COMP2350/6350 Database Systems

Practical – Week 5

Objectives:

- To understand the use of LOGICAL OPERATORS (AND, OR, NOT) in a SQL statement.
- To understand the use of DISTINCT in a SELECT clause.
- To understand the use of BETWEEN in a WHERE clause.
- To understand the use of LIKE in a WHERE clause.
- To understand MySQL CROSS JOIN (cartesian product) operations for multi-table queries.
- To formulate and execute queries based on these operators and keywords.

Task Specifications:

Task 1: Downloading and Executing Sample Database SQL Script to create and populate tables

- SQL Script of the sample database that will be used in this practical class is uploaded in iLearn. The data model of this SQL script can be found at the end of this document.
- Please download the SQL script and Execute it using MySQL Workbench.
- To execute the script, click **Query > Execute (All or selection)**.
- Ensure that each statement in the script has been executed correctly.

  You may want to use: `show tables` or `describe <table_name>` or `select * from <table_name>` commands for this purpose.

Task 2: Understanding the use of AND, OR, NOT in a WHERE clause

- The general syntax of a simple SQL statement is:
  ```
  SELECT <list of attributes>
  FROM <table_name>
  WHERE <predicate>;
  ```
- The WHERE clause is optional and it contains the predicate (condition) to find required tuples.
- **Example 1:** Find customers who live in Paris **and** has credit limit more than 75000.
  ```
  SELECT * FROM customers WHERE city = 'Paris' AND creditLimit >= 75000;
  ```
  (2 rows will be returned)
- **Example 2:** Find customers who live in Paris **or** has credit limit more than 75000.
  ```
  SELECT * FROM customers WHERE city = 'Paris' OR creditLimit >= 75000;
  ```
  (63 rows will be returned)
- **Example 3:** Find customers who live in any city except Paris.
  ```
  SELECT * FROM customers WHERE NOT city = 'Paris';
  ```
  (119 rows will be returned)

Task 3: Understanding the use of DISTINCT in a SELECT clause

- DISTINCT in a SELECT clause eliminates duplicate rows in a result set.
- The general syntax of using the DISTINCT is as follows.
  ```
  SELECT DISTINCT <list of attributes> FROM <table_name> WHERE <predicate>;
  ```
- **Example 4:** List the unique last names of employees.
  ```
  SELECT DISTINCT lastname FROM employees;
  ```
  (19 rows will be returned)

Task 4: Understanding the use of BETWEEN operator in a WHERE clause

- BETWEEN is used to determine whether a value is in a given range of values or not.
- **Example 5:** Find product code, product name of those whose buy prices are within the ranges of 90 and 100 (both inclusive).
  ```
  SELECT productCode, productName, buyPrice FROM products where buyPrice BETWEEN 90 and 100;
  ```
  (7 rows will be returned)

Task 5: Understanding the use of LIKE operator in a WHERE clause

- LIKE operator is used to select data based on patterns. Therefore, it is often used in the WHERE clause of SELECT statement.
- MySQL provides two wildcard characters for using with the LIKE operator.

  - The percentage (**%**) wildcard allows you to match any string of zero or more characters.
  - The underscore (**_**) wildcard allows you to match any single character.

- **Example 6:** List the employee number, employee first name and last name whose first name starts with character 'a'.
  ```
  SELECT employeenumber, firstname, lastname from employees where firstname LIKE 'a%';
  ```
  (2 rows will be returned)
- **Example 7:** List the employee number, employee first name and last name whose last name ends with 'on'.
  ```
  SELECT employeenumber, firstname, lastname from employees where lastname LIKE '%on';
  ```
  (4 rows will be returned)

- **Example 8:** List the employee number, employee first name and last name of those employees whose first name contains a substring of length 3 where first character is a 'T' and last character is an 'M'.
  ```
  SELECT employeenumber, firstname, lastname from employees where firstname LIKE '%t_m%';
  ```

  (1 row will be returned)

- **Please note that patterns are not case sensitive.**

Task 6: Understanding CROSS JOIN operation for Multi-table Queries

- CROSS JOIN operation returns the cartesian product of rows from the joined tables.
- **Example 9:** Shows cartesian product of employees and offices.
- ```
  SELECT * FROM employees CROSS JOIN offices;
  ```

  (161 rows will be returned)

- **CROSS JOIN may return some tuples where the values of common attribute(s) of two relations might not match. Please look at the result of the previous query. Can you find any such tuples?**
- Using a WHERE clause with appropriate condition can help us to remove those irrelevant tuples.

  Example 10: Find employees first name, last name, city and country of their offices.
  ```
  SELECT firstName, lastName, city, country from employees cross
  join offices where employees.officeCode = offices.officeCode;
  ```

  (23 rows will be returned)

- You may also rewrite the previous query as given below.
  ```
  SELECT firstName, lastName, city, country from employees,
  offices where employees.officeCode = offices.officeCode;
  ```
  CROSS JOIN can be replaced by a single comma (,).

Task 7: Writing and Executing Some Queries

Based on the previously introduced keywords and operators, your job is to write SQL statements for the following queries. Total number of tuples in the result set of each query is mentioned for your convenience.

- Find product code, product name of those whose buy prices are not within the ranges of 90 and 100.
  (Hints: use NOT and BETWEEN) [103 rows will be returned]
- List all product name and product vendor of those whose name contains the substring 'Ford' or vendor is 'Second Gear Diecast'.
  (Hints: use LIKE and OR) [22 rows will be returned]
- List the name of unique countries where an office is located.
  (Hints: use DISTINCT) [5 rows will be returned]
- Find order number, order date and customer name of those whose customers are based in Australia.
  (Hints: use CROSS JOIN with WHERE clause) [19 rows will be returned]
- Find product name, vendor name of products which are shipped in 2005.
  (Hints: use CROSS JOIN with WHERE clause) [444 rows will be returned]
  ```

Appendix: Data Model of the Sample Database

This database[1] is a retailer of scale models of classic cars database. It contains typical business data such as customers, products, sales orders, sales order line items, etc. The sample database schema consists of the following tables:

- Customers: stores customer's data.
- Products: stores a list of scale model cars.
- ProductLines: stores a list of product line categories.
- Orders: stores sales orders placed by customers.
- OrderDetails: stores sales order line items for each sales order.
- Payments: stores payments made by customers based on their accounts.
- Employees: stores all employee information as well as the organization structure such as who reports to whom.
- Offices: stores sales office data.

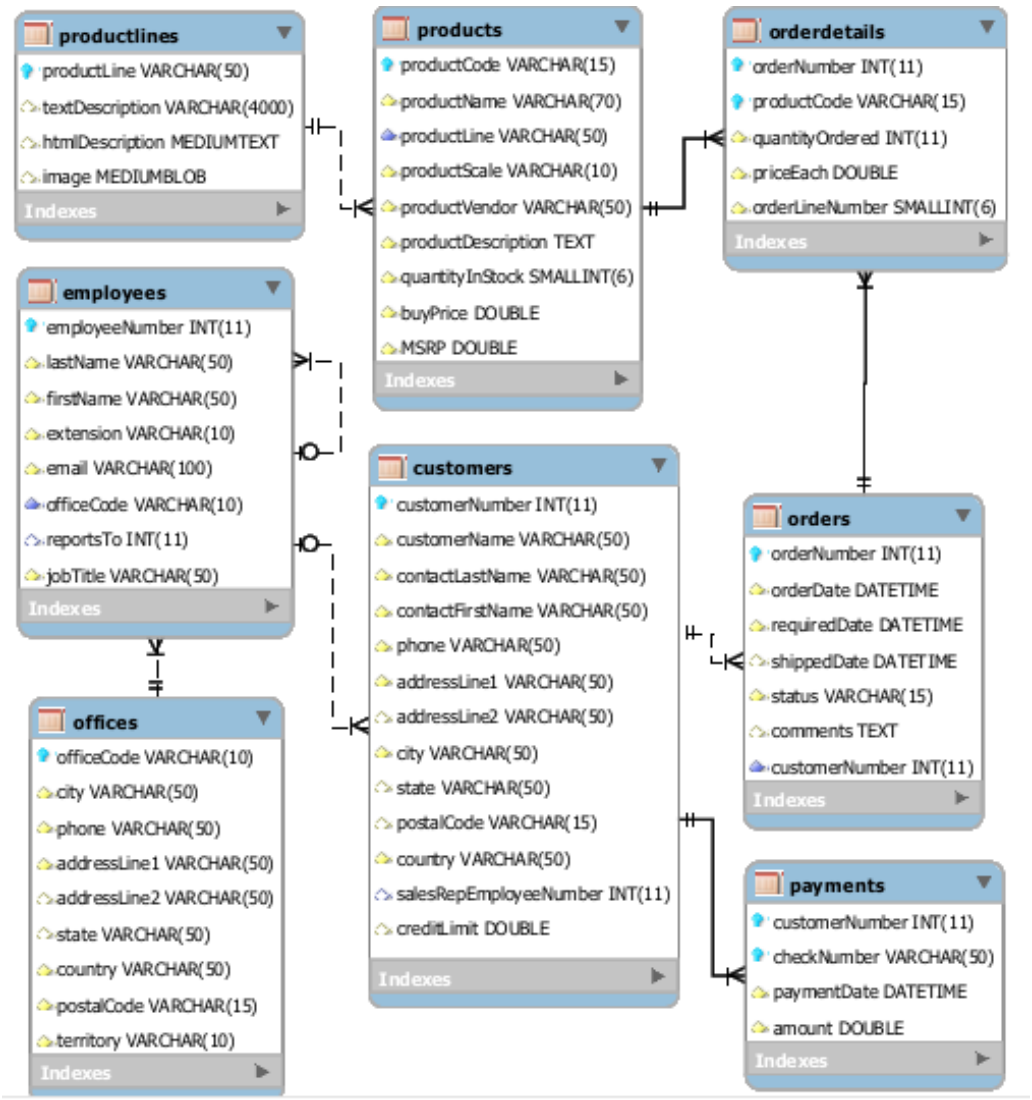

Figure 1: Data Model of Sample Database

[1] http://www.mysqltutorial.org/mysql-sample-database.aspx