TRANZACȚIILE ÎN SQL

Mariana TROFIN

Departamentul Ingineria Software și Automatică, grupa TI-191 F/R, Facultatea Calculatoare, Informatică și Microelectronică, Universitatea Tehnică a Moldovei, Chișinău, Republica Moldova

Autorul corespondent: Mariana TROFIN, e-mail: frunze.mariana@isa.utm.md

Coordonator științific: Dorian SARANCIUC, DISA, FCIM, UTM

Rezumat. *În articolul științific este descrisă utilizarea tranzacțiilor în SQL ca un instrument esențial pentru asigurarea integrității și fiabilității bazei de date. Tranzacțiile permit utilizatorilor să efectueze mai multe operații asupra bazei de date într-o singură unitate logică, cum ar fi adăugarea, modificarea sau ștergerea datelor. Cele trei comenzi de bază în tranzacții sunt COMMIT, ROLLBACK și SAVEPOINT, fiecare având rolul lor în gestionarea tranzacțiilor. Nivelele de tranzacții sunt: READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ și SERIALIZABLE, fiecare oferind un anumit nivel de protecție împotriva modificărilor concurente și a pierderii datelor.*

Cuvinte cheie*: tranzacții, commit, rollback, savepoint, nivelurile de izolare*

Tranzacțiile în SQL

Tranzacțiile în SQL sunt un mecanism esențial care permite efectuarea mai multor operații asupra bazei de date într-o singură unitate logică. Acestea sunt utilizate pentru a gestiona și a controla modificările efectuate asupra bazei de date, astfel încât să poată fi anulate orice modificări în cazul unei erori sau a unei probleme.

Tranzacțiile sunt gestionate prin intermediul unor comenzi specifice, cum ar fi COMMIT, care confirmă toate modificările efectuate în cadrul tranzacției, ROLLBACK, care anulează toate modificările efectuate în cadrul tranzacției, și SAVEPOINT, care permite crearea unui punct de salvare în cadrul tranzacției pentru a putea anula modificări specifice [1].

Utilizarea tranzacțiilor în SQL este esențială pentru orice aplicație care se bazează pe o bază de date, deoarece acestea asigură că orice modificare efectuată este validată și că baza de date rămâne întotdeauna în starea sa corectă.

Cum ajută la gestionarea și controlul modificărilor bazei de date

Tranzacțiile sunt un mecanism important pentru a gestiona și controla modificările bazei de date. Ele oferă următoarele beneficii: **atomicitate**, **consistență, izolare**, **durabilitate** [2].

Atomicitate: O tranzacție este o unitate atomică de lucru, care este executată în întregime sau nu este executată deloc. Aceasta înseamnă că, dacă o tranzacție conține mai multe modificări, toate acestea vor fi aplicate împreună sau niciuna dintre ele nu va fi aplicată.

Consistență: O tranzacție va menține baza de date într-o stare consistentă, indiferent dacă a fost executată sau nu.

Izolare: O tranzacție este izolată de alte tranzacții care sunt în curs de executare. Aceasta înseamnă că o tranzacție nu va afecta alte tranzacții sau va fi afectată de alte tranzacții.

Durabilitate: O tranzacție care a fost executată cu succes va fi permanent aplicată bazei de date.

Aceste caracteristici ale tranzacțiilor ajută la gestionarea și controlul modificărilor bazei de date, asigurând că baza de date este întotdeauna într-o stare consistentă, izolată și durabilă.

Exemple reale: COMMIT, ROLLBACK și SAVEPOINT

COMMIT, ROLLBACK și SAVEPOINT sunt operațiuni utilizate pentru a gestiona tranzacțiile în bazele de date [3].

COMMIT

COMMIT este utilizat pentru a confirma modificările efectuate într-o tranzacție și pentru a le face permanente în baza de date. Acest lucru înseamnă că modificările devin vizibile pentru toate tranzacțiile care vin după aceasta și nu mai pot fi revocate.

Iată un exemplu de instrucțiune COMMIT în SQL:

```
BEGIN TRANSACTION;
UPDATE employees SET salary = salary + 1000 WHERE department = 'sales';
COMMIT;
```

În acest exemplu, o tranzacție este inițiată cu instrucțiunea "BEGIN TRANSACTION". Apoi, salariul tuturor angajaților din departamentul "vânzări" este majorat cu 1000. Tranzacția este apoi confirmată cu instrucțiunea "COMMIT", ceea ce face ca actualizarea să fie permanentă în baza de date.

ROLLBACK

ROLLBACK este utilizat pentru a anula modificările efectuate într-o tranzacție. Acest lucru înseamnă că toate modificările efectuate în tranzacție sunt șterse, iar baza de date este readusă la starea sa anterioară.

Iată un exemplu de instrucțiune ROLLBACK în SQL:

BEGIN TRANSACTION;

UPDATE employees SET salary = salary + 1000 WHERE department = 'sales';

ROLLBACK;

În acest exemplu, o tranzacție este inițiată cu instrucțiunea "BEGIN TRANSACTION". Apoi, salariul tuturor angajaților din departamentul "vânzări" este majorat cu 1000. Cu toate acestea, dacă dintr-un motiv oarecare decidem că această actualizare nu ar trebui să fie confirmată, putem utiliza instrucțiunea "ROLLBACK" pentru a anula modificările efectuate în cadrul tranzacției [4].

Trebuie remarcat faptul că, instrucțiunea "rollback" va anula toate modificările efectuate de la ultima instrucțiune de confirmare, prin urmare, dacă aveți mai multe instrucțiuni în cadrul unei tranzacții și doriți să anulați doar una dintre ele, trebuie să utilizați puncte de salvare.

SAVEPOINT

SAVEPOINT este utilizat pentru a crea un punct de salvare în cadrul unei tranzacții. Acest lucru permite anularea doar a unui subset de modificări efectuate în tranzacție, în loc să anulați toată tranzacția.

Iată un exemplu de utilizare a unei instrucțiuni SAVEPOINT în SQL:

```
BEGIN TRANSACTION;
UPDATE employees SET salary = salary + 1000 WHERE department = 'sales';
SAVEPOINT sp1;
UPDATE employees SET department = 'marketing' WHERE department = 'sales';
SAVEPOINT sp2;
UPDATE employees SET address = 'New York' WHERE department = 'marketing';
ROLLBACK TO sp1;
```

În acest exemplu, o tranzacție este inițiată cu instrucțiunea "BEGIN TRANSACTION". Apoi, salariul tuturor angajaților din departamentul "vânzări" este majorat cu 1000. După aceea, se creează un punct de salvare "sp1". Apoi, departamentul tuturor angajaților din departamentul "vânzări" este schimbat în "marketing" și se creează un alt punct de salvare "sp2". În cele din urmă, adresa tuturor angajaților din departamentul "marketing" este schimbată în "New York". Cu toate acestea, decidem că nu dorim ultima actualizare, așa că folosim instrucțiunea "ROLLBACK TO" pentru a anula modificările efectuate după ce a fost creat punctul de salvare "sp1".

Prin urmare, efectul final va fi următorul:

Salariul tuturor angajaților din departamentul "vânzări" este mărit cu 1000.

Departamentul tuturor angajaților din departamentul "vânzări" se modifică în "marketing.

Trebuie remarcat faptul că, dacă declarația de revenire la anulare va anula toate modificările efectuate după punctul de salvare, nu este posibil să se revină la o singură declarație.

Nivelurile de izolare

În general, există patru niveluri de izolare, fiecare oferind un anumit nivel de protecție împotriva modificărilor concurente și a pierderii datelor.

Nivelul Read Uncommitted (CITIRE neconfirmată)

Acest nivel permite ca o tranzacție să citească date care sunt în curs de modificare de către o altă tranzacție, ceea ce poate duce la date inconsistente și poate afecta integritatea datelor. Acest nivel nu asigură nicio protecție împotriva pierderii datelor și poate duce la o serie de probleme precum anomalii de citire murdară și citiri fantomă. Acest nivel de izolare nu este recomandat pentru tranzacții importante.

De exemplu, în aplicații de monitorizare a performanței unde datele pot fi pierdute fără a afecta rezultatele finale.

Nivelul Read Committed (CITIRE confirmată)

Acest nivel permite ca o tranzacție să citească datele modificate de alte tranzacții numai după ce acestea au fost confirmate. Acest nivel asigură o protecție mai bună decât nivelul Read Uncommitted împotriva pierderii datelor, însă nu oferă nicio protecție împotriva citirilor fantomă. Acest nivel este adecvat pentru tranzacții care nu necesită o protecție strică împotriva modificărilor concurente și pierderii datelor.

Un exemplu ar fi o aplicație de gestiune a inventarului, unde este important să se asigure că stocurile sunt actualizate corespunzător, dar nu este necesar să se evite întotdeauna modificările concurente.

Nivelul Repeatable Read (CITIRE repetabilă)

Acest nivel asigură ca o tranzacție să poată citi aceleași date de mai multe ori, chiar și în cazul în care acestea sunt modificate de alte tranzacții în timpul execuției tranzacției actuale. Această izolare previne anomalii de citire murdară și citiri fantomă. Cu toate acestea, acest nivel nu oferă o protecție completă împotriva pierderii datelor și poate afecta performanța în cazul în care sunt implicate multe tranzacții concurente.

Un exemplu ar fi un sistem de plăți online, unde se dorește să se asigure că tranzacțiile sunt efectuate într-un mod sigur și nu pot fi modificate de alte tranzacții concurente.

Nivelul Serializable (SERIALIZABIL)

Acest nivel oferă cel mai înalt nivel de izolare și asigură că tranzacțiile concurente nu pot modifica sau citi aceleași date simultan. Acest nivel previne anomalii de citire murdară, citiri fantomă și pierderi de date. Acest nivel poate afecta performanța în cazul în care sunt implicate multe tranzacții concurente, dar este esențial pentru tranzacțiile care necesită o protecție strictă împotriva modificărilor concurente și pierderii datelor.

Un exemplu ar fi un sistem bancar, unde este crucial ca toate tranzacțiile să fie efectuate corect și să nu existe erori de sincronizare a datelor.

Concluzii

În concluzie, importanța tranzacțiilor în SQL și beneficiile lor în gestionarea și controlul modificărilor bazei de date sunt explicite în acest articol. Prin intermediul comenzilor specifice, cum ar fi COMMIT, ROLLBACK și SAVEPOINT, tranzacțiile permit efectuarea mai multor operații asupra bazei de date într-o singură unitate logică, ceea ce le face esențiale pentru orice aplicație care se bazează pe o bază de date. Utilizarea lor asigură că orice modificare efectuată este validată și că baza de date rămâne întotdeauna în starea sa corectă. Beneficiile precum atomicitate, consistență, izolare și durabilitate oferite de tranzacții asigură că baza de date este întotdeauna într-o stare consistentă, izolată și durabilă.

Referințe

1. V.Cotelea. M.Cotelea, Baze de date. Chișinău, 2016
2. James R. Groff. Paul N. Weinberg, "SQL: The Complete Reference"
3. C.J. Date, "SQL and Relational Theory: How to Write Accurate SQL Code"
4. John L. Viescas și Michael J. Hernandez, "SQL Queries for Mere Mortals: A Hands-On Guide to Data Manipulation in SQL"