RC25412 (WAT1309-091) September 30, 2013
Computer Science

IBM Research Report

A Functional Data Model for Analytics

Doug Kimelman
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 208
Yorktown Heights, NY 10598
USA

Manny Perez
IBM

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

A Functional Data Model for Analytics

Doug Kimelman
Manny Perez
IBM

ABSTRACT

This paper presents a functional data model – a model that is fundamental to supporting management analytics applications. This functional data model, henceforth referred to simply as "the functional model", is different from, but complementary to, the relational model. The functional model is also distinct from other similarly named concepts, including the DAPLEX functional database model, and functional language databases.

The functional model comprises multidimensional hierarchical consolidation, as is commonly found in OLAP technologies, relational-based and otherwise. But it goes beyond OLAP by requiring a spreadsheet-like cell orientation, and definition of cells calculated as functions of other cells. Because a functional definition of calculation according to formulas and consolidation is incorporated into the data model, databases implementing the model will inherently always return calculated cell values that are up to date and consistent with respect to the latest input cell values in interactive database update scenarios. These concepts are lacking in the relational model but are essential for support of flexible and interactive business performance management analytics.

The validity and effectiveness of the functional model is evident in the long-standing success of commercial product technology that embodies the model to deliver practical analytics solutions across a broad range of management analytics domains.

We present a formal definition of the functional model, a brief discussion of database technology that implements the model efficiently, and a comparison that shows how the relational model is inadequate in terms of expressiveness for scenarios fundamental to management analytics.

Categories and Subject Descriptors

H.2.1 [Database Management]: Logical Design

Keywords

Database, OLAP, Functional, Relational, Analytics, Spreadsheets, Multidimensional, Planning, What-if

1. Introduction

Analytics, especially forward looking or prospective analytics requires interactive modeling, "what if", and experimentation of the kind that most business analysts do with spreadsheets. This interaction with the data is enabled by the spreadsheet's cell orientation and its ability to let users define cells calculated as a function of other cells.

The relational database model has no such concepts and is thus very limited in the business performance modeling and interactivity it can support. Accordingly, relational-based analytics is almost exclusively restricted to historical data, which is static. This misses most of the strategic benefits of analytics, which come from interactively constructing views of the future.

This paper introduces the functional data model. The functional model is based on multidimensional arrays, or "cubes", of cells that, as in a spreadsheet, can be either externally input, or calculated in terms of other cells. Such cubes are constructed using dimensions which correspond to hierarchically organized sets of real entities such as products, geographies, time, etc. A cube can be seen as a function over the cartesian product of the dimensions, mapping coordinate n-tuples to cell values; thus the name "functional". The model retains the flexibility and potential for interactivity of spreadsheets, as well as the multidimensional hierarchical consolidations of relational-based OLAP tools. At the same time, the functional model overcomes the limitations of both the relational database model and classical spreadsheets.

Products that implement the principles of the functional model to varying degrees have been in existence for some time, including products such as Oracle® Hyperion® Essbase®, IBM® Cognos® TM1®, Alea, Microsoft® Analysis Services®, etc. [16, 19, 20, 21, 22, 23]. But other than publications such as the OLAP Report [17,18], and a paper by E. F. Codd [10], very little has been written on its mathematical foundation. Other than in *Spreadsheets in RDBMS for OLAP* [26], very little has been published on the technology used to implement functional databases. Much of it is considered proprietary and only some early patents [24, 25] give a glimpse of the technical challenges that had to be overcome for proper implementation.

This paper is intended to familiarize others with the concepts and significance of the functional model, to encourage the broader application of the technology, and to set its mathematical foundation as a base for future work to improve the functionality, scalability and performance of databases based on the model. Its ultimate goal is to improve the efficiency and effectiveness with which enterprises and other organizations are run, to the benefit us all.

This paper first covers business performance management analytics and how the functional model facilitates it, specifically:

- The nature of analytics, particularly management analytics and its importance to the more effective functioning of an enterprise.
- How the functional model fits in the context of analytics.
- The relationship between spreadsheets and the functional model.
- The key ideas behind the functional model and how it benefits analytics.
- How the functional model compares to the relational model.

We then present a formal definition of the functional model, along with a discussion of related work and future research directions.

2. Context

Enterprises, like all living organisms, must be aware of and adapt to their environment in order to survive and thrive. Their behavior

starts with a planning process that assesses the current business climate and attempts to predict its future evolution. This involves an understanding of its markets and customers, the competition, government regulations, technology, the economy, etc.

Once the plan – the enterprise's intended course of action – is established, the enterprise proceeds to execute the plan. As it does so, it periodically compares and analyzes its actual performance vis-à-vis the plan, and adjusts its behavior or the plan accordingly.

The process is effectively a feedback control loop – plan the work, work the plan, measure the results. Information Technology (IT) plays now a central role in making management control loops more efficient and effective. One could say this is a major benefit of IT.

The importance of automation to the success of an enterprise is now generally considered vital: The management control loop is its nervous system, and clearly its speed, accuracy and effectiveness can make the difference between success and failure. Improving the management control loop makes for a smarter and more successful enterprise.

2.1 Prospective and Retrospective Analytics

The management loop can be thought of as having two layers as shown in Figure 1.

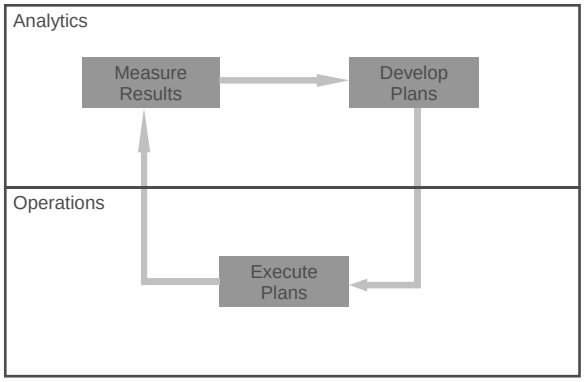

Figure 1. Management Cotrol Loop

The three components of the control loop have different time perspectives. Operations looks at the here and now, whereas analytics looks at the past and the future. The information needs of each component are different. Operations is concerned with recording transactions and keeping track of the current state of the business – inventory, work in progress etc. Analytics has two parts, prospective: develop plans, and retrospective: measure results.

In retrospective analytics, transactions resulting from operations are boiled down and accumulated into arrays of cells. These cells are identified by as many dimensions as are relevant to the business: time, product, customer, account, region, etc. The cells are typically arrayed in cubes that form the basis for retrospective analyses such as comparing actual performance to plan.

Prospective analytics develops similar cubes of data but for future time periods. The development of prospective data is typically the result of human input or mathematical models that are driven and controlled through user interaction.

2.2 Historical Perspective

It is instructive to consider the emergence of automated support for analytics, and the failings that constituted barriers to widespread adoption. Table 1 summarizes the history of the tools and technologies used in the three elements of the management control loop.

Table 1. History Control Loop Elements

Period	**Operations**	**Retrospective Analytics**	**Prospective Analytics**
Pre Automation	Ledger books	Profit & Loss, Balance Sheet reports	Paper spreadsheets
Early Automation	80-column card Sequential files Random Access DBMS	Reporting languages. 4GL	APL Math Programming Statistical Analytics
Late Automation	RDBMS SQL	Star Schema BI, DW OLAP, MDX	Electronic Spreadsheet Functional Database

Back in the days of quill pen and paper, operational transactions were recorded in a ledger books, and retrospective analytics amounted to financial reports (balance sheet, P&L, etc.) that were derived from the ledger. Prospective analytics at best was done on paper spreadsheets.

In the early days of automation, recording of operational transactions was the main priority. The first step was to put those records on 80 column punch cards. As electronics progressed, the records were moved, first to magnetic tape, then to disk. Software technology progressed as well and gave rise to database management systems that centralized the access and control of the data.

Databases made it then possible to develop languages that made it easy to produce reports for retrospective analytics. At about the same time, languages and systems were developed to handle multidimensional data and to automate mathematical techniques for forecasting and optimization as part of prospective analytics. Unfortunately, this technology required a high level of expertise and was not comprehensible to most end users. As a result, its user acceptance was limited, and consequently so were the benefits derived from it.

As information management software matured, record-based databases were formalized in the relational model and SQL [14], which became pervasive in support of operations. Subsequently, in support of analytics, a multidimensional construct based on the relational model – the star schema – was invented [15]. It provided much better functionality and flexibility for deriving value from historical data. Data warehousing and business intelligence were mostly based on this notion, and the OLAP query language MDX [16] further facilitated the presentation of multidimensional data.

However, the most important contribution to improving the management loop was the introduction of the electronic spreadsheet in support of both retrospective and prospective

analytics. For the first time end users had a tool that they could understand and control, and use it to model their business as they understood it. They could interact, experiment, adapt to changing situations, and derive insights and value very quickly. As a result, spreadsheets were adopted broadly, came to be regarded widely as a "killer app", and ultimately became pervasive. To this day, spreadsheets remain an indispensible tool of any enterprise.

The functional data model, as was discussed above, affords end users key capabilities and benefits similar to those of the spreadsheet, but goes beyond those by incorporating multiple dimensions and consolidation, overcoming the scalability limitations of the spreadsheet. Functional databases – those based on the functional model – additionally offer benefits typically associated with conventional database technology: data independence, concurrent multiuser access, integrity, scalability, security, audit trail, backup/recovery, and data integration. Functional database technology is being adopted in support of analytics by ever-growing numbers of enterprises.

3. Spreadsheets for Analytics

Business analysts get most of the information they use to make decisions about the future of their enterprises from what are in today's parlance business intelligence (BI) systems. Such systems consist primarily of report generation software acting on a data warehouse that stores historical transaction data.

Before analysts will base decisions and take action on the information provided by the BI systems, they will go through a process of interactive exploration and reflection. They will have what amounts to an intimate conversation with their data. It is through this interaction process that analysts arrive at insights and conclusions. They also develop confidence in the conclusions to the point of basing decisions on them. The tool they use for this conversation is almost exclusively an electronic spreadsheet. Even if their spreadsheets are not fed automatically from the BI systems, users will go through the pain of feeding them manually by typing in the data from the reports.

In order to extract insights and gain confidence, users will calculate other derived values. For example they may calculate the percent growth of a group of products from the prior period and compare the result with that of other product groups by various geographies. Or they may look at the historic costs of the components of a product, project what the cost will be in future periods and the effect on the cost of the product. In building those models, they will use their experience and intuition, focusing on those areas where they can see opportunities.

Spreadsheets have a key set of characteristics that facilitate modeling and analysis:

- Data from multiple sources can be brought together in one worksheet.
- Cells can be defined by means of calculation formulas in terms of other cells. So facts from different sources can be logically interlinked to calculate derived values.
- Calculated cells are updated automatically whenever any of the input cells on which they depend changes. When users have a "what if" question, they simply change some data cells, and automatically all dependent cells are brought up to date
- Cells are organized in rectangular grids and juxtaposed so that significant differences can be spotted at a glance or through associated graphic displays.
- Spreadsheet grids normally also contain consolidation calculations along rows and or columns. This permits discovering trends in the aggregate that may not be evident at a detailed level.

3.1 Limitations of Spreadsheets

While spreadsheets are fundamental to extracting insights and decisions from data, they nonetheless suffer from certain limitations:

- Limited data identification. Cells are identified by row and columns, not the business concepts they represent.
- Limited dimensionality. Spreadsheets are two dimensional, and multiple pages provide the semblance of three dimensions, but business data often has more dimensions.
- Data is tied to a spreadsheet. If users want to perform another analysis on the same set of data, the data needs to be duplicated. Spreadsheet links can sometimes be used, but most often are not practical.

The combined effect of these limitations is that there is a limit on the complexity of spreadsheets that can be built and managed. As the needs of the business drive users to more complexity, they find themselves in a situation commonly known as *spreadsheet hell.*

4. The Functional Model for Analytics

While the functional model retains the key features of the spreadsheet, it also overcomes its main limitations, which were identified above. With the functional model, data is arranged in a grid of cells, but cells are identified by business concept instead of just row or column. Rather than worksheets, the objects of the functional model are dimensions and cubes. Rather than two or three dimensions: row, column and sheet, the functional model supports as many dimensions as are necessary. And the dimensions of the cube support hierarchical structures that define consolidations implicitly.

4.1 Functional Databases

Another benefit of the functional model, as compared to the spreadsheet, is that it can be implemented using database technology, and that affords the end user all of the benefits typically associated with such technology. As was mentioned above, these benefits include: data independence, concurrent multiuser access, integrity, scalability, security, audit trail, backup/recovery, and data integration.

Data independence is of particularly high value for analytics. Data need no longer reside in spreadsheets. Instead the functional data base acts as a central information resource that can be shared by multiple spreadsheets and multiple users. Updates submitted by multiple users are available to all users subject to security rules. Accordingly there is always a single shared version of the truth.

The functional data model forms the theoretical basis for functional databases, and functional databases implement the functional data model with a particular focus on efficient representation, calculation, and consolidation for large sparse multidimensional datasets having hierarchies in each dimension. With a functional database, an analytics model can consist of multiple cubes where appropriate – for instance, where there are multiple collections of data, each having different dimensionality. The cubes are interlinked by calculation rules that contain inter-cube references.

4.2 Dimensions, Cubes and Cells

A dimension is a finite set of elements, or members, that partition or abstract the business reality into concepts, or bins, that can be used for predictions and comparisons, e.g., time periods, products, areas or regions, line items, etc.

Cubes are built using any number of dimensions. A cube is a collection of cells, each of which is identified by a tuple of elements, one from each dimension of the cube. Each cell in a cube contains a value. A cube is effectively a function that assigns a value to each n-tuple of the cartesian product of the dimensions.

The value of a cell may be:

- Assigned externally (input) and stored in the cell, or
- The result of a calculation that uses other cells, in the same cube or other cubes, and a formula that specifies the calculation.

As well, some cells may have no value – their value is not calculated, and they have not been assigned an input value. Such cells can be treated as having the value zero for purposes of consolidation.

As with spreadsheets, users need not worry about executing recalculation. When the value of a cell is requested, the value that is returned is up to date with respect to the values of all of the cells that go into its calculation i.e. the cells on which it depends.

Dimensions typically contain consolidation hierarchies where some elements are defined as parents of other elements, and a parent is interpreted as the sum of its children. Cells that are identified by a consolidated element in one or more dimensions are automatically calculated by the functional model as sums of cells having child elements in those dimensions. Please see the formal definition below for more detail. When the value of a consolidated cell is requested, the value that is returned is always up to date with respect to the values of all of the cells that it consolidates.

4.3 An Example

The concepts of the functional model will be illustrated with an example shown in Figure 2. Its functional database consists of five cubes that are interconnected by formulas into a single analytics model. The dimensions of the cubes are built from tables in a data warehouse shown at the top of the diagram. The cubes and their dimensions (in parentheses) are as follows:

- P&L (Region, Account, Currency, Time)
- Sales(Region, Product, Time)
- Payroll(Region, Employee, Time)
- Overhead(Account, Time)
- Foreign Exchange(Currency, Time)

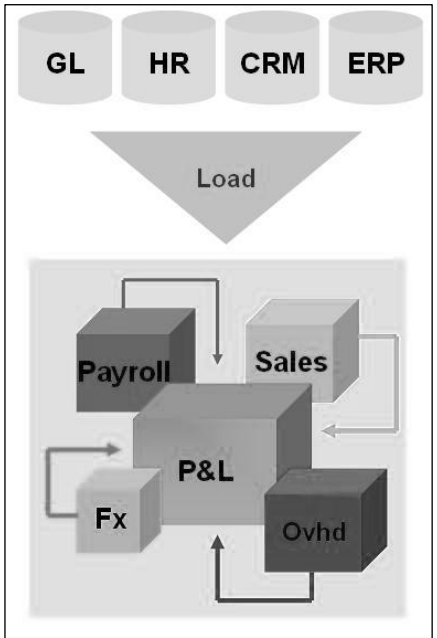

Figure 2. Example of a Functional Model

All cubes support the typical consolidation and pivoting operations of OLAP multidimensional databases. Furthermore, the cubes in the model are interconnected through formulas:

The P&L cube picks up the dollar costs from the payroll cube through a formula of the form:

> P&L(“Payroll”, “Dollars”) = Payroll (“All Employees”)

The dimensions that are omitted from the expression are assumed to range over all the leaf elements of those dimensions. Thus this expression is equivalent to:

> P&L(xRegion, “Payroll”, “Dollars”, xTime) = Payroll (xRegion, “All Employees”, xTime), for all leaves xRegion in Region and all leaves xTime in Time.

P&L also picks up sales revenue from the Sales cube through:

> P&L(“Sales”, “Dollars”) = Sales(“All Products”)

Overhead accounts are allocated by region on the basis of sales:

> P&L(“Region”, “Dollars”) = Ovhd() * Sales(“Region”) / Sales(“All Regions”)

Finally, other currencies are derived from the dollar exchange rate:

> P&L() = P&L(“Dollars”) * Fx()

Note: The expression syntax used is for illustration purposes and may not reflect syntax used in the formal model or in particular products that implement the functional model..

4.4 Use of the Model

The historical portion of the cubes is populated from the data warehouse. In this simplified example, the calculations just discussed may be done in the data warehouse for the historical portion of the cubes, but generally, the functional model supports the calculation of other functions, such as ratios and percentages that can deliver insight into historical data.

While the history is static, the future portion is dynamic and developed interactively by business analysts in various organizations and various backgrounds:

- Sales forecasts should be developed by experts from each region. They could use forecasting models and parameters that incorporate their knowledge and experience of that region, or they could simply enter them through a spreadsheet. Each region can use a different method with different assumptions.
- The payroll forecast could be developed by HR experts in each region.
- The overhead cube would be populated by people in headquarters finance, and so would the exchange rate forecasts.

The forecasts developed by regional experts are first reviewed and recycled within the region and then reviewed and recycled with headquarters.

The model can be expanded to include a Version dimension that varies based on, for example, various economic climate scenarios. As time progresses, each planning cycle can be stored in a different version, and those versions compared to actual and to one another.

At any time the data in all the cubes, subject to security constraints, is available to all interested parties. Users can bring slices of cubes dynamically into spreadsheets to do further analyses, but with a guarantee that the data is the same as what other users are seeing.

4.5 Functional Database Benefits for Interactive Analytics

Many of the benefits of databases that implement the functional model derive from both the multidimensional qualities of the model and the database qualities of the implementation. However, the most valuable benefit is uniquely the result of the model being cell-based and the implications inherent in that for support of interactive modeling via recalculation. Those unique benefits are highlighted and summarized in the following four areas:

4.5.1 Data integration

A functional database brings together data from multiple disparate sources and ties the disparate data sets into coherent consumable models. It also brings data scattered over multiple spreadsheets under control. This lets users see a summary picture that combines multiple components, e.g., to roll manpower planning into complete financial picture automatically. It gives them a single point of entry to develop global insights based on various sources. And by bringing spreadsheets under control delivers them from spreadsheet hell.

The relational model can be used to integrate multiple tables; however, to implement the calculation model that interlinks the cubes at the cell level, as in the example above, would require the development of a highly complex ETL application by a SQL expert. A simpler approach could implement the calculations by performing a series of single-cell SQL queries, but the performance of this approach is likely to be inadequate. Otherwise the application could transfer the data to cubes that are cell addressable. In the end, the ETL tool would effectively implement an ad hoc functional database, but one that would be read only. *Spreadsheets in RDBMS for OLAP* [26] describes in some detail the limitations of SQL in this context.

4.5.2 Interactive What If

A functional database, like spreadsheets, also lets users change input values while all dependent values are up to date. This facilitates what-if experimentation and creating and comparing multiple scenarios. Users can then see the scenarios side by side and choose the most appropriate.

When planning, users can converge on a most advantageous course of action by repeatedly recycling and interacting with results. Actionable insights come from this intimate interaction with data that users normally do with spreadsheets

4.5.3 Interactive Modeling

A functional database not only provides a common interactive data store. It also brings together models developed by analysts with knowledge of a particular area of the business that can be shared by all users.

To facilitate this, a functional database retains the spreadsheet's interactive cell-based modeling capability. This makes possible models that more closely reflect the complexities of business reality.

4.5.4 Collaborative Interaction

Perhaps a functional database's largest single contribution to analytics comes from promoting collaboration. It lets multiple individuals and organizations share single version of the truth, but a truth that is dynamic and constantly changing. Its automatic calculations quickly consolidate and reconcile inputs from multiple sources. This promotes interaction of various departments, facilitates recycle and makes it possible for differing viewpoints to converge and be reconciled. Also, since each portion of the model is developed by the people that are more experts in the area, it is able to leverage experience and insights that exist up and down the organization.

5. Formal Description of the Functional Model

5.1 Dimensions and Cubes

A functional model consists of a set of dimensions and a set of cubes.

A dimension D consists of a set of base or leaf elements **B**.D and a set of hierarchies **H**.D.

5.1.1 Cubes and Cells

A *cube* C is a function whose domain is the cartesian product of dimension leaf elements **B**.D1 × **B**.D2 × … × **B**.Dn and whose range is either the set of real numbers $\Re$, or the set of finite character strings *s*. If the range of C is $\Re$, the cube is called a numeric cube. For a numeric cube,

C: **B**.D1 × **B**.D2 × … × **B**.Dn → $\Re$

(Note: As in the relational model [4], for simplicity and flexibility the actual model should not require identifying the dimensions of a cube, or the elements of the domain of C as ordered n-tuples, rather it should identify the components of a tuple using the concept of *relationships* and *role tags* rather than a position index. This scheme will not be used or described further in this paper, instead we will use cartesian products and ordered n-tuples.)

The elements of the domain of C, i.e., the n-tuples of dimension leaf elements, are called leaf cells. The range element assigned to a leaf cell by the cube function is called the cell value. The value of a leaf cell may be either:

- An externally assigned (input) value effectively stored in the cell
- The result of a calculation formula that refers to other cells in the same cube or other cubes.
- Undefined or empty. Most actual cubes are mostly empty or sparse. Consolidations will normally skip such cells, effectively treating them as if their value was zero.

5.2 Consolidations

In addition to stored and calculated leaf cells, cubes also contain consolidations associated with the elements of hierarchies. Such consolidation calculations are kept automatically up to date whenever underlying values change.

5.2.1 Hierarchies

A hierarchy h of dimension D, h ∈ **H**.D, consists of a finite set **E**.h of hierarchy elements, and for each element e, a set of components **C**.e. The components of a hierarchy element can be

leaf elements, or other elements of the hierarchy, i.e., **C**.e ⊆ **E**.h ∪ **B**.D.

Components are also referred to as children. Note that unlike in biology, the value of parents is derived from its children.

The set of descendents of a hierarchy element e, **D**.e, is the union of **C**.e and the descendents of each of the elements of **C**.e. Hierarchies do not allow circular references, so no element can be its own descendent: ∀e∈ h (e∉**D**.e)

The leaf components of e, **B**.e are the descendents that are leaf elements of the dimension: **B**.e = **D**.e ∩ **B**.D.

For leaf elements, we define the set of components as a singleton containing the element itself , i.e.

∀b∈ **B**.D (**B**.b = **D**.b = { b })

5.2.2 Extended Set of Elements

The extended set of elements of a dimension **E**.D is the union of the leaf elements of D and the elements of all its hierarchies.

$$\mathbf{E}.D = \mathbf{B}.D \cup \bigcup_{h \in H.D} E.h$$

5.2.3 Consolidation Cube

Given a numeric-valued cube C,

C: **B**.D1 × **B**.D2 × … × **B**.Dn → ℜ

The consolidated cube C' is the extension of C to the Cartesian product of extended sets of elements of its dimensions:

C': **E**.D1 × **E**.D2 × … × **E**.Dn → ℜ

Where

$$C'(e1, e2, \ldots, en) = \sum_{b1 \in B.e1} \sum_{b2 \in B.e2} \ldots \sum_{bn \in B.en} C(b1, b2, \ldots, bn)$$

5.3 Multiple Hierarchies

Dimensions often originate from relational tables in operational systems or star schemata in data warehouses. Such dimension tables may be denormalized, with one row per leaf element, or may be stored as multiple tables corresponding to different levels in hierarchies as in snowflake schemata. For our purposes we will assume a single denormalized dimension table.

For example, respondents to a survey may have the following attributes: Age, Sex, Occupation, Income, Street Address, City, State, and Zip Code.

Note that respondent is a single dimension and that its attributes give rise to multiple hierarchies.

For purposes of analysis it may be desirable to group the respondents by: Age group, Sex, Occupation, Income Bracket, and Location. This would give rise to five different hierarchies.

Implementations of the functional model should provide the tools to automatically generate and update dimensions and hierarchies based on relational dimension tables.

In this example, analytical needs may require displaying different hierarchies of the same dimension along different axes, effectively treating them as independent dimensions. The functional model supports this requirement as described in the next section.

5.3.1 Pivoting Hierarchies

The functional model supports the presentation of slices of cubes. Various views of the cube can be shown by arranging some of the dimensions as rows, some as columns and some as titles. Single elements are selected from the title dimensions and subsets of elements for the axis (row and column) dimensions.

In situations where a dimension has multiple hierarchies as in the survey respondent example above, it is often desirable to present slices of a cube where the rows and columns are not from different dimensions, but from different hierarchies of the same dimension.

For example, if the respondent dimension has two hierarchies Age, with elements {young, old, all ages}, and Sex with elements {male, female, all sexes}, it may be of interest to see responses tabulated with sex as columns and age as rows. The cell under the column "female" in row "young" would consolidate all the respondents who are both young and female, which is the same as the intersection of the leaf children of element "young" and the leaf children of element "female".

To that end, cubes in the functional model support consolidations not only for the (extended) set of elements of its dimensions, but also for sets of such elements. Formally:

Given a consolidation cube C'

C': **E**.D1 × **E**.D2 × … × **E**.Dn → ℜ

The multi-hierarchy consolidated cube C'' is defined as follows:

C'': ℙ(E:D1) × ℙ(E:D2) × … × ℙ(E:Dn)

Where ℙ(S) is the power set of S, or the set of all subsets of S.

The value of a cell identified by subsets S1, S2, …, Sn, is given by:

$$C''(S1, S2, \ldots, Sn) = \sum_{b1 \in I1} \sum_{b2 \in I2} \ldots \sum_{bn \in In} C(b1, b2, \ldots, bn)$$

And where I1, I2, …, In is the intersection of the leaf children of the members of each set, i.e.:

$$Ik = \bigcap_{e \in Sk} B.e$$

In the example above, the value of the cell for young females would be given by:

C''(…, {young, female}, …)

where {young, female} appears in the position of dimension "respondent".

5.4 Updating Cubes

Information is input into a functional model by changing the value of leaf cells. The values of all calculated leaf cells and all consolidations will immediately reflect the change.

5.5 Leaf Cell Calculations

As mentioned before, in addition to consolidations, the functional model supports arbitrary calculations of cells in terms of other cells. As in the spreadsheet, where users typically copy formulas across rows, columns or ranges, the language that expresses calculation formulas does so for ranges of cells. Without going into the detailed syntax, a calculation is expressed by means of a *rule* that has the form:

<Range of Cells> = <Expression>;

Where <Range of Cells> is a subset of the leaf cells of a cube, and <Expression> is written in terms of references to cells in any cube, both leaf and consolidated, i.e., references to the *extended* version of any cube.

Such calculated leaf cells are treated as input cells for purposes of consolidation.

When <Range of Cells> for multiple rules overlap, the rule that appears first will override the others.

5.5.1 Element Calculation Rules

Calculations can also be defined for leaf elements of a dimension using a rule of the form:

<Element> = <Expression>

Where <Expression> is written using references to other leaf elements of the dimension.

Such a rule applies in all cubes that use the dimension to all cells identified by <Element>.

Element rules are overridden by cell rules that may be defined for such cells.

When multiple element rules apply to the same cell, the order is resolved based on the order in which the dimensions appear in the cube. Such default order can be overridden by writing cell rules that do so.

5.6 Nonlinear Summary Calculations

Rules can be written also for summary cells, in which case the rule overrides the normal consolidation that would otherwise be calculated. This can be used effectively to calculate ratios and also array functions such as Maximum, Minimum, Average, Count, etc. These functions are order dependent (unlike Sum which is the basis of consolidations) and cannot be defined globally the way consolidations are.

In the two dimensional cube below, rules are defined for calculating minima and maxima along each dimension.

	Jan	Feb	Mar	1Q	*Max*	*Min*
North	15	28	79	122	*79*	*15*
South	55	75	26	156	*75*	*26*
East	6	55	86	147	*86*	*6*
West	33	40	54	127	*54*	*33*
Total Region	109	198	245	552	*245*	*109*
Max Region	*55*	*75*	*86*	*156*	86	*33*
Min Region	*6*	*28*	*26*	*122*	*54*	6

Note that the value for Total Regions, Max Month, 245, is calculated by calculating the totals first and then taking their maxima, i.e., the rule is calculated "last". Note also that the cell for Min Region, Max Month, 54 is calculated by taking the minimum of the maxima. This is because we chose the Region rules to override the Month rules. If we did the opposite, the result would be different. Mixing such nonlinear calculations can be misleading and confusing so the recommended practice is to leave such "compound nonlinear" cells as undefined:

	Jan	Feb	Mar	1Q	*Max*	*Min*
North	15	28	79	122	*79*	*15*
South	55	75	26	156	*75*	*26*
East	6	55	86	147	*86*	*6*
West	33	40	54	127	*54*	*33*
Total Region	109	198	245	552	*245*	*109*
Max Region	*55*	*75*	*86*	*156*		
Min Region	*6*	*28*	*26*	*122*		

5.7 Relational Aspects of the Functional Model

While taking advantage of the benefits of the functional model, users should not have to give up the set manipulation power of the relational model. For example, when issuing queries to a functional server, the server should not just do the cell calculations, but should also support expressions that calculate sets of elements used in views, as in the OLAP multidimensional expressions (MDX) language.

The set calculations in the functional model should be as dynamic as those used to perform cube calculations. So, for example, if a set expression depends on values stored in a cube, when users update those values, the set should be automatically recalculated to reflect the change.

5.7.1 Dimension Attributes and Hierarchies

In the example of respondents to a survey that was presented above to illustrate multiple hierarchies, those hierarchies are derived from a dimension table with columns:

- Age
- Sex
- Occupation
- Income
- Street Address
- City
- State
- Zip Code

In the ROLAP star schema, such columns are called attributes of the dimension.

We could construct the various hierarchies in the example externally, based on the table, and then use them to update the hierarchies in the functional model.

5.7.2 Dynamic Hierarchies

Alternatively, this information can be kept in a set of functions, i.e., cubes that are directly updatable by the user. The functions are then used to define the components of hierarchy elements.

We define an Age Group dimension with elements:

AgeGroup. **B** = {"Child", "Youth", "Adult",
"Middle Age", "Senior"}

and two functions LowAge(AgeGroup) and HighAge(AgeGroup) to define the ages that fall in each group.

We define a hierarchy for the respondent dimension:

AgeGroup. **H** = AgeGroup. **B** ∪ {"All"}

Using the function Age(Respondent) we define the components of each age group by the expression:

∀ g ∈ AgeGroup. **B**
C .g = {y ∈ **B**.Respondents |
Age(y) >= Low Age(g) & Age(y) <= High Age(g)}

and the components of "All" by:

C.All = AgeGroup

This way the age of respondents, as well as the range of ages for the age groups could be adjusted dynamically.

5.7.3 Measures in the Functional Model

When a cube in a ROLAP star schema is translated to the functional model, multiple functions are created, one for each column in the fact cube, which is called the measures dimension in star schema parlance.

Rather than generate multiple cubes, an alternative formulation of the functional model could group all functions with the same dimensionality in one cube, by introducing an additional measures dimension. Each element of the measures dimension defines a range of values. Cubes thus formed become an agglomeration of functions rather than pure functions. Arguably, this formulation has simplicity benefits for users.

5.7.4 Storing Relational Tables in Cubes

As mentioned at the beginning of this paper, planning sometimes involves predicting individual future events or transactions, for example, capital projects, hiring for certain positions, or major predicted orders. Also, last minute adjustments, such as journal entries, may have to be made as part of the planning process. The occurrence of such relational inputs increases as the planning applications concern themselves with finer levels of detail. The functional model should have provisions to handle such transactions without having to resort to a relational server.

This can be accomplished by introducing an Index dimension whose members correspond to the natural numbers, 1, 2, …, and which is open-ended. This dimension in addition would have a single hierarchy with a Total element in it.

Let us suppose we are given a table with no candidate key, where some of the columns D1, D2, …, DN that correspond to dimensions in an existing functional model, and the rest of the columns V1, V2, …, VM, contain data values. We can store such a table in a cube with dimensions D1, D2, …, DN, Index, and Measures. The elements of Measures correspond to the data columns of V1, …, VM.

For example, a denormalized order detail table might contain the following columns:

- Customer ID
- **Region** (of Customer)
- Order No
- Date Placed
- **Month** (Placed)
- **Product Code**
- Qty

Assuming that Region, Month, and Product Code are dimensions, the information in the table can be stored in a cube with dimensions:

- Region
- Product Code
- Month
- Index
- Measures

Where Measure's elements are:

- Customer ID
- Order No
- Date Placed
- Qty

The table will yield quantity summaries at any level of aggregation for Region Product or Month by means of a query that requests

Order(region, product, month, "Total", "Qty").

Individual transactions can be displayed via a view that suppresses empty elements of Index.

6. Comparison with the Relational Model

In the relational model, the primary objects are flat files or tables. These can be formalized as relations. A relation is simply a subset of the cartesian product of two or more sets, that is to say a set of tuples. The operations on this model consist of selecting subsets of the rows, or joining multiple tables into larger tables, or summarizing sets of rows. Columns can be calculated in terms of other columns, but there is no concept of a cell, or of calculating individual cells.

The relational model can implement a relational OLAP, or ROLAP, which is able to accumulate, cross-tabulate and report on historical data using what is called a star schema. Most data warehousing and business intelligence (BI) applications use this approach. However, ROLAP provides no way of expressing cells as a function of other cells thus the ability to express modeling calculations is very limited. Also, the lack of support for a quick update / recalculation cycle do not make it appropriate for interactive analytics.

The functional and relational models are not two separate worlds. The mathematical concepts are quite close. In fact a function is a special kind of relation. If function f maps a set D to another set R, then f can be thought as a relation between D and R since f is a subset of $D \times R$.

Also a relation can be expressed as a function. Given a relation R where

$R \subseteq C1 \times C2 \times \ldots \times Cn$

R can be expressed as a function f_R where

$f_R: C1 \times C2 \times \ldots \times Cn \rightarrow \{ 0, 1 \}$

and the value of f_R for a tuple x is 1 if and only if $x \in R$.

In practice, the functional model depends on the relational model for a number of things:

- Most of the input for the functional model comes from actual transactions. Accordingly, the functional model must perform SQL queries to import and aggregate relational data from operations.
- Some planning applications require tracking major predictable transactions individually. This is clearly the case when forecasting product demand for large-ticket items.
- Adjustments, such as journal entries, must be recorded as transactions.
- The results of the planning process may need to be fed to operations in the form of transactions.
- Dimensions typically originate as relational tables, and most consolidation hierarchies are derived from attributes of the members stored in those tables.
- And for some analyses it may be necessary to calculate sets of members using expressions that are better expressed as relational queries.

7. Instantiation Considerations

Describing details of an instantiation of the functional model that delivers adequate performance would not be possible given the constraints of this paper. Another hurdle is that the technologies

involved are typically proprietary. Nonetheless, we can describe some general principles and considerations that any such implementation must adhere to.

7.1 Calculation on Demand

A functional data base must provide quick response to changes in data. Just as in a spreadsheet, calculations should be transparent to the user. They should occur automatically, without user initiation and the time to perform the calculation should be in the sub-second range in order not to interfere with the user's interactive thought process.

Since the functional model places no constraints on the size of cubes or dimensions, users should be able to create very large models. Users are also free to define calculations and thus a value in a cell may be used as a "parameter" in the calculation of millions of other cells. For example, a cell that contains a growth factor parameter may affect all the cells in a forecast period.

Changes to one cell may also affect a large number of consolidated cells. For example, in a cube with 10 dimensions where each dimension has 3 levels of consolidations in addition to the leaf level, the value of any leaf cell will affect the value of 410 – 1, or 1,048,575 consolidated cells. Such geometric "explosion" is very typical of large multidimensional models.

The use of parametric cells and geometric explosion of consolidated cells effectively rules out pre-calculation both from a response time and storage standpoint.

When users are interacting with models, they typically work on limited "slices" of cubes, which means they would be looking at a miniscule subset of all the cells that would be affected by changes they make. Accordingly, the only practical way of providing high speed interactive calculations is by performing calculations only on demand.

7.2 In-memory Hypersparsity

The ability to write expressions that access random cells in cubes requires that those cells be readily accessible and thus requires that they be stored using in-memory data structures. Such structures must not only provide quick random access, but must also efficiently store cubes that are "hypersparse", i.e., where only a very small portion of the cells are populated. Real business models with a density of less than 10^{-10} are not atypical.

Also, since values may be inserted into cubes at random and in real time, the structures used must be able to efficiently handle random insertions.

7.3 Calculation Caching and Invalidation

A parametric cell that is used in the calculation of many other cells may in turn be the result of a calculation. Clearly it would make no sense to repeatedly re-calculate the parametric cell. Thus to deliver good calculation performance, a functional data base must store calculated cells for potential future use.

Once those cells are stored, there also need to be mechanisms that invalidate them whenever the cells they depend on change values.

7.4 Parallelized Calculations

At the moment, the main scalability tool available for handling large scale calculations is massive parallelization. A properly implemented functional data base server must be able to perform calculations in parallel whenever possible while maintaining the proper order of calculation. This if further complicated by the need to maintain high performance and data integrity with multiple users making changes to the base data.

7.5 Mixed Disk/Memory Storage

In some applications of the functional model, for example in retail, the volume of data may be so large as to preclude the use of memory. Such large data volumes typically occur in historical periods where data do not change and thus random access in-memory data structures may not be required and thus a hybrid storage approach may be appropriate.

7.6 Simplicity

As with most new technology, the most difficult challenge is to keep things simple. This is a key to the success of the electronic spreadsheet and one that a functional data base must retain. In particular, it must self-optimize to relieve the user of as many "tuning knobs" as possible.

8. Relation to Previous Work

Spreadsheets in RDBMS for OLAP [26] proposes an cell oriented multidimensional extension to the SQL language as a way to overcome many of the limitations of relational data bases. It is not clear to what extent the instantiation it proposes attains the goal of implicit automatic calculations of the functional model proposed here.

We should note that the term *OLAP* [10] is used to refer to various technologies that are based on multidimensional cubes. The term *multidimensional data base* is also often used. [13]. In some cases MDB has been used [12] to refer to implementations of OLAP that are memory based and thus deliver much higher query performance. The term MOLAP (in-memory OLAP) has also been used to refer to such products. In most cases, however, simply using memory rather than disk does not provide the same level of interactivity, or responsiveness to updates as is contemplated in the functional model. In order to deliver that, products need to implement cell addressing, cell expressions and immediate recalculation after data or model changes, i.e., implement the functional model.

A cautionary note: one might be tempted to implement the functional model on top of an RDBMS containing cubes stored using the star schema. But, in order to provide the performance necessary for cell level calculations and interactivity, the data would have to be loaded into in memory cubes designed for that purpose; even if the RDBMS is implemented in memory. At the end there would be two versions of the data, one in the RDBMS and one in the in memory cubes. The updates could be easily synchronized but the RDBMS would not contain the derived cells available in the in-memory cubes. Attempting to also store calculations in the RDBMS and synchronize the two could introduce enormous delays. For example, if in the illustration of section 4.3 a single exchange rate, for one currency and one time period, in cube Fx were to be changed, a slice of cells, those corresponding to that currency and time period in P&L for all Products and Regions would need to be updated. In the end, both choices are undesirable: a) a rather useless set of raw data RDBMS cubes with no calculations, or b) an exact copy of the in-memory cubes available at the cost of inordinate delays.

The notion of functional or declarative programming has been around for some time [2, 5]. While the functional data model presented here contains functional programming elements – the expression of cell calculation rules is done through a functional language – the name functional does not refer to that aspect, but

rather that the fundamental data objects are expressed as set-theoretic functions instead of relations. The functional or declarative programming paradigm has been applied to OLAP [9].

The term *functional data model* has also been around for some time [3]. The closest prior work is the DAPLEX language [8]. It represents information as entities and functions, where entities are discrete sets and functions are defined over entities. Derived functions are expressed as queries, and their values can be singletons – numeric or string – or sets of entities.

The underlying data in DAPLEX is still a set of transactions, i.e., a relational model, while the underlying data in the functional model presented here consists of multidimensional arrays of values, i.e., single-valued functions defined over a cartesian product of the sets. It developed independently from the need to perform "spreadsheet" analysis to large volumes of multidimensional data. It concentrates on the functionality required for this purpose, for example defining a cell in a multidimensional grid as a function of other cells, and consolidating cells in a hierarchical structure. As in the case of the relational model, the functional model developed in response to a practical need and its formalization was developed subsequently.

9. Future Directions of Research

As a result of this history, the functional model is not as strong as DAPLEX or SQL in their ability to express set operations. Incorporating features such as set selection and functions whose results are sets rather than single values is one possible future direction of development for the current embodiment of the functional model.

There are also very significant challenges in the implementation of the functional model, primarily centered on scalability of data volumes and numbers of users. The fact that typically solution models are tightly integrated implies that as such models grow, decomposing them on multiple servers may not always be possible. Also, as the numbers of users for an application increase, handling the volume of updates and recalculations may require spreading the workload over multiple servers. This also presents significant challenges in order to maintain one consistent version of a dynamic truth.

Another potential opportunity for further scalability, made all the more challenging by the requirement of frequent updates, is the application of a massively parallel paradigm such as map reduce.

10. Conclusion

One key purpose of this paper is to encourage the use of a model that is different but complementary to the more established relational one and which delivers superior value when it comes to analytics.

Because of its generality, flexibility and simplicity, the relational model has supplanted all other transaction oriented technologies, and remains to this day the dominant paradigm. People in the IT community have been brought up on relational databases, and the relational model is often the only data modeling tool in their arsenal.

Managers that do analytics for the most part do not look at the world relationally. As explained before, their preferred paradigm is that of the electronic spreadsheet. The spreadsheet allows them to freely develop business forecasts and analyses: calculate cells containing future periods as a function of cells in prior periods, perform basic consolidations, and easily produce tabular reports that put relevant numbers next to each other so they can be compared. The mathematical models users develop with spreadsheets are fundamental to the automation of analytics since they incorporate the experience and insights of people who run the business. And while most of the historical data in such spreadsheets originates in reports from relational databases, such databases do not meet their analytics needs and users often go through the trouble of transcribing them manually in order to take advantage of the modeling flexibility of the spreadsheet.

This situation, which at first sight appears adequate, has at least two unfortunate outcomes:

- Spreadsheets have significant limitations, both in terms of the volumes of data they can handle, and their ability to handle applications with any complexity beyond the basics. They quickly become unmanageable and un-maintainable and put users in the infamous spreadsheet hell.
- The IT community mistrusts the spreadsheet because it does not offer the control and security of a database, so they attempt to deliver analytics solutions using the relational model. Thus most analytics facilities they provide are based on the relational model. Relational-based OLAP facilitates reporting but its modeling capabilities are severely limited and users are unable to interact with them. Their read-only nature precludes what-if analyses or interactive planning.

The net effect is that the most important and strategic business benefits of computing technology, those of prospective analytics, are not adequately captured. Thus the need to raise awareness of the functional model, particularly in the IT community.

Our hope is also to encourage others to further develop these notions, and to improve the functionality scalability and performance of functional databases thereby contributing to the efficiency with which organizations are run. This, in the end, should benefit us all.

11. Trademarks

12. References

[1] "Basic Set Theory". Stanford Encyclopedia of Philosophy. http://plato.stanford.edu/entries/set-theory/primer.html

[2] Bird R.S., Wadler P.L. *Introduction to Functional Programming*. Prentice Hall (1988).

[3] Buneman P., Functional Database Languages and the Functional Data Model. A position paper for the FDM workshop (June 1997) http://www.cis.upenn.edu/~peter/fdm-position.html.

[4] Codd, E. F. A Relational Model of Data for Large Shared Data Banks. Comm. ACM 13, 6 (June, 1970)

[5] Henderson P. *Functional Programming Application and Implementation*. Prentice Hall (1980).

[6] Hrbacek, K and Jech, T *Introduction to Set Theory*, Third Edition, Marcel Dekker, Inc., New York 1999.

[7] Lang, Serge (1987), *Linear algebra*, Berlin, New York: Springer-Verlag, ISBN 978-0-387-96412-6

[8] Shipman D.W. The functional data model and the data language DAPLEX. ACM Transactions on Database Systems 6(1), March 1981, p. 140-173.

[9] C.M. Necco, J.N. Oliveira, L. Quintas. A functional approach for on line analytical processing, 2006. WISBD, III Workshop de Ingeniería de Software y Bases de Datos. CACIC'06, XII Congreso Argentino de Ciencias de la Computación, Universidad Nacional de San Luis, Argentina.

[10] E. F. Codd. Providing olap to user-analysts: an it mandate, Apr. 1993. Technical Report, E. F. Codd and Associates.

[11] P. Trinder, A functional data base, D.Phil Thesis, Oxford University 1989.

[12] G. Colliat, Olap relational and multidimensional database systems, SIGMOD Record, 25(3), (1996)

[13] T. B. Pedersen, C. S. Jensen, Multidimensional database technology, IEEE Computer 34(12), 40-46, (2001)

[14] C. J. Date with Hugh Darwen: A Guide to the SQL standard : a users guide to the standard database language SQL, 4th ed., Addison Wesley, USA 1997, ISBN 978-0-201-96426-4

[15] Ralph Kimball and Margy Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition), p. 393

[16] George Spofford, Sivakumar Harinath, Chris Webb, Dylan Hai Huang, Francesco Civardi: MDX-Solutions: With Microsoft SQL Server Analysis Services 2005 and Hyperion Essbase. Wiley, 2006, ISBN 0-471-74808-0

[17] What is OLAP? www.olapreport.com/fasmi.htm

[18] BI Veredict http://www.bi-verdict.com/

[19] Oracle Essbase http://en.wikipedia.org/wiki/Essbase

[20] Jedox OLAP http://www.jedox.com/en/products/jedox-olap.html

[21] MIS Alea Server http://en.wikipedia.org/wiki/MIS_AG

[22] Infor PM OLAP Server http://www.infor.com/content/brochures/infor-pm-olap.pdf/

[23] Karsten Oehler Jochen Gruenes Christopher Ilacqua, IBM Cognos TM1 The Official Guide, McGraw Hill 2012

[24] http://www.google.com/patents/US5592666

[25] http://www.google.com/patents/US5319777

[26] Andrew Witkowski et al, Spreadsheets in RDBMS for OLAP, Proceedings of the 2003 ACM SIGMOD international conference on Management of data, June 09-12, 2003, San Diego, California