Compte rendu : Différences entre SQLite et RoomDB

Étudiante :
Balghouthi Sirine
Ouni Siwar
Smain Abir

Enseignante :
Maha Harzallah

Table des matières

1 Introduction 3

2 Présentation de SQLite 3

3 Présentation de RoomDB 3

4 Comparaison résumée 4

5 Exemple visuel 4

6 Conclusion 4

1 Introduction

Dans le développement Android, la gestion locale des données peut se faire avec deux solutions principales : **SQLite** et **Room**. **SQLite** est une base de données relationnelle intégrée dans Android, tandis que **Room** est une bibliothèque d'abstraction moderne introduite par Google pour simplifier son utilisation.

2 Présentation de SQLite

- **Définition :** SQLite est un moteur de base de données SQL intégré et léger.
- **Accès :** Manipulation directe à l'aide de commandes SQL brutes.
- **API :** Utilisation de classes comme `SQLiteDatabase`, `SQLiteOpenHelper`.

Avantages :

- Présent nativement dans Android.
- Contrôle total via les requêtes SQL.

Inconvénients :

- Verbeux et sujet aux erreurs.
- Pas de vérification de requêtes à la compilation.
- Gestion manuelle des conversions entre objets Java et données.

3 Présentation de RoomDB

- **Définition :** Room est une couche d'abstraction au-dessus de SQLite.
- **Fonctionnement :** Utilise des DAO (Data Access Object), des entités et des annotations.
- **Outils :** Génération automatique du code SQL.

Avantages :

- Plus sûr : vérification des requêtes SQL à la compilation.
- Moins de code répétitif.
- Intégration avec LiveData, Flow, RxJava.
- Gestion automatique des conversions via des *TypeConverters*.

Inconvénients :

- Légère surcharge de configuration.
- Moins de flexibilité pour des requêtes très complexes.

4 Comparaison résumée

Critère	SQLite	RoomDB
Type	API native	Bibliothèque d'abstraction
Syntaxe SQL	Requêtes manuelles	Requêtes via annotations ou DAO
Vérification à la compilation	Non	Oui
Boilerplate (Code répétitif et obligatoire)	Beaucoup de code	Réduit grâce aux DAO et annotations
Intégration Jetpack	Non	Compatible avec LiveData / ViewModel
Maintenance	Complexe	Plus simple et moderne

TABLE 1 – Comparaison entre SQLite et RoomDB

5 Exemple visuel

Avec SQLite

```
SQLiteDatabase db = dbHelper.getWritableDatabase();
Cursor cursor = db.rawQuery("SELECT * FROM user WHERE id=?", new String[]{"1"});
```

Avec Room

```
@Query("SELECT * FROM user WHERE id = :id")
User getUserById(int id);
```

6 Conclusion

Le choix entre **SQLite** et **Room** dépend du besoin et du contexte de l'application :

- **Room** est aujourd'hui la solution recommandée pour la plupart des projets Android modernes :
  - Elle réduit considérablement le code répétitif.
  - Elle offre une meilleure sécurité et maintenabilité grâce à la vérification à la compilation.

  - Elle s'intègre parfaitement dans les architectures Jetpack (LiveData, ViewModel, etc.).
- **SQLite**, quant à lui, reste une option intéressante pour :
  - Les petits projets simples.
  - Les développeurs ayant besoin d'un contrôle total sur les requêtes.
  - Des cas d'usage spécifiques où Room ne suffit pas.

En résumé, Room offre un surcroit de productivité et de robustesse qui justifie son adoption dans tout projet Android sérieux. Cependant, connaître SQLite reste essentiel pour comprendre les bases de la persistance locale dans Android.