Aggregations

An aggregation is a data provider which represents a database column that is aggregated over a set of records. At design-time aggregations have the following properties:

- **Name** - The name by which the aggregation will be available as a data provider throughout the solution in which it is declared, as well as any modules containing it.
- **Type** - There are five types of aggregations:
  - **Count** - The number of records in an entire foundset containing a non-null value for a column
  - **Sum** - The sum of all the values for numeric column in an entire foundset
  - **Minimum** - The smallest value for a numeric column in an entire foundset
  - **Maximum** - The largest value for a numeric column in an entire foundset
  - **Average** - The average value for a numeric column in an entire foundset
- **Column** - The column containing values that are aggregated

At runtime, aggregations are computed in the context of a foundset. The value is derived from a SQL query, which takes the form of a SQL Aggregate Function and appends the WHERE clause used by the foundset's query.

Example

Assume an aggregation, *record_count*, declared on the *customer* table. The aggregation type is *count* and the column is *customerid*. The aggregation is available for any foundset based on the *customers* table.

```
function printRecordCcount(event) {function printRecordCcount(event) {
        application.output(record_count);               // print record count before find
        if(foundset.find()){                               // Find all customers where city starts with 'B'
                city = 'B%';
                foundset.search();
                application.output(record_count);       // print the record count after find
        }
}
```

After the find, the aggregation is re-queried using the foundset's new WHERE clause.

```
SELECT COUNT(customerid) AS record_count FROM customers WHERE city LIKE ? LIMIT ?
```

Performance

Because aggregations are derived from SQL queries, they may not reflect data changes, seen in the client, but not yet committed to the database. Aggregations will refresh after outstanding changes are committed.

SQL Aggregate Functions may be expensive operations, depending on the size and structure of a database table and the nature of the aggregation. Developers are encouraged to use discretion when working with aggregations. For example, when an aggregation is shown in a table-view form, it may result in a query for each record in displayed on the form, and performance may degrade.