Interactive, Flexible, and Generic What-If Analyses Using In-Memory Column Stores

Stefan Klauck[1], Lars Butzmann[1] Stephan Müller[1], Martin Faust[1], David Schwalb[1], Matthias Uflacker[1], Werner Sinzig[2], and Hasso Plattner[1]

[1] Hasso Plattner Institute, University of Potsdam, Germany
{firstname.lastname}@hpi.de
[2] SAP SE, Walldorf, Germany,
werner.sinzig@sap.com

Abstract One well established method of measuring the success of companies are key performance indicators, whose inter-dependencies can be represented by mathematical models, such as value driver trees. While such models have commonly agreed semantics, they lack the right tool support for business simulations, because a flexible implementation that supports multi-dimensional and hierarchical structures on large data sets is complex and computationally challenging. However, in-memory column stores as the backbone of enterprise applications provide incredible performance that enables to calculate flexible simulation scenarios interactively even on large sets of enterprise data.
In this paper, we present the HPI Business Simulator as a tool to model and run generic what-if analyses in an interactive mode that allows the exploration of scenarios backed by the full enterprise database on the finest level of granularity. The tool comprises a meta-model to describe the dependencies of key performance indicators as a graph, a method to define data bindings for nodes, and a framework to specify rules that describe how to calculate simulation scenarios.

Keywords: Business Simulation, Column Store, What-If Analysis

1 Introduction

Today's reporting tools offer an unprecedented flexibility. Companies can dive into their data, i.e filter for arbitrary criteria and drill down into hierarchies to explore the data at the finest level of granularity. Companies wish to exploit this flexibility not only for reporting but also for forecasting and simulating. They want to define potential future scenarios and calculate how these influence their businesses. With the help of what-if analyses, they can evaluate simulation scenarios in terms of their goal fulfillment and support decisions in day-to-day operations.

Mathematical models, hereinafter also called calculation models, are one way to define the dependencies between measures, i.e. the logic how changes of one key performance indicator (KPI) influence other KPIs. Value driver trees, such

Accepted at DASFAA 2015.
The final authenticated version is available online at https://doi.org/10.1007/978-3-319-18123-3_29.

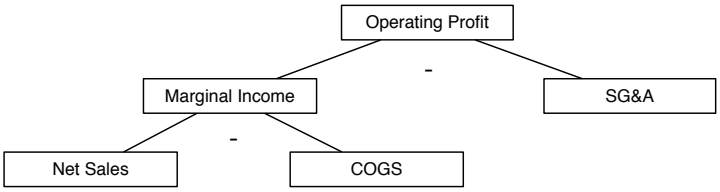

Figure 1: Value driver tree for the operation profit.

as the DuPont model [1], are a well-known method to model KPIs with linear dependencies among each other [2]. Figure 1 shows a driver tree for the operating profit, which can be calculated by subtracting selling, general and administrative expenses (SG&A) from the marginal income. The marginal income depends on net sales and cost of goods sold (COGS). The values of the nodes base on enterprise data, e.g. on G/L account transactions as *actuals* or a combination of the actuals and planned sales, production costs, and expenses as forecasted KPIs. On the basis of enterprise data, companies want to define scenarios, i.e. changes of the data, which reflect changing KPIs, and calculate the effects on other KPIs. The challenge to define and run simulation scenarios is not the mathematical complexity of the calculation, but the combination of a generic calculation model and the large amount of data the model builds on, which enables the users to define flexible scenarios and calculate them interactively.

For many years, the biggest obstacle has been the speed to access enterprise data with all the relevant criteria to allow flexible and interactive simulations. To run such simulations for net sales requires scanning sales documents with all their associated line items. The corresponding tables can comprise billions of records, specifying relevant attributes like sales date, sales volume, price, product, and customer, but also hundreds of other attributes. The advent of columnar in-memory databases has increased the performance of queries accessing few attributes of large data sets, which enables the development of new enterprise applications on top of it [3, 4].

This paper presents the HPI Business Simulator, a tool to create, modify and run what-if analyses interactively. This comprises two things: First, a way to define what-if simulations, which consist of a calculation model, the specification of data bindings between KPIs of the model and data of database tables, and the support to specify simulation scenarios. Second, the concept of a simulation tool to define and edit what-if simulations as well as to calculate scenarios.

After this short introduction into the problem domain, Section 2 presents the theoretical concepts of the HPI Business Simulator. Its implementation is shown in Section 3. Section 4 presents related work. The paper closes with a presentation of the conclusions and offering an outlook for future work.

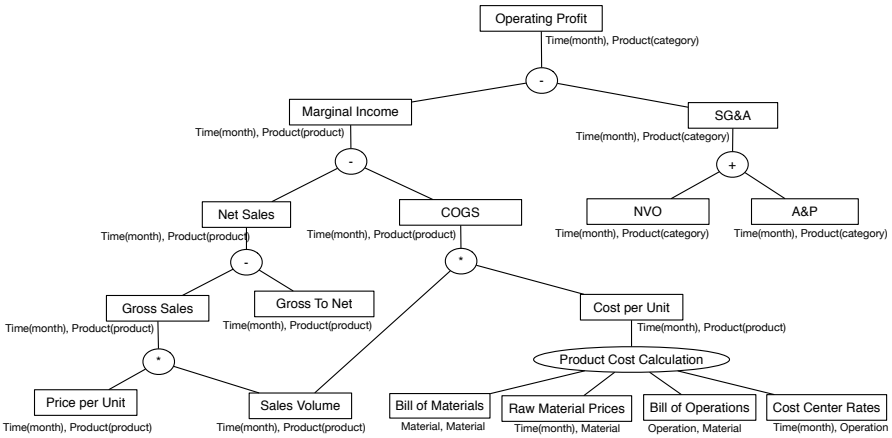

Figure 2: Calculation model for the operating profit as extension of the value driver tree in Figure 1.

2 Simulation Model

The HPI Business Simulator is a proof of concept for implementing generic what-if analyses. This section presents its theoretical concepts called *simulation model*. The simulation model consists of three parts: the generic calculation model to describe the drivers and the dependencies between them, the data binding to connect drivers with data, and a way to specify scenarios and to calculate their results.

2.1 Calculation Model

Calculation models are hypergraphs and extend value driver trees by supporting complex operations and the loosening of the tree structure for dependencies. Each node has a language dependent name, a measure with a unit, and freely definable dimensions for specifying further criteria of a measure. Nodes can be connected with other nodes by *operations*, which are hyperedges. The simulation model specifies available dimensions including their hierarchy levels. The time, for example, can be structured hierarchically into years and months. Operations define the dependencies between nodes, i.e. the way one can calculate the values for the result node based on the data of input nodes. Operations can be one of the four elementary operations addition, subtraction, multiplication, and division. Besides, users can define own, more complex operations, e.g. a product cost calculation based on raw material prices, the bill of materials, cost center rates, and the bill of operations. Figure 2 shows a calculation model for the operating profit.

Data of a node can be seen as a data cube. Combining data cubes with elementary operations works as for one-dimensional data. However, one has to

define rules for dimension handling. For additions and subtractions, the result data cube has the intersection of dimensions from all input data cubes. The records of the input data cubes with the same dimension values are combined to a single output record. Result data cubes for multiplications receive the union of dimensions from all input cubes. When combining data cubes with divisions, the calculation model has to define the dimensions for the result data cube so that combining these dimensions with the dimensions of the divisor results in the dimensions of the dividend. Table 1 shows a multiplication and subtraction of data cubes.

Product	Time	Quantity
Product 1	01/15	5
Product 1	02/15	10
Product 2	02/15	5

(a) Sales Volume.

Product	Price
Product 1	20
Product 2	5

(b) Product Prices.

Time	Expenses
01/15	50
02/15	100

(c) Expenses.

Product	Time	Sales
Product 1	01/15	100
Product 1	02/15	200
Product 2	02/15	25

(d) Sales (= Sales Volume * Product Prices).

Time	Profit
01/15	50
02/15	125

(e) Sales - Expenses.

Table 1: Data cube calculations.

User defined operations work in a similar way as elementary operations with the distinction that arbitrary algorithms can define the calculation logic for the result data cube. A simplified version of a product cost calculation with resolving a hierarchical bill of materials (BOM) is described in the following. We assume that product costs are only affected by raw material prices and thus ignore labor, energy, and machine costs, which would occur in a real world scenario. A calculation of these costs bases on the bill of operations (BOO) and follows the same calculation logic as the BOM resolution. Figure 3 and Table 2 present an exemplary BOM and its database representation.

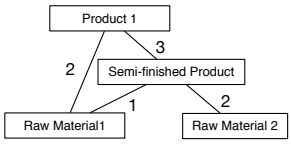

Figure 3: BOM as graph.

Child Material	Parent Material	Quantity
Raw Material 1	Product 1	2
Semi-finished Product	Product 1	3
Raw Material 1	Semi-finished Product	1
Raw Material 2	Semi-finished Product	2

Table 2: BOM as table.

Thereby, the table stores *parent-child* relationships, i.e. how much of a *child material* is needed to produce a *parent material*. Child materials can be raw materials and semi-finished products. Parent materials are semi-finished or end products. The second input of our cost calculation scenario is a table with raw material prices. To calculate product costs, we have to resolve the BOM so that products are represented as costs of raw materials, but no semi-finished products. Therefore, we traverse the BOM graph recursively or iteratively. The following equations present the resolution for the exemplary BOM.

$$\begin{aligned} costs_{Product1} &= 2 * costs_{RawMaterial1} + 3 * costs_{Semi-finishedProduct} \\ &= 5 * costs_{RawMaterial1} + 6 * costs_{RawMaterial2} \end{aligned} \tag{1}$$

2.2 Data Binding

Nodes of the calculation model can obtain their data cubes in two ways. First, they can query their data directly from data sources. Second, they can calculate their values by solving the equation defined by the operation between connected nodes and themselves. For the first case, data bindings are required. Our work focuses on relational databases as data sources. Data bindings define the database connection and query to calculate the data cube with all dimension values. Additionally, data bindings have to specify how to filter the cube and reduce the level of detail to calculate aggregates for higher levels of hierarchies. When specifying the data binding, we have to ensure that the values of all nodes can be calculated unambiguously, meaning that the data has to be sufficient and consistent.

2.3 Simulation Scenarios

Based on the calculation model and data binding, the data cubes of all nodes can be calculated. These data cubes are the basis of what-if analyses. In addition, it is required to specify in which direction changes propagate through the model. Thereby the direction of propagation is not allowed to contain cycles. A *simulation scenario* is a set of changes to the data of nodes. A single change specifies a node, optionally filter conditions to limit the change to a subset of records, and how the specified values are changed. The HPI Business Simulator implements three types of simulation changes: an overwrite for records of the data cube, an absolute adjustment by a delta value, and a relative adjustment by a linear factor.

3 HPI Business Simulator

This section describes the HPI Business Simulator, a proof of concept to implement generic what-if analyses. For the implementation, we have engaged with a Fortune 500 company in the consumer goods industry and discussed their needs in the area of what-if analyses. Based on their input and the data set they provided to us, we implemented the HPI Business Simulator. In this section, we

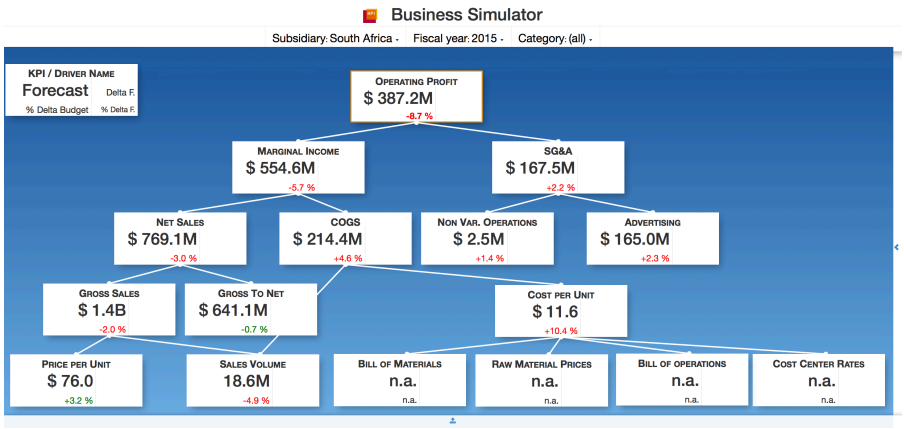

Figure 4: Screenshot of the HPI Business Simulator with anonymized data.

explain features and implementation details of the HPI Business Simulator, why in-memory column stores enable flexible and interactive simulations, and the benefits of the HPI Business Simulator compared to existing tools.

3.1 Features and Implementation

The HPI Business Simulator is a browser-based graphical tool to define simulation model instances and calculate simulation scenarios using SAP HANA. Figure 4 shows a screenshot of the business simulator with the calculation model from Figure 2. The legend in the top left corner shows the visualized metrics of the calculation model. Each node contains the driver name, a *forecast* value, and the delta of the forecast to the *budget*. Thereby, the forecast is calculated as a combination of the actuals up to today and the budget until the end of the planning horizon. After specifying simulation changes, the effects are included in the forecast and two additional values are displayed: the difference between the old and new forecast and the change in percentage.

The calculation model can be edited by adding, deleting, and dragging the graph components, i.e. nodes and edges. To change node properties as the name, unit of measure, available dimensions, and data binding, one can open a detailed view on the right side. The information to store simulation model instances consists of three parts, describing the available dimensions with their hierarchies, the nodes, and the operations. The top dropdown menu offers a way to drill down into the data, e.g. by selecting a specific product category. In this case, the HPI Business Simulator recalculates the drivers with the filter condition and displays the new values. To define simulation scenarios, the user can select a node and open the simulation interface in the bottom.

To calculate the operating profit for our use case, we worked with a denormalized table, which contained transactional as well as plan data. Following,

important attributes of the table are explained. The first two columns, *Document ID* and *is Plan Data*, identify single records and declare whether the record belongs to the actuals or the plan data. *G/L Account Description* indicates to which driver the record belongs to. The following attributes, i.e. *Product, Brand, Category, and Time*, specify criteria, which are mapped to dimensions of the simulation model. Finally, *Quantity, Quantity Measure, Amount, and Currency* describe the measures for the record. In particular, the values for *Quantity* and *Amount* are aggregated to calculate the displayed value for each node.

3.2 In-Memory Technology as Enabler for Interactive Simulations

In-memory databases provide high performance and flexibility. This allows to access the complete business data dynamically even of large companies at the finest level of granularity, opening completely new opportunities. However, the new dynamic nature and flexibility require adapting the way data is accessed and consumed. Recently, self-service tools for business analytics fulfill this need by providing intuitive interfaces to explore and analyze data. Typically, those tools are focused on historic data and do not consider dependencies of the underlying value drivers and metrics. With the HPI Business Simulator, we want to extend self-service tools for business analytics and provide an intuitive approach to model value dependencies in enterprises with an interactive simulation environment, leveraging the full computational power of in-memory databases.

In-memory column stores, such as SAP HANA [5], are the key enabler for interactive simulations on large enterprise data. First of all, analytical queries are accelerated compared to traditional disk-based systems by keeping all data in main memory. Analytical queries are the basis to retrieve the data for business simulations. Based on the specified parameters, the HPI Business Simulator calculates aggregates of disjoint data sets: the ones which are affected by simulation parameters and the ones which are not. The values of the nodes can then be calculated by applying changes and combining theses aggregates. The second benefit is the columnar data layout. Since the data bindings of the simulation model describe only columns that are required for a calculation, the amount of processed data is kept to a minimum. This functionality is especially beneficial in the context of enterprise systems where data is typically very wide and sparse, with up to 400 columns per table [6]. A third benefit is an aggregate cache, which is a transparent caching engine inside the database [7]. Other than classic materialized views, the aggregate cache does not create a hard copy of the data and as a result does not return stale data. The aggregate cache leverages the internal table representation in certain column stores like SAP HANA or Hyrise [8] and always works on the latest data. During a typical simulation session, different scenarios are analyzed and compared. The differences between the scenarios can vary, but are usually small. Consequently, the executed queries are similar and can therefore benefit from the aggregate cache.

The performance to calculate a simulation scenario depends on many factors like the number of records stored in the underlying tables, the number of nodes with a data source, the number of simulation changes, the number of columns

used to specify the filter criteria, the granularity of filter criteria, and the data distribution. For a preliminary performance test, our HPI Business Simulator ran on a data set comprising 300 million records of customer data. The initial on-the-fly calculation of an aggregate on a single enterprise class server with 128 cores and 256GB of RAM running SAP HANA was calculated within a second. In that way, simulations can be defined and run interactively.

3.3 Benefits Compared to Existing Tools

Generic model. Existing simulation tools are targeted for specific processes and are difficult to modify or extend to capture new use cases. Users may want to extend the calculation model for the operating profit (see Figure 2) to distinguish between advertising channels, which enables to simulate changes for a specific kind of advertisement. The HPI Business Simulator allows the modification of existing calculation models and definition of new ones without changing the source code of the simulation tool or rewriting the application.

Support for complex calculations. Calculation models should not be limited to tree structures and elementary operations as in the DuPont model. Instead, graph structures and custom calculations should be supported.

Using in-memory column stores. Exploiting information at the finest level of granularity requires the capability to operate on terabytes of data. Simulation tools without an enterprise database as backbone have to load pre-aggregated measures, which come along with a loss of information. The speed of current in-memory databases supports aggregating large amounts of data on the fly within seconds, which enables us to define and calculate flexible what-if scenarios interactively. In addition, simulations are always calculated on consistent and up-to-date data.

Collaborations. The HPI Business Simulator supports iterative what-if analyses of users in different roles: The management defines KPIs and adapts the definitions in case the calculation model does not support a desired simulation scenario. More technical staff with extensive knowledge about the data schema is responsible for providing the model with data and the integration of new data sources. Concrete simulation scenarios are discussed and worked out by potentially multiple controllers, which are responsible for different business areas.

4 Related Work

Golfarelli et al. introduce a methodology and process to design what-if analyses [9]. Compared to our approach, they describe what-if simulations in a more general way. In particular, they divide the process to design simulations into seven phases: goal analysis, business modeling, data source analysis, multidimensional modeling, simulation modeling, data design and implementation, and validation. Our approach uses a multi-dimensional data model, which Golfarelli et al. see as most suitable to design what-if analyses. To describe the actual simulation model, they propose an extension of UML 2 activity diagrams [10]. In

comparison to our work, they do not implement the simulation model instances as applications, specify data bindings, nor define how to calculate simulation scenarios. Furthermore, we see the dependencies described by a calculation model instance as subject to changes and extensions.

In the field of data cubes, which are the basis to calculate our simulation scenarios, most research focuses on materialized data cubes [11–15]. Gray et al. introduce a data cube operator as generalization and unification of following database concepts: aggregates, group by, histograms roll-ups and drill-downs, and cross tabs [11]. Further papers cover efficient implementation [12, 13] and maintenance techniques [14, 15] for materialized data cubes. In contrast to previous work, we calculate the required aggregates of the data cubes on the fly.

A further research area in the field of what-if analyses is the combination of spreadsheets and SQL. Spreadsheets have an easy to understand interface, but do not build on consolidated enterprise data, which is usually stored in a RDBMS. On the other side, SQL lacks the support for array-like calculations as Witkowski et al. claim in [16]. Their idea is to combine both and offer a spreadsheet-like computation in RDBMS through SQL extensions. In [17], they continue that research and introduce a way to translate MS Excel computations in SQL. Using their approaches for what-if analysis comes with two drawbacks. First, it does not encapsulate the definition of the simulation model so that it is not tangible, but only expressed by multiple formulas spread over many table cells. Second, MS Excel is limited to the two-dimensional representation and cannot visualize graphical dependencies between nodes appropriately.

5 Conclusion

In this paper, we presented our vision of generic, flexible, and interactive business simulations, enabled by the performance capabilities of columnar in-memory databases. In particular, we presented the HPI Business Simulator and its theoretical concepts to specify and run enterprise simulations. We proposed a meta-model to describe what-if analyses comprising a calculation model, the data binding and simulation parameters. Implementing this meta-model, simulation model instances can be created and edited, such as one for the operating profit. Based on a simulation model instance, scenarios are specified and calculated interactively. With the performance of in-memory column stores, such as SAP HANA, what-if analyses can include millions of records and work on the finest level of granularity to enable interactive and fully flexible simulations.

This paper names performance factors that influence the calculation of simulation scenarios. In this field a deep analysis can be conducted. Furthermore, future work can investigate how to optimize the calculation of scenarios, for example by exploiting cases in which nodes query the same table or by optimizing queried data cubes so that their granularity is sufficient to calculate connected nodes. The visualization of complex calculation models can be another field for future investigations.

References

1. Chandler, A., Salsbury, S.: Pierre S. Du Pont and the Making of the Modern Corporation. BeardBooks (2000)
2. Zwicker, E.: Prozeßkostenrechnung und ihr Einsatz im System der integrierten Zielverpflichtungsplanung. Techn. Univ. Berlin (2003)
3. Plattner, H.: A Common Database Approach for OLTP and OLAP Using an In-Memory Column Database. SIGMOD (2009)
4. Plattner, H.: The Impact of Columnar In-Memory Databases on Enterprise Systems. VLDB (2014)
5. Färber, F., Cha, S.K., Primsch, J., Bornhövd, C., Sigg, S., Lehner, W.: SAP HANA Database - Data Management for Modern Business Applications. SIGMOD (2011)
6. Krüger, J., Kim, C., Grund, M., Satish, N., Schwalb, D., Chhugani, J., Dubey, P., Plattner, H., Zeier, A.: Fast Updates on Read-Optimized Databases Using Multi-Core CPUs. VLDB (2011)
7. Müller, S., Plattner, H.: Aggregates Caching in Columnar In-Memory Databases. VLDB (2013)
8. Grund, M., Krüger, J., Plattner, H., Zeier, A., Cudre-Mauroux, P., Madden, S.: HYRISE: A Main Memory Hybrid Storage Engine. VLDB (2010)
9. Golfarelli, M., Rizzi, S., Proli, A.: Designing What-if Analysis: Towards a Methodology. DOLAP (2006)
10. Golfarelli, M., Rizzi, S.: UML-Based Modeling for What-If Analysis. DaWak (2008)
11. Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. Data Min. Knowl. Discov. (1997)
12. Harinarayan, V., Rajaraman, A., Ullman, J.D.: Implementing Data Cubes Efficiently. SIGMOD (1996)
13. Sismanis, Y., Deligiannakis, A., Roussopoulos, N., Kotidis, Y.: Dwarf: Shrinking the PetaCube. SIGMOD (2002)
14. Mumick, I.S., Quass, D., Mumick, B.S.: Maintenance of Data Cubes and Summary Tables in a Warehouse. SIGMOD (1997)
15. Roussopoulos, N., Kotidis, Y., Roussopoulos, M.: Cubetree: Organization of and Bulk Incremental Updates on the Data Cube. SIGMOD (1997)
16. Witkowski, A., Bellamkonda, S., Bozkaya, T., Dorman, G., Folkert, N., Gupta, A., Shen, L., Subramanian, S.: Spreadsheets in RDBMS for OLAP. SIGMOD (2003)
17. Witkowski, Andrew and Bellamkonda, Srikanth and Bozkaya, Tolga and Naimat, Aman and Sheng, Lei and Subramanian, Sankar and Waingold, Allison: Query by Excel. VLDB (2005)