SQL - 6

ACTUALIZAREA DATELOR

STUD

MATR	NUME	AN	GRUPA	DATAN	LOC	TUTOR	PUNCTAJ	CODS
1456	GEORGE	4	1141A	12-MAR-82	BUCURESTI		2890	11
1325	VASILE	2	1122A	05-OCT-84	PITESTI	1456	390	11
1645	MARIA	3	1131B	17-JUN-83	PLOIESTI		1400	11
3145	ION	1	2112B	24-JAN-85	PLOIESTI	3251	1670	21
2146	STANCA	4	2141A	15-MAY-82	BUCURESTI		620	21
3251	ALEX	5	2153B	07-NOV-81	BRASOV		1570	21
2215	ELENA	2	2122A	29-AUG-84	BUCURESTI	2146	890	21
4311	ADRIAN	3	2431A	31-JUL-83	BUCURESTI		450	24
3514	FLOREA	5	2452B	03-FEB-81	BRASOV		3230	24
1925	OANA	2	2421A	20-DEC-84	BUCURESTI	4311	760	24
2101	MARIUS	1	2412B	02-SEP-85	PITESTI	3514	310	24
4705	VOICU	2	2421B	19-APR-84	BRASOV	4311	1290	24

SPEC si BURSA

CODS	NUME	DOMENIU
11	MATEMATICA	STIINTE EXACTE
21	GEOGRAFIE	UMANIST
24	ISTORIE	UMANIST

TIP	PMIN	PMAX	SUMA
FARA BURSA	0	399	
BURSA SOCIALA	400	899	100
BURSA DE STUDIU	900	1799	150
BURSA DE MERIT	1800	2499	200
BURSA DE EXCEPTIE	2500	9999	300

OBIECTIV

- Scopul acestui capitol este descrierea operaţiilor care fac parte din limbajul de manipulare al datelor: adăugarea de linii într-o tabelă, modificarea valorilor unor linii şi ştergerea acestora.
- Este de asemenea prezentat mecanismul tranzacţiilor şi consistenţa la citire care permit utilizatorilor şi aplicaţiilor care accesează o bază de date să utilizeze o versiune consistentă a acesteia.

CERERI PREZENTATE

- Principalele cereri SQL prin care sunt efectuate aceste operaţii sunt:
  - INSERT efectuează adăugarea de noi linii,
  - UPDATE permite actualizarea (modificarea) valorilor dintr-o tabelă,
  - DELETE şterge linii dintr-o tabelă,
  - MERGE efectuează o actualizare sau inserare, după cum linia îndeplineşte sau nu o condiţie,
  - COMMIT, ROLLBACK şi SAVEPOINT permit controlul explicit al tranzacţiilor.
- Este prezentată de asemenea clauza WITH CHECK OPTION care controlează inserarea, ştergerea şi actualizarea datelor în cazul în care aceste operaţii se efectuează prin intermediul unei subcereri.

INSERT ... VALUES (1)

Inserarea unei linii prin specificarea valorilor acesteia

- Sintaxa cererii este următoarea:

```
INSERT INTO tabela [(lista_de_coloane)]
   VALUES (lista_de_valori)
```

În cazul în care lista de coloane nu este prezentă în cerere efectul va fi următorul:

- Se inserează în tabelă o linie completă,
- Numărul valorilor din listă trebuie să fie acelaşi cu numărul de coloane din tabelă,
- Ordinea valorilor din listă trebuie să fie aceeaşi cu ordinea coloanelor din cererea de creare a tabelei,
- Tipul valorilor trebuie să fie compatibil cu tipul coloanelor în care se stochează,

INSERT ... VALUES (2)

- În cazul în care pe o anumită coloană se doreşte inserarea unei valori nule, în lista de valori se foloseşte cuvântul cheie NULL,
- Dacă la crearea tabelei unei coloane i-a fost asociată o valoare implicită, pentru inserarea ei se foloseşte cuvântul cheie DEFAULT. În cazul în care se foloseşte DEFAULT dar coloana nu are o valoare implicită definită, în coloană se va insera o valoare nulă,
- Valorile pentru coloanele de tip şir de caractere sau dată calendaristică (doar cele în formatul standard) se pun între apostrofi,
- În cazul în care o valoare de tip dată calendaristică nu este în format standard se foloseşte funcţia TO_DATE pentru conversia şirului respectiv în dată calendaristică,
- Fiecare linie se inserează cu o cerere separată.

EXEMPLU

- : Inserarea specializărilor în tabela SPEC se face cu cererile:

```
INSERT INTO SPEC VALUES(11,
  'MATEMATICA', 'STIINTE EXACTE');
INSERT INTO SPEC VALUES(21,
  'GEOGRAFIE', 'UMANIST');
INSERT INTO SPEC VALUES(24, 'ISTORIE',
  'UMANIST');
```

- Pentru inserarea unei noi specializări având codul 30, nume încă necunoscut şi un domeniu implicit se poate folosi cererea:

```
INSERT INTO SPEC VALUES(30, NULL,
  DEFAULT);
```

EFECT

- Rezultatul celor patru inserări va fi următorul (am considerat că la crearea tabelei SPEC, pentru coloana DOMENIU s-a asociat valoarea implicită 'UMANIST'):

```
CODS NUME        DOMENIU
----- ---------- ---------------
   11 MATEMATICA STIINTE EXACTE
   21 GEOGRAFIE  UMANIST
   24 ISTORIE    UMANIST
   30            UMANIST
```

INSERT ... VALUES (3)

- În cazul în care cererea conţine şi o **listă de coloane**, rezultatul acesteia este inserarea unei linii incomplete: pentru coloanele care lipsesc din listă se vor insera automat valorile implicite - dacă există - sau valori nule în caz contrar.
- Observaţii:
  - Listele de coloane şi valori trebuie să conţină acelaşi număr de elemente,
  - Cele două liste se corespund poziţional: prima valoare va fi stocată în prima coloană din listă, s.a.m.d.
  - Tipul valorilor trebuie să fie compatibil cu tipul coloanelor în care se stochează.

EXEMPLU

- Exemplu: inserarea specializării cu codul 30 se putea face şi astfel:

```
INSERT INTO SPEC (CODS) VALUES(30);
```

  rezultatul obţinut fiind acelaşi ca mai sus.
- În toate cazurile, inserarea va eşua dacă noua linie nu respectă vreuna din constrângerile de integritate care sunt active în acel moment pentru tabela în care se face adăugarea.

CE PUTEM PUNE IN LISTA

- În lista de valori se mai pot folosi:
  - Expresii
  - Subcereri

EXPRESII

- In acest caz valoarea expresiei trebuie să fie compatibilă cu tipul coloanei.
- Expresiile pot conţine operatori şi funcţii.
- Exemplu: următoarea cerere de inserare este validă:

```
INSERT INTO STUD (MATR, NUME, DATAN,
  CODS) VALUES (1200+15, 'ION ' ||
  'DOBRE', SYSDATE, 24);
```

- Valorile inserate vor fi: 1215 , 'ION DOBRE', data curentă şi 24.

SUBCERERI

- Rezultatul unei subcereri necolelate care întoarce o singură valoare poate fi folosit în lista de valori a cererii INSERT.
- Exemplu: cererea:

```
INSERT INTO SPEC(CODS)
  VALUES ((SELECT MAX(CODS) FROM SPEC)+1);
```

va insera o specializare având codul 25 (în cazul în care valoarea maximă pe coloana CODS este 24).

INSERT .. SELECT (1)

- Sintaxa este în acest caz următoarea:

```
INSERT INTO tabela [(lista_de_coloane)]
  cerere_select
```

- Modul de execuţie al acestei cereri este următorul:
  - Se execută cererea SELECT,
  - Liniile rezultatului sunt inserate în tabelă,
  - Numărul de coloane al tabelei/listei de coloane trebuie să fie egal cu cel al rezultatului cererii SELECT iar tipurile trebuie să fie compatibile,
  - În cazul în care o linie din rezultatul cererii SELECT nu satisface constrângerile de integritate ale tabelei în care se face inserarea, întreaga comandă eşuează (nici celelalte linii nu sunt inserate).

EXEMPLU

- Fie o tabelă NRSTUD(CODS, NRSTUD) care conţine pentru fiecare specializare numărul de studenţi al acesteia.
- Umplerea ei se poate face prin cererea:

```
INSERT INTO NRSTUD
  SELECT CODS, COUNT(*) FROM STUD GROUP
 BY CODS;
```

- Conţinutul tabelei NRSTUD va fi:

```
CODS NRSTUD
----- ------
   11      3
   21      4
   24      5
```

DELETE

- Sintaxa cererii care efectuează ştergerea unor linii dintr-o tabelă este următoarea:

```
DELETE FROM tabela
[WHERE conditie]
```

- În clauza WHERE pot să apară aceleaşi elemente ca şi în cazul cererilor SELECT (deci inclusiv subcereri).
- Efectul cererii este ştergerea tuturor liniilor pentru care condiţia este verificată.
- În cazul absenţei clauzei WHERE se şterg toate liniile tabelei.

EXEMPLE

- Exemplul 1: dacă se doreşte ştergerea tuturor studenţilor specializării cu codul 11 cererea este:
  - `DELETE FROM STUD WHERE CODS = 11;`
- Exemplul 2: ştergerea liniilor corespunzătoare studenţilor cu cel mai mic punctaj din fiecare specializare se face cu cererea:

```
DELETE FROM STUD
WHERE MATR IN
   (SELECT MATR FROM STUD
    WHERE (CODS, PUNCTAJ) IN
        (SELECT CODS, MIN(PUNCTAJ)
         FROM STUD GROUP BY CODS));
```

OBSERVATIE

- Stergerea nu se va efectua şi va semnala o eroare dacă una dintre liniile care îndeplinesc condiţia conţine o cheie referită printr-o constrângere de tip FOREIGN KEY într-o tabelă din baza de date.

UPDATE

- Sintaxa cererii de actualizare (modificare) a valorilor dintr-o tabelă este:

```
UPDATE tabela
SET coloana1 = expresie1 [,
  coloana2=expresie2, ...]
[WHERE conditie];
```

- Efectul este următorul:
  - În WHERE pot să apară aceleaşi elemente ca şi în cazul cererilor SELECT,
  - Toate liniile care îndeplinesc condiţia din WHERE vor fi actualizate conform cu specificaţiile din SET. În cazul absenţei clauzei WHERE, toate liniile tabelei vor fi actualizate,
  - Expresiile din clauza SET se evaluează pornind de la valorile conţinute în linia care se modifică şi ele trebuie să aibă un tip compatibil cu al coloanelor asociate,
  - O aceeaşi coloană nu poate apare de două ori în clauza SET.

EXEMPLE

- Exemplul 1: Mărirea cu 10% a punctajului studenţilor de la specializarea cu codul 11 se face cu cererea:

```
UPDATE STUD
SET PUNCTAJ = PUNCTAJ * 1.1
WHERE CODS = 11
```

- Exemplul 2: În cazul în care, pentru toţi studenţii, pe lângă mărirea punctajului cu 10% se doreşte şi incrementarea anului de studii pentru studenţii din anii 1-4 şi stocarea pe coloana AN a valorii 0 pentru studenţii de anul 5, cererea este:

```
UPDATE STUD
SET PUNCTAJ = PUNCTAJ * 1.1,
    AN = DECODE (AN, 5, 0, AN+1);
```

EXEMPLE – cont.

- Exemplul 3: În cazul în care pentru o coloană se doreşte actualizarea valorilor prin aducerea lor la valoarea implicită asociată la crearea tabelei sau la o valoare nulă se folosesc de asemenea cuvintele cheie DEFAULT şi NULL.
- În cazul folosirii lui DEFAULT, dacă nu există o valoare implicită asociată, în coloana respectivă se va stoca o valoare nulă.

```
UPDATE SPEC SET DOMENIU = DEFAULT;
UPDATE SPEC SET DOMENIU = NULL;
```

SUBCERERI IN SET

- Expresiile din clauza SET pot conţine subcereri pe tabela care se actualizează sau pe o altă tabelă din baza de date.
- Exemplu: Pentru actualizarea numărului de studenţi din tabela NRSTUD descrisă anterior se poate folosi următoarea cerere corelată:

```
UPDATE NRSTUD N
SET NRSTUD =
   (SELECT COUNT(*)
    FROM STUD
    WHERE CODS = N.CODS);
```

SUBCERERI IN SET – cont.

- Subcererile pot fi prezente atât pe clauza WHERE cât şi în locul numelui tabelei (acest caz mai tarziu).
- De exemplu, dacă se doreşte actualizarea numărului de studenţi doar pentru specializările având mai mult de trei studenţi cererea este:

```
UPDATE NRSTUD
SET NRSTUD =
   (SELECT COUNT(*)
    FROM STUD WHERE CODS = NRSTUD.CODS)
WHERE CODS IN
   (SELECT CODS FROM STUD GROUP BY CODS
    HAVING COUNT(*) > 3);
```

EFECT CONSTRANGERI

- În cazul în care noile valori ale unei linii nu satisfac constrângerile de integritate active pentru tabela respectivă, cererea va eşua şi va returna un mesaj de eroare.
- Este cazul actualizărilor în care:
  - Noua valoare este nulă şi coloana are o constrângere de integritate de tip NOT NULL,
  - Noua valoare face ca linia să conţină o valoare de cheie care mai există în tabelă, într-o altă linie,
  - Noua valoare nu verifică o constrângere de tip CHECK,
  - Vechea valoare a cheii este referită prin constrângeri de tip FOREIGN KEY în alte tabele,
  - Noua valoare a unei chei străine nu se regăseşte în tabela referită prin FOREIGN KEY.

MERGE

- O cerere MERGE efectuează fie actualizarea fie inserarea unor linii într-o tabelă, după cum acestea îndeplinesc sau nu o condiţie de tip join.
- Sintaxa cererii este următoarea:

```
MERGE INTO nume_tabela_1 [alias_tabela_1]
USING nume_tabela_2 | nume_vedere | subcerere
ON (conditie_de_join)
WHEN MATCHED THEN
  UPDATE SET
    coloana1 = expresie1
    [, coloana2=expresie2, ...]
WHEN NOT MATCHED THEN
  INSERT [(lista_coloane)] VALUES
  (lista_valori);
```

MERGE – cont.

- Tabela în care se fac actualizări/inserări este cea specificată în clauza MERGE INTO. În continuare vom denumi această tabelă *tabela destinatie*.
- Valorile actualizate / inserate sunt determinate de tabela / vederea / subcererea din clauza USING. În continuare vom denumi această tabelă *tabela sursa*.
- Decizia privind operaţia efectuată (actualizare sau inserare) este luată pe baza condiţiei de join dintre tabela sursă şi tabela destinaţie, specificată în clauza ON,
- În cazul în care linia curentă din tabela sursă are linii corespondente prin condiţia ON în tabela destinaţie, se execută o actualizare a acestor linii pe baza specificaţiilor din clauza UPDATE SET,
- În cazul în care linia curentă din tabela sursă nu are nici o linie corespondentă prin condiţia ON în tabela destinaţie, se execută inserarea unei noi linii pe baza specificaţiilor din INSERT,
- Liniile din tabela destinaţie care nu au corespondent prin condiţia ON nu sunt afectate de această cerere (nu sunt nici actualizate, nici şterse).

EXEMPLU

- În cazul în care conţinutul tabelei STUD se modifică, aducerea tabelei NRSTUD în concordanţă cu acesta se poate face prin cererea:

```
MERGE INTO NRSTUD
USING (SELECT CODS C, COUNT(*) NR
       FROM STUD GROUP BY CODS)
ON (NRSTUD.CODS = C)
WHEN MATCHED THEN
  UPDATE SET NRSTUD.NRSTUD = NR
WHEN NOT MATCHED THEN
  INSERT VALUES (C, NR);
```

EXEMPLU – cont.

- Se evaluează subcererea din clauza USING. Rezultatul conţine codurile specializărilor şi numărul de studenţi corespunzător acestora calculate pe baza datelor din tabela STUD.
- În cazul în care un cod de specializare există atât în tabela NRSTUD cât şi în rezultatul subcererii, linia corespunzătoare din NRSTUD este actualizată să reflecte noul număr de studenţi, cel din rezultatul subcererii.
- În cazul în care au apărut studenţi la specializări noi (linia din rezultatul subcererii conţine un cod de specializare inexistent în tabela NRSTUD) se inserează o nouă linie în NRSTUD conţinând datele referitoare la aceste specializări.

ACTUALIZARI PRIN SUBCERERI: CLAUZA WITH CHECK OPTION

- În cazul cererilor INSERT, DELETE şi UPDATE se poate folosi o subcerere în locul numelui tabelei al cărei conţinut se modifică.
- Aceste subcereri trebuie să satisfacă restricţiile care vor fi prezentate în detaliu în capitolul următor în paragraful privind actualizarea datelor prin intermediul vederilor.
- Este vorba în principal despre subcereri care se referă la o singură tabelă şi nu conţin funcţii de grup sau grupare şi nici nu au coloane definite prin expresii.

WITH CHECK OPTION - cont

- Pentru un mai bun control al operaţiilor efectuate prin intermediul subcererilor, acestea pot conţine clauza WITH CHECK OPTION care specifică:
  - În cazul unui INSERT, noua linie îndeplineşte condiţiile pentru a fi regăsită prin subcererea prin care se face inserarea,
  - În cazul lui UPDATE, noile valori ale liniilor actualizate îndeplinesc de asemenea condiţiile pentru regăsirea lor prin intermediul subcererii,
  - În cazul cererilor DELETE se pot şterge doar linii care au corespondent în rezultatul subcererii. Folosirea lui WITH CHECK OPTION, deşi permisă, nu are nici un efect.

EXEMPLE: INSERARE

Inserarea unei noi specializări, cu şi fără WITH CHECK OPTION:

- Varianta 1: fără WITH CHECK OPTION. Deşi noua linie nu poate fi regăsită prin subcerere (codul specializării nu îndeplineşte condiţia CODS > 11), ea va fi totuşi inserată:

```
INSERT INTO (SELECT * FROM SPEC WHERE CODS >
  11)
  VALUES (9, 'UNNUME', 'UNDOMENIU');
```

- Varianta 2: cu WITH CHECK OPTION. Deoarece noua linie nu poate fi regăsită prin subcerere ea nu va fi inserată şi se va semnala mesajul de eroare *ORA-01402: view WITH CHECK OPTION where-clause violation*:

```
INSERT INTO
  (SELECT * FROM SPEC
   WHERE CODS > 11
   WITH CHECK OPTION)
VALUES (9, 'UNNUME', 'UNDOMENIU');
```

EXEMPLE: STERGERE

- ***Stergerea*** unor specializări, cu şi fără WITH CHECK OPTION: Următoarele două cereri au acelaşi efect: ştergerea specializării cu codul 21.
- Specializarea cu codul 24 nu este ştearsă, deşi domeniul ei este 'UMANIST', pentru că ea nu este regăsită de subcerere:

```
DELETE (SELECT * FROM SPEC
        WHERE CODS < 24)
WHERE DOMENIU = 'UMANIST';
```

Sau cu WITH CHECK OPTION:

```
DELETE (SELECT * FROM SPEC
        WHERE CODS < 24 WITH CHECK OPTION)
WHERE DOMENIU = 'UMANIST';
```

EXEMPLE: ACTUALIZARE

- ***Actualizarea*** unor specializări, cu şi fără WITH CHECK OPTION:
- Varianta 1: fără WITH CHECK OPTION. Unele noi valori nu pot fi regăsite prin subcerere (21 şi 24 devin > 50) dar cererea se va executa fără erori:

```
UPDATE (SELECT * FROM SPEC WHERE CODS <
  50)
SET CODS = CODS + 30;
```

- Varianta 2: cu WITH CHECK OPTION. În acest caz cererea va semnala o eroare (codurile unora dintre specializări depăşesc 50. Eşecul este la nivel de cerere şi nu sunt actualizate nici liniile care prin această operaţie pot fi încă regăsite de subcerere (cazul specializării cu codul 11):

```
UPDATE (SELECT * FROM SPEC
        WHERE CODS < 50 WITH CHECK OPTION)
SET CODS = CODS + 30;
```

CONSISTENTA LA CITIRE

- Sistemul Oracle implementează mecanisme care îi permit fiecărui utilizator (uman sau aplicaţie care utilizează baza de date) să aibă în orice moment o viziune consistentă a datelor.
- În continuare sunt prezentate elementele de bază privind consistenţa la citire, blocările implicite efectuate de sistem şi mecanismele de implementare a tranzacţiilor.

CONSISTENTA.. (2)

- În mod curent pe aceeaşi bază de date operează simultan mai mulţi utilizatori care pot efectua două tipuri de operaţii:
  - Operaţii de citire. În această categorie intră în principal regăsirile de date prin cereri SELECT.
  - Operaţii de scriere. În această categorie sunt operaţiile de INSERT, UPDATE, DELETE şi MERGE descrise anterior.
- În lipsa unei tratări necorespunzătoare, execuţia simultană a unei operaţii de scriere cu o altă operaţie (de citire sau de scriere) poate duce la inconsistenţe în utilizarea bazei de date.

EXEMPLU

- Fie doi utilizatori care lucrează simultan pe tabela STUD, primul incrementând anul de studii (o cerere UPDATE) şi al doilea afişând conţinutul tabelei (cerere SELECT). Fără mecanisme de păstrare a unei viziuni consistente asupra datelor este posibilă următoarea situaţie:
- Unele dintre linile afişate pentru al doilea utilizator conţin anul de studii încă nemodificat (cererea UPDATE nu a ajuns încă la aceste linii),
- Celelalte linii afişate conţin anul modificat de cererea UPDATE în curs.

CUM FACE ORACLE:

- Aceleaşi date pot fi citite simultan de oricâţi utilizatori,
- Aceleaşi date nu pot fi modificate simultan de doi utilizatori. În momentul în care unul din ei modifică o valoare, sistemul efectuează o blocare implicită a liniei care o conţine blocând accesul pentru scriere al altor utilizatori.
- Aceleaşi date pot fi modificate la un moment dat de un utilizator şi citite de oricâţi alţii,
- Modificările făcute de un utilizator în conţinutul tabelelor nu sunt vizibile pentru ceilalţi utilizatori până în momentul în care se execută implicit sau explicit înscrierea lor permanentă în baza de date (operaţie numită în terminologia de specialitate ***comitere***).

CUM FACE ORACLE – cont.

- Până la comiterea modificărilor, operaţiile de citire efectuate de alţi utilizatori returnează conţinutul de date anterior modificării lor,
- Până nu au fost comise, modificările pot fi revocate, anulându-se efectul cererilor care le-au efectuat,
- În momentul comiterii sau revocării modificărilor se ridică toate blocările aferente acestora,

În acest fel operaţiile de citire şi modificare a datelor se pot executa simultan fără să interfereze unele cu celelalte.

De asemenea sunt prevenite inconsistenţele care pot apare la modificarea simultană a aceloraşi date.

EXEMPLU

- Exemplu: să considerăm trei posturi de lucru, U1, U2 şi U3 de la care se execută următoarele cereri, la momentele de timp specificate (t1 < t2 < ... < t6).
- Iniţial presupunem că tabela NRSTUD conţine informaţii corecte despre specializările şi numărul de studenţi asociat lor:

```
U1/t1: UPDATE NRSTUD SET NRSTUD = 0;
U1/t2: SELECT * FROM NRSTUD;
U2/t3: SELECT * FROM NRSTUD;
U3/t4: UPDATE NRSTUD SET CODS = CODS + 1;
U1/t5: COMMIT;
U2/t6: SELECT * FROM NRSTUD;
```

```
U1/t1: UPDATE NRSTUD SET NRSTUD = 0;
U1/t2: SELECT * FROM NRSTUD;
U2/t3: SELECT * FROM NRSTUD;
U3/t4: UPDATE NRSTUD SET CODS = CODS + 1;
U1/t5: COMMIT;
U2/t6: SELECT * FROM NRSTUD;
```

- La momentul t1, U1 execută o actualizare a datelor, aducând la 0 valorile de pe coloana NRSTUD. Noile valori nu sunt încă permanente, fiind vizibile doar pentru U1.
- În acelaşi timp sistemul blochează liniile modificate, ele fiind accesibile pentru scriere doar pentru U1.
- La momentul t2, U1 afişează datele. Rezultatul conţine valorile modificate.

```
U1/t1: UPDATE NRSTUD SET NRSTUD = 0;
U1/t2: SELECT * FROM NRSTUD;
U2/t3: SELECT * FROM NRSTUD;
U3/t4: UPDATE NRSTUD SET CODS = CODS + 1;
U1/t5: COMMIT;
U2/t6: SELECT * FROM NRSTUD;
```

- La momentul t3, U2 afişează de asemenea conţinutul tabelei NRSTUD. Datele afişate vor conţine vechile valori din coloana NRSTUD (diferite de 0), cele modificate fiind disponibile decat pentru U1.
- La momentul t4, U3 lansează o cerere de actualizare pe tabela NRSTUD. Deoarece U1 nu a comis sau revocat încă modificările, cererea lui U3 intră în aşteptare, liniile necesare fiind blocate de sistem.
- La momentul t5 U1 comite modificările. Ele devin permanente iar liniile blocate din NRSTUD sunt deblocate. În consecinţă se execută şi cererea lui U3 care fusese plasată în aşteptare.

```
U1/t1: UPDATE NRSTUD SET NRSTUD = 0;
U1/t2: SELECT * FROM NRSTUD;
U2/t3: SELECT * FROM NRSTUD;
U3/t4: UPDATE NRSTUD SET CODS = CODS + 1;
U1/t5: COMMIT;
U2/t6: SELECT * FROM NRSTUD;
```

- Sistemul blochează pentru U3 liniile modificate din NRSTUD. Ele nu mai pot fi scrise decât de acesta.
- La momentul t6 se execută din nou o cerere de regăsire a lui U1. Datele afişate vor conţine valorile modificate ale numărului de studenţi (0) şi cele încă nemodificate pentru codul specializării (U3 nu a comis încă modificările).
- Pentru testarea acestui comportament se pot folosi trei ferestre SQL*Plus deschise simultan pe acelaşi calculator, cu acelaşi nume de utilizator Oracle.

TRANZACTII

- În contextul acestui capitol, o tranzacţie poate fi definită ca o succesiune de cereri DML (printre care pot exista şi cereri de regăsire a informaţiilor) executate într-o aceeaşi sesiune de lucru.
- Modificarile facute in baza de date pe parcursul unei tranzactii pot fi la final fie toate comise fie toate revocate
- Tranzactia este din acest punct de vedere o operatie atomica
- La încheierea unei tranzacţii, toate liniile blocate de sistem pentru aceasta sunt eliberate.

TRANZACTII – cont.

- Tranzacţia începe cu prima cerere DML şi se termină în momentul în care:
  - Se execută o cerere COMMIT care face permanente modificările efectuate de tranzacţie.
  - Se execută o cerere ROLLBACK care revocă modificările efectuate de tranzacţie.
  - Se execută o cerere DDL (de exemplu CREATE) sau o cerere DCL. Tranzacţia se încheie înainte de execuţia acesteia prin comiterea modificărilor
  - Se încheie sesiunea de lucru (ieşire din SQL*Plus). În acest caz modificările efectuate de tranzacţie sunt de asemenea comise.
  - Apare o cădere hardware sau software a sistemului. Modificările efectuate de tranzacţie sunt revocate.

CERERI DDL SI DCL

- În ceea ce priveşte cererile DDL şi DCL, acestea se constituie de asemenea în tranzacţii.
- Execuţia unei astfel de cereri se face astfel:
  - Anterior execuţiei este finalizată prin comitere tranzacţia în curs.
  - Se execută cererea DDL sau DCL.
  - Efectul acesteia este de asemenea comis. Cererile de acest tip nu pot fi revocate prin ROLLBACK.

SQL*PLUS

- În SQL*Plus există de asemenea posibilitatea de a specifica faptul că fiecare cerere DML trebuie comisă automat după execuţie. Setarea poate fi efectuată astfel:
  - SET AUTOCOMMIT OFF - Comiterea şi revocarea sunt la nivel de tranzacţie şi nu la nivel de cerere DML. Este opţiunea implicită.
  - SET AUTOCOMMIT ON - Fiecare cerere DML este o tranzacţie.
- Atentie: inchiderea ferestrei SQL*Plus (din butonul x) este considerata incident software si tranzactia curenta este revocata!

COMMIT SI ROLLBACK

- Sintaxa cererilor COMMIT şi ROLLBACK este următoarea:
  - **`COMMIT;`** - Toate modificările efectuate de cererile DML ale tranzacţiei sunt comise.
  - **`ROLLBACK;`** - Toate modificările efectuate de cererile DML ale tranzacţiei sunt revocate.
- In ambele cazuri tranzactia curenta se incheie.

SAVEPOINT

- În sistemul Oracle există şi posibilitatea ca o tranzacţie să fie revocată parţial.
- Pentru asta însă trebuiesc setate puncte de revenire cu cererea SAVEPOINT. Sintaxa aferentă acestor operaţii este următoarea:

```
SAVEPOINT nume;
ROLLBACK TO [SAVEPOINT] nume;
```

  - SAVEPOINT nume: specifică definirea unui punct de revenire, având asociat un nume. Pe cuprinsul execuţiei unei tranzacţii se pot fixa mai multe astfel de puncte.
  - ROLLBACK TO nume: specifică revocarea tuturor modificărilor efectuate după fixarea punctului de revenire cu numele respectiv. Modificările efectuate înainte de acest punct rămân şi nu sunt revocate. Ele pot fi comise sau revocate ulterior, revocarea putând fi de asemenea parţială. În acelaşi timp, toate punctele de revenire fixate ulterior celui în cauză se pierd.

EXEMPLU

- Fie următoarea succesiune de cereri SQL, executate imediat după intrarea într-o sesiune de lucru SQL*Plus.

```
INSERT INTO SPEC ...
SAVEPOINT P1;
UPDATE STUD ...
SAVEPOINT P2
DELETE SPEC ...
ROLLBACK TO P2
INSERT INTO STUD ...
ROLLBACK TO P1;
CREATE TABLE ...
```

```
INSERT INTO SPEC ...
   SAVEPOINT P1;
  UPDATE STUD ...
   SAVEPOINT P2
  DELETE SPEC ...
   ROLLBACK TO P2
INSERT INTO STUD ...
  ROLLBACK TO P1;
 CREATE TABLE ...
```

- Tranzacţia începe cu o inserare în tabela SPEC. După efectuarea acesteia se fixează punctul de revenire P1.
- Se execută o actualizare a tabelei STUD şi fixarea punctului de revenire P2.
- Se execută o ştergere de linii din SPEC. Urmează însă o revenire la punctul P2 care anulează efectele acestei ştergeri.

```
INSERT INTO SPEC ...
   SAVEPOINT P1;
  UPDATE STUD ...
   SAVEPOINT P2
  DELETE SPEC ...
   ROLLBACK TO P2
INSERT INTO STUD ...
  ROLLBACK TO P1;
 CREATE TABLE ...
```

- Se execută o inserare în tabela STUD. Ulterior însă se revine în punctul P1. Sunt revocate astfel efectele inserării şi actualizării tabelei STUD.
- Ultima cerere din exemplu este un CREATE. Înainte de execuţia sa sunt comise modificările executate de tranzacţie. Singura cerere DML care încă nu a fost revocată este prima inserare în SPEC şi va fi singura comisă.
- Se execută CREATE care fiind cerere DDL se comite automat.

CONCLUZII

- Mecanismele de tranzacţie şi blocările implicite puse la dispoziţie de sistem permit asigurarea consistenţei bazei de date în contextul operării concurente cu aceleaşi date.
- Pe lângă blocările implicite, Oracle pune la dispoziţia utilizatorilor şi mecanisme de blocare explicită care nu fac obiectul lucrării de faţă.

Sfarsitul capitolului

ACTUALIZAREA DATELOR