1

OLAP query personalisation and recommendation: an introduction

Patrick Marcel

Université François Rabelais Tours, Laboratoire d'Informatique, France
patrick.marcel@univ-tours.fr

Summary. The aim of this lecture is to present how popular user-centric techniques, namely personalisation and recommendation, can be adapted to an OLAP context. The presentation begins with an overview of query personalisation and query recommendation in relational databases. Then it introduces the approaches proposed for personalising OLAP queries with user preferences, and the approaches proposed for recommending OLAP queries. All the approaches are characterized in terms of formulation effort, prescriptiveness, proactiveness, expressiveness, and in terms of the data leveraged: the current state of the database, its history, or external information.

Key words: OLAP queries, preferences, query personalisation, collaborative filtering, recommender systems, query recommendation

1.1 Introduction

According to a recent article of The Economist, mankind created 150 exabytes (billion gigabytes) of data in 2005. In 2010, it will create 1,200 exabytes [3]. With such amounts of data, it is of paramount importance to be able to access relevant information efficiently, using sophisticated search tools. On the other hand, systems offering search facilities, like DBMSs, are quite uneasy to use, driving querying or navigation into huge amount of data a very tedious process. Those such facilities should be more user-friendly [26].

In domains like the Web, Information Retrieval or e-commerce, user-centric approaches like personalisation or recommendation have been proved successful (see e.g., [15]). It is for instance believed that Amazon makes around 30% of sales thanks to recommendations. Such approaches are very relevant in a database context. For instance, the user may not accept to spend too much time conceiving the query. In addition, she may not be happy if the query's answer shows too many or too few results. And, even if the size of the answer is acceptable, she may be relieved to see the system automatically suggesting

queries that will display other answers of interest, especially if she is left with the task of navigating the database to analyse the data it contains.

A typical example of such an analysis is that of a data warehouse navigated by decision makers using OLAP queries [44]. A data warehouse can be seen as a large database with a particular topology, shared by many analysts who have various interests and viewpoints, where data is seen as a cube, explored by sequences of OLAP queries that may return large answers. In such a context, being able to personalize or recommend queries is seen as particularly relevant [40].

This lecture introduces how personalisation and recommendation approaches have been adopted by the database community, and how they can be adapted to these particular databases that are data warehouses. More precisely, it will try to answer the following two questions:

- Given a database query q, how to to cope with too many/too few results? Personalisation can be used to answer this question. If the query result is too large then being able to add preferences to this query gives a way of ranking the query results to focus on the most relevant first. On the other hand, if the query result is too small, then selection predicates, also called strong constraints, could be turned into preferences (or soft constraints) to weaken this query.
- Given a sequence of queries over a database, how to suggest queries to pursue the session? In this case, what the user did in the past, or alternatively, what similar users did in the past, can serve as a basis for recommending relevant queries to complement the current query answer.

The lecture is organised as follows. The next section presents basic definitions and concepts of preferences and recommender systems. Section 1.3 delimits the scope of this lecture and precise how the approaches surveyed will be categorised. Section 1.4 and Section 1.5 introduce the existing approaches in relational databases and multidimensional databases, respectively. Section 1.6 concludes the lecture with a brief discussion and presents some open issues.

Note that this lecture assumes basic knowledge of relational database [4] and data warehousing [23]. Part of the material that focuses on query recommendation is borrowed from [35]. The slides illustrating this lecture are available on the author's web page [34].

1.2 Preliminaries

In this section, we provide the basic definitions underlying personalisation and recommendation.

1.2.1 Preference expression

We begin by explaining the basics of preference modelling. A more comprehensive introduction can be found in [1].

Qualitative and quantitative preferences

Two types of approaches are used to express preferences. Qualitative approaches express relative preferences i.e., "I like a better than b". Such a preference is noted $a > b$ where $>$ is usually a Strict Partial Order (SPO). An SPO is a binary relation $>$ over a set O which is

- Irreflexive, i.e., for all $a \in O$, $\neg(a > a)$
- Asymmetric, i.e., for all $a, b \in O$, if $(a \neq b)$ and $(a > b)$ then $\neg(b > a)$
- Transitive, i.e., for all $a, b, c \in O$, if $(a > b)$ and $(b > c)$ then $(a > c)$

Given a preference relation $>$, the indifference relation $\sim$ is defined by: $(a \sim b)$ if $\neg(a > b)$ and $\neg(b > a)$. It expresses that a and b are not comparable. Particular partial orders of interest are Total Orders (TO) and Weak Orders (WO). A relation $>$ is a TO if for every a and b, either $(a > b)$ or $(b > a)$. $>$ is a WO if $>$ is a SPO and $\sim$ is transitive.

Example 1. Consider the following database instance:

Movies	Author	Genre	Price	Duration
t1	Cohen	Comedy	5	90
t2	Cohen	Comedy	6	100
t3	Cohen	Comedy	7	80
t4	Allen	Drama	7	120
t5	Lynch	Drama	5	150

The preference "I prefer Lynch movies over Allen movies and Allen movies over Cohen movies" entails that tuple t5 is preferred to tuple t4 and tuple t4 is preferred to tuples t1, t2, t3. As preferences are assumed to be transitive, we say that t5 dominates all the other tuples. Note that this preference says nothing e.g., for t1 and t2, neither for t1 and t3.

Quantitative approaches express absolute preferences, i.e., I (do not) like a to a specific degree. They are based on Scoring / Utility Functions. I like a better than b is noted $u(a) > u(b)$ where u is a scoring function.

Quantitative approaches are often said to be less general than qualitative approaches in the sense that, in order to be representable using scoring functions, a preference relation has to be a WO, which implies that the corresponding indifference relation has to be transitive. But on the other hand, only scoring functions can express the intensity of preferences.

Example 2. The preference "I prefer Lynch movies over Allen movies and Allen movies over Cohen movies" can be expressed by the following scoring function (assuming that scores range from 0 to 1):

- "I like Lynch" corresponds to a score of 0.9
- "I like Allen" corresponds to a score of 0.8
- "I like Cohen" corresponds to a score of 0.5

It can easily be seen that for instance, preference "I prefer cheaper movies, given that author and genre are the same" (i.e., t1 is preferred to both t2 and t3, but it is not preferred to t4 or t5) cannot be expressed with scoring functions.

Preference composition

Preferences can be defined extensionally under the form relation instances, or intentionally. In the latter case, the intentional definition is called a model of preference. Models of preferences can be expressed with a given language like those of [31, 16] and/or by using composition of preference relations. Compositions follow the approach used to express preferences and thus can be qualitative or quantitative.

Qualitative composition

In what follows, let T be a set of tuples and $>_1$ and $>_2$ be two preference relations over T. We restrict here to single dimensional composition, where preferences are expressed over a single relation. Common single dimensional composition includes Boolean Composition, Prioritized Composition, and Pareto Composition.

Boolean composition involves a boolean operator, for instance:

- Intersection, i.e., $R = (>_1 \cap >_2)$ with $(t\ R\ t')$ if $(t >_1 t')$ and $(t >_2 t')$
- Union, i.e., $R = (>_1 \cup >_2)$ with $(t\ R\ t')$ if $(t >_1 t')$ or $(t >_2 t')$

Prioritized Composition imposes a priority of a preference over another. It is formally defined by $R = (>_1 \lhd >_2)$ with $(t\ R\ t')$ if $(t >_1 t')$ or $(\neg(t' >_1 t)$ and $(t >_2 t'))$

Pareto Composition assumes two preferences to be equally important. It is formally defined by: $R = (>_1 \otimes >_2)$ with $(t\ R\ t')$ if $((t >_1 t')$ and $(t >_2 t'$ or $t \sim_2 t'))$ or $((t >_2 t')$ and $(t >_1 t'$ or $t \sim_1 t'))$.

Example 3. Consider the two preferences: "I prefer Lynch movies over Allen movies and Allen movies over Cohen movies", called $P1$ and "I also prefer shorter movies" called $P2$. Composing them using intersection result in a particular SPO with no domination, that can be interpreted as everything is preferred since all tuples are undominated. Composing them with union violates irreflexivity and asymmetry, and thus the resulting relation is usually considered as not being a preference relation. Indeed, in this case, the resulting relation would mean for instance that both t5 should be preferred to t4 (according to $P1$) and t4 should be preferred to t5 (according to $P2$). Composing them with prioritisation results in a total order reflecting $P1$ first and then $P2$ only when $P2$ does not contradict $P1$. More precisely, we have t5 preferred to t4 preferred to t3 preferred to t1 preferred to t2. Finally, composing them using Pareto results in a preference relation where only t3 dominates

t1 and t1 dominates t2, and neither t4 dominates t5 nor t5 dominates t4 since $P1$ and $P2$ do not agree for these two tuples.

Note that properties preservation (irreflexivity, etc.) under various kind of composition operators has been deeply studied (see [1] for more details).

Quantitative composition

Quantitative composition is generally achieved through dedicated functions, like weighted functions, min, max, etc. An example of quantitative composition is, given preferences $P1$ modelled with $score_{P1}$ and preference $P2$ modelled with $score_{P2}$: $Score_{f(P1,P2)}(ti) = x \times score_{P1}(ti) + (1-x) \times score_{P2}(ti)$ where x is some weight.

Example 4. Consider the following preferences: "I prefer Lynch movies over Allen movies and Allen movies over Cohen movies" (P1) expressed by:

- "I like Lynch" with $score_{P1} = 0.9$
- "I like Allen" with $score_{P1} = 0.8$
- "I like Cohen" with $score_{P1} = 0.5$

and "I also prefer shorter movies" (P2) expressed by:

- "I like (duration=80)" with $score_{P2} = 1$
- "I like (duration=90)" with $score_{P2} = 0.9$
- "I like (duration=150)" with $score_{P2} = 0.6$

Suppose we use the scoring function defined above to compose P1 and P2, with $x = 0.5$. Then, for instance, the score of tuple $t1$ would be: $0.5 \times 0.5 + 0.5 \times 0.9 = 0.7$

1.2.2 Recommender systems

In this section, we briefly introduce the basics of recommender systems (see [5] for a more substantial presentation).

Basic formulation

A recommender system is typically modelled as follows. Let I be a set of items (e.g., products in a typical e-commerce application) and U be a set of users (e.g., customers in a typical e-commerce application). Let f be an utility function with signature $U \times I \rightarrow R$ for some totally ordered set R (typically reals between 0 and 1). Recommending s' to u is to choose for the user u the item s' that maximizes the user's utility, i.e., $s' = argmax_I f(u, i)$. The function f can be represented as a matrix $M = U \times I$, that records for any user u in U, any item i in I, the utility of i for u, that is $f(u, i)$. The problem of recommending items to users is that this matrix is both very large

and very sparse. Thus, many methods have been proposed for estimating the missing ratings. Moreover, achieving relevant user-specific recommendations is particularly difficult since it has been observed that everyone is a bit eccentric [21].

In general, recommendation methods are categorized [5] into: (i) Content-based, that recommend items to the user u similar to previous items highly rated by u, (ii) Collaborative, that consider users similar (i.e. having similar profiles) to the one for which recommendations are to be computed as a basis for estimating its ratings and (iii) Hybrid, that combine content-based and collaborative ones. Note that [6] propose a multidimensional generalisation of this basic two-dimensional formulation, especially to support profiling and contextualisation.

Content-based recommendation

Typical content-based recommendation is based on the comparison between item profiles and user profiles. For instance, it can rely on the following principle:

1. Build item profiles by using selected features and providing a score for each feature.
2. Build user profiles from highly rated item profiles, typically by computing a weighted average of item profiles.
3. Compare user profiles with non-rated item profiles to estimate the missing ratings. Typical similarity measures include vector-based similarities like cosine.
4. Recommend to the user those non-rated items achieving the best similarity scores.

Example 5. Consider the following matrix recording ratings:

	Donuts	Duff	Apple	Tofu	Water	Bud	Ribs
Homer	0.9	0.8				0.7	
Marge			0.8		0.6		
Bart	0.7	0.6	0.1				0.8
Lisa	0.2			0.8	0.6		
Maggie	0.6			0.5	0.6		

Suppose that the features chosen for the profiles are: (contains sugar, ok for a diet). Item profiles are modelled as vectors recording a score for these two features. Suppose here that the scores are automatically computed from the items' nutrition facts. For instance, the profile for Donuts is $(0.9, 0)$ and the profile for Apple is $(0.4, 0.6)$. Then user profiles are also modelled as vectors recording scores for the same two features, derived from the known ratings. For instance, Homer profile would be: $(0.9 \times (0.9, 0) + 0.8 \times (0.6, 0.1) + 0.7 \times$

$(0.6, 0.1))/3 = (0.8, 0.1)$[1]. Lisa profile would be: $(0.3, 0.8)$. The similarity score for the user profile with the item profile estimates the missing ratings.

The limitations of content-based approaches are the following: First finding a good set of features must be done very carefully since it impacts directly the score estimates and hence the quality of the recommendation. Another problem is that recommendations stick to the user profile. For instance, using the example above, Homer will never be recommended Tofu. Finally, this approach suffers from the cold-start problem, i.e., how to build a profile for a new user for who no ratings are known.

Collaborative recommendation

The main idea of collaborative approaches is to benefit from all users' ratings when estimating a user's missing ratings. Depending on how the matrix is used (row-wise or column-wise), two techniques are distinguished, that are based on computing similarities among users or items:

- User-user collaborative approaches estimate the ratings for items based on ratings of similar users.
- Item-item collaborative approaches estimate the ratings for items based on ratings for similar items.

Example 6. To illustrate the user-user approach, suppose that all user are modelled as vectors having as many components as there exists items, the value of the component being the rating for the item, or 0 if the rating is not known. For instance, Homer would be modelled as the following vector: (0.9,0.8,0,0,0,0.7,0). Similarity is computed between users, with cosine for instance, to find the users who are the most similar to the one for which the ratings are to be estimated. These users' ratings are derived to estimate the user's missing ratings. For instance, suppose that Bart is found the most similar to Homer, with a similarity score of 0.8. Then, given that Bart has a score for Ribs and Homer has not, Bart's score is used to estimate Homer's, by weighting Bart's score with the similarity between Bart and Homer, i.e., 0.8×0.8 is our example.

The limitations of the collaborative approach are that it relies on heavy pre-computation, and that new users or new items, for which no ratings are known, cannot be taken into account.

Hybrid methods

Hybrid approaches are used to cope with the limitations of both previous approaches. Such approaches include the aggregation of a content-based computed score with a collaborative computed score, the addition of content-based to collaborative filtering, or the use of item profiles to cope with the new item problem.

[1] This profile can be interpreted as: Homer contains sugar and is not ok for a diet.

1.3 Categorising the approaches

In this section we define precisely the scope of the lecture and present the criteria used to categorise the approaches.

1.3.1 Scope of the lecture

There is a lot of works in the database community that deal with query transformation, i.e., the process of, given a database query q, transforming q into another query q'. Among these work, we restrict our lecture to the following transformations:

- Query personalisation: given a database query q and some profile, compute a query $q' \subseteq q$ that has an added value w.r.t. the profile.
- Query recommendation: given a database query q and some profile, compute a query q' such that neither $q' \subseteq q$ nor $q \subseteq q'$, that has an added value w.r.t. the profile. Note that computing a query q' that include q would correspond to query relaxation (see e.g., [29, 37]).

Note that other forms of query transformation like e.g., relaxation, non relational data types (XML, etc.), and implementation and evaluation issues will not be covered.

1.3.2 Categorisation of the approaches

To describe and categorise the approaches presented in this lecture, we adopt the criteria introduced in [22], that are mostly used to differentiate personalisation approaches.

- Formulation effort: some approaches require the user to manually specify profile elements for each query, while in others the best they are inferred from the context and the user past actions.
- Prescriptiveness: some approaches use profile elements as hard constraints that are added to a query while in other as soft ones: tuples that satisfy as much profile criteria as possible are returned even if no tuples satisfies all of them.
- Proactiveness: distinguishes the approaches that suggest new queries based on the navigation log and on the context (but that does not execute them), with respect to those that change the current query or post process its results before returning them to the user.
- Expressiveness: personalization criteria have different expressivenesses and can be differently combined.

To precisely distinguish between the type of data needed, especially for recommendation techniques, we also use the taxonomy proposed in [47]. There, three categories are identified:

- Current-state approaches, exploiting the content and schema of the current query result and database instance. Current-state approaches can be based either on (i) the local analysis of the properties of the result of the posed query or (ii) the global analysis of the properties of the database. In both cases systems exploit (i) the content and/or (ii) the schema of the query result or the database.
- History-based approaches, using the query logs.
- External sources approaches, i.e., approaches exploiting resources external to the database.

1.4 Query personalisation and recommendation in relational databases

In this section, we give a brief overview of the approaches developed in the relational database context.

1.4.1 Personalisation in databases

We can distinguish two types of approaches:

- The use of explicit preference operators in queries. The most representative works in this category include Winnow [16], Preference SQL [31] and Skyline [10] This type of approaches requires high formulation effort, is not prescriptive not proactive, but is highly expressive.
- The rewriting (expansion) of regular database queries based on a profile. The most representative work is initiated in [32]. This approach requires a low formulation effort, is prescriptive and not proactive and has low expressiveness.

Use of dedicated operators

The basic definition of the operator computing dominating tuples is the following. Given a relation r with schema $sch(r)$ and a preference C over $sch(r)$ defining a preference relation $>_C$, the Winnow operator [16], denoted w_C, is defined by: $w_C(r) = \{t \in r | (\nexists t' \in r)(t' >_C t)\}$.

This operator can be used to order the query answer. Indeed, the answer to a query q can be partitioned according to C, i.e., $q = w_C(q) \cup w_C(q - w_C(q)) \cup \ldots$ meaning that the answer can be presented by displaying $w_C(q)$ first, then $w_C(q - w_C(q))$, etc.

Example 7. Suppose that preference C is "I prefer drama". The query "What are my most preferred affordable movies?" can be expressed by: $w_C(\sigma_{Price<7}(Movies))$. The answer can be presented by displaying t5 first, and then t1 and t2.

The work of Kiessling [31] extends this idea of having an operator dedicated to preference expression, to enrich the SQL syntax with a PREFERRING clause that enables the use of specific preference constructors. Each preference constructor can be use to express a specific atomic preference. Preferences can be composed with Pareto or prioritisation.

Example 8. Consider the following queries expressed with Preference SQL:

1. SELECT * FROM Movies PREFERRING HIGHEST(Duration)
2. SELECT * FROM Movies PREFERRING Genre IN (Drama,Thriller)

The model of preference specified by the first query is the following: for some duration x and y, $(x >_{HIGHEST} y)$ if value x is greater than value y, meaning that movies with highest durations will be preferred. For the second query, the model of preference says that Drama and Thriller movies will be preferred over any other genre. More formally, if x and y are two movie genres, $(x >_{IN(Drama,Thriller)} y)$ if $x \in \{Drama, Thriller\}$ and $y \notin \{Drama, Thriller\}$.

A restricted form of Kiessling's SQL extension is the addition of the Skyline operator to SQL [10]. This is a restriction in the sense that, originally, Skyline queries feature only numerical attributes and Pareto composition. This operator is defined as follows. The syntax of a skyline clause is:

$$\begin{aligned}\text{SKYLINE OF } & d_1 \text{ MIN}, \ldots, d_k \text{ MIN} \\ & d_{k+1} \text{ MAX}, \ldots, d_l \text{ MAX} \\ & d_{l+1} \text{ DIFF}, \ldots, d_m \text{ DIFF}\end{aligned}$$

The semantics of such a clause is that a tuple $p = (p_1, \ldots, p_n)$ dominates a tuple $q = (q_1, \ldots, q_n)$ if:

$p_i \leq q_i$, for $i = 1, \ldots, k$
$p_i \geq q_i$, for $i = k+1, \ldots, l$
$p_i = q_i$, for $i = l+1, \ldots, m$

Query expansion

In the absence of a dedicated preference operator, a regular user query can be processed and transformed with preferences, resulting in another regular query that is typically a subquery of the initial one. We use the work of [32] to illustrate this approach.

In this work, preferences come from a user profile which is modelled as a graph of atomic quantitative preferences of the form (selection condition, score), where selection is a regular selection predicate (that may be used to join two tables) and score is a real between 0 and 1 indicating the intensity of the preference. Atomic preferences are composed using a very simple principle: composition of preference $(s1, v1)$ with preference $(s2, v2)$ results in preference $(s1 \wedge s2, v1 \times v2)$.

Query expansion is performed as follows. First, given a query, the k most relevant preferences are selected from the profile. Then the selected preferences are added as hard constraints to the query, and the query is finally executed.

Example 9. Consider the following user query: SELECT title FROM Movies WHERE duration $<$ 120. Suppose that the best preference selected from the profile of the user who wrote the query is "I like Lynch as Author". Then the query is modified, resulting in: SELECT title FROM Movies WHERE duration $<$ 120 AND Author='Lynch'. Note that in this example, if the query is evaluated over the instance given in Example 1, then the result is be empty.

This work has been extended to take into account constraints like result cardinality or execution time [33].

1.4.2 Query recommendation in databases

Recently, to our knowledge, there has been only two attempts to formalize database query recommendations for exploration purpose [13, 47]. It is important to see that given the context of database exploration, a direct transposition of the users $\times$ items matrix is not relevant. Following [47], we use the categories introduced Section 1.3 to categorise these approaches.

Current state

In [47], the authors focus on the current state approach and propose two techniques to recommend queries based on the database instance and/or the current query answer. The first technique, called local analysis, analyse the answer to the user query to discover patterns and use these patterns to recommend. The second technique, called global analysis, extends this principle to the entire database instance. The instance would have to be analysed off-line, for instance to discover correlations among attribute values.

Example 10. Consider the current query: SELECT Author, Genre FROM Movies WHERE Duration $>$ 100. Suppose that by analysing this query answer, it is found that the result has a lot of tuples whose genre is Drama. Then, a possible recommendation would be: SELECT Author, Genre FROM Movies WHERE Genre='Drama'. Suppose now that a global analysis of the database instance shows that value Cohen for Author is correlated with value Comedy for Genre. Then, if the current query is: SELECT * FROM Movies WHERE Author='Cohen' a recommendation would be: SELECT * FROM Movies WHERE Genre='Comedy'.

History based

For [13], the problem of query recommendation is viewed as a sessions $\times$ tuples matrix. With this approach, a query is represented as a vector whose arity is

the number of tuples of the database instance. The basic approach considers that each component of such a query is either a 1, if the query used the tuple, or a 0 otherwise. A session is also represented by a binary vector which is the logical or of the vectors of the queries of the session. Consider a particular session S_c called the current session. Recommendations for S_c are computed as follows. First, sessions similar to S_c are found, using some vector similarity measures like e.g., cosine. For those session closest to S_c, the queries they contain are also compared to S_c using the same similarity measure. Finally, the queries of those sessions that are the most similar to S_c are recommended.

In subsequent works [7], the authors focus on fragments (attributes, tables, joins and predicates) of queries and consider thus a sessions $\times$ query fragments matrix. In this work, the matrix is used column-wise in the sense that recommendation computation relies on fragment similarity instead of session similarity.

Query completion

We conclude this section by noting that recommendation also make sense to assist the user writing a query. For instance, the SnipSuggest approach [30] is a collaborative approach that uses a query log to provide on-the-fly assistance to users writing complex queries. The query log is analysed to construct a graph whose nodes are the fragments appearing in queries and edges indicate the precedence. The edges are labelled with the probability that a fragment follows another fragment in the logged queries. Given the beginning of a current query, the graph is used to complete the query with the fragment that is the most likely to appear.

Example 11. Suppose that the query log contains only the following two queries: SELECT Title, Genre FROM Movies WHERE Actor=C. Lee and SELECT Title FROM Movies WHERE Author=Allen. Suppose that a user starts writing a query with only SELECT. It can be then suggested the attribute Title since this attribute is the most likely to appear according to the query log.

1.5 OLAP query personalisation and recommendation in data warehouses

In this section, we introduce the personalisation and recommendation approaches that are tailored to data warehouses queries. We start by reviewing the salient peculiarities of data warehouses compared to classical relational databases.

1.5.1 Peculiarities of data warehouses

As evidenced by e.g., [14, 45, 39] basic peculiarities of typical data warehouse can be summarized by:

1. A data warehouse is a read-mostly database and its instance has an inflationist evolution (data are added, never or very seldom deleted). It is for instance likely that a user issues periodically the same sequence of queries more than once.
2. A data warehouse is a database shared by multiple users whose interests may vary over time. It is argued in [8, 39, 24, 40] that user preferences are of particular importance in data warehouse exploration. It would for instance be important to issue recommendations computed from other users' habits (e.g., in a collaborative filtering fashion) and at the same time respecting the user interests.
3. A data warehouse has a particular schema that reflects a known topology, often called the lattice of cuboids, which is systematically used for navigation [25]. Rollup and drilldown operations that allow to see facts at various levels of detail are very popular in this context.
4. A typical analysis session over a data warehouse is a sequence of queries having an analytical goal, each one written based on the past results of the session. They may be expressed in a dedicated query language (like e.g., MDX [36]), may produce large results that are usually visualised as crosstabs. Moreover, the session has a sense w.r.t. some expectations. For instance, the user may assume a uniform distribution of the data [43, 44] or that two populations follow the same distribution [38]. Sessions (as sequences of queries) are of particular importance in this context since by this sequence the user navigates to discover valuable insights w.r.t. her expectations or assumptions.

We now describe how these peculiarities have been taken into account when personalising or recommending OLAP queries.

1.5.2 Personalisation of OLAP queries

To the best of our knowledge, only two works deal explicitly with the personalisation of OLAP queries. The first work [8] borrows from the query expansion paradigm, where a query expressed in MDX is transformed into another MDX query by using elements of the user profile. This approach features the same characteristics as that of [32] in terms of formulation effort, prescriptiveness, proactiveness and expressiveness. The second work [24, 9] is inspired by the use of explicit preference constructors enabling to express complex preferences directly within the query. This approach features the same characteristics as that of [31] in terms of formulation effort, prescriptiveness, proactiveness and expressiveness.

Query expansion

The work of [8] is inspired by that of [33], the main difference being that it does not use scoring function to represent preferences but relies on a qualitative model instead. It proposes to expand a current MDX query by another MDX query q using those elements of the profile that guarantee that (i) q is included (in the classical sense of query inclusion) in the current query, (ii) q only fetches preferred facts w.r.t. the profile and (iii) q respects a visualisation constraint. In this work, the user profile is given by a qualitative preference model relying on orders defined over dimension names and over attribute values. The visualisation constraint is used to indicate the maximum number of references (i.e., positions extracted from a data cube) that can be used for displaying the query answer. Note that this work assumes that the instances of the dimension tables can be used to compute the most preferred references in terms of the user profile.

Example 12. Consider the following MDX query:

```
SELECT CROSSJOIN({City.Tours, City.Orleans},Category.Members)
       ON ROWS
       {2003, 2004, 2005, 2006} ON COLUMNS
FROM   SalesCube
WHERE (Measures.quantity)
```

This query asks for 2 cities, 5 members of the Category level and 4 years, i.e., 40 cells of the cube. Suppose that the device for displaying the query result imposes the use of a cross tab with 2 axes with only 4 positions, i.e., only 16 cells. Suppose also that the user profile states that the most recent years are preferred and that, for the product dimension, the preference are: Electronics $<$ shoes $<$ cloth $<$ food $<$ drink.

The personalisation process is as follows. First the most preferred references are computed. In this example, that would be references *(Orleans, 2006, drink, quantity)* and *(Tours, 2006, drink, quantity)*. Then it is checked if this set of references complies with the visualisation constraint. In this example, as there are 2 preferred references for 16 available positions, this is indeed the case. When it is the case, then the second preferred references are added to the first preferred and again the whole set is checked against the visualisation constraint. The process stops when no more positions are available in the cross tab used to display the result, or all the references requested by the query have been included in the cross tab. Then the set of references is used to construct the personalised query, which, in this example, would be:

```
SELECT CROSSJOIN(City.Tours, City.Orleans,Category.Food, Category.drink)
       ON ROWS
       2003, 2004, 2005, 2006 ON COLUMNS
FROM   SalesCube
WHERE (Measures.quantity)
```

Use of dedicated operators

The work of [24, 9] is inspired by that of [31] where preferences are written for each query using a dedicated PREFERRING clause added to the MDX language. In this work, preference constructors are tailored to the multidimensional context. For expressing atomic preferences, the main differences with [31] are thus the following:

- The semantics of preferences over attribute values is hierarchy-driven.
- Preferences can be expressed over levels and thus over cuboids.
- Preferences can be expressed over measures.

As in [31], atomic preferences can be composed using Prioritization or Pareto. A specific implementation has been developed for evaluating preference queries expressed in this language.

Example 13. We illustrate two representative preference constructors. Suppose that one of the hierarchies of the 5-dimensional schema is named RESIDENCE and has levels City, State, Country. The preference PREFERRING City IN 'LA' not only means that City 'LA' is preferred over all other cities, but also that ancestors and descendants of 'LA' are preferred over values of this hierarchy that are neither ancestors nor descendants of 'LA'. For instance, the reference *(LA, all, 2010, F, all)* will be preferred over, say, *(NY,all,all,all,all)*, and the reference *(California,all,2009,all,all)* will be preferred over e.g., *(NY,all,2010,all,all)*. Note that *(LA, all, 2010, F, all)* $\sim$ *(California,all,2009,all,all)*

As another example, PREFERRING RESIDENCE CONTAIN City means that the facts grouping by level City are preferred over the facts grouping by another level of the RESIDENCE hierarchy. In that case, reference *(LA,all,2010,F,all)* is preferred over *(California,all,2009,all,all)*.

1.5.3 Recommending OLAP queries

Surprisingly, although there are many works in relational database dealing with query personalisation and quite a few works around query recommendations, in data warehouse, that seems to be the other way round. Indeed, to the best of our knowledge, the existing approaches for recommending OLAP queries are [27, 28, 12, 45, 43, 44, 46, 17, 18, 41, 42, 19, 20]. Note that the older works do not make use of the term query recommendation.

All these approaches are proactive, prescriptive, require a low formulation effort and have low expressiveness. To categorise them, we use and refine the categories proposed by [47]. We distinguish between (1) current-state methods exploiting external information (more precisely a user profile), (2) current-state methods based on expectations, (3) history-based methods exploiting query logs, and (4) hybrid (history and current-state) methods based on expectations.

Methods exploiting a profile

The works in this category [27, 28] suppose that a profile is provided together with the current query. The profile expresses user preferences over the tuples of the fact table using a quantitative approach. As in classical query expansion, the profile is used to modify the current query. The main difference with [32] is that the expanded query is not necessarily a subquery of the initial query.

Methods based on expectations

The works in this category [12, 45, 43, 44, 46] rely on discovery driven analysis, where a model on unseen data is used together with the already seen data, i.e., the results of the already launched queries of a given session. The strongest deviations to the model are recommended.

We briefly recall the concept of discovery driven analysis. To support interactive analysis of multidimensional data, [45] introduced discovery driven analysis of OLAP cubes as the definition of advanced OLAP operators to guide the user towards interesting regions of the cube, lightening the burden of a tedious navigation. These operators are of two kinds. The first kind tries to explain an unexpected significant difference observed in a query result by either looking for more detailed data contributing to the difference [43], or looking for less detailed data that confirm an observed tendency [46]. The second kind proposes to the user unexpected data in the cube w.r.t. the data she has already observed, by adapting the Maximum Entropy Principle [44].

Recommendation methods based on discovery driven analysis consist in recommending queries that result in data deviating the most from a consensual model (that we call expectation). Among the works in this approach, the main difference is the model used, i.e., the nature of the expectation. [43, 46, 44] rely on the assumption of a uniform data distribution. [12] assume a statistical independence of the cube's dimensions.

Example 14. We briefly illustrate the work of [44]. Suppose that a user asked for the total sale of Cheese in Europe for Quarter 1 of Year 2010. The result indicates that this sale is 100. Suppose that it is known (because the OLAP server transparently evaluates queries related to what the user is doing) that at the country level, this 100 is perfectly distributed among the 4 countries Europe drills down to (i.e., the sales of Cheese in these countries are 25 for each of France, Italy, Spain and U.K.). If such an uniform assumption is usually believed to be true, then this drill down does not correspond to a relevant recommendation. On the other hand, if at the month level, it is found that the sales of cheese in Europe is quite different from one month to another, then drilling down to the month level would be a good recommendation.

Note that [12] and [43, 46] compute suggestions using only the current query while [44] considers every former query result.

Methods exploiting query logs

The works in this category [17, 18, 41, 42] suppose that a query log is used to look for similarities between the current session and former sessions, to extract one past query as the recommendation.

The methods in this category mainly differ by the way they consider the similarity between sessions and/or queries. [18] use the classical Levenshtein (for session) and Hausdorff (for queries) distances. [41, 42] group queries by common projections and selections, and use a Markov model to represent sessions. [17] cluster queries using the Hausdorff distance and detects if the current session is a prefix of some existing session. [41, 42] and [17] identify a matching position for the current session in the closest former session and recommend the query after this position. [18] recommend the last query of the session that is the closest to the current one. The score that estimates the interest of a query for the session is computed based on the proximity of the current session with the log queries [18] or based on the probability to have the recommended query following the current query [41, 42].

Hybrid methods

The only work in this category is [19, 20]. In this work, that assumes a fix data warehouse instance, the query log is processed to detect discovery driven analysis sessions. Sessions are associated with a goal, and recommendations are those queries of former sessions having the same goal as that of the current session. More precisely, the model of expectation is the one proposed in [43, 46, 44]. The log is processed to discover pairs of facts that show a significant difference (like e.g., a drop of sales from one year to the following year). Those pairs are then arranged into a specialisation relation based on the hierarchies. At query time, if the current query investigates a pair that relates to the pairs discovered in the log, then the past queries featuring such pairs are recommended. The main difference with the techniques of [43, 46, 44] is that only the log is searched for interesting deviations.

Example 15. Suppose that the current query result shows that there is a big drop of sales in the sale of Cheese in Europe from year 2009 to year 2010. Suppose that in the log, it is found that past queries have already investigated a big drop of Cheese sales in France for the same couple of years. Then such queries will be recommended.

1.6 Conclusion

This lecture introduced OLAP query personalisation and recommendation, by first defining the basic concepts and then providing an overview of the existing approaches in relational and multidimensional databases. We restricted the

scope of this presentation to the problem of, given a database query q, compute a query $q' \subseteq q$ (personalisation) or compute a query q' such that neither $q' \subseteq q$ nor $q \subseteq q'$ (recommendation). We also categorise the approaches in terms of formulation effort, prescriptiveness, proactiveness, expressiveness, and the type of data needed: current state, history based, external data. With respect to this categorisation, it can be noted that many interesting combinations could investigated. For instance, to the best of our knowledge, there exists no approach that requires a low formulation effort, being proactive and not prescriptive while remaining highly expressive.

One of the main limitations in this field of research is that assessing effectiveness of the approach is very difficult since it should be based not only on real data but also on users' feedback, both being very difficult to obtain due to contexts the typically involves sensitive data. This is for instance illustrated by the fact that, in a data warehouse context, there exists no categorisation of the users' navigational behaviours, whereas such a categorisation exists in the web [11].

The domain of query personalisation and recommendation is still in its infancy. A lot of open issues can be listed, ranging from user privacy to the quality of recommendations and personalisations, most of these issues being also relevant beyond the data warehouse context. For instance, the learning and revision of preferences or navigational habits is one of these challenges. It is of paramount importance for computing more accurate and reliable personalisations and recommendations. A preliminary work tailored to the data warehouse context [2] is a first step in that direction.

References

1. A special report on managing information: Data, data everywhere. *The Economist*, February 2010. Available at http://www.economist.com/node/15557443.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
3. G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.
4. G. Adomavicius, A. Tuzhilin, and R. Zheng. REQUEST: A Query Language for Customizing Recommendations. *Information Systems Research*, 22(1):99–117, 2011.
5. J. Akbarnejad, M. Eirinaki, S. Koshy, D. On, and N. Polyzotis. SQL QueRIE Recommendations: a query fragment-based approach. In T. Catarci and Y. Stavrakas, editors, *PersDB*, 2010.
6. J. Aligon, M. Golfarelli, P. Marcel, S. Rizzi, and E. Turricchia. Mining preferences from olap query logs for proactive personalization. In J. Eder, M. Bieliková, and A. M. Tjoa, editors, *ADBIS*, volume 6909 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2011.

7. L. Bellatreche, A. Giacometti, P. Marcel, H. Mouloudi, and D. Laurent. A personalization framework for OLAP queries. In *DOLAP 2005, ACM 8th International Workshop on Data Warehousing and OLAP, Bremen, Germany, November 4-5, 2005, Proceedings*, pages 9–18, 2005.
8. P. Biondi, M. Golfarelli, and S. Rizzi. myOLAP: An Approach to Express and Evaluate OLAP Preferences. *IEEE Trans. Know. Data Eng.*, 23(7):1050–1064, 2011.
9. S. Börzsönyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, pages 421–430, 2001.
10. A. Z. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.
11. V. Cariou, J. Cubillé, C. Derquenne, S. Goutier, F. Guisnel, and H. Klajnmic. Built-In Indicators to Discover Interesting Drill Paths in a Cube. In *Data Warehousing and Knowledge Discovery, 10th International Conference, DaWaK 2008, Turin, Italy, September 2-5, 2008, Proceedings*, pages 33–44, 2008.
12. G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query Recommendations for Interactive Database Exploration. In *Scientific and Statistical Database Management, 21st International Conference, SSDBM 2009, New Orleans, LA, USA, June 2-4, 2009, Proceedings*, pages 3–18, 2009.
13. S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, 26(1):65–74, 1997.
14. P.-Y. S. Chen, S. yi Wu, and J. Yoon. The Impact of Online Recommendations and Consumer Feedback on Sales. In *Proceedings of the International Conference on Information Systems, ICIS 2004, December 12-15, 2004, Washington, DC, USA*, pages 711–724, 2004.
15. J. Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.
16. A. Giacometti, P. Marcel, and E. Negre. A framework for recommending OLAP queries. In *DOLAP 2008, ACM 11th International Workshop on Data Warehousing and OLAP, Napa Valley, California, USA, October 30, 2008, Proceedings*, pages 73–80, 2008.
17. A. Giacometti, P. Marcel, and E. Negre. Recommending Multidimensional Queries. In *Data Warehousing and Knowledge Discovery, 11th International Conference, DaWaK 2009, Linz, Austria, August 31 - September 2, 2009, Proceedings*, pages 453–466, 2009.
18. A. Giacometti, P. Marcel, E. Negre, and A. Soulet. Query recommendations for OLAP discovery driven analysis. In *DOLAP 2009, ACM 12th International Workshop on Data Warehousing and OLAP, Hong Kong, China, November 6, 2009, Proceedings*, pages 81–88, 2009.
19. A. Giacometti, P. Marcel, E. Negre, and A. Soulet. Query Recommendations for OLAP Discovery-Driven Analysis. *IJDWM*, 7(2):1–25, 2011.
20. S. Goel, A. Z. Broder, E. Gabrilovich, and B. Pang. Anatomy of the long tail: ordinary people with extraordinary tastes. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 201–210, 2010.
21. M. Golfarelli. Personalization of olap queries. In *6mes journes francophones sur les Entrepts de Donnes et l'Analyse en ligne (EDA 2010), Djerba, Tunisie*, volume B-6 of *RNTI*, page 1, Toulouse, Juin 2010. Cpadus. Article invit.
22. M. Golfarelli and S. Rizzi. *Data Warehouse Design: Modern Principles and Methodologies*. McGraw-Hill, 2009.

23. M. Golfarelli and S. Rizzi. Expressing OLAP Preferences. In *Scientific and Statistical Database Management, 21st International Conference, SSDBM 2009, New Orleans, LA, USA, June 2-4, 2009, Proceedings*, pages 83–91, 2009.
24. J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.
25. H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pages 13–24, 2007.
26. H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh. Preference-Based Recommendations for OLAP Analysis. In *Data Warehousing and Knowledge Discovery, 11th International Conference, DaWaK 2009, Linz, Austria, August 31 - September 2, 2009, Proceedings*, pages 467–478, 2009.
27. H. Jerbi, F. Ravat, O. Teste, and G. Zurfluh. A Framework for OLAP Content Personalization. In *Advances in Databases and Information Systems - 14th East European Conference, ADBIS 2010, Novi Sad, Serbia, September 20-24, 2010. Proceedings*, pages 262–277, 2010.
28. A. Kadlag, A. V. Wanjari, J. Freire, and J. R. Haritsa. Supporting Exploratory Queries in Databases. In *Database Systems for Advances Applications, 9th International Conference, DASFAA 2004, Jeju Island, Korea, March 17-19, 2004, Proceedings*, pages 594–605, 2004.
29. N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu. SnipSuggest: Context-Aware Autocompletion for SQL. *PVLDB*, 4(1):22–33, 2010.
30. W. Kießling. Foundations of Preferences in Database Systems. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pages 311–322, 2002.
31. G. Koutrika and Y. E. Ioannidis. Personalization of Queries in Database Systems. In *Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, 30 March - 2 April 2004, Boston, MA, USA*, pages 597–608, 2004.
32. G. Koutrika and Y. E. Ioannidis. Personalized Queries under a Generalized Preference Model. In *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5-8 April 2005, Tokyo, Japan*, pages 841–852, 2005.
33. P. Marcel. Personalization and recommendation of OLAP queries. http://www.info.univ-tours.fr/ marcel/BD/ebiss2011.pptx, 2011.
34. P. Marcel and E. Negre. A survey of query recommendation techniques for data warehouse exploration. In *7èmes journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 2011), Clermont-Ferrand*, volume B-7 of *RNTI*, pages 119–134, Paris, Juin 2011. Hermann.
35. Microsoft. MDX reference. http://msdn.microsoft.com/, 2009.
36. C. Mishra and N. Koudas. Interactive query refinement. In *EDBT 2009, 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24-26, 2009, Proceedings*, pages 862–873, 2009.
37. C. Ordonez and Z. C. 0002. Evaluating Statistical Tests on OLAP Cubes to Compare Degree of Disease. *IEEE Transactions on Information Technology in Biomedicine*, 13(5):756–765, 2009.
38. S. Rizzi. OLAP preferences: a research agenda. In *DOLAP 2007, ACM 10th International Workshop on Data Warehousing and OLAP, Lisbon, Portugal, November 9, 2007, Proceedings*, pages 99–100, 2007.

39. S. Rizzi. New Frontiers in Business Intelligence: Distribution and Personalization. In *Advances in Databases and Information Systems - 14th East European Conference, ADBIS 2010, Novi Sad, Serbia, September 20-24, 2010. Proceedings*, pages 23–30, 2010.
40. C. Sapia. On Modeling and Predicting Query Behavior in OLAP Systems. In *Proceedings of the Intl. Workshop on Design and Management of Data Warehouses, DMDW'99, Heidelberg, Germany, June 14-15, 1999*, page 2, 1999.
41. C. Sapia. PROMISE: Predicting Query Behavior to Enable Predictive Caching Strategies for OLAP Systems. In *Data Warehousing and Knowledge Discovery, Second International Conference, DaWaK 2000, London, UK, September 4-6, 2000, Proceedings*, pages 224–233, 2000.
42. S. Sarawagi. Explaining Differences in Multidimensional Aggregates. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 42–53, 1999.
43. S. Sarawagi. User-Adaptive Exploration of Multidimensional Data. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 307–316, 2000.
44. S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-Driven Exploration of OLAP Data Cubes. In *Advances in Database Technology - EDBT'98, 6th International Conference on Extending Database Technology, Valencia, Spain, March 23-27, 1998, Proceedings*, pages 168–182, 1998.
45. G. Sathe and S. Sarawagi. Intelligent Rollups in Multidimensional OLAP Data. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 531–540, 2001.
46. K. Stefanidis, M. Drosou, and E. Pitoura. "You May Also Like" Results in Relational Databases. In S. Amer-Yahia and G. Koutrika, editors, *PersDB*, pages 37–42, 2009.
47. K. Stefanidis, G. Koutrika, and E. Pitoura. A survey on representation, composition and application of preferences in database systems. *ACM Trans. Database Syst.*, 36(3):19, 2011.