On Modeling and Predicting Query Behavior in OLAP Systems

Carsten Sapia
FORWISS - Bavarian Research Center for Knowledge-Based Systems
Orleansstr. 34, D-81667 Munich, Germany
sapia@forwiss.de

Abstract

Interactive multidimensional data analysis tools (mostly OLAP systems) are the predominant frontend tools for end users in data warehouse environments. Thus, the design of these systems is an important part of the data warehouse design itself. This paper contributes to the important design step by discussing the modeling of user query behavior and its benefits. We present 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. To exemplarily show that the modeling of user behavior is not only beneficial during the logical and physical design phase, we also present an architecture to speed up OLAP systems at runtime by using speculative execution techniques based on a prediction of the user query behavior.

1 Introduction

OLAP systems are the predominant frontend tools for data warehouse systems. Their main characteristic is that the user has a multidimensional view of the data stored in the warehouse ([DSBH98]). Unlike with traditional information systems where the logical data schema is hidden underneath an application layer, the logical multidimensional schema of an OLAP system is directly used by the end user to formulate queries. Thus, the MD schema determines the types of queries the user can formulate. Therefore, the design of this schema is an important task of the data warehouse design itself. As the design of the multidimensional schema has a direct impact on the functionality of the system, the design process should be driven by the user analysis requirements. That means that knowledge about the anticipated behavior of the user is needed throughout the design and implementation process. Surprisingly enough, very little work has been devoted to models and notations to capture and formalize this knowledge. Many design methodologies are mainly driven by the structure of the operational data sources. We argue that it is desirable for the quality of the design to enrich these approaches with more user centric techniques.

Therefore, we start by demonstrating the benefits of modeling knowledge about the anticipated user behavior in OLAP systems (section 1.1) and examining the special characteristics of interactions (section 1.2). Section 2 surveys related work to show that the existing approaches do not fully match these characteristics. From the observations about the characteristics of OLAP systems, we derive a formal description technique for multidimensional query sequences and user/task specific interaction patterns and profiles (section 4). This formalism is based on a formal notion of a multidimensional schema as presented in section 3. To emphasize the benefits of our user centric modeling approach, we show how the user profiles can be used for improving the OLAP system performance at runtime by predicting the users next query. An architecture for this is shown in section 5. Finally, section 6 draws conclusions and presents directions for future work.

1.1 Why should we model user behavior?

The purpose of an OLAP system is to satisfy the user's analysis requirements. This should make clear that knowledge about the user query behavior is most important during the whole development cycle of the system (see figure 1).

The main goal of the **'Conceptual Design'** is to consolidate the requirements identified during **'Requirement Analysis'** to a conceptual MD schema. A user centered, use-case driven approach to schema design would increases the quality of the conceptual design. As the static data model (or schema) of an OLAP system is closely related to the dynamic aspect of the system (query behavior of the user), the schema determines which queries the

Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'99)
Heidelberg, Germany, 14. - 15. 6. 1999
(S. Gatziu, M. Jeusfeld, M. Staudt, Y. Vassiliou, eds.)
http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-19/

user can execute. Therefore, the design of the conceptual DW schema should be driven by a model of the user query behavior. Another reason is that our experiences from several real world projects show that end users usually can state very well which analytical task they want to perform, but have difficulties understanding a static data model or schema that is necessary to answer their queries.

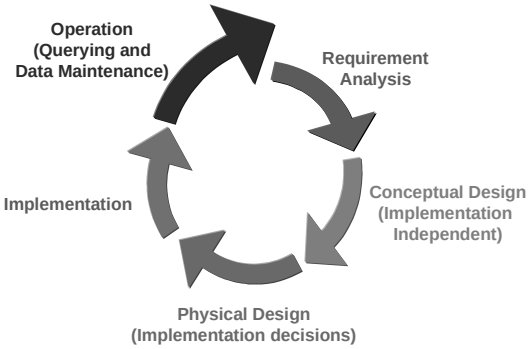

figure 1: The different phases of an iterating design and maintenance cycle

Knowledge about the anticipated user behavior is also a key input for the subsequent **physical data warehouse design**. During this phase, the conceptual model has to be mapped to a specific implementation (e.g. MOLAP or ROLAP). This includes decisions about physical optimization techniques (e.g. indexing schemes, physical clustering, precomputation strategies, denormalization). Comprehensive research of physical optimization strategies for Data Warehouses and OLAP systems has been done for specific implementation strategies (relational e.g. [GHR+97] or multidimensional e.g. [FB99]). Irrespective of the approach, all of these solutions require the definition of a typical workload as a prerequisite. The view selection problem (e.g. [TS98], [SDN98]) is a prominent example that has attracted a lot of research, lately. To our knowledge, no scientific work has so far been devoted on how to find such a workload in an OLAP environment. If the user behavior was modeled during the conceptual design using a formal modeling technique, typical workloads could be generated from this model, automatically.

Knowledge about the user query behavior can also be used at run time (during the **Implementation** and **Operation phase**). User and task profiles that contain information about typical query sequences can be used for runtime system optimization. Due to the interactive and navigational nature of the user's analysis it is promising to predict the next steps of the user. This information can be used by an intelligent OLAP database for optimizing caching and scheduling purposes. An interesting option is the usage of prediction information to speculatively execute possible next queries (this idea is further elaborated in section 5). If the user actually asks such a predicted query the systems answering time is greatly reduced which is one of the main factors of success for a data warehouse.

Summing up, knowledge about anticipated user behavior has an impact on all phases of OLAP system design. Therefore it should be fully incorporated into an OLAP design methodology. In combination with techniques to model the static part of the system (the multidimensional conceptual schema (e.g. [GMR98], [SBHD99]) a technique for modeling user behavior (the dynamic part of an OLAP system) can contribute to a comprehensive modeling technique for multidimensional information systems.

1.2 What is so characteristic about user interaction in OLAP systems?

As an example let us consider an extract of a real project we performed together with an industrial partner. A car manufacturer wants to analyze vehicle repairs to improve the technical quality of his products, to evaluate the warranty policy and to assess the quality of different garages. According to the basic philosophy of OLAP systems the user interactively analyzes the data by applying multidimensional operations. Thus, the typical interaction between the user and the system consists of a sequence of queries. We call all the interactions necessary to answer a business question (e.g. "Which garages offer a potential for cutting down repair costs?") or to verify a hypothesis (e.g. "Cars in southern countries are repaired less often.") a **session**.

The user normally uses a predefined entry point (e.g. a business report). From there the user navigates through the multidimensional data space by performing operations on the previous query result. These results are displayed using an MD view (for an example see figure 2). This view (mostly a table or a graph) visualizes a subspace of the original MD cube. Some dimensions are restricted to a single value (called a slicing operation). We call these dimensions the **selection dimensions** of the query. In the example the values to which the dimensions are restricted are shown in the left hand corner of the table (customers, year). The members of the remaining dimensions are shown in the visualization. In our example they are grouped along the axes (geogr. region, vehicle model). We call these dimensions the **result dimensions** of the query. The data displayed in the MD view are the values of a subset of all measures called the **result measures** of the query (e.g. # of repairs). The next query is formulated by manipulating the structure of the MD view, for example exchanging result and selection dimensions or altering the selection for a selection dimension.

To identify saving potentials a controller may start by accessing a predefined report (i.e. a query) ranking the geographical regions by number of repairs during the year 1998. He then picks a region in which enough repairs occurred to make it worthwhile further investigation. This is a cognitive process which takes some time and does not involve any system interaction. Therefore, from a systems point of view the process is a 'black box' that is not deterministic but predictable under certain circumstances.

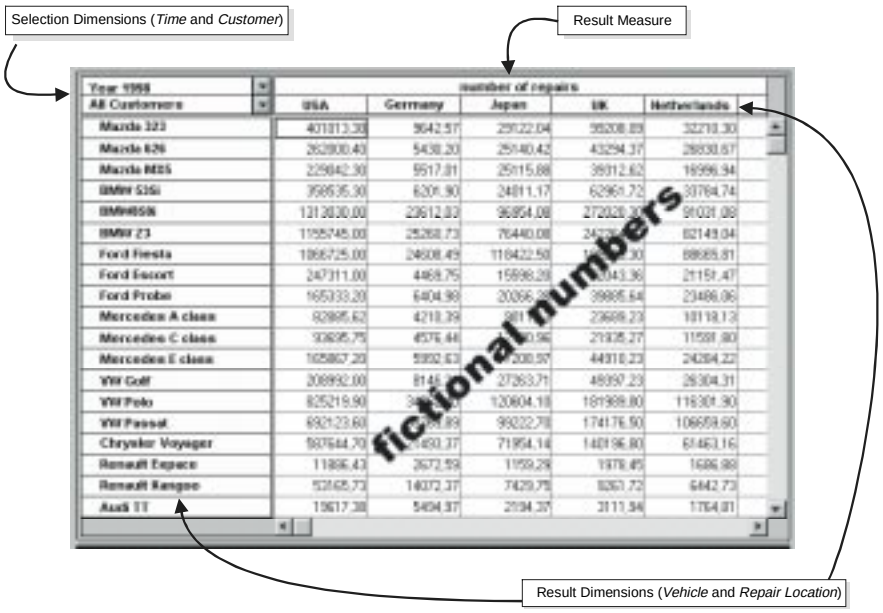

figure 2: typical user interface of an OLAP product (MD view)

The only thing the system can perceive (and use to guess the user's intention) is the users next query.

In our example the analyst, may perform a drill-down operation on the chosen region (e.g. by performing a double click on the region name) resulting in a list of garages in this region. Afterwards, he might be interested in the way how the number of repairs changed over the year. In this case, he executes a drill down on the year resulting in a two-dimensional view containing garages and month. Next, he might identify an interesting garage and might want to see how the number of repairs varies by car model (he performs a rotate operation of the data cube) and so on.

We call the consecutive operations executed during a session the **query sequence** of this session. As the OLAP systems do not restrict the user to a certain workflow, different sessions answering the same business query may have different query sequences as the user can skip steps or include additional steps. Nevertheless, each analytical task (e.g. "find garages which should participate in a promotion") possesses a characteristic interaction pattern. Furthermore, as different user have varying analysis habits, the patterns are also user specific.

2 Related Work

Several approaches have been published to formalize the multidimensional data model (e.g. [CT98a], [Vas98]). The papers present a description of the MD data structure and propose formalizations of single queries using algebraic operations (e.g. [Vas98]), a logical calculus (e.g. [CT98a]) or a graphical formalism ([CT98b]). Our static data model (section 3) is based on this work (esp. [Vas98]). However, the query descriptions in these papers are very operationally oriented as the aim is to provide a description that allows the efficient processing of single queries. Our approach supplements this work by giving a higher level abstraction of queries (needed for conceptual modeling in earlier phases and query pattern recognition) and taking into account the navigational nature of OLAP sessions.

Some work has been done in the field of design methodologies for data warehouses and OLAP systems. [GMR98] proposes a graphical representation of multidimensional schemas (DF model) and a methodology to build such a schema from the E/R models of the operational sources. "Query patterns" can be graphically represented in the model by marking dimension attributes that are of special interest to the user. The authors suggest that these markers can be used to determine sensible levels of preaggregation. The approach does not take into account the correlation between successive queries due to the navigational nature of OLAP applications which is the basic assumption of our work. Furthermore, the approach fundamentally differ from our approach by proposing a 'bottom up' data warehouse design, that means starting from the schemas of the operational sources and deriving a warehouse schema. Our approach is more user centered as we propose to start with a model of the users' analysis requirements independently from the structure of the operational systems.

[GR98] describes a methodology for data warehouse design that uses the DF model ([GMR98]) as its core. The description technique used for single queries in this paper has some similarity to our approach. However, it does not model the distinction between selection and result dimensions, the relationship between consecutive queries, typical query patterns and does not contain a graphical representation for query profiles.

[ABD+99] describes the design of an aggregate cache for OLAP systems. One of the parameters used to decide if a cached aggregate is replaced is its similarity to the last query of the user. Like in our approach, the similarity of queries is measured by the number of operations necessary to navigate between the queries. The presented measurements show that this is a sensible assumption. Our approach extends this work by not only taking into account the last query issued but the whole history of the user's session combined with domain knowledge captured by profiles.

To our knowledge no work exists on the prediction of queries in the domain of OLAP systems and Data Warehouses. Although, this idea has been researched for WWW based information systems. [ZAN99] presents a theoretical framework for using knowledge about user navigation for speculatively presending documents. A practical evaluation shows that significant speed-ups are possible. OLAP and WWW systems share the common characteristic of session oriented navigational data access. To a certain degree, the document structure of the WWW is also present in OLAP systems. Nevertheless, the operations available to the user are more complex in OLAP systems. Furthermore, in an OLAP system more information about the user and its task are available, therefore user/task specific profiles can be built.

3 Multidimensional Data Model

To formulate a formal conceptual model for the user behavior in OLAP systems, we first need a formal conceptual description of the underlying multidimensional paradigm. As mentioned in section 2, several data models have been published recently. Following an evaluation of MD data models ([SBH99]) our formalism is inspired by ([Vas98]).

Definition 3.1 (level): A dimension level (or short level) l is a finite set of elements from a domain *dom(l)*. Ψ denotes the set of all levels for an MD schema ◆

Definition 3.2 (dimension): A dimension d is defined as tuple $d := (H, class)$ with

- $H \subseteq \Psi$ being the set of levels that structure this dimension. Each dimension contains a special level *<dimension_name>.all* which contains only one member and is used to aggregate all the basic values to a single value.
- $class \subseteq H \times H$ defining the classification relationship between levels. $(l_1,l_2) \in class^*$ reads "l_1 can be classified according to l_2." The relation *class* is minimal in the sense that the following property is fulfilled: $(l_1,l_2) \in class \wedge (l_2,l_3) \in class \Rightarrow (l_1,l_3) \notin class$. The transitive, reflexive closure $class^*$ of *class* must fulfill the following property: $(l_1,l_2) \in class^* \Rightarrow (l_2,l_1) \notin class^*$. That means that $class^*$ defines a partial order on H. ◆

Thus, the tuple (H,$class^*$) defines a lattice over the levels.

Definition 3.3 (MD schema): A multidimensional schema is a finite set of dimensions $\Omega = \{d_1,..d_k, M\}$, where M is a special dimension containing the measures of the schema. ◆

The function *dim(l)*: $\Psi \rightarrow \Omega$ maps each level to its corresponding dimension, that means the following condition is fulfilled: $dim(l) = d \Leftrightarrow \exists d=(H_d,class_d) \in \Omega$ such that $l \in H_d$

Definition 3.4 (dimension path): A dimension path *dp* of a dimension D = (H, *class*) is defined as a path in the lattice (H, $class^*$), beginning from its least upper bound and ending in its greatest lower bound. Each dimension path is a linearly ordered list of dimension levels containing the special dimension level *<dimension_ name>.all* as the least upper bound. ◆

We use DP_D to denote the set of all dimension pathes of the dimension D.

For conceptual modeling purposes, multidimensional schemas can be represented graphically (e.g. by using the ME/R model [SBHD99]).

Example: A possible multidimensional schema for our example scenario (figure 3) looks like this:

$\Omega = \{d_{time}, d_{customer}, d_{vehicle}, d_{garage}, M\}$

d_{time}= ({ day, month, year, time.all },
{ (day,month), (month, year), (year, time.all)})

$d_{customer}$=({ customer, customer.all },
{ (customer, customer.all)})

$d_{vehicle}$= ({ vehicle, model, brand, vehicle.all},
{ (vehicle, model), (model, brand),
(brand, vehicle.all) })

d_{garage}= ({ garage, geogr. region, type of garage,
country, garage.all},
{ (garage, geogr. region),
(geogr. region, country), (country, garage.all),
(garage, type of garage),
(type of garage, garage.all)})

M = {measures, ∅} ◆

4 Modeling the User Behavior

The basic idea when modeling query behavior is that the user's queries follow certain patterns. E.g. when evaluating the number of repairs for a given period a user always uses an MD view that shows the number of repairs by geographic regions on one axis and vehicle models on the other axis. The individual query contains different values for the timespan covered (e.g. 1998, or 1999) but the structure of the MD view is unchanged. Thus, we use the structure of the MD view to define the notion of a query prototype (section 4.1).

As we are interested in similar queries to define patterns, we need a formal notion for this similarity. The basic idea is to define the distance between two queries q_1 and q_2 as the number of operations that is needed to transform the query prototype of q_1 to the prototype of q_2 (section 4.2).

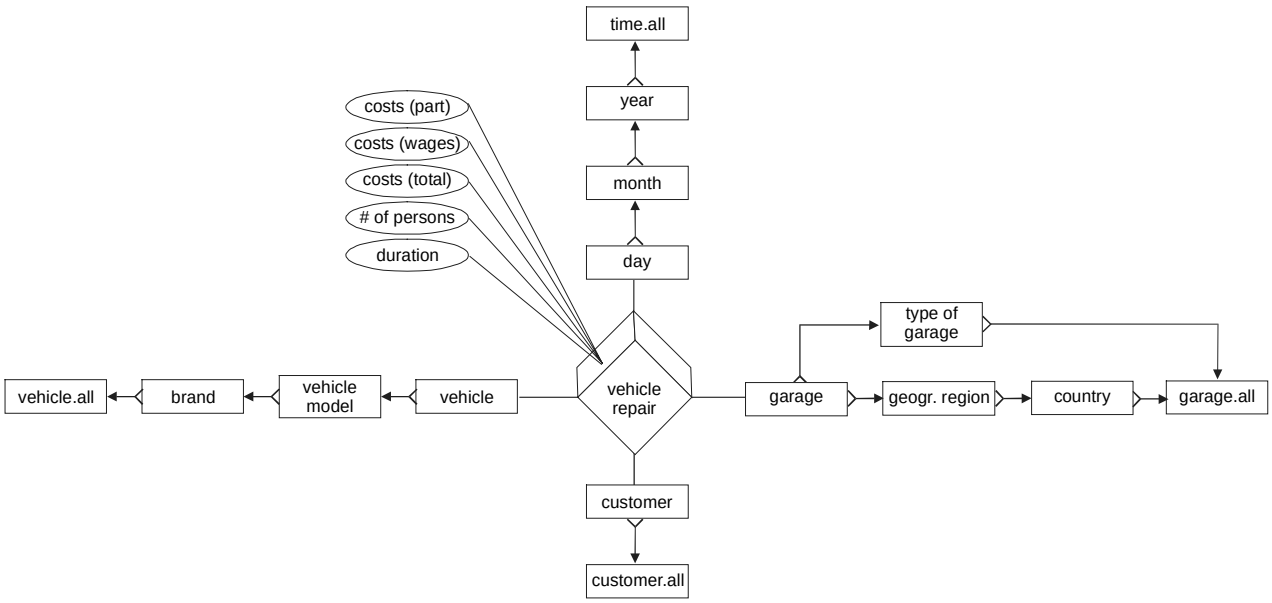

figure 3: A graphical representation of the MD schema of the vehicle repair example using the ME/R notation

Based on this, we can define a user/task profile that describes the typical interaction pattern for this task that may be user specific (section 4.3). These user/task profiles are usually created by a system designer together with domain experts during the requirement engineering and conceptual system design phase (Another possibility is the creation of profiles from database logs. This idea is further elaborated in section 5 and 6). The indispensable involvement of domain experts makes a graphical notation of profiles desirable (section 4.4).

4.1 Sessions, Queries and Query Prototypes

As already stated, the user interaction in OLAP systems is strongly session oriented. A session S consists of a sequence of queries $S := q_1,...,q_n$ where q_{i+1} is formulated applying interactive manipulation operations (like drilling, or rotation) to the results of q_i. The results of a single query $q \in S$ are displayed by the OLAP frontend using an MD view (table, graph etc.). Dimensions being restricted to one element are called **selection dimensions** of q while all other dimensions are called **result dimensions** of q. Let us assume that $d_k \in \Omega$ is a selection dimension of q limited to the single value *val*. The level $l \in \Psi$ to which *val* belongs is called the **selection granularity level** (or short selection level) of d_k for q, i.e. $val \in dom(l)$. For each result dimension d_r of a query q a level of granularity is given that is called the **result granularity level** (or short result level) for dimension d_r.

We are interested in modeling typical interaction patterns in query behavior. These patterns describe the similarities between different sessions that are similar with respect to the intention/task the user had in mind when executing the session. From a system's point of view, the current interest (or intention) of the user is mirrored in the structure of the query he issues. The distinction between selection and result dimensions in the current query contains information about the user's intention. It shows which correlation in the data the user is interested in. For example if the result dimensions are geographic region and time, the user is likely to be analyzing the correlation of these factors irrespective of e.g. the vehicle model.

Thus, we assume that the structure of the MD view (for the query q) contains the relevant information about the state of the user's current task at the moment he executes the query q. This information is described by a query prototype.

Definition 4.1 (Query Prototype): Let us assume a query q that has the result dimensions D_r and the selection dimensions D_s. The query prototype $\wp_q$ for this given query q is defined as a triple $\wp_q := (M_r, L_r, L_s)$ such that

- $\mathbf{M_r} \subseteq M$ is the set of measures referenced in query q as result measures
- $\mathbf{L_r}$ is the set of result granularity levels of the result dimensions D_r,
- $\mathbf{L_s}$ is the set of selection granularity levels of the selection dimensions D_s

The following conditions have to be fulfilled in order for $\wp$ to be a valid query pattern:

(1) $\bigcup_{l \in L_r} \dim(l) \cup \bigcup_{l \in L_s} \dim(l) = \Omega$

(2) $\dim(l_1) \neq \dim(l_2) \quad \forall l_1, l_2 \in (L_r \cup L_s) \quad l_1 \neq l_2$ ◆

Conditions (1) and (2) ensure that exactly one level of each dimension is either a result or a selection level.

Example: The query q_1 = "Give me a ranking of garages according to the number of repairs summed up to the geographic region for 1998" has the following prototype

$\wp_{q1}$ = (M_{q1} = {#of repairs}, D_r = {geogr.region}, D_s = {year, vehicle.all, customer.all}) ◆

Definition 4.2 (Query Space): The set Θ_Ω of all valid query prototypes for a given MD schema Ω is called the **query space** for this schema.

4.2 Similarity of Queries

We are interested in recognizing and formulating patterns of access. Two sessions have a common pattern if they contain similar queries. Thus, to define a pattern we first define the notion of similarity of queries. We approximate the similarity of two queries q_1 and q_2 by defining a distance measure that corresponds to the number of user interactions minimally needed to transform the query prototype of query q_1 to the prototype of q_2. A short distance means a great degree of similarity.

Assuming a query q with prototype $\wp_q = (M_q, L_r = \{r_1,\ldots,r_k\}, L_s = \{s_1,\ldots,s_m\})$, the following basic frontend interactions result in a query q' with prototype $\wp_{q'}$:

drill down/roll up: The user can increase (drill down) or decrease (roll up) the level of detail for a selection or result dimensions. For a *selection dimension* $\wp_{q'} = (M_q, L_r, L_s')$ with $L_s' = (s_1, \ldots, s_i', \ldots, s_m)$. Where s_i' has to fulfill the following condition:

$$\exists d=(H,class_d)\in\Omega: (s_i,s_i')\in class_d \vee (s_i',s_i)\in class_d$$

In case of a *result dimension*: $\wp_{q'} = (M_q, L_r', L_s)$ with $L_r' = (r_1, \ldots, r_i', \ldots, r_k)$ and

$$\exists d=(H,class_d)\in\Omega: (r_i,r_i')\in class_d \vee (r_i',r_i)\in class_d$$

rotation: the user can convert a result dimension to a selection dimension and vice versa (e.g. by a drag and drop mechanism). For result to selection dimension $\wp_{q'} = (M_q, L_r', L_s')$ with $L_s' = L_s \cup l$ and $L_r' = L_r - l$; $l \in L_r$.

For a selection to result dimension $\wp_{q'} = (M_q, L_r', L_s')$ with $L_r' = L_r \cup l$ and $L_s' = L_s - l$; $l \in L_s$

focus change: the user can add or delete a measure m from the MD view. If a measure is added $\wp_{q'} = (M_q', L_r, L_s)$ with $M_q' = M_q \cup \{m\}$; $m \in M$. If a measure is deleted $\wp_{q'} = (M_q', L_r, L_s)$ with $M_q' = M_q - \{m\}$; $m \in M$

Similar queries show that the user's analysis intention when executing them is similar. The change of intention differs for different types of operations. For example a rotate shows that the user is now interested in another combination of parameters what might indicate that the analysis task he is working on has evolved. Therefore, it seems sensible to assign a higher weight to rotate operations than to drilling operations when computing the distance. Thus, the weighted distance between two queries is informally defined as the weighted number of drilling operations plus the weighted number of rotations plus the weighted number of focus changes.

To determine how many drilling/rolling operations are necessary to navigate from one level to another level (drilling distance), we define the drilling graph as follows.

Definition 4.3 (drilling graph): The drilling graph for a given MD schema Ω with a set of corresponding levels Ψ is defined as a tuple $\mathcal{L}=(\Psi, E)$ where $E \subseteq \Psi\times\Psi$ is constructed using the following condition:

$$(l_1,l_2)\in E :\Leftrightarrow \exists d=(H,class_d)\in\Omega: \quad (l_1,l_2) \in class_d \vee (l_2,l_1) \in class_d$$

Intuitively, the drilling graph contains all the levels of the schema as nodes and the classification relationships of the different dimensions as edges. The edges of the drilling graph are undirected as drilling and rolling operations should not be distinguished.

Definition 4.4 (drilling distance of two levels): The drilling distance $|l_1\text{-}l_2|$ between two levels l_1 and l_2 is defined as the minimum length of the path in the drilling graph $\mathcal{L}$.

Definition 4.5 (distance between two query prototypes): We extend the definition of the function *dim*: $dim: 2^\Psi \to 2^\Omega$ maps a set of levels $L \subseteq \Psi$ to their corresponding dimensions.

$$\dim(L) := \bigcup_{l\in L} \dim(l)$$

Furthermore for two sets of levels $A,B \subseteq \Psi$ let $\chi(A,B)$ denote the corresponding level pairs that share the same dimension.

$$\chi(A,B) := \{(a,b)\in A\times B \mid \dim(a) = \dim(b)\}$$

The distance $|\wp_q - \wp_{q'}|$ between two query prototypes $\wp_q = (M_q, L_r, L_s)$ and $\wp_{q'} = (M_q, L_r, L_s)$ is defined as follows:

$$|\wp_q - \wp_{q'}| := \Delta drill + \Delta rotate + \Delta focus$$

with

$$\Delta drill = \sum_{(l,l')\in\chi(Lr\cup Ls, Lr'\cup Ls')} |l - l'|$$

$$\Delta rotate = \left|\left(\dim(L_r') \cup \dim(L_r)\right) - \left(\dim(L_r') \cap \dim(L_r)\right)\right|$$

$$\Delta focus = \left|M' \cup M - \left(M' \cap M\right)\right|$$

Assuming that $w_d, w_r, w_f \in \mathbb{R}$ are the weights for the different kinds of operations (e.g. drilling) The weighted distance $|\wp q - \wp q'|_w$ between two query prototypes is defined as:

$$|\wp_q - \wp_{q'}|_w := w_d \cdot \Delta drill + w_r \Delta rotate + w_f \Delta focus\,. \quad \blacklozenge$$

Definition 4.6 (distance between two queries): The (weighted) distance $|q_1\text{-}q_2|$ between two query q_1 and q_2 is defined as the (weighted) distance of their prototypes $|\wp q_1 - \wp q_2|$. ◆

Example: Let us compare query q_1 *"Give me a ranking of garages according to the number of repairs summed up to the geographic region for 1998"* with prototype

$\wp_{q1}$ = ({#of repairs}, {geogr.region},
{year, vehicle.all, customer.all})

and the query q_2 = *"Give me the number of repairs detailed by vehicle model and month for the garage 'XYZ'"* with

$\wp_{q2}$ = ({#of repairs}, {vehicle model, month},

{garage, customer.all}).

The following numbers of operation have to be performed: Δdrill=2, Δrotate=3, Δfocus=0. Thus, assuming the following weights $w_r=2, w_d = w_f = 1$ theses two queries have a weighted distance of

$$|q_1-q_2|_w = 2\cdot3+1\cdot2+1\cdot0 = 8.$$

The query q_3 = *"Give me the number of repairs for different garages during the year 1998"* has the protoype

$\wp_{q3}$= ({#of repairs}, {garage},
{year, vehicle.all, customer.all}).

Its weighted distance to query1 is $|q_1-q_3|_w=2\cdot0+1\cdot1+1\cdot0=1$ and its weighted distance to query2 is $|q_2-q_3|_w =2\cdot3+1\cdot1+1\cdot0 =7$. Therefore, query3 is more similar to query1 that to query2. ◆

4.3 User/Task Profiles

The previous section showed how to model the structure of individual sessions. If a user solves an analysis task, each of his sessions may be different as the user is free to skip steps or include additional steps. Nevertheless, each analysis task (e.g. "find garages which should participate in a promotion") possesses a characteristic interaction pattern.

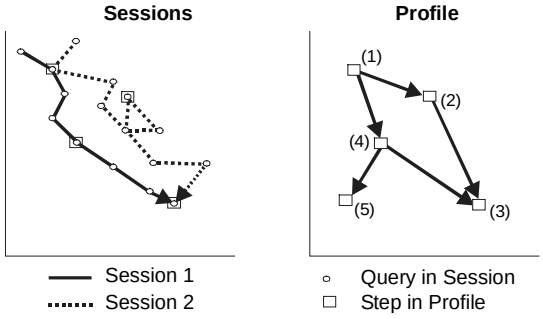

figure 4: an abstract visualization of two sessions(left) and a corresponding profile (right)

This section deals with the definition of these typical patterns (so-called user/task profiles). A session is a linear sequence of queries and two subsequent queries have a distance of 1. A user/task profile allows alternative navigation pathes and does only contain prototypes of queries that correspond to significant steps of the analysis process. Thus, a single profile is an abstraction of a set of sessions. The basic idea for the distinction between sessions and profiles is shown in figure 4. On the left two distinct sessions are visualized using an abstract two dimensional projection of the query space. A profile is shown on the right containing only queries that are significant to the analysis task.

Definition 4.5 (user/task profile): A user/task profile for a query space Θ_Ω is a labeled graph (V, E)**,** where

- $V \subseteq \Theta_\Omega$ is a finite set of query prototypes (called steps of the profile).
- $E \subseteq V\times V$ is the set of directed edges connecting query prototypes. An edge from query prototype $\wp_{q1}$ to $\wp_{q2}$ means that sessions typically contain a query q_1 (with prototype $\wp_{q1}$) before a query q_2 (with prototype $\wp_{q2}$). Notably, q_1 and q_2 can have an arbitrary distance.

To determine if a complete session 'matches' a given profile, we define the notion of instances of profiles. ◆

Definition 4.6 (Instance of a user/task profile): A session $S := q_1,...,q_n$ is called an instance of a profile P = (V,E) if there exists a profile path $v_1,...,v_m$ in P and a mapping $I: \mathbb{N}\rightarrow\mathbb{N}$ such that

(1) $\wp(q_{I(i)}) = v_i \ \forall 1\le i \le m$

(2) $i > j \Leftrightarrow I(i) > I(j) \ \forall 1\le i,j \le m$ ◆

Condition (1) formalizes the fact that for all of the steps of the profile path v_i the session S contains a query q with index I(i) that has the prototype v_i. Condition (2) ensures that the queries in S occur in same order as in the profile path.

The sessions shown in figure 4 are instances of the profile on the right. Session 2 contains the steps (1), (2), (3) of the profile and session 1 contains the steps (1), (4), (3). Other instances might also contain the sequence (1), (4), (5)

4.4 Graphical representation

As user and task profiles are modeled during the conceptual modeling phase together with the users of the system, an intuitive graphical representation is necessary. For this purpose we use an interaction graph which is a graphical representation of a task/user profile. Each node of this graph corresponds to a query prototype which is a mirror of the user current state of intention.

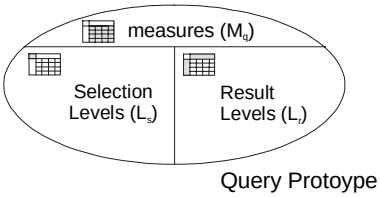

figure 5: graphical representation of a query prototype

It is depicted by an oval which contains three sections (see figure 5). The set of result measures is contained in the top part of the oval; selection levels are enumerated to left and result levels are shown on the right. Furthermore each section of the oval has an icon that depicts the corresponding category. The icon shows an abstraction of a table (as an example for an MD view) with the corresponding area being gray.

The edges of the profile correspond to the navigation of the user. Albeit, following one edge in the profile might include several individual queries steps for a session that is an instance of the profile.

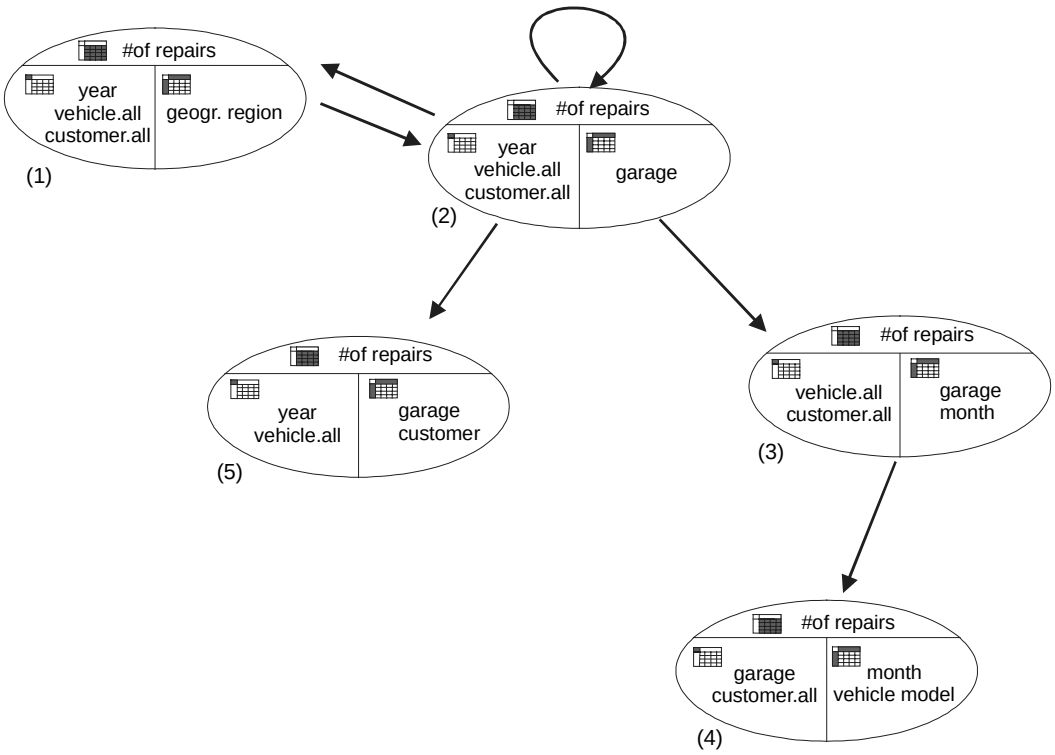

figure 6: the graphical representation of a query prototype

Figure 6 shows a sample profile that models the analysis task "find candidates for a new promotion" described in section 1.2. The node (1) shows the predefined report which serves as a starting point for this task. The navigation path (1)-(4) corresponds to the workflow described in the example. The alternative path from (2) to (5) models the possibility that the user decides to find the most promising garages for a promotion on the basis of the customer base of this garage. Therefore, he analyses the correlation between garages and customers for the current year. As shown with nodes (1) and (2) 'backward navigation' is also an option. This models the fact that the user first analyses the figures for regions then drilling to see the garages for one region but returning to the region view if the results are not satisfactory. A cycle occurs if the user executes queries that have the same prototype, i.e. queries of the same structure with different selection criteria (e.g. first analyzing 1997 figures the analyzing 1998 figures).

5 An Architecture for the Prediction of User Behavior

If typical interaction patterns (task and/or user specific) are known, this information can be used to build query profiles for users or user groups. This information can be used to optimize the performance of an OLAP system at runtime. The idea is to use the profiles together with the know prefix of the query session to predict which queries the user is likely to ask during the rest of the session. Thus these queries or parts of the results can already be computed while the user is busy formulating his next query. In this section we sketch an architecture that allows for extending an existing OLAP system with prediction functionality.

The component architecture of such an environment is show in figure 7. This abstract architecture applies to all OLAP systems irrespective of the storage strategy (MOLAP, ROLAP etc.). The white elements are part of the traditional OLAP system: an *MD user interface* which is used by the analyst to formulate the query and which visualizes the results. Via an interface (e.g. OLEDB for OLAP, see [Mic98]), the queries are passed to an MD query processor that performs optimization and maps the query to the physical storage format of the *MD data store* that contains a persistent image of the data. The query processor stores information about the executed queries in a query log. This abstract architecture applies to all OLAP systems irrespective of the storage strategy (MOLAP, ROLAP etc.).

The gray components are added to the system in order to provide the prediction functionality. The *profile information* stores the user/task specific profiles (using the formalism presented in section 4). These patterns are initially constructed during the conceptual *user modeling* process as described earlier in this paper. Another interesting alternative is to build, validate and adapt these patterns by analyzing the *query logs*. This task is carried out by the *profile builder* which uses data mining/pattern recognition techniques to derive new profiles or adapt existing profiles. The core of the architecture is a prediction unit, which is located between the query processor and the frontend tool. It has the same interface to the user inter-

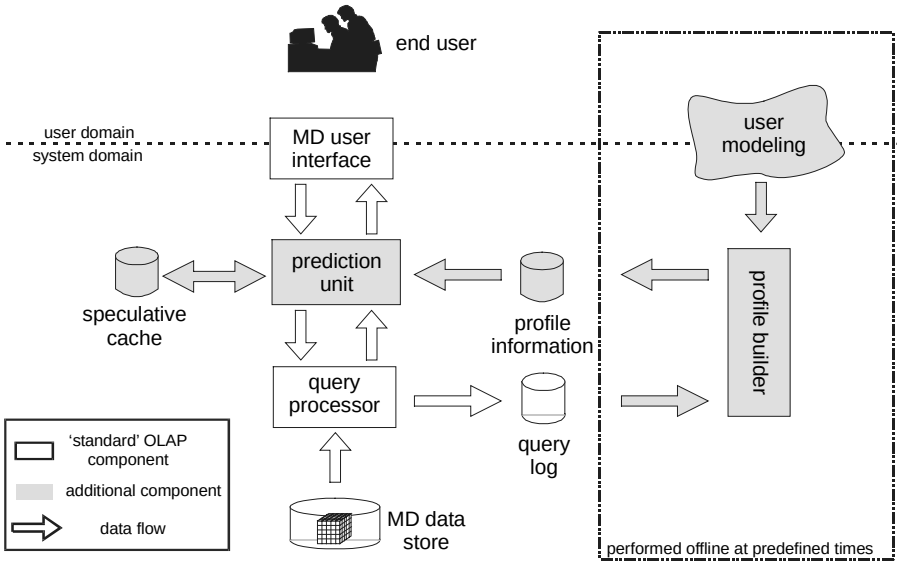

figure 7:the component architecture of a predictive OLAP system

face as the query processor (e.g. MDX) and predicts the possible next queries of the user from the status of the current session (the queries asked so far) and the profile information. Following a predefined strategy it passed predicted queries to the query processor for speculative execution. The results are stored in a *speculative result cache*.

In figure 8 the dataflow and the processes inside the prediction unit are shown (the parts where the formalism presented in this paper is used are marked gray). The frontend passes a new query to the request handler. The handler checks if this query can be answered from the speculative result cache, i.e. has been correctly predicted. If not, the query is passed to the query scheduler which is responsible for passing the query on to the OLAP query processor. In any case the session state for the current session is updated (i.e. the current query prototype is added to the session prefix). This information together with the profile information is used by the *'prediction model'*-process which generates queries that are most likely to be asked next and estimates their probability. In this process it makes use of the distance measure defined in section 4 as an approximation of similarity.

A cost model process passes the estimated database execution costs of the queries to a decision model process which decides if the query should be executed speculatively. In this case it passes the query to the query scheduler with a flag that the results of this query should be stored in the cache instead of being sent back to the user immediately. When the query scheduler receives a new query for execution, it first checks if such a query is already being executed at the moment. In this case the query is not passed on but answered using the results of the running query. This case can occur if a query that is being speculatively executed is actually asked by the user before the execution is finished. The results of all queries are either passed to the user or stored in the speculative result cache. The cache uses an appropriate replacement strategy (e.g. a time stamp method).

6 Conclusions and Future Work

The starting point of our research work was the observation that capturing knowledge about the query behavior of OLAP users is beneficial for the conceptual and physical design of such a system but also can improve the systems runtime performance. An analysis of the characteristics of OLAP query behavior showed that user access is typically multidimensional, navigational, explorative, session oriented with task and user specific patterns.

Therefore, we proposed a formal framework to model knowledge about user behavior taking into account these characteristics. We also presented a graphical notation to visualize the models and make them usable during conceptual design. To show the benefits of such an approach we exemplarily focused on the deployment of the modeled user/task profiles for increasing the system performance using prediction techniques.

The topic of our current and future work will be the integration of the presented notation into our conceptual modeling methodology and the validation of this modeling technique in several real world projects. For this purpose a modeling tool will extended to support the notation. topics are the relationship between static and dynamic model.

The architecture proposed in the previous section raises a lot of interesting research problems. It is necessary to develop a formal prediction model in order to implement the prediction algorithm. Another interesting point is the

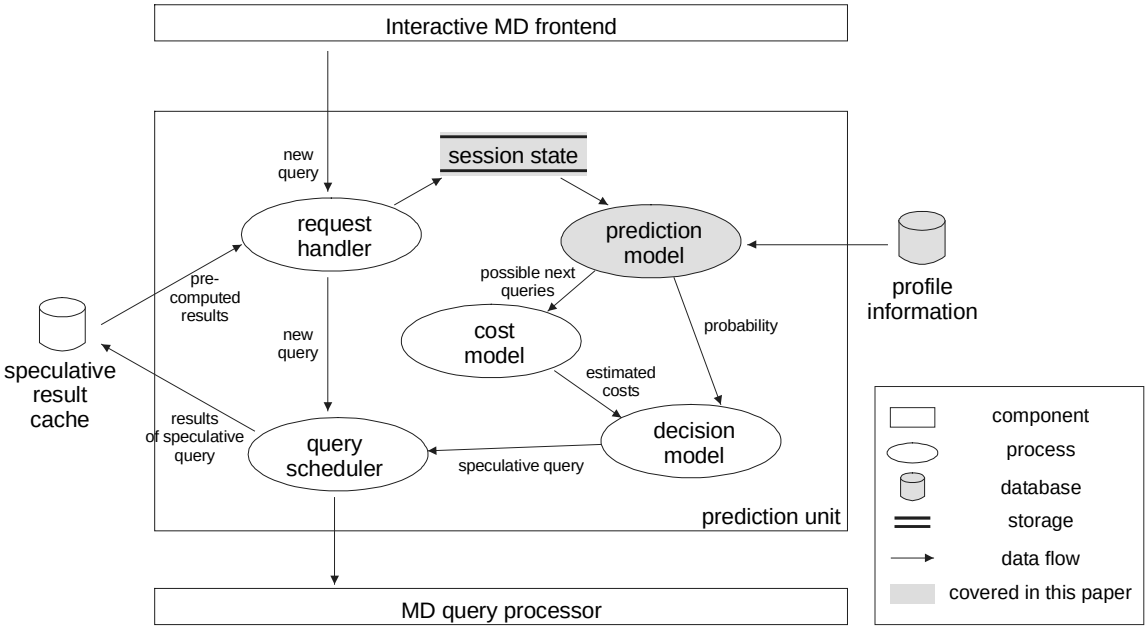

figure 8: A dataflow diagram of the prediction unit

derivation of user/task profiles from the saved interaction logs using data mining and pattern recognition techniques. The question of how to combine the manually modeled user profiles and the automatically generated ones immediately arises. Physical implementation issues are certainly worth further studies (e.g. how to organize the speculative result cache and which caching strategies to use).

Our plans are to implement the above architecture on top of a commercial OLAP system and to evaluate the prediction algorithm using real world interaction logs.

Until now we only focused on using the profile information outside the kernel of the OLAP system. Further potential optimization become possible when integrating the prediction functionality into the kernel of the OLAP query processing system (e.g. special caching techniques can make use of the prediction results).

Acknowledgments

I want to express my gratitude to my colleagues Markus Blaschka, Gabriele Höfling and Wolfgang Wohner for reading through several versions of this paper and discussing this work with me. Their most valuable comments played a central role during the development of this paper.

References

[ABD+99] J. Albrecht, A. Bauer, O. Deyerling, H. Günzel, W. Hümmer, W. Lehner, L. Schlesinger: *Management of Multidimensional Aggregates for Efficient Online Analytical Processing*. Proc. of the IDEAS 1999.

[CT98a] L. Cabibbo, R. Torlone: *A Logical Approach to Multidimensional Databases*. Proceedings of the EDBT 1998.

[CT98b] L. Cabibbo, R. Torlone. *From a Procedural to a Visual Query Language for OLAP*. In 10th IEEE International Conference on Scientific and Statistical Database Management (SSDBM-98), Capri, Italy

[DSBH98] B. Dinter, C. Sapia, G. Höfling, M. Blaschka: *The OLAP Market: State of the Art and Research Issues*, Proc.of First International Workshop on Data Warehousing and OLAP (DOLAP), 1998, Washington, D.C., USA..

[FB99] P. Furtado, P. Baumann: *Storage of Multidimensional Arrays based on Arbitrary Tiling*, Proc. of the 15th International Conference on Data Engineering (ICDE), Sydney 1999

[GHR+97] H. Gupta, V. Harinarayan, A. Rajaraman, J.D. Ullman: *Index Selection for OLAP*. Proceedings of the International Conference on Data Engineering (ICDE)1997

[GMR98] M. Golfarelli, S. Rizzi, *Conceptual design of data warehouses from E/R schemes*, Proc. 31st Hawaii Intl. Conf. on System Sciences, 1998.

[GR98] M. Golfarelli, S. Rizzi, *A Methodological Framework for Data Warehouse Design*, ACM DOLAP Workshop, Washington 1998

[Mic98] Microsoft Corp., *OLEDB for OLAP Specification 1.0*

[SBHD99] C. Sapia, M. Blaschka, G. Höfling, B. Dinter, *Extending the E/R Model for the Multidimensional Paradigm*, in "Advances in Database Technologies", Springer LNCS 1552, 1999.

[SBH99] C. Sapia, M. Blaschka, G. Höfling, *An Overview of Multidimensional Data Models*, FORWISS Report FR-1999-001, http://www.forwiss.tu-muenchen.de/~system42.

[SDN98] A. Shukla, P. Deshpande, J.F. Naughton: *Materialized View Selection for Multidimensional Datasets*, Proceedings of the VLDB 1998, Athens, Greece.

[TS98] D. Theodoratos, T. Sellis: *Data Warehouse Schema and instance Design*, in Conceptual Modeling – ER'98, Springer LNCS Vol. 1507.

[Vas98] P. Vassiliadis: *Modeling Multidimensional Databases, Cubes and Cube Operations*. Conference on Statistical and Scientific Databases (SSDBM) 1998.

[ZAN99] Zukerman, I., Albrecht, D., and Nicholson, A., *Predicting Users' Requests on the WWW*. In Proc. of the 7th International Conference on User Modeling, Springer-Verlag 1999.