Chapter 8: Oracle SQL Joins

🔗 Purpose of Joins in SQL

- A **join** is a SELECT statement that retrieves and merges data from two or more tables based on related columns.
- SQL databases are **relational** — tables relate via shared data elements (keys).

🔍 Types of Joins

1. Equijoins

- Use equality (=) to match rows between tables
- Most common type

2. Non-Equijoins

- Use non-equality operators (<, >, BETWEEN, etc.)
- Useful for range-based or conditional relationships

3. Inner Joins

- Return only rows with matching values in both/all tables

4. Outer Joins

- Include unmatched rows from one or more tables:
  - **LEFT OUTER JOIN**: All rows from the left table + matched rows from the right
  - **RIGHT OUTER JOIN**: All rows from the right + matched from the left
  - **FULL OUTER JOIN**: All rows from both tables, matched where possible

5. Natural Joins

- Automatically joins tables on columns with the same names and compatible data types
- No explicit condition needed

6. Self-Joins

- A table joins to itself using an alias
- Useful for hierarchical or recursive relationships

✅ JOIN Types and Syntax for Oracle SQL

1. Inner Joins

- Combines rows from two tables **only when there's a match** in both tables
- Use INNER JOIN or just JOIN (they're equivalent)

Syntax:

```sql
SELECT SHIP_ID, SHIP_NAME, PORT_NAME
FROM SHIPS INNER JOIN PORTS
ON HOME_PORT_ID = PORT_ID;
```

Can add filters:

```sql
WHERE PORT_NAME = 'Charleston'
```

2. Old Inner Join Syntax (Oracle-specific, not ANSI-compliant)

Uses commas and WHERE clause instead of JOIN and ON:

```sql
SELECT S.SHIP_ID, S.SHIP_NAME, P.PORT_NAME
FROM SHIPS S, PORTS P
WHERE S.HOME_PORT_ID = P.PORT_ID;
```

3. Table Aliases

- Used to shorten table names and disambiguate columns
- Scope is limited to the SQL statement
- Required when column names are ambiguous or repeated in joins

Two ways:

- **Full table prefix:**

  ```sql
  EMPLOYEES.EMPLOYEE_ID
  ```

- **Alias:**

```sql
FROM EMPLOYEES EM JOIN ADDRESSES AD
ON EM.EMPLOYEE_ID = AD.EMPLOYEE_ID
```

4. Natural Joins

- Automatically joins tables using **columns with identical names**
- **No ON clause** or table prefixes allowed for join columns

Syntax:

```sql
SELECT EMPLOYEE_ID, LAST_NAME, STREET_ADDRESS
FROM EMPLOYEES NATURAL JOIN ADDRESSES;
```

Supports:

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

Source precedence for ambiguous columns:

- Inner/Left Join → Left table
- Right Join → Right table
- Full Join → Merged from both

5. USING Clause

- Simplifies join when **identical column names** are involved
- No table prefixes for USING columns

Syntax:

```sql
SELECT EMPLOYEE_ID, LAST_NAME, STREET_ADDRESS
FROM EMPLOYEES LEFT JOIN ADDRESSES
USING (EMPLOYEE_ID);
```

Multiple columns:

```sql
USING (EMPLOYEE_ID, OFFICE_NAME);
```

6. Multitable Joins

- You can join **3+ tables** in a single query
- Add each table with JOIN and an ON condition

Syntax:

```sql
SELECT P.PORT_NAME, S.SHIP_NAME, SC.ROOM_NUMBER
FROM PORTS P JOIN SHIPS S ON P.PORT_ID = S.HOME_PORT_ID
JOIN SHIP_CABINS SC ON S.SHIP_ID = SC.SHIP_ID;
```

7. Non-Equijoins

- Joins **without equality condition**—use ranges or comparisons instead

Example with BETWEEN:

```sql
SELECT S.SCORE_ID, S.TEST_SCORE, G.GRADE
FROM SCORES S JOIN GRADING G
ON S.TEST_SCORE BETWEEN G.SCORE_MIN AND G.SCORE_MAX;
```

Can also use <, >, <=, >=, and logical operators like AND, OR.

Certification Objective 8.03: Self-Join

Definition

A **self-join** is when a table is joined to itself. This is typically done by comparing one column's value to another column in the same table.

Types

Self-joins can be:

- **INNER JOIN** or **OUTER JOIN**

- **EQUIJOIN** or **NON-EQUIJOIN**

Example Table: POSITIONS

Key columns:

- POSITION_ID
- POSITION
- REPORTS_TO (references another POSITION_ID)

Sample data:

POSITION_ID	POSITION	REPORTS_TO
1	Captain	(null)
2	Director	1
3	Manager	2
4	Crew Chief	2
5	Crew	4

REPORTS_TO Explanation

- It links an employee's position to their supervisor's POSITION_ID

Self-Referencing Foreign Key

(Optional, but recommended)

```sql
ALTER TABLE POSITIONS
ADD CONSTRAINT FK_PO_PO FOREIGN KEY (REPORTS_TO)
REFERENCES POSITIONS (POSITION_ID);
```

Self-Join Syntax

Steps:

1. Reference the same table twice with aliases
2. Use join condition A.REPORTS_TO = B.POSITION_ID
3. Use an **OUTER JOIN** to include positions with no boss

Example Query:

```sql
SELECT A.POSITION_ID, A.POSITION, B.POSITION AS BOSS
FROM POSITIONS A
LEFT OUTER JOIN POSITIONS B
ON A.REPORTS_TO = B.POSITION_ID
ORDER BY A.POSITION_ID;
```

Output:

POSITION_ID	POSITION	BOSS
1	Captain	(null)
2	Director	Captain
3	Manager	Director
4	Crew Chief	Director
5	Crew	Crew Chief

Exam Tip

Know all join types and syntax:

- NATURAL
- USING
- INNER, LEFT OUTER, etc.

🔷 Outer Joins Overview

- **Outer joins** return all matching rows + unmatched rows from one/both tables
- Three types: **LEFT**, **RIGHT**, and **FULL OUTER JOIN**

🔸 LEFT OUTER JOIN

Returns:

- All rows from the **left** table
- Matched rows from the **right** table
- NULLs for unmatched rows on the right

Syntax:

```sql
SELECT SHIP_ID, SHIP_NAME, PORT_NAME
FROM SHIPS LEFT [OUTER] JOIN PORTS
ON HOME_PORT_ID = PORT_ID;
```

- OUTER is optional
- LEFT JOIN = LEFT OUTER JOIN

◆ RIGHT OUTER JOIN

Returns:

- All rows from the **right** table
- Matched rows from the **left** table
- NULLs for unmatched rows on the left

Syntax:

```sql
SELECT SHIP_ID, SHIP_NAME, PORT_NAME
FROM SHIPS RIGHT [OUTER] JOIN PORTS
ON HOME_PORT_ID = PORT_ID;
```

- RIGHT JOIN = RIGHT OUTER JOIN

◆ FULL OUTER JOIN

Returns:

- All rows from both tables
- Matched rows merged
- NULLs where no match exists on either side

Syntax:

```sql
SELECT SHIP_ID, SHIP_NAME, PORT_NAME
FROM SHIPS FULL [OUTER] JOIN PORTS
ON HOME_PORT_ID = PORT_ID;
```

🔸 Legacy Oracle Outer Join Syntax (Not on Exam)

Uses + to represent outer joins.

LEFT OUTER JOIN:

```sql
SELECT SHIP_ID, SHIP_NAME, PORT_NAME
FROM SHIPS, PORTS
WHERE HOME_PORT_ID = PORT_ID(+);
```

RIGHT OUTER JOIN:

```sql
SELECT SHIP_ID, SHIP_NAME, PORT_NAME
FROM SHIPS, PORTS
WHERE HOME_PORT_ID(+) = PORT_ID;
```

FULL OUTER JOIN (simulated with UNION):

```sql
SELECT SHIP_ID, SHIP_NAME, PORT_NAME
FROM SHIPS, PORTS
WHERE HOME_PORT_ID = PORT_ID(+)
UNION
SELECT SHIP_ID, SHIP_NAME, PORT_NAME
FROM SHIPS, PORTS
WHERE HOME_PORT_ID(+) = PORT_ID;
```

⚠️ Not ANSI standard. Not on the Oracle SQL exam. Use ANSI-style JOIN syntax instead.

✅ Exam Tips

- Know **LEFT JOIN**, **RIGHT JOIN**, **FULL OUTER JOIN** syntax
- Understand **what rows are returned** in each case
- Remember OUTER is optional (e.g., LEFT JOIN = LEFT OUTER JOIN)
- Avoid legacy + syntax—it **won't be tested**