## Chapter 1.4 Review of Database Model and Language

***Content:***

In this chapter, do some simple review of SQL and Relational Model.

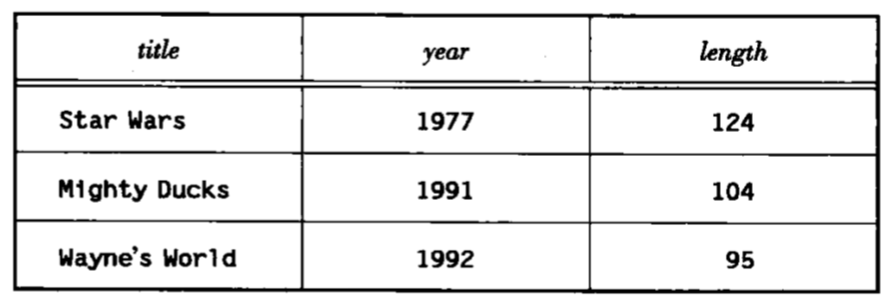
### Chapter 1.4.1 Review of Relational Model

***Definition:***

1. Relation is the collection of tuple, while tuple is the list of value.
2. All tuples in Relation has the same number of fields, while fields in different tuples has the same type.
3. Show tuple of one Relation is to show each column and row of the Relation.
4. The head of column is called attribute, it represents the meaning of each field.
5. The name of Relation, the name of Attribute and the type of property are all called as the Model of Relation.

***Example:***

The Relation Movie, it may contains the tuples below:



*The Model of Relation is:*

Movie(title, year, length)

1. The name of Relation is ‘Movie’, the name of Attribute are ‘title’, ‘year’ and ‘length’, assume that their type are string, integer and integer.
2. Each line of from the table is one tuple. The first line means that the movie Star Wars is produced in 1977 and the length of the movie is 124 minutes.

***Conclusion:***

Database Model is the collection of Relation Model. The below is the normally used Relation:

Movie(title, year, length, studioName)

MovieStar(name, address, gender, birthdate)

StarsIn(title, year, starName)

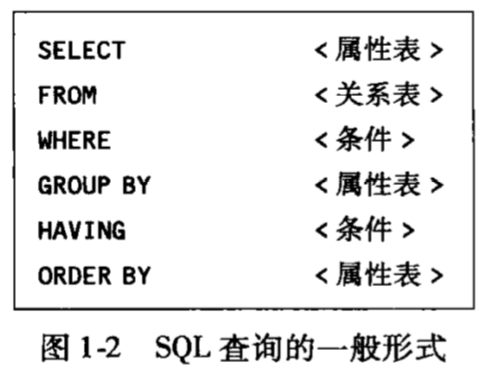
Studio(name, address)

### Chapter 1.4.2 Review SQL

***Definition:***

Database Language has a lot of ability, including the statements that are used to query and modify the database.

Database Modification includes *INSERT*, *DELETE* and *UPDATE*. Query usually uses the *select - from - where* to proceed.

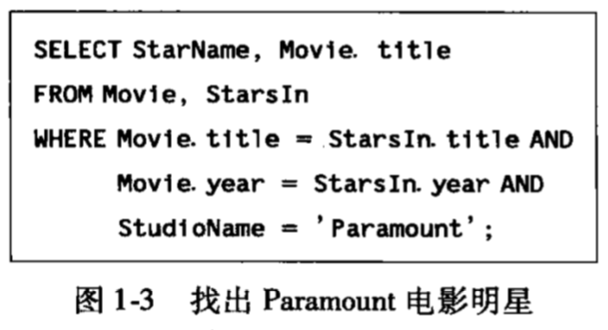


***Query Procedure:***

1. Get all tuples from Relations by statement *FROM <Relation Table>* and all tuple combinations.
2. Remove all tuples that do not satisfy statement *WHERE <Condition>*.
3. If exists the statement *GROUP BY <Attribute>*, then using the *<Attribute Name>* to group the left tuples.
4. If exists the statement *HAVING <Condition>*, then check each group based on <Condition>, remove the tuples that violates condition.
5. According to the statement *SELECT <Attribute>*, calculate tuples based on the assigned attributes and aggregations.
6. Sort the result tuples according to the statement *ORDER BY*.

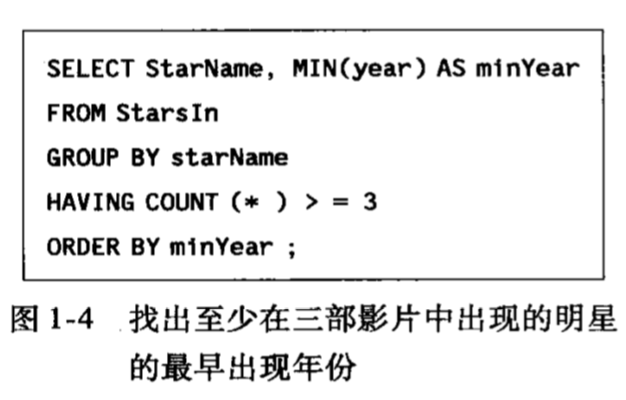
***Example:***

The query below is the simple SQL query and only includes three statements. It queries that all movies’ name and their corresponding movie stars from the *Paramount*.



***Example:***

The query below is one more complicated one. It queries that all stars’ name who has attended for at least three movies and the earliest year when the star attends the movie. The final result needs to be sorted by the earliest year that these movie stars attend the movie.

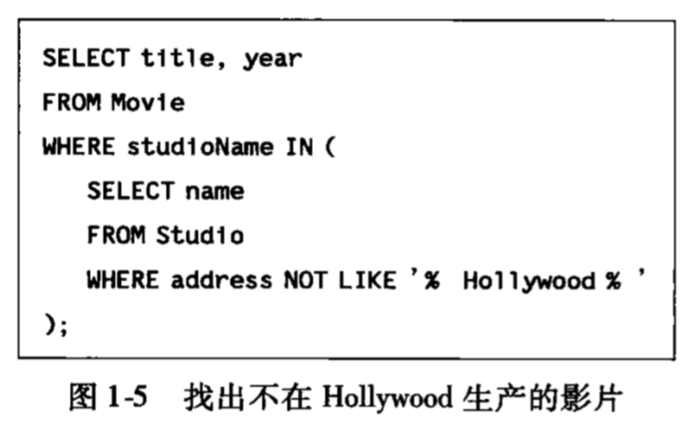


***Sub - Query:***

One of the most powerful feature is that SQL provides Sub - Query in *WHERE*, *FROM* or *HAVING*. The sub - query is also a whole *select - from - where* statement.

***Example:***

Below is one example that contains Sub - Query.



*The Sub - Query:*

*SELECT name*

*FROM Studio*

*WHERE address NOT LIKE ‘%Hollywood%’*

Sub - Query is to query all movies that do not produced by ‘*Hollywood’*.

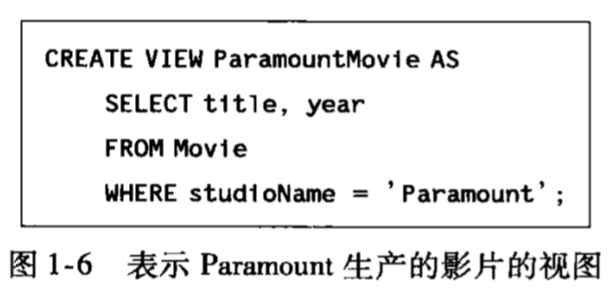
The whole Query is used to query all movies that do not produced by ‘*Hollywood’*, output all movie name and year.

***View:***

Another important point of SQL is to define view which is not saved into the storage but when we need this Relation, then construct based on all current Relations.

***Example:***

The view is to depict all movies produced in Paramount and output movie name and title of the movie.



The view ParamountMovie is defined as one part of the database module, but at this time, we do not calculate all tuples. If we need all tuples, then just merge the view into the Query and construct it’s tuples. *(Actually, all these tuples are not really stored in the database.)*