Programmation Android

Bases De Données, SQL-lite

Jérémy VINET

IE-Concept 2017

Bases De Données, SQL-lite

- Introduction
- SQL-lite
- Les Curseurs

Introduction

- Une base de données est un dispositif permettant de stocker un ensemble d'informations de manière structurée.

- L'agencement adopté pour organiser les informations s'appelle le schéma.

- L'unité de base de cette structure s'appelle la table.
  - Une table regroupe des ensembles d'informations qui sont composés de manière similaire.
  - Une entrée dans une table s'appelle un enregistrement, ou un tuple.
  - Chaque entrée est caractérisée par plusieurs renseignements distincts, appelés des champs ou attributs.

Introduction

- Par exemple, une table peut contenir le prénom, le nom et l'âge de plusieurs utilisateurs.

- Il est possible de représenter une table par un tableau, où les champs seront symbolisés par les colonnes du tableau et pour lequel chaque ligne représentera une entrée différente.

Id	Nom	Prénom	Age
1	Gates	Bill	62
2	Zuckerberg	Mark	33
3	Musk	Elon	46

Introduction

- **Une manière simple d'identifier les éléments dans une table est de leur attribuer une clé.**

- **Cette clé doit être unique, on dit qu'il s'agit d'une clé primaire.**
  - Généralement on utilise le champs id.

- **Cette clé peut aussi servir à faire le lien entre tables.**
  - Lorsque l'on utilise cette clé dans une autre table, on dit que le champ est une clé étrangère.
  - Elles garantissent la cohérence des données entre tables.

Introduction

Id	Nom	Prénom	Age	Métiers
1	Gates	Bill	62	1
2	Zuckerberg	Mark	33	1
3	Musk	Elon	46	2

Id	Nom métiers	Fonction
1	Milliardaire	Je sais pas trop
2	Excentrique	???
3	SDF	.....

SQL-lite

- Pour utiliser des bases de données sous Android, il est fortement conseillé d'utiliser SQL-lite.

- Contrairement à MySQL par exemple, SQLite ne nécessite pas de serveur pour fonctionner, ce qui signifie que son exécution se fait dans le même processus que celui de l'application.

- Par conséquent, une opération massive lancée dans la base de données aura des conséquences visibles sur les performances de votre application.

SQL-lite

- Il est donc important de maîtriser son implémentation afin de ne pas pénaliser le reste de l'application.
- SQLite a été inclus dans le cœur même d'Android, c'est pourquoi chaque application peut avoir sa propre base.
- Les bases de données sont stockées dans le répertoire DATA /data/APP_NAME/databases/FILENAME.
- Il est possible d'avoir plusieurs bases de données par application, cependant elles ne sont accessibles qu'au sein de l'application elle-même.

SQL-lite

- Afin de structurer le code Java, il est fortement conseillé de créer plusieurs classes Java, chacune ayant un rôle différent et de les placer dans un dossier différent du reste du code.

- Le fait de fragmenter son code en plusieurs classes permet une meilleure maintenabilité du code dans le temps.

- Structure conseillée :
  - Une classe contenant simplement des interfaces qui définissent les tables.
  - Une classe permettant la connexion avec la base de donnée.
  - Une classe par table, permettant d'effectuer des opérations sur la table.

SQL-lite

```
public class DatabaseAplis {

    public interface Person extends BaseColumns {
        String tableName            =   "person";
        String columnLastName       =   "last_name";
        String columnLastNameType   =   "TEXT";
        String columnFirstName      =   "first_name";
        String columnFirstNameType  =   "TEXT";
        String columnAge            =   "age";
        String columnAgeType        =   "INTEGER";
        String columnImg            =   "img";
        String columnImgType        =   "INTEGER";
        int    num_columnID         =   0;
        int    num_columnLastName   =   1;
        int    num_columnFirstName  =   2;
        int    num_columnAge        =   3;
        int    num_columnImg        =   4;

    }
}
```

- Exemple de classe contenant les interfaces de chaque tables.
- Aussi appelé définition du modèle.
- Hériter de BaseColumns permet de ne pas redéfinir le champs id.

SQL-lite

- Pour pouvoir accéder à la base de données, il faut créer une classe permettant la connexion.

- Cette classe doit hériter de SQLiteOpenHelper et redéfinir les méthodes :
  - onCreate(SQLiteDatabase db_), créait la BDD si elle n'existe pas.
  - onUpgrade(SQLiteDatabase db_, int oldVersion_, int newVersion_), permet de gérer les mises à jour de la base de données.
  - Attention, ces méthodes n'ont pas de lien avec le cycle de vie d'une activité.

- Généralement, elle contient aussi le code SQL permettant la création et la mise à jour de la BDD sous forme de constantes.

SQL-lite

```
public class DatabaseOpenHelper extends SQLiteOpenHelper {

    public static final int     databaseVersion     =   2;
    public static final String  databaseName        =   "dataBase.db";

    private static final String SQLCreateTablePerson =
            "CREATE TABLE "+ DatabaseAplis.Person.tableName + " (" +
                    DatabaseAplis.Person._ID + " INTEGER PRIMARY KEY," +
                    DatabaseAplis.Person.columnLastName + " " +
                    DatabaseAplis.Person.columnLastNameType + "," +
                    DatabaseAplis.Person.columnFirstName + " " +
                    DatabaseAplis.Person.columnFirstNameType + "," +
                    DatabaseAplis.Person.columnAge + " " +
                    DatabaseAplis.Person.columnAgeType + "," +
                    DatabaseAplis.Person.columnImg + " " +
                    DatabaseAplis.Person.columnImgType +
                    " )";

    private static final String SQLDeleteTablePerson =
            "DROP TABLE IF EXIST " + DatabaseAplis.Person.tableName;
```

SQL-lite

```
    public DatabaseOpenHelper(Context context_)
    {
        super(context_, databaseName, factory: null, databaseVersion);
    }

    @Override
    public void onCreate(SQLiteDatabase db_) {
        db_.execSQL(SQLCreateTablePerson);

    }

    @Override
    public void onUpgrade(SQLiteDatabase db_, int oldVersion_, int newVersion_) {
        db_.execSQL(SQLDeleteTablePerson);
        onCreate(db_);
    }
}
```

SQL-lite

- Une fois la classe permettant d'établir la connexion écrite, il faut écrire les classes (une par table) permettant de manipuler les tables.
- Chacune de ces classes doit avoir un accès à la classe gérant la connexion à la BDD.
- On nomme ces classes DAO (Data Access Objects).
- Elles font le lien entre des classes représentant des objets et les tables correspondant dans la BDD.

SQL-lite

```
public class DBPerson {

    private SQLiteOpenHelper m_sqlLiteHelper;

    public DBPerson(SQLiteOpenHelper sqlLiteOpenHelper_) { m_sqlLiteHelper = sqlLiteOpenHelper_; }

    public void insert(Person person_)
    {
        SQLiteDatabase db = m_sqlLiteHelper.getWritableDatabase();
        //---
        ContentValues values = new ContentValues();
        values.put(DatabaseAplis.Person.columnLastName, person_.getLastName());
        values.put(DatabaseAplis.Person.columnFirstName, person_.getFirstName());
        values.put(DatabaseAplis.Person.columnAge, person_.getAge());
        values.put(DatabaseAplis.Person.columnImg, person_.getImg());
        //---
        db.insert(DatabaseAplis.Person.tableName, nullColumnHack: null, values);
    }
```

SQL-lite

```
public void update(Person person_)
{
    SQLiteDatabase db = m_sqlLiteHelper.getWritableDatabase();
    //---
    ContentValues values = new ContentValues();
    values.put(DatabaseAplis.Person.columnLastName, person_.getLastName());
    values.put(DatabaseAplis.Person.columnFirstName, person_.getFirstName());
    values.put(DatabaseAplis.Person.columnAge, person_.getAge());
    values.put(DatabaseAplis.Person.columnImg, person_.getImg());
    //---
    String selection = DatabaseAplis.Person._ID + " = ?";
    String[] selectionArgs = {String.valueOf(person_.getId())};
    //---
    db.update(DatabaseAplis.Person.tableName, values,
            selection, selectionArgs);
}
```

SQL-lite

```
public void delete(Person person_)
{
    SQLiteDatabase db = m_sqlLiteHelper.getWritableDatabase();
    //---
    String selection = DatabaseAplis.Person._ID + " = ?";
    String[] selectionArgs = {String.valueOf(person_.getId())};
    //---
    db.delete(DatabaseAplis.Person.tableName,
            selection,
            selectionArgs);
}
```

Les curseurs

- Pour pouvoir manipuler les données retournées par les DAO, Android intègre une classe du nom de Cursor.
- Ce sont des objets qui contiennent les résultats d'une recherche dans une base de données.
- Ils fonctionnent comme des tableaux, ils contiennent les colonnes et lignes qui ont été renvoyées par la requête.

Les curseurs

- **Exemple de code permettant de retourner le contenu d'une table :**
  - Retourne un Cursor contenant la totalité de la table.

```
private Cursor getAllPersons()
{
    SQLiteDatabase db = m_sqlLiteHelper.getReadableDatabase();
    return db.query(DatabaseAplis.Person.tableName,
            columns: null, selection: null, new String[]{}, groupBy: null, having: null, orderBy: null);
}
```

Les curseurs

- **Pour parcourir les résultats d'une requête, il faut procéder ligne par ligne.**
  - boolean moveToFirst() : pour aller à la première ligne.
  - boolean moveToLast() : pour aller à la dernière ligne.
  - boolean moveToPosition(int position) : pour aller à la position voulue.
  - boolean moveToNext() : pour aller à la ligne suivante.
  - boolean moveToPrevious() : pour aller à la ligne précédente.

- **Les retours sont des booléens, true si l'opération a réussi, false si elle a échoué.**

Les curseurs

- Pour récupérer la position actuelle, il faut utiliser la méthode int getPosition() qui retourne la position courante.
- Lorsque la ligne est sélectionnée, il faut récupérer le contenu de chaque colonnes.
- Pour cela, il suffit d'utiliser une méthode du style X getX(int columnIndex).
  - long getLong(int columnIndex).
  - String getString(int columnIndex).
  - …

Les curseurs

```
public ArrayList<Person> getPersons()
{
    ArrayList<Person> persons = new ArrayList<~>();
    //---
    Cursor cursor = getAllPersons();
    //---
    cursor.moveToFirst();
    while (!cursor.isAfterLast()) {
        persons.add(new Person( cursor.getString(DatabaseAplis.Person.num_columnLastName),
                                cursor.getString(DatabaseAplis.Person.num_columnFirstName),
                                cursor.getInt(DatabaseAplis.Person.num_columnAge),
                                cursor.getInt(DatabaseAplis.Person.num_columnID),
                                cursor.getInt(DatabaseAplis.Person.num_columnImg)));
        cursor.moveToNext();
    }
    //---
    cursor.close();
    return persons;
}
```

Ressources

- https://developer.android.com/training/data-storage/room/index.html
- https://developer.android.com/reference/android/database/Cursor.html
- https://developer.android.com/