Global Journal of Computer Science and Technology
Volume 11 Issue 23 Version 1.0 December 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

"Globally Recorded binary encoded Domain Compression algorithm in Column Oriented Databases"

By Mehul Mahrishi, Devendra Kr. Sharma, Anita Shrotriya

Suresh Gyan Vihar University, Jaipur, Rajasthan, India

Abstract - through this study, we propose two algorithms. The first algorithm describes the concept of compression of domains at attribute level and we call it as "Attribute Domain Compression". This algorithm can be implemented on both row and columnar databases. The idea behind the algorithm is to reduce the size of large databases as to store them optimally. The second algorithm is also applicable for both concepts of databases but will optimally work for columnar databases. The idea behind the algorithm is to generalize the tuple domains by giving it a value say (n) such that all other n-1 tuples or at least maximum can be identified.

Keywords : *Compression, Columnar database, tuples, tables.*

GJCST Classification : *H.2.4*

"Globally Recorded binary encoded Domain Compression algorithm in Column Oriented Databases"

Mehul Mahrishi[α], Devendra Kr. Sharma[Ω], Anita Shrotriya[β]

Abstract - through this study, we propose two algorithms. The first algorithm describes the concept of compression of domains at attribute level and we call it as "Attribute Domain Compression". This algorithm can be implemented on both row and columnar databases. The idea behind the algorithm is to reduce the size of large databases as to store them optimally. The second algorithm is also applicable for both concepts of databases but will optimally work for columnar databases. The idea behind the algorithm is to generalize the tuple domains by giving it a value say (n) such that all other n-1 tuples or at least maximum can be identified.

Keywords : Compression, Columnar database, tuples, tables.

I. Introduction

Till now we have studied that a database is a collection of inter-related data which is organized in a matrix with rows and columns. Each column represents the attribute of that particular entity which is converted into the database table, while each row of the matrix generally called a tuple represents the different values that an attribute can possess. Each row in a table represents a set of related data, and every row in the table has the same structure.

For example, in a table that represents employee, each row would represent a single employee. Columns might represent things like employee name, employee street address, his SSN etc. In a table that represents the relationship of employees with departments, each row would relate one employee with one department.

Table 2.1 : a Typical Row oriented Database

	Column 1	Column 2	Column 3
Row 1	Row1 & Column 1	Row1 & Column 2	Row1 & Column 3
Row 2	Row2 & Column 1	Row2 & Column 2	Row2 & Column 3

Author [α] *: Dept. of Information Communication, Suresh Gyan Vihar University, Jaipur, Rajasthan, India. E-mail : mehul.aqua@gmail.com*
Author [Ω] *: Dept. of Information Communication, Suresh Gyan Vihar University, Jaipur, Rajasthan, India.*
E-mail : er.devendrasharma@gmail.com
Author [β] *: Dept. of Information Communication, Suresh Gyan Vihar University, Jaipur, Rajasthan, India.*
E-mail : anitashrotriya@gmail.com

II. Rise of Column Database

The relational databases present today are designed predominantly to handle online transactional processing (OLTP) applications. A transaction (e.g. an online purchasing a laptop through internet dealer) typically maps to one or more rows in a relational database, and all traditional RDBMS designs are based on a per row paradigm. For transactional-based systems, this architecture is well suited to handle the input of incoming data.

Data warehouses are used in almost every large organizations and research states that their size doubles after every third year. Moreover the hourly workload of these warehouses is huge and approximately 20lakhs SQL statements are encountered hourly.

Warehouses contain a lot of data and hence any leak or illegal publication of information risks the individuals' privacy. However, for applications that are very read intensive and selective in the information being requested, the OLTP database design isn't a model that typically holds up well. Business intelligence and analytical applications queries often analyze selected attributes in a database. The simplicity and performance characteristic of columnar approach provides a cost effective implementation.

Column oriented database generally known as *"columnar database"* reinvents how data is stored in databases. Storing data in such a fashion increases the probability of storing adjacent records on disk and hence odds of compression. This architecture suggests a different model in which inserting and deleting transactional data are done by a row-based system, but selective queries that are only interested in a few columns of a table are handled by columnar approach.

As we know that logical and critical queries requires more number of rows that that of physical I/O queries which are comparatively slower queries, the performance gap between row-oriented architectures and column-oriented architecture oftentimes widens as the database grows.

Different methodologies such as indexing, materialistic views, horizontal partitioning etc. are provided by row oriented databases which are rather better ways of query execution, but they also have some disadvantages of their own. For example, in business intelligence/analytic environments, the ad-hoc nature of such scenarios makes it nearly impossible to predict which columns will need indexing, so tables end up either being over-indexed (which causes load and maintenance issues) or not properly indexed and so many queries end up running much slower than desired.

III. Anonymization

Warehouses contain a lot of data and hence any leak or illegal publication of information risks the individuals' privacy. N-Anonymity is a major technique to deidentify a data set. The idea behind the technique is to determine the value of a tuple, say n, such that other remaining n-1 tuples or at least maximum tuples can be identified by the value of n.

The intensity of protection increases with increase the number of n. One way to produce n identical tuples within the identifiable attributes is to generalize values within the attributes, for example, removing city and street information in a address attribute.

There are many ways through which data unidentification can be done and one of the most appropriate approaches is generalization. Various generalization techniques include global recoding generalization multidimensional recoding generalization, and local recoding generalization. Global recoding generalization maps the current domain of an attribute to a more general domain. For example, ages are mapped from years to 10-year intervals.

Multidimensional recoding generalization maps a set of values to another set of values, some or all of which are more general than the corresponding premapping values. For example, {male, 32, divorce} is mapped to {male, [30, 40), unknown}. Local recoding generalization modifies some values in one or more attributes to values in more general domains.

IV. Problem Definition and Contribution

From the very beginning we have cleared that our objective is to make every tuple of a published table identical to at least n-1 other tuples. Identity-related attributes are those which potentially identify individuals in a table. For example, the record of an old-aged male in the rural area with the postcode of 302033 is unique in Table 4.1, and hence, his problem of asthma may be revealed if the table is published. To preserve his privacy, we may generalize Gender and Postcode attribute values such that each tuple in attribute set {Gender, Age, Postcode} has at least two occurrences.

Table 4.1 : Published Table

No.	Gender	Age	Postcode	Problem
01	Male	Young	302020	Heart
02	Male	Old	302033	Asthma
03	Female	Young	302015	Obesity
04	Female	Young	302015	Obesity

Table 4.2 : View of published table by Global recording

No.	Gender	Age	Postcode	Problem
01	*	Young	3020*	Heart
02	*	Old	3020*	Asthma
03	*	Young	3020*	Obesity
04	*	Young	3020*	Obesity

A view after this generalization is given in Table 4.2. Since various countries use different postcode schemes, we adopt a simplified postcode scheme, where its hierarchy {302033, 3020*, 30**, 3***, *} corresponds to {rural, city, region, state, unknown}, respectively.

a) Identifier attribute set

A set of attributes that potentially identifies the individuals in a table is a set of identifier attribute. For example, attribute set {Gender, Age, Postcode} in Table 1a is an identifier attribute set.

b) Equivalent Set (€)

An equivalent set of a table with respect to an attribute set is the set of all tuples in the table containing identical values for the attribute set. Table 4.1 forms a equivalent set with respect to attributes {Gender, Age, Postcode, Problem}. Therefore table 4.2 is the 2-Anonymity view of the table 4.1 since two attribute are used to deidentify the published table.

V. Quality Measures of Anonymization

After the study we can easily conclude that larger the size of equivalent set easier the compression and obviously cost of anonymization is a factor of equivalent set. On the basis of this theory, we can determine that:

$$CAVG = \frac{\left(\sum RECORDS\right)}{\sum €}$$

VI. Domain Compression Through Binary Conversion

We integrate two key methods, namely binary encoding of distinct values and pair wise encoding of attributes, to build our compression technique.

a) Encoding of Distinct values

This compression technique is based on the assumption that the table we have published contains minimum distinct domain of attributes and these values repeat over the huge number of tuples present in the database. Therefore, binary encoding of the distinct values of each attribute, followed by representation of the tuple values in each column of the relation with the corresponding encoded values would transform the entire relation into bits and thus compress it.

We will find out the number of distinct values in each column and encode the data into bits accordingly. For example consider an instant given below which represents the two major attributes of a relation Patients.

Table 4.3 : an instance of relation Student

Age	Problem
10	Cough & Cold
20	Cough & Cold
30	Obesity
50	Diabetes
70	Asthma

Now if we adopt the concept of N-Anonymization with global recording *(refer 4.2)*, we can map the current domain of attributes to more general domain. For example Age can be mapped into 10-Age interval as shown in the figure 4.4.

To examine the compression benefits achieved by this method assume that Age is of integer type and has 5 distinct values as in Table 4.3. Suppose if there are 50 patients then the total storage required by Age attribute will be 50*size of (int) = 50*4 = 200 bytes.

With our compression technique, we find that there are 9 distinct values for age therefore we need the upper bound of log (9) i.e. 4 bits to represent each data value in the Age field. It is easy to calculate that we would need 50*4 (bits) = 200 bits = 25 bytes which are reasonably less.

We call this as our *stage 1* of our compression which just transforms one column into bits. If we apply this compression to all columns of the table, the result will be significant.

Table 4.4 : Representing Stage 1 of compression technique

Age	Problem
10-20	Cough & Cold
30-40	Obesity
50-60	Diabetes
70-100	Asthma

Table 4.5 : Representing Stage 1 with binary compression

Age	Problem
00	Cough & Cold
01	Obesity
10	Diabetes
11	Asthma

b) Paired Encoding

It can be easily seen from the above example that besides optimizing the memory requirement of the relations, above encoding technique is also helpful in reducing redundancy (repetition values) from the relation. That is, it is likely that they are few distinct values of even (column1, column2) taken together, in addition to just column1's distinct values or column2's distinct values. We then represent the two columns together as a single column with pair values transformed according to the encoding. This constitutes Stage 2 of our compression in which we use the bit-encoded database from Stage 1 as input and further compress it by coupling columns in pairs of two, applying the distinct-pairs technique outlined. To examine the further compression advantage achieved, suppose that we couple 'Age' and 'Problem' columns. We can see in our table 4.3 that there are 5 distinct pairs (10, Cough & Cold), (20, cough & cold), (30, obesity), (50, Diabetes), (70, Asthma) and hence our upper bound is log (5) = 2 bits approx. Table 4.6 shows the result of *stage 2* compression.

Table 4.6 : Representing Stage 2 compression

Age	Problem
00	00
01	01
10	10
11	11

After compressing the attribute, pairing or coupling of attributes is done. All the columns are coupled in pair of two in a similar manner. If the database contains even number of columns it is straightforward. If the columns are odd, we can intelligently choose any of the columns to be uncompressed.

Table 4.7 : Representing Stage 2 compression coupling

Age- Problem
00
01
10
11

After this compression technique is applied we can easily calculate the space required i.e.

Before compression: 5*(4) +4*(4) = 36 bytes

After Compression and coupling: 4*2 = 8 bits.

VII. Conclusion

In this study we discuss two different compression techniques embedded with each other to form a "Globally Recorded binary encoded Domain Compression".

The first study defines generalization and discuss its different type in anonymization the attributes. It discusses how to handle a major problem in global recoding generalization, inconsistent domains in a field of a generalized table, and propose a method to approach the problem. The tables in the examples proposed global recoding method based on n-anonymity, and consistency.

The second technique focuses on the extension of existing compression by encoding the domain in binary form and further encoding pairs of column values. It shows how coupling of columns can be effective if attributes are properly rearranged. In particular I found that in most cases it is beneficial to couple the primary key with the column having the maximum number of distinct values. Also, columns with very few distinct values should be paired with columns with a large number of dissimilar values. Functional dependencies should be determined to achieve better compression of related attributes. Overall, a better knowledge of the data distribution leads to better compression. Based on the database and the application environment being targeted, the optimum stage up to which compression is feasible and worthy also needs to be determined, i.e. we need to decide the point at which the extra compression achieved is not worth the performance overhead involved.

References Références Referencias

1. http://www.tpc.org/tpch/spec/tpch2.1.0.pdf
2. Goldstein, J., Ramakrishna, R., Shaft, U.: Squeezing the most out of relational database Systems, In Proc. of ICDE (2000) page 81.
3. Software AG. ADABAS im Uberblick. Technical Report ADA5000D006IB, Software AG,Munich, Germany, January 1994.
4. A Deeper Look at Compression in Analytic Database Systems
5. Integrating Compression and Execution in Column-Oriented Database Systems
6. Jorge Vieira1, Jorge Bernardino2, Henrique Madeira3 Efficient compression of text attributes of data warehouse dimensions.
7. Till Westmann1∗ Donald Kossmann2 Sven Helmer1 Guido Moerkotte1 efficient compression of text attributes of data warehouse dimensions