Eroarea de tip Mutating Table în PL/SQL

Eroarea de tip mutating table este una dintre cele mai frecvente și mai dificil de înțeles erori întâlnite în dezvoltarea cu PL/SQL, în special în lucrul cu triggeri de tip DML la nivel de linie. Această eroare apare atunci când un trigger încearcă să acceseze (prin SELECT, INSERT, UPDATE sau DELETE) un tabel care se află deja într-un proces de modificare, ceea ce ar putea conduce la rezultate inconsistente sau la bucle recursive.

Ce este mutating table

Un „mutating table" este un tabel care se află într-o stare intermediară de modificare ca urmare a executării unei comenzi DML (INSERT, UPDATE sau DELETE). În timpul acestei operații, Oracle nu permite citirea sau modificarea aceluiași tabel din interiorul unui trigger row-level, deoarece datele nu sunt încă stabile și consistente.

Mesajul de eroare tipic este: ORA-04091: table <nume_tabel> is mutating, trigger/function may not see it

Această restricție este impusă de Oracle pentru a preveni:

- citirea unor date parțial modificate,
- recursivitatea necontrolată,
- inconsistența tranzacțiilor,
- bucle infinite de declanșare a triggerilor.

De ce apare eroarea mutating table

Eroarea apare în mod specific atunci când sunt îndeplinite simultan următoarele condiții:

1 Există un trigger DML la nivel de linie (FOR EACH ROW).

2 Triggerul este definit pe un anumit tabel (ex.: angajati).

3 În interiorul triggerului se încearcă:

- citirea acelui tabel prin SELECT,
- sau modificarea lui prin INSERT, UPDATE, DELETE.

Exemplu de situație problematică:

```
CREATE OR REPLACE TRIGGER trg_angajati
BEFORE UPDATE ON angajati
FOR EACH ROW
BEGIN
  SELECT COUNT(*) INTO v_nr
  FROM angajati
  WHERE salariu > :NEW.salariu;  -- mutating table
END;
```

Tabelul angajati este deja în curs de modificare, iar Oracle nu permite accesul la el în acest context.

Când NU apare eroarea mutating table

- În triggerii statement-level (fără FOR EACH ROW).
- În blocuri PL/SQL obișnuite (proceduri, funcții).
- Când se accesează alte tabele decât cel pe care este definit triggerul.
- În triggerii de tip AFTER STATEMENT.

Moduri de tratare a erorii mutating table

Există mai multe soluții consacrate pentru evitarea sau rezolvarea acestei erori, fiecare potrivită unui anumit scenariu.

1. Utilizarea triggerilor la nivel de comandă (STATEMENT-LEVEL)

Înlocuirea triggerului row-level cu unul statement-level permite accesul la tabel, deoarece modificarea a fost deja finalizată la nivel logic.

2. Separarea logicii în doi triggeri (ROW + STATEMENT). Se folosește:

- un trigger BEFORE/AFTER ROW pentru a salva datele necesare,
- un trigger AFTER STATEMENT pentru procesarea finală.

Datele pot fi salvate în variabile de pachet sau tabele termporare

2a. Utilizarea variabilelor de pachet (package variables)

Variabilele din pachete își păstrează valoarea pe durata sesiunii și pot fi utilizate pentru a stoca date intermediare.

```
-- pachet
CREATE OR REPLACE PACKAGE p_aux IS
  TYPE t_ids IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
  v_ids t_ids;
END;

-- trigger row-level
CREATE OR REPLACE TRIGGER trg_row
AFTER UPDATE ON angajati
FOR EACH ROW
BEGIN
  p_aux.v_ids(p_aux.v_ids.COUNT + 1) := :NEW.id;
END;

-- trigger statement-level
CREATE OR REPLACE TRIGGER trg_stmt
AFTER UPDATE ON angajati
BEGIN
  FOR i IN p_aux.v_ids.FIRST .. p_aux.v_ids.LAST LOOP
    -- acces permis la tabel
    UPDATE angajati SET ...
  END LOOP;
END;
```

2b. Utilizarea tabelelor temporare (GLOBAL TEMPORARY TABLES)

Se salvează datele afectate de trigger în tabele temporare, iar procesarea se face ulterior.

3. Refactorizarea logicii (fără trigger). De multe ori, triggerul poate fi eliminat complet și înlocuit cu proceduri explicite, logică în aplicație sau constrângeri declarative.

De ce Oracle interzice accesul la o mutating table

Oracle nu permite acest comportament deoarece:

- rândurile nu sunt complet procesate,
- rezultatele SELECT-ului ar fi nedeterministe,
- ar putea apărea bucle recursive infinite,
- ar fi încălcat principiul izolării tranzacțiilor.

Aceasta este o decizie de design pentru stabilitate și consistență.

Structura bazei de date Oracle

O bază de date Oracle este un sistem complex, organizat pe mai multe niveluri, care permite stocarea sigură, eficientă și coerentă a datelor. Pentru a înțelege modul în care funcționează o bază de date, este esențial să analizăm structura sa fizică, structura logică și mecanismele interne de descriere și administrare, reunite în dicționarul datelor.

Această organizare pe niveluri oferă independență între modul de stocare fizică a datelor și modul în care acestea sunt accesate logic de către utilizatori și aplicații.

Structura fizică a bazei de date

Structura fizică reprezintă modul concret în care datele sunt stocate pe disc. Ea este alcătuită din fișiere gestionate de sistemul de operare, dar controlate exclusiv de motorul Oracle.Principalele tipuri de fișiere fizice sunt:

1. Fișierele de date (Data Files)

Fișierele de date sunt cele mai importante componente ale bazei de date, deoarece conțin efectiv datele utilizatorilor: tabele, indecși, segmente temporare etc.

Caracteristici:

- Fiecare fișier de date aparține unui tablespace.
- O bază de date poate conține unul sau mai multe fișiere de date.
- Datele sunt organizate în blocuri gestionate de Oracle.

Exemple de informații stocate:

- rânduri din tabele,
- structuri de index,
- date temporare,
- segmente de rollback.

Fără fișierele de date, baza de date nu poate funcționa.

2. Fișierele de control (Control Files)

Fișierele de control conțin informații critice despre structura bazei de date. Ele sunt necesare la pornirea bazei de date. Conțin informații precum:

- numele bazei de date,
- locația fișierelor de date și a fișierelor de redo log,
- numerele de secvență ale jurnalelor,
- starea checkpoint-urilor.

Caracteristici importante:

- Sunt de dimensiuni relativ mici.
- Sunt replicate (de obicei minim 2 sau 3 copii) pentru siguranță.
- Dacă fișierele de control sunt pierdute, baza de date nu mai poate porni.

3. Fișierele de reluare (Redo Log Files)

Fișierele redo log înregistrează toate modificările efectuate asupra bazei de date, indiferent dacă tranzacțiile sunt confirmate (COMMIT) sau nu.

Rolul lor principal:

- permit recuperarea bazei de date în cazul unei căderi (crash recovery),
- asigură durabilitatea tranzacțiilor.

Caracteristici:

- sunt organizate în grupuri,
- sunt scrise secvențial,
- sunt reutilizate circular,
- pot fi arhivate (ARCHIVELOG mode).

Fără redo log-uri, Oracle nu poate garanta consistența datelor.

Structura logică a bazei de date

Structura logică reprezintă modul în care Oracle organizează și prezintă datele utilizatorilor, independent de locația fizică a acestora pe disc. Această structură este ierarhică și include următoarele componente:

1. Blocurile de date (Data Blocks) - Blocul de date este cea mai mică unitate logică de stocare gestionată de Oracle.

Caracteristici:

- Dimensiunea este stabilită la crearea bazei de date (ex.: 8KB).
- Un bloc poate conține datele propriu-zise, informații de control, spațiu liber pentru extinderi.

Blocurile sunt citite și scrise în memorie ca unități indivizibile.

2. Extensiile (Extents) - O extensie este o colecție de blocuri de date contigue, alocate simultan unui obiect.

Rol:

- facilitează alocarea eficientă a spațiului,
- permit creșterea progresivă a obiectelor.
- Un obiect poate avea una sau mai multe extensii, în funcție de mărimea sa.

3. Segmentele (Segments) - Un segment reprezintă spațiul total ocupat de un obiect logic din baza de date. Tipuri de segmente:

- segmente de tabel
- segmente de index
- segmente temporare
- segmente undo (rollback)

Un segment este format din una sau mai multe extensii.

4. Spațiile tabel (Tablespaces) - Tablespace-ul este unitatea logică de stocare care grupează segmentele și face legătura cu structura fizică.

Caracteristici:

- fiecare tablespace conține unul sau mai multe fișiere de date,
- un obiect aparține unui singur tablespace,
- tablespace-urile permit: separarea logică a datelor, administrarea spațiului, optimizarea performanței.

Exemple:

SYSTEM (dicționarul datelor),

USERS,

TEMP,

UNDO.

5. Schemele de obiecte (Schemas) - O schemă reprezintă colecția de obiecte aparținând unui utilizator.

Conține: tabele, vederi, indecși, proceduri, funcții, pachete, triggeri.

Schema nu ocupă spațiu fizic; obiectele din schemă sunt stocate în tablespace-uri.

Dicționarul datelor (Data Dictionary)

Dicționarul datelor este un set de tabele și vederi interne care descriu structura și metadatele bazei de date. El este creat și întreținut automat de Oracle.

Particularități ale dicționarului datelor:

- Este stocat în tablespace-ul SYSTEM.
- Este accesibil doar în citire pentru utilizatori.
- Este actualizat automat la fiecare operație DDL.
- Este esențial pentru funcționarea bazei de date.
- Oracle consultă dicționarul datelor pentru orice operație SQL.

Categorii de vederi din dicționar

USER_* informații despre obiectele utilizatorului curent.

ALL_* informații despre obiectele accesibile utilizatorului.

DBA_* informații despre întreaga bază de date (necesită privilegii).

Exemple de vederi importante

USER_TABLES

USER_OBJECTS

USER_TAB_COLUMNS

USER_CONSTRAINTS

USER_INDEXES

DBA_TABLESPACES

V$DATABASE, V$INSTANCE (vederi dinamice)

Utilitatea dicționarului datelor

Dicționarul datelor este utilizat pentru:

- verificarea existenței obiectelor,
- analiza structurii tabelelor,
- identificarea dependențelor dintre obiecte,
- monitorizarea stării bazei de date,
- audit și securitate,
- depanare și optimizare.

Aplicații practice ale dicționarului

- generarea automată de scripturi DDL,
- validarea dinamică a obiectelor,
- analizarea erorilor de compilare,
- documentarea structurii bazei de date,
- administrarea utilizatorilor și rolurilor.

Exemplu de interogare:

```
SELECT object_name, object_type, status
FROM user_objects
WHERE status = 'INVALID';
```

Exercitii

a) Introduceti un trigger care sa respinga orice inserare in tabelul STUDENTS cu media mai mica ca cea mai mica medie din tabel.
b) Selectati informatii despre toti triggerii creati in baza de date
c) Afisati cate obiecte au fost create in fiecare namespace
d) Afisati pentru fiecare tip de date cate coloane cu acesta exista in baza de date
e) Calculati pentru fiecare tabel lungimea ocupata in total de coloanele de tip varchar din cadrul sau