ZAKLJUČAVANJE PODATAKA

1. Istovremeni pristup podacima

Rad korisnika s bazom podataka najčešće se svodi na izvršavanje unaprijed definiranih transakcija. Pojam transakcije odnosi se na skup od jedne ili više radnji koje čuvaju konzistenciju i korektnost podataka, a moraju se sve izvršiti u potpunosti ili se ne smije izvršiti niti jedna. Iako transakcija sa korisničkog stanovišta predstavlja jednu nedjeljivu cjelinu, ona se obično realizira kao niz od nekoliko elementarnih zahvata u bazi.

Velike baze podataka moraju riješiti pitanje konkurentnosti baze i omogućiti istovremeni pristup podacima većem broju korisnika. U višekorisničkim bazama nekoliko transakcija izvodit će se paralelno, pri čemu će se osnovne operacije u transakciji vremenski ispreplitati. Da bi integritet baze bio očuvan, DBMS mora osigurati da učinak transakcija koje se izvršavaju paralelno mora biti kao da su se izvršavale slijedno, odnosno mora biti zadovoljeno svojstvo serijabilnosti (eng. *serializability*). Navedeno svojstvo ne može se a priori osigurati, a nekontrolirano izvršavanje transakcija može lako dovesti do neželjenih efekata, čiji su opisi dani u nastavku.

Potencijalni problemi kod paralelnog pristupanja podacima

- **Izgubljene izmjene** – dvije transakcije paralelno pišu u bazu, a sačuvan je učinak samo one koja je zadnja završila
- **Prljavo čitanje** – prva transakcija piše, dok druga čita. Ovisno u kojem trenutku druga transakcija čita podatak, može vidjeti n-torke koje nikad nisu postojale u bazi jer je transakcija završila s opozivom (ROLLBACK)
- **Neponovljivo čitanje** – prva transakcija čita određeni podatak, dok u međuvremenu druga napravi promjenu u bazi, nakon čega će prva transakcija za isti onaj upit čitanja podataka dobiti različiti rezultat
- **Sablasne n-toke** (eng. phantom rows) – prva transakcija dobiva različit rezultat prilikom primjerice prebrojavanja n-torki jer je druga transakciju u međuvremenu unijela promjenu u tablicu

Navedena problematika može se riješiti zaključavanjem odgovarajućih podataka. Transakcija koja želi pristupiti nekom podatku najprije mora „uzeti“ odgovarajući ključ i njime zaključati dotični dio baze. Čim je obavila svoju operaciju, transakcija treba „vratiti“ lokot i time otključati podatke. Kad transakcija naiđe na podatke koji su već zaključani, ona mora čekati dok ih prethodna transakcija ne otključa. Time se izbjegava (sasvim) istovremeni pristup istom podatku.

2. Protokol dvofaznog zaključavanja (2PL)

Svojstvo serijabilnosti bit će zadovoljeno ako sve transakcije prilikom upotrebe ključeva poštuju protokol dvofaznog zaključavanja (eng. *Two-phase locking protocol*). 2PL protokol sastoji se od 2 faze:

1. Faza pribavljanja ključeva (faza rasta - growing phase)
2. Faza otpuštanja ključeva (faza sužavanja - shrinking phase)

Prema protokolu transakcija mora zatražiti ključ prije obavljanja operacija nad objektom (tablicom, n-torkom…) te ga mora držati zaključanog sve dok ne završi potrebne operacije. Nakon otpuštanja ključa, transakcija više ne smije zatražiti nikakav ključ. Faza otpuštanja ključeva najčešće je izvedena kroz jednu operaciju na kraju transakcije (COMMIT ili ROLLBACK).

3. Vrste ključeva

U ovisnosti o operacijama koje će transakcija obavljati nad podacima, potrebno ih je zaključati s odgovarajućim ključem. Ovisno s kojom je vrstom ključa podatak zaključan, drugi proces će moći na ograničen način baratati s tim podacima. Postoje tri vrste ključa.

i. Ključ za pisanje/izmjenu – WRITE LOCK

Transakcija T1 će s ovim ključem zaključati podatak da bi mogla pisati. Riječ je o ekskluzivnom zaključavanju, odnosno za to vrijeme druga transakcija T2 ne može ni pisati niti čitati navedeni podatak, sve dok ga prva ne otključa. Isto tako ga ne može niti zaključati.

ii. Ključ za čitanje - READ LOCK

Transakcija T1 će s ovim ključem zaključati podatak da bi mogla čitati podatke. Riječ je o zajedničkom zaključavanju pri čemu transakcija T2 može zaključati podatak za čitanje (ne i za pisanje).

iii. Unaprijedivi ključ – PROMOTABLE LOCK

Ovo je ključ koji omogućava DBMS-u da predviđa buduće akcije. Na početku će T1 postaviti ključ za čitanje, a prije izmjene podataka ključ se unaprjeđuje u ključ za pisanje. Unaprijedivi ključ za sada ne postoji u MySQL bazi.

Tabela 1. Pravila po vrstama ključeva

	READ	PROMOTABLE	WRITE	BEZ ZAKLJUČAVANJA
READ	OK	OK	Greška	OK
PROMOTABLE	OK	Greška	Greška	OK
WRITE	Greška	Greška	Greška	OK

4. Granulacija (zrnatost) zaključavanja podataka

Zaključavanje podataka i istodobnost pristupa podacima dva su povezana pojma, pri čemu dobar sustav zaključavanja omogućava visoku istodobnost. Jedan način da se poveća istodobnost je povećanje zrnatosti zaključavanja. Umjesto da se zaključaju cijele tablice, zaključava se samo dio koji sadrži potrebne podatke koji se koriste, te se na taj način dopušta više izmjena nad podacima istovremeno. Loša strana ovog pristupa je smanjenje performansi, te je potrebno pronaći ravnotežu između potrebnih performansi i potrebne zrnatosti zaključavanja. U nastavku je dan pregled tipova zrnatosti zaključavanja u MySQL-u. Prilikom definicije određene zrnatosti zaključavanja, važno je napomenuti da ne podržavaju svi tipovi mehanizma pohranjivanja podataka (eng. *storage engine*) sve zrnatosti zaključavanja.

i. Zaključavanje baze podataka

Osnovna sintaksa – stavljanje ključa:

```
FLUSH TABLES WITH READ LOCK
```

Osnovna sintaksa – skidanje ključa:

```
UNLOCK TABLES
```

Obzirom da postavljanje ovog ključa zaustavlja trenutni rad cijele baze, loše je za konkurentnost i ne preporuča se korištenje osim kod izrade backup-a baze. Iako početak transakcije implicitno otključava zaključane podatke, obzirom da se u ovom slučaju radi o globalnom zaključavanju, početak transakcije neće otpustiti globalni ključ.

ii. Zaključavanje tablica

Osnovna sintaksa – stavljanje ključa:

```
LOCK TABLES imeTablice READ | [LOW PRIORITY] WRITE LOCK
```

Osnovna sintaksa – skidanje ključa:

```
UNLOCK TABLES
```

Najjednostavnije zaključavanje, koje ujedno najmanje opterećuje brzinu izvršavanja operacija, je zaključavanje na nivou tablice. Na ovaj način zaključava se čitava tablica. Moguće je postaviti ovakav ključ na privremenu tablicu ali ga *engine* ignorira. Obzirom da privremena tablica vrijedi samo u trenutnoj sesiji nije ju potrebno zaključavati jer je drugi korisnici ionako neće 'vidjeti'.

Primjer:

Potrebno je tablicu nalog u bazi autoradionica zaključati za čitanje. Otključati tablicu.

```
LOCK TABLES nalog READ;
UNLOCK TABLES;
```

iii. Zaključavanje n-torke

Osnovna sintaksa – stavljanje ključa:

```
SELECT selectIzjava WHERE whereIzjava LOCK IN SHARE MODE;
SELECT selectIzjava WHERE whereIzjava FOR UPDATE;
```

Osnovna sintaksa – skidanje ključa:

```
Automatski po završetku transakcije.
```

Klauzula LOCK IN SHARE MODE odnosi se na ključ za čitanje, pri čemu je ostalim transakcijama dozvoljeno čitati ali ne i mijenjati podatke. Ako se zaključa podatak FOR UPDATE, podatak je zaključan s ekskluzivnim ključem pri čemu se očekuje da će prva transakcija podatke mijenjati odnosno drugim procesima nije dozvoljeno ni čitanje niti pisanje.

Zaključavanje na nivou retka pruža najbolju zrnatost zaključavanja, ali ujedno i najviše usporava izvršenje operacija.

Primjer:

Zaključati za čitanje sve n-torke u tablici radnik koje se odnose na radnike prezimena Parlov.

```
SELECT * FROM radnik WHERE prezimeRadnik='Parlov' LOCK IN SHARE MODE;
```

Primjer:

Započeti transakciju i zaključati za pisanje n-torku u tablici kvar koje se odnose na kvar naziva 'Uštimavanje pokretnog krova'. Pokušati promijeniti navedenu n-torku u drugoj sesiji.

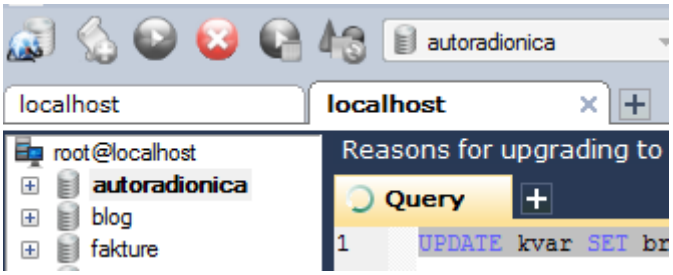
Prva sesija:

```
START TRANSACTION;
SELECT brojRadnika FROM kvar
        WHERE nazivKvar='Uštimavanje pokretnog krova' FOR UPDATE;
```

Druga sesija;

```
UPDATE kvar SET brojRadnika=brojRadnika + 1
        WHERE nazivKvar='Uštimavanje pokretnog krova';
```

U dugoj sesiji nije moguće provesti UPDATE naredbu jer je prva sesija zaključala n-torku za sebe. Pokušaj izvršavanja UPDATE naredbe u drugoj sesiji rezultira sljedećim:



Odnosno porukom o isteku vremena:

```
Error Code: 1205
Lock wait timeout exceeded; try restarting transaction
```

Prva sesija:

```
COMMIT WORK;
```

Tek bi se nakon završetka prve transakcije izvela UPDATE naredba u drugoj sesiji (ako u međuvremenu nije isteklo vrijeme – timeout).

iv. Zaključavanje indexa

Ovakva granulacija zaključavanja jest pod nadzorom DBMS-a. Indeks je potrebno zaključati s ciljem sprječavanja sablasnih n-torki (čitanja n-torki koje nikad nisu postojale).

5. Razine izoliranosti

Moguće je definirati hoće li rezultati transakcije biti vidljivi drugim transakcijama sve dok transakcija nije izvršena. Za navedeno se brine svojstvo izoliranosti, pri čemu postoje četiri nivoa izoliranosti: neizvršeno čitanje, izvršeno čitanje, ponavljano čitanje i serijalizacija. Navedeni nivoi određuju koje su promjene vidljive ostalim transakcijama odnosno koje nisu. Povećanjem razine izoliranosti sprečavaju se nepoželjne pojave nekonzistentnih čitanja podataka. U sljedećoj tablici prikazana je usporedba različitih razina izoliranosti.

Tabela 2. Usporedba različitih razina izoliranosti

Razina izoliranosti	Vidljivi rezultati neizvršenih transakcija	Pojava neponavljanih redaka	Pojava fantomskih redaka
Prljavo čitanje	da	da	da
Čitanje potvrđenih n-torki	ne	da	da
Ponavljano čitanje	ne	ne	da
Serializacija	ne	ne	ne

Osnovna sintaksa:

```
SET [GLOBAL | SESSION] TRANSACTION ISOLATION LEVEL
        {READ UNCOMMITTER | READ COMMITTED | REPEATABLE READ | SERIALIZABLE}
```

Razinu izolacije moguće je postaviti globalno za cijeli poslužitelj ili za trenutnu sesiju. Definira se na početku transakcije odnosno nakon BEGIN WORK.