

Proiectarea Bazelor de Date

Cornelia TUDORIE



Proiectarea Bazelor de Date - Cuprins

- A. Modelul Relațional.
- B. Limbajul SQL. Lucrul cu tabele.
- C. Bazele teoretice ale limbajelor relaționale
- D. Sisteme de baze de date. Performanța.
- E. Obiecte ale sistemului de baze de date. Performanța în utilizare
- F. **Programe pentru baze de date. Performanța în programare.**
- G. Protecția datelor.
- H. Sisteme Informatică. Proiectarea sistemelor de baze de date.

F. Programe pentru baze de date. Performanța în programare.

- 1. Fișiere de comenzi**
- 2. PL/SQL. Tipuri de programe**
- 3. Blocuri anonime**
- 4. Subprograme stocate. Pachete**
- 5. Declanșatori**
- 6. Tranzacții**

F. 6. Tranzacții.

Tranzacție

= o prelucrare coerentă a datelor care constă într-o succesiune de operații de actualizare a bazei de date

Începe la prima operație de actualizare

Durează până la următoarea comandă COMMIT, sau comandă LDD, sau la închiderea sesiunii

Se anulează la comanda ROLLBACK

F. 6. Tranzacții.

Tranzacție

= o prelucrare coerentă a datelor care constă într-o succesiune de operații de actualizare a bazei de date

INSERT

UPDATE

INSERT

DELETE

.

COMMIT

F. 6. Tranzacții.

Tranzacție



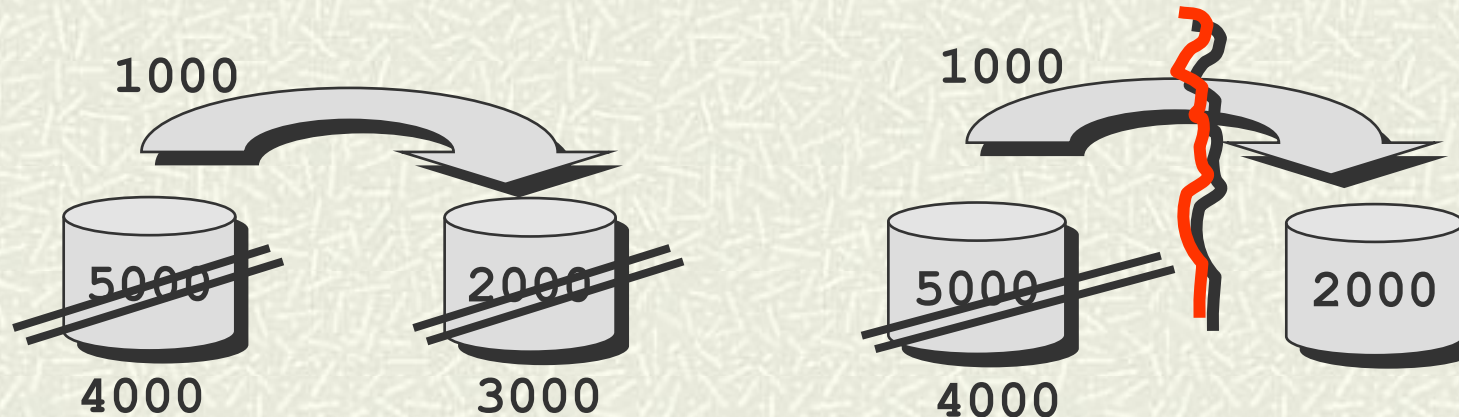
**Pentru asigurarea coerenței bazei de date,
toate tranzacțiile trebuie să se termine corect;
altfel, anulate!**



**Pentru asigurarea coerenței bazei de date,
tranzacțiile concurente nu trebuie să se suprapună!**

F. 6. Tranzacții.

Pentru asigurarea coerenței bazei de date, toate tranzacțiile trebuie să se termine corect; altfel, anulate!



F. 6. Tranzacții.

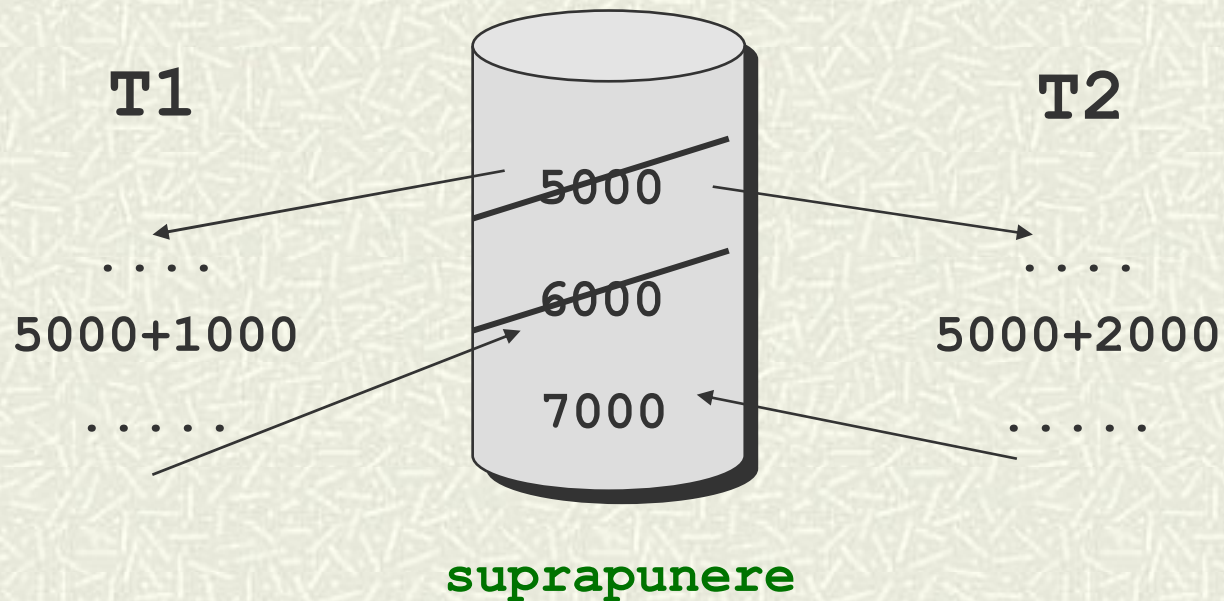
COMMIT;

ROLLBACK [*punct_de_întoarcere*] ;

SAVEPOINT *punct_de_întoarcere* ;

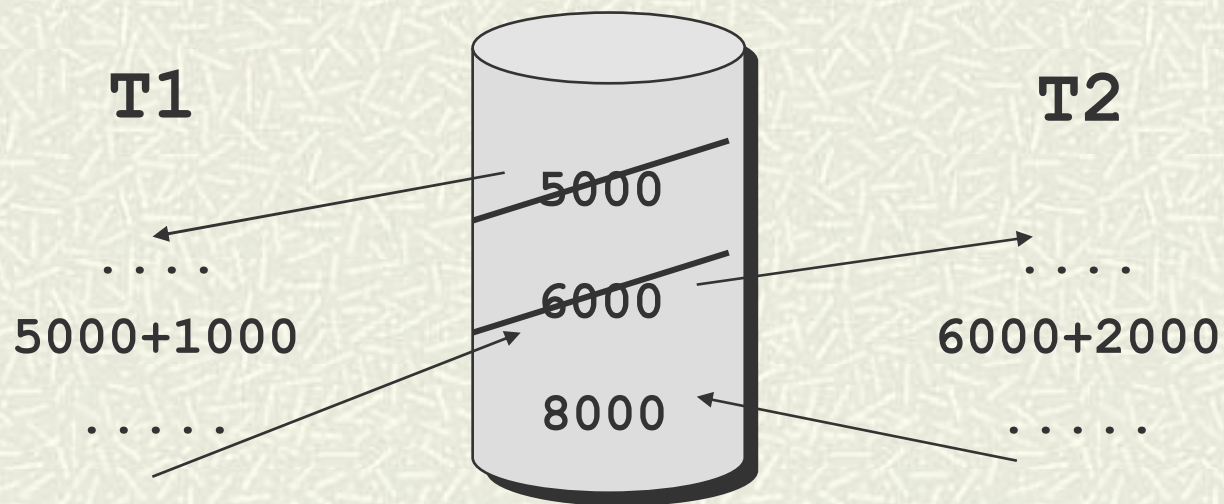
F. 6. Tranzacții.

**Pentru asigurarea coerenței bazei de date,
tranzacțiile concurente nu trebuie să se suprapună!**



F. 6. Tranzacții.

**Pentru asigurarea coerenței bazei de date,
tranzacțiile concurente nu trebuie să se suprapună!**



secvență de tranzacții

F. 6. Tranzacții.

Blocare

- # automată**
- # explicită**

F. 6. Tranzacții.

**Blocare
automată**

**Consistența
la Citire**

empno	sal	deptno
1	8000	1
2	10000	1

T1

```
update emp  
set sal=sal*1.1;  
  
insert into emp  
values (3, 12000, 2);  
  
select * from emp;
```

1	8800	1
2	11000	1
3	12000	2

T2

```
select * from emp;
```

1	8000	1
2	10000	1

Proiectarea Bazelor de Date -
Cornelia TUDORIE

F. 6. Tranzacții.

**Blocare
automată**

**Consistența
la Citire**

T1

```
commit;  
select * from emp;
```

1	8800	1
2	11000	1
3	12000	2

```
delete from emp  
where deptno=1;
```

```
select * from emp;
```

3	12000	2
---	-------	---

T2

```
select * from emp;
```

1	8800	1
2	11000	1
3	12000	2

```
select * from emp;
```

1	8800	1
2	11000	1
3	12000	2

F. 6. Tranzacții.

**Blocare
automată**

**Consistența
la Citire**

T1

`rollback;`

`select * from emp;`

1	8800	1
2	11000	1
3	12000	2

T2

`select * from emp;`

1	8800	1
2	11000	1
3	12000	2

F. 6. Tranzacții.

**Blocare
explicită**

SET TRANSACTION ...

sau

LOCK TABLE ...



Control complet

F. 6. Tranzacții.

Tranzacții și PL/SQL

Tranzacțiile sunt independente de blocurile PL/SQL

F. 6. Tranzacții.

Tranzacții și PL/SQL

```
INSERT INTO emp (empno,ename)
      VALUES (222, 'Popescu' );
UPDATE emp SET ename='Georgescu'
      WHERE empno=7902;
begin
      INSERT INTO emp (empno,ename)
            VALUES (333, 'Vasilescu' );
end;
DELETE FROM emp WHERE empno=7902;

COMMIT;
```

222 Popescu
333 Vasilescu

F. 6. Tranzacții.

Tranzacții și PL/SQL

```
INSERT INTO emp (empno,ename)
      VALUES (222, 'Popescu') ;
UPDATE emp SET ename='Georgescu'
      WHERE empno=7902;
begin
      INSERT INTO emp (empno,ename)
      VALUES (333, 'Vasilescu') ;
end;
DELETE FROM emp WHERE empno=7902;
```

7902 FORD

ROLLBACK;

F. 6. Tranzacții.

Tranzacțiile sunt independente de blocurile PL/SQL

Excepție: Tranzacțiile autonome

- # se execută în interiorul unui bloc, ca tranzacții separate de tranzacția principală
- # prin directiva
PRAGMA AUTONOMOUS_TRANSACTION
în secțiunea declarativă

F. 6. Tranzacții.

Fără tranzacții autonome. Exemple

```
CREATE OR REPLACE PROCEDURE p1 IS
BEGIN
    INSERT INTO emp (empno, ename)
        VALUES (100, 'ION');
END;
```

```
CREATE OR REPLACE PROCEDURE p2 IS
BEGIN
    INSERT INTO emp (empno, ename)
        VALUES (200, 'VASILE');
END;
```

```
BEGIN
p1;
p2;
COMMIT;
END;
```

100	ION
200	VASILE

F. 6. Tranzacții.

Fără tranzacții autonome. Exemple

```
CREATE OR REPLACE PROCEDURE p1 IS
BEGIN
    INSERT INTO emp (empno, ename)
        VALUES (100, 'ION');
END;
```

```
CREATE OR REPLACE PROCEDURE p2 IS
BEGIN
    INSERT INTO emp (empno, ename)
        VALUES (200, 'VASILE');
END;
```

```
BEGIN
p1;
p2;
ROLLBACK;
END;
... nimic...
```

F. 6. Tranzacții.

Tranzacții autonome. Exemple

```
CREATE OR REPLACE PROCEDURE p1 IS
```

```
BEGIN
```

```
    INSERT INTO emp (empno, ename)
```

```
        VALUES (100, 'ION');
```

```
END;
```

```
CREATE OR REPLACE PROCEDURE p2 IS
```

```
    PRAGMA AUTONOMOUS_TRANSACTION;
```

```
BEGIN
```

```
    INSERT INTO emp (empno, ename)
```

```
        VALUES (200, 'VASILE');
```

```
    COMMIT;
```

```
END;
```

```
BEGIN
```

```
    p1;
```

```
    p2;
```

```
    COMMIT;
```

```
END;
```

100	ION
200	VASILE

F. 6. Tranzacții.

Tranzacții autonome. Exemple

```
CREATE OR REPLACE PROCEDURE p1 IS
```

```
BEGIN
```

```
    INSERT INTO emp (empno, ename)
```

```
        VALUES (100, 'ION');
```

```
END;
```

```
CREATE OR REPLACE PROCEDURE p2 IS
```

```
    PRAGMA AUTONOMOUS_TRANSACTION;
```

```
BEGIN
```

```
    INSERT INTO emp (empno, ename)
```

```
        VALUES (200, 'VASILE');
```

```
    COMMIT;
```

```
END;
```

```
BEGIN
```

```
    p1;
```

```
    p2;
```

```
    ROLLBACK;
```

```
END;
```

200 VASILE

F. 6. Tranzacții.

Tranzacții autonome. Avantaje.

- # nu partajează resursele cu tranzacția principală**
- # nu depind de tranzacția principală**
- # actualizările sunt imediat vizibile celorlalți**

F. 6. Tranzacții.

Blocare prin cursor

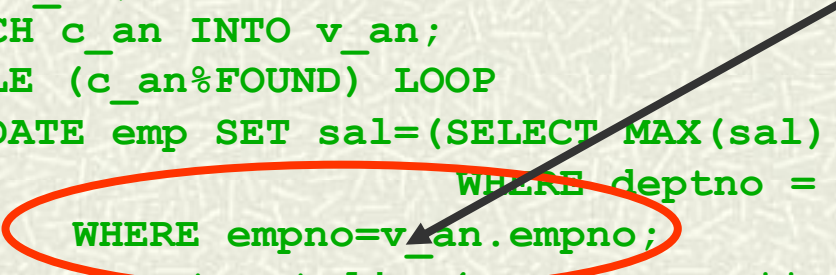
CURSOR *nume_cursor* **IS** *cerere*
FOR UPDATE [**OF** *coloană* [, *coloană*, ...]];

**tot ansamblul de linii returnate de cerere este blocat
până la sfârșitul tranzacției (COMMIT sau
ROLLBACK)**

F. 6. Tranzacții.

Blocare prin cursor. Exemplu

```
DECLARE
    CURSOR c_an IS
        SELECT * FROM emp where job='ANALYST' FOR UPDATE;
    v_an emp%ROWTYPE;
BEGIN
    OPEN c_an;
    FETCH c_an INTO v_an;
    WHILE (c_an%FOUND) LOOP
        UPDATE emp SET sal=(SELECT MAX(sal) FROM emp
                               WHERE deptno = v_an.deptno )
        WHERE empno=v_an.empno;
        Dbms_output.put_line(v_an.ename || ' castiga ' || v_an.sal );
        FETCH c_an INTO v_an;
    END LOOP;
    COMMIT;
    CLOSE c_an;
END ;
```



F. 6. Tranzacții.

Acces concurent și blocare. Exemplu

```
UPDATE dept set dname = 'JURIDIC' WHERE deptno=10; T1
```

```
-- blocare pe dept
```

```
DECLARE
```

```
    CURSOR c_an IS
```

```
        SELECT * FROM emp where job='ANALYST' FOR UPDATE;
```

```
BEGIN
```

```
    OPEN c_an; -- blocare pe emp T2
```

```
        UPDATE dept SET loc 'PARIS' WHERE deptno=10;
```

```
-- așteptare pe dept
```

```
        ROLLBACK;
```

```
        CLOSE c_an;
```

```
END ;
```

```
UPDATE emp set ename = 'MITICA' WHERE empno=7902;
```

```
-- așteptare pe emp (7902, este analist) T3
```