



## Experiment No: 5

Student Name: Nagesh Kumar

UID: 23BCS10195

Branch: B.E./C.S.E.

Section/Group: KRG\_2-A

Subject Name: ADBMS

Subject Code: 23CSP-333

### Medium Level Problem

Question : Normal View vs. Materialized View

1. 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.

2. Create a normal view and materialized view to for sales\_summary, which includes total\_quantity\_sold, total\_sales, and total\_orders with aggregation

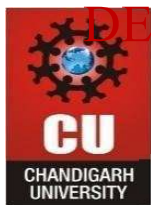
3. Compare the performance and execution time of both.

Solution:

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

```
Insert 1 million records for id = 1  
INSERT INTO transaction_data (id, value)  
SELECT 1, (random() * 100)::numeric  
FROM generate_series(1, 1000000);
```

```
Insert 1 million records for id = 2  
INSERT INTO transaction_data (id, value)
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
SELECT 2, (random() * 100)::numeric  
FROM generate_series(1, 1000000);
```

WITH 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;
```

EXPLAIN ANALYZE

```
SELECT * FROM sales_summary_view;
```

|    | QUERY PLAN  |  |
|----|---|--|
|    | text  |  |
| 1  | Finalize GroupAggregate (cost=25226.29..25279.46 rows=200 width=76) (actual time=364.318..375.012 rows=2 loops=1)         |  |
| 2  | Group Key: transaction_data.id  |  |
| 3  | -> Gather Merge (cost=25226.29..25272.96 rows=400 width=44) (actual time=364.304..374.995 rows=6 loops=1)                 |  |
| 4  | Workers Planned: 2  |  |
| 5  | Workers Launched: 2   |  |
| 6  | -> Sort (cost=24226.26..24226.76 rows=200 width=44) (actual time=289.350..289.351 rows=2 loops=3)                         |  |
| 7  | Sort Key: transaction_data.id   |  |
| 8  | Sort Method: quicksort Memory: 25kB   |  |
| 9  | Worker 0: Sort Method: quicksort Memory: 25kB   |  |
| 10 | Worker 1: Sort Method: quicksort Memory: 25kB   |  |
| 11 | -> Partial HashAggregate (cost=24216.12..24218.62 rows=200 width=44) (actual time=289.302..289.304 rows=2 loops=3)        |  |
| 12 | Group Key: transaction_data.id  |  |
| 13 | Batches: 1 Memory Usage: 40kB   |  |
| 14 | Worker 0: Batches: 1 Memory Usage: 40kB   |  |
| 15 | Worker 1: Batches: 1 Memory Usage: 40kB   |  |
| 16 | -> Parallel Seq Scan on transaction_data (cost=0.00..19226.21 rows=665321 width=36) (actual time=0.023..80.878 rows=66... |  |
| 17 | Planning Time: 0.276 ms   |  |
| 18 | Execution Time: 375.102 ms  |  |

WITH MATERIALIZED VIEW

```
CREATE MATERIALIZED VIEW sales_summary_mv AS SELECT
```




# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
id,  
COUNT(*) AS total_orders,  
SUM(value) AS total_sales,  
AVG(value) AS avg_transaction  
FROM transaction_data  
GROUP BY id;
```

```
EXPLAIN ANALYZE  
SELECT * FROM sales_summary_mv;
```

|   | QUERY PLAN   |   |
|---|--|---|
|   | text   |  |
| 1 | Seq Scan on sales_summary_mv (cost=0.00..17.80 rows=780 width=76) (actual time=0.014..0.016 rows=2 loops=... |   |
| 2 | Planning Time: 0.858 ms  |   |
| 3 | Execution Time: 0.031 ms   |   |

## Hard Level Problem

Question : Securing Data Access with Views and Role-Based Permissions

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:

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

Solution:

```
CREATE VIEW vW_ORDER_SUMMARY AS  
SELECT
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
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;  
  
SELECT * FROM vW_ORDER_SUMMARY;  
  
CREATE ROLE CLIENT_USER  
LOGIN  
PASSWORD 'client_password';  
  
GRANT SELECT ON vW_ORDER_SUMMARY TO CLIENT_USER;  
  
REVOKE SELECT ON vW_ORDER_SUMMARY FROM CLIENT_USER;
```