# **SQLITE baza podataka**

Android OS dolazi s već ugrađenom bazom podataka **sqlite**. Primjer kako pohraniti i izvući podatke iz baze.

## extend sqliteopenhelper

**sqliteopenhelper** je glavna klasa odgovorna za operacije baze i mora se naslijediti.

**public abstract class** dbmanager **extends** sqliteopenhelper{  
  
 **private static final** string ***db\_name*** = **"test.db"**;  
 **private** string **tablename**;  
 **private** string[] **columns**;  
  
 **public** dbmanager(context context,string tablename, string [] columns) {  
 **super**(context, ***db\_name***, **null**, 1);  
 **this**.**tablename** = tablename;  
 **this**.**columns** = columns;  
 }

Nakon proširenja sqliteopenhelpera, treba pokrenuti konstruktor *super* s odgovarajućim parametrima – kontekst (koja aktivnost se nasljeđuje), naziv datoteke db, cursorfactory (vraća podklasu kursora prilikom pozivanja upita, za početak vrijednost je null) i verzija baze db.

Context - omogućuje pristup resursima, omogućuje interakciju s drugim komponentama Androida slanjem poruka, daje informacije o okruženju aplikacije.

Radna dokumentacija

[Context](https://developer.android.com/reference/android/content/Context)

<https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper>

## Create table (stvaranje tablice)

SQL – CREATE TABLE ime\_tablice ( atribut1 integer primary key autoincrement, stupac1, stupac2,…stupacn);

**public abstract class** dbmanager **extends** sqliteopenhelper{  
  
 **private static final** string ***db\_name*** = **"test.db"**;  
 **private** string **tablename**;  
 **private** string[] **columns**;  
  
 **public** dbmanager(context context,string tablename, string [] columns) {  
 **super**(context, ***db\_name***, **null**, 1);  
 **this**.**tablename** = tablename;  
 **this**.**columns** = columns;  
 }  
  
 @override  
 **public void** oncreate(sqlitedatabase db) {  
 string sql = **"create table "** + **tablename** +  
 **" ("** + basecolumns.\_id + **" integer primary key autoincrement "**;  
  
 **for** (**int** i=0;i<**columns**.**length**; i++) {  
 sql+=**", "**+**columns**[i];  
 }  
  
 sql+=**" ) "**;  
 db.execsql( sql );

}

Jedna od funkcija koja se nasljeđuje od *sqliteopenhelpera* je *public void oncreate(sqlitedatabase db*), koja se pokreće kada je baza *db* stvorena. Ovu funkcija koristiti se za izradu tablice.

Metoda *execsql(naziv\_stringa)*, koristi se za izvršavanje SQL upita u bazi podataka, string sql počinje s *CreateTable…*

## Insert & Query (upis i upit)

**public void** insert(contentvalues values){  
 getwritabledatabase().insert(tablename, **null**, values);  
}  
  
**public** cursor getall(){  
 **return** getreadabledatabase().query(tablename, **null**, **null**, **null**, **null**, **null**, **null**);  
}

Radna dokumentacija:

<https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase>

public long insert (**[String](https://developer.android.com/reference/java/lang/String)** table,

[**String**](https://developer.android.com/reference/java/lang/String) nullColumnHack,

[**ContentValues**](https://developer.android.com/reference/android/content/ContentValues) values)

Upis retka(zapis ili slog) u bazu podataka. ContentValues ​​je objekt koji se koristi za čuvanje skupa parova ključ-vrijednost. Ovdje ključ mora biti valjano ime stupca tablice baze podataka, a vrijednost može biti bilo koji valjani podatak koji će se pohraniti.

Cursor - kada želimo komunicirati s bazom podataka, unutar Android aplikacije, moramo Aktivnosti dodijeliti pravo na pristup čitanju i pisanju sadržaju baze podataka. Iz tog razloga, Android ima sučelje kursora, koje pruža pristup čitanju i pisanju skupu rezultata koji vraća upit baze podataka.

public [**Cursor**](https://developer.android.com/reference/android/database/Cursor) query (**[String](https://developer.android.com/reference/java/lang/String)** table,

[**String[]**](https://developer.android.com/reference/java/lang/String) columns,

[**String**](https://developer.android.com/reference/java/lang/String) selection,

[**String[]**](https://developer.android.com/reference/java/lang/String) selectionArgs,

[**String**](https://developer.android.com/reference/java/lang/String) groupBy,

[**String**](https://developer.android.com/reference/java/lang/String) having,

[**String**](https://developer.android.com/reference/java/lang/String) orderBy,

[**String**](https://developer.android.com/reference/java/lang/String) limit)

Vraća rezultat upita. Ako upit vraća sve zapise, svi parametri su null osim naziva tablice.

public [**SQLiteDatabase**](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase) getWritableDatabase ()

Prije poziva metode potrebno je napraviti i/ili otvoriti bazu podataka koja će se koristiti za čitanje i pisanje. Prvi put kada se pozove, baza podataka će se otvoriti i pozvati onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int) i/ili onOpen(SQLiteDatabase).

Nakon uspješnog otvaranja, baza podataka se sprema u predmemoriju, tako da se može pozvati ovu metodu svaki put kada trebate pisati u bazu podataka. (Obavezno pozvati metodu close() kada više nema potrebe za bazu podataka). Pogreške kao što su loša dopuštenja ili pun disk mogu spriječiti realizaciju metode. Potrebno je otkloniti greške i ponovo pokrenuti metodu.

public [**SQLiteDatabase**](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase) getReadableDatabase ()

Prije poziva metode potrebno je napraviti i/ili otvoriti bazu podataka. Ovo će biti isti objekt koji vraća getWritableDatabase() osim ako postoji neki problem (pun disk). U tom slučaju objekt baze podataka biti će vraćen samo za čitanje.

Svaka jednostavna baza podataka ima 4 osnovne operacije koje se koriste kod aplikacija za trajnu pohranu. To su stvaranje(Create), čitanje (Read), ažuriranje (Update) i brisanje (Delete) podataka. Postoji kratica CRUD koja je je akronim za osnovne operacije nad podacima.

Primjeri:

public static [**SQLiteDatabase**](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase) openOrCreateDatabase (**[String](https://developer.android.com/reference/java/lang/String)** path,

[**SQLiteDatabase.CursorFactory**](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.CursorFactory) factory,

[**DatabaseErrorHandler**](https://developer.android.com/reference/android/database/DatabaseErrorHandler) errorHandler)

Path - String:

Factory, errorHandler - mogu imati vrijednost null

openOrCreateDatabase - metoda se koristi za stvaranje nove baze podataka ako ona ne postoji ili za otvaranje postojeće baze podataka na navedenom mjestu.

Context.MODE\_PRIVATE se koristi za deklariranje datoteka baze podatakakoja treba biti samo lokalna. Treći parametar je referenca za objekt CursorFactory. Ova vrijednost se prosljeđuje kao nulta referenca, budući da nema potrebe za prilagođenim objektom kursora (zadani objekt se preuzima prilikom čitanja iz baze podataka).

CREATE

**private void** createDatabase() {  
 SQLiteDatabase db = openOrCreateDatabase(**"Polaznici.db"**, Context.MODE\_PRIVATE, **null**);  
 db.execSQL(**"CREATE TABLE IF NOT EXISTS Polaznik(sifra integer PRIMARY KEY, ime TEXT, prezime TEXT, email TEXT);"**);  
 db.close();  
}

INSERT

**private void** insertPolaznik() {  
 SQLiteDatabase db = openOrCreateDatabase(**"Polaznici.db"**, Context.MODE\_PRIVATE, **null**);  
 ContentValues values = **new** ContentValues();  
 values.put(**"sifra"**, sifra.getText().toString());  
 values.put(**"ime"**, ime.getText().toString());  
 values.put(**"prezime"**, prezime.getText().toString());  
 values.put(**"email"**, email.getText().toString());  
 db.insert(**"Polaznik"**, **null**, values);  
 db.close();  
 **this**.finish();  
}

READ

**private** List<Polaznici> getPolaznik() {  
 List<Polaznici> list = **new** ArrayList<>();  
 SQLiteDatabase db = openOrCreateDatabase(**"Polaznici.db"**, Context.MODE\_PRIVATE, **null**);  
 Cursor cursor = db.rawQuery(**"SELECT \* FROM Polaznik"**, **null**);  
 cursor.moveToFirst();  
 **while**(!cursor.isAfterLast()) {  
 Polaznici polaznici = **new** Polaznici();  
 polaznici.setSifra(cursor.getString(0));  
 polaznici.setIme(cursor.getString(1));  
 polaznici.setPrezime(cursor.getString(2));  
 polaznici.setEmail(cursor.getString(3));  
 list.add(polaznici);  
 cursor.moveToNext();  
 }  
 cursor.close();  
 db.close();  
 **return** list;  
}

UPDATE

**private void** updatePolaznik() {  
 SQLiteDatabase db = openOrCreateDatabase(**"Polaznici.db"**, Context.MODE\_PRIVATE, **null**);  
 ContentValues values = **new** ContentValues();  
 values.put(**"sifra"**,sifra.getText().toString());  
 values.put(**"ime"**,ime.getText().toString());  
 values.put(**"prezime"**, prezime.getText().toString());  
 values.put(**"email"**, Email.getText().toString());  
 database.update(**"Polaznici"**, values, **"email = ?"**, **new** String[]{Polaznik.getEmail()});  
 db.close();  
 **this**.finish();  
}

DELETE

**private void** deletePolaznik() {  
 SQLiteDatabase db = openOrCreateDatabase(**"polaznici.db"**, Context.MODE\_PRIVATE, **null**);  
 db.delete(**"Polaznik"**, **"email= ?"**, **new** String[]{polaznik.getEmail()});  
 db.close();  
 **this**.finish();  
}

Metoda *rawQuery*()je ista kao *execSQL*(), koristi se za izvršavanje SQL upita u bazi podataka, ali se očekuje rezultat od metode rawQuery. Vraćeni rezultat je objekt Cursor, koji je objekt omota vraćenih redaka. Ovaj kod se ponavlja kroz kursor i stvara nove polaznike koristeći vraćene pojedinosti.

*rawQuery*() metoda ima dva parametra. Prvi je SQL upit kao String, a drugi su parametri za SQL upit kao niz String. Kako biste izbjegli SQL injekciju, preporuča se korištenje pripremljenih izraza. Drugi parametar se koristi za parametre pripremljene izjave. (metoda updateContact koristi ovu značajku)

db.delete(**"Polaznik"**, **"email= ?"**, **new** String[]{polaznik.getEmail()});

Prvi parametar metode SQLiteDatabase.update()je naziv tablice. Drugi parametar su ContentValues. Treći parametar je 'where klauzula', a posljednji parametar je niz parametara za pripremljenu izjavu. U ovom kodu se vraćena vrijednost polaznik.getEmail() koristi kao parametar klauzule.