Efficient Computation of Data Cubes and Aggregate Views

Leonardo Tininini
CNR - Istituto di Analisi dei Sistemi e Informatica "Antonio Ruberti," Italy

INTRODUCTION

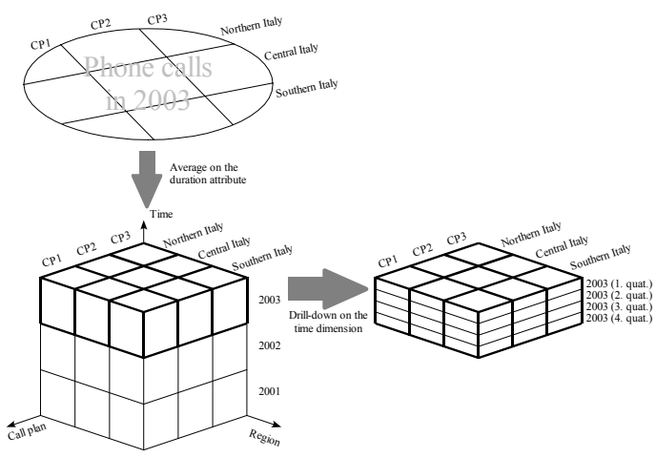
This paper reviews the main techniques for the efficient calculation of aggregate multidimensional views and data cubes, possibly using specifically designed indexing structures. The efficient evaluation of aggregate multidimensional queries is obviously one of the most important aspects in data warehouses (OLAP systems). In particular, a fundamental requirement of such systems is the ability to perform multidimensional analyses in *online* response times. As multidimensional queries usually involve a huge amount of data to be aggregated, the only way to achieve this is by pre-computing some queries, storing the answers permanently in the database and reusing these almost exclusively when evaluating queries in the multidimensional database. These pre-computed queries are commonly referred to as *materialized views* and carry several related issues, particularly how to efficiently compute them (the focus of this paper), but also which views to materialize and how to maintain them.

BACKGROUND

Multidimensional data are obtained by applying aggregations and statistical functions to elementary data, or more precisely to data groups, each containing a subset of the data and homogeneous with respect to a given set of attributes. For example, the data "Average duration of calls in 2003 by region and call plan" is obtained from the so-called fact table, which is usually the product of complex source integration activities (Lenzerini, 2002) on the raw data corresponding to each phone call in that year. Several groups are defined, each consisting of calls made in the same region and with the same call plan, and finally applying the average aggregation function on the duration attribute of the data in each group (see *Figure 1*). The triple of values (region, call plan, year) is used to identify each group and is associated with the corresponding average duration value. In multidimensional databases, the attributes used to group data define the dimensions, whereas the aggregate values define the measures.

The term multidimensional data comes from the well-known metaphor of the data cube (Gray, Bosworth, Layman, & Pirahesh, 1996). For each of n attributes, used to identify a single measure, a dimension of an n-dimensional space is considered. The possible values of the identifying attributes are mapped to points on the dimension's axis, and each point of this n-dimensional space is thus mapped to a single combination of the identifying attribute values and hence to a single aggregate value. The collection of all these points, along with

Figure 1. From facts to data cubes and drill-down on the time dimension



all possible projections in lower dimensional spaces, constitutes the so-called data cube. In most cases, dimensions are structured in hierarchies, representing several granularity levels of the corresponding measures (Jagadish, Lakshmanan, & Srivastava, 1999). Hence a time dimension can be organized into days, months, quarters and years; a territorial dimension into towns, regions and countries; a product dimension into brands, families and types. When querying multidimensional data, the user specifies the measures of interest and the level of detail required by indicating the desired hierarchy level for each dimension. In a multidimensional environment querying is often an exploratory process, where the user "moves" along the dimension hierarchies by increasing or reducing the granularity of displayed data. The drill-down operation corresponds to an increase in detail, for example, by requesting the number of calls by region and quarter, starting from data on the number of calls by region or by region and year. Conversely, roll-up allows the user to view data at a coarser level of granularity.

Multidimensional querying systems are commonly known as OLAP (Online Analytical Processing) Systems, in contrast to conventional OLTP (Online Transactional Processing) Systems. The two types have several contrasting features, although they share the same requirement of fast online response times:

- **Number of records involved:** One of the key differences between OLTP and multidimensional queries is the number of records required to calculate the answer. OLTP queries typically involve a rather limited number of records, accessed through primary key or other specific indexes, which need to be processed for short, isolated transactions or to be issued on a user interface. In contrast, multidimensional queries usually require the classification and aggregation of a huge amount of data.
- **Indexing techniques:** Transaction processing is mainly based on the access of a few records through primary key or other indexes on highly selective attribute combinations. Efficient access is easily achieved by well-known and established indexes, particularly B+-tree indexes. In contrast, multidimensional queries require a more articulated approach, as different techniques are required and each index performs well only for some categories of queries/aggregation functions (Jürgens & Lenz, 1999).
- **Current state vs. historical DBs:** OLTP operations require up-to-date data. Simultaneous information access/update is a critical issue and the database usually represents only the current state of the system. In OLAP systems, the data does not need to be the most recent available and should in fact be time-stamped, thus enabling the user to perform historical analyses with trend forecasts. However, the presence of this temporal dimension may cause problems in query formulation and processing, as schemes may evolve over time and conventional query languages are not adequate to cope with them (Vaisman & Mendelzon, 2001).
- **Target users:** Typical OLTP system users are clerks, and the types of query are rather limited and predictable. In contrast, multidimensional databases are usually the core of decision support systems, targeted at management level. Query types are only partly predictable and often require highly expressive (and complex) query language. However, the user usually has little experience even in "easy" query languages like basic SQL: the typical interaction paradigm is a spreadsheet-like environment based on iconic interfaces and the graphical metaphor of the multidimensional cube (Cabibbo & Torlone, 1998).

MAIN THRUST

In this section we briefly analyze the techniques proposed to compute data cubes and, more generally, materialized views containing aggregates. The focus here is on the exact calculation of the views from scratch. In particular, we do not consider (a) the problems of aggregate view maintenance in the presence of insertions, deletions and updates (Kotidis & Roussopoulos, 1999; Riedewald, Agrawal, & El Abbadi, 2003) and (b) the approximated calculation of data cube views (Wu, Agrawal, & El Abbadi, 2000; Chaudhuri, Das, & Narasayya, 2001), as they are beyond the scope of this paper.

(Efficiently) Computing Data Cubes and Materialized Views

A typical multidimensional query consists of an aggregate group by query applied to the join of the fact table with two or more dimension tables. In consequence, it has the form of an aggregate conjunctive query, for example:

```
SELECT D1.dim1, D2.dim2, AGG(F.measure)
FROM fact_table F, dim_table1 D1, dim_table2 D2
WHERE F.dimKey1 = D1.dimKey1
AND F.dimKey2 = D2.dimKey2
GROUP BY D1.dim1, D2.dim2                    (Q1)
```

where AGG is an aggregation function, such as SUM, MIN, AVG, etc.

For example, in the above-mentioned phone call data warehouse the fact table Phone_calls may have the (sim-