CARDINALITY ESTIMATION FOR THE OPTIMIZATION OF QUERIES ON ONTOLOGIES

[1]E. Patrick Shironoshita, MSEE
patrick@infotechsoft.com

[1]Michael T. Ryan
mryan@infotechsoft.com

[1,2]Mansur R. Kabuka, Ph.D.
kabuka@infotechsoft.com

[1]INFOTECH Soft, Inc.
9200 S Dadeland Blvd., Suite 620
Miami, FL 33156, USA
+1 (305) 670 5111

[2]University of Miami
Coral Gables, FL 33124

ABSTRACT

An effective, accurate algorithm for cardinality estimation of queries on ontology models of data is presented. The algorithm relies on the decomposition of queries into query pattern paths, where each path produces a set of values for each variable within the result form of the query. In order to estimate the total number of result set parameters for each path, a set of statistics is compiled on the properties of the ontology. Experimental analysis has shown that the algorithm produces estimates with high accuracy and with high correlation to actual values. Thus, this algorithm can be used as the cornerstone of an effective optimization strategy for queries on diverse, heterogeneous data sources modeled as ontologies.

Categories and Subject Descriptors

H.2.4. [**Database Management**]: Systems – *query processing.*

I.2.4. [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods – *Representations, Semantic networks.*

General Terms

Algorithms, Performance, Standardization.

Keywords

Ontology, semantic query, query optimization, cardinality estimation, OWL, SPARQL.

1. INTRODUCTION

An ontology – the explicit specification of a conceptualization [5] – is a means of representing semantic knowledge, and includes at least a controlled vocabulary of terms, and some specification of their meaning [7]. The modeling of data sources as ontologies mapped to standardized representations is essential for defining correspondence among entities belonging to different sources, providing a semantically consistent, unified, and evolving view of data regardless of its storage formats and naming conventions, and resolving conflicts among sources [12].

All but the most trivial queries over such distributed, heterogeneous sources can be executed according to different plans, each of which may require vastly different amounts of time and computational resources. Thus, there is a growing need to develop mechanisms for the optimization of query execution. The objective of query optimization is to select an efficient query plan according to a cost function or cost model. The optimization cost function depends on the estimation of properties of the input data (cardinalities and constraints), and of the operating environment (CPU, disk access) [6]. Most query optimization strategies rely heavily on accurate cardinality estimation based upon statistics such as value histograms [9]. There is a substantial body of research that has been dedicated to the optimization problem in relational database management systems [1], and more recently, in XML data sources [4][16][17]. While no algorithms or methods have been found in the literature for query optimization over ontologies, query algebras suitable for performing algebraic optimizations of queries over RDF data models have been proposed [3][11].

In this paper we present an effective, accurate algorithm for cardinality estimation of queries posed against ontologies. Result set cardinality estimation is a critical component of query optimization, especially in the context of distributed data sources, where network traffic times constitute a substantial portion of total query execution time. The ability to decide, with a high degree of certainty, which possible query plan results in the least amount of network traffic is crucial to the performance of any information integration system based on ontologies.

2. ONTOLOGIES AND DATA MODELS

An ontology $\mathcal{O}$ can be conceptualized as the union of four sets: a set of classes C, a set of properties P, a set of individuals I, and a set of literals L:

$$\mathcal{O} = C \cup P \cup I \cup L \ . \qquad (1)$$

We define the graph of the ontology, $G(\mathcal{O})$ as a set of triples that relate individuals to each other or to literals through properties, such that

$$G(\mathcal{O}) \subset I \times P \times (I \cup L). \tag{2}$$

Individuals in the ontology are necessarily instances of one or more classes in the ontology. Properties relate individuals to each other, or relate one individual to a literal; the former are called object properties, and the latter datatype properties.

The data model of an ontology is a subset of the total ontology comprised of classes and properties that have a direct relationship with the actual data source. A mechanism to create ontology data models from different types of data sources, including relational databases and XML filesystems, has been devised. In essence, it involves the creation or identification of a set of data classes C_D that correspond to all relevant element types within the structure of the data source, i.e., tables and columns in a relational database or node types in an XML document; and the conceptualization of individuals corresponding to each relevant element, that is, table cells in a database or nodes in XML. A class $c \in C_D$ if and only if there is a non-empty set of individuals I_c that are instances of class c and that are not instances of any subclass of c. Further, all individuals in the ontology must also belong to the set of individuals in the data model, and all individuals $i \in I$ are instances of one and only one class $c \in C_D$. A set of data properties P_D is also created to reflect actual relationships within the underlying data source, such as node edges in XML or table-column links in relational databases.

All individuals that belong to an ontology also belong to its data model. The data model of the ontology, then, can be defined as the union of the set of data classes, data properties, and individuals:

$$\mathcal{O}_D = C_D \cup P_D \cup I. \tag{3}$$

The classes and properties in the ontology that are not part of the data model may represent additional concepts or aggregations; these additional classes and properties are related to those in the data model through equivalence and subsumption relations.

3. QUERIES AND QUERY PATTERNS

The World Wide Web Consortium (W3C) has recently produced a Recommendation for a Web Ontology Language (OWL) [14], as a vocabulary extension to its Resource Description Framework (RDF) [8]. OWL is fast becoming the standard for ontology representation in the Web and beyond. We are particularly concerned with the DL fragment of OWL, where classes, properties, and individuals are disjoint sets. To address the need for querying over information modeled as ontologies, a number of RDF/OWL query languages have been developed [2][10][13], and the W3C has published a Last Call Working Draft for an RDF query language and protocol called SPARQL [11]. SPARQL is a declarative language, designed specifically to obtain information from RDF graphs.

In SPARQL, a query is a tuple consisting of a query pattern, a set of graphs, a set of solution modifiers, and a result form [11],

$$Q = \{QP, GS, SM, RF\}. \tag{4}$$

Queries in SPARQL are posed against a set of RDF graphs by attempting to match query patterns against this set of graphs, processing these matchings through solution modifiers, and preparing a result set according to the result form specified. The size or cardinality of this result set is given by the number of different individuals that match the variables specified by the result form. The objective of the algorithm presented here is to estimate this result set cardinality for the query Q, which we denote as $|Q|$.

For simplification purposes, the remainder of this paper deals with queries against a single graph, without solution modifiers, and with a result form that specifies a set of variable bindings to be returned. Extensions to this model to incorporate multiple graphs and modifiers is left for future work.

The basic query pattern in SPARQL is a set of triple patterns mixed with value constraints. The set of triple patterns is of the form

$$TP = (I \cup V) \times (P \cup V) \times (I \cup L \cup V). \tag{5}$$

where V is a set of variables. In other words, a triple pattern is a member of the graph of the ontology where zero or more of its three components is substituted by a variable. The three elements of a triple pattern are called the *subject*, *property*, and *object*, respectively.

Value constraints are Boolean expressions used to limit the allowable matchings of variables to literals or individuals in the ontology data set. If F denotes the set of possible Boolean expressions on values of the ontology, a conjunctive query pattern QP is a subset of the set of basic query patterns and value constraints, that is,

$$QP \subseteq (TP \cup F). \tag{6}$$

SPARQL defines two operators on query patterns besides conjunction: OPTIONAL and UNION. The cardinality of the result set of query patterns joined together through the UNION operator is the sum of the cardinalities of each pattern. Also, for cardinality estimation purposes, an OPTIONAL pattern can be conservatively substituted by a UNION and a conjunction: given two query patterns QP1 and QP2, the cardinality of QP1 OPTIONAL QP2 will be less than or equal to the cardinality of QP1 UNION (QP1.QP2). The rest of this paper deals only with conjunctive patterns.

A query pattern defined against an ontology $\mathcal{O}$ must first be re-written so that all terms in the query pattern refer to terms contained in the ontology data model $\mathcal{O}_D$. If a triple pattern t in a query pattern contains a term that is not in the data model $\mathcal{O}_D$, and if t cannot be transformed into a set of semantically equivalent triple patterns that only contain terms in $\mathcal{O}_D$, then the result set of the evaluation of t is empty. The design of algorithms for query transformation from terms in $\mathcal{O}$ to terms in $\mathcal{O}_D$ is a matter for future work and outside the scope of this paper.

4. QUERY PATTERN PATHS

A triple $t = (s, p, o)$, where o is a literal or individual, can be substituted by a triple $t' = (s, p, v)$ and a value constraint $(v = l)$, where v is a variable different from any other variable in the query pattern. A similar substitution can be made if the subject s is a literal or individual. Thus, after such substitutions, all triples in a query pattern can be considered to have only variables as subject and object.

A subset of a query pattern QP is considered a structural query pattern path for an ordered set of variables $W \subset V$ and an ordered set of properties or variables $R \subset (P \cup V)$, denoted $QPP(W,R)$, if it is of the form

$$QPP(W,R) = \left(v_1\, p_{12}\, v_2 \,.\, \cdots \,.\, v_{n-1}\, p_{(n-1)n}\, v_n\right) \qquad (7)$$

where $W=\{v_1 \ldots v_n\}$, $R=\{p_{12} \ldots p_{(n-1)n}\}$, and $\forall i,j,\ v_i \neq v_j$.

A maximal query pattern path $QPP^M(W,R)$ is one where there do not exist $W' \supset W$ and $R' \supset R$ such that $QPP(W',R') = QPP((W'-W),(R'-R)).QPP^M(W, R)$ is also a valid query pattern path. A literal query pattern path $LQPP(W,R)$ is one where the last variable in the path matches with literals rather than individuals.

For a structural query pattern path $QPP(W,R)$, every $v_k \in W$ has a (possibly empty) set of value constraints $F(v_k)$ consisting of every $f(v_i) \in F$ such that there exists a query pattern path from some $v_k \in W$ to v_i which does not contain any triple pattern that is also in $QPP(W,R)$. The total set of value constraints for query pattern $QPP(W,R)$ is then

$$F(QPP(W,R)) = \bigcup_{v_k \in W} F(v_k)\,. \qquad (8)$$

A complete query pattern path $QPP_C(W,R)$ is the union of a structural query pattern path and its total set of value constraints. The result set of a complete query pattern path, denoted as $RS(QPP_C(W,R))$, is given by the individuals that match the triples in the path, and where every variable $v_i \in W$ fulfills its set of value constraints. A complete maximal query path $QPP^M{}_C(W)$ is a complete query pattern path that contains a maximal query pattern path.

Given a query Q with query pattern QP and result form RF, the total result set according to QP for any variable v in RF, denoted as $RS(v,QP)$, is given by the intersection of the result sets for each complete maximal query path $QPP^M{}_C(W,R)$ where $v \in (W \cup R)$, that is

$$RS(v,QP) = \bigcap_{v \in (W \cup R)} RS(QPP_C^M(W,R)) \qquad (9)$$

The cardinality of $RS(v,QP)$ is upper-bound by the minimum cardinality of all $QPP^M{}_C(W,R)$.

The result set of a query Q, $RS(Q)$, is given by the union of the result sets for each of the variables $v_i \in RF$; the size of this result set, and therefore the cardinality of the query Q, is given by

$$|Q| = |RS(Q)| = \left| \bigcup_{v_i \in RF} RS(v_i, QP) \right|. \qquad (10)$$

The problem of estimating the cardinality of a query, then, can be solved by finding accurate estimates of each possible maximal complete query pattern path on every variable v in RF.

5. CARDINALITY ESTIMATION

5.1 Estimation Function

We define an estimation function for the cardinality of a maximal complete query pattern path as a probability distribution:

$$E(QPP_C^M(W,R)) = e(QPP_C^M(W,R)) \pm k\Delta e(QPP_C^M(W,R))\,. \qquad (11)$$

where e is the expected estimation value, Δe is a measure of potential error, and k is a tunable error factor. The actual estimate of cardinality is then calculated by choosing a k appropriate to the specific application: a k of zero chooses the most accurate estimate, but risks under-estimating cardinalities for query paths with large potential errors; a positive or negative k give a more conservative or more aggressive estimate, respectively.

We next consider the estimation of cardinality for two specific cases: the case where a maximal complete query pattern path is a structural path, i.e., it has no value constraints, and the case of a maximal complete query pattern path that is a lexical path with a set of constraints only on its last variable.

5.2 Estimation over Structural Query Pattern Paths

Structural query pattern paths define constraints only on the structure of the underlying data sources. To estimate cardinalities based on these structural constraints, statistics are kept on the properties comprising the data model.

For every property $p \in \mathcal{O}_D$, its total property cardinality, denoted as $|p|$, is defined as the total number of triples in the graph of the data model $G(\mathcal{O}_D)$ that contain p.

Further, for a given triple $t_{ab} = (i_a, p_{ab}, i_b)$ in the graph, the dependent property cardinality of a property p_{bc} with respect to t_{ab}, denoted $|t_{ab}, p_{bc}|$, is defined as the total number of triples $t_{bc} = (i_b, p_{bc}, i_c)$ in $G(\mathcal{O}_D)$ that follow

$t_{ab} = (i_a, p_{ab}, i_b)$, where t_{bc} is said to follow t_{ab} if and only if the object of triple t_{ab} is equal to the subject of triple t_{bc}.

Given two properties p_i, p_j, then, the <u>mean</u> of the dependent property cardinality of p_i with respect to p_j is given by

$$\mu(p_i, p_j) = \frac{1}{|p_i|} \sum_{t_i \in \texttt{extent}(p_i)} |t_i, p_j| . \tag{12}$$

The extent of a property, denoted $\texttt{extent}(p)$, is defined as the set of triples (i_a, p, i_b) in the graph that contain p. The <u>variance</u> of the dependent property cardinality is given by

$$\sigma^2(p_i, p_j) = \texttt{abs}\left(\left(\frac{1}{|p_i|} \sum_{t_i \in \texttt{extent}(p_i)} |t_i, p_j|^2 \right) - \mu^2(p_i, p_j) \right) \tag{13}$$

where abs is the absolute value. The <u>standard deviation</u> $\sigma(p_i, p_j)$ of the dependent property cardinality is given by the positive square root of the variance.

The distribution function of p_i with respect to p_j gives a probability estimate of the number of triples containing p_j that follow each specific triple containing p_i, and is defined as

$$D(p_i, p_j) = \mu(p_i, p_j) \pm k\sigma(p_i, p_j) . \tag{14}$$

Given a query pattern path $QPP(W,R)$, if p_i denotes the ith element in R, then the distribution function of the path is the product of the distribution functions of each property in the path,

$$D(QPP(W,R)) = \prod_{p_i, p_{i+1} \in R} D(p_i, p_{i+1}) . \tag{15}$$

If p_1 is the first property in R, then the estimate of the total cardinality of each structural query pattern path, denoted $E(QPP(W,R))$, is

$$E(QPP(W,R)) = |p_1| D(QPP(W,R)) . \tag{16}$$

and thus

$$e(QPP(W,R)) = |p_1| \prod_{p_i, p_{i+1} \in R} \mu(p_i, p_{i+1}) . \tag{17}$$

$$\Delta e(QPP(W,R)) = ke(QPP(W,R)) \left[\sum_{p_i, p_{i+1} \in R} \left(\frac{\sigma^2(p_i, p_{i+1})}{\mu^2(p_i, p_{i+1})} \right) \right]^{1/2} . \tag{18}$$

5.3 Estimation over Value Constraints

Value constraints in SPARQL can be defined against variables that match to literals or individuals, or where the type of object to be matched is unknown. A particularly important case of value constraints on variables matched to individuals is the test of whether an individual is an instance of a class; in SPARQL, this is done through a triple pattern using the pre-defined property `rdf:type`. The cardinality of such a value constraint can be kept directly by calculating class cardinality statistics. Other value constraints defined against matchings to individuals defined in [11] have less incidence on result set cardinality; their incorporation into our estimation model remains for future work.

In a literal query pattern path $LQPP(W,R)$, the last property in the ordered set R, $p_{(n-1)n}$, is by definition a datatype property. To estimate the cardinality of a complete literal query pattern path $LQPP_C(W,R)$ consisting of a literal path $LQPP(W,R)$ and a set of value constraints $F(v_n)$ over the last variable in W, statistics are also kept on the distribution of values for the objects of these datatype properties, in the form of value histograms.

Histograms are widely used in relational database systems to represent the data distribution of values on a table column [1]. In the ontology data model $\mathcal{O}_D$, values are conceptualized as the objects of datatype properties; thus, for each datatype property $p \in \mathcal{O}_D$, an equi-depth histogram is constructed to represent its distribution. The total number of buckets B in each histogram is determined according to the following:

$$B = \texttt{min}(|p|_{unique}, B_{\max}, \frac{|p|}{d}) . \tag{19}$$

where as before $|p|$ is the total property cardinality of p, and $|p|_{\text{unique}}$ represents the total number of unique values in the range of p. Both B_{max} and d are tunable parameters of the histograms; B_{max} indicates the maximum number of buckets that should be constructed for any histogram (to avoid excessive memory and storage space consumption), and d indicates the expected depth of each frequency in the histogram. At a minimum, a number of buckets equal to the total number of unique values is created.

The set of value constraints $F(v_n)$ over a variable v_n is processed against the histograms for the property $p_{(n-1)n}$, resulting in a <u>value ratio</u> $\rho(F(v_n))$ that indicates the estimated proportion of the possible value bindings for variable v_n that satisfy the value constraints. Note that all values satisfy an empty value constraint, and thus, if $F(v_n)$ is the empty set, $\rho(F(v_n)) = 1$.

The total cardinality of a complete literal query pattern path is then given by the product of the estimate of cardinality for the literal query pattern path and the value ratio for the end variable in the query,

$$E(LQPP_C(W,R)) = \rho(F(v_n)) \cdot E(LQPP(W,R)) . \tag{20}$$

5.4 General Cardinality Estimation of Maximal Complete Query Pattern Path

In the general case, a maximal complete query pattern path $QPP^M{}_C(W,R)$ consists of a maximal query pattern path $QPP^M(W,R)$ and a set of value constraints $F(QPP(W,R))$. The total cardinality estimate of such a path is obtained by

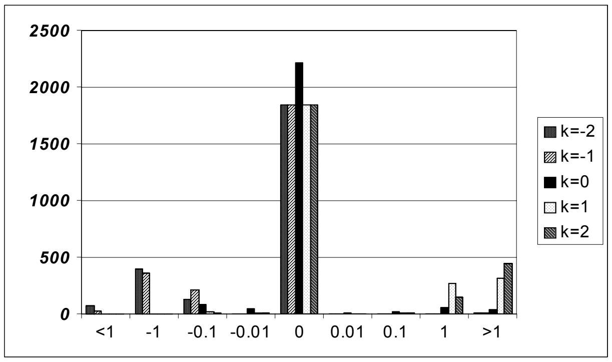

Figure 1. Histograms of estimated vs. real cardinality difference ratio

calculating the product of the cardinality estimate of its maximal query path with all the value ratios for every variable in the query pattern path:

$$E(QPP_C^M(W,R)) = E(QPP^M(W,R)) \cdot \prod_{v_k \in W} \rho(F(v_k)) \quad (21)$$

6. EXPERIMENTAL RESULTS

In order to validate the algorithm for cardinality estimation of semantic queries presented here, we obtained two XML data sets using the `eFetch` utility from the Entrez Gene website maintained by the National Center for Biotechnology Information (NCBI). Both datasets contains data from the `Gene` database; the dataset #1 contains data for a `geneId` of 2, while dataset #2 contains data for a sample set of six ATP-binding cassette transporter genes. Additionally, we also used an XML dataset obtained from a collection form in a mental health assessment system [15].

Each of these three datasets was odeled as an ontology in these cases, the data model of the ontology is equivalent to the ontology itself. Experimental analysis was then carried out over these models, to determine the validity of the cardinality estimation algorithm over both structural query pattern paths and value constraints.

6.1 Estimation over Structural Query Pattern Paths

To analyze the performance of the algorithm over structural constraints, cardinality was estimated for all possible query pattern paths containing between 2 and 4 properties, using error factor *k* of -2, -1, 0, 1, and 2. Each of these estimates was then compared with the real cardinality by taking the difference ratio, that is, the difference divided by the real cardinality value. Histograms of this difference ratio were obtained for each error factor, as shown in Figure 1. Aside from a few outlying values, in most cases this difference ratio is close to zero, indicating the accuracy of the estimation. The deviation of error factor from zero produces less accuracy, resulting in conservative estimates for positive error factors and aggressive estimates for negative factors.

Most importantly, the correlation between the actual and estimated total number of paths is very high, showing that the algorithm identifies, with high accuracy, the relative size between two query pattern paths. The correlation decreases as the error factor deviates from zero, also as expected; correlation also decreases, but is still very high, as the number of properties increases.

The degree of correlation between actual and estimate is highlighted in Table 1, where we present aggregate correlations for all possible query pattern paths in all datasets containing between 2 and 4 properties, and for all possible query plans in all datasets.

Table 1. Correlation between actual and estimated total number of paths

	Number of properties in path			
	2	**3**	**4**	**All**
Dataset #1	1.0000	0.9949	0.9785	0.9921
Dataset #2	1.0000	0.9997	0.9982	0.9994
Dataset #3	1.0000	0.9954	0.9214	0.9646
All datasets	1.0000	0.9994	0.9955	0.9985

6.2 Estimation over Value Constraints

To validate the use of histograms for the estimation of the cardinality of query pattern paths including value constraints, a total sampling of twenty different queries was chosen, including queries over ranges of values as well as queries over value equalities. A maximum bucket size of 5 was used, and in a few cases queries were redone using bucket sizes of 10 and 25. The aggregate results over this sampling are shown in Table 2. As can be observed, even with such a small number of buckets the estimates yield accurate results, and, more importantly, there is a high correlation between the estimates and the actual totals.

7. CONCLUSIONS AND FUTURE WORK

An algorithm for the estimation of cardinality for queries posed against ontology representations of data sources has been presented in this paper. Experimental results show that the proposed algorithm produces accurate estimates of cardinality, and more importantly, that the estimation of relative size among two queries is highly precise.

The estimation of the cardinality of a query is used to approximate the data transfer times of the result set, as part of the estimation of the total cost of executing a query. In a highly distributed architecture where data sources are located at diverse locations connected through the Internet, this is the most critical aspect of query execution time; however, in settings where very high speed connections are

Table 2. Statistics for estimation including value constraints

Number of different paths	20
Actual total number of paths	796
Estimated total number of paths	760
Correlation of estimated to actual	0.9863
Ratio of difference to total	
Average	0.03
Maximum	0.48
0.9 percentile	0.23
0.1 percentile	-0.20
Minimum	-0.42

available, other costs, such as CPU and disk access times, become significant; we are currently investigating the effects that the modeling of data sources as ontologies has on CPU execution times.

The estimation of the total cost of query execution is in turn used to select an optimal (or at least highly efficient) query plan. Algorithms for determining the set of different possible plans to be considered for selection are also currently under development. Work is also underway for the incorporation into our estimation model of multiple graphs and solution modifiers in a query, and of value constraints on matchings of variables to individuals.

A query optimization strategy is crucial to obtain reasonable performance over queries against ontology data models, especially if they are done over a highly distributed architecture. The algorithm proposed here is an important component for the construction of an efficient querying engine over ontology-modeled, distributed, heterogeneous data sources.

ACKNOWLEDGEMENTS

This work is supported by NIH grant R43RR018667. The authors also wish to acknowledge the contribution of Mr. Thomas Taylor and Dr. Akmal Younis of INFOTECHSoft, Inc.

REFERENCES

[1] Chauduri S. An Overview of Query Optimization in Relational Systems. Proc. of the 17th ACM Symp on Principles of Database Systems, Seattle, WA, USA, 1998:34-43.

[2] Chong EI, Das S, Eadon G, Srinivasan J. An efficient SQL-based RDF querying scheme. Proc. of the 31st Intl. Conf. on Very Large Data Bases, Trondheim, Norway. 2005:1216-1227.

[3] Frasincar F, Houben G-J, Vdovjak R, Barna P. RAL: An algebra for querying RDF. World Wide Web 2004;7(1):93-109.

[4] Freire J, Haritsa JR, Ramanath M, Roy P, Siméon J. StatiX: Making XML Count. Proc. ACM SIGMOD, 2002 Jun 4-6, Madison, WI, USA.

[5] Gruber TR. Toward principles for the design of ontologies used for knowledge sharing. Technical report KSL 93-04, Knowledge Systems Laboratory, Stanford University, Available from: ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-93-04.ps.gz.

[6] Ives ZG, Halevy AY, Weld DS. Adapting to source properties in processing data integration queries. Proc ACM SIGMOD Intl Conf on Mgmt of Data. 2004:395-406.

[7] Kohler J, Philippi S, Lange M. SEMEDA: ontology based semantic integration of biological databases. Bioinformatics. 2003 Dec 12;19(18):2420-2427.

[8] Manola F, Miller E, editors. RDF Primer. W3C Recommendation [updated 2004 Feb 10, accessed 2006 Mar 21]. Available from: http://www.w3.org/TR/rdf-primer/.

[9] Markl V, Raman V, Simmen D, Lohman G, Pirahesh H, Cilimdzic M. Robust query processing through progressive optimization. Proc ACM SIGMOD Intl Conf on Mgmt of Data. 2004:659-670.

[10] Pérez de Laborda C, Conrad S. Querying Relational Databases with RDQL. Berliner XML Tage 2005. [Accessed 2006 Mar 21]. Available from: http://dbs.cs.uni-duesseldorf.de/~perezdel/pdf/05PeCob.pdf.

[11] Prud'hommeaux E, Seaborne A, editors. SPARQL Query Language for RDF, W3C Working Draft [updated 2007 Mar 26, accessed 2007 Jun 11]. Available from: http://www.w3.org/TR/rdf-sparql-query/.

[12] Rodriguez MA, Egenhofer MJ. Determining Semantic Similarity among Entity Classes from Different Ontologies. IEEE Trans on Knowledge and Data Eng, 2003 15(2):442-456.

[13] Sattler K-U, Geist I, Schallehn E. Concept-based querying in mediator systems. The VLDB Journal. 2005;14(1):97-111.

[14] Smith MK, Welty C, McGuiness DL editors. OWL Web Ontology Language Guide, W3C Recommendation [updated 2004 Feb 10, accessed 2006 Mar 21]. Available from: http://www.w3.org/TR/owl-guide/.

[15] Taylor TJ, Kabuka MR, Shironoshita EP, Ryan MT, Younis AA, John, NM, et.al. Viability of Mental Health Assessment Software in Diverse Settings. 45th Annual NCDEU (New Clinical Drug Evaluation Unit), Boca Raton, FL, USA. June 6-9, 2005.

[16] Wu Y, Patel JM, Jagadish HV. Structural Join Order Selection for XML Query Optimization. Proc. of the 19th Intl Conf on Data Engineering. 2003 Mar 5-8: 443-454.

[17] Zhang N, Ozsu MT, Aboulnaga A, Ilyas IF. XSeed: accurate and fast cardinality estimation for XPath queries. [Accessed 2006 Mar 21]. Available from: http://www.cs.uwaterloo.ca/~ilyas/papers/synopsis.pdf.