Comp2350/6350 Database Systems

Practical (Week 7)

Objectives:

- To understand the use and difference between WHERE and HAVING in GROUP BY.
- To understand the use of SET OPERATIONS (UNION, INTERSECT and EXCEPT) in MySQL.
- To understand the use of ALL and ANY/SOME operators.
- To understand the use of NATURAL JOIN, LEFT JOIN and RIGHT JOIN.
- To understand the use of Subquery.
- To formulate and execute queries based on these operators and keywords.

Task Specifications:

Task 0: Ensuring the presence of tables with data in the database

- **You are going to work on the same database that you have created in Week 5. Ensure all tables with data exist in your database.**
- In case of any problems, please refer Task 1 of Week 5 in the appropriate document. You will find the SQL script inside Practical folder under Week 5.

Task 1: Understanding the use of WHERE and HAVING in GROUP BY

- GROUP BY clause was introduced in Week 6. **Using a WHERE clause before a GROUP BY clause selects appropriate tuples before forming the groups whereas a HAVING clause after a GROUP By clause selects tuples after forming the groups.**
- ***Example 1:*** Find the order numbers and the number of items sold only for products 'S18_1749' and 'S18_2248' per order.

```
SELECT ordernumber, sum(quantityordered)
FROM orderdetails
WHERE productcode in ('S18_1749','S18_2248')
GROUP BY ordernumber;
```

- ***Example 2:*** Find the order numbers and the number of items sold per order. Do not include order numbers having less than 500 items sold.

```
SELECT ordernumber, sum(quantityordered)
FROM orderdetails
GROUP BY ordernumber
HAVING sum(quantityOrdered) >= 500;
```

Task 2: Understanding the use of SET OPERATIONS

- **UNION** operator can combine two or more result sets from multiple SELECT statements into a single result set.

- ***Example 3:*** Find the cities where an office or a customer (or both) is located.
```
SELECT city FROM offices
UNION
SELECT city FROM customers;
```
- **MySQL does not support INERSECT and EXCEPT operator unlike standard SQL.** But the result of these operators can be achieved using other operators and subquery.

Task 3: Understanding the use of ALL and ANY/SOME operators

- **ALL** operator compares a value with a given set of values and returns true if the value satisfies the condition for all values in the given set.
- ***Example 4:*** Find the customer name that has the highest credit limit among all customers.
```
SELECT customername, creditlimit
FROM customers
WHERE creditlimit >= ALL (select creditlimit FROM customers);
```
- **ANY/SOME operator compares a value with a given set of values and returns true if the value satisfies the condition for at least some values in the given set.**
- ***Example 5:*** Find the customer names those do not have the highest credit limit..
```
SELECT customername, creditlimit
FROM customers
WHERE creditlimit <= ANY (select creditlimit FROM customers);
```
- **You can replace ANY with SOME. MySQL supports both.**
- **What change you might expect in the result set if you replace <= with < in Example 5?**

Task 4: Understanding the use of NATURAL JOIN and OUTER JOINS

- **CROSS JOIN (or only JOIN) was introduced in Week 5. You must put an equality condition on common attributes while performing JOIN to discard/remove invalid matchings of tuples from two relations.**
- **NATURAL JOIN puts the equality condition implicitly, without mentioning it in an explicit manner. However, in this case, the common attribute name must be same in both tables.**
- ***Example 6:*** List product names and product lines along with the product line text description only for those products manufactured by 'Second Gear Diecast'.
```
SELECT productname, productline, textDescription
FROM products natural join productlines
WHERE productvendor = 'Second Gear Diecast';
```
- **MySQL also supports outer joins – LEFT JOIN, RIGHT JOIN and FULL JOIN. See Textbook for details.**

Task 5: Understanding the use of Subquery

- **When a query (SQL Statement) contains another query (SQL Statement), the inner query is known as Subquery.**
- **Both Example 4 and Example 5 contain a subquery.**
- EXISTS/NOT EXISTS operator is also used to form a subquery.

- **EXISTS** returns true if the subquery has a non-empty result set.
- **NOT EXISTS** returns true if the subquery has an empty result set.
- ***Example 7:*** Find order number, customer name and country for those which are placed by customers in Australia.

```
SELECT o1.ordernumber, customerName, country
FROM orders o1, customers c1
WHERE o1.customerNumber = c1.customerNumber
      and EXISTS ( SELECT * FROM orders o2, customers c
                   WHERE o1.ordernumber = o2.ordernumber
                     and o1.customerNumber = c.customerNumber
                     and c.country = 'Australia');
```

- **Of course, query in Example 7 can be written in a simpler way.**

Task 6: Writing and Executing Some Queries

Based on the previously introduced keywords and operators, your job is to write SQL statements for the following queries. (*) marked queries are a bit challenging. A few tuples from the result set is given for your convenience. In case of any doubt, please verify the correctness of the SQL statement with your tutor.

i) (*) Find the total valuation/amount of all orders placed by each customer located in 'USA' and 'Australia'. The result set must include customer name, customer country and the total valuation of all orders by that customer. Sort the result set based on total valuation of orders in descending order. You must use HAVING clause in your statement.

customername	country	Total Valuation of Orders
Mini Gifts Distributors Ltd.	USA	591827.34
Australian Collectors. Co.	Australia	180585.07
Muscle Machine Inc	USA	177913.95
Land of Toys Inc.	USA	149085.15

ii) Rewrite the query as given in Example 2 without using HAVING clause.

ordernumber	total_quantity
10103	541
10105	545
10106	675
10108	561

iii) Find the cities where both offices and customers are located. The result set must include the name of the cities and their corresponding office codes.

officecode	city
1	San Francisco
2	Boston
3	NYC
4	Paris
7	London

iv) (*) Find the product name which has the lowest quantity in stock. You must not use ALL keyword.

productname	quantityInStock
1960 BSA Gold Star DBD34	15

v) Find the product code, product name and quantity in stock of those products which have higher quantity in the stock than the average quantity in stock of all products. Only include the top five products based on the quantity in stock and display them in descending order.

productcode	productname	quantityinstock
S12_2823	2002 Suzuki XREO	9997
S18_1984	1995 Honda Civic	9772
S700_2466	America West Airlines B757-200	9653
S24_3432	2002 Chevy Corvette	9446
S18_2325	1932 Model A Ford J-Coupe	9354

vi) Rewrite the query as given in Example 7 using NATURAL JOIN.

ordernumber	customerName	country
10120	Australian Collectors, Co.	Australia
10125	Australian Collectors, Co.	Australia
10223	Australian Collectors, Co.	Australia
10342	Australian Collectors, Co.	Australia

vii) (*) Find the products which are never ordered. Use NOT EXISTS operator.

productcode	productname
S18_3233	1985 Toyota Supra

viii) Find the employees (employee number, first name and last name) who supervises other employees.

employeenumber	firstname	lastname
1002	Diane	Murphy
1056	Mary	Patterson
1088	William	Patterson
1102	Gerard	Bondur
1143	Anthony	Bow
1621	Mami	Nishi

(*) Can you find the boss here? Assume that the boss is not supervised by anyone.

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.

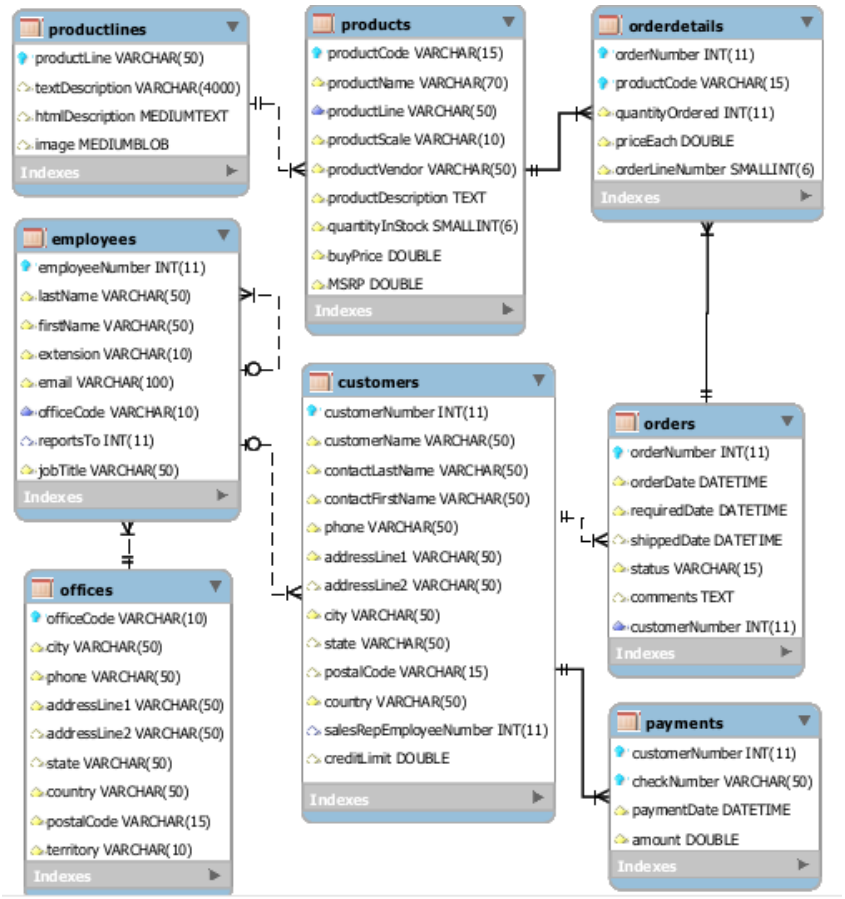

Figure 1: Data Model of Sample Database

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