SQLite

Uvod

SQLite je ''open source'' biblioteka koja omogućava rad sa relacionim bazama podataka. Male je veličine i laka za podešavanje te joj je najčešća primena kao lokalno skladište za ugrađene (eng. embedded) uređaje i pametne telefone.
Relacione baze podataka organizuju resurse u tabele čije vrste predstavljaju objekte (elemente). Jedna baza može sadržati jednu ili više tabela koje mogu da budu u relaciji. Objekti sadrže polja koja predstavljaju atribute (osobine) tog objekta i oni su definisani u kolonama tabele. Svaki objekat mora da bude unikatan, što znači da mora postojati bar jedno polje koje će jedinstveno identifikovati svaki element, kako bi se zaobišli konflikti prilikom selekcije željenih elemenata iz tabele.

Tipovi podataka za skladištenje

SQLite tipovi podataka specificiraju tip resursa koji se može čuvati u jednoj koloni tabele. Ta kolona predstavlja atribut koji opisuje sve objekte u tabeli, i svaka kolona može čuvati atribut različitog tipa. Postoji pet tipova podataka za SQLite bazu.

- NULL – prazno polje, informacija je trenutno nepoznata
- INTEGER – označeni celi broj
- REAL – decimalna vrednosti
- TEXT – niz karaktera
- BLOB – binarni objekat koji može čuvati bilo koju vrstu podataka

SQLite ne podržava boolean tip te se za tu potrebu može iskoristiti INTEGER (0 – false, 1 - true), takođe ne podržava datum (date) tip, za čiju alternativu se može koristiti TEXT tip.

SQLite komande

Komanda	Opis
CREATE	Kreira novu tabelu.
SELECT	Filtrira podake iz jedne ili više tabela.
INSERT	Formira novi element i stavlja ga u tabelu.

UPDATE	Menja element u tabeli.
DELETE	Briše element iz tabele.
DROP	Briše čitavu tabelu.

Upravljanje SQLite bazom se vrši uz pomoć navedenih komandi kombinujući ih sa odgovarajućim parametrima, formirajući upit tkzv. ''query'' koji obavlja jednu od predefinisanih operacija. SQLite sintaksa je ''case insensitive'' što znači da ne razaznaje razliku između velikih i malih slova.

Slede primeri i pojašnjenja najvažnijih SQLite komandi.

Primer kreiranja tabele STUDENT sa poljima ''FirstName'', ''SecondName'' i ''Age'':

```
CREATE TABLE STUDENT ( StudentId INTEGER, FirstName TEXT, SecondName TEXT, Age INTEGER );
```

Izgled prazne tabele posle izvršenja komande **CREATE**.

StudentId	FirstName	SecondName	Age

```
INSERT INTO  STUDENT VALUES (1, John , Perry , 23);

INSERT INTO  STUDENT VALUES (2, Mark, Smith , 30);

INSERT INTO  STUDENT VALUES (3, Trevor, Johnson , 45);
```

Izgled tabele posle dodavanja par elemenata koristići komandu **INSERT**.

StudentId	FirstName	SecondName	Age
1	John	Perry	23
2	Mark	Smith	30
3	Trevor	Johnson	45

Sledi primer brisanja unosa iz tabele koristeći komandu **DELETE**. Komanda delete zahteva uslov za selekciju na osnovu koga će tabela biti filtrirana a potom pronađeni jedan ili više elemenata će biti obrisani.

Delete pattern DELETE from <**TABLE_NAME**> WHERE <**SEARCH_CONDITION**>

```
DELETE FROM  STUDENT WHERE Age=23;
```

Dati upit će obrisati sve elemente koji sadrže polje Age čija vrednost iznosi 23 .

Kako bi selektovali (filtrirali) elemente iz tabele koristi se komanda **SELECT.** Slično komandi delete, select u upitu sadrži uslov na osnovu koga će podaci iz tabele biti filtrirani.

```
SELECT FistName FROM  STUDENT;
```

Dati primer će vratiti sve vrednosti iz tabele STUDENT u koloni ''FirstName'', što su u ovom slučaju imena- John, Mark, Trevor.

Select komandu je moguće pozivati zadajući uslov na osnovu koga će podaci iz tabele biti filtrirani. Uslova može biti i više, i oni se tada lančaju koristeći ključnu reč **AND** između više uslova.

```
SELECT FistName FROM  STUDENT WHERE Age=30 AND FirstName=''Mark''
```

SQLiteOpenHelper

SQLiteOpenHelper je klasa koju je potrebno naslediti kako bi se na lak način izvršila inicijalizacija baze i abstrahovale sve funkcionalnosti poput kreiranja i ažuriranja baze podataka. Ova klasa omogućava korišćenje dve bitne metode: getWritableDatabase i getReadableDatabase koje vraćaju instancu baze u modu za čitanje i pisanje. Ove metode su ''thread safe'' što znači da se može pristupati bazi sa sigurnošću da će sve niti koje pristupaju bazi biti sinhronizovane kako se nebi desilo da dve niti u isto vreme upisuju ili čitaju podatke, čime se zaobilaze konflikti. Klasa SQLiteOpenHelper sadrži dve bitne ''callback metode'': onCreate i onUpgrade. Metoda onCreate se poziva kada se prvi put kreira baza podataka i u njoj je potrebno izvršiti kreiranje i inicijalizaciju tabela. onUpgrade se poziva kada se promeni verzija baze podataka te je u ovom callback-u moguće odraditi određenu akciju u zavisnosti od verzije baze.

SQLiteDatabase

SQLiteDatabase klasa omogućava izvršavanje standardnih SQL komandi – CREATE, DELETE, EXECUTE. Biće navedene najbitnije metode ove klase kao i primeri korišćenja.

execSQL metoda omogućava izvršavanje SQL upita prosleđujući joj tekst upita kao String objekat a kao rezultat vraća rezultat tražene operacije.

Primer poziva execSQL kojoj se prosleđuje upit kreiranja tabele sa tri polja TEXT tipa.

```
db.execSQL("CREATE TABLE " + TABLE_NAME + " (" +
                    COLUMN_FIRST_NAME + " TEXT, " +
                    COLUMN_LAST_NAME + " TEXT, " +
                    COLUMN_INDEX + " TEXT);"
);
```

insert metoda koja omogućava dodavanje jedne vrste u bazu bez direktnog korišćenja SQL upita, ali kao parameter koristi ContentValues klasu kao vid privremenog skladišta. ContentValues prima podatke u vidu *ključ* i *vrednost*.

Sledi primer skladištenja podataka u ContentValues objektu koji se koristi za smeštanje podataka u tabelu.

```
SQLiteDatabase db = getWritableDatabase();

ContentValues values = new ContentValues();

 values.put(COLUMN_FIRST_NAME, student.getFirstName());

 values.put(COLUMN_LAST_NAME, student.getLastName());

 values.put(COLUMN_INDEX, student.getIndex());

db.insert(TABLE_NAME, null, values);

close();
```

Svaki put kada je završeno dodavanje ili čitanje iz baze potrebno je pozvati metodu close() koja zatvara instance baze.

delete je metoda koja omogućava brisanje svih unosa koji ispunjavaju predviđeni uslov.

```
SQLiteDatabase db = getWritableDatabase();

 db.delete(TABLE_NAME, COLUMN_INDEX + "=?", new String[] {index});

 close();
```

query je metoda koja omogućava izvršavanja upita nad određenom tabelom vraćajući objekat Cursor. *Cursor* je interfejs koji predstavlja dvodimenzionalnu tabelu baze. Cursor koji je dobijen kao povratna vrednost pokazuje na prvi element iz filtriranih rezultata i uz pomoć ugrađenih cursor metoda moguće je iterirati i preuzimati željene podatke.

```
SQLiteDatabase db = getReadableDatabase();

 Cursor cursor = db.query(TABLE_NAME, null, COLUMN_INDEX + "=?",new String[] {index}, null,
null,null);

 for(cursor.moveToFirst(); !cursor.isAfterLast(); cursor.moveToNext()) {

        // iteration through cursor object . . .

 }
```