Chapter 2
Summarizing Datacubes:
Semantic and Syntactic Approaches

Rosine Cicchetti
Aix-Marseille Universités, France

Lotfi Lakhal
Aix-Marseille Universités, France

Sébastien Nedjar
Aix-Marseille Universités, France

Noël Novelli
Aix-Marseille Universités, France

Alain Casali
Aix-Marseille Universités, France

ABSTRACT

Datacubes are especially useful for answering efficiently queries on data warehouses. Nevertheless the amount of generated aggregated data is huge with respect to the initial data which is itself very large. Recent research work has addressed the issue of summarizing Datacubes in order to reduce their size. In this chapter, we present three different approaches. They propose structures which make it possible to reduce the size of the data cube representation. The two former, the closed cube and the quotient cube, are said semantic and discard the redundancies captured within data cubes. The size of the underlying representations is especially reduced but the counterpart is an additional response time when answering the OLAP queries. The latter approach is rather syntactic since it enforces an optimization at the logical level. It is called Partition Cube and based on the concept of partition. We also give an algorithm to compute it. We propose a Relational Partition Cube, a novel R-Olap cubing solution for managing Partition Cubes using the relational technology. An analytical evaluation shows that the storage space of Partition Cubes is smaller than Datacubes. In order to confirm analytical comparison, experiments are performed in order to compare our approach with Datacubes and with two of the best reduction methods, the Quotient Cube and the Closed Cube.

DOI: 10.4018/978-1-60960-537-7.ch002

INTRODUCTION

In order to efficiently answer OLAP queries (Chaudhuri and Dayal, 1997), a widely adopted solution is to compute and materialize Datacubes (Gray et al., 1997). For example, given a relation *r* over the schema $\mathcal{R}$, a set of dimensions $\mathcal{D} = \{D_1, D_2, D_3\}$, $\mathcal{D} \subseteq \mathcal{R}$, a measure $\mathcal{M} \in \mathcal{R}$, an aggregate function *f*, the cube operator is expressed as follows:

```
SELECT D_1, D_2, D_3, f(M)
FROM r
GROUP BY CUBE(D_1, D_2, D_3)
```

Dimensions are also called categorical attributes and r a categorical database relation. The given query achieves all the possible *group-by* according to any attribute combination belonging to the power set of $\mathcal{D}$. It results in what is called a Datacube, and each sub-query performing a single *group-by* yields a cuboid. Computing Datacubes is exponential in the number of dimensions (the dimension powerset lattice must be explored), and the problem worsens when very large data sets are to be aggregated.

Datacubes are considerably larger than the input relation. Ross and Srivastava (1997) exemplify the problem by achieving a full Datacube encompassing more than 210 millions of tuples from an input relation having 1 million of tuples. The problem is originated by a twofold reason: on one hand the exponential number of dimensional combinations to be dealt, and on the other hand the cardinality of dimensions. The larger dimension domains are, the more aggregated results there are (according to each real value combination). Unfortunately, it is widely recognized that in OLAP databases, data can be very sparse (Ross and Srivastava, 1997; Beyer and Ramakrishnan, 1999) thus scarce value combinations are likely to be numerous and, when computing entirely the Datacubes (full Datacubes), each exception must be preserved. In such a context, (1) approaches favor the efficiency of OLAP queries to the detriment of storage space or (2) they favor an optimal representation of cubes but OLAP query performances are likely to be debased (Morfonios et al., 2007).

Related Work

The approaches addressing the issue of Datacube computation and storage attempt to reduce at least one of the quoted drawbacks. The algorithms BUC (Beyer and Ramakrishnan, 1999) and HCUBING (Han et al., 2001) enforce anti-monotone constraints and partially compute Datacubes (iceberg cubes) to reduce both execution time and disk storage requirements. The underlying argument is that OLAP users are only interested in general trends (and not in atypical behaviors). With a similar argumentation, other methods use the statistic structure of data to compute density distributions and give approximate answers to OLAP queries (see for details (Morfonios et al., 2007)).

The above mentioned approaches are efficient and meet their twofold objective (reduction of execution time and space storage). However, they are not able to answer whatever query (although OLAP queries are, by their very nature, ad hoc queries).

Another category of approaches is the so-called "information lossless". They aim to find the best compromise between OLAP query efficiency and storage requirements without discarding any possible query (even infrequent). Their main idea is to pre-compute and store frequently used aggregates while preserving all the data (possibly at various aggregation levels) needed to compute on line the result of a not foreseen query. They are mostly found in view materialization research.

The following five methods also fit in the information lossless trend:

- the Dwarf Cube (Sismanis et al., 2002),
- the Condensed Cube (Wei et al., 2002),