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 $c \in C$
   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 $A$
3. If $A$ is empty, then return $Q$; otherwise proceed to the next step
4. 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 Conjoin a clause body with predicates appeared in $A$
5. 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:

```erlang
main() ->
    NameList = mylib:read_file("ipum.NAMES"," ", [ ]),
    mylib:text_file(write,NameList,filetemp),
    [H | Tail] = NameList, [F, C | T] = lists:reverse(H),
    NewClass = [F | T], % make normal 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([ ]) -> [].
```

138
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, ","),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,_,X,[],_,_) -> format("~nfinal set=~p",[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 , 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", ", "),
    UsedData=map(fun(EachL)-->lists:zip(AttrName,EachL) end,LinesList ),
    allPossibleAttr([H|T]) ->
        format("~n------------START-create rules------------"),
        {_,[AllL]}=file:consult("set.raw"),
        AllAsso2=[list(X)|| X<->[AllL,length(list(X))>1 ],
        format("~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 ----------~n).
%% create prolog file Rules+Facts
transform_to_prolog([Body,Head,Conf],IO) ->
if (length(Head)=1) -> [Head1]=Head,
  io:format(IO,"~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,[Val|[]]) -> Val;
searchL(_,Set,[Another,_|T]) -> searchL(Set,T);
searchL(_,[]) -> 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.
### Table 1. Attribute Details of the Household Records

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>Record-ID</td>
</tr>
<tr>
<td>2</td>
<td>gq</td>
<td>Group quarters status</td>
</tr>
<tr>
<td>3</td>
<td>gqtypeg</td>
<td>Group quarters type -- general</td>
</tr>
<tr>
<td>4</td>
<td>farm</td>
<td>Farm status</td>
</tr>
<tr>
<td>5</td>
<td>ownershg</td>
<td>Ownership of dwelling -- general</td>
</tr>
<tr>
<td>6</td>
<td>value</td>
<td>Value of housing unit</td>
</tr>
<tr>
<td>7</td>
<td>rent</td>
<td>Monthly rental payment</td>
</tr>
<tr>
<td>8</td>
<td>frotinc</td>
<td>Family total income</td>
</tr>
<tr>
<td>9</td>
<td>nfams</td>
<td>Number of families within each household unit</td>
</tr>
<tr>
<td>10</td>
<td>ncouples</td>
<td>Number of married couples in a household</td>
</tr>
<tr>
<td>11</td>
<td>nmothers</td>
<td>Number of mothers within each household unit</td>
</tr>
<tr>
<td>12</td>
<td>nfathers</td>
<td>Number of fathers within each household unit</td>
</tr>
<tr>
<td>13</td>
<td>momloc</td>
<td>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)</td>
</tr>
<tr>
<td>14</td>
<td>stepmom</td>
<td>Probable step/adopted mother (whether a person’s mother was likely to have been the person’s stepmother or adoptive mother)</td>
</tr>
<tr>
<td>15</td>
<td>momrule</td>
<td>Rule for linking mother</td>
</tr>
<tr>
<td>16</td>
<td>poploc</td>
<td>Father’s location in the household (a constructed variable to identify social relationship such as stepfather, adoptive father, as well as biological father)</td>
</tr>
<tr>
<td>17</td>
<td>steppop</td>
<td>Probable step/adopted father</td>
</tr>
<tr>
<td>18</td>
<td>poprule</td>
<td>Rule for linking father</td>
</tr>
<tr>
<td>19</td>
<td>sploc</td>
<td>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)</td>
</tr>
<tr>
<td>20</td>
<td>sprule</td>
<td>Rule for linking spouse</td>
</tr>
<tr>
<td>21</td>
<td>famsize</td>
<td>Number of own family members in household</td>
</tr>
<tr>
<td>22</td>
<td>nchld</td>
<td>Number of own children in the household</td>
</tr>
<tr>
<td>23</td>
<td>nchld5</td>
<td>Number of own children under age 5 in household</td>
</tr>
<tr>
<td>24</td>
<td>famunit</td>
<td>Groups of related individuals in the household</td>
</tr>
<tr>
<td>25</td>
<td>eldch</td>
<td>Age of eldest own child in household</td>
</tr>
<tr>
<td>26</td>
<td>yngch</td>
<td>Age of youngest own child in household</td>
</tr>
<tr>
<td>27</td>
<td>nsibs</td>
<td>Number of own siblings in the household</td>
</tr>
<tr>
<td>28</td>
<td>relateg</td>
<td>Relationship to household head -- general</td>
</tr>
<tr>
<td>29</td>
<td>age</td>
<td>Age of a person</td>
</tr>
<tr>
<td>30</td>
<td>sex</td>
<td>Sex of a person</td>
</tr>
<tr>
<td>31</td>
<td>raceg</td>
<td>Race -- general</td>
</tr>
<tr>
<td>32</td>
<td>marst</td>
<td>Marital status</td>
</tr>
<tr>
<td>33</td>
<td>chborn</td>
<td>Number of children ever born to each woman</td>
</tr>
<tr>
<td>34</td>
<td>bplg</td>
<td>Birthplace -- general</td>
</tr>
</tbody>
</table>
**Table 1.** example of household records, each record has 34 attributes.

```plaintext
table1(1,gq_1,gqtypeg_0,farm_1,ownershg_1,value_6,rent_0,fiotinc_3,nfams_1,ncouples_1, nmothers_0,nfathers_0,momloc_0,stepmom_0,momrule_0,steppop_0,p oprule_0,sploc_2,sprule_1,famsize_2,nchild_0,ncht5_0,famunit_1,eldch_10,yngc h_10,nsibs_0,relateg_1,age_6,sex_1,raceg_1,marst_6,chborn_0,bplg_1).

table1(2,gq_1,gqtypeg_0,farm_1,ownershg_1,value_6,rent_0,fiotinc_3,nfams_1,ncouples_1, nmothers_0,nfathers_0,momloc_0,stepmom_0,momrule_0,steppop_0,p oprule_0,sploc_2,sprule_1,famsize_2,nchild_0,ncht5_0,famunit_1,eldch_10,yngc h_10,nsibs_0,relateg_2,age_6,sex_1,raceg_1,marst_1,chborn_0,bplg_1).

table1(3,gq_1,gqtypeg_0,farm_1,ownershg_1,value_7,rent_9,fiotinc_3,nfams_2,ncouples_0, nmothers_1,nfathers_0,momloc_0,stepmom_0,momrule_0,steppop_0,p oprule_0,sploc_2,sprule_1,famsize_2,nchild_1,ncht5_0,famunit_1,eldch_10,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,fiotinc_3,nfams_2,ncouples_0, nmothers_1,nfathers_0,momloc_0,stepmom_0,momrule_0,steppop_0,p oprule_0,sploc_2,sprule_1,famsize_2,nchild_0,ncht5_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,fiotinc_3,nfams_2,ncouples_0, nmothers_1,nfathers_0,momloc_0,stepmom_0,momrule_0,steppop_0,p oprule_0,sploc_2,sprule_1,famsize_1,nchild_0,ncht5_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

**Table 2.** example of person records, each record has 27 attributes.

```plaintext
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,incother_1,poverty_6, migrat5g_1,migplac5_0,movedin_7,verstat_1,tranwork_10,occupation_2).

table2(2,school_1,educrec_8,schltype_1,empstatg_1,labforce_2,occscore_3,sei_3, classwkg_1,wkswork2_3,hrswork2_6,yrlastwk_0,workedyr_2,inctot_3, incwage_2,incbus_2,incfarm_1,incother_1,poverty_6, migrat5g_1,migplac5_1,movedin_0,verstat_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,incother_1,poverty_2, migrat5g_2,migplac5_1,movedin_2,verstat_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,incother_4,poverty_2,migrat5g_1,migplac5_1,movedin_0,verstat_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,incother_1,poverty_4,migrat5g_2, migplac5_1,movedin_0,verstat_2,tranwork_0,occupation_2).
```

**Figure 5.** Example of Person Records, Each Record has 27 Attributes
### Table 2. Attribute Details of the Person Records

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

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.
Figure 6. The DES System with SQL Support (above) and Datalog Querying (below)
Figure 7. Part of Semantic Rules Transformed from the Association Mining Models

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 households. (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,
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,
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,

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> 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.

Query 5: Ask for educational record and total income of female person. (Joining of table1 and table2)


DES-Datalog> query5(X,Y).
\{ query5(educrec_0,incwage_4), query5(educrec_1,incwage_1), query5(educrec_1,incwage_2), query5(educrec_1,incwage_4), query5(educrec_2,incwage_1), query5(educrec_2,incwage_3), query5(educrec_3,incwage_1), query5(educrec_3,incwage_3), query5(educrec_3,incwage_4), query5(educrec_4,incwage_1), query5(educrec_4,incwage_2), query5(educrec_4,incwage_4), query5(educrec_5,incwage_1), query5(educrec_5,incwage_2), query5(educrec_5,incwage_4), query5(educrec_6,incwage_1),

149
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,

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_8,incwage_4),
  query5B(educrec_9,incwage_1),
  query5B(educrec_9,incwage_2),
  query5B(educrec_9,incwage_3),
  query5B(educrec_9,incwage_4) }
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.

**Query 6:** 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,B6,B7,B8,B9,B10,B11,B12,B13,B14,B15,B16,B17,B18,B19,B20,B21,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_1),
  query6(classwkg_1,occupation_2,inctot_2),
  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.

  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

<table>
<thead>
<tr>
<th>Query characteristics</th>
<th>Processing time (ms)</th>
<th>Reduced time (ms)</th>
<th>Time usage efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Direct answer</td>
<td>Intelligent answer</td>
<td></td>
</tr>
<tr>
<td>Query1: ask one information with a single condition, null answer</td>
<td>110</td>
<td>405</td>
<td>-295</td>
</tr>
<tr>
<td>Query2: ask one information with a single condition</td>
<td>1,029</td>
<td>1,325</td>
<td>-296</td>
</tr>
<tr>
<td>Query3: ask one information with a single condition</td>
<td>906</td>
<td>1,188</td>
<td>-282</td>
</tr>
<tr>
<td>Query4: ask one information with two conditions</td>
<td>1,028</td>
<td>688</td>
<td>340</td>
</tr>
<tr>
<td>Query5: ask two information with a single condition</td>
<td>10,423</td>
<td>1,060</td>
<td>9,363</td>
</tr>
<tr>
<td>Query6: ask three information with four conditions</td>
<td>9,852</td>
<td>1,467</td>
<td>8,385</td>
</tr>
</tbody>
</table>

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


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.