Example 5.27 Multiple Grouping Columns

Find number of properties handled by each staff member.

```
SELECT s.branchNo, s.staffNo, COUNT(*) AS count
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo
GROUP BY s.branchNo, s.staffNo
ORDER BY s.branchNo, s.staffNo;
```

Example 5.27 Multiple Grouping Columns

Table 5.27(a) Result table for Example 5.27.

branchNo	staffNo	count
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

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.

Computing a Join

4. If DISTINCT has been specified, eliminate any duplicate rows from the result table.

5. If there is an ORDER BY clause, sort result table as required.

- SQL provides special format of SELECT for Cartesian product:

```
SELECT [DISTINCT | ALL]   {* | columnList}
FROM Table1 CROSS JOIN Table2
```

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

- 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

- 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.

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow

Example 5.28 Left Outer Join

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;
```

Example 5.28 Left Outer Join

- 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

Example 5.29 Right Outer Join

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;
```

Example 5.29 Right Outer Join

- **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 NULLs.

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

Example 5.30 Full Outer Join

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;
```

Example 5.30 Full Outer Join

- Includes rows that are unmatched in both tables.
- Unmatched columns are filled with NULLs.

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

- 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 * ...)

Example 5.31 Query using EXISTS

Find all staff who work in a London branch.

```
SELECT staffNo, fName, lName, position
FROM Staff s
WHERE EXISTS
    (SELECT *
     FROM Branch b
     WHERE s.branchNo = b.branchNo AND
                 city = 'London');
```

Example 5.31 Query using EXISTS

Table 5.31 Result table for Example 5.31.

staffNo	fName	lName	position
SL21	John	White	Manager
SL41	Julie	Lee	Assistant

Example 5.31 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;
```

Example 5.31 Query using EXISTS

- Could also write this query using join construct:

```
SELECT staffNo, fName, lName, position
FROM Staff s, Branch b
WHERE s.branchNo = b.branchNo AND
        city = 'London';
```

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*.

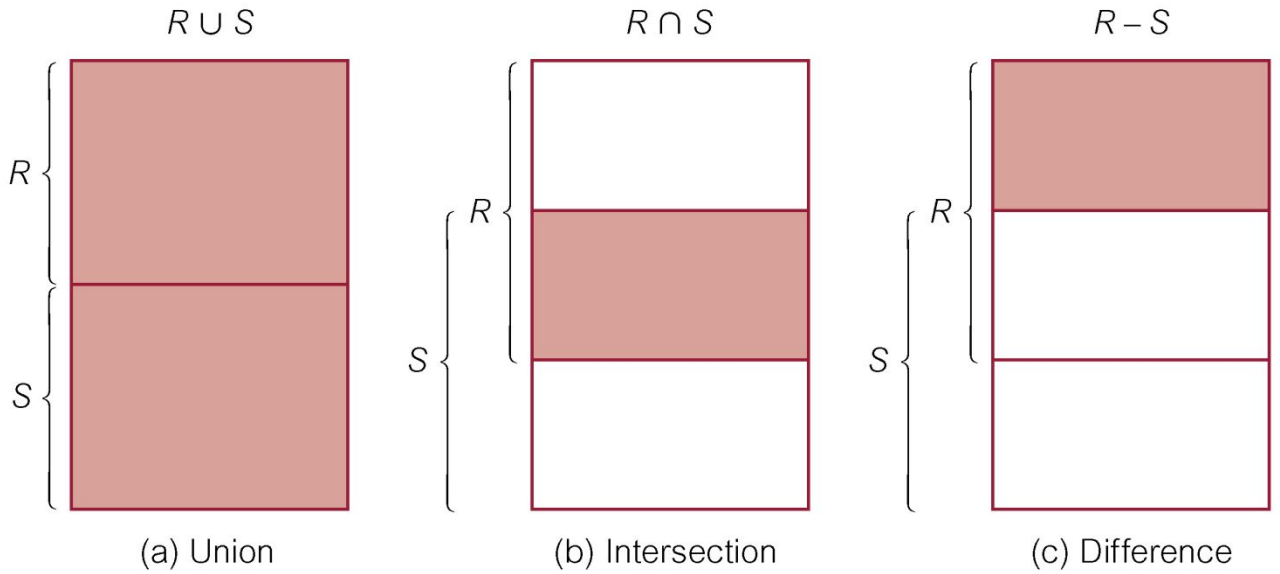
Union, Intersect, and Difference (Except)

- 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)



Example 5.32 Use of UNION

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

```
(SELECT city
    FROM Branch
    WHERE city IS NOT NULL)
UNION
    (SELECT city
    FROM PropertyForRent
    WHERE city IS NOT NULL);
```

Example 5.32 Use of UNION

- **Or**

```
(SELECT *
  FROM Branch
  WHERE city IS NOT NULL)
  UNION CORRESPONDING BY city
  (SELECT *
  FROM PropertyForRent
  WHERE city IS NOT NULL);
```

Example 5.32 Use of UNION

- Produces result tables from both queries and merges both tables together.

Table 5.32 Result table for Example 5.32.

city
London
Glasgow
Aberdeen
Bristol

Example 5.33 Use of INTERSECT

List all cities where there is both a branch office and a property.

```
(SELECT city FROM Branch)
INTERSECT
(SELECT city FROM PropertyForRent);
```

Example 5.33 Use of INTERSECT

- Or

```
(SELECT * FROM Branch)
INTERSECT CORRESPONDING BY city
(SELECT * FROM PropertyForRent);
```

Table 5.33 Result table for Example 5.33.

city
Aberdeen
Glasgow
London

Example 5.33 Use of INTERSECT

- Could rewrite this query without INTERSECT operator:

```
SELECT b.city
FROM Branch b PropertyForRent p
WHERE b.city = p.city;
```

- Or:

```
SELECT DISTINCT city FROM Branch b
WHERE EXISTS
 (SELECT * FROM PropertyForRent p
 WHERE p.city = b.city);
```

Example 5.34 Use of EXCEPT

List of all cities where there is a branch office but no properties.

```
(SELECT city FROM Branch)
EXCEPT
(SELECT city FROM PropertyForRent);
```

- Or

```
(SELECT * FROM Branch)
EXCEPT CORRESPONDING BY city
(SELECT * FROM PropertyForRent
```

Table 5.34 Result table for Example 5.34.

city
Bristol

Example 5.34 Use of EXCEPT

- Could rewrite this query without EXCEPT:

```
SELECT DISTINCT city FROM Branch
WHERE city NOT IN
 (SELECT city FROM PropertyForRent);
```

- Or

```
SELECT DISTINCT city FROM Branch b
WHERE NOT EXISTS
 (SELECT * FROM PropertyForRent p
 WHERE p.city = b.city);
```