PARTIȚIONĂRI

1. Se dă tabela *vanzari_ord_date_*** cu următoarele caracteristici:

- reprezintă o copie a tabelei *vanzari*;
- este partiționată prin ordonare, în submulțimi lunare corespunzătoare anului 2007, cheia de partiționare fiind coloana *timp_id* (de tip *date*)

a. Creați tabela *vanzari_ord_date_****, rulând următoarea comandă

```
CREATE TABLE vanzari_ord_date_***
PARTITION BY RANGE(timp_id)
(  PARTITION vanzari_jan2007
     VALUES LESS THAN(TO_DATE('01/02/2007','DD/MM/YYYY')),
   PARTITION vanzari_feb2007
     VALUES LESS THAN(TO_DATE('01/03/2007','DD/MM/YYYY')),
   PARTITION vanzari_mar2007
     VALUES LESS THAN(TO_DATE('01/04/2007','DD/MM/YYYY')),
   PARTITION vanzari_apr2007
     VALUES LESS THAN(TO_DATE('01/05/2007','DD/MM/YYYY')),
   PARTITION vanzari_mai2007
     VALUES LESS THAN(TO_DATE('01/06/2007','DD/MM/YYYY')),
   PARTITION vanzari_iun2007
     VALUES LESS THAN(TO_DATE('01/07/2007','DD/MM/YYYY')),
   PARTITION vanzari_iul2007
     VALUES LESS THAN(TO_DATE('01/08/2007','DD/MM/YYYY')),
   PARTITION vanzari_aug2007
     VALUES LESS THAN(TO_DATE('01/09/2007','DD/MM/YYYY')),
   PARTITION vanzari_sep2007
     VALUES LESS THAN(TO_DATE('01/10/2007','DD/MM/YYYY')),
   PARTITION vanzari_oct2007
     VALUES LESS THAN(TO_DATE('01/11/2007','DD/MM/YYYY')),
   PARTITION vanzari_nov2007
        VALUES LESS THAN(TO_DATE('01/12/2007','DD/MM/YYYY')),
   PARTITION vanzari_dec2007
        VALUES LESS THAN(TO_DATE('01/01/2008','DD/MM/YYYY')),
      PARTITION vanzari_rest
        VALUES LESS THAN (MAXVALUE))
   AS
   SELECT *
   FROM   vanzari;
```

b. Afișați numărul de produse vândute în fiecare lună. Colectați statistici pentru tabela *vanzari_ord_date_****. Consultați planul de execuție al cererii. Sunt accesate toate submulțimile tabelei?

```
ANALYZE TABLE nume_tabela COMPUTE STATISTICS;

EXPLAIN PLAN
SET STATEMENT_ID = 'st_1b_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_1b_***','serial'));
```

c. Consultați planul de execuției al cererii care utilizează tabela nepartiționată *vanzari*. Comparați cele două costuri. Ce observați?

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_1c_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_1c_***','serial'));
```

d. Afișați numărul de produse vândute în luna martie a anului 2007, utilizând în cerere următorul predicat:

```
TO_CHAR(timp_id,'YYYY-MM') = '2007-03'
```

Consultați planul de execuție al cererii. Sunt accesate toate partițiile tabelului?

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_1d_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_1d_***','serial'));
```

e. Afișați numărul de produse vândute în luna martie a anului 2007, utilizând în cerere următoarele predicate:

```
EXTRACT(YEAR FROM timp_id) = 2007
AND
EXTRACT(MONTH FROM timp_id) = 3
```

Consultați planul de execuție al cererii. Sunt accesate toate partițiile tabelului?

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_1e_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_1e_***','serial'));
```

f. Afișați numărul de produse vândute în luna martie a anului 2007, utilizând în cerere următorul predicat:

```
timp_id BETWEEN TO_DATE('01/03/2007','DD/MM/YYYY')
            AND TO_DATE('31/03/2007','DD/MM/YYYY');
```

Consultați planul de execuție al cererii. Sunt accesate toate partițiile tabelului?

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_1f_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_1f_***','serial'));
```

g. Consultați planul de execuției al cererii anterioare care utilizează tabela nepartiționată *vanzari*. Comparați cele două costuri. Ce observați?

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_1g_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_1g_***','serial'));
```

h. Afișați numărul de produse vândute în luna martie a anului 2007, precizând submulțimea partiției necesară cererii:

```
... FROM vanzari_ord_date_*** PARTITION(vanzari_mar2007)
```

Consultați planul de execuție al cererii. Comparați costul cu cel estimat pentru cererea de la punctul f.

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_1h_***'
```

```
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_1h_***','serial'));
```

2. Se dă tabela *vanzari_ord_number_**** cu următoarele caracteristici:

- reprezintă o copie parțială a tabelei *vanzari* (conține coloanele *id*, *produs_id*, *clienti_id*, *timp_id*, *factura*, *cantitate*, *pret_unitar_vanzare*, coloana *timp_id* fiind de tip *number*);
- este partiționată prin ordonare, în submulțimi lunare corespunzătoare anului 2007, cheia de partiționare fiind coloana *timp_id* (de tip *number*, reprezentând *Julian Day* corespunzătoare datei calendaristice la care este emisă factura - data calendaristică exprimată ca număr de zile scurse începând de la 1 ianuarie 4712 B.C.)

a. Creați tabela *vanzari_ord_number_****, rulând următoarea comandă

```
CREATE TABLE vanzari_ord_number_***
(id                  NUMBER,
 produs_id           NUMBER(4),
 client_id           NUMBER(4),
 timp_id             NUMBER(10),
 factura             NUMBER(8),
 cantitate           NUMBER(4),
 pret_unitar_vanzare NUMBER(10,2))
PARTITION BY RANGE(timp_id)
(
   PARTITION vanzari_jan2007
     VALUES LESS THAN (2454133),
     -- 2454133 = TO_CHAR(TO_DATE('01/02/2007','DD/MM/YYYY'),'J')
   PARTITION vanzari_feb2007
     VALUES LESS THAN (2454161),
     -- 2454161 = TO_CHAR(TO_DATE('01/03/2007','DD/MM/YYYY'),'J')
   PARTITION vanzari_mar2007
     VALUES LESS THAN (2454192),
     -- 2454192 = TO_CHAR(TO_DATE('01/04/2007','DD/MM/YYYY'),'J')
   PARTITION vanzari_apr2007
     VALUES LESS THAN (2454222),
     -- 2454222 = TO_CHAR(TO_DATE('01/05/2007','DD/MM/YYYY'),'J')
   PARTITION vanzari_mai2007
     VALUES LESS THAN (2454253),
     -- 2454253 = TO_CHAR(TO_DATE('01/06/2007','DD/MM/YYYY'),'J')
   PARTITION vanzari_iun2007
```

```
    VALUES LESS THAN (2454283),
    -- 2454283 = TO_CHAR(TO_DATE('01/07/2007','DD/MM/YYYY'),'J')
       PARTITION vanzari_iul2007
    VALUES LESS THAN (2454314),
    -- 2454314 = TO_CHAR(TO_DATE('01/08/2007','DD/MM/YYYY'),'J')
       PARTITION vanzari_aug2007
    VALUES LESS THAN (2454345),
    -- 2454345 = TO_CHAR(TO_DATE('01/09/2007','DD/MM/YYYY'),'J')
       PARTITION vanzari_sept2007
    VALUES LESS THAN (2454375),
    -- 2454375 = TO_CHAR(TO_DATE('01/10/2007','DD/MM/YYYY'),'J')
       PARTITION vanzari_oct2007
    VALUES LESS THAN (2454406),
    -- 2454406 = TO_CHAR(TO_DATE('01/11/2007','DD/MM/YYYY'),'J')
       PARTITION vanzari_nov2007
    VALUES LESS THAN (2454436),
    -- 2454436 = TO_CHAR(TO_DATE('01/12/2007','DD/MM/YYYY'),'J')
       PARTITION vanzari_dec2007
    VALUES LESS THAN (2454467),
    -- 2454467 = TO_CHAR(TO_DATE('01/01/2008','DD/MM/YYYY'),'J')
  PARTITION vanzari_rest
    VALUES LESS THAN (MAXVALUE)
);
```

b. Populați cu date tabela *vanzari_ord_number_*** folosind ca sursă tabela *vanzari.* Permanentizați tranzacția. Colectați statistici pentru tabela *vanzari_ord_number_****.

```
ANALYZE TABLE nume_tabela COMPUTE STATISTICS;
```

c. Definiți tabela *timp_ord_number_*** cu următoarele caracteristici:

- este o copie parțială a tabelei *timp* (conține coloanele *luna*, *luna_an*, *trimestru*, *trimestru_an*, *an*);
- cheia primară este coloana *id_timp*, de tip *number*, reprezentând *Julian Day* corespunzătoare datei calendaristice la care este emisă factura (data calendaristică exprimată ca număr de zile scurse începând de la 1 ianuarie 4712 B.C.);
- conține coloana *data* de tip date;
- este partiționată prin ordonare, în partiții lunare corespunzătoare anului 2007, cheia de partiționare fiind coloana *id_timp.*

```
CREATE TABLE timp_ord_number_***
 (id_timp    NUMBER PRIMARY KEY RELY DISABLE NOVALIDATE,
  data       DATE,
```

```
  luna       NUMBER(2),
  luna_an    VARCHAR2(7),
  trimestru     NUMBER(1),
  trimestru_an  VARCHAR2(6),
  an            NUMBER(4))
PARTITION BY RANGE(id_timp)
(  PARTITION timp_jan2007
     VALUES LESS THAN (2454133),
     -- 2454133 = TO_CHAR(TO_DATE('01/02/2007','DD/MM/YYYY'),'J')
   PARTITION timp_feb2007
     VALUES LESS THAN (2454161),
     -- 2454161 = TO_CHAR(TO_DATE('01/03/2007','DD/MM/YYYY'),'J')
   PARTITION timp_mar2007
     VALUES LESS THAN (2454192),
     -- 2454192 = TO_CHAR(TO_DATE('01/04/2007','DD/MM/YYYY'),'J')
   PARTITION timp_apr2007
     VALUES LESS THAN (2454222),
     -- 2454222 = TO_CHAR(TO_DATE('01/05/2007','DD/MM/YYYY'),'J')
   PARTITION timp_mai2007
     VALUES LESS THAN (2454253),
     -- 2454253 = TO_CHAR(TO_DATE('01/06/2007','DD/MM/YYYY'),'J')
   PARTITION timp_iun2007
     VALUES LESS THAN (2454283),
     -- 2454283 = TO_CHAR(TO_DATE('01/07/2007','DD/MM/YYYY'),'J')
        PARTITION timp_iul2007
     VALUES LESS THAN (2454314),
     -- 2454314 = TO_CHAR(TO_DATE('01/08/2007','DD/MM/YYYY'),'J')
        PARTITION timp_aug2007
     VALUES LESS THAN (2454345),
     -- 2454345 = TO_CHAR(TO_DATE('01/09/2007','DD/MM/YYYY'),'J')
        PARTITION vanzari_sept2007
     VALUES LESS THAN (2454375),
     -- 2454375 = TO_CHAR(TO_DATE('01/10/2007','DD/MM/YYYY'),'J')
        PARTITION timp_oct2007
     VALUES LESS THAN (2454406),
     -- 2454406 = TO_CHAR(TO_DATE('01/11/2007','DD/MM/YYYY'),'J')
        PARTITION timp_nov2007
     VALUES LESS THAN (2454436),
     -- 2454436 = TO_CHAR(TO_DATE('01/12/2007','DD/MM/YYYY'),'J')
        PARTITION timp_dec2007
     VALUES LESS THAN (2454467),
     -- 2454467 = TO_CHAR(TO_DATE('01/01/2008','DD/MM/YYYY'),'J')
   PARTITION timp_rest
```

```
    VALUES LESS THAN (MAXVALUE));
```

d. Populați cu date tabela *timp_ord_number_*** creată folosind ca sursă tabela *vanzari.* Permanentizați tranzacția. Colectați statistici pentru tabela *timp_ord_number_****. Adăugați constrângerea de cheie externă între cele 2 tabele nou definite (*vanzari_ord_number_*** și *timp_ord_number_***) cu opțiunile *RELY*, *DISABLE*, *NOVALIDATE*.

```
ANALYZE TABLE nume_tabela COMPUTE STATISTICS;

ALTER TABLE nume_tabelă
ADD CONSTRAINT nume_constrâmgere
FOREIGN KEY (coloană) REFERENCES nume_tabelă(coloană)
RELY DISABLE NOVALIDATE;
```

e. Afișați numărul de produse vândute în fiecare lună. Consultați planul de execuție al cererii. Verificați dacă se realizează operații de *join* la nivel de submulțimi.

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_2e_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_2e_***','serial'));
```

f. Afișați numărul de produse vândute în luna martie a anului 2007, folosind următorul predicat:

```
luna_an='2007-03'
```

Consultați planul de execuție al cererii.

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_2f_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_2f_***','serial'));
```

g. Afișați numărul de produse vândute în luna martie a anului 2007, precizând submulțimile necesare cererii:

```
... FROM vanzari_ord_number_*** PARTITION(vanzari_mar2007) v,
         timp_ord_number_***   PARTITION(timp_mar2007) t
```

Consultați planul de execuție al cererii.

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_2g_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_2g_***','serial'));
```

3. Se dă tabela *vanzari_hash_*** cu următoarele caracteristici:

- reprezintă o copie a tabelei *vanzari*;
- este partiționată *hash*, în 4 submulțimi, cheia de partiționare fiind coloana *produs_id*.

a. Creați tabela *vanzari_hash_***, rulând următoarea comandă

```
CREATE TABLE vanzari_hash_***
PARTITION BY HASH(produs_id)
PARTITIONS 4
--nr de submultimi trebuie sa fie o putere a lui 2
AS
SELECT *
FROM   vanzari;
```

b. Afișați pentru fiecare produs cantitatea vândută. Colectați statistici pentru tabela *vanzari_hash_***. Consultați planul de execuție al cererii. Sunt accesate toate submulțimile tabelei?

```
ANALYZE TABLE nume_tabela COMPUTE STATISTICS;

EXPLAIN PLAN
SET STATEMENT_ID = 'st_3b_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_3b_***','serial'));
```

c. Consultați planul de execuției al cererii care utilizează tabela nepartiționată *vanzari*. Comparați cele două costuri. Ce observați?

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_3c_***'
```

```
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_3c_***','serial'));
```

d. Afișați cantitatea vândută pentru produsul având codul 3010. Consultați planul de execuției al cererii.

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_3d_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
 table(dbms_xplan.display('plan_table','st_3d_***','serial'));
```

e. Afișați cantitatea vândută pentru produsul având codul 3010, precizând submulțimea care conține datele necesare cererii. Consultați planul de execuției al cererii.

```
... FROM   vanzari_hash_*** PARTITION (nume_submulțime);

SELECT PARTITION_NAME
FROM   USER_TAB_PARTITIONS
WHERE  TABLE_NAME = UPPER('vanzari_hash_***')
AND    PARTITION_POSITION = ORA_HASH('3010', 4 - 1) + 1;

EXPLAIN PLAN
SET STATEMENT_ID = 'st_3e_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
 table(dbms_xplan.display('plan_table','st_3e_***','serial'));
```

4. Se dă tabela *produse_list_**** cu următoarele caracteristici:

- reprezintă o copie a tabelei *produse*;
- este partiționată prin listă, cheia de partiționare fiind coloana *categorie_1* (valorile corespunzătoare celor 4 submulțimi fiind *IT*, *Papetarie*, *Industriale*, *Mobilier*).

a. Creați tabela *vanzari_list_****, rulând următoarea comandă

```
CREATE TABLE produse_list_***
  PARTITION BY LIST(categorie_1)(
```

```
   PARTITION prod_IT
      VALUES('IT'),
   PARTITION prod_papetarie
      VALUES('Papetarie'),
   PARTITION prod_industriale
      VALUES('Industriale'),
   PARTITION prod_mobilier
      VALUES('Mobilier'),
   PARTITION prod_altele VALUES(DEFAULT))
AS
SELECT *
FROM   produse;
```

b. Afișați numărul de produse din fiecare raion (*categorie_1*). Colectați statistici pentru tabela *produse_list_****. Consultați planul de execuție al cererii. Sunt accesate toate submulțimile tabelei?

```
ANALYZE TABLE nume_tabela COMPUTE STATISTICS;

EXPLAIN PLAN
SET STATEMENT_ID = 'st_4b_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_4b_***','serial'));
```

c. Consultați planul de execuției al cererii care utilizează tabela nepartiționată *produse*. Comparați cele două costuri. Ce observați?

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_4c_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_4c_***','serial'));
```

d. Afișați numărul de produse din raionul IT, folosind următorul predicat:

```
Categorie_1='IT'
```

Consultați planul de execuție al cererii.

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_4d_***'
```

```
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_4d_***','serial'));
```

e. Afișați numărul de produse din raionul IT, precizând submulțimea partiției necesară cererii:

```
FROM produse_list_*** PARTITION(prod_IT)
```

Consultați planul de execuție al cererii. Comparați costul cu cel estimat pentru cererea de la punctul d.

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_4e_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_4e_***','serial'));
```

5. Se dă tabela *vanzari_ord_hash_**** cu următoarele caracteristici:

- reprezintă o copie parțială a tabelei *vanzari*;
- este partiționată prin ordonare, utilizând cheia de partiționare *timp_id*, iar fiecare submulțime obținută este divizată în subpartiții *hash*, utilizând cheia de partiționare *produs_id*.

a. Creați tabela *vanzari_ord_hash_****, rulând următoarea comandă

```
CREATE TABLE vanzari_ord_hash_***
   (id                  NUMBER,
    produs_id           NUMBER(4),
    client_id           NUMBER(4),
    timp_id             DATE,
    factura             NUMBER(8),
    cantitate           NUMBER(4),
    pret_unitar_vanzare NUMBER(10,2))
PARTITION BY RANGE (timp_id)
SUBPARTITION BY HASH (produs_id) SUBPARTITIONS 4
(PARTITION vanz2007t1
  VALUES LESS THAN (TO_DATE('01/04/2007','DD/MM/YYYY')),
 PARTITION vanz2007t2
  VALUES LESS THAN (TO_DATE('01/07/2007','DD/MM/YYYY')),
 PARTITION vanz2007t3
```

```
 VALUES LESS THAN (TO_DATE('01/10/2007','DD/MM/YYYY')),
PARTITION vanz2007t4
 VALUES LESS THAN (TO_DATE('01/01/2008','DD/MM/YYYY')));
```

b. Populați cu date tabela *vanzari_ord_hash_**** folosind ca sursă tabela *vanzari.* Permanentizați tranzacția. Colectați statistici pentru tabela *vanzari_ord_hash_****.

```
ANALYZE TABLE nume_tabela COMPUTE STATISTICS;
```

c. Afișați cantitatea vândută din produsul având codul 3010 în primul trimestru al anului 2007. Consultați planul de execuției al cererii.

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_5c_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_5c_***','serial'));
```

d. Consultați planul de execuției al cererii care utilizează tabela nepartiționată *vanzari*. Comparați cele două costuri. Ce observați?

```
EXPLAIN PLAN
SET STATEMENT_ID = 'st_5d_***'
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_5d_***','serial'));
```

e. În ce submulțime a partiției se găsesc datele necesare cererii?

```
SELECT SUBPARTITION_NAME, SUBPARTITION_POSITION
FROM   USER_TAB_SUBPARTITIONS
WHERE  TABLE_NAME = UPPER('vanzari_ord_hash_***')
AND    PARTITION_NAME = UPPER('vanz2007t1')
AND    SUBPARTITION_POSITION = ORA_HASH('3010', 4 - 1) + 1;
```

f. Afișați cantitatea vândută din produsul având codul 3010 în luna martie a anului 2007, precizând submulțimea partiției necesară cererii. Consultați planul de execuției al cererii.

```
... FROM   vanzari_ord_hash_***  SUBPARTITION (nume_submultime);

EXPLAIN PLAN
SET STATEMENT_ID = 'st_5f_***'
```

```
FOR cerere_SQL;

SELECT plan_table_output
FROM
table(dbms_xplan.display('plan_table','st_5f_***','serial'));
```

6. Consultați vizualizările din dicționarul datelor pentru a afla informații despre tabelele partiționate definite:

- *user_tables* (*table_name*, *partitioned*, *last_analyzed*)
- *user_tab_partitions* (*table_name*, *partition_name*, *subpartition_count*, *partition_position*)
- *user_tab_subpartitions* (*table_name*, *partition_name*, *subpartition_name*, *subpartition_position*)