The Data Mining Query Language DMQL was proposed by Han, Fu, Wang, et al. for the DBMiner data mining system. The Data Mining Query Language is actually based on the Structured Query Language SQL.

Data Mining Query Languages can be designed to support ad hoc and interactive data mining. This DMQL provides commands for specifying primitives. The 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.

Syntax for Task-Relevant Data Specification

Here is the syntax of DMQL for specifying task-relevant data −

use database database_name

or

use data warehouse data_warehouse_name
in relevance to att_or_dim_list
from relation(s)/cube(s) [where condition]
order by order_list
group by grouping_list

Syntax for Specifying the Kind of Knowledge

Here we will discuss the syntax for Characterization, Discrimination, Association, Classification, and Prediction.

Characterization

The syntax for characterization is −

mine characteristics [as pattern_name]
analyze {measure(s)}

The analyze clause, specifies aggregate measures, such as count, sum, or count%. For example −

| Description describing customer purchasing habits. |
| mine characteristics as customerPurchasing |
| analyze count% |

Discrimination

The syntax for Discrimination is −

mine comparison [as {pattern_name}]
For {target_class} where {target_condition}
{versus {contrast_class_i}}
where {contrast_condition_i} |
analyze {measure(s)}

For example, a user may define big spenders as customers who purchase items that cost $100 or more on an average; and budgetspenders as customers who purchase items at less than $100 on an average. The mining of discriminant descriptions for customers from each of these categories can be specified in the DMQL as −

mine comparison as purchaseGroups
for bigSpenders where avg(I.price) ≥$100
versus budgetSpenders where \( \text{avg}(I\text{.price}) < 100 \)

**Association**

The syntax for Association is—

```sql
mine associations [ as \{pattern_name\} ]
\{matching \{metapattern\} \}
```

For Example –

```sql
mine associations as buyingHabits
matching \( P(X:customer,W) \wedge Q(X,Y) \geq \text{buys}(X,Z) \)
```

where \( X \) is key of customer relation; \( P \) and \( Q \) are predicate variables; and \( W, Y, \) and \( Z \) are object variables.

**Classification**

The syntax for Classification is –

```sql
mine classification [as \{pattern_name\}]
analyze classifying_attribute_or_dimension
```

For example, to mine patterns, classifying customer credit rating where the classes are determined by the attribute credit_rating, and mine classification is determined as classifyCustomerCreditRating.

```sql
analyze credit_rating
```

**Prediction**

The syntax for prediction is –

```sql
mine prediction [as \{pattern_name\}]
analyze prediction_attribute_or_dimension
\{set \{attribute_or_dimension_i= value_i\}\}
```

**Syntax for Concept Hierarchy Specification**

To specify concept hierarchies, use the following syntax –

```sql
use hierarchy \{hierarchy\} for \{attribute_or_dimension\}
```

We use different syntaxes to define different types of hierarchies such as–

- **Schema hierarchies**
  ```sql
define hierarchy time_hierarchy on date as [date, month, quarter, year]
```

- **Set-grouping hierarchies**
  ```sql
define hierarchy age_hierarchy for age on customer as
  level1: \{young, middle_aged, senior\} < level0: all
  level2: \{20, ..., 39\} < level1: young
  level3: \{40, ..., 59\} < level1: middle_aged
  level4: \{60, ..., 89\} < level1: senior
```

- **Operation-derived hierarchies**
  ```sql
define hierarchy age_hierarchy for age on customer as
  \{age_category(1), ..., age_category(5)\}
  := \text{cluster}(default, age, 5) < all(age)
```
- rule-based hierarchies

```plaintext
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
```

**Syntax for Interestingness Measures Specification**

Interestingness measures and thresholds can be specified by the user with the statement –

```plaintext
with <interest_measure_name> threshold = threshold_value
```

For Example –

```plaintext
with support threshold = 0.05
with confidence threshold = 0.7
```

**Syntax for Pattern Presentation and Visualization Specification**

We have a syntax, which allows users to specify the display of discovered patterns in one or more forms.

```plaintext
display as <result_form>
```

For Example –

```plaintext
display as table
```

**Full Specification of DMQL**

As a market manager of a company, you would like to characterize the buying habits of customers who can purchase items priced at no less than $100; with respect to the customer’s age, type of item purchased, and the place where the item was purchased. You would like to know the percentage of customers having that characteristic. In particular, you are only interested in purchases made in Canada, and paid with an American Express credit card. You would like to view the resulting descriptions in the form of a table.

```plaintext
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, purchase P, items_sold S, branch B
where I.item_ID = S.item_ID and P.cust_ID = C.cust_ID and
P.method_paid = "AmEx" and B.address = "Canada" and I.price ≥ 100
with noise threshold = 5%
display as table
```

**Data Mining Languages Standardization**

Standardizing the Data Mining Languages will serve the following purposes –

- Helps systematic development of data mining solutions.
- Improves interoperability among multiple data mining systems and functions.
- Promotes education and rapid learning.
- Promotes the use of data mining systems in industry and society.