Course Code: CS302

Database Management Systems

UNIT – II

Relational Model: Relational Data Model - Concept of relations, schema-instance distinction, keys, referential integrity and foreign keys, relational algebra operators, SQL - Introduction, data definition in SQL, table, Data Manipulation in SQL, Querying in SQL- aggregation functions group by and having clauses, embedded SQL, PL/SQL, Triggers. Introduction to NoSQL.

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Review Terms

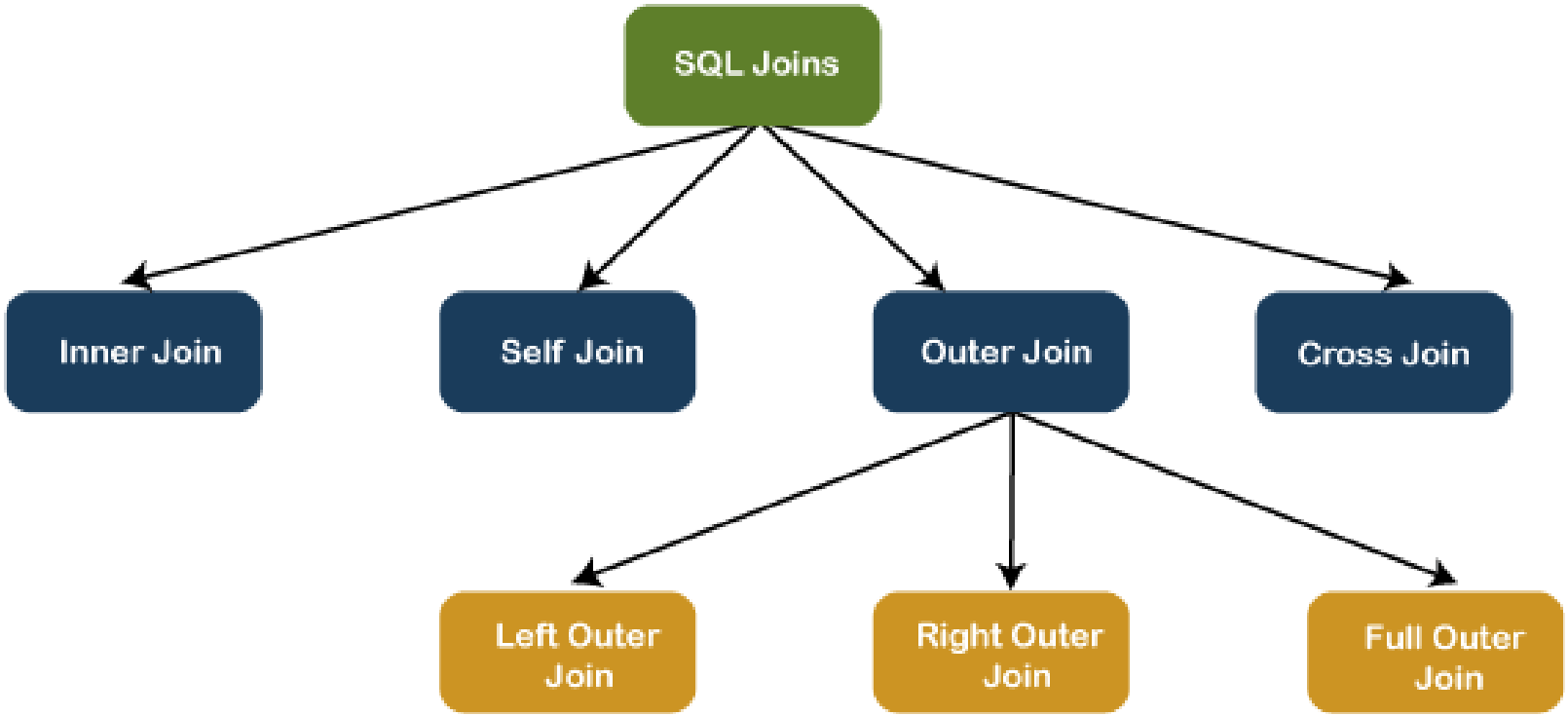
- Data-definition language
- Data-manipulation language
- Database schema
- Database instance
- Relation schema
- Relation instance
- Primary key
- Foreign key
  - Referencing relation
  - Referenced relation
- Null value
- Query language
- SQL query structure
  - **select** clause
  - **from** clause
  - **where** clause
- Multiset relational algebra
- **as** clause
- **order by** clause
- Table alias
- Correlation name (correlation variable, tuple variable)
- Set operations
  - **union**
  - **intersect**
  - **except**
- Aggregate functions
  - **avg, min, max, sum, count**
  - **group by**
  - **having**
- Nested subqueries
- Set comparisons
  - {<, <=, >, >=} { **some, all** }
  - **exists**
  - **unique**
- **lateral** clause
- **with** clause
- Scalar subquery
- Database modification
  - Delete
  - Insert
  - Update

Joins

- Join is used to combine rows from two or more tables, based on a related column between them.
- Cartesian product – combined information from multiple relations as in previous topics.
- Inner Join
  - Equi join
  - Non Equi join
  - Self join
- Outer Join
  - Left outer
  - Right outer
  - Full outer

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Types of Join



Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

- **(INNER) JOIN:** Returns records that have matching values in both tables

- **LEFT (OUTER) JOIN:** Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN:** Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN:** Returns all records when there is a match in either left or right table

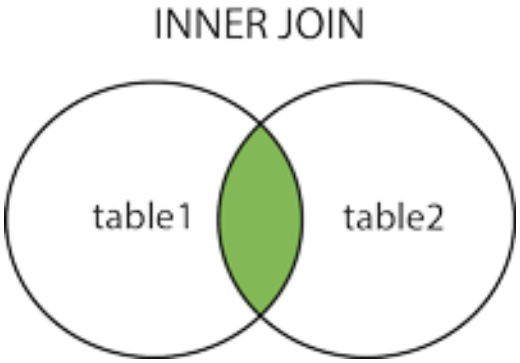
Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

INNER JOIN or SimpleJoin

- The INNER JOIN keyword selects records that have matching values in both tables.

Syntax

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```



Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

INNER JOIN or SimpleJoin

Example

Below is a selection from the "Orders" table:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

INNER JOIN or SimpleJoin

Example

And a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

INNER JOIN or SimpleJoin

Example

Example

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

INNER JOIN or SimpleJoin

Example - 2

Sample table: emp_mast

EMP_NO	EMP_NAME	JOB_NAME	MGR_ID	DEPT_NO
1234	Alex	Clerk	4567	15
2345	Jack	Consultant	3456	25
3456	Paul	Manager	1234	15
4567	Jenefer	Engineer	2345	45

Sample table: dep_mast

DEPT_NO	DEP_NAME	LOCATION
15	FINANCE	PARIS
25	MARKETING	LONDON
35	HR	DELHI

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

INNER JOIN or SimpleJoin

Example:

```
1  SELECT emp_no,emp_name,job_name,dep_name,location
2  FROM emp_mast
3  INNER JOIN dep_mast USING(dept_no);
```

Sample Output:

```
    EMP_NO EMP_NAME             JOB_NAME   DEP_NAME   LOCATION
---------- -------------------- ---------- ---------- ----------
      1234 Alex                 Clerk      FINANCE    PARIS
      2345 Jack                 Consultant MARKETING  LONDON
      3456 Paul                 Manager    FINANCE    PARIS
```

Outer Joins

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

INNER JOIN or SimpleJoin

- JOIN Three Tables

Example

```
SELECT Orders.OrderID, Customers.CustomerName, Shippers.ShipperName
FROM ((Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID)
INNER JOIN Shippers ON Orders.ShipperID = Shippers.ShipperID);
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Equijoins

- An equijoin is a join with a join condition containing an equality operator. This is represented by (=) sign. This join retrieves information by using equality condition.

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Equijoins
Example - 1

Sample table: emp_mast

EMP_NO	EMP_NAME	JOB_NAME	MGR_ID	DEPT_NO
1234	Alex	Clerk	4567	15
2345	Jack	Consultant	3456	25
3456	Paul	Manager	1234	15
4567	Jenefer	Engineer	2345	45

Sample table: dep_mast

DEPT_NO	DEP_NAME	LOCATION
15	FINANCE	PARIS
25	MARKETING	LONDON
35	HR	DELHI

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

E q u ijo in s

Example:

The following command shows that the two tables emp_mast and dep_mast are being joined based on an equality matching criteria i.e., "WHERE e.dept_no=d.dept_no".

```
1  SELECT emp_no,emp_name,job_name,dep_name
2  FROM emp_mast e,dep_mast d
3  WHERE e.dept_no=d.dept_no;
```

Sample Output:

```
    EMP_NO EMP_NAME             JOB_NAME   DEP_NAME
---------- -------------------- ---------- ----------
      1234 Alex                 Clerk      FINANCE
      2345 Jack                 Consultant MARKETING
      3456 Paul                 Manager    FINANCE
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Non-Equi Join

- An nonequi join is an inner join statement that uses an unequal operation (i.e.: <>, >, <, !=, BETWEEN, etc.) to match rows from different tables.

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Non-Equi Join

Example:

```
1  SELECT emp_no,emp_name,job_name,dep_name
2  FROM emp_mast e,dep_mast d
3  WHERE e.dept_no>d.dept_no;
```

Sample Output:

```
    EMP_NO EMP_NAME             JOB_NAME   DEP_NAME
---------- -------------------- ---------- ----------
      2345 Jack                 Consultant FINANCE
      4567 Jenefer              Engineer   FINANCE
      4567 Jenefer              Engineer   MARKETING
      4567 Jenefer              Engineer   HR
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Self Joins

- A self join is such a join in which a table is joined with itself.
- For example, when you require details about an employee and his manager (also an employee).

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Example - 1

Sample table: emp_mast

EMP_NO	EMP_NAME	JOB_NAME	MGR_ID	DEPT_NO
1234	Alex	Clerk	4567	15
2345	Jack	Consultant	3456	25
3456	Paul	Manager	1234	15
4567	Jenefer	Engineer	2345	45

Sample table: dep_mast

DEPT_NO	DEP_NAME	LOCATION
15	FINANCE	PARIS
25	MARKETING	LONDON
35	HR	DELHI

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Self Joins

Example:

```
1  SELECT a1.emp_no,a2.emp_name,a1.job_name,a2.dept_no
2  FROM emp_mast a1,emp_mast a2
3  WHERE a1.emp_no=a2.mgr_id;
```

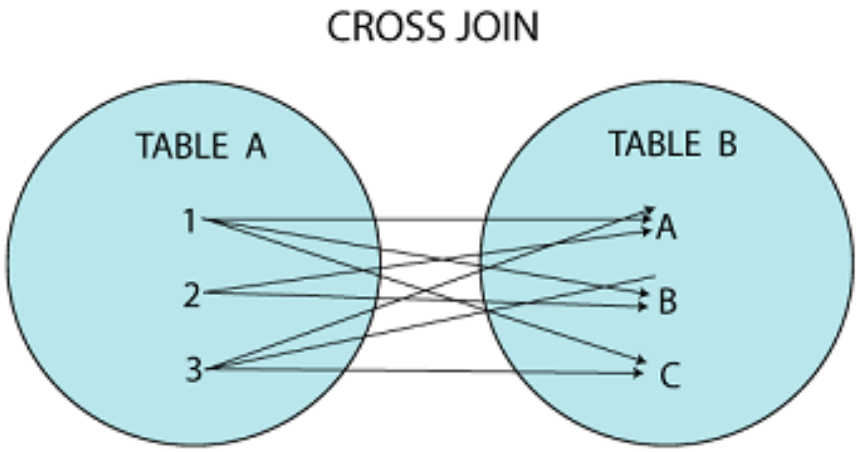
Sample Output:

```
    EMP_NO EMP_NAME            JOB_NAME      DEPT_NO
---------- ------------------- ---------- ----------
      4567 Alex                Engineer           15
      3456 Jack                Manager            25
      1234 Paul                Clerk              15
      2345 Jenefer             Consultant         45
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Cross Joins

- A Cross Join or Cartesian join or Cartesian product is a join of every row of one table to every row of another table.



Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Example:

```
1  SELECT emp_no,emp_name,job_name,dep_name,location
2  FROM emp_mast
3  CROSS JOIN dep_mast;
```

Sample Output:

```
    EMP_NO EMP_NAME             JOB_NAME   DEP_NAME   LOCATION
---------- -------------------- ---------- ---------- ---------
      1234 Alex                 Clerk      FINANCE    PARIS
      2345 Jack                 Consultant FINANCE    PARIS
      3456 Paul                 Manager    FINANCE    PARIS
      4567 Jenefer              Engineer   FINANCE    PARIS
      1234 Alex                 Clerk      MARKETING  LONDON
      2345 Jack                 Consultant MARKETING  LONDON
      3456 Paul                 Manager    MARKETING  LONDON
      4567 Jenefer              Engineer   MARKETING  LONDON
      1234 Alex                 Clerk      HR         DELHI
      2345 Jack                 Consultant HR         DELHI
      3456 Paul                 Manager    HR         DELHI
      4567 Jenefer              Engineer   HR         DELHI

12 rows selected.
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Outer Joins

- OUTER JOIN **returns all records from both tables** that satisfy the join condition.
- In other words, this join will not return only the matching record but also return all unmatched rows from one or both tables.

We can categories the OUTER JOIN further into three types:

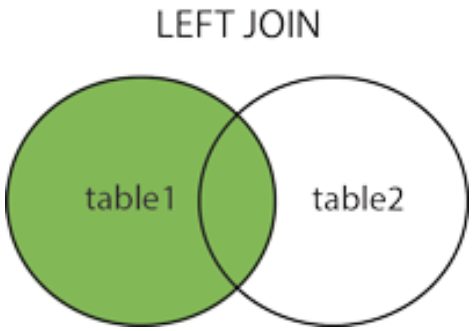
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Left Outer Join

- Left Outer Join returns all rows from the left (first) table specified in the ON condition and only those rows from the right (second) table where the join condition is met.
- Syntax

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```



Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Example:

```
1  SELECT emp_no,emp_name,job_name,dep_name,location
2  FROM emp_mast e LEFT OUTER JOIN dep_mast d
3  ON(e.dept_no=d.dept_no);
```

OR

```
1  SELECT emp_no,emp_name,job_name,dep_name,location
2  FROM emp_mast e, dep_mast d
3  WHERE e.dept_no=d.dept_no(+);
```

Sample Output:

```
    EMP_NO EMP_NAME            JOB_NAME   DEP_NAME   LOCATION
---------- ------------------- ---------- ---------- ----------
      3456 Paul                Manager    FINANCE    PARIS
      1234 Alex                Clerk      FINANCE    PARIS
      2345 Jack                Consultant MARKETING  LONDON
      4567 Jenefer             Engineer
```

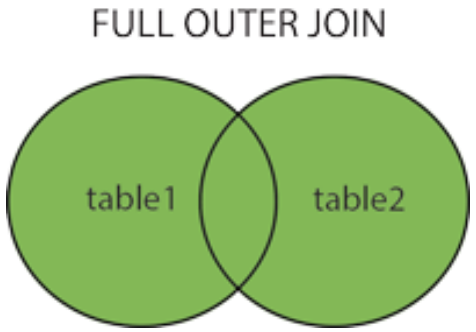
Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Right Outer Join

- The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

- Syntax

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```



Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Example:

```
1  SELECT emp_no,emp_name,job_name,dep_name,location
2  FROM emp_mast e RIGHT OUTER JOIN dep_mast d
3  ON(e.dept_no=d.dept_no);
```

OR

```
1  SELECT emp_no,emp_name,job_name,dep_name,location
2  FROM emp_mast e, dep_mast d
3  WHERE e.dept_no(+)=d.dept_no;
```

Sample Output:

```
    EMP_NO EMP_NAME             JOB_NAME   DEP_NAME   LOCATION
---------- -------------------- ---------- ---------- ---------
      1234 Alex                 Clerk      FINANCE    PARIS
      2345 Jack                 Consultant MARKETING  LONDON
      3456 Paul                 Manager    FINANCE    PARIS
                                           HR         DELHI
```

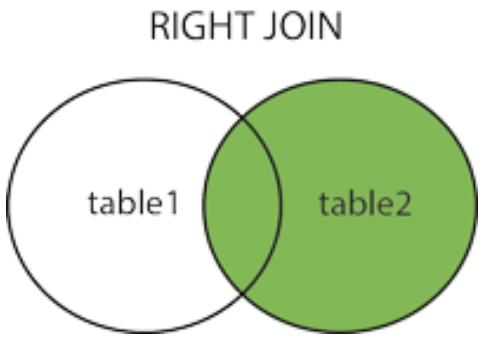
Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Full Outer Join

- The FULL OUTER JOIN **returns a result that includes all rows from both tables**.
- The columns of the right-hand table return NULL when no matching records are found in the left-hand table.
- And if no matching records are found in the right-hand table, the left-hand table column returns NULL.
- Tip: FULL OUTER JOIN and FULL JOIN are the same.

FULL OUTER JOIN Syntax

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition;
```



Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Example:

```
1  SELECT emp_no,emp_name,job_name,dep_name,location
2  FROM emp_mast e
3  FULL OUTER JOIN dep_mast d ON(e.dept_no=d.dept_no);
```

Sample Output:

```
    EMP_NO EMP_NAME             JOB_NAME   DEP_NAME   LOCATION
---------- -------------------- ---------- ---------- ----------
      1234 Alex                 Clerk      FINANCE    PARIS
      2345 Jack                 Consultant MARKETING  LONDON
      3456 Paul                 Manager    FINANCE    PARIS
      4567 Jenefer              Engineer
                                           HR         DELHI
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

This command gives the below result:

admission_no	first_name	last_name	course	amount_paid
3354	Luisa	Evans	Java	20000
2135	Paul	Ward	NULL	NULL
4321	Peter	Bennett	Python	18000
4213	Carlos	Patterson	NULL	NULL
5112	Rose	Huges	Machine Learning	30000
6113	Marielia	Simmons	NULL	NULL
7555	Antonio	Butler	Android	22000
8345	Diego	Cox	SQL	15000

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Natural Join

- A natural join is such a join that compares the common columns of both tables with each other.

Example:

```
1  SELECT  emp_no,emp_name,job_name,dep_name,location
2  FROM emp_mast
3  NATURAL JOIN dep_mast;
```

Sample Output:

```
    EMP_NO EMP_NAME             JOB_NAME   DEP_NAME   LOCATION
---------- -------------------- ---------- ---------- ----------
      1234 Alex                 Clerk      FINANCE    PARIS
      2345 Jack                 Consultant MARKETING  LONDON
      3456 Paul                 Manager    FINANCE    PARIS
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Antijoins

- An antijoin between two tables returns rows from the first table where no matches are found in the second table.
- Anti-Joins are only available when performing a NOT IN sub-query

Example:

```
1  SELECT * FROM emp_mast
2  WHERE dept_no NOT IN (
3  SELECT  dept_no FROM dep_mast);
```

Sample Output:

```
    EMP_NO EMP_NAME             JOB_NAME       MGR_ID    DEPT_NO
---------- -------------------- ---------- ---------- ----------
      4567 Jenefer              Engineer         2345         45
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Semijoins

- A semi-join is such a join where the EXISTS clause is used with a subquery.
- It can be called a semi-join because even if duplicate rows are returned in the subquery, only one set of matching values in the outer query is returned.

Example:

```
1  SELECT * FROM dep_mast a
2  WHERE EXISTS
3  (SELECT * FROM emp_mast b
4  WHERE a.dept_no = b.dept_no);
```

Sample Output:

```
   DEPT_NO DEP_NAME   LOCATION
---------- ---------- ----------
        15 FINANCE    PARIS
        25 MARKETING  LONDON
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

V I E W S ta te m e n t

- In SQL, a view is a virtual table based on the result-set of an SQL statement.
- A view contains rows and columns, just like a real table.
- The fields in a view are fields from one or more real tables in the database.
- You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.
- A view is created with the CREATE VIEW statement.

Ref:

https://www.tutorialspoint.com/sql/sql-rename-view.htm

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

C R E A T E V I E W

- A view is created with the CREATE VIEW statement.
- Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

C R E A T E V I E W

- A view is created with the CREATE VIEW statement.
- Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Example

```
CREATE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName
FROM Customers
WHERE Country = 'Brazil';
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

C R E A T E V I E W

- A view can be viewed using SELECT Query.

Example

SELECT * FROM [Brazil Customers];

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Updating a View

- A view can be updated with the CREATE OR REPLACE VIEW statement.

SQL CREATE OR REPLACE VIEW Syntax

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Updating a View

- The following SQL adds the "City" column to the "Brazil Customers" view:

Example

```
CREATE OR REPLACE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName, City
FROM Customers
WHERE Country = 'Brazil';
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Dropping a View

- A view is deleted with the DROP VIEW statement.

SQL DROP VIEW Syntax

```
DROP VIEW view_name;
```

The following SQL drops the "Brazil Customers" view:

Example

```
DROP VIEW [Brazil Customers];
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

sp_rename Stored Procedure

Syntax

Following is the basic syntax to rename a view in SQL –

```
EXEC sp_rename 'old_view_name', 'new_view_name'
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Example

Example
In this example, let us first try to create a table with the name **'CUSTOMERS'** which contains the personal details of customers including their name, age, address and salary etc. as shown below −

```
CREATE TABLE CUSTOMERS (
   ID INT NOT NULL,
   NAME VARCHAR (20) NOT NULL,
   AGE INT NOT NULL,
   ADDRESS CHAR (25),
   SALARY DECIMAL (18, 2),
   PRIMARY KEY (ID)
);
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Example

Now insert values into this table using the INSERT statement as follows −

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'kaushik', 23, 'Kota', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (7, 'Muffy', 24, 'Indore', 10000.00 );
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Example

The table will be created as follows –

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

Example

```
Create view CUSTOMERS_VIEW AS SELECT * FROM CUSTOMERS;
```

You can verify the contents of a view using the select query as shown below –

```
SELECT * from CUSTOMERS_VIEW;
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

E x a m p l e

```
EXEC sp_rename CUSTOMERS_VIEW, VIEW_CUSTOMERS;
```

Output

The result obtained is as shown below –

```
Caution: Changing any part of an object name could break scripts and stored
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe

```
SELECT * FROM VIEW_CUSTOMERS;
```

The view displayed is as follows –

```
+----+----------+-----+-----------+---------+
| ID | NAME     | AGE | ADDRESS   | SALARY  |
+----+----------+-----+-----------+---------+
|  1 | Ramesh   |  32 | Ahmedabad | 2000.00 |
|  2 | Khilan   |  25 | Delhi     | 1500.00 |
|  3 | kaushik  |  23 | Kota      | 2000.00 |
|  4 | Chaitali |  25 | Mumbai    | 6500.00 |
|  5 | Hardik   |  27 | Bhopal    | 8500.00 |
|  6 | Komal    |  22 | MP        | 4500.00 |
|  7 | Muffy    |  24 | Indore    |10000.00 |
+----+----------+-----+-----------+---------+
```

Copyright © 2016 Ramez Elmasri and Shamkant B. Navathe