**Annexe 12 SQLite**

**Appels répétés**

**SQLite**

SqLite est intégrée au moteur d’exécution d’Android.

Il s’agit d’un SGBD très apprécié car il fournit une interface SQL tout en nécessitant un minimum de mémoire et en fournissant une rapidité satisfaisante.

**SQLite vs Oracle**

Similitudes :

* Commandes courantes : INSERT, CREATE TABLE, SELECT

Différences :

* Typage des données : SQLite permet de stocker une valeur de n’importe quel type dans n’importe quelle colonne quel que soit son type déclaré.
* On ne peut pas faire de contraintes de clé étrangères, de jointures externes

\*\*\*Android ne fournit aucune base de données au départ 🡪 on doit donc créer la base de données et les tables.

**Types de données dans SQLite**

|  |  |  |  |
| --- | --- | --- | --- |
| INTEGER | REAL | TEXT | NULL |

\*\* boolean : on utilise integer 0 ou 1

\*\* index : INTEGER PRIMARY KEY AUTOINCREMENT ( doit s'appeler \_id )

**Exercice** : Construire une base de données incluant une table Inventeur regroupant les informations suivantes sur certains grands inventeurs de l’Histoire dans le but de l’utiliser dans un quiz.

|  |  |  |  |
| --- | --- | --- | --- |
| Nom | Origine | Invention | Annee |
| Laszlo Biro | Hongrie | Stylo à bille | 1938 |
| Benjamin Franklin | Etats-Unis | Paratonnerre | 1752 |
| Mary Anderson | Etats-Unis | Essuie-glace | 1903 |
| Grace Hopper | Etats-Unis | Compilateur | 1952 |
| Benoit Rouquayrot | France | Scaphandre | 1864 |

**Marche à suivre**

1. Créer une classe Inventeur qui permettra de modéliser nos données
2. Créer notre classe de gestion de tout ce qui a rapport à la BD, sous-classe de SQLiteOpenHelper (permet de créer nos tables et les opérations sur elles )
3. Travailler avec la base de données via nos activités

**Étape 1 : Créer une classe-modèle pour représenter les différents inventeurs**

**Étape 2 : sous-classe de SQLiteOpenHelper ( singleton )**

**Qu’est-ce qu’un singleton ? Voir présentation**

En programmation orientée objet, une classe java singleton est une classe qui ne peut avoir qu'un seul objet (une instance de la classe) à la fois

L'objectif principal d'une classe Java Singleton est de limiter la limite du nombre de créations d'objets à une seule. Cela garantit souvent qu'il existe un contrôle d'accès aux ressources, par exemple, une connexion à un socket ou à une base de données.

* Constructeur :
  + Contexte : on l’assignera par getApplicationContext dans l’activité
  + Nom : nom de la base de données( et pas le nom de la table )
  + Factory : Fabrique de curseur, avec un curseur pn va se promener dans notre ensemble de resultat. Curseur particulier, afin d’accomplir des taches qui me semble complexe
  + Version : version de notre base de données. Dans l’exemple nous avons ecrit 1
* Méthode onCreate ( SQLiteDatabase db )
* A partir du paramètre, on peut créer la / les tables et y ajouter des enregistrements initiaux s’il y a lieu
* IMPORTANT : cette méthode n’est appelée qu’une fois, lors de l’installation de l’app
* Méthode onUpgrade ( SQLiteDatabase db, int oldVersion, int newVersion )

Cliquez ou appuyez ici pour entrer du texte.

Exercice : Créer cette sous-classe, coder la méthode onCreate en y créant la table

|  |  |  |
| --- | --- | --- |
| Méthodes importantes de la classe SQLiteDatabase | | |
| execSQL | Pour les énoncés autres qu’un insert/select  ( create table ) |  |
| insert | Pr. Insérer un enregistrement  3 paramètres :  Nom de la table  Null  ContentValues |  |
|  |  |  |
| Objet ContentValues | Fonctionne un peu comme une Hashtable(cle – valeur) |  |
|  |  |  |

Exercice : Coder des méthodes pour :

* Ouvrir une instance de la base de données ( l’appeler dans le constructeur )
* Fermer l’accès à la base de données

**Pour vous aider :**

**Petit lexique non-exhaustif des équivalences JDBC / SQLite**

|  |  |  |
| --- | --- | --- |
| Opérations sur la table | Méthode sur l’objet SQLiteDatabase | JDBC |
| insert | Insert (les données à insérer doivent être englobées dans un objet ContentValues ) |  |
| delete | delete |  |
| update | update |  |
| Select | rawQuery / query  retourne un Cursor | createStatement / prepareStatement  retourne un ResultSet |
| Autres commandes qui ne retournent pas de résultats ( create table, etc. ) | execSQL |  |

**Méthodes à utiliser sur un Cursor**

|  |  |  |
| --- | --- | --- |
| Opérations | Méthode sur le Cursor | Équivalent JDBC ( ResultSet ) |
| Placer le curseur à la première ligne de résultats | moveToFirst() | next() |
| Passer à la ligne suivante | moveToNext() | next() |
| Tant qu'il y a un résultat | while ( ! curseur.isAfterLast())  ou **while ( curseur.moveToNext())** | while ( !rs.next()) |
| Trouver la valeur int à un certain champ | getInt(numéro du champ dans l’ensemble de résultats ( commence à 0 ) ) | getInt( numéro du champ ) |

Exercices :

* Faire une requête permettant de retourner le nom de toutes les inventions de la table
* Faire une requête permettant de retourner si le nom d’un inventeur et l’invention fournis en paramètre sont une bonne association ou non

**Étape 3. Travailler avec la base de données via nos activités**

**exercice : Annexe 12B**