Shri Ramdeobaba College of Engineering and Management Nagpur, 440013

Department of Computer Science Engineering

Practical 3

Name - *Shantanu Mane*
Roll No. - *E63*
Batch - *E4*
Subject - *Data Mining and Warehousing Lab* **Date** - *22/3/2023*

AIM - To execute following data partitioning technique in data warehouse, (operations can be demonstrated on any schema.)

1. Range Partitioning
2. List Partitioning
3. Multi-column Partitioning
4. Interval Partitioning
5. Composite Partitioning
6. Reference Partitioning

Q1. Write a query to create range partitioned table:

- Creates a table named- Sales consisting of four partitions, one for each quarter of sales. The column sale_date are the partitioning columns, while their values constitute the partitioning key of a specific row.
- Each partition is given a name (sales_q1, sales_q2, ...), and each partition is contained in a separate tablespace (tsa, tsb, ...)
- The columns for table must be prod_id, cust_id, promo_id, quantity sold, amount_sold – all in number format and sale_date.

```
-- CREATE TABLESPACE FOR THE PARTITIONED TABLES
CREATE TABLESPACE tsa
    DATAFILE 'tsa.dbf' SIZE 100 M
    AUTOEXTEND ON NEXT 10 M MAXSIZE 1 G;
    AUTOEXTEND ON NEXT 10 M MAXSIZE 1 G;

CREATE TABLESPACE tsb
    DATAFILE 'tsb.dbf' SIZE 100 M
    AUTOEXTEND ON NEXT 10 M MAXSIZE 1 G;


CREATE TABLESPACE tsc
    DATAFILE 'tsc.dbf' SIZE 100 M
    AUTOEXTEND ON NEXT 10 M MAXSIZE 1 G;

CREATE TABLESPACE tsd
    DATAFILE 'tsd.dbf' SIZE 100 M
    AUTOEXTEND ON NEXT 10 M MAXSIZE 1 G;

-- CREATE TABLE
CREATE TABLE sales_two
(
    prod_id       number(4),
    cust_id       number(4),
    promo_id      number(4),
    quantity_sold number(4),
    amount_sold   number(8, 2),
    sale_date     date
)
    PARTITION BY RANGE (sale_date)
(
    PARTITION sales_q1 VALUES LESS THAN (TO_DATE('01-APR-2023', 'DD-MON-YYYY')) TABLESPACE
    PARTITION sales_q2 VALUES LESS THAN (TO_DATE('01-JUL-2023', 'DD-MON-YYYY')) TABLESPACE
    PARTITION sales_q3 VALUES LESS THAN (TO_DATE('01-OCT-2023', 'DD-MON-YYYY')) TABLESPACE
    PARTITION sales_q4 VALUES LESS THAN (TO_DATE('01-JAN-2024', 'DD-MON-YYYY')) TABLESPACE
);

-- ADDING SAMPLE DATA TO THE TABLE
INSERT INTO sales_two
VALUES (1, 1, 1, 1, 1, TO_DATE('01-JAN-2023', 'DD-MON-YYYY'));
INSERT INTO sales_two
VALUES (2, 2, 2, 2, 2, TO_DATE('01-APR-2023', 'DD-MON-YYYY'));
INSERT INTO sales_two
VALUES (3, 3, 3, 3, 3, TO_DATE('01-JUL-2023', 'DD-MON-YYYY'));
INSERT INTO sales_two
VALUES (4, 4, 4, 4, 4, TO_DATE('01-OCT-2023', 'DD-MON-YYYY'));

SELECT tablespace_name, partition_name, partition_position, high_value
FROM dba_tab_partitions
WHERE table_name = 'SALES_TWO';
```

Output:

```
+---------------+--------------+------------------+-----------------------------------------
|TABLESPACE_NAME|PARTITION_NAME|PARTITION_POSITION|HIGH_VALUE
+---------------+--------------+------------------+-----------------------------------------
|TSA            |SALES_Q1      |1                 |TO_DATE(' 2023-04-01 00:00:00', 'SYYYY-M
|TSB            |SALES_Q2      |2                 |TO_DATE(' 2023-07-01 00:00:00', 'SYYYY-M
|TSC            |SALES_Q3      |3                 |TO_DATE(' 2023-10-01 00:00:00', 'SYYYY-M
|TSD            |SALES_Q4      |4                 |TO_DATE(' 2024-01-01 00:00:00', 'SYYYY-M
+---------------+--------------+------------------+-----------------------------------------
```

Q2. Create the same table as in Q1. With a different name with ENABLE ROW MOVEMENT. Bring out the difference in these two tables.

```
-- CREATE A TABLE WITH ENABLE ROW MOVEMENT AND PARTITIONS
CREATE TABLE sales_two_erm
(
    prod_id       number(4),
    cust_id       number(4),
    promo_id      number(4),
    quantity_sold number(4),
    amount_sold   number(8, 2),
    sale_date     date
)
    PARTITION BY RANGE (sale_date)
(
    PARTITION sales_q1 VALUES LESS THAN (TO_DATE('01-APR-2023', 'DD-MON-YYYY')) TABLESPACE
    PARTITION sales_q2 VALUES LESS THAN (TO_DATE('01-JUL-2023', 'DD-MON-YYYY')) TABLESPACE
    PARTITION sales_q3 VALUES LESS THAN (TO_DATE('01-OCT-2023', 'DD-MON-YYYY')) TABLESPACE
    PARTITION sales_q4 VALUES LESS THAN (TO_DATE('01-JAN-2024', 'DD-MON-YYYY')) TABLESPACE
)
    ENABLE ROW MOVEMENT;

-- ADDING SAMPLE DATA TO THE TABLE
INSERT INTO sales_two_erm
VALUES (1, 1, 1, 1, 1, TO_DATE('01-JAN-2023', 'DD-MON-YYYY'));
INSERT INTO sales_two_erm
VALUES (2, 2, 2, 2, 2, TO_DATE('01-APR-2023', 'DD-MON-YYYY'));
INSERT INTO sales_two_erm
VALUES (3, 3, 3, 3, 3, TO_DATE('01-JUL-2023', 'DD-MON-YYYY'));
INSERT INTO sales_two_erm
VALUES (4, 4, 4, 4, 4, TO_DATE('01-OCT-2023', 'DD-MON-YYYY'));

-- QUERYING THE TABLESPACE
SELECT tablespace_name, partition_name, partition_position, high_value
FROM dba_tab_partitions
WHERE table_name = 'SALES_TWO_ERM';
```

Output:

```
+---------------+--------------+------------------+---------------------------------------
|TABLESPACE_NAME|PARTITION_NAME|PARTITION_POSITION|HIGH_VALUE
+---------------+--------------+------------------+---------------------------------------
|TSA            |SALES_Q1      |1                 |TO_DATE(' 2023-04-01 00:00:00', 'SYYYY-M
|TSB            |SALES_Q2      |2                 |TO_DATE(' 2023-07-01 00:00:00', 'SYYYY-M
|TSC            |SALES_Q3      |3                 |TO_DATE(' 2023-10-01 00:00:00', 'SYYYY-M
|TSD            |SALES_Q4      |4                 |TO_DATE(' 2024-01-01 00:00:00', 'SYYYY-M
+---------------+--------------+------------------+---------------------------------------

* * *

DIFFERENCE BETWEEN NORMAL AND ENABLE ROW MOVEMENT PARTITIONED TABLES
DIFFERENCE BETWEEN NORMAL AND ENABLE ROW MOVEMENT PARTITIONED TABLES

→ Normal partitioned tables are not allowed to move rows from one partition to another par
→ Enable row movement partitioned tables are allowed to move rows from one partition to ar
```

Q3. Create a table with list partition as follows:

- Table having columns deptno, deptname, quarterly_sales and state.
- Create partition on state:
  - Northwest on OR and WA
  - Southwest on AZ, UT and NM
  - northeast on NY, VM and NJ
  - southeast on FL and GA
  - northcentral on SD and WI
  - southcentral on OK and TX
- Add the following entries into the table and make conclusion to which partition the entry maps:
  - (10, 'accounting', 100, 'WA')
  - (20, 'R&D', 150, 'OR')
  - (30, 'sales', 100, 'FL')
  - (40, 'HR', 10, 'TX')
  - (50, 'systems engineering', 10, 'CA')

```sql
CREATE TABLE SALES_Q3
(
    deptno          number(4),
    deptname        varchar2(20),
    quarterly_sales number(4),
    state           varchar2(2)
)
    PARTITION BY LIST
(
    state
)
(
    PARTITION northwest VALUES ('OR', 'WA'),
    PARTITION southwest VALUES ('AZ', 'UT', 'NM'),
    PARTITION northeast VALUES ('NY', 'VM', 'NJ'),
    PARTITION southeast VALUES ('FL', 'GA'),
    PARTITION northcentral VALUES ('SD', 'WI'),
    PARTITION southcentral VALUES ('OK', 'TX')
);

-- ADDING SAMPLE DATA TO THE TABLE
INSERT INTO SALES_Q3
VALUES (10, 'accounting', 100, 'WA');
INSERT INTO SALES_Q3
VALUES (20, 'R&D', 150, 'OR');
INSERT INTO SALES_Q3
VALUES (30, 'sales', 100, 'FL');
INSERT INTO SALES_Q3
VALUES (40, 'HR', 10, 'TX');
INSERT INTO SALES_Q3 -- THIS WILL NOT GO TO ANY PARTITION (SINCE IT IS NOT IN THE LIST)
```

```
VALUES (50, 'systems engineering', 10, 'CA');

-- QUERYING THE TABLE TO FIND OUT WHICH PARTITION THE DATA IS GOING TO
SELECT *
FROM SALES_Q3 PARTITION (NORTHWEST)
WHERE STATE = 'WA';

SELECT *
FROM SALES_Q3 PARTITION (northcentral)
WHERE STATE = 'WA'; -- THIS WILL NOT RETURN ANYTHING
```

Output:

```
+------+----------+--------------+-----+
|DEPTNO|DEPTNAME  |QUARTERLY_SALES|STATE|
+------+----------+--------------+-----+
|10    |accounting|100           |WA   |
+------+----------+--------------+-----+
```

Q4. Create a multi-column range partitioned table as directed:

- Create a table with the actual DATE information in three separate columns: year, month, and day. Also amount_ sold.
- Create following partitions:
  - Before 2001: Less than jan 2001
  - Less than april 2001
  - Less than july 2001
  - Less than oct 2001
  - Less than jan 2002
  - Future with max incoming value
- Insert values into table and show to which partition does the value belong.
  - (2001,3,17, 2000);
  - (2001,11,1, 5000);
  - (2002,1,1, 4000);

Make conclusion for each result.

```
CREATE TABLE sales_Q4
(
    year        NUMBER(4),
    month       NUMBER(2),
    day         NUMBER(2),
    amount_sold NUMBER
)
    PARTITION BY RANGE (year, month, day)
(
    PARTITION p1 VALUES LESS THAN (2001, 1, 1),
    PARTITION p2 VALUES LESS THAN (2001, 4, 1),
    PARTITION p3 VALUES LESS THAN (2001, 7, 1),
    PARTITION p4 VALUES LESS THAN (2001, 10, 1),
    PARTITION p5 VALUES LESS THAN (2002, 1, 1),
    PARTITION p6 VALUES LESS THAN (MAXVALUE, MAXVALUE, MAXVALUE)
);

INSERT INTO sales_Q4
VALUES (2001, 3, 17, 2000);
INSERT INTO sales_Q4
VALUES (2001, 11, 1, 5000);
INSERT INTO sales_Q4
VALUES (2002, 1, 1, 4000);

EXPLAIN PLAN FOR
SELECT *
FROM sales_Q4
WHERE year = 2001
  AND month = 3
  AND day = 17;

SELECT *
FROM TABLE (DBMS_XPLAN.DISPLAY);
```

```
+------------------------------------------------------------------------------------------
|PLAN_TABLE_OUTPUT
+------------------------------------------------------------------------------------------
|Plan hash value: 1715653740
|
|------------------------------------------------------------------------------------------
|| Id  | Operation              | Name     | Rows  | Bytes | Cost (%CPU)| Time     | Pstart
|------------------------------------------------------------------------------------------
||   0 | SELECT STATEMENT       |          |     1 |    52 |     3   (0)| 00:00:01 |
||   1 |  PARTITION RANGE SINGLE|          |     1 |    52 |     3   (0)| 00:00:01 |      2
||*  2 |   TABLE ACCESS FULL    | SALES_Q2 |     1 |    52 |     3   (0)| 00:00:01 |      2
|------------------------------------------------------------------------------------------
|
|Predicate Information (identified by operation id):
|---------------------------------------------------
|
|   2 - filter("YEAR"=2001 AND "MONTH"=3 AND "DAY"=17)
|
|Note
|-----
|   - dynamic sampling used for this statement (level=2)
+------------------------------------------------------------------------------------------
```

Q5. Create a table with range partition as follows:

```
CREATE TABLE supplier_parts
(
    supplier_id NUMBER,
    partnum     NUMBER,
    quantity    NUMBER
)
```

```
    PARTITION BY RANGE (supplier_id, partnum)
(
    PARTITION p1 VALUES LESS THAN (6, 50),
    PARTITION p2 VALUES LESS THAN (11, 100),
    PARTITION p3 VALUES LESS THAN (MAXVALUE, MAXVALUE)
);

INSERT INTO supplier_parts
VALUES (5, 5, 1000);
INSERT INTO supplier_parts
VALUES (5, 150, 1000);
INSERT INTO supplier_parts
VALUES (10, 100, 1000);

SELECT *
FROM SUPPLIER_PARTS PARTITION (P1);
SELECT *
FROM SUPPLIER_PARTS PARTITION (P2);
SELECT *
FROM SUPPLIER_PARTS PARTITION (P3);
```

Output:

```
+-----------+-------+--------+
|SUPPLIER_ID|PARTNUM|QUANTITY|
+-----------+-------+--------+
|5          |5      |1000    |
|5          |150    |1000    |
+-----------+-------+--------+

+-----------+-------+--------+
|SUPPLIER_ID|PARTNUM|QUANTITY|
+-----------+-------+--------+
|10         |100    |1000    |
+-----------+-------+--------+
```

Q6. Create a table with range partition as follows:

```
CREATE TABLE sales_Q6
(
    prod_id       NUMBER,
    cust_id       NUMBER,
    promo_id      NUMBER,
    quantity_sold NUMBER,
    amount_sold   NUMBER,
    time_id       DATE
)
    PARTITION BY RANGE (time_id) INTERVAL (NUMTOYMINTERVAL(1, 'MONTH'))
(
    PARTITION sales_q1 VALUES LESS THAN (TO_DATE('01-APR-2022', 'DD-MON-YYYY')),
    PARTITION sales_q2 VALUES LESS THAN (TO_DATE('01-JUL-2022', 'DD-MON-YYYY')),
    PARTITION sales_q3 VALUES LESS THAN (TO_DATE('01-OCT-2022', 'DD-MON-YYYY')),
    PARTITION sales_q4 VALUES LESS THAN (TO_DATE('01-JAN-2023', 'DD-MON-YYYY'))
);
```

```sql
INSERT INTO sales_Q6
VALUES (1, 1, 1, 10, 100, TO_DATE('01-JAN-2022', 'DD-MON-YYYY'));
INSERT INTO sales_Q6
VALUES (2, 2, 2, 20, 200, TO_DATE('01-OCT-2022', 'DD-MON-YYYY'));
INSERT INTO sales_Q6
VALUES (3, 3, 3, 30, 300, TO_DATE('01-JUL-2022', 'DD-MON-YYYY'));

SELECT *
FROM sales_Q6 PARTITION (sales_q1);
SELECT *
FROM sales_Q6 PARTITION (sales_q2);
SELECT *
FROM sales_Q6 PARTITION (sales_q3);
SELECT *
FROM sales_Q6 PARTITION (sales_q4);
```

Output:

```
+-------+-------+--------+-------------+-----------+----------+
|PROD_ID|CUST_ID|PROMO_ID|QUANTITY_SOLD|AMOUNT_SOLD|TIME_ID   |
+-------+-------+--------+-------------+-----------+----------+
|1      |1      |1       |10           |100        |2022-01-01|
|2      |2      |2       |20           |200        |2022-02-01|
|3      |3      |3       |30           |300        |2022-03-01|
|1      |1      |1       |10           |100        |2022-01-01|
+-------+-------+--------+-------------+-----------+----------+

+-------+-------+--------+-------------+-----------+----------+
|PROD_ID|CUST_ID|PROMO_ID|QUANTITY_SOLD|AMOUNT_SOLD|TIME_ID   |
+-------+-------+--------+-------------+-----------+----------+
+-------+-------+--------+-------------+-----------+----------+

+-------+-------+--------+-------------+-----------+----------+
|PROD_ID|CUST_ID|PROMO_ID|QUANTITY_SOLD|AMOUNT_SOLD|TIME_ID   |
+-------+-------+--------+-------------+-----------+----------+
|3      |3      |3       |30           |300        |2022-07-01|
+-------+-------+--------+-------------+-----------+----------+

+-------+-------+--------+-------------+-----------+----------+
|PROD_ID|CUST_ID|PROMO_ID|QUANTITY_SOLD|AMOUNT_SOLD|TIME_ID   |
+-------+-------+--------+-------------+-----------+----------+
|2      |2      |2       |20           |200        |2022-10-01|
+-------+-------+--------+-------------+-----------+----------+
```

Q7. Create a table with range partition as follows:

```sql
-- Create the table
CREATE TABLE customer
(
    cust_id       NUMBER,
    cust_name     VARCHAR2(50),
    cust_state    CHAR(2),
    time_id       DATE,
    quantity_sold NUMBER,
    amount_sold   NUMBER
)
    PARTITION BY RANGE (time_id)
    SUBPARTITION BY LIST
(
    cust_state
)
    SUBPARTITION TEMPLATE
(
    SUBPARTITION
    west
    VALUES ('MH', 'GJ'),
    SUBPARTITION
    south
    VALUES ('TN', 'AP'),
    SUBPARTITION
    north
    VALUES ('UP', 'HP'),
    SUBPARTITION
    unknown
    VALUES (DEFAULT)
)
(
    PARTITION old VALUES LESS THAN (TO_DATE('01-JAN-2005', 'DD-MON-YYYY')),
    PARTITION acquired VALUES LESS THAN (TO_DATE('01-JAN-2010', 'DD-MON-YYYY')),
    PARTITION recent VALUES LESS THAN (TO_DATE('01-JAN-2015', 'DD-MON-YYYY')),
    PARTITION unknown VALUES (MAXVALUE)
);

-- Insert data into the table
INSERT INTO customer
VALUES (1, 'John', 'MH', TO_DATE('01-JAN-2003', 'DD-MON-YYYY'), 100, 1000);
INSERT INTO customer
VALUES (2, 'David', 'TN', TO_DATE('01-JAN-2008', 'DD-MON-YYYY'), 200, 2000);
INSERT INTO customer
VALUES (3, 'Sara', 'UP', TO_DATE('01-JAN-2013', 'DD-MON-YYYY'), 300, 3000);
INSERT INTO customer
VALUES (4, 'Michael', 'RJ', TO_DATE('01-JAN-2017', 'DD-MON-YYYY'), 400, 4000);

-- Check partitions of the data
SELECT cust_id, cust_name, cust_state, time_id, DBMS_ROWID.ROWID_OBJECT(rowid)
FROM customer;
```

Q8. Create a table with range partition as follows:

```
sql
-- Create the parent table Orders with range partitioning on Order Date
CREATE TABLE Orders
(
    order_id    NUMBER,
    order_date  DATE,
    customer_id NUMBER,
    shipper_id  NUMBER,
    CONSTRAINT pk_orders PRIMARY KEY (order_id)
) PARTITION BY RANGE (order_date) INTERVAL (NUMTOYMINTERVAL(3, 'MONTH'))
```

```
(
    PARTITION q1_orders VALUES LESS THAN (TO_DATE('01-APR-2022', 'DD-MON-YYYY')),
    PARTITION q2_orders VALUES LESS THAN (TO_DATE('01-JUL-2022', 'DD-MON-YYYY')),
    PARTITION q3_orders VALUES LESS THAN (TO_DATE('01-OCT-2022', 'DD-MON-YYYY')),
    PARTITION q4_orders VALUES LESS THAN (TO_DATE('01-JAN-2023', 'DD-MON-YYYY'))
);

-- Create the child table order_items with reference partitioning on order_id
CREATE TABLE order_items
(
    order_id   NUMBER,
    product_id NUMBER,
    price      NUMBER,
    quantity   NUMBER,
    CONSTRAINT fk_order_items_orders
        FOREIGN KEY (order_id)
            REFERENCES Orders (order_id)
) PARTITION BY REFERENCE(fk_order_items_orders);

-- Delete the partitions and drop the tables
ALTER TABLE Orders
    DROP PARTITION q1_orders;
ALTER TABLE Orders
    DROP PARTITION q2_orders;
ALTER TABLE Orders
    DROP PARTITION q3_orders;
ALTER TABLE Orders
    DROP PARTITION q4_orders;
DROP TABLE order_items;
DROP TABLE Orders;
```