CSC 443 – Data Base Management Systems

Lecture 8 – More Complex Queries in SQL

Subqueries

- Some SQL statements can have a **`SELECT`** embedded within them.
- A subselect can be used in **`WHERE`** and **`HAVING`** clauses of an outer **`SELECT`**, where it is called a *subquery* or *nested query*.
- Subselects may also appear in **`INSERT`**, **`UPDATE`**, and **`DELETE`** statements.

Subqueries – An Example

```
mysql> select branchNo
    -> from Branch
    -> where street='163 Main St';
+----------+
| branchNo |
+----------+
| B003     |
+----------+
1 row in set (0.00 sec)

mysql>
```

```
mysql> select staffNo, fName, lName, position
    -> from Staff
    -> where branchNo=
    ->     (select branchNo
    ->      from Branch
    ->      where street='163 Main St');
+---------+-------+-------+------------+
| staffNo | fName | lName | position   |
+---------+-------+-------+------------+
| SG14    | David | Ford  | Supervisor |
| SG37    | Ann   | Beech | Assistant  |
| SG5     | Susan | Brand | Manager    |
+---------+-------+-------+------------+
3 rows in set (0.00 sec)

mysql>
```

Example - Subquery with Equality

- Inner **SELECT** finds branch number for branch at **'163 Main St'** (**'B003'**).
- Outer **SELECT** then retrieves details of all staff who work at this branch.
- Outer **SELECT** then becomes:

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo = 'B003';
```

Subquery with Aggregate

Cannot write '**WHERE salary > AVG(salary)**'

Instead, use subquery to find average salary (**17000**), and then use outer SELECT to find those staff with salary greater than this:

```
SELECT staffNo, fName, lName,
position,
     salary – 17000 As salDiff
FROM Staff
WHERE salary > 17000;
```

Example Subquery with Aggregate

```
mysql> select avg(salary)
    -> from Staff;
+--------------+
| avg(salary)  |
+--------------+
| 17000.000000 |
+--------------+
1 row in set (0.00 sec)

mysql>
```

Formatting the Query

```
mysql> select cast(avg(salary) as decimal(9,2))
    -> as AverageSalary
    -> from Staff;
+---------------+
| AverageSalary |
+---------------+
|      17000.00 |
+---------------+
1 row in set (0.00 sec)

mysql>
```

Example Subquery with Aggregate

```
mysql>  select StaffNo, fName, lName, position,
    ->      salary-(select cast(avg(salary)
    ->          as decimal(9, 2))from Staff)
    ->      as SalDiff
    ->      from Staff
    ->      where salary >
    ->        (select avg(salary)
    ->         from Staff);
+---------+-------+-------+------------+----------+
| StaffNo | fName | lName | position   | SalDiff  |
+---------+-------+-------+------------+----------+
| SG14    | David | Ford  | Supervisor |  1000.00 |
| SG5     | Susan | Brand | Manager    |  7000.00 |
| SL21    | John  | White | Manager    | 13000.00 |
+---------+-------+-------+------------+----------+
3 rows in set (0.00 sec)
```

Subquery rules

- **ORDER BY** clause may not be used in a subquery (although it may be used in outermost **SELECT**).
- Subquery **SELECT** list must consist of a single column name or expression, except for subqueries that use **EXISTS**.
- By default, column names refer to table name in **FROM** clause of subquery. Can refer to a table in **FROM** using an *alias*.

Subquery Rules (continued)

- When subquery is an operand in a comparison, subquery must appear on right-hand side.
- A subquery may not be used as an operand in an expression.

Building a Subquery Using **IN**

```
mysql> select branchNo
    -> from Branch
    -> where street='163 Main St';
+----------+
| branchNo |
+----------+
| B003     |
+----------+
1 row in set (0.00 sec)

mysql>
```

```
mysql> select staffNo
    -> from Staff
    -> where branchNo =
    ->    (select branchNo
    ->     from Branch
    ->     where street = '163 Main St');
+---------+
| staffNo |
+---------+
| SG14    |
| SG37    |
| SG5     |
+---------+
3 rows in set (0.00 sec)

mysql>
```

Example- Nested subquery: use of IN

```
mysql> select propertyNo, street, city, postcode,
type, rooms, rent
    -> from PropertyForRent
    -> where staffNo in
    ->   (select staffNo
    ->    from Staff
    ->    where branchNo=
    ->      (select branchNo
    ->       from Branch
    ->       where street='163 Main St'));
```

```
+------------+--------------+---------+----------+-
------+-------+------+
| propertyNo | street       | city    | postcode |
type  | rooms | rent |
+------------+--------------+---------+----------+-
------+-------+------+
| PG16       | 5 Novar Dr   | Glasgow | G12 9AX  |
Flat  |     4 |  450 |
| PG21       | 18 Dale Rd   | Glasgow | G12      |
House |     5 |  600 |
| PG36       | 2 Manor Rd   | Glasgow | G32 4QX  |
Flat  |     3 |  375 |
| PG4        | 6 Lawrence St | Glasgow | G11 9QX  |
Flat  |     3 |  350 |
+------------+--------------+---------+----------+-
------+-------+------+
```

ANY and **ALL**

- **ANY** and **ALL** may be used with subqueries that produce a single column of numbers.
- With **ALL**, condition will only be true if it is satisfied by *all* values produced by subquery.
- With **ANY**, condition will be true if it is satisfied by *any* values produced by subquery.
- If subquery is empty, **ALL** returns true, **ANY** returns false.
- **SOME** may be used in place of **ANY**.

```
mysql> select staffNo, fName, lName, position,
    -> salary
    -> from Staff;
+---------+-------+-------+------------+----------+
| staffNo | fName | lName | position   | salary   |
+---------+-------+-------+------------+----------+
| SA9     | Mary  | Howe  | Assistant  |  9000.00 |
| SG14    | David | Ford  | Supervisor | 18000.00 |
| SG37    | Ann   | Beech | Assistant  | 12000.00 |
| SG5     | Susan | Brand | Manager    | 24000.00 |
| SL21    | John  | White | Manager    | 30000.00 |
| SL41    | Julie | Lee   | Assistant  |  9000.00 |
+---------+-------+-------+------------+----------+
6 rows in set (0.00 sec)

mysql>
```

Building the Query That Uses **SOME**

```
mysql> select salary
    -> from Staff
    -> where branchNo = 'B003';
+----------+
| salary   |
+----------+
| 18000.00 |
| 12000.00 |
| 24000.00 |
+----------+
3 rows in set (0.00 sec)

mysql>
```

The Query Using **SOME**

```
mysql> select staffNo, fName, lName, position,
    -> salary
    -> from Staff
    -> where salary > SOME
    ->           (SELECT salary
    ->            FROM Staff
    ->            WHERE branchNo='B003');
+---------+-------+-------+------------+----------+
| staffNo | fName | lName | position   | salary   |
+---------+-------+-------+------------+----------+
| SG14    | David | Ford  | Supervisor | 18000.00 |
| SG5     | Susan | Brand | Manager    | 24000.00 |
| SL21    | John  | White | Manager    | 30000.00 |
+---------+-------+-------+------------+----------+
```

Explaining the Query

- Inner query produces set {**12000**, **18000**, **24000**} and outer query selects those staff whose salaries are greater than any of the values in this set.

Multi-Table Queries

- Can use subqueries provided result columns come from same table.
- If result columns come from more than one table must use a join.
- To perform join, include more than one table in FROM clause.
- Use comma as separator and typically include WHERE clause to specify join column(s).

Multi-Table Queries (continued)

- Also possible to use an alias for a table named in **`FROM`** clause.
- Alias is separated from table name with a space.
- Alias can be used to qualify column names when there is ambiguity.

Example – Simple Join

- List names of all clients who have viewed a property along with any comment supplied.
- Only those rows from both tables that have identical values in the **`clientNo`** columns (**`c.clientNo = v.clientNo`**) are included in result.

```
mysql> SELECT c.clientNo, fName, lName,
    ->        propertyNo, comment
    -> FROM Client c, Viewing v
    -> WHERE c.clientNo = v.clientNo;
+----------+-------+---------+------------+----------------+
| clientNo | fName | lName   | propertyNo | comment        |
+----------+-------+---------+------------+----------------+
| CR56     | Aline | Stewart | PA14       | too small      |
| CR56     | Aline | Stewart | PG36       | NULL           |
| CR56     | Aline | Stewart | PG4        | NULL           |
| CR62     | Mary  | Tregear | PA14       | no dining room |
| CR76     | John  | Kay     | PG4        | too remote     |
+----------+-------+---------+------------+----------------+
5 rows in set (0.00 sec)

mysql>
```

JOIN - Alternate Ways

- SQL provides alternative ways to specify joins:

```
FROM Client c JOIN Viewing v
    ON c.clientNo = v.clientNo
FROM Client NATURAL JOIN Viewing
```

- In each case, **FROM** replaces original **FROM** and **WHERE**. However, first produces table with two identical clientNo columns.

Example - Using **JOIN**

```
mysql> select fName, lName
    -> FROM Client c JOIN Viewing v
    -> ON c.clientNo = v.clientNo;
+-------+---------+
| fName | lName   |
+-------+---------+
| Aline | Stewart |
| Aline | Stewart |
| Aline | Stewart |
| Mary  | Tregear |
| John  | Kay     |
+-------+---------+
5 rows in set (0.00 sec)

mysql>
```

Example - Using **JOIN**

```
mysql> select fName, lName
    -> FROM Client NATURAL JOIN Viewing;
+-------+---------+
| fName | lName   |
+-------+---------+
| Aline | Stewart |
| Aline | Stewart |
| Aline | Stewart |
| Mary  | Tregear |
| John  | Kay     |
+-------+---------+
5 rows in set (0.00 sec)

mysql>
```

Example – Sorting a **JOIN**

- For each branch, list numbers and names of staff who manage properties, and properties they manage.

```
mysql> SELECT s.branchNo, s.staffNo, fName, lName,
    ->        propertyNo
    -> FROM Staff s, PropertyForRent p
    -> WHERE s.staffNo = p.staffNo
    -> ORDER BY s.branchNo, s.staffNo, propertyNo;
+----------+---------+-------+-------+------------+
| branchNo | staffNo | fName | lName | propertyNo |
+----------+---------+-------+-------+------------+
| B003     | SG14    | David | Ford  | PG16       |
| B003     | SG37    | Ann   | Beech | PG21       |
| B003     | SG37    | Ann   | Beech | PG36       |
| B003     | SG5     | Susan | Brand | PG4        |
| B005     | SL41    | Julie | Lee   | PL94       |
| B007     | SA9     | Mary  | Howe  | PA14       |
+----------+---------+-------+-------+------------+
```

Three-table Join

- For each branch, list the staff numbers and names of staff who manage properties, including the city in which the branch is located and the properties that the staff manage.

```
mysql> select b.branchNo, b.city, s.staffNo, fName,
    -> lName
    -> from Branch b, Staff s
    -> where b.branchNo=s.branchNo
    -> order by b.branchNo,s.staffNo;
+----------+----------+---------+-------+-------+
| branchNo | city     | staffNo | fName | lName |
+----------+----------+---------+-------+-------+
| B003     | Glasgow  | SG14    | David | Ford  |
| B003     | Glasgow  | SG37    | Ann   | Beech |
| B003     | Glasgow  | SG5     | Susan | Brand |
| B005     | London   | SL21    | John  | White |
| B005     | London   | SL41    | Julie | Lee   |
| B007     | Aberdeen | SA9     | Mary  | Howe  |
+----------+----------+---------+-------+-------+
6 rows in set (0.00 sec)
mysql>
```

```
mysql> select b.branchNo, b.city, s.staffNo, fName,
    -> lName, propertyNo
    -> from Branch b, Staff s, PropertyForRent p
    -> where b.branchNo=s.branchNo AND
    -> s.staffNo=p.staffNo
    -> order by b.branchNo, s.staffNo, propertyNo;
+----------+----------+---------+-------+-------+------------+
| branchNo | city     | staffNo | fName | lName | propertyNo |
+----------+----------+---------+-------+-------+------------+
| B003     | Glasgow  | SG14    | David | Ford  | PG16       |
| B003     | Glasgow  | SG37    | Ann   | Beech | PG21       |
| B003     | Glasgow  | SG37    | Ann   | Beech | PG36       |
| B003     | Glasgow  | SG5     | Susan | Brand | PG4        |
| B005     | London   | SL41    | Julie | Lee   | PL94       |
| B007     | Aberdeen | SA9     | Mary  | Howe  | PA14       |
+----------+----------+---------+-------+-------+------------+
6 rows in set (0.00 sec)

mysql>
```

```
mysql> select *
    -> from Branch join Staff;
+----------+-------------+----------+----------+---------+-------+-------+------------
+------+------------+----------+----------+
| branchNo | street      | city     | postcode | staffNo | fName | lName | position
| sex  | DOB        | salary   | branchNo |
+----------+-------------+----------+----------+---------+-------+-------+------------
+------+------------+----------+----------+
| B002     | 56 Clover Dr | London   | NW10 6EU | SA9     | Mary  | Howe  | Assistant
| f    | 1970-02-19 |  9000.00 | B007     |
| B003     | 163 Main St  | Glasgow  | G11 9QX  | SA9     | Mary  | Howe  | Assistant
| f    | 1970-02-19 |  9000.00 | B007     |
| B004     | 32 Manse Rd  | Bristol  | BS99 1NZ | SA9     | Mary  | Howe  | Assistant
| f    | 1970-02-19 |  9000.00 | B007     |
| B005     | 22 Deer Rd   | London   | SW1 4EH  | SA9     | Mary  | Howe  | Assistant
| f    | 1970-02-19 |  9000.00 | B007     |
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  | SA9     | Mary  | Howe  | Assistant
| f    | 1970-02-19 |  9000.00 | B007     |
| B002     | 56 Clover Dr | London   | NW10 6EU | SG14    | David | Ford  | Supervisor
| M    | 1958-03-24 | 18000.00 | B003     |
| B003     | 163 Main St  | Glasgow  | G11 9QX  | SG14    | David | Ford  | Supervisor
| M    | 1958-03-24 | 18000.00 | B003     |
| B004     | 32 Manse Rd  | Bristol  | BS99 1NZ | SG14    | David | Ford  | Supervisor
| M    | 1958-03-24 | 18000.00 | B003     |
| B005     | 22 Deer Rd   | London   | SW1 4EH  | SG14    | David | Ford  | Supervisor
| M    | 1958-03-24 | 18000.00 | B003     |
```

```
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  | SG14    | David | Ford  | Supervisor
| M    | 1958-03-24 | 18000.00 | B003     |
| B002     | 56 Clover Dr | London   | NW10 6EU | SG37    | Ann   | Beech | Assistant
| f    | 1960-11-10 | 12000.00 | B003     |
| B003     | 163 Main St  | Glasgow  | G11 9QX  | SG37    | Ann   | Beech | Assistant
| f    | 1960-11-10 | 12000.00 | B003     |
| B004     | 32 Manse Rd  | Bristol  | BS99 1NZ | SG37    | Ann   | Beech | Assistant
| f    | 1960-11-10 | 12000.00 | B003     |
| B005     | 22 Deer Rd   | London   | SW1 4EH  | SG37    | Ann   | Beech | Assistant
| f    | 1960-11-10 | 12000.00 | B003     |
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  | SG37    | Ann   | Beech | Assistant
| f    | 1960-11-10 | 12000.00 | B003     |
| B002     | 56 Clover Dr | London   | NW10 6EU | SG5     | Susan | Brand | Manager
| f    | 1940-06-03 | 24000.00 | B003     |
| B003     | 163 Main St  | Glasgow  | G11 9QX  | SG5     | Susan | Brand | Manager
| f    | 1940-06-03 | 24000.00 | B003     |
| B004     | 32 Manse Rd  | Bristol  | BS99 1NZ | SG5     | Susan | Brand | Manager
| f    | 1940-06-03 | 24000.00 | B003     |
| B005     | 22 Deer Rd   | London   | SW1 4EH  | SG5     | Susan | Brand | Manager
| f    | 1940-06-03 | 24000.00 | B003     |
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  | SG5     | Susan | Brand | Manager
| f    | 1940-06-03 | 24000.00 | B003     |
| B002     | 56 Clover Dr | London   | NW10 6EU | SL21    | John  | White | Manager
| M    | 1945-10-01 | 30000.00 | B005     |
| B003     | 163 Main St  | Glasgow  | G11 9QX  | SL21    | John  | White | Manager
| M    | 1945-10-01 | 30000.00 | B005     |
| B004     | 32 Manse Rd  | Bristol  | BS99 1NZ | SL21    | John  | White | Manager
| M    | 1945-10-01 | 30000.00 | B005     |
```

```
| B005     | 22 Deer Rd   | London   | SW1 4EH  | SL21    | John  | White | Manager
| M    | 1945-10-01 | 30000.00 | B005     |
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  | SL21    | John  | White | Manager
| M    | 1945-10-01 | 30000.00 | B005     |
| B002     | 56 Clover Dr | London   | NW10 6EU | SL41    | Julie | Lee   | Assistant
| f    | 1965-06-13 |  9000.00 | B005     |
| B003     | 163 Main St  | Glasgow  | G11 9QX  | SL41    | Julie | Lee   | Assistant
| f    | 1965-06-13 |  9000.00 | B005     |
| B004     | 32 Manse Rd  | Bristol  | BS99 1NZ | SL41    | Julie | Lee   | Assistant
| f    | 1965-06-13 |  9000.00 | B005     |
| B005     | 22 Deer Rd   | London   | SW1 4EH  | SL41    | Julie | Lee   | Assistant
| f    | 1965-06-13 |  9000.00 | B005     |
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  | SL41    | Julie | Lee   | Assistant
| f    | 1965-06-13 |  9000.00 | B005     |
+----------+--------------+----------+----------+---------+-------+-------+-----------
+------+------------+----------+----------+
30 rows in set (0.01 sec)

mysql>
```

```
mysql> select * from Branch join Staff using(branchNo);
+----------+--------------+----------+----------+---------+------
-+-------+------------+------+------------+----------+
| branchNo | street       | city     | postcode | staffNo | fName
| lName | position   | sex  | DOB        | salary   |
+----------+--------------+----------+----------+---------+------
-+-------+------------+------+------------+----------+
| B003     | 163 Main St  | Glasgow  | G11 9QX  | SG14    | David
| Ford  | Supervisor | M    | 1958-03-24 | 18000.00 |
| B003     | 163 Main St  | Glasgow  | G11 9QX  | SG37    | Ann
| Beech | Assistant  | f    | 1960-11-10 | 12000.00 |
| B003     | 163 Main St  | Glasgow  | G11 9QX  | SG5     | Susan
| Brand | Manager    | f    | 1940-06-03 | 24000.00 |
| B005     | 22 Deer Rd   | London   | SW1 4EH  | SL21    | John
| White | Manager    | M    | 1945-10-01 | 30000.00 |
| B005     | 22 Deer Rd   | London   | SW1 4EH  | SL41    | Julie
| Lee   | Assistant  | f    | 1965-06-13 |  9000.00 |
| B007     | 16 Argyll St | Aberdeen | AB2 3SU  | SA9     | Mary
| Howe  | Assistant  | f    | 1970-02-19 |  9000.00 |
+----------+--------------+----------+----------+---------+------
-+-------+------------+------+------------+----------+
6 rows in set (0.00 sec)
mysql>
```

```
mysql> select branchNo, city, staffNo, fName, lName
    -> from (Branch join Staff using (branchNo) );
+----------+----------+---------+-------+-------+
| branchNo | city     | staffNo | fName | lName |
+----------+----------+---------+-------+-------+
| B003     | Glasgow  | SG14    | David | Ford  |
| B003     | Glasgow  | SG37    | Ann   | Beech |
| B003     | Glasgow  | SG5     | Susan | Brand |
| B005     | London   | SL21    | John  | White |
| B005     | London   | SL41    | Julie | Lee   |
| B007     | Aberdeen | SA9     | Mary  | Howe  |
+----------+----------+---------+-------+-------+
6 rows in set (0.01 sec)

mysql>
```

```
mysql> select b.branchNo, b.city, staffNo, fName, lName,
    -> propertyNo
    -> from (Branch b join Staff s using (branchNo))
    -> join PropertyForRent using (staffNo);
+----------+----------+---------+-------+-------+------------+
| branchNo | city     | staffNo | fName | lName | propertyNo |
+----------+----------+---------+-------+-------+------------+
| B003     | Glasgow  | SG14    | David | Ford  | PG16       |
| B003     | Glasgow  | SG37    | Ann   | Beech | PG21       |
| B003     | Glasgow  | SG37    | Ann   | Beech | PG36       |
| B003     | Glasgow  | SG5     | Susan | Brand | PG4        |
| B005     | London   | SL41    | Julie | Lee   | PL94       |
| B007     | Aberdeen | SA9     | Mary  | Howe  | PA14       |
+----------+----------+---------+-------+-------+------------+
6 rows in set (0.00 sec)

mysql>
```

Multiple Grouping Columns

- Find the number of porperties handled by each staff member, along with the branch number of the member of staff.

```
mysql> select s.branchNo, s.staffNo,
    -> count(*) as myCount
    -> from Staff s, PropertyForRent p
    -> where s.staffNo=p.staffNo
    -> group by s.branchNo, s.staffNo
    -> order by s.branchNo, s.staffNo;
+----------+---------+---------+
| branchNo | staffNo | myCount |
+----------+---------+---------+
| B003     | SG14    |       1 |
| B003     | SG37    |       2 |
| B003     | SG5     |       1 |
| B005     | SL41    |       1 |
| B007     | SA9     |       1 |
+----------+---------+---------+
5 rows in set (0.02 sec)
mysql>
```

Computing a Join

Procedure for generating results of a join are:

1. Form Cartesian product of the tables named in **`FROM`** clause.
2. If there is a **`WHERE`** clause, apply the search condition to each row of the product table, retaining those rows that satisfy the condition.
3. For each remaining row, determine value of each item in **`SELECT`** list to produce a single row in result table.

Outer Joins

- If one row of a joined table is unmatched, row is omitted from result table.
- Outer join operations retain rows that do not satisfy the join condition.
- Consider following tables:

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

Outer Joins (continued)

- The (inner) join of these two tables:

```
SELECT b.*, p.*
FROM Branch1 b, PropertyForRent1 p
WHERE b.bCity = p.pCity;
```

Table 5.27(b) Result table for inner join of Branch1 and PropertyForRent1 tables.

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

Outer Joins (continued)

- Result table has two rows where cities are same.
- There are no rows corresponding to branches in Bristol and Aberdeen.
- To include unmatched rows in result table, use an Outer join.

Left Outer Join - Example

List branches and properties that are in same city along with any unmatched branches.

```
SELECT b.*, p.*
FROM Branch1 b LEFT JOIN
PropertyForRent1 p ON b.bCity = p.pCity;
```

Left Outer Join - Example

- Includes those rows of first (left) table unmatched with rows from second (right) table.
- Columns from second table are filled with NULLs.

Table 5.28 Result table for Example 5.28.

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

Right Outer Join Example

List branches and properties in same city and any unmatched properties.

```
SELECT b.*, p.*
FROM Branch1 b RIGHT JOIN
PropertyForRent1 p ON b.bCity = p.pCity;
```

Right Outer Join Example

- Right Outer join includes those rows of second (right) table that are unmatched with rows from first (left) table.
- Columns from first table are filled with **NULL**s.

Table 5.29 Result table for Example 5.29.

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London

Full Outer Join Example

- List branches and properties in same city and any unmatched branches or properties.

```
SELECT b.*, p.*
FROM Branch1 b FULL JOIN
     PropertyForRent1 p
     ON b.bCity = p.pCity;
```

Full Outer Join Example

- Includes rows that are unmatched in both tables.
- Unmatched columns are filled with **NULL**s.

Table 5.30 Result table for Example 5.30.

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London

EXISTS and **NOT EXISTS**

- **EXISTS** and **NOT EXISTS** are for use only with subqueries.
- Produce a simple true/false result.
- True if and only if there exists at least one row in result table returned by subquery.
- False if subquery returns an empty result table.
- **NOT EXISTS** is the opposite of **EXISTS**.

EXISTS and **NOT EXISTS** (continued)

- As (**NOT**) **EXISTS** check only for existence or non-existence of rows in subquery result table, subquery can contain any number of columns.
- Common for subqueries following (**NOT**) **EXISTS** to be of form:

```
(SELECT * ...)
```

```
mysql> select staffNo, fName, lName, position
    -> from Staff s
    -> where exists
    -> (select *
    ->  from Branch b
    ->  where s.branchNo=b.branchNo
    ->  AND city='London');
+---------+-------+-------+-----------+
| staffNo | fName | lName | position  |
+---------+-------+-------+-----------+
| SL21    | John  | White | Manager   |
| SL41    | Julie | Lee   | Assistant |
+---------+-------+-------+-----------+
2 rows in set (0.00 sec)

mysql>
```

Query Using **EXISTS**

- Note, search condition

  s.branchNo = b.branchNo

  is necessary to consider correct branch record for each member of staff.
- If omitted, would get all staff records listed out because subquery:

  SELECT * FROM Branch WHERE city='London'

  would always be true and query would be:

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE true;
```

Rewriting the Query Using **JOIN**

```
mysql> select staffNo, fName, lName, position
    -> from Staff s, Branch b
    -> where s.branchNo=b.branchNo
    -> AND city='London';
+---------+-------+-------+-----------+
| staffNo | fName | lName | position  |
+---------+-------+-------+-----------+
| SL21    | John  | White | Manager   |
| SL41    | Julie | Lee   | Assistant |
+---------+-------+-------+-----------+
2 rows in set (0.00 sec)

mysql>
```

Union, Intersect, and Difference (Except)

- Can use normal set operations of Union, Intersection, and Difference to combine results of two or more queries into a single result table.
- Union of two tables, A and B, is table containing all rows in either A or B or both.
- Intersection is table containing all rows common to both A and B.
- Difference is table containing all rows in A but not in B.
- Two tables must be *union compatible*.

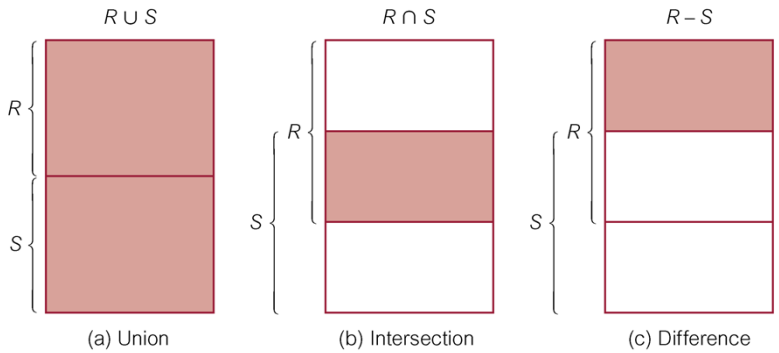
Union, Intersect, and Difference (Except)

- Format of set operator clause in each case is:

```
op [ALL] [CORRESPONDING [BY
        {column1 [, ...]}]]
```

- If **CORRESPONDING BY** specified, set operation performed on the named column(s).
- If **CORRESPONDING** specified but not **BY** clause, operation performed on common columns.
- If **ALL** specified, result can include duplicate rows.

Union, Intersect, and Difference (Except)



Use of **UNION** - Example

- List all cities where there is either a branch office or a property.

Use of **UNION** - Example

```
mysql> select staffNo, fName, lName, position
    -> from Staff s, Branch b
    -> where s.branchNo=b.branchNo
    -> AND city='London';
+---------+-------+-------+-----------+
| staffNo | fName | lName | position  |
+---------+-------+-------+-----------+
| SL21    | John  | White | Manager   |
| SL41    | Julie | Lee   | Assistant |
+---------+-------+-------+-----------+
2 rows in set (0.00 sec)

mysql>
```

INSERT

```
INSERT INTO TableName [ (columnList) ]
   VALUES (dataValueList)
```

- *columnList* is optional; if omitted, SQL assumes a list of all columns in their original **CREATE TABLE** order.
- Any columns omitted must have been declared as **NULL** when table was created, unless **DEFAULT** was specified when creating column.

INSERT

- *dataValueList* must match *columnList* as follows:
  - number of items in each list must be same;
  - must be direct correspondence in position of items in two lists;
  - data type of each item in *dataValueList* must be compatible with data type of corresponding column.

INSERT VALUES – An Example

Insert a new row into Staff table supplying data for all columns.

```
mysql> insert into Staff values
    -> ('SG16', 'Alan', 'Brown', 'Assistant', 'M',
    -> '1957-05-25', 8300,
    -> 'B003');
Query OK, 1 row affected (0.02 sec)

mysql>
```

```
mysql> select staffNo, fName, lName, position
    -> from Staff;
+---------+-------+-------+------------+
| staffNo | fName | lName | position   |
+---------+-------+-------+------------+
| SA9     | Mary  | Howe  | Assistant  |
| SG14    | David | Ford  | Supervisor |
| SG16    | Alan  | Brown | Assistant  |
| SG37    | Ann   | Beech | Assistant  |
| SG5     | Susan | Brand | Manager    |
| SL21    | John  | White | Manager    |
| SL41    | Julie | Lee   | Assistant  |
+---------+-------+-------+------------+
7 rows in set (0.00 sec)

mysql>
```

INSERT Using Defaults

- Insert a new row into Staff table supplying data for all mandatory columns.

INSERT Using Defaults

```
mysql> insert into Staff(staffNo, fName, lName,
    -> position, salary, branchNo)
    -> values ('SG44', 'Anne', 'Jones', 'Assistant',
8100, 'B003');
Query OK, 1 row affected (0.00 sec)

mysql> insert into Staff
    -> values('SG45', 'Anna', 'Smith', 'Assistant',
    -> NULL, NULL, 8200, 'B002');
Query OK, 1 row affected (0.01 sec)

mysql>
```

INSERT…SELECT

- Second form of **INSERT** allows multiple rows to be copied from one or more tables to another:

```
INSERT INTO TableName
       [ (columnList) ]
SELECT ...
```

INSERT…SELECT

- Assume there is a table StaffPropCount that contains names of staff and number of properties they manage:

  StaffPropCount(staffNo, fName, lName, propCnt)

- Populate StaffPropCount using Staff and PropertyForRent tables.

INSERT…SELECT – An Example

```
INSERT INTO StaffPropCount
  (SELECT s.staffNo, fName, lName, COUNT(*)
  FROM Staff s, PropertyForRent p
  WHERE s.staffNo = p.staffNo
  GROUP BY s.staffNo, fName, lName)
  UNION
  (SELECT staffNo, fName, lName, 0
  FROM Staff
  WHERE staffNo NOT IN
   (SELECT DISTINCT staffNo
   FROM PropertyForRent));
```

INSERT…SELECT – An Example

- If second part of UNION is omitted, excludes those staff who currently do not manage any properties.

Table 5.35 Result table for Example 5.37.

staffNo	fName	lName	propCount
SG14	David	Ford	1
SL21	John	White	0
SG37	Ann	Beech	2
SA9	Mary	Howe	1
SG5	Susan	Brand	0
SL41	Julie	Lee	1

UPDATE

```
UPDATE TableName
SET columnName1 = dataValue1
     [, columnName2 = dataValue2...]
[WHERE searchCondition]
```

- *TableName* can be name of a base table or an updatable view.
- **SET** clause specifies names of one or more columns that are to be updated**.**

UPDATE (continued)

- **WHERE** clause is optional:
  - if omitted, named columns are updated for all rows in table;
  - if specified, only those rows that satisfy *searchCondition* are updated.
- New *dataValue(s)* must be compatible with data type for corresponding column.

UPDATE All Rows – An Example

- Give all staff a 3% pay increase.
- Give all Managers a 5% pay increase.

UPDATE All Rows – An Example

```
mysql> update Staff
    -> set salary = salary * 1.03;
Query OK, 9 rows affected (0.01 sec)
Rows matched: 9  Changed: 9  Warnings: 0

mysql> update Staff
    -> set salary = salary * 1.05
    -> where position = 'manager';
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql>
```

UPDATE Multiple Columns – An Example

- Promote David Ford (staffNo='SG14') to Manager and change his salary to £18,000.

```
mysql> update Staff
    -> set position = 'manager', salary = 18000
    -> where staffNo = 'SG14';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

DELETE

```
DELETE FROM TableName
[WHERE searchCondition]
```

- *TableName* can be name of a base table or an updatable view.
- *searchCondition* is optional; if omitted, all rows are deleted from table. This does not delete table. If *search_condition* is specified, only those rows that satisfy condition are deleted.

DELETE Specific Rows

- Delete all viewings that relate to property PG4.

```
mysql> delete from Viewing
    -> where propertyNo = 'PG4';
Query OK, 2 rows affected (0.00 sec)

mysql>
```

- Delete all records from the Viewing table.

```
DELETE FROM Viewing;
```