPDATE ProductionCosts
SET DateCreated = DateCreated + INTERVAL '1' SECOND
WHERE DateDone BETWEEN DATE '[enter low date in YY-MM-DD format]'
                AND DATE '[enter high date in YY-MM-DD format]'
```

How does this work?

The database runs the "GENERATION" process whenever the data in a row is updated. This means that "bumping" the "DateCreated" field by 1 second forces the "TypeValue" function to re-run. If the Value in the Description field has been updated, this will cause the recomputation of the UnitPrice field with the new value.

Note that if the data contains conditional statements (for example only running the Generation if a Complete Boolean field is false) then the SQL must ALSO "bump" the value of this field to ensure that Generation occurs.