Database Analysis Models used for Studying the Residential Assemble Market

MIRELA-CATRINEL VOICU, ANDREEA BÂNCIU, MIHAI DRAGOTA, RAUL TURCU
Faculty of Economics Business Administration
West University of Timişoara
ROMANIA
mirela.voicu@feaa.uvt.ro, andreeabanciu_ro@yahoo.com, mihai.dragota86@yahoo.com, raulandrei.turcu@yahoo.com,
http://www.feaa.uvt.ro

Abstract: - In this paper we present our own study on residential assemblies, starting from a particular set presented in [6]. We build a database and we present our results concerning these data. We present an algorithm for obtaining aggregated values sets. In order to exemplify its application we make a study on residential assemblies including apartments with three rooms.

Key-Words: residential assemblies, aggregated value sets, economical analyses, relational databases, programming environment.

1 Introduction

Optimism or pessimism (see [5]), certainty or uncertainty (see [10]), price evolution (see [10], [12], [13]), all these are key words concerning the residential assemblies market of Romania. We find data in [10] about the residential rating and about the parameters of project evaluation. Some basic rules and snares in real estate transactions are presented in [12]. In [1], some factors which influence the residential assemblies market are presented. Some residential assemblies are presented in [10], [12] and [13]. In [1] and [6] we find catalogs which describe many residential assemblies.

Starting from the data presented in [6], in our paper we build a database, we present a model of analysis of these data and algorithms in order to explore the database. We consider our algorithms interesting for clients, as well as for developers and researchers.

2 A database model for the analysis of residential assemblies

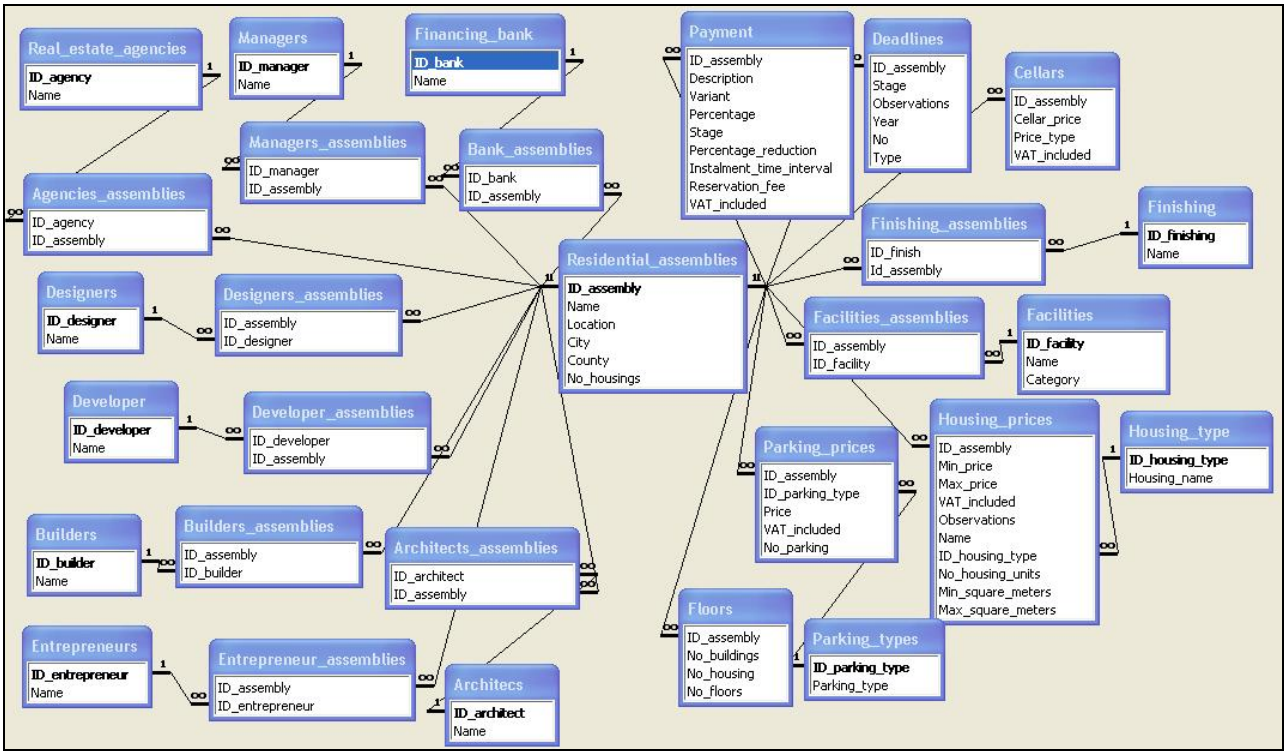


Fig. 1 Residential assemblies – the database structure

Starting from the data presented in catalog from [6], we built the database presented in *Figure 1*. Now, we present the tables from *Figure 1,* the significance of fields in these tables and some of our results. In the tables *Financing_bank, Managers, Real_estate_agencies, Designers, Developers, Builders, Entrepreneurs* and *Architects* we have two fields: *ID* and *Name*. For each of these tables we can use more data, like address, city, county, phone number, etc.

For each residential assembly we find one or more developers, as followings: *Adama Holding Ltd.* for 6 assemblies, *Conarg Real Estate* for 2, *Globe Trade Center Romania* for 2, *Copper Beech* for 2, *CAN SERV* for 2, *Shapir Group* for 2, *Well Group* for 2. For the other residential assemblies, we find only one developer.

The financing banks are specified for 21 residential assemblies (with *Alpha Bank* for 5 from them, *Piraeus Bank* for 3, *Raiffaisen Bank* for 3 *Banca Romaneasca* for 2, *BCR* for 2). We find the other banks only in the case of one residential assembly.

We find real estate agencies for 63 of all cases, e.g. *CBRE Eurisko* for 10, *Coldwell Banker* for 7, *Colliers International* for 6, *Regatta* for 5, *Neocasa* for 5, *Media City* for 5, *Euroest Invest* for 4, *Colliers* for 4 and *DTZ Echinocțiu* for 2.

The builders are specified for 23 assemblies, with *Ozer Construction* for 2 assemblies. We find the other builders only in the case of one residential assembly.

The entrepreneurs are specified for 21 residential assemblies, with *Bouygues* for 2 and *Conarg Consulting* for 2.

The architects are specified for 40 residential assemblies, with *Axa* for 2 assemblies, *Westfourth Architecture* for 2 and *Arhis Design* for 2. We find the other architects only in the case of one residential assembly.

	No of RA
0 <no_housings ≤200	2
100< no_housings ≤200	17
200< no_housings ≤300	14
300< no_housings ≤400	8
400< no_housings ≤500	7
500< no_housings ≤1000	18
1000< no_housings ≤1500	11
1500< no_housings ≤2000	3
2000< no_housings ≤3000	0
3000< no_housings ≤4600	5

Table 1 Number of housings

In the *Residential_assemblies* table we have data about 85 residential assemblies from 17 localities (58 residential assemblies from *Bucharest*, 5 from *Braşov*, 4 from *Constanța*, 2 from *Buftea*, 2 from *Iaşi*, 2 from *Mamaia*, 2 from *Sibiu*, 1 from *Bacău*, 1 from *Băneasa*, 1 from *Cluj-Napoca*, 1 from *Craiova*, 1 from *Pipera*, 1 from *Sat Ploieştiori - Com. Blejoi*, 1 from *Sinaia*, 1 from *Ştefanestii de Jos*, 1 from *Tunar* and 1 from *Vâlcea*) and from 10 counties (6 assemblies from *Constanța*, 6 from *Ilfov*, 5 from *Braşov*, 2 from *Iaşi*, 2 from *Prahova*, 2 from *Sibiu*, 1 from *Bacău*, 1 from *Cluj*, 1 from *Dolj*, 1 from *Vâlcea*) and 58 from *Bucharest*.

The minimum number of housings is 50 (villas), as in the case of the residential assembly *The Fifties* from *Craiova*. The maximum number of housings is 4600, as in the case of the residential assembly *Cosmopolis* from *Stefanestii de Jos* locality, *Ilfov* county (see also *Table 1*). In all tables, *No of RA* means the number of residential assemblies.

In the *Floors* table, we save data such as number of buildings, number of apartments and number of floors. The minimum number of floors is 1 for residential assemblies which include houses or villas (see also *Table 2*).

The maximum number of floors is 30 for the residential assembly *NeoPeninsula Residences* from *Bucharest*.

The managers and designers are specified only for a small number of assemblies.

	No of RA
0 < floors_number ≤5	28
5 < floors_number ≤ 10	30
10 < floors_number ≤ 15	22
15 < floors_number ≤ 20	12
20 < floors_number ≤ 25	2
25 < floors_number ≤ 30	1

Table 2 Number of floors

The residential assembly *Sigma Residence & Gardens* has buildings with 5 different floor levels. *Global City* and *Laguna Residence* have 4 different floor levels for their buildings. *Asmita Gardens* and *Alia Apartments* have 3 different floor levels for their buildings. *Green City Residence, Green Wave Mamaia, Green Lake* and *Garden of Eden* have buildings with 2 different floor levels. The other assemblies have buildings with the same floor number or this number is not specified.

Seven of these assemblies have only one building and *Greenfield* has 328 buildings (which are villas). See also *Table 3*.

In the *Parking_types* table we have two types of

parking options: underground parking and aboveground parking. We find that 70 assemblies have parking and for 59 of them, the price is specified. For 35 assemblies we find both types of parking, 14 assemblies have only underground parking and 10 assemblies have only aboveground parking. The price is between 3500 (for *Rams Residential*) and 18500 euros (for *Parcul Privighetorilor*). See also *Table 4*. The VAT is not included.

	No of RA
0 < buildings_number ≤ 5	38
5 < buildings_number ≤ 10	16
10 < buildings_number ≤ 15	5
15 < buildings_number ≤ 20	4
20 < buildings_number ≤ 25	2
25 < buildings_number ≤ 30	1
30 < buildings_number ≤ 40	0
40 < buildings_number ≤ 45	1
45 < buildings_number ≤ 50	2
50 < buildings_number ≤ 100	5
100 < buildings_number ≤ 300	1
300 < buildings_number ≤ 350	1

Table 3 Number of buildings

	No of RA
0 < parking_ price≤ 4000	9
4000 < parking_ price ≤ 6000	10
6000 < parking_ price ≤ 8000	10
8000 < parking_ price ≤ 10000	15
10000 < parking_ price ≤ 12000	17
12000 < parking_ price ≤ 14000	8
14000 < parking_ price ≤ 16000	8
16000 < parking_ price ≤ 18000	2
18000 < parking_ price ≤ 19000	1

Table 4 Parking price in euro

For 2 assemblies we find 3 different prices for parking, for 35 assemblies we have 2 prices (generally, a price for underground parking and a price for aboveground parking) and for the other 22 there is 1 price.

We find cellars for 22 assemblies. The price is between 650 and 850 euros per square meter or between 700 and 8000 euros per cellar, when the VAT is not included.

We find the following finishing: air conditioning, central heating, kitchen with domestic appliances, solar installations, marble in the bathroom, furnished bathroom and furnished kitchen.

The facilities belong to the following categories: landscape gardening (fountains, gardens, lakes, parks, green spaces), bank (banking agencies, ATM-Automatic teller machine), business (offices, business park), commerce (gas station, bakery, commercial center, dry cleaning station, flower shop, bookshop and stationery, pastry, market, catering services, launderette, car wash), entertainment (clubs, playground for children), education (baby sitting service, nursery school, school), event (amphitheater, conference center, projection room), hotel, care and beauty (massage, spa center, hairdresser, Jacuzzi, beauty salon, sauna, solar, hydro massage), local (bar, coffee bar, casino and restaurant), parking, post, church, health center (medical center, pharmacy), security (secure access, Interphone, guard services, surveillance systems), sport (bowling track, tennis court, basketball court, horseback riding, sport club, rink, swimming pool, track cycling, jogging track, fitness club, gymnastics club, weightlifting club, squash room, golf course), transport (heliport, means of transport available to residents). See also *Tables 5* and *6*.

	No of RA
0 < no_facilities ≤ 5	42
5 < no_facilities ≤ 10	29
10 < no_facilities ≤ 15	5
15 < no_facilities ≤ 20	2
20< no_facilities ≤ 26	1

Table 5 Number of facilities

Category	No of RA	Category	No of RA
sport	81	landscape gardening	31
parking	70	health	22
commerce	59	event	9
security	40	business	7
entertainment	38	bank	4
local	38	hotel	4
education	36	church	3
care and beauty	32	transport	2
		post	1

Table 6 Categories of facilities

In the *Deadlines* table, in the field *type* we use the following values: *flat, building, housing units* and *villa*. The field *No* refers to their number, in the field *Observations* we use values such as: a month, a quarter of a year or a season. For each assembly we find between 1 and 4 stages of constructions. We find deadlines for these stages, in the following way: in 2007 for 1 assembly, in 2008 for 25, in 2009 for

50, in 2010 for 41, in 2011 for 7, in 2012 for 4, in 2013 for 3 and in 2015 for 2.
We find flats with 1 room for 3 assemblies, flats with 2 rooms for 81, flats with 3 rooms for 74, flats with 4 rooms for 48, flats with 5 rooms for 8, flats with 6 rooms for 1, houses for 1, duplex for 3, duplex with 2 rooms for 1, duplex with 3 rooms for 8, duplex with 4 rooms for 2, duplex with 5 rooms for 1, studio flats for 57, penthouses for 25 and villas for 11 residential assemblies.
In the *Housing_price* table, the field *Observations* specifies if the price is per square meter or per housing unit. We find that 36 square meters is the minimum value for a housing unit (for a studio) and that 759 is the maximum value (for a villa). See also *Table 7*.

	No of RA
0 < square meters ≤ 40	51
40 < square meters ≤ 60	27
60 < square meters ≤ 100	51
100 < square meters ≤ 200	69
200 < square meters ≤ 300	21
300 < square meters ≤ 500	7
500 < square meters ≤ 700	0
700 < square meters ≤ 800	1

Table 7 Square meters for housing units

For the cases in which the price is per square meter, we provide in *Table 8* the lowest and highest price in euros for each housing type.

Housing unit	Min price	Max price
flat with 1 room	1100	2300
flat with 2 rooms	875	3200
flat with 3 rooms	875	3200
flat with 4 rooms	1000	2300
flat with 5 rooms	1100	2000
flat with 6 rooms	1100	1100
duplex	1150	1700
duplex with 2 rooms	1450	1450
duplex with 3 rooms	1080	1450
duplex with 4 rooms	1080	1080
duplex with 5 rooms	1360	1360
studio flat	875	3200
penthouse	1000	3200
villa	675	2000

Table 8 Prices per square meter for housing units

For 65 assemblies we find specified between 1 and 5 means of payment.

3 Algorithm for obtaining aggregated values sets

3.1 Algorithm presentation

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 aggregation, many tools are known. We remind some of 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 remind 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.
Now, we consider the table presented in *Fig. 2*. Here, the fields *field1, field2, field3, field4, field5* form the maximal set used for grouping and the field *fvalue* is used for aggregation.

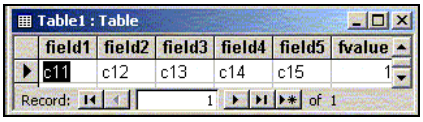


Fig. 2 An initial table

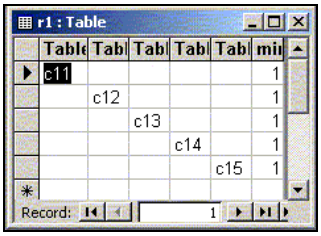


Fig. 3 The result for *m f m*

The specification *m f m* produces the results presented in *Fig. 3* (which correspond to five aggregation types).
The specification *m f u f m* produces the results presented in *Fig. 4* (which correspond to three aggregation types).
The specification *f m f m* produces the results presented in *Fig. 5* (which correspond to four aggregation types).
In such specifications we can also eliminate some fields for a certain *f*.

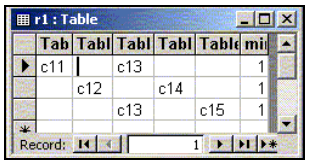


Fig. 4 The result for *m f u f m*

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.

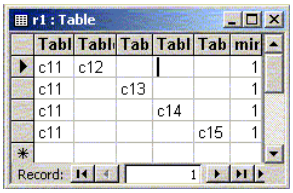


Fig. 5 The result for *f m f m*

For the algorithm implementation we use a programming environment (we worked in *Delphi*) and a database (there we used databases from *Access*).
For grouping and for aggregation, we can use fields from one or more tables.
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 6*.

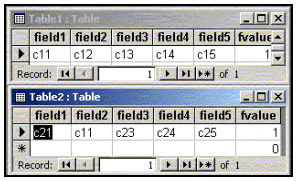


Fig. 6 Initial tables

Tables, fields, relationships, aggregation functions
The user must specify the tables, fields, relationships, aggregation functions like in *Figure 7*. In this figure 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);
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 8*).

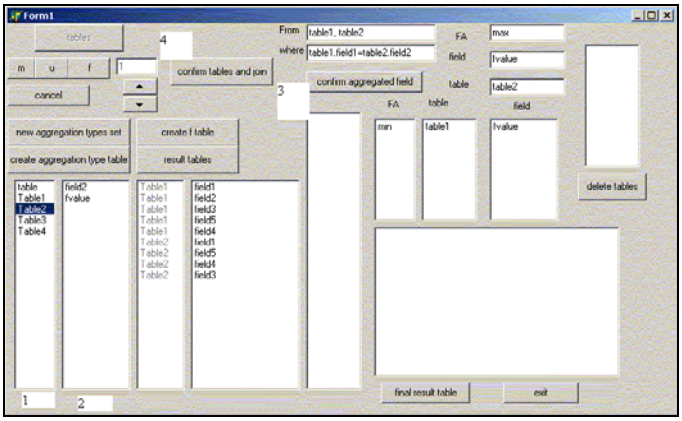


Fig. 7 Tables, fields, relationships, aggregation functions

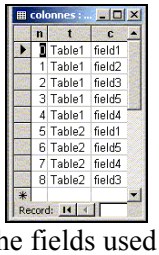


Fig. 8 Indexes for the fields used for grouping

Specification of 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);

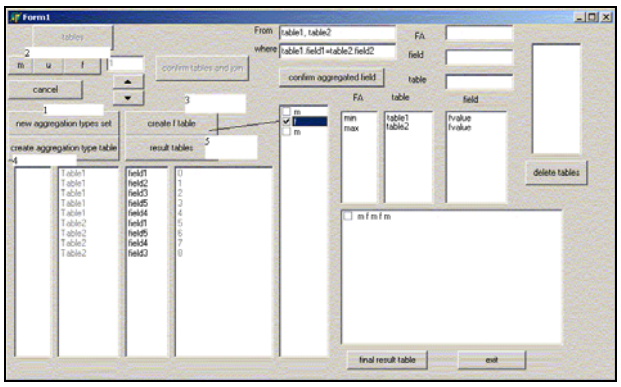
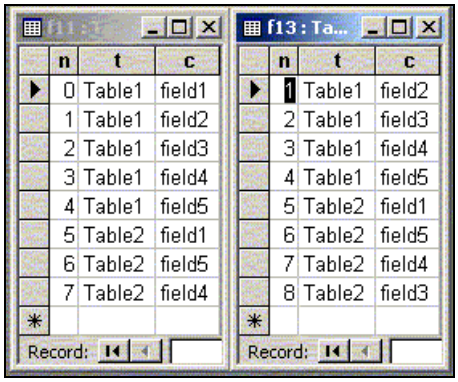


Fig. 9 Specification of aggregation types

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 9*), 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 10* (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

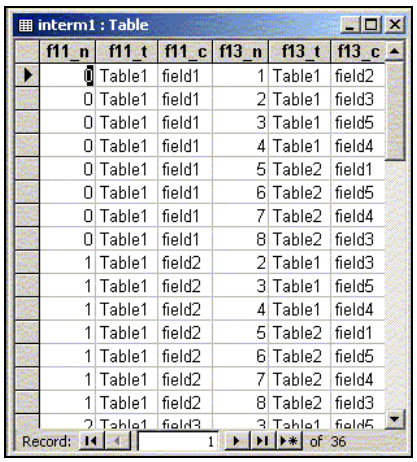
n	t	c
1	Table1	field2
2	Table1	field3
3	Table1	field4
4	Table1	field5
5	Table2	field1
6	Table2	field5
7	Table2	field4
8	Table2	field3

Fig. 10 The corresponding tables for each *f* from the specification *m f m f m*

When we have the table for each *f* from a specification (see *Figure 10*), 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 11*). Here we have a cartesian product between the records from the tables corresponding at each *f* (presented in *Figure 10*). 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 that will be specified all aggregation types. We will obtain the results at the following step (see *Figure 9*):



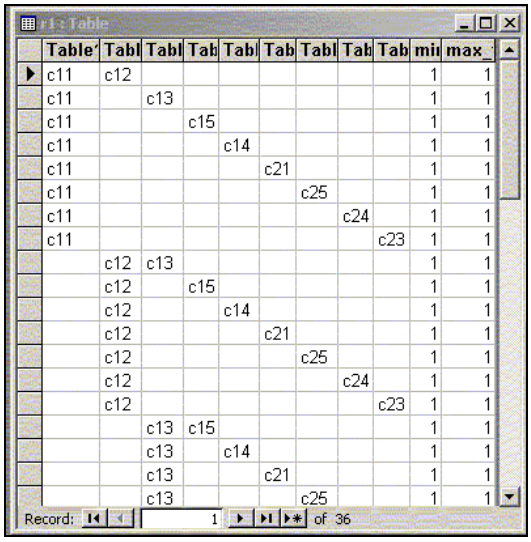
interm1 : Table

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

Record: 1 of 36

Fig. 11 A table, which contains as records the aggregation types

5 – with a click on the command button "result tables", for each table like the table presented in *Figure 11* (these tables correspond to each specification of aggregation types), we will construct a corresponding result table, which contains the aggregated values, like in *Figure 12*.



r1 : Table

Table'	Tabl	Tabl	Tab	Tab	Tab	Tabl	Tab	Tab	mi	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

Record: 1 of 36

Fig. 12 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 11,* 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 11*, 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 12*), 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.

Final tables

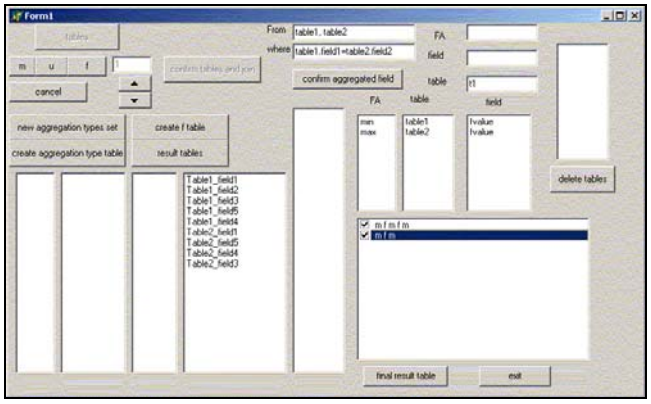


Fig. 13 Final confirmations for result tables

We confirm the desired specifications of aggregation types like in *Figure 13*. With a click on the field name we eliminate the field from the result table.
For the case presented in *Figure 13*, we will obtain the result table presented in *Figure 14*.
Also, we can build new tables with fields concerning criteria on the fields from the initial tables, like in the following example.

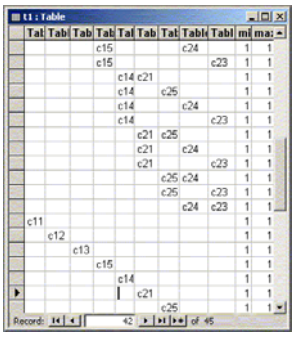


Fig. 14 A result table for aggregated values

3.2 A study concerning the categories of facilities for residential assemblies including apartments with three rooms.

Now, we consider that we are interested to know the categories of facilities for residential assemblies including apartments with three rooms. Moreover, as example, we are interested to know only those which contain the following 4 categories of facilities: *landscape gardening*, *commerce*, *parking* and *sport*.

First we create a table which has the following fields: *ID_assembly*, *c1,…c16*.
For space economy in the presentation, we denote the category fields in the following way: landscape gardening - *c1*, commerce - *c2*, parking - *c3*, sport - *c4*, bank - *c5*, business - *c6*, entertainment - *c7*, education- *c8*, event -*c9*, hotel - *c10*, care and beauty - *c11*, local - *c12*, church - *c13*, health - *c14*, security - *c15*, transport - *c16*.
In this table, in the fields *c1…c16* we use the value 1 if we find the category for the corresponding *ID_assembly* and the value 0, if we do not.

We use the following specification of aggregation types:
Case1: fffffm - which refers to the number of residential assemblies which contain the categories *landscape gardening*, *commerce* and *parking, sport.*
Case2: ffffmfm – which refers to the number of residential assemblies which contain 5 categories where the first 4 categories from them are: *landscape gardening*, *commerce* and *parking, sport.*
Case 3: ffffmfmfm – 6 categories
Case 4: ffffmfmfmfm – 7 categories
Case 5: ffffmfmfmfmfm –8 categories
Case 6: ffffmfmfmfmfmfm – 9 categories
Case 7: ffffmfmfmfmfmfmfm –10 categories
Case 8: ffffmfmfmfmfmfmfmfm –11 categories
Case 9: ffffmfmfmfmfmfmfmfmfm –12 categories
Case10: ffffmfmfmfmfmfmfmfmfmfm –13 categories

Case11: ffffmfmfmfmfmfmfmfmfmfmfm–14 categories
Case12: ffffmfmfmfmfmfmfmfmfmfmfm –15 categories
Case13: fffffmfmfmfmfmfmfmfmfmfmfm –16 categories

The fields used for grouping are fields *c1…c16*. The field used for aggregation is the field *ID_assembly* and the aggregation function is *count*. From the result tables corresponding to each specification of aggregation types, we save only the records for which the sum of the fields *c1…c16* is equal with the minimum of categories specified for each case. Such results are presented in the *Tables 9-15,* where *no_a* refers to the number of residential assemblies and *no_f* refers to the minimum number of category of facilities included.

no_f	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	no_a
4	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	6

Table 9 Results for *Case 1*

no_f	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	no_a
5	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	0	0	0	0	0	0	5
5	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	0	0	0	0	0	0	3
5	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	**1**	0	0	0	0	3
5	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	**1**	0	0	3
5	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	**1**	0	3
5	**1**	**1**	**1**	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0	1
5	**1**	**1**	**1**	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0	1
5	**1**	**1**	**1**	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	1
5	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	0	0	0	0	0	1
5	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	**1**	1

Table 10 Results for *Case 2*

no_f	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	no_a
6	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	0	**1**	0	0	0	0	3
6	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	0	0	0	**1**	0	0	3
6	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	0	0	0	0	**1**	0	3
6	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	0	0	0	0	0	0	2
6	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	0	**1**	0	0	0	0	2
6	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	0	0	0	**1**	0	0	2
6	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	0	0	0	0	**1**	0	2
6	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	**1**	0	**1**	0	0	2
6	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	**1**	0	0	**1**	0	2
6	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	**1**	**1**	0	2
6	**1**	**1**	**1**	**1**	0	**1**	0	**1**	0	0	0	0	0	0	0	0	1
6	**1**	**1**	**1**	**1**	0	0	**1**	0	**1**	0	0	0	0	0	0	0	1
6	**1**	**1**	**1**	**1**	0	0	**1**	0	0	**1**	0	0	0	0	0	0	1
6	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	**1**	0	0	0	0	0	1
6	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	0	0	0	0	0	**1**	1
6	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	**1**	0	0	0	0	0	1
6	**1**	**1**	**1**	**1**	0	0	0	0	**1**	**1**	0	0	0	0	0	0	1
6	**1**	**1**	**1**	**1**	0	0	0	0	**1**	0	0	**1**	0	0	0	0	1
6	**1**	**1**	**1**	**1**	0	0	0	0	**1**	0	0	0	0	0	0	1	1
6	**1**	**1**	**1**	**1**	0	0	0	0	0	**1**	0	**1**	0	0	0	0	1
6	**1**	**1**	**1**	**1**	0	0	0	0	0	**1**	0	0	0	0	0	1	1
6	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	**1**	0	0	0	0	1
6	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	0	0	**1**	0	0	1
6	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	0	0	0	**1**	0	1
6	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	**1**	0	0	0	**1**	1

Table 11 Results for *Case 3*

no_f	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	no_a
7	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	0	**1**	0	0	0	0	2
7	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	0	0	0	**1**	0	0	2
7	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	0	0	0	0	**1**	0	2
7	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	0	**1**	0	**1**	0	0	2
7	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	0	**1**	0	0	**1**	0	2
7	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	0	0	0	**1**	**1**	0	2
7	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	0	**1**	0	**1**	0	0	2
7	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	0	**1**	0	0	**1**	0	2
7	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	0	0	0	**1**	**1**	0	2
7	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	**1**	0	**1**	**1**	0	2
7	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	**1**	0	0	0	0	0	1
7	**1**	**1**	**1**	**1**	0	0	**1**	0	**1**	**1**	0	0	0	0	0	0	1
7	**1**	**1**	**1**	**1**	0	0	**1**	0	**1**	0	0	**1**	0	0	0	0	1
7	**1**	**1**	**1**	**1**	0	0	**1**	0	**1**	0	0	0	0	0	0	**1**	1
7	**1**	**1**	**1**	**1**	0	0	**1**	0	0	**1**	0	**1**	0	0	0	0	1
7	**1**	**1**	**1**	**1**	0	0	**1**	0	0	**1**	0	0	0	0	0	**1**	1
7	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	**1**	**1**	0	0	0	0	1
7	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	**1**	0	0	**1**	0	0	1
7	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	**1**	0	0	0	**1**	0	1
7	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	0	**1**	0	0	0	**1**	1
7	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	**1**	**1**	0	0	0	0	1
7	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	**1**	0	0	**1**	0	0	1
7	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	**1**	0	0	0	**1**	0	1
7	**1**	**1**	**1**	**1**	0	0	0	0	**1**	**1**	0	**1**	0	0	0	0	1
7	**1**	**1**	**1**	**1**	0	0	0	0	**1**	**1**	0	0	0	0	0	**1**	1
7	**1**	**1**	**1**	**1**	0	0	0	0	**1**	0	0	**1**	0	0	0	**1**	1
7	**1**	**1**	**1**	**1**	0	0	0	0	0	**1**	0	**1**	0	0	0	**1**	1
7	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	**1**	0	**1**	0	0	1
7	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	**1**	0	0	**1**	0	1
7	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	0	0	**1**	**1**	0	1

Table 12 Results for *Case 4*

no_f	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	no_a
8	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	0	**1**	0	**1**	0	0	2
8	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	0	**1**	0	0	**1**	0	2
8	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	0	0	0	**1**	**1**	0	2
8	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	0	**1**	0	**1**	**1**	0	2
8	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	0	**1**	0	**1**	**1**	0	2
8	**1**	**1**	**1**	**1**	0	0	**1**	0	**1**	**1**	0	0	0	0	0	**1**	1
8	**1**	**1**	**1**	**1**	0	0	**1**	0	**1**	0	0	**1**	0	0	0	**1**	1
8	**1**	**1**	**1**	**1**	0	0	**1**	0	0	**1**	0	**1**	0	0	0	**1**	1
8	**1**	**1**	**1**	**1**	0	0	0	0	**1**	**1**	0	**1**	0	0	0	**1**	1
8	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	**1**	**1**	0	0	0	0	1
8	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	**1**	0	0	**1**	0	0	1
8	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	**1**	0	0	0	**1**	0	1
8	**1**	**1**	**1**	**1**	0	0	**1**	0	**1**	**1**	0	**1**	0	0	0	0	1
8	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	**1**	**1**	0	**1**	0	0	1
8	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	**1**	**1**	0	0	**1**	0	1
8	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	**1**	0	0	**1**	**1**	0	1
8	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	**1**	**1**	0	**1**	0	0	1
8	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	**1**	**1**	0	0	**1**	0	1
8	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	**1**	0	0	**1**	**1**	0	1
8	**1**	**1**	**1**	**1**	0	0	0	0	0	0	**1**	**1**	0	**1**	**1**	0	1

Table 13 Results for *Case 5*

no_f	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	no_a
9	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	0	**1**	0	**1**	**1**	0	2
9	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	**1**	**1**	0	**1**	0	0	1
9	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	**1**	**1**	0	0	**1**	0	1
9	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	**1**	0	0	**1**	**1**	0	1
9	**1**	**1**	**1**	**1**	0	0	**1**	0	**1**	**1**	0	**1**	0	0	0	**1**	1
9	**1**	**1**	**1**	**1**	0	0	**1**	0	0	0	**1**	**1**	0	**1**	**1**	0	1
9	**1**	**1**	**1**	**1**	0	0	0	**1**	0	0	**1**	**1**	0	**1**	**1**	0	1

Table 14 Results for *Case 6*

no_f	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	no_a
10	**1**	**1**	**1**	**1**	0	0	**1**	**1**	0	0	**1**	**1**	0	**1**	**1**	0	1

Table 15 Results for *Case 7*

The value 1 shows that this category is included.
For the *Cases 8 – 13* we find 0 records. In *Case 3* we find 25 records, in *Case 4* we find 30 records and in *Case 5* we find 20 records. In *Case 6* we find 7 records and in the *Case 7* we find 1 record. The result for the *Cases 1, 2, 6* and *7* are presented in the *Tables 9, 10, 11* and *12,* respectively.

4 Conclusions

Using algorithms like the one presented in *Subsection 3.2* and analyses presented in *Section 2*, we can perform different studies on the residential assemblies market, which can be interesting for clients, but also for developers. The case presented in this paper is just one example and we consider that it can be adapted for situations concerning many other residential assemblies in Romania or from other countries.

References:

[1] *Biz – Residential Guidebook*, Romania, October 2008.

[2] Bogdan Ghilic-Micu, Marian Stoica, Marinela Mircea - How to Succeed in Business Intelligence Initiative: A Case Study for Acquisitions in Romania Public Institutions, *WSEAS TRANSACTIONS on BUSINESS and ECONOMICS, Issue 6, Volume 5, June 2008, pag.298- 309*

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

[4] *Business Magazine*, Romania, No 216, 2/2009.

[5] *Business Magazine*, Romania, No 217, 3/2009.

[6]*Catalog of Residential Assemblies, Business Magazine*, Romania, October 2008.

[7] *100%construct magazine*, Romania, December 2008 – January 2009

[8] Huanzhuo Ye, Shuai Chen, Fang Gao - On Application of SOA to Continuous Auditing, *WSEAS TRANSACTIONS on COMPUTERS Volume 7, 200, pag. 532 – 541.*

[9] 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)

[10] *NewsIn Real Estate Guidebook for clients and developers*, No 1, Romania, 2008

[11] *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)

[12] *Real Estate Magazine*, Romania, Octobre 2008.

[13] *Real Estate Magazine*, Romania, December 2008 – January 2009.

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

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

[16] Voicu M.C., 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

[17] Voicu M.C. - *Algorithms used to obtain aggregated value sets from relational databases* - 9th WSEAS International Conferen3ce on MATHEMATICS & COMPUTERS IN BUSINESS &ECONOMICS (MCBE '08), Bucharest, Romania, June 24-26, 2008.

[18] Xin Jin - Research on E-business Intelligent Examination System, *WSEAS TRANSACTIONS on INFORMATION SCIENCE & APPLICATIONS, Volume 6, 2009, pag. 21-30*