Préparation SQL

TABLE DES MATIÈRES

1.1 - INTRODUCTION

1.1.1 - Présentation du document

Ce document est une préparation à un cours SQL Initiation qui lui même sera suivi d'un cours SQL Avancé. C'est une pré-initiation !

L'accent est mis sur le langage SQL de base.

L'architecture Client/Serveur n'est pas abordée.

Les seules connaissances informatiques requises sont de connaître Windows ou Linux en mode graphique ou Mac OS, plus exactement la manipulation d'un explorateur de fichiers.

Pratique de ce document.

Vous devez :

créer un dossier,
copier dedans le fichier cours.db,
télécharger et installer SQLite (cf 1.2.2.),
lancer SQLite en spécifiant le chemin et le nom de la BD.

1.1.2 - Gérer votre vidéothèque

Vos films … dans un tableau d'un document de traitement de textes (MS-Word / OpenOffice Writer) ou dans une feuille de calcul d'un tableur (MS Excel / OpenOffice Calc).

Titre	Acteurs	Réalisateurs	Support	Année de production	Titre original	Genre
Collision	Sandra Bullock, Don Cheadle, Matt Dillon	Paul Haggis	DD	2004	Crash	Drame
21 grammes	Sean Penn, Benicio del Toro, Naomi Watts, Charlotte Gainsbourg	Inarritu	Cassette	2003	21 Grams	Drame
Nicotina	Diego Luna, Daniel Giménez-Cacho, Lucas Crespi	Hugo Rodriguez	DVD	2004	Nicotina	Comédie

Vous pouvez imaginer les autres données utiles : synopsis, pays producteurs, durée, scénariste, photo(s), bande(s) annonce, musique, ...

La problématique :

Comment lister tous les films avec Charlotte Gainsbourg ?
Comment lister tous les films avec Sean Penn ?
Comment lister tous les films produits par la France ou les Etats-Unis ou les deux ?

Solution : pour cela il faut «exploser» les données en plus petits «paquets» d'informations – plusieurs tableaux en fait - reliés entre eux.

La première solution serait de créer plusieurs tableaux dans le document texte ou plusieurs feuilles de calcul dans un tableur. Mais cette solution n'est pas viable car la liaison entre les tableaux nécessiterait une programmation difficile et peu fiable.

De plus, et là on passe au niveau professionnel, si l'on veut partager les données entre plusieurs utilisateurs, éventuellement de façon simultanée, il faut un outil qui le permette.

Il faut un logiciel d'une autre catégorie : un Système de Gestion de Bases de Données (MS-Access, OpenOffice Base, MySQL, Oracle, etc, …, MongoDB, Cassandra, etc).

1.1.3 - L'exigence d'un SGBDR

Un SGBDR s'impose !

Quelques notions

Notion	Description
SGBDR	Système de Gestion de Bases de Données Relationnelles. Logiciel qui permet de gérer une Base de Données Relationnelle. MySQL est un SGBDR. Les principaux objets gérés par un SGBDR sont les bases de données, les utilisateurs, les tables, les index, … Le R de SGBDR provient de l'Algèbre **R**elationnelle de CODD et pas des relations qui peuvent exister entre les tables.
Tables	Ensemble de colonnes et de lignes, un tableau. C'est dans une table que sont stockées les données. Une ligne représente un objet du monde réel. Pensez à la table acteurs ou à la table films.
Colonnes	Une donnée élémentaire dans la structure des objets à stocker dans la table. Pensez à la colonne nom_acteur ou à la colonne titre_film.
Types de colonne	Chaque colonne est d'un type particulier. Les principaux types sont les suivants : numériques, textes, date, … Chaque type contient des sous-types. INTEGER, INT, DECIMAL, FLOAT, DOUBLE ..., pour les types numériques. VARCHAR, CHAR, TEXT, …, pour les types textes. DATE, DATETIME, TIME, TIMESTAMP pour les types date. La colonne nom_acteur est de type VARCHAR qui est un sous-type de Texte. La date de production du film est de type DATE.
Clé primaire	Colonne (parfois plusieurs) qui permet d'identifier – de rendre unique – une ligne d'une table. Pensez au n° de sécurité sociale, au code commune, au code d'un produit, etc.
Clé étrangère	Une colonne (parfois plusieurs) qui correspond à une clé primaire dans une autre table (sa référence). Le genre d'un film (comédie, drame, …) doit exister dans la table genres avant de pouvoir être saisi dans la table films.
Index	Un accélérateur pour accéder plus rapidement à une information.
SQL	Structured Query Langage. Le langage standardisé pour la gestion des BDR.

Notre objectif est de présenter les notions de base qui concernent principalement les tables.

1.1.4 - SQL

SQL = Strutured Query Language.

SQL est une implémentation de l'Algèbre Relationnelle de CODD.

C'est le langage standardisé (à l'ANSI) pour la gestion des BDR.

SQL se décompose en 4 sous-langages :

Sous-langage	Description
LDD	Langage de définition des données Permet de gérer les objets Trois verbes : CREATE (création), ALTER (modification), DROP (suppression)
LMD	Langage de manipulation des données Permet de gérer les données des tables Quatre verbes : SELECT (extraction), INSERT (ajout), DELETE (suppression), UPDATE (modification)
LCD	Langage de contrôle des données Permet de gérer l'accès aux données Deux verbes : GRANT et REVOKE
LCT	Langage de contrôle des transaction Permet de gérer les transactions (mises à jour) Deux verbes : COMMIT et ROLLBACK

1.2 - SQLite

1.2.1 - Présentation

SQLite est un SGBDR léger.
Il est principalement utilisé dans les systèmes embarqués.

SQLite est sponsorisé par le SQLite Consortium qui a comme membres Oracle, Nokia, Mozilla, Adobe, ...

1.2.2 - Installation de SQLite

1.2.2.1 - Sous Linux

```
$ sudo apt-get install sqlite3
```

1.2.2.2 - Sous Windows

Téléchargement : https://www.sqlite.org/download.html

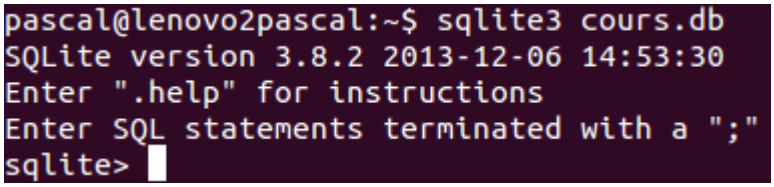
Vous dézippez le .zip téléchargé (sqlite-shell-win32-x86-3090200.zip pour du X86, etc).

Le résultat est l'exécutable sqlite3.exe.

1.3 - Quelques commandes de SQLite

Lancer SQLite :

cmd>sqlite3



Quitter la console sqlite :

sqlite>.exit

Aide :

sqlite>.help

Quelques commandes SQLite (hors SQL standard et sans ; à la fin) :

.help	L'aide
.tables	Affiche la liste des tables de la BD
.schema nomDeTable	Affiche l'ordre CREATE de la table
.open nomDeBD.db	Ouvre une BD
.databases	Affiche la liste de BD attachées
.quit	Comme .exit

1.4 - Création/Ouverture d'une Base de Données

Cette commande crée une base de données si elle n'existe pas ou ouvre une base de données si elle existe.
Une base de données SQLite est donc un unique fichier d'extension .db.
L'extension doit être précisée.

Syntaxe

```
cmd>sqlite3 nomDeLaBD.db
```

Exemple

```
sqlite3 cours.db
```

1.5 - Suppression d'une base de Données

Syntaxes

```
cmd>rm nomDeLaBD.db
```

```
cmd>del nomDeLaBD.db
```

Exemples

```
$ rm cours.db
```

```
cmd>del cours.db
```

1.6 - Les types de SQLITE

TEXT,
INT ou INTEGER,
INTEGER PRIMARY KEY,
NUM ou NUMERIC,
REAL,
BLOB,
NONE, ...

```
CREATE TABLE exemple1(
  a VARCHAR(10),
  b NVARCHAR(15),
  c TEXT,
  d INTEGER,
  e FLOAT,
  f BOOLEAN,
  g CLOB,
  h BLOB,
  i TIMESTAMP,
  j NUMERIC(10,5)
  k VARYING CHARACTER(24),
  l NATIONAL VARYING CHARACTER(16)
);
```

1.7 - Création d'une table

Définition

Une table est un ensemble de colonnes et de lignes.
Les colonnes sont typées.
Une table doit posséder une clé primaire sur une ou plusieurs colonnes.
La clé primaire identifie la ligne dans la table.
Les colonnes peuvent subir des contraintes.
Une table peut avoir zéro, une ou plusieurs clés étrangères (cf plus loin).

Syntaxe

```
CREATE TABLE [IF NOT EXISTS] [bd.]nomDeTable(clePrimaire TYPE,
col1 TYPE, col2 TYPE);
```

Exemple

```
sqlite>CREATE TABLE pays(
id_pays INTEGER PRIMARY KEY AUTOINCREMENT,
code_pays VARCHAR(5) NOT NULL UNIQUE,
nom_pays VARCHAR(50) NOT NULL UNIQUE
);
```

Quelques contraintes de colonnes

Contrainte	Description
PRIMARY KEY	Clé primaire donc UNIQUE, NOT NULL et indexée
NOT NULL	NOT NULL : la valeur doit être renseignée
UNIQUE	Les valeurs de la colonne sont uniques
CHECK(expression)	La valeur doit être validée par l'expression. Salaire > 1000 par exemple
COLLATE nom_de_collation	Nom de la collation (utilisée pour les tris) : BINARY, NOCASE, RTRIM
DEFAULT	Valeur par défaut

1.8 - Afficher la structure d'une table

Syntaxes

Affiche la structure de la table

```
sqlite>PRAGMA table_info(nomDeTable);
```

Affiche l'ordre SQL CREATE correspondant à la table

```
sqlite>.schema nomDeTable;
```

Exemples

```
sqlite>PRAGMA table_info(pays);
```

0|id_pays|INTEGER|0||1
1|code_pays|VARCHAR(5)|1||0
2|nom_pays|VARCHAR(50)|1||0

```
sqlite>.schema pays;
```

CREATE TABLE pays(
id_pays INTEGER PRIMARY KEY AUTOINCREMENT,
code_pays VARCHAR(5) NOT NULL UNIQUE,
nom_pays VARCHAR(50) NOT NULL UNIQUE
);

1.9 - Lister les tables d'une BD

```
sqlite>.tables
```

sqlite> .tables
pays villes

1.10 - Suppression d'une table

Syntaxe

```
DROP TABLE nomDeTable;
```

Exemple

```
DROP TABLE pays;
```

1.11 - Ajout d'un enregistrement

Syntaxe

```
INSERT INTO nomDeTable(liste de colonnes)
VALUES(liste de valeurs);
```

Notes

Les valeurs de type texte (VARCHAR, NVARCHAR, TEXT, VARYING CHARACTER, NATIONAL VARYING CHARACTER) sont entre ' (simple quote).
Les valeurs de type numérique (INTEGER, FLOAT, NUMERIC) sont sans ' (simple quote).

Exemple

```
sqlite>INSERT INTO pays(code_pays, nom_pays) VALUES('FR', 'France');
```

1.12 - Affichage de tous les enregistrements d'une table

Syntaxe

```
sqlite>SELECT * FROM nomDeTable;
```

Exemple

```
sqlite>SELECT * FROM pays;
```

1|FR|France
2|IT|Italie
4|ES|ESPAGNE

Note : si vous ne voulez afficher que quelques colonnes vous remplacez * par une liste de colonnes, chaque colonne est séparée par une virgule.

```
sqlite>SELECT nom_pays, code_pays FROM pays;
```

1.13 - Affichage de quelques les enregistrements d'une table

1.13.1 - Syntaxe et premier exemple

Syntaxe

```
sqlite>SELECT * FROM nomDeTable WHERE condition;
```

La clause WHERE permet de filtrer des lignes de la table.

Exemple

```
sqlite>SELECT * FROM pays WHERE id_pays = 1;
```

1|FR|France

1.13.2 - Les opérateurs de comparaison

OPÉRATEUR	DESCRIPTION
=	Égal
!= , <>	Différent de
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à

1.13.3 - Les opérateurs logiques

Opérateur	Description
AND	Et logique
OR	Ou logique
NOT	La négation logique

Rappel sur les opérateurs logiques : table de vérités

C1	C2	AND	OR	NOT C1	XOR
Vrai	Vrai	Vrai	Vrai	Faux	Faux
Vrai	Faux	Faux	Vrai	Faux	Vrai
Faux	Vrai	Faux	Vrai	Vrai	Vrai
Faux	Faux	Faux	Faux	Vrai	Vrai

AND : Le résultat est VRAI si les opérandes C1 et C2 sont VRAI.
OR : Le résultat est VRAI si un des opérandes C1 et C2 est VRAI.
NOT : Le résultat est VRAI l'opérande C1 FAUX.
XOR : Le résultat est VRAI si un et un seul des opérandes C1 et C2 est VRAI.

Exemples :

AND : pour être authentifié le pseudo et mot de passe doivent être valides.
OR : pour afficher les produits des catégories Eaux et Sodas il faut utiliser l'opérateur OR.
NOT : pour afficher les villes hors de France il faut utiliser l'opérateur NOT.
XOR : n'existe pas en SQL. Il faudra utiliser la clause exists et des requêtes ensemblistes.

1.13.4 - Les opérateurs ensemblistes

Opérateur	Descripteur
[NOT] IN(V1, V2, …)	Egal à n'importe quelle valeur d'une liste de valeurs
[NOT] BETWEEN x AND y	x >= valeur <= y
[NOT] LIKE	Comparer deux chaînes de caractères avec l'utilisation des caractères génériques : _ pour un caractère et % pour une chaîne de caractères
IS [NOT] NULL	Tester la valeur NULL (ou non) dans une colonne Le NULL est la valeur dite indéterminée indépendamment du type

1.13.5 - Opérateurs divers

Concaténation : ||

Exemple :

```
SELECT id_pays || ' : ' || nom_pays FROM pays;
```

1.13.6 - Autres exemples

Les pays dont l'id est supérieur à 1

```
sqlite>SELECT * FROM pays WHERE id_pays > 1;
```

Les pays dont l'id est égal à 1 ou à 4

```
sqlite>SELECT * FROM pays WHERE id_pays = 1 OR id_pays = 4;
```

Les pays dont l'id est égal à 1 ou à 4

```
sqlite>SELECT * FROM pays WHERE id_pays IN(1,4);
```

Les pays dont l'id est compris entre 1 et 3

```
sqlite>SELECT * FROM pays WHERE id_pays BETWEEN 1 AND 3;
```

Les pays dont l'id n'est pas égal à 4

```
sqlite>SELECT * FROM pays WHERE NOT id_pays = 4;
```

Les pays dont le nom commence par 'fr'

```
sqlite>SELECT * FROM pays WHERE nom_pays LIKE 'fr%';
```

1.14 - Suppression d'un enregistrement

Syntaxe

```
sqlite>DELETE FROM nomDeTABLE [WHERE condition];
```

Exemple

```
sqlite>DELETE FROM pays WHERE nom_pays = 'Espagne';
```

Note : sans la clause WHERE tous les enregistrements de la table sont supprimés.

1.15 - Modification d'un enregistrement

Syntaxe

```
sqlite>
UPDATE pays
SET nom_de_colonne = valeur [nom_de_colonne = valeur]
[WHERE condition];
```

Exemple

```
sqlite>UPDATE pays SET nom_pays = 'ESPAGNE' WHERE id_pays = 4;
```

1.16 - Création d'une clé étrangère

Définition

Une clé étrangère (Foreign Key) est une contrainte d'intégrité référentielle. Cela signifie qu'un enregistrement ne peut exister dans une table que si il existe un enregistrement correspondant dans une autre table. Par exemple Pays et villes, produits et commandes, etc.

La création d'une clé étrangère doit être réalisée dans l'ordre de création de la table.

Syntaxe

```
CREATE TABLE nomDeTable(
colonnePK INTEGER PRIMARY KEY AUTOINCREMENT,
colonne1 type contrainte,
colonne2 type contrainte,
colonneFK type contrainte,
CONSTRAINT nom_de_la_FK
FOREIGN KEY (colonneFK)
REFERENCES tableDeLaTableDeReference(PK de la table de référence)
);
```

PK : primary key (clé primaire).
FK : foreign key (clé étrangère).

Exemple

```
CREATE TABLE villes(
id_ville INTEGER PRIMARY KEY AUTOINCREMENT,
cp VARCHAR(5) NOT NULL,
nom_ville VARCHAR(50) NOT NULL,
id_pays INTEGER NOT NULL,
CONSTRAINT fk_villes_pays
FOREIGN KEY (id_pays)
REFERENCES pays(id_pays)
);
```

Ajout d'enregistrements dans la table villes :

vérifiez avant vos id_pays dans la table pays !

```
INSERT INTO villes(cp, nom_ville, id_pays) VALUES('75000', 'Paris',1);
INSERT INTO villes(cp, nom_ville, id_pays) VALUES('69000', 'Lyon',1);
INSERT INTO villes(cp, nom_ville, id_pays) VALUES('99999', 'Roma',2);
```

1.17 - Affichage des enregistrements de plusieurs tables

Prémisses

Pour afficher des enregistrements de plusieurs tables il faut faire une jointure.
Il faut que les tables est une colonne commune (clé primaire/clé étrangère).

Syntaxe de la jointure

```
sqlite>SELECT * | liste de colonnes
FROM table_1 INNER JOIN table_2
ON table_1. colonne_de_jointure = table_2.colonne_de_jointure;
```

Note : il existe d'autres syntaxes.

Exemple

```
sqlite>SELECT *
FROM pays INNER JOIN villes
ON pays.id_pays = villes.id_pays;
```

1|FR|France|1|75000|Paris|1
1|FR|France|2|69000|Lyon|1
2|IT|Italie|3|99999|Roma|2

1.18 - Exercices

Créez une base de données nommée « videotheque ».

Créez une table nommée « genre » (colonnes : id_genre, libelle_genre).

Ajoutez 3 enregistrements dans la table « genre » (comédie, thriller, historique).

Affichez le contenu de la table.

Créez une table nommée « film » (colonnes : id_film, titre, duree, id_genre).

Ajoutez 6 films dans la table « film ».

Affichez le contenu de la table.

Affichez la liste des titres.

Affichez la liste des titres avec le libellée du genre à côté.

1.19 - Bibliographie et Webographie

1.19.1 - Bibliographie

Frédéric Brouard, Christian Soutou et Rudy Bruchez, « SQL », Editions Pearson, 304 pages, 2012, EAN13 : 9782744076305, 29 euros

Christian Soutou, « Programmer avec MySQL », Eyrolles, 480 pages, 2015, EAN13 : 9782212143027, 29,90 euros

1.19.2 - Webographie

Documentation officielle de SQLite
https://www.sqlite.org/docs.html

1.20 - Corrigés des exercices

Créez une base de données nommée « videotheque ».

```
cmd>sqlite3 videotheque.db
```

Créez une table nommée « genre » (colonnes : id_genre, libelle_genre).

```
sqlite>CREATE TABLE genre(
id_genre INTEGER PRIMARY KEY AUTOINCREMENT,
libelle_genre VARCHAR(50) NOT NULL UNIQUE
);
```

Ajoutez 3 enregistrements dans la table « genre » (comédie, thriller, historique).

```
sqlite>INSERT INTO genre(libelle_genre) VALUES('Comédie');
sqlite>INSERT INTO genre(libelle_genre) VALUES('Thriller');
sqlite>INSERT INTO genre(libelle_genre) VALUES('Historique');
```

Affichez le contenu de la table.

```
sqlite>SELECT * FROM genre;
```

Créez une table nommée « film » (colonnes : id_film, titre, duree, id_genre).

```
sqlite>CREATE TABLE film(
id_film INTEGER PRIMARY KEY AUTOINCREMENT,
titre VARCHAR(50) NOT NULL UNIQUE,
duree INTEGER NOT NULL,
id_genre INTEGER NOT NULL
);
```

Ajoutez 6 films dans la table « film ».

```
sqlite>INSERT INTO film(titre, duree, id_genre) VALUES('Coup de foudre à
Nottinghill', 120, 1);
sqlite>INSERT INTO film(titre, duree, id_genre) VALUES('4 mariages et un
enterrement', 90, 1);
sqlite>INSERT INTO film(titre, duree, id_genre) VALUES('Psycho', 120, 2);
sqlite>INSERT INTO film(titre, duree, id_genre) VALUES('Outland', 100, 2);
sqlite>INSERT INTO film(titre, duree, id_genre) VALUES('La reine Margot',
100, 3);
sqlite>INSERT INTO film(titre, duree, id_genre) VALUES('Lagardère', 90,
3);
```

```
```

Affichez le contenu de la table.

```
sqlite>SELECT * FROM film;
```

Affichez la liste des titres.

```
sqlite>SELECT titre FROM film;
```

Affichez la liste des titres avec le libellée du genre à côté.

```
sqlite>SELECT film.titre, genre.libelle_genre
FROM film JOIN genre
ON film.id_genre = genre.id_genre;
```