OLAP
Slicing – Sizing – Rising
A Theory of Datacube and Pivot-Tables

Tony Hürlimann
info@matmod.ch

November 12, 2023

Abstract

A pivot table is a powerful means to represent structured, multi-dimensional data on a two-dimensional space. They enable a user to show a large amount of structured data from different angles and perspectives.

This paper gives a survey and implementation in LPL on datacube and pivot-tables in order to understand and use OLAP functionalities. It is argued that slicing, sizing, and rising are fundamental data operations of multidimensional datacubes. They are the basic building block of every OLAP tool, hence our efforts are concentrated on them. The datacube operations are explained as functions which transforms n-dimensional datacubes into datacubes of dimensions less, equal or higher than n. All operations are viewed in this perspective. This reduces various kinds of OLAP-operations considerably and looks at them in a new unified way, which also might be interesting in implementing OLAP-tools.

Then this paper exposes the connections of n-dimensional datacubes with their many 2-dimensional representations, the pivot-tables. The many different pivot-table representations of an n-dimensional datacube are also understood in a unified way, such that all representations can be generated from each other by a very limited number of operations on the datacube which could be subsumed under the general operator pivoting. It is shown how a datacube of any dimension is transformed into any pivot-table representation using just pivoting and the mentioned cube operations slicing, sizing, and rising.

It's a little-known fact... Shakespeare was working on an Excel spreadsheet and created a formula that inspired one of his most famous formula:

```
=OR(B2, NOT(B2))
```

Spreadsheet Poem:

You may spreadsheet in columns
You may spreadsheet in rows
But the more you spreadsheet
The faster it grows.

Contents

1 Introduction

Data is an increasing value resource in many contexts, for example, in a company to schedule and monitor effectively the company's activities. Hence, a data manipulation system must collect and classify the data, by means of integrated and suitable procedures, in order to produce in time and at the right levels the synthesis to be used to support the decisional process, as well as to administrate and globally control the company's activity. While **databases** are the place where data are *collected*, **data warehouses** are systems that *classify* the data. According to William Inmon, widely considered the father of the modern data warehouse, a Data Warehouse is a " Subject-Oriented, Integrated, Time-Variant, Non-volatile collection of data in support of decision makin". Data Warehouses tend to have these distinguishing features: (1) Use a subject oriented dimensional data model; (2) Contain publishable data from potentially multiple sources and; (3) Contain integrated reporting tools.

OLAP (On-Line Analytical Processing) is a key component of data warehousing, and OLAP Services provides essential functionality for a wide array of applications ranging from reporting to advanced decision support. According to [www.olapcouncil.org] OLAP "... is a category of software technology that enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the user. OLAP functionality is characterized by dynamic multidimensional analysis of consolidated enterprise data supporting end user analytical and navigational activities including calculations and modeling applied across dimensions, through hierarchies and/or across members, trend analysis over sequential time periods, slicing subsets for on-screen viewing, drilldown to deeper levels of consolidation, rotation to new dimensional comparisons in the viewing area etc.". The focus of OLAP tools is to provide multidimensional analysis to the underlying information. To achieve this goal, these tools employ multidimensional models for the storage and presentation of data. Data are organized in cubes (or hypercubes), which are defined over a multidimensional space, consisting of several dimensions. Each dimension comprises of a set of aggregation levels. Typical OLAP operations include the aggregation or deaggregation of information (roll-up and drill-down) along a dimension, the selection of specific parts of a cube (dicing) and the reorientation of the multidimensional view of the data on the screen (pivoting).

OLAP functionality is characterized by dynamic multi-dimensional analysis of consolidated enterprise data supporting end user analytical and navigational activities including:

- calculations and modeling applied across dimensions, through hierarchies and/or across members trend analysis over sequential time periods (data mining)
- slicing subsets for on-screen viewing
- drill-down to deeper levels of consolidation
- reach-through to underlying detail data
- rotation to new dimensional comparisons in the viewing area

Two main approaches to support OLAP are (1) ROLAP architecture (Relational On-Line Analytical Processing), and (2) MOLAP architecture (Multidimensional On-Line Analytical Processing). The advantage of the MOLAP architecture is, that it provides a direct multidimensional view of the data and is normally easy to use, whereas the ROLAP architecture is just a multidimensional interface to relational data and requires normally an advanced knowledge on

the SQL queries. On the other hand, the ROLAP architecture has two advantages: (a) it can be easily integrated into other existing relational information systems, and (b) relational data can be stored more efficiently than multidimensional data.

Hence ROLAP is based directly on the architecture of relational databases and different venders just extend the SQL standard language in various ways in order to implement the OLAP functionality. For example in ORACLE the cube and rollup operator expands a relational table, by computing the aggregations over all the possible subspaces created from the combinations of the attributes of such a relation. Practically, the introduced CUBE operator calculates all the marginal aggregations of the detailed data set. The value 'ALL' is used for any attribute which does not participate in the aggregation, meaning that the result is expressed with respect to all the values of this attribute.

The MOLAP architecture is basically *cube-oriented*. This does not mean that they are far from the relational paradigm in fact all of them have mappings to it but rather that their main entities are cubes and dimensions. In practice, the cube's data are extracted from databases using standard SQL queries and stored in "cubes", which beside the data extracted contain pre-calculated data in order to speed up the viewing. One of the big advantage of the cube-oriented approach is its intuitive use of generating a multitude of various "views" of the data; another advantage is speed: Having the data in a cube, it easier to reorganize the data than in ROLAP.

In this paper, the cube-oriented architecture is analyzed. In section 2 a definition of the fundamental concepts, such as datacube, is given. In section 3, the operators on datacubes are defined and analyzed. An interpretation of these operations in the light of OLAP is given in section 4. Section 5 introduces the basic concept of pivot-table, which is identified as a 2-dimensional representation of a datacube. Section 6 briefly summarizes the connection to modeling and to LPL. Finally, some reflections about spreadsheets as a modeling tool are given. See also [5], [3], [8].

2 Definition of Datacube

A set of $(n+1)$-tuples $(d_1, d_2, \ldots, d_n, m)$ with $d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n, m \in M$ is called an ***n*-dimensional datacube**. $D_1, D_2, \ldots, D_n$ are finite sets of *members* and are the **dimensions** of the cube. M is also a set of *values* (normally numerical) called the **measurement**.

A dimension is a *structural* attribute of the cube, that is, a list of members all of which are of a similar type in the user's perception of the data. For example, all months, quarters, years, etc., make up a time dimension; likewise all cities, regions, countries, etc., make up a geography dimension. A dimension acts as an index for identifying values within a multi-dimensional array. If one member of the dimension is selected, then the remaining dimensions in which a range of members (or all members) are selected define a sub-cube. If all but two dimensions have a single member selected, the remaining two dimensions define a spreadsheet (or a "slice" or a "page"). If all dimensions have a single member selected, then a single **cell** is defined. A cell can also be identified as a single tuple of the datacube. Dimensions offer a very concise, intuitive way of organizing and selecting data for retrieval, exploration and analysis. A **member** is a discrete name or identifier used to identify a data item's position and description within a dimension. For example, *January*, *1989* or *1Qtr93* are typical examples of members of a time dimension. *Wholesale*, *Retail*, etc., are typical examples of members of a distribution channel dimension.

As an example see the Figure 1. It represents a 3-dimensional cube with three dimensions *time, region*, and *product*. The measurement is *Sale*. The members of

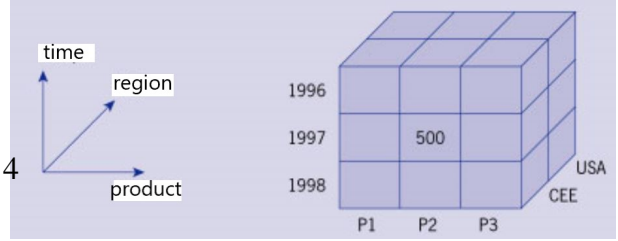


Figure 1: A 3-dimensional cube

time are $\{1996, 1997, 1998\}$, the members of region are $\{CEE, USA\}$, and the members of product are $\{P1, P2, P3\}$. The cell $(1997, CEE, P2)$ contains the value 500. The interpretation is that 500 units of the product P2 has been sold in 1997 in CEE. The cube contains $3 \times 3 \times 2 = 18$ cells.

A member combination is an exact description of a unique cell in the datacube which contains a single value (the measurement). A datacube can also be seen as a multi-dimensional array. A cell – a unique tuple – can be seen as a single data point that occurs at the intersection defined by selecting one member from each dimension in a multi-dimensional array. The maximal number of cells is given by the cardinalities of the dimensions as: $|D_1| \cdot |D_2| \cdot \ldots \cdot |D_n|$. The tuple-set is also called the Cartesian Product of the dimensions. A datacube can be dense or sparse. It is called dense if a relatively high percentage of the possible combinations of its dimension members contain data values, otherwise it is called sparse. It is important to see, that very sparse datacubes are very common in practice.

From a database point of view, an n-dimensional datacube is typically stored as a database table containing $n + 1$ fields, the first n fields representing the dimensions and the $(n + 1)$-th field represents the measurement (the data value). The first n fields are typically (but not necessary) foreign keys pointing to a table filled with basis "identifier" lists. It is important to note that a table is a more general concept than a datacube: (1) The same tuple of members in a datacube – defining a cell – can occur only once in a datacube, while it can occur several times in a database table, except when a primary key is defined on the "dimensional" fields. In praxis, however, this is not a limitation, because we mostly analyze data which can be classified according some dimensions and hence have distinct tuples. (2) A database table – besides the "dimensional" fields – can contain several "measurements" fields. If this is the case, then several datacubes with the same dimensions can be built, or a datacube with an additional dimension, which contains a "measurement" field name as members, if the measurements are all of the same data type (numeric, alphanumeric, etc.). If this is not the case the db table can be mapped to a set of datacubes.

We call the list of tuples defining a datacube – normally printed in a vertical way – on a piece of paper, the *standard view* or the *db-view* of the datacube. As we shall see, this is nothing else than a particular pivot table view.

In a mathematical notation, a cube can be represented as follows:

$$m_{i_1,i_2,\ldots,i_n} \quad \text{forall } i_1 \in D_1, i_2 \in D_2, \ldots, i_n \in D_n, m \in M$$

$i_1, i_2, \ldots, i_n$ are called indexes, and the notation is called the **indexed notation** (see [9]).

3 Operations on Datacubes

Several operations can be applied to datacubes. The result of these operations is again a datacube. The operators are: *slicing, dicing, sizing*, and *rising*. The operation *slicing* on an n-dimensional datacube generates a datacube of dimension $\leq n$, while dicing and sizing generates a datacube of dimension n, and rising generates a datacube of dimension $\geq n$.

3.1 Slicing

Slicing means to *fix* a particular member of one or several of the dimensions (we call them "fixed" dimensions) by discarding all tuples containing any of the other members of the fixed dimensions. Then we remove the particular (fixed) members from all tuples. The result is a datacube with fewer dimensions.

Figure 2 shows a 3-dimensional cube. The left part selects a slice – the grey part – of a 2-dimensional cube by fixing the member **3** of dimension 1. The right part selects a slice of 1-dimensional cube by fixing the member **3** of dimension 1 *and* the member **1** of dimension 3. In both cases, the dimension is reduced and the result is another datacube of lower dimension.

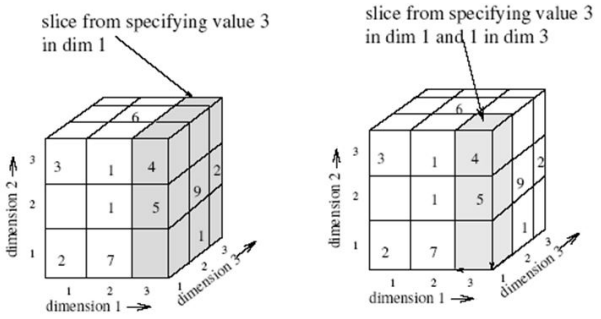


Figure 2: Slicing a Cube

In fact, the operation "slicing on more than one dimensions" can be reduced to a sequence of "slicing on a single dimension". The operation is commutative and associative on the dimensions, meaning that the order which it is applied to the "fixed" dimensions does not matter. "Slicing on a single dimension" reduced the dimension by one. Hence, applying this sliding operation to an n-dimensional cube, generates a $(n-1)$-dimensional cube. One can apply this operation at most n times successively to get a 0-dimensional cube, say a single cell of the original datacube.

In the mathematical notation, slicing is to generate a sub-cube as follows. As an example, let's start with a 3-dimensional cube $a_{i,j,k}$ with $i \in I, j \in J, k \in K$ and $I = \{i_1, i_2, \ldots, i_m\}$, $J = \{j_1, j_2, \ldots, j_n\}$, and $K = \{k_1, k_2, \ldots, k_p\}$. Let's fix the member $k_2 \in K$. The result is a cube of dimension 2, say $b_{i,j}$.

$$b_{i,j} \leftarrow a_{i,j,k_2}$$

means that for each $(i, j) \in I \times J$ tuple in the cube $a_{i,j,k}$, the value of the cell is copied (or mapped) to the corresponding cell in $b_{i,j}$. It is in fact a *matrix-assignment*. Fixing two members in the different dimensions I and K generates a 1-dimensional cube, say c_j:

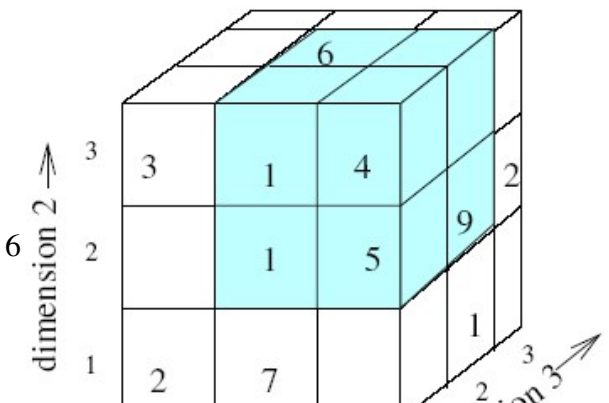
$$c_j \leftarrow a_{i_3,j,k_2} \quad \text{forall } j \in J$$

The operation $\leftarrow$ means that for each (i, j)-combination in the cube a, the value of the cell is copied to the corresponding cell in b. It is in fact *matrix-assignment*.

3.2 Dicing

Dicing (it may be called also *down-sizing*) a n-dimensional datacube means to select a (possibly empty) subset of members from each dimension and discard all tuples which contains at least one member not in the selection (see Figure 3

Note that the resulting datacube has the same dimension as the datacube before applying the operation. Even if the selection on a particular dimension contains a single member, it is not the same as slicing. The dimen-



sion is not reduced and a set containing a single member after all is also a set.

If no member is selected of any dimension, we get the *empty cube*. (Note that the empty cube is not the same than a 0-dimensional cube, which contains a single cell.) If all members are selected from all dimensions, the resulting cube is the same as the original cube.

Mathematically, the operation is simply stated as follows. As above, let's start again with the 3-dimensional cube $a_{i,j,k}$. Let $P_{i,j,k}$ be a predicate (relation) which is true for some (i, j, k)-triples and false for the others. Then dicing $a_{i,j,k}$ on $P_{i,j,k}$ means to create a cube $d_{i,j,k}$ that contains all the (i, j, k)-triples of $a_{i,j,k}$ for which $P_{i,j,k}$ is true. The result is an assignment:

$$d_{i,j,k} \leftarrow a_{i,j,k} \quad \text{forall } (i, j, k) \in P_{i,j,k}$$

A shorter notation is :

$$d_{i,j,k|P} \leftarrow a_{i,j,k}$$

3.3 Sizing

Sizing (it may be called also *up-sizing*) a n-dimensional datacube means to add tuples which contains members in a particular dimension that were not part of that dimension. The dimensions are extended: elements are added to one or more dimensions.

The consequence is an extension of the datacube in one or several dimensions. This operation does not change the dimension of the resulting cube – just its size in one or several dimension is changed (see Figure 4. A particular operation is when only one dimension is extended. Then this operation can also be seen as "merging" a n-dimensional datacube with a $(n-1)$-datacube in which the $n-1$ dimensions occur in both cubes. It is "adding a slice" to the cube.

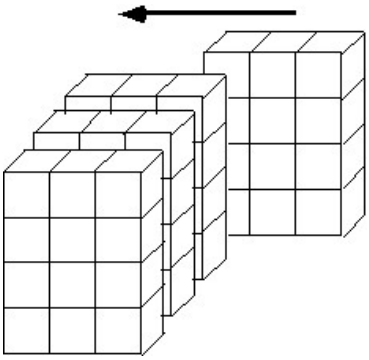


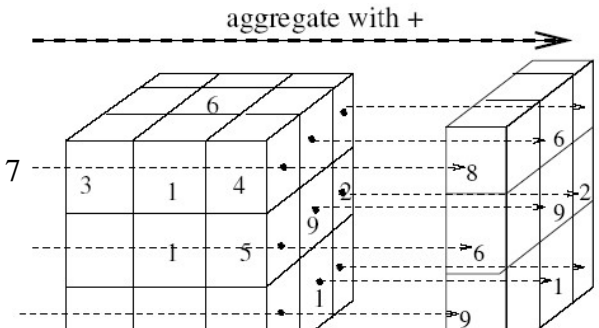
Figure 4: Dicing a Cube

This last special case can be implemented as another assignment (where $'j'_{n+1}$ is a new element in the dimesion J) :

$$f_{i,j,k} \leftarrow \begin{cases} a_{i,j,k} & \text{if } j \in J \\ b_{i,k} & \text{if } j =' j_{n+1'} \end{cases}$$

Another important special case occurs when the added $(n-1)$-slice is an aggregated cube. A *aggregated cube* is constructed from the n-dimensional cube as follows: Choose one dimension out of the n dimensions in this cube, then choose an operation that can be applied to the cells along the chosen dimension (p.e. summation, maximum, etc., for numerical cells) (see Figure 5).

The aggregated cube is an $(n-1)$-dimensional cube. Mathematically, it is a another assignment:



$$b_{i,k} \leftarrow \sum_{j \in J} a_{i,j,k} \quad \text{forall } i \in I, k \in K$$

Of course, one can also aggregate along the other dimensions i and k, in this example, and one gets the 2-dimensional cubes as follows:

$$b'_{i,j} \leftarrow \sum_{k \in K} a_{i,j,k} \quad \text{forall } i \in I, j \in J \qquad b''_{j,k} \leftarrow \sum_{i \in I} a_{i,j,k} \quad \text{forall } j \in J, k \in K$$

The aggregation can be continued to generate 1-dimension cubes:

$$c'_i \leftarrow \sum_{j,k} a_{i,j,k} \qquad c''_j \leftarrow \sum_{i,k} a_{i,j,k} \qquad c'''_k \leftarrow \sum_{i,j} a_{i,j,k}$$

Finally, one gets the 0-dimensional cube by aggregating all cells:

$$d \leftarrow \sum_{i,j,k} a_{i,j,k}$$

Figure 6 shows all possible aggregated cubes from a 0 to 4-dimensional cube. Each node in the graph denotes a aggregated cube and the (down)-links represent the aggregation operation. For example, from the 4-dimensional cube $abcd$, one can generated a 3-dimensional cube $* \; bcd$ by aggregation along the dimension a. This shows that the generation of all aggregated cubes forms a lattice.

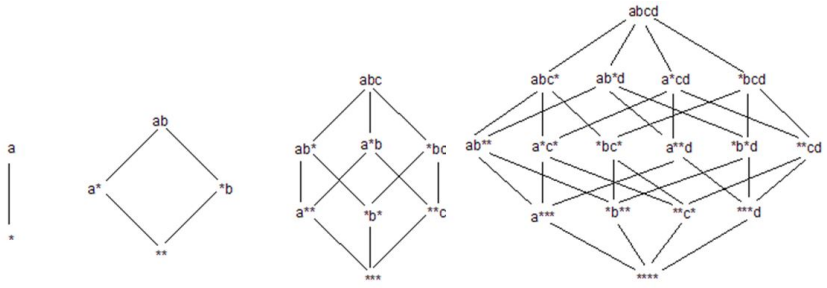


Figure 6: Lattice of All Aggregated Cubes

Represented in another way, the 3-dimensional cube of dimension $2 \times 3 \times 4$ (left-bottom cube in Figure 7) is augmented by all its aggregated cubes.

In general, from a n-dimensional cube with dimension cardinalities $c_1, c_2, \ldots, c_n$, one can generate $\binom{n}{i}$ aggregated i-dimensional cubes. Totally, there are $2^n - 1$ aggregated cubes. All aggregated cubes together with the original cube can be merged together to form a new cube – the *augmented cube*. It is also an n-dimensional cube, for which all dimensions are extended by just one element.

The order in which the aggregated cubes are build to form an augmented cube is not important supposing the augmented operation is commutative and associative. There are

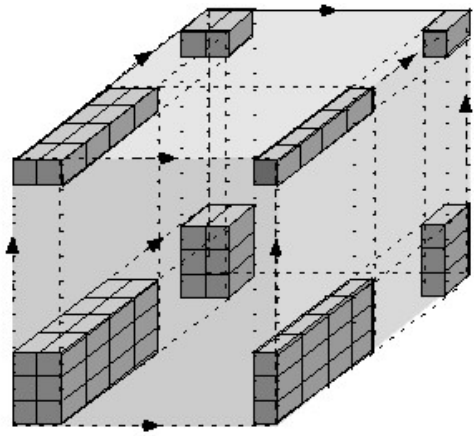


Figure 7: All Aggregated Cubes

several aggregate operations (for numerical data) that can be considered: Sum, Average, Max, Min, Count, Variance, Standard deviation, and others. More than one aggregate operation may be applied at the same time.

3.4 Rising

Rising an n-dimensional cube is an operation that "lifts" the datacube to a higher dimension. This can occur in two important applications. (1) Two datacubes having the same number of dimensions and having the same dimensions can be merged together. Figure 8 displays an example.

This case is especially interesting for comparative studies of two or several datacubes, comparing scenarios and outcoming of the same datacube. In LPL this is implemented as *Multiple Snapshot* Analysis. Rising implies to introduce an additional dimension into the resulting datacube. LPL automatically adds a set called `_SNAP_`. Hence the resulting datacube has then $n + 1$. dimensions.

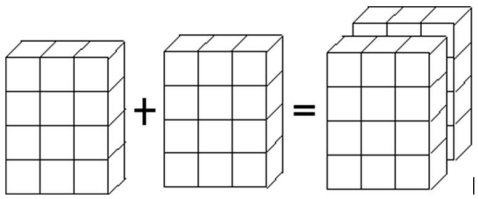


Figure 8: Rising Identical Cubes

(2) The second important application arises when the actual cube has to be partitioned into several groups, for example, along a time dimension, the months, one has to group the dimension along quarters; or along a product dimension, one has to group them into various product categories, etc (see Figure 9).

The original cube is partitioned into the desired parts and build from each part a complete datacube of the original size – by getting eventually a very sparse cube. Then these cubes are risen by "merging" them along a new dimension, which implements the partition, see Figure 10.

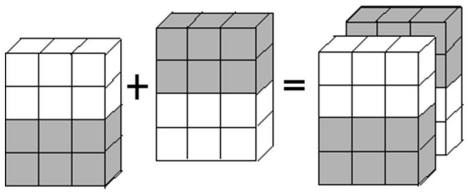
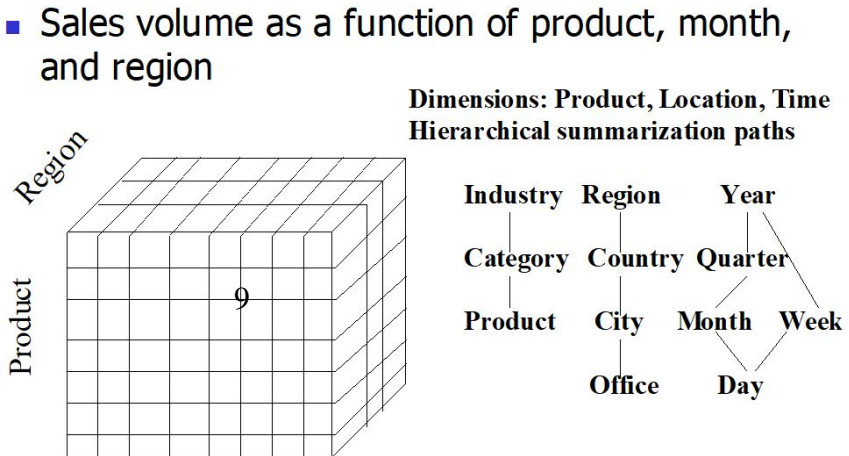


Figure 9: Rising Cubes by Extending and Merging Dimensions

The idea behind this partition of a cube is that certain dimensions can be structured into hierarchies (Year – Month – Week – Day, Continent – Country – Region – State). The partition can be arbitrarily however, it can be even on several dimensions, that is, certain tuples of the cube may belong to one part and other tuples to another part. The two most important applications of this operation are grouping and hierarchy building in OLAP tool and grouping and subgrouping in reports.

3.5 Selecting and Ordering

Selecting a subset of elements in one or several dimension results in a datacube of the same dimension with fewer tuples. This operation can be neatly integrated into a



visualisation tool of datacube. Another similar operation is ordering: The elements in one or several dimension are sorted along a criterion. For example, a distance matrix can be sorted in the order of the distance (global sorting) to get the 5 shortest distance in the matrix, but one could also sort it along one dimension (dimensional sorting), to get the 5 closest neighbors of each location. This may be interesting in a heuristic to find a good TSP tour.

4 Interpretation of the Operations in the Light of OLAP

On-Line Analytical Processing (OLAP) is a category of software technology that enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the user.

OLAP is implemented in a multi-user client/server mode and offers consistently rapid response to queries, regardless of database size and complexity. OLAP helps the user synthesize enterprise information through comparative, personalized viewing, as well as through analysis of historical and projected data in various "what-if" data model scenarios. This is achieved through use of an OLAP Server.

We use the OLAP glossary to go through the different "operations" needed to be able to characterize a system an OLAP, see [www.OLAPCouncil.org].

AGGREGATE (CONSOLIDATE, ROLL-UP)

"Multi-dimensional databases generally have hierarchies or formula-based relationships of data within each dimension. Consolidation involves computing all of these data relationships for one or more dimensions, for example, adding up all Departments to get Total Division data. While such relationships are normally summations, any type of computational relationship or formula might be defined." See Figure 11.

The Roll-Up operation is typically to partition a cube and then to rise it a long the partition. This operation can be repeated several times in order to generate hierarchies of dimensions. Aggregation has been extensively discussed.

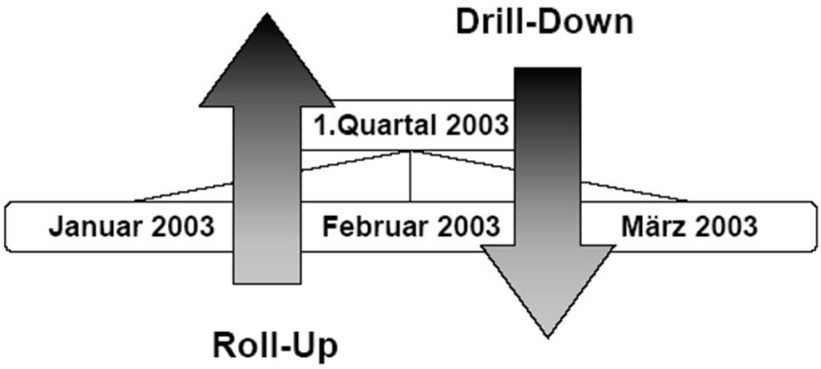


Figure 11: Roll-Op/Drill-Down Operations

DRILL DOWN/UP

"Drilling down or up is a specific analytical technique whereby the user

navigates among levels of data ranging from the most summarized (up) to the most detailed (down). The drilling paths may be defined by the hierarchies within dimensions or other relationships that may be dynamic within or between dimensions. For example, when viewing sales data for North America, a drill-down operation in the Region dimension would then display Canada, the eastern United States and the Western United States. A further drill- down on Canada might display Toronto, Vancouver, Montreal, etc."

The Drill-Down means slicing a cube along the hierarchies defined before by a rolling-up process.

MULTI-DIMENSIONAL ANALYSIS

"The objective of multi-dimensional analysis is for end users to gain insight into the meaning contained in databases. The multi-dimensional approach to analysis aligns the data content with the analyst's mental model, hence reducing confusion and lowering the incidence of erroneous interpretations. It also eases navigating the database, screening for a particular subset of data, asking for the data in a particular orientation and defining analytical calculations. Furthermore, because the data is physically stored in a multi- dimensional structure, the speed of these operations is many times faster and more consistent than is possible in other database structures. This combination of simplicity and speed is one of the key benefits of multi-dimensional analysis."

Multi-dimensional analysis is nothing else than cube manipulation!

CALCULATED MEMBER

"A calculated member is a member of a dimension whose value is determined from other members' values (e.g., by application of a mathematical or logical operation). Calculated members may be part of the OLAP server database or may have been specified by the user during an interactive session. A calculated member is any member that is not an input member."

By rising a cube, one can add member to a dimensions the cells of which are calculated. The aggregates are typical such calculated cells.

CHILDEN AND HIERARCHICAL RELATIONSHIPS

"Members of a dimension that are included in a calculation to produce a consolidated total for a parent member. Children may themselves be consolidated levels, which requires that they have children. A member may be a child for more than one parent, and a child's multiple parents may not necessarily be at the same hierarchical level, thereby allowing complex, multiple hierarchical aggregations within any dimension."

"Any dimension's members may be organized based on parent-child relationships, typically where a parent member represents the consolidation of the members which are its children. The result is a hierarchy, and the parent/child relationships are hierarchical relationships."

"Members of a dimension with hierarchies are at the same level if, within their hierarchy, they have the same maximum number of descendants in any single path below. For example, in an Accounts dimension which consists of general ledger accounts, all of the detail accounts are Level 0 members. The accounts one level higher are Level 1, their parents are Level 2, etc. It can happen that a parent has two or more children which are different levels, in which case the parent's level is defined as one higher than the level of the child with the highest level."

Using rising a cube it was shown how a hierarchy can be implemented.

NAVIGATION

"Navigation is a term used to describe the processes employed by users to explore a cube interactively by drilling, rotating and screening, usually using a graphical OLAP client connected to an OLAP server."

See implementation of pivot-tables in LPL.

NESTING (OF MULTI-DIMENSIONAL COLUMNS AND ROWS)

January				February				March			
Actual		Budget		Actual		Budget		Actual		Budget	
Prod A	Prod B	Prod A	Prod B	Prod A	Prod B	Prod A	Prod B	Prod A	Prod B	Prod A	Prod B

Figure 12: Horizontal Nesting Display

"Nesting is a display technique used to show the results of a multi-dimensional query that returns a sub-cube, i.e., more than a two-dimensional slice or page. The column/row labels will display the extra dimensionality of the output by nesting the labels describing the members of each dimension. For example, the display's columns may be as seen in Figure 12. These columns contain three dimensions, nested in the user's preferred arrangement."

Chocolate Bars	Unit Sales	xxxx	xxxx	xxxx
	Revenue	xxxx	xxxx	xxxx
	Margin	xxxx	xxxx	xxxx
Fruit Bars	Unit Sales	xxxx	xxxx	xxxx
	Revenue	xxxx	xxxx	xxxx
	Margin	xxxx	xxxx	xxxx

Figure 13: Vertical Nesting Display

"Likewise, a report's rows may contain nested dimensions", see Figure 13.

This was consistently implemented into the modeling environment of LPL (lplw.exe).

PAGE DISPLAY (PIVOT, ROTATE, ROW DIMENSION, COLUMN DIMENSION, HORIZONTAL DIMENSION, VERTICAL DIMENSION

"The page display is the current orientation for viewing a multi-dimensional slice. The horizontal dimension(s) run across the display, defining the column dimension(s). The vertical dimension(s) run down the display, defining the contents of the row dimension(s). The page dimension-member selections define which page is currently displayed. A page is much like a spreadsheet, and may in fact have been delivered to a spreadsheet product where each cell can be further modified by the user."

"To change the dimensional orientation of a report or page display we use pivoting. For example, rotating may consist of swapping the rows and columns, or moving one of the row dimensions into the column dimension, or swapping an off-spreadsheet dimension with one of

the dimensions in the page display (either to become one of the new rows or columns), etc. A specific example of the first case would be taking a report that has Time across (the columns) and Products down (the rows) and rotating it into a report that has Product across and Time down. An example of the second case would be to change a report which has Measures and Products down and Time across into a report with Measures down and Time over Products across. An example of the third case would be taking a report that has Time across and Product down and changing it into a report that has Time across and Geography down."

PAGE DIMENSION

"A page dimension is generally used to describe a dimension which is not one of the two dimensions of the page being displayed, but for which a member has been selected to define the specific page requested for display. All page dimensions must have a specific member chosen in order to define the appropriate page for display."

See: "take out/in" operator in the next section.

SELECTION

"A selection is a process whereby a criterion is evaluated against the data or members of a dimension in order to restrict the set of data retrieved. Examples of selections include the top ten salespersons by revenue, data from the east region only and all products with margins greater than 20 percent." See: dicing

SLICE AND DICE

"A slice is a subset of a multi-dimensional array corresponding to a single value for one or more members of the dimensions not in the subset. For example, if the member Actuals is selected from the Scenario dimension, then the sub-cube of all the remaining dimensions is the slice that is specified. The data omitted from this slice would be any data associated with the non-selected members of the Scenario dimension, for example Budget, Variance, Forecast, etc. From an end user perspective, the term slice most often refers to a two- dimensional page selected from the cube."

"The user-initiated process of navigating by calling for page displays interactively, through the specification of slices via rotations and drill down/up."

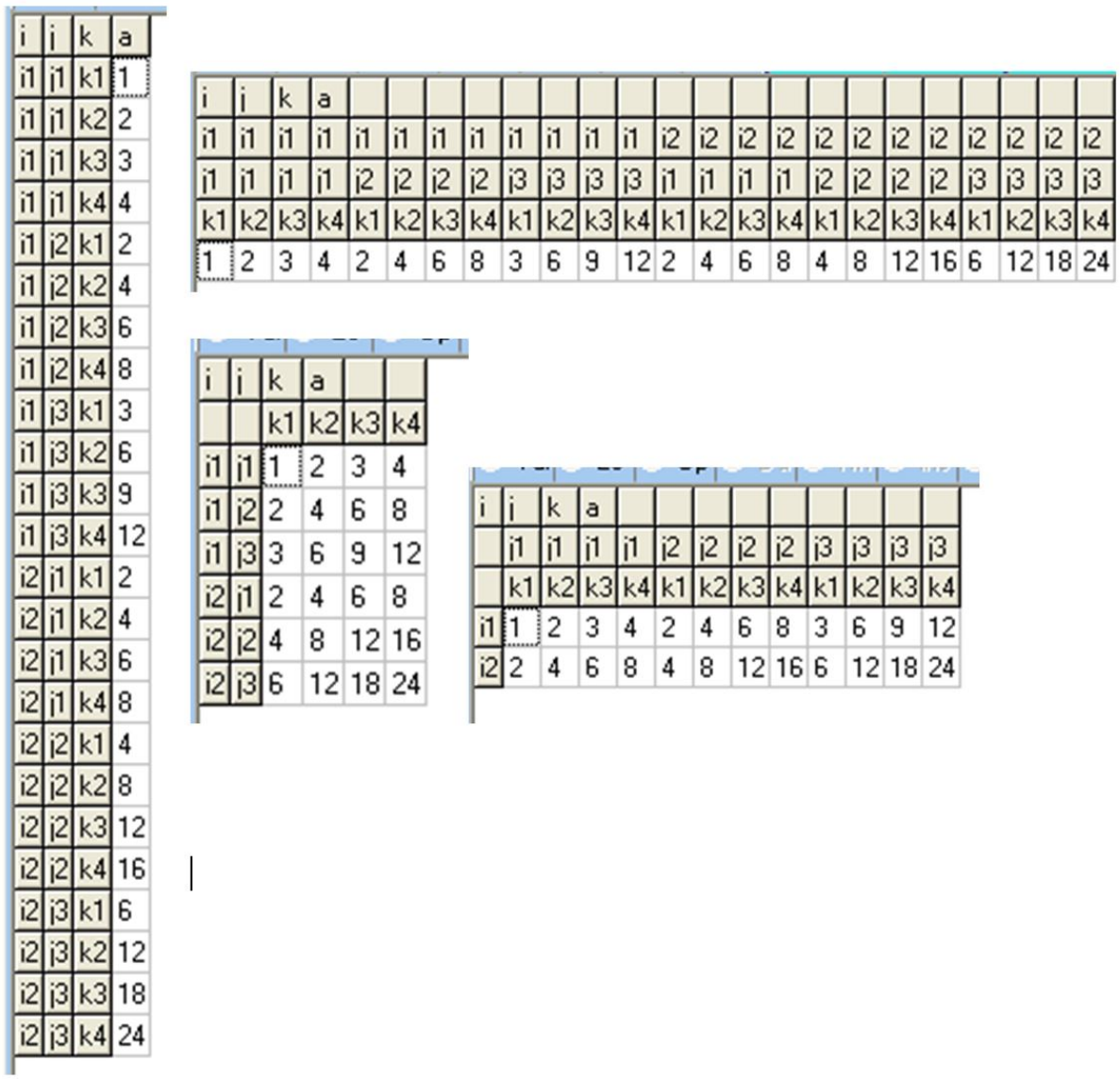
Slicing and dicing have been explained.

5 Pivot-Table: 2-dimensional Representation of Datacube

Datacubes must be represented on sheet of papers for reports or on a screen in order to be viewed by human beings, that is, they must be projected onto a two-dimensional space. We call a two-dimensional representation of a cube **pivot-table**.

Datacubes can be represented in many ways on a two-dimensional space. Such representations are shown in Figure 14. The standard view – or database view, is a vertical (normally top-down) listing of all tuples in the datacube in a given order. Another is the horizontal view, other are mixtures. The Figure shows an example is the 3-dimensional cube $a_{i,j,k}$ with $i \in \{i1, i2\}$, $j \in \{j1, j2, j3\}$, and $k \in \{k1, k2, k3, k4\}$. The measurement of the cube is $i * j * k$. In general for an n-dimensional cube this kind of switching horizontally/vertically the dimensions gives us $n + 1$ pivot-table representations.

Furthermore, one may consider any permutation order on the dimensions, to obtain $n!$ possibilities of pivot-tables. These permutations are a rich source on projecting the datacube on to a 2-dimensional space. We call this going from one permutation to another *pivoting*. Hence the name of *pivot-table*. Three examples of these permutations are shown in Figure 15.



i	j	k	a
i1	j1	k1	1
i1	j1	k2	2
i1	j1	k3	3
i1	j1	k4	4
i1	j2	k1	2
i1	j2	k2	4
i1	j2	k3	6
i1	j2	k4	8
i1	j3	k1	3
i1	j3	k2	6
i1	j3	k3	9
i1	j3	k4	12
i2	j1	k1	2
i2	j1	k2	4
i2	j1	k3	6
i2	j1	k4	8
i2	j2	k1	4
i2	j2	k2	8
i2	j2	k3	12
i2	j2	k4	16
i2	j3	k1	6
i2	j3	k2	12
i2	j3	k3	18
i2	j3	k4	24

i	j	k	a																				
i1	i1	i1	i1	i1	i1	i1	i1	i1	i1	i1	i1	i2	i2	i2	i2	i2	i2	i2	i2	i2	i2	i2	i2
j1	j1	j1	j1	j2	j2	j2	j2	j3	j3	j3	j3	j1	j1	j1	j1	j2	j2	j2	j2	j3	j3	j3	j3
k1	k2	k3	k4	k1	k2	k3	k4	k1	k2	k3	k4	k1	k2	k3	k4	k1	k2	k3	k4	k1	k2	k3	k4
1	2	3	4	2	4	6	8	3	6	9	12	2	4	6	8	4	8	12	16	6	12	18	24

i	j	k	a		
		k1	k2	k3	k4
i1	j1	1	2	3	4
i1	j2	2	4	6	8
i1	j3	3	6	9	12
i2	j1	2	4	6	8
i2	j2	4	8	12	16
i2	j3	6	12	18	24

i	j	k	a									
	j1	j1	j1	j1	j2	j2	j2	j2	j3	j3	j3	j3
	k1	k2	k3	k4	k1	k2	k3	k4	k1	k2	k3	k4
i1	1	2	3	4	2	4	6	8	3	6	9	12
i2	2	4	6	8	4	8	12	16	6	12	18	24

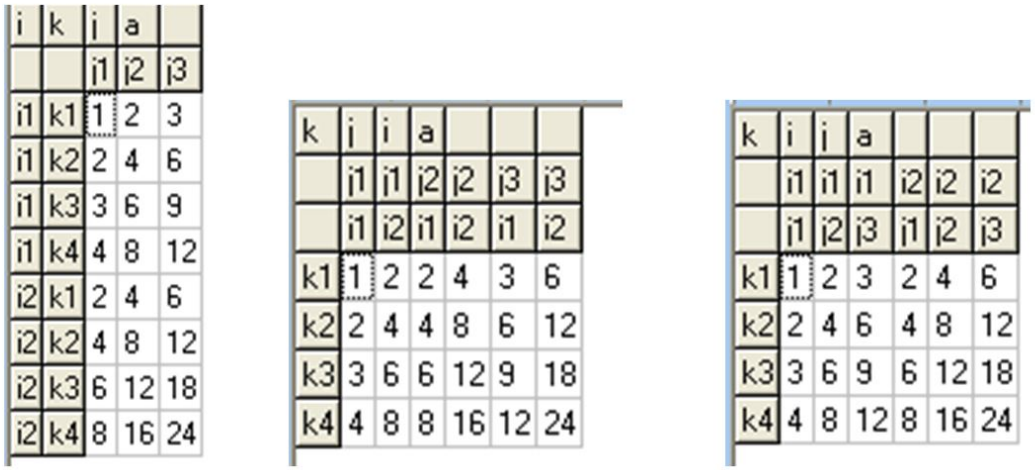
Figure 14: Pivot Tables by Switching Dimensions Horizontally/Vertically

In the left one, 1 dimension is projected horizontally and the permutation is (i, k, j). The second projects 2 dimensions horizontally and the permutation is (k, j, i). The right one also projects 2 dimensions horizontally and the permutation is (k, i, j).

The aggregated cubes can be viewed by "taking out" one or several dimensions from the original cube. So Figure 16 represents the aggregated cubes $b_{j,k} = \sum_i a_{i,j,k}$ and $c_{i,k} = \sum_j a_{i,j,k}$. The aggregate operator was SUM.

To summarize: Given any n-dimensional cube and a permutation on the dimension as well as two numbers h and k and an aggregate operator, one can generate any pivot-table of the cube or one of its aggregated cubes, h being the number of dimensions that are projected horizontally, and k being the number of dimensions of "taken out". The permutation is than interpreted as follows: project the first dimensions vertically, the h following horizontally and the last k once are "taken out". For example, the standard view of a 5-dimensional cube would be given as: $perm = (1, 2, 3, 4, 5), h = 0, k = 0$. The two pivot-tables in Figure 16 are defined by:

$$perm = (2, 3, 1), h = 1, k = 1, op = sum \quad \text{and} \quad perm = (1, 3, 2), h = 1, k = 1, op = sum$$

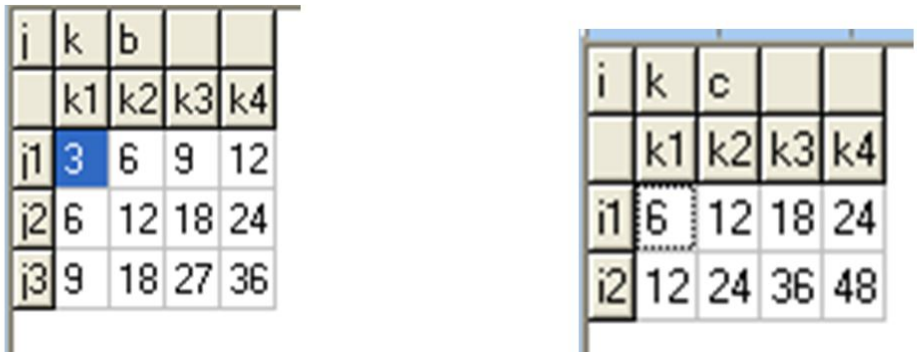


i	k	j	a	
		j1	j2	j3
i1	k1	1	2	3
i1	k2	2	4	6
i1	k3	3	6	9
i1	k4	4	8	12
i2	k1	2	4	6
i2	k2	4	8	12
i2	k3	6	12	18
i2	k4	8	16	24

k	j	i	a			
	j1	j1	j2	j2	j3	j3
	i1	i2	i1	i2	i1	i2
k1	1	2	2	4	3	6
k2	2	4	4	8	6	12
k3	3	6	6	12	9	18
k4	4	8	8	16	12	24

k	i	j	a			
	i1	i1	i1	i2	i2	i2
	j1	j2	j3	j1	j2	j3
k1	1	2	3	2	4	6
k2	2	4	6	4	8	12
k3	3	6	9	6	12	18
k4	4	8	12	8	16	24

Figure 15: Pivot Tables by Permuting Dimensions



j	k	b		
	k1	k2	k3	k4
j1	3	6	9	12
j2	6	12	18	24
j3	9	18	27	36

i	k	c		
	k1	k2	k3	k4
i1	6	12	18	24
i2	12	24	36	48

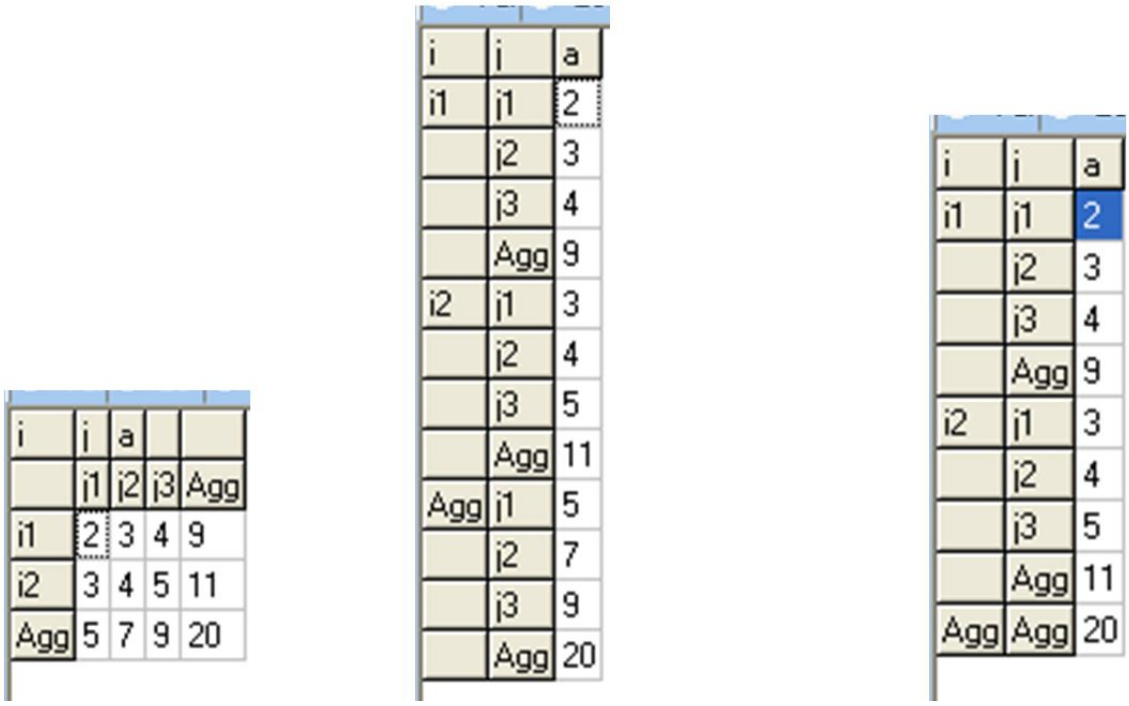
Figure 16: Two Aggregated Cubes

Furthermore, *ordering* can be taken into account. Each cell in the datacube is determined by the tuple of its members. Hence, the order in which the tuples are given does not matter. However, one could exploit this freedom to impose a specified order on each dimension's member list. The order than imposed the sequence in which the cells are listed in a particular pivot-table. Any permutation order on the members on each dimension gives a particular pivot-table. The ordering is easy to specify: Given an initial order of the members, one only need to attach a permutation vector to each dimension.

Another operation in showing a cube as a particular pivot-table is *selection*. One can dice a cube first and then display the diced cube as a pivot-table. Another way to view this is to attach a Boolean on each member, and set its value to TRUE if the particular member should be displayed in the pivot-table and FALSE else wise. Hence, each dimension needs a boolean vector of the size of the dimension to specify a selection.

The augmented cube can also be integrated easily into the pivot-table presentations. In the terminology we used earlier, we first *size* the cube with its aggregates and then show the sized cube as a pivot-table. Another way again is to integrate this information into the displaying operations: given a cube, we add (1) a permutation, (2) define a h and k and an (3) an aggregate-op.

Normally however, there is no need to compute all aggregated cubes in order to display them in a particular pivot-table. Let's explain this in the example of a 2-dimensional and then of a 3-dimensional cube.



i	j	a		
	j1	j2	j3	Agg
i1	2	3	4	9
i2	3	4	5	11
Agg	5	7	9	20

i	j	a
i1	j1	2
	j2	3
	j3	4
	Agg	9
i2	j1	3
	j2	4
	j3	5
	Agg	11
Agg	j1	5
	j2	7
	j3	9
	Agg	20

i	j	a
i1	j1	2
	j2	3
	j3	4
	Agg	9
i2	j1	3
	j2	4
	j3	5
	Agg	11
Agg	Agg	20

Figure 17: A Pivot Table with Aggregated Cubes

Figure 17 shows a cube of dimension 2 on the left side a pivot-table with $h = 1$. All three aggregated cubes (a*, *b, **) are visible and attached as last row and last column. All aggregated cubes are needed (ab, a*, *b, and **), see Figure 18. The middle pivot-table with $h = 0$ displays also all aggregates. However, one could argue that in the last four rows only the last is needed (the total of all totals), hence, the aggregated cube *b is not needed. A more "natural way" to display the pivot-table would be the picture on the right.

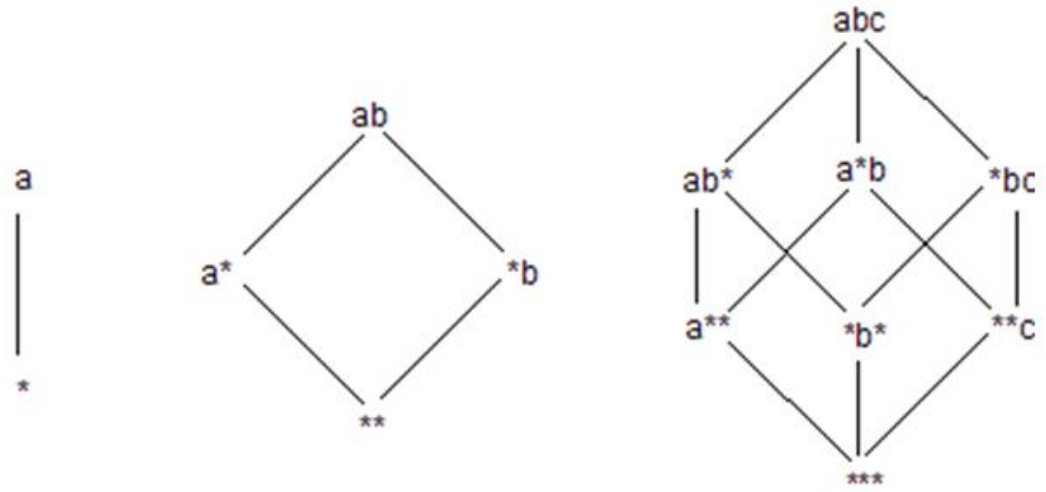


Figure 18: A Pivot Table with Aggregated Cubes

For a 3-dimensional cube with h=0 only the aggregates abc, ab*, a**, *** (see Figure 18) are needed, with $h = 1$, we need abc, ab*, a**, ***, and a*b, **c. The rule is the following: We need all aggregates along maximally two paths from abc to *** in the lattice. It is easy to find

them. If we have a 5-dimensional cube, for example, with the dimensions abcde (the top node of the lattice) then the dimensions are partitioned into the vertically and horizontally displayed dimensions for a particular pivot-table (say $h = 2$, then we have abc|de). The two aggregates needed – following the paths in the lattice – are : ab*|de and abc|d*. Hence, following the path in which the stars (*) are filled from right to left beginning with the very last entry and beginning with the entry left to |. Following the path down to ***** this way, generates to paths and the corresponding aggregates to calculate and integrate into the pivot-table are given by collecting them in the two paths. Now we also see, why in the cases of $h = 0$ (all vertically) and $h = n - k$ (all horizontally displayed) a single path is needed only. It is easy to see, why this works in general.

Formatting: A pivot-table is – first of all – a 2-dimensional representation of a datacube. The parts of the tables can be thought to be "printed" in rows and columns, in vertically/horizontally arranged cells as shown in Figure 19 where the parts are just displayed in the grid without formatting.

Product						
Tiere	Periods	TimeLag	AMOUNT			
		7-9	9-11	11-14	14-18	SUM
Alpha	Per_1	0	3	86	20	27
	Per_2	67	31	16	37	42
	Per_3	8	47	7	84	5
	SUM	29	91	36	77	32
Betas	Per_2	69	84	71	30	16
	Per_3	32	46	24	82	27
	SUM	48	14	87	28	77
SUM	SUM	97	49	88	82	2

Figure 19: Unformatted Pivot Table

However, not all cells in the grid have the same meaning. While some "cells" in the pivot-table display the name of the dimensions, others display the data. Basically, a pivot-table can be partitioned into 4 sections: (1) a header, where the dimensions are displayed, (2) the member names of the dimensions to identify a row and a column (3) the aggregates which are "SUM" rows/columns, and (4) the data part. The formatting of these sections is independent from the layout and we are free to enforce visually by colors and other attributes the different sections. An example is shown in Figure 20. Different formatting could be chosen.

Another kind of formatting is data formatting: (1) the data can be formatted along a mask like `##.###` (with three decimals, if they are numbers), (2) certain data can be shown in a different color (for example if they are negative), (3) The data can be shown as percent of a total, or as difference from a given value, etc.

The spreadsheet software Excel offers the functionality of pivot-tables. But it is a somewhat neglected tool and have serious disadvantages: : (1) The table is a one-way construct, one cannot change any data, (2) certain formats get lost if the table is manipulated by pivoting, (3) it takes quite a time to understand, what you can do, many operations can be done by mans different ways

Product						
Tiere	Periods	TimeLag	AMOUNT			
		7-9	9-11	11-14	14-18	SUM
Alpha	Per_1	0	3	86	20	27
	Per_2	67	31	16	37	42
	Per_3	8	47	7	84	5
	SUM	29	91	36	77	32
Betas	Per_2	69	84	71	30	16
	Per_3	32	46	24	82	27
	SUM	48	14	87	28	77
SUM	SUM	97	49	88	82	2

Figure 20: Formatted Pivot Table

In the LPL modeling system pivot table manipulation are fully integrated and easy to handle (see user manual of LPL). The LPL modeling system is designed to define and manipulate datacubes. The dimensions must be modeled as SETs and a datacube then is a multi-indexed entity in LPL. For example, to define a 3-dimensional cube one needs the declaration of four entities: 3 SETs, representing the dimensions and a `parameter` (or a `variable`, or whatever is indexed).

```
set i:=[i1 i2]; j:=[j1 j2 j3]; k:=[k1 k2 k3 k4];
parameter a{i,j,k} := i*j*k;
```

The different operations on datacubes can be implemented in various ways depending often from the context.
Slicing:

```
parameter b{j,k} := a['i1',j,k];
```

Dicing:

```
parameter c{i,j,k|a>=10} := a[i,j,k];
```

Sizing:

```
set h:=[1 i2 i3];
parameter d{h,j,k|a>=10} := if(h in i, a[h,j,k], sum{i} a[h,j,k];
```

Rising:

```
set i:=[1..3]; j:=[a b c]; k:=[1 2 3 a b c]; m:=[1..2];
parameter a{i}:=i; b{j}:=10*j;
  c{k}:=if(k<=#i,a[k],b[k-#i]);
  d{m,i}:=if(m=1,a[i],b[i]);
```

The generation of a pivot-table is a function of an extended Write statement. The input information that an algorithm must have to generate a particular pivot-table is:

1. a datacube
2. a permutation of the dimensions , h , k, aggregate operator
3. a order for the members of each dimensions
4. a Boolean for each member indicating of selecting it or not
5. Formatting: (1) of the cell: mask, alignment, font, border, pattern, (2) of the value: as percent, as difference etc., (3) depend on an expression (p.e., negative number with another color).

6 Spreadsheets

Spreadsheet modeling represents one of the most successful and widespread applications of present desktop computers. Since their introduction in the late 1970s, spreadsheet programs transformed the way of end-user computing. It created a new paradigm that offers a unique combination of ease of use and unprecedented modeling and calculation power, accessible to every user. As a result, spreadsheet programs became the most widely used decision support tool in modern business. Today's spreadsheet programs are very powerful, versatile, and user friendly. Yet in spite of this technological progress, the basic ideas for building a spreadsheet model remained the same in the last 25 years. Let us briefly explain what these basic ideas are and where they came from. "Spread sheets" have been used by accountants for hundreds of years [1], [7]. The history of accounting and bookkeeping is intrinsically linked to the history of calculation. It is, therefore, not surprising that the first published book on *double-entry*-accounting (Luca Pacioli's famous Summa de Arithmetica, geometria, proportioni et proportionalita, at 1494) appeared in a book of mathematics. In the realm of accounting, a "spread sheet" was and is a large sheet of paper with columns and rows that organizes data about transactions for a business person to examine. It spreads all of the costs, income, taxes, and other related data on a single sheet of paper for a manager to examine when making a decision. However, according to [7], until the 19-th century accounting was pure "arithmetics". Only at that time "algebraic" considerations entered the picture in accounting in the form of simple equations such as "Assets = Liabilities + Owner's Equity", compound interest rate calculations or present value considerations. The present value approach in accounting, for instance, played an important role in the German Railway Statutes of 1863 and subsequent legislations.

Spreadsheets are great for small models, although they are used frequently today for complex models with dubious success. The number of papers reporting and analyzing "errors" in spreadsheets is large. As soon as the model becomes complex the disadvantages of using traditional spreadsheets become evident. Since all calculation is cell-oriented, formulas must be copied again and again, revising and rearranging the layout turn out to be a nightmare, adding a new "item" having the same pattern as the already entered once is error-prune. The logic and structure behind the model get lost or invisible. It became evident already at the beginning of the 1990's that tools are needed to separate the different concepts: structure and logic, data and presentation. "In many ways, the present state of spreadsheet modeling is reminiscent of the state of data management prior to the 'database era.' Before data definition was elevated to the DBMS level, file structures were a fixed part of a program's code. In a similar vein, the logic

and documentation of spreadsheet models are often 'buried in the formulae,' and are largely inaccessible to people other than the spreadsheets' creators. In both cases, the implications are similar: redundant and inconsistent file and model structures, respectively. To complete the analogy, these problems arise because spreadsheet programs lack a high-leuel means to support the design and maintenance of spreadsheet models." [2].

In 1991, Lotus – then the leader of spreadsheet technology with its Lotus 1-2-3 – shipped a new product: Improv. Its inventor was Pito Salas, who can be called the father of pivot-table. The concepts of Lotus Improv revolved around the concept of separating the three parts: *data*, *views*, and *formulas*. The data are described in terms of categories such as "Territory", "Month", or "Product" and items of those categories such as "Europe", "January" or "Leather Gloves". The data can easily be reorganized. Categories are listed on tiles that you simply drag and drop to rearrange your data. Just pick up a category tile and move it to another axis, or rearrange it with respect to the tiles currently on its axis. Because data is associated with categories and items, rearranging the data has no effect on the validity of formulas. Multi-dimensional data – not just 2-dimensional once – can easily be set up. Formulas are written and stored in a separate formula area, which can be viewed simultaneously with the spreadsheet itself. Formulas are written in plain English, making it easier to read and write them. Improv was shipped in 1991 with the new NeXT computer, it was very popular and was one of the killer application on that computer. Unfortunately, Lotus did not port Improv to Windows until 1993. The main reason was that they did not want it to compete directly with their cash-cow: 1-2-3. In the meantime, however, the users already turned away from Lotus 1-2-3 to buy the product Excel. One reason between others was that Lotus put all its effort to the newest version of 1-2-3, which was delayed by more than one year and they did only half-hearted develop Improv. Lotus took Improv from the market. As the inventor of Improv sees it in retro-perspective, is "...that the key strategy mistake was to try to market Improv to the existing spreadsheet market. Instead, if the product were marketed to a segment where the more structured model was a 'feature' not a 'bug' would have given Lotus the time to learn and improve and refine the model to a point where it would have satisfied the larger market as well." [Pito Salas at November 28, 2004]. Improv was more a product in the realm of OLAP, a market that just had emerged at that time. It seems "that the OLAP vendors of the time, the likes of Comshare, Holos and IRI were worried about Improv as it had much the same functionality as product such as Holos and Express but with the backing of Lotus, who at the time were number one in the spreadsheet market with 1-2-3. In the end though, Lotus marketed Improv more as 'spreadsheets done right' (referring to the separation of data and formulas, and the more rigid structure of an Improv model) rather than its OLAP capabilities, which unfortunately had the effect of confusing those customers who now had to choose between 1-2-3 and Improv, and chose Microsoft Excel instead." [Rittman Mark, March 3, 2006]. So, it seems that Improv has failed due to its marketing debacle and less due to its "complexity". It is undeniable that creating a model with Improv requires some preparation and a structured and conceptual approach. However, it is also true that this greatly outweights the chaotic and "any-thing-goes" of today's spreadsheet building by producing a cleaner and more error-free model.

In any case, the story does not end here. Various attempts have been made to reanimate the ideas of Improv. FlexSheet, for instance, is a free software that implements a subset of Improv. In 2004, a company called Quantrix, has marketed its flagship software Quantrix Modeler, a real superset of Improv and more. Interestingly enough, the inventors of Quantrix Modeler, Peter Murray and Chris Houle, do not see their product as a direct competitor of traditional spreadsheet programs. Quantrix Modeler has even an export function to Excel! However, they analyzed very carefully the deficiencies of the traditional spreadsheet technology [6]:

1. Spreadsheets are two-dimensional: Data that are defined in more than two dimensions are difficult to build. Either one need to spread the data over several work sheets or an arbitrary predefined layout must be specified. Reorganizing the data afterwards is difficult and involving. Excel includes the concept of pivot-table, but they are define only as read-only.

2. Formulas are written with arbitrary coordinates: This make formula cryptic to read and reorganizing the data needs to rewrite many formulas. They are difficult to maintain consistently.

3. Logic is tied to the initial layout of the data: Because the formulas refer directly to cell positions, the logic of the model is inextricably tied to the presentation of the data within the spreadsheet's grid of cells. It is therefore a time-consuming task to rearrange the presentation of the data to highlight or juxtapose particular numbers in the model. Similarly, when trying to present the data through graphs and charts, the user is constrained to work within the program's "wizards", which force the data to be displayed in the way it is depicted within the spreadsheet grid.

4. Lack of dynamism: Modeling is an iterative process. This process must be structured in order to come to a meaningful, consistent and traceable result. Furthermore, the supposed benefits of spreadsheets is the ability to do "what if" analysis, to work through various scenarios. Changing the content of some cells in order to calculate a variant destroys the "original" model - but these changes are and must be an inevitable part of the spreadsheet modeling process. The task to verify the changes is time-consuming and error-prune. There is no true variant analysis, several variants cannot be inspected at the same time without large reorganization of the work sheets. Users must remember all the formula and cell dependencies and make multiple insertions and deletions to ensure consistency. However, one must remember that the typical spreadsheet developer is a business person not a professional software engineer who is specifically trained in handling such programming details.

5. Replicated logic: In spreadsheet application we often have "similar patterns", since each cell contains its own formula, this means that formulas must be copied and copied again over the whole spreadsheet with their cryptic hardcoded references to other cells. One never can be sure of whether the formulas are copied correctly resulting in momentous and far-reaching errors that are difficult to trace.

6. Row and Column limitations: The most widely used spreadsheet, limits the users to 65,536 rows and 256 columns. When initially designed, this was not considered a problem. However, as models have increased in complexity, this constraint is encountered more frequently.

7. File size: Due to the design of the traditional spreadsheet application that stores a formula in each calculated cell, models quickly become enormous files which are difficult to transfer with email or over a network.

The results are flawed models. This is alarming especially in financial context where a lot of money may be involved.[1] It also results in lack of auditability, in dependencies from its author,

[1] "A slip of the hand in a computer spreadsheet for bidding on electricity transmission contracts in New York will cost TransAlta Corp. $24-million (U.S.), wiping out 10 per cent of the company's profit this year." ... "A

in lack of portability and extensibility. One of the main functions of modeling is insight. This is largely absent. Since the logic and structure is mixed with the data and the formatted layout, the structure of the model cannot be separated and inspected on its own. Precisely what is the power of mathematics – insight – is made hidden and intransparent in the spreadsheet, because basically spreadsheets are "arithmetic" not "algebraic". So many scenarios are not calculated because it is too time-consuming to do it! However, the reality is that business professionals must create increasingly complex models for their decision making processes and their software is based on 25 year old innovations which mapped the paper-ledger into software! There exists an increasing need for better tools. Still!

7 Conclusion

Data Viewing is an important subject whenever mass of data is involved in modeling. For this we use *reporting*, *data browsing tools* and *data editing tools*. When uniformed data are stored along several dimensions, then we may pack them into datacubes. These data are best viewed and edited through pivot-tables, the 2-dimensional representation of any datacubes.

This paper tries to give a unified theory on datacube and pivoting. It was shown that all data operations in OLAP can by reduced to slicing, dicing, sizing, and rising operations in a multidimensional datacube. The goal is to reduce all kind of proposed operations in OLAP to a few operations in manipulating datacubes. This gives also a new view in implementing OLAP tools. It is, furthermore, important to note that all aspects of data viewing of unified mass of data stored as datacubes can be accessed through pivot-tables. Pivot-tables are easy to understand (if implemented correctly) and easy to manipulate – at least from the point of view of the user. This paper shows that – given a datacube – a few operations and options determine a particular pivot-table. If the user understands these few operations, it is easy to use them. Unfortunately, Excel pivot-tables do not have these properties.

References

[1] Marinoni M. Cilloni A. Spreadsheet, Chessboard and Matrix Accounting: The origin and development of advanced accounting instruments. *The 10th World Congress of accounting Educators*, 2006, Istambul.

[2] Lucas H.C.JR. Isakowitz T., Schocken S.

[3] Malliaris M. Jukic N., Jukic B. Processing (OLAP) for Decision Support. *Handbook on Decision Support Systems 1, eds. Burstein F., Holsapple C.W., Springer, pp. 259ff*, 2008.

[4] MatMod. Homepage for Learning Mathematical Modeling : https://matmod.ch.

[5] Arnott D. Meredith R., O'Donnell P. Databases and Data Warehouses for Decision Support. *Handbook on Decision Support Systems 1, eds. Burstein F., Holsapple C.W., Springer, pp. 207ff*, 2008.

[6] Houle C. Murray P. You Can't Always Get from A to B by Way of X and Y. *a Quantrix White Paper at www.quantrix.com*, 2006.

spreadsheet created in 3-months, full-time work by a highly-paid professional has a cost-accounting value in excess of $25,000." [**?**].

[7] Mattessich R. A Concise History of Analytical Accounting: Examining the Use of Mathematical Notions in our Discipline. *De Computis Spanish Journal of Accounting History, Vol. 2, pp 123ff*, 2005.

[8] Winston W.L. Seref M.H., Ahuja R.K. Developing Spreadsheet-Based Decision Support Systems (Using Excel and VBA for Excel). 2007, Belmont, MA: Dynamic Ideas.

[9] Hürlimann T. Index Notation in Mathematics and Modeling Language LPL: Theory and Exercises. https://matmod.ch/lpl/doc/indexing.pdf.