Query Answering Engine for the Next Generation Database Systems

Nittaya Kerdprasop, Kittisak Kerdprasop

Data Engineering Research Unit, School of Computer Engineering,
Suranaree University of Technology, Nakhon Ratchasima, Thailand

Corresponding Author: nittaya@sut.ac.th

Abstract

The first generation of databases has emerged since 1970's as hierarchical and network systems that unified data collection and manipulation as a single set of data records. The second generation has started since the early 1980 with the concept of relational databases and some variations such as object databases and deductive databases. The idea of moving databases toward the next generation has proposed since the 1990's as a better system to support new kinds of objects structures and business rules. With the advancement of data mining and learning techniques, we propose that the next generation database systems should include the learning capability. We therefore introduce the design of a query answering system that can support the database system to be more intelligent than the current systems. We also conduct some experiments to show the performance of the proposed query rewriting system.

Keywords: Next generation database systems, query optimization, intelligent query answering, data mining.

1. Introduction

Querying a database is a common task for traditional database systems. To query a database is to find some answers from stored data. Traditional database systems return exactly what is being asked. This is a method of direct query answering and a user is required to construct a query intelligently and properly. To remove the burden of intelligence from the database users, the concept of intelligent or cooperative query answering has emerged(1,2). The process of intelligent query answering consists of analyzing the intent of query, rewriting the query based on the intention and other kinds of knowledge, and providing answers in an intelligent way(3). Intelligent answers could be generalized, neighborhood or associated information relevant to the query. This concept is based on the assumption that some users might not have a clear idea of the database content and schema. Therefore, it is difficult to pose queries correctly to get some useful answers.

Knowledge, either intentional or extensional, is the key ingredient of intelligence. Many researchers(2-4) propose to integrate data mining techniques as a knowledge discovery engine to serve an intelligent query answering purpose. We extend this idea by incorporating both virtual mining and materialized views in the query answering system. Virtual mining views(5,6) are data mining rules discovered from databases and stored as tables, whereas materialized views are view relations computed and stored in the database as well. We consider virtual mining and materialized views as semantic constraints capable of transforming queries to be processed intelligently.

The idea of coupling data mining capability with the database systems gives rise to the new concept of *inductive databases*(7-9). An inductive database is a database that contains not only data, but also patterns which are generalized information induced from data. By providing this tightly integration framework of data management system and pattern discovery engine, users can access patterns in the same manner as querying data. To achieve this aim a number of SQL-based inductive query languages, such as DMQL(10), MINE RULE(11), MSQL(12), have been proposed and implemented. Most of these languages are an SQL extension with some primitives to support the data mining task, that is, users can pose queries to induce, access and update patterns.

Besides the front-end functionalities, we propose that the induced patterns can also be useful in the back-end part of query answering. The induced patterns are viewed as a repository of semantic knowledge highly beneficial to the optimization process. The purpose of query optimization is to rewrite a given query into an equivalent one that uses less time and resources. Equivalence is defined in terms of identical answer sets. Query optimization utilizes syntactic and logic equivalence transformation of a given query expression. Semantic query optimization (SQO), on the contrary, uses not only syntactic transformations, but also semantic knowledge, such as integrity constraints and various forms of data generalization, to transform a query into an optimized one.

Early work on SQO(13,14) transforms query by reasoning via heuristics such as index and restriction introduction, join elimination, contradiction detection. Since the introduction of SQO concept in 1981(14), semantic-based transformation techniques have been developed constantly. Some proposed techniques in the literature are resolution refutation method(15) and knowledge deduction(16). Recently, the interest on SQO has moved toward the setting of intelligent query answering(2,3), which is defined as a procedure that can answer incorrect or incompletely specified query cooperatively and intelligently. The intelligence is obtained by analyzing the intent of a query and provide some generalized or associated answers. Necib and Freytag(17) propose an ontology-based optimization approach to rewrite a query into another one which is not necessary equivalent but can provide more meaningful result satisfying the user's intention.

Our research follows the direction of intelligent query answering with the emphasis on semantic-based optimization. We consider acquiring semantic knowledge using a data mining approach in which knowledge can be induced from the database content. The main purpose of this paper is to illustrate the idea of inducing and integrating knowledge in the query rewriting and optimization process to produce an intelligent answer. We present the optimization process within the framework of inductive database systems that both data content and patterns are stored in the databases. The rest of the paper is organized as follows. Section 2 presents some major concepts and related work. Section 3 explains architecture of the proposed query answering system. Section 4 discusses the implementation and some experimental results. Section 5 concludes this paper.

2. Preliminary and Related Work

Inductive databases can be viewed as an extension of the traditional database systems in that the databases do not only store data, but they also contain patterns of those data.

$\mathcal{R}$

X	Y	Z
1	0	0
1	1	1
1	0	1
0	1	1

$\mathcal{P}$

pattern	support	confidence
$X \Rightarrow Y$	0.25	0.33
$X \Rightarrow Z$	0.50	0.66
$Y \Rightarrow X$	0.25	0.50
$Y \Rightarrow Z$	0.50	1.00
$Z \Rightarrow X$	0.50	0.66
$Z \Rightarrow Y$	0.50	0.66
$XY \Rightarrow Z$	0.25	1.00
$XZ \Rightarrow Y$	0.25	0.50
$YZ \Rightarrow X$	0.25	0.50

Fig. 1. An example of inductive database instance

Mannila(18) formalized a framework of inductive database I as a pair (R, P) where R is a database relation and P is a nested relation of the form (Q_R, e) in which Q_R is a set of patterns obtained from querying the base data and e is the evaluation function measuring some metrics over the patterns. As an example, consider the database (adapted from(19)) consisting of one base relation, R. The induced patterns P are a set of rules represented as an implication LHS $\Rightarrow$ RHS; therefore, $Q_R = \{ \text{LHS} \Rightarrow \text{RHS} \mid \text{LHS}, \text{RHS} \subseteq R\}$ and the rule's quality metrics are support and confidence(20). An inductive database I = (R, P) containing one base relation R and a set P of all association patterns induced from R is shown in Fig. 1.

Given the framework of an inductive database I, users can query both the stored data (the part of $I.R$ in Fig. 1) as well as the set of patterns (the $I.P$ part). Formalization of inductive queries to perform data mining tasks has been studied by several research groups(7,21). We are, however, interested in the concept of inductive databases from a different perspective. Instead of using a sequence of queries and operations to create the induced patterns such as association rules(20), we shift our focus towards the induction of rules and then deploy the stored information to support query answering.

Evaluating queries efficiently and intelligently requires an important step of query rewriting and modification. Query rewriting is a basic step in query processing aiming at transforming a given query into another more efficient one that uses less time and resources to execute. A rewritten query normally produces the same answer set as the original query.

Query modification(22) interprets query rewriting in a more relaxing way as a query refining process to produce answers that might be a superset of the expected answers.

The advantage of query relaxation is the increased possibility of obtaining desired answers when users have limited knowledge about the problem domain and the database schemas.

Early research in query modification[1,22] has focused on rewriting the query using generalization concept, neighborhood, and type abstraction hierarchy. The work of Han et al[17] is among the early research in intelligent query answering that incorporates data mining techniques to rewrite users' queries. Their query relaxation approach employed the notion of generalization to build concept hierarchy.

Lin et al[18] proposed to integrate neighborhood information and data mining rules discovered from the databases to rewrite the queries. Muslea[23] introduced the LOQR algorithm to learn some knowledge about the problem domain using a small subset of the database. Then the learned information is used to relax the constraints in the query that originally returns an empty answer.

Aragao and Fernandes[4] proposed a unified foundation for query answering and knowledge discovery. The combined system is called CIDS (Combined Inference Database Systems).

The integration of knowledge discovery and query answering system is also the basis of our research. However, we propose to extend the idea by incorporating not only the knowledge discovered from databases, but also the materialized views in the process of query rewriting and answering.

Materialized views are pre-computed data that are stored in the database. Answering queries using views has long been extensively studied[24-28]. Materialized views can provide useful information in query processing especially in the context of web searching applications. We thus design our system to employ both learned knowledge and materialized views to refine the given query.

3. The Design of Query Answering System

The proposed query answering system composed of a number of autonomous modules working cooperatively to pursue the common goal of rewriting and producing answers in an intelligent way. The general framework of these modules is depicted in Fig. 2. Our design of intelligent query answering system includes the semantic optimizer to utilize materialized views and data mining models as major sources of knowledge in semantically transforming the users' queries. The framework of the optimizer is provided in Fig. 3

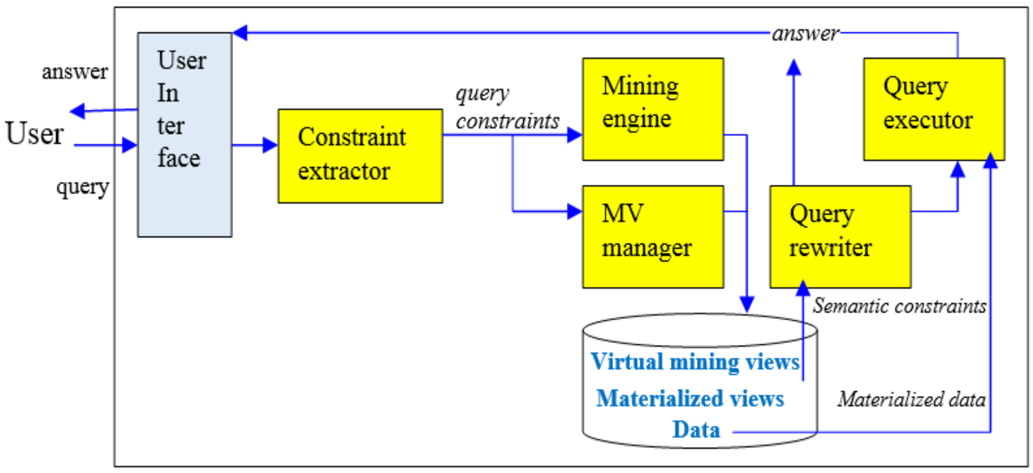


Fig. 2. A general framework of the inductive database.

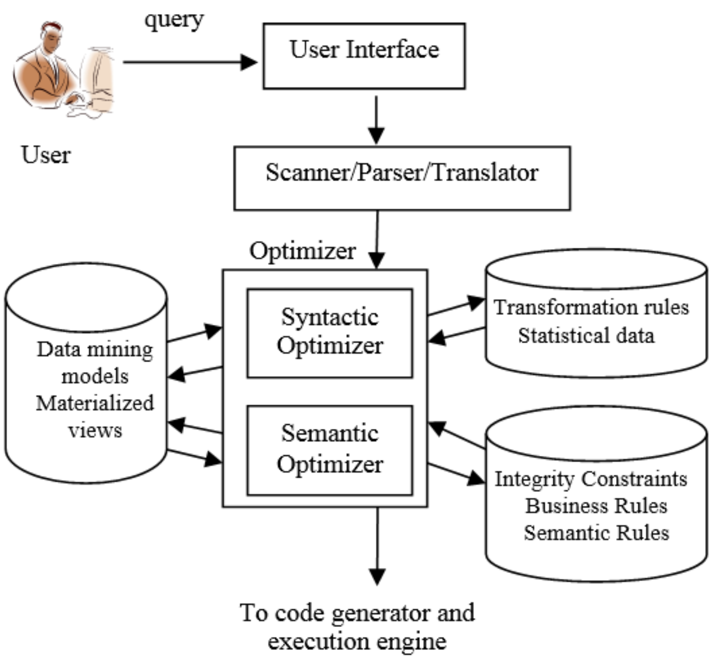

query
User Interface
User
Scanner/Parser/Translator
Optimizer
Syntactic Optimizer
Semantic Optimizer
Data mining models Materialized views
Transformation rules Statistical data
Integrity Constraints Business Rules Semantic Rules
To code generator and execution engine

Fig. 3. A framework of query processor and optimizer.

User interface is a front-end module to get query from the user and return a final answer set. Constraint extractor is responsible for extracting constraints from the original query and inputs these constraints to the mining engine and the materialized view (MV) manager. A mining engine is thus driven by the query constraints to discover knowledge such as association rules[20] that are relevant to the given query. MV manager is a module responsible for view creation, selection and modification. The data storage thus contains three kinds of information: base relations (data), materialized views created by the MV manager, and virtual mining views discovered by the mining engine.

Materialized view definitions and virtual mining views are to be used as semantic constraints by the query rewriter in transforming the given query. Some queries can be answered at this stage, whereas the more complicate ones are sent to the query executor in which base relations and materialized data might be accessed. The algorithm for query optimizer is given in Fig. 4.

```
Input:   a database D,
         a set of semantic rules S,
         a set of materialized views V,
         current user's query Q
Output:  a new query Q′
Steps:
1.    Extract conditions C from the user's query Q
2.    For each $c \in C$
3.        Search for applicable semantic rules from S by
3.1           assert $c$ as a temporary database fact
3.2           search for predicates in S that are related to $c$
3.3           report searching result as an answer set A
4.    If A is empty, then return Q; otherwise proceed to
          the next step
5.    Form a new query Q′ by
5.1       Construct a head of query clause with C
              appeared as arguments
5.2       Construct clause body with applicable
              materialized view from V
5.3       Conjunct a clause body with predicates
              appeared in A
6.    Return a new query Q′
```

Fig. 4. Semantic query optimizer algorithm.

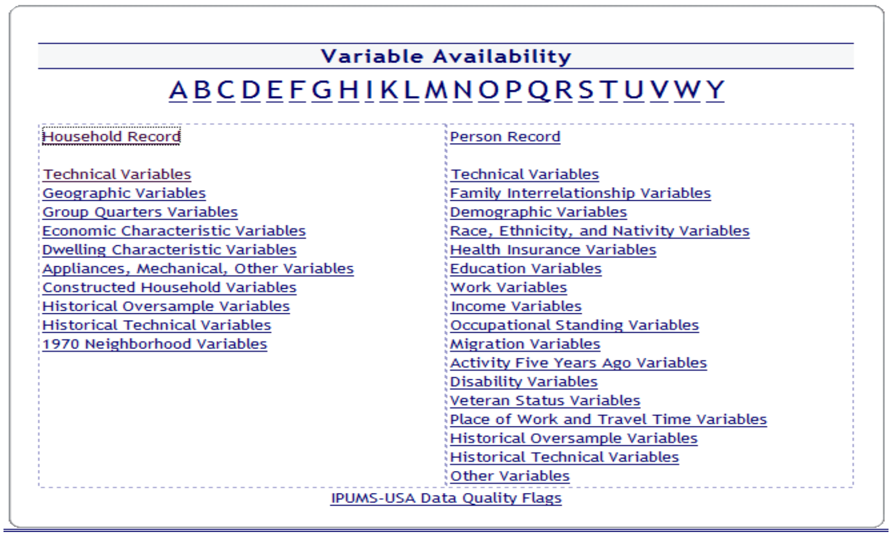
4. Experimentation and Results

On a series of experimentation, we draw a sample set containing 1,000 data records from the IPUMS-USA database which has been made available to public by Minnesota Population Center[29]. This database contains the United States socio-economic data of the year 1999 with household and person information as shown in Fig. 5.

The household records contain information such as value of household unit, monthly rental payment, family total income, number of families within each household unit, age of a person, relationship to household head, and other related information. Examples of household records are illustrated in Fig. 6.

For the person records, the information is about education, employment status, occupation, income of a person, and other personal information. Examples of person records are shown in Fig. 7.

To test the efficiency of query optimization with materialized views and data mining models, we create a database using the DES system[30]. Data mining models used in the experimentation are association rules.


Variable Availability

A B C D E F G H I K L M N O P Q R S T U V W Y

Household Record	Person Record
Technical Variables	Technical Variables
Geographic Variables	Family Interrelationship Variables
Group Quarters Variables	Demographic Variables
Economic Characteristic Variables	Race, Ethnicity, and Nativity Variables
Dwelling Characteristic Variables	Health Insurance Variables
Appliances, Mechanical, Other Variables	Education Variables
Constructed Household Variables	Work Variables
Historical Oversample Variables	Income Variables
Historical Technical Variables	Occupational Standing Variables
1970 Neighborhood Variables	Migration Variables
	Activity Five Years Ago Variables
	Disability Variables
	Veteran Status Variables
	Place of Work and Travel Time Variables
	Historical Oversample Variables
	Historical Technical Variables
	Other Variables

IPUMS-USA Data Quality Flags

Fig. 5. Variables as appeared in the household and person records.

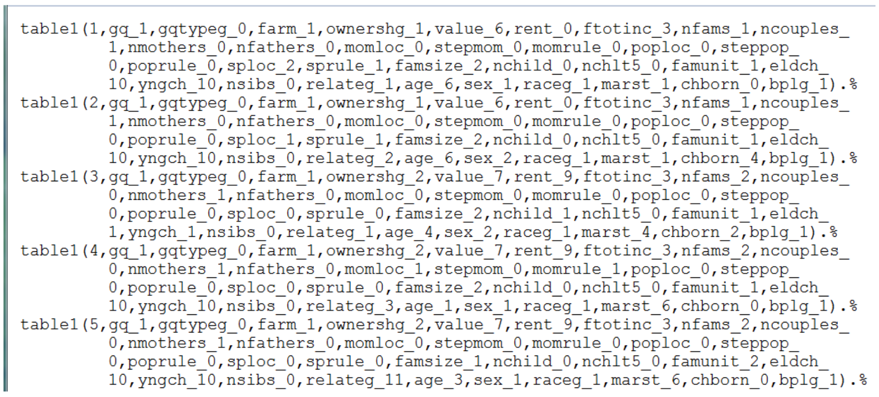

```
table1(1,gq_1,gqtypeg_0,farm_1,ownershg_1,value_6,rent_0,ftotinc_3,nfams_1,ncouples_
         1,nmothers_0,nfathers_0,momloc_0,stepmom_0,momrule_0,poploc_0,steppop_
         0,poprule_0,sploc_2,sprule_1,famsize_2,nchild_0,nchlt5_0,famunit_1,eldch_
         10,yngch_10,nsibs_0,relateg_1,age_6,sex_1,raceg_1,marst_1,chborn_0,bplg_1).%
table1(2,gq_1,gqtypeg_0,farm_1,ownershg_1,value_6,rent_0,ftotinc_3,nfams_1,ncouples_
         1,nmothers_0,nfathers_0,momloc_0,stepmom_0,momrule_0,poploc_0,steppop_
         0,poprule_0,sploc_1,sprule_1,famsize_2,nchild_0,nchlt5_0,famunit_1,eldch_
         10,yngch_10,nsibs_0,relateg_2,age_6,sex_2,raceg_1,marst_1,chborn_4,bplg_1).%
table1(3,gq_1,gqtypeg_0,farm_1,ownershg_2,value_7,rent_9,ftotinc_3,nfams_2,ncouples_
         0,nmothers_1,nfathers_0,momloc_0,stepmom_0,momrule_0,poploc_0,steppop_
         0,poprule_0,sploc_0,sprule_0,famsize_2,nchild_1,nchlt5_0,famunit_1,eldch_
         1,yngch_1,nsibs_0,relateg_1,age_4,sex_2,raceg_1,marst_4,chborn_2,bplg_1).%
table1(4,gq_1,gqtypeg_0,farm_1,ownershg_2,value_7,rent_9,ftotinc_3,nfams_2,ncouples_
         0,nmothers_1,nfathers_0,momloc_1,stepmom_0,momrule_1,poploc_0,steppop_
         0,poprule_0,sploc_0,sprule_0,famsize_2,nchild_0,nchlt5_0,famunit_1,eldch_
         10,yngch_10,nsibs_0,relateg_3,age_1,sex_1,raceg_1,marst_6,chborn_0,bplg_1).%
table1(5,gq_1,gqtypeg_0,farm_1,ownershg_2,value_7,rent_9,ftotinc_3,nfams_2,ncouples_
         0,nmothers_1,nfathers_0,momloc_0,stepmom_0,momrule_0,poploc_0,steppop_
         0,poprule_0,sploc_0,sprule_0,famsize_1,nchild_0,nchlt5_0,famunit_2,eldch_
         10,yngch_10,nsibs_0,relateg_11,age_3,sex_1,raceg_1,marst_6,chborn_0,bplg_1).%
```

Fig. 6. Examples of household records, each record has 34 attributes

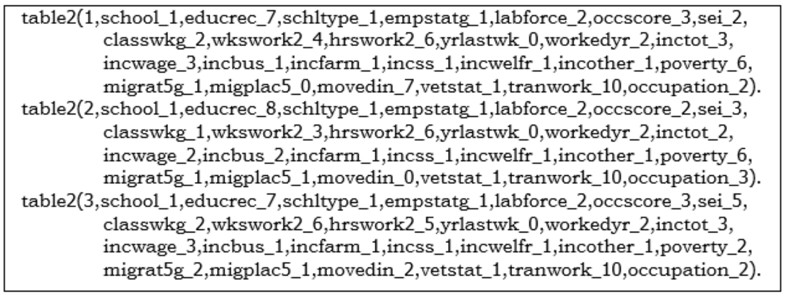

```
table2(1,school_1,educrec_7,schltype_1,empstatg_1,labforce_2,occscore_3,sei_2,
         classwkg_2,wkswork2_4,hrswork2_6,yrlastwk_0,workedyr_2,inctot_3,
         incwage_3,incbus_1,incfarm_1,incss_1,incwelfr_1,incother_1,poverty_6,
         migrat5g_1,migplac5_0,movedin_7,vetstat_1,tranwork_10,occupation_2).
table2(2,school_1,educrec_8,schltype_1,empstatg_1,labforce_2,occscore_2,sei_3,
         classwkg_1,wkswork2_3,hrswork2_6,yrlastwk_0,workedyr_2,inctot_2,
         incwage_2,incbus_2,incfarm_1,incss_1,incwelfr_1,incother_1,poverty_6,
         migrat5g_1,migplac5_1,movedin_0,vetstat_1,tranwork_10,occupation_3).
table2(3,school_1,educrec_7,schltype_1,empstatg_1,labforce_2,occscore_3,sei_5,
         classwkg_2,wkswork2_6,hrswork2_5,yrlastwk_0,workedyr_2,inctot_3,
         incwage_3,incbus_1,incfarm_1,incss_1,incwelfr_1,incother_1,poverty_2,
         migrat5g_2,migplac5_1,movedin_2,vetstat_1,tranwork_10,occupation_2).
```

Fig. 7. Examples of person records, each record has 27 attributes.

We test the query processing performance with six kinds of queries (experimental results are graphically shown in Fig. 8 and summarized in Table I). Some queries (Query1, Query2, and Query3) cannot benefit from the presence of semantic rules because the queries' conditions do not fit the rules. For this case, the system takes more time to search for semantic rules than traditional direct querying method. But for some queries that fit the rule antecedents, the intelligent method does significantly save the database searching time.

5. Conclusions

Our query answering scheme presented in this paper is based on the setting of inductive databases. An inductive database is the concept proposed as the next generation of database systems. Within this framework, data and patterns are stored together as database objects. In such tightly coupling architecture, patterns are considered first-class objects in that they can be created, accessed, and updated in the same manner as persistent data. We present the framework and techniques of query rewriting and answering that use stored patterns as semantic knowledge.

We design and implement a query answering system to provide an integrated and efficient platform for the next generation database management system. To answer queries effectively and intelligently, the association mining component and the materialized view manager are two key players to derive useful knowledge relevant to the given query. Query rewriter, which is supported by intelligent transformation rules and co-operated with query executor, is expected to produce answers in an intelligent way. The preliminary experimental results satisfy this expectation. We are, however, improving the capability of these components to analyze the user's intent and preferences to better providing associated information. Extending the scope of this research towards the distributed environment is also the direction of our future work.

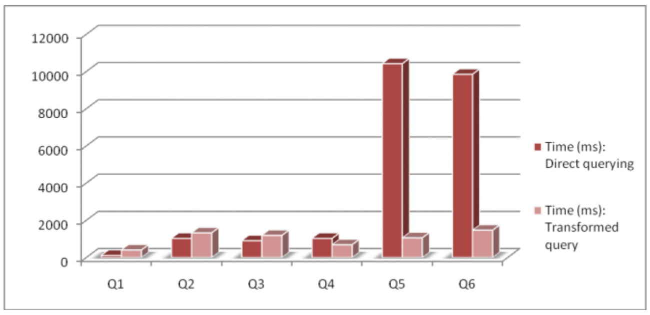


Fig. 8. Query processing time comparison of direct querying versus intelligent method using views and association rules.

Table 1. Processing time summarization of direct querying versus intelligent query answering with semantic transformation using materialized views and mining models.

Query characteristics	***Processing time (msec.)***		***Reduced time***	***Time usage efficiency***
	Direct answer	***Intelligent answer***		
Query1: ask one information with a single condition, null answer	110	405	-295	-268.18%
Query2: ask one information with a single condition	1,029	1,325	-296	-28.76%
Query3: ask one information with a single condition	906	1,188	-282	-31.12%
Query4: ask one information with two conditions	1,028	688	340	33.07%
Query5: ask two information with a single condition	10,423	1,060	9,363	89.83%
Query6: ask three information with four conditions	9,852	1,467	8,385	85.10%

Acknowledgment

This research has been conducted at the data engineering research unit, school of computer engineering, Suranaree University of Technology (SUT). This research unit is financially supported by research grants from the SUT and National Research Council of Thailand (NRCT).

References

(1) W. Chu, and Q. Chen : “A structured approach for cooperative query answering”, IEEE Transactions on Knowledge and Data Engineering, Vol.6, pp. 738-749, 1994.

(2) J. Han, Y. Huang, N. Cercone, and Y. Fu : “Intelligent query answering by knowledge discovery techniques”, IEEE Transactions on Knowledge and Data Engineering, Vol.8, No.3, pp. 373-390, 1996.

(3) T. Lin, N. Cercone, X. Hu, and J. Han : “Intelligent query answering based on neighborhood systems and data mining techniques”, Proc. International Database Engineering and Applications Symposium, pp. 91-96, 2004.

(4) M. Aragao, and A. Fernandes : “Logic-based integration of query answering and knowledge discovery”, Proc. 6th International Conference on Flexible Query Answering Systems, pp. 68-83, 2004.

(5) T. Calders, B. Goethals, and A. Prado : “Integrating pattern mining in relational databases”, Proc. 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, pp. 454-461, 2006.

(6) H. Blockeel, T. Calders, E. Fromont, B. Goethals, and A. Prado : “Mining views: database views for data mining”, Proc. 24th IEEE International Conference on Data Engineering, pp. 1608-1611, 2008.

(7) J. Boulicaut, M. Klemettinen, and H. Mannila : “Modeling KDD processes within the inductive database framework”, Proc. 1st International Conference on Data Warehousing and Knowledge Discovery (DaWaK’99), pp. 293-302, 1999.

(8) L. De Raedt : “A perspective on inductive databases”, *SIGKDD Explorations*, Vol. 4, No. 2, pp. 69-77, 2002.

(9) R. Meo : “Inductive databases: Towards a new generation of databases for knowledge discovery”, Proc. 16th International Workshop on Database and Expert Systems Applications, pp. 1003-1007, 2005.

(10) J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane : “DMQL: A data mining query language for relational databases”, Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, pp. 27-34, 1996.

(11) J. Boulicaut, M. Klemettinen, and H. Mannila : “Querying inductive databases: A case study on the MINE RULE operator”, Proc. 2nd European Symposium on Principles of data Mining and Knowledge Discovery (PKDD), pp. 194-202, 1998.

(12) T. Imielinski, and A. Virmani : “MSQL: A query language for database mining”, Data Mining and Knowledge Discovery, Vol. 2, No. 4, pp. 373-408, 1999.

(13) M. Hammer, and S. Zdonik : “Knowledge base query processing”, Proc. 6th International Conference on Very Large Data Bases (VLDB), pp. 137-147, 1980.

(14) J. King : “QUIST: A system for semantic query optimization in relational databases”, Proc. 7th International Conference on Very Large Data Bases, pp. 510-517, 1981.

(15) U. Charkravarthy, J. Grant, and J. Minker : “Logic-based approach to semantic query optimization”, ACM Transactions on Database Systems , Vol. 15, No. 2, pp. 162-207, 1990.

(16) M. Siegel, E. Sciore, and S. Salveter : “A method for automatic rule derivation to support semantic query optimization”, ACM Transactions on Database Systems, Vol. 17, No. 4, pp. 563-600., 1992.

(17) C. Necib, and J. Freytag : “Semantic query transformation using ontologies”, Proc. International Database Engineering and Applications Symposium (IDEAS) 187-199, 2005.

(18) H. Mannila : “Inductive databases and condensed representations for data mining”, Proc. International Logic Programming Symposium, pp. 21-30, 1997.

(19) F. Bonchi : “Frequent pattern queries: Language and optimizations”, Ph.D. Thesis, Computer Science Department, University of Pisa, Italy, 2003.

(20) R. Agrawal, T. Imielinski, and A. Swami : “Mining association rules between sets of items in large databases”, Proc. ACM SIGMOD, pp. 207-216, 1993.

(21) L. De Raedt, M. Jaeger, S. Lee, and H. Mannila : “A theory of inductive query answering”, Proc. IEEE International Conference on Data Mining (ICDM), pp. 123-130, 2002.

(22) S. Chaudhuri : “Generalization and a framework for query modification”, Proc. IEEE ICDE, pp. 138-145, 1990.

(23) I. Muslea : "Machine learning for online query relaxation", Proc. ACM SIGMOD, pp. 246-255, 2004.
(24) F. N. Afrati, C. Li, and J. D. Ullman : "Using views to generate efficient evaluation plans for queries", Journal of Computer and System Sciences, Vol. 73, No. 5, pp. 703-724, 2007.
(25) J. Chang, and S. Lee : "Query reformulation using materialized views in data warehousing environment", Proc. 1st ACM International Workshop on Data Warehousing and OLAP (DOLAP), pp. 54-59, 1998.
(26) S. Chaudhuri, S. Krishnamurthy, S. Potamianos, and K. Shim : "Optimizing queries with materialized views", Proc. IEEE ICDE, pp. 190-200, 1995.
(27) G. Gou, M. Kormilitsin, and R. Chirkova : "Query evaluation using overlapping views: completeness and efficiency", Proc. ACM SIGMOD, pp. 37-48, 2006.
(28) A. Halevy : "Answering queries using views: a survey", VLDB Journal, Vol. 10, No. 4, pp. 270-294, 2001.
(29) Integrated Public Use Microdata Series (IPUMS), Minnesota Population Center, http://www.ipums.org
(30) Datalog Educational System, version 2.0, http://www.fdi.ucm.es/profesor/fernan/DES/