Tema: Modele de tranzacții. Tranzacții imbricate. Puncte de reluare

Tranzacții în SQL

In limbajul SQL, o tranzacție începe cu debutul unei sesiuni de lucru sau imediat după încheierea tranzacției precedente. Ea se termină cu o instrucțiune de validare explicită (**COMMIT**) sau de anulare (**ROLLBACK**)

Utilizatorul poate, în orice moment, să valideze (să finalizeze) tranzacția, apelînd la comanda **COMMIT.** Modificările devin permanente şi vizibile pentru toate celelalte tranzacții.

Unele comenzi SQL, inclusive definițiile de date (CREATE TABLE…), provoacă o validare automata a tranzacției.

Structura tranzacțiilor în standardul SQL este plată şi înlănțuită:

- Două Tranzacții nu se pot suprapune
- Tranzacția începe atunci cînd precedent se termină

Acest model nu este totdeauna corespunzător unei situații concrete, în special, pentru Tranzacțiile de lungă durată si Tranzacțiile în bazele de date stabilite pe mai multe stații:

- Tranzacțiile pot fi o sursă de frustrare pentru utilizator, dacă toate lucrările realixate de la începutul acestuia sunt anulate
- Tranzacțiile păstrează pînă la sfîrşitul lor lacătele, fapt ce afecteaza accesul concurential.

Tranzacțiile sunt utilizate pentru a controla în ce condiții se desfăşoară o succesiune a instrucțiunilor de manipulare a datelor.

Tranzacții imbricate

Modelul de tranzacții imbricate permite unei tranzacții să posede subtranzacții fiice, care, la rîndul lor, pot deasemenena, avea subtranzactii.

Anulare unei tranzacții-părinte derulează înapoi toate subtranzacțiile – fiice, însă anularea unei tranzacții-fiică nu anulează neapărat tranzacția – părinte. Dacă tranzacâia – părinte nu este anulată, restul tranzacâiilor – fiice nu sunt anulate.

La anularea unei tranzacții-fiică, tranzacția-părinte poate:

- Stabili un tratament de substituție
- Relua tranzacția anulată
- Să se anuleze(în cazul în care nu poate trece tranzacția anulată)
- Ignora anularea (în cazul în care tratamentul ce trebuia)

Astfel, un tratament efectuat într-o tranzacție-fiică poate fi reluat sau poate fi înlocuit cu un tratament alternative pentru a ajunge la sfîrşitul tratamentului complet. Acest model este foarte potrivit pentru tranzacțiile cu termen lung şi care activează pe multă staţii, care trebuie să facă față problemelor în ce caz de eşec de rețea

Limbajul SQL nu suportă acest model de tranzacții.

Puncte de reluare

Fără a trece la modelul de tranzacții imbricate, cele mai recente versiuni ale principlalelor SGBD-uri

au îmbunătățit modelul tranzacțiilor plate, cu puncta de control(numite, deasemenea, puncte de salvare; ***savepoint*** în limbă engleză).

Punctele de control, într-o tranzacție, pot fi desemnate cu instrucțiunea:

SAVEPOINT <nume punct>

Exisă posibilitatea de anularea a tuturor modificărilor effectuate, de la un punct de control, în cazul în care au existat probleme:

ROLLBACK <nume punct>

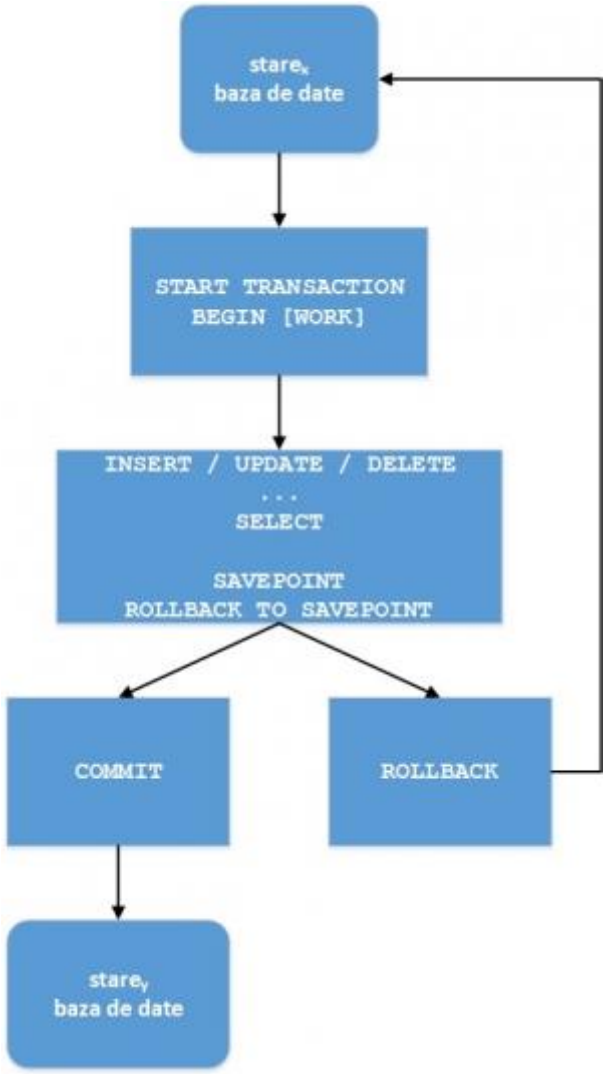
Astfe, se evită anularea întregii tranzacții și se poate încerca remedierea problemei în locul anulării tranzacției globale.

Instrucțiuni

- **START TRANSACTION** (sau **BEGIN WORK**, **BEGIN TRANSACTION**, în funcție de dialectul SQL) Început de tranzacție.
- **SAVE TRANSACTION** (sau **SAVEPOINT**) salvează starea bazei într-un punct al transacției
- **COMMIT** Operează toate operațiile tranzacției ca fiind permanente.
- **ROLLBACK** Anulează toate operațiile tranzacției începând cu ultimul COMMIT.

Instrucțiunile **COMMIT** și **ROLLBACK** termină tranzacția curentă și deblochează datele.

Schema:



OPERAȚIILE EFECTUATE DE O TRANZACȚIE

1. **Begin:** această operație marchează începutul execuției unei tranzacții.
2. **Read sau write**: sunt operații de citire sau scriere a articolelor în baza de date, executate în cadrul unei tranzacții.

3. **End:** această operație marchează terminarea operațiilor de scriere sau citire din baza de date, ceea ce înseamnă că tranzacția se poate termina; totuşi, este posibil să fie necesare unele operații de verificare înainte de validarea (commit) tranzacției.

SINTAXA

BEGIN TRAN[SACTION] [*nume_tranzacție*] **COMMIT** [TRAN[SACTION] [*nume_tranzacție*]

ROLLBACK [TRAN[SACTION] [*nume_tranzacție*]

EXEMPLU

BEGIN TRAN update_royalty(acordati dreptul de autor)

IF EXISTS (SELECT nume, prenume, nota

FROM student a, note_exam b

WHERE a.st_id = b.st_id

AND curs_id=10) UPDATE note_exam SET nota= 8 **WHERE** nota>5

ELSE ROLLBACK TRAN update_royalty

Tranzacții exemple:

1.Condiție: tranzacția adaugă în tabela **Elevi** un nou atribut **Procente** care constituie **2%** din salariu.

Tranzacția:

BEGIN TRAN Update_Tabela
ALTER TABLE Salarii add Procente float *//se adugă coloana **Procente** de tip **int***
BEGIN TRAN Update_Tabela
UPDATE Salarii set Procente=(Salariu*3)/100 *//se calculează **2%** din **Salariu***

Răspunsul obținut:

	IDAntrenor	Salariu	Procente
1	11	5010	100,2
2	22	7600	152
3	33	6790	135,8
4	44	4566	91,32
5	55	8900	178
6	66	7820	156,4
7	77	9900	198

2. Condiția: tranzacția inserează un nou tabel **Infermiere** cu atributele **Idinf** și

NumeInf Tranzacția:

```
BEGIN TRANSACTION;
CREATE TABLE Infermiere (IDInf int primary key, NumeInf text);
INSERT INTO Infermiere VALUES(234, 'Busuioc Ana');
INSERT INTO Infermiere VALUES(654, 'Panus Diana');
INSERT INTO Infermiere VALUES(294, 'Blindu Diana');
INSERT INTO Infermiere VALUES(980, 'Calancea Elena');
INSERT INTO Infermiere VALUES(267, 'Climciuc Aurica');
COMMIT;
SELECT * FROM Infermiere;
```

Răspunsul obținut:

	IDInf	NumeInf
1	234	Busuioc Ana
2	267	Climciuc Aurica
3	294	Blindu Diana
4	654	Panus Diana
5	980	Calancea Elena

3.Condiție: tranzacția șterge din tabela **Antrenori** antrenorii a căror nume începe cu litera **G** și au salariu **mai mare** de **6000**

Tranzacția:

```
BEGIN TRANSACTION;
DELETE FROM Antrenori
 WHERE NumeAntrenor Like'G%' and Salariu > 6000;
COMMIT;
```

Răspunsul obținut:

	IDAntrenor	NumeAntrenor	PrenumeAntrenor
1	11	Gladun	Artiom
2	33	Chiriac	Olga
3	44	Livadari	Olga
4	55	Cornogolub	Sergiu
5	66	Istrati	Pavel
6	77	Cecoi	Nicolai

http://www.cs.ubbcluj.ro/~horea/SD/fisiere/plsqlt.htm