World Academy of Science, Engineering and Technology
International Journal of Economics and Management Engineering
Vol:3, No:10, 2009

Multidimensional Performance Management

David Wiese

***Abstract*—**In order to maximize efficiency of an information management platform and to assist in decision making, the collection, storage and analysis of performance-relevant data has become of fundamental importance. This paper addresses the merits and drawbacks provided by the OLAP paradigm for efficiently navigating large volumes of performance measurement data hierarchically. The system managers or database administrators navigate through adequately (re)structured measurement data aiming to detect performance bottlenecks, identify causes for performance problems or assessing the impact of configuration changes on the system and its representative metrics. Of particular importance is finding the root cause of an imminent problem, threatening availability and performance of an information system. Leveraging OLAP techniques, in contrast to traditional static reporting, this is supposed to be accomplished within moderate amount of time and little processing complexity. It is shown how OLAP techniques can help improve understandability and manageability of measurement data and, hence, improve the whole Performance Analysis process.

***Keywords*—**Data Warehousing, OLAP, Multidimensional Navigation, Performance Diagnosis, Performance Management, Performance Tuning.

I. Introduction

DATABASES are growing rapidly in scale and complexity. High performance, availability and further service level agreements need to be satisfied under any circumstances to please customers. In order to tune the database management systems (DBMSs) within their complex environments, maximize productivity and efficiency and minimize the total cost of ownership of an information management platform, Performance Management, that is the collection, storage and analysis of performance-relevant data for monitoring, capacity planning and tuning purposes, has become of fundamental importance [1][2].

Performance management of complex database information systems can be regarded as part-science and part-art. In practice, proactive and reactive strategies ought to be developed and complement one another in order to ensure acceptable end user experience. Performance monitoring and analysis tools aim high at assisting database and system administrators in coping with these tedious tasks.

However, reporting against and using such tools often reveals an overwhelming flood of data, making it almost impossible for analysts and even experienced database administrators to separate the wheat from the chaff.

David Wiese is with the University of Jena, 07743 Jena, Germany (phone: +49-3641-9-46367; fax: +49-3641-9-46302; e-mail: david.wiese@uni-jena.de).

Consequentially, it seems more than warrantable to yield a more intuitive and flexible source of information on which to base performance-analytic decision-making.

Facing the associated problems and challenges, this paper discusses the potentials of widely adopted multidimensional concepts to represent hierarchical layers of data and allow explorative, interactive and intuitive problem analyses. Such a multidimensional model can reduce overhead and diminish the learning curve involved in understanding performance traces and their correspondence to performance reports. Decision making will be accelerated and adequate reaction to changes can be sped up. But, most importantly, one of the most valuable assets - information - will be leveraged.

The rest of this paper is organized as follows. Section 2 starts with a general introduction into performance management as well as common performance data collection and storage techniques. We then present the basic principles of OLAP and the multidimensional data model in Section 3, and discuss the application of multidimensional analysis, followed along with an exemplary scenario that shows the principles of the multidimensional methodology in Section 4. Section 5 critically examines merits and drawbacks of the OLAP paradigm in Performance Management. Lastly, we conclude with a compulsory summary and an outlook of ongoing and future work.

II. Performance Management

Information systems and database usage scenarios range from stand-alone systems to complex combinations of database servers and clients running on multiple platforms. In order to meet business requirements the achievement of adequate performance is critical to all these environments.

In order to achieve at least sufficient performance, pursuing a customized, proactive and reactive performance management strategy becomes vitally important. Performance management involves the collection, storage and management of measurement data in order to enable resource monitoring, tuning and optimization, early problem diagnosis and repair, as well as capacity planning and workload forecasting (see Figure 1). Instead of detecting problems when they occur, or, worse, have already begun degrading system performance, the proactive effort targets at avoiding problems in the first place. Usually, this is done by capacity planning of IT resources, adopting best practices in architecture and application design, as well as performing regular monitoring of central performance indicators and storing the collected data in a long-term repository for subsequent analyses and predictions.

However, not all adverse effects on the system can be avoided beforehand. Hence, a sophisticated reactive methodology for localizing the problem, pinpointing when and where it occurs, who is affected and what resources are involved, determining the root-cause(s) and finally fixing the situation is needed as well.

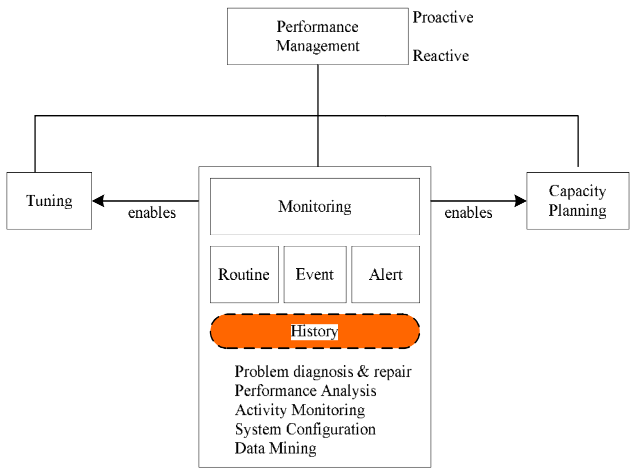


Fig. 1: Performance Management Overview

Commonly monitoring can be categorized as subsumed below [3]:

1. *Routine monitoring*

   Involves the regular collection of information about the workload and the stress on the system during periods of normal and peak activity. This monitoring technique is an essential routine for both system and database administrators and typically involves ascertaining minimums, maximums, average, hit ratios, etc. for key performance elements (such as transactions, users, CPU, memory, disk space, and SQL) over days and weeks with the main purposes to keep components under supervised control, document how the system and the database are performing day-to-day and based on the collected history, identify and isolate potential problems and perform capacity planning.

2. *Event Monitoring*

   Routine monitoring assures the total control over the system, most of the time. Nevertheless, unanticipated problems (delays, deadlocks, increased number of transactions) might sometimes occur. Event monitoring involves looking out for specific events in a short interval of time (short history) that may either identify a specific problem known to degrade performance or conceivable problems in the near to immediate future in order to take quick corrective action to rectify the problem. In other words, there probably needs to be a very short delay between information collection and a corrective response.

3. *Alert Monitoring*

   This type of monitoring is required when end users discover or suspect a problem, or predefined lower and upper thresholds for special performance variables are being exceeded. The database or system administrators are notified and need to identify the situation's root cause in order to apply the appropriate action. Unlike routine and event monitoring, which are planned occurrences and are designed to have low overheads on the managed system, alert monitoring is driven by imminent problem situations and may impose significant overhead on the managed system. The more detailed information one collects, the more processing is necessary in order to collect it.

The next subsections briefly present further goals, techniques and sources of collecting, as well as storing different kinds performance data. Collecting and storing monitoring history is accomplished to gain (quick) access to and analyze (recent) past-performance data, compare it to present data, perform strategic and preventive planning and enable data mining[1]

A. *Performance Data Collection*

Performance of the system and its resources is typically measured using different metrics. Those measurement variables represent resource allocations and the overall system state at a specific point in time or over a period of time.

Ideally, there should be a single, global monitor that supervises all layers in the software stack and provides a holistic view by continuously extracting useful performance indicators for the application, network, DBMS, operating system and hardware (CPU, memory, disk) into a central, consistent and integrated short- (nearly real time) and long-term diagnostic performance metric repository. The truth is that in practice every layer has its own set of proprietary monitoring and analysis tools which hardly communicate and do not deliver an acceptable basis for analytical decision making.

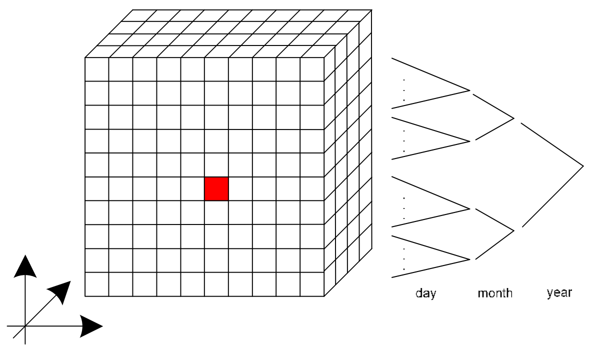
Considering IBM DB2 UDB metrics originate from various internal sources [4] (these are, among others, the System Monitor, Explain Facility, DB and DBM configuration parameters, registry variables, schema definitions, log files, etc.) as well as from external monitoring facilities (e.g. operating system and network data, DB2 Performance Expert [5], IBM Tivoli Monitoring[2], etc.) with different locations and formats.

As in most DBMSs, within DB2, commonly encountered problems that need proper attention arise from an inappropriate database, application design, the overall architecture, bad programming and incorrect use of SQL as well as insufficient tuning and can be prioritized as follows:

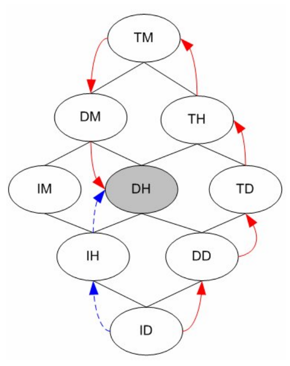
1. Constrained resources (CPU, I/O, memory and network bandwidth)

[1] Extracting valid, useful, previously unknown, and comprehensive information from data and using it for the automated prediction of trends and behaviors, as well as the automated discovery of previously unknown patterns.

[2] IBM Tivoli Monitoring, IBM Corporation, http://www.tivoli.com/products/index/monitor/.



(a) Multidimensional concepts: dimensions, measures



(b) Navigation visualized with a lattice

Fig. 2: Multidimensional model as basis for navigation

2. Locking contention / conflicts
3. DB shared memory shortages (buffer pools, sort heaps, application global memory, and catalog and package caches)
4. Lack of appropriate indexes (sub-optimal access paths) and inadequate maintaining (runstats and reorgs)
5. Poorly written applications (inefficient database design, inefficient SQL code)

Despite of the origin and format, metric data typically belongs to one of the following element types [6]:

1. *Counter*
   Accumulates the number of times a specific event or activity occurs. Hence, counter values only increase during monitoring. Some examples of DBMS counters include deadlocks detected, number of lock escalations, number of rows read, or number of sort overflows.
2. *Gauge*
   Indicates the current value for an item. Gauge values can go up and down, depending on database activity. Some examples of gauges include Locks Held, Total Lock List Memory in Use, and Connections Involved in Deadlock.
3. *Water mark*
   Indicates the highest (maximum) or lowest (minimum) value an element has reached since monitoring was started. Some examples of water marks include maximum number of concurrent connections or maximum number of coordinating agents.
4. *Information*
   Provides details of monitoring activities. This can include items such as database names, partition names, aliases and path details.
5. *Timestamp*
   Indicates the date and time that an activity took place, e.g. by providing the number of seconds and microseconds that have elapsed since January 1, 1970.
6. *Time*
   Returns the number of seconds and microseconds spent on an activity.

B. Storage and Presentation of Performance Data

Collected measurement data needs to be stored to allow sophisticated analyses. As conclusions and predictions about performance, tracking back problems or even correlating imminent incidences to past events can only be made with regard to the past, a long-term repository seems more than warrantable.

Storing performance data can be done in manifold manners. The expressive capabilities of the implied data model are crucial for later analysis potentials. Actually, one might argue that the physical basis of the data is out of relevance, as long a suitable, intuitive interface to the user, transparently hiding complexity and structure, exists. However, we presume an applicable physical model as a basis for analyses, as no further, complex and time-consuming integrations and transformations need to be performed at run-time.

An in-depth classification and evaluation of the most common storage mechanisms like flat files, RDBMS, spreadsheets, etc. can be found in [7].

Present performance reporting capabilities, delivered either from monitoring tools or from the RDBMS itself, rather overwhelm or confuse end-users like administrators and analysts with a poorly conceived information overload than supporting in decision making. Furthermore, the rudimentary SQL analysis capabilities turned out to be unsuitable for more sophisticated reporting requirements. Manually filtering out the feasible facts from relational tables becomes exhausting, time-consuming and error-prone, even for experienced users.

Online Analytical Processing (OLAP) technologies have gained ground within the last decade in business-oriented decision support environments and the business community. Therefore, it seems legitimate to evaluate the merits of utilizing OLAP techniques for analyses of performance-relevant measures.

III. Multidimensional Performance Analysis

OLAP can be seen as a set of technologies and tools that assist in the quick analysis of (business) data [8][9][10]. For the analysis using OLAP, the manifold relationships among (business) data are mapped to multidimensional data structures.

TABLE I SAMPLE DIMENSIONS AND HIERARCHIES

Dimension	Hierarchy Name	Hierarchy Paths
DB_OBJECTS	DB_OBJECTS_LOGICAL DB_OBJECTS_PHYSICAL DB_OBJECTS_LOGICAL_EEE DB_OBJECTS_PHYSICAL_EEE	TABLE - SCHEMA - DB - INSTANCE TABLE - TBS - BP - DB - INSTANCE TABLE - SCHEMA - DBPG - DB - INSTANCE TABLE - TBS - BP - DBPG - DB - INSTANCE
TIME	TIME	MICROSEC - SEC - MIN - HOUR - DAYTIME - DAY - MONTH - YEAR
WORKLOAD	WORKLOAD_NORMAL WORKLOAD_DETAIL	STMT - APPL STMT_OP - STMT - TA - CONNECTION - APPL
USER	USER	USER - OS_GROUP
PARTITION	PARTITION	PARTITION

TABLE II EXEMPLARY MEASUREGROUPS

Measuregroup	Measuregroup Association List (MGAL)
BP_IO_INFO[1]	MICROSEC (x TBS) (x STMT_OP x USER) (x PARTITION)
STMT_COUNT	(APPL x USER x) MICROSEC (x DB) (x PARTITION)
APPL_STATUS	APPL x DB x MICROSEC
STMT_INFO	(DB x) MICROSEC x STMT
STMT_OP_INFO	(DB x) MICROSEC x STMT_OP
UOW_INFO	MICROSEC x TA (x DB)
TABLE_ACTIVITY	TABLE x MICROSEC (x PARTITION)
INTERNAL_COUNTS	(STMT_OP x USER x) MICROSEC (x DB) (x PARTITION)
ROW_COUNTS	(STMT_OP x USER x) (TABLE x) MICROSEC (x PARTITION)
AGENTS_N_CONNS	INSTANCE x MICROSEC (x PARTITION)
LOCKS_N_DEADLOCKS	DB x MICROSEC (x PARTITION)
INFORMATIONAL_DBCFG MODIFIABLE_DBCFG DBMCFG	DB x MICROSEC (x PARTITION) DB x MICROSEC (x PARTITION) INSTANCE x MICROSEC

TABLE III EXEMPLARY CUBES

CubeNo	Cube Dimension Hierarchies	Measuregroups (MG(CubeNo))
1	TIME x WORKLOAD_DETAIL	BP_IO_INFO, INTERNAL_COUNTS, UOW_INFO
2	TIME x DBOBJECTS_LOGICAL	BP_IO_INFO, STMT_COUNT, TABLE_ACTIVITY, INTERNAL_COUNTS, ROW_COUNTS, AGENTS, LOCKS_N_DEADLOCKS, INFORMATIONAL_DBCFG, MODIFIABLE_DBCFG, DBM_CFG
3	TIME x DBASE x USER x WORKLOAD_DETAIL	BP_IO_INFO, STMT_COUNT, APPL_STATUS, INTERNAL_COUNTS, STMT_INFO, STMT_OP_INFO, UOW_INFO
4	TIME x DBOBJECTS_PHYSICAL	MG(2)
5	TIME x DBASE x WORKLOAD_NORMAL	STMT_COUNT, APPL_STATUS
6	TIME x DBOBJECTS_LOGICAL_EEE x PARTITION	MG(1) \ {DBM_CFG}
7	TIME x DBOBJECTS_PHYSICAL_EEE x PARTITION	MG(1) \ {DBM_CFG}

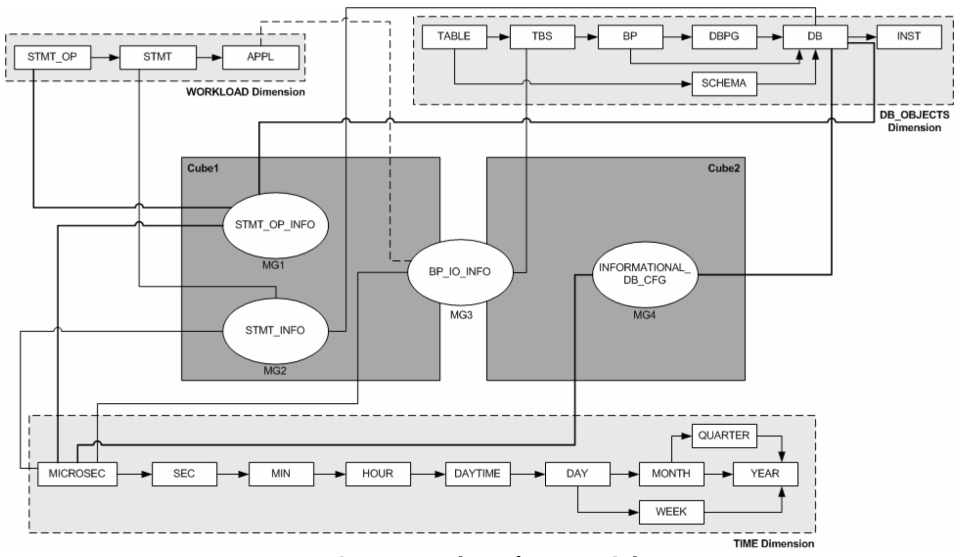


Fig. 3: Two sample Performance Cubes

This section highlights multidimensional representation of performance indicators and shows how OLAP techniques can help improve understandability and manageability of measurement data and, hence, improve the whole performance analysis and analytic decision making process.

A. *The Multidimensional Data Model*

The multidimensional paradigm can be regarded as an extension of the relational approach. Within two-dimensional relations each tuple attribute value is determined by the intersection of a specific row and column, both can be regarded as dimensions. Dimensions in multidimensional models are simply higher-level perspectives on the data. In fact, most of the deployed multidimensional data models are implemented relationally (ROLAP approach) via star or snowflake schemas [11].

The basis of the multidimensional data model (see Figure 2(a) and detailed explanation in [12]) is rooted in the difference between qualifying and quantifying data that are reflected by two key concepts: dimensions and measures.

Dimensions in multidimensional models serve for the unambiguous, orthogonal structuring of the data space and describe different ways of looking at the information (e.g. time, database objects and workload).

The intersection of dimensions acts as an index and identifies the data points the analysts intend to analyze, the so-called measures (e.g. index pool and data page hit ratios, physical writes/reads, number of commits/rollbacks as well as dynamic/static SQL statements). In contrast to the descriptive, textual and qualifying dimension attributes, these are mostly numerical, additive and quantifying information.

The structure of the data is similar to that of an array. Like the dimensions of an array, dimension levels provide the indexes for identifying individual cube cells, as well as the situation in which the measures were taken and where they are meaningful. For example, a buffer pool hit-ratio of 13% is useless. Provided with the TIME and name of the BUFFER POOL its meaning and validity becomes clear. As it is often sufficient to draw decisions from a high-level point of view, not everybody is interested in detailed (raw) data. In order to increase manageability, dimensions are broken down into hierarchy levels (e.g. table - tablespace - buffer pool - database - instance; secs - mins - hours - days) with differing granularities. Such an abstraction process is an instinctively known human activity. Utilizing these hierarchical relationships, analysts can drill-down/roll-up along their data to view different levels of granularity and abstraction and only deal with the level of information appropriate to their current assignment.

Basically, measures can be divided into atomic and non-atomic, derived ones. Derived measures can be computed from either other measures (atomic or derived) or from dimension attributes. Typically, measures are of a numerical nature that allows the classification into additive (can be aggregated by simple arithmetical operations), semi-additive (can be aggregated along some of the dimensions' hierarchies and not along others) and non-additive (cannot be aggregated at all) ones. Despite the fact of structural aggregation-capability, the computation of aggregation functions might not be semantically meaningful for all the measures (e.g. SUM(high-water-mark) does not make sense along a hierarchy, MAX(HWM) seems more appropriate). Informative measures that are neither aggregatable nor numeric are no rarity (e.g.

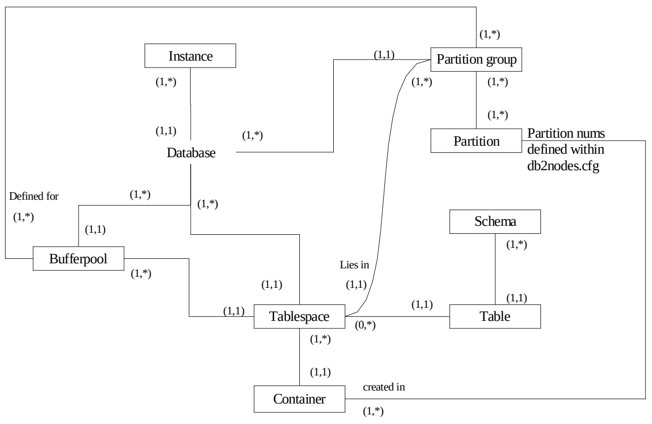


Fig. 4: Implicit DB2 Database Object Hierarchies

DB2 System Monitor elements of type information). Type and semantical meaning of a measure is solely determined by the analysis scenario and the prevalent environment.

Abstraction cannot solely be achieved with the concept of dimension hierarchies. Measuregroups (MG) group sets of measures semantically by subject area. This can be useful for systems with a countless number of measures in order to reduce and manage complexity.

B. Multidimensional Navigation

Aligning complex data by dimensions that influence the key business factors as well as representing and capturing natural hierarchical relationships in data gives users the ability to comprehend the conceptual scheme and recognize dependencies and implications more intuitively.

Typical multidimensional navigation is a top-down approach. With the intention to obtain a "big-picture" view, the user normally starts analysis from an upper level of abstraction (entry point) by looking at a specific set of summarized problem- or situation-relevant key performance metrics (totals, averages, counts, etc.) that might not seem quite right. Issuing a combination of sequential and mutually dependent OLAP queries [13], from there he crawls along paths through the multidimensional data space in order to reveal inter-connections and dependencies between metrics that should be examined next. When requiring insight into more fine granular data, the analyst can incrementally increase the level of detail with a drill-down operation. Rolling-up does the opposite, it moves upwards in hierarchical relationships, thereby decreasing level of detail. Besides, setting focus on specific regions of data that appear promising can be performed by slicing, dicing and pivoting on the current data cube.

A typical business analyst will not query against relational tables directly[3]. Reasons for that are insufficient knowledge and understanding or experience as well as missing time to build complex SQL statements. Instead, he specifies his

[3] Presuming a relational OLAP implementation via star or snowflake schemas.

requests with a graphical navigation interface[4]. In contrast to common static reporting, it is not necessary anymore to generate an entirely new report in order to see more details or to change the perspective on the data. With an appropriate graphical tool, it can be as easy as a mouse click, and the requested points of interest can be displayed. Access to valuable information can be gained quickly and intuitively without having to learn a new query or programming language or dialect. However, as the number of dimensions and therefore complexity grows, it naturally becomes more difficult to visualize the model of the database and make it understandable for human users. Therefore, the number of dimensions should be kept small and clearly arranged.

Regarding multidimensional performance analyses there are many ways for finding the actual cause of the problem. Utilizing dimension hierarchies for navigation provides the analyst with the flexibility to explore various paths of interest that may all lead to the potential root cause(s). Some paths seem more naturally intuitive to the user, but require more time for analysis. In order to reduce system down times or time for decisive analyses, the paramount goal is to find the cause(s) as quickly as possible. From a set of representative query sequences, ideally, it seems optimal to select the one with the lowest accumulated execution time. Apparently, this can be difficult and depends on the skill and experience of the user. Interestingly, the authors of [14] propose a layer on top of multidimensional data that provides for automated problem isolation of performance problems. They claim that automated drill down is sufficiently general so that it offers the possibility of improved productivity in a wide range of computing environments. However, analysts still must explore different dimensions manually in order to determine which provides the best characterization of the problem being isolated.

The sequential nature and diversity of exploration possibilities (possible paths) in multidimensional analysis can be clarified visually using the lattice approach (see [15] for details). The multidimensional lattice represents the solution space of the problem with vertices that present unique combinations of dimension levels, hence queries on the base data. Edges between vertices represent single drill-down/roll-up operations. The art in analyses lies in considering only those queries of importance. Figure 2(b) aims at showing various "movement" strategies and illustrating the possible navigation process of an analyst. Each node is labeled with a combinatory instance of dimension levels. The chosen hierarchies for both dimensions are: (T)able - (D)atabase - (I)nstance and (M)inute - (H)our - (D)ay. The lowest node represents the highest level of abstraction with the least cardinality in contrast to the uppermost one which describes information at highest detail. The arrows (dotted and drawn through) ought to represent two of numerous sequential paths (with operations restricted and simplified to drilling-down and rolling-up) an analyst could take to best localize the (database-related) problem. Obviously, the dotted path seems more effective regarding time and effort to arrive at the conclusion. Regrettably, the shortest path is not always the most evident to a human analyst. In addition, analysis paths may also lead to a dead-end. Often, users have to go back in history, back-trace their routes through the data and find alternative paths to draw conclusions.

[4] See www.tdwi.org/marketplace for a comprehensive list of vendors and tools.

IV. Exemplary DB2 Performance Analysis Scenario

In order to practically demonstrate some basic structures and scenarios, we will use IBM DB2 UDB. The presented concepts, however, can be generalized to common performance scenarios with differing DBMS, OS and further layers of the software stack.

A. Multidimensional structures within IBM DB2

Fortunately, the DB2 architecture already provides a comprehensive set of object hierarchies as depicted in Figure 4. Relationships are mostly "contains" and are visualized using undirected arrows with the appropriate (n,m)-cardinalities.

The following sample dimensions and parallel hierarchies can therefore be derived (Table 1). Hierarchies roll up from child to the parent and abbreviations are basically the same used in various DB2 documentations (DBPG = Database Partition Group, TA = Transaction). The time dimension is often considered separately and can be found in almost every multidimensional data model. It allows time-series analyses and comparisons among different points in time. This is essential for performance analysis in that the root cause of a problem might originate a long time ago in the past, leading to an irresolvable bottleneck at the end.

Table 2 provides the list of possible measuregroups and the associated dimensions (brackets indicate alternative hierarchy levels). The established dimensions and measuregroups can be combined in many ways forming the cubes presented in Table 3. Two cubes and four of the defined measuregroups as well as their associated (shared) dimensions are visually depicted in Figure 3.

B. Exploring DB2 performance data multidimensionally using SQL

Aside from utilizing sophisticated tools, multidimensional data can also be analyzed using SQL directly, applying the ANSI-SQL99 OLAP extensions [13]. Operationss include the extended GROUP BY functionality (grouping sets, GROUPING function, ROLLUP and CUBE operators, etc.) as well as OLAP functions (aggregating, partitioning, windowing, ranking, etc.) to form the typical queries on star schemas. Those queries are characterized by affecting a large number of tables and mainly applying equi-join predicates between fact and dimension tables, as well as local selective predicates on the dimension tables.

A typical problem isolation scenario consists of multiple successional steps and might look like the following (the measures have been taken from the Performance Expert long-term repository and are embedded within a star schema

TABLE IV START OF ANALYSIS

DB	BP	TBS	Time Interval	BPHR
PEDEMO	BUILDINGS	ALL	ALL	99.61
PEDEMO	FRUITS	ALL	ALL	20.35
PEDEMO	IBMDEFAULTBP	ALL	ALL	89.55
PEDEMO	LOCKTEST	ALL	ALL	99.69
PEDEMO	MOTION	ALL	ALL	98.98
PEDEMO	ALL	ALL	ALL	83.08
SAMPLE	IBMDEFAULTBP	ALL	ALL	85.35
SAMPLE	ALL	ALL	ALL	85.35
TESTDB	IBMDEFAULTBP	ALL	ALL	82.39
TESTDB	ALL	ALL	ALL	82.39
ALL	ALL	ALL	ALL	83.49

TABLE V DRILL-DOWN AND RANKING

DB	BP	TBS	Time Interval	BPHR
PEDEMO	FRUITS	GROWTH	ALL	15.27
PEDEMO	FRUITS	TRADE	**ALL**	49.77
TESTDB	IBMDEFAULTBP	USERSPACE1	ALL	65.72
TESTDB	IBMDEFAULTBP	SYSCATSPACE	ALL	81.96
SAMPLE	IBMDEFAULTBP	SYSCATSPACE	ALL	85.02
PEDEMO	IBMDEFAULTBP	SYSCATSPACE	ALL	88.93
TESTDB	IBMDEFAULTBP	SYSTOOLSPACE	ALL	97.03
SAMPLE	IBMDEFAULTBP	SYSTOOLSPACE	ALL	98.75
PEDEMO	MOTION	VEHICLES	ALL	99.01
PEDEMO	BUILDINGS	LOCATIONS	ALL	99.64

TABLE VI FURTHER DRILL-DOWN

DB	BP	TBS	Time Interval	BPHR
PEDEMO	FRUITS	GROWTH	Month11/Day2	57.20
PEDEMO	FRUITS	GROWTH	Month11/Day2	2.12
PEDEMO	FRUITS	GROWTH	Month11/Day8	24.41
PEDEMO	FRUITS	GROWTH	Month11/Day8	33.58
PEDEMO	FRUITS	GROWTH	Month11/Day9	55.71
PEDEMO	FRUITS	GROWTH	Month11/Day9	14.09
PEDEMO	FRUITS	GROWTH	Month11/Day9	10.76
PEDEMO	FRUITS	GROWTH	Month12/Day14	12.52
PEDEMO	FRUITS	GROWTH	Month12/Day15	21.97
PEDEMO	FRUITS	TRADE	Month11/Day2	76.51
PEDEMO	FRUITS	TRADE	Month11/Day2	52.07
PEDEMO	FRUITS	TRADE	Month11/Day8	56.99
PEDEMO	FRUITS	TRADE	Month11/Day8	54.79
PEDEMO	FRUITS	TRADE	Month11/Day9	77.90
PEDEMO	FRUITS	TRADE	Month11/Day9	58.45
PEDEMO	FRUITS	TRADE	Month11/Day9	45.58
PEDEMO	FRUITS	TRADE	Month12/Day14	45.54
PEDEMO	FRUITS	TRADE	Month12/Day15	55.50

TABLE VII DETERMINING ROOT CAUSE TIME OF PROBLEM

DB	BP	TBS	Time Interval	BPHR
PEDEMO	FRUITS	GROWTH	Month11/Day2/12:45-13:00	65.99
PEDEMO	FRUITS	GROWTH	Month11/Day2/13:00-13:15	10.85
PEDEMO	FRUITS	GROWTH	Month11/Day2/13:15-13:30	10.50
PEDEMO	FRUITS	GROWTH	Month11/Day2/13:30-13:45	10.01
PEDEMO	FRUITS	GROWTH	Month11/Day2/13:45-14:00	8.98
PEDEMO	FRUITS	GROWTH	Month11/Day2/14:00-14:15	8.46
PEDEMO	FRUITS	GROWTH	Month11/Day2/14:15-14:30	8.22
PEDEMO	FRUITS	GROWTH	Month11/Day2/14:30-14:45	8.18
PEDEMO	FRUITS	GROWTH	Month11/Day2/14:45-15:00	7.97
PEDEMO	FRUITS	GROWTH	Month11/Day2/15:00-15:15	8.37

equivalent of cube number 4 from Table 3). Only basic SQL constructs that support OLAP have been used, such as ROLLUP, RANK, GROUP BY.

The database administrator (DBA) observes (possibly in return to user complaints) increasing (database) response times. Assuming this problem occurs due to multiple applications questioning the database at the same time, he decides to first monitor buffer pool activity, as one possible indicator of overall system health.

Starting root cause analysis he computes the buffer pool hit ratios for the highest level of abstraction in each hierarchy at first. The subsequent query determines the average buffer pool hit ratios[5] on a buffer pool and database level and results in Table 4.

```
SELECT   dbo.db_name as DB, dbo.BP_name as BP,
         AVG(fact.pool_hit_ratio) as BPHR
FROM     d_db_objects dbo, f_bp_io_info fact,
         d_time t
WHERE    fact.d_db_objects_id = dbo.d_db_objects_id
         AND fact.d_time_id = t.d_time_id
GROUP BY ROLLUP(dbo.db_name, dbo.BP_name)
```

Next, the DBA further drills down into tablespace level and applies ranking functionality to obtain the 10 worst buffer pool hit ratios (see Table 5) using a similar SQL statement.

[5] As the buffer pool hit ratio metric is of type gauge, we can use the AVG aggregation function to present values of coarser detail.

Navigating the cube's data, the analyst notices that the BPHR decreases below 30%. Having detected the problem, he tries to judge which abstraction hierarchy best localizes the problem. In our simple example it is obvious to drill more into detail for the FRUITS buffer pool and the GROWTH tablespace as it was done in Table 6. By steadily descending lower in the (time) hierarchy, he seeks more in detail and constraints the data for further navigations. Now, it seems interesting to pinpoint the exact time interval the performance started suffering (Table 7). From observing configuration change history at the pinpointed time interval(s), it turns out that the size of the buffer pool has been decreased at run-time at about 1 PM. From that moment, the hit ratio rapidly dropped down.

V. Critical Evaluation

Existing performance management solutions, however, cannot become multidimensional without effort as several fundamental challenges and problems before analyzing the multidimensional structures prevail. In order to harness powerful multidimensional analysis potentials, the following additional steps are required beforehand or even repeatedly at run-time, causing additional overhead:

1. *Identification*
   Determination of appropriate sources, their structure and deliverable data.
2. *Conceptualization/Creation*
   Establishment of multidimensional target structures, as well as their relational representations.
3. *Mapping*
   Specification how to translate data from one representation to another.
4. *Filling*
   Transferring of the source data into the created multidimensional target representatives in compliance with predefined mapping guidelines.

Although a large number of commercial solutions are currently available on the market to support multidimensional structuring and ETL processing in a comprehensive way, in [12] we argue that the price, lack of standards, differing functionality and proprietary nature, as well as diverse other problems legitimate the idea of deploying a lightweight, platform-independent, general multi-purpose framework with little footprint on the system that is supposed to comply with all four above steps in an intuitive way for end users. Hence, in [12] we show how a simple, lightweight and extensible data mart creation framework (XDMF) has been developed to easily rearrange and aggregate performance data and construct subject-oriented multidimensional data marts as well as valid and complex SQL-based data transformation and movement mappings in order to support subsequent analyses. Data marts can then be created on-demand at run-time in order to organize data (into symptoms or categories) and to allow subsequent analyses over time, tracking back problems or even correlating imminent incidences to past events. The framework has been utilized for creating data marts with performance-relevant data on demand. It turned out that performance monitoring (especially historical performance analysis) is greatly facilitated by using multidimensional structures.

Another great difficulty for users is to find and select the appropriate set of cubes for conducting navigational analyses on. This can be a highly non-trivial and possibly iterative trial-and-error process that should be accomplished by experienced DBAs only. At the beginning, the user has to get some idea which measures (cubes) are best suited for addressing the problem and providing an entry point into following analyses. In [12] some aspects of an autonomic cube advising and selection component in the area of performance tuning are introduced. Basic elements of the therein described architecture are a knowledge base, correlating typical and recent performance problems to possible performance cubes, a learning component that analyzes previous cube determinations and success ratios to adopt the priority list of cube proposals, and triggering mechanisms (e.g. health indicators exceeding user defined thresholds) for automatically initiating the advising process.

Furthermore, fast query times are crucial for OLAP. The ability to intuitively manipulate huge quantities of data and to accomplish analyses within and across dimensions in order to answer important (business) questions requires quick retrieval of information. In practice, however, "quick" applies only for the most common requests that, in return, need to rely on a well-tuned underlying physical schema. Response times, however, suffer the more calculation needs to be done. Aside from widespread indexing techniques, pre-aggregating and storing frequently accessed data in materialized views helps to reduce run-time calculation overhead. Techniques have been developed for deciding what subsets of a data cube to pre-compute, for estimating the size of multidimensional aggregates, and for indexing pre-computed views. Approaches to aggregation affect both the size of the database and the response time of queries. If more values are pre-calculated, a user is more likely to request a value that has already been calculated, thus the response time will be faster. However, if all possible values are pre-calculated, not only will the size of the database be unmanageable, but the time it takes to aggregate will be intolerably long. Furthermore, updating and refreshing the according materialized views can be time-consuming and must be considered as well. Optimization techniques emphasize lowering query costs with little regard to maintenance (e.g. refreshing tables and materialized views). Update costs are believed to be unimportant as systems are oriented towards nightly or offline batch updates. The truth is, in upcoming business scenarios and in our performance management scenario, data warehouses become more and more real-time storage and access containers (real-time warehousing). Furthermore, global 24x7 businesses do not have any downtimes or timeframes of low activity and least of all "night"-periods.

Nevertheless, effectively determining pre-aggregation candidates and managing the update of the materialized tables are far beyond this work's scope. Further, in-depth information can be found in [15] [8].

Designing effective materialized views requires adequate prior planning. The designer has to anticipate the query workload and the likely user behavior to identify patterns for accessing tables, and frequently performed aggregations. A highly interesting approach addressing this problematic can be found in [16]. The author presents a mathematical model and a graphical notation for capturing knowledge about typical multidimensional interaction patterns in OLAP systems, taking into account the session oriented, interactive and navigational nature of the user query behavior. Furthermore, an architecture is presented to speed up OLAP systems at runtime by using speculative execution techniques based on a prediction of the user query behavior. The concept of predicting query behavior can be considered rather irrelevant for performance analysis requirements. Irregular system states that could not be anticipated through proactive routine monitoring occur at run-time and require immediate reactive actions by means of multidimensional analyses to isolate the problem and finding its root cause.

VI. Summary and Outlook

Applying OLAP techniques to performance and system management supports sophisticated problem detection and analysis instead of naively guessing problem causes and parameters that take effect. In fact, OLAP technologies offer a wide variety of features to simplify decision making. By storing decision-supporting data in spreadsheet-like multidimensional data structures, end users (DBAs, knowledge worker, analysts) can access their large (historical) data in a simple and understandable way - by the dimensions of their business. Dimensions are closely related to the nature of (business) data and offer a very intuitive way of organizing and selecting data for retrieval and analysis. Beyond it, individual, subject-specific multidimensional cubes are capable of providing a basis for several reports that can be shared or stored for later (re)use.

Data access is enterprise-wide, but each user can flexibly manipulate (by slicing, dicing, rotating, drilling, etc.), view, analyze and compare data from various perspectives and see only facts relevant for the assigned activities. An appropriate tools collection presumed, access to valuable information can be gained without having to learn a new query or programming language or dialect. These kinds of analyses are able to enhance productivity and skills of even inexperienced analysts.

This, however, requires an initial learning and preparation time. Requirements need to be specified, an appropriate multidimensional data model must be designed, a complete Warehousing architecture, seamlessly integrating a set of sophisticated tools, needs to be established and last but not least, administration, maintenance and end user personnel must be trained. One of the biggest challenges lies in the trade-off between more intuitive analyses and the overhead in creating, filling and navigating respective structures. Chances, however, are promising, especially, looking at the development of OLAP for business decision making.

However, it must be kept in mind that neither the most advanced analysis techniques can belie an offer of information (the available performance measures) of low quality. Therefore, it must be of paramount importance to retrieve detailed and consistent, but only relevant and expressive measures for later analyses.

Further research work delves into the possibilities of enhancing autonomic database performance tuning with multidimensional performance data collection, storage and analysis techniques. In [17] we describe our workload driven system for best-practice oriented autonomic database tuning, called Autonomic Tuning Expert (ATE). ATE’s architecture is based on widely accepted and influential technologies and industry-proven products that are combined in a way to build a component-based MAPE loop for automating typical tuning tasks. The ATE infrastructure is designed to be the core component of an ecosystem that enables DBAs to design, exchange, adapt, and execute best-practice tuning methods. It is intended to leverage the multidimensional model for more sophisticated automated problem detection and diagnosis, as well as trend prediction.

The research field of autonomic database performance tuning is very promising. However, we do not believe that highly skilled DBAs will ever get replaced by intelligent autonomic database administrating tools. Automation is in fact a great option for the “usual case” but there always will be exceptional cases that need to be taken care of. Those even more justifying the need of a profound, holistic, short-term and long-term knowledge base.

References

[1] S. Chaudhuri, G. Weikum. Rethinking Database System Architecture: Towards a Self-tuning RISC style Database System. In Proceedings of VLDB, 2000.

[2] IBM Corp. DB2 UDB ESE V8 non-DPF Performance Guide for High Performance OLTP and BI. IBM Redbooks, 2004.

[3] Alur, N; Balaji, R.; Miskimen, M.; Stolz-Hofmann, D.: IBM DB2 UDB Performance Expert for Multiplatforms - A usage guide. Redbook. IBM Corp, 2003.

[4] Alur, N.; Falos, A.; Lau, A.; Lindquist, S.; Varghese, M.: DB2 UDB/WebSphere Performance Tuning Guide. Redbook. IBM Corp, 2003.

[5] Chen, W-J; Ma A.; Markovic, A.; Midha, R.; Miskimen, M.; Siders, K.; Taylor, K; Weinerth, M.: DB2 Performance Expert for Multiplatforms V2. Redbook. IBM Corp, 2005.

[6] IBM Corp.: DB2 Universal Database - System Monitor Guide and Reference (Version 8). IBM Corp, 2004.

[7] Sriram, C.; Martin, P.; Powley, W.: A Data Warehouse for Performance Management of Distributed Systems. Dept. of Computing and Information Science. Queen's University at Kingston, 1998.

[8] Bauer, A.; Günzel, H.: Data Warehouse Systeme Architektur, Entwicklung, Anwendung. dpunkt. Heidelberg. 2000.

[9] Codd, E.F. et al.: Providing OLAP to User-Analysts: An IT mandate. Arbor Software, 1993.

[10] Inmon, W.H.: Building the Data Warehouse. 3rd edition. Wiley. New York, 2002.

[11] Kimball, R.: The Datawarehouse Toolkit. John Wiley & Sons, New York 1996.

[12] Wiese, D.: Framework for Data Mart Design and Implementation in DB2 Performance Expert. Diploma thesis. University of Jena and IBM Böblingen. 2005.

[13] Mogin, P.: OLAP Queries and SQL1999. Issues in Database and Information Systems. Victoria University of Wellington, 2005.

[14] Hart, D.G.; Hellerstein, J.L.; Yue, P.C.: Automated drill down: An approach to automated problem isolation for performance management. In: Proceedings of the Computer Measurement Group, 1999.

[15] Harinarayan, V.; Rajaraman, A.; Ullman, J. D.: Implementing data cubes efficiently. Proceedings of the 1996 ACM SIGMOD international conference on Management of Data (SIGMOD'96). ACM Press, June 1996.

[16] Sapia, C.: On Modeling and Predicting Query Behavior in OLAP Systems. Proceedings of the International Workshop on Design and Management of Data Warehouses. Heidelberg, Germany, 14. - 15. 6. 1999.

[17] Wiese, D.; Rabinovitch, G.; Reichert, M. and Arenswald, S.: Autonomic Tuning Expert - A framework for best-practice oriented autonomic database tuning. In Proceedings of Centre for Advanced Studies on Collaborative Research (CASCON 2008). Ontario, Canada, October 27 - 30, 2008.