Aplicatii mobile pentru afaceri

Master SIA

Lect. Octavian Dospinescu

Tematica generala curs

- SQLite – baze de date locale in Android
  - Creare BD locala
  - Utilizare pentru insert/delete/update
  - Clase specifice:
    - SQLiteDatabase
    - SQLiteOpenHelper
    - Cursor

SCOPUL CURSULUI

- Deprinderea notiunilor legate de lucrul cu baze de date locale SQLite
- Efectuarea de operatiuni in baza de date:
  - Citire
  - Scriere (INSERT/DELETE/UPDATE)

Caracteristici SQLite

SQLite este o bază de date relațională care este cel mai adesea răspândită în cadrul dispozitivelor mobile, datorită următoarelor avantaje:

- Nu necesită configurare, este simplu de folosit de către dezvoltatori;
- Nu necesită un server pentru a rula;
- Întreaga bază de date este stocată într-un singur fișier, pentru fiecare aplicație în parte;
- Este open source.

Clase utilizate

- SQLite OpenHelper
- SQLiteDatabase
- Cursor

Clasa SQLiteOpenHelper

Are rolul de a facilita crearea unei baze de date pe dispozitivul local.

Evenimente:

- **onCreate()** – apare atunci cand trebuie creata o noua BD (la prima rulare a aplicatiei)
- **onUpgrade()** – apare atunci cand se face upgrade la aplicatie (de vazut DB_VERSION)

Clasa SQLiteOpenHelper - exemplu

```
  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);
}
```

Clasa SQLite OpenHelper - exemplu

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

Clasa SQLiteOpenHelper

Permite obtinerea unei instante a unei baze de date, prin intermediul **constructorului** si a metodei **getWritableDatabase()**.

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

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

Exemplu metoda pentru adaugarea unei noi inregistrari

Se folosesc perechi de valori (camp-valoare) prin intermediul unor obiecte de tip **ContentValues**.

```
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);
}
```

Clasa Cursor

- Din punct de vedere conceptual, este asemanatoare cu un cursor Oracle.

- Este folosita pentru a “prinde” rezultatele interogarilor din baza de date.

Clasa Cursor – exemplu de utilizare

```
        // 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);
        }
```

Clasa Cursor – exemplu complex

```
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();
```

Implementare practica ☺