Mobile applications

Master SDBIS/SIA

Octavian Dospinescu

2021

General topics

- SQLite – local databases in Android
  - Creating a local DB
  - How to use insert/delete/update
  - Specific classes:
    - SQLiteDatabase
    - SQLiteOpenHelper
    - Cursor

THE GOAL OF THE COURSE

- Learning concepts related to working with local database in SQLite
- Making operations in the database:
  - Read
  - Write (INSERT/DELETE/UPDATE)

SQLite characteristics

SQLite is a relational database that is often prevalent in mobile devices due to the following advantages:

- It requires no configuration. It is simple to be used by developers.
- It does not require a server to run.
- The entire database is stored in a single file for each application.
- It is open source.

Classes to be used

- SQLiteOpenHelper
- SQLiteDatabase
- Cursor

Class SQLiteOpenHelper

- It aims to facilitate the creation of a database on our local device .

  Events:

- **onCreate()** – It occurs when we have created a new database (on first run the application)

- **onUpgrade()** – It occurs when we upgrade the application (see DB_VERSION)

Class SQLiteOpenHelper - example

```
  private static final String CREATE_TABLE_CLIENTI = " create table “ + 
DBAdapter.DB_TABLE + " (_id integer primary key autoincrement," + " nume text, “ + "
adresa text, " + " telefon text);";
```

```
@Override
public void onCreate(SQLiteDatabase db) {
      db.execSQL(CREATE_TABLE_CLIENTI);
      // adaug clientii existenti
      this.adaugaClienti(db);
}
```

Class SQLite OpenHelper - example

```
                    @Override
                    public void onUpgrade(SQLiteDatabase db, int oldVersion, int 
newVersion)
                    {
                             db.execSQL("DROP TABLE IF EXISTS " + DBAdapter.DB_TABLE);
                             onCreate(db);
                    }
```

Class SQLiteOpenHelper

It allows to get an instance of the database, by using the **constructor** and the method **getWritableDatabase()**.

```
  private SQLiteDatabase myDb;
  private DatabaseHelper myDbHelper;  //DatabaseHelper extends SQLiteOpenHelper
```

```
public DBAdapter open() throws SQLException {
         this.myDbHelper = new DatabaseHelper(this.context, DB_NAME, null,
                                  DB_VERSION);
         this.myDb = this.myDbHelper.getWritableDatabase();
         return this;
}
```

Example: method for adding a new record

- We use value pairs (field - value) through objects of **ContentValues** type.

```
public void adaugaClienti(SQLiteDatabase db) {
        ContentValues cv = new ContentValues();
        cv.put("nume", "Berariu Iulian");
        cv.put("adresa", "Str. Florilor, nr.1, Iasi");
        cv.put("telefon", "0723123321");
        db.insert("clienti", null, cv);
        cv.put("nume", "Spiridon Marcica");
        cv.put("adresa", "Str. Liliacului, nr.3, Comanesti");
        cv.put("telefon", "0745234543");
        db.insert("clienti", null, cv);
        cv.put("nume", "Straton Mircea");
        cv.put("adresa", "Str. Teiului, nr.2, Valea Lupului");
        cv.put("telefon", "0723123321");
        db.insert("clienti", null, cv);
}
```

Class Cursor

- From a conceptual standpoint, it is similar to an Oracle cursor.
- It is used to “catch” the results of the queries from the database.

Class Cursor – example

```
        // Metode pentru a interoga baza de date
        public Cursor getAllClients() {
                return this.myDb.query(DBAdapter.DB_TABLE, new String[] { "_id",
                                "nume", "adresa", "telefon" }, null, null, null,
null, null);
        }
```

Class Cursor – a complex example

```
Cursor cursorClienti = myDBAdapter.getAllClients();

    //pun clientii intr-un vector pentru a-i putea afisa
    int numarValori;
    numarValori = cursorClienti.getCount();

    String[] valori;
    valori = new String[numarValori];


    cursorClienti.moveToFirst();

    int i=0;
    while(cursorClienti.isAfterLast() == false) {
            //creez un obiect de tip Client
            Client client;
            client = new Client();
            client.setId(Integer.valueOf(cursorClienti.getInt(0)));
            client.setNume(cursorClienti.getString(1));
            client.setAdresa(cursorClienti.getString(2));
            client.setTelefon(cursorClienti.getString(3));

            //il adaug in vector
            valori[i] = client.getNume();
            //merg la urmatorul client
            i++;
            cursorClienti.moveToNext();
    }

    //inchid cursorul
    cursorClienti.close();
```

Practical implementation ☺