



## EXPERIMENT- 05

**Student Name: Ruchi Sharma**

**UID: 23BCS10713**

**Branch: BE-CSE**

**Section/Group: KRG 1(B)**

**Semester: 05**

**Date of Performance: 23/09/25**

**Subject Name: ADBMS**

**Subject Code: 23CSP-333**

### **Performance Benchmarking: Normal View VS Materialized View (Medium Level)**

#### **1. Aim:**

- i. Create a large dataset
  - Create a table names `transaction_data` (`id` , `value`) with 1 million records.
  - take `id` 1 and 2, and for each `id`, generate 1 million records in `value` column
  - Use `Generate_series ()` and `random()` to populate the data.
- ii. Create a normal view and materialized view to for `sales_summary`, which includes `total_quantity_sold`, `total_sales`, and `total_orders` with aggregation.
- iii. Compare the performance and execution time of both.

#### **2. Objective:**

- Create a large dataset `transaction_data` with 1 million records for each `id`.
- Create a **normal view** and a **materialized view** to summarize sales (`total_orders`, `total_sales`, `avg_transaction`).
- Compare performance using `EXPLAIN ANALYZE`.

#### **3. DBMS script and output:**

```
CREATE TABLE transaction_data (  
    id INT,  
    value INT  
);
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
-- Insert random data for id = 1
INSERT INTO transaction_data (id, value)
SELECT 1, random() * 1000
FROM generate_series(1, 1000000);

-- Insert random data for id = 2
INSERT INTO transaction_data (id, value)
SELECT 2, random() * 1000
FROM generate_series(1, 1000000);

-- Show data
SELECT * FROM transaction_data;

-- Normal view
CREATE OR REPLACE VIEW sales_summary_view AS
SELECT
    id,
    COUNT(*) AS total_orders,
    SUM(value) AS total_sales,
    AVG(value) AS avg_transaction
FROM transaction_data
GROUP BY id;

-- Run normal view
EXPLAIN ANALYZE SELECT * FROM sales_summary_view;

-- Materialized view
CREATE MATERIALIZED VIEW sales_summary_mv AS
SELECT
    id,
    COUNT(*) AS total_orders,
    SUM(value) AS total_sales,
    AVG(value) AS avg_transaction
FROM transaction_data
GROUP BY id;

-- Run materialized view
EXPLAIN ANALYZE SELECT * FROM sales_summary_mv;

-- New table
CREATE TABLE random_tabl (
    id INT,
    val DECIMAL
);

-- Insert random values for id = 1
INSERT INTO random_tabl
SELECT 1, random()
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
FROM generate_series(1, 1000000);

-- Insert random values for id = 2
INSERT INTO random_tabl
SELECT 2, random()
FROM generate_series(1, 1000000);

-- Normal query
SELECT id, AVG(val), COUNT(*)
FROM random_tabl
GROUP BY id;

-- Materialized view for query
CREATE MATERIALIZED VIEW mv_random_tabl AS
SELECT id, AVG(val), COUNT(*)
FROM random_tabl
GROUP BY id;

-- Show materialized view
SELECT * FROM mv_random_tabl;

-- Refresh if data changes
REFRESH MATERIALIZED VIEW mv_random_tabl;
```

## 4. Output:

	QUERY PLAN
	text
1	HashAggregate (cost=5.00..5.02 rows=2 width=52) (actual time=0.135..0.138 rows=2 loops=1)
2	Group Key: transaction_data.id
3	Batches: 1 Memory Usage: 24kB
4	-> Seq Scan on transaction_data (cost=0.00..3.00 rows=200 width=8) (actual time=0.030..0.050 rows=200 loop=...
5	Planning Time: 0.168 ms
6	Execution Time: 0.186 ms

^ normal view performance

	QUERY PLAN
	text
1	Seq Scan on sales_summary_mv (cost=0.00..20.20 rows=1020 width=52) (actual time=0.043..0.046 rows=2 loops=...
2	Planning Time: 0.169 ms
3	Execution Time: 0.085 ms

^ materialized view performance



Data Output Messages Notifications

REFRESH MATERIALIZED VIEW

Query returned successfully in 216 msec.

## Securing Data Access with Views and Role-Based Permissions (Hard Level)

### 1. Aim:

The company **TechMart Solutions** stores all sales transactions in a central database. A new reporting team has been formed to analyze sales but **they should not have direct access to the base tables** for security reasons.

The database administrator has decided to:

- i. Create **restricted views** to display only summarized, non-sensitive data.
- ii. Assign access to these views to specific users using **DCL commands** (GRANT, REVOKE).

### 2. Objective:

- To create **restricted views** that display only summarized, non-sensitive sales data for reporting.
- To use **DCL commands (GRANT, REVOKE)** for controlling user access to views.
- To ensure reporting users can only **view data** without direct access to base tables.

### 3. DBMS script and output:

```
-- 1. Create customer_master
CREATE TABLE customer_master (
  customer_id VARCHAR(5) PRIMARY KEY,
  full_name VARCHAR(50) NOT NULL,
  phone VARCHAR(15),
  email VARCHAR(50),
  city VARCHAR(30)
);
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

-- 2. Create product\_catalog

```
CREATE TABLE product_catalog (  
    product_id VARCHAR(5) PRIMARY KEY,  
    product_name VARCHAR(50) NOT NULL,  
    brand VARCHAR(30),  
    unit_price NUMERIC(10,2) NOT NULL  
);
```

-- 3. Create sales\_orders

```
CREATE TABLE sales_orders (  
    order_id SERIAL PRIMARY KEY,  
    product_id VARCHAR(5) REFERENCES product_catalog(product_id),  
    quantity INT NOT NULL,  
    customer_id VARCHAR(5) REFERENCES customer_master(customer_id),  
    discount_percent NUMERIC(5,2),  
    order_date DATE NOT NULL  
);
```

```
INSERT INTO customer_master (customer_id, full_name, phone, email, city) VALUES  
( 'C1', 'Amit Sharma', '9876543210', 'amit.sharma@example.com', 'Delhi'),  
( 'C2', 'Priya Verma', '9876501234', 'priya.verma@example.com', 'Mumbai'),  
( 'C3', 'Ravi Kumar', '9988776655', 'ravi.kumar@example.com', 'Bangalore'),  
( 'C4', 'Neha Singh', '9123456789', 'neha.singh@example.com', 'Kolkata'),  
( 'C5', 'Arjun Mehta', '9812345678', 'arjun.mehta@example.com', 'Hyderabad'),  
( 'C6', 'Sneha Reddy', '9090909090', 'sneha.reddy@example.com', 'Chennai'),  
( 'C7', 'Vikram Das', '9123412345', 'vikram.das@example.com', 'Pune'),  
( 'C8', 'Rohit Gupta', '9000000001', 'rohit.gupta@example.com', 'Lucknow'),  
( 'C9', 'Pooja Nair', '9898989898', 'pooja.nair@example.com', 'Kochi'),  
( 'C10', 'Ankit Yadav', '9345678901', 'ankit.yadav@example.com', 'Ahmedabad');
```

```
INSERT INTO product_catalog (product_id, product_name, brand, unit_price) VALUES  
( 'P1', 'Smartphone X100', 'Samsung', 25000.00),  
( 'P2', 'Laptop Pro 15', 'Dell', 65000.00),  
( 'P3', 'Wireless Earbuds', 'Sony', 5000.00),  
( 'P4', 'Smartwatch Fit', 'Apple', 30000.00),  
( 'P5', 'Tablet 10.5', 'Lenovo', 22000.00),  
( 'P6', 'Gaming Console', 'Sony', 45000.00),  
( 'P7', 'Bluetooth Speaker', 'JBL', 7000.00),  
( 'P8', 'Digital Camera', 'Canon', 55000.00),  
( 'P9', 'LED TV 55 inch', 'LG', 60000.00),
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

('P10', 'Power Bank 20000mAh', 'Mi', 2500.00);

```
INSERT INTO sales_orders (product_id, quantity, customer_id, discount_percent, order_date) VALUES
('P1', 2, 'C1', 5.00, '2025-09-01'),
('P2', 1, 'C2', 10.00, '2025-09-02'),
('P3', 3, 'C3', 0.00, '2025-09-03'),
('P4', 1, 'C4', 8.00, '2025-09-04'),
('P5', 2, 'C5', 5.00, '2025-09-05'),
('P6', 1, 'C1', 12.00, '2025-09-06'),
('P7', 2, 'C2', 0.00, '2025-09-07'),
('P8', 1, 'C3', 10.00, '2025-09-08'),
('P9', 1, 'C6', 15.00, '2025-09-09'),
('P10', 4, 'C7', 0.00, '2025-09-10'),
('P1', 1, 'C8', 5.00, '2025-09-11'),
('P2', 2, 'C9', 10.00, '2025-09-12'),
('P3', 2, 'C10', 0.00, '2025-09-13'),
('P4', 1, 'C5', 8.00, '2025-09-14'),
('P5', 3, 'C6', 5.00, '2025-09-15'),
('P6', 1, 'C7', 12.00, '2025-09-16'),
('P7', 2, 'C8', 0.00, '2025-09-17'),
('P8', 1, 'C9', 10.00, '2025-09-18'),
('P9', 1, 'C10', 15.00, '2025-09-19'),
('P10', 5, 'C4', 0.00, '2025-09-20');
```

-- Create view for order summary

```
CREATE VIEW vW_ORDER_SUMMARY AS
SELECT
```

```
    O.order_id,
    O.order_date,
    P.product_name,
    C.full_name,
    (P.unit_price * O.quantity) - ((P.unit_price * O.quantity) * O.discount_percent / 100) AS final_cost
FROM customer_master AS C
JOIN sales_orders AS O
    ON O.customer_id = C.customer_id
JOIN product_catalog AS P
    ON P.product_id = O.product_id;
```

-- Check data in the view

```
SELECT * FROM vW_ORDER_SUMMARY;
```

-- User access

-- Create client user



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
CREATE ROLE RUCHI LOGIN PASSWORD 'ruchi';

-- Give select access to view
GRANT SELECT ON vW_ORDER_SUMMARY TO RUCHI;

-- Optional: revoke access
-- REVOKE SELECT ON vW_ORDER_SUMMARY FROM ALOK;

-- Employee table
CREATE TABLE EMPLOYEE (
    empId INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    dept TEXT NOT NULL
);

-- Insert sample data
INSERT INTO EMPLOYEE VALUES (1, 'Clark', 'Sales');
INSERT INTO EMPLOYEE VALUES (2, 'Dave', 'Accounting');
INSERT INTO EMPLOYEE VALUES (3, 'Ava', 'Sales');

-- View table data
SELECT * FROM EMPLOYEE;

-- View with check option
CREATE VIEW vW_STORE_SALES_DATA AS
SELECT empId, name, dept
FROM EMPLOYEE
WHERE dept = 'Sales'
WITH CHECK OPTION;

-- Check view data
SELECT * FROM vW_STORE_SALES_DATA;
```

## 5. Output:

Data Output Messages Notifications

GRANT

Query returned successfully in 94 msec.



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

									SQL
	empid integer		name text		dept text				
1	1		Clark		Sales				
2	3		Ava		Sales				