Unit 6

Data Mining Query Languages

Problem Description

- We are armed with two powerful tools
  - Database management systems
  - Efficient and effective data mining algorithms and frameworks
- Generally, this work asks:
  - “How can we merge the two?”
  - “How can we integrate data mining more closely with traditional database systems, particularly querying?”

Three Different Answers

- **DMQL: A Data Mining Query Language for Relational Databases (Han et al, Simon Fraser University)**
- Integrating Data Mining with SQL Databases: **OLE DB for Data Mining** (Netz et al, Microsoft)
- **MSQL:** A Query Language for Database Mining (Imielinski & Virmani, Rutgers University)

Some Common Ground

- Create and manipulate data mining models through a SQL-based interface ("Command-driven" data mining)
- Data mining should be performed on data in the database (should not need to export to a special-purpose environment)
- Approaches differ on what kinds of models should be created, and what operations we should be able to perform

Data Mining Languages Standardization

- Standardizing the Data Mining Languages will serve for the following purposes:
  - Systematic Development of Data Mining Solutions.
  - Improve interoperability among multiple data mining systems and functions.
  - Promote the education.
  - Promote use of data mining systems in industry and society.

Data Mining Query Language (DMQL)→ Defination

- The Data Mining Query Language is actually based on the Structured Query Language (SQL).
- Data Mining Query Languages can be designed to support interactive data mining.
- DMQL can work with databases and data warehouses as well.
- DMQL can be used to define data mining tasks.
- Particularly we examine how to define data warehouses and data marts in DMQL.

Data Mining Query Language (DMQL)

- DMQL provides a display in command to view resulting rules, but no advanced way to query them.
- Suggests that a GUI interface might aid in the presentation of these results in different forms (charts, graphs, etc.)

Data Mining Query Language (DMQL) → Primitives

DMQL Commands specify the following :

1. The set of data relevant to the data mining task (the training set)
2. Background knowledge

   Concept hierarchies based on attribute relationships, etc.
3. Various thresholds

   Minimum support, confidence, etc.
4. The kinds of knowledge to be discovered

   Generalized relation, Characteristic rules, Discriminant rules, Classification rules, Association rules

Data Mining Query Language (DMQL) → Syntax

- Syntax for specification of
  - task relevant data
  - the kind of knowledge to be mined
  - concept hierarchy specification
  - interestingness measure
  - pattern presentation and visualization
- Putting it all together — a DMQL query

Data Mining Query Language (DMQL) → Syntax

- Syntax

use database <database_name>

Specify background knowledge → {**use hierarchy** <hierarchy_name> for <attribute>}

Specify rules to be discovered → <rule_spec>

related to <attr_or_agg_list>

Relevant attributes or aggregations → **from** <relation(s)>

Collect the set of relevant data to mine → [**where** <conditions>]

[**order by** <order list>]

Specify threshold parameters → {**with** [<kinds of>] **threshold** = <threshold_value> [for <attribute(s)>]}

Syntax For Task-Relevant Data Specification

- The first step in defining a data-mining task is the specification of the task-relevant data, that is, the data on which mining is to be performed.

- This involves specifying the database and tables or data warehouse containing the relevant data, conditions for selecting the relevant data, the relevant attributes or dimensions for exploration, and instructions regarding the order or grouping of the data retrieved.

Syntax For Task-Relevant Data Specification

- DMQL provides clauses for the specification of such information, as follows:

  - **Use database** (database_name) or **use data warehouse** (data_warehouse_name): the use clause directs the mining task to the database or data warehouse specified.
  - **From relation** (s)/cubes(s) [where (condition)]: the from and where clauses respectively specify the database tables or data cubes involved, and the conditions defining the data to be retrieved
  - **in relevance** to att_or_dim_list (This clause lists the attributes or dimensions for exploration.)
  - **order by** order_list (The order by clause specifies the sorting order of the task relevant data.
  - **group by** grouping_list (The group by clause specifies criteria for grouping the data.)
  - **having** condition (The having clause specifies the condition by which groups of Data are considered relevant.)

Data Mining Query Language (DMQL) → Example

```
use database AllElectronics_db
use hierarchy location_hierarchy for B.address
mine characteristics as customerPurchasing
analyze count%
in relevance to C.age, I.type, I.place_made
from customer C, item I, purchases P,items_sold S,works_at W, branch
where I.item_ID = S.item_ID and S.trans_ID = P.trans_ID
    and P.cust_ID = C.cust_ID and P.method_paid = ``AmEx''
    and P.empl_ID = W.empl_ID and W.branch_ID =B branch ID B.branch_ID
    and B address B.address = ``Canada" and I price I.price >= 100
with noise threshold = 0.05

display as table
```

The kind of knowledge to be mined

- The (Mine_Knowledge_Specification) statement is used to specify the kind of knowledge to be mined.
- Its syntax is defined below for

characterization,
discrimination,
Prediction,
association, and
classification.

Characterization

(Mine_Knowledge_Specification)::=mine characteristics[as (pattern_name)]
Analyze (measure(s))

- This specifies that characteristic descriptions are to be mined.
- The **analyze clause**, when used for characterization, specifies aggregate measure, such as count, sum, or count% (percentage count, i.e., the percentage of tuples in the relevant data set with the specified (characteristics).
- These measures are to be computed for each data characteristic found

Discrimination

Mine_Knowledge_Specification ::=
mine comparison as [pattern_name]
for target_class where target_condition
{versus contrast_class_i where contrast_condition_i}
analyze measure(s)

Data discrimination, also called discrimination by algorithm, is bias that occurs when predefined data types or data sources are intentionally or unintentionally treated differently than others.

Association

```
Mine_Knowledge_Specification ::=
mine associations as [pattern_name]
```

Association rules are created by searching data for frequent if-then patterns and using the criteria support and confidence to identify the most important relationships.

Classification

```
Mine_Knowledge_Specification ::=
mine classification [as pattern_name]
analyze classifying_attribute_or_dimension
```

Classification in data mining is a common technique that separates data points into different classes. It allows to organize data sets of all sorts, including complex and large datasets as well as small and simple ones.

Prediction

```
Mine_Knowledge_Specification ::=
mine prediction [as pattern_name]
analyze prediction_attribute_or_dimension
{set {attribute_or_dimension_i= value_i}}
```

Prediction in data mining is to identify data points purely on the description of another related data value

Concept hierarchy specification

- Concept hierarchies allow the mining of knowledge at multiple levels of abstraction.
- In order to accommodate the different viewpoints of users with regard to the data, there may be more than one concept hierarchy per attribute or dimension.

- *For instance, some users may prefer to organize branch locations by provinces and states, while others may prefer to organize them according to languages used.*
- *In such cases, a user can indicate which concept hierarchy is to be used with statement.*

Concept hierarchy specification

- To specify what concept hierarchies to use
  use hierarchy <hierarchy> for <attribute_or_dimension>
- We use different syntax to **define different type of hierarchies**
  - Schema hierarchies
    define hierarchy time_hierarchy on date as [date,month, quarter,year]
  - Set-grouping hierarchies
    ***define hierarchy** age_hierarchy for age on customer as*
    level1: {young, middle_aged, senior} < level0: all
    level2: {20, ..., 39} < level1: young
    level2: {40, ..., 59} < level1: middle_aged
    level2: {60, ..., 89} < level1: senior

Concept hierarchy specification

- Operation-derived hierarchies

```
define hierarchy age_hierarchy for age on customer as
{age_category(1), ..., age_category(5)} := cluster(default,age, 5) <
    all(age)
```

- rule-based hierarchies

```
define hierarchy profit_margin_hierarchy on item as
level_1: low_profit_margin < level_0: all
if (price - cost)< $50
level_1: medium-profit_margin < level_0: all
if ((price - cost) > $50) and ((price - cost) <=$250))
level_1: high_profit_margin < level_0: all if (price -
cost) > $250
```

Syntax for Interestingness Measure Specification

- The user can control the number of uninteresting patterns returned by the data mining system by specifying measures of pattern interestingness and their corresponding thresholds.
- Interestingness measures and thresholds can be specified by the user with the statement:

  with *<interest_measure_name>* ***threshold***=*threshold_value*
- Example:

  with *support* ***threshold*** *= 0.05*

  with *confidence* ***threshold*** *= 0.7*

Measurements of Pattern's Interestingness

- **Simplicity**
  Association rule length, decision tree size
- **Certainty**
  Confidence, P(A|B) = n(A and B)/ n(B), classification reliability or accuracy, certainty factor, rule strength, rule quality, discriminating weight
- **Utility**
  Potential usefulness,support(association),noise threshold (description)
- **Novelty**
  Not previously known, surprising (used to remove redundant rules)

Syntax for Pattern Presentation and Visualization Specification

- **"How can users specify the forms of presentation and visualization to be used in displaying the discovered patterns?"**
- Our data mining query language needs syntax that allows users to specify the display of discovered patterns in one or more forms, including rules, tables cross tabs, pie or bar charts, decision trees ,cubes ,curves or surfaces, We define the DMQL display statement for this purpose
- We have syntax which allows users to specify the display of discovered patterns in one or more forms.

display *as <result_form>*
where <result_form> could be any of the knowledge presentation or visualization forms listed .

Syntax for Pattern Presentation and Visualization Specification

- The user can alternately view the patterns at different levels of abstractions with the use of following DMQL syntax:

*(**Multilevel_Manapulation**)::= **roll up on***
(attribute_or_dimension)
*|**drill down on** (attribute_or_dimension)*
*|**add** (attribute_or_dimension)*
*|**drop** (attribute_or_dimension)*

An example of a DMQL query → Scenario

Suppose, as a marketing manager of *ABCompany*, you would like to *characterize* the buying habits of the customers who purchase items priced at no less than Rs.100, *with respect to* the customer's age, the type of item purchased, and the place in which the *item was made.*
For each characteristic discovered, we would like to know the percentage of customers having that characteristic. In particular we are only interested in *purchases made* in Nepal, and paid for with a Visa, ("Visa") credit card. We would like to view the resulting descriptions in the form of a table.

Express This data mining query in DMQL.

An example of a DMQL query → Solution

```
use database ABCompany_db
use hierarchy location_hierarchy for B.address
mine characteristics as customerPurchasing
analyze count%
in relevance to C.age, I.type, I.place_made
from customer C, item I, purchase P, items_sold S, works_at W,
    branch B
where I.itemID=S.itemID
    And S.transID=P.transID
    And P.custID=C.custID
    And P.method_paid= “Visa”
    And B.address=”Nepal”
    And I.Cost < 100
With noise threshold: =5%

Display as table
```

An example of a DMQL query → Solution

- The data mining query is parsed to form an SQL query that retrieves the set of task-relevant data from the ABCompany database.
- The concept hierarchy is used to generalize branch locations to high level concept levels such as “Nepal”.
- An algorithm for mining characteristic rules, which uses the generalized data, can then be executed.
- The mined characteristic descriptions, derived from the attributes age, type, and place made, are displayed as a table, or generalized relation.
- The percentage of task-relevant tuples satisfying each generalized tuple is shown as count%.
- If no visualization form is specified a default form is used.
- The noise threshold of 5% means any generalized tuple found that represents less than 5% if the total count is omitted from display
- Similarly, the complete DMQL specification of data mining queries
- for discrimination, association, classification, and prediction can be given.

Other Data Mining Languages and Standardization efforts → Other than DMQL

- **OLE DB for Data Mining**
  - Just think about the types of data you use on a regular basis. The data is found in text files, emails, Excel spreadsheets, Word documents, and so on. You want to access all of this data in a way similar to the way you access relational data, ideally through the same API. OLE DB was introduced for this purpose
  - Object Linking and Embedding (OLE) DB is based on the Microsoft Component Object Model (COM) infrastructure.
  - The main purpose of OLE DB is to provide a standard way to access tabular data

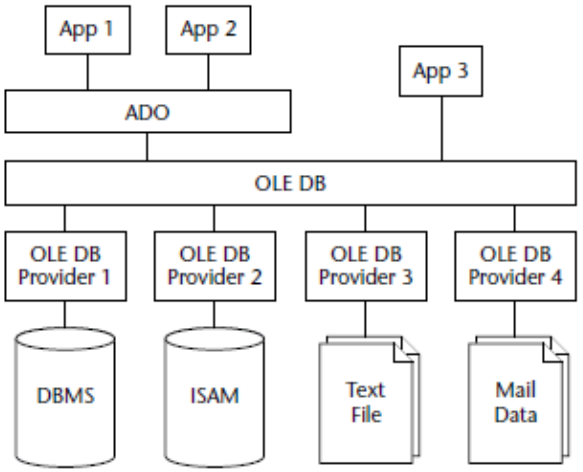

Figure shows an overview of the architecture of OLE DB. OLE DB providers implement a set of predefined interfaces on top of native data storage software such as DBMSs, ISAM, text file, mail data, or Excel. Consumer applications can access these data sources in a same way, directly through OLE DB or indirectly through Active Data Objects (ADO). ADO is a high-level interface on top of OLE DB that most applications developers use.

Other Data Mining Languages and Standardization efforts → Other than DMQL

- **MSQL**

  It extracts rules that are based on descriptors, each descriptor being an expression of the type $(A_i = a_{ij})$, where A_i is an attribute and a_{ij} is a value or a range of values in the domain of A_i.

  MSQL extracts propositional rules of the form A → B, where A is a conjunctset and B is a descriptor. As a consequence, only one attribute can appear in the consequent of a rule.

```
GetRules(C) [INTO <rulebase name>]
    [WHERE <rule constraints>]
    [SQL-group-by clause]
    [USING encoding-clause]
```

C is the source table and **rule constraints** are conditions on the desired rules. The **USING** part enables to discretize numerical values. **rulebase name** is the name of the object in which rules will be stored.

Designing Graphical User Interfaces Based on a Data Mining Query Language

- A data mining query language provides necessary primitives that allow users to communicate with data mining systems.
- However, inexperienced users may find data mining query languages occurred to use and the syntax difficult to remember.
- Instead, users may prefer to communicate with data mining systems through a graphical user interface (GUI).
- In relational data base technology, SQL serves as a standard “core” language for relational systems, on top of which GUI’s can easily be designed.

Designing Graphical User Interfaces Based on a Data Mining Query Language

A data mining GUI may consist of the following functional components

- **Data collection and data mining query compositions:** This component allows the user to specify task-relevant data sets and to compose data mining queries. It is similar to GUI's used for the specification of relational queries.

- **Presentations of discovered patterns:** This component allows the display of the discovered patterns in various forms, including tables, graphs, charts, curves and other visualization techniques

Designing Graphical User Interfaces Based on a Data Mining Query Language

- **Manipulation of data mining primitives:**
  This component may allow the dynamic adjustment of the data mining thresholds, as well as the selection, display and modification of concept hierarchies. It may also allow the modification of previous data mining queries or conditions.
- **Interactive multilevel mining:**
  This component should allow roll-up or drill-down operations on discovered patterns.
- **Other miscellaneous information:**
  This component may include on-line help manuals indexed search, debugging, and other interactive graphical facilities. The design of a graphical user interface should also take into consideration different classes of users of a data mining system.

Methods for integration of DM with DW System

Based on different architecture designs, a DM system can be integrated with a DB/DW system using the following coupling schemes.

- No Coupling
- Loose coupling,
- Semi-tight coupling, and
- Tight coupling.

Methods for integration of DM with DW System

- No Coupling

  No coupling means that a DM system will not utilize any function of a DB or DW system. It may fetch data from a particular source (such as a file system), process data using some data mining algorithms,and then store the mining results in another file. Such a system, though simple, suffers from several drawbacks.

Methods for integration of DM with DW System

- Loose Coupling

  Loose coupling means that a DM system will use some facilities of a DB or DW system, fetching data from a data repository managed by these systems, performing data mining, and then storing the mining results either in a file or in a designated place in a database or data warehouse.

  Loose coupling is better than no coupling since it can fetch any portion of data stored in databases or data warehouses by using query processing.

Methods for integration of DM with DW System

- Semi Tight Coupling

  Semi tight coupling means that besides linking a DM system to a DB/DW system, efficient implementations of a few identical data mining functions can be provided in the DB/DW system.
  These primitives can include sorting, indexing, aggregation, histogram analysis, multilayer join, and precipitation of some essential statistical measures, such as sum, count, max, min, standard deviation and so on.

Methods for integration of DM with DW System

- Tight Coupling

  Tight coupling means that a DM system is smoothly integrated into the DB/DW system. The data mining subsystem is treated as one functional component of an information system.
  Data mining queries and functions are optimized based on mining query analyses, data structures, indexing schemes, and query processing methods of a DB/DW system.