

# Bases De Données SQLite

**Marouane ABAKARIM (FI)**

**08/11/2017**



## Bases de données SQLite

---

*SQLite est une base de données open source, qui supporte les fonctionnalités standards des bases de données relationnelles comme la syntaxe SQL.*

*SQLite est intégrée dans chaque appareil Android. L'utilisation d'une base de données SQLite sous Android ne nécessite pas de configuration ou d'administration de la base de données.*

- ✓ *La base de données nécessite peu de mémoire lors de l'exécution (env. 250 ko).*

*Après la création une base de données, par défaut enregistrée dans le répertoire Data : /data/APP\_NAME/databases/FILENAME.*

- ✓ *DATA : le chemin retourné par la méthode `Environment.getDataDirectory()`.*
- ✓ *APP\_NAME : le nom de votre application.*
- ✓ *FILENAME : le nom de la base de données que vous renseignez dans le code de votre application.*

## Pré-requis

---

- *Pas de connaissance particulières hors de SQL.*
- *Un savoir sur la structure d'une base de données.*
- *Capacité de faire des simple requet SQL.*

## Code source

---

- *Code Initiale :*
- *Code Finale :*

## Explications du TP

---

Normalement, Nous avons besoin d'un moyen efficace de stocker et gérer des données complexes et d'y accéder sous des applications Android.

Les bases de données pour Android sont fournies à l'aide de [SQLite](#).

Dans ce TP, nous verrons comment créer une Base de doonnées , et comment créer une table ,insérer , modifier supprimer une données dans cette base de données.

## Classe Utiliser

---

- SQLiteDatabase :
  - la connexion à la DB.
  - Création des tables.
  - Mise à jour de données.
- SQLiteOpenHelper :
  - Création d'une Base de données.

### Etape 1: Créer une Base de données :

Pour créer et mettre à jour une base de données dans une application Android, faut avoir une classe qui hérite de SQLiteOpenHelper. Dans le constructeur de la sous-classe, **faut appelez la méthode super(Context context,String name,CursorFactory factory, int version)** de SQLiteOpenHelper, avec le nom de la BD et sa version.

- Dans la sous classe, faut redéfinir cette méthodes pour créer la base de données.
- On définit la requét SQL pour la création de la Table « Contact » dans une String .

```
/* private static final String CREATE_BDD = "CREATE TABLE " + TABLE_NAME + "("  
    + COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "  
    + COL_NAME + " TEXT NOT NULL, "  
    + COL_PRENOM + " TEXT NOT NULL, "  
    + COL_TEL + " TEXT NOT NULL, "  
    + COL_EMAIL + " TEXT NOT NULL, "  
    + COL_ADDRESS + " TEXT NOT NULL, "  
    + COL_COMMENTAIRE +" TEXT NOT NULL );"; */
```

- **onCreate(SQLiteDatabase db)** : est appelée par le framework pour accéder à une base de données qui n'est pas encore créée.
- **SQLiteDatabase** : qui est la représentation Java de la base de données.

```
public void onCreate(SQLiteDatabase db) {  
  
    /* CREATE_BDD contient la définitions de la table */  
    db.execSQL(CREATE_BDD);  
  
}
```

## Etape 2: La mise à jour de la Base de données :

Dans la sous classe, faut redéfinir cette méthodes qui met à jour la base de données.

- `onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)` - est appelée lorsque la version de la base de données est augmentée. permet de mettre à jour la base de données existant ou supprimer la base de données existante et la recréer avec `onCreate()`.
- **SQLiteDatabase** : qui est la représentation Java de la base de données.

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
    String RequetMiseAJour = "DROP TABLE " + TABLE_NAME + ";";  
    db.execSQL(RequetMiseAJour);  
    onCreate(db);  
}
```

## Etape 3: Ouvrir et fermer de la base de données

- ✓ **Ouvrir en mode ecriture** : `bdd = maBaseSQLite.getWritableDatabase();`
- ✓ **Fermer la base de données** : `bdd.close();`

## Etape 3: Insérer des données :

La classe `SQLiteOpenHelper` fournit les méthodes `getReadableDatabase()` et `getWritableDatabase()` pour accéder à un objet `SQLiteDatabase` en lecture, respectif en écriture.

```
public void insertPersonne(Personne personne){  
    /* Création d'un ContentValues (fonctionne comme une HashMap) */  
    ContentValues values = new ContentValues();  
    /* on lui ajoute une valeur associé à une clé (qui est le nom de la colonne dans  
    * laquelle on veut mettre la valeur) */  
    values.put( Nom_Colonne1 , sa_Valeur1);  
    values.put( Nom_Colonne2 , sa_Valeur2);
```

```

/* on insère l'objet dans la BDD via le ContentValues */
bdd.insert(Nom_Table, null, values);
}

```

## Etape 7: Les Curseur dans la bases de données :

Réprésente le résultat d'une Requête SQLite.

```

/* passé au le première ligne */
c.moveToFirst();

/* Boucle arrête quand le curseur arrivées à le ligne après le dernier de la base */
while(!c.isAfterLast()){
/* avancé au ligne suivant */
    c.moveToNext();
}

```

## Etape 6: Récupérer Et Afficher les données :

```

public Cursor getAllData(){

    String[] Nom_Colonne = new String[] {COL_ID, COL_NAME,
    COL_PRENOM, COL_TEL,

        COL_EMAIL, COL_ADDRESS, COL_COMMENTAIRE} ;

    String Where = null ;

    String WhereArgs = null ;

    String GroupBy = null ;

    String GroupFilter = null ;

    String Having = null ;

    String OrderBy = null ;

    Cursor res = bdd.query(Nom_Table, Nom_Colonne, Where, WhereArgs ,
    GroupBy, GroupFilter, Having, OrderBy);

    Return res ;

}

```

#### Etape 4: Modification des données :

- /\* La mise à jour d'un Contact dans la BDD fonctionne plus ou moins comme une
- \* insertion
- \* il faut simple préciser quelle contact on doit mettre à jour grâce à l'ID \*/

```
public int updateById(int id, Personne personne){  
    /* Création d'un ContentValues (fonctionnement comme une HashMap) */  
    ContentValues values = new ContentValues();  
  
    /* on lui ajoute une valeur associée à une clé (qui est le nom de la colonne dans  
    laquelle on  
    * veut mettre la valeur) */  
    values.put( Nom_Colonne1 , sa_Valeur1);  
    values.put( Nom_Colonne2 , sa_Valeur2);  
    .....  
    String Where = COL_ID + "=" + id ;  
    String WhereArgs = null ;  
    return bdd.update( Nom_Table, ContentValues, Where, WhereArgs);  
}
```

#### Etape 5: Supprimer des données :

```
public void SupprimerWithId(int id){  
    /* Supprimer le contact de l'identification « id » */  
    String Where = COL_ID+"=" + id ;  
    String WhereArgs = null ;  
    bdd.delete(Nom_table, Where, WhereArgs);  
}
```

## Remarques sur SQLite

---

- ✓ créer tout les méthode d'ouvrir, fermer la base de données aussi d'insérer et modifier, supprimer dans une autre classe ,pour mieux faire la gestion de la base de données.
- ✓ mieux Créer l'identifiant de la table « ID » en AUTOINCREMENT avec l'utilisation de la contrainte de la clé primaire.

## Références

---

**SQLite** : <http://vogella.developpez.com/tutoriels/android/utilisation-base-donnees-sqlite/>

**Wiki** : <https://fr.wikipedia.org/wiki/SQLite>

## Exercice

---

Complétez le code pour qu'on peut faire des fonctionnalité suivant :

- Ajouter des Contacts.
- Visualier les Contacts.
- Supprimer des Contacts.
- Modifier des Contacts.
- Rechercher à partir un Mot-clé.