OBLIKOVANJE BAZA PODATAKA

Predavanje 10

Blic

- https://forms.gle/c6jByKu5co3r9Sm56

Transakcije

Uvod

- **Transakcija** (engl. *transaction*) **u relacijskim bazama** predstavlja količinu posla koju baza treba odraditi kao cjelinu, na dobro definiran način

- Transakcije možemo podijeliti na dva tipa:
  - Transakcije na nivou jedne naredbe
    - Svaka SQL naredba je transakcija sama po sebi
      - Ako zadamo DELETE za 5 redaka, ne može se desiti da se 2 retka obrišu, a 3 ne
  - Eksplicitne transakcije koje obuhvaćaju proizvoljan broj naredbi
    - Korisnik zadaje početak i kraj transakcije

Korisnost transakcija

- Transakcije rješavaju dva problema:

1. Baza uvijek mora biti u **ispravnom (konzistentnom)** stanju
   - Svaka naredba se mora izvršiti do kraja ili uopće ne
     - Radimo UPDATE na 20 redaka
   - Ponekad je potrebno osigurati da se **niz SQL naredbi** odradi kompletan ili da se ne odradi niti jedna naredba iz niza
     - Želimo nekome prebaciti 1.000 kuna na račun:
       - UPDATE koji skida 1.000 kuna s našeg računa
       - UPDATE koji stavlja 1.000 kuna na odredišni račun
   - Što ako se u oba slučaja usred posla desi nestanak struje?
2. Pri svemu tome **jedan korisnik baze ne smije ometati druge**, tj. mora biti **izoliran** od drugih korisnika
   - Što ako jedan korisnik mijenja redak kojeg drugi briše?

Svojstva transakcija

- Transakcije u svim RDBMS-ovima posjeduju **četiri svojstva** koja na engleskom jeziku tvore kraticu **ACID**:
  - **A**tomicity (hrv. *nedjeljivost*)
    - Transakcija se mora obaviti u cijelosti ili uopće ne
  - **C**onsistency (hrv. *ispravnost*)
    - Baza prije i nakon transakcije mora biti u konzistentnom stanju (sva ograničenja moraju biti ispoštovana)
  - **I**solation (hrv. *odvojenost*)
    - Jedna transakcija ne smije utjecati na drugu
  - **D**urability (hrv. *izdržljivost*)
    - Stanje baze dobiveno završetkom transakcije mora biti trajno sačuvano (niti pad sustava ne smije rezultirati gubitkom stanja)

Transakcijska datoteka

- Transakcijska datoteka omogućava postojanje transakcija
- Svaka relacijska baza se obavezno sastoji od barem jedne **transakcijske datoteke**
  - Sadržava povijest akcija rađenih nad bazom
    - Svaka akcija nad bazom se zapisuje o transakcijsku datoteku zajedno s podacima na koje je utjecala
  - Kod SQL Servera se ta datoteka naziva i **transakcijski zapisnik** (engl. *transactional log*) i ima ekstenziju **.ldf**
- Zbog performansi je dobro transakcijsku datoteku držati na drugom fizičkom disku
  - Na taj način se mogu paralelno zapisivati podaci i transakcijske aktivnosti

Korištenje transakcijske datoteke (1/2)

- Izvršavanje SQL naredbe se izvodi na sljedeći način:

1. Korisnik zadaje naredbu, primjerice:

```
UPDATE Zaposlenik
SET Placa *= 1.1
WHERE ID Zaposlenik IN (12, 2342, 22133)
```

2. Pronalaze se stranice koje sadržavaju retke na koje se odnosi naredba (u memoriji ili s diska) i mijenjaju se
3. U transakcijsku datoteku u memoriji se zapisuje naredba zajedno sa stanjem redaka prije i poslije njenog izvršavanja
4. Izmjene transakcijske datoteke se odmah spremaju (tzv. ***write-ahead* način rada**) i već tada RDBMS kaže da je naredba uspješno obavljena

Korištenje transakcijske datoteke (2/2)

- U ovom trenutku:
  - Podaci su promijenjeni na stranicama u memoriji
  - Transakcijski log je zapisan na disk
  - Korisnik je dobio potvrdu da je naredba uspješno izvršena
  - **U .mdf datoteci na disku još nema izmijenjenih podataka**
- Podaci će zbog performansi biti grupno zapisani na disk u nekom kasnijem trenutku:
  - Procesom *lazywritera*
  - Zadavanjem naredbe CHECKPOINT
  - Izradom sigurnosne kopije, ...
- Ako se u međuvremenu desi pad sustava, nakon pokretanja će podaci biti rekonstruirani iz transakcijske datoteke

Važni trenuci u životu jedne transakcije

- Transakcija se sastoji od sljedećih važnih trenutaka:
  - **Početak** transakcije
  - **Kraj** transakcije
    - Potvrda transakcije:
      - Sve operacije koje čine transakciju su **uspješno** obavljene i time je i transakcija označena uspješnom
    - Odustajanje od transakcije
      - Desila se jedna ili više grešaka i transakcija je **neuspješna**
      - Vraćam bazu na stanje kakvo je bilo točno prije početka transakcije
  - Opcionalno postavljanje **kontrolne točke** (engl. ***savepoint***)
- Svaki trenutak u životu transakcije je predstavljen odgovarajućom SQL naredbom

T-SQL sintaksa za upravljanje transakcijama

- Osnovne T-SQL naredbe su sljedeće:
  - Početak transakcije: **BEGIN TRAN**[SACTION]
  - Potvrda transakcije: **COMMIT TRAN**[SACTION] (ili samo **COMMIT**)
  - Odustajanje od transakcije: **ROLLBACK TRAN**[SACTION] (ili samo **ROLLBACK**)

- Dodatno, za rad s kontrolnim točkama koristimo:
  - Postavljanje kontrolne točke: **SAVE TRAN**[SACTION] **naziv_kontr_točke**
  - Vraćanje na kontrolnu točku: **ROLLBACK TRAN**[SACTION] **naziv_kontr_točke**

Primjer transakcije

- Svaka transakcija se odnosi na konekciju na kojoj je zadana
  - Otvorimo SSMS, napravimo novu konekciju na neku bazu i zadamo naredbe:

- Nakon kraja transakcije u bazi će ostati promjene napravljene naredbama 1, 2 i 4
- Učinak naredbe 3 je poništen vraćanjem na kontrolnu točku naziva spa

Primjeri

1. Napravite tablicu Osoba. Pokrenite transakciju i umetnite 3 zapisa u Osoba. Probajte dohvatiti podatke iz tablice. Odustanite od transakcije. Probajte dohvatiti podatke iz tablice.

2. Riješite zadatak 1, ali umjesto odustajanja potvrdite transakciju.

3. U transakciji umetnite 1 zapis u Osoba i postavite kontrolnu točku. Umetnite još 1 zapis. Na kraju odustanite od transakcije. Što je u tablici?

4. U transakciji umetnite 1 zapis u Osoba i postavite kontrolnu točku. Umetnite još 1 zapis. Na kraju potvrdite transakciju. Što je u tablici?

Primjeri

5. U transakciji umetnite 1 zapis u Osoba i postavite kontrolnu točku. Umetnite još 1 zapis i postavite kontrolnu točku. Na kraju odustanite od transakcije.

6. U transakciji umetnite 1 zapis u Osoba i postavite kontrolnu točku. Umetnite još 1 zapis i postavite kontrolnu točku. Na kraju potvrdite transakciju.

7. U transakciji umetnite 1 zapis u Osoba i postavite kontrolnu točku. Umetnite još 1 zapis i vratite se na kontrolnu točku. Na kraju odustanite od transakcije.

8. U transakciji umetnite 1 zapis u Osoba i postavite kontrolnu točku. Umetnite još 1 zapis i vratite se na kontrolnu točku. Na kraju potvrdite transakciju.

Korištenje transakcija

Sistemska funkcija @@TRANCOUNT

- Sistemsku funkciju @@TRANCOUNT koristimo za provjeru jesmo li unutar transakcije
  - Funkcija vraća cijeli broj:
    - Ako je veći od 0, unutar transakcije smo
    - Ako je jednak 0, nismo unutar transakcije

- Način rada funkcije @@TRANCOUNT

  - BEGIN TRAN povećava @@TRANCOUNT za 1

  - COMMIT TRAN smanjuje @@TRANCOUNT za 1

  - ROLLBACK TRAN smanjuje @@TRANCOUNT na 0
    - Povratak na kontrolnu točku ne mijenja @@TRANCOUNT

Tipično korištenje transakcija

- Čest je način korištenja transakcija uz `TRY/CATCH`
  - Ako su sve naredbe u transakciji uspješno izvršene, na kraju `TRY` bloka se radi `COMMIT TRAN`
  - Ako neka od naredbi uzrokuje grešku, u `CATCH` bloku se radi `ROLLBACK TRAN` (ako smo unutar transakcije)

```
BEGIN TRY
    BEGIN TRAN
        naredbe_u_transakciji
    COMMIT TRAN
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0 --Samo ako transakcija postoji
        ROLLBACK TRAN
END CATCH
```

Upravljanje transakcijama

- Transakcijama možemo upravljati:
  - Iz aplikacije (kolegij PPPK)
  - Iz SQL kôda
    - Upravljanjem **unutar** procedure osiguravamo da se cijela procedura ponaša kao transakcija

```
CREATE PROC UmetniCijeliRacun … AS
BEGIN TRY … BEGIN TRAN …
```

    - Upravljanjem **izvan** procedure osiguravamo da se pozivanje više procedura ponaša kao transakcija

```
BEGIN TRY … BEGIN TRAN …
   EXEC UmetniRacun …
   EXEC UmetniStavku …
   EXEC UmetniStavku …
    …
```

Ugniježđene transakcije

- Ako i procedura i njen pozivatelj žele upravljati transakcijom, imamo **ugniježđenu transakciju** (engl. *nested transaction*)
  - **Vanjska** transakcija je ona pozivateljeva
  - **Unutarnja** transakcija je ona u proceduri

- Moguće su sljedeće situacije:
  - Unutarnja kaže `COMMIT`
    - Ako i vanjska kaže `COMMIT`, sve izmjene se potvrđuju
    - Ako vanjska kaže `ROLLBACK`, odustaje se od svih izmjena (i od onih unutarnje transakcije, bez obzira što je rekla `COMMIT`)
  - Unutarnja kaže `ROLLBACK`
    - Odustaje se od svih izmjena i završava i unutarnja i vanjska transakcija

Primjeri

9. Napišite proceduru za brisanje proizvoda. Neka procedura prima 1 parametar, IDProizvod. Transakciju vodite izvan procedure. Ispišite uspjeh ili neuspjeh.
   - Pozovite 3 puta proceduru s vrijednostima parametara jednakim 1001, 1002 i 1003.
   - Pozovite 3 puta proceduru s vrijednostima parametara jednakim 1001, 1002 i 777.

10. Osigurajte da u tablici Drzava ne mogu postojati dvije države s istim nazivom. Napišite proceduru koja prima 1 XML parametar koji sadržava podatke potrebne za umetnuti novu državu i tri njena grada. Transakciju vodite unutar procedure. Ispišite uspjeh ili neuspjeh. Pozovite proceduru 2 puta s istim XML dokumentom kao parametrom.

11. Unutar vanjske transakcije pozovite prethodnu proceduru s nekim drugim parametrom. Nakon toga odustanite od vanjske transakcije. Ispišite uspjeh ili neuspjeh. Je li umetanje napravljeno?

Implicitne transakcije

- **Implicitna transakcija** je ona koja automatski započinje zadavanjem neke naredbe i traje do potvrde ili odustajanja
- Ne miješati s AUTOCOMMIT načinom rada
- Svaki RDBMS je specifičan u upravljanju implicitnim transakcijama, ali se uglavnom mogu podijeliti u tri skupine:
  - Oni koji koriste implicitne transakcije
  - Oni koji ne koriste implicitne transakcije
  - Oni koji ih ne koriste, ali ih mogu koristiti
- **Oracle** i **DB2** spadaju u prvu skupinu, **SQL Server** u treću

- Uključivanje/isključivanje implicitnih transakcija u SQL Serveru (za svaku konekciju posebno):

```
SET IMPLICIT_TRANSACTIONS ON | OFF
```

Primjeri

12. Uključite implicitne transakcije i zadajte naredbu za umetanje nove osobe. Zatvorite konekciju.