CAPITOLUL 14

Limbajul de manipulare a datelor

Acest capitol explica cum se fac schimbari liniilor intr-o tabela, cum se adauga noi linii sau cum se sterg. Este introdus conceptul de tranzactie. Consistenta la citire este deasemenea discutata.

Inserarea de noi linii intr-o tabela

Comanda INSERT este folosita pentru a adauga linii unei tabele Sintaxa comenzii INSERT este:

```
INSERT    INTO nume tabela [ (coloana,coloana,....)]
          VALUES (valoare,valoare,....);
```

Este posibila inserarea unei noi linii cu valori in fiecare coloana, in care caz lista de coloane nu este ceruta. Este recomandat ca COLUMN LIST sa fie intotdeauna specificata. Daca lista nu este specificata, software-ul va cere modificari oriunde definitia tabelei este modificata.

Pentru a insera un nou departament, introduceti:

```
INSERT    INTO DEPT (DEPTNO,DNAME,LOC)
VALUES    (50,'MARKETING','SAN JOSE');
```

Nota aceasta comanda adauga o singura linie unei tabele.

Pentru a intra intr-un departament nou, omitand numele departamentului, lista de coloane trebuie specificata:

```
INSERT     INTO DEPT (DEPTNO,LOC)
VALUES     (50,'SAN JOSE');
```

Alternativ, daca numele departamentului nu este cunoscut, un NULL ar putea fi specificat:

```
INSERT     INTO DEPT (DEPTNO,DNAME,LOC)
VALUES     (50,NULL,'SAN JOSE');
```

Valorile CHARACTER si DATE trebuie puse in ghilimele simple.

Folosirea Variabilelor de Substitutie pentru a insera linii

Dupa cum am mentionat anterior, INSERT este o comanda pentru o singura lini- e. Folosind variabile de substitutie este posibil sa se mareasca viteza de intrare.

```
INSERT    INTO DEPT (DEPTNO,DNAME,LOC)
VALUES    (&D_NUMBER, '&D_NAME', '&LOCATION');
```

Cand comanda este rulata, valorile sunt afisate de fiecare data.

Inserarea informatiilor de data si timp

Cand se insereaza o valoare DATE, formatul DD-MON-YY este de obicei folo- sit. Cu acest format secolul implicit este secolul 20 (19nn). Data deasemenea contine informatii de timp, care daca nu sunt specificate, implicit devin miezul noptii (00:00:00).

Daca o data trebuie introdusa in alt secol si un timp specific este cerut deasemenea, folositi functia TO_DATE:

```
INSERT   INTO EMP
         (EMPNO,ENAME,JOB,MGR,HIREDATE,SAL,COMM,DEPTNO)
VALUES   (7658,
         'MASON',
         'ANALYST',
         7566,
         TO_DATE('24/06/2084 9:30','DD/MM/YYYY HH:MI'),
         3000,
         NULL,
         20);
```

Copierea de linii din alta tabela

```
INSERT   INTO  tabela [(coloana, coloana, ....)]
         SELECT lista-select
         FROM tabela(e)
```

Aceasta forma a declaratiei INSERT va permite sa inserati cateva linii intr-o tabela unde valorile sunt derivate din continutul tabelelor existente in baza de date.

Pentru a copia toate informatiile din departamentul 10 in tabela D10HISTORY, introduceti:

```
INSERT          INTO D10HISTORY
                (EMPNO,ENAME,SAL,JOB,HIREDATE
                FROM EMP
                WHERE DEPTNO=10;
```

Notati ca cuvantul cheie 'VALUES' nu este folosit aici.

Actualizarea liniilor

Declaratia UPDATE va permite sa schimbati valori in liniile unei tabele.

```
UPDATE        tabela[alias]
SET           coloana [, coloana...] = {expresie, subcerere}
[WHERE        conditie];
```

De exemplu:

Pentru a actualiza linia lui Scott, introduceti:

```
UPDATE   EMP
SET      JOB='SALESMAN',
         HIREDATE = SYSDATE,
         SAL = SAL*1.1
WHERE    ENAME  = 'SCOTT';

1 record updated.
```

Daca clauza WHERE este omisa, toate liniile din tabela vor fi actualizate. Este posibil sa folositi subcereri inlantuite si subcereri corelate in decla- ratia UPDATE.

Sa presupunem ca ati avut o cifra noua de comisioane pentru angajati siguri. De exemplu, tabela COMMISSION de mai jos este folosita pentru a actualiza li- nii sigure ale tabelei EMP:

COMMISSION

EMPNO	COMM
7499	1100
7654	500
7844	3500
7844	2000
7844	1500

EMP

EMPNO	COMM
7499	300
7654	1400
7844	0

Schimbarile listate in tabela COMMISSION pot fi aplicate tabelei EMP, folosind o subcerere corelata si o subcerere inlantuita, ca mai jos:

Exemplul 1:

```
UPDATE EMP
SET COMM = (SELECT COMM FROM COMMISSION C
              WHERE C.EMPNO = EMP.EMPNO)
WHERE EMPNO IN (SELECT EMPNO FROM COMMISSION);

3 records updated.
```

Tabela COMMISSION poate contine mai mult decat o intrare pentru fiecare angajat, ca in exemplul de mai jos :

COMISSION

EMPNO	COMM
7499	1100
7654	500
7654	100
7844	2000
7844	1500

Daca doriti sa inlocuiti (REPLACE) valorile din tabela EMP pentru comision cu comisionul TOTAL pentru fiecare angajat listat in tabela COMISSION, atunci puteti utiliza urmatorul SQL :

Exemplul 2:

```
UPDATE EMP
SET COMM = ( SELECT SUM(COMM) FROM COMISSION C
        WHERE C.EMPNO = EMP.EMPNO)
WHERE EMPNO IN (SELECT EMPNO FROM COMISSION);

3 inregistrari modificate.

Tabela EMP reflecta comisioanele modificate :

        EMP
EMPNO           COMM
-----           ----
 7499           1100
 7654            600
 7844           3500
```

O alta posibilitate este cea de a adauga (ADD) la valorile comisionului in tabela COMISSION la comisioanele existente in tabela EMP mai mult decat inlocuirea lor. Exemplul 3 realizeaza acest lucru :

Exemplul 3:

```
UPDATE EMP
SET COMM = ( SELECT SUM(COMM) + EMP.COMM
        FROM COMISSION C
        WHERE C.EMPNO = EMP.EMPNO)
WHERE EMPNO IN (SELECT EMPNO FROM COMISSION);

Tabela EMP reflecta comisioanele schimbate :

        EMP
EMPNO           COMM
-----           ----
 7844           3500
 7499           1400
 7654           2000
```

Stergerea Coloanelor dintr-o Tabela

Comanda DELETE permite stergerea unei sau mai multor linii dintr-o tabela.

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

Pentru a sterge toate informatiile despre departamentul 10 din tabela EMP, introduceti :

```
DELETE          FROM EMP
WHERE           DEPTNO = 10;
```

Daca clauza WHERE este omisa, atunci toate liniile vor fi sterse.

TRANZACTII

Procesarea unei Tranzactii

Ce este o tranzactie ?

O tranzactie este o operatie asupra unei baze de date care implica una sau mai multe modificari in una sau mai multe tabele.

Exista doua clase de tranzactii. Tranzactii DML care contin un numar oarecare de blocuri DML si pe care ORACLE le trateaza ca o singura entitate sau o singura unitate logica de lucru, si tranzactii DDL care contin un singur bloc DDL.

Nu pot exista situatii "jumatate de drum" in timpul executiei unei tranzactii, asa incat unele modificari specificate in tranzactie sa fie aplicate bazei de date si altele nu. Pentru fiecare tranzactie ori toate modificarile sunt aplicate bazei de date, ori nici una din modificari nu este indeplinita (sunt toate abandonate - discarded).

O tranzactie incepe cand prima comanda executabila DML sau DDL este intalnita si se termina in una din urmatoarele situatii :

- Intalneste COMMIT/ROLLBACK
- Comanda DDL se termina
- Anumite erori (DEADLOCK)
- EXIT - iesire din SQL*Plus
- Eroare sistem

Un bloc DDL este executat automat si de aceea implicit incheie o tranzactie.

Dupa incheierea unei tranzactii, urmatorul bloc executabil SQL va lansa automat urmatoarea tranzactie.

Permanentizarea Modificarilor

Pentru ca modificarile sa ramana permanente, ele trebiue executate asupra bazei de date. comanda COMMIT realizeaza permanentizarea modificari- lor; ROLLBACK permite sa abandonam sau sa anulam modificarile. Modifiacrea, sau modificarile, executate asupra bazei de date intre 2 comenzi COMMIT reprezinta o tranzactie. Pana cand tranzactia nu este executata, nici una din modificari nu este vizibila utilizatorilor.

Inlaturarea Modificarilor Nedorite

Modificarile neexecutate pot fi abandonate prin comanda ROLLBACK. ROLLBACK va atribui datelor valorile care acestea le aveau imediat dupa executarea ultimului COMMIT prin anularea tuturor modificarilor facute dupa ultimul COMMIT.

Erorile de Sistem

Cand o tranzactie este intrerupta de o eroare serioasa, de exemplu o eroare de sistem, intreaga tranzactie este anulata. Aceasta previne erorile datorate modificarilor nedorite asupra datelor, si realizeaza intoarcerea tabelelor la starile de dupa ultimul COMMIT. In acest fel SQL*Plus protejea- za integritatea tabelelor.

Anularea automata este cauzata cel mai des de catre o eroare de sistem, ca de exemplu o resetare a sistemului sau o cadere de tensiune. Erorile de tastare a comenzilor, ca de exemplu tastarea gresita a unor nume de coloane sau incercarile de a realiza operatii neautorizate asupra tabelelor altor utilizatori, nu intrerup tranzactia si nu realizeaza anu- larea automata. Aceasta se datoreaza faptului ca aceste erori sunt detectate in cursul compilarii (de catre PARSER) (cand un bloc SQL este scanat si verificat), si nu in timpul executiei.

O tranzactie noua este lansata urmand unui COMMIT sau ROLLBACK - adica cand primul bloc executabil DML sau DDL este intalnit.

Semnificatia Tranzactiilor

ORACLE asigura consistenta datelor bazata pe tranzactii. Tranzacti ile dau utilizatorului mai multa flexibilitate si control la lucrul asupra datelor, si asigura consistenta datelor in cazul unei erori a procesului utilizator sau a unei erori de sistem.

Tranzactiile ar trebui sa contina doar acele comenzi DML care realizeaza o singura modificare asupra datelor. De exemplu un transfer de fonduri (sa spunem 1000$) intre 2 conturi ar trebui sa implice un debit al unui cont de 1000$ si un credit al altui cont de 1000$. Ambele actiuni ar trebui sa se incheie cu succes sau sa dea eroare impreuna. Creditul nu ar trebui executat fara debit.

Controlul Tranzactiilor cu Instructiuni SQL

Urmatoarele instructiuni SQL sunt utilizate cand apar executii (commit) sau refaceri (rollback) :

- COMMIT[WORK]
- SAVEPOINT nume_savepoint
- ROLLBACK[WORK] to [SAVEPOINT] nume_savepoint

De notat ca COMMIT si ROLLBACK sunt instructiuni (blocuri) SQL.

Cele 3 blocuri SQL utilizate pentru controlul tranzactiilor sunt explicate mai jos:

COMMIT[WORK]

Sintaxa : COMMIT[WORK];

- Permanentizeaza schimbarile in tranzactia curenta
- Sterge toate punctele de salvare (savepoint) din tranzactie
- Termina tranzactia
- Elibereaza toate blocarile (Lock) tranzactiei
- Cuvantul cheie WORK este optional
- Utilizatorul trebuie sa expliciteze sfarsitul tranzactiei in programul aplicatie utilizand COMMIT (sau ROLLBACK). Daca nu se executa explicit tranzactia si programul se termina anormal, ultima tranzactie executata va fi anulata.
- Executii implicite (commit) apar in urmatoarele situatii :
    - inainte de o comanda DDL
    - dupa o comanda DDL
    - la inchiderea mormala a unei baze de date

Blocurile DDL cauzeaza mereu executii (commit) in timpul executiei lor. Daca introduceti un bloc DDL dupa cateva blocuri DML, blocul DDL cauzeaza aparitia unui commit inaintea propriei executii, incheind tranzactia curenta. Astfel daca blocul DDL este executat pana la capat, este si inregistrat.

SAVEPOINT

Sintaxa : SAVEPOINT nume_savepoint

Exemplu :

```
SAVEPOINT terminare_actualizari
```

- Poate fi utilizat pentru a imparti o tranzactie in bucati mai mici
- Punctele de salvare (savepoints) permit utilizatorului sa retina toata munca sa la orice moment din timp, cu optiunea de a inregistra mai tarziu totul sau a anula totul sau o parte din ea. Astfel, pentru o tran- zactie lunga, se pot salva parti din ea, pe masura executiei, la sfarsit inregistrandu-se sau refacandu-se continutul initial. La aparitia unei erori nu trebuie executat din nou fiecare bloc.
- La crearea unui nou punct de salvare cu acelasi nume ca al unuia dinainte, primul punct este sters.
- Numarul maxim de puncte de salvare pentru un proces utilizator este implicit 5. Aceasta limita poate fi schimbata.
-
- `ROLLBACK[WORK] to [SAVEPOINT] nume_punct_salvare`
-
- Instructiunea ROLLBACK este utilizata pentru a reface un lucru.
- Cuvantul cheie "work" este optional. Intoarcerea la un punct de salvare este de asemenea optionala.
- Daca se utilizeaza ROLLBACK fara clauza TO SAVEPOINT, atunci :
    - se termina tranzactia
    - se anuleaza modificarile din tranzactia curenta
    - sterge toate punctele de salvare din tranzactie
    - elibereaza blocarile tranzactiei

Intoarcerea la Nivel de Bloc

O parte a unei tranzactii poate fi anulata. Daca un singur bloc DML da eroare, ORACLE V6 va intoarce inapoi doar acel bloc. Aceasta facilitate este cunoscuta ca STATEMENT LEVEL ROLLBACK. Intoarcerea la nivel de bloc inseamna daca un singur segment DML da eroare la executia unei tranzactii, efectul lui este anulat, dar schimbarile realizate de precedentul bloc DML in tranzactie nu vor fi anulate si pot fi inscrise (COMMIT) sau intoarse (ROLLBACK) explicit de catre utilizator.

Daca blocul este unul de tip DDL, inscrierea (commit) care precede imediat acest bloc nu este anulata (schimbarile au fost facute deja permanen- te). ORACLE realizeaza intoarcerea la nivel de bloc prin crearea unui punct de salvare implicit inainte de executarea fiecarei comenzi DML. Utilizatorul nu poate referi caest punct de salvare in mod direct.

Astfel, daca va intoarceti la un punct de salvare, atunci:

- anulati o parte din tranzactie
- se retine punctul de salvare pentru intoarcere, dar se pierd toate celelalte punct create dupa punctul de salvare numit.
- se elibereaza toate blocarile de tabele si linii.

Intoarceri Implicite

Intoarcerile implicite apar cand se intalnesc terminari anormale ale executiei (de exemplu cand se intrerupe un proces utilizator). Intoarcerile implicite la nivel de bloc apar la eroarea de executie a unui bloc.

Este recomandat ca tranzactiile sa se termine explicit utilizand COMMIT[WORK] ori ROLLBACK[WORK].

Urmatorul exemplu demonstreaza utilizarea unui punct de salvare, si a instructiunilor ROLLBACK si COMMIT.

```
INSERT INTO DEPT
VALUES
( 50, 'TESTING', 'LAS VEGAS' );

SAVEPOINT insert_done;

UPDATE DEPT
SET DNAME = 'MARKETING';

ROLLBACK TO insert_done ( modificarile sunt abandonate );

UPDATE DEPT
SET DNAME = 'MARKETING' ( revizuim comanda UPDATE )
WHERE DNAME = 'SALES';

COMMIT;

AUTOCOMMIT
```

COMMIT sau ROLLBACK pot fi date manual sau automat prin utilizarea optiunii AUTOCOMMIT a comenzii SET. Optiunea AUTOCOMMIT controleaza cand schimbarile intr-o baza de date sunt facute permanente.

Exista doua setari :

COMANDA + DESCRIEREA

SET AUTO[COMMIT] ON
COMMIT este utilizat automat la fiecare INSERT, UPDATE sau DELETE
SET AUTO[COMMIT] OFF
COMMIT poate fi utilizata de utilizator explicit. De asemenea, COMMIT se executa cand apare comanda Z (specifica VAX), cand se executa comenzile DROP, ALTER sau CREATE, sau la iesirea din SQL*Plus. ROLLBACK poate fi executat explicit de catre utilizator pentru refacerea bazei de date.

De retinut ca SET este o comanda SQL*Plus.

Consistenta la Citire

Utilizatorii bazelor de date fac doua tipuri de accesuri asupra bazelor de date:

- Operatii de citire (SELECT)
- Operatii de scriere (INSERT, UPDATE, DELETE)

Cititorului si scriitorului unei baze de date trebuie sa i se garanteze o vedere consistenta asupra bazei de date. Cititorii nu trebuie sa vizulaizeze o data care este in curs de modificare. Si scriitorii trebuie sa fie siguri ca schimbarile intr-o baza de date sunt facute intr-un mod consistent : schimbarile facute de un scriitor sa nu distruga sau sa intre in confilct cu schimbarile pe care le face un alt scriitor.

Scopul consistentei la citire este acela de a asigura faptul ca fiecare utilizator vede data ca fiind cea de la ultimul COMMIT, inainte ca o operatie DML sa inceapa.

Consistenta la citire este implementata prin tinerea unor copii partiale ale bazei de date in segmente de intoarcere (ROLLBACK).

Cand de executa operatii de scriere intr-o baza de date, ORACLE va face o copie a datelor onainte de schimbare si o va scrie intr-un segment de intoarcere.

Toti cititorii, exceptandu-i pe cei care au facut schimbarile, inca mai vad baza de date care exista inainte ca schimbarile sa fie facute - ei vad segmentul de intoarcere de fapt.

Oricum, inainte ca schimbarile sa fie facute permanente in baza de date, doar utilizatorul care modifica datele poate sa vada baza de date cu alteratiile incorporate. Toti ceilalti vad baza nemodificata (fereastra din segmentul de intoarcere). Aceasta garanteaza citirea unor date consistente care nu fac subiectul unor modificari in curs.

Cand un bloc DML se executa, schimbarile facute in baza de date devin vizibile oricarui utilizator care executa SELECT. Modificarile sunt facute 'universale' si acum toti utilizatorii vad baza de date cu modificarile incorporate.

Spatiul ocupat de catre 'vechile' date in segmentul de intoarcere este eliberate pentru a fi reutilizat.

Daca tranzactia este anulata (ROLLBACK), atunci toate schimbarile sunt 'anulate' :

- Versiunea veche ('originala') a bazei de date aflata in segmentul de intoarcere este scrisa inapoi ('recuperata') in baza de date.
- Toti utilizatorii vad baza de date existenta inainte de inceperea tranzactiei.

Tranzactii de Citire

Implicit, modelul consistent al ORACLE RDBMS garanteaza ca rezultatul executiei unui bloc este consistenet. Oricum, in anumite situatii se poate dori chestionarea unor blocuri multiple asupra datelor din mai multe tabele si se doreste asigurarea ca datele sunt consistente. Adica, re- zultatele in orice tabela sunt consistente in timp in raport cu rezultatele din orice alta tabela.

Linia SQL : SET TRANSACTION READ ONLY este utilizata pentru a incepe o tranzactie de citire exclusiv.

Consistenta la citire pe care READ ONLY o furnizeaza este implementa- ta in acelasi fel cu consistenta la nivel de bloc - utilizand segmente de intoatcere. Fiecare bloc vede implicit o fereastra consistenta a datelor la momentul inceperii blocului. Aceasta facilitate este foarte folositoare pentru rapoarte care ruleaza mai multe chestionari asupra mai multor tabele in timp ce alti utilizatori actualizeaza aceleasi tabele.

Note :

- SET TRANSACTION READ ONLY este utilizata pentru a incepe o tran- zactie doar de citire.
- Doar cereri (blocuri SELECT) sunt permise. Comenzile DML nu sunt permise. SELECT FOR UPDATE va genera o eroare.
- O instructiune COMMIT, ROLLBACK, sau un bloc DDL va termina tran- zactia de citire (de retinut ca blocurile DDL genereaza implicit suprascrieri - COMMIT). Cu blocuri DDL, nu este data nici o indi- catie referitoare la faptul ca tranzactia se termina implicit.
- In timpul tranzactiei de citire, toate cererile se refera la aceeasi copie a bazei de date (schimbarile sunt efectuate inainte ca tranzactia de citire sa inceapa).
- Alti utilizatori pot continua sa citeasca sau sa modifice datele.

Urmatoarele seturi de instructiuni pot fi rulate pentru a extrage datele din tabelele EMP si DEPT.

```
COMMIT;
SET TRANSACTION READ ONLY;

SELECT DEPTNO, ENAME, JOB
FROM EMP;

SELECT DEPTNO, DNAME
FROM DEPT;

COMMIT;
```

Ultimul COMMIT este necesar pentru a termina explicit tranzactia de citire.

EXERCITII - limbajul de manipulare a datelor

Exercitiile acopera inserarea si modificarea de coloane in tabelele create anterior.

1. Inserati in tabela PROJECTS urmatoarele informatii :

PROJID	1	2
P_DESC	WRITE C030 COURSE	PROOF READ NOTES
P_START_DATE	02-JAN-88	01-JAN-89
P_END_DATE	07-JAN-88	10-JAN-89
BUDGET_AMOUNT	500	600
MAX_NO_STAFF	1	1
COMMENTS	BE CREATIVE	YOUR CHOICE

2. Inserati in tabela ASSIGNMENTS urmatoarele coloane:

PROJID	1	1	2
EMPNO	7369	7902	7844
A_START_DATE	01-JAN-88	04-JAN-88	01-JAN-89
A_END_DATE	03-JAN-88	07-JAN-88	10-JAN-89
BILL_RATE	50.00	50.00	45.50
ASSIGN_TYPE	WR	WR	PF
HOURS	15	20	30

3. Modificati campul ASSIGNMENT TYPE astfel incat sa se citeasca 'WT' in loc de 'WR'. Un tip ca 'PF' trebuie sa ramana nemodificat.

4. Inserati inca doua proiecte cu atribuiri la alagere.

REZOLVARI:

```
1.      INSERT INTO PROJECTS
        (PROJID, P_DESC, P_START_DATE, P_END_DATE,
        BUDGET_AMOUNT, MAX_NO_STAFF)
        VALUES
        (1,'WRITE C030 COURSE','02-JAN-88','07-JAN-
88',500,2);

        INSERT INTO PROJECTS
        (PROJID, P_DESC, P_START_DATE, P_END_DATE,
        BUDGET_AMOUNT, MAX_NO_STAFF)
        VALUES
        (2,'PROOF READ NOTES','01-JAN-89','10-JAN-
89',600,1);

2.      INSERT INTO ASSIGNMENTS
        (PROJID, EMPNO, A_START_DATE,
A_END_DATE,BILL_RATE,
        ASSIGN_TYPE, HOURS)
        VALUES
        (1,7369,'01-JAN-88','03-JAN-88',50.00,'WR',15);

        INSERT INTO ASSIGNMENTS
        (PROJID, EMPNO, A_START_DATE,
A_END_DATE,BILL_RATE,
        ASSIGN_TYPE, HOURS)
        VALUES
        (1,7092,'04-JAN-88','07-JAN-88',50.00,'WR',20);

        INSERT INTO ASSIGNMENTS
        (PROJID, EMPNO, A_START_DATE,
A_END_DATE,BILL_RATE,
        ASSIGN_TYPE, HOURS)
        VALUES
        (2,7844,'01-JAN-89','10-JAN-89',45.50,'PF',30);

3.      UPDATE ASSIGNMENTS
        SET ASSIGN_TYPE='WT'
        WHERE ASSIGN_TYPE='WR';
```