ADVANCED SQL

TOPICS

- JOINS
- SQL CONSTRAINTS
- SQL SUBQUERIES
- SQL LOGICAL OPERATORS
- SQL RELATIONAL OPERATORS
- SQL LIKE CLAUSE
- SQL GROUP BY / HAVING

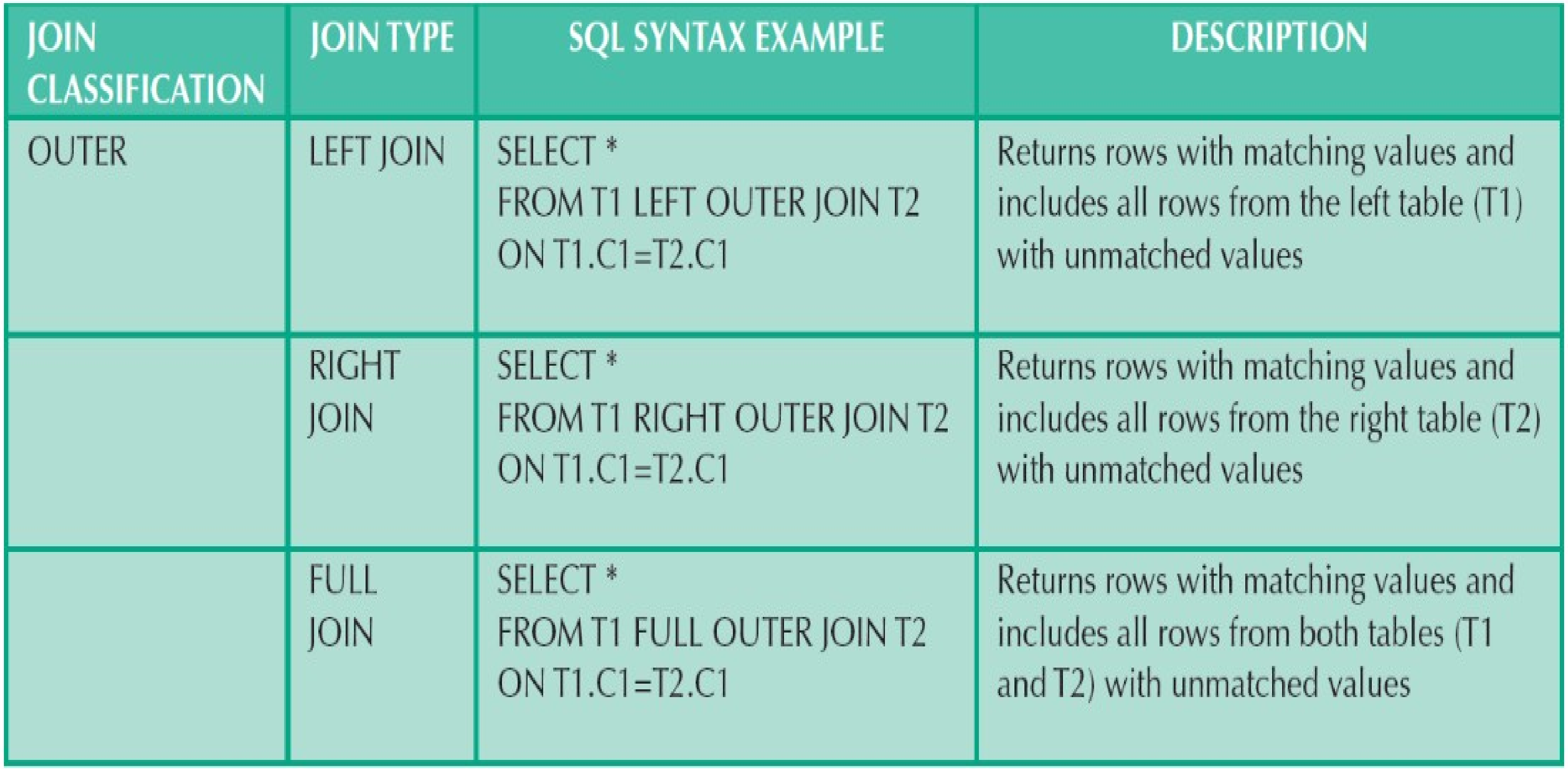
SQL JOINS

- combine rows from two or more tables, based on a related column between them.
- Types of Joins
  - Cartesian joins
  - Inner Joins
  - Left Joins
  - Right joins
  - Full Joins
  - Self joins

SQL JOINS

JOIN CLASSIFICATION	JOIN TYPE	SQL SYNTAX EXAMPLE	DESCRIPTION
CROSS	CROSS JOIN	SELECT * FROM T1, T2	Returns the Cartesian product of T1 and T2 (old style)
		SELECT * FROM T1 CROSS JOIN T2	Returns the Cartesian product of T1 and T2
INNER	Old-style JOIN	SELECT * FROM T1, T2 WHERE T1.C1=T2.C1	Returns only the rows that meet the join condition in the WHERE clause (old style); only rows with matching values are selected
	NATURAL JOIN	SELECT * FROM T1 NATURAL JOIN T2	Returns only the rows with matching values in the matching columns; the matching columns must have the same names and similar data types
	JOIN USING	SELECT * FROM T1 JOIN T2 USING (C1)	Returns only the rows with matching values in the columns indicated in the USING clause
	JOIN ON	SELECT * FROM T1 JOIN T2 ON T1.C1=T2.C1	Returns only the rows that meet the join condition indicated in the ON clause

S QL JOINS (Contd.)



JOIN CLASSIFICATION	JOIN TYPE	SQL SYNTAX EXAMPLE	DESCRIPTION

OUTER	LEFT JOIN	SELECT *
		FROM T1 LEFT OUTER JOIN T2
		ON T1.C1=T2.C1
			Returns rows with matching values and includes all rows from the left table (T1) with unmatched values

	RIGHT JOIN	SELECT *
		FROM T1 RIGHT OUTER JOIN T2
		ON T1.C1=T2.C1
			Returns rows with matching values and includes all rows from the right table (T2) with unmatched values

	FULL JOIN	SELECT *
		FROM T1 FULL OUTER JOIN T2
		ON T1.C1=T2.C1
			Returns rows with matching values and includes all rows from both tables (T1 and T2) with unmatched values

SQL Joins Contd.

- Self join example
  - Find the names of sellers who have same age

    -
      ```
      SELECT A.sname as seller1, B.sname as seller2,
      A.AGE as age
        FROM seller A, seller B
        WHERE A.sid <> B.sid AND A.AGE = B.AGE
      ```

SQL CONSTRAINTS

- NOT NULL
- DEFAULT
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK

SQL CONSTRAINTS (Contd.)

- NOT NULL
  - Ensures that a column cannot have NULL value.
- Example
  - CREATE TABLE seller(s_id int NOT NULL, s_name varchar(20) NOT NULL, age int NOT NULL, mob_number int, PRIMARY KEY (s_id))
  - ALTER TABLE seller MODIFY mob_number int NOT NULL;

SQL CONSTRAINTS (Contd.)

- DEFAULT
  - provides a default value to a column.
- Example
  - CREATE TABLE seller(s_id int NOT NULL, s_name varchar(20) NOT NULL, age int NOT NULL, mob_number int DEFAULT 9999888877, PRIMARY KEY (s_id))
  - ALTER TABLE seller MODIFY mob_number int DEFAULT 9999888877;
- To drop constraint
  - ALTER TABLE seller ALTER COLUMN mob_number DROP DEFAULT;

SQL CONSTRAINTS (Contd.)

- UNIQUE
  - prevents two records from having identical values in a particular column.
- Example
  - CREATE TABLE seller(s_id int NOT NULL, s_name varchar(20) NOT NULL, age int NOT NULL UNIQUE, mob_number int DEFAULT 9999888877, PRIMARY KEY (s_id))
  - ALTER TABLE seller MODIFY age int NOT NULL UNIQUE;
  - ALTER TABLE seller ADD CONSTRAINT myUniqueConstraint UNIQUE(AGE, mob_number);
- To drop constraint
  - ALTER TABLE seller DROP CONSTRAINT myUniqueConstraint;

SQL CONSTRAINTS (Contd.)

- PRIMARY KEY
  - A primary key is a field in a table which uniquely identifies each row/record
- Example
  - CREATE TABLE seller(s_id int NOT NULL, s_name varchar(20) NOT NULL, age int NOT NULL , mob_number int DEFAULT 9999888877, PRIMARY KEY (s_id))
  - ALTER TABLE seller ADD PRIMARY KEY (s_id);
- To drop constraint
  - ALTER TABLE seller DROP PRIMARY KEY;

SQL CONSTRAINTS (Contd.)

- FOREIGN KEY
  - a key used to link two tables together.
  - Foreign Key is a column or a combination of columns, whose values match a Primary Key in a different table.
- Example
  - CREATE TABLE stock(stock_id int NOT NULL, s_id int REFERENCES seller(s_id), b_id int REFERENCES buyer(b_id), v_id int REFERENCES vehicle(v_id), sold_date date, PRIMARY KEY (stock_id))
  - ALTER TABLE stock ADD FOREIGN KEY (s_id) REFERENCES seller(s_id);
- To drop constraint
  - ALTER TABLE stock DROP FOREIGN KEY;

SQL CONSTRAINTS (Contd.)

- CHECK
  - enables a condition to check the value being entered into a record.
- Example
  - CREATE TABLE seller(s_id int NOT NULL, s_name varchar(20) NOT NULL, age int NOT NULL CHECK (age >=18), mob_number int DEFAULT 9999888877, PRIMARY KEY (s_id))
  - ALTER TABLE seller MODIFY age int NOT NULL CHECK (age >= 18)
  - ALTER TABLE seller ADD CONSTRAINT myCheckConstraint CHECK(age >= 18);
- To drop constraint
  - ALTER TABLE stock DROP CONSTRAINT myCheckConstraint;

SUB QUERIES

- Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN etc.

- Find the id and name of seller with highest age.

1)SELECT * FROM SELLER

2)SELECT MAX(age) FROM seller

3)SELECT sid,sname FROM seller WHERE age = 60

OR

4)SELECT sid, sname FROM seller WHERE age = (SELECT MAX(age) FROM seller)

CORRELATED QUERIES

- Each subquery is executed once for every row of the outer query.
- Find the name of seller with 2nd highest age.
  - SELECT sname FROM seller s1 WHERE 2= (SELECT COUNT(s2.age) FROM seller s2 WHERE s2.AGE >= s1.AGE)

SQL LOGICAL OPERATORS

- **ALL**
  - The ALL operator is used to compare a value to all values in another value set.
  - `SELECT first_name FROM employees WHERE salary > ALL(SELECT salary FROM employees WHERE department_id = 2);`
- **ANY**
  - The ANY operator is used to compare a value to any applicable value in the list according to the condition.
  - `SELECT sid FROM seller WHERE sid = any (SELECT sid FROM stock)`
- **BETWEEN**
  - The BETWEEN operator is used to search for values that are within a set of values, given the minimum value and the maximum value.
  - `SELECT * FROM stock WHERE date BETWEEN '2020-10-01' AND '2020-10-05'`

SQL LOGICAL OPERATORS (Contd.)

- **EXISTS**
  - The EXISTS operator is used to search for the presence of a row in a specified table that meets certain criteria.
- **IN**

  The IN operator is used to compare a value to a list of literal values that have been specified.
- **LIKE**

  The LIKE operator is used to compare a value to similar values using wildcard operators.
- **UNIQUE**

  The UNIQUE operator searches every row of a specified table for uniqueness (no duplicates).
- **IS NULL**

  The NULL operator is used to compare a value with a NULL value.

SQL LIKE Clause

- Used to compare a value to similar values using wildcard operators.
  - The percent sign (%)
  - The underscore (_)

Examples

- WHERE SALARY LIKE '200%'
  Finds any values that start with 200

- WHERE SALARY LIKE '%200%'
  Finds any values that have 200 in any position

- WHERE SALARY LIKE '_00%'
  Finds any values that have 00 in the second and third positions

SQL LIKE Clause

- WHERE SALARY LIKE '2_%_%'
  Finds any values that start with 2 and are at least 3 characters in length
- WHERE SALARY LIKE '%2'
  Finds any values that end with 2
- WHERE SALARY LIKE '_2%3'
  Finds any values that have a 2 in the second position and end with a 3
- WHERE SALARY LIKE '2___3'
  Finds any values in a five-digit number that start with 2 and end with 3

SQL RELATIONAL OPERATORS

- **Union compatible** – Number of attributes must be same and their corresponding data types are alike.
- **UNION**
  - Combines rows from 2 queries
  - avoid duplicates
  - SELECT column_name(s) FROM table1 UNION SELECT column_name(s) FROM table2;
- **INTERSECT**
- **EXCEPT/MINUS**
- **UNION ALL**
  - Retain duplicates.

SQL GROUP BY / HAVING

- Find the seller names and number of vehicles they sold.

  -
    ```
    SELECT sname,count(stock.sid) as '#sold'
     FROM seller,stock
      WHERE seller.sid = stock.sid
      GROUP BY stock.sid
    ```
- The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.
  - Find the seller names who sold more than one vehicle.
  - SELECT sname,count(stock.sid) as '#sold' FROM seller,stock WHERE seller.sid = stock.sid GROUP BY stock.sid HAVING count(stock.sid) > 1

Thank you!