Algorithm using hypercube for aggregations
MIRELA-CATRINEL VOICU, GABRIELA MIRCEA
Faculty of Economic Sciences
West University of Timișoara
ROMANIA

Abstract: - For \( n \) fields (used for grouping) from a database, we can obtain \( 2^n \) aggregation types - the maximal set possible. In this paper we are focused on presenting an algorithm with which the user can obtain any subsets of aggregations types from the maximal set. Our algorithm supposes that, before inserting records in tables, we know the desired aggregation types. We present our algorithm for a snowflake scheme, using a fact table and some dimension tables. Our algorithm can be easily extended to a constellation scheme.

Key-Words: aggregated value sets, hypercube, star (snowflake, constellation) scheme

1 Introduction
Data analysis is used in more departments or sectors like finance departments, marketing departments, manufacturing sector, sales departments etc. Data analysis applications typically aggregate data across many dimensions (\( n \geq 0 \)). For aggregations, many tools are known. We recall (as example) some from these:

An SQL aggregate function (\( AF \)) produces one answer:
\[
\text{Select } AF \text{ (attribute\_value) from table which corresponds to one aggregation type.}
\]

An SQL aggregate function (\( AF \)) and the \( \text{Group by} \) operator produce also one answer:
\[
\text{Select attribute\_1,…,attribute\_n, } AF \text{ (attribute\_value) from table group by attribute\_1,…,attribute\_n which corresponds to one aggregation type.}
\]

The Rollup operator (from Oracle) – corresponds to \( n+1 \) aggregation types.

The Cube operator – corresponds to \( 2^n \) aggregation types (the maximal set possible).

In the case in which \( n \) is not small, \( 2^n \) is a considerable value. In the case in which the user wants to obtain (in the same result table) other subsets of aggregated values than the sets given by the known tools, we propose one algorithm.

In the beginning, we consider a database with a star or snowflake scheme. We suppose that the user wants to obtain one result table (containing aggregated values) or many results tables having the same header. In the last case, the difference between result tables is given by the used aggregation types.

The algorithm presented in this paper supposes that, before introducing the records in tables, we will specify: the fact table, dimension tables, the relationships of the tables used for aggregations and the aggregation types (see Fig. 6 and Fig.8).

In the moment in which we will introduce a new record in the fact table, we also will construct a hypercube (see Fig. 11). Its records will be formed as the set of all subsets obtained from the fields used for grouping and the fields used for aggregations, corresponding to the new record (from the fact table) and the records from the dimension tables corresponding to the specified relationships.

From this hypercube we will save in certain tables (additionally used in database) only the records which form the sets which provide the values used for the specified aggregation types (see Fig. 12). After this moment the hypercube will be deleted. From these last additional tables, using sample SQL statements and the \( \text{Group By} \) operator we will obtain the desired result tables (containing the aggregated values), in a very short time as in Fig. 13.

At this moment, we recall how we want to refer to the sets of aggregation types (see [16],[17]). In order to specify the aggregation types, we propose that the user make specifications, which contain combinations of “\( m \)” and/or “\( f \)” and/or “\( u \)”, where:

\( f \) – means one field used for grouping,
\( u \) – means one field not used for grouping,
\( m \) – means zero, one or more fields not used for grouping.

![Fig. 1 An initial table](image)

Now, we consider the table presented in Fig. 1.
Here, the fields field1, field2, field3, field4, field5 form the maximal set used for grouping and the field fvalue is used for aggregation.

The specification m f m produces the results presented in Fig. 2 (which correspond to five aggregation types).

The specification m f u f m produces the results presented in Fig. 3 (which correspond to three aggregation types).

The specification f m f m produces the results presented in Fig. 4 (which correspond to four aggregation types).

In such specifications we can also eliminate some fields for a certain field.

The user must specify the n fields used for grouping. Using specifications, which are composed of “f” or/and “m” or/and “u”, the user can obtain any wanted subsets of aggregation types for the n specified fields.

Now, we will present our algorithm. The implementation is made in a programming environment (we work here, for example, in Delphi) and with a database (here we use databases from Access).

2 Specification of the aggregation types

2.1 Create a table

In the moment in which we create a table, we save some data (referring to all the fields of the new table) in an additional table (see Fig. 5). The additional table from the Fig. 5 contains the following fields:

- The field table_name contains the name of the tables from database.
- The field field_name contains the name of the fields from database.
- The field data_type contains the data type of each field from database.
- The field t refers to the name of an additional table corresponding to a work table from the database.
- The field tf represents a code for the unique identification of each field from the database.

When we create a new table, the corresponding table (its name is given by application and it is saved in the field t from Fig. 5) is also created. For each field from the initial table it has as fields: the field code (given by tf from Fig. 5) and the field value (for an example of such a table, see Fig. 5, Table 2 and Fig. 10). In addition, we have a field norecord for the number of each record in the initial table. This last field also exists in the initial tables and it is generated automatically by application.

In the moment in which we have created all the tables, we will pass to the following step: we will specify the fields used for grouping, for aggregation and the relationships between tables.

2.2 Tables, fields, relationships, aggregation functions, fact table

Now, we consider a database with seven tables (see the Table 1, i between 1 and 7), each table with six fields. The field norecord is used to generate automatically the number of a record from a table, for its unique identification.

The user must specify the tables, fields, relationships, aggregation functions like in Fig. 6. In Fig. 6 we must follow these steps:
1 – select the used tables;

<table>
<thead>
<tr>
<th>Table_name</th>
<th>field_name</th>
<th>data_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>tab1</td>
<td>norecord</td>
<td>Integer</td>
</tr>
<tr>
<td>tabi</td>
<td>f1 PK</td>
<td>text(10)</td>
</tr>
<tr>
<td>tabi</td>
<td>f2</td>
<td>text(10)</td>
</tr>
<tr>
<td>tabi</td>
<td>f3</td>
<td>text(10)</td>
</tr>
<tr>
<td>tabi</td>
<td>f4</td>
<td>text(10)</td>
</tr>
<tr>
<td>tabi</td>
<td>f5</td>
<td>Integer</td>
</tr>
</tbody>
</table>

Table 1 Database structure

2 – select the fields used for grouping (in order in which they form the header for the result table – these fields will be indexed from 0 to \(n-1\), where \(n\) represents the number of used fields);

3 – here the user must specify the used tables, the relationships and the fact table.

4 – this step is used to allow the user to introduce the aggregation functions (one or more);

5 – at this step many tables will be created:

5.1. - a table which contains, for each field used for grouping, the following information: the field name, its table name and its field index.

5.2. - a table which contains information on the fields used for grouping (initial and additional tables).
- a table which contains information on the fields used for aggregation (initial and additional tables).

For each field used for grouping or aggregation, in these new tables we have a corresponding field in which we save as records: the field name, its table name, the field code, the name of the additional table (see Fig. 5). We present such a table in Fig. 7 (this table contains information on fields used for aggregation).

5.3. – we will have a table (for the additional tables) which contains information on the relationships specified in Fig. 6.

5.4. – a table which contains the additional table name, corresponding to the fact table.

2.3 Specification of the aggregation types

Now, the user must specify the sets of aggregation types in the following way (see Fig.8):

1 – “prepare” the form for a new specification of aggregation types (clear some components on the form);

2 – selection of \(m, f, u\) in the desired way;

3 – the user must select each \(f\) (from CheckListBox), step by step. For a selected \(f\), with a click on the field name, the user can eliminate the field (all possible fields are displayed in a ListBox, like in Fig. 8), which will not be used. In this moment a table will be created, which contains the field index, the table name and the field name for all selected fields for the corresponding \(f\).

In the moment in which we have the table for each \(f\) from a specification (see Step 3), we can pass to the following step:

4 – with a click on the command button “create aggregation type table” we will obtain two tables (presented in Fig. 9), which contain as records the aggregation types (for the initial and additional tables). Here we have a cartesian product between the
records from the tables corresponding to each \( f \). Using the indexes, we can formulate conditions for \( where \) clauses (according to the presence of \( m \) or \( u \) at the left or right of each \( f \)).

In Fig. 9, in the left table, the first record - “tab3f31, tab3f32” means that for grouping we will have the fields \( f31 \) and \( f32 \) from the table \( tab3 \).

In Fig. 9, in the right table, the first record - “t3f1t3f2t1f5t3f5” means that for grouping we will have the fields \( t3f1 \) and \( t3f2 \) from the table \( t3 \). We use “t1f5t3f5” corresponding to the fields used for aggregations (\( t1f5 \) from \( t1 \) and \( t3f5 \) from \( t3 \)).

Now, we repeat the steps 1-4 until the moment when we will have specified all aggregation types.

3 Insert records

We consider the records presented in Table 2. Here \( i \) is between 1 and 7.

<table>
<thead>
<tr>
<th>Table i</th>
<th>f1</th>
<th>f2</th>
<th>f3</th>
<th>f4</th>
<th>f5</th>
</tr>
</thead>
<tbody>
<tr>
<td>tab1</td>
<td>a1</td>
<td>b1</td>
<td>c1</td>
<td>d1</td>
<td>1</td>
</tr>
<tr>
<td>tab2</td>
<td>a2</td>
<td>a4</td>
<td>a5</td>
<td>b2</td>
<td>1</td>
</tr>
<tr>
<td>tab3</td>
<td>a1</td>
<td>a2</td>
<td>a3</td>
<td>b3</td>
<td>1</td>
</tr>
<tr>
<td>tab4</td>
<td>a1</td>
<td>a2</td>
<td>a3</td>
<td>b4</td>
<td>1</td>
</tr>
<tr>
<td>tab5</td>
<td>a3</td>
<td>a6</td>
<td>b4</td>
<td>c4</td>
<td>1</td>
</tr>
<tr>
<td>tab6</td>
<td>a4</td>
<td>b5</td>
<td>c5</td>
<td>d5</td>
<td>1</td>
</tr>
<tr>
<td>Tab7</td>
<td>a5</td>
<td>b6</td>
<td>c6</td>
<td>d6</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2 Records from tables

At this step we must consider two situations. The first situation is when we insert a new record in a different table than the fact table (i.e. in a dimension table). In this moment, in the corresponding additional table, we will add two records, like in Fig. 10 (\( t1 \) is the additional table corresponding to \( tab1 \) – see also Fig. 5).

The first record is (”", null, "", null, "", null, "", null, "", null, no_record) and the second record is (field_1_code, field_1_value, field_2_code, field_2_value, field_3_code, field_3_value, field_4_code, field_4_value, field_5_code, field_5_value, no_record).

The second situation is when we insert a new record in the fact table. Here we also insert the corresponding records in the additional table, but in addition we will make the following three operations:

1 - Using the table (see also step 5.3 from Subsection 2.2) which corresponds to the relationships from Fig. 6 (relationships on the additional tables), we extract from the additional tables (see field \( t \) from Fig. 5 and Fig. 10) the records which correspond to all the relationships. For each table used in the relationship specified in Fig. 6, we will obtain only two records (like the records from Fig. 10).

For the case presented in Fig. 6, we will obtain seven such tables corresponding to the new record from the fact table and to the records from the dimension tables which will respect all relationships.

2 - Now, we will generate a hypercube – a table which contains all subsets of the set formed with twelve fields (ten for grouping and two for aggregation), using:

- the last tables (which have the same structure as the table presented in Fig. 10),
- the table which contains information on the fields used for grouping (ten fields in our example presented in Fig. 6; see step 5.2 from Subsection 2.2),
- the table which contains information on the fields used for aggregation (two fields in our example presented in Fig. 6; see step 5.2 from Subsection 2.2).

In addition, the hypercube contains a field \( f \) which has as values a concatenation of all code fields corresponding to the each record (see Fig. 11). This field contains, in fact, all possible aggregation types using the twelve fields.
3 – Here we use the hypercube and the tables which contain the aggregation types, like in the right table from Fig. 9. For each specification of aggregation types, we will create a table which contains the corresponding values from hypercube. For example, for the specification $m f m$ we will obtain the table presented in Fig. 12. When we have constructed all these tables, we will delete the hypercube. These tables (like the table from Fig. 12) contain only the records which are necessary in order to obtain the aggregated values for a certain specification (like $m f m$ in Fig. 8).

![Fig. 11 Hypercube structure](image)

In Fig. 12, the field $nti$ (here $i$ is between 1 and 7) represents the record number from the table $ti$ (see Fig. 10). Using theses fields, when we modify or delete a record in the initial tables (like table), the same changes affect the corresponding additional table and the tables like the one in Fig. 12.

![Fig. 12 Table corresponding to a specification of aggregation types (here, $m f m$ from Fig. 8)](image)

4 Final result tables

Using the tables like the one presented in Fig. 12 (which provide the values corresponding to a specification of aggregation types), and the table from Fig. 7, now, we can obtain the final results. We will use, as example, the form presented in Fig. 13.

![Fig. 13 Creating result tables](image)

Here we can select if we want only one table which contains the results corresponding to all aggregation types (for which we have selected the corresponding checkbox), or we can create a result table for each specification. For the last case, in Fig. 14, we present such a result table for the specification $m f m$.

![Fig. 14 Result table (here, corresponding to specification $m f m$ from Fig. 8)](image)

5 Conclusions

Using this algorithm we can easily obtain any subsets of aggregation types, from the maximal possible set.

We have presented the algorithm for a snowflake scheme. This algorithm is the same for a star
scheme. If we want to use a constellation scheme or we use aggregation types when the result tables have different headers, we must repeat all our constructions (see Figs. 6-9) separately for each case and we must use them when we introduce new records in fact table (see Figs. 11, 12).

Our algorithm allows us to obtain any set of aggregation types in a very short period of time, but it requires us to specify all the aggregation types, before inserting the records in the tables.

References:
[12] Pinto. H, Han J., Pei J. et al. – Multi-dimensional Sequential Pattern Mining – Information and Knowledge Management (CIKM ’01), Atlanta, USA, November 2001
[17] Voicu M., Mircea G. - Algorithm for obtaining aggregated value sets from multidimensional databases – Conference Proceedings of 5th WSEAS International Conference on APPLIED INFORMATICS and COMMUNICATIONS (AIC 05), Malta, September 15-17, 2005