- J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29-54, 1997.
- V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data, pages 205-216, Montreal, Canada, June 1996.
- Microsoft. OLEDB for OLAP programmer's reference version 1.0. In http://www.microsoft.com/data/oledb/olap, 1998.
- K. Ross and D. Srivastava. Fast computation of sparse datacubes. In Proc. 1997 Int. Conf. Very Large Data Bases, 116-125, Athens, Greece, Aug. 1997.
- K. A. Ross, D. Srivastava, and D. Chatziantoniou. Complex aggregation at multiple granularities. In Proc. Int. Conf. of Extending Database Technology (EDBT'98), 263-277, Valencia, Spain, March 1998.
- S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cu[illegible]. In Proc. Int. Conf. of Extending Database Technology (EDBT'98), pages 168-182, Valencia, Spain, March 1998.
- E. Thomsen. OLAP Solutions: Building Multidimensional Information Systems. John Wiley & Sons, 1997.
- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. In Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data, 159-170, Tucson, Arizona, May 1997.


# Unit 3

# DATA MINING PRIMITIVES, LANGUAGES AND SYSTEM ARCHITECTURES

## INTRODUCTION

We all know that in Data mining process, automated extraction of patterns takes place. In practice, the generated patterns may easily surpass the size of the database. Moreover user may not use all the patterns found in large datasets, because certain patterns are irrelevant for his analysis. Due to lack of interaction with data mining system user cannot have provision of indication regarding portions of the database, kinds of patterns to choose, methods of visualization of patterns etc. Furthermore, many patterns found to be difficult to understand even though they are related to their analysis. So, a set of *data mining primitives* designed to facilitate users to communicate with the data mining systems and provide efficient and fruitful knowledge discovery. This allows the user to interactively communicate with the data mining system in order to examine the findings from different angles or depths and direct them in successful path. The main objectives of this chapter is to realize the difference between data mining operations and become aware of the process of specifying data mining tasks, exhibit a brief introduction to a query language for data mining language DMQL and explained about architecture of data mining with the present technologies.

### 3.1 Data Mining Primitives

Every user will keep in mind about what data analysis he would like to perform on datasets. This thought can be presented as data mining task in the form of a query to the data mining system. Users must be provided with a set of primitives to be used to communicate with the data mining system. By incorporating these primitives in a data mining query language, user interaction with the system becomes more flexible, gives foundation for the design of graphical user interface and standardize the data mining industry and practice. The input data mining query is defined in terms of the following:

- **Task – relevant data:** This is the First primitive that the user can specify portions of the database on which mining is to be performed. The data selected for analysis is nothing but relevant data. In real scenario, many of the patterns generated may be irrelevant according to the interests of the user. User can also specify the attributes of interest to be considered in the mining process. These attributes are referred as *relevant attributes*. Eg: If you are interested to find out relation ships between student and examination attended, then the attributes name of the student, exam attended, grade secured can be specified as relevant attributes for mining.
- **Kinds of knowledge :** This can be performed by using data mining functions like association, clustering, classification, discrimination , characterization analysis etc. Eg: Associations between customers and the items these customers are likely to buy can be mined.
- **Background knowledge:** Domain knowledge about the mining is known as background knowledge. The is so important to know about the discovery process and to evaluates the patterns found. To accomplish this we use "Concept hierarchies". They are useful to mine at various levels of abstraction.
- **Interesting measures:** Uninterested patterns can be eliminated from the knowledge. This can guide the mining process after discovery in order to evaluate discovery patterns. For Example: In Association mining process support and confidence can be considered as interesting measures so that Association rules which are less than given support and confidence can be eliminated from the discovered patterns.
- **Presentation and Visualization of discovered patterns:** The Form in which the discovered patterns are displayed are referred as presentation and visualization of discovered patterns. For Ex: Patterns can be displayed charts, graphs, tables, decision trees , cubes and rules.



The above Primitives can be explained in detail as follows:

#### 3.1.1 Task – Relevant Data

This is the first primitive to specify the data on which the data mining should be performed. Since data stored in huge amount in databases, or data warehouses. Whole data can not be used for data mining process. The performance issue of generation of patterns depends totally upon database size, it is impractical to mine whole database. Further more, the interested attributes requested by the user only participate in the mining process. So, relevant data only takes part in the mining process.

In relational database, the task relevant data can be collected from the user in the form of a Query with operations involving selection, projection, join, aggregation etc. The data collection process results in as a new relation called *Initial relation.* This initial relation will undergo process of sorting, ordering, and grouping according with the conditions given in the query. The data will be cleaned and transformed before applying data mining analysis. The initial relation may or may not correspond to the physical relation, the task related data for data mining is called as *minable view* which are similar like virtual tables, *views* in relational databases.

In relational database the task relevant data can be specified by providing the following information:

1. Database or datawarehouse name
2. Database tables or datawarehouse cubes
3. Condition for data selection
4. Relevant attributes or dimensions
5. Data grouping criteria.

In data warehouse, typically the data can be stored in multidimensional databases called data cubes. The data cube is like a multidimensional array of relations. The operations that can be performed on data cubes are slicing (extracting data of selected attribute), and dicing (extracting the intersection of several slices).

In a mining query, the user can specify relevant attributes in a particular database or database or data cube. User should specify the condition that should be conceptually higher the database or data warehouse. For Example: In a item relation, different items are available like "Refrigerator", "TV", "Washing machine" etc. The concept hierarchy on item that specifies "home appliances" as a higher concept level composed of the lower level concepts "Refrigerator", "TV" and "Washing machine" can be used as collection of task relevant data. Selection of relevant attributes is a typical task for a user. He may have rough idea about it. In such cases he can use some mechanisms like evaluation of rank attributes that help to give a precise specification.

#### 3.1.2 The Kind of Knowledge

The different kinds of knowledge used in mining process include concept description (discrimination and characterization), classification, association, clustering, prediction and evolution

analysis. To specify the kind of knowledge to be mined, the user should be more specific to provide pattern templates that all patterns should match. These templates guide mining process and explained briefly in the following example.

**Example**

A user is studying a buying habit of Home appliances customers. He choose to mine association rule of the form

$$P(X,customer, W) \wedge Q(X,Y) \Rightarrow buys(X,Z) \quad \text{---------- (3.1)}$$

The above given is a Pattern template, Where X is a key of customer relation; P and Q are predicate variables that can be used to initiate relevant variables. W, Y and Z are the object variables that can take values of respective predicates. The search for association rules which are matched for above template, such as

$$Age(X,"30....50") \wedge income(X,"5000.....50000") \Rightarrow buys(X,"TV")\ [2.2\%,60\%] \text{ ------- (3.2)}$$

$$Occupation(X,"Student") \wedge age(X,"20....29") \Rightarrow buys(X,"DVD")\ [1.4\%,80\%] \text{ -------- (3.3)}$$

The fist rule states that the customers between age 30 and 50 and have monthly income 5000 to 50000 likely to purchase TV, and such cases represent 2.2% in the total number of transactions. The another rule represents that the students whose age is between 20 to 29 are likely to purchase DVD, such cases are 1.4% in the total number of transactions.

### 3.1.3 Background Knowledge

Discovery may be performed with the assistance of relatively strong background knowledge (such as conceptual hierarchy information, etc.) or with little support of background knowledge. The discovery of conceptual hierarchy information itself can be treated as a part of a knowledge discovery process. However, the availability of relatively strong back ground knowledge not only improves the efficiency of a discovery process but also expresses user's preference for guided generalization which may lead to an efficient and desirable generalization process.

The third primitive, the background knowledge, is a set of concept hierarchies or generalization operators which provide corresponding higher level concepts and assist generalization processes. A concept hierarchy defines a sequence of mappings from low level concepts to high level concepts. A concept hierarchy for a location is described as follows in the figure 3.1.

In the figure 3.1 shown the concept of hierarchy is represented as a set of nodes organized as a tree where each node represents a concept. All is the root node, generalized value of the given dimension Location and started with level 0. Country represents level 1, state represents level 2 and city represents level 3.The leaves are represented as the raw values of the dimension.

The concept hierarchy allows raw data to be handled at generalized levels of abstraction. The way of replacing primitive level or raw data(level 3) to next high level in the *Generalization* called Rollup. Generalization makes mining process to take few input / output operations and be more efficient on a larger data set. Here compressing the data is an added advantage.



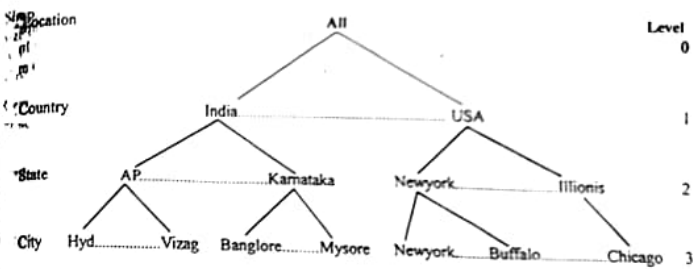


**Figure 3.1** A Concept hierarchy for the dimension Location.

In concept hierarchies, *specialization* occurs when data appears over generalized. The way of replacing higher level to lower level of concept called Drilling down. Eg: Replacing country level by state level concept is drilling down.

There are Four major types of Concept hierarchies. They are:

1. Schema hierarchies
2. Set- grouping hierarchies
3. Operation- derived hierarchies
4. Rule – based hierarchies

- **Schema hierarchy**: A total or partial order among attributes in the database schema. It represents semantic relation ships among the attributes. Schema hierarchy represents data warehouse dimension.

Eg: In the figure 3.2 the schema hierarchy Activity is illustrated. The relation ships among the schemas are represented as

{business , other_activity } < Activity

{low_bussiness_Activity, high_bussiness_activity} < bussiness

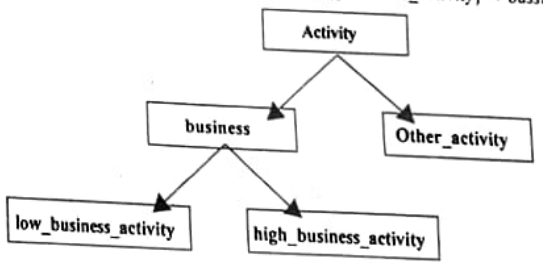


**Figure 3.2** Schema Hierarchies.

- **Set-Grouping Hierarchies:** It Organize values for given attributes or dimensions into groups of constants or range of values. A total or partial order can be defines among groups. Eg: In the figure 3.3, The range of values are represented by the attributes 2 Km .. +∞ Km and 0 m ... 1,999 m . The set of grouping hierarchies are represented as follows:

  level1 : {far, near} < level0: Distance;
  level2: {0, 1999} < level1 : near;
  level2: {2000, + inf} < level1 : far;

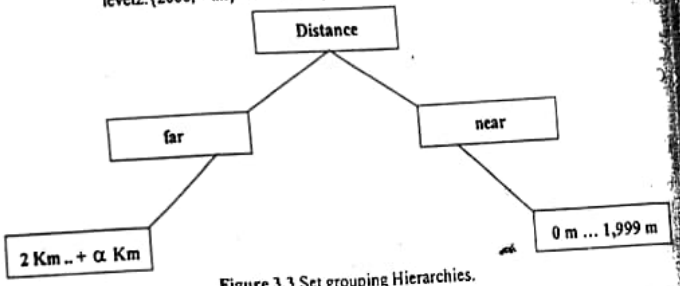


Figure 3.3 Set grouping Hierarchies.

- **Operation-derived hierarchies:** This is based on operations specified by users, experts, or the data mining system. Operations can include decoding of information encoded strings, information extraction from complex data objects and data clustering. For example: An email-is address dmbook@jntu.ap.in gives the partial order "login-name < university < state < country" forming a concept hierarchy. Mathematical and statistical operations such as data clustering and data distribution algorithms, can be used for concept hierarchies.

- **Rule-based Hierarchies:** Rule based Hierarchies occurs when either a whole concept hierarchy or portion of it is defined by a set of rules and is evaluated dynamically based on the current database. For Example: In Home Appliances, categorize the items as low profit items, medium profit items and high profit items. Profit can be calculated as difference between retail price and actual cost of X. If the profit is less than Rs.250 may be defined as lower profit, if items earned profit between Rs.250 and Rs.1000 defined as medium profit and items earned profit greater than Rs.1000 defined as high profit items. Then this rule based hierarchies can be represented as follows:

$$\text{Lowprofit}(X) \Leftarrow \text{price}(X,p1) \wedge \text{cost}(X,p2) \wedge ((p1 - p2) < 250)$$
$$\text{Mediumprofit}(X) \Leftarrow \text{price}(X,p1) \wedge \text{cost}(X,p2) \wedge ((p1 - p2) \geq 250) \wedge ((p1 - p2) < 1000)$$
$$\text{Lowprofit}(X) \Leftarrow \text{price}(X,p1) \wedge \text{cost}(X,p2) \wedge ((p1 - p2) > 1000)$$

### 3.1.4 Interestingness Measures

In discovery of patterns the users need to confine the number of uninteresting patterns returned by the process. This can be achieved by interesting measures that estimate the simplicity, certainty, utility and novelty of patterns. Each measure is associated with threshold that can be controlled by the user.



**Simplicity:** Simplicity can be viewed as functions of pattern structures defined in terms of pattern size in bits or the number of attributes or operators appearing in the pattern. For instance, rule length is a simplicity measure. It can be defined as the number of conjuncts in the rule. Association or classification rules whose length exceeds user-defined threshold can be considered uninteresting.

**Certainty:** The certainty measure for association rules of the form "A⇒B", where A and B are sets of items i.e. confidence. The confidence of "A⇒B" is defined as

$$\text{confidence}(A \Rightarrow B) = \frac{no\_tuples\_containing\_both\_A\_and\_B}{no\_tuples\_containing\_A} \quad \text{---------- (3.4)}$$

A confidence value of 100% indicates that the rule is always correct and such rules are called Exact. The measure of certainty is also referred as reliability or accuracy in classification rules.

**Utility:** The usefulness of a pattern is a factor defining its interestingness. This can be estimated by utility function, support. The support of an association pattern of the form "A⇒B" is defined as

$$\text{Support}(A \Rightarrow B) = \frac{no\_tuples\_containing\_both\_A\_and\_B}{total\_no\_tuples} \quad \text{---------- (3.5)}$$

The association rules that satisfy both user defined minimum confidence threshold and user specified minimum support threshold are referred as strong association rules. Rules with low support likely represent noise or exceptional cases.

**Novelty:** Novel patterns are those patterns which are new, surprising that increases performance to the given pattern set. To detect novelty the strategy should be followed is removing redundant patterns. If the discovered rule can be implied by another rule, that is already in the knowledge base or in the derived rule set, then either rule should be reexamined in order to remove the potential redundancy.

### 3.1.5 Presentation and Visualization of Discovered Patterns

Data mining should display the discovered patterns in effective and efficient manner in different forms such as tables, charts, bar graphs, rules, pie-charts, cross tabs and data cubes as shown in figure 3.4. A user should be able to specify the form of presentation to display the discovered patterns. The generalization concept hierarchies allows to view the patterns in high level concept which may be more understandable to users than rules expressed in terms of raw data. Data mining system should employ concept hierarchies to implement drill down and rollup operations, so that users can see the patterns in different levels of abstraction. The patterns resulted from data cubes which are generalized can be viewed by different perspectives with the help of pivoting (rotating), slicing, and dicing operations.

Rules

age (X, "young") and income (X, "high") => class (X, "A")
age (X, "young") and income (X, "low") => class (X, "B")
age (X, "old") => class (X, "C")

TABLE

| Age | Income | Class | Count |
|---|---|---|---|
| Young | High | A | 1,402 |
| Young | low | B | 1,038 |
| Old | High | C | 786 |
| Old | Low | C | 1,274 |

CROSSTAB

| | Income | | Class | | |
|---|---|---|---|---|---|
| Age | High | Low | A | B | C |
| Young | 1,402 | 1,038 | 1,402 | 1,038 | 0 |
| Old | 786 | 1,374 | 0 | 0 | 2,160 |
| Count | 2,188 | 2,412 | 1,402 | 1,038 | 2,160 |

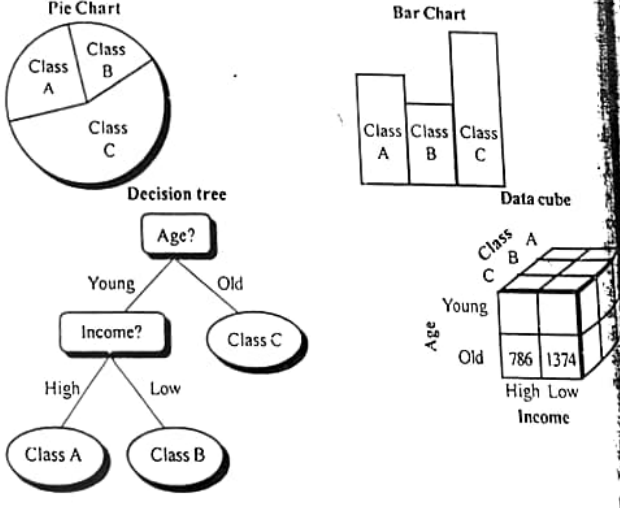


Figure 3.4 Presentation and visualization of various forms of discovered patterns.



## 3.2 DMQL

DMQL has been designed at the Simon Fraser University, Canada. It has been designed to support various rule mining extractions (eg: classification rules, comparison rules, association rules).. The standardization of relational query languages, which occurred at the early stages of relational database development, is leads the relational database fields in successful path. Having a Good query language for data mining may help to standardize the development of data mining systems. The designing of effective query language requires deep understanding of the limitations and mechanisms of various kinds of data mining tasks.

The data mining primitives also plays an important role in the design of a data mining query language. The primitives to be considered are:

- The set of primitives to be mined
- Kinds of knowledge to be mined
- The background language to be used in the discovery process
- The interesting measures and thresholds for pattern evaluation
- The expected representation of visualization and discovered patterns

DMQL is used to query different kinds of knowledge from the relational databases, data warehouses in multiple levels of abstraction.

### Syntax of DMQL for Mining Different Kinds of Rules

DMQL adopts an SQL-like syntax to facilitate high level data mining and natural integration with relational query language, SQL. The DMQL language 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, as shown below.

```
<DMQL> ::=
        use database <database name>
                {use hierarchy <hierarchy name> for <attribute>}
                <rule spec>
                in relevance to <attr_or_agg list>
                from <relation(s)>
                [where <condition>]
                [order by <order list>]
                {with [<kinds of>] threshold = <threshold value>
                [for <attribute(s)>]}
```

In <DMQL>, "use database <database name>" directs the mining task to a specific database <database name>", and the optional statement, "use hierarchy <hierarch> for <attribute>", assigns <hierarchy> to a particular attribute <attribute> (otherwise, a default hierarchy is used). The statement, <rule spec> is the specification of the kind of rules to be discovered. For discovering different kinds of rules, the rule specification should be in different formats, which will be presented in detail in the following sections. The related-to statement, "in relevance to <att_or_agg_ list>", selects a list of

relevant attributes and/or aggregations for generalization. The "from" and "where" clauses, fro
<relation(s)> [where <condition>], form an SQL query to collect the set of relevant data. The "ord
by" clause simply specifies the order of rows to be printed. The "with-threshold" statement specifi
various kinds of thresholds.

### 3.2.1 Syntax for Task Relevant Specification

The clauses include in the task relevant specification are Use database or use datawarehou
From clause, in relevance to clause, order by clause, group by clause and having clause

Ex: use databaseAllelectronics
in relevance to I.name, I.price, C.income,C.age
from customer C, Item I, purchase P, item_sold S
where I.item_*id* = S.item_id and S.trans_id = P.trans_id and P.cust_id = C.cust_
and C.country = "Canada"

group by P.date

### 3.2.2 Syntax for specifying the Kind of Knowledge to be Mined

For rule specification, the following kinds of rules are considered in DMQL.Consider
following database university for all the given below examples.

University database with the following schema.

student(name, sno, status, major, gpa, birth date,birth place, address)
course(cno, title, department)
grading(sno, cno, instructor, semester, grade)

**1. Mining characteristic rules**

The syntax for characteristics rule is :

*mine characteristics* [*as* pattern_name]
*analyze* measure(s)

**Example:** Query (q1) is to find the general characteristics of the graduate students in comput
science in relevance to attributes gpa, birth place and address, for the students born in Canada

(q1) : **use database university**
**in relevance to gpa, birth place, address**
**from student**
**where status = "graduate" and major = "cs" and birth place = "Canada"**
**with noise threshold = 0.05**
**mine characteristics gradstudent**
**analyze count**

This data mining query will first retrieve data from the database using a transformed
query, where the high level constants "Canada" and "graduate" are transformed into low
primitive concepts in the database according to the provided (default) concept hierarchy for
attribute. The algorithm for finding characteristic rules is then executed with the data general
high level for manipulation and presentation. The set of generalized data grouped according



high level concept values of the attributes gpa, birth place and address are presented, associated with the corresponding count (i.e., the count of tuples in the corresponding group in proportion to the total number of tuples). The noise threshold 0.05 means that a generalized tuple taking less than 5% of the total court will not be included in the final result.

**2. Mining discriminant rules**

The syntax for discriminant rule is :

*mine comparison* [*as* pattern_name]
*for* target_class *where* target_condition
{*versus* contrast_class_*i* *where* contrast_condition_*i*}
*analyze* measure(s)

**Example:** Query (q2) is to find the discriminant features to compare graduate students versus undergraduate students in computing science in relevance to attributes gpa, birth place and address, for the students born in Canada.

(q2): **use database university**
**in relevance to, birth place, address, count(*)%**
**from student**
**where major = "cs" and birth place = "Canada"**
**mining comparision as find_discriminant**
**for csgrads where status = "graduate"**
**versus undergrads where status = "undergraduate"**
**analyze count**

This data mining query will first retrieve data into two classes, "cs grads" and "cs undergrads", using a transformed SQL query which maps the high level constants into low level ones. The algorithm for finding discriminant rules is then executed for data mining and result manipulation.

**3. Data classification and mining classification rules**

The syntax for classification rule is as follows:

*mine classification* [*as* pattern_name]
*analyze* classifying_attribute_or_dimension

**Example:** Query (q3) is to classify students according to their gpa's and and their classiffication rules for those majoring in computing science and born in Canada, with the attributes birth place and address in consideration.

(q3): **use database university**
**in relevance to birth place, address**
**from student**
**where major = "cs" and birth place = "Canada"**
**mine classification as stud_mine**
**analyze gpa**

This query will first collect the relevant set of data, and then execute some data classification algorithm, to classify students according to their gpa's and present each class and its associated characteristics.

**4. Mining association rules**

The syntax for Association rule is :

*mine associations* |*as* pattern_name|
|matching <metapattern>|

Example:Query (q4) is to find strong association relationships for those students majoring in computing science and born in Canada, in relevance to the attributes gpa, birth place and address.

(q4) : **use database university**
**in relevance to gpa, birth place, address**
**from student**
**where major ="cs" and birth place = "Canada"**
**mine associations as stud_association**
**with support threshold = 0.05**
**with confidence threshold = 0.7**

This query will first collect the relevant set of data and then execute an association mining algorithm to find a set of interesting association rules. The support and confidence thresholds are specified (otherwise using default values) for mining strong rules.

**5. Mining prediction**

The syntax for mining prediction is as follows:

*mine prediction* |*as* pattern_name|
*analyze* prediction_attribute_or_dimension
{*set* {attribute_or_dimension_*i*= value_*i*}}

### 3.2.3 Specification of Interestingness and Thresholds

A data mining task may need to specify a set of thresholds to control its data mining process including guiding an induction process, constraining search for interesting knowledge, testing the interestingness or significance of the discovered knowledge, etc. This requires the introduction of the fourth set of primitives, a set of data mining thresholds, in DMQL.

Different kinds of rule mining may need to specify different kinds of thresholds which can be categorized into at least three classes, as follows.

**1. Significance threshold:** It indicates that there should exist at least some reasonably substantial evidence (support) of a pattern in the data set in order to warrant its presentation. In mining association rules, this threshold is called the minimum support and the patterns passing this support threshold are called large (or frequent) data items; whereas in mining characteristic rules, it is called noise threshold, and the patterns which cannot pass this threshold are treated as noise.

**2. Rule confidence threshold.** It indicates that the probability of B under condition of A in rule A(B, i.e., Prob(B/A), must pass this threshold to make sure that the implication relationship is reasonably strong.

**3. Rule redundancy threshold.** It indicates that the rules to be presented are not similar to those already there.



The syntax of the threshold specification is as below:

**with |<kinds of>| threshold = <threshold value>**
**|for <attribute(s)>|**

where <kinds of> can be support, confidence, noise, redundancy, etc. Threshold values can be set and modified interactively using similar language primitives.

### 3.2.4 Syntax for Specification of Concept Hierarchies

Concept hierarchy (or lattice) provides useful background knowledge for expressing data mining results in concise, high-level terms. Concept hierarchies can be specified based on database attribute relationships, particular grouping operations, etc. Any given concept hierarchies should be able to be adjusted dynamically based on current data distributions. Moreover, hierarchies for numerical attributes should be able to be constructed automatically based on data distributions.

DMQL handles concept hierarchy specifications as follows.

**1. Hierarchy specification at the schema level.** Concept hierarchies can be specified at the schema level based on database attribute relationships since such relationships may exist in the attribute semantics.

For example: Address(num; street; city; province; country) may indicate that city is more general than street but less general than province and country. Similarly, date(day; month; year) forms naturally a built in concept hierarchy for generalization.

DMQL specifies concept hierarchy at the schema level in the following syntax.

**Define hierarchy <hierarchy_name> for <attribute_or_dimension>**
**on <relation *or* cube> as <hierarchy description>**
**|where <condition>|**

An example of such is shown below.

**Define hierarchy location_hierarchy on address as**
**|street,city province, country|**

**2. Hierarchy specification by set grouping.** Some hierarchical information can be specified by concept grouping which explicitly shows that one group of concepts is at a level lower than another. An example in DMQL is as follows.

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**

**3. Modification of concept hierarchies.** A given hierarchy should be modifiable via some simple statements. One may insert a term as a subordinate concept of a super ordinate one in a hierarchy or delete a term from it.

### 3.2.5 Syntax for Interesting Measure Specification

The user can control the uninteresting patterns in the data mining system by specifying the measures of the pattern interestingness and their thresholds. The measures are confidence, support, noise and novelty. Interestingness measures and thresholds can be specified with the following statement:

**with** [ (interest_measure_name) ] **threshold**=(threshold_value)

Example: In mining association rules, the minimum support and minimum confidence threshold of 5% and 70% respectively can be stated as

**with support threshold** = 5%
**with confidence threshold** = 70%

### 3.2.6. Syntax for pattern presentation & visualization specification:

Data mining query language requires needs syntax that allow users to display the discovered patterns in the form of rules, tables, crosstabs, pie or bar charts, decision trees, cubes, curves or surfaces. The syntax for displaying discovered patterns in DMQL is stated as follows:

**display as** (result_form)

### 3.2.7 Other Data mining languages and the standardization of data mining primitives:

**MSQL**

MSQL (Imielinski and Virmani, 1999) has been designed at the Rutgers University. It extracts rules that are based on descriptors, each descriptor being an expression of the type ($Ai = aij$), where $Ai$ is an attribute and $aij$ is a value or a range of values in the domain of $Ai$. We define a *conjunctset* as the conjunction of an arbitrary number of descriptors such that there are no couple of descriptors built on the same attribute. MSQL extracts prepositional rules of the form $A \Rightarrow B$, where $A$ is a conjunctset and $B$ is a descriptor. As a consequence, only one attribute can appear in the consequent of a rule. Notice that MSQL defines the support of an association rule $A \Rightarrow B$ as the number of tuples containing $A$ in the original table and its confidence as the ratio between the number of tuples containing $A$ et $B$ and the support of the rule. From a practical point of view, MSQL can be seen as an extension of SQL with some primitives tailored for association rule mining (given their semantics of association rules). Specific queries are used to mine rules (inductive queries starting with GetRules) while other queries are post-processing queries over a materialized collection of rules (queries starting with SelectRules). The global syntax of the language for rule extraction is the following one:

```
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, e.g., the kind of descriptors which must appear in rule components, the minimal frequency or confidence of the rules or some mutual exclusion constraints on attributes which can appear in a rule. The USING part enables to discretize numerical values. rulebase name is the name of the object in which rules will be stored. Indeed, using MSQL, the analyst can explicitly materialize a collection of rules and then query it with the following generic statement where <conditions> can specify constraints on the body, the head, the support or the confidence of the rule:

```
SelectRules(rulebase name)
            [where <conditions>]
```

Finally, MSQL provides a few primitives for post-processing. Indeed, it is possible to use Satisfy and Violate clauses to select rules which are supported (or not) in a given table.

**MINE RULE**

MINE RULE (Meo *et al.*, 1998) has been designed at the *University of Torino* and *the Politecnico di Milano*. It is an extension of SQL which is coupled with a relational DBMS. Data can be selected using the full power of SQL. Mined association rules are materialized into relational tables as well. MINE RULE extracts association rule between values of attributes in a relational table. However, it is up to the user to specify the form of the rules to be extracted. More precisely, the user can specify the cardinality of body and head of the desired rules and the attributes on which rule components can be built. An interesting aspect of MINE RULE is that it is possible to work on different levels on grouping during the extraction (in a similar way as the GROUP BY clause of SQL). If there is one level of grouping, rule support will be computed w.r.t. the number of groups in the table. Defining a second level of grouping leads to the definition of clusters (sub-groups). In that case, rules components can be taken in two different clusters, eventually ordered, inside a same group. It is thus possible to extract some elementary sequential patterns (by clustering on a time-related attribute). For instance, grouping purchases by customers and then clustering them by date, we can obtain rules like *Butter* $\wedge$ *Milk* $\Rightarrow$ *Oil* to say that customers who buy first *Butter* and *Milk* tend to buy *Oil* after. Concerning interestingness measures, MINE RULE enables to specify minimal frequency and confidence thresholds. The general syntax of a MINE RULE query for extracting rules is:

```
MINE RULE <TableName> AS
SELECT DISTINCT [<Cardinality>] <Attributes>
                AS BODY,
                [<Cardinality>] <Attributes>
                AS HEAD
                [,SUPPORT] [,CONFIDENCE]
FROM <Table> [WHERE <WhereClause>]
GROUP BY <Attributes> [HAVING <HavingClause>]
[CLUSTER BY <Attributes>
                [HAVING <HavingClause>]]
EXTRACTING RULES WITH
                SUPPORT:<real>, CONFIDENCE:<real>
```

### OLE DB for DM

OLE DB for DM has been designed by Microsoft Corporation (Netz *et al.*,2000). It is an extension of the OLE DB API to access database systems. More precisely, it aims at supporting the communication between the data sources and the solvers that are not necessarily implemented inside the query evaluation system. It can thus work with many different solvers and types of patterns. To support the manipulation of the objects of the API during a KDD process, OLE DB for DM proposes a language as an extension to SQL. The concept of OLE DB for DM relies on the definition of Data Mining Models (DMM), i.e. object that correspond to extraction contexts in KDD. Indeed, whereas the other language proposals made the assumption that the data almost have a suitable format for the extraction, OLE DB for DM considers it is not always the case and let the user defines a virtual object that will have a suitable format for the extraction and that will be populated with the needed data. Once the extraction algorithm has been applied on this DMM, the DMM will become an object containing patterns or models. It will then be possible to query this DMM as a rule base or to use it as a classifier. The global syntax for creating a DMM is the following:

**CREATE MINING MODEL <DMM name>**
**(<columns definition>)**
**USING <algorithm>**
**[(<algorithm parameters>)]**

For each column, it is possible to specify the data type and if it is the target attribute of the model to be learnt in case of classification. Moreover, a column can correspond to a nested table, which is useful when populating the mining model with data taken in tables linked by a one-to-many relationship. For the moment, OLE DB for DM is implemented in the SQL Server 2000 software and it provides only two mining algorithms: one for decision trees and one for clustering. However, the 2005 version of SQL server should provide neural network and association rule extractors. This latter one will enable to define minimal and maximal rule support, minimal confidence, and minimal and maximal sizes of itemsets on which the rules are based.

CRISP-DM ("Cross-Industry Standard Processing for Data Mining") is a standardization effort related to data mining. It focus on technology by addressing the needs of all levels of users in deploying data mining technology to solve business problems. CRISP-DM is an international project that define and validate a data mining process that is generally applicable to diverse sectors. It addresses issues like:

1) Mapping from business issues to data mining problems.
2) Capturing and understanding data
3) Identifying and solving problems within the data
4) Applying data mining techniques
5) Interpreting data mining results within the business context.
6) Deploying and maintaining data mining results.
7) Capturing and transferring expertise to ensure future projects benefit from experience.

## 3.3 Interactive Data Mining and Graphical User Interfaces

Although a data mining task can be specified flexibly using the primitives discussed above, it is difficult to predict the mining results at the time of query submission. Thus, interactive refining of a mining task or mining results becomes essential for effective mining. Interactive refining of a mining



task often requires easy modification of a query condition, thresholds, relevant attributes, selected hierarchies, or letting a hierarchy be dynamically adjusted, etc. Such tasks should be accomplished conveniently by a graphical user interface although they can be specified (but not so conveniently) using DMQL language primitives. For interactive refining of data mining results, one should display the results using rule visualization tools or in different output forms, including generalized relations, projected statistical tables, bar charts, pie charts, curves, surfaces, quantitative rules, etc. Report writers or graphical display softwares may help this process. DMQL provides the following primitives for displaying results in different forms.

**(result displaying) ::= display in (result form)**

where the (result form) could be projected statistical tables, bar charts, curves, etc. when appropriate. Moreover, with the availability of concept hierarchies, knowledge can be expressed at different levels of abstractions. Interactive mining should facilitate the discovered knowledge to be viewed at different concept levels and from different angles. This can be accomplished by transforming data mining results conveniently with "roll-up" and "drill-down" operations. Notice that roll-up can be done by climbing up the concept hierarchy of an attribute or dropping some attribute(s); whereas drill-down by stepping down the concept hierarchy of an attribute or adding some attributes. DMQL provides the following primitives for traversing through different levels of abstractions.

The goal of designing the DMQL language is to provide necessary primitives for data mining engines to work on. However, a data mining user may prefer to use flexible GUIs to interact with a data miner for fruitful and convenient data mining. In current relational technology, SQL provides the "core" language of relational systems on top of which various GUIs have been constructed. Similarly, a data mining query language may serve as a "core language" for data mining system implementations, on top of which various kinds of GUIs should be developed for effective data mining. Based on our experience, a data mining GUI may consist of the following functional components.

1) Data collection, which is an interface for collecting the relevant set of data. Such an interface will be very much like a GUI for construction of relational queries.
2) Presentation of data mining results, which is an interface for displaying data mining results in different forms, including tables, graphics, charts, curves, visualization of data mining results, etc.
3) Manipulation of data mining primitives, which may include dynamic adjustment of various kinds of thresholds, selection, display, and modifications of concept hierarchies, or modification of other mining requests or conditions, etc.
4) Interactive multi-level mining, which may include rollup or drill-down of data mining results on any selected attributes.
5) Other miscellaneous information, which may include on-line manuals, help, indexed search, debugging, and many other interactive graphical facilities. Some data mining GUIs works on both the UNIX and PC platforms in the DBMiner system. By examination of many data mining products or prototypes, it seems difficult to set up a standard on data mining GUIs. However, classification of GUI primitives and exploration of their underlying mechanisms are quite useful at designing a good data mining query language.

## 3.4 Architectures of Data Mining Systems

The data mining software's are developed and become most popular in database technologies at present and also in the future. To apply data mining in diverse applications the architecture and design of data mining is critically important. Since Data mining is still a new area, much of the focus has been on doing mining relational databases. However data mining can also be applied to other kinds of databases such as object oriented, multimedia, distributed, heterogeneous, and legacy databases, architectural support will be increasingly important.

### 3.4.1 Technology Integrated Architecture

One needs architectural support for integrating the different data technologies. Figure 3.5 shows a pyramid like structure as how the various technologies fit with each other. As shown in the figure, communications, network and system level support at the lowest level. Then we have middleware support. This is followed by database management and data warehousing. After that data mining technologies are applies. Finally, we have the decision support systems that take the results of data mining and help the users to make effective decisions. These users could be managers, analysts, programmers and any other user of information systems.

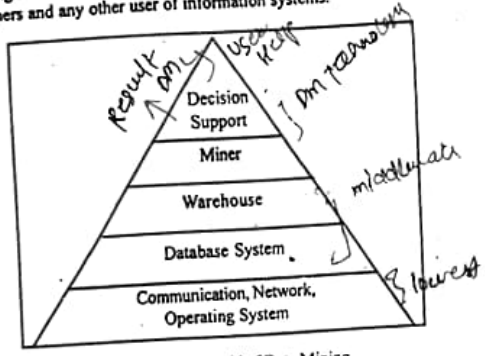


Figure 3.5 Pyramid of Data Mining.

When one builds systems, the various technologies involves may not fit the pyramid identically as shown above. For Example, Some may skip data warehousing stage and directly got to data mining. At present we do not have well defined standards except some of the standards and interface definition languages emerging from various groups such as Object management Group.

The possible ways of integrating or coupling a data mining system and a data base/ datawarehouse are as follows:

- **No coupling :** DM system will not utilize any function of DB/DW system. Mos: data have been stored in DB/DW systems. With no coupling the DM system will need other tools to extract, integrate into an information processing environment. This is a poor design.



- **Loose coupling :** Loosely coupling means that a DM system will utilize some facilities of DB/ DW system, fetching data from repository, performing data mining and storing the results in file or DB or DW. It is difficulty to achieve high scalability but better than No coupling.
- **Semitight coupling:** Semitight coupling means besides linking a DM system to a DB/DW system, efficient implementations of a few essential data mining primitives like sorting, indexing, aggregation, some statistical measure such as sum, count, max, min can be provided in DB/DW system. This design will enhance the DM system.
- **Tight coupling:** Tight coupling means DM system is smoothly integrated with DB/DW system. Data mining queries and functions are optimized based on mining query analysis, data structures, indexing schemes, and query processing methods of DW/DB systems. Here DB, DW and DM integrate together as one information system with multiple functionalities.

Multiple databases are also integrated through some middleware and subsequently form a data warehouse which is then mined. The data mining component is also integrated into this setting so that the databases are mined directly. The middleware can also integrate distributed objects and web information accessed through the web. The three dimensional view of data mining is illustrated in figure 3.6. In one plane we have included warehousing, machine learning and statistical learning. All base technologies are placed in one plane. The third plane has technologies like parallel processing, visualization, metadata management and secure access which are important to carry out data mining.

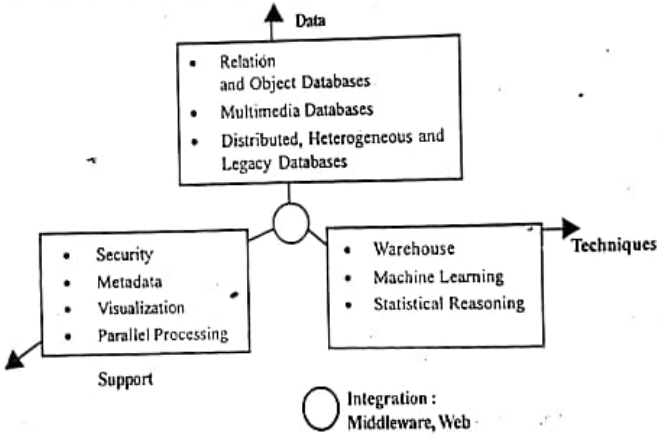


Figure 3.6 Three dimensional view of data mining.

### 3.4.2 Functional Architecture

If the data mining was one of the Functional module of a DBMS then it is called as mining DBMS. One approach to organize Mining DBMS is considering data mining is an extension to the query processor. The Query Processor modules such as the Query Optimizer could be extended to handle data mining. Data mine can have the following components.

- Training sets and learns various strategies
- Data analyzer which analyses the data based on what it has learnt
- Results producing component that does classification, clustering, associations and other tasks.

There is interaction between all the three components. The training component feeds information to the data analyzer and data analyzer feeds information to results producing component

### 3.4.3 System Architecture

System architecture consists of components such as the middleware, database management system and the data warehousing system for data mining. Many of the database systems are based on client-server architectures. That means multi vendor clients communicate with multi vendor servers in a transparent manner. The figure 3.7 illustrates the client server data mining. Middleware also includes de facto standard such as ODBC or distributes object based systems such as OMG'S Common Object Request Broker Architechture(CORBA). Here one possibility to encapsulate database servers as objects and clients to issue appropriate requests and access the servers through an Object Request Broker(ORB). There are three major compoents in CORBA.

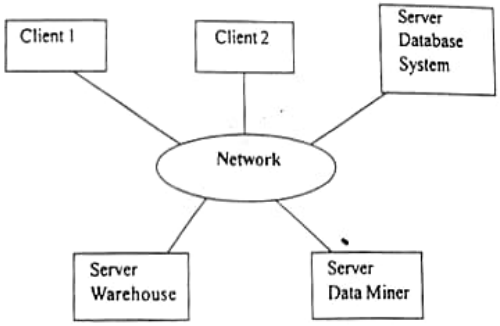


**Figure 3.7** Client server based data mining

- **Object model** which includes most of the constructs
- **ORB** through which clients and servers communicate
- **Interface definition Language(IDL)** specifies interfaces for client server communication



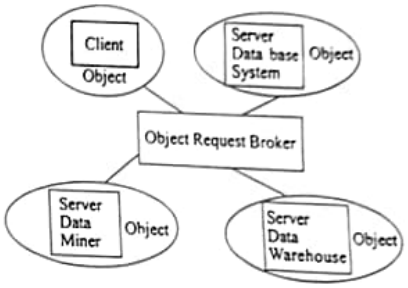


**Figure 3.8** Data Mining through ORB

Figure 3.8 illustrates the client server based data mining through ORB. The data miner could be used as a server, the DBMS could be another server and data warehouse could be third server. The client issues requests to the database system, warehouse and the miner. One could use ORB for data mining. In data mining, one could utilize three tier architecture where data miner is placed in the middle tier. Then data miner could be developed as a collection of objects or components. The advantage is one could purchase components from different vendors and assemble them together to form a system that means here components are reused. For example: Let us assume that the modules are the data source integrator, the data miner, the results pruner and the report generator. Then these modules are encapsulated as objects and used ORBS to integrate different objects. So that we can use plug and play approach to develop data mining tools.

## Summary

Users must be provided with a set of primitives like Task – relevant data, Kinds of knowledge, Background knowledge, Interesting measures, Presentation and Visualization of discovered patterns to be used to communicate with the data mining system.

These primitives are incorporated in a data mining query language. With this the user interaction with the system becomes more flexible, gives foundation for the design of graphical user interface and standardize the data mining industry and practice.

DMQL has been designed to support various rule mining extractions (eg: classification rules, comparison rules, association rules). The data mining primitives also plays an important role in the design of a data mining query language.

Interactive refining of a mining task or mining results becomes essential for effective mining. Interactive refining of a mining task often requires easy modification of a query condition, thresholds, relevant attributes, selected hierarchies, or letting a hierarchy be dynamically adjusted, etc. Such tasks should be accomplished conveniently by a graphical user interface although they can be specified (but not so conveniently) using DMQL language primitives. For interactive refining of data mining results, one should display the results using rule visualization tools or in different output

forms, including generalized relations, projected statistical tables, bar charts, pie charts, curve surfaces, quantitative rules, etc.

The data mining software's are developed and become most popular in database technologies at present and also in the future. To apply data mining in diverse applications the architecture and design of data mining is critically important and architectural support will be increasingly important.

## Exercises

1. Briefly explain the primitives of data mining task ?
2. What is the importance of concept hierarchies in the process of knowledge discovery in data mining ?
3. What is concept hierarchy? Explain the major types of concept hierarchies with an example
4. What do you understand about standardization of data mining primitives?. List out the data mining languages whose effort related to standardization?
5. Explain the various interesting measures required in the discovery of patterns?
6. Consider the University database includes the student : name, address, status (undergraduate and graduate), major, and GPA(cumulative grade point average). Answer the following:
   I) Propose concept hierarchy for the attributes address, status, major and GPA. What type of concept hierarchy is it ?
   II) Define each above proposed concept hierarchy with DMQL syntax.
   III) Write DMQL query to compare the students majoring in science with majoring in arts.
   IV) Write DMQL query to predict student grades in "Computer science" based on student GPA and course instructor.
   V) Write a DMQL to find the characteristics of students who have an Excellent GPA.
   VI) Write DMQL query to find associations between students majoring in arts and residing at Canada in relevance to GPA.
7. Explain the system architecture of data mining?
8. What are the functional components of Data Mining Graphical user interface?
9. Describe the architectures for the integration of a data mining system with a database or datawarehouse system?
10. Write a DMQL syntax for defining concept hierarchy for the schema Date (date, month, quarter, year)?
11. Write a DMQL definition for item_hierarchy which includes two relations as follows:
    Item(item_id,brand, type, place_made,supplier)
    Supplier(name, type,headquareter_location, owner, size,assets, revenue)



12. Explain briefly how MSQL helps in extracting the prepositional rules? Give the global sytax of MSQL Language.
13. Explain the specification of comcept hirarchies along with syntax?
14. What is CRISP-DM? How it addresses the issues in the process of data mining?
15. Explain how MINE RULE Works on different levels of grouping during the extraction?

## Multiple choice questions

1. In Data Warehouse the data stored in databases as
   a. One dimension
   b. Two dimensions
   c. **Multiple dimensions**
   d. No dimension

2. Which operations can be performed on Data cubes?
   a. slicing
   b. dicing
   c. Roll up
   d. Roll down
   i) Only a ii) Only b iii) none of the above **iv) All of the Above**

3. Which of the following is not a data mining primitive?
   a. Task-relevant data
   b. Interesting measures
   c. Domain knowledge
   d. **Data mining query language**

4. Which concept hierarchy represents the semantic relationship between the attributes?
   a. Operation- derived hierarchies
   b. **Schema hierarchies**
   c. Rule based hierarchies
   d. Set- grouping hierarchies

5. Which Language extracts association rule between values of attributes in a relational table?
   a. **DMQL**
   b. MSQL
   c. MineRule
   d. OLEDB for DM

6. In the architecture of Data mining, The Data miner level comes after
   a. Data base System
   b. Decision support System
   **c. Data Warehouse**
   d. Communications, network and system level

7. which of the following is not a interesting measure in the discovery of patterns?
   a. Novelty
   b. Utility
   c. Certainity
   **d. Classify**

8. The possible ways of integrating or coupling a data mining system and a database datawarehouse are
   a. Tight Coupling
   b. Loose Coupling
   c. Semitight Coupling
   d. No coupling
   i) Only a ii) Only b iii) Only c **iv) All of the above**

9. In three tier architecture of data mining, The data miner belongs to
   a. Client tier
   **b. middle tier**
   c. Database tier

10. Which of the following can be specified based on database attribute relationships particularly grouping operations?
    **a) Concept hierarchies**
    b) Support
    c) confidence
    d) noise

11. The goal of designing DMQL is to
    a) Collect the data
    **b) Provide necessary primitives for data mining engine**
    c) Present data mining results in GUI
    d) Create Data cube

12. One approach to organize Mining DBMS is considering Data mining is an extension to
    a) SQL
    **b) Query processor**
    c) PLSQL
    d) Databases



13. The clients and server communicate through
    a) Data miner
    b) Database System
    **c) Object Request Browser**
    d) Data warehouse

14. Which of the following supports minimum threshold in association rules?
    a) Rule confidence threshold
    b) Rule redundancy threshold
    **c) Significance threshold**

15. The kind of knowledge used in mining process include
    a) Association
    b) Classification
    c) Clustering
    d) Prediction
    i) Only A ii) Only B **iii) All of the above** iv) none of the above

16. In which Presentation the operations like pivoting, slicing, and dicing operations can be viewed by different perspectives?
    a) Pie chart
    b) Bar chart
    **c) Data cube**
    d) Decision tree

17. which of the following are generalized levels of abstraction in concept hierarchy?
    a) Generalization
    b) Specialization
    **c) Both a and b**
    d) Neither a nor b

## Object type Questions

18. What is data cube?
19. In Relational Database, the data collection process results in as a new relation called—
20. What are the different kinds of knowledge used in data mining process ?
21. Which primitive is a set of concept hierarchies or generalization of operators?
22. What is concept Hierarchy?
23. What is the role of Generalization in mining process?
24. The way of replacing higher level to lower level of concept is called _____.
25. List out the types of concept hierarchies.

26. Define confidence?
27. The measure of certainty is also referred as _____.
28. The support of an association pattern is represented as_____.
29. How can you detect novelty in the given pattern set?
30. Which threshold indicates that the probability of B under condition of A in rule A | B.
31. Which thresholds are to be specified in mining association rules?
32. How the DMQL can handle the specification of concept hierarchy?
33. Which Language extracts elementary sequential patterns by clustering on a time-related attribute?
34. CRISP-DM stands for _____.
35. Interactive mining facilitates the discovered knowledge to be viewed at different concept levels and from different angles with _____and _____ operations.
36. Which coupling integerates Data bases,Data warehouse and Dataminig together as one information system with multiple functionalities?
37. What are the three major components in CORBA?
38. What is CORBA?
39. Which integrates distributed objects and web information accessed through the web?
40. Which Language is core language for Data mining?

## References


R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996.

Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Proc. CL 2000*, volume 1861 of *LNCS*, pages 972–986. Springer-Verlag, 2000.

M. Botta, J. F. Boulicaut, C. Masson, and R. Meo. Query languages supporting descriptive rule mining: a comparative study. In *Database Technologies for Data Mining - Discovering Knowledge with Inductive Queries*, volume 2682 of *LNCS*, pages 27–54. Springer-Verlag, 2004.

J. F. Boulicaut. Inductive databases and multiple uses of frequent itemsets: the cInQ approach. In *Database Technologies for Data Mining – Discovering Knowledge with Inductive Queries*, volume 2682 of *LNCS*, pages 3–26. Springer-Verlag, 2004.

J. F. Boulicaut and B. Jeudy. Constraint-based data mining. In *Data Mining and Knowledge Discovery Handbook*. Kluwer, 2005. (this volume).





T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *Proc. PKDD'02*, volume 2431 of *LNCS*, pages 74–85. Springer-Verlag, 2002.

B. Catania, A. Maddalena, M. Mazza, E. Bertino, and S. Rizzi. A framework for data mining pattern management. In *Proc. PKDD'04*, volume 3202 of *LNAI*, pages 87–98. Springer-Verlag, 2004.

L. De Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69–77, 2003.

F. Giannotti and G. Manco. Querying inductive databases via logic-based user defined aggregates. In *Proc. PKDD'99*, volume 1704 of *LNCS*, pages 125– 135. Springer-Verlag, 1999.

J.Han , M.Kamber .Data Mining concepts and techniques. Academic Press, 2001.

J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. DMQL: a data mining query language for relational databases. In R. Ng, editor, *Proc. ACM SIGMOD Workshop DMKD'96*, Montreal, Canada, 1996.

T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, November 1996.

T. Imielinski and A. Virmani. MSQL: A query langugage for database mining. *Data Mining and Knowledge Discovery*, 3(4):373–408, 1999.

T. Imielinski, A. Virmani, and A. Abdulghani. Major-application programming interface for database mining. *Data Mining and Knowledge Discovery*, 3(4):347–372, 1999.

B. Jeudy and J.-F. Boulicaut. Optimization of association rule mining queries. *Intelligent Data Analysis*, 6(4):341–357, 2002.