SQL - Oracle Topics

Important Topics

By Dhandapani Yedappalli Krishnamurthi Sep 24, 2025

Oracle SQL Important Topics

---

1. Common Table Expressions (CTEs)

Description:
A CTE is a temporary result set defined with the `WITH` clause that can be referenced in a subsequent query. It improves readability and helps organize complex queries.

Example:

```
WITH SalesCTE AS (
    SELECT customer_id, SUM(amount) AS total_sales
    FROM sales
    GROUP BY customer_id
)
SELECT c.customer_name, s.total_sales
FROM customers c
JOIN SalesCTE s ON c.customer_id = s.customer_id;
```

Explanation:

- `WITH SalesCTE` creates a temporary named result set.
- Inside, we calculate `total_sales` per customer.

- The main query then joins the CTE with `customers` to show customer names with sales totals.

Illustration:

```
[Sales Table] → (aggregate sales per customer) →
[SalesCTE]
       |
[Customers Table] → (join with SalesCTE) → Final
Result
```

---

2. Stored Procedures

Description:
A Stored Procedure is a block of PL/SQL code stored in the database, which can be reused and executed with parameters.

Example:

```
CREATE OR REPLACE PROCEDURE GetCustomerSales
(p_customer_id IN NUMBER) AS
    v_total_sales NUMBER;
BEGIN
    SELECT SUM(amount) INTO v_total_sales
    FROM sales
    WHERE customer_id = p_customer_id;

    DBMS_OUTPUT.PUT_LINE('Total Sales: ' ||
v_total_sales);
END;
```

Explanation:

- `p_customer_id` is an input parameter.

- The procedure calculates total sales for that customer.
- `DBMS_OUTPUT.PUT_LINE` prints the result.

Illustration:

```
Client → Calls Procedure (customer_id) → Runs Query
→ Returns Output
```

---

3. Triggers

Description:
A Trigger is a PL/SQL block that automatically executes when a specified event (INSERT, UPDATE, DELETE) occurs on a table.

Example:

```
CREATE OR REPLACE TRIGGER trg_audit_sales
AFTER INSERT ON sales
FOR EACH ROW
BEGIN
    INSERT INTO sales_audit (sale_id, action_date,
action)
    VALUES (:NEW.sale_id, SYSDATE, 'INSERT');
END;
```

Explanation:

- This trigger fires **after every insert** on `sales`.
- It inserts a log record into `sales_audit`.
- `:NEW` refers to new row values after insert.

Illustration:

```
INSERT INTO Sales → Trigger Fires → Log Entry in
Sales_Audit
```

---

4. Window Functions

Description:
Window (Analytic) functions perform calculations across a set of rows related to the current row (without collapsing results like GROUP BY).

Example:

```
SELECT employee_id, department_id, salary,
       RANK() OVER (PARTITION BY department_id ORDER
BY salary DESC) AS dept_rank
FROM employees;
```

Explanation:

- `RANK()` assigns a rank within each department (`PARTITION BY department_id`).
- `ORDER BY salary DESC` ranks employees by salary, highest first.
- Unlike GROUP BY, all rows are kept.

Illustration:

```
Dept A → Employees ranked by salary
Dept B → Employees ranked by salary
```

---

5. Indexes

Description:
Indexes are used to speed up data retrieval from tables at the cost of extra storage and slower writes.

Example:

```
CREATE INDEX idx_customer_name ON
customers(customer_name);
```

Explanation:

- Creates an index on the `customer_name` column.
- Queries filtering by customer_name will be faster.

Illustration:

```
[Customers Table]
      ↕
[Index on customer_name] → Fast lookups
```

---

6. Sequences

Description:
Sequences generate unique numbers, often used for primary keys.

Example:

```
CREATE SEQUENCE seq_order START WITH 1 INCREMENT BY
1;
INSERT INTO orders (order_id, order_date) VALUES
(seq_order.NEXTVAL, SYSDATE);
```

Explanation:

- `seq_order` starts at 1 and increases by 1.
- `NEXTVAL` generates the next number.

Illustration:

```
Seq: 1 → 2 → 3 → 4 … assigned to order_id
```

---

7. Constraints

Description:
Constraints enforce rules on data (e.g., uniqueness, relationships).

Example:

```
ALTER TABLE employees
ADD CONSTRAINT fk_dept FOREIGN KEY (department_id)
REFERENCES departments(department_id);
```

Explanation:

- Ensures every `department_id` in `employees` exists in `departments`.
- Prevents invalid data entries.

Illustration:

```
Employees.department_id → must exist in →
Departments.department_id
```
