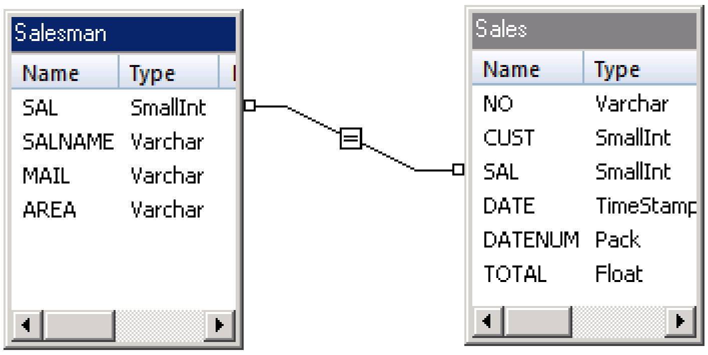
Knowledge Base Article: Article 261

Subqueries in Builder - Click & DECiDE BAI 21 (nested SELECT in the FROM clause)

Click and DECiDE BAI version 21 provides a new feature in Builder for making a nested SELECT in the FROM clause easily and friendly, by viewing and using a subquery as a table in the main query.

1. Use example

Working on the Demo database which is installed by the product, suppose we need a query that counts the number of salesmen and the total sales amount by area.
Involved tables are *DataSetReport.demo.Salesman* and *DataSetReport.demo.Sales* :



Salesman

Name	Type
SAL	SmallInt
SALNAME	Varchar
MAIL	Varchar
AREA	Varchar

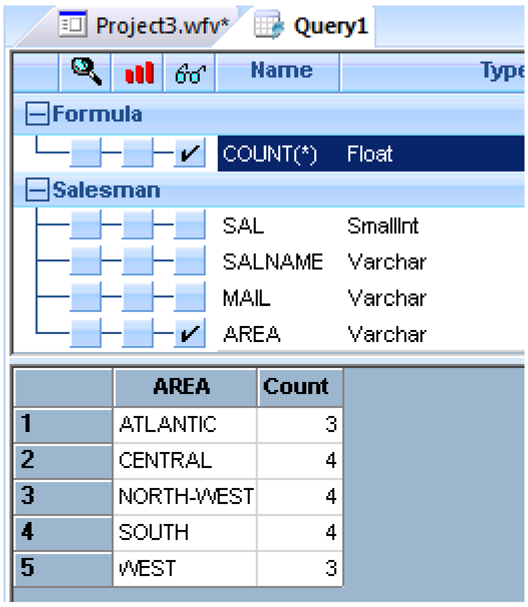
Sales

Name	Type
NO	Varchar
CUST	SmallInt
SAL	SmallInt
DATE	TimeStamp
DATENUM	Pack
TOTAL	Float

We cannot aggregate the fields SAL (COUNT) and TOTAL (SUM) in the same query because the relation between the two tables is a join of type 1-N: there are several sales for the same salesman, so every salesman would be counted for each sale and the result would be wrong.

We must do it in two steps:

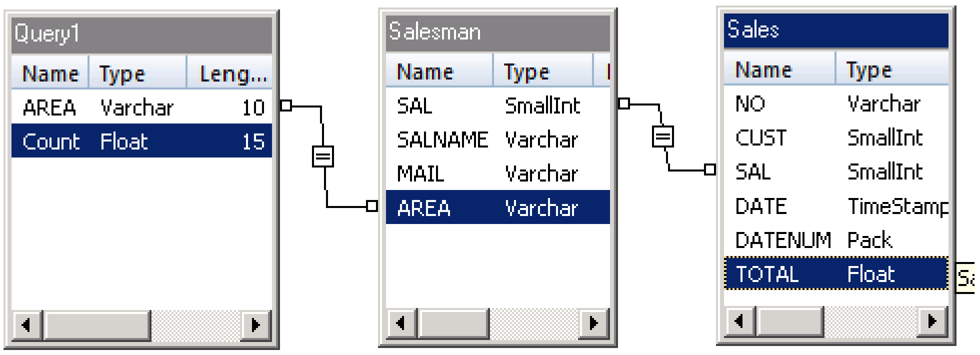
1) First, we get the number of salesmen by area in the query "Query1".



Name	Type
Formula	
✓ COUNT(*)	Float
Salesman	
SAL	SmallInt
SALNAME	Varchar
MAIL	Varchar
✓ AREA	Varchar

	AREA	Count
1	ATLANTIC	3
2	CENTRAL	4
3	NORTH-WEST	4
4	SOUTH	4
5	WEST	3

2) And then, we use the result of the first query for adding in a new query the total sales amount.



Query1

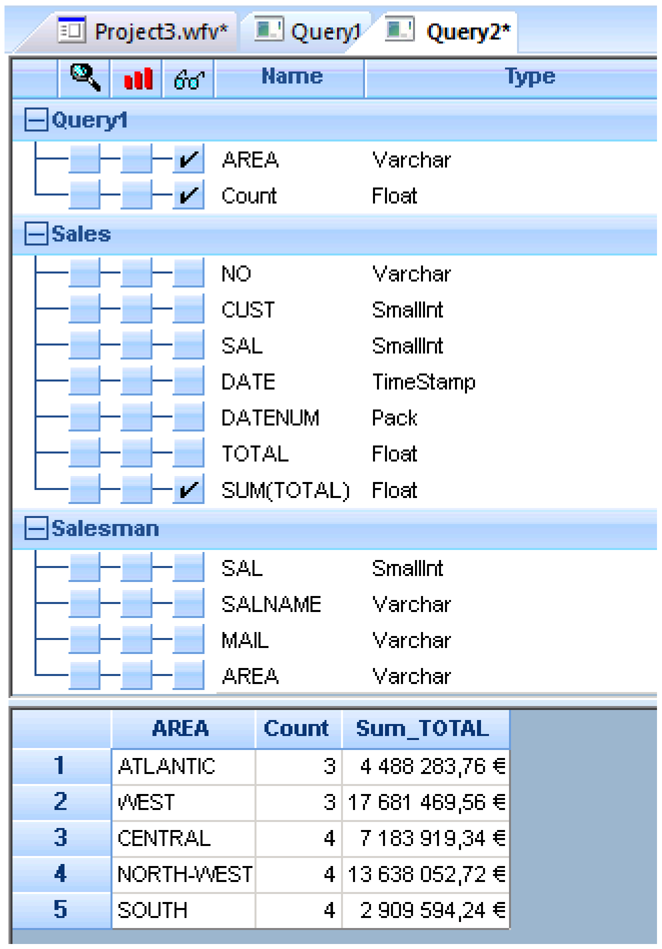
Name	Type	Leng...
AREA	Varchar	10
Count	Float	15

Salesman

Name	Type
SAL	SmallInt
SALNAME	Varchar
MAIL	Varchar
AREA	Varchar

Sales

Name	Type
NO	Varchar
CUST	SmallInt
SAL	SmallInt
DATE	TimeStamp
DATENUM	Pack
TOTAL	Float



	Name	Type
Query1		
✓	AREA	Varchar
✓	Count	Float
Sales		
	NO	Varchar
	CUST	SmallInt
	SAL	SmallInt
	DATE	TimeStamp
	DATENUM	Pack
	TOTAL	Float
✓	SUM(TOTAL)	Float
Salesman		
	SAL	SmallInt
	SALNAME	Varchar
	MAIL	Varchar
	AREA	Varchar

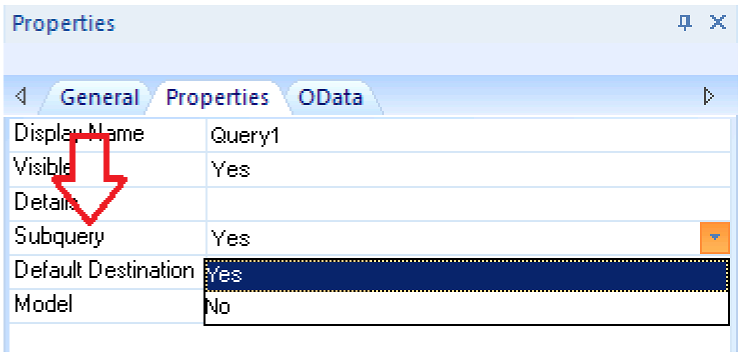
	AREA	Count	Sum_TOTAL
1	ATLANTIC	3	4 488 283,76 €
2	WEST	3	17 681 469,56 €
3	CENTRAL	4	7 183 919,34 €
4	NORTH-WEST	4	13 638 052,72 €
5	SOUTH	4	2 909 594,24 €

In previous versions of BAI, this was possible with the “Click and DECiDE Queries” data source which allows to use the queries from a Click & DECiDE project file (*.wfv) as a data source, especially useful for multi sources queries.
We can now do it directly in Builder with the main query based on the initial datasource.
This new feature has the following advantages:

- Easy and user-friendly way to do it: only one project including subquery and main query, no additional data source to be configured, easier to manage query parameters, etc.
- More efficient: only one query is run at database level (a global native SQL is built; it includes the subquery with a nested SELECT in the FROM clause).

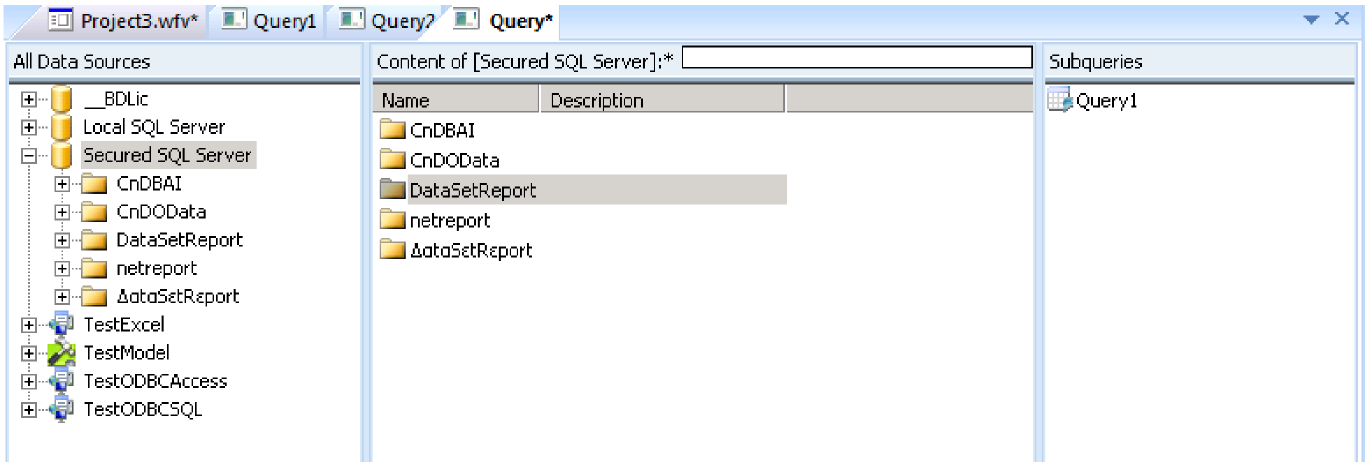
2. Subquery configuration

❖ A query can be used as a table if its property ***Subquery*** is enabled. Default value is *No*.



Display Name	Query1
Visible	Yes
Details	
Subquery	Yes
Default Destination	Yes
Model	No

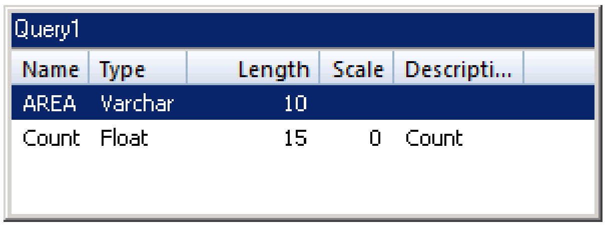
❖ When a query has the ***Subquery*** flag enabled, it is available in the new panel ***Subqueries*** in the ***Tables*** View:



When a data source is selected in the left panel ***All Data Sources***, only subqueries based on this datasource are shown in the right panel **Subqueries**.

The ***Subqueries*** panel is hidden when no subquery is available.

❖ Drag and drop a subquery onto the bottom panel or double-click on the subquery item for adding it as a new table in the query.
The subquery is then shown as a table, the columns of which correspond to the selected fields in the subquery:

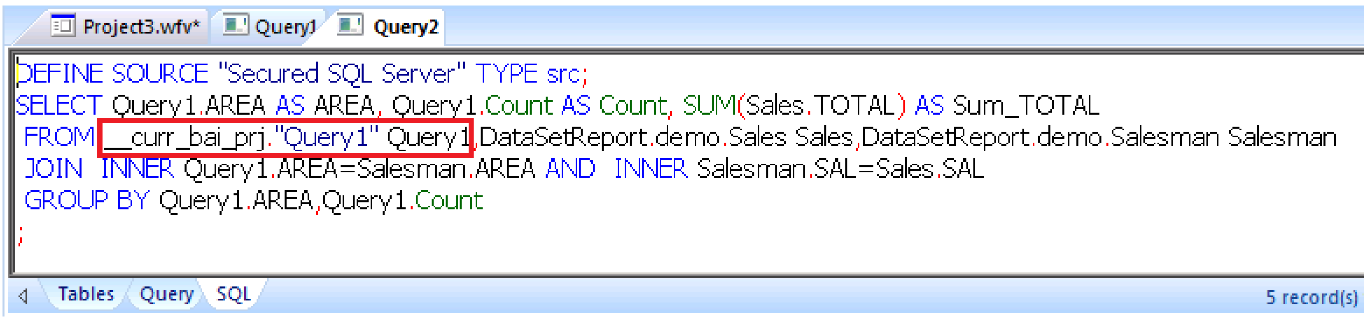

Query1

Name	Type	Length	Scale	Descripti...
AREA	Varchar	10		
Count	Float	15	0	Count

Column names are obtained from the subquery: it is either the column description if defined, either the column header if defined, otherwise it is the name of the column.

3. Click & DECiDE and native SQL (for advanced users)

❖ In the Click and DECiDE SQL, the subquery is stored as a table of the "schema" ***__curr_bai_prj*** (= the current project):



```
DEFINE SOURCE "Secured SQL Server" TYPE src;
SELECT Query1.AREA AS AREA, Query1.Count AS Count, SUM(Sales.TOTAL) AS Sum_TOTAL
 FROM __curr_bai_prj."Query1" Query1,DataSetReport.demo.Sales Sales,DataSetReport.demo.Salesman Salesman
 JOIN  INNER Query1.AREA=Salesman.AREA AND  INNER Salesman.SAL=Sales.SAL
 GROUP BY Query1.AREA,Query1.Count
;
```

❖ The native SQL built for run includes the subquery as a nested SELECT in the FROM clause:

SELECT Query1.AREA AREA,Query1.Count Count,SUM(Sales.TOTAL) Sum_TOTAL
FROM (**SELECT Salesman.AREA AREA,COUNT(*) Count**
FROM DataSetReport.demo.Salesman Salesman GROUP BY Salesman.AREA) Query1
INNER JOIN DataSetReport.demo.Salesman Salesman ON Query1.AREA = Salesman.AREA
INNER JOIN DataSetReport.demo.Sales Sales ON Salesman.SAL = Sales.SAL
GROUP BY Query1.AREA,Query1.Count

4. Parameters

Parameters of a subquery work like for a standard query:

❖ Parameters of type *Value*, *Formula*, *Query*, *User Property*: the value is obtained/calculated at runtime.
❖ Parameters of type *Input*, *Input List*, *Input Query List*: the user is prompted to enter the value.

5. Limitations

❖ ORDER BY is not supported in a subquery, except if it contains a TOP. This is a limitation at the database level for a nested SELECT in the FROM clause.
❖ Click and DECiDE formula are of course not supported in subqueries, because they are calculated after SQL execution.