Some Extensions to the Multidimensional Data Model

Ina Naydenova, Kalinka Kaloyanova
“St. Kliment Ohridski” University of Sofia,
Faculty of Mathematics and Informatics, Sofia 1164, Bulgaria
ina@fmi.uni-sofia.bg, kkaloyanova@fmi.uni-sofia.bg

Abstract

Business intelligence applications involve complex queries on very large databases. Users typically view the data as multidimensional data cubes. Computing multidimensional aggregates in large data cubes is a performance bottleneck for many OLAP applications. Calculating the answer of an aggregation query can be too expensive in terms of time and storage space. In this paper we describe some of the problems that could arise in the process of building multidimensional applications with Oracle OLAP Option. We pay special attention to the sparsity of high-dimensional data cubes. We present some extensions to the common multidimensional data model which could solve the described problems. They also enable a more flexible interface not only for the developer of an OLAP application but for the end users as well.

1. Introduction

Online analytical processing (OLAP) is a term that has been used since the early 1990s to describe a class of computer systems designed and optimized for analysis. End users of OLAP systems can easily perform unpredictable ad hoc queries and have the results displayed in a various layouts [1].

During the development of a business intelligence system with Oracle OLAP Option we came across some difficulties related to the user requirements and the high-dimensionality of the requested reports. The solution to those problems gave us some ideas for an extension of the multidimensional data model. The extensions could help overcome the difficulties and also enable new features for users of OLAP systems. We start with a brief description of the logical multidimensional model and the Oracle OLAP. Then we consider the problems that we came across during the development of our information system, and finally we present ideas of how to extend the multidimensional model to avoid the encountered problems.

2. Multidimensional Data Model and Oracle OLAP Option

OLAP systems organize data using the multidimensional paradigm. Human thinking is multidimensional by nature. When you ask questions, you put restraints, which formulate the questions in many dimensions. The factors, which influence the company business (for example: time, products, geography) are taken as dimensions in the multidimensional model. This way you receive a hypercube. Through the analysis process the user can see the data from different view points, in different sections and solve specific problems.

There have been many attempts to define a common basis for multidimensional data models ([9], [10], [11], [12], [13], [14], [15]). According Oracle OLAP option a multidimensional data model is constructed based on a set of dimensions, measures and data cubes. Measures are the objects of analysis. They represent factual data. The dimensions are structure attributes of the measures. They are organized in hierarchies of categories and levels. Dimensional attributes are non-hierarchical identifiers that provide additional information about the data (they can be applied to dimension members at any level of summarization within the model). Cubes provide means of organizing measures that have the same shape, that is, they have exactly the same dimensions. Measures in the same cube have the same relationships to other logical objects and can easily be analyzed and displayed together.

Oracle OLAP provides the query performance and calculation capability previously found in multidimensional Express platform. Multidimensional data is stored in analytic workspaces (AW), where it
can be manipulated by the OLAP engine in Oracle Database. The technology that underlies analytic workspaces is based on an indexed multidimensional array model, which provides direct cell access [6].

A basic characteristic of business analysis is hierarchically structured data. Measures can be additive (summed) over certain dimensions and use other aggregation methods (for example min, max, average, weighted average) on other dimensions. They can also be non-additive over any dimension. An analytic workspace initially contains only base-level data loaded from its data sources. Aggregates can be stored permanently in the analytic workspace, or for the duration of an individual session, or only for a single query. Aggregation rules identify which aggregates are stored, and which aggregates are calculated on the fly. The creation and maintenance of summary data is a serious issue for database administrators. If no summary data is stored, then all summarizations must be performed in response to individual queries. This can easily result in unacceptably slow response time. At the other extreme, if all summary data is stored, then the database can quickly multiply in size [6]. Data stored in an analytic workspace is typically very sparse (that is, there are many empty cells), particularly at lower levels of the data model. To help manage the physical storage of sparse data a composite dimension can be used [8]. A composite dimension contains tuples of dimension member values from two or more dimensions. Oracle OLAP contains a Sparsity Advisor and an Aggregate Advisor. These advisors analyze the tables in a relational scheme and provide recommendations for data storage in an analytic workspace [6].

Version 10g of Oracle OLAP provides new data storage and aggregation algorithm optimized for sparse data. You can build the so called “compressed cubes”. The compressed cube technology deals with the sparsity patterns in hierarchical data sets by mapping more that one cell within a node of a hierarchy to a single stored value rather than replacing that same value up to the hierarchy [7]. Unfortunately at present this option still has many limitations. You can also partition your cube along with any dimensions and at any level to improve the build and aggregation performance of your AW.

3. The problems

When there is a relatively large number of empty cells in a cube, the cube is “sparse”. Sparsity refers to a natural phenomenon, evident in all multidimensional data to some degree: not all of the cells in the logical cube (the total possible combinations of all the dimension members for each dimension of the cube) will ever contain data. It is very common for a relatively small percentage of the possible combinations to actually store data. For example, if you are a manufacturer of consumer packaged goods, you do not sell one or more of every single product you make to every customer, every day, through every sales channel. Different customers buy different products, at different time intervals, and each customer probably has a preferred channel. Different products may display different sparsity patterns: Ice creams and cold drinks tend to be sold faster in summer, while warm arctic coats are more popular in winter (particularly in cold locations) [1].

First of all, we want to make the following specification: we divide the cube sparsity in two types - random and regular sparsity. If one cell is empty because of the semantic of the modelled business area (the semantic enforces lack of value), then we witness “regular sparsity”. If the cell is empty, but it is possible to have a value, we talk about “random sparsity”. Missing data in random sparsity usually expresses zero values, whereas the regular sparsity expresses inapplicable values. For example in the market basket problem we can observe random sparsity. In a grocery store, there might be 10000 products, and the average size of a basket (the collection of distinct items that a customer purchases in a typical transaction) is on average 50 products. For the rest 9950 products we have no values but in fact their meaning is “zero”. If the customer has bought only 2 bars of chocolate he has bought 0 ice-creams, 0 wafers etc.

All the problems that we observed are related to the regular sparsity, we just have defined. We have the following situation: a holding that consists of several separate companies - insurance companies, banks and superannuation fund. We want to have a unified point of view of our holding clients. It is important for us to know how many clients we have for each service that we offer in each geographic region. We want to distinguish also our clients according to their type: private persons or organizations. For our private person clients we save information like their gender, age, education, marital status and so on. For our corporate clients we save information like their branch of business, type of company etc.

3.1. Irrelevant dimensions

The first problem that we are going to discuss is the problem of irrelevant dimensions.

Example: We have designed a cube varying over 7 dimensions: time, services, sales channels, regions, client types, gender and branch of business. When we
select client type to be “private person”, the measure
doesn’t have values by the dimension branch of
business. Vice versa when we choose “organization”,
gender is irrelevant. This is regular sparsity because
we know in advance that some dimensions become
irrelevant for some known members or tuples from the
rest of the dimensions.

We can choose other service, region, sales channel
or period of time but the highlighted areas in figure 1

incompatibility of some dimension segments with the
members of other dimension/dimensions.

3.3. Dimensions changed over time

The problem with modification of dimensions
during the life cycle of business intelligent application
is well known. Those changes could provoke
reorganization of the hierarchical structure or only add

are always empty.

Description: For specific measures (or the entire
cube) we have a sparse region that occurs because of
incompatibility of the whole dimension/dimensions
with the members of other dimension/dimensions.

3.2. Segmentation of dimensions

This problem is similar to the previous as far as it
generates sparse regions in the cube, but its nature is
different.

Example: Let’s look at the services dimension.
Because our holding is made up of companies with
different business areas, the services that it offers to its
clients are various. There are services that are specific
for some company or for companies that have similar
business. For example a service like “deposits” is
relevant only for bank companies. The situation with
organization units is similar – many of the geographic
units belong only to one company and offer only
specific services.

Because there are no insurance offices in Varna,
Stara Zagora and Burgas (figure 2), services that are
related with such type of business enforce sparsity. Other services are connected with bank business and
are relevant only for offices of the bank companies.

Description: For specific measures (or the entire
cube) we have sparse areas that occur because of

new members while many of the old ones are not yet
up-to-date.

Example: Our holding decides to promote new
services and stop offering some of the current ones.
We cannot remove old services because for the old
time periods there is data that we are interested in, but
from the specific period there is no further data about
these services. The newly added service also generates
a lot of sparsity because this service doesn’t exist for
the old periods.

Description: We have sparse regions that are a
result from the introduction of new or retired
dimension members.

3.4. What to aggregate and store preliminary

Often it is difficult to decide what part of the data
to aggregate preliminary and store it in database and
what part of the data to calculate on the fly. Choosing a
balance between static and dynamic aggregation
depends on many factors including disk space,
available memory, the frequency of the queries and
others. If you could decide what percentage of the data
to preaggregate, you can use tools like Oracle
Aggregate Advisor. This advisor analyzes the
distribution of dimension members within hierarchies
and identifies a set of dimension members to
preaggregate but it doesn’t consider factors like
frequency of the queries and users’ behavior.
If you deal with high dimensional cubes it is very likely that many of the possible intersections are never used while others are repeatedly selected.

3.5. Our approach

Every multidimensional engine offers solutions about sparse cubes. With Oracle OLAP option you can use conjoint and composites, compression, partitioning, set different storage parameters. But our application has specific characteristics and we cannot use some of them or their usage is not powerful enough for building application with good time and space performance. We overcame the difficulties but this was related with additional work and compromises. The support of the application is also a complex process. We could not implement initially designed cubes so we redesigned the model of the system. The main strategy of our approach was to break down sparse cubes to smaller compact subcubes. Let’s get back to the example for irrelevant dimensions. Instead of the described cube we could create 3 derived cubes: one for private person clients varying over time, services, sales channels, regions and gender, another for clients that are organizations varying over time, services, sales channels, regions and branch of business, and finally the most compact cube varying over time, services, sales channels and regions that consists of aggregated layers for all holding clients regardless of their type.

4. Extensions of multidimensional data model

4.1. Map of feasible tuples

One of the main characteristics of regular sparsity is the knowledge of it existence in advance. We want to subjoin that knowledge to the multidimensional model as an additional object that we call a permissive map or map of feasible tuples (later shortened to map). This object describes dense areas of the cube or potential non-empty cells. One of the possible ways to do that is to use a negative approach and to describe regular sparsity. Because the cells of a cube could be more than a million that process must be very flexible and handy. Under the observation of the described problems it is important to have an easy way to indicate irrelevant and segmented dimensions. For example, a well-designed rule based system is a proper solution for the population of our map. The map could be associated to specific measures, to selected cubes or to the whole model (all cubes).

Preliminary knowledge of available combinations enables the following opportunities:

- avoiding unnecessary space overhead on physical level and hence redundant time overhead;
- developing a strainer module to automatically hide/avoid the areas of regular sparsity. When a business analytic selects some dimension member, the members of the other dimensions will be restricted to the set of meaningful tuples. Such functionality will be very convenient when we have dimensions changed over time. For a fixed period of time the end user of the OLAP application will see the business such as it is at the selected point of time;
- developing decomposition module for intelligent splitting of reports with many empty cells to subreports without missing values

Example: Let’s go back to our irrelevant dimension example cube. To describe the sparse regions we can use the following rules:

   Rule1: When client types=‘organization’ then gender = ‘All’.
Rule2: When client types = ‘private person’ then branch of business = ‘All’.

If the cardinalities of dimensions member sets are as follows \(|\text{time}| = n_1, |\text{services}| = n_2, |\text{sales channels}| = n_3, |\text{regions}| = n_4, |\text{client types}| = n_5, |\text{branch of business}| = n_6 \text{ and } |\text{gender}| = n_7, we declare that \(n_1, n_2, n_3, n_4, (2.n_6 - n_6 - n_7)\) points of multidimensional cube space are irrelevant. Let’s assume that an OLAP engine uses B-tree composite index over all dimensions to manage the access to data cube cells. The size of the index entries is specified by the size of every member in the tuple \((m_1, m_2, m_3, m_4, m_5, m_6, m_7)\). According to rule 1 in leaves of the tree it is impossible to have tuples where \(m_3\) is equal to ‘organization’ and \(m_4\) is different by ‘All’. Then we can decrease the size of about \(n_1,n_2,n_3,n_4,n_5\) entries (some of dimension combinations are not present in the leaves because of the random sparsity) because the value of gender determined when client type is ‘organization’. In the same way the size of potentially \(n_1,n_2,n_3,n_4,n_7\) entries could be decreased on the basis of rule 2.

Now, let’s imagine that in a certain moment of time \(t_k\) we stop offering the service \(s_j\). We need of a new rule:

When time > \(t_k\) then services = all except \(s_j\).

In the process of typical business intelligence slice and dice operation, the end user of the system can fix time dimension to \(t_{k+1}\). In fact, he is interested in a specific part of the entire cube. The strainer module reads the rules of the map, preprocesses the user request and returns a reduced sub-cube without \(s_j\) layer. The module facilitates the navigation through data after and before \(t_k\) because the user receives relevant data automatically, otherwise he must do that manually by means of different saved selections before and after \(t_k\). It could be a very long-winded process especially when the business is dynamic.

When we have an appropriately constructed map of feasible tuples the program implementation of the strainer module will be simple. More interesting is the realization of decomposition module. This requires proper algorithm to coordinate transformation to minimize dense areas in the selected multidimensional space and return several derived sub-cubes. Note that if we apply the method bearing in mind not only the regular but also the random sparsity we could use it like a method for extraction of hardly perceptible and potentially useful information from data (knowledge discover method).

4.2. Dimension classes

The idea for extensions of the model with object-oriented techniques is not new, but we are going to present the benefits of such techniques with respect to the context of our real case. If you have designed a lot of similar cubes to eliminate the problem with irrelevant or segmented dimensions it will be good for you to have some object-oriented techniques. For example the ability to define dimension classes and to inherit them will facilitate definition and support of derived cubes.

Example: We have 5 cubes – one for every company in the holding. All of them varying by time, regions, client types, gender and branch of business and 2 additional dimensions for every cube represented services and sales channels in the company. Therefore we have 2 sets of 5 dimensions that have equal attributes, levels and hierarchies, similar labels etc. If changes in services or sales channels are needed we must introduce them repeatedly, 5 times. We want to declare dimension class ‘Services’ and define 5 instances – one for every company. Every instance can have specific features, but inherit common properties.

In fact, common dimension attributes are not a special case. Almost all dimensions (over 40) in our system have an attribute “Code” that presents a specific code of used nomenclatures.

4.3. Links between cubes

One useful extension to the multidimensional model is also the ability for defining links between cubes. If you have a report that shows a number of clients distributed by type of clients, you may want to
see a detailed report only for clients that are private persons. You are interested in people involved in the aggregated number but distributed according to their gender, education, marital status and so on. Since listed characteristics are specific for only the set of private persons we suppose that they are organized in a separate data cube. The easiest way to do that is to jump among cubes. Links between created measures or cubes could define suitable shortcuts for the end users. If there are common dimensions between related cubes, every jump determines a particular slice of the target cube (figure 3).

4.4. Saving thresholds

When Oracle OLAP users work with business intelligence application, in fact, they work with a copy of the analytical workspace. In their ad hoc queries often figure data cells are not preliminary stored and OLAP engine calculates them online. There is an option to cache data over session and results of the engine calculation are stored in a temporary analytical workspace. As we mentioned in high dimensional data cubes it is very likely that most of the possible intersections are never used. The ability to save online calculated cells permanently in the original workspace could avoid the problem of beforehand aggregation of data that are never used. If we have a counter how many times one cell was calculated on the fly we could set some threshold that indicates which cell must be preserved. Thus the huge time that is needed for preliminary aggregation will be delayed and distributed over time. Database administrators can adjust the values of saving thresholds according to the resources of the system – available free space and required response time.

5. Conclusions

One of the serious problems in the field of online analytical processing is connected with phenomena like data sparsity and high dimensionality. The available solutions are not good enough – they are concentrated mainly on the physical optimization and methods of compression. There are still no flexible techniques to handle dimensions changes over time. Researches in that sphere haven’t stopped. An idea for fragmentation of high dimensional cubes we found in [5]. In this paper we outline some general ideas of how to extend the common multidimensional model. We discussed the sparsity problems with irrelevant dimensions, segmented dimensions, dimensions changed over time and determining which of the data to be preaggregated. Then we proposed extensions (like a map of feasible tuples, links between cubes, definition of dimensional classes and their inheritance, thresholds for preservation of online calculations), that could help solve the described problems. The extensions mainly affect the developer’s point of view. They also impact on the end user because that enables additional functionality for analysis, but the model still looks so simple. We look forward to working on their improvement and, eventually, implementation of our proposals.

Acknowledgements: This paper is supported by the projects SUGrid (Cr. N. NSF-110/2005) and Sofia University SRF under Contract No. 91/2006.

References

4. Oracle Corporation, Oracle® OLAP Reference 10g Release 2 (10.2), Part No B14350-01, June 2005.
5. Li. Xiaolei, J.Han, and H.Gonzalez, High-Dimensional OLAP: A Minimal Cubing Approach, Proc.VLDB 2004.