Algorithms used to obtain aggregated value sets from relational databases

MIRELA-CATRINEL VOICU
Faculty of Economic Sciences,
"Nicholas Georgescu-Roegen" Interdisciplinary Platform
West University of Timişoara
ROMANIA

Abstract: - For n fields used for grouping we can obtain 2^n aggregation types. When n is not small, 2^n is a considerable value. With a new model for the specification of aggregation types, we present ways in which we can obtain any subsets of aggregation types, starting from n fields used for grouping. The algorithms concern cases in which:
- The specification of aggregation types is made after we insert the data in the tables.
- The specification of aggregation types is made before we insert the data in the tables. In order to obtain the results really fast, we also use hypercubes.
- The case of one table (the fields used for grouping and for aggregation are from the same table) where we use one hypercube and we can specify the aggregation types in any moment (after or before the data insertion in the table) - model which can be used for questionnaires.
These algorithms can be used in many economic analyses, in order to obtain complex reports. We will also present an example which concerns the sales in a supermarket chain. In our study we have worked with databases from *Access* and we have made the algorithm implementation in *Delphi*, but the implementation can also be made in other programming environments and we can also use other relational databases.

Key-Words: aggregated value sets, hypercube, economic analyses, relational databases, programming environment

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>=0). For aggregations, many tools are known. We recall some from these:

An *SQL* aggregate function (*AF*) produces one answer:
Select AF (attribute_value) from table
which corresponds to one aggregation type.

An *SQL* aggregate function (*AF*) and the *Group by* operator produce also one answer:
Select attribute_1,…,attribute_n, AF (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 two algorithms.

In the beginning, we recall how we want to refer to the sets of aggregation types (see [6], [7]). 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.

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.

Table1 : Table

field1	field2	field3	field4	field5	fvalue
c11	c12	c13	c14	c15	1

Fig. 1 An initial table

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

r1 : Table

Table	Tabl	Tabl	Tabl	Tabl	mi
c11					1
	c12				1
		c13			1
			c14		1
				c15	1

Fig. 2 The result for *m f m*

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 *f*.

r1 : Table

Tab	Tabl	Tabl	Tabl	Table	mi
c11		c13			1
	c12		c14		1
		c13		c15	1

Fig. 3 The result for *m f u f m*

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.

r1 : Table

Tabl	Tabl	Tab	Tabl	Tab	mir
c11	c12				1
c11		c13			1
c11			c14		1
c11				c15	1

Fig. 4 The result for *f m f m*

Now, we will present our algorithms. 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 Algorithm in which, before inserting records in tables, we will specify the desired aggregation types

For starters, we consider a database with a 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 latter 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 *Group By* operator we will obtain the desired result tables (containing the aggregated values), in a very short time as in *Fig. 13*.

2.1 Specification of aggregation types

2.1.1 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:

Tables : Table

nr	t	tf	table_name	field_name	data_typ
1	t1	t1f1	tab1	f11	text(10)
2	t1	t1f2	tab1	f12	text(10)
3	t1	t1f3	tab1	f13	text(10)
4	t1	t1f4	tab1	f14	text(10)
5	t1	t1f5	tab1	f15	integer
6	t2	t2f1	tab2	f21	text(10)
7	t2	t2f2	tab2	f22	text(10)
8	t2	t2f3	tab2	f23	text(10)
9	t2	t2f4	tab2	f24	text(10)
10	t2	t2f5	tab2	f25	integer
11	t3	t3f1	tab3	f31	text(10)
12	t3	t3f2	tab3	f32	text(10)
13	t3	t3f3	tab3	f33	text(10)
14	t3	t3f4	tab3	f34	text(10)
15	t3	t3f5	tab3	f35	integer
16	t4	t4f1	tab4	f41	text(10)
17	t4	t4f2	tab4	f42	text(10)
18	t4	t4f3	tab4	f43	text(10)

Fig. 5 Data for a table

- 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 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.

2.1.2 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.

Table_name	field_name	data_type
tab***i***	norecord	Integer
tab***i***	f***i***1 PK	text(10)
tab***i***	f***i***2	text(10)
tab***i***	f***i***3	text(10)
tab***i***	f***i***4	text(10)
tab***i***	f***i***5	Integer

Table 1 Database structure

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;

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.

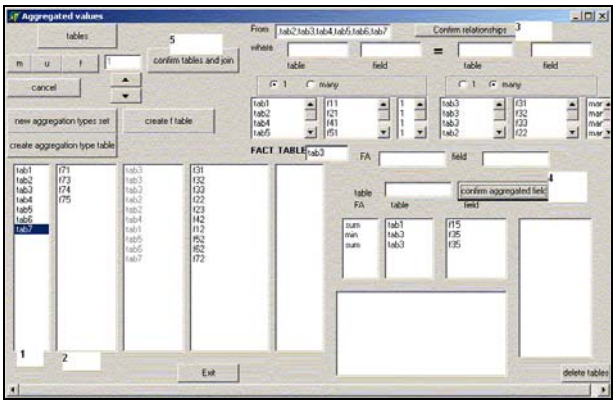
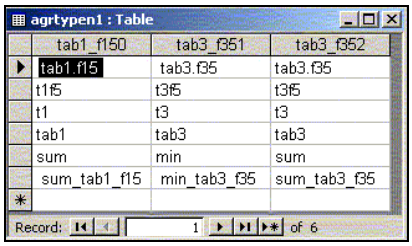


Fig. 6 Tables, fields, relationships, aggregation functions

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.*



agrtypen1 : Table

tab1_f150	tab3_f351	tab3_f352
tab1.f15	tab3.f35	tab3.f35
t1f5	t3f5	t3f5
t1	t3	t3
tab1	tab3	tab3
sum	min	sum
sum_tab1_f15	min_tab3_f35	sum_tab3_f35

Record: 1 of 6

Fig. 7 Information on fields used for aggregation

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

2.1.3 Specification of the aggregation types

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

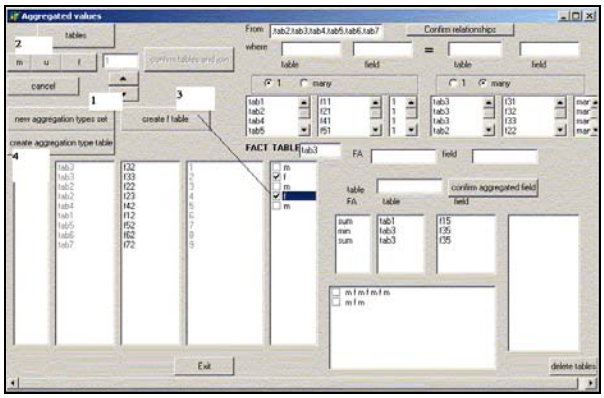


Fig. 8 Aggregation types

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*.

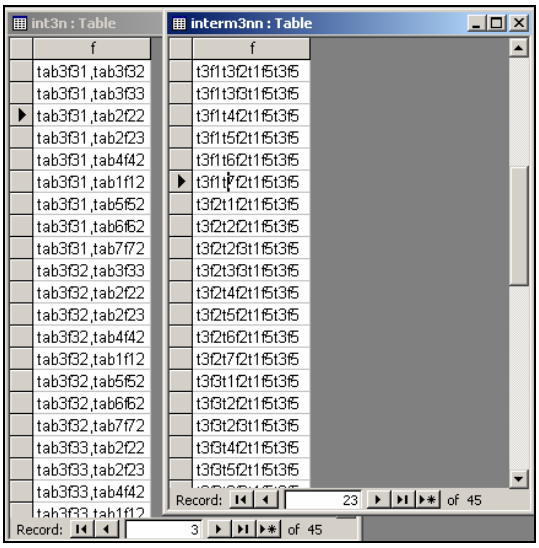


Fig. 9 Tables, which contain as records the aggregation types of the specification *m f m f m* from *Fig. 8* (the left table corresponds to the initial tables and the right table corresponds to the additional tables)

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.

2.2 Insert records

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

Table *i*	***fi*1**	***fi*2**	***fi*3**	***fi*4**	***fi*5**
tab1	a1	b1	c1	d1	1
tab2	a2	a4	a5	b2	1
tab3	a1	a2	a3	b3	1
tab3	a1	a2	a3	b4	1
tab4	a3	a6	b4	c4	1
tab5	a4	b5	c5	d5	1
Tab6	a5	b6	c6	d6	1
Tab7	a6	b7	c7	d7	1

Table 2 Records from tables

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:

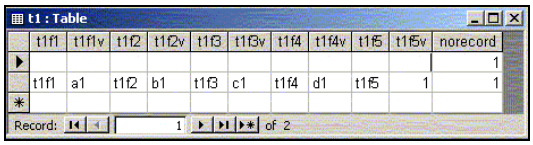


Fig. 10 Records in the additional table

1 - Using the table (see also *step 5.3* from *Sub-subsection 2.1.2*) 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 *Sub-subsection 2.1.2*),
- 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 *Sub-subsection 2.1.2*).

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.

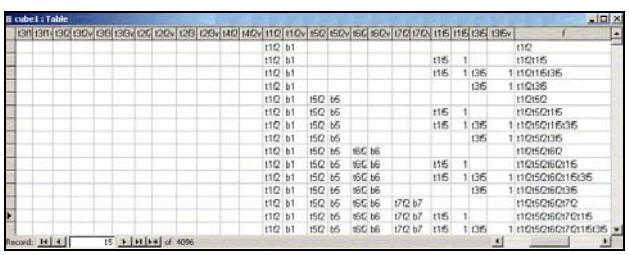


Fig. 11 Hypercube structure

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*).

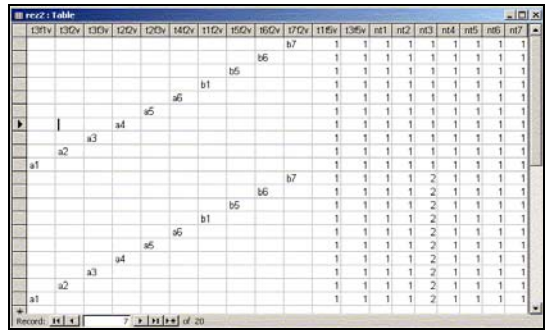


Fig. 12 Table corresponding to a specification of aggregation types (here, *m f m* from *Fig.8*)

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 tab***i***), the same changes affect the corresponding additional table and the tables like the one in *Fig. 12*.

2.3 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*.

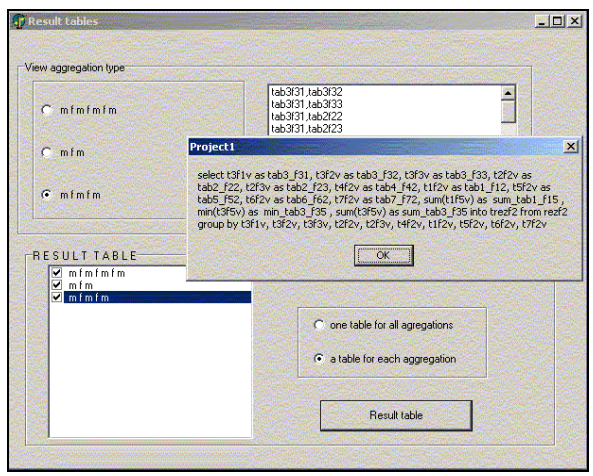


Fig. 13 Creating result tables

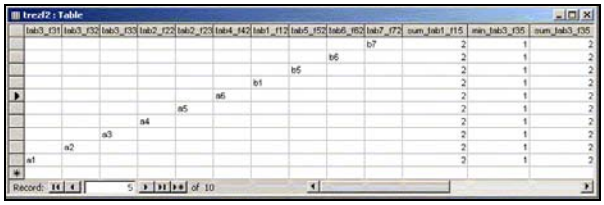


Fig. 14 Result table (here, corresponding to specification *m f m* from *Fig.8*)

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*.

3 One table, one hypercube and all aggregation types

In the case in which for aggregations we use fields from only a single table, we can construct a hypercube associated with this table, in order to specify the aggregation types in any moment (i.e., also when we have records in table). We present in this section such an algorithm.

3.1 Create a table and insert records

3.1.1 Create a table

In the moment in which we create a new table, we save some data (referring to all the fields of the new table) in an additional table (as the table from *Fig. 5*).

This additional table 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 the additional table (hypercube) corresponding to a table from the database. The corresponding table hypercube has the same structure as the table from the *Fig.10* (see also *Fig.15*).
- The field *tf* represents a code for the unique identification of each field from the database.

3.1.2 Insert a record

Now, we consider, as example, a table named *tab1* which contains twelve fields (named ***f1i,*** where *i* is between 1 and 12). We consider the first ten fields with *text(10)* data type and the last two fields with *integer* data type. Automatically, in each new table, the application adds a new field, named *norecord*, which is used for the unique identification of each record in table.

In table *tab1* we insert the following record: ("a","b","c","d","e","f","g","h","i","j",1,1). When we insert a new record, the record number will be automatically generated in the initial table and in hypercube table.

In this moment we will also insert in the table *cube_t1f1* (the additional table hypercube associate to the table *tab1*) the corresponding hypercube of the new record (see *Fig.15*). Our record is a set with twelve elements and its hypercube contains the set of all subsets formed with these twelve values.

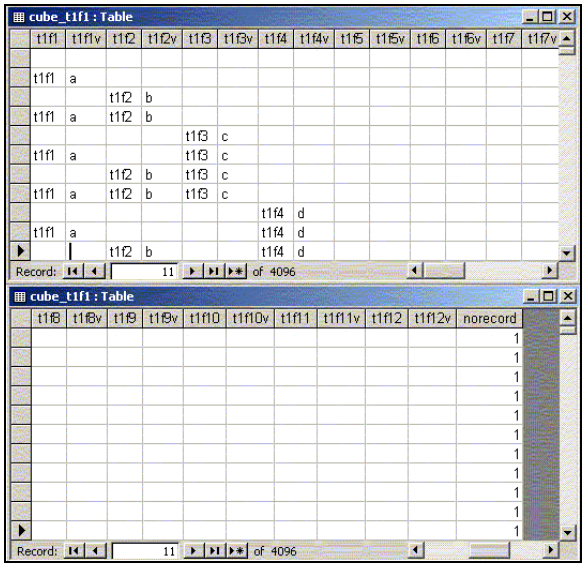


Fig. 15 Records of hypercube

For each new record in the initial table *tab1*, we will add its hypercube in the corresponding table *cube_t1f1*. When we want to delete or modify a record from the initial table (here *tab1*), using the field *norecord*, we will make the same operation in its hypercube.

3.2 Specification of aggregation types

3.2.1 Table, fields, aggregation functions

The user must specify the tables, fields, relationships, aggregation functions as in *Fig. 16*. In this figure we must follow these steps:

1 – select the used table;

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);

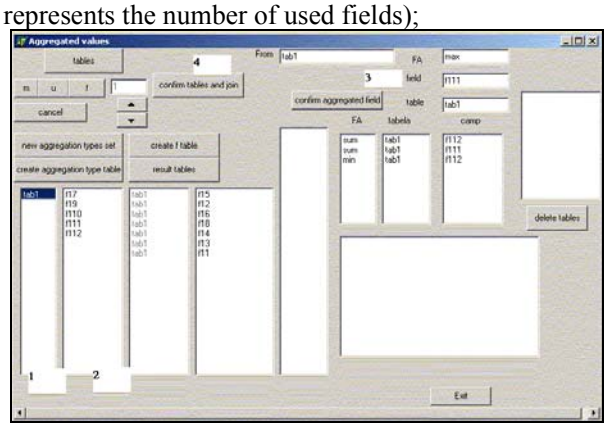


Fig. 16 Table, fields, aggregation functions

3 – this step is used to allow the user to introduce the

aggregation functions (one or more);

4 – at this step many tables will be created:
4.1.- a table which contains, for each field used for grouping, the following information: the field name, its table name and its field index.

4.2.- a table which contains information on the fields used for grouping (initial and additional table).
- a table which contains information on the fields used for aggregation (initial and additional table).
- a table which contains information on the fields which will not be used for grouping or for aggregation (initial and additional table);

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, as example, *Fig. 7*).

3.2.2 Aggregation types

Now, the user must specify the sets of aggregation types in the following way:

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. 17*), 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*.

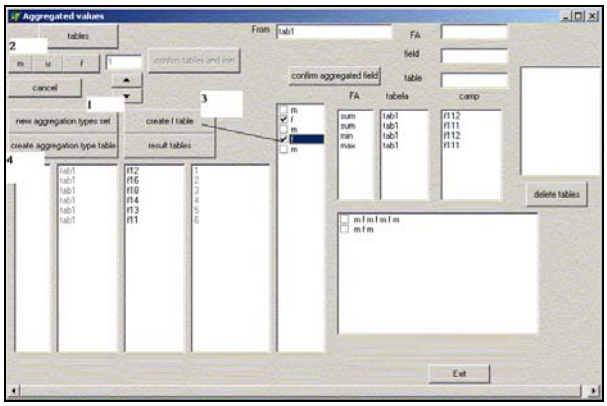


Fig. 17 Aggregation types

In the moment in which we have the table for each *f* from a specification (see *Fig. 17*), we can pass to the following step:

4 – with a click on the command button "create aggregation type table" we will obtain a table (see, as example, such tables in *Fig.9*), which contains as records the aggregation types (for the additional tables). Here we have a cartesian product between the records from the tables corresponding to each *f* from a specification of aggregation types.

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

3.3 Final result tables

We can select if we want only one table which contains the results corresponding to all aggregation types (see, as example, *Fig.13*), or we can create a result table for each specification.

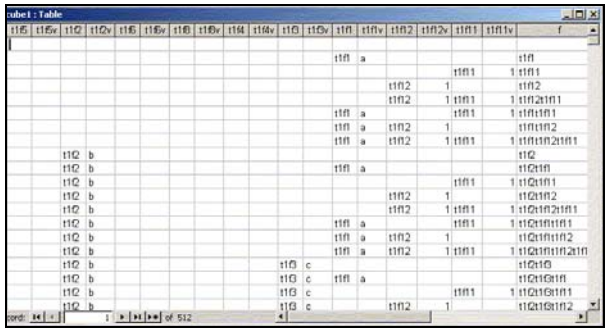


Fig. 18 Sub-hypercube used for aggregations

For the last case, in *Fig. 20*, we present such a result table for the specification *m f m f m* from *Fig.17*.

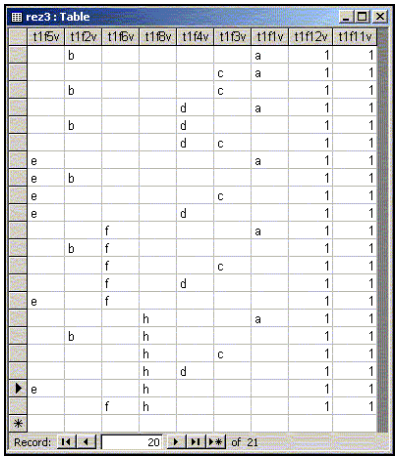


Fig. 19 The set of records which provide the values used for an aggregation type specification (here, *m f m f m* from *Fig.17*)

In both cases, first, we will extract from hypercube (see *Fig.18*) the set which corresponds to the fields used for grouping and for aggregation (see *step 4.2* from *Sub-subsection 3.2.1*). In addition, we will have a new field - *f* - which has as values a concatenation of all code fields corresponding to the each record. This field

contains, in fact, all possible aggregation types using the nine fields (seven for grouping and two for aggregation). The table from the *Fig.18* is in fact a sub-hypercube of the hypercube from *Fig.15* (corresponding only to the nine fields: seven used for grouping and two used for aggregation).

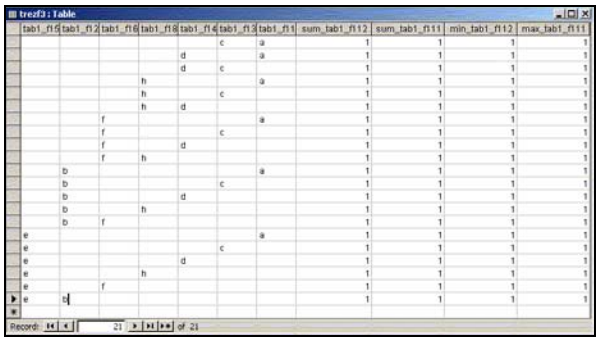


Fig. 20 Result table (here, corresponding to the specification *m f m f m* from *Fig.17*)

From the sub-hypercube from *Fig.18*, we will extract only the records which provide the values used for an aggregation type specification. For each specification of aggregation types we will have such a table, like one presented in *Fig.19*. In the moment in which we have obtained all these last tables, we will delete the sub-hypercube.

With the tables like the one in *Fig. 19* and using simple *SQL* statements (see, as example, *Fig.13*), we will obtain the result tables (see *Fig.20*).

4 The algorithm used for any database

This algorithm supposes the construction of tables, which contain as records the aggregation types and the use of *SQL* statements (by application) for each such record. This means that if we have (for example) 100 aggregation types there are in use 100 *SQL* statements for aggregations.

We can make the connection with any database (here, from *Access*), because additional constructions on the databases are not necessary.

We consider, for example, the tables presented in *Figure 21*.

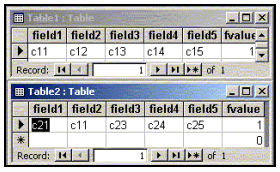


Fig. 21 Initial tables

4.1 Tables, fields, relationships, aggregation functions

The user must specify the tables, fields, relationships, aggregation functions like in *Figure 22*. In this figure we must follow these steps (like in *Sub-subsection 2.1.2*):

1 – select the used tables;

2 – select the fields used for grouping (in order in which they form the header for the result table – these fields will be indexed);

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

4 – here the user must specify the tables used and (if necessary) the relationships. Here a table will be created, which has as record the fields (and the corresponding indexes) used for grouping (see the *Figure 23*).

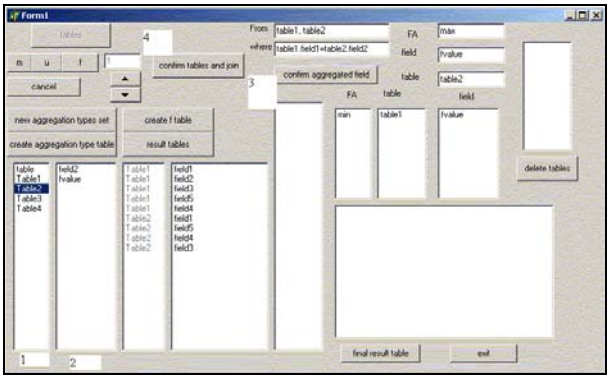


Fig. 22 Tables, fields, relationships, aggregation functions

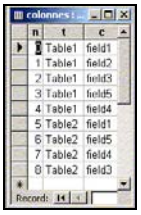


Fig. 23 Indexes for the fields used for grouping

4.2 Specification of aggregation types

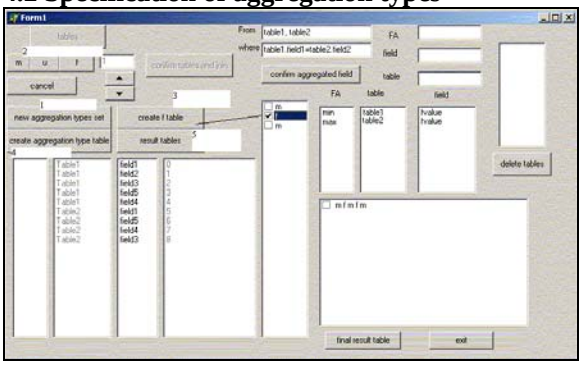
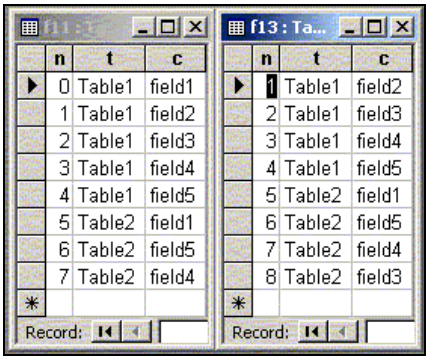


Fig. 24 Specification of aggregation types

Now, the user must specify the sets of aggregation types in the following way (like in *Sub-subsection 2.1.3*):

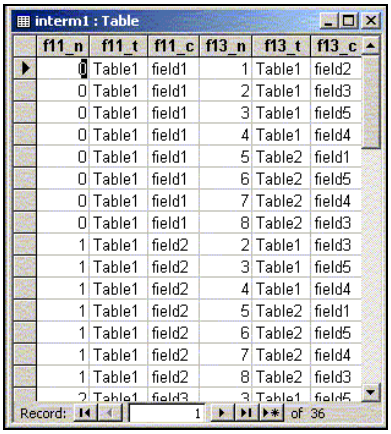
1 – "prepare" the form for a new specification of aggregation types (clear some components on the form);
2 – selection of *m, f, u* in the desired way;
3 – the user must select each *f* (from *CheckListBox*), step by step. For a selected *f*, with a click on the field name, he can eliminate the field (all possible fields are displayed in a *ListBox*, like in *Figure 24*), which will not be used. At 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*. We present such tables in *Figure 25* (for the specification *m f m f m*).



n	t	c
0	Table1	field1
1	Table1	field2
2	Table1	field3
3	Table1	field4
4	Table1	field5
5	Table2	field1
6	Table2	field5
7	Table2	field4

n	t	c
1	Table1	field2
2	Table1	field3
3	Table1	field4
4	Table1	field5
5	Table2	field1
6	Table2	field5
7	Table2	field4
8	Table2	field3

Fig. 25 The corresponding tables for each *f* from the specification *m f m f m*



interm1 : Table

f11_n	f11_t	f11_c	f13_n	f13_t	f13_c
0	Table1	field1	1	Table1	field2
0	Table1	field1	2	Table1	field3
0	Table1	field1	3	Table1	field5
0	Table1	field1	4	Table1	field4
0	Table1	field1	5	Table2	field1
0	Table1	field1	6	Table2	field5
0	Table1	field1	7	Table2	field4
0	Table1	field1	8	Table2	field3
1	Table1	field2	2	Table1	field3
1	Table1	field2	3	Table1	field5
1	Table1	field2	4	Table1	field4
1	Table1	field2	5	Table2	field1
1	Table1	field2	6	Table2	field5
1	Table1	field2	7	Table2	field4
1	Table1	field2	8	Table2	field3

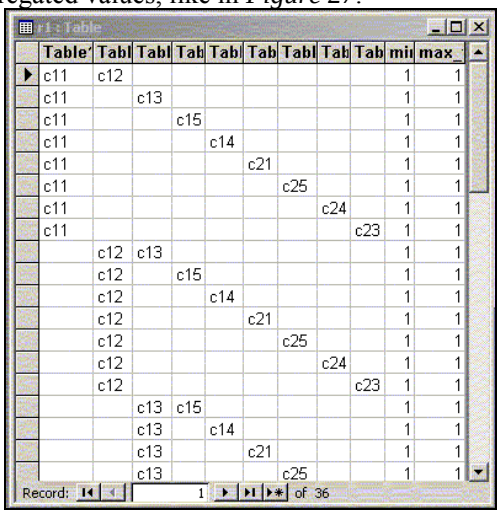
Fig. 26 A table, which contains as records the aggregation types

When we have the table for each *f* from a specification (see *Figure 25*), we can pass to the following step:

4 – with a click on the command button "create aggregation type table" we will obtain a table, which contains as records the aggregation types (see *Figure 26*). Here we have a cartesian product between the records from the tables corresponding at each *f* (presented in *Figure 25*). Using the indexes, we can formulate conditions for *where* clauses (according to the presence of *m* or *u* at left or right of each *f*).

Now, we repeat the step 1-4 until the moment when we will have specified all aggregation types. We will obtain the results at the following step (see *Figure 24*):

5 – with a click on the command button "result tables", for each table like the table presented in *Figure 26* (these tables correspond to each specification of aggregation types), we will construct a corresponding result table, which contains the aggregated values, like in *Figure 27*.



r1 : Table

Table	Tabl	Tabl	Tab	Tabl	Tab	Tabl	Tab	Tab	min	max_
c11	c12								1	1
c11		c13							1	1
c11			c15						1	1
c11				c14					1	1
c11					c21				1	1
c11						c25			1	1
c11							c24		1	1
c11								c23	1	1
	c12	c13							1	1
	c12		c15						1	1
	c12			c14					1	1
	c12				c21				1	1
	c12					c25			1	1
	c12						c24		1	1
	c12							c23	1	1
		c13	c15						1	1
		c13		c14					1	1
		c13			c21				1	1
		c13				c25			1	1

Fig. 27 A result table corresponding to the specification *m f m f m* of aggregation types

For example, for the second record from the table *interm1* from *Figure 26*, the application executes the following *SQL* statement:

insert aggregated values in a result table
Insert into r1

the fields used for grouping
Select Table1.field1 as "Table1_field1", NULL as "Table1_field2", Table1.field3 as "Table1_field3", NULL as "Table1_field5", NULL as "Table1_field4", NULL as "Table2_field1", NULL as "Table2_field5", NULL as "Table2_field4", NULL as "Table2_field3 ",

the aggregated data
min(Table1.fvalue) as min_Table1_fvalue, max(Table2.fvalue) as max_Table2_fvalue

From Table1, Table2
the relationships
Where Table1.field1=Table2.field2
the fields used for grouping
Group by Table1.field1, Table1.field3

This means that for the table presented in *Figure 26*, the application executes 36 *SQL* statements.

For this step the time can be a problem. When the number of aggregation types is increasing, the necessary time for obtaining the results is also increasing.

From the start, we can create only one result table which will contains the aggregated values, but we prefer to create a separated table for each specification (like in *Figure 27*), because, in this way, we can easily obtain the results only for some specifications presented before (not for all of them), without repeating the algorithm. We can also eliminate some fields from the result table. We present this possibility in the following subsection.

4.3 Final tables

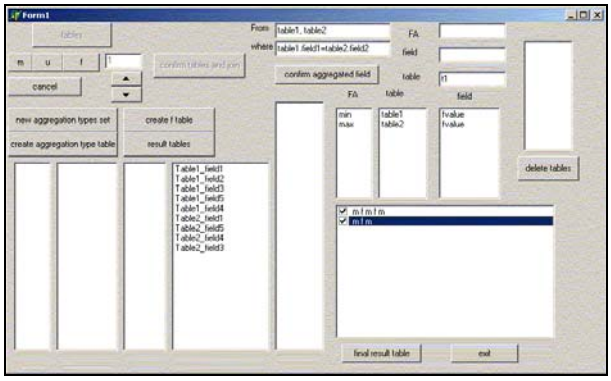


Fig. 28 Final confirmations for result tables

Fig. 29 A result table for aggregated values

We confirm the desired specifications of aggregation types like in *Figure 28*. With a click on the field name we eliminate the field from the result table.

For the case presented in *Figure 28*, we will obtain the result table presented in *Figure 29*.

5 An example which concerns the sales in a supermarket chain

Data analysis is used in more departments or sectors like finance departments, marketing departments, the manufacturing sector, sales departments etc. Data analysis applications typically aggregate data across many dimensions (n>=0). Many analyses can be very complex and can concern a considerable number of aggregation types.

In order to illustrate the importance of this subject and its practical applicability, in this paper we consider as example data corresponding to the Monoprix supermarket chain. For analysis, we use information from the web site www.monoprix.fr. The database structure corresponding to the supermarkets chain is presented in *Figure 30*.

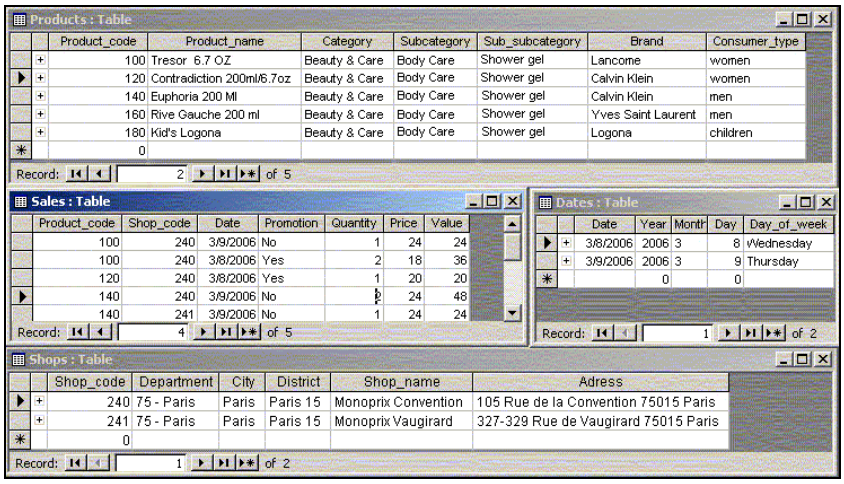
In the Monoprix supermarkets chain we find the following product categories: *Mode & Accessories* (subcategories: Skirts, Shirts, Tops, Accessories, Uenderwear, T-Shirt, Dress, Trousers, Jacket/Waistcoat &Winter coat); *Beauty & Care* (subcategories: Body Care, Face Care, Make-up); *House & Leisure* (subcategories: Pieces of furniture, Decoration Accessories, Kitchen Materials, Stationery); *Food/Maintenance* (subcategories: Soft drinks, Bread- pastry, Maintenance, Fruits, Vegetables, Breakfast, Cooked dishes, Fish, dairy product, Dry Products, Frozen, Meat, Alcoholic drinks, ice cream, sausages).

Starting from the database presented in *Figure 30*, we can make many analyses concerning the product sales. These analyses can concern the sales at country, department, city, district or shop level, etc. In order to simplify the presentation, we present a possible analysis type at shop level.

In order to easily observe the implementation of the algorithm from the *Section 4* and the result tables, we fix the values for the following fields: *Shop_name* (eg. Monoprix Convention), *Category* (eg. Beauty&Care), *Subcategory* (Body_Care) and *year* (eg. 2006). In this case we can consider for grouping, only the fields *Sub_subcategory, Brand, Consumer_type, Product_name, Promotion, Date, Month.* Now we present the types of aggregations and their specifications (see the *Table 3*).

We select the tables and the fields, we specify the relationships, criteria and the aggregation functions like in *Figure 31*.

We specify the aggregation types and for each *f* we confirm the fields which will be used for grouping like in *Figure 32*.



Products : Table

Product_code	Product_name	Category	Subcategory	Sub_subcategory	Brand	Consumer_type
100	Tresor 6.7 OZ	Beauty & Care	Body Care	Shower gel	Lancome	women
120	Contradiction 200ml/6.7oz	Beauty & Care	Body Care	Shower gel	Calvin Klein	women
140	Euphoria 200 Ml	Beauty & Care	Body Care	Shower gel	Calvin Klein	men
160	Rive Gauche 200 ml	Beauty & Care	Body Care	Shower gel	Yves Saint Laurent	men
180	Kid's Logona	Beauty & Care	Body Care	Shower gel	Logona	children
0						

Record: 2 of 5

Sales : Table

Product_code	Shop_code	Date	Promotion	Quantity	Price	Value
100	240	3/9/2006	No	1	24	24
100	240	3/8/2006	Yes	2	18	36
120	240	3/8/2006	Yes	1	20	20
140	240	3/9/2006	No	2	24	48
140	241	3/9/2006	No	1	24	24

Record: 4 of 5

Dates : Table

Date	Year	Month	Day	Day_of_week
3/8/2006	2006	3	8	Wednesday
3/9/2006	2006	3	9	Thursday
	0		0	

Record: 1 of 2

Shops : Table

Shop_code	Department	City	District	Shop_name	Adress
240	75 - Paris	Paris	Paris 15	Monoprix Convention	105 Rue de la Convention 75015 Paris
241	75 - Paris	Paris	Paris 15	Monoprix Vaugirard	327-329 Rue de Vaugirard 75015 Paris
0					

Record: 1 of 2

Fig. 30 Database structure, for a supermarket chain

Case 2.1. - f f m Sub_subcategory, Brand
Case 3.1. - f f m f m Sub_subcategory, Brand, Consumer_type Sub_subcategory, Brand, Product_name Sub_subcategory, Brand, Promotion Sub_subcategory, Brand, Date Sub_subcategory, Brand, Month
Case 4.1. - f f f m f m Sub_subcategory, Brand, Consumer_type, Product_name Sub_subcategory, Brand, Consumer_type, Promotion Sub_subcategory, Brand, Consumer_type, Date Sub_subcategory, Brand, Consumer_type, Month
Case 4.2. - f f u f m f m Sub_subcategory, Brand, Product_name, Promotion Sub_subcategory, Brand, Product_name, Date Sub_subcategory, Brand, Product_name, Month
Case 4.3. - f f u u f m f m Sub_subcategory, Brand, Promotion, Date Sub_subcategory, Brand, Promotion, Month
Case 5.1. - f f f f m f m Sub_subcategory, Brand, Consumer_type, Product_name, Promotion Sub_subcategory, Brand, Consumer_type, Product_name, Date Sub_subcategory, Brand, Consumer_type, Product_name, Month
Case 5.2. - f f f u f m f m Sub_subcategory, Brand, Consumer_type, Promotion, Date Sub_subcategory, Brand, Consumer_type, Promotion, Month
Case 5.3. - f f u f f m f m Sub_subcategory, Brand, Product_name, Promotion, Date Sub_subcategory, Brand, Product_name, Promotion, Month
Case 6 - f f f f f m f m Sub_subcategory, Brand, Consumer_type, Product_name, Promotion, Date Sub_subcategory, Brand, Consumer_type, Product_name, Promotion, Month

Table 3 Aggregation types

For obtaining result tables we can select all specifications of aggregation types or only some of them, like in *Figure 33*.

In the case in which we select all specifications of aggregation types, we will obtain the result presented in *Figure 34*.

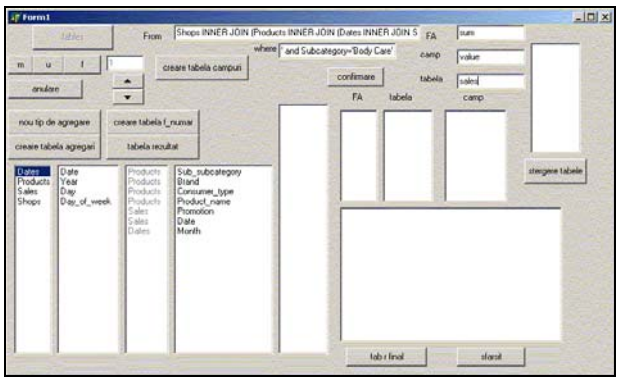


Fig. 31 Tables, fields, aggregation functions

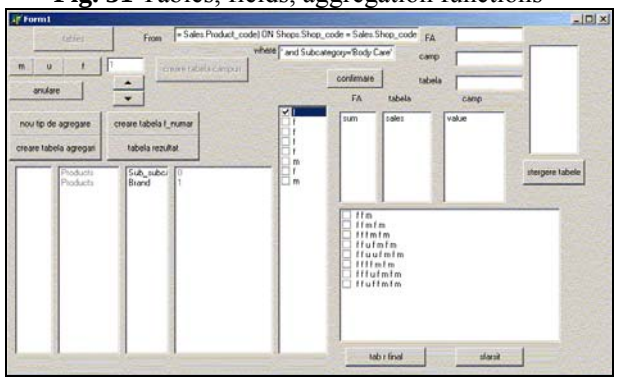


Fig. 32 Specification of aggregation types

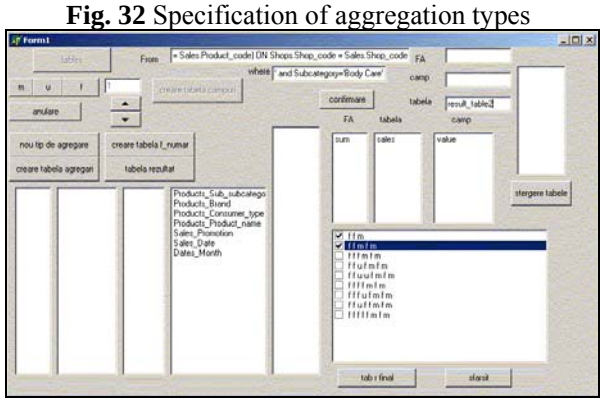


Fig. 33 Final confirmation for result tables

result_table1 : Table

Sub_subcategory	Brand	Consumer_type	Product_name	Promotion	Date	Month	Total_value
Shower gel	Calvin Klein						68
Shower gel	Lancome						60
Shower gel	Calvin Klein	men					48
Shower gel	Calvin Klein	women					20
Shower gel	Lancome	women					60
Shower gel	Calvin Klein		Contradiction 20				20
Shower gel	Calvin Klein		Euphoria 200 Ml				48
Shower gel	Lancome		Tresor 6.7 OZ				60
Shower gel	Calvin Klein			No			48
Shower gel	Calvin Klein			Yes			20
Shower gel	Lancome			No			24
Shower gel	Lancome			Yes			36
Shower gel	Calvin Klein				3/8/2006		20
Shower gel	Calvin Klein				3/9/2006		48
Shower gel	Lancome				3/8/2006		36
Shower gel	Lancome				3/9/2006		24
Shower gel	Calvin Klein					3	68
Shower gel	Lancome					3	60
Shower gel	Calvin Klein	men	Euphoria 200 Ml				48
Shower gel	Calvin Klein	women	Contradiction 20				20
Shower gel	Lancome	women	Tresor 6.7 OZ				60
Shower gel	Calvin Klein	men		No			48
Shower gel	Calvin Klein	women		Yes			20
Shower gel	Lancome	women		No			24
Shower gel	Lancome	women		Yes			36
Shower gel	Calvin Klein	men			3/9/2006		48
Shower gel	Calvin Klein	women			3/8/2006		20
Shower gel	Lancome	women			3/8/2006		36
Shower gel	Lancome	women			3/9/2006		24
Shower gel	Calvin Klein	men				3	48
Shower gel	Calvin Klein	women				3	20
Shower gel	Lancome	women				3	60
Shower gel	Calvin Klein		Contradiction 20	Yes			20
Shower gel	Calvin Klein		Euphoria 200 Ml	No			48
Shower gel	Lancome		Tresor 6.7 OZ	No			24
Shower gel	Lancome		Tresor 6.7 OZ	Yes			36
Shower gel	Calvin Klein		Contradiction 20		3/8/2006		20
Shower gel	Calvin Klein		Euphoria 200 Ml		3/9/2006		48
Shower gel	Lancome		Tresor 6.7 OZ		3/8/2006		36
Shower gel	Lancome		Tresor 6.7 OZ		3/9/2006		24
Shower gel	Calvin Klein		Contradiction 20			3	20
Shower gel	Calvin Klein		Euphoria 200 Ml			3	48
Shower gel	Lancome		Tresor 6.7 OZ			3	60
Shower gel	Calvin Klein			No	3/9/2006		48
Shower gel	Calvin Klein			Yes	3/8/2006		20
Shower gel	Lancome			No	3/9/2006		24
Shower gel	Lancome			Yes	3/8/2006		36
Shower gel	Calvin Klein			No		3	48
Shower gel	Calvin Klein			Yes		3	20
Shower gel	Lancome			No		3	24
Shower gel	Lancome			Yes		3	36
Shower gel	Calvin Klein	men	Euphoria 200 Ml	No			48
Shower gel	Calvin Klein	women	Contradiction 20	Yes			20
Shower gel	Lancome	women	Tresor 6.7 OZ	No			24
Shower gel	Lancome	women	Tresor 6.7 OZ	Yes			36
Shower gel	Calvin Klein	men	Euphoria 200 Ml		3/9/2006		48
Shower gel	Calvin Klein	women	Contradiction 20		3/8/2006		20
Shower gel	Lancome	women	Tresor 6.7 OZ		3/8/2006		36
Shower gel	Lancome	women	Tresor 6.7 OZ		3/9/2006		24
Shower gel	Calvin Klein	men	Euphoria 200 Ml			3	48
Shower gel	Calvin Klein	women	Contradiction 20			3	20
Shower gel	Lancome	women	Tresor 6.7 OZ			3	60
Shower gel	Calvin Klein	men		No	3/9/2006		48
Shower gel	Calvin Klein	women		Yes	3/8/2006		20
Shower gel	Lancome	women		No	3/9/2006		24
Shower gel	Lancome	women		Yes	3/8/2006		36
Shower gel	Calvin Klein	men		No		3	48
Shower gel	Calvin Klein	women		Yes		3	20
Shower gel	Lancome	women		No		3	24
Shower gel	Lancome	women		Yes		3	36
Shower gel	Calvin Klein		Contradiction 20	Yes	3/8/2006		20
Shower gel	Calvin Klein		Euphoria 200 Ml	No	3/9/2006		48
Shower gel	Lancome		Tresor 6.7 OZ	No	3/9/2006		24
Shower gel	Lancome		Tresor 6.7 OZ	Yes	3/8/2006		36
Shower gel	Calvin Klein		Contradiction 20	Yes		3	20
Shower gel	Calvin Klein		Contradiction 20	Yes		3	20
Shower gel	Calvin Klein		Euphoria 200 Ml	No		3	48
Shower gel	Lancome		Tresor 6.7 OZ	No		3	24
Shower gel	Lancome		Tresor 6.7 OZ	Yes		3	36
Shower gel	Calvin Klein	men	Euphoria 200 Ml	No	3/9/2006		48
Shower gel	Calvin Klein	women	Contradiction 20	Yes	3/8/2006		20
Shower gel	Lancome	women	Tresor 6.7 OZ	No	3/9/2006		24
Shower gel	Lancome	women	Tresor 6.7 OZ	Yes	3/8/2006		36
Shower gel	Calvin Klein	men	Euphoria 200 Ml	No		3	48
Shower gel	Calvin Klein	women	Contradiction 20	Yes		3	20
Shower gel	Lancome	women	Tresor 6.7 OZ	No		3	24
Shower gel	Lancome	women	Tresor 6.7 OZ	Yes		3	36

Record: 75 of 86

Fig. 34 Result table corresponding to all specifications of aggregation types

If we select only the two first specifications of aggregation types (like in *Figure 33*), we will obtain the results presented in *Figure 35*. In both result tables (see *Figures 34* and *35*) the result presentation is affected by the order in which we introduce the specifications of aggregation types (see *Figures 33*). According to the situation, we can choose the intended order of specifications.

result_table2 : Table

Sub_subcategory	Brand	Consumer_type	Product_name	Promotion	Date	Month	Total_value
Shower gel	Calvin Klein						68
Shower gel	Lancome						60
Shower gel	Calvin Klein	men					48
Shower gel	Calvin Klein	women					20
Shower gel	Lancome	women					60
Shower gel	Calvin Klein		Contradiction 20(				20
Shower gel	Calvin Klein		Euphoria 200 Ml				48
Shower gel	Lancome		Tresor 6.7 OZ				60
Shower gel	Calvin Klein			No			48
Shower gel	Calvin Klein			Yes			20
Shower gel	Lancome			No			24
Shower gel	Lancome			Yes			36
Shower gel	Calvin Klein				3/8/2006		20
Shower gel	Calvin Klein				3/9/2006		48
Shower gel	Lancome				3/8/2006		36
Shower gel	Lancome				3/9/2006		24
Shower gel	Calvin Klein					3	68
Shower gel	Lancome					3	60

Record: 1 of 18

Fig. 35 Result table corresponding to the specifications *f f m* and *f f m f m*

The field *value* from the table *sales* and the fields *year, month, day* and *day_of_week* from the table *dates* are additional fields. These fields are necessary only when we start the specification of the aggregation types, like in *Figure 31*, in order to use the algorithm proposed in the *Section 4*. When we have the result tables, like in the *Figures 34* and *35*, we can delete these fields.

6 Conclusion

The algorithms can be used for any type of databases. In the programming environment we use *SQL* statements to exploit the database. Here we have worked with databases from *Access*. In the case in which we want to work with another type of relational database, we must only modify (in the program) the *SQL* statement (if it is different from the corresponding *SQL* statement from *Access*).

We have presented the implementation in *Delphi*, but the implementation can also be made in other programming environments.

For *n* fields (used for grouping) from a database, we can obtain 2^n aggregation types. With our algorithms, we can obtain any subsets of aggregation types.

The algorithms from the *Sections 2* and *3* concern a very short time for obtaining the results, but requires that the database be exploited only using the application.

We can propose this algorithm when it is often necessary to obtain various sets of aggregation values and the number of aggregation types is not very small.

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

In the algorithm from the *Section 2*, before inserting records in tables, we must specify the desired aggregation types. We have presented the algorithm for a snowflake scheme. This algorithm is the same for a star scheme and can be extended to a constellation scheme or when the result tables have different headers. In the last two cases 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*).

In the algorithm from the *Section 3* we can specify the desired aggregation types in any moment, but this algorithm is limited to the case in which we use fields from only one table (it can be used, for example, for questionnaires).

In the case in which a database is not exploited with the proposed applications from *Sections 2* and *3*, with the third algorithm, presented in the *Section 4*, we can also obtain the desired sets of aggregation types.

We must remember that if the number *n* of aggregation types is greater, the time for obtaining the results can be a problem (because *n SQL* statements are executed, instead of only one *SQL* statement - like in the first algorithm).

However, we can propose this algorithm when it is not often necessary to obtain such results or the number of aggregation types is not greater.

Our algorithms can be used in many other economic situations, as well. We have presented a situation at shop level but the analysis can be used in more departments or sectors such as finance departments, marketing departments, manufacturing sector, tourism etc.

References:

[1] *Borland Delphi 6 for Windows, Developer'Guide*, 2001.

[2] Messaoud R. B., Boussaid O., Rabaseda S.- A New OLAP Aggregation Based on the AHC Technique - *Workshop Proceedings DOLAP 2004* - November 12-13, 2004, Hyatt Arlington Hotel, Washington,D.C.,USA (http://www.cis.drexel.edu/faculty/song/dolap/dolap04/wproceedings.htm)

[3] *Oracle OLAP Developer's Guide to the OLAP API* 10*g* Release 2 (10.2), June 2005, (http://www.oracle.com/technology/documentation/olap.html)

[4] Silberschatz A., Korth H.F., Sudarshan – Database System Concepts – *McGraw-Hill*, Fifth Edition, 2005

[5] Tanasescu A , Boussaid O., Bentayeb F. - Preparing Complex Data for Warehousing - 3rd *ACS/IEEE International Conference on Computer Systems and Applications*, Cairo, Egipt. 2005

[6] Voicu M., Mircea G. - *Constructing and Exploiting Hypercubes in order to Obtain Aggregated Values– WSEAS Transactions on Information Science and Applications, Issue 10, Volume 3, October 2006, ISSN 1790-0832,* pag. 2008-201

[7] Voicu M., Mircea G. - Algorithm using hypercube for aggregations - *Conference Proceedings of 10th WSEAS International Conference on COMPUTERS, July 13-15, 2006,* Vouliagmeni Beach, Athens, Greece

[8]Voicu M.C. – Algorithms for database and applications in Delphi, (book- Vol. II), *Mirton Publishing House*, Timişoara, 2006

[9] Voicu M.C., Mircea G. –Viewpoints on obtaining aggregated value sets - *Proceedings of the 11th WSEAS International Conference on COMPUTERS,* Agios Nikolaos, Crete Island, Greece, July 26-28, 2007, pag. 633- 638

[10] Voicu M.C. – Database applications and data structures in Java, using JBuilder, (book) *West University Publishing House*, Timişoara, 2007