Le langage SQL

Dans le cadre du modèle relationnel, Edgar F. Codd a développé un ensemble d'opérations mathématiques servant à manipuler les objets de son modèle relationnel que l'on nomme algèbre relationnelle. C'est sur elle qu'est fondé le langage de requêtes SQL (pour Structured Query Language) utilisé par les SGBGR.

Pour présenter le langage SQL nous nous appuierons sur SQLite et nous reprendrons l'exemple des relations Ouvrage et Auteur décrites dans le chapitre **Bases de données**. Pour rappel, le schéma relationnel de la base de données est :

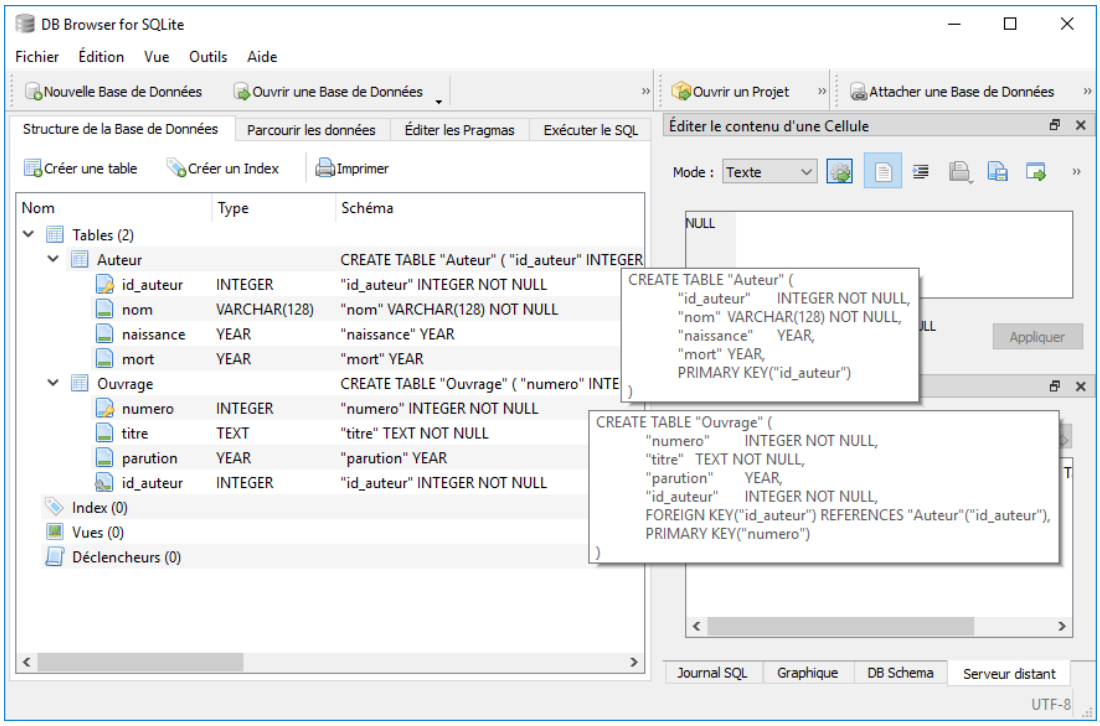
> Auteur(id_auteur:entier, auteur:caractères[128], naissance:année, mort:année)
>
> Ouvrage(numero:entier, titre:texte, date:année, *#id_auteur*:entier)

Remarque SQLite implémente un nombre restreint de types de données. En particulier il n'existe pas de chaîne limitée à un nombre prédéfini de caractères ni de type année. Il reste cependant possible d'utiliser ces types dans la définition du schéma relationnel, mais il faut garder à l'esprit que ceux-ci se verront convertis automatiquement en texte ou en entier.

La création d'une base de donnée (CREATE)

Pour créer une base de données on commence tout simplement par définir son schéma relationnel dans le SGBDR. Ceci peut se faire à l'aide d'outils graphiques ou bien directement en langage SQL.

Nous allons illustrer la première méthode en utilisant le logiciel libre DB Browser for SQLite.



Cette première étape consiste à créer une nouvelle base de données, puis à créer les tables qui la constituent en définissant leur schéma relationnel et en précisant les clés primaires et étrangères (*contrainte d'intégrité de référence*).

La création d'une base de données peut s'effectuer aussi dans l'interface de ligne de commande **sqlite3** ou bien sous l'onglet « Exécuter le SQL » de DB Browser for SQLite avec la commande **CREATE TABLE** en suivant la syntaxe :

```
CREATE TABLE table (
      attribut_1 type,
      attribut_2 type,
      ...
      attribut_n type,
      PRIMARY KEY(attribut_1),
      FOREIGN KEY(attribut_k) REFERENCES autre_table(attribut)
);
```

Pour notre exemple Auteur-Ouvrage, il faut saisir les instructions SQL suivantes :

```
CREATE TABLE "Auteur" (
      "id_auteur" INTEGER NOT NULL,
      "nom" VARCHAR(128) NOT NULL,
      "naissance" YEAR,
      "mort"      YEAR,
      PRIMARY KEY("id_auteur")
);

CREATE TABLE "Ouvrage" (
      "numero"    INTEGER NOT NULL,
      "titre"     TEXT NOT NULL,
      "parution"  YEAR,
      "id_auteur" INTEGER NOT NULL,
      PRIMARY KEY("numero"),
      FOREIGN KEY("id_auteur") REFERENCES "Auteur"("id_auteur")
);
```

L'insertion de données (INSERT INTO)

Le peuplement initial de la base de données s'effectue avec la commande **INSERT INTO** selon la syntaxe suivante :

```
INSERT INTO table VALUES(tuple);
```

Rapporté à notre exemple, cela correspond aux instructions SQL :

```
INSERT INTO "Auteur" VALUES(63,"Comtesse de Ségur",1799,1874);
INSERT INTO "Auteur" VALUES(75,"Victor Hugo",1802,1885);
INSERT INTO "Auteur" VALUES(81,"Jules Verne",1828,1905);

INSERT INTO "Ouvrage" VALUES(142,"Le Dernier Jour d'un condamné",1829,75);
INSERT INTO "Ouvrage" VALUES(241,"Voyage au centre de la Terre",1864,81);
INSERT INTO "Ouvrage" VALUES(323,"Les Petites Filles modèles",1858,63);
INSERT INTO "Ouvrage" VALUES(358,"Les Misérables",1862,75);
INSERT INTO "Ouvrage" VALUES(419,"Les Malheurs de Sophie",1858,63);
INSERT INTO "Ouvrage" VALUES(492,"Le Tour du monde en quatre-vingts jours",1872,81);
INSERT INTO "Ouvrage" VALUES(677,"Mémoires d'un âne",1860,63);
INSERT INTO "Ouvrage" VALUES(760,"Vingt mille lieues sous les mers",1869,81);
INSERT INTO "Ouvrage" VALUES(806,"Notre-Dame de Paris",1831,75);
INSERT INTO "Ouvrage" VALUES(815,"De la Terre à la Lune",1865,81);
```

On peut réaliser la même opération dans DB Browser for SQLite, en se rendant dans l'onglet « Parcourir les données ». Il faut alors créer de nouveaux enregistrements et saisir les informations des différents tuples pour chacune des tables.

Nous venons de voir comment créer une base de données en définissant les tables de son schéma relationnel, avec la commande **CREATE**, et comment remplir celle-ci en insérant les tuples de chaque relation dans les différentes tables au moyen de la commande **INSERT INTO**. Passons maintenant aux commandes usuelles.

La projection (SELECT)

La projection consiste à créer une nouvelle relation en conservant certains attributs (en partie ou en totalité). Autrement dit, la projection opère une sélection sur les colonnes d'une table. Elle se réalise avec la commande **SELECT**, qui est de loin la commande la plus utilisée en SQL, en respectant la syntaxe :

SELECT [*attribut_1,attribut_2,...,attribut_n*|*] **FROM** *table*;

(où * est un raccourci qui signifie « *tous les attributs* »)

- Ainsi pour obtenir le contenu entier de la table Auteur, il convient de faire :

```
SELECT * FROM Auteur;
```

```
63|Comtesse de Ségur|1799|1874
75|Victor Hugo|1802|1885
81|Jules Verne|1828|1905
```

	id_auteur	nom	naissance	mort
1	63	Comtesse de Ségur	1799	1874
2	75	Victor Hugo	1802	1885
3	81	Jules Verne	1828	1905

- Tandis que pour obtenir seulement le nom et l'année de naissance, on fera :

```
SELECT nom,naissance FROM Auteur;
```

```
Comtesse de Ségur|1799
Victor Hugo|1802
Jules Verne|1828
```

	nom	naissance
1	Comtesse de Ségur	1799
2	Victor Hugo	1802
3	Jules Verne	1828

Remarque Le langage SQL s'affranchit quelque peu de l'algèbre relationnelle en autorisant les lignes identiques comme on peut le voir avec la commande suivante :

```
SELECT parution FROM Ouvrage;
```

```
1829
1864
1858     apparaît une première fois
1862
1858     apparaît une seconde fois
...
1865
```

La commande **DISTINCT** permet d'éviter ces redondances.

```
SELECT DISTINCT parution FROM Ouvrage;
```

```
1829
1864
1858
1862
1872
1860
1869
1831
1865
```

La sélection (WHERE)

La sélection, aussi appelée restriction, crée une nouvelle relation en sélectionnant les tuples sur la base de conditions à définir et qui portent sur les attributs. Autrement dit, la sélection opère une sélection sur les lignes selon un prédicat qui doit être précisé. Elle s'effectue à l'aide de la commande **WHERE** selon la syntaxe :

SELECT [*attributs*|*] **FROM** *table* **WHERE** *prédicat*;

Le prédicat s'écrit au moyen des opérateurs suivants :

- comparaisons : **=** **<** **>** **<=** **>=** **<>** (différent)
- opérateurs logiques : **AND** **OR** **NOT**
- encadrements : **BETWEEN**
- motifs de chaînes : **LIKE** (**_** pour un caractère quelconque, **%** pour plusieurs)

Voici quelques exemples de sélections :

- Les ouvrages parus après 1865.

```
SELECT * FROM Ouvrage WHERE parution > 1865;
```

```
492|Le Tour du monde en quatre-vingts jours|1872|81
760|Vingt mille lieues sous les mers|1869|81
```

- Les ouvrages parus en 1862 ou 1872.

```
SELECT * FROM Ouvrage WHERE parution = 1862 OR parution = 1872;
```

```
358|Les Misérables|1862|75
492|Le Tour du monde en quatre-vingts jours|1872|81
```

- Les ouvrages parus entre 1860 et 1865.

```
SELECT * FROM Ouvrage WHERE parution BETWEEN 1860 AND 1865;
```

```
241|Voyage au centre de la Terre|1864|81
358|Les Misérables|1862|75
677|Mémoires d'un âne|1860|63
815|De la Terre à la Lune|1865|81
```

- Les ouvrages dont le titre commence par « Les ».

```
SELECT * FROM Ouvrage WHERE titre LIKE "Les%";
```

```
323|Les Petites Filles modèles|1858|63
358|Les Misérables|1862|75
419|Les Malheurs de Sophie|1858|63
```

- Le titre et l'année de parution des ouvrages dont le titre ne contient pas « i ».

```
SELECT titre,parution FROM Ouvrage WHERE NOT titre LIKE "%i%";
```

```
Voyage au centre de la Terre|1864
De la Terre à la Lune|1865
```

La jointure interne (INNER JOIN)

La jointure consiste à créer une nouvelle relation en composant plusieurs relations. Autrement dit, la jointure crée une nouvelle table à partir de plusieurs tables.

Il existe plusieurs sortes de jointures qui apparient différemment les tuples des relations. Parmi celles-ci, la jointure interne crée une nouvelle table en liant uniquement les lignes de deux tables qui respectent une certaine condition portant sur les colonnes (un prédicat). Elle s'opère avec la commande **JOIN … ON**.

SELECT [*attributs*|*] **FROM** *table1* **[INNER] JOIN** *table2* **ON** *prédicat*;

Lorsque le prédicat utilise des noms de colonnes identiques dans les deux tables on doit préfixer ces noms par leur table avec un point de séparation (`table.colonne`).

Voici quelques exemples de jointures internes :

- Les ouvrages appariés à leurs auteurs.

```
SELECT * FROM Ouvrage JOIN Auteur ON Ouvrage.id_auteur = Auteur.id_auteur;
```

```
142|Le Dernier Jour d'un condamné|1829|75|75|Victor Hugo|1802|1885
241|Voyage au centre de la Terre|1864|81|81|Jules Verne|1828|1905
323|Les Petites Filles modèles|1858|63|63|Comtesse de Ségur|1799|1874
358|Les Misérables|1862|75|75|Victor Hugo|1802|1885
419|Les Malheurs de Sophie|1858|63|63|Comtesse de Ségur|1799|1874
492|Le Tour du monde en quatre-vingts jours|1872|81|81|Jules Verne|1828|1905
677|Mémoires d'un âne|1860|63|63|Comtesse de Ségur|1799|1874
760|Vingt mille lieues sous les mers|1869|81|81|Jules Verne|1828|1905
806|Notre-Dame de Paris|1831|75|75|Victor Hugo|1802|1885
815|De la Terre à la Lune|1865|81|81|Jules Verne|1828|1905
```

- Les titres des ouvrages de Jules Verne.

```
SELECT titre FROM Ouvrage JOIN Auteur
                ON Ouvrage.id_auteur = Auteur.id_auteur
             WHERE nom = "Jules Verne";
```

```
Voyage au centre de la Terre
Le Tour du monde en quatre-vingts jours
Vingt mille lieues sous les mers
De la Terre à la Lune
```

- Les noms des auteurs et les titres de leurs ouvrages uniquement lorsqu'ils contiennent le mot « jour ».

```
SELECT DISTINCT nom,titre FROM Auteur JOIN Ouvrage
                            ON Auteur.id_auteur = Ouvrage.id_auteur
                         WHERE titre LIKE "%jour%";
```

```
Victor Hugo|Le Dernier Jour d'un condamné
Jules Verne|Le Tour du monde en quatre-vingts jours
```

Notez comment LIKE est insensible à la casse des caractères.

Remarques

La jointure est particulièrement utilisée lorsqu'il s'agit d'exploiter les clés étrangères.
Même si on peut formuler la sélection (**WHERE**) dans le prédicat, il est préférable de séparer la condition de jointure de la règle de filtrage.
Lorsque le prédicat est une égalité on parle d'équi-jointure.

La mise à jour (UPDATE)

La mise à jour consiste à modifier les valeurs dans des lignes. Elle s'effectue à l'aide de la commande **UPDATE … SET** en précisant la table concernée, ainsi que les colonnes à modifier avec leurs nouvelles valeurs. Elle est le plus souvent complétée d'une sélection (**WHERE**) afin d'identifier les lignes à mettre à jour ; en son absence ce sont toutes les lignes qui sont alors modifiées. La syntaxe est la suivante :

UPDATE *table* **SET** *attribut_1=valeur_1*, …, *attribut_n=valeur_n* [**WHERE** *prédicat*];

Voici un exemple de mise à jour :

- Changement du titre de l'ouvrage portant le numéro 492.

```
UPDATE Ouvrage SET titre = "Le tour du monde en 80 jours" WHERE numero = 492;
SELECT * FROM Ouvrage WHERE numero = 492;
```

```
492|Le tour du monde en 80 jours|1872|81
```

La suppression (DELETE)

La suppression consiste à supprimer une ou plusieurs lignes dans une table. Elle se fait généralement sur la base d'un prédicat, qui sert à identifier les lignes concernées. En l'absence de ce prédicat, ce sont alors toutes les lignes de la table qui sont supprimées. La commande de suppression **DELETE** suit la syntaxe :

DELETE FROM *table* [**WHERE** *prédicat*];

Voici deux exemples de suppression :

- Suppression des ouvrages dont le `id_auteur` n'est pas 75 (*Victor Hugo*).

```
DELETE FROM Ouvrage WHERE id_auteur <> 75;
-- affichage des ouvrages restants après la suppression
SELECT * FROM Ouvrage;
```

```
142|Le Dernier Jour d'un condamné|1829|75
358|Les Misérables|1862|75
806|Notre-Dame de Paris|1831|75
```

- Suppression des auteurs dont le `id_auteur` n'est pas 75 (*Victor Hugo*).

```
DELETE FROM Auteur WHERE id_auteur <> 75;
-- affichage des auteurs restants après la suppression
SELECT * FROM Auteur;
```

```
75|Victor Hugo|1802|1885
```

Remarque Si on commence par le second exemple, un message d'erreur est affiché indiquant que la commande ne respecte pas la contrainte de référence imposée par l'utilisation d'une clé étrangère. Il est en effet interdit de supprimer un auteur tant que des ouvrages de cet auteur sont présents dans la table Ouvrage.

```
-- PRAGMA foreign_keys=ON;
DELETE FROM Auteur WHERE id_auteur <> 75;
```

```
Error: FOREIGN KEY constraint failed
```