CHAPTER 9

Data Mining Query Language

9.1 Data Mining Query Language

9.2 Syntax for Task-Relevant Data Specifications:

9.3 Syntax for Concept Hierarchy Specification

9.4 Mining characteristic descriptions:

9.5 Designing Graphical User Interfaces Based on a Data Mining Query Language

9.6 Architectures of data mining system:

9.7 Review Question

9.8 References

9. Data Mining Query Language

9.1 A Data Mining Query Language:

A desired feature of data mining systems is the ability to support ad hoc and interactive data mining in order to facilitate the flexible and effective knowledge discovery. Data mining query languages can be designed to support such a feature.

The importance of the design of a good data mining query language can also be seen from observing the history of relational data base systems. Relational data base systems have dominated the database market for decades. The standardization of relational query languages, which occurred at the early stages of relational database development, is widely credited for the success of the relational database field. Although each commercial relational database system has its own graphical user interface, the underlying core of each interface is a standardized relational query language. The standardization of relational query languages provided a foundation on which relational systems for developed and evolved. It facilitated information exchange and technology transfer, and promoted commercialization and wide acceptance of relational database technology. The recent standardization activities in database systems, such as work relating to SQL-3, and so on, further illustrate the importance of having a standard database language for success the development and commercialization of database systems. Hence, having a good query language for data mining may help standardize the development of platforms for data mining systems.

Designing a comprehensive data mining language is challenging because data mining covers a wide spectrum of tasks, from data characterization to mining association rules, data classification, and evolution analysis. Each task has different requirements the design of an effective data mining query language requires a deep understanding of the power, limitation, and underlying mechanisms of the various kinds of data mining tasks.

How would you design a data mining query language? Earlier in this chapter, we looked art primitives for defining a data mining task in the form of a data mining query. The primitives specify the following:

- The set of task-relevant data to be mined
- The kind of knowledge to be mined
- The background to be mined
- The background knowledge to be used in the discovery process
- The interestingness measures and thresholds for pattern evolution
- The expected representation for visualizing the discovered patterns

Based on these primitives, we design a query language for data mining called DMQL (data mining query language), DMQL allows the ad hoc mining of several kinds of knowledge from relational databases and data warehouses at multiple levels of abstraction.

The language adopts an SQL-like syntax, so that it can easily be integrated with the relational query language SQL. The syntax of DMQL is defined in an extended BNF grammar, where "[ ]" represents 0 or one occurrence, "{ }" represents 0 or more occurrences, and words in sans serif font represent keywords.

9.2 Syntax for Task-Relevant Data Specifications:

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

Use database (database_name) or use the data warehouse (data_warehouse_name): the use clause directs the mining task to the database or data warehouse specified.

From (relation (s)/cubes(s)) [ware (condition)]: the from and where clauses respectively specify the database tables or data cubes involved, and the conditions defining the data to be retrieved

(DMQL)::= (DMQL_statement} ;{(DMQL_Statement)

(DMQL_Statement)::= (Data_Mining_Statment)

| {Concept_Hierarchy,.Denmtion_Statement)

| (Visualization_and_Presentation)

(Data_Mining_Statement)::=

Use database (database_name) | use data warehouse (data_warehouse_name)

|<use hierarchy (hierarchy_name>for (attribute_or_dimension)}

<Mint_Knowledge_Specification>

In relevance to {attribute_or_dimension_list}

From (relation (s) /cube(s))

[Where (condition)]

[Order by (order list)]

[Group by (grouping_list)]

[Having (condition)]

[With [(interest_measure_name)] threshold={threshold_value)

[For (attribute(s))]

(Mine_Knowledge_Specification)::=(Mine_Char) |(Mine_Discr) |(Mine_Assoc) |(Mine_Class)

{Mine_Char)::= mine characteristics [as (pattern_name)]

Analyze {measure(s))

<Mine_Discr>::= mine comparison [as(pattern_name)]

For (target class) where (tarhet_condition)

(Verses (contrast_class_i)

(contrast_condition_i)}

Analyze (measure(s))

{Minc_Assoc}::= mine associations [as{pattern_name}]

[Matching {met pattern)]

{Mine_Class}::= mine classification [as{pattern_name}]

Analyze {classifying_attribute_or_dimension)

(Concept_Hierarchy _Dennition_Statement)::=

Define hierarchy (hierarchy_name)

[for (attribute_or_dimension}]

On (relation _or_cube_or_hierarchy)

As (hierarchy_description)

[Where (condition}]

{Visualization_and_Presentation}:=

Display as (result_form) | {MultUevel_Manipulation)}

(Multilevel _Manipulation)::= roll up on

(attribute_or_dimension}

| drill down on (attribute_or_dimension)

| add (attribute_or_dimension)

| Drop (attribute _or_dimension)

In relevance to (attribute_or_dimensionlist): 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.

These clauses form an SQL query to collect the task-relevant data.

Example: This example shows how to use DMQL to specify the task-relevant data, the mining of associations between items frequently purchased at ABCompany by Sri Lankan customers, with respect to customer income and age. In addition, the user specifies that the data are to be grouped by date. The data are retrieved from a relational database.

Use database ABCompany_db

In relevance to I .name,I.price, C.income, C.age

From customer C, item I, purchases P, items_sold S

Where I.item_ID=S. item.JD and S.trans_ID =P.trans_ID and P .custJD=C.cust_ID and C. country –“Sri Lanka”

Group by p.data

Syntax for Specifying the Kind of Knowledge to be mined

The (Mine_Knowledge_Specification) statement is used to specify the kind of knowledge to be mined. In other words, it indicates the data mining functionality to be performed. Its syntax is defined below for characterization, discrimination, 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.

9.3 Syntax for 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 use hierarchy (hierarchy_name) for {attribute _or_dimension) Otherwise, a default hierarchy per attribute or dimension is used.

Syntax for Interestingness Measure Specification:

The user can help 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 be the user with the statement with {(interest_measure_name)] threshold-(threshold_value)

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; display a

(Result _form)

Where the (result_form) could be any of the knowledge presentation or visualization forms listed above.

Interactive mining should allow the discovered patterns to be viewed at different concept levels or from different angles. This can be accomplished with roll-up and drill-down operations. Patterns can be rolled up, or viewed at a more general level, by climbing up the concept hierarchy of an attribute or dimension (replacing lower=-level concept values by higher-level values).Dropping attributes or dimensions con also perform generalization .For example, suppose that a pattern contains the attribute city given the location hierarchy city < province_or_state<country <continent, then dropping the attribute, province_or _state. Patterns can be drilled down on, or viewed at a less general level, by stepping down the concept hierarchy of an attribute or dimension patterns can also be made less general by adding attributes or dimensions to their description. The attribute added must be one of the attributes listed in the in relevance to clause for task-relevant 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)

Putting all together-An example of a DMQL query

In the above discussion, we presented DMQL syntax for specifying data mining queries in terms of the five data mini8ng primitives. For a given query, these primitive define the task-relevant data, the kind of knowledge to be mined, the concept hierarchies and interestingness measures to be used, and the representation forms for pattern visualization. Here we put these components together. Let's look at an example for the full specification of a DQML query.

9.4 Mining characteristic descriptions:

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, you would like to know the percentage of customers having that characteristic. In particular you are only interested in purchases made in Sri Lanka, and paid for with a Visa, ("Visa") credit card. You would like to view the resulting descriptions in the form of a table. This data mining query is expressed in DMQL as follows:

Use hierarchy location^hierarchy for B.address

Mine characteristics as customer purchasing

Analyze count%

N 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 L.itemJD=S.itemJD and S.transJD=P.transJD and P.custJD=C.custJD

And P.method_paid= "Visa" and P.empl_ID=W.empl_ID

And W.branch_ID=B.branch_ID and B.address="Sri Lanka" and Lprice_100

With noise threshold: =5%

Display **as table**

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 "Sri Lanka". 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.

9.5 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. Similarly, a data mining query language may serve as a "core language" for data mining system implementation is providing a basis for the development of GUI's for effective data mining.

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.
- **Hierarchy specification and manipulation:** this component allows for concept hierarchy specification, either manually by the user or automatically (based on analysis of the data at hand). In addition, this component should allow concept hierarchies to be modified by the user or adjusted automatically based on the given data set distribution.
- **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. In general, users of data mining systems can be classified into two categories: business analysts and business executives. Business analysts would like to have flexibility and convenience in selecting different portions of the data manipulating dimensions and levels, setting mining parameters, and tuning data mining processes. On the other hand, a business executive needs clear presentation and interpretation of data mining results, flexibility in viewing and comparing different data mining results and easy integration of data mining results interpret writing and presentation process. A well-designed data mining systems should provide friendly user interfaces for both kinds of users.

9.6 Architectures of data mining system:

With popular and diverse applications of data mining, it is expected that a good variety of data mining systems will be designed and developed in future years. Although rich and powerful data mining functions from the core of a data mining systems, like most software systems, the architecture and design of a data mining system is critical important. A good system architecture and design will facilities the system to make best use of the software environment, accomplish data mining tasks in an efficient and timely manner intemperate and exchange information with other information systems, be adaptable user's diverse requirements, and evolve with time.

With decades of research and development in the data base and information industry, database systems and data warehouse systems have become the mainstreams information systems. Tremendous amounts of data and information have been store and/or integrated in such systems. Moreover, comprehensive information processing and data analysis infrastructure have been or will be continuously and semantically constructed surrounding database systems and data warehouses. These include accessing, integration, consolidation, and transformation of multiple, heterogeneous databases, ODBC/OLE DB connections web-accessing and service facilities and reporting and OLAP analysis tools.

Under this situation,a critical question in the design of a data mining system becomes whether we should couple or integrate a data mining (DM) system with a data base system and / or a data warehouse (DW)system , and if we should, how to do it correctly, To answer these questions, we need to examine the possible ways of coupling or integrating a DM system and a DB/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. Let us examine these one by one.

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. First, a DB system provides a great deal of flexibility and efficiency at storing, organizing, accessing, and processing data. Without using a DB/DW system, a DM system may spend a substantial amount of time finding, collecting, cleaning, and transforming data. In DB and/or DW systems, data tend to be well organized, indexed, cleaned integrated, or consolidated, so that finding the task-relevant, high-quality data becomes an easy task. Second, there are many tested, scalable algorithms and data structures implemented in DB and DW systems. It is feasible to realize efficient, scalable implementations using such systems. Moreover, most data have been or will be stored in DB/DW systems. Without any coupling of such systems, a DM system will need to use other tools to extract data, making it difficult to integrate such a system into an information processing environment. Thus, no coupling represents a poor design.

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 con fetch any portion of data stored in databases or data warehouses by using query processing. Indexing and other system facilities. It incurs some advantages of the flexibility, efficiency, and other features provided by such systems. However, many loosely coupled min\ ing systems are mainly memory-based. Since mining itself does not explore data structures and query optimization methods provided by DB or DW systems, it is difficult for loose coupling to achieve high scalability and good performance with large data sets.

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. Moreover, some frequently used intermediate mining results can be pre-computed and stored in the DB/DW system. Since these intermediate mining results are either pre-computed or can be computed efficiently, this design will enhance the performance of a DM system.

Tight coupling:

Tight coupling means that a DM system is smoothly integrated into the DB/W 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. With further technology advances, DM, DB, and DW systems with evolve and integrate together as one information system with multiple functionalities .This will provide a uniform information processing environment.

This approach is highly desirable since it facilitates efficient implementations aof data mining functions, high system performance, and an integrated information processing environment.

With this analysis, one can see that mining system should be coupled with DB/DW system. Loose coupling, though not efficient, is better than no coupling since it makes use of both data and system facilities of a DB/SW system. Tight coupling is highly desirable, but its implementation is nontrivial and more research is needed in the area. Semi tight coupling is a compromise between loose and tight coupling. It is important to identify commonly used data mining primitives and provide efficient implementations of such primitives in DB or DW systems.

9.7 Review Question

1 Explain about Data Mining Query Language

2 Discus Syntax for Concept Hierarchy Specification

3. How Designing Graphical User Interfaces Based on a Data Mining Query Language can be possible

9.8 References

[1]. Data Mining Techniques, Arun k pujari 1st Edition

[2] .Data warehousung,Data Mining and OLAP, Alex Berson ,smith.j. Stephen

[3].Data Mining Concepts and Techniques ,Jiawei Han and MichelineKamber

[4]Data Mining Introductory and Advanced topics, Margaret H Dunham PEA

[5] The Data Warehouse lifecycle toolkit , Ralph Kimball Wiley student Edition