Les Bases des données avec SQLite

SQLite

- SQLite permet de créer des bases de données légères internes aux applications Android,
- 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.
- SQLite est embarquée dans tous les appareils Android.

SQLite

Création d'une base de données SQLite

- Android fournit une classe d'aide SQLiteOpenHelper permettant de gérer la création et la mise à jour des bases de données.
- Il faut créer une classe qui en hérite de la classe SQLiteOpenHelper .

SQLite

- En héritant de la classe parente SQLiteOpenHelper, la classe fille doit surcharger deux méthodes :
- la méthode onCreate. Celle-ci permet de spécifier les requêtes de création des tables de la base et une méthode mise à jour onUpgrade .

```
public class BDDAssistant extends SQLiteOpenHelper {
  private static final int VERSION_BDD = 1;
  private static final String NOM_BDD = "maBDD";

  public BDDAssistant(Context context) {
    super(context, NOM_BDD, null, VERSION_BDD);
  }

@Override
public void onCreate(SQLiteDatabase db) {
// Ajoutez votre code de création ici ...
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
// Ajoutez votre code de mise à jour ici ..
}
  }
}
```

```
class Database(context: Context) :
SQLiteOpenHelper(context,"etudiant.bd",null,1){
    override fun onCreate(db: SQLiteDatabase?) {

    }


     override fun onUpgrade(db: SQLiteDatabase?, anc_ver:
Int, nouv_ver: Int) {
    }


      }
```

SQLite

- La classe SQLiteOpenHelper fournit la méthode **getWritableDatabase** pour ouvrir la base de données. Elle retourne un objet de type SQLiteDatabase accessible en écriture permettant de modifier la base de données.
- La classe SQLiteOpenHelper fournit la méthode **getReadableDatabase** pour récupérer la base de données. (la base sera en lecture seule)

Exemple

```
writableDatabase.insert( "users", null , values)

readableDatabase.rawQuery("SELECT * FROM users",
null)
```

Traitements et requêtes SQL

- L’objet de type SQLiteDatabase récupéré dans la section précédente permet l’exécution de traitements et requêtes SQL : CREATE TABLE, DELETE, INSERT… L’exécution des requêtes SQL est réalisée par les méthodes **execSQL** et **rawQuery**.

SQLite

- execSQL(String sql) pour CREATE, ALTER, DROP qui ne retournent pas de données.
- rawQuery(String sql, ...) pour des SELECT qui retournent des enregistrements.

TYPES DE DONNÉES POUR SQLITE

- INTEGER pour les entiers
- REAL pour les nombres réels
- TEXT pour les chaînes de caractères.
- BLOB pour les données brutes (image)

- `db?.execSQL("CREATE TABLE users (num_insc INTEGER PRIMARY KEY,nom TEXT, prenom TEXT)")`

- `rawQuery("SELECT * FROM users", null)`

ContentValues(valeurs de contenu)

- ContentValues : un conteneur clé/valeur qui insère des données dans une ligne d’une table.
- Les clés correspondent aux noms de colonnes de la table et les valeurs sont les données à saisir dans la table.

ContentValues(valeurs de contenu)

```
val values = ContentValues()
values.put("num_insc",etud.num_insc)
values.put("nom",etud.nom)
values.put("prenom",etud.prenom)
```

Navigation dans les résultats

- Les curseurs sont des objets qui contiennent les résultats d'une recherche dans une base de données.
- ils contiennent les colonnes et lignes qui ont été renvoyées par la requête.

Navigation dans les résultats

- L’interface Cursor fournit toutes les méthodes permettant de naviguer dans le jeu de résultat d’une requête.
- L’accès aux enregistrements peut se faire de manière séquentielle (méthode moveToNext) ou directement en indiquant un numéro d’enregistrement (méthode moveToPosition).

Navigation dans les résultats

- À noter : il ne faut pas oublier d’invoquer la méthode moveToFirst avant le parcours séquentiel d’un lot d’enregistrements,
- la méthode rawQuery positionnant le curseur avant le premier enregistrement.