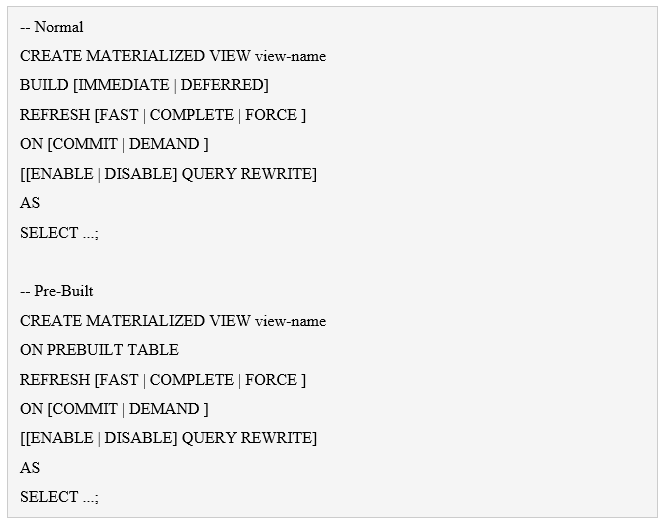
**---------------------------------------------------------------------------------------------**

**DWM EXPERIMENT NO. : 05** **Roll no :50**

**---**-**-----------------------------------------------------------------------------------------Problem definition :**

To implement materialized view on given data warehouse for various OLAP and Aggregate operations. Also to compare query response time for a given query.

**---------------------------------------------------------------------------------------------**

****

**Perform the following tasks :**

**Q1. Connect with scott/tiger and create the following view :**

CREATE MATERIALIZED VIEW emp\_mv

BUILD IMMEDIATE

REFRESH FORCE

ON DEMAND

AS

SELECT \* FROM EMP;

Materialized view created.

**Q2. Gather stats after building the materialized view.**

BEGIN

DBMS\_STATS.GATHER\_TABLE\_STATS(

CREATE MATERIALIZED VIEW LOG SCOTT\_EMP

TABLESPACE USERS

WITH PRIMARY KEY

INCLUDING NEW VALUES;

/

PL/SQL procedure successfully completed.

**Q3. Create the materialized view log**

CREATE MATERIALIZED VIEW LOG ON scott.emp

TABLESPACE users

WITH PRIMARY KEY

INCLUDING NEW VALUES;

Materialized view log created.

**Q4. Create a refresh group to refresh every minute and assign the created materialized view to it.**

EXEC DBMS\_MVIEW.refresh('EMP\_MVV');

ALTER SESSION SET QUERY\_REWRITE;

**Q5. Clean up to remove all objects.**

SQL> DROP MATERIALIZED VIEW emp\_mv1;

Materialized view dropped.

SQL> DROP DATABASE LINK DB1.WORLD;

Database link dropped.

SQL> BEGIN

2 DBMS\_REFRESH.destroy(name => 'SCOTT.MINUTE\_REFRESH');

3 END;

4 /

PL/SQL procedure successfully completed.

SQL> DROP MATERIALIZED VIEW LOG ON scott.emp;

Materialized view log dropped.

**Q6. Execute the following query**

**SET AUTOTRACE TRACE EXPLAIN**

SQL> SET AUTOTRACE TRACE EXPLAIN

SQL> SELECT deptno, SUM(sal)

2 FROM emp

3 GROUP BY deptno;

Execution Plan

----------------------------------------------------------

Plan hash value: 4067220884

---------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

---------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 3 | 21 | 4 (25)| 00:00:01 |

| 1 | HASH GROUP BY | | 3 | 21 | 4 (25)| 00:00:01 |

| 2 | TABLE ACCESS FULL| EMP | 14 | 98 | 3 (0)| 00:00:01 |

---------------------------------------------------------------------------

**Q7. Write a materialized view to perform the aggregation in advance (make sure you specify the ENABLE QUERY REWRITE clause)**

CREATE MATERIALIZED VIEW SALES\_AGGR\_MVV

BUILD IMMEDIATE

REFRESH FORCE

ON DEMAND

ENABLE QUERY REWRITE

AS

SELECT CHANNELS.CHANNEL\_DESC, CALENDAR\_MONTH\_DESC,

COUNTRIES.COUNTRY\_ISO\_CODE,

TO\_CHAR(SUM(AMOUNT\_SOLD), '9,999,999,999') SALES$

FROM SALES, CUSTOMERS, TIMES, CHANNELS, COUNTRIES

WHERE SALES.TIME\_ID=TIMES.TIME\_ID

AND SALES.CUST\_ID=CUSTOMERS.CUST\_ID

AND CUSTOMERS.COUNTRY\_ID = COUNTRIES.COUNTRY\_ID

AND SALES.CHANNEL\_ID = CHANNELS.CHANNEL\_ID

AND UPPER(CHANNELS.CHANNEL\_DESC) IN ('DIRECT SALES', 'INTERNET')

AND TIMES.CALENDAR\_MONTH\_DESC IN ('2000-09', '2000-10')

AND COUNTRIES.COUNTRY\_ISO\_CODE IN ('GB', 'US')

GROUP BY

ROLLUP(CHANNELS.CHANNEL\_DESC, CALENDAR\_MONTH\_DESC, COUNTRIES.COUNTRY\_ISO\_CODE);

Materialized view created.

SQL> EXEC DBMS\_STATS.gather\_table\_stats(USER, 'EMP\_AGGR\_MV');

PL/SQL procedure successfully completed.

**Q8. Rewrite the same query to take advantage of the pre-aggregated data in the materialized view, instead of the session doing the work for itself.**

CREATE MATERIALIZED VIEW SALES\_AGGR\_MV\_FAST

BUILD IMMEDIATE

REFRESH FAST

ON DEMAND

ENABLE QUERY REWRITE

AS

SELECT CHANNELS.CHANNEL\_DESC, CALENDAR\_MONTH\_DESC,

COUNTRIES.COUNTRY\_ISO\_CODE,

TO\_CHAR(SUM(AMOUNT\_SOLD), '9,999,999,999') SALES$

FROM SALES, CUSTOMERS, TIMES, CHANNELS, COUNTRIES

WHERE SALES.TIME\_ID=TIMES.TIME\_ID

AND SALES.CUST\_ID=CUSTOMERS.CUST\_ID

AND CUSTOMERS.COUNTRY\_ID = COUNTRIES.COUNTRY\_ID

AND SALES.CHANNEL\_ID = CHANNELS.CHANNEL\_ID

AND UPPER(CHANNELS.CHANNEL\_DESC) IN ('DIRECT SALES', 'INTERNET')

AND TIMES.CALENDAR\_MONTH\_DESC IN ('2000-09', '2000-10')

AND COUNTRIES.COUNTRY\_ISO\_CODE IN ('GB', 'US')

GROUP BY

ROLLUP(CHANNELS.CHANNEL\_DESC, CALENDAR\_MONTH\_DESC, COUNTRIES.COUNTRY\_ISO\_CODE);

Execution Plan

----------------------------------------------------------

Plan hash value: 2456459487

--------------------------------------------------------------------------------

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)

| Time |

--------------------------------------------------------------------------------

| 0 | SELECT STATEMENT | | 3 | 21 | 3 (0)

| 00:00:01 |

| 1 | MAT\_VIEW REWRITE ACCESS FULL| EMP\_AGGR\_MV | 3 | 21 | 3 (0)

| 00:00:01 |

--------------------------------------------------------------------------------

**Q9. Compare the query response times of all queries created in practical 2 with the query response time of their materialized views.**

The materialized views result in quicker response and improved efficiency.

The performance of Materialized view is better than normal View of lab 2 because the data of materialized is stored in table and table may be [indexed](http://java67.blogspot.sg/2012/10/difference-between-clustered-vs-nonclustered-index-sql-database.html) so faster for joining. Also joining is done at the time of materialized views refresh time so no need to every time fire join statement as in case of view.

**---------------------------------------------------------------------------------------**  **E N D**

**---------------------------------------------------------------------------------------**