# Relier deux entités

Repérer le lien entre deux entités dans un texte, l'écrire avec ses cardinalités, le traduire en clé étrangère, puis le poser dans la base pour qu'elle refuse les incohérences.

4TTR 5TTR 6TTR Découverte

Dans la base d'une école, les élèves ont leur table et les classes vont avoir la leur. Il reste à dire quel élève est dans quelle classe. Cet article montre comment repérer ce lien dans un texte, comment l'écrire sur papier, puis comment le poser dans la base pour qu'elle refuse les incohérences.

> 💡 **BASE DE TRAVAIL**
>
> Télécharge `mon-ecole-relier.db` dans la carte « Téléchargements » de la sidebar et ouvre-la avec DB Browser for SQLite. Elle contient deux tables : `eleve` (12 élèves) et `professeur` (6 professeurs). En suivant l'article jusqu'au bout, tu obtiendras exactement la base `mon-ecole-classes.db` .

## 🎯 Objectifs

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

1. **Repérer** une association dans un texte avec la méthode sujet – verbe – complément.
2. **Écrire et lire** les cardinalités d'une association, en justifiant chaque chiffre par une phrase.
3. **Traduire** un MCD en MLD, en plaçant la clé étrangère du bon côté.
4. **Créer** la table `classe` et la clé étrangère `classe_id` dans ta base.
5. **Activer** la vérification des clés étrangères et **constater** ce que la base refuse.

## Pourquoi relier deux tables

Quand on range les élèves, leur classe et le local de cette classe dans un seul tableau, le même local est recopié sur la ligne de chaque élève. La moindre faute de frappe crée alors une contradiction que personne ne voit. La solution consiste à donner à chaque entité sa propre table : une table `eleve` , une table `classe` .

Pour rappel, une **entité** est une chose dont on veut garder la trace, comme un élève ou une classe. Dans la base, chaque entité devient une table, et chaque exemplaire concret (Lucas Adam, la 4TTR) devient une ligne. Dans le vocabulaire de l'analyse, cet exemplaire s'appelle une **occurrence**.

Une question se pose 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 ? Pour y répondre, on commence par l'analyse, sur papier. On passe à la base ensuite.

## Trouver les associations dans un texte

Voici comment le directeur décrit son école :

> « Chaque élève appartient à une classe. Une classe occupe un local. Chaque classe a un professeur titulaire. »

Pour trouver les liens entre les entités, une méthode simple suffit.

> 💡 **LA MÉTHODE SUJET – VERBE – COMPLÉMENT**
>
> Cherche les phrases dont le **sujet** et le **complément** sont deux entités. Le **verbe** qui les relie désigne très probablement un lien entre elles.

Découpe le texte du directeur :

| Sujet | Verbe | Complément | Conclusion |
|---|---|---|---|
| élève | appartient à | classe | un lien entre ELEVE et CLASSE |
| classe | a pour titulaire | professeur | un lien entre CLASSE et PROFESSEUR |
| classe | occupe | local | le local est-il une entité ? |

Les deux premières lignes relient deux entités. La troisième demande réflexion. Un local est-il une chose dont on veut garder la trace pour elle-même, avec sa capacité, son étage et son équipement ? Ou juste un code écrit sur la classe ? Pour l'école, le code suffit : `local` reste un simple **attribut** de la classe.

> 📖 **NOUVELLE NOTION : L'ASSOCIATION**
>
> Une **association** est un lien entre deux entités. On la repère grâce au verbe qui les relie, et on lui donne ce verbe pour nom, conjugué à la 3e personne : `appartient`.
>
> En anglais : GB ***relationship***.

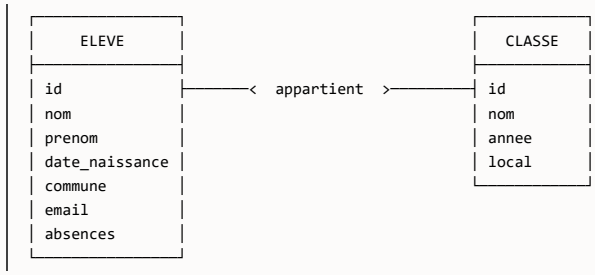
> ⚠️ **ASSOCIATION, PAS RELATION**
>
> Tu entendras parfois le mot « relation » pour désigner ce lien. Dans ce parcours, on dit toujours **association** : dans le vocabulaire des bases de données, une relation désigne une table, et les deux sens se mélangeraient.

Si tu ne trouves aucun verbe entre deux mots, méfie-toi : l'un des deux n'est peut-être pas une entité, mais un attribut de l'autre, comme le local.

Dans cet article, on traite l'association entre l'élève et sa classe. Le titulaire fera l'objet d'un exercice.

## Dessiner les entités et l'association

On dessine chaque entité dans un rectangle, avec son nom en MAJUSCULES et ses attributs en dessous. L'association se dessine dans un losange, entre les deux rectangles :




L'attribut `id` est l'**identifiant** de chaque entité : il désigne une occurrence et une seule. Dans la base, il deviendra la clé primaire.

Ce dessin est un **schéma entité-association** : c'est lui que tu réalises, sur papier, chaque fois que tu analyses un nouveau domaine.

> 📖 **NOUVELLE NOTION : LE SCHÉMA ENTITÉ-ASSOCIATION**
>
> Le **schéma entité-association** est le dessin qui représente les données d'un domaine : chaque **entité** est un rectangle qui contient ses attributs, et chaque **association** est un losange, nommé par un verbe, qui relie deux entités. On y ajoute ensuite les cardinalités.
>
> En anglais : GB ***entity-relationship diagram***, souvent abrégé ERD.

> 📖 **NOUVELLE NOTION : LE MCD**
>
> Le **MCD**, ou **modèle conceptuel de données**, est la description des entités, de leurs attributs et des associations qui les relient ; on le représente par un schéma entité-association. Il décrit ce que l'on veut retenir, sans se préoccuper du logiciel qui stockera les données.
>
> En anglais : GB ***conceptual data model***.

> 📖 **NOUVELLE NOTION : MERISE**
>
> **Merise** est une méthode française de conception de bases de données, née à la fin des années 1970. Elle fixe la manière de dessiner le schéma entité-association : des rectangles pour les entités, des losanges pour les associations, et deux chiffres de chaque côté de l'association. C'est la notation utilisée dans ce parcours.
>
> Merise est un nom propre : il ne se traduit pas.

# Les cardinalités

Il manque encore une information sur le dessin : combien de classes peut avoir un élève ? Et combien d'élèves peut avoir une classe ? Pour le savoir, on pose deux questions de chaque côté de l'association.

Place-toi du côté de l'élève :

- Un élève appartient **au minimum** à combien de classes ? **0** : un élève qui vient d'arriver n'a pas forcément encore de classe.

- Un élève appartient **au maximum** à combien de classes ? **1** : on n'est pas dans deux classes à la fois.

Place-toi maintenant du côté de la classe :

- Une classe a **au minimum** combien d'élèves ? **0** : une classe qui vient d'ouvrir est vide.
- Une classe a **au maximum** combien d'élèves ? **Plusieurs**, sans nombre fixé : on écrit **n**.

Côté élève, on écrit donc `(0,1)`. Côté classe, on écrit `(0,n)`.

> 📖 **NOUVELLE NOTION : LA CARDINALITÉ**
>
> La **cardinalité** est le couple de chiffres écrit à côté d'une entité, de part et d'autre d'une association. Le premier chiffre est le **minimum**, le second le **maximum** : ils disent à combien d'occurrences de l'autre entité **une** occurrence de cette entité peut être liée.
>
> En anglais : GB ***cardinality***.

Il n'existe que quatre cardinalités :

| Cardinalité | Minimum | Maximum | Se lit |
|---|---|---|---|
| `(0,1)` | 0 | 1 | au plus un, éventuellement aucun |
| `(1,1)` | 1 | 1 | exactement un |
| `(0,n)` | 0 | plusieurs | autant qu'on veut, éventuellement aucun |
| `(1,n)` | 1 | plusieurs | au moins un, autant qu'on veut |

Elles s'écrivent toujours entre parenthèses, avec un `n` minuscule.

## Écrire et lire une cardinalité

Sur une ligne de texte, l'association entre l'élève et sa classe s'écrit ainsi :

```
ELEVE (0,1) —appartient— (0,n) CLASSE
```

Les chiffres collés à `ELEVE` parlent d'**un élève**. Les chiffres collés à `CLASSE` parlent d'**une classe**. Pour ne jamais te tromper, transforme chaque cardinalité en phrase.

> 🧠 **LA RÈGLE DE LECTURE**
>
> Commence toujours ta phrase par « **un** » suivi du nom de l'entité **du côté où les chiffres sont écrits**.
>
> - `(0,1)` à côté de ELEVE : « **Un élève** appartient au minimum à 0 classe et au maximum à 1 classe. »
> - `(0,n)` à côté de CLASSE : « **Une classe** a au minimum 0 élève et au maximum n élèves qui lui appartiennent. »

> ⚠️ **LE PIÈGE : LIRE LA CARDINALITÉ À L'ENVERS**
>
> Le `(0,1)` écrit à côté de ELEVE ne veut pas dire « il y a zéro ou un élève ». Il parle d'**un** élève, et compte ses classes. Si ta phrase ne commence pas par « un élève », tu es en train de la lire à l'envers.

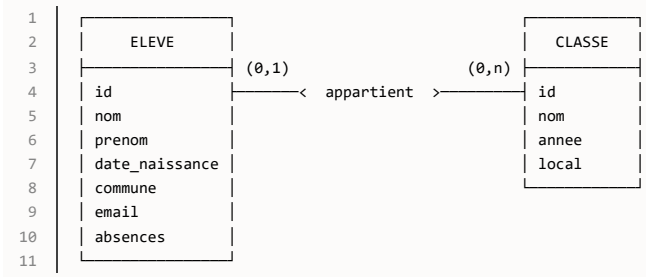
## Le minimum est une décision de gestion

Le maximum découle souvent du bon sens : un élève n'est pas dans deux classes. Le minimum, lui, traduit un choix de l'école. Compare :

- Côté ELEVE `(0,1)` : l'école **accepte** d'enregistrer un élève qui n'a pas encore de classe.
- Côté ELEVE `(1,1)` : l'école **refuse**. Tout élève doit recevoir une classe dès son inscription.

Les deux choix sont défendables. Ce qui ne l'est pas, c'est de choisir sans y penser. Ici, l'école garde `(0,1)` : Sarah Lambert est arrivée en cours d'année et n'a pas encore de classe.

Voici le MCD complet, avec ses cardinalités :



📖 NOUVELLE NOTION : L'ASSOCIATION 1-N

Une association est dite **1-N** (« un à plusieurs ») quand le maximum vaut 1 d'un côté et n de l'autre. Ici, un élève a au maximum une classe, et une classe a au maximum plusieurs élèves. Seuls les maximums comptent pour lui donner ce nom.

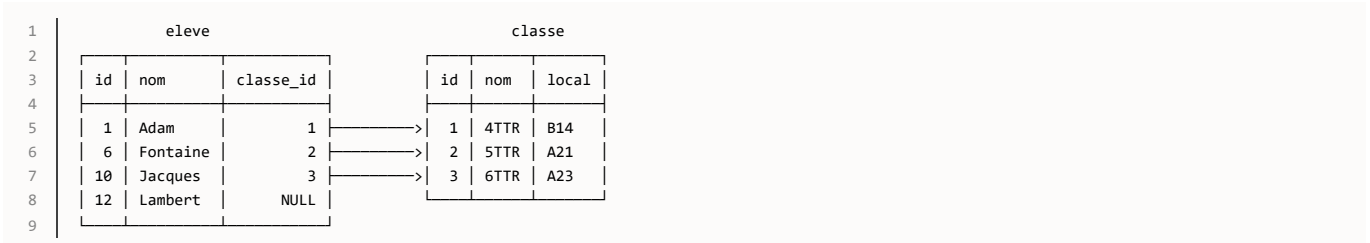
En anglais : GB ***one-to-many relationship***.

---

# De l'association à la clé étrangère

Le MCD dit ce qu'on veut retenir. Il faut maintenant le traduire en tables. Pour relier un élève à sa classe, on ajoute dans la table `eleve` une colonne qui contient l'identifiant de sa classe.

Pour rappel, la **clé primaire** d'une table est la colonne qui désigne une ligne et une seule. Dans chacune de nos tables, c'est la colonne `id`. Voici un extrait des deux tables une fois reliées :




La valeur `1` sur la ligne de Lucas Adam désigne la ligne de `classe` dont l'`id` vaut 1 : la 4TTR. Sarah Lambert n'a pas de classe : sa case est vide (`NULL`). C'est le « 0 » de la cardinalité `(0,1)`.

📖 NOUVELLE NOTION : LA CLÉ ÉTRANGÈRE

Une **clé étrangère** est une colonne qui contient la clé primaire d'une ligne d'une **autre** table. Elle fonctionne comme une adresse : la valeur `1` dans `eleve.classe_id` permet de retrouver la classe dont l'`id` vaut 1.

En anglais : GB ***foreign key***.

## De quel côté placer la clé étrangère

On aurait pu, au contraire, ajouter une colonne `eleve_id` dans la table `classe`. Pour choisir, une seule règle suffit, et elle se lit dans les cardinalités.

🧠 LA RÈGLE DE PLACEMENT

La clé étrangère va dans la table de l'entité qui porte le maximum 1 — celui qui n'en a qu'un garde l'adresse de l'autre.

Côté ELEVE, la cardinalité est `(0,1)` : son maximum vaut 1. Un élève n'a qu'une classe, il garde donc l'adresse de sa classe. La clé étrangère `classe_id` va dans la table `eleve`.

Pourquoi pas de l'autre côté ? La 4TTR compte cinq élèves. Avec une colonne `eleve_id` dans `classe`, la ligne de la 4TTR devrait contenir cinq identifiants dans une seule case, ce qui est impossible. L'autre possibilité serait de recopier cinq fois la ligne de la 4TTR, avec son local : on retomberait sur la redondance du départ.

⚠️ LE PIÈGE : LA CLÉ ÉTRANGÈRE DU MAUVAIS CÔTÉ

Ne te fie pas à l'intuition (« la classe est plus importante, donc c'est elle qui reçoit la colonne »). Regarde uniquement les maximums : la clé étrangère va du côté du **1**, jamais du côté du **n**.

💡 NOMMER LA CLÉ ÉTRANGÈRE

La colonne s'appelle `classe_id` : le nom de la table désignée, suivi de `_id`. En lisant ce nom, on sait tout de suite ce que contient la colonne et vers quelle table elle mène.

## Écrire le MLD

Pour décrire les tables sans les dessiner, on les écrit sous forme de liste. Voici celle de notre école :

- classe (id, nom, annee, local)
- eleve (id, nom, prenom, date_naissance, commune, email, absences, #classe_id → classe.id)

Chaque ligne donne le nom d'une table, puis ses colonnes entre parenthèses. La clé primaire `id` est soulignée. L'écriture `#classe_id → classe.id` se lit : « la colonne `classe_id` est une clé étrangère qui désigne la colonne `id` de la table `classe` ».

> 📖 **NOUVELLE NOTION : LE MLD**
>
> Le **MLD**, ou **modèle logique de données**, est la traduction du MCD en liste de tables. Chaque entité y devient une table, chaque attribut une colonne, et chaque association 1-N une clé étrangère.
>
> En anglais : GB ***logical data model***.

Remarque que les noms changent de forme en passant du MCD au MLD : l'entité `ELEVE` devient la table `eleve`, au singulier, en minuscules et sans accent.

> 🧠 **VÉRIFIER SON MLD**
>
> Compte les associations 1-N de ton MCD, puis les clés étrangères de ton MLD : il doit y en avoir **autant**. Ici, une association 1-N donne une clé étrangère.

## Ce que la clé étrangère règle

Le local `B14` n'est plus écrit qu'une seule fois, sur la ligne de la 4TTR. Les anomalies du tableau unique disparaissent :

| Situation | Avec un seul tableau | Avec deux tables reliées |
|---|---|---|
| La 4TTR change de local | cinq lignes à modifier, au risque d'en oublier une | une seule ligne à modifier dans `classe` |
| L'école ouvre une classe sans élève | impossible sans inventer un élève | on ajoute une ligne dans `classe` |
| Les deux élèves de 6TTR partent | la classe 6TTR disparaît avec eux | la ligne de la 6TTR reste dans `classe` |
| Une faute de frappe dans le nom d'une classe | invisible, recopiée sur une seule ligne d'élève | le nom n'est écrit qu'une fois : la faute se voit et se corrige |

# Mettre le lien en place dans la base

Place à la pratique. Toutes les instructions de cette partie se tapent dans l'onglet *Exécuter le SQL* de DB Browser, sauf le réglage du pragma qui a son propre onglet.

## Activer la vérification des clés étrangères, avant tout le reste

SQLite sait vérifier que chaque clé étrangère désigne une ligne qui existe. Mais cette vérification est **désactivée par défaut**. Il faut l'activer avant de créer quoi que ce soit.

Dans DB Browser, ouvre l'onglet *Éditer les pragmas*, coche la case **Foreign Keys**, puis clique sur *Enregistrer*. Selon les réglages de ton DB Browser, la case est peut-être déjà cochée : vérifie quand même.

En SQL, le même réglage s'écrit :

```
PRAGMA foreign_keys = ON;
```

Pour vérifier qu'il est actif, demande sa valeur :

```
PRAGMA foreign_keys;
```

```
foreign_keys
------------
1
```

La valeur `1` veut dire « activé » ; `0` voudrait dire « désactivé ».

> 📖 **NOUVELLE NOTION : LE PRAGMA** `FOREIGN_KEYS`
>
> Un **PRAGMA** est une commande de réglage propre à SQLite : elle ne lit ni n'écrit de données, elle change le comportement de la base. `PRAGMA foreign_keys = ON` veut dire « clés étrangères : activées ». Tant qu'il n'est pas activé, SQLite enregistre les clés étrangères mais ne vérifie jamais leur contenu.
>
> En anglais aussi, on dit GB ***pragma***.

> ⚠️ **UN RÉGLAGE QUI NE DURE PAS**
>
> Le pragma ne s'enregistre pas dans le fichier de la base. Il vaut seulement **pour la session en cours**, c'est-à-dire tant que la base reste ouverte dans le logiciel qui l'a activé.

> Chaque logiciel et chaque programme qui ouvre la base part donc d'un pragma désactivé et doit le réactiver. En Python, cela s'écrit `conn.execute("PRAGMA foreign_keys = ON")`, juste après l'ouverture de la base.

## Créer la table `classe`

`CREATE TABLE` veut dire *créer une table*, et `INSERT INTO` veut dire *insérer dans*. La colonne `id` déclarée `INTEGER PRIMARY KEY` se remplit toute seule : on ne la donne pas dans l'`INSERT`.

```sql
CREATE TABLE classe (
    id    INTEGER PRIMARY KEY,
    nom   TEXT    NOT NULL UNIQUE,
    annee INTEGER NOT NULL,
    local TEXT
);

INSERT INTO classe (nom, annee, local) VALUES
    ('4TTR', 4, 'B14'),
    ('5TTR', 5, 'A21'),
    ('6TTR', 6, 'A23');
```

Vérifie le contenu de la table :

```sql
SELECT * FROM classe;
```

```
id | nom  | annee | local
---+------+-------+------
1  | 4TTR | 4     | B14
2  | 5TTR | 5     | A21
3  | 6TTR | 6     | A23
```

Le `UNIQUE` sur `nom` empêche d'enregistrer deux fois la même classe. Il n'empêche pas une faute de frappe comme `4TRR` : ce serait une valeur nouvelle, donc acceptée. Mais le nom de la classe n'est plus écrit qu'une seule fois : une faute se remarque tout de suite et se corrige en une ligne.

N'oublie pas d'enregistrer ton travail avec *Écrire les modifications* (Ctrl + S) après chaque étape.

## Ajouter la clé étrangère dans `eleve`

La table `eleve` existe déjà et contient douze élèves. On ne la recrée pas : on lui ajoute une colonne.

```sql
ALTER TABLE eleve ADD COLUMN classe_id INTEGER REFERENCES classe(id);
```

`ALTER TABLE` veut dire *modifier la table*, et `ADD COLUMN` veut dire *ajouter une colonne*. La nouvelle colonne `classe_id` est vide (`NULL`) pour les douze élèves.

> 📖 **NOUVELLE NOTION : `REFERENCES`**
>
> `REFERENCES` veut dire *fait référence à*. `classe_id INTEGER REFERENCES classe(id)` déclare que la colonne `classe_id` est une clé étrangère vers la colonne `id` de la table `classe`. Quand le pragma est activé, cette colonne ne peut contenir que des valeurs présentes dans `classe.id`, ou rester vide.
>
> En anglais, la déclaration s'appelle une GB ***foreign key constraint***.

> ℹ️ En SQLite, `ALTER TABLE` ne sait pas faire grand-chose. Ajouter une colonne fait partie des rares modifications qu'il accepte, c'est pourquoi on peut compléter `eleve` sans la recréer.

## Remplir les liens

Pour rappel, `UPDATE` veut dire *mettre à jour* et `SET` veut dire *régler* : `UPDATE table SET colonne = valeur WHERE condition;` modifie les lignes où la condition est vraie. `BETWEEN 1 AND 5` veut dire *entre 1 et 5*, et `IN (10, 11)` veut dire *parmi 10 et 11*.

```sql
UPDATE eleve SET classe_id = 1 WHERE id BETWEEN 1 AND 5;
UPDATE eleve SET classe_id = 2 WHERE id BETWEEN 6 AND 9;
UPDATE eleve SET classe_id = 3 WHERE id IN (10, 11);
```

Ces trois instructions modifient respectivement 5, 4 et 2 lignes. Vérifie le résultat :

```sql
SELECT id, nom, prenom, classe_id FROM eleve;
```

```
id | nom       | prenom  | classe_id
---+-----------+---------+----------
1  | Adam      | Lucas   | 1
2  | Bastin    | Emma    | 1
3  | Charlier  | Noah    | 1
4  | Delvaux   | Léa     | 1
```

```
5  | Englebert | Hugo    | 1
6  | Fontaine  | Chloé   | 2
7  | Gérard    | Nathan  | 2
8  | Hubert    | Manon   | 2
9  | Istas     | Louis   | 2
10 | Jacques   | Camille | 3
11 | Kevers    | Arthur  | 3
12 | Lambert   | Sarah   | NULL
```

Sarah Lambert reste sans classe. C'est voulu, et c'est permis par la cardinalité `(0,1)`.

## Vérifier que la base refuse l'incohérent

Une clé étrangère ne sert à rien si la base ne la fait pas respecter. Après l'avoir posée, fais toujours les deux tests suivants. Ils sont faits pour échouer.

> 📖 **NOUVELLE NOTION : L'INTÉGRITÉ RÉFÉRENTIELLE**
>
> L'**intégrité référentielle** est la règle qui garantit que chaque clé étrangère désigne une ligne qui existe vraiment. C'est une **contrainte** : une règle que la base vérifie à chaque écriture et qui refuse ce qui ne la respecte pas, comme elle le fait déjà pour `NOT NULL` ou `UNIQUE`.
>
> En anglais : GB ***referential integrity***.

**Premier test : rattacher un élève à une classe qui n'existe pas.** Il n'y a pas de classe numéro 99.

```sql
UPDATE eleve SET classe_id = 99 WHERE id = 12;
```

```
FOREIGN KEY constraint failed
```

Le message veut dire « la contrainte de clé étrangère a échoué ». La modification est refusée, et rien n'a changé :

```sql
SELECT id, nom, prenom, classe_id FROM eleve WHERE id = 12;
```

```
id | nom     | prenom | classe_id
---+---------+--------+----------
12 | Lambert | Sarah  | NULL
```

**Second test : supprimer une classe qui contient encore des élèves.** La 4TTR (`id` 1) en compte cinq.

```sql
DELETE FROM classe WHERE id = 1;
```

```
FOREIGN KEY constraint failed
```

Là aussi, la suppression est refusée et la table `classe` contient toujours ses trois lignes :

```sql
SELECT * FROM classe;
```

```
id | nom  | annee | local
---+------+-------+------
1  | 4TTR | 4     | B14
2  | 5TTR | 5     | A21
3  | 6TTR | 6     | A23
```

Si la base avait accepté cette suppression, les cinq élèves de 4TTR auraient gardé `classe_id = 1`, alors que plus aucune classe ne porterait cet `id`.

> 📖 **NOUVELLE NOTION : LA LIGNE ORPHELINE**
>
> Une **ligne orpheline** est une ligne dont la clé étrangère désigne une ligne qui n'existe pas, ou plus. L'intégrité référentielle sert précisément à empêcher leur apparition.
>
> En anglais : GB ***orphan row***.

> 🧠 **LES DEUX TESTS À RETENIR**
>
> 1. Rattacher une ligne à une ligne inexistante doit être **refusé**.
> 2. Supprimer une ligne encore désignée par une clé étrangère doit être **refusé**.
>
> Si l'un des deux passe sans erreur, le pragma n'est pas actif : retourne dans *Éditer les pragmas* et coche la case.

## Et si le pragma avait été oublié ?

Voici ce que donnent les deux mêmes tests avec le pragma désactivé, sur une copie de la base. Ne le fais pas dans ta propre base : l'exercice 5 te propose de le vérifier sur une copie.

Le rattachement à la classe 99 est accepté :

```
UPDATE eleve SET classe_id = 99 WHERE id = 12;
SELECT id, nom, prenom, classe_id FROM eleve WHERE id = 12;
```

```
id | nom     | prenom | classe_id
---+---------+--------+----------
12 | Lambert | Sarah  | 99
```

La suppression de la 4TTR est acceptée aussi. La table `classe` n'a plus que deux lignes :

```
id | nom  | annee | local
---+------+-------+------
2  | 5TTR | 5     | A21
3  | 6TTR | 6     | A23
```

Pourtant, les élèves de 4TTR désignent toujours la classe 1 :

```
id | nom       | classe_id
---+-----------+----------
1  | Adam      | 1
2  | Bastin    | 1
3  | Charlier  | 1
4  | Delvaux   | 1
5  | Englebert | 1
```

Aucun message d'erreur : la base contient maintenant six lignes orphelines, et rien ne te le signale.

---

# 🧪 Exercices

## Exercice 1 — Repérer les associations ★★☆☆☆

Applique la méthode sujet – verbe – complément à ce texte :

> « Le club de sport gère ses membres. Chaque membre est inscrit dans une catégorie d'âge. Un entraîneur encadre plusieurs séances. Chaque séance se déroule dans une salle. Un membre participe à plusieurs séances. »

1. Dresse le tableau sujet – verbe – complément – conclusion, comme dans l'article.
2. Pour chaque association trouvée, donne les deux entités (en MAJUSCULES) et le nom de l'association (un verbe).
3. Une de ces associations n'est pas une association 1-N : un membre participe à plusieurs séances, **et** une séance accueille plusieurs membres. Repère-la et explique pourquoi. Tu la traiteras plus tard, avec un autre outil : ici, signale-la seulement.

## Exercice 2 — Poser les cardinalités ★★☆☆☆

Pour chaque association, écris les cardinalités selon la notation `ENTITE (?,?) –verbe– (?,?) ENTITE`. Justifie **chacun des quatre chiffres** par une phrase qui commence par « Un … » ou « Une … ».

| Lettre | Association |
|---|---|
| a | `CLIENT –passe– COMMANDE` (une boutique en ligne) |
| b | `VEHICULE –appartient– CLIENT` (un garage) |
| c | `VILLE –se trouve dans– PAYS` |
| d | `SEANCE –se déroule dans– SALLE` (le club de sport) |

Exemple de justification attendue, pour l'école : « Un élève appartient au minimum à 0 classe, parce qu'un élève qui arrive peut ne pas encore en avoir. »

Pour chaque minimum, demande-toi s'il s'agit d'une évidence ou d'une décision de gestion. Un garage accepte-t-il d'enregistrer un client sans véhicule ? Justifie ton choix plutôt que de chercher « la » bonne réponse.

## Exercice 3 — Où va la clé étrangère ? ★★☆☆☆

1. Pour chaque association de l'exercice 2, dis dans quelle table va la clé étrangère et comment tu la nommes. Justifie avec la règle de placement.
2. Pour l'association (a), explique en deux ou trois phrases ce qui se passerait si tu mettais la clé étrangère **de l'autre côté**.
3. Écris le MLD du club de sport pour ses associations 1-N (sans l'association signalée à l'exercice 1). Invente deux ou trois attributs par entité.

4. Vérifie ton MLD : combien d'associations 1-N as-tu dans ton MCD, et combien de clés étrangères dans ton MLD ?

## Exercice 4 — Le titulaire ★★★☆☆

Travaille sur une **copie** de ta base terminée (ou de `mon-ecole-classes.db`). On ajoute l'association « une classe a pour titulaire un professeur ».

1. Écris l'association avec ses cardinalités, sous la forme `CLASSE (?,?) —a pour titulaire— (?,?) PROFESSEUR`. Justifie chaque minimum par une phrase : une classe peut-elle exister sans titulaire ? Un professeur doit-il être titulaire d'une classe ?
2. Dans quelle table va la clé étrangère ? Comment s'appelle-t-elle ? Écris la ligne du MLD qui change.
3. Active le pragma, puis écris l'`ALTER TABLE` qui ajoute la clé étrangère `professeur_id`.
4. Écris les `UPDATE` qui donnent un titulaire à chaque classe : Ludovic Lambrechts pour la 4TTR, Sophie Dubois pour la 5TTR et Marc Vermeulen pour la 6TTR. Cherche d'abord leur `id` dans la table `professeur`.
5. Fais les deux tests de l'intégrité référentielle sur cette nouvelle clé étrangère, et note les messages obtenus.
6. Peux-tu supprimer Paul Renard, qui n'est titulaire d'aucune classe ? Et Ludovic Lambrechts ? Explique la différence.

## Exercice 5 — Le pragma oublié ★★★☆☆

Travaille sur une **copie** de ta base terminée (ou de `mon-ecole-classes.db`).

1. Dans *Éditer les pragmas*, décoche **Foreign Keys** et enregistre.
2. Insère trois élèves de ton choix, rattachés aux classes 55, 66 et 77. Les insertions sont-elles acceptées ?
3. Recoche **Foreign Keys** et enregistre. Les trois lignes fautives ont-elles disparu ?
4. Pour retrouver toutes les lignes fautives d'un coup, exécute :

```
PRAGMA foreign_key_check;
```

   Ce pragma veut dire *vérification des clés étrangères*. Il affiche une ligne par problème : la colonne `table` donne la table fautive, `rowid` l'identifiant de la ligne, et `parent` la table désignée. Combien de lignes obtiens-tu ? Retrouve les élèves concernés grâce à leur `id`.
5. Supprime ces trois élèves, puis relance `PRAGMA foreign_key_check;`. Que renvoie-t-il maintenant ?
6. Réactiver le pragma a-t-il réparé la base ? Qu'en conclus-tu sur le moment où il faut l'activer ?

---

# ✍️ À retenir

- Une phrase **sujet – verbe – complément** entre deux entités signale une **association**, et le verbe lui donne son nom.
- Une **cardinalité** donne le minimum et le maximum ; elle se lit en commençant par « **un** » suivi de l'entité du côté où elle est écrite : `ELEVE (0,1) —appartient— (0,n) CLASSE`.
- Le **minimum** est une décision de gestion : 0 veut dire « on accepte que ce soit vide ».
- La clé étrangère va dans la table de l'entité qui porte le maximum 1 — celui qui n'en a qu'un garde l'adresse de l'autre.
- Dans le **MLD**, elle s'écrit `#classe_id → classe.id`, et il y a autant de clés étrangères que d'associations 1-N.
- `PRAGMA foreign_keys = ON` est **désactivé par défaut** et vaut seulement pour la session : il faut le réactiver dans chaque logiciel et chaque programme.
- Après chaque clé étrangère, deux tests : lien vers une ligne inexistante, et suppression d'une ligne encore désignée. Les deux doivent être refusés.

---

# Suite

Tes deux tables sont reliées. Mais un `SELECT` sur `eleve` affiche `classe_id = 1`, pas `4TTR`. L'article Interroger deux tables t'apprend à lire les deux tables ensemble.