Journal of Information & Knowledge Management, Vol. 10, No. 2 (2011) 1–16

DOI: 10.1142/S0219649211002936

Automated Generation of Personal Data Reports from Relational Databases

Georgios John Fakas*, Ben Cawley† and Zhi Cai‡
Department of Computing and Mathematics
Manchester Metropolitan University
Manchester, M1 5GD, UK
**g.fakas@mmu.ac.uk*
†*b.m.cawley@mmu.ac.uk*
‡*z.cai@mmu.ac.uk*

Abstract. This paper presents a novel approach for extracting personal data and automatically generating Personal Data Reports (PDRs) from relational databases. Such PDRs can be used among other purposes for compliance with Subject Access Requests of Data Protection Acts. Two methodologies with different usability characteristics are introduced: (1) the G^{DS} *Based Method* and (2) the *By Schema Browsing Method*. The proposed methdologies combine the use of graphs and query languages for the construction of PDRs. The novelty of these methodologies is that they do not require any prior knowledge of either the database schema or of any query language by the users. An optimisation algorithm is proposed that employs hash tables and reuses already found data. We conducted several queries on two standard benchmark databases (i.e. TPC-H and Microsoft Northwind) and we present the performance results.

Keywords: Data extraction; privacy protection; relational databases.

1. Introduction

Data Protection Acts (DPAs), such as the US Privacy Act of 1974 (2004 Edition) (US Privacy Act, 1974), the United Kingdom DPA of 1984 and 1998 and the EU Directives on Privacy of 1995 (95/462/EC) (Data Protection Act, 1998), give individuals (namely "Data Subjects" (DS)) who are subject of personal data a general right of access to the personal data which relates to them. For example, according to the UK DPA of 1998, a DS has the right (namely the "subject access right") to request access to records from any person or organisation (namely the "data controller") who may process (by holding, disclosing or using) such personal information. Such a request is called a Subject Access Request (SAR). More precisely, a DS is entitled, under section 8(1) of the DPA 1998, to be given a copy in an intelligible form of all the information constituting any personal data of which that individual is the DS (Data Protection Act, 1998). Various countries such as the USA (Safe Harbour, 1998; US Privacy Act, 1974), Japan (Japan's Personal Information Protection Act, 2003), Australia (Australian Privacy Amendment Act, 2000) have implemented their own acts covering the protection of personal data and ensuring that personal data is accessible.

As a consequence, the capability for organisations to quickly access and generate Personal Data Reports (PDRs) with information held about Data Subjects (DSs) is central to promoting compliance with SARs of DPAs. Especially when organisations collect and store vast amounts of information about individuals then the tracking, extraction and presentation in an intelligible format of such data requires a significant investment of both time and effort. Therefore, the need of a formal methodology that can automate the generation of such reports is very apparent.

1.1. *Problems and challenges*

Although current DBMSs such as ORACLE (Oracle, 2010), SQL Server (Microsoft, 2010), provide advanced report generation facilities, they do not provide any specialised formal methodologies or facilities for the automated extraction of personal data and the generation of PDRs. Nevertheless, they provide techniques that can be used in this context, such as Full-Text Search (e.g., Microsoft, 2004a). Full-Text search facilities identify tuples containing keywords (such as names, IDs) associated with the DS, however, these tuples do not comprise the complete set of personal data held about an individual. This is because of the nature of relational databases where relations are linked with other relations (containing additional data) via primary key (PK) to foreign key (FK)

*Corresponding author.

May 3, 2011 6:47:56pm WSPC/188-JIKM 00293 ISSN: 0219-6492

relationships. This is a challenging problem: how to extract the complete set of such personal data but also exclude any data that will breach the privacy of other Third Party DSs (3PDSs).

Another relevant challenging problem is when personal data about a particular DS is held in more than one R^{DS}s, for instance (for the Northwind database (Microsoft, 2004b)) in Customers and Employees relations. If tuples from the two R^{DS}s are associated (through primary key to foreign key relationships), this indicates semantically that Employees have served themselves as Customers. In such cases, the challenge is to obtain, present and emphasise this semantic accordingly. Finally, another challenging problem is to present PDR to users in an intelligible form.

1.2. *Approach*

In this paper, we propose two methods that are based on the fact that some relations in a database schema comprise a central point of data held about Data Subjects (DSs). We denote such relations as R^{DS}s (e.g. $R^{\text{Employees}}$, $R^{\text{Customers}}$ etc.) and relations linked around them include additional information about the particular DS.

The first approach, namely the G^{DS}*Based Method* produces a PDR fully compliant with the DPA SARs. In this approach, users do not need to have any prior knowledge of the database schema or query language, however, a DBA (e.g. the Data Controller) is required to register the database. The DBA registers in the *Data Subject Schema Graph* (denoted as G^{DS}), the database schema and the personal data to be extracted from neighbouring relations that he/she wishes to include in the PDR.

The second approach, namely the *By Schema Browsing Method*, also produces a PDR fully compliant with the DPA SARs. However, a DBA is no longer required to pre-register the database but the user, during the construction of the PDR, browses the database schema and chooses which relations, relationships and attributes to include in the G^{DS}. This method does not require the user to have any query language knowledge. Comparing the usability of the two methods we observe that the first method facilitates the fast and efficient generation of large amounts of SAR PDRs from a particular database (since the same G^{DS} is used for all PDRs) whilst the second method facilitates the fast and instant generation of occasional SAR PDRs (since no prior G^{DS} registration is required).

1.3. *Novel contributions*

The contributions of this paper are the following:

- The formal definition of SAR PDR queries and the proposition of two methodologies (namely the G^{DS}*Based Method* and *By Schema Browsing Method*) for the automated generation of SAR Personal Data Reports (PDRs) from relational databases. The novelty of these approaches is that they do not require any knowledge of a query language at all whereas at the same time they minimise DBAs' intervention. The proposition of the two alternative methods facilitates the fast generation of both big batches of and occasional reports.
- The presentation of PDRs in all methods has a hierarchical format which is an intelligible form for users; this is achieved by combining the use of graphs and SQL.
- The optimisation techniques employed during the generation of PDR facilitates the reuse of already found data. Experimental results quantify the effectiveness of this optimisation approach and also verify the scalability of the system.

1.4. *Paper organisation*

The rest of the paper is structured as follows. Section 2 discusses related work. Section 3 presents the system overview, Sec. 4 describes the Master Index and Sec. 5 describes the Graph Generator. Section 6 describes the Plan Generator and Execution and Sec. 7 describes the Optimisation Algorithm. Section 8 presents the experimental evaluation while Sec. 9 concludes the paper.

2. Related Work

To the best of our knowledge, no related work has been found in the area of the automated generation of SAR PDRs. Nevertheless, we present and compare related work of the area.

2.1. *DBMS report generation facilities*

Almost all commercial DBMS provide powerful and user friendly wizards for the generation of reports that require limited programming knowledge and effort from users (e.g., Oracle, 2010). However, such facilities still require a lot of time, effort and good SQL knowledge to produce such SAR PDRs because of the specialisation (and consequently the complexity) of the problem. An additional problem of such facilities is their difficulty to deal with big G^{DS} containing multiple levels and branches of relations, such as the Northwind Customer and Employee G^{DS}. Consequently, this is not a very feasible solution for organisations that need to quickly produce such reports from large and often different databases. Such tools are based only on the use of SQL, in contrast, to our approach that combines the use of graphs and SQL and gives us the opportunity to deal with arbitrary sizes of G^{DS}s. Finally,

May 3, 2011 6:47:57pm WSPC/188-JIKM 00293 ISSN: 0219-6492

they do not take advantage of optimisation potentials that arise from the specialised nature of the problem (namely the reuse of already found data from M:1 relationships).

2.2. *Query by browsing systems*

The *By Schema Browsing* method used here resembles other Query By Browsing techniques proposed in other studies such as those by Carey and Polyviou (Carey *et al.*, 1996; Polyviou *et al.*, 2005). Such techniques facilitate users to navigate through the dataset based on the data associations (through primary keys and foreign keys). These techniques usually unify querying and browsing of data and emphasise mainly on the navigation and GUI aspects of the problem. Our approach also unifies querying and browsing by facilitating the concurrent generation of the PDR with the selection of schema entities; however, it is beyond the scope of this paper to investigate the GUI issues. Using such tools for the generation of PDRs, although possible, is time consuming. For instance, consider we have a schema $R_{\text{Customer}} \rightarrow R_{\text{Order}} \rightarrow R_{\text{OrderDetails}}$ and we want to retrieve information about Customer c_4 who has 10 Orders and each Order has another 10 tuples of OrderDetails and so on. Then, the user would have to navigate manually through this dataset subset exhaustively. Although such techniques, e.g. QBB (Polyviou *et al.*, 2005), provide facilities such as Folder Templates that can support the generation of PDRs, these facilities, among other differences, still require some additional programming.

2.3. *Other data protection act applications*

Almost all the work in the DPA domain concerns only the limited disclosure of data. For instance, Hippocratic databases (Agrawal *et al.*, 2002; LeFevre *et al.*, 2004; Massacci *et al.*, 2006) and IBM Tivoli Privacy Manager (Ashley and Moore, 2002). Other data privacy work that gained publicity recently is k-anonymity where in a k-anonymised dataset each record is indistinguishable from at least $k-1$ other records (e.g., Sweeney, 2002; Machanavajjhala *et al.*, 2006; Meyerson and Williams, 2004).

2.4. *Keyword search (KWS) in relational databases*

Keyword search techniques facilitate the discovery of data from relational databases using keywords queries rather than complex SQL queries. Keyword search, similarly to our work, emphasises the liberation of users from database schema and SQL knowledge by exploiting the association of tuples through their primary keys and foreign keys. Keyword search query results comprised of sets of joining tuples that are associated through their keys and collectively contain all query keywords. For instance, Object Summaries (Fakas, 2008, 2011; Fakas and Cai, 2009), DISCOVER (Hristidis and Papakonstantinou, 2002), DBXplorer (Agrawal *et al.*, 2002), Mragyati (Sarda and Jain, 2001), and relational stream keyword search (Markowetz *et al.*, 2007) use series of SQL statements in order to execute such keyword queries whilst other keyword search systems such as BANKS (Aditya *et al.*, 2002; Bhalotia *et al.*, 2002; Hulgeri *et al.*, 2001), DBSurfer (Wheeldon *et al.*, 2004) convert the whole database into a data graph.

2.5. *Full-text searching in relational databases*

Full-Text searching techniques facilitate the discovery of tuples that contain queried keywords. These techniques allow users to create full text indices on single attributes and then perform keyword queries, e.g. Oracle 9i Text (Oracle, 2006), IBM DB2 Text Information Extender (IBM, 2006) and Microsoft SQL Server 2000 (Microsoft, 2002).

3. System Overview

This section introduces the functionality of the system with an example of an SAR PDR query and the system architecture.

3.1. *An example of an SAR PDR query*

For example, we consider the PDR shown in Fig. 1, the Microsoft Northwind schema in Fig. 2, a sample of the dataset in Fig. 3 and the G^{DS} for Employees in Fig. 5. The arrows in the schema point in the direction of the primary key to the foreign key (i.e. 1:M) relationships between relations.

Now, we consider that a user is trying to produce an SAR PDR for "Janet Leverling". Our system will automatically generate the report presented in Fig. 1. This keyword "Janet Leverling" is found in tuple e_3; however, this tuple does not comprise the whole set of the data held about the particular DS as it is also associated via primary to foreign key relationships with other tuples. Since e_3 belongs to the Employees relation then Employees relation is considered as the R^{DS} and consequently the Employees G^{DS} will be used for the generation of the report. Our system will insert tuple e_3 at the root node of the PDR tree under construction and then will start traversing the dataset based on the Employees G^{DS} in order to generate the rest of the report. For instance, e_2 is

SAR for "Janet Leverling"

Employee

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	Address	...	
3	Leverling	Janet	Sales Representative	Ms.	722 Moss Bay Blvd .	...	(e_3)

Employee (Reports To)

LastName	FirstName	
Fuller	Andrew	(e_2)

Territories, Region

TerritoryDescription	RegionDescription	
Atlanta	Southern	(et_1, t_4, r_2)

Orders

OrderID	ShipName	ShipAddress	OrderDate	RequiredDate	...	
10273	QUICK -Stop	Taucherstae 10	1996 -08 -05	1996 -09 -02	...	(o_2)

Customer

CustomerID	CompanyName	ContactName	
Quick	QUICK -Stop	Margaret Peacock	(c_2)

Shippers

ShipperID	CompanyName	
3	Federal Shipping	(s_3)

Orders Details

UnitPrice	Quantity	Discount	
15.2000	50	0.2	(od_1)

Products

ProductName	QuantityPerUnit	
Chang	24 - 12 oz bottles	(p_2)

Categories

CategoryName	Description	
Beverages	Soft drinks , coffees , teas , beers , ...	(ca_1)

Fig. 1. A PDR for the "Janet Leverling" SAR.

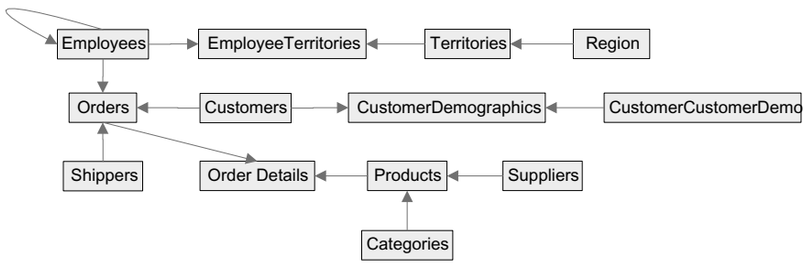


Fig. 2. The northwind database schema.

another Employee that the DS has a ReportTo relationship and o_2 is an Order that the DS has served and so on.

In some cases, Data Controllers are obliged by DPA SARs to disclose thorough reports about DSs from their databases, such as data about an employee's performance at work (e.g. PDRs including Employee's Orders in the Northwind example). However, according to DPAs, Data Controllers also must protect the privacy of 3PDS, for instance, they cannot disclose to "Janet Leverling" private information (such as ID, address, telephone number, salary) about e_2 or c_2 (i.e. the reason that these 2 tuples are projections of their relations). Data Controllers must be very careful before replying to an SAR that the content of a PDR does not disclose information that may compromise the identification of data about 3PDS. K-anonymity techniques investigate such re-identification attacks on disclosed data about DSs (see Sec. 2.3).

3.2. *System architecture*

A high level representation of the proposed architecture is presented in Fig. 4. In this subsection, we give a high level description of the components of our system.

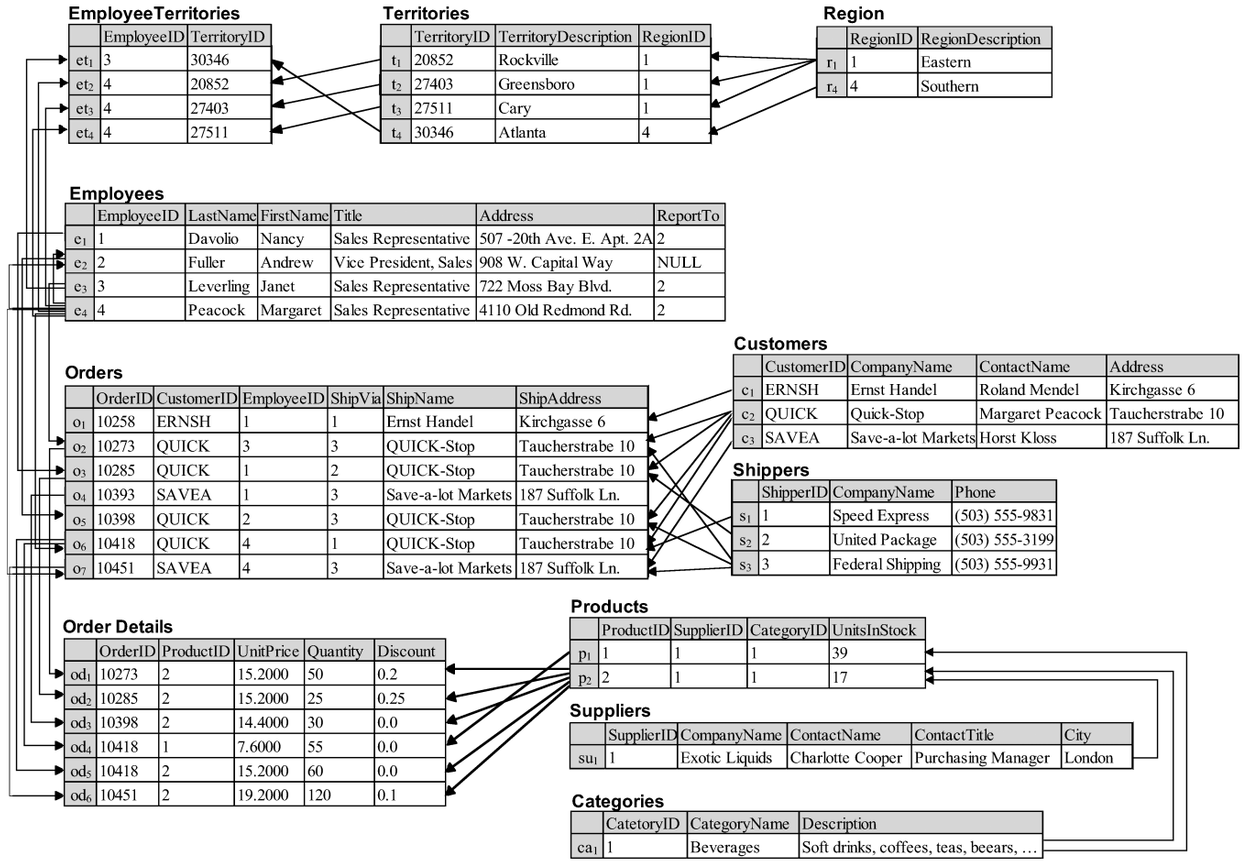


Fig. 3. A sample of the dataset from the northwind database.

First, the user enters the identifying keywords for a DS into the system and the Master Index returns all tuples that contain these keywords. The second step is the generation of necessary schema graphs (G^{DS}, $G^{\mathrm{DS}}_{\mathrm{T}}$ etc.) depending on the method followed. For instance, for the G^{DS}*Based Method*, the DBA needs to define the G^{DS}. Then, the Graph Generator module generates automatically all necessary graphs for each method accordingly. Plan Generator and Plan Execution modules take as input the tuple sets and graphs and produce the PDR of the DS. Finally, the Present PDR Report module prints in an intelligible form the PDR (due to lack of space we excluded the description of this final module from the paper). During the Present PDR Report module, we can also (1) hide any unnecessary physical keys or intersection tuples (i.e. tuples mapping M:N relationships) or even (2) merge tuples that are associated with a 1:1 relationship.

4. Master Index

The user can give a query in two formats: either (1) as a set of keywords or (2) as a set of keywords, relations and attributes where each keyword will be searched only in the given relation's attribute. Either query format can uniquely identify tuples associated with DSs. These identifying keywords (id-kw) are looked up in the master index (Hristidis and Papakonstantinou, 2002), which returns the set of tuples that contain the identifying keywords as part of an attribute value. We define the terms $\mathrm{DB}^{\text{id-kw}}$ and $R_i^{\text{id-kw}}$ as the set of tuples for a given identifying keyword query per the whole database or per a particular relation R_i respectively. The user can exclude tuples for further processing from these tuple sets if they are irrelevant for the DS under investigation. For instance, irrelevant data is very likely if identifying keyword queries are not very narrow (usually when using the first query format).

For instance (from the sample dataset of Fig. 3), the following id-kw queries (1) "Leverling", "Janet", (2) "Peackock", "Margaret", (3) "4", Employees.EmployeeID and (4) "QUICK", Customers.CustomerID will produce the following $\mathrm{DB}^{\text{id-kw}}$s (1) $\{e_3\}$, (2) $\{e_4, c_2\}$, (3) $\{e_4\}$ and (4) $\{c_2\}$ respectively.

The master index has been implemented using the Microsoft SQL Server 2000 Full-Text Search feature

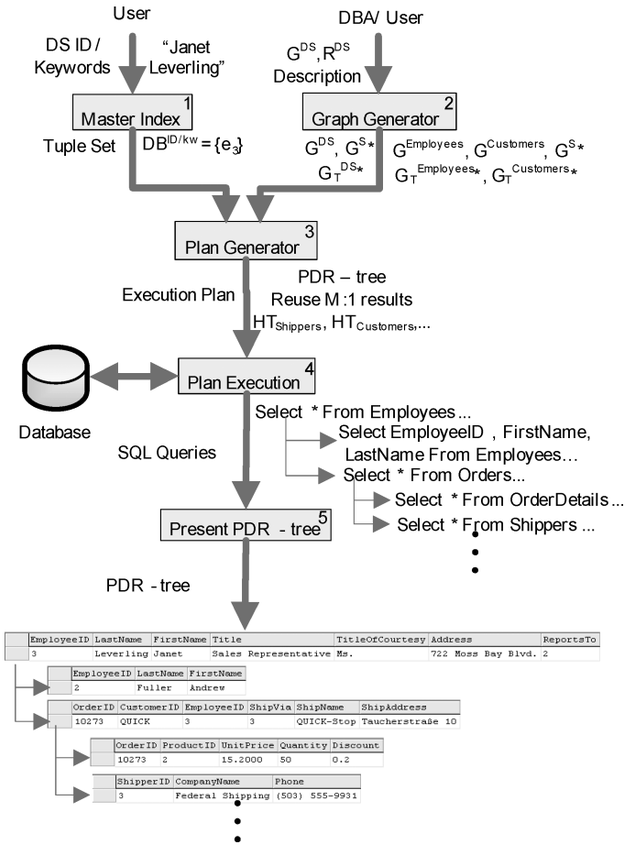


Fig. 4. The proposed architecture.

(which builds full text external catalogues in order to index attributes of relations). The master index inspects the index of each attribute and combines the results in order to produce $\mathrm{DB}^{\text{id-kw}}$ (Microsoft, 2004a).

5. Graph Generator

In this section, we describe the generation and the content of the necessary graphs for each method. For the G^{DS}*Based Method*, the generation of a schema graph G^{S} and a G^{DS} for each R^{DS} (found in the schema) are required, and for the *By Schema Browsing Method*, a schema graph G^{S} and a G_T^{DS} for each R^{DS} are required. G^{DS}s and G_T^{DS}s are generated with DBA's and user's contribution respectively whilst G^{S} is automatically generated by the system.

5.1. G^{DS} based method: Generation of the data subject schema graph G^{DS}

The Data Controller and DBA decide what information to include in the PDR for each DS. Certainly, a user-friendly GUI will simplify this task for DBAs. However, this is beyond the scope of this work. The DBA carefully studies the database schema and then selects which **relations**, **attributes** and **relationships** to include in the G^{DS} (Fig. 5). For instance, attributes from Employees and Customers relations are excluded from Customers and Employees G^{DS} respectively for 3PDS privacy reasons.

Definition 1: A Data Subject Schema Graph $G^{\mathrm{DS}}(V, E)$ is a Rooted Directed Labeled Graph that captures a subset of the database schema with a root node R_0 representing R^{DS}. 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 R_i consists of set(s) of projection attributes (i.e. Public Data (PD) and 3PDS Public Data (3PPD)) that the DBA wishes to extract from the particular Relation (in order to protect privacy of third

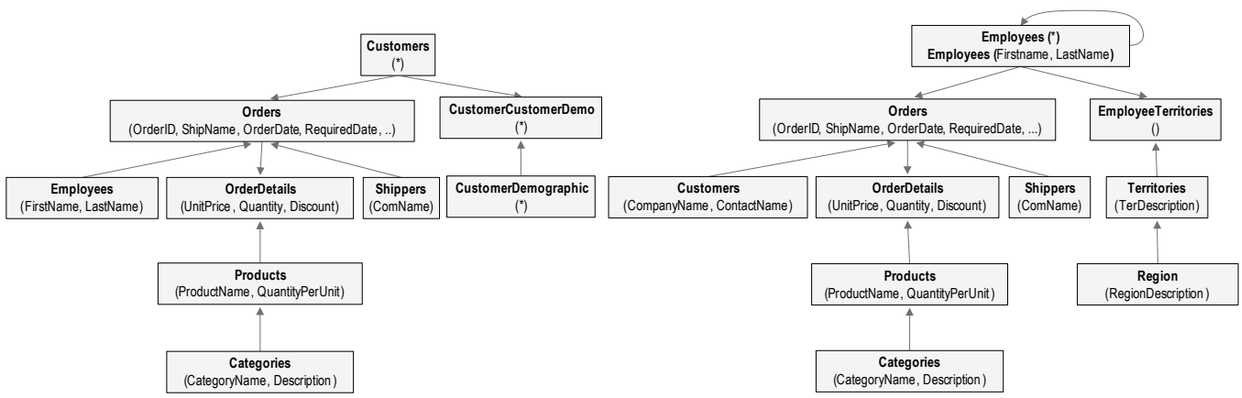


Fig. 5. G^{DS}s for customers and employees R^{DS}s of the northwind database.

May 3, 2011 6:48:52pm WSPC/188-JIKM 00293 ISSN: 0219-6492 1st Reading

party DSs; 3PPD will be used if R_i participates in recursive or loop relationships).

- A set of edges $E = \{(R_i \rightarrow R_j) | R_i, R_j \in V$, 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.

Loops or recursive relationships can also be included in the G^{DS}, e.g. the recursive relationship *Reports To* on *Employees* (Fig. 5).

5.2. *By schema browsing method: Generation of the G^{S} and G_T^{DS}*

In this method, the user gradually generates concurrently the G_T^{DS} and the PDR tree by including additional relations with T relationships away from R^{DS}. We denote the intermediary DS schema graph of this approach as G_T^{DS} (where T denotes the current value of T) and the final and complete graph as G^{DS} (similarly to the G^{DS}*Based Method*, note that G_T^{DS} graphs are subsets of G^{DS}) because eventually this graph is becoming identical with the G^{DS} graph (assuming it is correctly generated). The user is given the option to generate the schema graph by incrementing the value for T and additionally by selecting the set of relationships, relations and attributes (for each T, namely from "$GT_T^{\mathrm{DS}} - GT_{T-1}^{\mathrm{DS}}$") to include in the G_T^{DS}.

For instance for the R^{DS} Customers, the user is given the option to choose for $T = 1$ from Orders and CustomerCustomerDemo and for $T = 2$ Employees, OrderDetails, Shippers, CustomerDemographic relations to include in the G_T^{DS} respectively. The user accepts all of these relations to be included in the G_T^{DS}. For $T = 3$, the user has to choose from ReportsTo Employees (recursive relationship), EmployeeTerritories and Products relations; however, the user will select only Products. Similarly for $T = 4$, the user will select only the Categories relation.

6. Plan Generator and Execution

In this section we describe how we generate the SAR PDR, namely the PDR trees.

6.1. *G^{DS} based method: Generation of an SAR PDR*

This module takes as input (1) the tuple set $\mathrm{DB}^{\mathrm{id\text{-}kw}}$ and (2) the G^{DS} graphs and produces the PDR tree. The algorithm (Fig. 6) is breadth-first (using queues) and starts traversing the G^{DS} graphs for each $\mathrm{DB}^{\mathrm{id-kw}}$ tuple with the $\mathrm{DB}^{\mathrm{id-kw}}$ tuple(s) as the initial entry(s) of the PDR tree. The algorithm employs relational operations with the general format $\pi_{\mathrm{PD}}(\sigma_{R_i.\mathrm{FK}=t_j.\mathrm{PK}}(R_i))$ for $R_j \rightarrow R_i$ (i.e. 1:M) relationships and $\pi_{\mathrm{PD}}(\sigma_{R_i.\mathrm{PK}=t_j.\mathrm{FK}}(R_i))$ for $R_j \leftarrow R_i$ (i.e. M:1) relationships for each node t_j of the PDR-tree. Where R_i is the current relation, PD is the Public Data set of R_i (or 3PPD in the case of recursive relationships) 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 PDR tree as child nodes of t_j.

For instance, in the context of the example of Fig. 1, the tuple set consists of tuple e_3 which belongs to Employees

```
Algorithm: Generate_PDR-tree
Input: Graph G^DS, Tuple t //t belongs to DB^id-kw
Output: Tree PDR-tree
{
   AddRoot(PDR-tree, t)
   EnQueue(Q, t)
   WHILE !IsEmptyQueue (Q) {
     t_j=DeQueue(Q)
     R_j= GetRelation(G^DS, t_j)
     For each relationship rel of R_j in G^DS {
       Get R_i which has a relationship rel with R_j
       IF the relationship rel is R_j→R_i
          THEN dataset="SELECT R_i.PD
                        FROM R_i
                        WHERE R_j.PK= R_i.FK"
          ELSE dataset="SELECT R_i.PD
                        FROM R_i
                        WHERE R_j.FK= R_i.PK"
       For each tuple t_i of dataset {
         AddChild (PDR-tree, t_j, t_i)
         EnQueue (Q, t_i)
       }
     }
   }
}
```

Fig. 6. The generate_PDR-tree algorithm.

relation and consequently the Employees G^{DS} will be used. The algorithm will traverse firstly the root node of the G^{DS} by using the operation $\pi_{\mathrm{PD}}(e_3)$ and then proceed traversing relations of the graph by following the relationships using either the operation $\pi_{\mathrm{PD}}(\sigma_{R_i\cdot\mathrm{FK}=t_j\cdot\mathrm{PK}}(R_i))$ or $\pi_{\mathrm{PD}}(\sigma_{R_i\cdot\mathrm{PK}=t_j\cdot\mathrm{FK}}(R_i))$. The operations π_{3PPD} $(\sigma_{\mathrm{PK}} = e_{3\cdot\mathrm{FK}}$ $(\mathit{Employees}))$, $\pi_{\mathrm{3PPD}}(\sigma_{\mathrm{FK}=e_3\cdots\mathrm{PK}}$ $(\mathit{Employees}))$, $\pi_{\mathrm{PD}}(\sigma_{\mathrm{FK}} = e_{3\cdot\mathrm{PK}}$ $(\mathit{Orders}))$, and $\pi_{\mathrm{PD}}(\sigma_{\mathrm{FK}} = e_3 \cdot \mathrm{PK}$ $(\mathit{EmployeeTerritories}))$, where primary key represents the primary key of the current relation, will return $\{\}$, $\{e_2\}$, $\{o_2\}$, and $\{et_1\}$ respectively. These sets of tuples are appended on the PDR tree as children nodes of e_3. Following the next level of relationships, for each tuple (namely $\{e_2\}, \{o_2\}, \{et_1\}$) we run the operations $\pi_{\mathrm{PD}}(\sigma_{\mathrm{PK}} = o_{2\cdot\mathrm{FK}}$ $(\mathit{Customers}))$, $\pi_{\mathrm{PD}}(\sigma_{\mathrm{FK}=o_2\cdot\mathrm{PK}}$ $(\mathit{OrderDetails}))$, π_{PD} $(\sigma_{\mathrm{PK}} = o_{2\cdot\mathrm{FK}}$ $(\mathit{Shippers}))$ and $\pi_{\mathrm{PD}}(\sigma_{\mathrm{PK}} = et_{1\cdot\mathrm{FK}}$ $(\mathit{Territories}))$ which will return $\{c_2\}, \{od_1\}, \{s_3\}$ and $\{t_4\}$ respectively. This procedure continues until we reach the end of the G^{DS} graph.

The proposed algorithm supports the processing of looped, recursive, l:1, 1:M, and M:1 relationships. It should be emphasised that the PDR tree is a tree structure and results from recursive or looped relationships that are presented in a tree structure rather than in a cyclic way. For instance, in our example, e_3 has a child e_2 as a result of the recursive *ReportsTo* Employees relationship. Similarly, results from l:1, 1:M and M:1 (and therefore M:N) relationships are also presented in tree structure.

Definition 2: *A* **PDR tree** is the result of an SAR PDR Query and is a hierarchical report which consists of:

- A set of nodes representing projections of tuples $V = \{t_1 \cdots t_n | n \geq 1$, where each tuple t_i belongs to relation R_k, $R_k \in G^{\mathrm{DS}}$, the root node is t_1, $t_1 \in R^{\mathrm{DS}}$, $t_1 \in \mathrm{DB}^{\mathrm{id\text{-}kw}}\}$. The projection on each tuple $t_i \in R_k$ is defined in R_k.
- A set of edges $E = \{(t_i \rightarrow t_j) | t_i, t_j \in V, t_i \in R_k,\ t_j \in R_l$: $(R_k \rightarrow R_l \wedge t_i \cdot \mathrm{PK} = t_j \cdot \mathrm{FK}) \vee (R_k \leftarrow R_l \wedge t_i \cdot \mathrm{FK} = t_j \cdot \mathrm{PK})\}$.

6.1.1. *Existence of DS tuples in more than one* R^{DS}

When we are searching for a particular DS, it is always possible for tuples associated with the particular DS to be found in more than one R^{DS} that appear on the same G^{DS}. For instance, DS "Peacock Margaret" is found in both Customers and Employees R^{DS}s (namely e_4 and c_2) that appear on both Employees and Customers G^{DS}s. In such cases, a simplified approach would produce two separate and independent reports, i.e. one for each R^{DS}. However, there is some additional semantic that needs to be considered in such cases. Namely, in the context of the above example, such information that is associated with both R^{DS}s indicates that Employees (i.e. e_4) have served themselves as Customers (i.e. c_2). The set of such common data should be represented in the PDR tree in such a way that indicates this semantic (e.g. by using a different presentation format such as underlined, colored). In this subsection, we explain how we can identify and mark such data in order to indicate this semantic.

The proposed solution consists of the following amendment to the algorithm of the previous subsection. First, we need to separate the set of the common data on G^{DS} graphs so we can present it accordingly to the user. We observe that this data is extracted from the shortest paths between the primary and the secondary R^{DS}s of the G^{DS} graphs. We define as primary R^{DS} (i.e. R^{DS1}) the root R^{DS} of the G^{DS}, and as secondary the rest (namely $R^{\mathrm{DS2}}, R^{\mathrm{DS3}}, \ldots$), e.g. for Customers G^{DS}, the primary R^{DS} is Customers whilst Employees and Shippers are considered as secondary. We denote such paths as Intersection Paths ($\mathrm{IP}^{\mathrm{DSn}}$), i.e. the shortest path between the primary R^{DS} and the secondary R^{DSn}. For instance, for Customers G^{DS}, we have an $\mathrm{IP}^{\mathrm{DS}}$ for Employees (namely $R_{\mathrm{Customers}} \rightarrow R_{\mathrm{Orders}} \leftarrow R_{\mathrm{Employees}}$) and for Shippers (namely $R_{\mathrm{Customers}} \rightarrow R_{\mathrm{Orders}} \leftarrow R_{\mathrm{Shippers}}$). Eventually, based on the proposed $\mathrm{IP}^{\mathrm{DS}}$s, the system will be able to identify and mark the common data.

The proposed algorithm by default proposes the shortest path among R^{DS}s in order to generate IPs and that may not be always the most meaningful or desirable path. For example, in the TPC-H schema, if we assume that both G^{DS}, for Customer and Supplier include both R^{DS}, then the shortest path between Customer and Supplier is through the Nation relation (which semantically means that Customer and Suppliers come from the same nation), whilst an alternative and longer path is through Orders relation (which semantically means Customer and Supplier associated with the same Order) and possibly the latter is more meaningful and interesting. Our algorithm gives DBAs the choice of path.

Second, we proceed with the construction of the PDR tree where we maintain Label Lists (LL) for the tuples extracted from relations belonging to $\mathrm{IP}^{\mathrm{DS}}$ and are associated with tuples from the secondary R^{DS}s. More precisely, we generate the PDR tree as usual and when we generate tuples from the secondary R^{DS} (e.g. Employees), we check whether the generated tuple belongs to the tuple set of the secondary R^{DS} (i.e. $R^{\mathrm{id-kw}}_{\mathit{Employees}}$). If true, then we backtrack until we reach the root of the PDR tree or reach a tuple that is already marked then mark the LL of the tuple by adding the tuple id (Fig. 7). For instance in our example, during the construction of Customers R^{DS} (where the tuple set consists of c_2 and e_4) when the e_4 tuple is generated and added to the tree, the marking

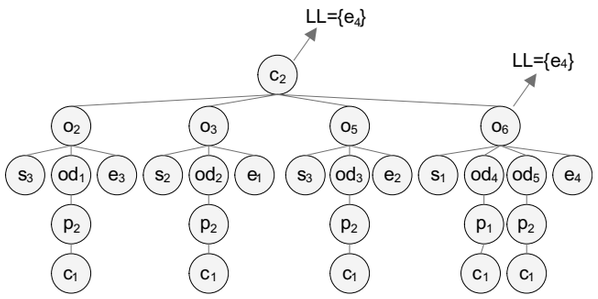


Fig. 7. Label lists on tuples: Mark common data among R^{DS}s.

function will be called and that will add e_4 identifier on LLs of tuples o_6 and c_2.

This problem resembles keyword searching techniques (such as BANKS, DISCOVERY (Hristidis and Papakonstantinou, 2002; Aditya *et al.*, 2002; Bhalotia *et al.*, 2002; Hulgeri *et al.*, 2001)) when searching for keys located in different relations. Such techniques use Steiner trees or Minimum Total Join Networks (MTJN) (Hristidis and Papakonstantinou, 2002; Aditya *et al.*, 2002; Bhalotia *et al.*, 2002; Hulgeri *et al.*, 2001) in order to traverse the dataset graph. Of course, the differences are: (1) during the generation of PDRs we mark already retrieved relevant tuples that are associated with more than one R^{DS} whilst in keyword searching we investigate the efficient extractions of joins of tuples by minimising the retrieval of irrelevant data; and (2) here we are still interested in data extracted from the IP sub-trees (e.g. data extracted from the Orders subtree) in contrast to keyword searching techniques where this is excluded due to the minimality criterion.

6.1.2. *Existence of more than one DS tuples in each R^{DS}*

Although we do not expect more than one tuple in the same relation storing information about the same DS in a well-designed database system, this "anomaly" is still possible (namely $|R_i^{\text{id-kw}}| > 1$). For instance, an R^{DS} Customers tuple set includes 2 or more tuples (i.e. $R^{\text{id-kw}}_{\text{Customers}} = \{c_1, c_{12}\}$) for the same person. This case can easily be dealt with by generating two separate PDR trees, namely one for each tuple. In contrast to the case of the previous sub-section, in this case we do not have any intersection of data between tuples of the same relation. However, if the same DS is also found in another R^{DS} such as Employees (e.g. $R^{\text{id-kw}}_{\text{Employees}} = \{e_9, e_{13}\}$), we need to treat this as described in the previous subsection, i.e. find all the association of data between the tuples of the different R^{DS}. We generalise our algorithm to deal with such a case by adding to LL all these associations of tuples.

6.2. *By schema browsing method: Generation of an SAR PDR*

This module takes as input: (1) the tuple set $\text{DB}^{\text{id-kw}}$, (2) the current G_T^{DS} and (3) T and produces the PDR tree. This method facilitates the gradual generation of a PDR tree by browsing the database schema. For each T, both the G_T^{DS} graph and the PDR tree are expanded concurrently until the user reaches the completion of the report. That means that for each T the system generates the tuples belonging to the sub-graph "$G_T^{\text{DS}} - G_{T-1}^{\text{DS}}$" and then appends these tuples accordingly to the current PDR tree (that means that the system does not reconstruct from scratch the PDR tree for each G_T^{DS}). This also means that performance-wise this approach should have very similar properties with the G^{DS} Based Method. This method generates PDR trees in exactly the same way as the previous method and the same rules about its content apply.

7. Optimisation Algorithm

During the generation of a PDR tree the reuse of already retrieved tuples can be used as an optimisation technique in order to improve performance.

Heuristic 1: *During the generation of a PDR tree, subtree results starting from a tuple belonging to Relation R_i can be reused in other sub-queries on R_i when $t_j > 1$ AND $R_j \leftarrow R_i$ AND $t_j = t_i - 1$ (i.e. R_j and R_i participate in an M:1 relationship) where t_j and t_i denote the path length from R^{DS} to R_j and R_i respectively. The rationale is that results already found from M:1 relationships may be reused rather than extracted from the database and added on the PDR tree again.*

For instance, from the dataset example of Figs. 3 and 8, we observe that the tuple p_2 points on od_1, od_2 etc. Thus, we could store the $p_2 \leftarrow c_1$ subtree result for od_1 and reuse it for od_2 etc. We observe that this query optimisation opportunity happens due to the M:1 relationship $R_{\text{OrderDetails}} \leftarrow R_{\text{Products}} \leftarrow R_{\text{Categories}}$.

However, storing and reusing subtrees starting from the root tuple of the PDR tree is not necessary. This concept is not very apparent in the Northwind database because all relationships on R^{DS}s of both Employees and Customers G^{DS} are 1:M. On the contrary in the TPC-H database, where the Customer R^{DS} includes the $R_{\text{Customer}} \leftarrow R_{\text{Nation}} \leftarrow R_{\text{Region}}$ relationships, we can observe that there is no need to store Nation and Region sub-tree results since the instance of Customer DS is always one (e.g. Customer#000143500) and, therefore, there is no need of reuse in the same query. This is the reason we include the condition $t_j > 1$ in the heuristic.

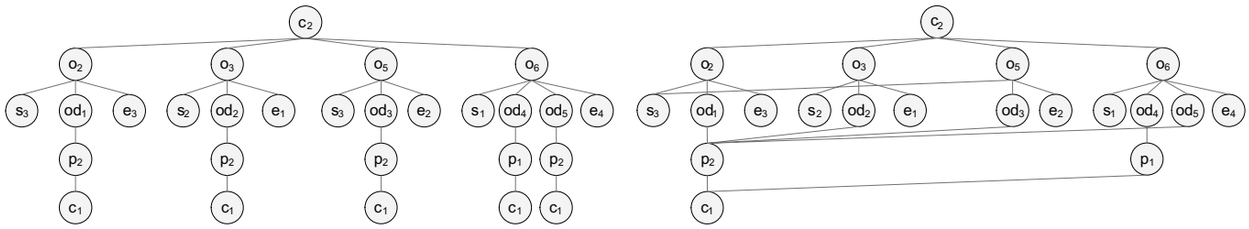


Fig. 8. PDR tree: Optimised/naïve approach.

In the context of a G^{DS}, relationships are considered to have R^{DS} as a directional point of reference. This is defined in the heuristic with the sub-condition $t_j = t_i - 1$.

7.1. *Reuse already found information*

From the above heuristic, we realise that we need to create some effective storage and indexing facility for tuples from each Relation on a G^{DS} that satisfies the heuristic conditions, e.g. for $R_{\mathrm{Employees}}$, R_{Shippers}, R_{Products}, $R_{\mathrm{Categories}}$, $R_{\mathrm{CustomerDemographics}}$ for Customers G^{DS}. We create a Hash Table (HT_R) for each one of these Relations where we store the found primary keys.

For instance, continuing on the example of Figs. 3 and 8, for Customer c_2 we generate o_2, o_3, o_5 and o_6 (from Orders), and then od_1, od_2, od_3, od_4 and od_5 (from OrderDetails). Now, we need to generate Products tuples where R_{Products} satisfies the M:1 relationship condition (with OrderDetails). Dealing firstly with od_1, we generate p_2 and store p_2 in HT_{Products}. Dealing now with od_2, we first search in HT_{Products} (for tuple p_2 as we know we are searching for p_2 from the od_2 foreign key) where we find the key in the HT_{Products} and therefore, we do not need to fetch it from the database but simply point od_2 directly on the existing subtree of p_2 (Fig. 8).

Experimental results of the optimisation technique are presented in the following section. It is apparent from the comparative results (naïve/optimisation) (see Figs. 9 and 10) that the more data is reused, the more beneficial the optimised approach is. HT_Rs were implemented by using the rehashing technique (also called double hashing) (provided by .NET Framework Base Class Library) (Microsoft).

7.2. *Preventing the invocation of the optimisation algorithm*

However, the potential performance benefits of the proposed optimisation algorithm do not depend solely on the schema properties (i.e. on M:1 relationships) but also on the dataset distribution properties. For instance, even if a G^{DS} schema includes several M:1 relationships if the association between the tuples of these relationships is 1:1, then invoking the proposed optimisation algorithm will result in additional performance cost rather than benefit (due to the use of Hash Tables, etc.).

Assuming that data are uniformly distributed among the relations of a database then based on relations' cardinality we can estimate the reuse potentials of the particular dataset. Let $r(R_i)$ denote the times that each tuple from R_i will potentially be reused for the generation of a $t(R^{\mathrm{DS}})$ PDR and $R_o \rightarrow R_1 \rightarrow \cdots \rightarrow R_n \leftarrow R_{n+1}$ be a path on G^{DS} (where R_o and R_{n+1} denote R^{DS} and R_i respectively). Then, a $t(R^{\mathrm{DS}})$ maybe associated with $\prod_{i=0}^{n} \frac{|R_i|}{|R_{i+1}|} = \frac{|R_n|}{|R_0|} = \frac{|R_n|}{|R^{\mathrm{DS}}|}$ tuples from R_n and since a particular tuple $t(R_n)$ may reuse $\frac{|R_n|}{|R_{n+1}|}$ times tuples from R_{n+1}, hence, we can infer that a $t(R^{\mathrm{DS}})$ may reuse $\frac{|R_n|}{|R^{\mathrm{DS}}|\cdot|R_i|}$ times tuples from R_i. If R_i has a sub-tree then the subtree tuples will also be reused for $r(R_i)$ times; we can use the same approach to estimate the reuse of the data inside the subtree (and then multiply it by $r(R_i)$).

Let us discuss the usefulness of the proposed algorithm on the two databases; Northwind database has very good reusability properties in contrast to the TPC-H database. In Appendices 1 and 2, the schemata of the 2 database benchmarks also depict their relations' cardinality. For instance, the Employees G^{DS} from the Northwind database has the following reusability properties: $r(R_{\mathrm{Shippers}}) = |R_{\mathrm{Orders}}|/(|R_{\mathrm{Employees}}| \cdot |R_{\mathrm{Shippers}}|) = 830/(9 \cdot 3) = 30.7 \approx 31$ and similarly $r(R_{\mathrm{Product}}) = 3.11 \approx 3$ while $r(R_{\mathrm{Categories}}) = |R_{\mathrm{Product}}|/|R_{\mathrm{Categories}}| \cdot r(R_{\mathrm{Product}}) = 29.9 \approx 30$. On the other hand, the Customer G^{DS} from the TPC-H database has the following properties: $r(R_{\mathrm{Partsupp}}) = |R_{\mathrm{lineitem}}|/(|R_{\mathrm{Customer}}| \cdot |R_{\mathrm{Partsupp}}|) = 5 \cdot 10^{-5}$and $r(R_{\mathrm{Part}}) = 20 \cdot 10^{-5}$; therefore we can infer that an individual customer is not expected to be associated with any tuples from R_{partsupp} and R_{part} more than once. Hence, the utilisation of the optimisation algorithm on the particular TPC-H G^{DS} will result in performance reductions rather than improvements (e.g. maintenance of Hash Tables for 40 Partsupp tuples etc., since each Customer is approximately associated with 40 Partsupp tuples).

Based on the above data reusability estimations, we can choose a policy on prevention/invocation of the

Optimisation Algorithm for any R_i participating in an M:1 relationship. For instance, we invoke the algorithm for all R_is in the Northwind whilst in TPC-H we prevent it for all R_is (with Q10 the only exception). Experimental results presented in the next section are based on this policy.

8. Experiments

The system was evaluated with two databases, namely Northwind and TPC-H. The TPC-H benchmark was used to also validate the scalability of the system on giga-scale datasets. We measured the performance of the system in terms of execution time and memory requirements and then compared the naïve with the optimised method. For these experiments, we used the Microsoft SQL Server 2000 DBMS and a PC with Intel Pentium M Processor, 1.7 Ghz and 512 MB of main memory. The DBMS Maximum Server Memory was set to 80 MB.

8.1. *Experimental datasets and queries*

The size of the TPC-H database is 1 GB (with Scale Factor 1) and the size of Northwind is 3.7 MB. Although the Northwind database is very small in comparison to the TCP-H, it was very useful for the evaluation of the proposed methodologies as its dataset facilitated the comparative measurement of the optimised/naïve approaches. In contrast, the TPC-H due to its schema nature and data distribution did not facilitate the optimised/naïve measurement (i.e. why we used an artificial Query e.g. Q10). For instance, the Supplier G^{DS} has no M:1 relationships and although the Customer G^{DS} contains M:1 relationships (e.g. $R_{\mathrm{Lineitem}} \leftarrow R_{\mathrm{Partsupp}} \leftarrow R_{\mathrm{Part}}$) data distribution is such that it does not facilitate any reuse. The TPC-H schema did not facilitate the study of queries with multiple tuples appearing in several R^{DS}s and G^{DS}s either. Nevertheless, the TPC-H database was still a very useful benchmark for evaluating the performance of the proposed methodologies on large scale databases.

The Queries used to evaluate the proposed system are described (in terms of tuples and relations cardinality) in detail in Appendix 3. Due to the uniqueness of the evaluating problem, we proposed our own Queries and also made some minor alterations on both database datasets. The alterations were made in order to produce PDRs with results from multiple R^{DS}s (e.g. Q6, Q7 and Q8) and no alterations were made on the distribution of data (with Q10 being the only exception). For instance, for Q6 (Northwind database) we changed the ContactName to "Margaret Peacock" in tuple Customers(Quick) so the "Margaret Peacock" $\mathrm{DB}^{\mathrm{id-kw}}$ will include both Employees (4) and Customer (Quick) tuples. Q10 is an altered version of Q9 where Orders contain the same lineitems several times; although this is not semantically meaningful, it was very interesting for our evaluation (since, we can now study the optimisation benefit from the reuse of data).

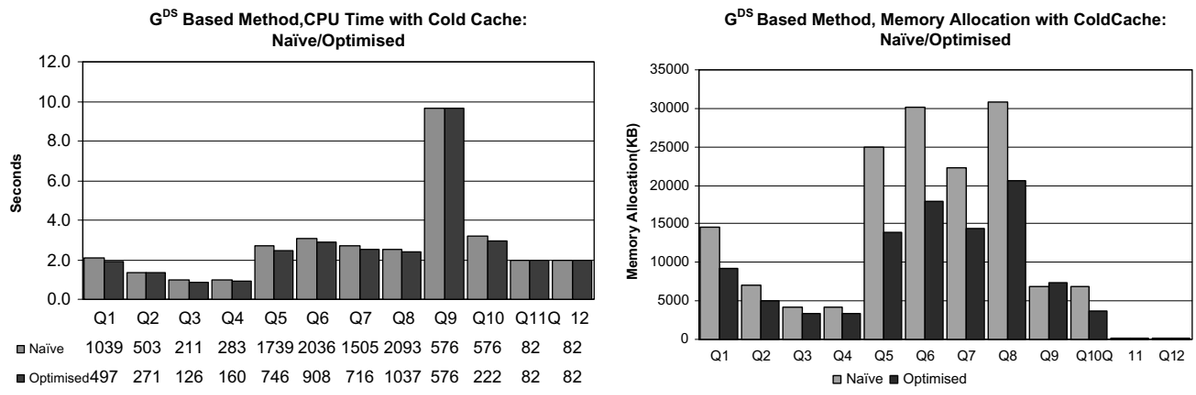
8.2. *Performance evaluation*

The following results describe the performance of the G^{DS}*Based Method*; for space limitation *By Schema Browsing Method*'s results are omitted since they are almost identical to G^{DS}*Based Method*'s results. We indicate in each set of experiments what the cache status is (either warm or cold) as the cache status significantly affects the performance of the queries. We run each query 20 times, excluded the worst and best measurements and then calculated the average of the remaining 18 measurements.

The figures below give the CPU execution time and Memory consumption for the naïve and optimised approach. The benefit of the optimised approach is very apparent when (a) the PDR is large and also (b) the amount of reused tuples is large. In Appendix 3, the difference between the "Naïve" and "Optimised" Number of Tuples in the |PDR| columns indicates the amount of reused tuples per query.

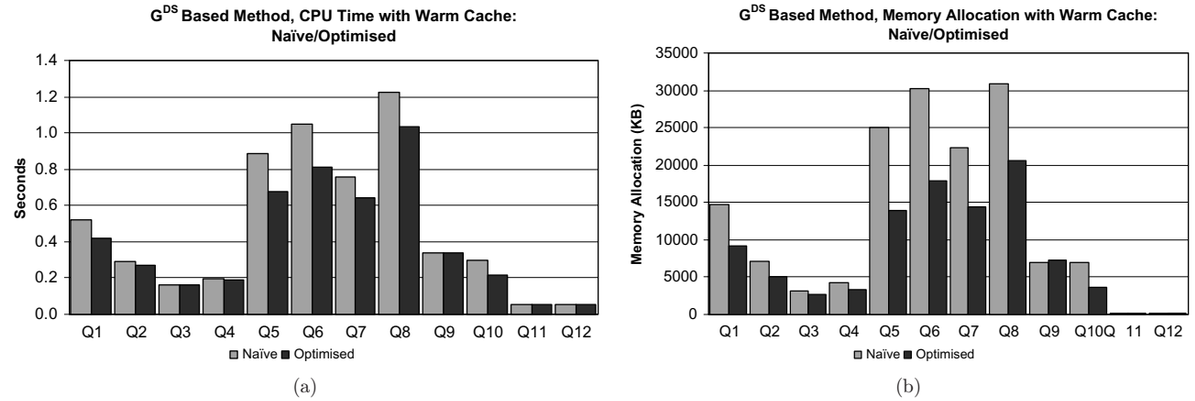
The results obtained from the optimised approach are almost always better, in both CPU and Memory terms, than the naïve approach. We also notice, as expected, that (a) both optimised and naïve results are a function of the size of the PDR (i.e. |PDR|) and of the size of the dataset and that (b) the ratios $\lambda^{\mathrm{Mem}}_{\mathrm{Naive,Optim}} = \frac{Q^{\mathrm{Mem}}_{\mathrm{Naive}}}{Q^{\mathrm{Mem}}_{\mathrm{Optim}}}$ and $\lambda^{\mathrm{CPU}}_{\mathrm{Naive,Optim}} = \frac{Q^{\mathrm{CPU}}_{\mathrm{Naive}}}{Q^{\mathrm{CPU}}_{\mathrm{Optim}}}$ (where $Q^{\mathrm{Mem}}_{\mathrm{Naive}}$, $Q^{\mathrm{Mem}}_{\mathrm{Optim}}$, $Q^{\mathrm{CPU}}_{\mathrm{Naive}}$, $Q^{\mathrm{CPU}}_{\mathrm{Optim}}$ denote the Memory and CPU time consumptions of a Query for the naïve/optimised approaches respectively) are correlated with $\lambda^{\mathrm{PDR-Tree}}_{\mathrm{Naive,Optim}} = \frac{|\mathrm{PDR-tree_{Naive}}|}{|\mathrm{PDR-tree_{Optim}}|}$ (where $|\mathrm{PDR} - \mathrm{tree_{Naive}}|$ and $|\mathrm{PDR} - \mathrm{tree_{Optim}}|$ denote the size of the PDR tree in terms of tuples). This means that the bigger the PDR is in combination with the bigger reuse of tuples needed for the generation of a PDR, the better resource savings we get. For the G^{DS}*Based Method* with cold memory, the largest values for $\lambda^{\mathrm{Mem}}_{\mathrm{Naive,Optim}}$ and $\lambda^{\mathrm{CPU}}_{\mathrm{Naive,Optim}}$ ratios were 1.90 and 1.10 with $\lambda^{\mathrm{PDR-Tree}}_{\mathrm{Naive,Optim}}$ 576/222 and 1739/746 obtained from Q10 and Q5 respectively.

Figures 9(a) and 9(b) show the results with cold cache whilst Figs. 10(a) and 10(b) display then with warm cache for the G^{DS} Based Method. Figure 9(a) also depicts the sizes of PDR (i.e. $|\mathrm{PDR} - \mathrm{tree_{Naive}}|$ and $|\mathrm{PDR} - \mathrm{tree_{Optim}}|$). Comparing cold/warm results, we observe that CPU time is reduced significantly, i.e. the average $\lambda^{\mathrm{CPU}}_{\mathrm{Cold,Warm}} \approx 12$ for both optimised and naïve approaches whilst the Memory



(a) (b)

Fig. 9. (a, b) Performance results of G^{DS} based method with cold cache.



(a) (b)

Fig. 10. (a, b) Performance results of G^{DS} based method with warmd cache.

consumption remains the same (i.e. the average $\lambda^{\mathrm{Mem}}_{\mathrm{Cold,Warm}} \approx 1$). For the G^{DS} Based Method with warm memory, the largest values for $\lambda^{\mathrm{Mem}}_{\mathrm{Naive,Optim}}$ and $\lambda^{\mathrm{CPU}}_{\mathrm{Naive,Optim}}$ ratios were 1.90 and 1.37 with $\lambda^{\mathrm{PDR-Tree}}_{\mathrm{Naive,Optim}}$ 576/222 obtained both from Q10.

Comparing naïve/optimised measures, we also notice that the effect of the optimised approach is larger on Memory than on the CPU execution time (i.e. $\lambda^{\mathrm{Mem}}_{\mathrm{Naive,Optim}} > \lambda^{\mathrm{CPU}}_{\mathrm{Naive,Optim}}$). One should expect that the optimised approach would have given much better results for $\lambda^{\mathrm{CPU}}_{\mathrm{Naive,Optim}}$; however, we observe that even the largest $\lambda^{\mathrm{CPU}}_{\mathrm{Naive,Optim}}$ value is only limited to 1.37, i.e. Q10 with warm cache, with $\lambda^{PDR-Tree}_{Naive,Optim} = 576/222 = 2.59$. The explanation is that DBMS caches query results anyway and thus the majority of already found results are fetched from the cache rather than the database. This also explains the observation that $\lambda^{\mathrm{Mem}}_{\mathrm{Naive,Optim}} > \lambda^{\mathrm{CPU}}_{\mathrm{Naive,Optim}}$, since $\lambda^{\mathrm{Mem}}_{\mathrm{Naive,Optim}}$ is benefited more than the $\lambda^{\mathrm{CPU}}_{\mathrm{Naive,Optim}}$ from the optimisation technique. This also very interestingly justifies CPU execution times of Q9 and Q10, where in Q10 even during the naïve approach, the big percentage of reuse of tuples significantly reduces execution time.

9. Conclusions

This paper introduces two formal methodologies for the automated generation of Personal Data Reports from relational databases. This problem faces considerable challenges because of the nature of the relational model where relations are linked with other relations via relationships. For instance, how can we extract the complete set of such personal data? An additional challenge is

when personal data about a particular DS is held in more than one R^{DS}s and tuples from these R^{DS}s are associated (through foreign key to primary key relationships). Thus, we need to present this intersection semantic accordingly in an intelligible form.

We face these challenges with the following contributions. We proposed two methodologies, namely (1) the G^{DS}*Based Method* and the (2) *By Schema Browsing Method*. These approaches produce PDRs that can be fully compliant with the DPA SAR. These methods are based on the G^{DS} and G_T^{DS} graphs respectively which are defined by the DBAs and users. These methods do not require any SQL knowledge by DBAs or users.

We also use Label Lists in order to deal with the intersection semantic of data from different R^{DS}s by marking tuples that are associated with more than one R^{DS}. In addition, we show how we deal we recursive loops, 1:1, 1:M and M:1 relationships. Finally, an optimisation technique is employed that uses Hash Tables in order to reuse already found tuples from M:1 relationships.

We conducted experiments in two databases, namely Northwind and TPC-H. The presented results also show the comparative benefit from the optimisation technique.

A direction of future work concerns the investigation of k-Anonymity techniques in order to protect 3PDS data. Another direction of future work concerns the addition of querying features to the current techniques. For instance, a user may wish to define the maximum size of the PDR (in this case we should incorporate ranking mechanisms) or query conditions on PDR data (e.g. include in a PDR data about a DS after a particular date). A different direction concerns a methodology that will semi-automate the generation of G^{DS} by proposing a G^{DS} to users.

Appendix 1: The Northwind Database

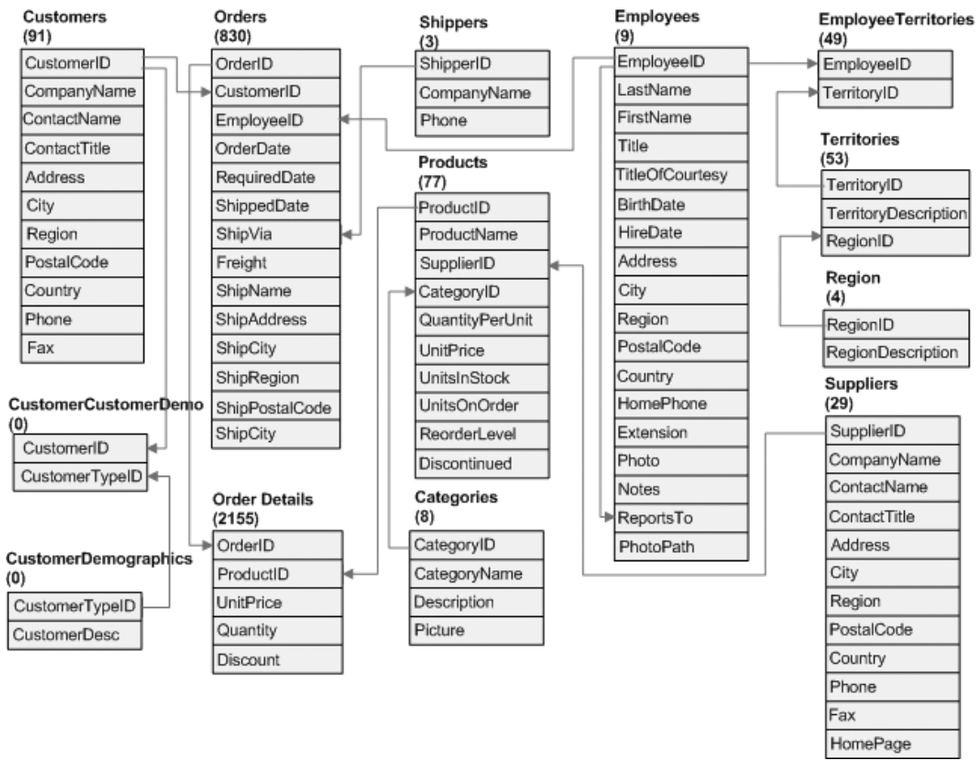


Fig. 11. The northwind DB schema (with the cardinality of each relation).

Appendix 2: The TPC-H Database

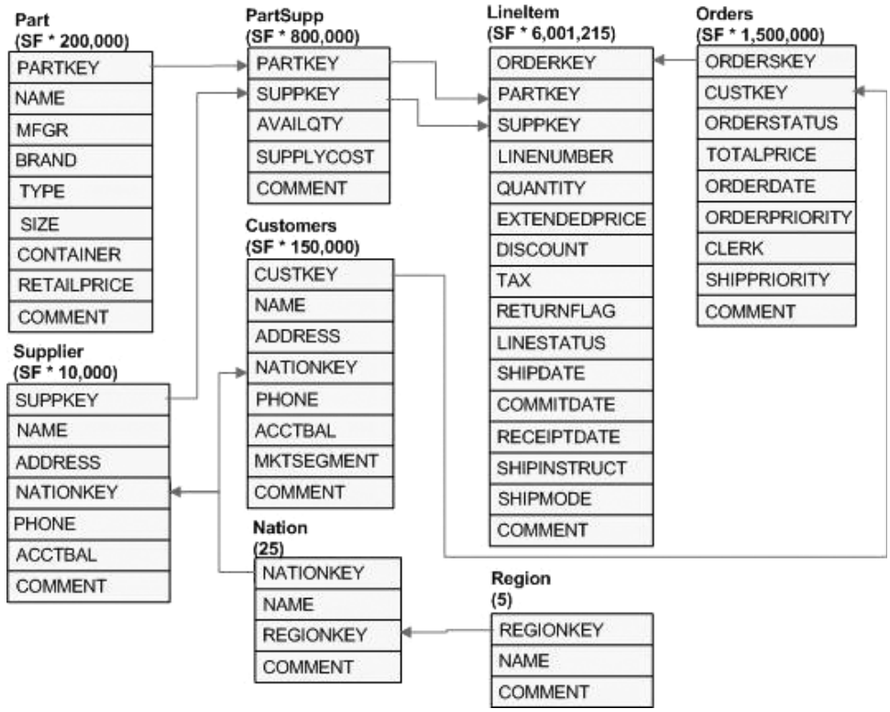


Fig. 12. The TPC-H DB schema (with the cardinality of each relation where SF (scale factor) = 1).

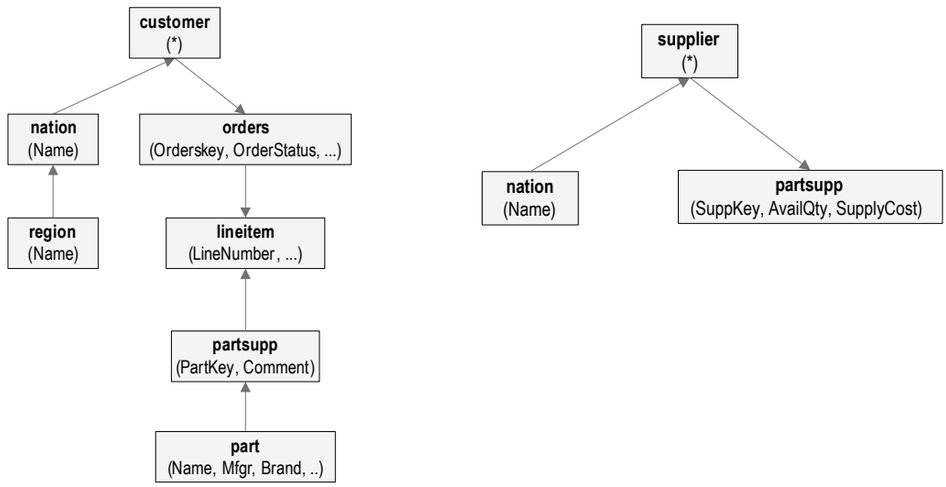


Fig. 13. The G^{DS} for customer and supplier.

Appendix 3: The 12 Queries

	Id-kw	DB	DB^{id-kw}	[1]$\lvert G^{DS}\rvert$ ($\lvert G^{DS1}UG^{DS2}\rvert$)	[2]|PDR|	
					Naive	Optimised
Q1	"Andrew Fuller"	Northwind	Employees (2)	10	1039	497
Q2	"Steven Buchanan"	Northwind	Employees (5)	10	503	271
Q3	"Christina Berglund"	Northwind	Customers (Bergs)	9	211	126
Q4	"Paula Wilson"	Northwind	Customers (Rattc)	9	283	160
Q5	4, Employees.EmployeeId	Northwind	Employees (4)	10	1739	746
Q6	"Margaret Peacock"	Northwind	Employees (4), Customers (Quick)	10 + 9 (12)	2036	908
Q7	"Horst Kloss"	Northwind	Employees (8), Customers (Savea)	10 + 9 (12)	1505	716
Q8	"Janet Leverling"	Northwind	Employees (3) Employees (9) Customers (Hungo) Customers (Warth)	10 + 9 (12)	2093	1037
Q9	"Customer#000143500"	TPC-H	Customer (143500)	7	576	576
Q10	"Customer#000143500"	TPC-H	Customer (143500) (Modified)	7	576	222
Q11	"Supplier#000000101"	TPC-H	Supplier (101)	3	82	82
Q12	"Supplier#000000546"	TPC-H	Supplier (546)	3	82	82

[1]$\lvert G^{DS}\rvert$ and $\lvert G^{DS1}UG^{DS2}\rvert$ indicate their sizes in terms of relations. [2]|PDR| indicates the size in terms of tuples.

References

Aditya, B, G Bhalotia, S Chakrabarti, A Hulgeri, C Nakhe, Parag and S Sudarshan (2002). BANKS: Browsing and keyword searching in relational databases. *Proceedings of the 28th VLDB Conference*, Hong Kong, China.

Agrawal, R, R Bayardo, C Faloutsos, J Kiernan, R Rantzau and R Srikant (2004). Auditing compliance with a Hippocratic database. *Proceedings of the 30th VLDB Conference*, Toronto, Canada.

Agrawal, R, J Kiernan, R Srikant and Y Xu (2002). Hippocratic databases. *Proceedings of the 28th VLDB Conference*, Hong Kong, China.

Agrawal, S, S Chaudhuri and G Das (2002). DBXplorer: A system for keyword-based search over relational databases. *Proceedings of the 18th International Conference on Data Engineering*, San Jose, USA.

Ashley, P and D Moore (2002). Enforcing privacy within an enterprise using IBM Tivoli Privacy Manager for e-business. Retrieved from http://www-128.ibm.com/developerworks/tivoli/library/t-privacy/index.html.

Australian Privacy Amendment Act (2000). Retrieved from http://www.privacy.gov.au/business/index.html.

Bhalotia, G, A Hulgeri, C Nakhe, S Chakrabarti and S Sudarshan (2002). Keyword searching and browsing in databases using BANKS. *Proceedings of the 18th International Conference on Data Engineering*, San Jose, USA.

Carey, MJ, LM Haas, V Maganty and JH Williams (1996). PESTO: An integrated query/browser for object databases. *Proceedings of the 22nd VLDB conference*, Mumbai, India. pp. 203−214. (Also Available Online at citeseer.ist.psu.edu/haas96pesto.html).

Data Protection Act (1998). HMSO, London. Retrieved from www.hmso.gov.uk/acts/acts1998/19980029.htm.

Fakas, G (2011). Automated generation of object summaries from relational database: A novel keyword searching paradigm. *Data & Knowledge Engineering*, 70, 208−229.

Fakas, G and Z Cai (2009). Ranking of object summaries. *DBRank Workshop 2009, ICDE*.

Fakas, G (2008). Automated Generation of Object summaries from relational database: A novel keyword searching paradigm. *DBRank Workshop 2008, ICDE*.

Hristidis, V and Y Papakonstantinou (2002). DISCOVER: Keyword search in relational databases. *Proceedings of the 28th VLDB Conference*. Hong Kong, China.

Hulgeri, A, G Bhalotia, C Nakhe, S Chakrabarti and S Sudarshan (2001). Keyword search in databases. *Institute of Electrical and Electronics Engineers Data Engineering Bulletin*, 24(3), 22−32.

IBM (2006). Retrieved from http://www.ibm.com/software/data/db2/extenders/textinformation/index.html.

Japans Personal Information Protection Act (2003). Retrieved from http://www.privacyexchange.org/japan/japanPIPA2003v3_1.pdf.

LeFevre, K, R Agrawal, V Ercegovac, R Ramakrishnan, Y Xu and D DeWitt (2004). Limiting disclosure in Hippocratic databases. *Proceedings of the 30th VLDB Conference*, Toronto, Canada.

Li, Y, C Yu and HV Jagadish (2004). Schema-Free XQuery. *Proceedings of the 30th VLDB Conference*, Toronto, China.

Machanavajjhala, A, J Gehrke and D Kifer (2006). ℓ-Diversity: Privacy beyond k-anonymity. *Proceedings of the International Conference on Data Engineering*.

Markowetz, A, Y Yang and D Papadias (2007). Keyword search on relational data streams. *Proceedings of ACM Conference on the Management of Data (SIGMOD)*, Beijing, China.

Massacci, F, J Mylopoulos and N Zannone (2006). Hierarchical Hippocratic databases with minimal disclosure for virtual organisations. *The VLDB Journal*, 15, 370−387.

Meyerson, A and R Williams (2004). On the complexity of optimal k-anonymity. *Proceedings of the Symposium on Principles of Database Systems*, pp. 223−228.

Microsoft (2002). Retrieved from http://msdn2.microsoft.com/en-us/library/aa902674(SQL080).aspx.

Microsoft (2004a). Microsoft SQL Server 2000 Full-Text Search Deployment white paper (Q323739). Retrieved from http://support.microsoft.com/kb/323739.

Microsoft (2004b). Retrieved from http://www.microsoft.com/downloads/details.aspx?familyid=06616212-0356-46a0-8da2-eebc53a68034&displaylang=en.

Microsoft (2005). An Extensive Examination of Data Structures using C# 2.0. Part: 2 The Queue, Stack and Hash table. Retrieved from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvs05/html/datastructures20_2.asp.

Microsoft (2010). Retrieved from www.microsoft.com/sql.

Oracle (2006). Retrieved from http://technet.oracle.com/products/text/content.html.

Oracle (2010). Retrieved from www.oracle.com.

Polyviou, S, G Samaras and P Evripidou (2005). A relationally complete visual query language for heterogeneous data sources and pervasive querying. *Proceedings of International Conference of Data Engineering*, pp. 471−482.

Safe Harbour (1998). Retrieved from http://www.export.gov/safeharbor/index.html.

Sarda, NL and A Jain (2001). Mragyati: A system for keyword-based searching in databases, Technical report, *Computing Research Repository* (CoRR) cs.DB/0110052.

Sweeney, L (2002). K-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5), 557−570.

TPC (2005). Retrieved from http://www.tpc.org/tpch/default.asp.

US Privacy Act (1974). Retrieved from http://www.usdoj.gov/oip/04_7_1.html.

Wheeldon, R, M Levene and K Keenoy (2004). DBSurfer: A search and navigation tool for relational databases. *Annual British National Conference on Databases*.