Computation of OLAP Data Cubes

Amin A. Abdulghani
Data Mining Engineer, USA

INTRODUCTION

The focus of online analytical processing (OLAP) is to provide a platform for analyzing data (e.g., sales data) with multiple dimensions (e.g., product, location, time) and multiple measures (e.g., total sales or total cost). OLAP operations then allow viewing of this data from a number of perspectives. For analysis, the object or data structure of primary interest in OLAP is a data cube. A detailed introduction to OLAP is presented in (Han & Kambler, 2006).

BACKGROUND

Consider a 3-D cube model shown in Figure 1 representing sales data. It has three dimensions year, product and location. The measurement of interest is total sales. In olap terminology, since this cube models the base data, it forms a 3-D *base cuboid*. A *cuboid* in general is a group-by of a subset of dimensions of the base data, obtained by aggregating all tuples on these dimensions. So, for example for our sales data we have three 2-d cuboids namely (year, product), (product, location) and (year, location), three 1-d cuboids (year), (location) and (product) and one 0-d cuboid in which aggregation is performed on the whole data. A cuboid which is not a base cuboid is called an *aggregate cuboid* while a 0-D cuboid is called an *apex cuboid*. For base data, with n diemensions, the union of of all *k*-dimensional (k<=n) cuboids forms *an n-dimensional data cube*. A *cell* represents an association of a measure *m* (e.g., total sales) with a member of every dimension in a cuboid e.g. C1(product="toys", location="NJ", year="2004"). The dimensions not present in the cell are aggregated over all possible members. For example, you can have a two-dimensional (2-D) cell, C2(product="toys", year="2004"). Here, the implicit value for the dimension location is '*', and the measure *m* (e.g., total sales) is aggregated over all locations. Any of the standard

Figure 1. A 3-D base cuboid with an example 3-D cell.

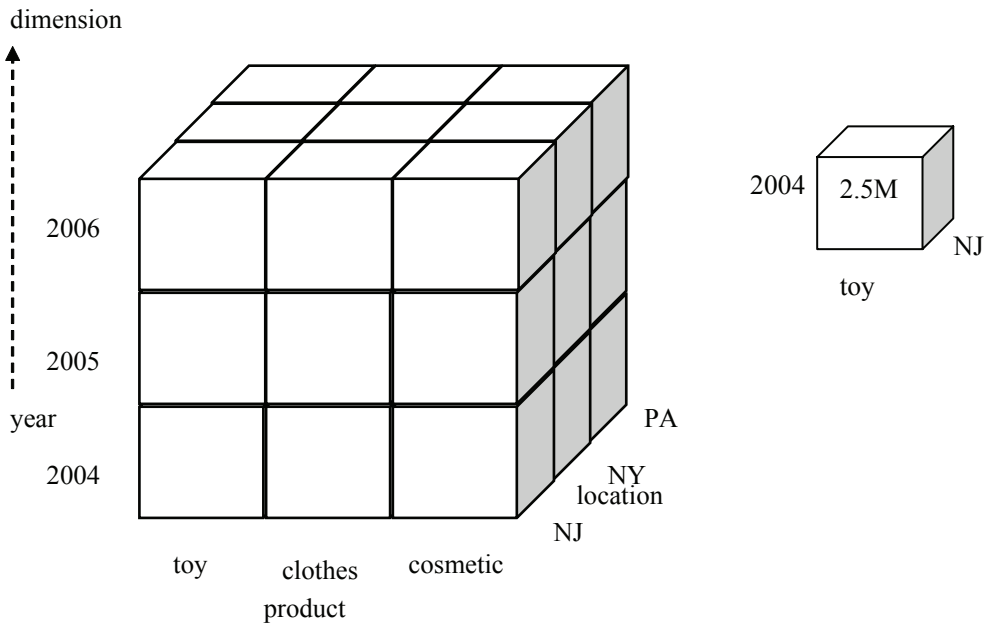
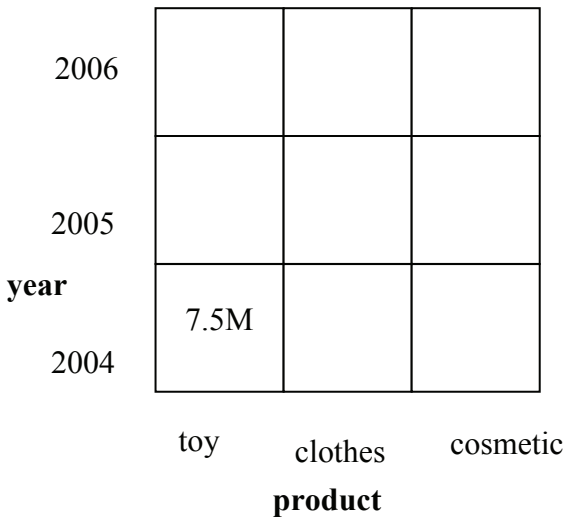


Figure 2. An example 2-D cuboid on (product, year) for the 3-D cube in Figure 1 (location=''); total sales needs to be aggregated (e.g., SUM)*



aggregate functions such as count, total, average, minimum, or maximum can be used for aggregating. Suppose the sales for toys in 2004 for NJ, NY, PA were $2.5M, $3.5M, $1.5M respectively and that the aggregating function is total. Then, the measure value for cell C2 is $7.5M.

In theory, no special operators or SQL extensions are required to take a set of records in the database and generate all the cells for the cube. Rather, the SQL group-by and union operators can be used in conjunction with dimensional sorts of the dataset to produce all cuboids. However, such an approach would be very inefficient, given the obvious interrelationships between the various group-bys produced.

MAIN THRUST

I now describe the essence of the major methods for the computation of OLAP cubes.

Top-Down Computation

In a seminal paper, Gray, Bosworth, Layman, and Pirahesh (1996) proposed the data cube operator.. The algorithm presented there forms the basis of the top-down approach.

The top-down cube computation works with non-holistic functions. An aggregate function F is called holistic if the value of F for an n-dimensional cell cannot be computed from a constant number of aggregates of the (n+1)–dimensional cell. Median and mode are examples of holistic functions, sum and avg are examples of non-holistic functions.The non-holistic functions have the property that more detailed aggregates (i.e., more dimensions) can be used to compute less detailed aggregates. This property induces a partial-ordering (i.e., a *lattice*) on all the group-bys of the cube. A group-by is called a child of some parent group-by if the parent can be used to compute the child (and no intermediate group-bys exist between the parent and child). Figure 3 depicts a sample lattice where A, B, C, and D are dimensions, nodes represent group-bys, and the edges show the parent-child relationship.

The basic idea for top-down cube construction is to start by computing the base cuboid (group-by for which no cube dimensions are aggregated). A single pass is made over the data, a record is examined, and the appropriate base cell is incremented. The remaining group-bys are computed by aggregating over already computed finer grade group-by. If a group-by can be computed from one or more possible parent group-bys, then the algorithm uses the parent smallest in size. For example, for computing the cube ABCD, the algorithm starts out by computing the base cuboid for ABCD. Then, using ABCD, it computes the cuboids for ABC, ABD, and BCD. The algorithm then repeats itself by computing the 2-D cuboids, AB, BC, AD, and