OBLIKOVANJE BAZA PODATAKA

Predavanje 11

Blic

- https://bit.ly/3uFHdPj

Konkurentnost, lokoti i transakcije

Konkurentnost i izolacija

- Relacijska baza podataka je **višekorisničko okruženje**
- U isto vrijeme mnogi korisnici mogu koristiti bazu i njezine objekte, što se naziva **konkurentnost** (engl. *concurrency*)
  - Što je **veća konkurentnost**, to **više korisnika** može paralelno nesmetano koristiti bazu
  - Što je **veća izolacija**, to su korisnici više zaštićeni jedan od drugoga, ali **manje korisnika** može raditi paralelno
  - Porastom izolacije pada konkurentnost i obrnuto
  - Međusobno suprotni zahtjevi: želimo što veću konkurentnost i što veću izolaciju
    - Kako to postići? Kompromisom.

Dijeljeni resursi i konflikti

- **Dijeljeni resurs** predstavlja element baze podataka kojemu korisnici mogu pristupati

- Najvažnije vrste dijeljenih resursa su:
  - Redak u tablici
  - Tablica
  - Stranica
  - Ključ indeksa

- **Konflikt** - kad različite transakcije pristupaju u isto vrijeme istom resursu na nekompatibilne načine
  - Dvije transakcije žele pročitati isti redak – nema konflikta
  - Jedna želi pročitati redak, a druga promijeniti – **konflikt**

Rješavanje konflikata

- RDBMS-ovi konflikte rješavaju na dva načina:
  - **Zaključavanjem** resursa (engl. *locking*)
    - Koriste se **lokoti** (engl. *lock*)
    - Zajednički mehanizam za sve RDBMS-ove
  - **Verzioniranjem** (engl. *versioning*)
    - Koristi se više verzija istog retka
    - Podržavaju samo poneki RDBMS-ovi (SQL Server od verzije 2005)

- U ovom kolegiju ćemo proučavati mehanizme zaključavanja

- Zanimaju nas dvije vrste lokota:
  - **X lokot** ili **ekskluzivni** lokot (engl. *exclusive lock*)
  - **S lokot** ili **dijeljeni** lokot (engl. *shared lock*)

Zaključavanje resursa

- Princip korištenja resursa zaključavanjem:
  1. Transakcija na resurs postavlja svoj lokot
  2. Transakcija koristi resurs (transakciji ne smetaju vlastiti lokoti)
  3. Transakcija uklanja lokot
- Kako se time rješavaju konflikti:
  - Između koraka 1 i 3 **druga transakcija** dolazi do **istog resursa**
    a. Ako je način korištenja kompatibilan, postavlja i svoj lokot na resurs i koristi resurs na isti način kao i prva transakcija
    b. Ako nije kompatibilan, druga transakcija je **blokirana** (engl. *blocked*) i čeka dok prva transakcija ukloni lokot
- Podešavanje koliko je transakcija spremna biti blokirana:

```
SET LOCK_TIMEOUT broj_milisekundi
```

Primjeri

- Kod rješavanja zadataka s transakcijama potrebne su dvije SQL skripte, svaka u svojoj konekciji
  - Svaka SQL naredba treba sadržavati **redni broj izvršavanja** jedinstven u obje skripte

1. Napravite dvije transakcije. Neka prva transakcija pristupi jednom retku i promijeni neku vrijednost. Neka druga transakcija dohvati sve retke iz tablice. Je li se dogodio konflikt? Odustanite od obje transakcije.
2. Napravite dvije transakcije. Neka prva transakcija pristupi jednom retku i promijeni neku vrijednost. Neka druga transakcija dohvati sve retke iz tablice osim onog kojeg je prva transakcija promijenila. Je li se dogodio konflikt? Odustanite od obje transakcije.

Izolacijski nivoi

Izolacijski nivoi

- **Izolacijski nivo** (engl. *isolation level*) definira ponašanje transakcije u konkurentnom okruženju

- Standard definira četiri izolacijska nivoa:
  - READ UNCOMMITTED
  - READ COMMITTED (predefinirano)
  - REPEATABLE READ
  - SERIALIZABLE

- Izolacijski nivo se može postaviti na sljedeće načine:
  - Prilikom pristupa pojedinoj tablici
  - Za cijelu konekciju
    - Mi ćemo koristiti ovaj pristup

SQL sintaksa

- Model zaključavanja korišten na ovom kolegiju će biti znatno pojednostavljen:
  - Ne zanimaju nas eskalacije zaključavanja, zaključavanje roditeljskih resursa ni indeksa niti drugi tipovi lokota osim S i X

- Podrazumijevano je aktivan izolacijski nivo READ COMMITTED

- Bilo koji nivo za cijelu konekcije postavljamo na sljedeći način:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

READ COMMITTED

- Transakcija pokrenuta pod ovim izolacijskim nivoom se ponaša na sljedeći način:

1. Ako transakcija želi **čitati** redak:
   a. Ako na njemu nema lokota ili postoji jedan ili više S lokota, postavit će na njega svoj S lokot i otpustiti ga **odmah nakon čitanja**
   b. Ako na njemu postoji X lokot, transakcija će biti blokirana

2. Ako transakcija želi **mijenjati** (I/U/D) redak:
   a. Ako na njemu nema nikakvih lokota, postavit će na njega svoj X lokot i držati ga do **kraja transakcije**
   b. Ako na njemu postoji bilo kakav tuđi lokot, transakcija će biti blokirana

READ UNCOMMITTED

- Podcrtano su označene razlike u odnosu na READ COMMITTED

- Transakcija pokrenuta pod ovim izolacijskim nivoom se ponaša na sljedeći način:

1. Ako transakcija želi **čitati** redak:
   a. Pročitat će ga bez postavljanja svojih i poštivanja tuđih lokota

2. Ako transakcija želi **mijenjati** redak:
   a. Ako na njemu nema nikakvih lokota, postavit će na njega svoj X lokot i držati ga do kraja transakcije
   b. Ako na njemu postoji bilo kakav tuđi lokot, transakcija će biti blokirana

REPEATABLE READ

- Transakcija pokrenuta pod ovim izolacijskim nivoom se ponaša na sljedeći način:

1. Ako transakcija želi **čitati** redak:
   a. Ako na njemu nema lokota ili postoji jedan ili više S lokota, postavit će na njega svoj S lokot i držati ga do kraja transakcije
   b. Ako na njemu postoji X lokot, transakcija će biti blokirana

2. Ako transakcija želi **mijenjati** redak:
   a. Ako na njemu nema nikakvih lokota, postavit će na njega svoj X lokot i držati ga do kraja transakcije
   b. Ako na njemu postoji bilo kakav tuđi lokot, transakcija će biti blokirana

SERIALIZABLE

- Transakcija pokrenuta pod ovim izolacijskim nivoom se ponaša na sljedeći način:

1. Ako transakcija želi **čitati** redak:
   a. Ako na njemu nema lokota ili postoji jedan ili više S lokota, postavit će na njega svoj S lokot i držati ga do kraja transakcije
      i. Dodatno definira **raspon redaka** prema WHERE uvjetu i na njega stavlja S lokote – spriječeno umetanje novih redaka
   b. Ako na njemu postoji X lokot, transakcija će biti blokirana

2. Ako transakcija želi **mijenjati** redak:
   a. Ako na njemu nema nikakvih lokota, postavit će na njega svoj X lokot i držati ga do kraja transakcije
   b. Ako na njemu postoji bilo kakav tuđi lokot, transakcija će biti blokirana

Primjeri

3. Napravite dvije transakcije, obje na READ COMMITTED. Neka prva transakcija pristupi jednom retku i promijeni mu vrijednost. Neka druga transakcija dohvati baš taj redak. Je li se dogodio konflikt? Odustanite od obje transakcije.

4. Napravite dvije transakcije, prvu na READ COMMITTED, drugu na READ UNCOMMITTED. Neka prva transakcija pristupi jednom retku i promijeni mu vrijednost. Neka druga transakcija dohvati baš taj redak. Je li se dogodio konflikt? Odustanite od obje transakcije.

5. Napravite dvije transakcije, prvu na REAPEATABLE READ, drugu na READ COMMITTED. Neka prva transakcija dohvati neke retke. Neka druga transakcija pokuša promijeniti jedan od tih redaka. Je li se dogodio konflikt? Odustanite od obje transakcije.

6. Napravite dvije transakcije, prvu na SERIALIZABLE, drugu na READ COMMITTED. Neka prva transakcija dohvati nazive svih proizvoda iz tablice. Neka druga transakcija pokuša pročitati neki redak i umetnuti novi. Je li se dogodio konflikt? Odustanite od obje transakcije.

Problemi kod konkurentnosti (1/2)

- Postoje tri osnovna **problema** koji mogu nastati kad više korisnika pristupa istom podatku:
  - Prljavo čitanje (engl. *dirty read*)
    1. Prva transakcija promijeni vrijednost retka
    2. Druga transakcija pročita tu vrijednost – **prljavo čitanje**
    3. Prva transakcija odustane
  - Neponovljivo čitanje (engl. *non-repeatable read*)
    1. Prva transakcija pročita vrijednost stupca u nekom retku – **neponovljivo čitanje**
    2. Druga transakcija promijeni upravo vrijednost i potvrdi se
    3. Prva transakcija istim SELECT-om sad čita drukčiju vrijednost

Problemi kod konkurentnosti (2/2)

- Fantom (engl. *phantom*)
  1. Prva transakcija zada SELECT koji vrati *n* redaka
  2. Druga transakcija umetne redak u raspon
  3. Prva transakcija zada isti SELECT koji sad vrati *n* + 1 redaka – **fantom**

Kako izolacijski nivoi rješavaju probleme

- Svaki izolacijski nivo rješava jedan ili više problema
- Tablica prikazuje koji problem se može pojaviti u kojem izolacijskom nivou
  - označava da smo sigurni od problema ✔
  - označava da se problem može dogoditi ⚠

Izolacijski nivo	Prljavo čitanje	Neponovljivo čitanje	Fantom
SERIALIZABLE	✔	✔	✔
REPEATABLE READ	✔	✔	⚠
READ COMMITTED	✔	⚠	⚠
READ UNCOMMITTED	⚠	⚠	⚠

Primjeri

7. Demonstrirajte problem prljavog čitanja na tablici Drzava. Kako ga rješavamo?

8. Demonstrirajte problem neponovljivog čitanja na tablici Drzava. Kako ga rješavamo?

9. Demonstrirajte problem fantoma na tablici Drzava. Kako ga rješavamo?