SQLite

Sadržaj

Relacione baze podataka i SQLite

- Baze podataka omogućavaju čuvanje, ažuriranje i pretragu vrednosti
- Relacione baze podataka imaju koncept tabele (relacije) i elemenata u tabeli (n-torke)
- Relacione baze podataka: SQLite, MySQL, Microsoft SQL Server, Oracle Database itd.
- SQLite nema poseban servis niti zahteva poseban server da bude pokrenut kako bi se obavljale operacije nad bazom

SQL

- Relacioni upitni jezik
- Naredbe za kreiranje i brisanje tabela
  - CREATE TABLE
  - DROP TABLE
- Naredbe za upis, ažuriranje i pretragu podataka u bazi
  - INSERT
  - UPDATE
  - DELETE
  - SELECT

JDBC API

- Java Database Connectivity (JDBC) API (Application programming interface) standardan API koji omogućava pristup relacionim bazama podataka
- klase i interfejsi u paketu java.sql
- komunikacija putem SQL-a
- API implementira konkretna biblioteka za konkretnu bazu – SQLite, MySQL, Oracle, ...
- promena baze (npr. SQLite → MySQL) ne [bi trebalo da] zahteva promenu našeg koda

Uspostavljanje konekcije

- Inicijalizujemo drajver klasu

```
Class.forName("org.sqlite.JDBC");
```

- Uspostavimo konekciju sa bazom preko koje možemo da pošaljemo naredbe bazi

```
Connection c =
    DriverManager.getConnection("jdbc:sqlite:prodavnicaSQLite.db");
//naredbe koje saljemo bazi
c.close();
```

- Dok je konekcija uspostavljena možemo da šaljemo naredbe bazi
- Nakon što pošaljemo sve naredbe potrebno je pozvati metodu `close()` kako bi omogućili drugim programima da se konektuju na bazu

Izuzeci

- Mehanizam prijavljivanja greške
- Greška se signalizira ”bacanjem” izuzetka
- Metoda koja poziva potencijalno ”grešnu” metodu ”hvata” izuzetak
- Hijerarhija:
- Throwable – roditeljska klasa
  - Error – ozbiljne sistemske greške
  - Exception – bazna klasa za sve standardne izuzetke
    - unchecked: RuntimeException i njene naslednice – ne moraju da se obuhvate try/catch blokom
    - checked: Ostale klase koje nasleđuju Exception klasu i koje moraju da se obuhvate try/catch blokom

Izuzeci

- Checked (Exception i njene naslednice) – moraju da se uhvate
  - **EOFException**
  - **SQLException**
  - **...**
- Unchecked (RuntimeException i njene naslednice) – ne moraju da se uhvate, jer mogu da se programski spreče
  - **NullPointerException**
  - **IndexOutOfBoundsException**
  - **...**

Try catch finally blok

- Naredba za konekciju sa bazom će izazvati izuzetak kada se desi da ne možemo da otvorimo konekciju ka bazi ili kad ne uspe slanje SQL naredbe bazi
- finally blok će se izvršiti i kada se desi izuzetak i kada sve naredbe unutar try catch bloka završe bez greške

```
try {
  // kod koji moze da izazove
  // izuzetak
}
catch (java.io.EOFException ex) {
  System.out.println("Kraj datoteke pre vremena!");
}
catch (IndexOutOfBoundsException ex) {
  System.out.println("Pristup van granica niza");
}
catch (Exception ex) {
  System.out.println("Svi ostali izuzeci");
}
finally{
   // naredbe koje treba da se izvrse bez obzira da li se desio izuzetak
}
```

Izuzeci

- Programsko izazivanje izuzetka

```
throw new Exception("Ovo je jedan izuzetak");
```

- Korisnički definisani izuzeci

```
class MojException extends Exception {
  MojException(String s) {
            super(s);
          }
}
```

Izuzeci

- Ključna reč **throws**

```
void f(int i) throws MojException { ... }
```

- Propagacija izuzetaka
  - ne moramo da obuhvatimo try-catch blokom, već da deklarišemo da i pozivajuća metoda takođe baca izuzetak
  - tako možemo da prebacujemo odgovornost hvatanja izuzetka na gore

CREATE TABLE izraz

- Za kreiranje nove tabele koristi se SQL izraz

```
CREATE TABLE <naziv tabele>
(<naziv kolone1> <tip podatka> [NOT NULL],
<naziv kolone> <tip podatka> [NOT NULL], ..)
```

- Ključna reč NOT NULL se može izostaviti, ako se navede određuje se da ta kolona mora imati postavljenu vrednost

Kreiranje tabele za Racun

- Za SQL naredbe mora se kreirati objekat tipa `Statement`

```
Statement stmt = c.createStatement();
String sql = "CREATE TABLE artikal " +
    "(id       INT PRIMARY KEY     NOT NULL," +
    " naziv    TEXT     NOT NULL, " +
    " opis     TEXT      NOT NULL, " +
    " cena     REAL)";
stmt.executeUpdate(sql);
stmt.close();
```

- Da bi se SQL naredba izvršila poziva se metoda `executeUpdate` i prosleđuje joj se SQL naredba u vidu `String` objekta
- Nakon poslate naredbe treba osloboditi resurse koje zauzima objekat tipa `Statement` pozivanjem metode `close`

Tipovi podataka

- Osnovni tipovi koji se koriste u internim operacijama SQLite baze podataka

INTEGER	standardna vrednost celih brojeva sa dužinom zapisa od 1, 2, 3, 4, 6, ili 8 bajta
REAL	vrednost realnog broja sa 8 bajtova u standardnom IEEE zapisu
TEXT	tekstualna vrednost zapisana u jednom od podržanih načina zapisa (UTF-8, UTF-16BE ili UTF-16LE)
BLOB	vrednost koja se čuva onako kako je prosleđena u bazu
NUMERIC	može da čuva vrednosti bilo kojeg tipa vrednosti ako je moguća konverzija u broj

- Ostali tipovi se konvertuju u ove tipove podataka
- Ostali tipovi se koriste kada hoćemo da smanjimo količinu memorije koju baza zauzima

Tipovi podataka

INT INTEGER TINYINT SMALLINT MEDIUMINT BIGINT UNSIGNED BIG INT INT2 INT8	INTEGER
CHARACTER(20) VARCHAR(255) VARYING CHARACTER(255) NCHAR(55) NATIVE CHARACTER(70) NVARCHAR(100) TEXT CLOB	TEXT
REAL DOUBLE DOUBLE PRECISION FLOAT	REAL
NUMERIC DECIMAL(10,5) BOOLEAN DATE DATETIME	NUMERIC

Primarni ključ

- Vrednost koja jedinstveno određuje svaki red u tabeli je primarni ključ
- Može biti skup vrednosti iz više kolona ili samo vrednost jedne kolone
- U tabeli `artikal` to je kolona sa nazivom id i prilikom definisanja kolone dodate su ključne reči `PRIMARY KEY`
- Kolona id ima samo namenu da čuva vrednosti koje jedinstveno određuju svaki red tabele i to je veštački primarni ključ
- Prirodni primarni ključ su kolone koje imaju dodatnu namenu pored toga što jedinstveno određuju svaki red u tabeli, na primer: kolona index u tabeli Student

Strani ključ

- Kada hoćemo da definišemo da postoji veza između dve tabele navodimo posebnu kolonu koja će čuvati vrednosti primarnog ključa iz tabele sa kojom postoji veza
- Ove vrednosti se zovu strani ključ
- Kolonu za strane ključeve navodimo u tabeli koja je na više kraju veze
- Kada se definiše tabela sa stranim ključem potrebno je navesti posebnu kolonu za strani ključ i takođe navesti naredbu za očuvanje referencijalnog integriteta

```
FOREIGN KEY(naziv_kolone_stranog_kljuca)
REFERENCES naziv_tabele(naziv_kolone_primarnog_kljuca)
```

Tabela Stavka

- Tabela stavka sa kolonom `artikal_id` za strane ključeve i naredbom za očuvanje referencijalnog integriteta `FOREIGN KEY(artikal_id) REFERENCES artikal(id)`

```
CREATE TABLE stavka
(id              INT PRIMARY KEY     NOT NULL,
 kolicina        INT     NOT NULL,
 artikal_id      INT     NOT NULL,
 FOREIGN KEY(artikal_id) REFERENCES artikal(id))
```

INSERT izraz

- Za dodavanje novih vrednosti koristi se SQL izraz

```
INSERT INTO naziv_tabele
(naziv_kolone_atr1, naziv_kolone_atr2, ...)
VALUES (vrednost_atr1, vrednost_atr2, ... )
```

- Navodi se
  - naziv tabele u koju se upisuju vrednosti
  - naziv kolona u koje se upisuju vrednosti
  - vrednosti koje upisujemo u odgovarajuće kolone
- Redosled navođenja naziva kolona mora da odgovara redosledu navođenja vrednosti koje se upisuju u kolone

Dodavanje vrednosti za Artikal

- Potrebno je kreirati Statement objekat nad kojim se pozove executeUpdate metoda i prosledi SQL insert naredba u vidu String-a

```
Statement stmt = c.createStatement();
String sql = "INSERT INTO artikal (id,naziv,opis,cena) " +
"VALUES (1, 'Mleko', 'Mleko u flasi 1L', 40.99 );";
stmt.executeUpdate(sql);
stmt.close();
```

- Nakon poslate naredbe treba osloboditi resurse koje zauzima objekat tipa `Statement` pozivanjem metode `close`

SELECT izraz

- Za prikaz i pretragu vrednosti koristi se SQL izraz

```
SELECT lista_kolona
FROM lista_tabela
WHERE kriterijum_selekcije_redova_za_prikaz
ORDER BY naziv_kolone smer_sortiranja
```

- Za listu kolona se navode nazivi kolona čije vrednosti želimo da dobijemo, ako hoćemo vrednosti svih kolona navodimo `*`
- Za listu tabela se navodi iz kojih tabela želimo da pretražujemo vrednosti i vratimo vrednosti
- Za kriterijum selekcije navodimo uslov koji treba da bude zadovoljen da bi se vratile samo vrednosti iz određenih redova tabela koje pretražujemo
- Sa ORDER BY klauzulom se određuje kako želimo da vraćeni redovi tabela budu sortirani, navođenjem kolone po čijoj se vrednosti sortira i za smer sortiranja se može navesti ASC (predefinisana vrednost) sortiranje u rastućem redosledu ili DESC u opadajućem redosledu

SELECT izraz

- Za uslov u kriterijumu selekcije koristimo nazive kolona i sledeće operatore
  - Relacioni operatori
  - Logički operatori
  - Operator BETWEEN
    - Vrednost kolone se nalazi u zadatom opsegu
    - naziv_kolone BETWEEN min_vrednost AND max_vrednost
  - Operator IN
    - Kolona sadrži vrednosti iz skupa vrednosti
    - naziv_kolone IN (vrednost,vrednost,...)
  - Operator LIKE
    - Poređenje kolone sa zadatim šablonom
    - naziv_kolone LIKE vrednost
    - Šablon može da sadrži znake %, predstavlja bilo koji mogući znak ili skup znakova, i _, predstavlja samo jedan bilo koji znak
  - Operator IS NULL
    - Da li je vrednost dodeljena koloni
    - naziv_kolone IS NULL

Select izraz

- Logički operatori, navedeni po prioritetu:
  - NOT
  - AND
  - OR
- Relacioni operatori

=	jednako
<>	nije jednako (različito)
<	manje od
>	veće od
<=	manje ili jednako od
>=	veće ili jednako od

Select izraz primeri

- Potrebno je kreirati Statement objekat nad kojim se pozove executeQuery metoda i prosledi SQL select izraz u vidu String-a
- Kao rezultat se dobije ResultSet objekat koji u sebi čuva vrednosti svih traženih kolona i redova

```
Statement stmt = c.createStatement();
ResultSet rs = stmt.executeQuery( "SELECT * FROM artikal" );
while ( rs.next() ) {
        int id = rs.getInt("id");
        String naziv = rs.getString("naziv");
        String opis = rs.getString("opis");
        double cena = rs.getDouble("cena");
        System.out.println( "ID = " + id );
        System.out.println( "NAZIV = " + naziv );
        System.out.println( "OPIS = " + opis );
        System.out.println( "CENA = " + cena );
        System.out.println();
}
rs.close();
stmt.close();
```

- Prolazak kroz svaki poseban red omogućen je pozivom metode next nad ResultSet objektom
- Vrednost odgovarajuće kolone se dobije pozivom get metode za odgovarajući tip vrednosti koju kolona čuva
- Kad se prođe kroz sve vrednosti tabele treba osloboditi resurse koje koriste ResultSet i Statement objekti pozivom metode close

Select izraz primeri

- Prikaz svih vrednosti kolona artikala kod kojih se u nazivu pojavljuje slovo e i sortiranih po nazivu u rastućem redosledu

```
SELECT *
FROM artikal
WHERE naziv
LIKE '%e%'
ORDER BY naziv ASC
```

- Prikaz svih vrednosti kolona stavki gde je vrednost količine veća od 2 i sortiranih po količini u rastućem redosledu

```
SELECT *
FROM stavka
WHERE kolicina>2
ORDER BY kolicina
```

- Prikaz svih vrednosti kolona artikala kod kojih se cena nalazi u intervalu od 30 do 60

```
SELECT *
FROM artikal
WHERE cena BETWEEN 30 AND 60
```

Spajanje dve tabele JOIN

- Eksplicitno navođenjem u WHERE klauzuli koja kolona stranog ključa se poklapa sa kojom kolonom primarnog ključa tabela navedenih u FROM klauzuli

```
SELECT *
FROM artikal t1,stavka t2
WHERE t1.id=t2.artikal_id
```

- Koristeći JOIN operaciju spajanje dve tabele

```
SELECT *
FROM artikal a INNER JOIN stavka s
ON a.id=s.artikal_id
```

- Koristeći JOIN operaciju spajanje više tabela

```
SELECT *
FROM artikal a INNER JOIN stavka s
ON a.id=s.artikal_id
INNER JOIN racun r
ON s.racun_id=r.id
```

UPDATE izraz

- Za izmenu vrednosti u tabeli koristi se SQL izraz

```
UPDATE naziv_tabele
SET naziv_kolone1=vrednost1, naziv_kolone2=vrednost2, ...
WHERE kriterijum_selekcije_reda_za_izmenu
```

- Navodi se
  - naziv tabele u kojoj se menjaju vrednosti
  - naziv kolona čije vrednosti treba izmeniti
  - vrednosti koje upisujemo u odgovarajuće kolone
  - za kriterijum selekcije reda navodi se uslov kojim se pronalazi odgovarajući red za izmenu u tabeli
- Može se izmeniti više različitih redova sa jednim izrazom

Izmena vrednosti za Artikal

- Potrebno je kreirati Statement objekat nad kojim se pozove executeUpdate metoda i prosledi SQL update naredba u vidu String-a

```
Statement stmt = c.createStatement();
String sql = "UPDATE artikal SET " +
             "cena = 30.0,opis='Beli hleb 300g' " +
             "WHERE id=2;";
stmt.executeUpdate(sql);
stmt.close();
```

- Nakon poslate naredbe treba osloboditi resurse koje zauzima objekat tipa `Statement` pozivanjem metode `close`

DELETE izraz

- Za brisanje redova iz tabele koristi se SQL izraz

```
DELETE FROM naziv_tabele
WHERE kriterijum_selekcije_reda_za_brisanje
```

- Navodi se
  - naziv tabele iz koje se brišu redovi
  - za kriterijum selekcije reda navodi se uslov kojim se pronalaze svi redovi koje treba obrisati u tabeli
- Može se obrisati više različitih redova sa jednim izrazom
- Prvo treba obrisati redove iz tabela koje imaju postavljen referencijalni integritet, a tek onda se mogu obrisati redovi iz tabela koje su bile referencirane

Brisanje vrednosti za Stavku i Artikal

- Potrebno je kreirati Statement objekat nad kojim se pozove executeUpdate metoda i prosledi SQL delete naredba u vidu String-a

```
Statement stmt = c.createStatement();
String sql = "DELETE FROM stavka where ID=1;";
stmt.executeUpdate(sql);
stmt.close();
```

- Nakon poslate naredbe treba osloboditi resurse koje zauzima objekat tipa `Statement` pozivanjem metode `close`
- Kad je obrisana stavka može se obrisati i artikal koji je referencirala stavka

```
Statement stmt = c.createStatement();
String sql = "DELETE FROM artikal where ID=1;";
stmt.executeUpdate(sql);
stmt.close();
```

DROP TABLE izraz

- Za brisanje tabele koristi se SQL izraz

  `DROP TABLE IF EXISTS <naziv tabele>`
- Navodi se
  - naziv tabele koju hoćemo da obrišemo
- Brisanjem tabele brišu se i sve vrednosti koje su bile upisane u tu tabelu
- Prvo treba obrisati tabele koje imaju postavljen referencijalni integritet, a tek onda se mogu obrisati tabele koje su bile referencirane

Brisanje tabele za Stavke i Artikal

- Potrebno je kreirati Statement objekat nad kojim se pozove executeUpdate metoda i prosledi SQL drop table naredba u vidu String-a

```
Statement stmt = c.createStatement();
String sql = "DROP TABLE IF EXISTS stavka";
stmt.executeUpdate(sql);
stmt.close();
```

- Nakon poslate naredbe treba osloboditi resurse koje zauzima objekat tipa `Statement` pozivanjem metode `close`
- Kad je obrisana tabela stavka može se obrisati i tabela artikal na koju je postojala zaštita referencijalnog integriteta iz tabele stavka

```
Statement stmt = c.createStatement();
sql = "DROP TABLE IF EXISTS artikal;";
stmt.executeUpdate(sql);
stmt.close();
```