P82 Mobile Application Development - Android

Databases

Databases

- Example uses :
  - For a list of contacts
  - For a list of music
  - To manage a large amount of information

- Problems :
  - The data is stored on the phone, so only accessible by the phone
  - No servers for SQLite... A large amount of data to process influences the performance of the phone
  - Accessible only within the application (MODE_PRIVATE)

Creating Databases

- Create a class that inherits from the SQLiteOpenHelper object,
- Syntax to create a table:

```
CREATE TABLE table_name (
   field_name_1 type {constraints},
   field_name_2 type {constraints},
   ...);
```

- Only 5 types in SQLite
  - NULL for NULL data.
  - INTEGER for integers (without a comma).
  - REAL for real numbers (with comma).
  - TEXT for strings.

Constraints

- PRIMARY KEY to designate the primary key on an attribute;
- NOT NULL to indicate that this attribute can not be NULL;
- CHECK to verify that the value of this attribute is consistent;
- DEFAULT is used to specify a default value.

Creating Databases

```
public static final String STD_KEY = "id";
public static final String STD_LASTNAME = "name";
public static final String STD_FIRSTNAME = "firstname";
public static final String STD_AGE = "age";
public static final String STD_TABLE_NAME = "student";

public static final String STD_TABLE_CREATE =
"CREATE TABLE" + STD_TABLE_NAME + "("
+ STD_KEY + "INTEGER PRIMARY KEY AUTOINCREMENT,"
    + STD_LASTNAME + "TEXT,"
    + STD_FIRSTNAME + "TEXT,"
+ STD_AGE + "INTEGER);";

public static final String STD_TABLE_DROP =
"DROP TABLE IF EXISTS" + STD_TABLE_NAME + ";";
```

Database Creation / Updating

SQLiteOpenHelper has two methods to override

Method called the first time if there is not this database in the phone:

```
@Override
public void onCreate (SQLiteDatabase db) {
    db.execSQL (STD_TABLE_CREATE);
}
```

Method called and executed if the version number changes

```
public static final String STD_TABLE_DROP = "DROP TABLE IF EXISTS" +
STD_TABLE_NAME + ";";
@Override
public void onUpgrade (db SQLiteDatabase, int oldVersion, int newVersion) {
    db.execSQL (STD_TABLE_DROP);
    onCreate (db);
}
```

Make Entities out of our tables

```
public class Student {
int id;
String lastName;
String firstName;
int age;

public Student () {}

public Student (String lastName, String firstName, int age) {
     this.lastName = lastName;
     this.firstName = firstName;
     this.age = age;
}
// getter / setter
}
```

Getting data from a database

To get the database you can use

```
SQLiteDatabase db = gDB.getWritableDatabase ();
```

Where gDB and our class that inherits from SQLiteOpenHelper

Create a class database Connection

- public class DatabaseConnection {

```
    private static SQLiteDatabase db;
    private static String dataBase = "myDB";
    private static int version = 1;

    public static SQLiteDatabase getdb(Context ctx){
        Gestiondb gDB = new Gestiondb(ctx,dataBase,null,version);
        return db = gDB.getWritableDatabase();
    }

    public static void closedb(){
        db.close();
    }

}
```

Managing Table (Add)

```
public static long add (STD student, Context ctx) {
     ContentValues cv = new ContentValues ();
     cv.put (DBAdministrationSTD_LASTNAME, STD.getLastName ());
     cv.put (Management.db.STD_FIRSTNAME, STD.getFirstName ());
     cv.put (Management.STD_AGE, STD.getAge ());

     SQLiteDatabase db = ConnectionDB.getdb (ctx);

     long id = db.insert (DB.STD_TABLE_NAME, null, CV);

     ConnectionDB.closedb ();
     return id;
}
```

Use a ContentValues to pass the parameters
Call the insert method on our db instance
LASTNAME of the table, case or value are null, value to insert

Managing Table (Remove)

```
public static void delete (int id, Context ctx) {
         SQLiteDatabase db = ConnectionDB.getdb (ctx);
        db.delete (Gestiondb.STD_TABLE_NAME,
        Managementdb.STD_KEY + "=?",
        new String [] {String.valueOf (id)});
}
```

int delete (String table, String whereClause, String [] whereArgs).

table is the identifier of the table.

Where Clause is the WHERE in SQL.
"Id = 1"
"Id =? "
whereArgs is an array of values that will replace the "?” in whereClause
whereClause is "LASTNAME LIKE? AND age>?"
whereArgs a String[] of the type {"b", "10"}.

Managing Table (Modify)

```
public static void update (STD Student, Context ctx) {
    ContentValues value = new ContentValues ();
    value.put (DBManagementSTD_FIRSTNAME, STD.getFirstName());
    value.put (DBManagement.STD_LASTNAME, STD.getLastName());

    SQLiteDatabase db = ConnectionDB.getdb (ctx);
    db.update (STD.STD_TABLE_NAME, value, DB.STD_KEY + "=?", new String [] {
String.valueOf (STD.getId())});
}
int update (String table, ContentValues values, String whereClause, String [] whereArgs)
```

Managing Table (Selection)

```
private static String querySelectAge = "Select * from "+
Managedb.STD_TABLE_NAME + " where " + Managedb.STD_AGE +" > ?";
public static ArrayList<Student> getByAge(int age, Context ctx){
   ArrayList<Student> listSTD = new ArrayList<Student>();
   SQLiteDatabase db = ConnectionDB.getdb(ctx);
   Cursor c = db.rawQuery(querySelectAge, new String[]{String.valueOf(age)});

   while (c.moveToNext()) {
      Student STD = new Student();
      STD.setId(c.getInt(0));
      STD.setLastName(c.getString(1));
      STD.setFirstName(c.getString(2));
      STD.setAge(c.getInt(3));
      listSTD.add(STD);
   }
   c.close();

   return listSTD;
}
```

Managing Table (Selection)

Cursor c
Browse :
while (c.moveToNext ()) {
}
Method:
boolean moveToFirst () to go to the first line.
boolean moveToLast () to go to the last one.
boolean moveToPosition (int position) to go to the desired position, LASTNAMEbrows rows with the method int getCount ().
boolean moveToNext ()
boolean moveToPrevious ()

Cursor rawQuery (String sql, String [] selectionArgs)
SQL query, Arguments replacing?
Ex: "Select * from Student where age?"