Capturing Semantics from Bitmap Indices for Data Analysis

Carlo DELL'AQUILA, Ezio LEFONS, and Filippo TANGORRA
Dipartimento di Informatica
Università di Bari
via Orabona 4, 70125 Bari
ITALY

Abstract: - Bitmap indexing is a diffuse approach for processing efficiently complex queries in decision support activities. Besides this common use of bitmap structures, the use of bitmap indices to represent analytical views of user's data is presented here. In this approach, bitmaps can be created and utilized not only to index different domain attribute values, but also to pre-compute legal relational algebra query expressions useful for the analytic purposes. According to this approach, problems of data integrations and conceptual correlations of analytical data can be efficiently solved.

Key Words: - Bitmap index, Data semantics, Data semantic integration, Analytical data view, Complex query pre-processing, Decisional user

1 Introduction

The need to process and analyze huge volumes of information in OLAP or data-mining applications has led to the extraction, integration, and organization of enterprise data. The commonly adopted solution is to build data warehouses, very large repositories that integrate data from operational databases of several enterprise sectors for decisional analysis. Data warehouses contain data consolidated from several operational databases and they must solve two crucial problems, namely, data integration, and efficient access to stored data.

The integration of data coming from different sources is necessary for such data are stored on heterogeneous, relational and legacy databases that are components of information systems owned by the decisional organization or other external organizations. The ETL procedures are the tools of the data warehouse system that allow the decisional data administrator to extract, to transform, to load, to refresh, and to integrate data from the several source data described by different data schemas in the data warehouse. The technological solutions of distributed services for information systems are adopted, such as, for example, the management of the data inconsistency and structure incompatibility. The definition of the ETL procedures is a very difficult step, because generally inadequate or no tools at all are furnished by the system supporting the data warehouse building up. Therefore, often, the user itself must implement manual procedures for this purpose analyzing source data and schemas in order to discovery inconsistent data, incompatible data structures, and data granularity.

Accessing the data in warehouses in an efficient and, at the same time, flexible fashion is a difficult objective to obtain, because decision support data are collected on very large historical repositories. On one part, the efficiency of accesses takes advantage when using repetitive and predefined queries for they allow to predispose adequate internal structures, such as indices [2, 8, 10], materialized views [1, 5, 14], summary tables [12], and statistical data profiles [6, 7, 11]. On the other hand, the data analysis is an iterative and exploratory process in nature characterized by a sequence of unpredictable and occasional queries, in which each successive query against the data warehouse could be influenced by the results of previous ones. This process can vanish the predefined structures and require flexible data organization.

Bitmap indexing is a very popular technique proposed for processing complex queries in the data warehouse environment supporting decisional and OLAP applications. In fact, this methodology is particularly adapt to provide fast query processing in a decisional context characterized by non-volatile and read-mostly data.

The use of bitmap indices has been considered by researchers pre-eminently from the physical point of view.

In this paper, we present the bitmap structure at an high abstraction level and show its capability to capture semantics to solve integration problems, efficiency in accessing large data amount, and flexibility in answering to decisional needs.

2 Bitmap indices

Bitmap indices are widely used in data warehousing environments, that typically have large amounts of data and ad hoc queries, but a low level of concurrent updating transactions.

In its basic form, a bitmap index on an attribute consists of one vector of bits (or, bitmap) per attribute value, where the size of each bitmap is equal to the number of records in the relation (that is, each bitmap index is dense with respect to data in the relation). The bitmaps are encoded such that the i^{th} record (identified with a row identifier RID) has a value of *v* in the indexed attribute if and only if :

(a) the i^{th} bit in the bitmap associated with the attribute value v is set to 1, and

(b) the i^{th} bit is set to 0 in each of the other bitmaps related to the same indexed attribute.

The bitmap indices are recognized to have the following advantages: (1) they reduce the response time to ad hoc queries, and (2) they require lower amount of storage than other indexing techniques. In fact, if the number of different key values is small (that is, for those attribute columns in which the number of distinct values is small compared with the number of rows in the table), then bitmap indices are very space efficient. Moreover, bitmap indexing efficiently merges indices that correspond to several conditions in a WHERE clause. Rows that satisfy some, but not all, conditions are filtered out before the table itself is accessed. This improves the response time, often considerably. In decision support systems, complex and ad hoc queries accessing data warehouse are frequently used for analytical purposes. Bitmap indices allow to retrieve data very quickly. In fact, very complex selection predicates can be processed efficiently by executing the Boolean *and*, *or*, and *not* operators by means of the basic machine instructions AND, OR, and NOT on bit strings. As an example of an analytical query with a complex selection conditions, let us consider the database schema *CUSTOMER (Name, Lives_in, Works_in, Car, Number_of_children, Has_cable, Has_cellular)* (*cf.*, [2]). If we want to find all customers who work in a state different from the one in which they live, have one or more children, subscribe to a cable service, but do not have a cellular phone, then the corresponding selection condition *C* can be written as

C:= ((*Lives_in* = "AL" AND NOT(*Works_in* = "AL"))
OR ··· OR
(*Lives_in* = "WY" AND NOT(*Works_in* = "WY")))
AND NOT(*Number_of_children* = 0)
AND *Has_cable* = "Y" AND *Has_cellular* = "N".

While common indices in general cannot handle these types of selections easily, bitmap indices can.

However, bitmaps indices become space consuming for high attribute domain cardinality, and they are not very efficient for (low dimensional) range queries, which are typical for data warehouse systems. Several approaches have been proposed to overcome these drawbacks [3, 4, 13, 15]; for example, compression techniques, and a comparative performance algorithm is reported in [9].

3 Analytic bitmap indices

In this Section, we present the use of bitmap indices for representing analytical views of user's data. In this approach, bitmaps are created and used not only to index the attribute data values, but also to pre-compute any legal relational algebra query expressions useful for analytic purposes. In this case, the bitmap encodes setting to 1 the bits corresponding to those, and only records which satisfy the query.

With reference to the example in the previous section, the selection condition *C* can be equivalently and more efficiently expressed as any of, but not limited to, the following conditions *C′* and *C″*:

C′:= NOT(*Lives_in* = *Works_in*)
AND NOT(*Number_of_children* = 0)
AND *Has_cable* = "Y" AND *Has_cellular* = "N",

C″:= *Lives_in* < > *Works_in*
AND *Number_of_children* > 0
AND *Has_cable* = "Y" AND *Has_cellular* = "N".

In Figure 1, a relation instance on the relation schema *FLAT (Flat#, District#, Proprietor#, Year_of_construction, Story, Surface, Room_no, Kitchen_utility_no, Bath_room_no, Telephon_no, Heating-system#, ...)* is exemplified. Each tuple in the *FLAT.Data.* relation refers to a flat in a building.

In Figure 2, analytical bitmap indices (i.e., columns in the *FLAT.Bitstring.* file) are exemplified with reference to data in Figure 1. Each bitmap index contains the boolean truth-values of the relative user's analytical query, as defined and described in the bitmap queries *FLAT.AnalyticView.* file, and it is applied to the instance of the *FLAT.Data.* relation. For example, the 2^{nd} bit in the i^{th} tuple of the *Flat* bitstring (i.e., the i^{th} row in the *FLAT.Bitstring.* file) represents the truth value of the second query (q_2) on the i^{th} tuple of the *FLAT.Data.* relation, or, in other words, whether or not the i^{th} flat is located in the central district. (This case shows the

RID	Flat #	District #	Proprietor #	Year of construction	Story	Surface (m^2)	Room no.	Kitchen and utility room no.	Bath room no.	Telephone no.	Heating-system #	…
1	1	3	1	1940	1	248	5	3	2	0	2	…
2	2	3	1	1940	2	165	3	2	4	1	1	…
.	3	2	1	0	0	45	2	1	1	0	0	…
.	4	3	1	1967	2	86	3	3	1	1	1	…
	5	0	2	1968	3	150	4	3	1	0	1	…
i	6	1	5	1910	2	310	6	5	1	2	2	…
	7	1	5	1910	1	325	6	4	3	1	2	…
.	8	2	0	1950	1	23	1	0	0	0	3	…
.	9	3	1	0	2	175	4	3	1	1	0	…
10	10	2	1	0	1	100	3	1	1	0	0	…

(FLAT.Schema. / FLAT.Data.)

Fig. 1. *FLAT* relation instance.

basic use of the bitmap indices relating to domain values of attribute *District#*.)

Suppose now we are interested in finding the "luxury flats". Here, we define the concept of *luxury flat* in terms of the following selection condition:

Q:= *District* = "centre" AND *Surface* > 200 AND
Bath_room_no ≥ 2 AND
NOT (*Heating-system* = "none"),

which is equivalent, in terms of query views, to:

$$q_2 \wedge q_8 \wedge (q_{30} \vee q_{31}) \wedge (\neg\, q_{36}).$$

The analytical bitmap corresponding to the *Luxury_flat* condition can be added to bitmap indices (as q_{40}, for example; *cf.*, Figure 2).

In the next, we show how the analytical bitmap indices are able to solve some data integration problems and how they can by used for decision support in analytical activities.

FLAT.AnalyticView.

		description	(°)
q_1	*District#*	n.v.;	0
q_2		centre;	1
q_3		suburbs;	2
q_4		isolated building;	3
q_5	*Surface:*	n.v.;	
q_6		≤ 100 m^2;	
q_7		> 100 m^2;	
q_8		> 200 m^2;	
q_9		> 400 m^2;	
…			
q_{18}	*Room no.:*	n.v.;	
q_{19}		= 1;	
q_{20}		= 2;	
q_{21}		= 3;	
q_{22}		= 4;	
q_{23}		between 5 and 10;	
q_{24}		> 10;	
q_{25}	*Kitchen / utility room no.:*	n.v.;	0
q_{26}		≤ 3;	
q_{27}		> 3;	
q_{28}	*Bath room no.:*	n.v.;	0
q_{29}		= 1;	1
q_{30}		= 2;	2
q_{31}		> 2;	3
q_{32}		= 0;	4
q_{33}	*Heating system:*	n.v.;	0
q_{34}		condominjum;	1
q_{35}		individual;	2
q_{36}		none;	3
…			
q_{40}	*Luxury flat:*	$q_2 \wedge q_8 \wedge (q_{30} \vee q_{31}) \wedge (\neg\, q_{36})$	

Legenda: (°) = code in *FLAT.Data.*
n.v. = null value

FLAT.Bitstring.

RID	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	…	q_{18}	q_{19}	q_{20}	q_{21}	q_{22}	q_{23}	q_{24}	q_{25}	q_{26}	q_{27}	q_{28}	q_{29}	q_{30}	q_{31}	q_{32}	q_{33}	q_{34}	q_{35}	q_{36}	…	q_{40}
1	0	0	0	1	0	0	1	1	0	…	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	…	0
.	0	0	0	1	0	0	1	0	0	…	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	…	0
.	0	0	1	0	0	1	0	0	0	…	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	…	0
	0	0	0	1	0	1	0	0	0	…	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	…	0
	1	0	0	0	0	0	1	0	0	…	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	…	0
i	0	1	0	0	0	0	1	1	0	…	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0	…	0
	0	1	0	0	0	0	1	1	0	…	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	…	1
.	0	0	1	0	0	1	0	0	0	…	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	…	0
.	0	0	0	1	0	0	1	0	0	…	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	…	0
10	0	0	1	0	0	1	0	0	0	…	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	…	0

Fig. 2. Analytic user's view of *FLAT* database relation.

3.1 Data integration with analytic bitmap

The definition and materialization of bitmaps corresponding to analytical views defined by the user by means of query expressions in OLAP or data mining processes can aid the user to discover logical relationships among data. In fact, analytical bitmaps represent the implementation of a personalized vision closed to a certain user's intuition in an explorative data analysis. At this level, the user itself can resolve, also dynamically, some conceptual data integration in the data warehouse that are most difficult to solve with predefined ETL (back end) processes and, however, impossible to solve dynamically during the (front end) decisional process.

With reference to the surface attribute of the *FLAT.Data.* relation in Figure 1, all the instanced flats have different surface values. The four analytical views defined on this attribute (*viz.*, q_6, q_7, q_8, and q_9) specify only four classes of surface values significant for user investigation. For example, on the basis of q_6, the flats identified by RID (Flat#) value 3 (which has the original surface value of m^2 45), 4 (m^2 86), 8 (m^2 23), and 10 (m^2 100) are considered to produce equivalent surface information. (This reflects in the *FLAT.Bitstring.* file: the 6th bitmap has value 1 in the 3rd, 4th, 8th, and 10th tuples, and 0 otherwise).

As an example of data integration, we consider the problem of measure unit conversion (usually supported by transform tool in the ETL phase). If the user wishes to relate data values expressed in unit *a* (*e.g.*, centimetres) from source α, with data values referring to an homogeneous entity set expressed in unit *b* (*e.g.*, inches) from source β, then the respective relations α*.Data.* and β*.Data.* are not directly comparable. The integration that reflects the desired classification of the user can be obtained by defining the proper analytical queries in α*.View.* (that is, with reference to 'centimetres') and the physically equivalent one (1cm = 0.3937 inches) in β*.View.*. It must be observed that, in this way, no real data conversion is performed (thus, saving time and storage), and that the *.Bitstring.* files contain fully comparable and union–compatible data (the information expressed by boolean values is independent of the measure unit).

3.2 Semantic Join using Bitmaps

When considering the bitmaps of two or more relations, not necessarily distinct among them, the decisional user can perform *analytical joins*, by opportunely combining analytical criteria on the corresponding views and bitmaps, in order to establish conceptual associations among data in the relations. For example, suppose that the proprietors of the instanced flats have instructed a real estate agency to sell their flats. By a suitable preliminary manipulation of (a copy of) the *FLAT* relation, the agency classifies the flats into the following orthogonal categories: de_luxe flats, first_class flats, second_class flats, and economic ones, as exemplified in Figure 3. The actual criteria adopted by the agency are not relevant to the comprehension of the example, and therefore they are not listed. The flat RIDs 1 and 7 have been classified as the luxe ones, flats 2, 5, and 6 as first class, flats 4 and 9 as second class, and the others as economic flats. Moreover, the agency has at its disposal data referring to potential buyers: data include the code of the client and her/his available funds. In order to propose a selected list of adequate flats to each client, the agency states the following classification of the available funds (shown in the *CLIENT* analytic view in Figure 4):

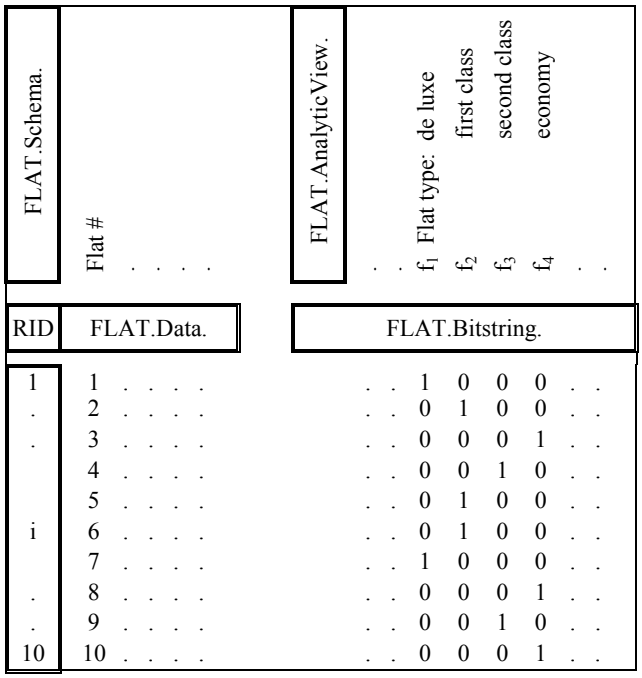

Fig. 3. The compound of flats for sale.

- Very High Availability (*VHA*) is defined as funds availability ≥ 140 (therefore, clients 'a' and 'b' have *VHA*),
- High Availability (*HA*) is defined as available funds in [80,150] (therefore, client 'b' is classified having both *VHA* and *HA*),
- Mean Availability (*MA*) is defined as available funds in [50,90] (therefore, clients 'c', 'e', and 'f' have *MA*), and, finally,

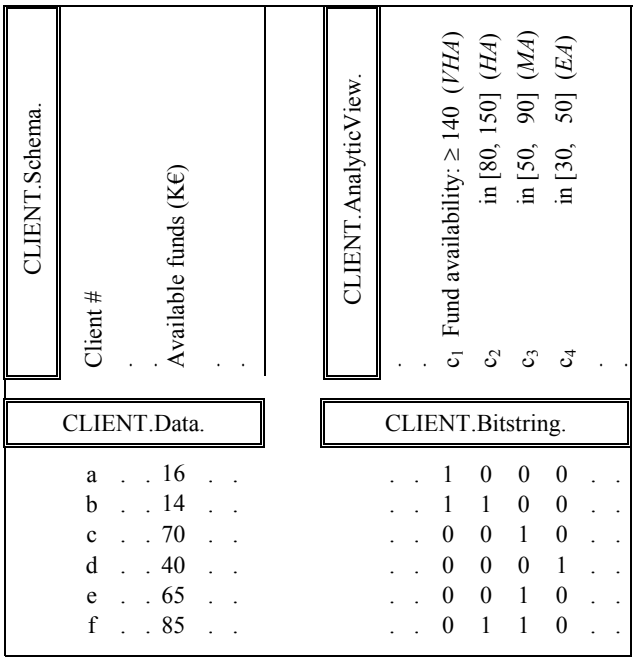

Fig. 4. The compound of possible buyers of flats.

- Economic Availability (*EA*) is defined as available funds in [30,50] (client 'd' has *EA*).

For defining the analytical join of relations *CLIENT* and *FLAT* in order to produce the association between each client and possible flats he/she could buy, the agency has to establish the proper semantic correspondence between types of flat and client's funds availability.

If the association based on the strict semantic correspondence given in Table 1 is adopted, then the corresponding condition of analytical criteria (boolean function of the views given in Figures 3 and 4) is:

$$(f_1 \wedge c_1) \vee (f_2 \wedge c_2) \vee (f_3 \wedge c_3) \vee (f_4 \wedge c_4).$$

The resulting sale proposals are those reported in Table 2 where only the client and flat codes are indicated.

On the other hand, if an enlarged semantics of association of flats to clients has to be adopted, or if the former fails to give the desired results, then Table 1 must be properly extended. Table 3 shows a possible extension that consists of relating each funds availability class to its immediate predecessor and successor classes.

On the basis of table 3, the join criterium

$$(f_1 \wedge (c_1 \vee c_2)) \vee (f_2 \wedge (c_1 \vee c_2 \vee c_3)) \vee$$
$$(f_3 \wedge (c_2 \vee c_3 \vee c_4)) \vee (f_4 \wedge (c_3 \vee c_4))$$

is derived. The resulting sale proposals are those listed in Table 4.

Flat category	Fund availability
de luxe	*VHA*
1st class	*HA*
2nd class	*MA*
economy	*EA*

Table 1. Strict semantic correspondence between types of flat and fund availability.

Client code	List of proposable flats
a	(1, 7)
b	(1, 2, 5, 6, 7)
c	(4, 9)
d	(3, 8, 10)
e	(4, 9)
f	(2, 4, 5, 6, 9)

Table 2. Sale proposal of flats underlying the semantics in Table 1.

Flat category	Fund availability
de luxe	*VHA, HA*
1st class	*VHA, HA, MA*
2nd class	*HA, MA, EA*
economy	*MA, EA*

Table 3. Extended semantics version of Table 1.

Client code	List of proposable flats
a	(1, 2, 5, 6, 7)
b	(1, 2, 4, 5, 6, 7, 9)
c	(2, 3, 4, 5, 6, 8, 9, 10)
d	(3, 4, 8, 9, 10)
e	(2, 3, 4, 5, 6, 8, 9, 10)
f	(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

Table 4. Sale proposal of flats underlying the semantics in Table 3.

4 Conclusions

In decision support activities, it is very important to provide the user with feasible strategies to answer complex queries, besides furnishing fast query response time. The bitmap indices are successfully used in data warehouse systems where prevalently read-only attributes are collected, so avoiding expensive index update. At the same time, bitmaps offer the possibility of a simple index organization and fast data access for answering to complex queries. This takes advantage from the efficiency of the execution of basic logical machine instructions

on the bitstrings.

In this paper, we have shown that, in addition to the commonly recognized advantages in the physical organization of data warehouse systems, bitmaps can be used to represent information at the conceptual level that can be processed efficiently. Examples of analytical data integration in the data warehouse and principally of conceptual associations among relations using bitmaps based and constructed on analytical user criteria have been shown.

Future work regards the implementation of the system interface to obtain an easy and dynamic definition of the bitmaps representing the user views of data expressed by analytical criteria.

References:

[1] Agrawal D., El Abbadi A., Singh A., and Yurek T., Efficient View Maintenance at Data Warehouses, *Proc. of the 1997 ACM SIGMOD Conf.,* Tucson, Arizona, 1997, pp. 417-427.

[2] Amer-Yahia S., and Johnson T., Optimizing Queries on Compressed Bitmaps, *Proc. of the 26th VLDB Conf.,* Cairo, Egypt, 2000, pp. 329-338.

[3] Chan C.Y., and Ioannidis Y. E., An Efficient Bitmap Encoding Scheme for Selection Queries, *Proc. of the 1999 SIGMOD Conf.,* Philadelphia, Pennsylvania, 1999, pp. 215-226.

[4] Chan C.Y., and Ioannidis Y. E., Bitmap Index Design and Evaluation, *Proc. of the 1998 SIGMOD Conf.,* Seattle, Washington, 1998, pp. 355-366.

[5] dell'Aquila C., Lefons E., and Tangorra F., Building Data Warehouse: Design and Case Analysis, *WSEAS Transactions on Information Science and Applications,* vol. 3, no. 3, 2006, pp. 510-517.

[6] dell'Aquila C., Lefons E., and Tangorra F., Approximate Query Processing in Decision Support System Environment, *WSEAS Transactions on Computers,* vol. 3, no. 3, 2004, pp. 581-586.

[7] dell'Aquila C., Lefons E., and Tangorra F., Decision Portal Using Approximate Query Processing, *WSEAS Transactions on Computers,* vol. 2, no. 2, 2003, pp. 386-392.

[8] Gupta H., Harinarayan V., Rajaraman A., and Ullman J.D., Index Selection for OLAP, *Proc. of the 1997 IEEE ICDE Conf.,* Birmingham, UK, 1997, pp. 208-219.

[9] Johnson T., Performance Measurements of Compressed Bitmap Indices, *Proc. of the 25th VLDB Conf.,* Edinburgh, Scotland, 1999, pp. 278-289.

[10] Johnson T., and Shasha D., Some Approaches to Index Design for Cube Forests, *IEEE Data Engineering Bull.,* vol. 22, no. 4, 1999, pp. 22-30.

[11] Mannino M.V., Chu P., and Sager T., Statistical Profile Estimation in Database Systems, *ACM Computing Surveys,* vol. 20, no. 3, 1988, pp. 191-221.

[12] Mumick I.S., Quass D., and Mumick B.S., Maintenance of Data Cubes and Summary Tables in a Warehouse, *Proc. of the 1997 ACM SIGMOD Conf.,* Tucson, Arizona, 1997, pp. 100-111.

[13] O'Neil P., and Quass D., Improved Query Performance with Variant Indexes, *Proc. of the 1997 ACM SIGMOD Conf.,* Tucson, Arizona, 1997, pp. 38-49.

[14] Zhuge Y., Garcia-Molina H., Hammer J., and Widom J., View Maintenance in a Warehousing Environment, *Proc. of the 1995 ACM SIGMOD Conf.,* San Jose, California, 1995, pp. 316-327.

[15] Wu K., and Shoshani A., On the Performance of Bitmap Indices for High Cardinality Attributes, *Proc. of the 30th VLDB Conf.,* Toronto, Canada, 2004, pp. 24-35.