

Database Management Systems (Lab)

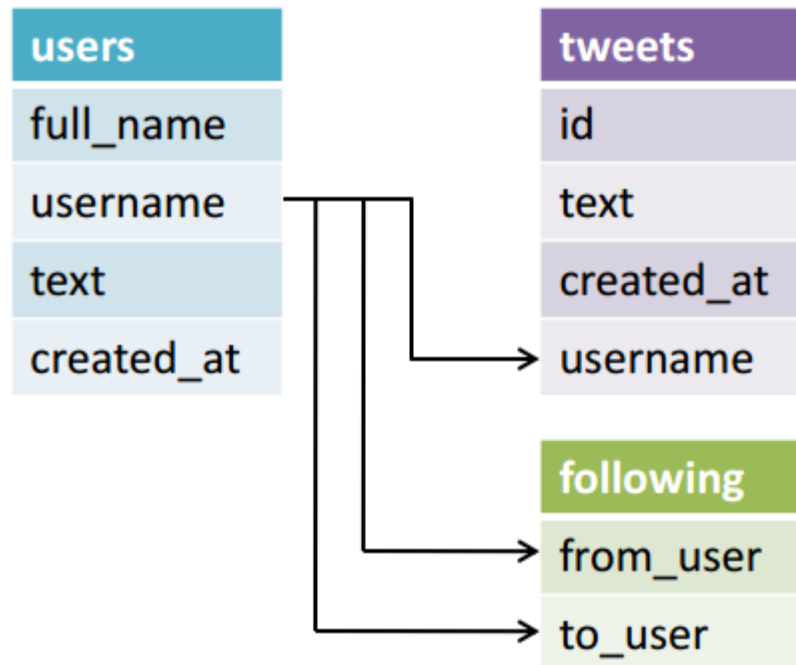
Vikas Bajpai

Relational Database Concept:

- Dr. E. F. Codd proposed the relational model for database systems in 1970.
- It is the basis for the relational database management system (RDBMS)
- The relational model consists of the following:
 - Collection of objects or relations
 - Set of operators to act on the relations
 - Data integrity for accuracy and consistency

Definition of a Relational Database:

- A relational database is a collection of relations or two-dimensional tables.



Difference between DBMS and RDBMS



No. DBMS	RDBMS
1) DBMS applications store data as file .	RDBMS applications store data in a tabular form .
2) In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3) Normalization is not present in DBMS.	Normalization is present in RDBMS.
4) DBMS does not apply any security with regards to data manipulation.	RDBMS defines the integrity constraint for the purpose of ACID (Atomocity, Consistency, Isolation and Durability) property.
5) DBMS uses file system to store data, so there will be no relation between the tables .	in RDBMS, data values are stored in the form of tables, so a relationship between these data values will be stored in the form of a table as well.
6) DBMS has to provide some uniform methods to access the stored information.	RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information.
7) DBMS does not support distributed database .	RDBMS supports distributed database .
8) DBMS is meant to be for small organization and deal with small data . it supports single user .	RDBMS is designed to handle large amount of data . it supports multiple users .
9) Examples of DBMS are file systems, xml etc.	Example of RDBMS are mysql, postgre, sql server, oracle etc.

A Relational Database:

- Can be accessed and modified by executing structured query language (SQL) statements.
- Contains a collection of tables with no physical pointers
- Uses a set of operators

Structures Query Language:

- Structures Query Language
 - Acronym: SQL
 - Pronounced “Sequel” or “S-Q-L”
 - Originally developed by IBM as the SEQUEL language in 1970s
 - Designed to support Edgar Codd’s relational model
 - Based on Relational Algebra
 - ANSI/ISO standard

SQL Defined:

- SQL is not a programming language, but rather is a data sub-language
- SQL is comprised of
 - A Data Definition Language (DDL)
 - Used to define and manage database structures
 - A data manipulation language (DML)
 - Data updating
 - Data retrieval (Queries)
 - A data Control Language (DCL)
 - For creating user accounts, managing permissions etc.

Cells:

- All DBMS allow users to create containers for data storage and management.
- These containers are called 'Cells' (also called as Field).

Information for cell construction:

- The cell Name
- The cell Length
- The cell Data type

Cell Name

20 Characters

Four basic Database operations:

1. Create
2. Read
3. Update
4. Delete

From Relations to Entities to Tables:

- Item (itemId, description, cost, listPrice, quantityOnHand)

Writing Basic SQL Statements

SQL Statements:

•SELECT	Data Retrieval/Query Language
•INSERT	
•UPDATE	
•DELETE	
•CREATE	Data Manipulation Language
•ALTER	
•DROP	
•RENAME	
•TRUNCATE	
•COMMIT	Data Definition Language
•ROLLBACK	
•SAVEPOINT	
•GRANT	Transaction Control
•REVOKE	

About SQL statements:

- SQL statements are **not** case **sensitive**
- SQL statements can be on one or **more lines**
- Keywords **cannot** be **abbreviated** or split across lines
- Clauses are usually placed on separate lines
- Tabs and indents are used to enhance readability

Basic SELECT Statement:

```
SQL> SELECT [DISTINCT] FROM table;
```

- SELECT identifies what columns
- FROM identifies which table

Selecting all columns:

```
SQL> SELECT * FROM dept;
```

DEPT_NO	DEPT_NAME	LOCATION
10	CSE	JAIPUR
20	ECE	JODHPUR
30	CCE	UDAIPUR
40	MME	JAISALMER

Selecting specific Columns:

```
SQL> SELECT DEPT_NO, LOCATION FROM dept;
```

DEPT_NO	LOCATION
10	JAIPUR
20	JODHPUR
30	UDAIPUR
40	JAISALMER

Arithmetic Expressions:

- Create expressions on NUMBER and DATE data by using arithmetic operators.

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide

Using Arithmetic Operators:

```
SQL>  SELECT  DEPT_NO,  DEPT_NO  +10,  
          LOCATION FROM dept;
```

DEPT_NO	DEPT_NO + 10	LOCATION
10	20	JAIPUR
20	30	JODHPUR
30	40	UDAIPUR
40	50	JAISALMER

Operator Precedence:



- Multiplication and division take priority over addition and subtraction.
- Operators of the same priority are evaluated from left to right.
- Parentheses are used to force prioritized evaluation and to clarify statements.

Operator Precedence:

- SQL> SELECT DEPT_NO, 12*DEPT_NO +10, LOCATION FROM dept;

DEPT_NO	12*DEPT_NO + 10	LOCATION
10	130	JAIPUR
20	250	JODHPUR
30	370	UDAIPUR
40	490	JAISALMER

Using Parentheses:

- SQL> SELECT DEPT_NO, 12*(DEPT_NO +10),
LOCATION FROM dept;

DEPT_NO	12*(DEPT_NO + 10)	LOCATION
10	240	JAIPUR
20	360	JODHPUR
30	480	UDAIPUR
40	600	JAISALMER

Null Value

- If a row lacks the data value for a particular column, that value is said to be **null**, or to contain null.
- A null value is a value that is **unavailable, unassigned, unknown, or inapplicable**.
- A null value is **not the same** as zero or a space. Zero is a number, and a space is a character.
- Columns of any datatype can contain null values, unless the column was defined as NOT NULL or as PRIMARY KEY when the column was created.

Defining a Null Value:

```
SQL> SELECT DEPT_NO, LOCATION,  
           NULL_EXAMPLE FROM dept;
```

DEPT_NO	LOCATION	NULL_EXAMPLE
10	JAIPUR	
20	JODHPUR	
30	UDAIPUR	
40	JAISALMER	

Null Values in Arithmetic Expressions:

- Arithmetic expressions containing a null value evaluate to null.

```
SQL> SELECT DEPT_NO, LOCATION,  
           DEPT_NO+NULL_EXAMPLE FROM dept;
```

DEPT_NO	LOCATION	DEPT_NO + NULL_EXAMPLE
10	JAIPUR	
20	JODHPUR	
30	UDAIPUR	
40	JAISALMER	

Defining a Column Alias:

- Renames a column heading
- Is useful with calculations
- Immediately follows the column name - there can also be the optional AS keyword between the column name and alias
- Requires double quotation marks if it contains spaces or special characters or is case sensitive

Using Column Aliases:

```
SQL> SELECT DEPT_NO AS DEPT, LOCATION LOC  
FROM dept;
```

OR

```
SQL> SELECT DEPT_NO AS DEPT, LOCATION AS  
LOC FROM dept;
```

↓
optional

DEPT	LOC
10	JAIPUR
20	JODHPUR
30	UDAIPUR
40	JAISALMER

Using Column Aliases:

```
SQL> SELECT DEPT_NO AS DEPT, LOCATION LOC  
FROM dept;
```

OR

```
SQL> SELECT DEPT_NO "DEPT", LOCATION  
"LOC" FROM dept;
```

DEPT	LOC
10	JAIPUR
20	JODHPUR
30	UDAIPUR
40	JAISALMER

Using Column Aliases:

```
SQL> SELECT 12*DEPT_NO AS DEPT, LOCATION  
        LOC FROM dept;
```

DEPT	LOC
120	JAIPUR
240	JODHPUR
360	UDAIPUR
480	JAISALMER

Concatenation Operator:

- Concatenates columns or character strings to other columns
- Is represented by two vertical bars (||)
- Creates a resultant column that is a character expression

Using Concatenation Operator:

```
SQL> SELECT DEPT_NO || LOCATION AS  
        CONCAT_EXMP FROM dept;
```

CONCAT_EXMP
10JAIPUR
20JODHPUR
30UDAIPUR
40JAISALMER

Literal Character Strings:

- A literal is a character, a number, or a date included in the SELECT list.
- Date and character literal values must be enclosed within single quotation marks.
- Each character string is output once for each row returned.

Using Literal Character Strings:

```
SQL> SELECT DEPT_NAME || ' ' || 'from' || ' ' ||
```

```
2. LOCATION AS "NAME_LOCATION"
```

```
3. FROM dept;
```

NAME_LOCATION
CSE from JAIPUR
ECE from JODHPUR
CCE from UDAIPUR
MME from JAISALMER

Lets have some duplicate rows first

DEPT_NO	DEPT_NAME	LOCATION
10	CSE	JAIPUR
10	CSE	JAIPUR
20	ECE	JODHPUR
20	ECE	JODHPUR
30	CCE	UDAIPUR
40	MME	JAISALMER

Duplicate Rows:

- The default display of queries is all rows, including duplicate rows.

```
SQL> SELECT DEPT_NO FROM dept;
```

DEPT_NO	DEPT_NAME	LOCATION
10	CSE	JAIPUR
10	CSE	JAIPUR
20	ECE	JODHPUR
20	ECE	JODHPUR
30	CCE	UDAIPUR
40	MME	JAISALMER

Eliminating Duplicate Rows:

- Eliminate duplicate rows by using the DISTINCT keyword in the SELECT clause.

```
SQL> SELECT DISTINCT DEPT_NO FROM dept;
```

DEPT_NO
10
20
30
40

SQL Statements Versus SQL*Plus Commands

SQL	SQL Plus
A language	An environment
ANSI standard	Oracle proprietary
Keyword cannot be abbreviated	Keyword can be abbreviated
Statements manipulate data and table definitions in the database	Commands do not allow manipulation of values in the database

Displaying Table Structure:

- Use the *iSQL*Plus* DESCRIBE command to display the structure of a table.

```
SQL> DESCRIBE dept;
```


Restricting and Sorting Data

Restricting and Sorting Data:

- Possibilities:
 - Limit the rows retrieved by a query
 - Sort the rows retrieved by a query

Limiting Rows Using a Selection:

DEPT_NO	DEPT_NAME	LOCATION
10	CSE	JAIPUR
20	ECE	JODHPUR
30	CCE	UDAIPUR
40	MME	JAISALMER
50	CCE	UDAIPUR
60	MME	JAISALMER



Displaying
information
about CCE
department only

DEPT_NO	DEPT_NAME	LOCATION
30	CCE	UDAIPUR
50	CCE	UDAIPUR

Limiting Rows Selected:

- Restrict the rows returned by using the WHERE clause.

```
SQL>SELECT [DISTINCT]  
2. {* | column | expression [alias],...}  
3. FROM    table  
4. [WHERE  condition(s)];
```

- The WHERE clause follows the FROM clause.

Using the WHERE Clause:

```
SQL> SELECT DEPT_NO, DEPT_NAME, LOCATION
```

```
2. FROM dept
```

```
3. WHERE DEPT_NAME='CCE';
```

DEPT_NO	DEPT_NAME	LOCATION
30	CCE	UDAIPUR

Character Strings and Dates

- Character strings and date values are enclosed in **single quotation marks**.
- Character values are **case sensitive**, and date values are **format sensitive**.
- The default date format is DD-MON-YY.

```
SQL> SELECT DEPT_NO, LOCATION
```

```
2. FROM dept
```

```
3. WHERE DEPT_NAME='CCE';
```

Comparison Operators:

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Using the Comparison Operators:

```
SQL> SELECT DEPT_NO, DEPT_NAME, LOCATION
```

```
2. FROM dept
```

```
3. WHERE DEPT_NO <= 40;
```

DEPT_NO	DEPT_NAME	LOCATION
10	CSE	JAIPUR
20	ECE	JODHPUR
30	CCE	UDAIPUR
40	MME	JAISALMER

Other Comparison Operators:

Operator	Meaning
BETWEEN ...AND...	Between two values (inclusive),
IN (set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Using the BETWEEN Operator:

- Use the BETWEEN condition to display rows based on a range of values.

```
SQL> SELECT DEPT_NO, DEPT_NAME, LOCATION
```

```
2. FROM dept
```

```
3. WHERE DEPT_NO BETWEEN 20 AND 50;
```

DEPT_NO	DEPT_NAME	LOCATION
20	ECE	JODHPUR
30	CCE	UDAIPUR
40	MME	JAISALMER
50	CCE	UDAIPUR

Both lower limit and upper limit values will be included

Using the IN Operator:

- Use the IN operator to test for values in a list

```
SQL> SELECT DEPT_NO, DEPT_NAME, LOCATION
```

```
2. FROM dept
```

```
3. WHERE DEPT_NO IN (20, 50);
```

DEPT_NO	DEPT_NAME	LOCATION
20	ECE	JODHPUR
50	CCE	UDAIPUR

Using the LIKE Operator:


- Use the LIKE condition to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers:
 - % denotes zero or many characters.
 - _ denotes one character.

Using the LIKE Operator:

```
SQL> SELECT LOCATION
```

```
2. FROM dept
```

```
3. WHERE LOCATION LIKE 'J%';
```



This means that first character should be J and after that anything

LOCATION
JAIPUR
JODHPUR
JAISALMER

Using the LIKE Operator:

- You can combine pattern-matching character

```
SQL> SELECT LOCATION
```

```
2. FROM dept
```

```
3. WHERE LOCATION LIKE '_A%';
```

LOCATION
JAIPUR
JAISALMER

This means that first letter can be anything then A and after that any substring

Using the LIKE Operator:

- For matching the last character we use %A

```
SQL> SELECT LOCATION
```

```
2. FROM dept
```

```
3. WHERE LOCATION LIKE '%R';
```

LOCATION
JAIPUR
JODHPUR
UDAIPUR
JAISALMER


Using the IS NULL Operator:

- Test for null values with the IS NULL operator.

```
SQL> SELECT DEPT_NO, DEPT_NAME
```

```
2. FROM DEPT
```

```
3. WHERE LOCATION IS NULL;
```



This will display the list of DEPT_NO, DEPT_NAME where the value of LOCATION is NULL

Logical Operators

Operator	Meaning
AND	Returns TRUE if <i>both</i> component conditions are true
OR	Returns TRUE if <i>either</i> component condition is true
NOT	Returns TRUE if the following condition is false

Using the AND Operator:

- AND requires both conditions to be TRUE.

```
SQL>SELECT DEPT_NO, DEPT_NAME, LOCATION  
2. FROM dept  
3. WHERE DEPT_NO >=10  
4. AND DEPT_NAME='CSE';
```

DEPT_NO	DEPT_NAME	LOCATION
10	CSE	JAIPUR

Using the OR Operator:

- AND requires either condition to be TRUE.

```
SQL>SELECT DEPT_NO, DEPT_NAME, LOCATION  
2. FROM dept  
3. WHERE DEPT_NO >20  
4. OR DEPT_NAME='CSE';
```

DEPT_NO	DEPT_NAME	LOCATION
30	CCE	UDAIPUR
40	MME	JAISALMER

Using the NOT Operator:

```
SQL>SELECT DEPT_NO, DEPT_NAME, LOCATION  
2. FROM dept  
3. WHERE LOCATION NOT IN ('JAISALMER',  
    'JAIPUR');
```

DEPT_NO	DEPT_NAME	LOCATION
20	ECE	JODHPUR
30	CCE	UDAIPUR
50	CCE	UDAIPUR

Rules of Precedence:

Order Evaluated	Operator
1	All comparison operators
2	NOT
3	AND
4	OR

- Override rules of precedence by using parenthesis.

Rules of Precedence:

SQL> SELECT DEPT_NO, DEPT_NAME, LOCATION

2. FROM dept

3. WHERE DEPT_NAME = 'MME'

4. OR DEPT_NAME = 'CCE'

5. AND DEPT_NO>10;

DEPT_NO	DEPT_NAME	LOCATION
30	CCE	UDAIPUR
40	MME	JAISALMER

Rules of Precedence:

- Use parentheses to force priority

```
SQL> SELECT DEPT_NO, DEPT_NAME, LOCATION
```

```
2. FROM dept
```

```
3. WHERE (DEPT_NAME = 'MME'
```

```
4. OR DEPT_NAME = 'CCE')
```

```
5. AND DEPT_NO>10;
```

Firstly braces will be operated then the result will be treated with outer operators

DEPT_NO	DEPT_NAME	LOCATION
30	CCE	UDAIPUR
40	MME	JAISALMER

ORDER BY Clause:

- Sort rows with the ORDER BY clause
 - **ASC**: ascending order, default
 - **DESC**: descending order
- The ORDER BY clause **comes last** in the SELECT statement.

```
SQL> SELECT DEPT_NO, DEPT_NAME, LOCATION  
2. FROM dept  
3. ORDER BY DEPT_NO;
```

ORDER BY Clause:

```
SQL> SELECT DEPT_NO, DEPT_NAME, LOCATION
```

```
2. FROM dept
```

```
3. ORDER BY DEPT_NO DESC;
```

DEPT_NO	DEPT_NAME	LOCATION
40	MME	JAISALMER
30	CCE	UDAIPUR
20	ECE	JODHPUR
10	CSE	JAIPUR

Sorting by Column Alias:

```
SQL> SELECT 12*DEPT_NO NUMBER,  
           DEPT_NAME, LOCATION
```

```
2. FROM dept
```

```
3. ORDER BY NUMBER DESC;
```

DEPT_NO	DEPT_NAME	LOCATION
40	MME	JAISALMER
30	CCE	UDAIPUR
20	ECE	JODHPUR
10	CSE	JAIPUR

Sorting by Multiple Columns:

SQL> SELECT DEPT_NO, DEPT_NAME, SALARY

2. FROM dept

3. ORDER BY DEPT_NO, SALARY DESC;

DEPT_NO	DEPT_NAME	SALARY
40	MME	5000
30	CCE	2500
20	ECE	2000
10	CSE	3000

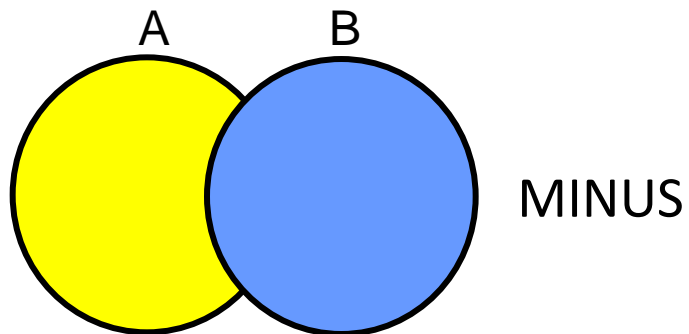
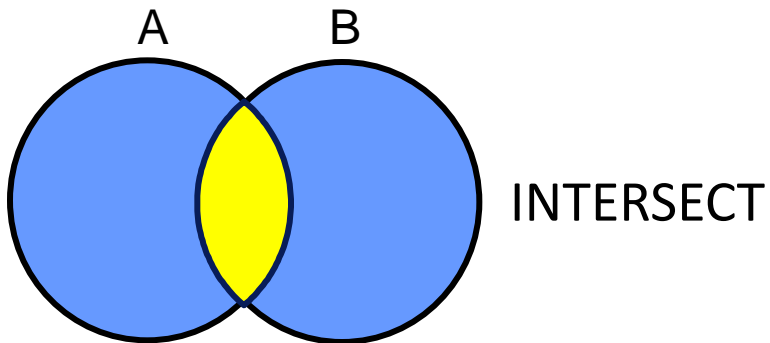
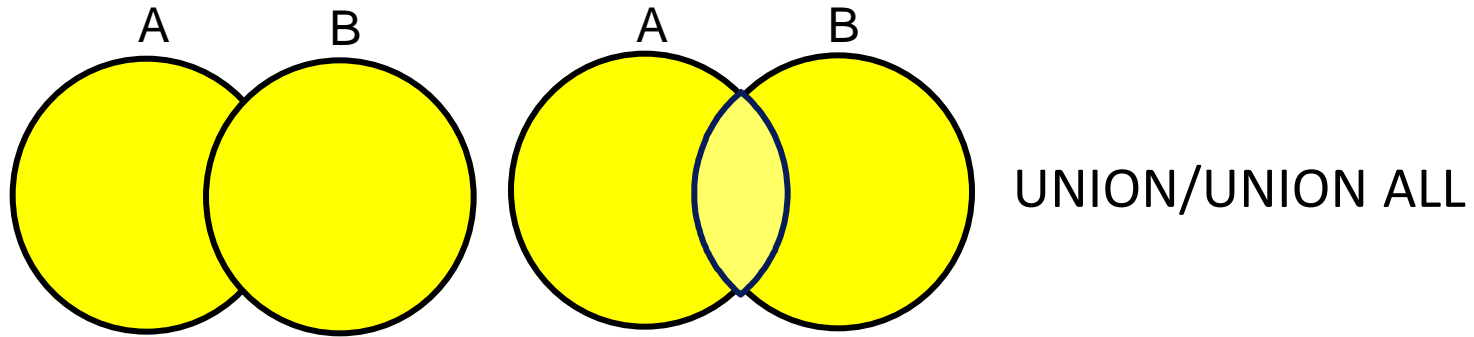
- You can sort by a column that is not in the SELECT list

Important :

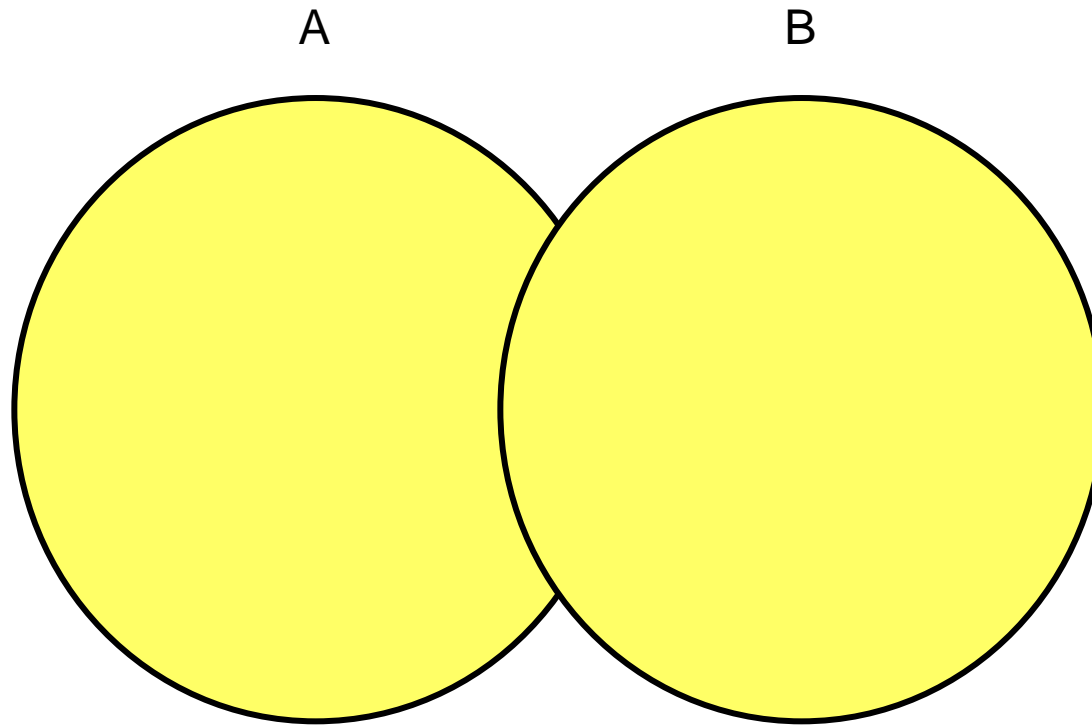
- While using WHERE clause no need of codes when comparing with any integer values.
- Only character or varchar needs codes and case should be kept in mind.

Using SET Operators

Set Operators



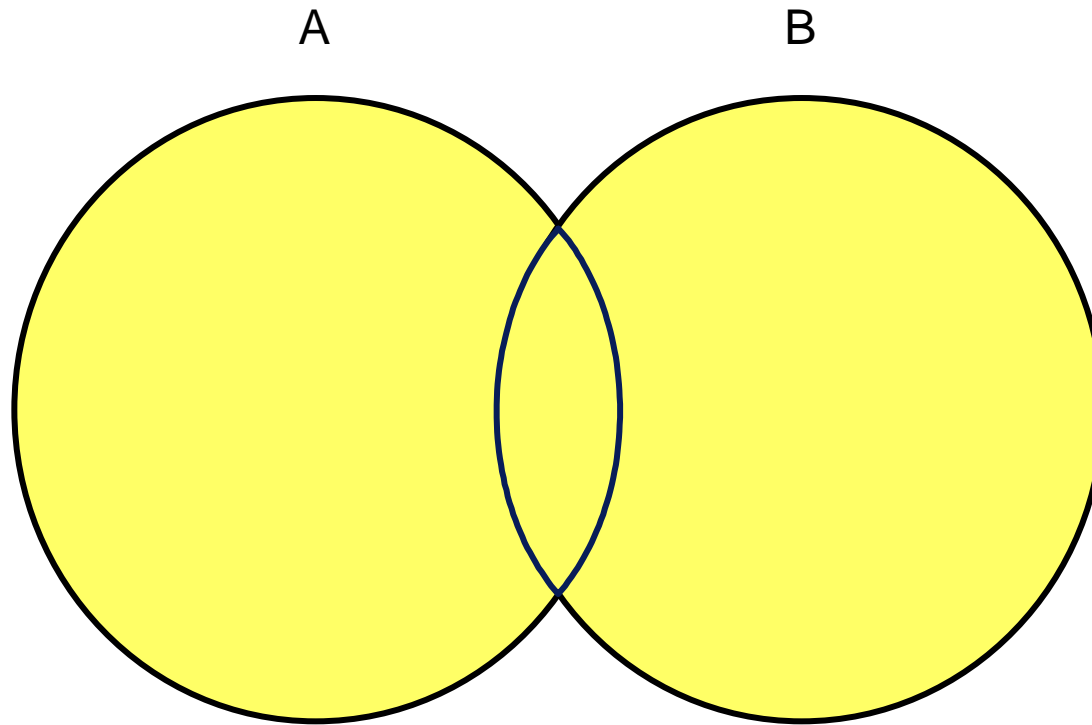
UNION Operator



The UNION operator returns all distinct rows

Using the UNION Operator

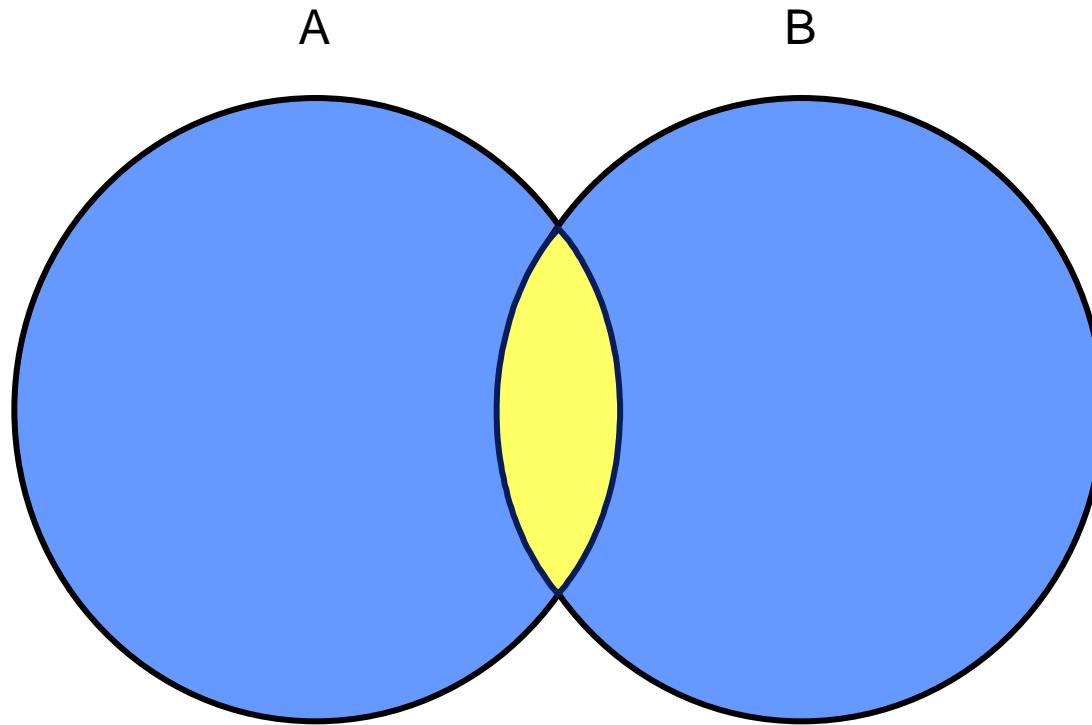
UNION ALL Operator



The `UNION ALL` operator returns results from both queries, including all duplications.

Using the UNION ALL Operator

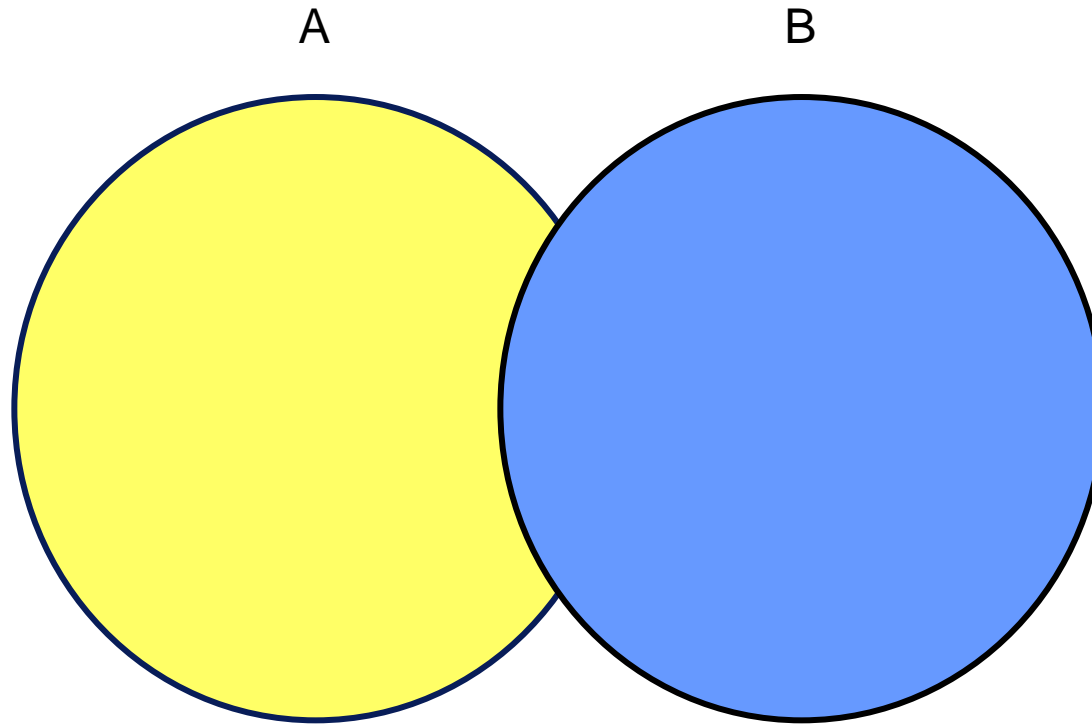
INTERSECT Operator



The `INTERSECT` operator returns rows that are common to both queries.

Using the INTERSECT Operator

MINUS Operator



The `MINUS` operator returns rows in the first query that are not present in the second query.

