Exploitation d'une base de données relationnelle avec SQLite sur MorphOS

(Tutoriel développé et écrit par Sébastien Jeudy en Février 2021)

Présentation :

SQLite est un système de gestion de bases de données relationnelles embarquées, libre et open source. Contrairement aux autres SGBDR, il n'est pas orienté "client-serveur", ses bases de données sont uniquement exploitées en local (idéal pour les logiciels devant embarquer la base de données, donc sans partage des données en réseau avec d'autres utilisateurs).

Pour cela, il est également ultra-léger (< 1 Mo).

Autre intérêt : l'intégralité d'une base de données est stockée dans un fichier indépendant de la plateforme. La gestion des droits d'accès et de modification des données se fait alors par le système de fichiers du système d'exploitation.

Intégré à de nombreux logiciels grand public, produits professionnels, bibliothèques standards, langages de programmation, systèmes embarqués, smartphones et tablettes, SQLite est au final le moteur de bases de données le plus utilisé au monde.

Une chance pour nous : la version AmigaOS / MorphOS (3.34.0 à ce jour) est aussi la dernière disponible, identique à la version Windows / Linux / Mac OS X !

Etant spécialiste en bases de données et utilisateur de MorphOS, j'ai donc profité de la disponibilité de SQLite sur MorphOS pour approfondir ce SGBDR que je n'avais jamais utilisé.

L'objectif de ce tutoriel est de vous apporter un cas concret d'exploitation de base de données relationnelle avec SQLite, mais pas d'enseigner en détail le langage SQL ni l'implémentation de SQLite dans une application logicielle. Cependant, il est présenté de sorte que le novice peut tout de même s'y retrouver et en comprendre la finalité.

Enfin, l'environnement choisi est MorphOS mais l'utilisation de SQLite est logiquement la même sur AmigaOS ou tout autre système.

Pour découvrir davantage SQLite : https://fr.wikipedia.org/wiki/SQLite

Site officiel : https://www.sqlite.org

Téléchargement :

https://www.morphos-storage.net/?find=sqlite

Après avoir téléchargé l'archive, la désarchiver et copier son dossier à l'endroit souhaité sur disque.

Personnellement, j'ai renommé son dossier en "SQLite".

Utilisation en lignes de commandes avec l'outil SQLite nommé "sqlite3" :

Ouvrir un Terminal Shell, puis lancer l'outil SQLite - le tout en créant une nouvelle base de données - en tapant sur une seule ligne (avec un espace après "sqlite3" et attention à vos propres chemins) :

Work:Applications/SQLite/build-ppc-morphos/bin/sqlite3
Work:Developpement/SQLite/BDD/Bibliotheque

Sachant que chez moi l'outil "sqlite3" est dans "Work:Applications/SQLite/build-ppc-morphos/bin/" et que le fichier de la base de données créée s'appellera "Bibliotheque" dans "Work:Developpement/SQLite/BDD/".

Une autre solution serait de copier "sqlite3" dans "C:", ce qui donnerait :

sqlite3 Work:Developpement/SQLite/BDD/Bibliotheque

Puis dans l'outil SQLite, saisir l'ensemble des commandes suivantes (après l'invite de commande "sqlite> "), dans l'ordre précisé afin de suivre correctement ce tutoriel - pas à pas - jusqu'au bout.

Voir l'ensemble des commandes internes intégrées à l'outil SQLite (le point en préfixe est obligatoire pour les commandes internes) :

.help

Pour quitter l'outil SQLite :

.quit

Pour retourner dans SQLite et dans la base de données "Bibliotheque", retaper (chez moi) :

Work:Applications/SQLite/build-ppc-morphos/bin/sqlite3
Work:Developpement/SQLite/BDD/Bibliotheque

Ou avec "sqlite3" dans "C:" :

sqlite3 Work:Developpement/SQLite/BDD/Bibliotheque

Dans SQLite, vérifier les bases de données utilisées :

.databases

Créer la nouvelle table "Auteurs" dans la base de données "Bibliotheque" (le point-virgule en fin d'instruction est obligatoire pour les requêtes SQL) :

create table Auteurs(IdAuteur integer primary key not null, Nom varchar(30) not null, Prenom varchar(30) not null, DateNaissance date, DateDeces date);

Précision SQLite : avec ce genre de "primary key" sur une seule colonne de type entier (integer), sa valeur sera automatiquement incrémentée au fur et à mesure des insertions de lignes dans la table.

Autre rappel : la clé primaire permet d'éviter les doublons de lignes dans la table mais aussi d'identifier une ligne de façon unique.

Créer la nouvelle table "Livres" dans la base de données "Bibliotheque" :

create table Livres(IdLivre integer primary key not null, Titre varchar(100) not null, IdAuteur integer not null, Parution year(4) not null, NbPages integer,

```
constraint fk_auteurs foreign key (IdAuteur) references Auteurs(IdAuteur));
```

Précision SQLite : la contrainte "foreign key" permet de relier la table "Livres" à la table "Auteurs" via les colonnes respectives "IdAuteur", celle-ci ne pouvant pas être rajoutée ultérieurement avec un "alter table" (comme dans d'autres SGBDR).

Autre rappel : une clé étrangère permet de contrôler que les données d'une (ou plusieurs) colonne(s) de table correspondent bien aux données de la clé primaire d'une autre table (de référence). Ici, chaque livre de la table "Livres" doit être affecté à un auteur existant dans la table "Auteurs".

Activer le contrôle des clés étrangères (sinon inactif) :

```
pragma foreign_keys = on;
```

Attention : à réactiver à chaque nouvelle utilisation de la base de données (accès à SQLite) !

Vérifier les tables créées (dans la base de données courante "Bibliotheque") :

```
.tables
```

Revoir le schéma ("create table") des tables existantes :

```
.schema Auteurs

.schema Livres
```

Insérer des lignes de données dans la table "Auteurs" :

```
insert into Auteurs (Nom, Prenom) values ('Verne', 'Jules');
insert into Auteurs (Nom, Prenom) values ('Stevenson', 'Robert Louis');
insert into Auteurs (Nom, Prenom) values ('De Saint-Exupéry', 'Antoine');
```

Interroger toute la table "Auteurs" :

```
select * from Auteurs;
```

Résultat :

```
1|Verne|Jules||
2|Stevenson|Robert Louis||
3|De Saint-Exupéry|Antoine||
```

Rappel : une "*" affiche toutes les colonnes et un "select" sans condition "where" retourne toutes les lignes.

Insérer des lignes de données dans la table "Livres" :

```
insert into Livres (Titre, IdAuteur, Parution) values ('Le Tour du Monde en 80 Jours', 1, 1872);
insert into Livres (Titre, IdAuteur, Parution) values ('De la Terre à la Lune', 1, 1865);
insert into Livres (Titre, IdAuteur, Parution) values ('Autour de la Lune', 1,
```

```
1870);
insert into Livres (Titre, IdAuteur, Parution) values ('L''Ile au Trésor', 2,
1883);
insert into Livres (Titre, IdAuteur, Parution) values ('L''Etrange Cas du
Docteur Jekyll et de M. Hyde', 2, 1886);
insert into Livres (Titre, IdAuteur, Parution) values ('Vol de Nuit', 3, 1931);
insert into Livres (Titre, IdAuteur, Parution) values ('Le Petit Prince', 3,
1943);
insert into Livres (Titre, IdAuteur, Parution) values ('Terre des Hommes', 3,
1939);
```

Interroger toute la table "Livres" :

```
select * from Livres;
```

Résultat :

```
1|Le Tour du Monde en 80 Jours|1|1872|
2|De la Terre à la Lune|1|1865|
3|Autour de la Lune|1|1870|
4|L'Ile au Trésor|2|1883|
5|L'Etrange Cas du Docteur Jekyll et de M. Hyde|2|1886|
6|Vol de Nuit|3|1931|
7|Le Petit Prince|3|1943|
8|Terre des Hommes|3|1939|
```

Avec tri selon la colonne "Parution" :

```
select * from Livres order by Parution asc;
```

Résultat :

```
2|De la Terre à la Lune|1|1865|
3|Autour de la Lune|1|1870|
1|Le Tour du Monde en 80 Jours|1|1872|
4|L'Ile au Trésor|2|1883|
5|L'Etrange Cas du Docteur Jekyll et de M. Hyde|2|1886|
6|Vol de Nuit|3|1931|
8|Terre des Hommes|3|1939|
7|Le Petit Prince|3|1943|
```

Précision : "asc" pour ascendant et "desc" pour descendant.

Afficher le nombre de lignes des tables :

```
select count(*) from Auteurs;

select count(*) from Livres;
```

Essayer d'insérer un livre dont l'auteur n'existe pas (violation de clé étrangère) :

```
insert into Livres (Titre, IdAuteur, Parution) values ('Le Vieil Homme et la
Mer', 4, 1952);
```

Résultat :

Error: FOREIGN KEY constraint failed

Essayer de supprimer un auteur affecté à des livres (violation de clé étrangère) :

delete from Auteurs where IdAuteur = 1;

Résultat :

Error: FOREIGN KEY constraint failed

Interroger plusieurs tables reliées avec jointure (liaison entre clé étrangère et clé primaire, en général mais pas obligatoirement) :

select Titre, Prenom, Nom, Parution from Livres L join Auteurs A on L.IdAuteur = A.IdAuteur;

Résultat (liste des livres avec leur auteur et leur date de parution) :

Le Tour du Monde en 80 Jours|Jules|Verne|1872
De la Terre à la Lune|Jules|Verne|1865
Autour de la Lune|Jules|Verne|1870
L'Ile au Trésor|Robert Louis|Stevenson|1883
L'Etrange Cas du Docteur Jekyll et de M. Hyde|Robert Louis|Stevenson|1886
Vol de Nuit|Antoine|De Saint-Exupéry|1931
Le Petit Prince|Antoine|De Saint-Exupéry|1943
Terre des Hommes|Antoine|De Saint-Exupéry|1939

Créer la nouvelle vue "LivresAuteurs" (table image résultant d'un "select" interrogeant une ou plusieurs tables) :

create view LivresAuteurs as select Titre, Prenom, Nom, Parution from Livres L join Auteurs A on L.IdAuteur = A.IdAuteur;

Résultat :

.tables

.schema LivresAuteurs

Rappel : une fois créée, une vue s'utilise comme une table mais sans contenir de données.

Exemple d'utilisation de la vue "LivresAuteurs" :

select * from LivresAuteurs order by Parution asc;

Résultat :

De la Terre à la Lune|Jules|Verne|1865
Autour de la Lune|Jules|Verne|1870
Le Tour du Monde en 80 Jours|Jules|Verne|1872
L'Ile au Trésor|Robert Louis|Stevenson|1883
L'Etrange Cas du Docteur Jekyll et de M. Hyde|Robert Louis|Stevenson|1886
Vol de Nuit|Antoine|De Saint-Exupéry|1931
Terre des Hommes|Antoine|De Saint-Exupéry|1939
Le Petit Prince|Antoine|De Saint-Exupéry|1943

Rappel : une vue allège les syntaxes et optimise les traitements des requêtes SQL complexes.

Autre exemple d'utilisation de la vue "LivresAuteurs" :

```
select * from LivresAuteurs where Nom = 'Verne' or Nom like '%saint%' order by Parution desc;
```

Résultat :

```
Le Petit Prince|Antoine|De Saint-Exupéry|1943
Terre des Hommes|Antoine|De Saint-Exupéry|1939
Vol de Nuit|Antoine|De Saint-Exupéry|1931
Le Tour du Monde en 80 Jours|Jules|Verne|1872
Autour de la Lune|Jules|Verne|1870
De la Terre à la Lune|Jules|Verne|1865
```

Afficher le nombre de livres par auteur (avec regroupement par nom) :

```
select count(*), Nom from LivresAuteurs group by Nom;
```

Résultat :

```
3|De Saint-Exupéry
2|Stevenson
3|Verne
```

Attention, pas la même chose que :

```
select count(*), Nom from LivresAuteurs;
```

(résultat erroné ou ambigu)

Autre exemple de regroupement (avec condition "having") :

```
select count(*), Nom from LivresAuteurs group by Nom having count(*) >= 3;
```

Résultat :

```
3|De Saint-Exupéry
3|Verne
```

Avec alias "as Total" pour "count(*)" :

```
select count(*) as Total, Nom from LivresAuteurs group by Nom having Total >= 3;
```

Même résultat :

```
3|De Saint-Exupéry
3|Verne
```

Afficher le résultat de plusieurs "select" à la suite (union de sélections) :

```
select Prenom, Nom from Auteurs union all select Titre, Parution from Livres;
```

Résultat :

```
Jules|Verne
Robert Louis|Stevenson
Antoine|De Saint-Exupéry
Le Tour du Monde en 80 Jours|1872
De la Terre à la Lune|1865
Autour de la Lune|1870
L'Ile au Trésor|1883
L'Etrange Cas du Docteur Jekyll et de M. Hyde|1886
Vol de Nuit|1931
Le Petit Prince|1943
Terre des Hommes|1939
```

Précisions : les "select" doivent retourner le même nombre de colonnes. "all" conserve les lignes en doublon.

Utiliser des fonctions d'agrégat (retournent une valeur unique à partir d'un ensemble de valeurs) :

Pour rappel : `select Parution from Livres;`

Comptage : `select count(Parution) from Livres;`

Minimum : `select min(Parution) from Livres;`

Maximum : `select max(Parution) from Livres;`

Somme : `select sum(Parution) from Livres;`

Moyenne : `select avg(Parution) from Livres;`

Précisions : les fonctions d'agrégat ne tiennent pas compte des valeurs vides (null), sauf le count(*). "count", "min" et "max" fonctionnent également sur les valeurs alphanumériques (textes).

Modifier un enregistrement dans une table :

```
update Auteurs set DateNaissance = '1828-02-08', DateDeces = '1905-03-24' where IdAuteur = 1;
```

Vérifier :

```
select * from Auteurs where IdAuteur = 1;
```

Résultat :

```
1|Verne|Jules|1828-02-08|1905-03-24
```

Attention : un "update" sans condition "where" modifie toutes les lignes de la table !

Rechercher des valeurs de colonne à vide (null) ou non vide (not null) :

```
select * from Auteurs where DateNaissance is null;

select * from Auteurs where DateNaissance is not null;
```

Supprimer un enregistrement dans une table :

```
delete from Livres where IdLivre = 8;
```

Vérifier :

```
select * from Livres;
```

Résultat :

```
1|Le Tour du Monde en 80 Jours|1|1872|
2|De la Terre à la Lune|1|1865|
3|Autour de la Lune|1|1870|
4|L'Ile au Trésor|2|1883|
5|L'Etrange Cas du Docteur Jekyll et de M. Hyde|2|1886|
6|Vol de Nuit|3|1931|
7|Le Petit Prince|3|1943|
```

Attention : un "delete" sans condition "where" supprime toutes les lignes de la table !

Utiliser une transaction (pour éventuellement annuler les modifications avec un "rollback;") :

Démarrer la transaction :

```
begin;
```

Effectuer des modifications, par exemple vider la table "Livres" :

```
delete from Livres;
```

```
select * from Livres;
```

Annuler les modifications (et quitter la transaction) :

```
rollback;
```

Vérifier :

```
select * from Livres;
```

Résultat :

```
1|Le Tour du Monde en 80 Jours|1|1872|
2|De la Terre à la Lune|1|1865|
3|Autour de la Lune|1|1870|
4|L'Ile au Trésor|2|1883|
5|L'Etrange Cas du Docteur Jekyll et de M. Hyde|2|1886|
6|Vol de Nuit|3|1931|
7|Le Petit Prince|3|1943|
```

Précision : un "commit;" à la place du "rollback;" aurait enregistré les modifications (et quitté la transaction).

Créer un trigger / déclencheur sur la table "Auteurs" (programme s'exécutant automatiquement lorsqu'un évènement insert / update / delete survient sur la table associée) :

```
create trigger suppr_auteur_et_livres before delete on Auteurs begin delete from
Livres where Livres.IdAuteur = old.IdAuteur; end;
```

Résultat :

```
.schema Auteurs
```

Précision : "old" fait référence à la table "Auteurs" avant suppression ("before delete").

Exemple d'utilisation du trigger "suppr_auteur_et_livres" (lorsqu'un auteur est supprimé dans la table "Auteurs", ses livres sont aussi automatiquement supprimés dans la table "Livres") :

```
delete from Auteurs where IdAuteur = 2;
```

Résultats :

```
select * from Auteurs;

1|Verne|Jules|1828-02-08|1905-03-24
3|De Saint-Exupéry|Antoine||

select * from Livres;

1|Le Tour du Monde en 80 Jours|1|1872|
2|De la Terre à la Lune|1|1865|
3|Autour de la Lune|1|1870|
6|Vol de Nuit|3|1931|
7|Le Petit Prince|3|1943|
```

Exporter toutes les données de la table "Livres" dans le fichier "RAM:Livres.csv" :

```
.once RAM:Livres.csv

select * from Livres;
```

Vérifier (en utilisant la commande interne ".system" d'appel de commandes système depuis SQLite) :

```
.system dir RAM:

.system type RAM:Livres.csv
```

Résultat :

```
1|Le Tour du Monde en 80 Jours|1|1872|
2|De la Terre à la Lune|1|1865|
3|Autour de la Lune|1|1870|
6|Vol de Nuit|3|1931|
7|Le Petit Prince|3|1943|
```

Réimporter toutes les données du fichier "RAM:Livres.csv" dans la table "Livres" (à vider au préalable) :

```
delete from Livres;
```

```
select * from Livres;

.import RAM:Livres.csv Livres

select * from Livres;
```

Résultat :

```
1|Le Tour du Monde en 80 Jours|1|1872|
2|De la Terre à la Lune|1|1865|
3|Autour de la Lune|1|1870|
6|Vol de Nuit|3|1931|
7|Le Petit Prince|3|1943|
```

Faire la sauvegarde complète (backup) de la base de données "Bibliotheque" dans le fichier "RAM:Bibliotheque.bak" :

```
.backup RAM:Bibliotheque.bak
```

Vérifier :

```
.system dir RAM:

.system type RAM:Bibliotheque.bak
```

Précision SQLite : cette sauvegarde (et sa restauration) peut être également réalisée en copiant simplement le fichier sur disque de la base de données.

Détruire les trigger, vue et tables créés :

```
drop trigger suppr_auteur_et_livres;

drop view LivresAuteurs;

drop table Livres;

drop table Auteurs;
```

Vérifier :

```
.tables
```

Faire la restauration complète (restore) de la base de données "Bibliotheque" depuis le fichier "RAM:Bibliotheque.bak" :

```
.restore RAM:Bibliotheque.bak
```

Vérifier :

```
.tables

.schema Auteurs

.schema Livres

.schema LivresAuteurs

select * from Auteurs;
```

```
select * from Livres;

select * from LivresAuteurs;
```

Faire la sauvegarde SQL complète (dump) de la base de données "Bibliotheque" dans le fichier "RAM:Bibliotheque.sql" (notamment pour une migration vers un autre SGBDR) :

```
.output RAM:Bibliotheque.sql

.dump

.quit
```

Vérifier (dans le Terminal Shell) :

```
dir RAM:

type RAM:Bibliotheque.sql
```

Le mot de la fin :

Vous voilà venu à bout de ce tutoriel qui vous a introduit l'essentiel des fonctionnalités d'exploitation d'une base de données relationnelle avec SQLite. Un SGBDR digne de ce nom, qui dispose d'un langage SQL riche et relativement standard. En définitive une excellente solution pour gérer les données dans le développement de vos logiciels sur MorphOS (et AmigaOS) !