SQLite, Introduction

- Définition

SQLite est un moteur de bases de données libre qui implémente la plus part des fonctionnalités du SQL standard. Il utilise donc la plus part des fonctionnalités de SGBDR

C’est le SGBDR embarqué le plus utilisé au monde, il ne nécessite aucune configuration, ni serveur pour fonctionner. Entièrement écrit en C, ce qui le rend très performant.

SQLite est également transactionnel, c'est-à-dire qu'il respecte les principes ACID(opération atomique, Consistance des données, indépendant des processus et durable → longévité des données).

Évidemment, il est public d’où sa popularité.

SQLite, Introduction

Avantages:

- Performant
- Ne nécessite pas de serveur de base de données. Les données tiennent dans un fichier.
- Utilise la plus part des commandes SQL
- Peut être utilisé comme Bd personnelle ou commerciale (selon la taille)
- Il est embarqué.

Inconvénients

- Il est embarqué, donc léger ce qui veut dire que vous pouvez oublier les procédures et fonctions.
- Si votre BD est assez volumineuse, il vaut mieux utiliser un autre SGBD. Vous ne pouvez pas faire des jointures à ne plus en finir.

SQLite, Introduction

Installation:

Vous pouvez télécharger SQLite à l'adresse : https://sqlite.org/download.html

Le fichier est sqlite-dll-win64-x64-3140200.zip (pour Windows 7 et 10).

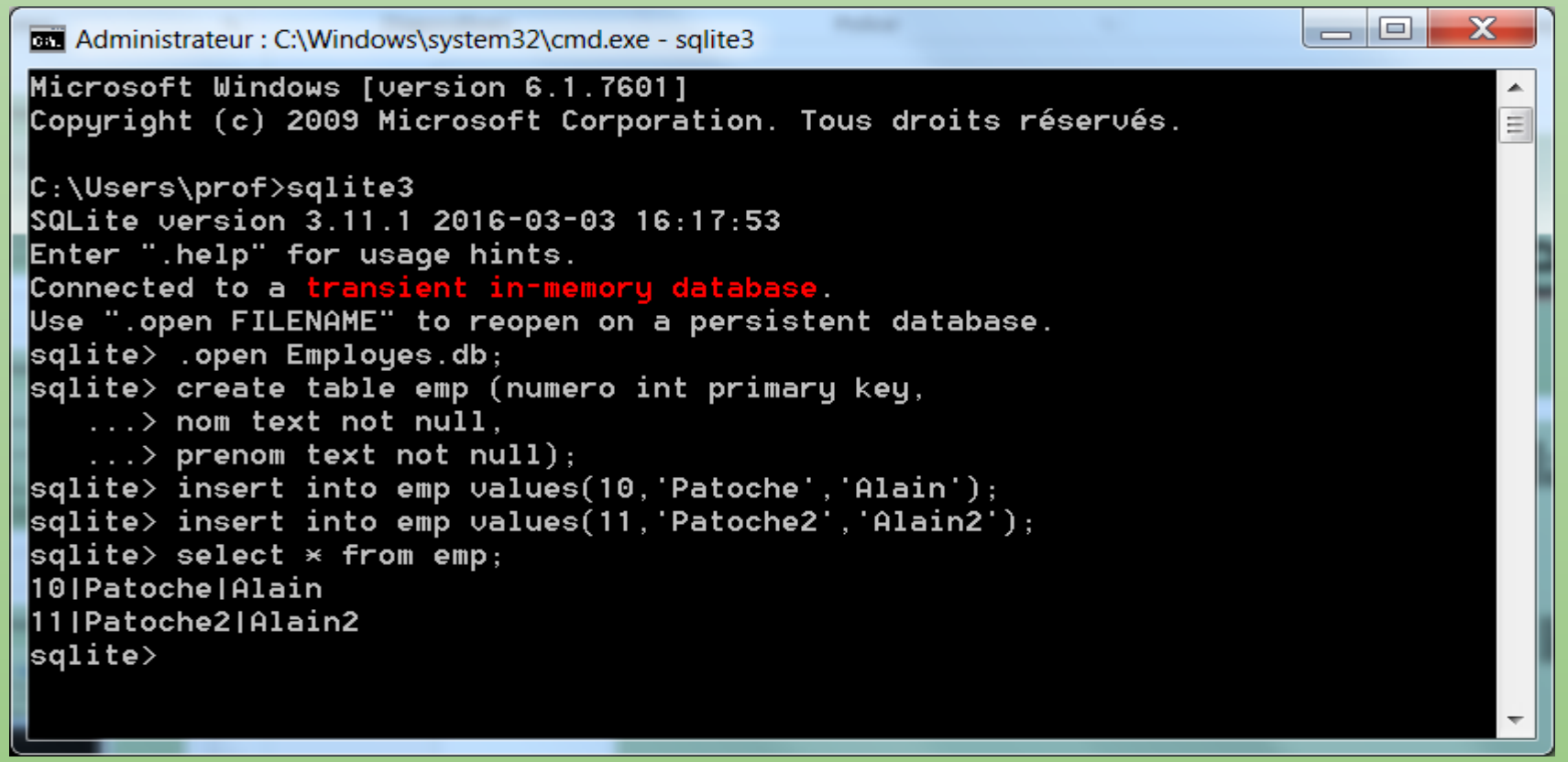
Une fois dé-zipper , le dossier va contenir un fichier sqlite3.dll, qui est votre sqlite.

Lorsque vous lancez votre sqlite en mode ligne des commande en tapant sqlite3, vous allez avoir :

Pour créer la bd à un autre endroit (dossier) que celui par défaut

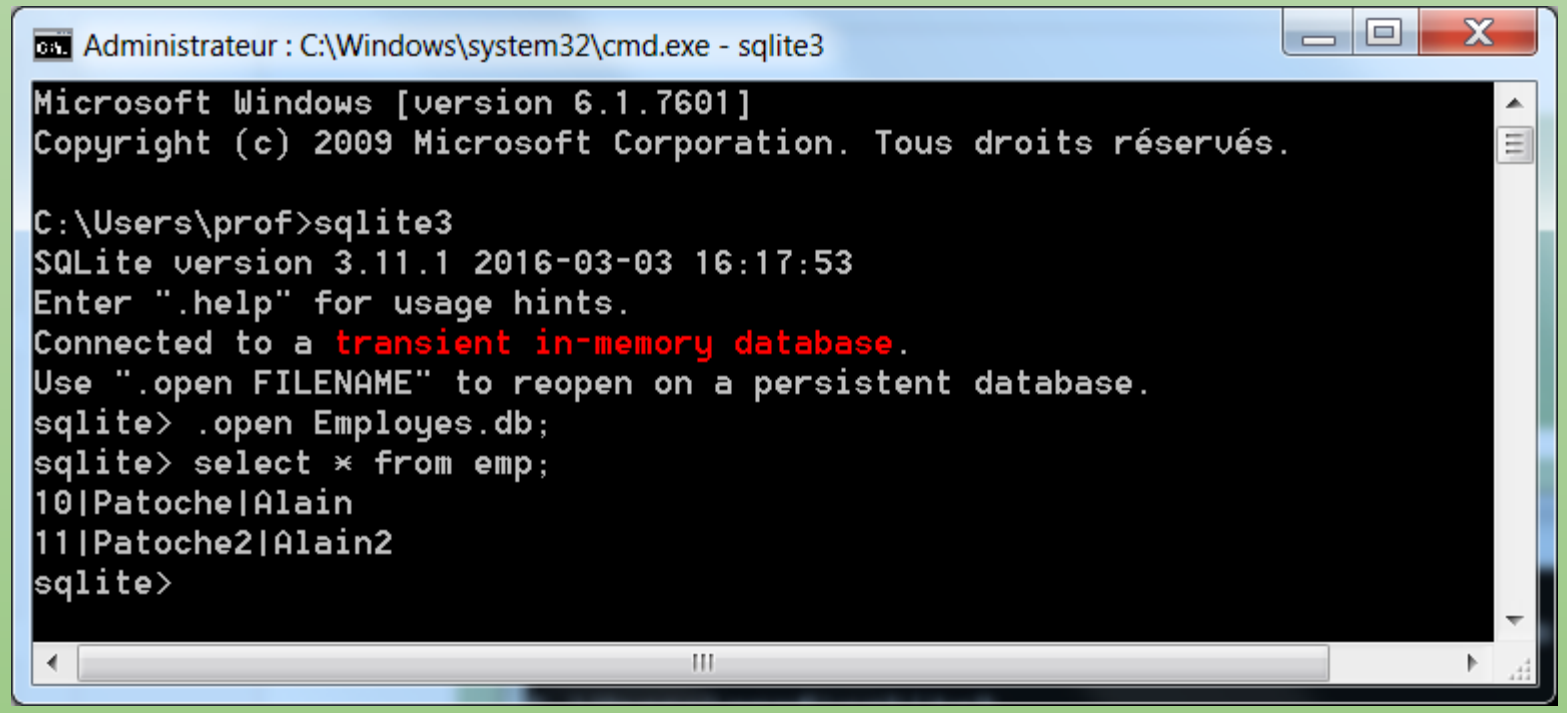
.open c:/GestionJoueurs/Joueurs.db",

SQLite, Introduction



SQLite, Introduction

On peut retrouver la bd que nous avons créée tout



SQLite, Introduction

On peut également utiliser SQLiteBrowser ou DB Browser For Sqlite qui offre une belle interface graphique pour créer et gérer votre base de données. (démonstration plus tard)

Vous pouvez également utiliser un langage de programmation pour créer directement la BD, les table et en faire la gestion.

Exemple avec Java.

En Java, on utilise le pilote JDBC: org.sqlite.JDBC, avec la méthode forName de la classe Class.

```
Class.forName("org.sqlite.JDBC");
```

SQLite, Introduction

Vous pouvez accéder à la base de donnée existante ou en créer une nouvelle. Il suffit d'utiliser la calsse DriverManager et la méthode getConnection.

Dans le code qui suit, le programme va créer la base de données BdChats, si celle-ci n'existe pas.

Une fois la BD créée, vous pouvez créer les tables et etc..

SQLite, Introduction

```
try {
    Class.forName("org.sqlite.JDBC");

    System.out.println("Pilote Chargé");
     }
catch ( ClassNotFoundException e )
    {
    System.out.println(e.getMessage());
    System.exit(0);
    }
try {
   conn = DriverManager.getConnection("jdbc:sqlite:BdChats");
   System.out.println(Bd Crée ou Connexion ouverte");
}
catch (SQLException sed )
{
    System.out.println(sed.getMessage());
    System.exit(0);
}
```

SQLite, Introduction

```
String sqlc ="create table minou (nom text, race text)";
 String sqli ="insert into  minou values ('remi','abyssin')";
String sql2 = "Select race from minou where nom ='remi'";
//Charger le driver puis créer la bd
 try{
       Statement   stmc = conn.createStatement();
      stmc.executeUpdate(sqlc);
      Statement stmi = conn.createStatement();
      stmi.executeUpdate(sqli);
     stm = conn.createStatement();
     rst = stm.executeQuery(sql2);
      if (rst.next())
        {
  System.out.println("la race est" + "   "+ rst.getString(1) );
       }

   }
```

SQLite, Introduction

Types de données supportés par SQLite.

SQLite ne supporte que 4 types de données:

Type	Définition
NULL	Valeur vide
INTEGER	Entier signé
REAL	Nombre réel
TEXT	Champ texte
BLOB	Champ binaire (image)

Mais pour être compatible avec les autres SGBD, SQLite prend en charge le concept Affinity; ce qui veut dire que si vous déclaré une colonne d'un type qui n'est pas SQLite, alors SQLite lui associe un type parmi ceux qui sont dans le tableau

SQLite, Introduction

L'affinité d'une colonne est déterminée par le type déclaré de la colonne, selon les règles suivantes dans l'ordre indiqué:

R1: Si le type déclaré contient la chaine INT alors celui-ci est associé à un INTEGER

R2: Si le type déclaré de la colonne contient l'une des chaînes "CHAR", "CLOB", ou "TEXT", alors le type est associé à un TEXT (VARCHAR contient la chaine CHAR).

R3: Si le type déclaré contient la chaine BLOB alors celui-ci est associé à un BLOB

R4: Si le type déclaré pour une colonne contient l'une des chaînes "REAL", "Floa", ou "DOUB", alors il le type est associé à un REAL

R5: dans tous les autre cas, le type est associé à NUMERIC.

Source : http://www.sqlite.org/datatype3.html

SQLite, Introduction

Example Typenames From The CREATE TABLE Statement or CAST Expression	Resulting Affinity	Rule Used To Determine Affinity
INT INTEGER TINYINT SMALLINT MEDIUMINT BIGINT UNSIGNED BIG INT INT2 INT8	INTEGER	1
CHARACTER(20) VARCHAR(255) VARYING CHARACTER(255) NCHAR(55) NATIVE CHARACTER(70) NVARCHAR(100) TEXT CLOB	TEXT	2
BLOB no datatype specified	BLOB	3
REAL DOUBLE DOUBLE PRECISION FLOAT	REAL	4
NUMERIC DECIMAL(10,5) BOOLEAN DATE DATETIME	NUMERIC	5

SQLite, Introduction

Lorsque la base de données SQLite est embarquée (dans un appareil mobile) alors, vous l'avez compris, aucune installation supplémentaire n'est requise. Pour exploiter un BD SQLite, deux méthodes s'offrent à nous:

- Une qui utilise surtout la classe **SQLiteDatabase (proche de la bd)** et celle qui utilise la classe **SQLiteOpenHelper et SQLiteDatabase (proche de la programmation.).**
- Nous allons commencer par détailler la méthode qui utilise SQLiteDataBase.
- Cette classe est contenue dans le package:

android.database.sqlite.SQLiteDatabase;

SQLite, Introduction

La classe **SQLiteDatabase: Méthodes minimales**

(https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html

Méthodes	**Rôles**
execSQL(String sql, Object[] bindArgs)	Execute une instruction SQL qui n'est pas un SELECT (void)
rawQuery(String sql, String[] selectionArgs)	Execute une requête SQL de type SELECT et retourne un ensemble de résultat dans un curseur : CURSOR
openOrCreateDatabase(String name, int mode, CursorFactory factory)	Retourne un SQLiteDatabe. Les paramètres sont : nom pour la base de données. Il Peut être null. Le Context Mode, ici (Context. MODE_PRIVATE) qui veut dire que seule votre activité a droit d'y accéder. C'est le mode par défaut

SQLite, Introduction

Exemples: pour créer la base de données bd

bd = openOrCreateDatabase("GestionJoueurs", Context.MODE_PRIVATE,null);

Après execution, nous aurons créé notre base de données bd.

Pour creer les tables, il suffit d'appeler la méthode execSQL de la classe SQLiteDatabe.

Exemple :

bd.execSQL("CREATE TABLE IF NOT EXISTS Joueurs(num integer primary key autoincrement, nom VARCHAR,prenom VARCHAR);");

bd.execSQL("INSERT INTO joueurs(nom,prenom) VALUES('Patoche','Alain');");

SQLite, Introduction

La méthode rawQuery(String sql, String[] selectionArgs), cette méthode retourne un curseur après qu'une requête de type SELECT soit passée.

Exemple:

```
Cursor c=bd.rawQuery("SELECT * FROM Joueurs",null);
```

Ou encore

```
Cursor c = bd.rawQuery(" SELECT nom, prenom FROM joueurs  where salaire > ?", new String[]{"1000"});
```

→ la liste d'arguments est 1, ce qui veut dire qu'on ramène tous les noms et prénom dont le salaire est > 1000.

```
Cursor c  =rawQuery("SELECT  nom, prenom FROM joueurs WHERE codeEquipe = ? and salaire > ?", new String[] {« MTL", "2000"});
```

SQLite, Introduction

Récupérer le contenu du curseur.

Pour récupérer le contenu du curseur on utilise les méthode suivantes:

moveToFirst () , se place au début du curseur

moveToNext (), parcours en avant

moveToPrevious() parcours en arrière

moveToLast() se place à la fin du curseur

La méthode **isAfterLast()** permet de vérifier si la fin du résultat de la requête a été atteint.

getCount() permet d'obtenir le nombre d'enregistrements de la requête.

Cursor fournit les méthodes get * (), par exemple getLong (columnIndex), getString (columnIndex) pour accéder aux données de l'enregistrement courant. Le "columnIndex" est l'index de la colonne à accéder.

Un curseur doit être fermé avec la méthode **close().**

SQLite, Introduction

```
//----------DébutAfficher TOUT
    if(view==btnAffichertout)
    {
       Cursor c=bd.rawQuery("SELECT * FROM Joueurs",null);
            if(c.getCount()==0)
                {
            AfficheMessage("Erreur", "Aucune donnée");
             return;
                }
       StringBuffer buffer=new StringBuffer();
       while(c.moveToNext())
        {
          buffer.append("Numéro: "+c.getInt(0)+"\n");
          buffer.append("Non: "+c.getString(1)+"\n");
          buffer.append("Prénom: "+c.getString(2)+"\n");
        }
       c.close();
       AfficheMessage("Les joueurs", buffer.toString());
    }
```

SQLite, Introduction

```
public void AfficheMessage(String titre,String message)
 {
       Builder builder=new Builder(this);
       builder.setCancelable(true);
      builder.setTitle(titre);
       builder.setMessage(message);
       builder.show();
   }
```

SQLite, Introduction

```
// Bouton afficher le premier
            if(view==btnPremier) {
          try{
          c1 = bd.rawQuery("SELECT * FROM Joueurs", null);
          if (c1.getCount() == 0)
                {
             AfficheMessage("Erreur", "Aucune donnée");
             return;
                 }
       if(c1.moveToFirst())
             {
        edtNom.setText(c1.getString(1));
        dtPrenom.setText(c1.getString(2));
             }
       } // fin du try
     catch(Exception se)  {
      Toast.makeText(MainActivity.this, "message"+ se.getMessage().toString(), Toast.LENGTH_LONG).show();
                                  }
 } // Fin du premier
```

SQLite, Introduction

```
// bouton suivant :
        if(view ==btnSuivant)
    {
          if(c1.moveToNext())
          edtNom.setText(c1.getString(1));
          edtPrenom.setText(c1.getString(2));
       }
 if(view ==btnPrecdent)
        {
          if(c1.moveToPrevious())
             edtNom.setText(c1.getString(1));
          edtPrenom.setText(c1.getString(2));
       }
```