
**University Institute of Engineering**

**Department of Computer Science & Engineering**

# EXPERIMENT:5

**NAME : SATYA PRAKASH SWAIN** **UID : 23BCS11072**
**BRANCH : BE-CSE** **SECTION : KRG_1A**
**SEMESTER : 5TH** **SUBJECT : 23CSP-339**
**SUBJECT NAME : ADBMS**

## 1. AIM:-

**[MEDIUM]**

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.

**[HARD]**

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).

## 2. TOOLS USED :-

PgAdmin4

## 3. CODE:-

**[MEDIUM]**

```
CREATE TABLE transaction_data (
```

```
    id INT,
    value INT
);

-- For id = 1
INSERT INTO transaction_data (id, value)
SELECT 1, random() * 1000  -- simulate transaction amounts 0-1000
FROM generate_series(1, 1000000);

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

SELECT *FROM transaction_data


--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;


--WITH 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;
```

```
EXPLAIN ANALYZE
SELECT * FROM sales_summary_mv;
create table random_tabl (id int, val decimal)

insert into random_tabl
select 1, random() from generate_series(1,1000000);

insert into random_tabl
select 2, random() from generate_series(1,1000000);


--normal execution
select id, avg(val), count(*)
from random_tabl
group by id;

--execution by materialized view
create materialized view mv_random_tabl
as
select id, avg(val), count(*)
from random_tabl
group by id;

select *from mv_random_tabl

--if you update anything in table, the mv doesn't gets updated
---for that we have to refresh it

refresh materialized view mv_random_tabl;
```

**[HARD]**

```
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;


CREATE ROLE VANSH
LOGIN
PASSWORD 'vansh';

GRANT SELECT ON vW_ORDER_SUMMARY TO  VANSH;
REVOKE SELECT ON vW_ORDER_SUMMARY FROM VANSH;


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

-- insert
INSERT INTO EMPLOYEE VALUES (0001, 'Clark', 'Sales');
INSERT INTO EMPLOYEE VALUES (0002, 'Dave', 'Accounting');
INSERT INTO EMPLOYEE VALUES (0003, 'Ava', 'Sales');

select *from employee;

CREATE VIEW vW_STORE_SALES_DATA
AS
   SELECT EMPID, NAME, DEPT
   FROM EMPLOYEE
   WHERE DEPT = 'Sales'
   WITH CHECK OPTION;

SELECT *FROM vW_STORE_SALES_DATA;
```

INSERT INTO vW_STORE_SALES_DATA(EMPID, NAME, DEPT) VALUES (5, 'Aman', 'Admin'); --VIOLATION CONDITION

## 4. OUTPUT:-

Data Output | Messages | Notifications

| | QUERY PLAN text |
|---|---|
| 1 | Finalize GroupAggregate (cost=26394.39..26394.93 rows=2 width=76) (actual time=253.023..261.774 rows=2 loops=1) |
| 2 | Group Key: transaction_data.id |
| 3 | -> Gather Merge (cost=26394.39..26394.86 rows=4 width=44) (actual time=253.005..261.753 rows=6 loops=1) |
| 4 | Workers Planned: 2 |
| 5 | Workers Launched: 2 |
| 6 | -> Sort (cost=25394.37..25394.37 rows=2 width=44) (actual time=212.358..212.359 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=25394.33..25394.36 rows=2 width=44) (actual time=212.331..212.332 rows=2 loops=3) |
| 12 | Group Key: transaction_data.id |
| 13 | Batches: 1 Memory Usage: 24kB |
| 14 | Worker 0: Batches: 1 Memory Usage: 24kB |
| 15 | Worker 1: Batches: 1 Memory Usage: 24kB |
| 16 | -> Parallel Seq Scan on transaction_data (cost=0.00..19144.33 rows=833333 width=10) (actual time=0.013..45.481 rows=66... |
| 17 | Planning Time: 0.158 ms |
| 18 | Execution Time: 261.820 ms |

Data Output | Messages | Notifications

| | QUERY PLAN text |
|---|---|
| 1 | Seq Scan on sales_summary_mat (cost=0.00..1.02 rows=2 width=76) (actual time=0.018..0.019 rows=2 loops=... |
| 2 | Planning Time: 0.101 ms |
| 3 | Execution Time: 0.034 ms |

## 5. LEARNING OUTCOMES:-

1. Ability to create and populate large datasets efficiently using generate_series() and random().
2. Understanding of table design and schema creation for transactional data.
3. Skills in writing aggregate queries (SUM, COUNT, GROUP BY) for reporting purposes.
4. Learn the difference between a normal view and a materialized view and when to use each.
5. Ability to analyze and compare query performance and execution times.
6. Understanding of performance optimization and caching benefits with materialized views.

7. Ability to create restricted views that provide summarized, non-sensitive data.
8. Understanding data security best practices by limiting direct access to base tables.
9. Skills in Data Control Language (DCL) commands (GRANT and REVOKE) to manage user access.
10. Learn to balance usability and security by providing controlled access to sensitive business data.

I