YMAL: Ένα Σύστημα Συστάσεων Σχεσιακών Βάσεων Δεδομένων

Eftychia Koletsou[1], Kostas Stefanidis[2], Marina Drosou[1], Evaggelia Pitoura[1]

[1]*Dept. of Computer Science, University of Ioannina, Greece.* {ekoletso, mdrosou, pitoura}@cs.uoi.gr

[2]*Dept. of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong.* kstef@cuhk.edu.hk

Abstract— Σε αυτήν την επίδειξη λογισμικού παρουσιάζουμε τις YMAL συστάσεις, ένα πλαίσιο εργασίας που επεκτείνει τα σχεσιακά συστήματα βάσεων δεδομένων με λειτουργικότητα συστάσεων. Συγκεκριμένα, προτείνουμε, μαζί με τα αποτελέσματα μίας ερώτησης, να συστήνουμε στο χρήστη επιπλέον αποτελέσματα που ονομάζουμε **"You May Also Like"** ή YMAL αποτελέσματα. Για να υπολογίσουμε τα YMAL αποτελέσματα για μία συγκεκριμένη ερώτηση, χρησιμοποιούμε είτε μόνο το περιεχόμενο του αποτελέσματος της ερώτησης (τοπική προσέγγιση) είτε και το περιεχόμενο της βάσης δεδομένων (καθολική προσέγγιση).

I. Introduction

The typical interaction of a user with a database system is by formulating queries. This interaction mode assumes that users are to some extent familiar with the content of the database and also have a clear understanding of their information needs. However, as databases get larger and accessible to a more diverse and less technically-oriented audience, a new "recommendation"-oriented form of interaction seems attractive and useful.

In this paper, motivated by the way recommenders work, we consider "recommending" to the users tuples not in the results of their queries but of potential interest. For instance, when asking for movies with *detectives*, we could recommend movies with *policemen* as well. When looking for drama movies produced in *England* with *Oscar* nominations, we could also recommend similar movies with *BAFTA* awards. We call such results *"You May Also Like"* or YMAL results for short. YMAL results are useful because they let users see other tuples in the database that they may be unaware of.

Extending database queries with recommendations has also been suggested in two very recent works, namely [2] and [1]. [2] proposes a general framework and a related engine for the declarative specification of the recommendation process, while here, instead, we propose a specific recommendation method for relational databases. Recommendations in [1] are based on the past behavior of similar users whereas we consider the content of the database. A preliminary version of our recommendation functionality is presented in [3].

In this demonstration paper, we briefly describe YMAL recommendations for relational databases and present the architecture of our system for processing recommendation requests (Section II). We also describe our demonstration of YMAL recommendations for a relational movies database system (Section III).

II. YMAL Recommendation System Overview

Assume a relational database system $\mathcal{D}$ and a set of users interacting with it by posing traditional select-project-join (SPJ) queries. Given a query Q, a typical database system returns a set of results $Res(Q)$ in the form of tuples, possibly produced by joining several relations of $\mathcal{D}$. Besides $Res(Q)$, we would like to locate and recommend to users a set of tuples that may also be of interest to them. We call this set of tuples *"You May Also Like"* tuples or YMAL results. We denote this set as YMAL(Q).

To compute YMAL recommendations, we exploit the content and schema of the current query result and database instance. We consider an SPJ query Q of the form:

```
select A from R where P1 AND P2,
```

R denotes a set of relations of $\mathcal{D}$, A a set of attributes $\{A_1, \dots, A_n\}$ of the relations of $\mathcal{D}$, P1 the conjunction of the join conditions for Q and P2 the conjunction of the remaining selection conditions for Q. For processing recommendation requests, we re-write the submitted query Q into a set of queries referred to as YMAL *queries*. An YMAL query is of the form:

```
select B from S where P.
```

In the following, we will show our query re-writing mechanism for computing B, S, and P. This mechanism is based on either (i) local analysis of the intrinsic properties of the result $Res(Q)$ or (ii) global analysis of the properties of the database $\mathcal{D}$. The union of the results of YMAL queries constitute the YMAL recommendations, or YMAL(Q).

Next, we describe the main components of the architecture of our system. A high level representation is depicted in Fig. 1. Once a query Q is submitted, we compute its actual result set. Using the query results, we construct a set of YMAL queries by re-writing Q. The results of these queries correspond to the YMAL results.

Attribute Selection Generator. This component takes as input the submitted query Q and returns the attributes B that will appear in the select clause of the YMAL queries. In a local-based approach, where only $Res(Q)$ is employed, all YMAL queries have the same select clause which contains all attributes B that appear in the conditions P2 of the initial query.

In a global-based approach, where the content of the database is also taken into account, we use a special select clause. We simply construct this clause by removing from the set of attributes appearing in the relations of the from clause of the under-construction YMAL query, the non-informative attributes, such as the primary and foreign key attributes containing meaningless values.

Relation Selection Generator. This component takes as input the query Q and returns the relations S that will appear in the from clause of the YMAL queries. In the local-based approach, S = R, while in the global-based approach, S is a superset of R. To compute S, we maintain correlations among relations. A relation R_i is correlated to a relation R_j with a score p that

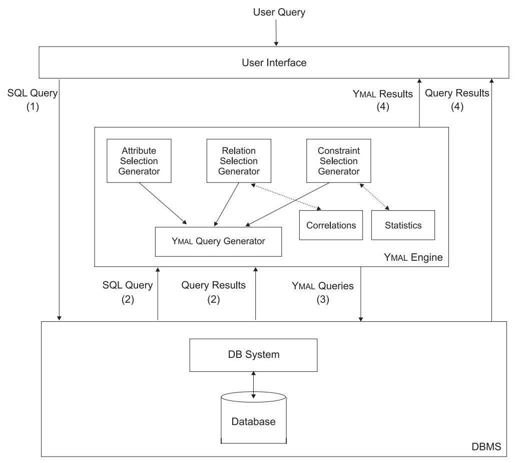


Fig. 1. YMAL system architecture.

reflects the result size of their join. This way, we construct S by adding to R the k most correlated relations to those in R, where k is determined by an input cardinality constraint. The higher the score p of a relation, the more correlated the relation.

Constraint Selection Generator. This component generates the selection conditions that the YMAL queries contain taking as input both Q and $Res(Q)$. Again, we distinguish between two cases; selection conditions are located either with respect to a local analysis of the intrinsic properties of the result $Res(Q)$, or based on a global analysis of the properties of the database $\mathcal{D}$.

In the former case, given a query Q with attributes $\mathcal{A} = \{A_1, \ldots, A_m\}$ appearing in the relations R, we first compute the power-set $\mathcal{A}^*$ consisting of all subsets of $\mathcal{A}$. Then, the selection operator constructs the YMAL queries' constraints with respect to $\mathcal{A}^*$.

Definition 1: Given a query Q, its attribute power-set $\mathcal{A}^*$ and the result set $Res(Q)$, the *selection operator* outputs, for each set of attributes of the form $\{A_i, \ldots, A_j\}$ in $\mathcal{A}^*$, a selection condition of the form:

$$A_i = a_{i_x} \texttt{ AND } \ldots \texttt{ AND } A_j = a_{j_y},$$

where $a_{i_x} \in dom(A_i)$, $a_{j_y} \in dom(A_j)$ and the set of values $\{a_{i_x}, \ldots, a_{j_y}\}$ is the most frequently appeared value set in $Res(Q)$ for $\{A_i, \ldots, A_j\}$.

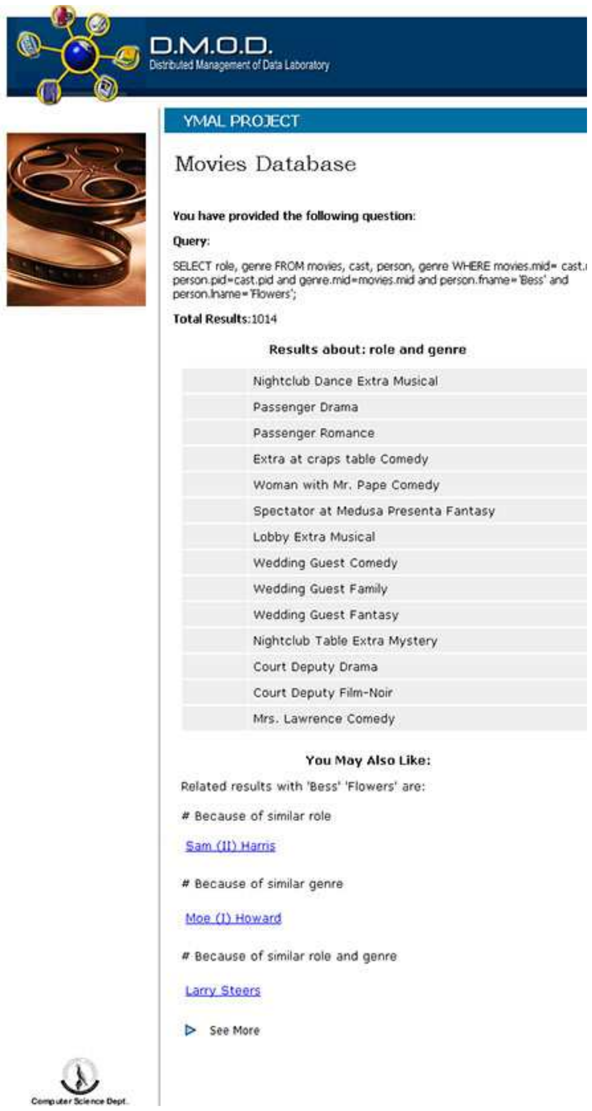
For each selection condition that is returned by the selection operator, a different YMAL query is constructed.

In the latter case, selection conditions are generated taking into consideration not only the query results but also statistics maintained for the database content. In particular, for each attribute set in $\mathcal{A}^*$, we locate the relative value set $\mathcal{V}$ that appears frequently in both $Res(Q)$ and $\mathcal{D}$ using the formula: $\frac{freq^{Res(Q)}(\mathcal{V})}{freq^{\mathcal{D}}(\mathcal{V})}$ where $freq^{Res(Q)}(\mathcal{V})$ denotes the number of occurrences of $\mathcal{V}$ in $Res(Q)$ and $freq^{\mathcal{D}}(\mathcal{V})$ denotes the number of occurrences of $\mathcal{V}$ in $\mathcal{D}$.

YMAL *Query Generator.* In this step, we construct the YMAL queries. To perform this operation, we employ the outputs of the previous steps. This component is also responsible for identifying the join conditions for each generated query. YMAL(Q) consists of the union of results of the produced YMAL queries.

III. Demonstration

We have implemented our system for computing a set of recommended results for a given query in Java on top of



Movies Database

You have provided the following question:

Query:

SELECT role, genre FROM movies, cast, person, genre WHERE movies.mid= cast.
person.pid=cast.pid and genre.mid=movies.mid and person.fname='Bess' and
person.lname='Flowers';

Total Results:1014

Results about: role and genre

Nightclub Dance Extra Musical
Passenger Drama
Passenger Romance
Extra at craps table Comedy
Woman with Mr. Pape Comedy
Spectator at Medusa Presenta Fantasy
Lobby Extra Musical
Wedding Guest Comedy
Wedding Guest Family
Wedding Guest Fantasy
Nightclub Table Extra Mystery
Court Deputy Drama
Court Deputy Film-Noir
Mrs. Lawrence Comedy

You May Also Like:

Related results with 'Bess' 'Flowers' are:

# Because of similar role

Sam (II) Harris

# Because of similar genre

Moe (I) Howard

# Because of similar role and genre

Larry Steers

See More

Fig. 2. Query results and YMAL recommendations.

MySQL. Implementation on top of an existing database system has a number of advantages, such as portability and ease of implementation. Our system can be accessed via a simple web browser using an intuitive GUI.

We demonstrate our method using a movies database. Users can submit their queries via SQL or by employing available input forms. After executing the query, users are presented with the results of the query and also a set of YMAL results (Fig. 2). An *explanation* is also provided along with each YMAL result, i.e. how this specific recommendation is related to the original query result. At first, we present one result tuple of each YMAL query. If the user clicks on this, more tuples from the result of this YMAL query appear as well.

References

[1] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recommendations for interactive database exploration. In *SSDBM*, 2009.

[2] G. Koutrika, B. Bercovitz, and H. Garcia-Molina. Flexrecs: Expressing and combining flexible recommendations. In *SIGMOD*, 2009.

[3] K. Stefanidis, M. Drosou, and E. Pitoura. "you may also like" results in relational databases. In *PersDB*, 2009.