Database Management Systems (CSE221)

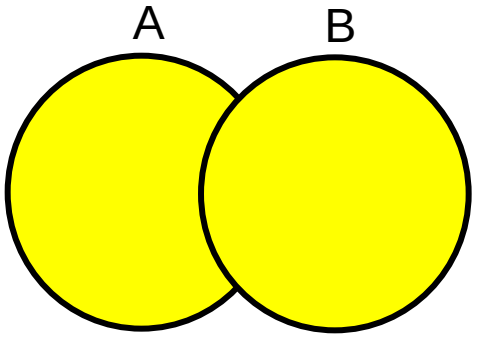
Vikas Bajpai

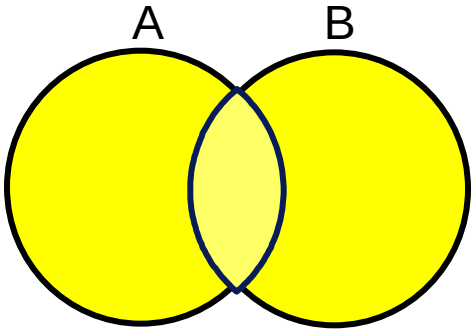
Acknowledgement:

- Khaleeq Ahmad Siddiqui, Sr. Software Engineer (OCP), IIT-Kanpur

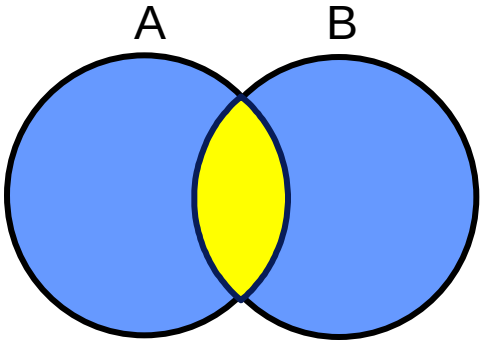
Using SET Operators

Set Operators

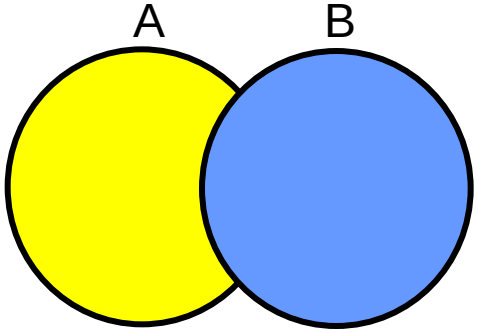




UNION/UNION ALL

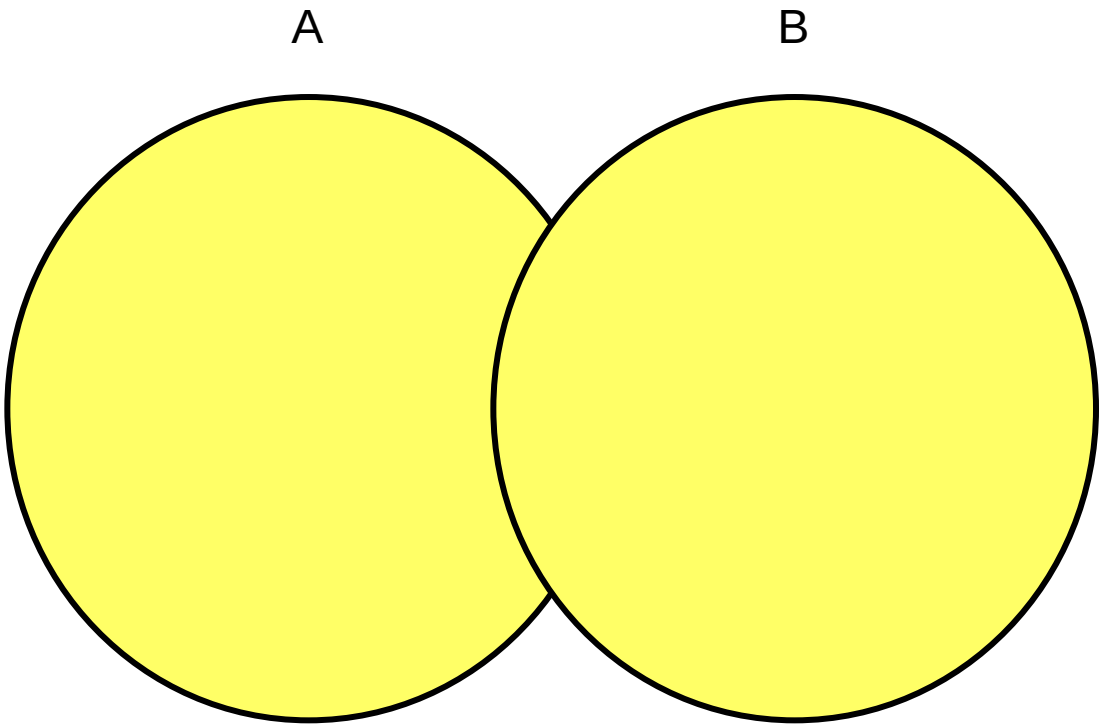


INTERSECT



MINUS

UNION Operator



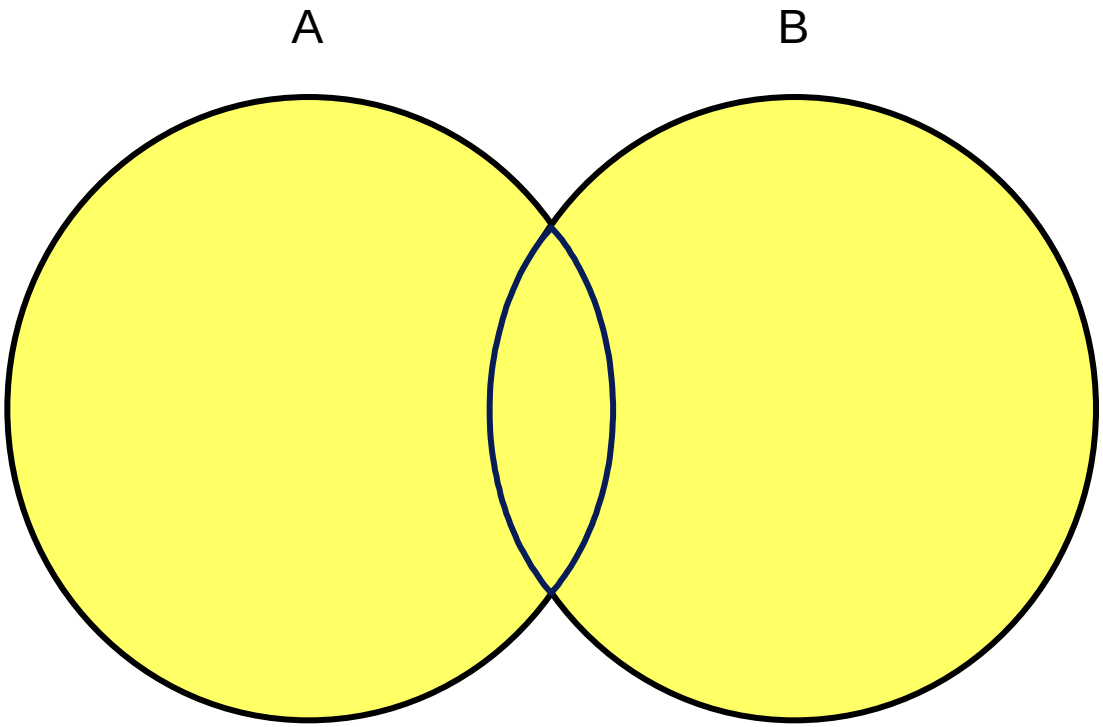
The UNION operator returns all distinct rows

Using the UNION Operator

```
SQL> SELECT DEPT_NO, DEPT_NAME
2. FROM dept
3. UNION
4. SELECT DEPT_NO, DEPT_NAME
5. FROM emp;
```

- Data type should be same while performing these operation
- At a time 255 tables can be operated using any of these operators.

UNION ALL Operator:

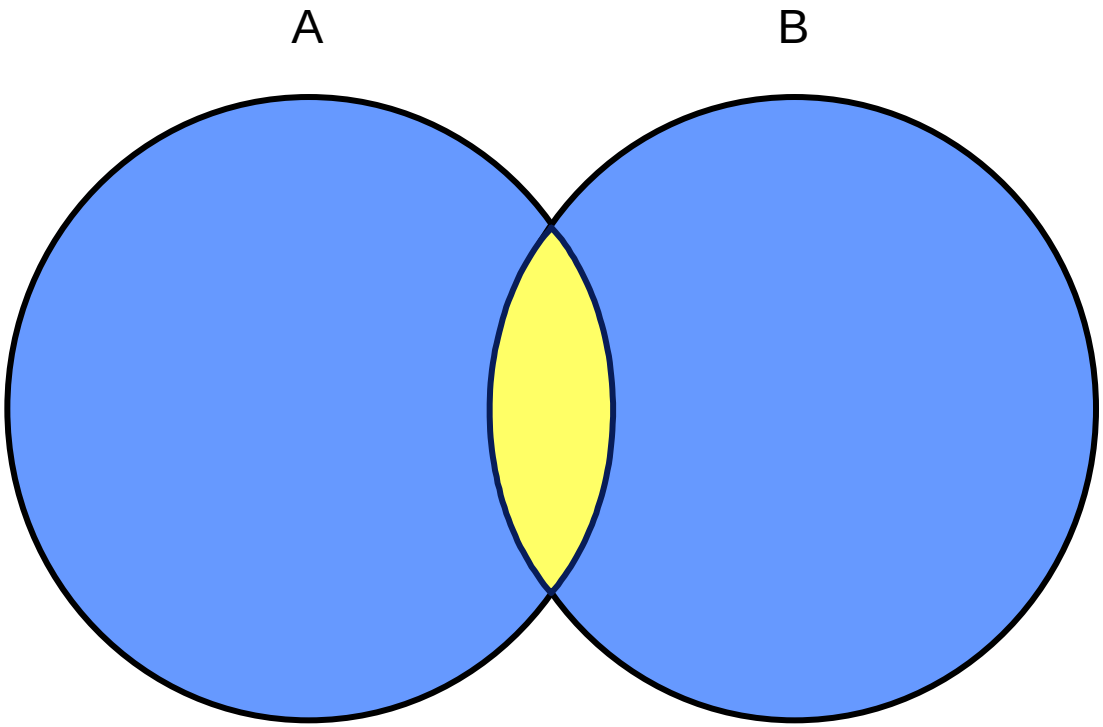


The UNION ALL operator returns all rows including all duplications.

Using the UNION ALL Operator

```
SQL> SELECT DEPT_NO, DEPT_NAME
2. FROM dept
3. UNION ALL
4. SELECT DEPT_NO, DEPT_NAME
5. FROM emp;
```

INTERSECT Operator:

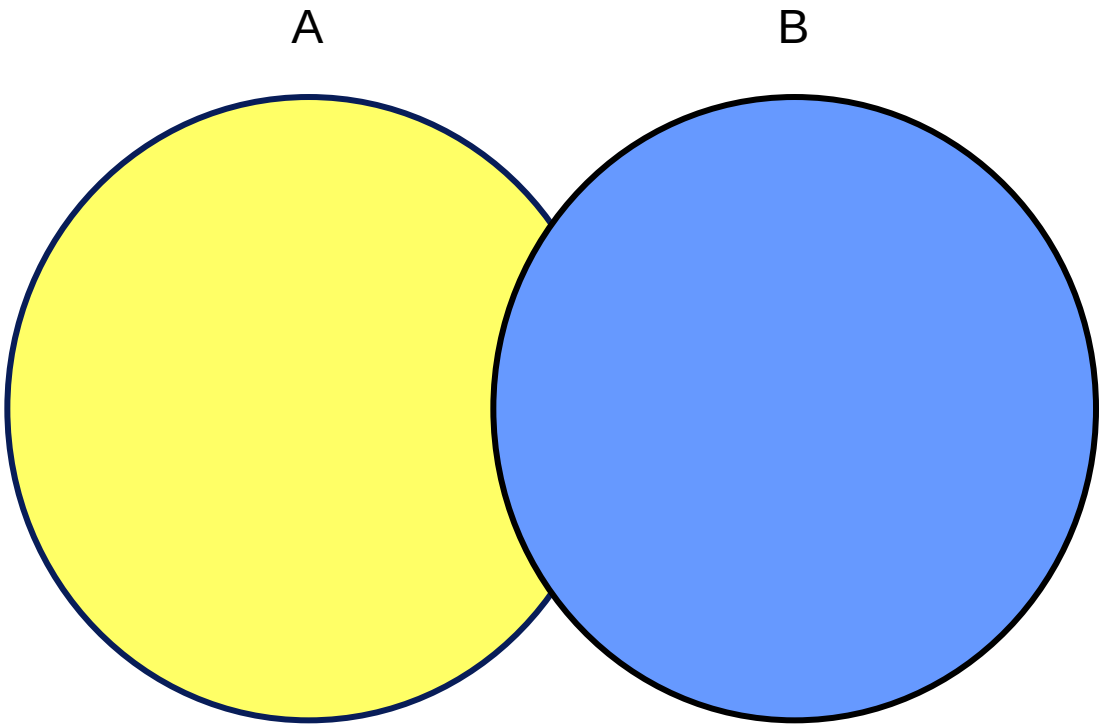


The INTERSECT operator returns rows that are common to both queries.

Using the INTERSECT Operator

```
SQL> SELECT DEPT_NO, DEPT_NAME
2. FROM dept
3. INTERSECT
4. SELECT DEPT_NO, DEPT_NAME
5. FROM emp;
```

MINUS Operator:



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

Using the MINUS Operator

```
SQL> SELECT DEPT_NO, DEPT_NAME
2. FROM dept
3. MINUS
4. SELECT DEPT_NO, DEPT_NAME
5. FROM emp;
```

SET Operator Rules:

- The expressions in the SELECT lists must match in number and datatype.
- Duplicate rows are automatically eliminated except in UNION ALL.
- Column names from the first query appear in the result.
- The output is sorted in ascending order by default except in UNION ALL.
- Parentheses can be used to alter the sequence of execution.

Matching the SELECT Statements

- Using the UNION operator, display the department ID, location, and hire date for all employees.

```
SELECT department_id, TO_NUMBER(null)
       location, hire_date
FROM   employees
UNION
SELECT department_id, location_id,  TO_DATE(null)
FROM   departments;
```

DEPARTMENT_ID	LOCATION	HIRE_DATE
10	1700	
10		17-SEP-87
20	1800	
20		17-FEB-96
...		
110	1700	
110		07-JUN-94
190	1700	
		24-MAY-99

27 rows selected.

Creating and Managing Tables, Views, Indexes, Sequence, Synonyms

Database Objects:

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Generates primary key values
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

Naming Conventions:

- Must begin with a letter
- Can be 1–30 characters long
- Must contain only A–Z, a–z, 0–9, _, $, and #
- Must not duplicate the name of another object owned by the same user
- Must not be an Oracle/DB2 Server reserved word

The CREATE TABLE Statement:

- You must have:
  - CREATE TABLE privilege
  - A storage area

```
CREATE TABLE [schema] table ( column
datatype [DEFAULT expr] [.....]);
```

- You specify:
  - Table name
  - Column name, column datatype and column size

Referencing Another User’s Tables:

- Tables belonging to other users are not in the user’s schema.
- You should use the owner’s name as a prefix to the table.

The DEFAULT Option:

- Specify a default value for a column during an insert.

```
... Hiredate DATE DEFAULT SYSDATE ...
```

- Legal values are literal value, expression or SQL function
- Illegal values are another column's name or pseudo-column.
- The default datatype must match the column datatype.

Datatypes:

Datatype	Description
VARCHAR(size)/ VARCHAR2(size)	Variable-length character data
CHAR(size)	Fixed-length character data
NUMBER(p,s)	Variable-length numeric data
DATE	Date and time values
LONG	Variable-length character data up to 2 gigabytes
CLOB	Single-byte character data up to 4 gigabytes
RAW and LONG RAW	Raw binary data
BLOB	Binary data up to 4 gigabytes
BFILE	Binary data stored in an external file; up to 4 gigabytes

Creating Tables:

```
SQL> CREATE TABLE dept
2            (DEPT_NO    NUMBER(2),
3            DEPT_NAME VARCHAR2(15),
4            LOCATION VARCHAR2(15));
```

Creating a table by using a Subquery:

- Create a table and insert rows by combining the CREATE TABLE statement and AS subquery option.

```
CREATE TABLE table_name [(column, column...)]
AS subquery
```

- Match the number of specified columns to the number of subquery columns.
- Define columns with column names and default values.

Creating a table by using a Subquery

```
SQL> CREATE TABLE dept_by_Subquery
2 AS
3   SELECT DEPT_NO, DEPT_NAME, LOCATION
4   FROM dept
5   WHERE DEPT_NO<=20;
```

Sub-query

The ALTER TABLE Statement:

- Use the ALTER TABLE statement to:
  - Add a new column
  - Modify an existing column
  - Define a default value for a new column

Adding a Column:

- Use ADD clause to add columns.

```
SQL> ALTER TABLE dept
2. ADD (NEW_COLUMN VARCHAR(10));
```

- The new column becomes the last column.

Guidelines for Adding a Column:

- You can add or modify columns, but you cannot drop them from a table.
- You cannot specify where the column is to appear.
- The new column becomes the last column
- The new column is initially null for all the rows.

Modifying a Column:

- Use MODIFY clause to modify columns

```
SQL> ALTER TABLE dept
2. MODIFY (DEPT_NAME VARCHAR(4));
```

Changing column's dataype using MODIFY clause

Dropping a Table:

- All data and structure in the table is deleted.
- Any pending transactions are committed.
- All indexes are dropped.
- You cannot roll back this statement.

```
SQL> DROP TABLE dept;
```

Changing the Name of the Object:

- To change the name of a table, view, sequence, or synonym, you execute the RENAME statement.

```
SQL> RENAME dept TO department;
```

- You must be the owner of the object.

Truncating a Table:

- The TRUNCATE TABLE statement:
  - Removes all rows from a table
  - Releases the storage space used by that table

```
SQL> TRUNCATE TABLE dept;
```

- You cannot roll back row removal when using TRUNCATE.
- Alternatively, you can remove rows by using the DELETE statement.

Adding Comments to a Table:

- You can add comments to a table or column by using the COMMENT statement.

```
SQL> COMMENT ON TABLE dept
2. IS Student Information';
```

- Comments can be viewed through the data dictionary views.
  - ALL_COL_COMMENTS
  - USER_COL_COMMENTS
  - ALL_TAB_COMMENTS
  - USER_TAB_COMMENTS

Dropping a Column:

- Use the DROP COLUMN clause to drop columns you no longer need from the table.

```
SQL> ALTER TABLE dept
2. DROP COLUMN LOCATION;
```

The INSERT Statement:

- Add new rows to a table by using the INSERT statement.
- Only one row is inserted at a time with this statement.

Inserting New Rows:

- Insert a new row containing values for each column.
- List values in the default order of the columns in the table.
- Optionally list the columns in the INSERT clause

```
SQL> INSERT INTO dept( DEPT_NO, DEPT_NAME, LOCATION)
2. VALUES (10, 'CSE', 'JAIPUR');
```

- Enclose character and date values within single quotation marks.

Creating Views, Sequences, Indexes, Private and Public Synonyms

What Is a View?

EMPLOYEES table

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALA
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	240
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	170
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	170
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	90
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	42
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	58
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	35
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	31
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	26
					JUL-96	ST_CLERK	25
					JAN-00	SA_MAN	105
					MAY-96	SA_REP	110
					MAR-98	SA_REP	86
176	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	70
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	44
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	130
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	60
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	120
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	83

20 rows selected.

EMPLOYEE_ID	LAST_NAME	SALARY
149	Zlotkey	10500
174	Abel	11000
176	Taylor	8600

View:

- View is a single table that is derived from other table(s). These other tables can be tables or previously defined views.
- A view doesn’t necessarily exist in physical form, it is considered as a Virtual table.
- View is used for displaying or hiding some data.

Why Use Views:

- To restrict database access
- To make complex queries easy
- To allow data independence
- To present different views of same data

Advantages of Views

- Views restrict access to the data because the view can display selected columns from the table.
- Views can be used to make simple queries to retrieve the results of complicated queries. For example, views can be used to query information from multiple tables without the user knowing how to write a join statement.
- Views provide data independence for ad hoc users and application programs. One view can be used to retrieve data from several tables.
- Views provide groups of users access to data according to their particular criteria.

Simple Views and Complex Views

Feature	Simple Views	Complex Views
Number of tables	One	One or more
Contain functions	No	Yes
Contain groups of data	No	Yes
DML operations through a view	Yes	Not always

Creating a View:

- You embed a subquery within the CREATE VIEW statement
- The subquery can containg complex SELECT syntax
- The subquery cannot contain an ORDER BY clause.

Creating a View

- Create the deptVIEW view

```
SQL> CREATE VIEW deptVIEW
2. AS SELECT DEPT_NO, LOCATION
3. FROM dept
4. WHERE DEPT_NO <=40;
```

- Describe the structure of the view by using the *i*SQL*Plus DESCRIBE command:

```
SQL> DESCRIBE deptVIEW
```

Guidelines for Creating a View:

- The subquery that defines a view can contain complex SELECT syntax, including joins, groups, and subqueries.
- If you do not specify a constraint name for a view created with the WITH CHECK OPTION, the system assigns a default name in the format SYS_C*n*.
- You can use the OR REPLACE option to change the definition of the view without dropping and re-creating it or regranting object privileges previously granted on it.

Creating a View

- Create a view by using column aliases in the subquery:

```
SQL> CREATE VIEW deptVIEW
2. AS SELECT DEPT_NO DEPARTMENT, LOCATION LOC
3. FROM dept
4. WHERE DEPT_NO <=40;
```

- Select the columns from this view by the given alias names

Retrieving Data from a View:

```
SQL> SELECT *
2. FROM deptVIEW;
```

Modifying a View:

- Modify a view by using CREATE OR REPLACE VIEW clause.
- Add an alias for each column name.

Modifying a View

```
SQL> CREATE OR REPLACE VIEW deptVIEW
2. (DEPARTMENT , LOC)
3. AS SELECT DEPT_NO, LOCATION
4. FROM dept
5. WHERE DEPT_NO <=40;
```

- Column aliases in the CREATE OR REPLACE VIEW clause are listed in the same order as the columns in the subquery.

Complex VieW

- If a view is made of using GROUP BY function then the view is called Complex View.

Creating a Complex View

- Create a complex view that contains group functions to display values from two tables:

```
CREATE VIEW        dept_sum_vu
  (name, minsal, maxsal, avgsal)
AS SELECT     d.department_name, e.salary,
              e.salary, e.salary
   FROM       employees e, departments d
   WHERE      e.department_id = d.department_id
   GROUP BY  d.department_name;
```

Rules for Performing DML Operations on a View

- You can usually perform DML operations on simple views.
- You cannot remove a row if the view contains the following:
  - Group functions
  - A `GROUP BY` clause
  - The `DISTINCT` keyword
  - The pseudocolumn `ROWNUM` keyword

Rules for Performing DML Operations on a View

- You cannot modify data in a view if it contains:
  - Group functions
  - A `GROUP BY` clause
  - The `DISTINCT` keyword
  - The pseudocolumn `ROWNUM` keyword
  - Columns defined by expressions

Rules for Performing DML Operations on a View

- You cannot add data through a view if the view includes:
  - Group functions
  - A `GROUP BY` clause
  - The `DISTINCT` keyword
  - The pseudocolumn `ROWNUM` keyword
  - Columns defined by expressions
  - `NOT NULL` columns in the base tables that are not selected by the view

Using the WITH CHECK OPTION Clause

- You can ensure that DML operations performed on the view stay in the domain of the view by using the WITH CHECK OPTION clause:

```
CREATE OR REPLACE VIEW empvu20
AS SELECT       *
  FROM     employees
  WHERE    department_id = 20
  WITH CHECK OPTION CONSTRAINT empvu20_ck ;
```

- Any attempt to change the department number for any row in the view fails because it violates the WITH CHECK OPTION constraint.

Denying DML Operations:

- You can ensure that no DML operations occur by adding the WITH READ ONLY option to your view definition.
- Any attempt to perform a DML operation on any row in the view results in an Oracle server error.

Denying DML Operations

```
CREATE OR REPLACE VIEW empvu10
   (employee_number, employee_name, job_title)
AS SELECT    employee_id, last_name, job_id
  FROM     employees
  WHERE    department_id = 10
  WITH READ ONLY ;
```

Removing a View

- You can remove a view without losing data because a view is based on underlying tables in the database.

```
SQL> DROP VIEW viewname;
```

```
SQL> DROP VIEW empvu80;
```

- Only the creator or a user with the DROP ANY VIEW privilege can remove a view.

SQL – Relations, Tables & Views

When we say Relation, it could be a Table or a View.

There are three kind of relations:

1. Stored relations → tables

   We sometimes use the term "base relation" or "base table"

2. Virtual relations → views
3. Temporary results

Database Objects: Sequences

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Generates primary key values
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

Sequences

- Can automatically generate unique numbers
- Is a sharable object
- Can be used to create a primary key value
- Replaces application code
- Speeds up the efficiency of accessing sequence values when cached in memory

CREATE SEQUENCE Statement: Syntax

- Define a sequence to generate sequential numbers automatically:

```
CREATE SEQUENCE sequence
    [INCREMENT BY n]
    [START WITH n]
    [{MAXVALUE n | NOMAXVALUE}]
    [{MINVALUE n | NOMINVALUE}]
    [{CYCLE | NOCYCLE}]
    [{CACHE n | NOCACHE}];
```

By default

Creating a Sequence

In the syntax:	
sequence	is the name of the sequence generator
INCREMENT BY *n*	specifies the interval between sequence numbers, where *n* is an integer (If this clause is omitted, the sequence increments by 1.)
START WITH *n*	specifies the first sequence number to be generated (If this clause is omitted, the sequence starts with 1.)
MAXVALUE *n*	specifies the maximum value the sequence can generate
NOMAXVALUE	specifies a maximum value of 10^27 for an ascending sequence and –1 for a descending sequence (This is the default option.)
MINVALUE *n*	specifies the minimum sequence value
NOMINVALUE	specifies a minimum value of 1 for an ascending sequence and –(10^26) for a descending sequence (This is the default option.)

Creating a Sequence

`CYCLE | NOCYCLE` specifies whether the sequence continues to generate values after reaching its maximum or minimum value (`NOCYCLE` is the default option.)

`CACHE` *n* `| NOCACHE` specifies how many values the Oracle server preallocates and keeps in memory (By default, the Oracle server caches 20 values.)

Creating a Sequence

- Create a sequence named DEPT_DEPTID_SEQ to be used for the primary key of the DEPARTMENTS table.
- Do not use the CYCLE option.

```
CREATE SEQUENCE dept_deptid_seq
        INCREMENT BY 10
        START WITH 120
        MAXVALUE 9999
        NOCACHE
        NOCYCLE;
```

NEXTVAL and CURRVAL Pseudocolumns

- NEXTVAL returns the next available sequence value. It returns a unique value every time it is referenced, even for different users.
- CURRVAL obtains the current sequence value.
- NEXTVAL must be issued for that sequence before CURRVAL contains a value.

NEXTVAL and CURRVAL Pseudocolumns

- **Rules for Using NEXTVAL and CURRVAL**
- You can use NEXTVAL and CURRVAL in the following contexts:
  - The SELECT list of a SELECT statement that is not part of a subquery
  - The SELECT list of a subquery in an INSERT statement
  - The VALUES clause of an INSERT statement
  - The SET clause of an UPDATE statement
- You cannot use NEXTVAL and CURRVAL in the following contexts:
  - The SELECT list of a view
  - A SELECT statement with the DISTINCT keyword
  - A SELECT statement with GROUP BY, HAVING, or ORDER BY clauses
  - A subquery in a SELECT, DELETE, or UPDATE statement
  - The DEFAULT expression in a CREATE TABLE or ALTER TABLE statement

Using a Sequence

- Insert a new department named “Support” in location ID 2500:

```
INSERT INTO departments(department_id,
        department_name, location_id)
VALUES     (dept_deptid_seq.NEXTVAL,
        'Support', 2500);
```

- View the current value for the DEPT_DEPTID_SEQ sequence:

```
SELECT    dept_deptid_seq.CURRVAL
FROM      dual;
```

Caching Sequence Values

- Caching sequence values in memory gives faster access to those values.
- Gaps in sequence values can occur when:
  - A rollback occurs
  - The system crashes
  - A sequence is used in another table

Modifying a Sequence

- Change the increment value, maximum value, minimum value, cycle option, or cache option:

```
ALTER SEQUENCE dept_deptid_seq
        INCREMENT BY 20
        MAXVALUE 999999
        NOCACHE
        NOCYCLE;
```

Guidelines for Modifying a Sequence

- You must be the owner or have the ALTER privilege for the sequence.
- Only future sequence numbers are affected.
- The sequence must be dropped and re-created to restart the sequence at a different number.
- Some validation is performed.
- To remove a sequence, use the DROP statement:

```
DROP SEQUENCE dept_deptid_seq;
```

Database Objects: Synonyms

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Generates primary key values
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

Synonyms

Simplify access to objects by creating a synonym (another name for an object). With synonyms, you can:

- Create an easier reference to a table that is owned by another user
- Shorten lengthy object names

CREATE [PUBLIC] SYNONYM *synonym*
FOR *object*;

Creating and Removing Synonyms

- Create a shortened name for the DEPT_SUM_VU view:

```
CREATE SYNONYM  d_sum
FOR  dept_sum_vu;
```

- Drop a synonym:

```
DROP SYNONYM d_sum;
```