Une seule table ne suffit pas

Tout ranger dans une seule table, comme dans un tableur, semble la solution la plus simple. On la teste sur les données d'une école et on mesure les dégâts : redondance, anomalies et fautes invisibles.

4TTR 5TTR 6TTR Découverte

L'école veut connaître la classe de chaque élève, avec le local de cette classe et le professeur titulaire. Le réflexe naturel consiste à tout ranger dans un seul tableau. Dans cet article, tu testes cette solution sur de vraies données et tu regardes précisément ce qui casse.

> 💡 **BASE DE TRAVAIL**
>
> Télécharge `bulletin-plat.db` dans la carte « Téléchargements » de la sidebar, puis ouvre-le avec DB Browser for SQLite. Ce fichier suffit pour suivre tout l'article.

🎯 Objectifs

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

1. **Reconnaître** la redondance dans une table.
2. **Nommer** les problèmes qu'elle provoque : les anomalies de modification, d'insertion et de suppression, et les incohérences silencieuses.
3. **Détecter** des incohérences avec `SELECT DISTINCT`.
4. **Repérer** plusieurs entités mélangées dans une même ligne.
5. **Expliquer** la règle « une entité = une table, un fait écrit à un seul endroit ».

Tout dans une seule table

La demande du secrétariat tient en une phrase :

> « Ajoutez la classe de chaque élève, avec son local et le professeur titulaire. »

La solution la plus rapide consiste à tout mettre dans un seul tableau, comme on le ferait dans un tableur. C'est ce que contient le fichier `bulletin-plat.db`. Ouvre l'onglet *Parcourir les données* et choisis la table `bulletin_plat` :

```
eleve_nom | eleve_prenom | classe | classe_local | titulaire_nom      | titulaire_email
----------+--------------+--------+--------------+--------------------+---------------------
Adam      | Lucas        | 4TTR   | B14          | Lambrechts Ludovic | l.lambrechts@ecole.be
Bastin    | Emma         | 4TTR   | B14          | Lambrechts Ludovic | l.lambrechts@ecole.be
Charlier  | Noah         | 4TTR   | B14          | Lambrechts Ludovic | l.lambrechts@ecole.be
Delvaux   | Léa          | 4TRR   | B14          | Lambrechts Ludovic | l.lambrechts@ecole.be
Englebert | Hugo         | 4TTR   | B14          | Lambrechts Ludovic | l.lambrechts@ecole.be
Fontaine  | Chloé        | 5TTR   | A21          | Dubois Sophie      | s.dubois@ecole.be
Gérard    | Nathan       | 5TTR   | A21          | Dubois Sophie      | s.dubois@ecole.be
Hubert    | Manon        | 5TTR   | A21          | Dubois Sophie      | s.dubois@ecoles.be
Istas     | Louis        | 5TTR   | A21          | Dubois Sophie      | s.dubois@ecole.be
Jacques   | Camille      | 6TTR   | A23          | Vermeulen Marc     | m.vermeulen@ecole.be
Kevers    | Arthur       | 6TTR   | A23          | Vermeulen Marc     | m.vermeulen@ecole.be
```

Toutes les informations demandées sont là. Et pourtant, **deux erreurs se sont déjà glissées dans ces onze lignes**. Avant de lire la suite, essaie de les trouver à l'œil. Note ce que tu trouves : on vérifiera plus bas.

Le même fait écrit plusieurs fois

Compte combien de fois l'adresse `l.lambrechts@ecole.be` est écrite : cinq fois. Et le local `B14` ? Cinq fois aussi. Pourtant, Ludovic Lambrechts n'a qu'une seule adresse, et la classe 4TTR n'a qu'un seul local.

> 📖 **NOUVELLE NOTION : LA REDONDANCE**
>
> Il y a **redondance** quand un même fait est écrit à plusieurs endroits d'une base. Ici, « le local de la 4TTR est le B14 » est écrit cinq fois, une fois par élève de la classe.
>
> En anglais : GB ***redundancy***.

On pourrait croire que la redondance fait seulement perdre de la place. C'est faux : elle rend la base fragile. Chaque fois qu'on ajoute, modifie ou supprime des données, on risque de créer une contradiction.

> 📖 **NOUVELLE NOTION : L'ANOMALIE**
>
> Une **anomalie** est un problème qui apparaît quand on ajoute, modifie ou supprime des données dans une table qui mélange plusieurs choses. La base ne signale rien : elle accepte l'opération, mais le résultat est incomplet ou contradictoire.
>
> En anglais : GB ***anomaly***.

On distingue trois anomalies, une par type d'opération. Un quatrième problème, plus sournois, les accompagne.

Modifier : l'anomalie de modification

Ludovic Lambrechts change d'adresse e-mail. Dans ce tableau, il faut modifier **cinq lignes**. Avec vingt-cinq élèves par classe, il en faudrait vingt-cinq.

Si tu en oublies une, la base contient deux adresses différentes pour la même personne. Plus rien ne dit laquelle est la bonne.

> 🧠 **L'ANOMALIE DE MODIFICATION**
>
> Un seul changement doit être répété sur plusieurs lignes. Si on en oublie une, la base se contredit elle-même, sans aucun message d'erreur.

Ajouter : l'anomalie d'insertion

L'école ouvre une classe de 3TTR pour la rentrée prochaine, dans le local B10, avec Anne Leroy comme titulaire. Aucun élève n'y est encore inscrit.

Où enregistres-tu cette classe ? Nulle part. Dans ce tableau, chaque ligne décrit un élève : **une classe n'existe que si un élève y est inscrit**. Tu pourrais créer une ligne en laissant vides les colonnes de l'élève, mais tu obtiendrais un élève sans nom, qui fausserait tous tes comptages.

> 🧠 **L'ANOMALIE D'INSERTION**
>
> On ne peut pas enregistrer une information sans en inventer une autre qui n'a rien à voir avec elle.

Supprimer : l'anomalie de suppression

Camille Jacques et Arthur Kevers quittent l'école. Ce sont les deux seuls élèves de 6TTR : tu supprimes leurs deux lignes.

Du même coup, tu effaces la classe 6TTR, son local A23 et le fait que Marc Vermeulen en est le titulaire. Personne n'a demandé ça.

> 🧠 **L'ANOMALIE DE SUPPRESSION**
>
> Effacer une information en efface une autre au passage.

Les incohérences silencieuses

Revenons aux deux erreurs du début. Lire les lignes une à une est lent, et l'œil se trompe vite. Il est plus sûr de demander à la base la liste des valeurs différentes d'une colonne.

> ℹ️ **RAPPEL : `SELECT DISTINCT`**

`DISTINCT` veut dire *distinct*, *différent*. `SELECT DISTINCT colonne FROM table` affiche chaque valeur de la colonne une seule fois, sans répétition.

Commence par la colonne `classe` :

```sql
SELECT DISTINCT classe FROM bulletin_plat ORDER BY classe;
```

```
classe
------
4TRR
4TTR
5TTR
6TTR
```

L'école compte trois classes, mais la base en affiche quatre. `4TRR` est une faute de frappe : c'est la ligne de Léa Delvaux. Recommence avec les adresses des titulaires :

```sql
SELECT DISTINCT titulaire_email FROM bulletin_plat ORDER BY titulaire_email;
```

```
titulaire_email
--------------------
l.lambrechts@ecole.be
m.vermeulen@ecole.be
s.dubois@ecole.be
s.dubois@ecoles.be
```

Trois titulaires, mais quatre adresses : Sophie Dubois en a deux. La ligne de Manon Hubert contient `ecoles` au lieu de `ecole`.

💡 UN RÉFLEXE DE CONTRÔLE

Sur une colonne qui ne devrait contenir que quelques valeurs (des classes, des locaux, des adresses de professeurs), lance un `SELECT DISTINCT`. Toute valeur en trop est suspecte.

Ces fautes ont des conséquences bien réelles. Demande la liste des élèves de 4TTR :

```sql
SELECT eleve_prenom, eleve_nom FROM bulletin_plat WHERE classe = '4TTR';
```

```
eleve_prenom | eleve_nom
-------------+----------
Lucas        | Adam
Emma         | Bastin
Noah         | Charlier
Hugo         | Englebert
```

La base répond sans erreur, mais Léa Delvaux n'apparaît pas. Si le secrétariat imprime les bulletins de la 4TTR à partir de cette liste, Léa n'aura pas le sien.

> 🧠 **LES INCOHÉRENCES SILENCIEUSES**
>
> Aucune des deux fautes n'a provoqué de message d'erreur. La base les a acceptées, parce que rien ne lui dit que la classe `4TRR` n'existe pas. La redondance ne crée pas les fautes de frappe : elle leur laisse la place de se glisser, puis elle les cache.

Une ligne qui raconte plusieurs choses

Pourquoi tous ces problèmes ? Regarde de près la ligne de Lucas Adam :

```
Adam | Lucas | 4TTR | B14 | Lambrechts Ludovic | l.lambrechts@ecole.be
└─ élève ──┘   └ classe ┘   └──────────────── professeur ─────────────────┘
```

Cette ligne raconte **trois choses différentes** : un élève, une classe et un professeur. Dans une base de données, chacune de ces choses est une **entité** : une chose dont on veut garder la trace, décrite par ses attributs (un nom, un local, une adresse…).

Or chaque entité vit sa propre vie. Un élève change de classe, une classe change de titulaire, un professeur change d'adresse. Quand on les mélange dans une même ligne, chaque changement de l'une oblige à toucher aux données des autres.

> 🧠 **LA RÈGLE QUI DÉCOULE DE TOUT ÇA**
>
> **Une entité = une table. Un fait est écrit à un seul endroit.**
>
> S'il faut écrire l'adresse de Ludovic Lambrechts cinq fois, c'est qu'il manque une table `professeur`. S'il faut écrire `B14` cinq fois, c'est qu'il manque une table `classe`.

Découper le tableau en trois tables règle la redondance. Mais une nouvelle question apparaît aussitôt : si les élèves sont dans une table et les classes dans une autre, comment la base sait-elle que Lucas Adam est en 4TTR ?

🧪 Exercices

Tous les exercices portent sur `bulletin-plat.db`. Ne corrige pas les fautes : le but est de mesurer les dégâts. L'exercice 3 modifie la base : fais-le sur une copie du fichier, ou télécharge-le à nouveau ensuite.

Exercice 1 — Chasse aux incohérences ★★☆☆☆

1. Lance un `SELECT DISTINCT` sur la colonne `classe_local`. Combien de valeurs obtiens-tu ? Ce résultat révèle-t-il la faute `4TRR` ?
2. Fais de même sur la colonne `titulaire_nom`. Pourquoi cette colonne ne révèle-t-elle **pas** la faute dans l'adresse de Sophie Dubois ?
3. Que penses-tu d'un contrôle qui ne porterait que sur une seule colonne ?

Exercice 2 — Mesurer les dégâts ★★☆☆☆

1. Écris la requête qui affiche le prénom et le nom des élèves dont le titulaire a l'adresse `s.dubois@ecole.be`. Combien en trouves-tu ?
2. Combien devrait-il y en avoir ? Qui manque ?
3. Sophie Dubois envoie un message à chaque élève de sa classe à partir de cette liste. Que se passe-t-il pour l'élève oublié ?
4. Écris une requête qui retrouve les quatre élèves malgré la faute. Pourquoi n'est-ce pas une vraie solution ?

Exercice 3 — Simuler un changement ★★★☆☆

> ℹ️ **RAPPEL : UPDATE**
>
> `UPDATE` veut dire *mettre à jour* et `SET` veut dire *régler*. `UPDATE table SET colonne = nouvelle_valeur WHERE condition;` modifie les lignes où la condition est vraie ; sans `WHERE`, toutes les lignes sont modifiées.

Sophie Dubois change d'adresse e-mail. Sa nouvelle adresse est `s.dubois-renard@ecole.be`.

1. Combien de lignes faudrait-il modifier dans `bulletin_plat` ?
2. Écris la requête `UPDATE` qui remplace l'adresse `s.dubois@ecole.be` par la nouvelle, puis exécute-la. DB Browser affiche le nombre de lignes modifiées : note-le.
3. Relance le `SELECT DISTINCT` sur `titulaire_email`. Que constates-tu ? Explique pourquoi.
4. Combien de lignes aurait-il fallu modifier si les professeurs avaient leur propre table ?

Exercice 4 — Les trois anomalies ★★★☆☆

Pour chaque demande, écris ce que tu tenterais, puis explique précisément ce qui bloque et à quelle anomalie cela correspond.

1. Enregistrer la nouvelle classe 3TTR (local B10, titulaire Anne Leroy), qui n'a encore aucun élève.
2. Supprimer les deux élèves de 6TTR sans perdre l'existence de la classe.
3. Enregistrer Julie Martin, une professeure qui n'est titulaire d'aucune classe.

Exercice 5 — Proposer un découpage ★★★☆☆

Sans écrire de SQL, sur papier :

1. Combien de tables faudrait-il, et lesquelles ?
2. Pour chaque table, écris la liste de ses colonnes.
3. Vérifie ta proposition : chaque fait n'est-il plus écrit **qu'une seule fois** ?
4. Ta proposition doit encore dire dans quelle classe se trouve Lucas Adam. Comment ferais-tu ? Note ton idée, même incomplète.

✍️ À retenir

- La **redondance**, c'est un même fait écrit à plusieurs endroits.
- Elle provoque des **anomalies** : de **modification** (une ligne oubliée), d'**insertion** (impossible d'enregistrer une information seule), de **suppression** (effacer une information en efface une autre).
- Elle permet aussi des **incohérences silencieuses** : des fautes de frappe que la base accepte sans aucun message d'erreur.
- `SELECT DISTINCT colonne` liste les valeurs différentes d'une colonne : c'est un outil simple pour repérer les fautes.
- Une ligne qui raconte plusieurs choses à la fois mélange plusieurs entités.
- La règle : **une entité = une table, un fait écrit à un seul endroit**.

Suite

Découper les données en plusieurs tables supprime la redondance, mais il faut alors relier chaque élève à sa classe. C'est l'objet de l'article Relier deux entités.