Du MCD au modèle relationnel

Comment transformer un schéma entité-association en un ensemble de tables prêtes à être créées : règles de passage, clés primaires, clés étrangères.

4TTR niveau

Tu as un MCD complet : entités, attributs, relations nommées, cardinalités. Tu sais aussi que chaque ligne d'une table doit avoir un identifiant unique. Il manque une étape : **comment relier les tables entre elles** pour matérialiser les relations du MCD.

L'objectif de ce cours est d'apprendre les **règles de traduction** d'un MCD vers un **modèle logique de données** — le **MLD**. À la fin, tu auras une liste de tables avec toutes leurs colonnes, prêtes à être créées dans le SGBDR au prochain cours.

🎯 Objectifs

À la fin de cette séquence, tu seras capable de :

1. Transformer chaque entité en une table avec sa clé primaire
2. Traduire une relation `0,n / 1,1` en clé étrangère
3. Écrire un MLD complet en notation textuelle
4. Repérer où placer la clé étrangère selon les cardinalités

Partie 1 — La clé étrangère, le maillon manquant

Un exemple pour comprendre

Reprenons deux entités du MCD billetterie :

On a vu dans le cours sur les identifiants que chaque table aura un `id` :

```
client(id, nom, prenom)
ticket(id, prix, statut)
```

Mais avec juste ça, **rien ne relie un ticket à son client**. Si je regarde la ligne `ticket` n°42, je ne sais pas qui l'a acheté.

La solution : une colonne qui pointe

L'idée est simple : on ajoute dans la table `ticket` une colonne qui **contient l'id du client** propriétaire :

id	prix	statut	client_id
1	12	payé	**3**
2	12	payé	**3**
3	15	réservé	**7**

Tickets n°1 et n°2 appartiennent au client n°3. Ticket n°3 appartient au client n°7. La relation `achète` du MCD est maintenant matérialisée par cette colonne.

Cette colonne porte un nom précis : **clé étrangère** (GB ***foreign key, FK***). Elle est *étrangère* parce qu'elle référence la clé primaire d'une **autre** table.

> 💡
> **RÈGLE D'OR**
> : une clé étrangère contient **toujours** une valeur qui existe dans la clé primaire de la table référencée. Le SGBDR refusera d'insérer un ticket avec `client_id = 999` si aucun client n°999 n'existe.

Partie 2 — Les règles de traduction

Règle 1 — Chaque entité devient une table

Pour chaque entité du MCD :

- créer une **table** du même nom (en minuscules, au singulier par convention)
- chaque attribut devient une **colonne**
- ajouter une colonne `id` comme **clé primaire** (`INT AUTO_INCREMENT`)

```
Entité ÉVÉNEMENT (titre, date, prix_base)
       │
       ▼
Table  evenement (id, titre, date, prix_base)
```

Règle 2 — Une relation `0,n / 1,1` devient une clé étrangère

C'est le cas le plus fréquent et celui qui te servira partout.

Principe : la clé étrangère est placée du côté `1,1`.

Pourquoi ? Parce que du côté `1,1`, chaque occurrence pointe vers **exactement un** élément de l'autre côté — une seule colonne suffit pour stocker cette référence.

```
CLIENT  ———0,n—— achète ——1,1——►  TICKET
                                  (↑ la FK va ici : client_id)
```

À l'inverse, du côté `0,n`, un client peut avoir plusieurs tickets — on ne pourrait pas stocker plusieurs ids dans une seule colonne.

Résultat sur le MCD billetterie :

```
client    (id, nom, prenom, email, telephone)
evenement (id, titre, date, prix_base)
ticket    (id, prix, statut, date_achat, #client_id, #evenement_id)
```

Le `#` signale une clé étrangère. La table `ticket` reçoit **deux** clés étrangères parce qu'elle participe à **deux** relations `0,n / 1,1` (avec `client` et avec `evenement`).

Règle 3 — Une relation `0,1 / 0,n` se traite pareil

Même logique : la FK va du côté du **maximum à 1**. La différence avec `1,1` : la colonne peut être vide (`NULL` autorisé).

```
LIEU   ———0,n—— accueille ——0,1——►  ÉVÉNEMENT
                                    (FK : lieu_id, NULL autorisé)
```

Un événement peut ne pas avoir de lieu encore défini → `lieu_id` peut être `NULL`.

Partie 3 — Notation textuelle du MLD

Sur papier (ou dans un cahier), on écrit un MLD avec les conventions suivantes :

- **Nom de la table** en minuscules, suivi des colonnes entre parenthèses
- La **clé primaire** est **soulignée**
- Les **clés étrangères** sont préfixées d'un `#`
- Une flèche `→` (ou une note) précise vers quelle table pointe la FK

Exemple complet — la billetterie

```
client    (id, nom, prenom, email, telephone)
            —
```

```
evenement (id, titre, date, prix_base)
            —

ticket    (id, prix, statut, date_achat, #client_id, #evenement_id)
            —                              └──→ client.id
                                                      └──→ evenement.id
```

Cette écriture est **suffisante** pour créer ensuite les tables dans le SGBDR. Toutes les informations nécessaires sont là.

Partie 4 — Quel type pour chaque colonne ?

Au moment de créer la table, le SGBDR exigera un **type** pour chaque colonne. Voici les plus utiles à ce stade :

Type	Pour stocker	Exemple
`INT`	un entier	un id, un prix en centimes, un nombre
`VARCHAR(n)`	du texte court (max `n` caractères)	un nom, un email, un titre
`TEXT`	du texte long sans limite pratique	une description, un commentaire
`DATE`	une date `AAAA-MM-JJ`	une date d'événement
`DATETIME`	une date + heure	une date d'achat précise
`DECIMAL(p,d)`	un nombre à virgule fixe	un prix : `DECIMAL(6,2)` = jusqu'à 9999,99
`BOOLEAN`	vrai / faux	un statut binaire

> ⚠️ **ÉVITE `FLOAT` POUR DE L'ARGENT.**
> `DECIMAL` garantit la précision exacte ; `FLOAT` arrondit subtilement et provoque des erreurs de centimes.

MLD enrichi avec les types

```
client (
    id          INT  PK  AUTO_INCREMENT
    nom         VARCHAR(100)  NOT NULL
    prenom      VARCHAR(100)  NOT NULL
    email       VARCHAR(150)
    telephone   VARCHAR(20)
)

evenement (
    id          INT  PK  AUTO_INCREMENT
    titre       VARCHAR(200)  NOT NULL
    date        DATETIME      NOT NULL
    prix_base   DECIMAL(6,2)  NOT NULL
)
```

```
ticket (
    id           INT  PK  AUTO_INCREMENT
    prix         DECIMAL(6,2)  NOT NULL
    statut       VARCHAR(20)   NOT NULL
    date_achat   DATETIME      NOT NULL
    client_id    INT  FK → client.id        NOT NULL
    evenement_id INT  FK → evenement.id     NOT NULL
)
```

C'est ce que tu vas littéralement encoder dans HeidiSQL au prochain cours.

Partie 5 — Cas particulier : la relation `0,1 / 0,1`

Plus rare, mais ça arrive : deux entités liées au maximum une seule fois de chaque côté.

Exemple : un `employe` est rattaché à au plus un `casier`, et un `casier` est attribué à au plus un employé.

```
EMPLOYE  ———0,1—— possède ——0,1——►  CASIER
```

Règle : on peut placer la FK **d'un côté ou de l'autre** — peu importe techniquement. On choisit :

- selon ce qui est le plus naturel (l'employé existe d'abord, le casier est attribué après → FK côté casier)
- en ajoutant une contrainte `UNIQUE` sur la FK pour interdire qu'elle se répète (sinon on retomberait sur du `0,n / 0,1`)

```
employe (id, nom, prenom)
casier  (id, numero, #employe_id UNIQUE)
```

> 💡 Ce cas est **RARE EN PRATIQUE**. On préfère souvent regrouper les deux entités dans une seule table quand la relation est aussi exclusive. Mais c'est utile à connaître.

Partie 6 — La méthode, étape par étape

Quand tu reçois un MCD, voici la procédure à suivre :

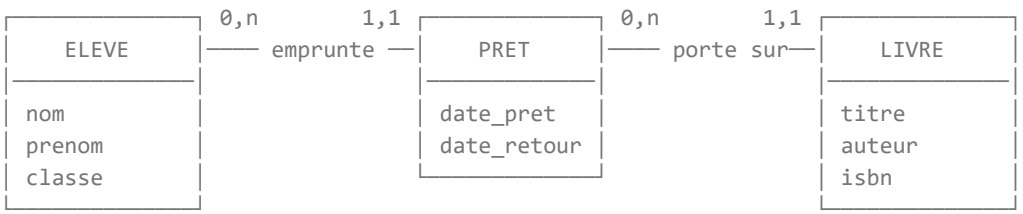
1. **Une table par entité**, avec un `id` comme clé primaire
2. **Recopier les attributs** en choisissant un type pour chacun
3. **Pour chaque relation** `0,n / 1,1` : ajouter une FK dans la table du côté `1,1`
4. **Pour chaque relation** `0,n / 0,1` : ajouter une FK dans la table du côté `0,1`, autorisée à `NULL`

5. **Vérifier** : chaque relation du MCD doit se retrouver sous forme de FK quelque part

Si tu termines avec autant de relations que de FK ajoutées, ton MLD est cohérent.

🧪 Exercice 1 — Bibliothèque scolaire

Voici un MCD simplifié :



Questions :

1. Combien de tables ce MLD aura-t-il ?
2. Pour chacune, donne sa clé primaire.
3. Où va la clé étrangère `eleve_id` ? La clé étrangère `livre_id` ?
4. Écris le MLD complet en notation textuelle (avec les types).

🧪 Exercice 2 — Refaire la billetterie sans regarder

Sans consulter la Partie 3, écris le MLD complet du système de billetterie en notation textuelle. Compare ensuite avec la correction.

🧪 Exercice 3 — Ton projet

Reprends le MCD que tu as construit en séquence 4 (garage, formation, Discord, ou autre). Applique la procédure de la Partie 6 et écris le MLD complet — types compris.

> ⚠️ Vérification finale : chaque relation du MCD doit avoir donné lieu à **EXACTEMENT UNE** clé étrangère dans ton MLD. Ni plus, ni moins.

✍️ À retenir

- Une **entité** devient une **table** avec un `id` comme clé primaire.
- Une **relation** se matérialise par une **clé étrangère** : une colonne qui contient l'id d'une ligne d'une autre table.
- Pour une relation `0,n / 1,1` (ou `0,n / 0,1`) : la **FK va du côté** `1`.
- Le SGBDR garantit que chaque FK pointe vers une ligne existante — c'est l'**intégrité référentielle**.
- Un MLD bien écrit (avec PK, FK et types) contient tout ce qu'il faut pour créer les tables.

Suite

Tu as maintenant un MLD prêt à l'emploi. Dans le prochain cours, on ouvre HeidiSQL, on lance MySQL via Laragon, et on crée concrètement la première base et ses premières tables. Les clés étrangères, elles, auront leur cours dédié juste après.