Data & Knowledge Engineering 70 (2011) 208–229

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

A novel keyword search paradigm in relational databases: Object summaries

Georgios John Fakas *

Department of Computing and Mathematics, Manchester Metropolitan University, UK
Chester Street, Manchester, M1 5GD, UK

ARTICLE INFO

Article history:
Received 2 January 2010
Received in revised form 3 November 2010
Accepted 3 November 2010
Available online 23 November 2010

Keywords:
Information retrieval
Data extraction
Relational databases
Keyword search

ABSTRACT

This paper introduces a novel keyword search paradigm in relational databases, where the result of a search is an Object Summary (OS). An OS summarizes all data held about a particular Data Subject (DS) in a database. More precisely, it is a tree with a tuple containing the keyword(s) as a root and neighboring tuples as children. In contrast to traditional relational keyword search, an OS comprises a more complete and therefore semantically meaningful set of information about the enquired DS.

The proposed paradigm introduces the concept of Affinity in order to automatically generate OSs. More precisely, it investigates and quantifies the Affinity of relations (i.e. Affinity) and their attributes (i.e. Attribute Affinity) in order to decide which tuples and attributes to include in the OS. Experimental evaluation on the TPC-H and Northwind databases verifies the searching quality of the proposed paradigm on both large and small databases; precision, recall, f-score, CPU and space measures are presented.

1. Introduction

Keyword search is the leading search paradigm on the Web (W-KwS). For example, searching for information about a particular Data Subject (DS) (e.g. Janet Peacock), a W-KwS engine (e.g. Google, Yahoo!, etc.), as a result, will return a set of web pages containing the keyword(s). The users can read through these web pages and find enough information about the particular DS they are interested in; sometimes they can even find a complete answer to their enquiry in a single web page (for instance, in the DS's personal web page). A DS usually represents an individual who is a subject of personal data; in this paper however, a DS has a broader meaning and represents any object (e.g. a product, order, etc.) that is a subject of data.

The keyword search paradigm has also recently been successfully used in relational databases (R-KwS). Such paradigms have significant usability advantages as they liberate users from technical details such as database schemata and query languages. For instance, Full-Text techniques will return tuples (e.g. from Customers or Employees relations) containing the "Janet Peacock" keywords. However, such a tuple(s) (unlike W-KwS results) does not comprise a complete result since additional information is required to comprise a meaningful result for the DS; namely her Nation, Orders, etc. Other R-KwS techniques facilitate users to do advanced search using a set of keywords [1–3]; e.g. "Peacock Leverling" which will return trees of nodes containing information associating the two keywords such as Orders of Customer Leverling prepared by Employee Peacock. As a rule, such R-KwS results are comprised of either individual tuples (when only one keyword is given) or trees of tuples (when two or more keywords are given). Yet, unlike W-KwS results (namely a web page), R-KwS results fail to provide a comprehensive and therefore meaningful summary of information about a particular DS.

In this paper, a novel keyword search paradigm is proposed that produces results which are comprised of a more complete set of information about the particular DS; namely an Object Summary (OS). For instance, when searching information about "Janet Peacock" the proposed paradigm will produce an OS that summarizes all information about Janet Peacock (Fig. 1). More precisely

* Tel.: +44 161 2473537; fax: +44 161 2471483.
E-mail address: g.fakas@mmu.ac.uk.

0169-023X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2010.11.003

Employee

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	Address	...
3	Peacock	Janet	Sales Representative	Ms.	722 Moss Bay Blvd.	...

EmployeeTerritories

TerritoryDescription	RegionDescription
Atlanta	Southern

Region

Employee (Reports To)

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	...
2	Fuller	Andrew	Vice President, Sales	Dr.	...

Orders

OrderID	ShipName	ShipAddress	OrderDate	RequiredDate	...
10273	QUICK-Stop	Taucherstae 10	1996-08-05	1996-09-02	...

Customer

CustomerID	CompanyName	ContactName	Address	...
Quick	QUICK-Stop	Nick Leverling	Taucherstae 10	...

Shippers

ShipperID	CompanyName	...
3	Federal Shipping	...

Orders Details

UnitPrice	Quantity	Discount
15.2000	50	0.2

Products

ProductName
Chang

Fig. 1. The OS for "Janet Peacock".

in the Northwind database context (schema and sample dataset are shown in Figs. 2 and 3 respectively) [4], an OS will be composed of an Employee tuple (with FirstName="Janet" and LastName="Peacock") as a root and child nodes including additional information about her Nation, Regions, Orders she served, etc. In order to produce an OS, the proposed paradigm traverses the data-graph as follows: it starts from a tuple containing the keyword(s) (denoted as t^{DS}) and continues traversing neighboring tuples as long as the data traversed is relevant to t^{DS}.

The proposed paradigm, similarly to R-KwS, liberates users from query language and schema technical details. On the other hand, its semantics differentiate fundamentally from traditional R-KwS's semantics. The proposed paradigm facilitates the extraction of information and hence the generation of an OS about the particular DS even with a single keyword (e.g. solely "Peacock"); whereas R-KwS generates a set of joining tuples that collectively contain a set of query keywords.

1.1. Challenges

The proposed search paradigm faces several challenges. The primary challenge is the classification of neighboring data as relevant or irrelevant to t^{DS}. For this reason, the semantic of *Affinity* of surrounding relations to the relation (denoted as R^{DS}) of t^{DS} is investigated and quantified so as to select which relations to traverse. These Affinity scores in combination with a threshold (e.g. 0.70) either proposed by the system or provided by the DBA (or users) will facilitate the decision on which relations to retrieve in the context of an OS and therefore liberate the user from the schema details.

A secondary challenge is the classification of attributes in each relation as relevant or irrelevant to R^{DS}. For example, consider the generation of an OS for Customer "Leverling", details such as BirthDate, HomePhone, etc. concerning the Employee Peacock

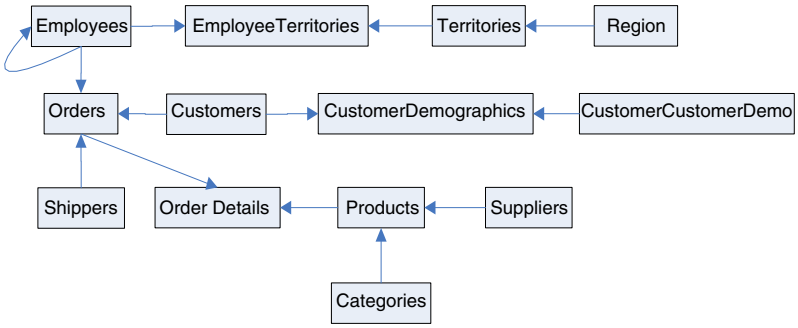

Fig. 2. The Northwind database schema.

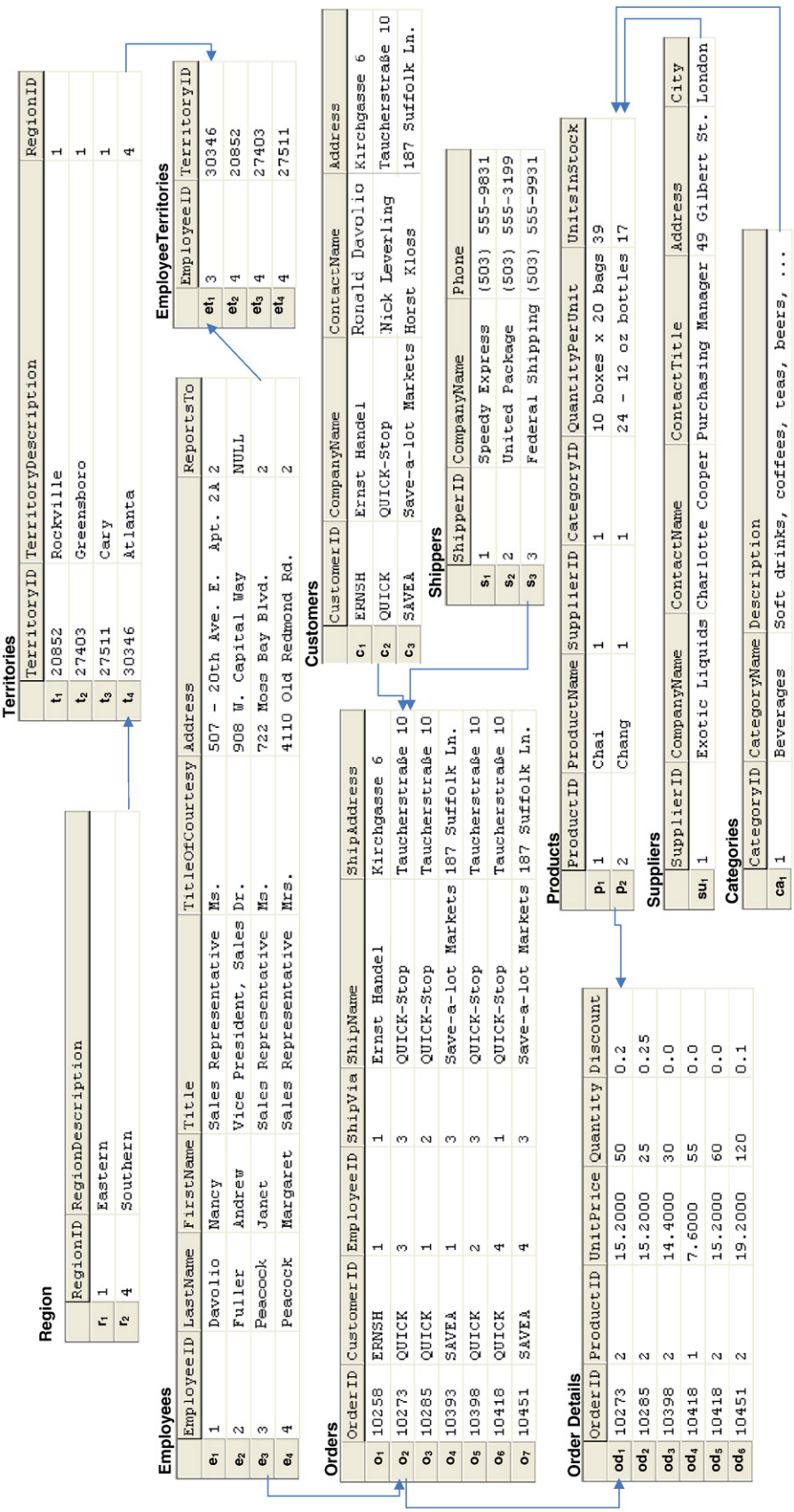

Fig. 3. A sample of the dataset from the Northwind database.

who has served the particular Customer are certainly not relevant to Leverling. For this reason, the following novel approach is proposed: the attributes of each relation are firstly clustered to attribute clusters and then each attributes' cluster is assigned an *Attribute Affinity* score. Hence, the semantic of Attribute Affinity to R^{DS} also needs to be investigated and quantified. For attribute clustering, a technique that combines linguistic and attributes' data types matching is employed.

The selection of effective thresholds that satisfy users' needs is another challenging problem. Since, this may be a difficult and unfriendly task for users as they cannot easily predict how comprehensive or abstract an OS may be with a particular threshold. We propose a technique that facilitates the quick selection of thresholds by estimating the sizes of OSs.

Finally, the ranking of OSs is also challenging since existing ranking semantics of traditional R-KwS are completely inappropriate for OS ranking. As in R-KwS, a result of a small size has generally a higher ranking semantic than another result of a larger size [1–3]. In contrast, an OS containing many well connected tuples should have certainly greater importance. In this paper, a ranking paradigm is proposed that ranks OSs descending their Importance scores, (denoted as $Im(OS)$) that combines (1) each comprising tuple's local Importance score (denoted as $Im(OS, t_i)$) and (2) the size of the OS (denoted as |OS|); where $Im(OS, t_i)$ is a function of (1) tuple's global Importance score (denoted as $Im(t_i)$) and (2) tuple's Affinity score (denoted as $Af(t_i)$).

1.2. Novel contributions

The novel contributions of this paper are the following:

- The formal definition of the novel keyword search paradigm which automatically produces an Object Summary from relational databases about a particular DS. The novelty of this paradigm is that it requires minimum contribution from the user (i.e. only a keyword) and does not require any prior database registration, prior knowledge of the database schema or prior knowledge of any query language.
- The formal definition of Relations Affinity to R^{DS} and the proposition of the Affinity formula in relational databases. Good Affinity calculation is the building block of the paradigm and this is achieved through a thorough analysis of the semantics of relational databases. The Affinity formula considers (1) the schema design, (2) data distributions and also (3) the impact of "hub" relations. The impact of the latter metric (3) although is very significant on Affinity, has not been considered before in this context. The excellent precision, recall, f-score (P/R/F) and Affinity Ranking Correctness results of OSs validate the quality of the proposed Affinity formula.
- The formal definition and calculation of Attribute Affinity to R^{DS}. Attribute Affinity facilitates the filtering of irrelevant attributes to R^{DS} and therefore the generation of more meaningful OSs. Furthermore, such an attributes' filtering also reduces the size of OSs. The automated filtering of attributes is another feature that has not been attempted yet in keyword search. P/R/F results together with space savings are presented.
- The proposition of a technique that facilitates the quick and user friendlier selection of thresholds by estimating the sizes of OSs.
- The proposition of a novel ranking paradigm of OSs. We propose a combining function that considers comprising tuples' local Importance and the size of the OS in order to estimate the Importance of each OS.

1.3. Paper organization

The rest of the paper is structured as follows: Section 2 discusses related work. Section 3 presents the proposed searching paradigm by utilizing examples whilst Section 4 describes the system architecture. Sections 5 and 6 discuss Relations and Attribute Affinity semantics in relational databases and present the respective Affinity formulas whereas Section 7 introduces a technique that facilitates the estimation of thresholds. Section 8 discuses the ranking of OSs. Section 9 presents the experimental evaluation and finally Section 10 concludes the paper.

2. Related work

To the best of our knowledge there is not any related work at all in the area of the automated generation of OSs (apart from the early work in [5] that included only preliminary descriptions). We present and compare related work of the area.

2.1. Keyword search

Full-Text search techniques in relational databases facilitate the discovery of tuples that contain queried keywords; e.g. Oracle 9i Text [6], IBM DB2 Text Information Extender [7], Microsoft SQL Server 2000 [8], MySQL Full-Text search [9], etc. Whilst, R-KwS techniques facilitate the discovery of joining tuples (i.e. Minimal Total Join Networks of Tuples (MTJNTs) [3]) that collectively contain all query keywords and are associated through their keys; for this purpose the concept of Candidate Networks is introduced. MTJNTs are (1) total as they must contain all query keywords and (2) minimal as every leaf node must contain a unique keyword. For instance for the keyword query "Leverling Peacock", R-KwSs will return MTJNTs with Orders of Customer Leverling prepared by Employee Peacock, etc. (e.g. Fig. 4). Note that because of the minimality constraint, o_2 cannot contain elaborating and potentially interesting data such as its OrderDetails and that both leaf nodes c_2 and e_3 contain unique keywords. Note also that for the same reason, if a keyword query consists of only one keyword, e.g. "Peacock", the results will be composed by single nodes (Fig. 5(b)). DISCOVER [3], DBXplorer [10], Mragyati [11], and relational stream keyword search [12] use series of SQL

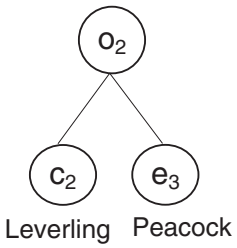

Fig. 4. An MTJNT result for keyword query "Leverling Peacock" (based on the sample dataset of Fig. 3).

statements in order to execute such keyword queries whilst other systems such as BANKS [1,2] convert the whole database into a data-graph. Other work in R-KwS also investigates efficiency, effectiveness and ranking [13–24].

Evidently, the proposed paradigm does not only differentiate semantically from traditional R-KwSs but also technically. Semantically as illustrated by Figs. 4 and 5, the proposed paradigm produces an OS for each t^{DS} containing the keyword(s), whereas R-KwS produces MTJNTs that collectively contain all keywords. In other words, the proposed paradigm produces summaries of information about particular DSs, whilst traditional R-KwSs produce results describing the associations of the given keywords. Furthermore, the proposed paradigm can produce meaningful OSs even with a single keyword in contrast to R-KwS that requires a set of keywords to produce meaningful MTJNTs. Technically, the proposed paradigm considers G^{DS} graphs (described in Sections 3 and 4.2), Affinity and Attribute Affinity in order to generate OSs, whereas R-KwSs consider Candidate Networks in order to produce MTJNTs.

Earlier work in keyword search includes proximity searches [25]. Proximity searches are specified by a pair of queries: A "Find query" specifies a *Find set* of objects that are potentially of interest and a "Near query" specifies a *Near set*. The objective is to rank objects in the *Find set* according to their distance to the *Near objects*. For instance, the query "*Find* Orders *Near* Peacock Leverling" will return Orders ranked based on their proximity to keywords Peacock and Leverling. Namely, o_2 which has distance one from both *Near* keywords will be ranked higher than o_3 which has distance one only to Peacock. Although this search paradigm differentiates semantically from our work, technically employs similar techniques. For instance, ranking considers the proximity (i.e. distance) of objects and the impact of "hub" relations. As we explain in Affinity Calculation section, these are metrics that we also build upon for the calculation of Affinity. Similar metrics have also been considered in semantic Web search [26].

Précis queries resemble our work because they include additional information to the nodes containing the keywords by considering neighboring relations and their attributes [27–29]. More precisely, a précis query result is a logical subset of the original database (i.e. a subset of relations and a subset of tuples). For instance, the précis for the query "Peacock" is a subset of the database that includes the tuples containing "Peacock" (e.g. e_3, e_4) and their neighboring tuples from Orders, OrderDetails, Products, Employees, etc. (Fig. 5(c)). Précis also support the narrative presentation of the results, e.g. "Peacock is an employee …", "Peacock has an order …", "Order 10273 has orderdetails ….", "Orderdetails 10273 consists of product 2 …", "Product 2 is …", etc. In contrast to our result which is a set of two separate OSs (Fig. 5(a)). OSs produce more personalized summaries about the DSs, i.e. a hierarchical structure with the DS tuple as a root (rather than a database subset or its equivalent narrative presentation). In this spirit, some tuples are replicated in OSs when necessary. For instance, the Employee e_2, which Employees e_3 and e_4 report to, appears in both OSs and tuple r_1 is replicated twice in the e_4 OS. On the contrary, précis do not replicate such information and as a consequence, users may find big précis difficult to understand (even their narrative presentation).

A significant technical difference is that the selection of neighboring relations and their attributes is based on weights manually set by DBAs (in contrast to our approach where Affinity is automatically generated). For instance, for the Northwind database, this will require, in addition to the threshold, 112 manually set weights: two weights per relationship (2*13 relationships) plus one weight per attribute (86 attributes). Evidently, this is a demanding and difficult task for the DBA and probably an impossible task for ordinary users.

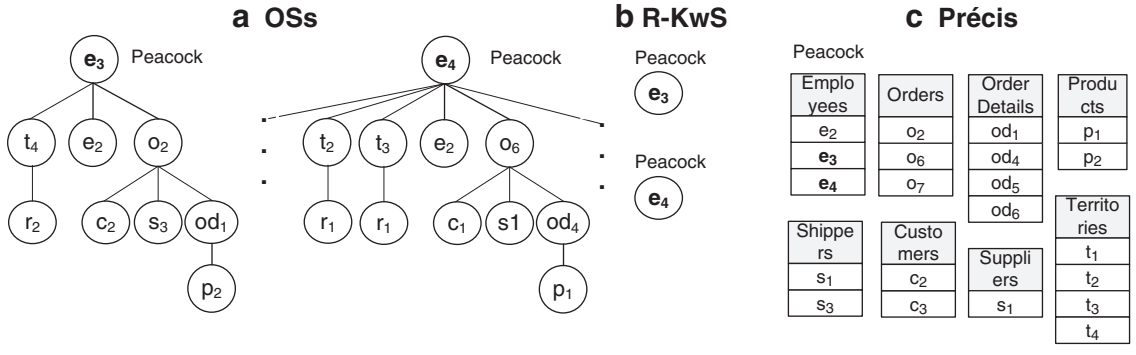

Fig. 5. The results for the keyword query "Peacock" (based on the sample dataset of Fig. 3).

2.1.1. Ranking in R-KwS

R-KwS techniques also facilitate the ranking of their results; such ranking paradigms consider (combined functions) of the following important (among other) metrics:

(1) The *result's size*, i.e. the consideration of the inverse of the amount of joins. The intuition is that, the less joins an MTJNT contains the more important semantically it is. On the contrary, this ranking metric is inappropriate for OS ranking where an OS containing many and well connected tuples should have certainly greater importance. For instance, an Employee who has served many orders should have higher ranking score than another Employee with fewer orders.
Note also that almost all such paradigms try to capture somehow the Affinity (as proposed in this paper) of tuples comprising an MTJNT by considering the inverse of the amount of (weighted) joins (e.g. [1–3]). The amount of joins corresponds to the relations distance metric of our Affinity formula (Section 5.1). However, as we show in Section 5, this is only one metric of Affinity; as schema design, data distributions and spurious short-cuts also affect Affinity. Therefore, we believe that even for MTJNT ranking, the proposed Affinity may be a more appropriate ranking metric rather than solely the amount of joins (i.e. distance).

(2) *IR-style* techniques [14–18], i.e. the consideration of the amount of times keywords (terms) appear in MTJNT results and also in tuples and attributes of the database. Such techniques have several limitations when applying on databases in general and also on OSs in particular as they miss important tuples that do not contain the keyword(s). For instance consider the keyword query "Peacock", the territories that Peacock serves also have some Importance (analogous to the Importance of Peacock and the other employees serving the particular territories) even though they do not include the particular keyword. Recall that usually an OS contains the keyword(s) only once, at the t^{DS} (i.e. the root node of the OS) and therefore such techniques will fail to rank effectively the remaining tuples of the OS.

(3) *Tuples' Importance*, i.e. the consideration of the authority flow through relationships in the corresponding data-graph of the database, e.g. PageRank [30], ObjectRank [31,32], ValueRank [33], BANKS (i.e. PageRank inspired) [1,2], etc. For instance, ObjectRank and ValueRank are extensions of PageRank on databases that employ the concept of Authority Transfer Rates between the tuples of each relation of the database in order to control the flow of authority.
Such ranking systems, unlike IR-style techniques, can produce effective Importance scores for all tuples of the database regardless whether they contain the keywords or not. This is very useful in ranking OSs, as we can assign scores to all tuples of the OS (including those that do not include the keywords). Hence, they can be used in OS ranking in order to estimate the global Importance score for each tuple of the database.

2.2. Schema summarization

Schema clustering and summarization techniques investigate similar semantic properties of schemata such as Affinity and Importance in order to cluster and summarize schemata respectively; where Affinity and Importance are two different concepts. More precisely, Affinity measures the closeness between entities; e.g. Employee e_3 is semantically closer to Order o_2 than to Product p_2 (Fig. 5(a)). Whereas, Importance ranks entities according to their significance using metrics such as connectivity (centrality) influenced mainly by PageRank [30]; e.g. Product p_2 is more important than p_1 because it is more connected (as it participates in more orders). Earlier related work in schema clustering relies only on schema properties without considering the data distribution, e.g. on semantics embedded in relationships; examples of such work are [34,35].

More recent work on schema summarization [36,37] investigates the Affinity and Importance of schema elements in order to produce a schema summary of XML and relational databases respectively. For instance, according to [36] the Affinity of a relation R_i to R^{DS}, denoted as $Af-H_{R_i \to R^{DS}}$ (or simply $Af-H_{R_i}$), can be calculated as follows:

$$Af-H_{R_i \to R^{DS}} = \frac{1}{fd_i} \cdot \prod_{j=2}^{fd_i} \frac{1}{RC\left(R_{j-1} \to R_j\right)} \quad (1)$$

where fd_i is the distance of R_i to R^{DS}, j ranges over all relations along the path from R_i to R^{DS} and RC is the Relative Cardinality between those relations (all these terms are explained in detail in Section 5). An important similarity of our work with the Affinity formulas proposed in [36,37] is the consideration of both schema and data distribution (e.g. distance and Relative Cardinality). On the other hand, there are several differences that make the Affinity proposed in [36,37] inappropriate to apply on our paradigm. Firstly, the Affinity in [36,37] produces scores which are very difficult to combine with a common threshold in order to generate OSs, mainly because they are skewed, and secondly, schema summaries correspond to the whole schema in contrast to OSs which correspond to an object (thus are tuple-oriented i.e. t^{DS}-rooted) and therefore, Affinity has a different meaning. Another semantic difference is that they disregard properties specific to relational databases such as "spurious short-cuts" created by hub relations (the impact of hub relations on Affinity is discussed in Section 5). Nevertheless, Affinity comparative results and further discussion are presented in Section 9.

2.3. Attribute filtering

Although attribute filtering and ordering in databases' queries has been examined in many database contexts, unfortunately none of this previous work is suitable for the automated filtering of attributes in OSs. For instance in R-KwS, précis queries facilitate

the filtering of attributes [27]; however this filtering is not completely automatic since it is based on weights manually set by DBAs. In other related work, the filtering decision is based on query's attributes which do not exist in our case, e.g. [38].

Related work in schema and ontology matching investigate the matching of schema and ontology elements respectively [39–41]. Such methodologies employ, among other techniques, linguistic and attributes' data type matching; application of such techniques has been used for attribute clustering in this work.

3. The proposed searching paradigm: an overview with examples

In this section, the proposed paradigm is illustrated with examples of keyword queries on the Northwind database based on the sample dataset of Fig. 3. The detailed schemata of the Northwind [4] and TPC-H [42] databases are included in the Appendix (Figs. 18 and 19).

In the context of the proposed novel keyword search paradigm, a keyword query is a set of keywords that can identify DSs; e.g. (1) "Peacock", (2) "Janet Peacock". The result of such a keyword query is a ranked set of OSs; where an OS is a tree comprised of (projections of) tuples, with the t^{DS} tuple as the root node and t^{DS}'s neighboring tuples (i.e. tuples that are associated through their keys) the child nodes. An OS is generated for each tuple (t^{DS}) found in the database that contains the keyword(s) as part of an attribute's value. Therefore, for the above query examples an OS will be generated for tuples (1) $\{e_3, e_4\}$ and (2) $\{e_3\}$ respectively. Full-Text facilities are employed to find all such tuples (Section 4.1).

In order to construct OSs, the proposed approach combines the use of graphs and SQL. The rationale is based on the fact that some relations, denoted as R^{DS} (where $t^{DS} \in R^{DS}$ contains the keyword(s)), hold information about the Data Subjects (DSs) and the relations linked around R^{DS}s contain additional information about the particular DS. For each R^{DS}, a Data Subject Schema Graph (G^{DS}) is automatically generated; namely a directed labeled tree that captures a subset of the database schema with R^{DS} as a root. The challenge now is the selection of the relations and attributes from G^{DS} that have the higher Affinity with the R^{DS} that need to be traversed in order to create a good OS. For this reason, Affinity measures of relations and attributes in G^{DS} are investigated, quantified and annotated on the G^{DS}. Provided an Affinity threshold θ a subset of G^{DS} can be produced; denoted as $G^{DS}(\theta)$. Furthermore, provided both Affinity and Attribute Affinity thresholds θ and θ', a $G^{DS}(\theta, \theta')$ can be produced. Finally, by traversing the $G^{DS}(\theta, \theta')$ we can now proceed with the generation of OSs.

For instance, for the keyword query "Janet Peacock", $\theta = 0.7$ and $\theta' = 0.7$ the system will automatically generate the report presented in Fig. 1. Since the keywords are found in tuple e_3 (i.e. the t^{DS} which belongs to the Employees relation) then Employees relation is considered as the R^{DS} and consequently the Employees G^{DS} (Fig. 7) will be generated by our G^{DS} Generator. Our system will insert tuple e_3 at the root node of the OS tree and then will start traversing the dataset based on the Employees G^{DS} in order to generate the rest of the report. Note also that some attributes are filtered out, e.g. from Products all attributes are filtered out except from ProductName.

4. System architecture

A high level representation of the proposed architecture is presented in Fig. 6. In this section, a brief description of the system's modules is presented; yet, more important modules are described in more detail in the following sections.

Firstly, the user enters the identifying keywords for a DS together with an Affinity and Attribute Affinity thresholds into the system. Then, the Master Index returns all tuples that contain these keywords (Section 4.1). The second step is the generation and annotation (with Cardinality and Relative Cardinality information) of necessary schema graphs (G^{DS}, G^{S}, etc.) (Section 4.2). Then, the Affinity and Attribute Affinity Calculation modules will calculate the Relation Affinity and Attribute Affinity to R^{DS} respectively; these scores are annotated on the G^{DS} (Sections 5 and 6 respectively). Steps 2–4, i.e. G^{DS} Generation, Affinity and Attribute Affinity Calculation modules, can be pre-computed and their outputs can be reused during a keyword search. OS Generation module takes as input the tuple set and $G^{DS}(\theta, \theta')$ graphs in order to generate the OS (Section 4.3). Finally, the OS Ranking and Presentation module ranks and prints in an intelligible form the ranked set of OSs (Section 4.4 and Section 8).

4.1. Master Index

The user enters his/her query as a set of keywords into the Master Index; which has been implemented using the MySQL Full-Text search [9]. The Master Index will return all tuples that contain the keywords as part of an attribute value; the set of such tuples forms the $DB^{id\text{-}kw}$ set. During this phase we can limit the search space with more narrowed searches; namely to a set of relations and attributes (Boolean expressions such as OR are also supported). For instance, Employees.LastName, Naming Clusters A_N of attributes (i.e. clusters composed of naming attributes, e.g. Name, Surname, First name, Company name, etc.), DS IDs such as primary keys, etc. This can be very useful in order to eliminate keywords found in physical relations or A_C clusters (A_N and A_C clusters are described in Section 6) that potentially will give meaningless OSs.

4.2. G^{DS} Generator

This module takes as input the $DB^{id\text{-}kw}$ tuple set and automatically generates (1) a G^{DS} for each R^{DS} that its tuples exists in $DB^{id\text{-}kw}$ and (2) a schema graph G^{S}; both types of schema graphs are generated without any user intervention. Alternatively, the generation of G^{DS} graphs can be pre-computed for all potentially interesting R^{DS}s.

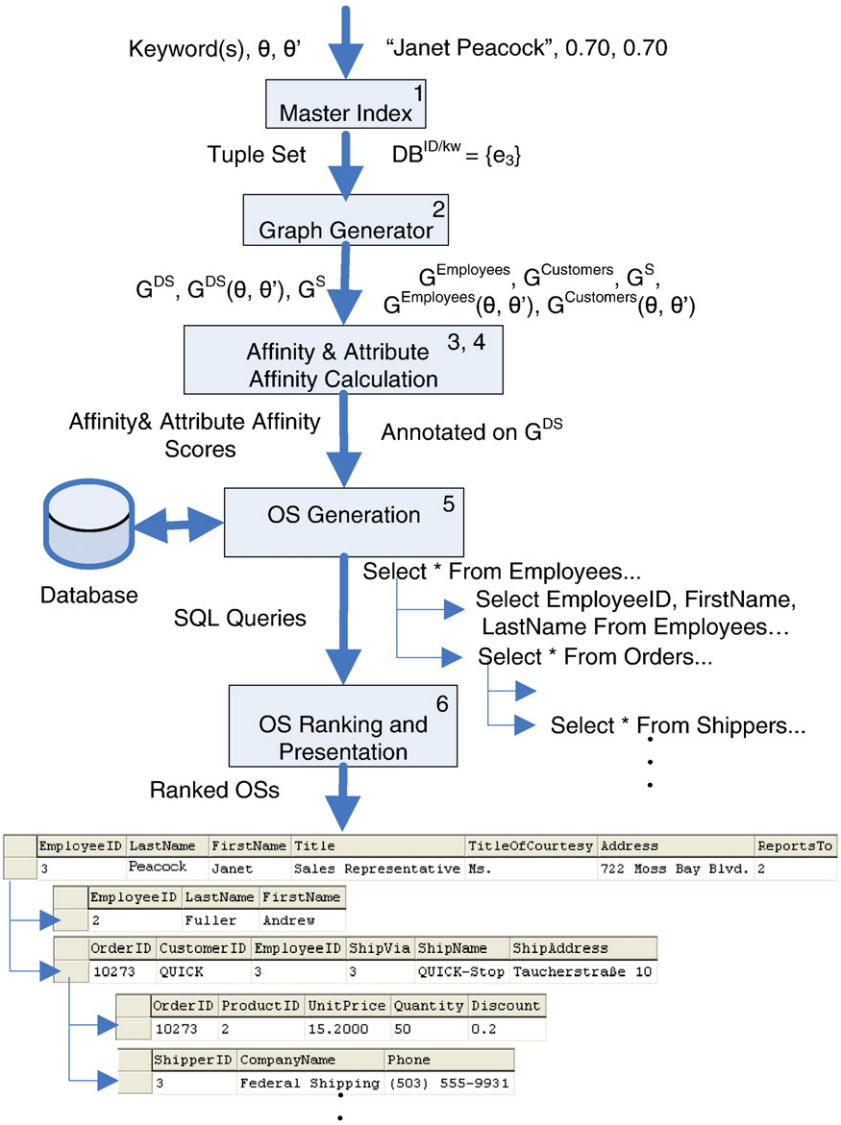

Fig. 6. The proposed architecture.

Definition 1 (Data Subject Schema Graph (G^{DS})). *A G^{DS} (V, V′, E) is a directed labeled tree that has an R^{DS} as a root node and captures the subset of the schema surrounding R^{DS}; where any surrounding relations participating in loop relationships are replicated. It consists of:*

- *a set of nodes $V = \{R_0, \ldots, R_n | n \geq 0\}$ where each R_i represents a relation of the database schema and the root node R_0 has the additional property that represents R^{DS}. Each node is labeled with Cardinality, Affinity and Attribute Affinity scores;*
- *a set of nodes $V' = \{R_{n+1}, \ldots, R_{n+m}\}$ where each R_i from V′ is a replication of R_j from V that participates either in a loop or recursive relationship. Again, each node is labeled with Cardinality, Affinity and Attribute Affinity scores;*
- *a set of edges $E = \{(R_i \rightarrow R_j) | R_i, R_j \in V \cup V'$, each $R_i \rightarrow R_j$ captures the primary to foreign key relationship between relations R_i and $R_j\}$. Each edge $R_i \rightarrow R_j$ is labeled with a name, Relative Cardinality and Reverse Relative Cardinality.* □

All G^{DS}s have a common maximum depth T; i.e. the longest distance fd_i of any R_i to R^{DS}. For the exemplary databases that we consider in this paper, T was empirically set to 4 as evaluators selected in most cases relations in G^{DS}s with maximum $fd_i \leq 3$ and exceptionally with maximum $fd_i = 4$ (since relations with $fd_i \geq 4$ represent completely irrelevant data to R^{DS}). In general, other values of T can also easily be considered; although intuitively we observe that $T \leq 4$ is generally adequate. G^{DS} Generator, whilst constructing G^{DS}, actually does two tasks: (1) "Treealizes" the schema graph G^S by inserting R^{DS} as a root, adding neighboring relations as child nodes and replicating looped relations and relationships and (2) annotates the G^{DS} with properties such as Cardinality, Relative Cardinality, etc.

Treealization is very useful for later operations on G^{DS}; e.g. it simplifies their annotation and enumeration. More precisely, relations and relationships participating in loop or recursive relationships (e.g. the relationship ReportsTo on Employees) are Treealized, i.e. replicated accordingly and then included in the G^{DS} (denoted with rounded corners in diagrams; see figure below). Such relations are retrieved as a closed path (namely first node is retrieved up to twice (as first and last node) and the rest intermediary nodes only once) when the loop length is smaller than T. Similarly recursive relationships are retrieved up to twice. Preliminary precision and recall measures revealed that this approach gives better results.

The complexity of the G^{DS} Generator algorithm is rather simple and this is due to the T and close path (of looped relations) constrains. Fig. 7 shows the G^{DS} for Customers and Employees R^{DS} for $T=4$. Relations are annotated with their Cardinality and relationships with their Relative Cardinality (RC) and Reverse Relative Cardinality ($\overline{RC}$). The Appendix includes additional examples of G^{DS}s for the TPC-H database (Fig. 20).

4.3. OS generation

This module takes as input (1) the tuple set $DB^{id\text{-}kw}$ and (2) the corresponding $G^{DS}(\theta, \theta')$ graphs and produces an OS tree for each tuple in $DB^{id\text{-}kw}$. For the generation of an OS, a breadth-first traversal of the corresponding $G^{DS}(\theta, \theta')$ with the t^{DS} tuple as the initial root entry of the OS is employed. The algorithm employs relational operations with the general format $\sigma_{R_i.FK=t_j.PK}(R_i)$ for $R_j \rightarrow R_i$ (i.e. 1:M) relationships and $\sigma_{R_i.PK=t_j.FK}(R_i)$ for $R_j \leftarrow R_i$ (i.e. M:1) relationships for each node t_j of OS, where R_i is the current relation and t_j is a tuple that belongs to the parent relation R_j. The results of each execution of these operations are appended to the OS tree as child nodes of t_j.

4.4. OS ranking and presentation

This module takes as input the (1) set of OSs, (2) relations' Affinity scores and (3) global Importance scores of each comprising tuple (global Importance scores may be maintained in a database). It calculates the Importance of each OS and then prints them descending their Importance. In Section 8, we describe in detail the formulas and parameters we use for ranking OSs.

During presentation, we can merge tuples that are associated with a 1:1 relationship (Fig. 1).

5. Affinity calculation

In this section, we investigate the semantic of Affinity $Af_{R_i \rightarrow R^{DS}}$ (or simply Af_{R_i}) between a relation R_i to the R^{DS} in relational databases, define metrics that will assist to calculate Affinity and finally present an Affinity formula. The concept of Affinity has also been investigated and applied in schema clustering and summarization [34–37]. However, in those cases the Affinity between relations has a more general meaning and its semantics are slightly different from Affinity applied here. According to our knowledge the definition, investigation and quantification of Affinity as proposed in our work have not been attempted before.

5.1. The Affinity formula

For the quantification of Affinity the following metrics are considered.

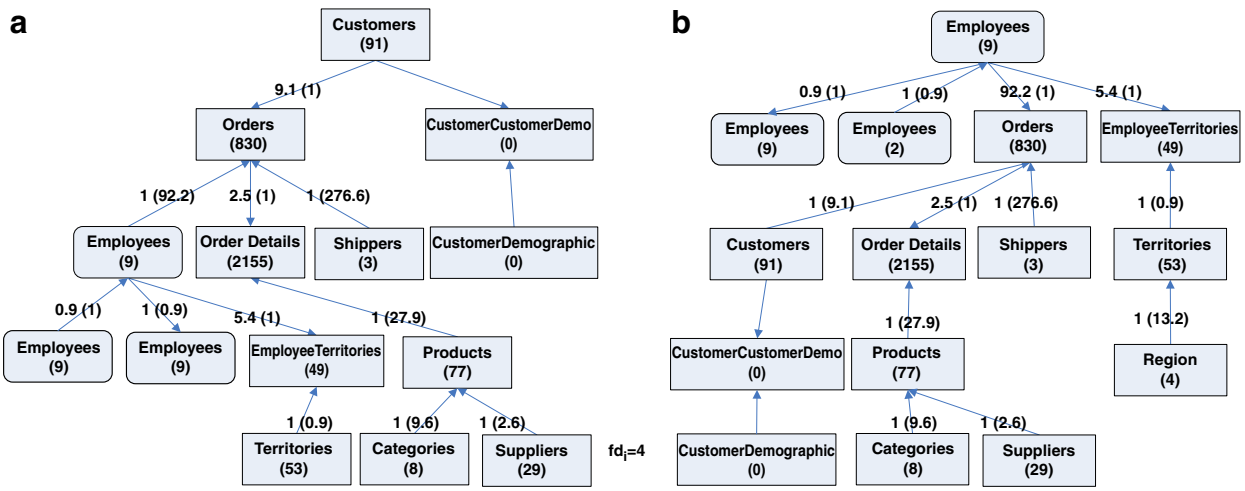

Fig. 7. Customers and Employees G^{DS}s.

5.1.1. Relations' distance

The primary metric for closeness between an R_i to the R^{DS} is their distance on the G^{DS} schema graph, namely the length (i.e. the number of relationships) of the path from the R_i to the R^{DS}. The shorter the distance is the bigger the Affinity between the two relations is. For instance, considering the Customers R^{DS} of our running example, we can easily see that Orders is semantically closer than EmployeeTerritories to the Customer R^{DS}.

If a path from R_i to the R^{DS} contains a physical relation (such as an M:N relation that consists only of the primary keys (PKs) of the participating relations) then we disregard the additional relationship. Let ld_i and fd_i be the logical and physical distances on a path respectively and |M:N| be the amount of M:N physical relations in the path then $ld_i = fd_i - |M:N|$. We see that Orders and CustomerDemographic, although they have different fd_i they have equal Affinity to R^{DS} since they have equal $ld_i (=1)$. Note that relations such as Orders, OrderDetails, although they act sometimes as intersection relations they are not considered as physical relations. (We only consider logical relations to include in an OS and assume that R_i is also a logical relation.).

5.1.2. Connectivity

A secondary metric of closeness between R_i to R^{DS} is R_i's connectivity on both the database schema, denoted as Schema Connectivity (Co_i) and the data-graph, denoted as Relative Cardinality ($RC_{i \to j}$). Let the Schema Connectivity Co_i of R_i be the amount of relationships R_i participates in. For instance, the relation CustomerDemographic has $Co_i = 1$ since it participates only in one relationship (i.e. with relation Customers) and Orders has $Co_i = 4$ since it participates in four relationships (i.e. with relations Customers, Employees, Shippers and OrderDetails). Therefore, the relation CustomerDemographic is closer to the Customer R^{DS} than Orders since they have common logical distance (i.e. $ld_i = 1$) but different Schema Connectivity (i.e. 1 against 4). This is a very interesting closeness property because we notice that CustomerDemographic exists only to provide additional information about Customers in contrast to Orders which is connected with several other relations. This is some kind of semantic dedication from CustomerDemographic to Customers.

Let $RC_{i \to j}$ represent the Relative Cardinality of R_i and R_j, namely the average number of tuples of R_i that is connected with each tuple from R_j (this concept was also used in [36,37]). However, our interest in Relative Cardinality of R_i is only limited towards its parent relation on the G^{DS}; therefore let RC_i be the $RC_{i \to j}$ where R_j is the parent relation on G^{DS}. Relative Cardinality can easily be calculated as follows: Assuming that all tuples are connected and there are no null values on keys: $RC_{i \to j} = |R_i| / |R_j|$ for 1:M relationships and $RC_{i \to j} = 1$ for M:1 relationships. Alternatively, $RC_{i \to j}$ will need to be calculated by counting the connections on the data-graph. For instance, the $RC_{Orders \to Employees}$ represents how many Order tuples are associated with a particular Employee, i.e. $|R_{Orders}| / |R_{Employees}| = 830/9 = 92$. Similarly, $RC_{EmployeeTerritories \to Employees} = 5.5$, therefore EmployeesTerritories semantically is closer than Orders to Employees. Another example is $RC_{Nation \to Customer}$ from the TPC-H database, an M:1 relationship, where only one tuple from Nation is associated with a particular Customer. Generally we can conclude that M:1 relationships give higher closeness.

It should also be noted that RC is not commutative. For instance, whilst $RC_{Orders \to Employees} = 92$ $RC_{Employees \to Orders} = 1$. Now, let Reverse Relative Cardinality, denoted as $\overline{RC_{i \to j}}$, be the reverse of $RC_{i \to j}$ (i.e. $\overline{RC_{i \to j}} = RC_{j \to i}$). Similarly to RC_i, let $\overline{RC_i}$ be the $RC_{i \to j}$ where R_j is the parent relation on G^{DS}. Apparently, 1:1 relationships have $\overline{RC_{i \to j}} = RC_{j \to i} = 1$ and therefore result to higher closeness. For instance, if we hypothetically assume that $\overline{RC_{Nation \to Customer}} = RC_{Nation \to Customer} = 1$ (i.e. each Customer belongs to a different Nation) and therefore their relationship is 1:1, we can easily infer that their Affinity is even closer.

5.1.3. Penalization of lateral data (from hub relations)

Analyzing further relational database schemata, we realize that 'hub' relations give lateral data to the R^{DS} [1,2,25]; such paths containing 'hubs' have the following structure: $R^{DS} \ldots \leftarrow R_{hub} \rightarrow R_2$. More precisely, the bigger the $RC_{Rhub \to R2}$ the more irrelevant data we get; i.e. in fact lateral information for the DS. For instance in the TPC-H database, $R_{Supplier}$ with $ld_i = 2$ from the $R_{Customers} \leftarrow R_{Nation} \rightarrow R_{Supplier}$ path will result to a big list of Suppliers coming from the same Nation as the Customer; something not directly relevant to Customer (but only laterally relevant). Furthermore, the cardinality of hub–child relations, e.g. R_2, in large databases is huge having a significant impact on f-score results. For this reason, lateral data extracted by such "spurious short-cuts" should be excluded from OSs. This can be penalized by increasing the impact of ld_i and Relative Cardinality.

5.1.4. Affinity Descriptor of R_i ($DAf(R_i)$)

Let the Affinity Descriptor of R_i to R^{DS} be a list of weighted metrics; namely, $DAf(R_i) = \{(m_1, w_1), (m_2, w_2), \ldots (m_n, w_n)\}$ where $\sum w_j = 1$. In this context, we define metrics $m_1 \ldots m_n$ as follows: $m_1 = f_1(ld_i)$, $m_2 = f_1(\log(10 * RC_i))$, $m_3 = f_1(\log(10 * \overline{RC_i}))$, $m_4 = f_1(Co_i)$. In the case of hub–child relations we penalize them as follows: $m_1 = f_1(ld_i * h)$ and $m_2 = f_1(RC_i)$ where $h = 1.6$ is a penalizing constant, and where $f_1(\alpha) = \begin{cases} (11-\alpha)/10 & 1 \le \alpha \le 10 \\ 1 & 0 < \alpha < 1 \\ 0 & \alpha > 10 \end{cases}$.

These metrics measure the distance (m_1), the Relative Cardinality (m_2), the Reverse Cardinality (m_3) and the Schema Connectivity (m_4) of R_i. The function f_1 aims to scale accordingly all metrics to 1 and thus to control their variation so the Affinity value does not skew excessively. For the same reason, since RC_i and $\overline{RC_i}$ values may vary significantly, we depress their values using logarithms. Other normalization methods, e.g. producing values in the range [0, 1], can also be investigated for depressing

RC_i and $\overline{RC_i}$. Note that the former approach produces absolute m_2 and m_3 metrics (since the absolute values are depressed) whereas the latter produces relative metrics (since normalized values are produced in the range [0, 1]).

Definition 2 (Affinity). *The Semantic of Affinity of R_i to R^{DS}, denoted as $Af_{R_i \to R^{DS}}$ (or simply Af_{R_i}), with respect to a schema and a database conforming to the schema, can be calculated with the following formula:*

$$Af_{R_i \to R^{DS}} = \sum_j m_j w_j \cdot Af_{R_{Parent} \to R^{DS}} \quad (2)$$

where j ranges over all metrics in $DAf(R_i)$, $Af_{R^{DS} \to R^{DS}} = 1$ and $Af_{R_{Parent} \to R^{DS}}$ is the Affinity of the R_i's Parent to R^{DS}. □

The above Affinity formula produces a score in the range [0, 1] and the R^{DS} is assigned the maximum Affinity score (i.e. 1). Also note that the proposed formula (2) may also produce an Affinity score 1 for a relation R_i (where $R_i \neq R^{DS}$), if all metrics of R_i have a score of 1; i.e. $ld_i = 1$, $RC_i = 1$, $\overline{RC_i} = 1$ and $Co_i = 1$. In other words, a relation R_i that participates only in one relationship (therefore $Co_i = 1$) and this unique relationship is 1:1 with the R^{DS} (therefore $ld_i = 1$ and $RC_i = \overline{RC_i} = 1$). Such a relation can be merged with R^{DS} without violating normalization rules; which means that R_i semantically is a part of R^{DS} but for some reason during the database design it was created as a separate relation.

5.2. Discussion

Experimental analysis of Affinity formula has revealed that usually any R_i with ld_i up to 3 gives good f-score results as long as lateral data from hub relations are excluded. In fact, the proposed Affinity formula has been tuned accordingly so as an Affinity threshold $\theta = 0.7$ in combination with the following metric weights: $w_1 = 0.5$, $w_2 = 0.4$, $w_3 = 0.05$ and $w_4 = 0.05$ will include only such relations for the particular two databases (but also for any other database). In general, DBAs and users can employ the technique proposed in Section 7 for the selection of a threshold closer to their needs (that estimates the OS size). Regarding metrics' weights, they should note that metrics m_1 (distance) and m_2 (Relative Cardinality) are the most influential for the Affinity estimation and thus give analogously more weights to w_1 and w_2 (e.g. $w_1 = 0.5$, $w_2 = 0.4$). Fig. 8 illustrates the Affinity results of relations for Employees, Customer, Order and Shipper R^{DS}s for the Northwind database based on these weight settings.

Furthermore, thorough user evaluation also revealed that user needs or even user perceptions regarding the comprehensiveness (and therefore abstraction) of an OS may vary. For instance for the Employees G^{DS}, some evaluators consider Product as relevant whilst others as irrelevant and detailed data. DBAs and users can deal with this requirement by controlling the level of comprehensiveness by adjusting θ (or the estimated OS size, discussed in Section 7) accordingly. For this reason, the proposed Affinity Ranking r_{Ri}^{Af} (or simply r_{Ri}) of each R_i and hence the overall Affinity Ranking Correctness (ARC) of relations was also investigated and compared with users' perceived Affinity rankings (Section 9.1.2).

6. Attribute Affinity calculation

In this section, we investigate the semantic of Affinity $AAf_{R_i.Aj \to R^{DS}}$ (or simply $AAf_{A_{ij}}$) between attributes A_j (e.g. a cluster of attributes denoted as A_j) of the relation R_i to R^{DS} and then propose an Attribute Affinity formula. The quantification of Attribute Affinity is useful in order to filter out attributes that are not really relevant to R^{DS} since not all attributes of all R_is are relevant. As explained in the Related work section, existing work in attribute filtering is completely inappropriate for our paradigm.

In order to achieve a good quantification of Attribute Affinity we need to investigate the semantics and the affinity properties of attributes in relational databases not only in general but also in the context of G^{DS}. In a relational database, a relation comprised of two sets of attributes: (1) the set of physical attributes denoted as A_F and (2) the set of logical attributes denoted as A_L. The former comprised of physical data required by the relational model; namely primary keys and foreign keys and can be further clustered into A_{PK} and A_{FK} sets respectively. Attributes in A_{FK} are not semantically interesting for users and therefore can be excluded from

R^{DS} / R_i	Employees			Customer	Order	Shipper
	ld_i, RC_i, $\overline{RC_i}$, Co_i	$m_1..m_4$	Af_{Ri}	$Af_{Ri}(r_{Ri})$	$Af_{Ri}(r_{Ri})$	$Af_{Ri}(r_{Ri})$
Employees	**R^{DS}**	**R^{DS}**	**1.00**	0.88 (3)	0.97 (4)	0.82 (4)
Employees (ReportsTo)	1, 1, 0.9, 4	1, 1, 1, 0.7	0.98	0.78 (5)	0.91 (5)	0.73 (5)
Employees (ReportedBy)	1, 0.9, 1, 4	1, 1, 1, 0.7	0.98	0.70(7)	0.85 (7)	0.66 (7)
Territories	1, 5.4, 1, 2	1, 0.9, 1, 0.9	0.96	0.55 (10)	0.66 (10)	0.51 (10)
Region	2, 1, 13.2, 1	0.9, 1, 0.88, 1	0.91	0.46 (11)	0.59 (11)	0.43 (11)
Order	1, 92.2, 1, 4	1, 0.8, 1, 0.7	0.90	0.94 (1)	**1 (R^{DS})**	0.89 (1)
Customer	2, 1, 9.1, 2	0.9, 1, 0.9, 0.9	0.85	**1 (R^{DS})**	0.99 (1)	0.83 (2)
Shipper	2, 1, 276.6, 1	0.9, 1, 0.75, 1	0.85	0.88 (2)	0.98 (2)	**1 (R^{DS})**
OrderDetails	2, 2.5, 1, 2	0.9, 0.96, 1, 0.9	0.84	0.88 (4)	0.97 (3)	0.82 (3)
Product	3, 1, 43.9, 4	0.8, 1, 0.83ρ0.8	0.74	0.77 (6)	0.91 (6)	0.73 (6)
Supplier	4, 1, 1.6, 1	0.7, 1, 0.9, 1	0.63	0.65 (8)	0.82 (8)	0.62 (8)
Categories	4, 1, 6.1, 1	0.7, 1, 0.92, 1	0.62	0.65 (9)	0.81 (9)	0.61 (9)
CustDemographics	3, null, null, 1	0.8, null, null, 1	Null	Null	Null	Null

Fig. 8. Affinity for four G^{DS}s from the Northwind database (Affinity Ranking).

the OS. On the other hand, information described by attributes in A_L is logically meaningful to users and need to be considered for inclusion in OSs. The clustering of attributes into the above sets can easily be achieved since primary and foreign keys are known.

The challenge now is how to assign an Affinity score to the attributes of A_L; for this reason, the following approach is proposed. A_L is further clustered into disjoint sets of attributes $A_1,\ldots,A_n$ and then each cluster A_j is assigned an Attribute Affinity score ($AAf_{A_{ij}}$). For instance, a General (Universal) Clustering Approach of attributes that aims to address any arbitrary database is proposed as follows (Fig. 9): (1) the A_N cluster comprising of Naming attributes, such as Name, Surname, First Name, Company Name, etc.; (2) the A_G cluster comprising of general attributes, such as Address, Date of birth, etc.; and finally (3) the A_C cluster comprising of attributes including comments, descriptions, multimedia data, etc. Section 6.1 presents the Attribute Affinity formula, Section 6.2 describes the proposed attribute clustering technique whilst Section 6.3 discusses alternative clustering approaches.

6.1. Attribute Affinity formula

In this context, the following metrics are considered for the calculation of Attribute Affinity.

6.1.1. Relations' Affinity

The primary metric of Attributes $R_i.A_j$ Affinity to R^{DS} is the Affinity of R_i. This is based on the intuitive observation that the more Af_{R_i} decreases the less attributes of R_i are really relevant to the R^{DS}. In many cases, although an R_i is included in a $G^{DS}(\theta)$, only its naming (e.g. Name, Surname, etc.) attributes are in fact relevant. For example for the Northwind Employees G^{DS}, a user will be interested in more attributes included in Orders relation than in Products relation since $Af_{ROrder} > Af_{RProduct}$. More precisely, a user would expect to see attributes OrderDate, RequiredDate, ShipName, etc. from Orders whilst the attribute ProductName from Products will be adequate.

6.1.2. Attribute cluster's Affinity ranking (d_{Aj})

A secondary metric is that A_L clusters can intuitively be ranked by their Affinity to the R^{DS}. For example for the proposed General Clustering Approach, clusters have the following intuitive Affinity Ranking: A_N, A_G, and A_C. In other words, if we had to prune out some attributes from a relation (e.g. Employees; Fig. 9(b)), we would certainly prune out first A_C, then A_G and lastly A_N. This order represents in some way the ranking of abstractness of attributes.

Therefore an attribute cluster's Affinity Ranking (d_{Aj}) is assigned for each cluster; e.g. for A_N, A_G, and A_C the d_{Aj} values 1, 2 and 3 respectively (other continuous ranking values may also be considered). Because some of these clusters are sometimes empty, a more dynamic way of assigning d_{Aj} is required in order to increase a cluster's ranking when all its predecessor clusters are empty. For this reason, the corresponding $d(A_j)$ function is introduced. As shown in Fig. 9(a), $d_{AG}=1$ if A_N is empty and $d_{AC}=1$ if both A_N and A_G are empty. In other words, if a relation does not include an A_N cluster then d_{AG} is set to 1 instead of 2; similarly if a relation does not include neither A_N nor A_G clusters then d_{AC} is set to 1 instead of 3.

a

Attributes Cluster	d(A_j)	Clustering Rules
A_N	$d_{AN}=1$	Linguistic and Data type matching of "NAME"
A_G	IF A_N==Ø THEN $d_{AG}=1$ ELSE $d_{AG}=2$	$A_G=A_L-A_C-A_N$
A_C	IF A_N==A_G==Ø THEN $d_{AC}=1$ ELSE $d_{AC}=3$	Memo, Text with size>75, multimedia Attributes, etc.

b

Employees

A_F	A_{PK}	EmployeeID	N(4)
	A_{FK}	ReportsTo	N(4)
A_L	A_N	LastName	T(20)
		FirstName	T(10)
	A_G	Title	T(30)
		TitleOfCourtesy	T(25)
		BirthDate	Date
		HireDate	Date
		Address	T(60)
		City	T(15)
		Region	T(15)
		PostalCode	T(10)
		Country	T(15)
		HomePhone	T(24)
		Extension	T(4)
	A_C	Photo	OLE
		Notes	Memo
		PhotoPath	T(255)

Order Details

A_F	A_{PK}	OrderID	N(4)
	A_{FK}	ProductID	N(4)
A_L	A_G	UnitPrice	N(8)
		Quantity	N(2)
		Discount	N(4)

Fig. 9. (a) The General Clustering Affinity Approach. (b) Attribute clusters (with each attribute's name and data type (size)).

Definition 3 (Attribute Affinity). *The semantic of Affinity of a cluster of attributes A_j of a relation R_i to R^{DS}, denoted as $AAf_{R_i.A_j \to R^{DS}}$ (or simply $AAf_{A_{ij}}$), can be calculated with the following formula:*

$$AAf_{R_i.A_j \to R^{DS}} = f_2\left(d_{A_j}\right) \cdot Af_{R_i \to R^{DS}} \tag{3}$$

where the function f_2 is defined as $f_2(\alpha) = 1 - w \cdot (\alpha - 1)/10$, w is a tuning weight, $Af_{R_i \to R^{DS}}$ is the Affinity of R_i to R^{DS} and d_{A_j} is the Cluster's Affinity Ranking defined by the function $d(A_j)$. □

The function f_2 (similarly to f_1) aims to scale accordingly d_{Aj} to 1. On the other hand, the tuning weight w is necessary for the definition of a threshold value θ'; for instance for $\theta' = 0.7$, w is set to 1.5. The above formula produces a score in the range [0, 1]. Note that according to the above formula only the R^{DS} (with $Af_{R^{DS}} = 1$) can have all of its clusters with $AAf_{A_{ij}} \geq 0.70$ and therefore all clusters filtered in the OS for $\theta' = 0.7$, whilst for the rest R_is, for example, always $AAf_{AiC} < 0.70$ and therefore A_C is always filtered out for $\theta' = 0.7$. Fig. 10 illustrates the Attribute Affinity results for the Northwind Employees R^{DS} using the same weight settings as those used in Fig. 8.

6.2. Attribute clustering (the General Clustering Approach)

According to the proposed clustering approach, each cluster A_j is associated with a set of rules that can be used for the clustering of A_j. Schema matching [40] techniques, which combine linguistic and attributes' data types matching can be used to cluster attributes based on these clustering rules.

The clustering rules of the General Clustering Approach are summarized in Fig. 9(a) and more precisely its clustering is achieved as follows. Firstly, the A_C is created by considering the data types of attributes, namely by including attributes of text with big size (e.g. >75 characters; i.e. comments or descriptions, etc.), multimedia content, etc. Secondly, the A_N is created by combining linguistic and data type matching. For this purpose, a NamingDictionary containing 21 synonym and hyponym words (together with their stems) for the words "Name" and "Surname" (e.g. Surname, Family name, Patronymic, etc.) was generated from the WordNet Dictionary [43]. Finally A_G is created by deducting A_N and A_C from the A_G set (namely $A_G = A_L - A_C - A_N$). These problems have already been well addressed by schema and ontology matching techniques [39–41] and it is beyond the scope of this paper to examine them in further detail.

6.3. Discussion

The clustering of attributes can also be described by ontologies; the use of ontologies will enhance clustering quality and flexibility (e.g. more and finer clusters). The proposed General Clustering Approach of attributes (i.e. A_N, A_G, and A_C clusters) is general (universal) and aims to address any domain theme of databases; ranging from trading (e.g. Northwind and TPC-H), bibliographic (e.g. DBLP [44], IMDB [45]), university (e.g. [46]) databases, etc. Certainly, the consideration of a particular domain theme of a database during attribute clustering in combination with the use of ontologies describing such domains will improve results quality; since it will facilitate finer and more accurate clustering. For example, in a bibliographic database, the naming attribute of a publication (e.g. a paper [44,45]) is usually called a "Title" instead of a "Name". In our work, this bibliographic domain idiom for a naming attribute will be missed out and clustered as A_G instead. In addition, the automated identification of the database theme domain (e.g. trading, bibliographic, etc.) and their relations' themes (e.g. relations Employees describe a physical person whilst Shippers and Customers describe Companies, etc.) in combination with predefined ontologies will further improve these facilities. These are plans for future work.

R^{DS} / R_i(Attr. Clust.)	Employees		
	$f_2(d_{Aj})$	Af_{Ri}	$AAf_{A_{ij}}$
Employees (A_N, A_G, A_C)	**R^{DS}**	**1**	1, 0.85, 0.7
Employees (ReportsTo) (A_N, A_G, A_C)	1, 0.85, 0.7	0.98	0.99, 0.84, 0.69
Employees (ReportedBy) (A_N, A_G, A_C)	1, 0.85, 0.7	0.98	0.99, 0.84, 0.69
Territories (A_G)	1	0.96	0.97
Region (A_G)	1	0.91	0.91
Order (A_G)	1	0.90	0.90
Customer (A_N, A_G)	1, 0.85	0.85	0.85, 0.72
Shipper (A_N, A_G)	1, 0.85	0.85	0.85, 0.72
OrderDetails (A_G)	1	0.84	0.84
Product (A_N, A_G)	1, 0.85	0.74	0.74, 0.63

Fig. 10. Attribute Affinity for Employees G^{DS} (Northwind database).

7. Estimation of thresholds

The selection of appropriate thresholds according to the user needs may be a difficult task. As with small thresholds, we may obtain extremely large OSs or alternatively with large thresholds we may obtain very small OSs. We propose a technique that facilitates users to select more quickly the suitable Affinity and Attribute Affinity thresholds by estimating the size of OSs for the particular thresholds. This approach is not only more convenient for users but also simpler to comprehend.

In order to estimate an Affinity threshold, we estimate the size of an OS in tuples using the Relative Cardinalities of relations in $G^{DS}(\theta)$. Let $OS(\theta)$ be the OS generated with $G^{DS}(\theta)$, $|OS(\theta)|$ be the amount of tuples of $OS(\theta)$, $R(OS)_i$ be any relation in $G^{DS}(\theta)$, $|R(OS)_i|$ be the amount of tuples from the relation R_i in the $OS(\theta)$ and $RC_{i\to j}$ be the Relative Cardinality between R_i and R_j (as defined in Section 5). Then, the estimated $|OS(\theta)|$ can be calculated with $|OS(\theta)| = \sum|R(OS)_i|$ where $|R(OS)_i| = |R(OS)_j| * RC_{i\to j}$ and R_j is the parent relation of R_i. Fig. 11(a) gives us an indication of estimated $|OS(\theta)|$s for the Employees G^{DS}. For instance $|OS(0.95)|$ is calculated as follows. $G^{DS}(0.95)$ includes Employees (ReportsTo), Empoloyees (Reported By) and Territories. Their $|R(OS)_i|$s are 0.9 (=1*0.9), 1 (=1*1) and 5.4 (=1*5.4) respectively; since they all have a common $|R(OS)_j| = 1$ (i.e. the t^{DS}) and respective $RC_{i\to j}$s are 0.9, 1 and 5.4. Therefore, estimated $|OS(0.95)| = 1 + 0.9 + 5.4 = 7.3$. Finally, we round the estimated numbers of $|OS(\theta)|$ as to avoid any confusions of users.

Similarly, we can estimate an Attribute Affinity threshold. Namely, for a given $G^{DS}(\theta, \theta')$ we estimate the size of an OS in terms of attribute clusters (alternatively, we can estimate the amount of attributes or bytes). Let $OS(\theta, \theta')$ be the OS generated with $G^{DS}(\theta, \theta')$, $|OS(\theta, \theta')|$ be the total amount of attribute clusters of the $OS(\theta, \theta')$ and $|R(\theta')_i|$ be the amount of attribute clusters in the relation R_i (i.e. with $AAf_{Aij} > \theta'$). Then, the estimated $|OS(\theta, \theta')|$ can be calculated with $|OS(\theta, \theta')| = \sum(|R(OS)_i| * |R(\theta')_i|)$, where $|R(OS)_i|$ is calculated as above. Fig. 11(b) gives an indication of estimated $|OS(0.70, \theta')|$s for the Employees G^{DS}.

8. Ranking of OSs

The result of a query may comprise a set of OSs, e.g. the result of the query "Peacock" on the dataset of Fig. 3 comprises two OSs (Fig. 5). In some cases the set of OSs may be very large and thus an effective ranking of OSs is necessary. Recall that existing ranking semantics of traditional R-KwS are inappropriate for OS ranking as they rank higher results with small size [1–3]. In contrast, an OS containing many well connected tuples should have certainly greater Importance. For instance, in the aforementioned example, the Employee Margaret Peacock who has served two orders is more important than Employee Janet Peacock who has served only one. Thus their corresponding OSs should be ranked accordingly.

In order to rank OSs we estimate an Importance score for each OS, denoted as $Im(OS)$. We treat each OS as a document comprising of |OS| tuples where each tuple is associated with a local Importance score denoted as $Im(OS, t_i)$. We propose that the Importance of an OS should combine:

(1) *The local Importance of each tuple ($Im(OS, t_i)$) in the OS*. More precisely, the more important tuples an OS contains the higher Importance it has. Hence, the summation of the local Importance of tuples should be considered.
(2) *The size of the OS ($|OS|$)*. Because solely considering the summation of local Importance of comprising tuples will favor large OSs, we also penalize large OSs by normalizing with their log(|OS|). We use log to depress the impact of the size. Also note that excluding the log will result to the average of $Im(OS, t_i)$ which is not desired as it is misleading.

Therefore, the following combining formula can be used for estimating the Importance of an OS:

$$Im(OS) = \frac{\sum \mathrm{Im}(OS, t_i)}{log(|OS|) + 1} \tag{4}$$

where the local Importance of each tuple t_i in an OS (namely $Im(OS, t_i)$) can be calculated with:

$$Im(OS, t_i) = Im(t_i) * Af(t_i) \tag{5}$$

where $Im(t_i)$ is the global Importance of t_i in the database. We can use ranking systems that consider authority flow through relationships to calculate global Importance scores, such as PageRank [30], ObjectRank [31], ValueRank [33], etc. $Af(t_i)$ is the Affinity of t_i to the t^{DS}; namely the Affinity of the corresponding relation it belongs to, to R^{DS} (namely $Af(R_i)$ and can cheaply be obtained from the G^{DS}). The product of $Im(t_i)$ with $Af(t_i)$ aims to reduce the global Importance contribution of each tuple accordingly to the $Im(OS, t_i)$ (since $Af(t_i) \leq 1$). This was necessary as discrimination of tuples with different Affinity is considered crucial.

a

Θ	Estimated |OS(θ)|
0.95	7
0.90	105
0.85	289
0.80	520
0.75	520
0.70	750

b

θ'	Estimated |OS(0.70,θ')|
0.95	8
0.90	106
0.85	291
0.80	524
0.75	524
0.70	940

Fig. 11. (a) Estimated $|OS(\theta)|$ in tuples. (b) Estimated $|OS(\theta, \theta')|$ in attributes clusters.

9. Experimental evaluation

The proposed keyword search paradigm was evaluated with two databases, namely TPC-H and Northwind. The size of the TPC-H database is 1 GB ($\approx 8.6*10^6$ tuples; Scale Factor 1) and the size of the Northwind database is 3.7 MB ($\approx 3.3*10^3$ tuples). For the experiments, we used Java, MySQL and a PC with AMD Phenom 9650 (Quad-Core) processor, 2.3 GHz and 4 GB of main memory. The DBMS maximum memory was set to 80 MB. We firstly present the search quality, then the performance and finally comparative results with other existing work.

9.1. OS quality (evaluation methodology)

In this section, we investigate the quality of OS and $G^{DS}(\theta)$ in terms of accuracy and completeness, where the correct results were provided by humans. For this reason, a thorough survey was conducted where ten lecturers and students from our department and also from the School of Computer Science of the University of Manchester participated. The volunteers had no involvement in this work and they were firstly introduced to the new keyword search paradigm and the semantics of OS results.

The quality of OSs (and thus $G^{DS}(\theta)$s) is measured by standard precision, recall and f-score (P/R/F) metrics. Precision measures accuracy (i.e. the fraction of correct results), recall measures completeness (i.e. the fraction of all correct results actually captured), and f-score represents their weighted harmonic mean.

9.1.1. Quality of relations filtering (P/R/F results)

For this evaluation task, the participants were given twelve G^{DS}s (i.e. six from the TCP-H and six from the Northwind database) and were asked to define manually their own G^{DS}s, denoted as $G^{DS}(h)$s. The participants did not provide completely identical $G^{DS}(h)$s since their $G^{DS}(h)$s had some minor differences (this was anticipated anyway, since this is a subjective task). For example, a comprehensive enquiry about a Supplier may demand information such as Products, Categories and Orders; on the other hand an abstract enquiry may be limited only to the Products the particular Supplier supplies. For this reason, in order to obtain common comparative criteria, we requested from the participants comprehensive but at the same good $G^{DS}(h)$s.

The following experiments also investigate the impact of metrics' weights and thresholds. The following weight schemes and thresholds were investigated: $\mathbf{W_1 = <0.5, 0.4, 0.05, 0.05>}$, $W_2 = <0.4, 0.3, 0.15, 0.15>$, $W_3 = <0.6, 0.3, 0.05, 0.05>$, $W_4 = <0.7, 0.2, 0.05, 0.05>$, $W_5 = <0.4, 0.5, 0.05, 0.05>$ and $W_6 = <0.25, 0.25, 0.25, 0.25>$ and $\mathbf{\theta_1 = 0.70}$, $\theta_2 = 0.65$, $\theta_3 = 0.75$, and $\theta_4 = 0.80$, with W_1 and θ_1 as the initial setting.

Fig. 12(a) summarizes the quality of the $G^{DS}(\theta)$ produced with initial setting: θ_1 and W_1 in terms of P/R/F results. More precisely, it depicts the average of P/R/F results of $G^{DS}(\theta)$ and each $G^{DS}(h)$ (namely by considering relations that are included or

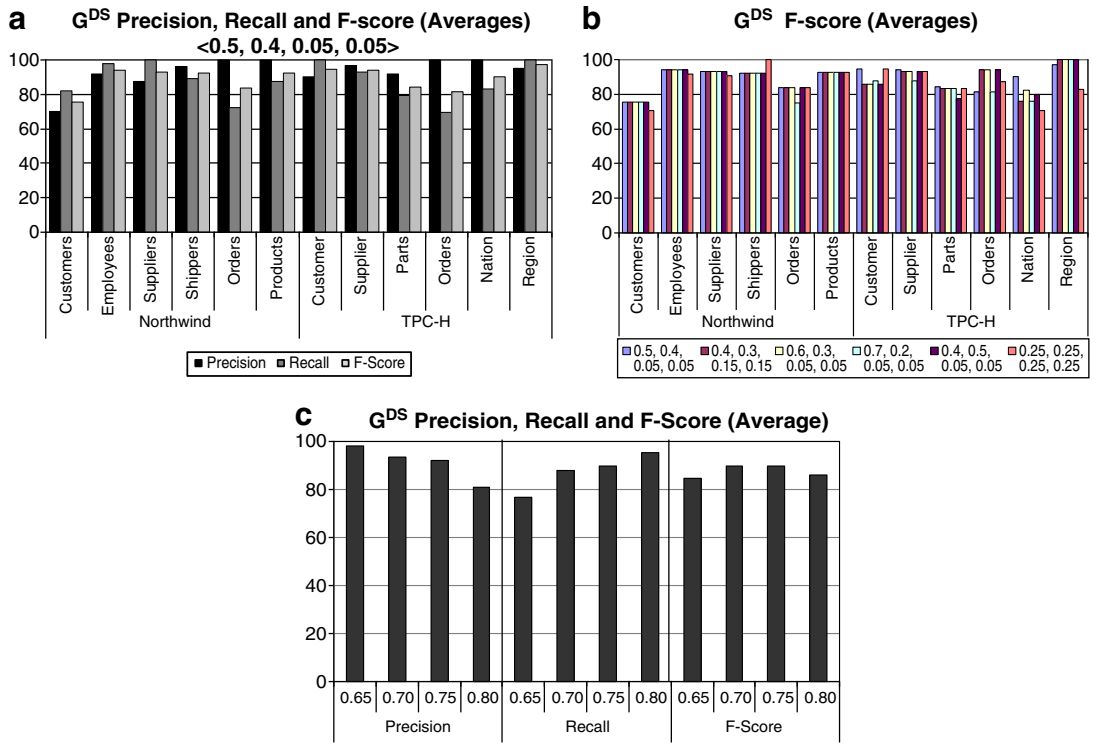

Fig. 12. G^{DS} quality (P/R/F results).

excluded in $G^{DS}(\theta)$). Fig. 12(b) summarizes the impact of descriptors weights by presenting f-scores of $G^{DS}(\theta)$. The initial setting (W_1) gives the best average of f-score results, i.e. 89.44% whist W_5 the worst i.e. 86.77%. Note that the proposed Affinity formula was designed based on the initial weight scheme. Evidently, there is not a significant impact of the weight settings on P/R/F of G^{DS}s; since f-score range is [86.77, 89.44]. Whilst, Fig. 12(c) depicts the impact of thresholds; thresholds 0.70 and 0.75 produce the best f-scores ($\approx$89%). As expected whilst θ increases, precision decreases whereas recall increases.

Fig. 13(a) summarizes the quality of OSs produced by a $G^{DS}(\theta)$ (with θ_1 and W_1). More precisely, it depicts the average of P/R/F results of OSs produced by each $G^{DS}(\theta)$ for 10 randomly selected t^{DS}s (or for all t^{DS}s if $|R^{DS}|<10$) against OSs produced by the corresponding $G^{DS}(h)$. The average size in tuples of the randomly selected OSs per G^{DS} is also indicated (below the G^{DS} name). In summary, f-scores range from 96% (i.e. for Northwind Shippers G^{DS}) to 56% (i.e. for TPC-H Nation G^{DS}) and the average P/R/F scores are 90.3%, 83.9% and 82.2% respectively. The poor results for Nation OSs were to some extent expected because of the large Relative Cardinality involved. In summary, these are very good P/R/F results.

Fig. 13(b) summarizes the impact of different descriptor weight schemes on OS P/R/F scores by depicting f-scores per G^{DS}. As shown, the impact on the Northwind G^{DS}s is insignificant. On the other hand, on the TPC-H database some weight schemes (e.g. W_4 and W_6) result to very poor scores (e.g. for Nation and Region G^{DS}s). Fig. 13(c) shows the average P/R/F scores for the two databases per weight scheme. It verifies again that the impact is significant on the TCP-H database; this is due to the large Relative Cardinality involved. In conclusion, W_1, W_2, W_3, and W_5 give very good results with averages of f-score ranging from 81.1% to 83.8%. These weight schemes also reveal that m_1 and m_2 have the most significant effect and that if their corresponding weights belong to the range [0.4, 0.6] can yield to very good results.

Comparing results from Figs. 12 and 13, we observe that W_1 is the safest weighting scheme to choose since it performs well on both cases for both databases. On the G^{DS} level it returns f-score = 89.44% (i.e. the best P/R/F results of all), whilst on the OS level it returns f-score = 82.20% (i.e. an insignificant difference from the best f-score = 83.8% produced by W_5). This is due to the Relative Cardinality effect on the P/R/F results of OSs. Certainly, the P/R/F results on the OS level are significantly more important than the ones on a G^{DS} level. In conclusion, a DBA must investigate for the suitability of the proposed initial weighting scheme for each particular dataset.

9.1.2. Relation's Affinity Ranking Correctness (ARC)

During the first task of the evaluation (Section 9.1.1), it was revealed that a critical factor of the variation of human's perception of a good summary was because of the different perception regarding the abstraction or comprehensiveness of an OS. Based on this observation, we realized that another accurate measurement of the quality of our approach is the correctness of the Affinity calculation and more precisely the ranking of relations based on their Affinity. Since, users can experiment with abstraction or comprehensiveness by adjusting (by either increasing or decreasing) gradually the threshold accordingly. For instance, if a user believes that the OSs produced are very over-detailed then he/she can increase θ or alternatively if he/she believes that OSs are

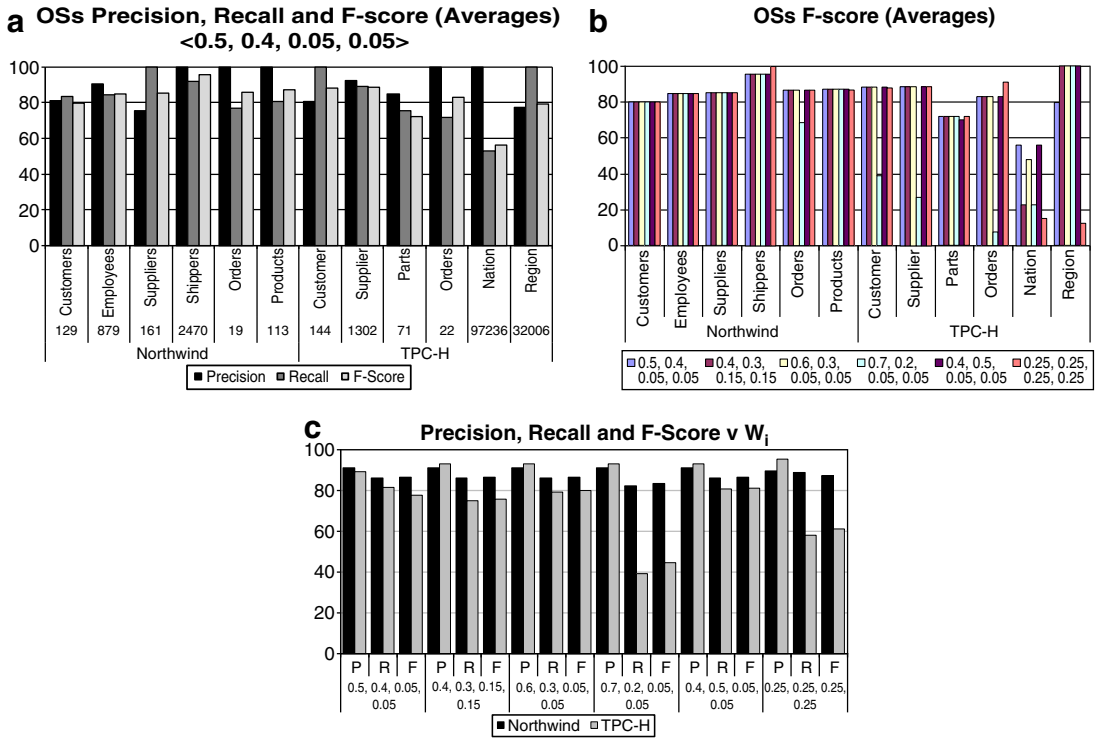

Fig. 13. OS quality (P/R/F results).

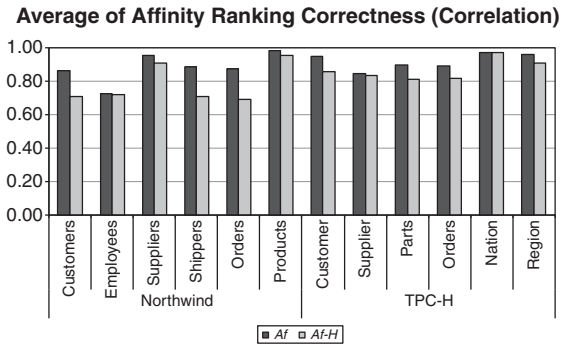

Fig. 14. The average of the Affinity Ranking Correctness (produced by the proposed (*Af*) and [36] (*Af-H*) Affinity formulas).

very abstract and additional information about the DS is retrievable then θ can be decreased. For this reason, we also asked our volunteers for each G^{DS} to rank relations based on their Affinity to the R^{DS}.

The results in Fig. 14 denoted with *Af* depict the average of Affinity Ranking Correctness (ARC) of the proposed Affinity formula (produced using the initial settings) against evaluators' rankings. (Whereas comparative results, denoted with *Af-H*, depict ARC produced by Affinity formula (1) [36] and are discussed in Subsection 9.3.) We use the correlations to measure ARC. More precisely, we calculate the correlation between the proposed Affinity Ranking (i.e. r_{Ri}^{Af}) for each R_i in G^{DS} and the evaluator's Affinity Ranking (denoted as r_{Ri}^{h}). In conclusion, the average results were very encouraging with values ranging from 0.72 (for Employees G^{DS}) to 0.97 (for Products G^{DS}) both from the Northwind database. Note that the maximum correlation is 1 and that is achieved when the two rankings are identical.

9.1.3. Quality of attribute filtering (P/R/F results)

The following P/R/F results compare attribute filtering proposed by $G^{DS}(\theta, \theta')$ against evaluators' $G^{DS}(\theta, h)$ on both G^{DS} and OS levels. For comparative reasons both $G^{DS}(\theta, \theta')$ and $G^{DS}(\theta, h)$ were comprised of the same set of relations, therefore they were both generated with a common Affinity threshold $\theta = 0.70$ and weighting settings (namely W_1). The Attribute Affinity threshold and tuning weight *w* were set to 0.70 and 1.5 respectively; attribute clusters included in each relation of $G^{DS}(\theta, h)$ were defined by evaluators.

Fig. 15(a) depicts the average of P/R/F results of $G^{DS}(\theta, \theta')$ against each evaluator's $G^{DS}(\theta, h)$ (namely by considering attribute clusters of each relation in $G^{DS}(\theta, \theta')$ and $G^{DS}(\theta, h)$). f-Scores range from 85.7% to 100% and the average P/R/F scores are 88.0%, 96.0% and 91.5% respectively. On the other hand, Fig. 15(b) summarizes the average P/R/F results of OSs produced by each $G^{DS}(\theta, \theta')$ for the 10 selected t^{DS}s (same t^{DS}s used in previous experiments) against OSs produced by the corresponding evaluator's $G^{DS}(\theta, h)$. In summary, f-scores range from 86.3% to 100% and the average P/R/F scores are 93.3%, 98.6% and 95.6% respectively. In summary, these are very good P/R/F results.

9.2. Performance evaluation

In this subsection, we present performance results of the proposed system. Firstly, we investigate the space required by the OS contents and the space savings achieved by employing attribute filtering. Fig. 16(a) summarizes the size of OSs (in bytes) with and without attribute filtering for the experiments of Section 9.1.3. These results exclude the structural space required for the OS tree construction. Evidently for the two benchmarks, the space savings after filtering attributes are very significant; more precisely, size reductions range from 86% to 8.7% (i.e. for TPC-H Region with a reduction from 7,117,106 to 928,326 bytes and Northwind Suppliers with a reduction from 18,108 to 16,520 bytes respectively), whilst the average reduction is 50%. These significant

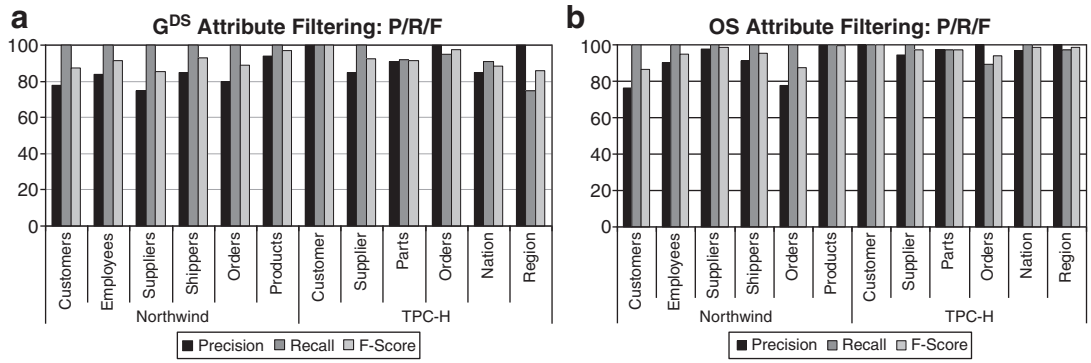

Fig. 15. Attribute filtering quality.

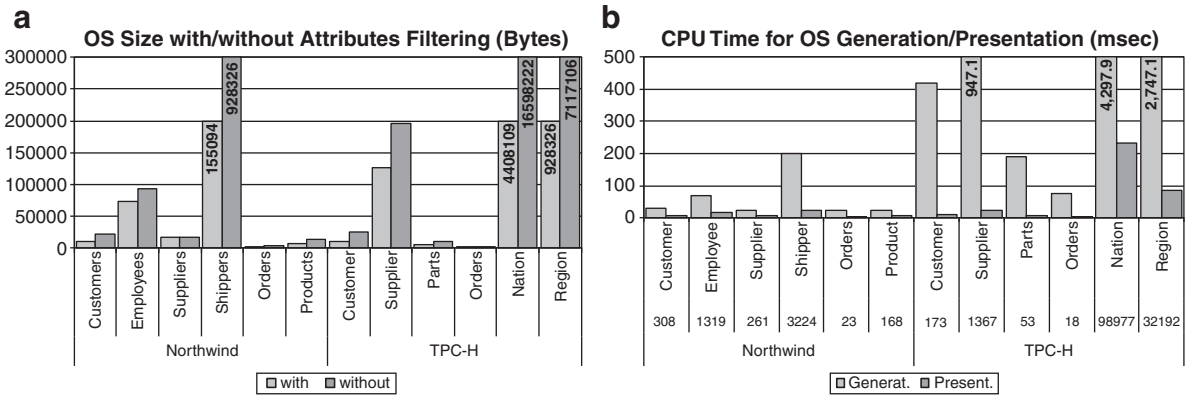

Fig. 16. OSs' size and CPU time.

reductions are due to the extensive repetitions of relations such as Customers and Suppliers in large OSs such as Region and Nation. These relations include big in size A_C clusters and thus their exclusion impacts positively the OS's size.

Secondly, we investigate the time required by the various modules of the system. More precisely, the average time for the generation of a G^{DS} graph and the calculation of the Affinity and Attribute Affinity of a G^{DS} are 16 ms and 6 ms respectively. We are very satisfied with these times (and therefore do not elaborate discussions) because these tasks can be pre-computed (i.e. one time tasks) and then reuse their output during keyword search. Note that both these tasks are cheap to execute because their input, i.e. the schema graph, is small and also memory based.

The CPU execution time required for the generation and presentation of OSs was more thoroughly investigated. As expected, experimentation revealed that attribute filtering did not have any important impact on the CPU time for the OS generation. Therefore, the following results (Fig. 16(b)) do not present any comparative results but only an indication of the time required for the generation of an OS for each G^{DS}. The size in terms of tuples for each OS is also indicated (below the G^{DS} name). The results revealed that both (1) OS size and (2) database size affect significantly CPU time. For instance for the Northwind database, the particular OS for Customers required less time than the OS for Shippers, since OS sizes are 308 and 3224 tuples respectively. Similarly, comparing the time for the OS generation for the particular Northwind Shippers (with size 3224) with the time for the TPC-H Customers OS (with size only 173 tuples), we observe that the TPC-H Customers OS requires double time.

Fig. 16(b) also presents comparative results between OS generation and presentation. The results revealed that OS generation is much more demanding than OS presentation and this is because of the big I/O operations required. The results also revealed that only the OS size affected CPU time for the OS presentation (in contrast to OS generation where the database size also affected).

An interesting observation is the extremely large times required by TPC-H Nation, Region and Shipper OSs (especially during OS generation), i.e. due to the huge size of the corresponding OSs. In such cases, the users can very easily detect potentially demanding OSs by using the threshold estimation techniques (discussed in Section 7) so they can act accordingly.

We run each OS generation experiment 20 times, excluded the worst and best measurements and then calculated the average of the remaining 18 measurements. In all experiments cold cache memory was used.

9.3. Comparative results

According to our knowledge there is not any other existing work directly relevant to OSs so we can compare our results with. Nevertheless we discuss and compare where possible our results with existing related work.

9.3.1. R-KwS

Traditional R-KwS query results are not comparable with our work as they differ fundamentally both semantically and technically. More precisely, R-KwSs produce results describing the associations of the given keywords (e.g. “Leverling Peacock”) whilst OSs produce summaries of all information about particular DSs (e.g. “Janet Peacock”).

9.3.2. Précis

Précis queries have some similarities with our work as they also summarize information, although they have slightly different semantics. For instance, in OS queries, given an identifying set of keywords (e.g. “Janet Peacock”), an OS is generated for each DS; whereas in précis queries, given a more general set of keywords (e.g. “Janet Peacock” or “Peacock sales”), a subset (summary) of the whole database is generated. Using the same evaluators, we compared OSs and précis results in terms of their friendliness and ease of use when trying to obtain a thorough report of information about a DS (which is the objective of the proposed keyword search paradigm).

More precisely, we presented to our evaluators results in the OS and précis formats and asked them to score them in the scale 1–10 and also to justify their scores. We emphasized to them the objective of this keyword search paradigm (which is the generation of thorough personalized reports of DSs) and we asked them to score the friendliness and ease of use of the contents of

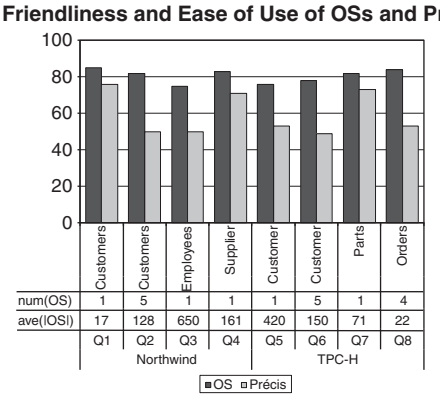

Fig. 17. Comparative user evaluation of OSs and précis queries (friendliness and ease of use).

the results. We presented the queries, the corresponding results and their format in a randomized order to the evaluators so we do not make them biased in favor to one method. We presented to the evaluators results from various queries; ranging the size and amount of OSs per query, the G^{DS} and the corresponding database. The results for each query, in either format, contain the same contents as correctness of results is not what we investigate but their friendliness and ease of use. E.g. similarly to the results for the "Peacock" keyword, presented in the two formats in Fig. 5(a) and (c) respectively that correspond to the same contents.

Fig. 17 depicts the average of the comparative evaluation results of our survey, where num(OS) represent the amount of OSs per query result and ave(|OS|) their average in size. In general, the evaluators expressed more preference to OSs' format. More particularly, for large in size and amount of OS results (e.g. Q2, Q3, Q5, Q6, and Q8), they strongly preferred OSs (e.g. in Q2 the difference of preference of the two approaches is 32%). The users explained that they preferred OSs mainly for presentation reasons, since OS format replicate information not only inside each OS but also in all OSs per query result and this made OSs more self-contained and thus easier to comprehend and use. In contrast, in the précis format (which is a subset of the database); it was difficult for the evaluators to connect the data and make sense what information is contained for each DS. For small results with fewer and smaller OSs the difference of preference was smaller (e.g. Q1, Q4 and Q7; more precisely in Q1 the difference of preference is only 9%). This is because for smaller results précis were not that difficult to use in comparison to OSs. In general, regardless of the method, evaluators expressed more preference to smaller in size OS and précis results in comparison to larger ones. In summary, the evaluation revealed that OS results are always more preferable by users than précis for all queries in terms of friendliness and ease of use. This was anticipated as OSs and précis have slightly different semantics and this is the reason that précis were not that preferred by the evaluators for our purpose (i.e. for summaries of data of DSs).

In terms of quality correctness, précis results are produced by manually set weights, thus there is not any meaning comparing précis results (since their contents is selected manually) against OSs. However, assuming that we had an algorithm that transforms a précis query result (i.e. the database subset) to an OS format, then this "Précis OS" will correspond to the OS produced by the $G^{DS}(h)$ proposed by users or DBAs.

9.3.3. Affinity

In schema summarization the concept of Affinity is also investigated and quantified, e.g. the Affinity formula (1) [36]. Although this formula is not completely appropriate for our paradigm, we try to apply it in order to calculate an Affinity for each R_i for comparative reasons. Yet, it is not very easy to use it in order to generate OSs and then compare OS quality since a corresponding threshold is needed. The selection of an effective common threshold for all G^{DS}s of a database is very difficult as Affinity values were very skewed (with a fast decline; this is because metrics were not normalized). For instance, for the TPC-H Customer G^{DS}, ordered Affinity values were 1, 0.5, 0.1, 0.013, 0.008, etc. whereas for Parts G^{DS} 0.25, 0.12, 0.083, 0.062, etc. On the contrary, our formula was designed to support more easily the selection of a common threshold.

ARC is a fair comparative metric since it considers the ranking of relations according to their Affinity (regardless of any thresholds). In Fig. 14 we also present ARC results created with Affinity [36]. In summary, our results are better for all twelve G^{DS}s. In average, the difference of the correlation coefficient of the two affinity rankings is 0.10 for the twelve G^{DS}s and more precisely the difference for the Northwind database is 0.09 whilst for the TPC-H database is 0.05 (recall that the maximum correlation is 1). This difference was expected since Affinity in [36] ignores Affinity properties specific to relational databases, such as hub relations, schema connectivity, etc. Note that even this small difference of ARC is enough to yield very poor P/R/F results (especially on large datasets).

10. Conclusion and future work

This paper introduces a novel keyword search paradigm that facilitates the automated extraction of data held about DSs in a relational database. According to the best of our knowledge, this keyword search paradigm has not been attempted before. Such a

searching paradigm, that liberates users from schema and query languages, will certainly be a great contribution especially now with the wider use of web accessible databases.

The result of a keyword query of the proposed paradigm is an OS; where an OS summarizes all data held about a particular DS (e.g. a person or any other logical object) in a database. The proposed paradigm produces OSs by traversing the data-graph; for this purpose the G^{DS}, R^{DS} and t^{DS} concepts are introduced which correspond to the sub-schema graph, relation and tuple containing the keywords. It starts from a tuple containing the keyword, i.e. the t^{DS} and continues traversing neighboring tuples (by traversing G^{DS}) as long as the data traversed is relevant to t^{DS}.

Certainly the primary challenge of this work is the selection of which relations and attributes in the G^{DS} to traverse. Therefore, an important contribution of this paper was the introduction, investigation and quantification of Relation and Attribute Affinity in the G^{DS} context. These Affinity scores in combination with the thresholds provided by the DBA (or users) will facilitate the automated generation of OSs. A technique is also proposed that facilitates users to select more quickly the suitable thresholds by estimating the size of OSs for the particular thresholds.

We conducted experiments on two databases, namely TPC-H and Northwind. The excellent P/R/F and Affinity Ranking Correctness results proved the quality of the proposed Affinity formulas.

A direction of future work concerns the investigation of snippets of OSs, i.e. the generation of more synoptic OSs with smaller size containing only the most important and distinguishable tuples of each OS; since some OSs may be very large in size. For this purpose efficient and effective algorithms will be investigated. A different direction of future work concerns further investigation of attribute clustering techniques, the automated identification of database theme domains and their relations' themes. For these problems, ontologies will be considered.

Acknowledgment

This work was partially supported by the "Hosting of Experienced Researchers from Abroad" programme (ΠΡΟΣΕΛΚΥΣΗ/ΠΡΟΕΜ/0308) funded by the Research Promotion Foundation, Cyprus.

Appendix A

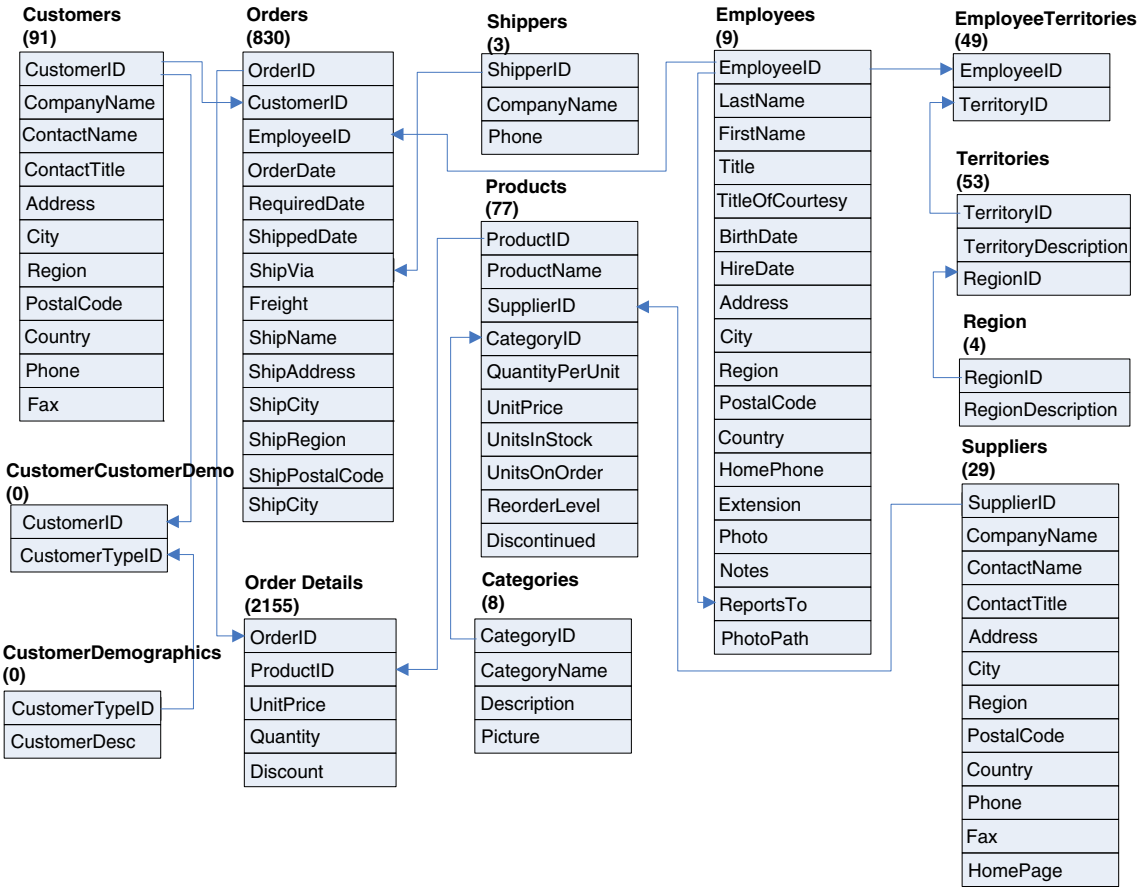

Fig. 18. The Northwind database schema (with the Cardinality of each relation).

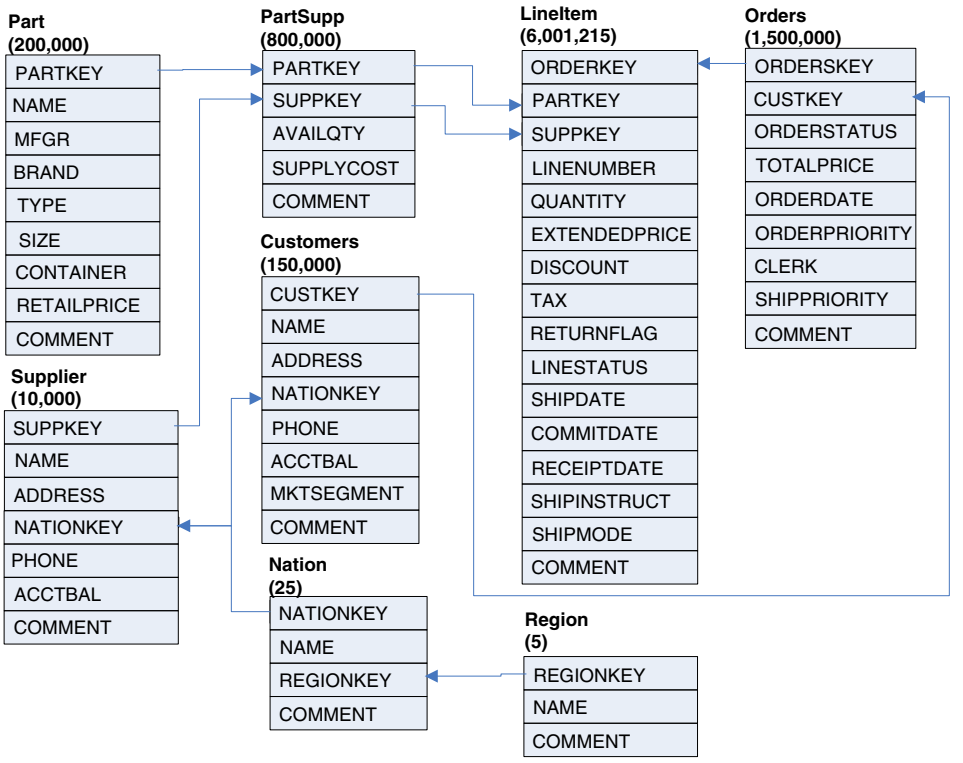

Fig. 19. The TPC-H database schema (with the Cardinality of each relation for Scale Factor 1).

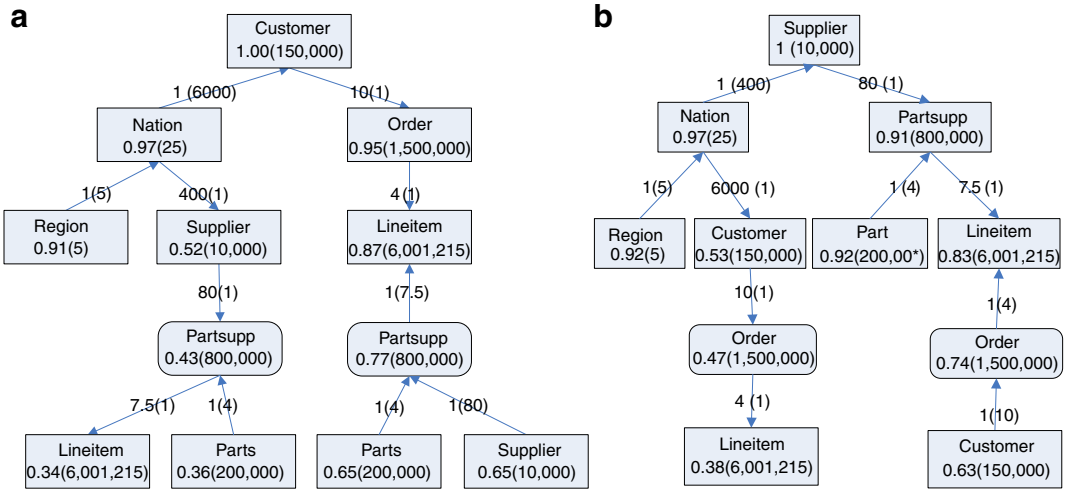

Fig. 20. The TPC-H customers and supplier G^{DS}s. Relationships are annotated with RC and ($\overline{RC}$) whilst relations with $\mathbf{Af_{Ri}}$ and ($|R_i|$), where Af_{Ri} was calculated with the weighting schemes W_1.

References

[1] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, Parag, S. Sudarshan, BANKS: browsing and keyword searching in relational databases, Proc. Very Large Data Bases Conf., 2002.
[2] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, S. Sudarshan, Keyword searching and browsing in databases using BANKS, Proc. Very Large Data Bases Conf., 2002.
[3] V. Hristidis, Y. Papakonstantinou, DISCOVER: keyword search in relational databases, Proc. Very Large Data Bases Conf., 2002.
[4] Microsoft. Northwind Database, 2004. http://www.microsoft.com/downloads/details.aspx?familyid=06616212-0356-46a0-8da2-eebc53a68034&displaylang=en.
[5] G. Fakas, Automated generation of object summaries from relational databases: a novel keyword searching paradigm, Proc. ICDE Workshop on Ranking in Databases (DBRank'08), 2008.
[6] Oracle. http://technet.oracle.com/products/text/content.html. 2006.

[7] IBM. http://www.ibm.com/software/data/db2/extenders/textinformation/index.html. 2006.
[8] Microsoft. Microsoft SQL Server 2000 Full-Text Search Deployment white paper (Q323739), 2004. http://support.microsoft.com/kb/323739.
[9] MySQL. Full-Text Search Functions, 2010. http://dev.mysql.com/doc/refman/5.5/en/fulltext-search.html.
[10] S. Agrawal, S. Chaudhuri, G. Das, DBXplorer: a system for keyword-based search over relational databases, Proc. ICDE Conf., 2002.
[11] N. L. Sarda, and A. Jain, Mragyati, A system for keyword-based searching in databases, Technical report, computing research repository (CoRR) cs. DB/0110052, 2001.
[12] A. Markowetz, Y. Yang, D. Papadias, Keyword search over relational tables and streams, TODS 34 (3) (2009).
[13] A. Markowetz, Y. Yang, D. Papadias, Reachability indexes for relational keyword search, Proc. ICDE, 2009.
[14] V. Hristidis, L. Gravano, Y. Papakonstantinou, Efficient IR-style keyword search over relational databases, Proc. Very Large Data Bases Conf., 2003.
[15] F. Liu, C. Yu, W. Meng, A. Chowdhury, Effective keyword search in relational databases, Proc. ACM SIGMOD Conf., 2006.
[16] Y. Luo, X. Lin, W. Wang, X. Zhou, SPARK: top-k keyword query in relational databases, Proc. ACM SIGMOD Conf., 2007.
[17] Q-H. Vu, B-C. Ooi, D. Papadias, A. Tung, A graph method for keyword-based selection of the top-K databases, Proc. ACM SIGMOD Conf., 2008.
[18] B. Yu, G. Li, K. Sollins, A.K.H. Tung, Effective keyword-based selection of relational databases, Proc. ACM SIGMOD, 2007.
[19] A. Kashyap, V. Hristidis, M. Petropoulos, S. Tavoulari, Effective navigation of query results based on concept hierarchies, IEEE TKDE, 2010.
[20] G. Weikum, G. Kasneci, M. Ramanath, F. Suchanek, Database and information-retrieval methods for knowledge discovery, Commun. ACM 52 (4) (2009).
[21] E. Chu, BaidA. , X. Chai, A.H. Doan, J.F. Naughton, Combining keyword search and forms for ad hoc querying of databases, Proc. ACM SIGMOD Conf., 2009.
[22] G. Li, J. Feng, J. Wang, L. Zhou, An effective and versatile keyword search engine on heterogenous data sources, PVLDB, 2008.
[23] X. Yang, B. Wang, G. Wang, G. Yu, Enhancing Keyword Search in Relational Databases Using Nearly Duplicate Records, IEEE Data Eng. Bull. 33 (1) (2010).
[24] L. Qin, J. Xu Yu, L. Chang, Keyword search in databases: the power of RDBMS, Proc. ACM SIGMOD Conf., 2009.
[25] R. Goldman, N. Shivakumar, S. Venkatasubramanian, H. Garcia-Molina, Proximity search in databases, Proc. Very Large Data Bases Conf., 1998.
[26] R. Guha, R. McCool, E. Miller, Semantic search, Proc. WWW Conf., 2003.
[27] G. Koutrika, A. Simitsis, Y. Ioannidis, Précis: the essence of a query answer, Proc. ICDE Conf., 2006.
[28] A. Simitsis, G. Koutrika, Y. Ioannidis, Generalized précis queries for logical database subset creation, Proc. ICDE Conf., 2007.
[29] A. Simitsis, G. Koutrika, Y. Ioannidis, Précis: from unstructured keywords as queries to structured databases as answers, VLDB J. 17 (1) (2008).
[30] S. Brin, L. Page, The anatomy of a large-scale hypertextual Web search engine, WWW Conference, 1998.
[31] A. Balmin, V. Hristidis, Y. Papakonstantinou, Objectrank: authority-based keyword search in databases, Proc. Very Large Data Bases Conf., 2004.
[32] R. Varadarajan, V. Hristidis, L. Raschid, Explaining and reformulating authority flow queries, Proc. ICDE Conf., 2008.
[33] G. Fakas, Z. Cai, Ranking of object summaries, Proc. ICDE Workshop on Ranking in Databases (DBRank'09), 2009.
[34] S. Castano, V.D. Antonellis, M. Fugini, B. Pernici, Conceptual schema analysis: techniques and applications, TODS 23 (3) (1998).
[35] D. Moody, A. Filtman, A methodology for clustering entity relationship models—a human information processing approach, In ER, 1999.
[36] C. Yu, H.V. Jagadish, Schema summarization, Proc. Very Large Data Bases Conf., 2006.
[37] X. Yang, C.M. Procopiuc, D. Srivastava, Summarizing relational databases, PVLDB 2 (1) (2009) 634–645.
[38] G. Das, V. Hristidis, N. Kapoor, S. Sudarshan, Ordering the attributes of query results, Proc. ACM SIGMOD Conf., 2006.
[39] N. Choi, I.Y. Song, H. Han, A survey on ontology mapping, SIGMOD Rec. 35 (3) (2006) 34–41.
[40] J. Madhavan, P.A. Bernstein, E. Rahm, Generic schema matching with cupid, Proc. Very Large Data Bases Conf., 2001.
[41] E. Rahm, P.A. Bernstein, A survey of approaches to automatic schema matching, VLDB J. 10 (4) (2001) 334–350.
[42] TPC, 2005. http://www.tpc.org/tpch/default.asp.
[43] WordNet, A Lexical Database for the English Language, 2010. http://wordnet.princeton.edu/.
[44] DBLP, 2010. http://www.informatik.uni-trier.de/ley/db/.
[45] Internet Movie Database, 2010. www.imdb.com.
[46] S. Polyviou, G. Samaras, P. Evripidou, A relationally complete visual query language for heterogeneous data sources and pervasive querying, Proc. ICDE Conf., 2005.

Georgios John Fakas is a Senior Lecturer in the Department of Computing and Mathematics of the Manchester Metropolitan University, UK. He also worked as a Research Associate at the Institute for Automatic Control of the Swiss Federal Institute of Technology - Lausanne (EPFL), Switzerland and at the Computer Science Department of the University of Cyprus, Cyprus. He obtained his Ph.D. in Computation in 1998 from the Department of Computation, UMIST, Manchester, UK. His research interests include keyword search and ranking in relational databases.