Semantic-based Query Answering Supported by Association Patterns and Materialized Views

Nittaya Kerdprasop and Kittisak Kerdprasop

Data Engineering Research Unit, School of Computer Engineering,
Suranaree University of Technology, Nakhon Ratchasima 30000 Thailand
nittaya@sut.ac.th, kittisakThailand@gmail.com

Abstract

Querying a database is a common task for most 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 users are 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. 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. 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. Producing answers effectively depends largely on users' knowledge about the query language and the database schema. Knowledge, either intentional or extensional, is the key ingredient of intelligence. In order to improve effectiveness and convenience of querying databases, we design a systematic way to analyze user's request and revise the query with data mining models and materialized views. The models obtained from the automatic knowledge extraction process is a set of association rules discovered from the database contents. Materialized views are pre-computed and normally aggregated data from base tables to speed up the processing of frequently asked queries. This paper presents the knowledge acquisition method focusing on association pattern mining, its implementation, and a systematic method of rewriting query with association patterns and materialized views. We perform preliminary efficiency tests of the proposed system. The experimental results demonstrate the effectiveness of our system in answering queries sharing the same pattern as the available knowledge and the pre-computed views.

Keywords: *Query answering, Query optimization, Association mining, Materialized views, Erlang programming, Deductive database*

1. Introduction

Since the emergence of data mining as a new research area two decades ago, it has been argued among database researchers that the database management system (DBMS) should support both data processing and data mining tasks [3, 4, 5, 9]. With the data mining functionalities, a database can contain not only data contents and schemas, but also data models which are generalized information induced from the data contents. By providing such an extension framework of the DBMS, users can manipulate and access data models in the same manner as querying and processing the data contents. To achieve this aim, a number of database system extensions, such as the IBM intelligent miner [17] and Microsoft OLE DB for data mining [20], have been implemented.

Besides the front-end support for mining database contents, we propose that the next generation DBMS should also utilize the data mining models at the back-end part to support query answering and optimization. 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. Semantic knowledge such as integrity constraints and data mining models can be used to transform a query into an optimized one. Recent advance on semantic knowledge utilization has moved toward the setting of cooperative or intelligent query answering [11, 16].

The process of intelligent query answering consists of analyzing the intent of a query, rewriting the query based on the intention and other kinds of knowledge, and providing answers in an intelligent way [19]. 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 contents and schemas. Therefore, it is difficult to pose queries correctly to get some useful answers.

Knowledge, either intentionally or extensionally stated, is the key ingredient of intelligence. Many researchers [2, 7, 16, 19] 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 data mining models and materialized views in the query answering system.

Data mining models are generalized rules discovered from databases and stored as tables, whereas materialized views are view relations computed and stored in the database as well. We consider data mining models and materialized views as semantic constraints capable of transforming queries to be processed intelligently.

We design a query optimizing and answering system using a platform of deductive database that can be easily integrated with induced knowledge obtained from the constrained association mining program implemented with the Erlang language. The remainder of the paper is organized as follows. Section 2 presents related work. Section 3 explains architecture and implementation of the proposed system. Section 4 discusses the experimentation and its results. Section 5 concludes the paper and indicates our future work.

2. Related Work

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 [7] 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 [7, 11] has focused on rewriting the query using generalization concept, neighborhood, and type abstraction hierarchy. The work of Han et al [16] 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 [19] proposed to integrate neighborhood information and data mining rules discovered from the databases to rewrite the queries. Muslea [21] 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 [3] 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 [1, 6, 8, 10, 14, 15, 22]. 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 and Implementation of Semantic-based Query Optimization

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 and its algorithm are provided in Figures 1 and 2, respectively.

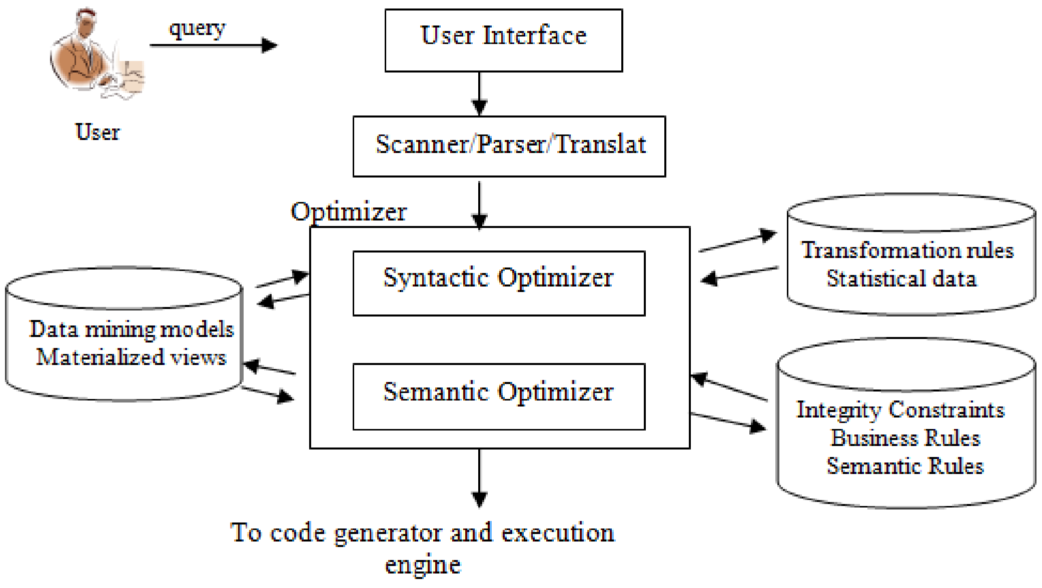


Figure 1. A Framework of Semantic-based Query Optimizer

```
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 ∈ 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′
```

Figure 2. Semantic Query Optimization Algorithm

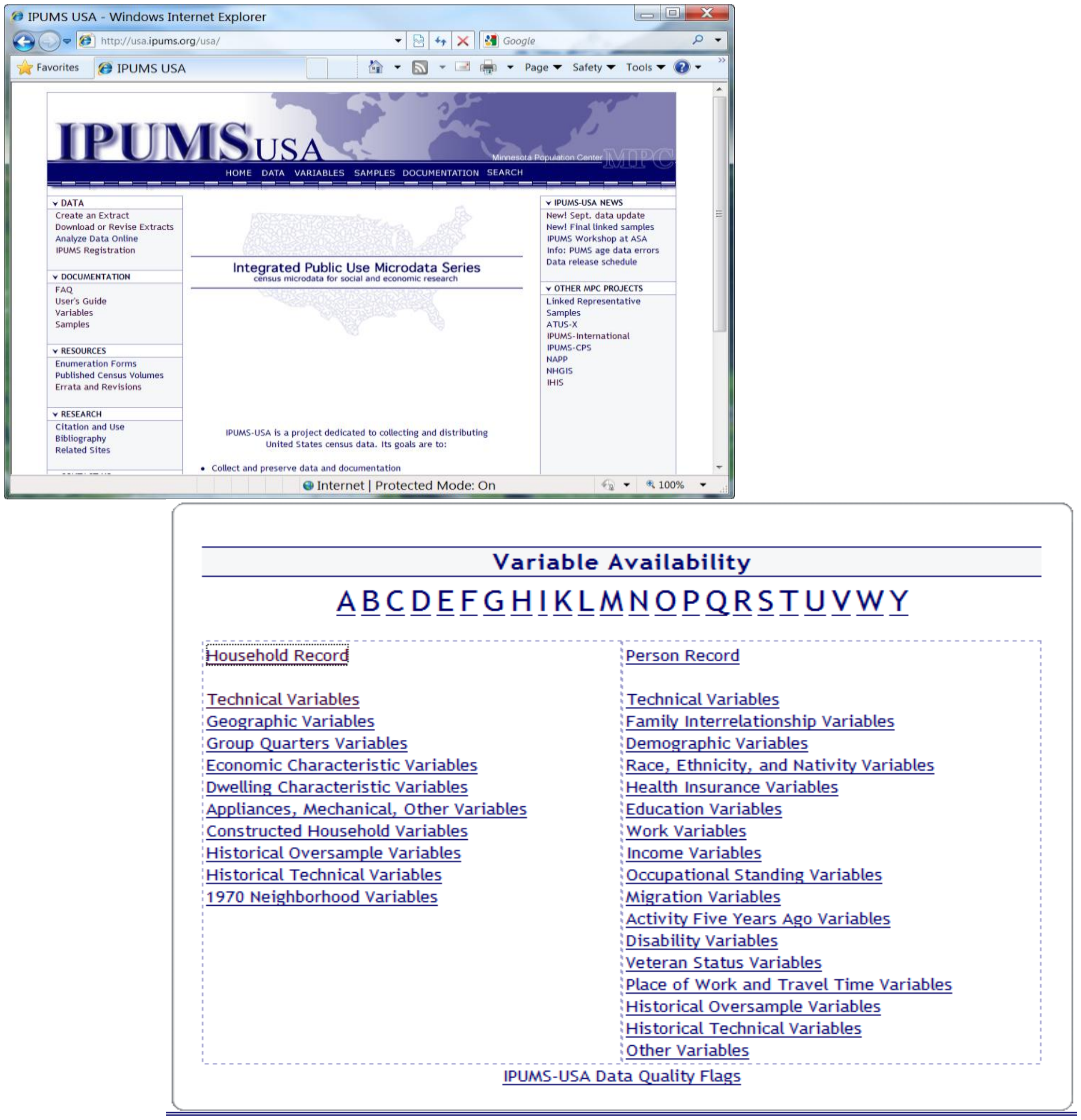
Semantic rules as mentioned in the framework and in the algorithm are association rules [2] induced from the database contents and constrained to induce only knowledge relevant to users' queries. The association mining component is implemented with the Erlang programming language [13]. The program is written to search for association rules with confidence 1.0, that is, 100% correct. Main part of the source code can be displayed as follows:

```
main() ->
    NameList=mylib:read_file( "ipum.NAMES"," ,.\t:|" ),
    mylib:text_file(write,NameList,filetemp),
    [H|Tail]=NameList, [F,C|T]=lists:reverse(H),NewClass=[F|T], % make_nornal_class
    LL=[NewClass|Tail],
    [A1|A2]=map(fun([H|_])->H end, LL),AttrName=A2++[A1],
    [P1|P2]=lists:map(fun(L)->allPossibleAttr(L) end,LL),%swap class to the last
    PossibleValue=P2++[P1],
    mylib:text_file(write,PossibleValue,filetemp1),
    mylib:text_file(write,AttrName,filetemp2),AllInput=input(AttrName),
         %% write fact to file
    {_,IO}=file:open("factc.pl",[write]),
    lists:map(fun(EachR)->to_prolog(EachR,IO) end,AllInput),
    _=file:close(IO),
    DB=myToSet(AllInput),  MinSup=inputSup(AllInput), mylib:c(20,MinSup),
    mylib:text_file(write,AllInput,allinput),
    Items=my_flat(PossibleValue),
    AllL=apriori1(DB, Items,MinSup),
    format("~n   END ~n").

my_flat([H|T]) -> H++my_flat(T);
my_flat([]) -> [].
```

```
to_prolog(Fact,IO) ->
      io:format(IO,"~n~p(",[rec]),
      print_fact(IO,Fact),io:format(IO,").% ",[]).

print_fact(IO,[H]) -> io:format(IO,"~p",[list_to_atom(H)]) ;
print_fact(IO,[H|T]) -> io:format(IO,"~p,",[list_to_atom(H)]),print_fact(IO,T).

apriori1(DB,Items,Min) -> %% findSup(Set,ListOfSet)
    mylib:text_file(append,myToList(DB),apriori1),mylib:text_file(append,Items,apriori1),
    C1=[{from_list([X]),findSup(from_list([X]),DB)}|| X<-Items ],
    CkPrint=[ {to_list(FS),Sup} || {FS,Sup}<-C1],
    L1=[{FS,Sup} || {FS,Sup}<-C1,Sup>=Min],
    LkPrint=[ {to_list(FS),Sup,Sup/length(DB)*100} || {FS,Sup}<-L1],
    mylib:text_file(write,CkPrint,lkfile),
    K=2, LS=[FS||{FS,_}<-L1],
    AllSet=aprioriAll(L1,DB,LS,K,Min),
    mylib:term_file(write,AllSet,"set.raw").

inputSup(AllInput)->Total=length(AllInput),
     {_,Per}=io:read(" input percent> "),
     MinSup=Total*Per/100 .

aprioriAll(AllL,_,[],_,_) -> format("~nfinal set=~p~n",[AllL]),AllL;  % return final Set
aprioriAll(AllL,_,[_],_,_) -> format("~nfinal set=~p~n",[AllL]),AllL;
aprioriAll(AllL,DB,LS,K,Min) -> Com=combi(LS),
      C_=myDistinct(usedCombi(Com,K)),
      Ck=[{X,findSup(X,DB)}|| X<-C_],mylib:c(35,Min),
      Lk=[ {FS,Sup} || {FS,Sup}<-Ck,Sup>=Min],
      LkS=[FS||{FS,_}<-Lk],
      LkPrint=[ {to_list(FS),Sup,Sup/length(DB)*100} || {FS,Sup}<-Lk],
      format("~nK=~w-~p,  has ~w set ~n  ",[K,LkPrint,length(LkPrint)]),
      aprioriAll(AllL++Lk,DB,LkS,K+1,Min) .

allPossibleAttr([H|T]) -> [H++ET||ET<-T].

input(AttrName) ->  LinesList=mylib:read_file( "ipums-1999-1Krecords.bak"," ," ),
      Zip=map( fun(EachL)->lists:zip(AttrName,EachL) end,LinesList ),
      UsedData=map(fun(LineOfTuple)->concat_line_tuple(LineOfTuple) end,Zip).

% shift([a,b,c]) --> [b,c,a]
shift([H|T]) -> T++[H].

genR(_,Max,Max) -> [];
genR(L,N,Max) -> {H,T}=lists:split(N,L), [{H,T}]++genR(L,N+1,Max).

% genRule([2,3,5],3,3).
genRule(_,0,_)->[];
genRule(L,Count,Len)-> genR(L,1,Len)++genRule(shift(L),Count-1,Len).
findConf({H,B},AllL)  -> {H,B,searchL(set(H++B),AllL)/searchL(set(H),AllL) }.

% main1() is for creating rules.
main1() ->
      format("~n------------START-create rules--------------"),
      {_,[AllL]}=file:consult("set.raw"),
      AllAsso2=[list(X)|| {X,_} <-AllL,length(list(X))>1 ],
        %gen Rules
      AllRuleGen=lists:flatten([genRule(L,length(L),length(L))||L<-AllAsso2]),
      AllRuleConf=[findConf(X,AllL)||X<-AllRuleGen],
      format("~nAllRule=~p ,~nThere are ~p rules ",[AllRuleConf,length(AllRuleConf)]),
        mylib:text_file(write,AllRuleConf,allrule),
        Sorted= lists:sort(fun({_,_,C1},{_,_,C2})->C1>=C2 end,AllRuleConf),
        mylib:text_file(write,Sorted,"allsortedrule.txt"),
        Conf1=lists:filter(fun({A,B,C})->C==1.0 end,Sorted),
          %% write to file
          %%% create prolog file Rules+Facts
        {_,IO}=file:open("rules.pl",[write]),
        lists:map(fun(EachR)->transform_to_prolog(EachR,IO) end,Conf1),
        _=file:close(IO),
        mylib:text_file(append,length(Conf1),"allsortedrule.txt"),
        mylib:text_file(append,length(Sorted),"allsortedrule.txt"),
        format("~n-----------end main1() process ----------------") .
          %%% create prolog file Rules+Facts

    transform_to_prolog({Body,Head,Conf},IO) ->
         if (length(Head)==1) -> [Head1]=Head,
                                 io:format(IO,"~np(~p):-",[list_to_atom(Head1)]),
                                 print_body(IO,Body), io:format(IO,"% Conf=~p",[Conf]);
            true -> io:format("")
         end.

    print_body(IO,[H]) -> io:format(IO,"p(~p).",[list_to_atom(H)]) ;
    print_body(IO,[H|T]) -> io:format(IO,"p(~p),",[list_to_atom(H)]),print_body(IO,T).

    set(X) -> from_list(X).
    list(X) -> to_list(X).
    searchL(Set,[{Set,Val}|_])   ->  Val;
    searchL(Set,[{_Another,_}|T])  -> searchL(Set,T);
    searchL(_Set,[]) ->  1  .% Cannot find Set

    concat_line_tuple(LineOfTuple) -> map(fun({A,B})->A++B end, LineOfTuple).

    myToSet(L) -> [from_list(X)||X<-L].
    myToList(SL) -> [to_list(S)||S<-SL].

    findSup(_,[]) -> 0;
    findSup(Set,DB) -> [H|T]=DB,
        Cond = is_subset(Set,H),
        if Cond -> 1+findSup(Set,T);
          true -> findSup(Set,T)
        end.

    myDistinct(List) -> to_list(from_list(List)).

    combi([H|T]) -> [[H,Te] || Te<-T]++ combi(T);
    combi([]) -> [].

    usedCombi([H|T],K) ->  Union=union(H),
        Len=ordsets:size(Union),
        if Len==K -> [Union|usedCombi(T,K)];
           true -> usedCombi(T,K)
        end ;
    usedCombi([],_) -> [].


 %-----------  end of association rule mining program ------------------
```

4. Experimentation and Query Answering 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 [18]. This database contains the United States socio-economic data of the year 1999 with household and person information as shown in Figure 3. The household records contain information such as value of household unit, monthly rental payment, family total income, and other related information. Detail information of the household records is given in Table 1, and its example data records are shown in Figure 4. For the person records, the information is about education, employment status, occupation, income and other personal information. The details are in Table 2 and its record examples are in Figure 5.



IPUMS-USA is a project dedicated to collecting and distributing United States census data. Its goals are to:

- Collect and preserve data and documentation

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

Figure 3. The IPUMS-USA Database and its Variables (or attributes) as Appeared in the Household and Person Records

Table 1. Attribute Details of the Household Records

Attribute	Name	Description
1	ID	Record-ID
2	gq	Group quarters status
3	gqtypeg	Group quarters type -- general
4	farm	Farm status
5	ownershg	Ownership of dwelling -- general
6	value	Value of housing unit
7	rent	Monthly rental payment
8	ftotinc	Family total income
9	nfams	Number of families within each household unit
10	ncouples	Number of married couples in a household
11	nmothers	Number of mothers within each household unit
12	nfathers	Number of fathers within each household unit
13	momloc	Mother's location in the household (a variable that indicates whether the person's mother lived in the same household and, if so, gives the number of mother)
14	stepmom	Probable step/adopted mother (whether a person's mother was likely to have been the person's stepmother or adoptive mother)
15	momrule	Rule for linking mother
16	poploc	Father's location in the household (a constructed variable to identify social relationship such as stepfather, adoptive father, as well as biological father)
17	steppop	Probable step/adopted father
18	poprule	Rule for linking father
19	sploc	Spouse's location in household (a constructed variable that indicates whether the person's spouse lived in the same household and, if so, gives the person number of the spouse)
20	sprule	Rule for linking spouse
21	famsize	Number of own family members in household
22	nchild	Number of own children in the household
23	nchlt5	Number of own children under age 5 in household
24	famunit	Groups of related individuals in the household
25	eldch	Age of eldest own child in household
26	yngch	Age of youngest own child in household
27	nsibs	Number of own siblings in the household
28	relateg	Relationship to household head -- general
29	age	Age of a person
30	sex	Sex of a person
31	raceg	Race -- general
32	marst	Marital status
33	chborn	Number of children ever born to each woman
34	bplg	Birthplace -- general

```
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,p
          oprule_0,sploc_2,sprule_1,famsize_2,nchild_0,nchlt5_0,famunit_1,eldch_10,yngc
          h_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,p
          oprule_0,sploc_1,sprule_1,famsize_2,nchild_0,nchlt5_0,famunit_1,eldch_10,yngc
          h_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,p
          oprule_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,p
          oprule_0,sploc_0,sprule_0,famsize_2,nchild_0,nchlt5_0,famunit_1,eldch_10,yngc
          h_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,p
          oprule_0,sploc_0,sprule_0,famsize_1,nchild_0,nchlt5_0,famunit_2,eldch_10,yngc
          h_10,nsibs_0,relateg_11,age_3,sex_1,raceg_1,marst_6,chborn_0,bplg_1).%
```

Figure 4. Example of Household Records, Each Record has 34 Attributes

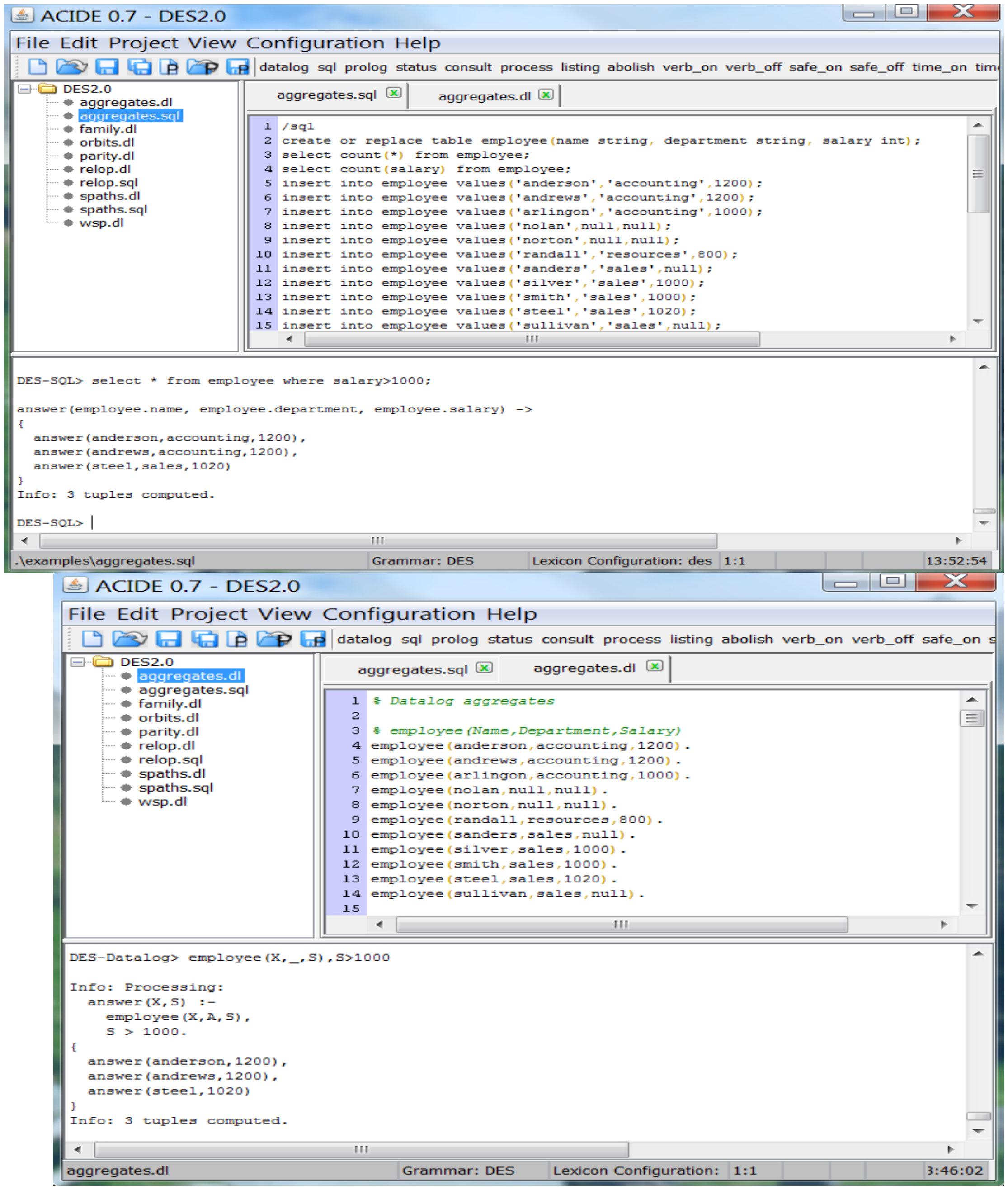
```
table2(1,school_1,educrec_7,schltype_1,empstatg_1,labforce_2,occscore_3,sei_2,
          classwkg_2,wkswork2_4,hrswork2_6,yrlastwk_0,workedyr_2,inctot_3,
          incwage_3,incbus_1,incfarm_1,incss_1,incwelfr_1,incother_1,poverty_6,
          migrat5g_1,migplac5_0,movedin_7,vetstat_1,tranwork_10,occupation_2).%
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).%
table2(4,school_2,educrec_2,schltype_2,empstatg_0,labforce_0,occscore_1,sei_1,class
          wkg_0,wkswork2_0,hrswork2_0,yrlastwk_0,workedyr_0,inctot_1,incwage_4,
          incbus_4,incfarm_4,incss_3,incwelfr_3,incother_3,poverty_2,migrat5g_1,mi
          gplac5_1,movedin_0,vetstat_0,tranwork_0,occupation_5).%
table2(5,school_1,educrec_7,schltype_1,empstatg_2,labforce_2,occscore_5,sei_8,class
          wkg_2,wkswork2_6,hrswork2_0,yrlastwk_10,workedyr_2,inctot_3,incwage_
          3,incbus_1,incfarm_1,incss_1,incwelfr_1,incother_1,poverty_4,migrat5g_2,
          migplac5_1,movedin_0,vetstat_2,tranwork_0,occupation_2).%
```

Figure 5. Example of Person Records, Each Record has 27 Attributes

Table 2. Attribute Details of the Person Records

Attribute	Name	Description
1	ID	Record-ID
2	school	School attendance
3	educrec	Highest year of school or degree completed
4	schltype	School type
5	empstatg	Employment status -- general
6	labforce	Labor force status
7	occscore	Occupational income score (the median total income in hundreds of dollars)
8	sei	Duncan Socioeconomic Index score to each occupation (SEI is the measure of occupational status based upon the income level and educational attainment associated with each occupation)
9	classwkg	Class of worker -- general
10	wkswork2	Weeks worked last year, intervalled
11	hrswork2	Hours work last week, intervalled
12	yrlastwk	Year last worked
13	workedyr	Worked last year
14	inctot	Total personal income
15	incwage	Wage and salary income
16	incbus	Non-farm business income
17	incfarm	Farm income
18	incss	Social security income (social security pensions, survivors benefits, permanent disability insurance)
19	incwelfr	Welfare income (federal/state supplemental security income payments to elderly, blind, or disabled persons with low incomes; families with dependent children)
20	incother	Other income
21	poverty	Poverty status (It expresses each family's total income for the previous year as a percentage of the poverty threshold established by the Social Security Administration.)
22	migrat5g	Migration status, 5 years – general
23	migplac5	State or country of residence 5 years ago
24	movedin	Number of years ago that the householder moved into the dwelling unit
25	vetstat	Veteran status
26	tranwork	Means of transportation to work
27	occupation	Occupational classification

To test the efficiency of query optimization with materialized views and data mining models, we create a database using the DES system [12]. The DES system provides facility to support querying in both the SQL and Datalog languages as shown in Figure 6. The database stores table1 and Table 2 (Figures 4 and 5) to contain household and person information, respectively. Both tables are in the format of Datalog clauses. Data mining models used in the experimentation are association rules that are also transformed to be Datalog clauses as shown some part in Figure 7. Examples of materialized views used in the experimentation are displayed in Figure 8.



```
/sql
create or replace table employee(name string, department string, salary int);
select count(*) from employee;
select count(salary) from employee;
insert into employee values('anderson','accounting',1200);
insert into employee values('andrews','accounting',1200);
insert into employee values('arlingon','accounting',1000);
insert into employee values('nolan',null,null);
insert into employee values('norton',null,null);
insert into employee values('randall','resources',800);
insert into employee values('sanders','sales',null);
insert into employee values('silver','sales',1000);
insert into employee values('smith','sales',1000);
insert into employee values('steel','sales',1020);
insert into employee values('sullivan','sales',null);
```

```
DES-SQL> select * from employee where salary>1000;

answer(employee.name, employee.department, employee.salary) ->
{
  answer(anderson,accounting,1200),
  answer(andrews,accounting,1200),
  answer(steel,sales,1020)
}
Info: 3 tuples computed.

DES-SQL> |
```

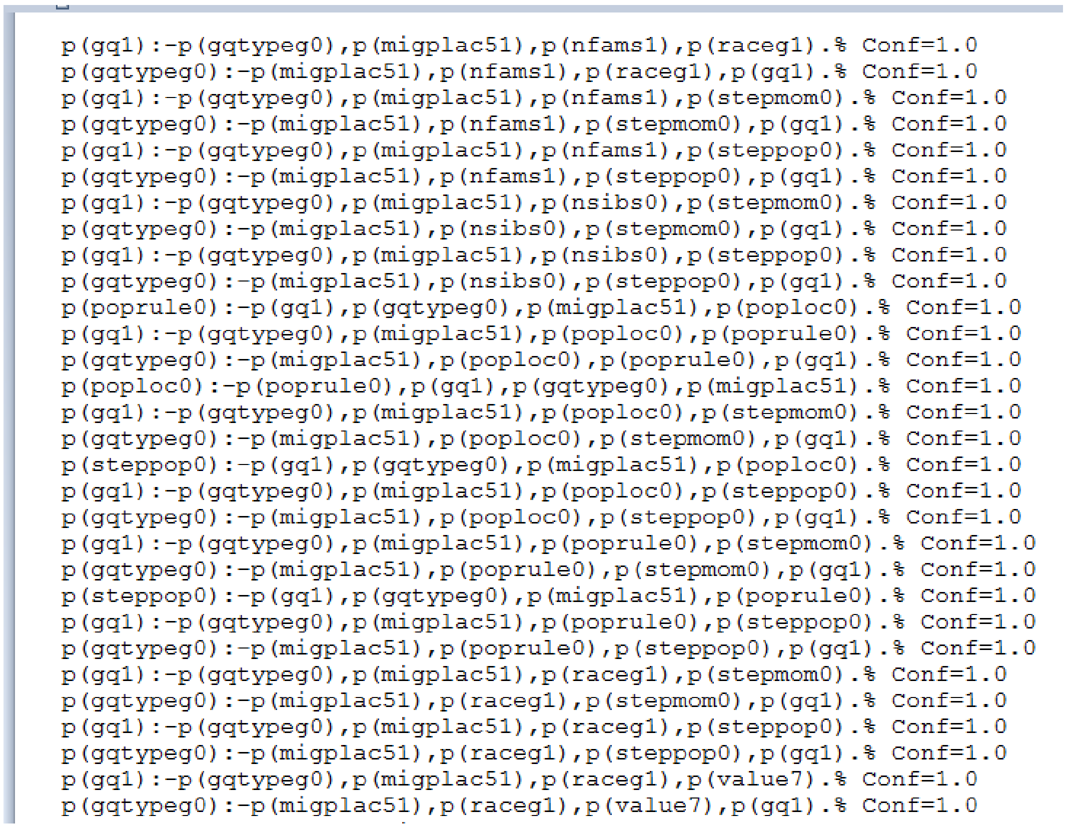
.\examples\aggregates.sql | Grammar: DES | Lexicon Configuration: des | 1:1 | 13:52:54

```
% Datalog aggregates

% employee(Name,Department,Salary)
employee(anderson,accounting,1200).
employee(andrews,accounting,1200).
employee(arlingon,accounting,1000).
employee(nolan,null,null).
employee(norton,null,null).
employee(randall,resources,800).
employee(sanders,sales,null).
employee(silver,sales,1000).
employee(smith,sales,1000).
employee(steel,sales,1020).
employee(sullivan,sales,null).
```

```
DES-Datalog> employee(X,_,S),S>1000

Info: Processing:
  answer(X,S) :-
    employee(X,A,S),
    S > 1000.
{
  answer(anderson,1200),
  answer(andrews,1200),
  answer(steel,1020)
}
Info: 3 tuples computed.
```

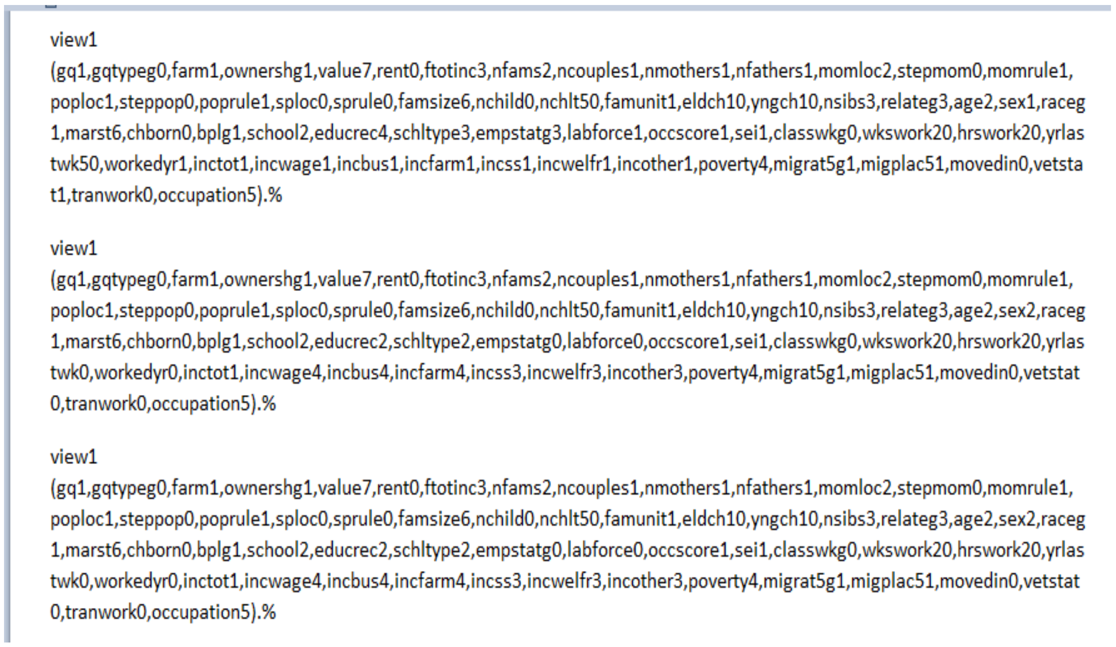
aggregates.dl | Grammar: DES | Lexicon Configuration: | 1:1 | 3:46:02

Figure 6. The DES System with SQL Support (above) and Datalog Querying (below)



```
p(gq1):-p(gqtypeg0),p(migplac51),p(nfams1),p(raceg1).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(nfams1),p(raceg1),p(gq1).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(nfams1),p(stepmom0).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(nfams1),p(stepmom0),p(gq1).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(nfams1),p(steppop0).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(nfams1),p(steppop0),p(gq1).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(nsibs0),p(stepmom0).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(nsibs0),p(stepmom0),p(gq1).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(nsibs0),p(steppop0).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(nsibs0),p(steppop0),p(gq1).% Conf=1.0
p(poprule0):-p(gq1),p(gqtypeg0),p(migplac51),p(poploc0).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(poploc0),p(poprule0).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(poploc0),p(poprule0),p(gq1).% Conf=1.0
p(poploc0):-p(poprule0),p(gq1),p(gqtypeg0),p(migplac51).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(poploc0),p(stepmom0).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(poploc0),p(stepmom0),p(gq1).% Conf=1.0
p(steppop0):-p(gq1),p(gqtypeg0),p(migplac51),p(poploc0).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(poploc0),p(steppop0).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(poploc0),p(steppop0),p(gq1).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(poprule0),p(stepmom0).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(poprule0),p(stepmom0),p(gq1).% Conf=1.0
p(steppop0):-p(gq1),p(gqtypeg0),p(migplac51),p(poprule0).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(poprule0),p(steppop0).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(poprule0),p(steppop0),p(gq1).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(raceg1),p(stepmom0).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(raceg1),p(stepmom0),p(gq1).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(raceg1),p(steppop0).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(raceg1),p(steppop0),p(gq1).% Conf=1.0
p(gq1):-p(gqtypeg0),p(migplac51),p(raceg1),p(value7).% Conf=1.0
p(gqtypeg0):-p(migplac51),p(raceg1),p(value7),p(gq1).% Conf=1.0
```

Figure 7. Part of Semantic Rules Transformed from the Association Mining Models



```
view1
(gq1,gqtypeg0,farm1,ownershg1,value7,rent0,ftotinc3,nfams2,ncouples1,nmothers1,nfathers1,momloc2,stepmom0,momrule1,
poploc1,steppop0,poprule1,sploc0,sprule0,famsize6,nchild0,nchlt50,famunit1,eldch10,yngch10,nsibs3,relateg3,age2,sex1,raceg
1,marst6,chborn0,bplg1,school2,educrec4,schltype3,empstatg3,labforce1,occscore1,sei1,classwkg0,wkswork20,hrswork20,yrlas
twk50,workedyr1,inctot1,incwage1,incbus1,incfarm1,incss1,incwelfr1,incother1,poverty4,migrat5g1,migplac51,movedin0,vetsta
t1,tranwork0,occupation5).%

view1
(gq1,gqtypeg0,farm1,ownershg1,value7,rent0,ftotinc3,nfams2,ncouples1,nmothers1,nfathers1,momloc2,stepmom0,momrule1,
poploc1,steppop0,poprule1,sploc0,sprule0,famsize6,nchild0,nchlt50,famunit1,eldch10,yngch10,nsibs3,relateg3,age2,sex2,raceg
1,marst6,chborn0,bplg1,school2,educrec2,schltype2,empstatg0,labforce0,occscore1,sei1,classwkg0,wkswork20,hrswork20,yrlas
twk0,workedyr0,inctot1,incwage4,incbus4,incfarm4,incss3,incwelfr3,incother3,poverty4,migrat5g1,migplac51,movedin0,vetstat
0,tranwork0,occupation5).%

view1
(gq1,gqtypeg0,farm1,ownershg1,value7,rent0,ftotinc3,nfams2,ncouples1,nmothers1,nfathers1,momloc2,stepmom0,momrule1,
poploc1,steppop0,poprule1,sploc0,sprule0,famsize6,nchild0,nchlt50,famunit1,eldch10,yngch10,nsibs3,relateg3,age2,sex2,raceg
1,marst6,chborn0,bplg1,school2,educrec2,schltype2,empstatg0,labforce0,occscore1,sei1,classwkg0,wkswork20,hrswork20,yrlas
twk0,workedyr0,inctot1,incwage4,incbus4,incfarm4,incss3,incwelfr3,incother3,poverty4,migrat5g1,migplac51,movedin0,vetstat
0,tranwork0,occupation5).%
```

Figure 8. Example of Materialized Views Used in the Querying Experimentation

We test the query processing performance with six kinds of queries (experimental results are graphically shown in Figure 9 and summarized in Table 3). 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. The details of each query as well as its running time report are provided as follows:

Query 1: Ask for the value of farm housholds. (Null answer; all households are non-farm)

```
DES-Datalog> /assert query1(X) :-
                    table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,
                    A13,A14,A15,A16,A17,A18,A19,A20,A21,A22,A23,A24,A25,
                    A26,A27,A28,A29,A30,A31,A32,A33,A34),A4=farm_2,X=A6.

DES-Datalog> query1(X).
          {}
          Info: 0 tuples computed.
          Total elapsed time: 110 ms.
```

Search for semantic rules to transform query: (the rule's predicate is 'farm_2')

```
DES-Datalog> /assert p(farm_2).
DES-Datalog> p(C).
       {  p(farm_2) }
       Info: 1 tuple computed.
       Total elapsed time: 295 ms.
```

The query returns the same predicate 'farm_2'; that means no rules can be applied. Thus, the semantic-based query transformation wastes time = 295 ms.

Query 2: Ask for the value of non-farm households.

```
DES-Datalog> /assert query2(X) :-
                    table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,
                    A13,A14,A15,A16,A17,A18,A19,A20,A21,A22,A23,A24,A25,
                    A26,A27,A28,A29,A30,A31,A32,A33,A34), A4=farm_1, X=A6.
DES-Datalog> query2(X).
          {  query2(value_1),
             query2(value_2),
             query2(value_3),
             query2(value_4),
             query2(value_5),
             query2(value_6),
             query2(value_7) }
          Info: 7 tuples computed.
          Total elapsed time: 1029 ms.
```

Search for semantic rules to transform query: (the rule's predicate is 'farm_1')

```
DES-Datalog> /assert p(farm_1).
DES-Datalog> p(C).
       {  p(farm_1) }
       Info: 1 tuple computed.
```

```
                Total elapsed time: 296 ms.
```

The query returns the same predicate 'farm_1'; that means no rules can be applied. Thus, the semantic-based query transformation wastes time = 296 ms.

Query 3: *Ask for total family income for a household unit with two families.*

```
  DES-Datalog> /assert query3(X):-
                      table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,
                      A13,A14,A15,A16,A17,A18,A19,A20,A21,A22,A23,A24,A25,
                      A26,A27,A28,A29,A30,A31,A32,A33,A34),A24=famunit_2,X=A8.

  DES-Datalog> query3(X).
                {  query3(ftotinc_1),
                   query3(ftotinc_2),
                   query3(ftotinc_3) }
                Info: 3 tuples computed.
                Total elapsed time: 906 ms.
```

Search for semantic rules to transform query: (the rule's predicate is 'famunit_2')

```
      DES-Datalog> /assert p(famunit_2).
      DES-Datalog> p(C).
                {  p(famunit_2) }
                Info: 1 tuple computed.
                Total elapsed time: 282 ms.
```

The query returns the same predicate 'famunit_2'; that means no rules can be applied. Thus, the semantic-based query transformation wastes time = 282 ms.

Query 4: *Ask for family size of non-farm households with family income at level 3 (10,000-99,999 US$).*

```
  DES-Datalog> /assert query4(X):-
          table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,
          A13,A14,A15,A16,A17,A18,A19,A20,A21,A22,A23,A24,A25,
          A26,A27,A28,A29,A30,A31,A32,A33,A34),A4=farm_1,A8=ftotinc_3,X=A21.

  DES-Datalog> query4(X).
        {  query4(famsize_1),
           query4(famsize_2),
           query4(famsize_3),
           query4(famsize_4),
           query4(famsize_5),
           query4(famsize_6),
           query4(famsize_7) }
        Info: 7 tuples computed.
        Total elapsed time: 1028 ms.
```

Search for semantic rules to transform query:
(the rule's predicates are 'farm_1' and 'ftotinc_3')

```
          DES-Datalog> /assert p(farm_1).
          DES-Datalog> /assert p(ftotinc_3).
          DES-Datalog> p(C).
                {  p(farm_1),
                   p(ftotinc_3),
        p(gq_1),
        p(gqtypeg_0)  }
      Info: 6 tuples computed.
      Total elapsed time: 594 ms.
```

Semantically transformed query:

```
DES-Datalog> /assert query4B(X):-table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,
             A11,A12,A13,A14,A15,A16,A17,A18,A19,A20,A21,A22,A23,A24,
             A25,A26,A27,A28,A29,A30,A31,A32,A33,A34), A4=farm_1,
             A8=ftotinc_3, A2=gq_1, A3=gqtypeg_0, X=A21.

DES-Datalog> query4B(X).
      {  query4B(famsize_1),
         query4B(famsize_2),
         query4B(famsize_3),
         query4B(famsize_4),
         query4B(famsize_5),
         query4B(famsize_6),
         query4B(famsize_7)  }
      Info: 7 tuples computed.
      Total elapsed time: 94 ms.
```

Query processing time saving = 1028-(594+94)= 340 ms.

<u>Query 5</u>: *Ask for educational record and total income of female person. (Joining of table1 and table2)*

```
DES-Datalog> /assert query5(X,Y):-table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,
             A12,A13,A14,A15,A16,A17,A18,A19,A20,A21,A22,A23,A24,A25,
             A26,A27,A28,A29,A30,A31,A32,A33,A34),table2(B1,B2,B3,B4,B5,B
             6,B7,B8,B9,B10,B11,B12,B13,B14,B15,B16,B17,B18,B19,B20,B2
             1,B22,B23,B24,B25,B26,B27),A1=B1,A30=sex_2,X=B3,Y=B15.

DES-Datalog> query5(X,Y).
             { query5(educrec_0,incwage_4),
               query5(educrec_1,incwage_1),
               query5(educrec_1,incwage_2),
               query5(educrec_1,incwage_3),
               query5(educrec_1,incwage_4),
               query5(educrec_2,incwage_1),
               query5(educrec_2,incwage_3),
               query5(educrec_2,incwage_4),
               query5(educrec_3,incwage_1),
               query5(educrec_3,incwage_2),
               query5(educrec_3,incwage_3),
               query5(educrec_3,incwage_4),
               query5(educrec_4,incwage_1),
               query5(educrec_4,incwage_2),
               query5(educrec_4,incwage_3),
               query5(educrec_4,incwage_4),
               query5(educrec_5,incwage_1),
               query5(educrec_5,incwage_2),
               query5(educrec_5,incwage_3),
               query5(educrec_5,incwage_4),
               query5(educrec_6,incwage_1),
               query5(educrec_6,incwage_2),
               query5(educrec_6,incwage_3),
               query5(educrec_7,incwage_1),
               query5(educrec_7,incwage_2),
               query5(educrec_7,incwage_3),
               query5(educrec_8,incwage_1),
               query5(educrec_8,incwage_2),
               query5(educrec_8,incwage_3),
               query5(educrec_9,incwage_1),
               query5(educrec_9,incwage_2),
               query5(educrec_9,incwage_3),
               query5(educrec_9,incwage_4)  }
       Info: 756 tuples computed.
       Total elapsed time: 10423 ms.
```

Transformed query with materialized view:

```
DES-Datalog> /assert query5B(X,Y):-
                 view1(V1,V2,V3,V4,V5,V6,V7,V8,V9,V10,V11,
                 V12,V13,V14,V15,V16,V17,V18,V19,V20,V21,V22,V23,V24,V25,
                 V26,V27,V28,V29,V30,V31,V32,V33,V34,V35,V36,V37,V38,V39,
                 V40,V41,V42,V43,V44,V45,V46,V47,V48,V49,V50,V51,V52,V53,
                 V54,V55,V56,V57,V58,V59),V29=sex_2,X=V35,Y=V47.
DES-Datalog> query5B(X,Y).
             { query5B(educrec_0,incwage_4),
               query5B(educrec_1,incwage_1),
               query5B(educrec_1,incwage_2),
               query5B(educrec_1,incwage_3),
               query5B(educrec_1,incwage_4),
               query5B(educrec_2,incwage_1),
               query5B(educrec_2,incwage_3),
               query5B(educrec_2,incwage_4),
               query5B(educrec_3,incwage_1),
               query5B(educrec_3,incwage_2),
               query5B(educrec_3,incwage_3),
               query5B(educrec_3,incwage_4),
               query5B(educrec_4,incwage_1),
               query5B(educrec_4,incwage_2),
               query5B(educrec_4,incwage_3),
               query5B(educrec_4,incwage_4),
               query5B(educrec_5,incwage_1),
               query5B(educrec_5,incwage_2),
               query5B(educrec_5,incwage_3),
               query5B(educrec_5,incwage_4),
               query5B(educrec_6,incwage_1),
               query5B(educrec_6,incwage_2),
               query5B(educrec_6,incwage_3),
               query5B(educrec_7,incwage_1),
               query5B(educrec_7,incwage_2),
               query5B(educrec_7,incwage_3),
               query5B(educrec_8,incwage_1),
               query5B(educrec_8,incwage_2),
               query5B(educrec_8,incwage_3),
               query5B(educrec_9,incwage_1),
               query5B(educrec_9,incwage_2),
               query5B(educrec_9,incwage_3),
               query5B(educrec_9,incwage_4)  }
       Info: 33 tuples computed.
       Total elapsed time: 1060 ms.
```

Query processing with the transformed query with materialized view:

Time saving = 10423-1060 = 9363 ms.

<u>Query 6</u>: *Ask for class of work, occupation, and income of white immigrants moving from Europe and being a single family unit. (Joining table1 and table2)*

```
DES-Datalog> /assert query6(X,Y,Z):-
                table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,
                A12,A13,A14,A15,A16,A17,A18,A19,A20,A21,A22,A23,A24,A25,
                A26,A27,A28,A29,A30,A31,A32,A33,A34),table2(B1,B2,B3,B4,B5,B
                6,B7,B8,B9,B10,B11,B12,B13,B14,B15,B16,B17,B18,B19,B20,B2
                1,B22,B23,B24,B25,B26,B27), A1=B1, A9=nfams_1, A31=raceg_1,
                A34=bplg_4, X=B9, Y=B27, Z=B14.
DES-Datalog> query6(X,Y,Z).
              { query6(classwkg_0,occupation_5,inctot_1),
                query6(classwkg_0,occupation_5,inctot_2),
                query6(classwkg_0,occupation_5,inctot_3),
                query6(classwkg_1,occupation_2,inctot_3),
                query6(classwkg_2,occupation_1,inctot_1),
                query6(classwkg_2,occupation_1,inctot_2),
                query6(classwkg_2,occupation_1,inctot_3),
                query6(classwkg_2,occupation_1,inctot_4),
                query6(classwkg_2,occupation_2,inctot_1),
                query6(classwkg_2,occupation_2,inctot_2),
                query6(classwkg_2,occupation_2,inctot_3)  }
        Info: 121 tuples computed.
        Total elapsed time: 9852  ms.
```

Transformed query with semantic rules and views:

```
DES-Datalog> /assert p(nfams_1)
DES-Datalog> p(C)
            {  p(famunit_1),
               p(farm_1),
               p(nfams_1)  }
            Info: 3 tuples computed.
            Total elapsed time: 452 ms.
```

```
 DES-Datalog> /assert query6B(X,Y,Z):-view1(V1,V2,V3,V4,V5,V6,V7,V8,V9,V10,
                V11,V12,V13,V14,V15,V16,V17,V18,V19,V20,V21,V22,V23,V24,V25
                ,V26,V27,V28,V29,V30,V31,V32,V33,V34,V35,V36,V37,V38,V39,V4
                0,V41,V42,V43,V44,V45,V46,V47,V48,V49,V50,V51,V52,V53,
                V54,V55,V56,V57,V58,V59), V3=farm_1, V8=nfams_1,
                V23=famunit_1, V30=raceg_1, V33=bplg_4, X=V41, Y=V59, Z=V46.
```

```
DES-Datalog> query6B(X,Y,Z).
                 {query6B(classwkg_0,occupation_5,inctot_1),
                  query6B(classwkg_0,occupation_5,inctot_2),
                  query6B(classwkg_0,occupation_5,inctot_3),
                  query6B(classwkg_1,occupation_2,inctot_3),
                  query6B(classwkg_2,occupation_1,inctot_1),
                  query6B(classwkg_2,occupation_1,inctot_2),
                  query6B(classwkg_2,occupation_1,inctot_3),
                  query6B(classwkg_2,occupation_1,inctot_4),
                  query6B(classwkg_2,occupation_2,inctot_1),
                  query6B(classwkg_2,occupation_2,inctot_2),
                  query6B(classwkg_2,occupation_2,inctot_3)}
            Info: 11 tuples computed. Info:
            Total elapsed time: 1015 ms.
```

Query processing with the transformed query with semantic rules and materialized view:

Time saving = 9852-(452+1015)= 8385 ms.

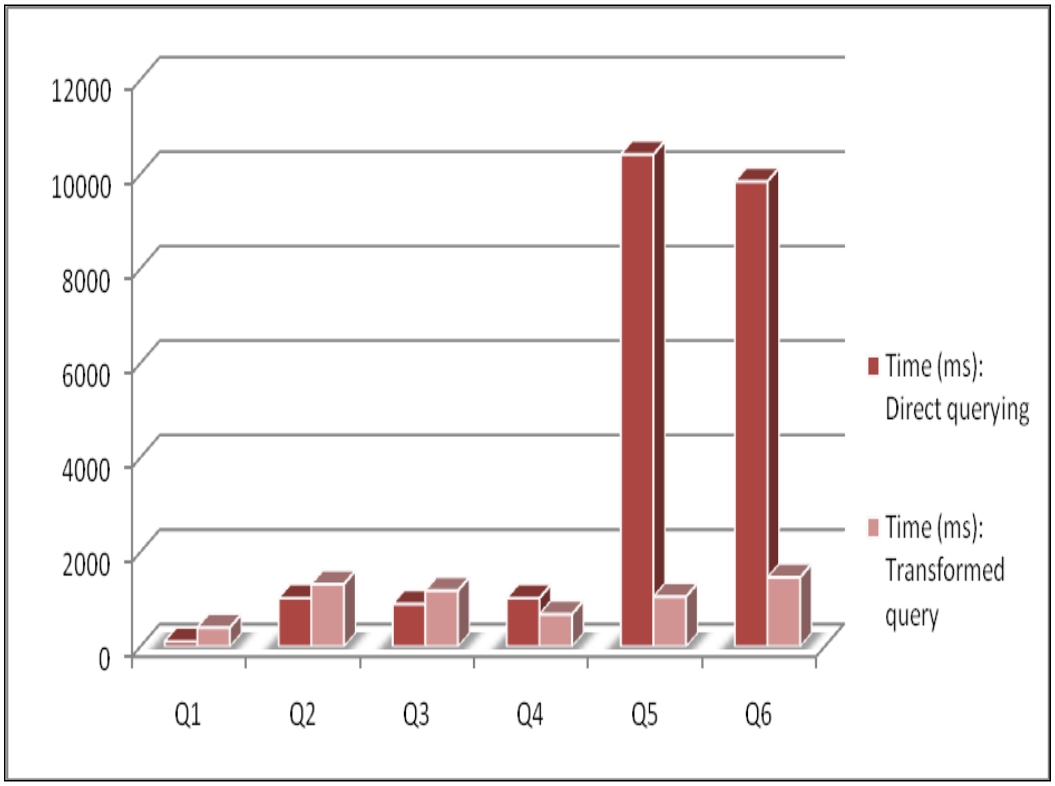


Figure 9. Time Comparison of Direct Querying versus Transforming Queries with Materialized Views and Data Mining Models Prior to Accessing the Database Contents

Table 3. Processing Time Summarization of Direct Querying versus Intelligent Querying Based on Semantic Transformation Using Materialized Views and Mining Models

Query characteristics	Processing time (ms)		Reduced time (ms)	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%

5. Conclusion

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 the 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.

Acknowledgements

This work has been supported by grants from the National Research Council of Thailand (NRCT) and Suranaree University of Technology via the funding of Data Engineering Research Unit.

References

[1] Afrati FN, Li C, Ullman JD, “Using views to generate efficient evaluation plans for queries”, Journal of Computer and System Sciences 73, 5, (2007), pp. 703-724.

[2] Agrawal R, Srikant R, “Fast algorithm for mining association rules”, In: VLDB, **(1994)**, pp. 487-499.

[3] Arago M, Fernandes A, “Logic-based integration of query answering and knowledge discovery”, In: 6th Flexible Query Answering Systems, **(2004)**, pp. 68-83.

[4] Blockeel H, Calders T, Fromont E, Goethals B, “Mining views: data base views for data mining”, In: IEEE ICDE, **(2008)**, pp. 1608-1611.

[5] Calders T, Goethals B, Prado A, “Integrating pattern mining in relational databases”, In: PKDD, **(2006)**, pp. 454-461.

[6] Chang J, Lee S, “Query reformulation using materialized views in data warehousing environment”, In: ACM Int. Workshop on Data Warehousing and OLAP, **(1998)**, pp. 54-59.

[7] Chaudhuri S, “Generalization and a framework for query modification”, In: IEEE ICDE, **(1990)**, pp. 138-145.

[8] Chaudhuri S, Krishnamurthy S, Potamianos S, Shim K, “Optimizing queries with materialized views”, In: IEEE ICDE, **(1995)**, pp. 190-200.

[9] Chaudhuri S, Narasayya V, Sarawagi S, “Extracting predicates from mining models for efficient query evaluation”, ACM Trans. Database Systems, vol. 29, no. 3, **(2004)**, pp. 508-544.

[10] Chen CM, Rossopoulos N, “The implementation and performance evaluation of the ADMS query optimizer: integrating query result caching and matching”, In: EDBT, **(1994)**, pp. 323-336.

[11] Chu W, Chen Q, “A structured approach for cooperative query answering”, IEEE Trans. Knowledge and Data Engineering, vol. 6, **(1994)**, pp. 738-749.

[12] Datalog Educational System, version 2.0. http://www.fdi.ucm.es/profesor/fernan/DES/.

[13] Erlang Programming Language, release 14. http://www.erlang.org.

[14] Gou G, Kormilitsin M, Chirkova R, “Query evaluation using overlapping views: completeness and efficiency”, In: ACM SIGMOD, **(2006)**, pp. 37-48.

[15] Halevy A, “Answering queries using views: a survey”, The VLDB Journal, vol. 10, no. 4, **(2001)**, pp. 270-294.

[16] Han J, Huang Y, Cercone N, Fu Y, “Intelligent query answering by knowledge discovery techniques”, IEEE Trans. Knowledge and Data Engineering, vol. 8, no. 3, **(1996)**, pp. 373-390.

[17] IBM, IBM intelligent miner scoring, administration and programming for DB2 version 7.1. New York: IBM **(2001)**.

[18] Integrated Public Use Microdata Series (IPUMS), Minnesota Population Center. http://www.ipums.org.

[19] Lin T, Cercone N, Hu X, Han J, “Intelligent query answering based on neighborhood systems and data mining techniques”, In: IEEE IDEAS, **(2004)**, pp. 91-96.

[20] Microsoft Corporation, OLE DB for data mining. Redmond, WA: Microsoft Corporation, **(2000)**.

[21] Muslea I, “Machine learning for online query relaxation”, In: ACM SIGMOD, **(2004)**, pp. 246-255.

[22] Srivastava D, Das S, Jagadish HV, Levy AY, “Answering queries with aggregation using views”, In: VLDB, **(1996)**, pp. 318-329.

Authors

Nittaya Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. She received her B.S. from Mahidol University, Thailand, in 1985, M.S. in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, USA, in 1999. She is a member of ACM and IEEE Computer Society. Her research of interest includes Knowledge Discovery in Databases, Artificial Intelligence, Logic Programming, Deductive and Active Databases.

Kittisak Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. He received his bachelor degree in Mathematics from Srinakarinwirot University, Thailand, in 1986, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in computer science from Nova Southeastern University, USA., in 1999. His current research includes Data mining, Artificial Intelligence, Functional and Logic Programming Languages, Computational Statistics.