Supprimer des lignes : DELETE

Une ligne encodée par erreur doit disparaître. Avant de supprimer, tu sauvegardes la base dans un fichier SQL ; puis tu supprimes avec DELETE, en vérifiant d'abord par un SELECT.

4TTR 5TTR 6TTR Découverte

Une ligne de test oubliée, un doublon encodé par erreur : parfois, une ligne doit disparaître. Une base de données n'a pas de corbeille. Une suppression enregistrée est définitive. Avant de supprimer quoi que ce soit, tu vas donc apprendre à sauvegarder ta base.

> 💡 **BASE DE DÉPART**
>
> Télécharge `mon-ecole-relier.db` dans la barre latérale et ouvre-la avec DB Browser for SQLite. Elle contient deux tables : `eleve` (12 élèves) et `professeur` (6 professeurs, dont Paul Renard, qui n'est plus actif). Le SQL s'écrit dans l'onglet **Exécuter le SQL**.

🎯 Objectifs

À la fin de cet article, tu seras capable de :

1. **Sauvegarder** une base complète dans un fichier `.sql`, et la restaurer.
2. **Supprimer** des lignes avec `DELETE FROM … WHERE`.
3. **Vérifier** par un `SELECT` les lignes qui vont disparaître, avant de supprimer.
4. **Annuler** une suppression ratée tant qu'elle n'est pas enregistrée.
5. **Distinguer** `DELETE`, qui vide des lignes, de `DROP TABLE`, qui supprime la table.
6. **Expliquer** pourquoi des numéros d'`id` manquent après une suppression.

Avant de supprimer : sauvegarder

Une base SQLite est un simple fichier. Le copier dans un autre dossier est déjà une sauvegarde. Mais il existe une autre forme de sauvegarde, plus lisible et plus pratique à transmettre : le fichier SQL.

> 📖 **NOUVELLE NOTION : LA SAUVEGARDE**
>
> Une **sauvegarde** est une copie de la base, rangée à part, qui permet de retrouver les données si une manipulation tourne mal. On la fait **avant** une opération risquée, pas après. Une sauvegarde qu'on n'a jamais essayé de restaurer n'est pas une garantie.
>
> En anglais : GB ***backup***.

Exporter la base en fichier SQL

1. Clique d'abord sur **Écrire les modifications** (`Ctrl+S`), pour que la sauvegarde contienne l'état enregistré de la base.
2. Menu **Fichier > Exporter > Base de données vers fichier SQL…**
3. Sélectionne toutes les tables et garde les options proposées par défaut (structure et données).
4. Enregistre le fichier sous le nom `sauvegarde-mon-ecole.sql`.

> 📖 **NOUVELLE NOTION : L'EXPORT SQL**
>
> **Exporter** une base en SQL, c'est la transformer en un script : un fichier texte qui contient les `CREATE TABLE` de toutes les tables, puis un `INSERT` par ligne. En exécutant ce script dans une base vide, on reconstruit la base à l'identique. On appelle aussi ce fichier un *dump*, un « vidage » de la base.
>
> En anglais : GB ***export*** ou GB ***dump***.

Ce que contient le fichier

Ouvre `sauvegarde-mon-ecole.sql` avec le Bloc-notes. Voici la partie qui concerne les professeurs. La présentation exacte (guillemets, espaces, options) peut varier légèrement selon la version de DB Browser :

```
BEGIN TRANSACTION;
CREATE TABLE professeur (
    id        INTEGER PRIMARY KEY,
    nom       TEXT    NOT NULL,
    prenom    TEXT    NOT NULL,
    email     TEXT    NOT NULL UNIQUE,
    actif     INTEGER NOT NULL DEFAULT 1,
    telephone TEXT
);
INSERT INTO "professeur" VALUES(1,'Dubois','Sophie','s.dubois@ecole.be',1,'0471 23 45 67');
INSERT INTO "professeur" VALUES(2,'Lambrechts','Ludovic','l.lambrechts@ecole.be',1,'0485 11 22 33');
INSERT INTO "professeur" VALUES(3,'Vermeulen','Marc','m.vermeulen@ecole.be',1,NULL);
INSERT INTO "professeur" VALUES(4,'Leroy','Anne','a.leroy@ecole.be',1,NULL);
INSERT INTO "professeur" VALUES(5,'Martin','Julie','j.martin@ecole.be',1,NULL);
INSERT INTO "professeur" VALUES(6,'Renard','Paul','p.renard@ecole.be',0,NULL);
COMMIT;
```

Tu reconnais le `CREATE TABLE` de la table, puis un `INSERT` par professeur, avec toutes ses valeurs dans l'ordre des colonnes. La table `eleve` est construite de la même façon, juste au-dessus. `BEGIN TRANSACTION` et `COMMIT` encadrent le tout : si une instruction échoue pendant la restauration, rien n'est à moitié importé.

> 🧠 **LE FICHIER `.DB` CONTIENT LA BASE ; LE FICHIER `.SQL` CONTIENT LES INSTRUCTIONS POUR LA RECONSTRUIRE.** Le second est du texte : on peut le lire, l'envoyer par mail, le ranger avec un projet, ou le rendre comme travail.

Restaurer une sauvegarde

Une sauvegarde ne vaut que si elle se restaure. Essaie tout de suite, dans une base neuve, pour ne pas toucher à la tienne :

1. Menu **Fichier > Nouvelle base de données**, nomme-la `restauration.db`. Si DB Browser ouvre une fenêtre pour créer une table, ferme-la sans rien créer.
2. Menu **Fichier > Importer > Base de données depuis un fichier SQL…**, puis choisis `sauvegarde-mon-ecole.sql`. Si DB Browser te propose de créer une nouvelle base pour accueillir les données, garde plutôt `restauration.db`, qui est vide.
3. Ouvre l'onglet **Parcourir les données** : `eleve` contient 12 élèves et `professeur` 6 professeurs.
4. Clique sur **Écrire les modifications**, puis rouvre `mon-ecole-relier.db` pour la suite.

Ajouter une ligne de test

Pour s'exercer sans rien abîmer, on ajoute un faux professeur. `INSERT INTO professeur (colonnes) VALUES (valeurs)` ajoute une ligne en rangeant chaque valeur dans la colonne de même position. On ne donne ni `id` ni `actif` : SQLite attribue le numéro suivant et la valeur par défaut `1`.

```
INSERT INTO professeur (nom, prenom, email)
VALUES ('Test', 'Essai', 'test@ecole.be');
```

```
SELECT * FROM professeur;
```

```
id | nom        | prenom  | email                 | actif | telephone
---+------------+---------+-----------------------+-------+--------------
1  | Dubois     | Sophie  | s.dubois@ecole.be     | 1     | 0471 23 45 67
2  | Lambrechts | Ludovic | l.lambrechts@ecole.be | 1     | 0485 11 22 33
3  | Vermeulen  | Marc    | m.vermeulen@ecole.be  | 1     | NULL
4  | Leroy      | Anne    | a.leroy@ecole.be      | 1     | NULL
5  | Martin     | Julie   | j.martin@ecole.be     | 1     | NULL
6  | Renard     | Paul    | p.renard@ecole.be     | 0     | NULL
7  | Test       | Essai   | test@ecole.be         | 1     | NULL
```

Le faux professeur porte le numéro 7.

Supprimer une ligne : le SELECT d'abord

Étape 1 : vérifier ce qui va disparaître

On écrit la condition dans un `SELECT`. `WHERE` veut dire *où* : on garde les lignes où la condition est vraie.

```
SELECT *
FROM professeur
WHERE id = 7;
```

```
id | nom  | prenom | email         | actif | telephone
---+------+--------+---------------+-------+----------
7  | Test | Essai  | test@ecole.be | 1     | NULL
```

Une ligne, et c'est bien la ligne de test.

Étape 2 : transformer le SELECT en DELETE

On garde le `FROM` et le `WHERE`, et on remplace `SELECT *` par `DELETE` :

```
DELETE FROM professeur
WHERE id = 7;
```

`DELETE FROM` veut dire *supprime de* : « supprime de la table `professeur` les lignes où `id` vaut 7 ».

> 📖 **NOUVELLE NOTION : DELETE**

L'instruction `DELETE FROM table WHERE condition` supprime toutes les lignes où la condition est vraie. Elle supprime des lignes entières : on ne cite aucune colonne. La table, elle, reste en place.

En anglais : GB ***delete***, *supprimer*.

DB Browser annonce une ligne affectée. Vérifie :

```
SELECT * FROM professeur;
```

```
id | nom        | prenom  | email                | actif | telephone
---+------------+---------+----------------------+-------+--------------
1  | Dubois     | Sophie  | s.dubois@ecole.be    | 1     | 0471 23 45 67
2  | Lambrechts | Ludovic | l.lambrechts@ecole.be | 1     | 0485 11 22 33
3  | Vermeulen  | Marc    | m.vermeulen@ecole.be | 1     | NULL
4  | Leroy      | Anne    | a.leroy@ecole.be     | 1     | NULL
5  | Martin     | Julie   | j.martin@ecole.be    | 1     | NULL
6  | Renard     | Paul    | p.renard@ecole.be    | 0     | NULL
```

Clique sur **Écrire les modifications** : la ligne de test a disparu du fichier.

💡 **VIDER UNE SEULE CASE, CE N'EST PAS UN `DELETE`.** Pour effacer le téléphone d'un professeur sans supprimer le professeur, on modifie la ligne : `UPDATE professeur SET telephone = NULL WHERE id = …`.

⚠️ **IL N'Y A PAS DE CORBEILLE.** Une fois les modifications écrites, la ligne supprimée n'existe plus nulle part, sauf dans ta sauvegarde.

L'expérience du DELETE sans WHERE

DB Browser garde tes changements **en attente** jusqu'à ce que tu cliques sur **Écrire les modifications**. Tant que ce n'est pas fait, **Annuler les modifications** revient au dernier enregistrement. Tu viens d'enregistrer : l'expérience est sans danger.

Exécute :

```
DELETE FROM professeur;
```

Aucune erreur, aucune confirmation. DB Browser annonce **six** lignes affectées.

```
SELECT * FROM professeur;
```

```
id | nom | prenom | email | actif | telephone
---+-----+--------+-------+-------+----------
```

Tous les professeurs ont disparu. La table existe toujours, avec ses six colonnes, mais elle est vide.

Clique sur **Annuler les modifications**, puis relance le `SELECT` : les six professeurs sont revenus.

🧠 **UN `DELETE` SANS `WHERE` VIDE TOUTE LA TABLE.** Avant chaque `DELETE`, écris le `SELECT` avec la même condition, compte les lignes, et compare avec le nombre annoncé par DB Browser après l'exécution. S'il ne correspond pas, annule avant d'écrire.

DELETE ou DROP TABLE

Après le `DELETE FROM professeur`, la table était vide mais présente. Essaie maintenant de la supprimer elle-même. `DROP TABLE` veut dire *laisse tomber la table* :

```
DROP TABLE professeur;
```

```
SELECT * FROM professeur;
```

```
no such table: professeur
```

Cette fois, la table n'existe plus du tout : ni lignes, ni colonnes, ni contraintes. Clique sur **Annuler les modifications** pour la retrouver.

Instruction	Famille	Ce qui disparaît	Ce qui reste
`DELETE FROM professeur WHERE id = 7`	DML	les lignes où la condition est vraie	la table et les autres lignes
`DELETE FROM professeur`	DML	toutes les lignes	la table vide, avec ses colonnes
`DROP TABLE professeur`	DDL	la table entière, avec ses lignes	rien

ℹ️ Le DML regroupe les instructions qui travaillent sur les lignes (`SELECT` , `INSERT` , `UPDATE` , `DELETE`). Le DDL regroupe celles qui travaillent sur la structure (`CREATE TABLE` , `ALTER TABLE` , `DROP TABLE`).

Les trous dans la numérotation

La ligne de test portait le numéro 7. Que se passe-t-il pour les numéros quand on supprime au milieu ? Fais l'expérience. Ajoute deux lignes de test :

```
INSERT INTO professeur (nom, prenom, email)
VALUES
    ('Test', 'Un',   'test1@ecole.be'),
    ('Test', 'Deux', 'test2@ecole.be');
```

Supprime la première, puis ajoute une troisième ligne :

```
DELETE FROM professeur WHERE id = 7;

INSERT INTO professeur (nom, prenom, email)
VALUES ('Test', 'Trois', 'test3@ecole.be');
```

```
SELECT id, nom, prenom FROM professeur;
```

```
id | nom        | prenom
---+------------+--------
1  | Dubois     | Sophie
2  | Lambrechts | Ludovic
3  | Vermeulen  | Marc
```

```
4  | Leroy      | Anne
5  | Martin     | Julie
6  | Renard     | Paul
8  | Test       | Deux
9  | Test       | Trois
```

Le numéro 7 manque, et la nouvelle ligne n'est pas venue le reboucher : SQLite prend le plus grand numéro existant et ajoute 1.

> 🧠 **LES TROUS DANS LES `ID` SONT NORMAUX.** Un identifiant sert à désigner une ligne, pas à compter les lignes. Ne cherche jamais à « renuméroter proprement » : si un numéro change, tout ce qui faisait référence à cette ligne désignerait soudain une autre ligne.

> ℹ️ Si tu supprimes la **DERNIÈRE** ligne de la table, son numéro peut être redonné à la ligne suivante, puisque SQLite part du plus grand numéro qui reste.

Clique sur **Annuler les modifications** : la table retrouve ses six professeurs, numérotés de 1 à 6.

🧪 Exercices

Après chaque essai qui modifie des données, clique sur **Annuler les modifications**. Garde ta sauvegarde `sauvegarde-mon-ecole.sql` à portée de main.

Exercice 1 — Annoncer le nombre de lignes ★☆☆☆☆

Pour chaque instruction, pars de la base de l'article. Annonce **avant d'exécuter** combien de lignes vont disparaître, écris le `SELECT` qui le prouve, exécute, compare avec DB Browser, puis annule.

```sql
-- a)
DELETE FROM eleve WHERE commune = 'Perwez';

-- b)
DELETE FROM eleve WHERE absences = 0;

-- c)
DELETE FROM eleve WHERE id = 99;

-- d)
DELETE FROM professeur WHERE actif = 0;

-- e)
DELETE FROM eleve;
```

Exercice 2 — Lire et corriger ★★☆☆☆

L'objectif de chaque instruction est de supprimer l'élève Noah Charlier (`id` 3). Exécute-la, recopie le message, trouve l'erreur et écris la version correcte. N'oublie pas d'annuler après la version correcte.

```sql
-- a)
DELETE * FROM eleve WHERE id = 3;

-- b)
```

```sql
DELETE eleve WHERE id = 3;

-- c)
DELETE FROM eleve WHERE nom = Charlier;

-- d)
DROP TABLE eleve WHERE id = 3;
```

Exercice 3 — DELETE, UPDATE ou DROP TABLE ? ★★☆☆☆

Pour chaque situation, choisis l'instruction adaptée et écris-la complètement.

1. Arthur Kevers (`id` 11) a quitté l'école en cours d'année, et on ne veut plus garder sa ligne.
2. Emma Bastin (`id` 2) ne veut plus que son adresse e-mail figure dans la base, mais elle reste élève.
3. Une table `import_eleve`, qui a servi à un import, n'est plus utile.
4. Une table `essai` doit être vidée pour recommencer un test, mais sa structure doit rester.

Exercice 4 — Sauvegarder, casser, restaurer ★★★☆☆

1. Exporte ta base dans `sauvegarde-exercice.sql`.
2. Exécute `DELETE FROM eleve;`, puis clique sur **Écrire les modifications**. Tes élèves ont disparu pour de bon.
3. Crée une base neuve et restaure-y la sauvegarde. Vérifie que les 12 élèves sont revenus.
4. Explique en deux phrases pourquoi **Annuler les modifications** ne pouvait plus t'aider à l'étape 3.

Exercice 5 — Supprimer ou désactiver ? ★★★☆☆

Paul Renard a quitté l'école. Dans la base, on a mis `actif = 0` au lieu de supprimer sa ligne.

1. Donne deux situations où l'école a encore besoin de ses informations après son départ.
2. Quelles informations seraient perdues avec `DELETE FROM professeur WHERE id = 6;` ?
3. Écris la requête qui affiche uniquement les professeurs inactifs.

✍️ À retenir

- **Sauvegarde avant de supprimer** : *Fichier > Exporter > Base de données vers fichier SQL…* produit un script avec les `CREATE TABLE` et les `INSERT`.
- Une sauvegarde se restaure dans une base vide avec *Fichier > Importer > Base de données depuis un fichier SQL…*.
- `DELETE FROM table WHERE condition` supprime des lignes entières ; **le** `SELECT` **d'abord**, avec la même condition.
- **Sans** `WHERE`**, toutes les lignes disparaissent**, sans confirmation.
- Tant que les modifications ne sont pas écrites, **Annuler les modifications** rattrape l'erreur ; ensuite, seule la sauvegarde le peut.
- `DELETE` vide des lignes, la table reste ; `DROP TABLE` supprime la table elle-même.
- Les trous dans les `id` sont normaux : on ne renumérote jamais.

Suite

Tu sais maintenant créer une table et y ajouter, modifier ou supprimer des lignes. Révise l'essentiel avec la Fiche à étudier — Définir et modifier ses données, puis passe au niveau suivant : Relier deux entités (1-N).