ANDROID PROGRAMIRANJE

Vežba 6: Rad sa SQLite bazama podataka

(radi se dve nedelje)

Teorijske napomene

SQLite je **relaciona baza podataka** ugrađena u Android operativni sistem. Koristi se za čuvanje lokalnih podataka aplikacije i omogućava lako upravljanje podacima. SQLite baza je laka, brza i ne zahteva dodatnu instalaciju. Svi podaci se skladište u fajl na uređaju, što je čini pogodnom za aplikacije koje rade offline.

Ključne karakteristike SQLite baze podataka:

1. **Lagana i ugrađena** – SQLite dolazi uz Android, pa nije potrebna dodatna instalacija ili konfiguracija.
2. **Relaciona baza podataka** – Podaci se organizuju u tabele sa kolonama i redovima.
3. **Podržava SQL upite** – Omogućava izvršavanje standardnih SQL komandi za upravljanje podacima.
4. **Dostupnost podataka** – Podaci se čuvaju u fajlu i ostaju dostupni čak i kada je aplikacija zatvorena.

DB Browser for SQLite je alat za pregled i manipulaciju SQLite baza podataka na računaru. Kroz ovaj alat možete:

- Kreirati, uređivati i brisati baze podataka.
- Izvršavati SQL upite.
- Dodavati, ažurirati i brisati podatke u tabelama.

Kako instalirati DB Browser for SQLite

- Preuzmite alat sa zvaničnog sajta: https://sqlitebrowser.org.
- Instalirajte ga i pokrenite ga prateći korake instalacionog programa za vaš operativni sistem (Windows, macOS ili Linux).

Kreiranje baze podataka

Kao što je već napomenuto, **DB Browser for SQLite** je jednostavan alat koji omogućava kreiranje i upravljanje SQLite bazama podataka, gde čak nije ni potrebno pisati SQL kod s obzirom da ima pregledan grafički korisnički interfejs. Koristan je za testiranje ili brzo podešavanje strukture baze podataka pre nego što je povežete sa Android aplikacijom. Slede najbitniji koraci za kreiranje i testiranje baze podataka:

1. **Kreiranje baze podataka**:
    - Pokrenite DB Browser for SQLite.
    - Kliknite na **New Database** (Nova baza podataka).
    - Unesite ime baze (npr. students.db) i izaberite lokaciju za čuvanje.
    - Dodajte tabelu:
        - Kliknite na **Create Table** (Kreiraj tabelu).
        - Definišite ime tabele (npr. students) i dodajte sledeće kolone:
            - id (INTEGER, PRIMARY KEY, AUTOINCREMENT),
            - name (TEXT, NOT NULL),
            - age (INTEGER, NOT NULL).
    - Sačuvajte promene.
2. **Dodavanje stranog ključa**:
    - SQLite ima opciju za strane ključeve, ali ona nije uključena po podrazumevanoj vrednosti.
    - Kada koristite DB Browser, ovo se automatski omogućava ako u strukturi tabele definišete strani ključ.
    - Na primer, kreirajte tabelu courses.

      Kolone:

        - course_id (INTEGER, PRIMARY KEY, AUTOINCREMENT),
        - course_name (TEXT, NOT NULL),
        - student_id (INTEGER, FOREIGN KEY koji referencira id iz tabele students).
    - Zatim, na dnu prozora za kreiranje tabela kliknite na Foreign Keys.
        - Kliknite na Add.
        - U polju From izaberite kolonu student_id.
        - U polju Table izaberite tabelu students.
        - U polju To izaberite id (primarni ključ iz glavne tabele).
        - Kliknite OK i sačuvajte promene.

DB Browser će osigurati da u tabeli courses, student_id uvek referencira validni id iz tabele students.

3. **Provera podataka**:
    - Kliknite na **Browse Data** da biste pregledali podatke u tabeli.
    - Kliknite na **Edit Pragmatically** da biste uneli test podatke, npr.:
        - (1, "Marko", 20)
        - (2, "Jelena", 22).
    - Potom učinite isto i sa courses tabelom:

      Unesite podatke u tabelu courses:
        - (1, "Matematika", 1)
        - (2, "Programiranje", 2)

Ako želite da dodate strani ključ direktno pomoću SQL upita:

```
CREATE TABLE students (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    age INTEGER NOT NULL
);

CREATE TABLE courses (
    course_id INTEGER PRIMARY KEY AUTOINCREMENT,
    course_name TEXT NOT NULL,
    student_id INTEGER,
    FOREIGN KEY(student_id) REFERENCES students(id)
);
```

Napomena o greškama pri unosu stranog ključa

- Ako unesete student_id u tabelu courses koji ne postoji u tabeli students, baza će prikazati grešku jer strani ključ ne može da referencira nepostojeći zapis.
- Da biste izbegli probleme:
    - Prvo unesite podatke u glavnu tabelu (students).
    - Zatim unesite podatke u detaljnu tabelu (courses).

Povezivanje baze sa Android aplikacijom

Android aplikacije sa SQLite bazom podataka omogućavaju skladištenje, preuzimanje i generalno manipulaciju podacima. Ovo je korisno za čuvanje korisničkih informacija, beleženje aktivnosti, offline rad i još mnogo toga. U nastavku su dati najbitniji koraci za komunikaciju baze podataka sa komponentama Android aplikacije.

- **Dodavanje baze u aplikaciju:**
  Nakon kreiranja baze u DB Browseru, dodajte .db fajl u folder **assets** u vašem projektu. Ako nemate već kreiranu bazu, ispratite korake na narednoj strani da vidite kako možete i direktno iz Android aplikacije da je napravite.
- **Kreiranje SQLiteOpenHelper klase**:
  SQLiteOpenHelper je klasa u Androidu koja pojednostavljuje rad sa SQLite bazom. Ova klasa pomaže u kreiranju, otvaranju i upravljanju verzijama baze.
- **Pisanje SQL upita za kreiranje i migraciju**:
  Unutar onCreate metode definišu se tabele.
  Unutar onUpgrade metode vrši se ažuriranje baze ako promenite njenu verziju.
- **Otvaranje baze i izvršavanje operacija**:
  Nakon što je baza povezana, koristi se za CRUD operacije.

Slede primeri koda za navedene operacije.

Ako koristite postojeću bazu, možete preskočiti pisanje SQL upita za kreiranje tabela. U ovom slučaju, fokus je na kopiranju baze iz assets foldera u aplikacioni folder gde baza može biti korišćena.

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "students.db";
    private static final int DATABASE_VERSION = 1;
    private static String DATABASE_PATH = "";
    private final Context context;
```

```
    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
        this.context = context;
        DATABASE_PATH = context.getFilesDir().getPath() + "/" + DATABASE_NAME;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // Nije potrebno, baza je već kreirana u DB Browseru
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Ako promenite strukturu baze, obavite migraciju ovde
    }

    public void copyDatabase() {
        try {
            File dbFile = new File(DATABASE_PATH);
            if (!dbFile.exists()) {
                InputStream inputStream = context.getAssets().open(DATABASE_NAME);
                OutputStream outputStream = new FileOutputStream(DATABASE_PATH);

                byte[] buffer = new byte[1024];
                int length;
                while ((length = inputStream.read(buffer)) > 0) {
                    outputStream.write(buffer, 0, length);
                }
                outputStream.flush();
                outputStream.close();
                inputStream.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public SQLiteDatabase openDatabase() {
        return SQLiteDatabase.openDatabase(DATABASE_PATH, null,
                                           SQLiteDatabase.OPEN_READWRITE);
    }
}
```

U glavnoj aktivnosti aplikacije, inicijalizujte i povežite bazu:

```
DatabaseHelper dbHelper = new DatabaseHelper(this);
dbHelper.copyDatabase();
SQLiteDatabase db = dbHelper.openDatabase();

// Provera
if (db != null) {
    Log.d("Database", "Baza uspešno otvorena!");
}
```

Ako nemate kreiranu bazu i želite da je napravite u aplikaciji, možete implementirati klasu koja kreira bazu podataka unutar aplikacije ako ona ne postoji, i omogućava ažuriranje baze pomoću onUpgrade metode.

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "students.db";
    private static final int DATABASE_VERSION = 1;

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // Kreiranje tabele
        String CREATE_TABLE = "CREATE TABLE students (" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "name TEXT NOT NULL, " +
                "age INTEGER NOT NULL);";
        db.execSQL(CREATE_TABLE);
    }
```

```
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Ažuriranje baze
        db.execSQL("DROP TABLE IF EXISTS students");
        onCreate(db);
    }
}
```

onCreate: Sadrži SQL upit za kreiranje potrebnih tabela (u ovom slučaju tabela students).

onUpgrade: Koristi se kada se promeni verzija baze. Na primer, možete dodati novu tabelu ili kolonu.

Otvaranje baze u glavnoj aktivnosti:

```
DatabaseHelper dbHelper = new DatabaseHelper(this);
SQLiteDatabase db = dbHelper.getWritableDatabase();

// Provera
if (db != null) {
    Log.d("Database", "Baza uspešno kreirana i otvorena!");
}
```

Razlike između ova dva pristupa

Korak	Postojeća baza (DB Browser)	Kreirana baza u aplikaciji
Dodavanje baze	Kopiranje `.db` fajla u `assets` folder	Nije potrebno – baza se kreira unutar aplikacije
Kreiranje tabela	Nije potrebno (već definisane u DB Browseru)	SQL upiti u `onCreate` metodi
Upotreba SQLiteOpenHelper	Koristi se samo za kopiranje baze iz `assets` foldera	Koristi se za kreiranje i upravljanje bazom
Migracija baze	Promene u DB Browser i ponovno dodavanje baze	Definišu se u `onUpgrade` metodi

Oba pristupa omogućavaju iste funkcionalnosti i podršku u radu sa SQLite bazama u Android aplikacijama. Ako baza već postoji (generisana u DB Browseru), celokupni process je lakši jer ne morate ručno da definišete tabele. S druge strane, kreiranje baze unutar aplikacije pruža veću fleksibilnost za dinamične aplikacije.

CRUD operacije

1. **Create (Dodavanje podataka)**:
   - Koristi se za umetanje novih redova u tabelu.
   - U SQLite se to postiže SQL upitom INSERT INTO.
   - U Androidu možemo koristiti metodu **insert()** za umetanje podataka.
2. **Read (Čitanje podataka)**:
   - Koristi se za dobijanje podataka iz baze.
   - Možete dohvatiti sve ili određene redove na osnovu uslova (WHERE).
   - U SQLite se ovo obavlja SQL upitom SELECT.
   - Android koristi metode poput **query()** ili **rawQuery()*.**
3. **Update (Ažuriranje podataka)**:
   - Koristi se za izmenu postojećih podataka.
   - U SQLite se koristi UPDATE sa uslovom WHERE da biste ažurirali određene redove.
   - Android pruža metodu **update().**
4. **Delete (Brisanje podataka)**:
   - Koristi se za brisanje određenih podataka iz baze.
   - U SQLite se ovo postiže SQL upitom DELETE FROM uz uslov WHERE.
   - Android koristi metodu **delete().**

*** query() vs. rawQuery()**

Koristite query() ako:

- Imate jednostavne SELECT upite i želite da aplikacija bude lakša za održavanje.
- Radite sa jednom tabelom i osnovnim filtriranjem podataka.

Koristite rawQuery() ako:

- Imate kompleksne upite, poput spajanja tabela (JOIN), grupisanja (GROUP BY), ili prilagođenih SQL funkcija.
- Želite potpunu kontrolu nad SQL sintaksom.

Npr, ako aplikacija zahteva spajanje tabela students i grades, rawQuery() je jedini izbor:

```
Cursor cursor = db.rawQuery(
    "SELECT s.id, s.name, g.grade FROM students s JOIN grades g ON s.id =
            g.student_id WHERE g.grade > ?", new String[]{"85"});
```

Pretpostavljamo da već imamo SQLiteOpenHelper klasu nazvanu **DatabaseHelper**, sa tabelom **students** koja ima sledeće kolone:

- id (INTEGER, primarni ključ)
- name (TEXT)
- age (INTEGER)

1. Create (Dodavanje podataka)

Dodavanje novog studenta u tabelu:

```
public long addStudent(String name, int age) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put("name", name);
    values.put("age", age);

    long id = db.insert("students", null, values);
    db.close(); // Zatvaranje baze nakon operacije

    return id; // Vraća ID novog reda
}
```

Pozivanje metode:

```
DatabaseHelper dbHelper = new DatabaseHelper(this);
long newId = dbHelper.addStudent("Marko", 22);
Log.d("DB", "Novi student dodat sa ID: " + newId);
```

2. Read (Čitanje podataka)

Čitanje svih podataka iz tabele, koristimo **Cursor** da prolazimo kroz ove rezultate.

Cursor se kreira kada izvršite upit na SQLite bazi koristeći metode query() ili rawQuery().

Cursor omogućava iteraciju kroz rezultate koristeći metode kao što su:

- moveToFirst(): Pomeranje na prvi red.
- moveToNext(): Pomeranje na sledeći red.
- moveToPrevious(): Pomeranje na prethodni red.
- moveToLast(): Pomeranje na poslednji red.
- isAfterLast(): Provera da li je pređena poslednja linija.

```
public List<String> getAllStudents() {
    List<String> students = new ArrayList<>();
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.query("students", null, null, null, null, null, null);
    if (cursor.moveToFirst()) {
        do {
            String name = cursor.getString(cursor.getColumnIndexOrThrow("name"));
            int age = cursor.getInt(cursor.getColumnIndexOrThrow("age"));
            students.add(name + " (" + age + " godina)");
        } while (cursor.moveToNext());
    }
    cursor.close();
    db.close();
    return students;
}
```

Pozivanje metode:

```
List<String> students = dbHelper.getAllStudents();
for (String student : students) {
    Log.d("DB", "Student: " + student);
}
```

Čitanje specifičnih podataka (npr. studenta po ID-ju):

```
public String getStudentById(int id) {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.query("students", null, "id=?", new
                    String[]{String.valueOf(id)}, null, null, null);
    String studentInfo = null;
    if (cursor.moveToFirst()) {
        String name = cursor.getString(cursor.getColumnIndexOrThrow("name"));
        int age = cursor.getInt(cursor.getColumnIndexOrThrow("age"));
        studentInfo = name + " (" + age + " godina)";
    }
    cursor.close();
    db.close();
    return studentInfo;
}
```

Pozivanje metode:

```
String student = dbHelper.getStudentById(1);
Log.d("DB", "Student sa ID 1: " + student);
```

3. Update (Ažuriranje podataka)

Ažuriranje podataka o studentu (npr. promena imena):

```
public int updateStudent(int id, String newName, int newAge) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put("name", newName);
    values.put("age", newAge);

    int rowsAffected = db.update("students", values, "id=?", new
                                  String[]{String.valueOf(id)});
    db.close();
    return rowsAffected; // Vraća broj ažuriranih redova
}
```

Pozivanje metode:

```
int rows = dbHelper.updateStudent(1, "Nikola", 23);
Log.d("DB", "Broj ažuriranih redova: " + rows);
```

4. Delete (Brisanje podataka)

Brisanje studenta na osnovu ID-ja:

```
public int deleteStudent(int id) {
    SQLiteDatabase db = this.getWritableDatabase();

    int rowsDeleted = db.delete("students", "id=?", new
                                 String[]{String.valueOf(id)});
    db.close();
    return rowsDeleted; // Vraća broj obrisanih redova
}
```

Pozivanje metode:

```
int rows = dbHelper.deleteStudent(1);
Log.d("DB", "Broj obrisanih redova: " + rows);
```

Zadatak za samostalni rad

U ovom zadatku treba proširiti prethodnu aplikaciju za rad sa bazama tako što ćete dodati nove funkcionalnosti. Novi zahtevi uključuju dodavanje novih kolona u tabelu students, proširenje interfejsa da prikazuje sve studente preko RecyclerView, filtriranje studenata po smerovima, i prikazivanje 3 najuspešnijih studenata prema njihovom proseku. Sve neophodne funkcionalnosti će se implementirati kroz SQLite operacije i UI komponente.

Zahtevi zadatka

1. **Proširenje tabele students**
   Tabela students treba da sadrži sledeće dodatne kolone:
   - department (TEXT) – smer studenta
   - gpa (REAL) – trenutni prosek studenta
2. **UI Komponente**
   - Kreirati RecyclerView koji će prikazivati listu svih studenata sa njihovim imenom, smerom i prosekom.
   - Dodati Spinner (padajuću listu) za filtriranje studenata prema smeru.
   - Dodati dugme za prikazivanje 3 najuspešnijih studenata prema njihovom proseku. Rezultat može izaći kao pop-up prozor ili u nekom tekstualnom polju da se prikaže na ekranu. Preporučujem da to bude pop-up prozor.
3. **Funkcionalnosti:**
   - Filtriranje studenata po smeru koristeći Spinner.
   - Dugme za prikazivanje 3 studenta sa najvećim prosekom.
   - Dodavanje novih studenata sa podacima o smeru i proseku.

Korisni resursi

Tutorijali i sajtovi sa praktičnim vežbama:

1. https://www.geeksforgeeks.org/how-to-create-and-add-data-to-sqlite-database-in-android/
2. https://medium.com/@shivani.patel18/a-comprehensive-guide-to-sqlite-databases-in-android-development-df74f01df6c3
3. https://www.javatpoint.com/android-sqlite-tutorial

Video tutorijali (prvi je za DB Browser, a drugi i treći za razvoj Android aplikacije):

4. https://www.youtube.com/watch?v=b0Dplx4M5zg&ab_channel=BoostMyTool
5. https://youtube.com/playlist?list=PLSrm9z4zp4mGK0g_0_jxYGgg3os9tqRUQ&si=G9KOdrA_hTzi1Oll