Limiter le nombre de lignes avec LIMIT

Les trois élèves les plus absents, le plus âgé, les deux derniers de la liste : LIMIT ne garde que le début d'un résultat trié. C'est aussi l'occasion de fixer l'ordre des clauses et une méthode pour traduire une question en requête.

4TTR 5TTR 6TTR Découverte

La direction veut rencontrer les trois élèves qui ont le plus d'absences. Ranger les élèves du plus absent au moins absent, c'est un tri. Mais la direction ne veut pas la liste entière : seulement le haut de la liste.

> 💡 Télécharge `mon-ecole-eleves.db` dans la sidebar, ouvre-la dans DB Browser for SQLite, puis va dans l'onglet **Exécuter le SQL**. Tape chaque requête dans la zone du haut et exécute-la avec ▶ ou Ctrl+Entrée.

🎯 Objectifs

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

1. limiter le nombre de lignes d'un résultat avec `LIMIT` ;
2. expliquer pourquoi `LIMIT` doit presque toujours accompagner un `ORDER BY` ;
3. départager les égalités avant de couper un résultat ;
4. écrire les clauses d'une requête dans l'ordre imposé ;
5. traduire une question en français en requête, étape par étape.

Garder les premières lignes

La base contient une table `eleve`, avec une ligne par élève. Pour la lire, on écrit une **requête** : `SELECT` (*sélectionne*) donne les colonnes à afficher, `FROM` (*depuis*) donne la table où les chercher. `ORDER BY` (*ordonner par*) trie les lignes, et `DESC` rend le tri décroissant.

Pour les trois élèves les plus absents, on trie d'abord, puis on ajoute `LIMIT 3` à la fin :

```
SELECT prenom, nom, absences
FROM eleve
```

```sql
ORDER BY absences DESC
LIMIT 3;
```

```
prenom | nom       | absences
-------+-----------+---------
Louis  | Istas     | 12
Hugo   | Englebert | 8
Noah   | Charlier  | 5
```

`LIMIT` veut dire *limite*. SQLite construit d'abord le résultat complet et le trie, puis il ne garde que les trois premières lignes.

> **📖 NOUVELLE NOTION : LIMIT**
>
> `LIMIT n` (*limite*) ne garde que les `n` premières lignes du résultat. La limite s'applique **en dernier**, après le filtre et le tri. Si le résultat contient moins de `n` lignes, il est renvoyé en entier, sans erreur.
>
> En anglais : GB ***limit***.

Le même principe donne l'élève le plus âgé : on trie par date de naissance, la plus ancienne en premier, et on garde une ligne.

```sql
SELECT prenom, nom, date_naissance
FROM eleve
ORDER BY date_naissance
LIMIT 1;
```

```
prenom | nom      | date_naissance
-------+----------+---------------
Chloé  | Fontaine | 2009-09-08
```

LIMIT sans ORDER BY ne veut rien dire

Retire le `ORDER BY` de la première requête :

```sql
SELECT prenom, nom, absences
FROM eleve
LIMIT 3;
```

```
prenom | nom      | absences
-------+----------+---------
Lucas  | Adam     | 2
Emma   | Bastin   | 0
Noah   | Charlier | 5
```

La requête fonctionne, mais le résultat ne répond à aucune question. Ce sont « les trois premières lignes » dans l'ordre où SQLite a lu la table. Et sans `ORDER BY`, cet ordre n'est pas garanti : il peut changer après une modification de la table.

🧠 « Les trois premiers » n'a de sens que si tu dis **SELON QUEL ORDRE**. `LIMIT` s'écrit presque toujours avec un `ORDER BY` .

Attention aux égalités

« Les trois élèves qui ont le moins d'absences » : la question paraît simple. Mais quatre élèves ont zéro absence. Lesquels garder ?

Sans consigne, SQLite choisit trois des quatre, et rien ne te garantit lesquels. Pour que le résultat soit toujours le même, ajoute une deuxième colonne de tri qui départage les égalités :

```sql
SELECT prenom, nom, absences
FROM eleve
ORDER BY absences, nom
LIMIT 3;
```

```
prenom | nom      | absences
-------+----------+---------
Emma   | Bastin   | 0
Chloé  | Fontaine | 0
Manon  | Hubert   | 0
```

Le résultat est maintenant prévisible. Il reste un problème de fond : Sarah Lambert a aussi zéro absence, et elle est écartée uniquement à cause de son nom. La question « les trois moins absents » n'a pas de bonne réponse ici. Un résultat correct sur le plan technique peut rester trompeur.

⚠️ Avant de couper un résultat avec `LIMIT` , regarde s'il y a des égalités à la frontière. Si oui, décide comment les départager, ou signale-le à celui qui a posé la question.

ℹ️ `OFFSET` , qui veut dire *décalage*, saute un certain nombre de lignes avant d'appliquer la limite. `ORDER BY nom LIMIT 3 OFFSET 3` renvoie les lignes 4 à 6 : Léa Delvaux, Hugo Englebert et Chloé Fontaine. C'est ce qu'utilisent les sites web pour afficher un long résultat page par page.

L'ordre imposé des clauses

Une requête peut maintenant contenir cinq parties. Chacune commence par un mot-clé et joue un rôle précis.

📖 NOUVELLE NOTION : LA CLAUSE

Une **clause** est un morceau de requête qui commence par un mot-clé : `SELECT …`, `FROM …`, `WHERE …`, `ORDER BY …`, `LIMIT …`. Chaque clause a un rôle précis. Les clauses ne sont pas toutes obligatoires, mais celles qui sont présentes doivent apparaître dans un **ordre imposé**.

En anglais : GB ***clause***.

```sql
SELECT ...      -- ce que je veux voir
FROM ...        -- où sont les données
WHERE ...       -- quelles lignes
ORDER BY ...    -- dans quel ordre
LIMIT ...;      -- combien
```

Si tu inverses deux clauses, SQLite refuse la requête. Place `LIMIT` avant `ORDER BY` :

```sql
SELECT prenom, nom
FROM eleve
LIMIT 3
ORDER BY nom;
```

```
near "ORDER": syntax error
```

Même chose avec `ORDER BY` placé avant `WHERE` :

```sql
SELECT prenom, nom
FROM eleve
ORDER BY nom
WHERE commune = 'Wavre';
```

```
near "WHERE": syntax error
```

Le message se lit *erreur de syntaxe près de…* : SQLite indique le mot où il s'est perdu. C'est souvent la clause mal placée.

Clause	Traduction	Rôle	Obligatoire ?
`SELECT`	sélectionne	les colonnes à afficher	oui
`FROM`	depuis	la table où chercher	oui, dès qu'on lit une table
`WHERE`	où	les lignes à garder	non
`ORDER BY`	ordonner par	l'ordre des lignes	non
`LIMIT`	limite	le nombre de lignes	non

💡 L'ordre des clauses suit l'ordre naturel des questions qu'on se pose : qu'est-ce que je veux voir, où, lesquelles, dans quel ordre, combien. Retiens les questions, et l'ordre des clauses suit.

Traduire une question en requête

Écrire une requête d'un seul jet est difficile. La méthode suivante découpe la question en cinq petites questions, une par clause.

Question à se poser	Clause
Qu'est-ce que je veux voir ?	`SELECT`
Où sont les données ?	`FROM`
Quelles lignes ?	`WHERE`
Dans quel ordre ?	`ORDER BY`
Combien ?	`LIMIT`

Applique-la à cette demande :

> « Donne-moi le prénom, le nom et le nombre d'absences des deux élèves de Jodoigne qui ont le plus d'absences. »

Qu'est-ce que je veux voir ?

Le prénom, le nom et le nombre d'absences.

```
SELECT prenom, nom, absences
```

Où sont les données ?

Toutes ces colonnes sont dans la table `eleve`.

```
FROM eleve
```

Quelles lignes ?

Seulement les élèves de Jodoigne.

```
WHERE commune = 'Jodoigne'
```

Dans quel ordre ?

« Ceux qui ont le plus d'absences » : on range du plus absent au moins absent.

```sql
ORDER BY absences DESC
```

Combien ?

Deux élèves.

```sql
LIMIT 2
```

On assemble les cinq morceaux dans l'ordre, et on exécute :

```sql
SELECT prenom, nom, absences
FROM eleve
WHERE commune = 'Jodoigne'
ORDER BY absences DESC
LIMIT 2;
```

```
prenom | nom   | absences
-------+-------+---------
Louis  | Istas | 12
Lucas  | Adam  | 2
```

⚠️ Dans cette base, la commune de Sarah Lambert a été encodée `jodoigne`, en minuscules. `WHERE commune = 'Jodoigne'` ne la trouve donc pas. Ici, le résultat n'est pas faussé, parce qu'elle n'a aucune absence. Mais sur une autre question, cette faute de frappe pourrait changer la réponse sans que tu le remarques.

🧪 Exercices

Exercice 1 — Les premiers et les derniers ★☆☆☆☆

Écris une requête qui affiche :

1. le prénom et le nom de l'élève le plus jeune ;
2. les deux derniers élèves par ordre alphabétique de nom ;
3. les quatre élèves les plus âgés, avec leur date de naissance ;
4. l'élève de Hannut qui a le plus d'absences.

Exercice 2 — Prédire le résultat ★★☆☆☆

Sans exécuter, écris le résultat de chaque requête. Exécute ensuite pour vérifier.

```sql
-- a)
SELECT prenom, nom FROM eleve ORDER BY nom DESC LIMIT 2;

-- b)
SELECT prenom, nom, absences FROM eleve WHERE absences > 0 ORDER BY absences LIMIT 4;

-- c)
SELECT prenom, nom FROM eleve ORDER BY nom LIMIT 3 OFFSET 6;

-- d)
SELECT prenom, nom FROM eleve WHERE commune = 'Namur' LIMIT 3;
```

Exercice 3 — Réparer des requêtes ★★★☆☆

Chacune de ces requêtes provoque une erreur ou ne répond pas à la question. Pour chacune : exécute-la, observe, explique ce qui ne va pas, puis corrige-la.

```sql
-- a) Les trois premiers élèves par ordre alphabétique
SELECT prenom, nom FROM eleve LIMIT 3 ORDER BY nom;

-- b) Les élèves de Wavre, par ordre alphabétique
SELECT prenom, nom FROM eleve ORDER BY nom WHERE commune = 'Wavre';

-- c) Les trois élèves les plus absents
SELECT prenom, nom, absences FROM eleve LIMIT 3;

-- d) Les trois élèves les plus absents
SELECT prenom, nom, absences FROM eleve ORDER BY absences LIMIT 3;

-- e) Le nom des trois premiers élèves
SELECT LIMIT 3 nom FROM eleve;
```

Exercice 4 — Cinq questions à traduire sur papier ★★★☆☆

Cet exercice se fait **sans ordinateur**. Pour chaque question, recopie la grille sur ta feuille, remplis-la, puis écris la requête complète au crayon, clauses dans le bon ordre.

Question à se poser	Ta réponse	Clause
Qu'est-ce que je veux voir ?		`SELECT`
Où sont les données ?		`FROM`
Quelles lignes ?		`WHERE`
Dans quel ordre ?		`ORDER BY`
Combien ?		`LIMIT`

1. Le prénom et le nom des élèves de Wavre, par ordre alphabétique de nom.
2. Le nom, le prénom et la date de naissance des trois élèves nés en 2010 les plus âgés.
3. Le nom et le nombre d'absences de l'élève de Hannut le plus absent.
4. Le prénom, le nom et l'email des élèves sans adresse email, triés par nom. La condition « n'a pas d'adresse » s'écrit `email IS NULL`.
5. Les cinq élèves qui ont le plus d'absences ; en cas d'égalité, par ordre alphabétique de nom.

Quand tes cinq requêtes sont écrites, passe à l'ordinateur et vérifie-les une par une. Pour chaque requête fausse, note ce qui n'allait pas : clause oubliée, mal placée, condition incorrecte…

✍️ À retenir

- `LIMIT n` (*limite*) ne garde que les `n` premières lignes, après le filtre et le tri.
- `LIMIT` sans `ORDER BY` renvoie des lignes au hasard de la lecture : « les premiers » n'a pas de sens.
- En cas d'égalité à la frontière, ajoute une colonne de tri pour départager.
- Ordre imposé des clauses : `SELECT`, `FROM`, `WHERE`, `ORDER BY`, `LIMIT`.
- Une clause mal placée provoque une erreur `near "…": syntax error`.
- Pour écrire une requête : qu'est-ce que je veux voir, où sont les données, quelles lignes, dans quel ordre, combien.

Suite

Parfois, la question ne porte pas sur les élèves eux-mêmes, mais sur les valeurs d'une colonne : « dans quelles communes habitent-ils ? ». Pour obtenir chaque valeur une seule fois, passe à Éliminer les doublons avec DISTINCT.