Découvrir une base de données de type MySQL

I. Explication du modèle relationnel via une base de données relationnelle

1) Systèmes de Gestion de Base de Données Relationnelle = SGBDR

Base de Données Relationnelle : Une collection organisée de données structurées en tables interconnectées.

Système de Gestion de Base de Données Relationnelle : Un logiciel qui permet de persister, d'accéder et de manipuler des données stockées de manière organisée et liée entre elles.

Objectifs du SGBDR :

- Persister les données de manière durable.
- Accéder aux données depuis une application de manière normalisée.
- Garantir l'intégrité et la consistance des données.

2) SQL = SOFTWARE QUERY LANGAGE

- Manipuler des données relationnelles depuis une application.
- Normaliser
- Gérer et structurer une BD.

DML (Data Manipulation Language) et DDL (Data Definition Language) sont deux types de langages SQL utilisés dans le contexte des systèmes de gestion de base de données relationnelles (SGBDR). Ils servent à des tâches distinctes liées à la manipulation et à la définition des données.

DDL est principalement utilisé pour **définir et gérer la structure de la base de données**, tandis que DML est utilisé pour **manipuler les données elles-mêmes**. Ces deux langages sont essentiels pour le développement, la maintenance et l'utilisation quotidienne des bases de données relationnelles.

- **DML (Data Manipulation Language) :**

Objectif : Le langage de manipulation des données est utilisé pour manipuler (ajouter, mettre à jour, supprimer) les données présentes dans la base de données.

Principales commandes DML :

- SELECT: Récupère des données de la base de données.
- INSERT: Ajoute de nouvelles données à la base de données.
- UPDATE: Modifie les données existantes dans la base de données.
- DELETE: Supprime des données de la base de données.

Utilisation typique :

- Extraction d'informations de la base de données.
- Ajout, modification ou suppression de données existantes.

DDL (Data Definition Language) :

Objectif : Le langage de définition des données est utilisé pour définir et gérer la structure globale de la base de données, c'est-à-dire la façon dont les données sont organisées et les relations entre elles.

Principales commandes DDL :

- CREATE: Crée des objets de base de données tels que des tables, des vues, des index, etc.
- ALTER: Modifie la structure d'un objet existant dans la base de données.
- DROP: Supprime un objet de la base de données.
- RENAME, TRUNCATE, COMMENT…

Utilisation typique :

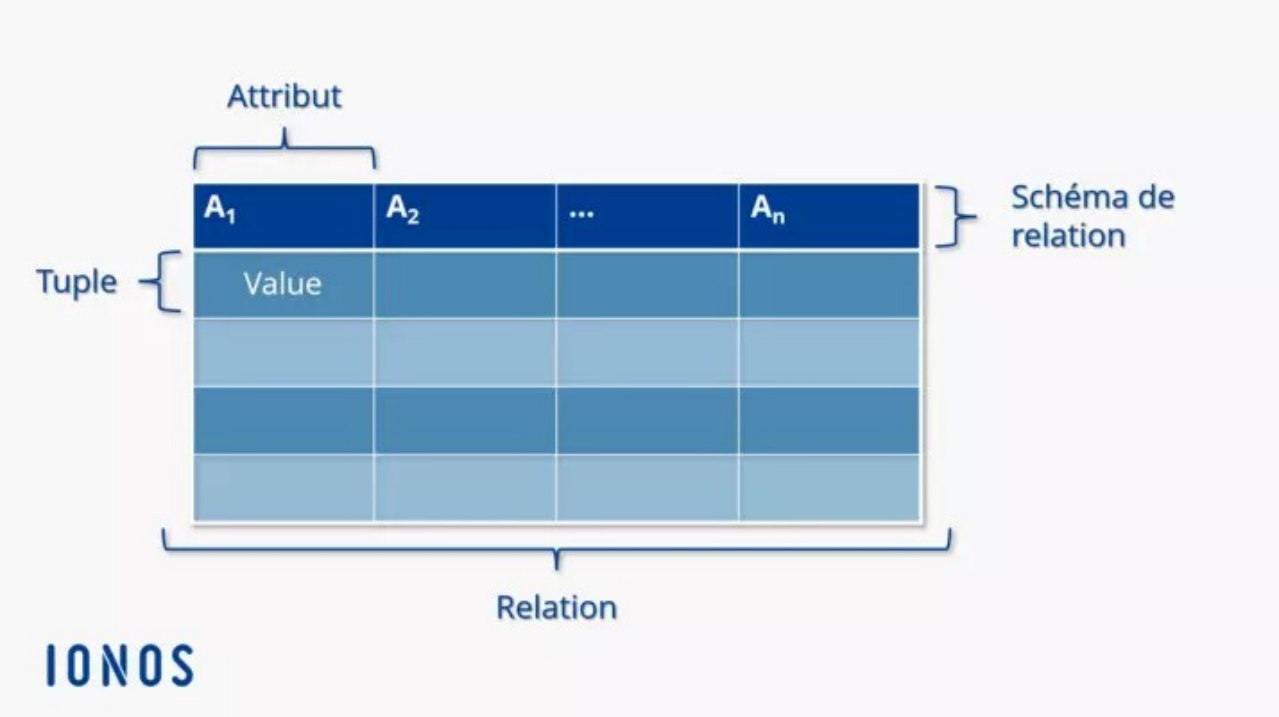
- Création de tables, d'index, de vues, etc.
- Modification de la structure de la base de données.

II. Concepts fondamentaux du modèle relationnel

1) Tables, lignes et colonnes

Tables : une table est une structure de stockage pour les données. Elle représente une collection de données organisées en lignes (horizontales) et colonnes (verticales) qui permettent de **collecter et d'afficher les informations sous une forme ordonnée**. Chaque table a un nom unique et est utilisée pour stocker des informations spécifiques.

Lignes et colonnes : Les lignes représentent des enregistrements individuels dans la table (aussi appelé « tuple »), tandis que les colonnes spécifient les attributs de ces enregistrements.



e_id	nom	prénom	numéro de sécurité sociale	rue	code postale	ville
1	Dupond	Jacques	25 120512 S 477	Grand Boulevard 1	11111	Lacité
2	Martin	Jean	25 100615 M 694	Rue de la Gare 5	22222	Laville
3	Petit	Hélène	25 091225 M 463	Place du Marché 3	33333	Levillage

2) Types de données

Les types de données définissent le genre d'information que chaque colonne peut stocker.

a) INT: Entier

b) VARCHAR(n): Chaîne de caractères de longueur maximale n

c) DECIMAL(p, s): Nombre décimal avec p chiffres au total, s après la virgule

3) Relations entre les tables

Les relations définissent les liens entre les différentes tables, généralement établies par des clés étrangères. FOREIGN KEY: Clé étrangère liée à la clé primaire d'une autre table (Clients dans cet exemple).

III. Notion de Clé Primaire et de Clé Étrangère

La clé primaire assure l'unicité des enregistrements, tandis que la clé étrangère établit des relations entre les tables, garantissant l'intégrité référentielle. Ces concepts fondamentaux sont essentiels pour la conception et la maintenance de bases de données relationnelles.

1) Clé primaire

Définition : Une colonne (ou un ensemble de colonnes) qui identifie de manière unique chaque enregistrement dans une table.

La clé primaire dans une base de données est comme une "étiquette unique" pour chaque ligne dans une table. Elle sert à identifier de manière spécifique chaque élément, évite les répétitions, accélère les recherches, et permet de créer des liens logiques entre différentes parties de la base de données. En bref, c'est une sorte de numéro d'identification unique pour chaque morceau d'information dans la base de données.

Caractéristiques d'une clé primaire appropriée *:*

- **Unicité** : La valeur de la clé primaire doit être unique parmi toutes les lignes de la table, ce qui garantit l'unicité de chaque enregistrement.
- **Non-Nullité** : La valeur de la clé primaire ne peut pas être nulle.
- **Stabilité** : La valeur de la clé primaire ne doit pas changer au fil du temps. En empêchant la duplication des enregistrements, la clé primaire contribue à maintenir l'intégrité des données. Elle assure qu'aucune information n'est enregistrée plus d'une fois dans la table.

Avantages :

- **Identification Univoque** : Permet d'identifier chaque enregistrement de manière univoque.
- **Optimisation des Recherches** : Améliore les performances des requêtes de recherche et de jointure. La clé primaire est souvent utilisée comme base pour les index.
- **Références dans d'autres tables** : Lorsqu'une table a une clé primaire, d'autres tables peuvent y faire référence en utilisant une clé étrangère. Cela établit des relations entre les tables et contribue à garantir l'intégrité référentielle, assurant que les références entre les données sont valides et cohérentes.

Implémentation dans une table :

L'implémentation d'une clé primaire dans une table se fait généralement lors de la création de la table. Supposons que nous avons une table "Utilisateurs" avec les colonnes "ID" (identifiant), "Nom" et "Email". On peut implémenter une clé primaire comme ceci :

```sql
CREATE TABLE Utilisateurs (
    ID INT PRIMARY KEY,
    Nom VARCHAR(50),
    Email VARCHAR(100)
);
```

- ***ID INT PRIMARY KEY*** : Crée une colonne nommée "ID" de type entier (INT) et déclare que cette colonne est la clé primaire de la table. L'utilisation de "PRIMARY KEY" garantit que chaque valeur dans la colonne "ID" sera unique.

- On peut également avoir une clé primaire composée de plusieurs colonnes. Par exemple, dans une table "Commandes", les colonnes "Numéro de Commande" et "ID Utilisateur" composent la clé primaire qui garantit l'unicité de chaque enregistrement dans la table.

```sql
CREATE TABLE Commandes (
    NumeroCommande INT,
    IDUtilisateur INT,
    PRIMARY KEY (NumeroCommande, IDUtilisateur)
);
```

2) Clé étrangère

Définition : Une colonne (ou un ensemble de colonnes) qui établit une relation entre deux tables en reliant la clé primaire d'une table à une colonne de l'autre table.

La clé étrangère est essentielle pour établir, maintenir et optimiser les relations entre les tables dans une base de données relationnelle.

Caractéristiques d'une clé étrangère appropriée ***:***

- **Référence à une Clé Primaire** : La clé étrangère fait référence à la clé primaire d'une autre table.
- **Intégrité Référentielle** : Garantit qu'une valeur dans la clé étrangère correspond toujours à une valeur existante dans la clé primaire.

Avantages :

- **Maintien de la Cohérence** : Assure la cohérence des données entre les tables.
- **Établissement de Relations** : Permet de définir des relations logiques entre différentes tables. Elle connecte les données de deux tables en utilisant une colonne commune.
- **Cascade des Opérations** : Permet de définir des actions en cascade lors de la modification ou suppression des données dans la table parente.

Implémentation dans une table :
Si on imagine deux tables, "Commandes" et "Clients". La clé étrangère permet de dire à la table "Commandes" : "Regarde dans la table 'Clients', et prends le 'ClientID' qui correspond à la

personne qui a passé la commande." La clé étrangère s'assure que si un client est supprimé de la table "Clients", toutes les commandes associées dans la table "Commandes" ne soient pas laissées sans propriétaire, mais également supprimées.

Actions en cascade :

Les actions en cascade sont des paramètres qu'on peut spécifier lors de la définition d'une clé étrangère dans une base de données relationnelle. Ces actions déterminent le comportement de la base de données lorsqu'une modification est apportée à la clé primaire d'une table parente, à laquelle une clé étrangère dans une table enfant fait référence. Voici quelques-unes des actions en cascade couramment utilisées :

- **CASCADE** : Lorsqu'une clé primaire est mise à jour ou supprimée, les modifications correspondantes sont automatiquement appliquées à la clé étrangère dans la table enfant. Par exemple, si un client est supprimé de la table parente, toutes les commandes associées dans la table enfant seront également supprimées. Existe pour **ON DELETE** si la clef primaire est supprimée et pour **ON UPDATE** si la clef primaire est modifiée.

FOREIGN KEY (ClientID) REFERENCES Clients(ID) ON DELETE CASCADE
FOREIGN KEY (ClientID) REFERENCES Clients(ID) ON UPDATE CASCADE

- **SET NULL** : Lorsqu'une clé primaire est mise à jour ou supprimée, les valeurs correspondantes dans la clé étrangère de la table enfant sont définies à NULL. Par exemple, si un client est supprimé, le "ClientID" dans toutes les commandes associées serait défini à NULL.

FOREIGN KEY (ClientID) REFERENCES Clients(ID) ON DELETE SET NULL

- **SET DEFAULT** : Similaire à SET NULL, sauf que la clé étrangère prend la valeur par défaut définie pour cette colonne. Cela nécessite que la colonne ait une valeur par défaut définie.

FOREIGN KEY (ClientID) REFERENCES Clients(ID) ON DELETE SET DEFAULT

- **NO ACTION / RESTRICT** : Cela signifie qu'aucune action n'est effectuée sur la clé étrangère si une mise à jour ou une suppression est tentée sur la clé primaire. Cependant, cela pourrait provoquer une violation de l'intégrité référentielle si la modification viole la clé étrangère.

FOREIGN KEY (ClientID) REFERENCES Clients(ID) ON DELETE NO ACTION

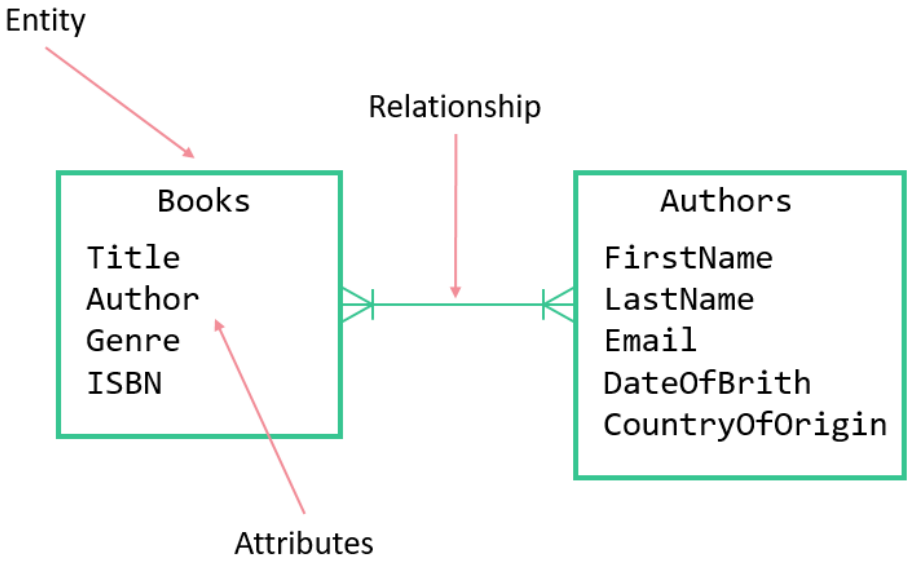
```sql
CREATE TABLE Clients (
    ID INT PRIMARY KEY,
    Nom VARCHAR(50)
);

CREATE TABLE Commandes (
    NumeroCommande INT PRIMARY KEY,
    ClientID INT,
    FOREIGN KEY (ClientID) REFERENCES Clients(ID) ON UPDATE CASCADE
);
```

IV. Présenter un MCD (Modèle Conceptuel de Données) et un MPD (Modèle Physique de Données)

1) Modèle Conceptuel de Données (MCD)

La **conception conceptuelle** est la première étape du processus de conception de bases de données. Elle vise à comprendre les besoins du domaine métier sans se soucier de la structure de stockage des données. Le Modèle Conceptuel de Données (MCD) est utilisé pour représenter ces concepts de manière abstraite et indépendante du système de gestion de base de données.

https://creately.com/blog/fr/uncategorized-fr/tutoriel-sur-le-diagramme-er/
https://creately.com/diagram-community/popular/t/erd



Utilisation de Diagrammes Entité-Association/Relation (ER) :

- **Entités** : Les entités représentent des objets ou des concepts du monde réel qui ont des données à stocker. Par exemple, dans une base de données pour une bibliothèque, **"Livre" et "Auteur" peuvent être des entités**.

*Une **entité faible** est une entité qui n'a pas de clé primaire unique propre, et n'a donc pas la capacité de s'identifier de manière autonome. Contrairement aux entités fortes qui ont une clé primaire indépendante et unique pour les identifier, les entités faibles dépendent d'une autre entité pour leur identification, elles doivent être liées à une entité forte via une relation appelée "entité propriétaire".*

***EXEMPLE** : Dans le cas d'une base de données pour la gestion des étudiants et de leurs cours, une entité faible pourrait être une "Note", qui dépend de l'entité propriétaire "Cours" pour son identification (via une clé étrangère). Une note seule n'a pas de sens sans référence à un cours spécifique. La clé primaire de "Cours" (CodeCours) est utilisée comme partie de la clé de "Note", et une note spécifique est identifiée par la combinaison de CodeCours et d'autres attributs comme la date. La "Note" n'a pas de clé primaire propre indépendante de "Cours".*

Entité Propriétaire : Cours
- Clé Primaire : CodeCours

Entité Faible : Note
- Clé Étrangère vers Cours (CodeCours)
- Autres Attributs : Note, Date

- **Relations** : Les relations décrivent comment les entités sont liées les unes aux autres. Par exemple, un livre est écrit par un auteur. ***La relation entre "Livre" et "Auteur" pourrait être appelée "Écrit par".***

*Une **relation faible** est une relation dans un modèle de données qui ne peut exister que lorsqu'elle est associée à une entité faible. Contrairement à une relation classique, une relation faible n'a pas de clé primaire indépendante et doit être connectée à une entité faible pour être identifiable. Elle est souvent utilisée pour modéliser des associations dépendantes, où l'existence de la relation dépend de l'existence de l'entité faible associée.*

***EXEMPLE** : Une relation faible "Assister" peut représenter la participation d'un étudiant à un cours spécifique, et dépend de l'existence de l'entité faible "Note". La clé étrangère vers "Étudiant" dans "Assister" fait partie de la clé de cette relation, montrant la dépendance entre la relation et l'entité faible. La relation "Assister" n'aurait de sens que lorsqu'il y a une note associée à un étudiant.*

Entité Propriétaire : Étudiant
- Clé Primaire : Matricule

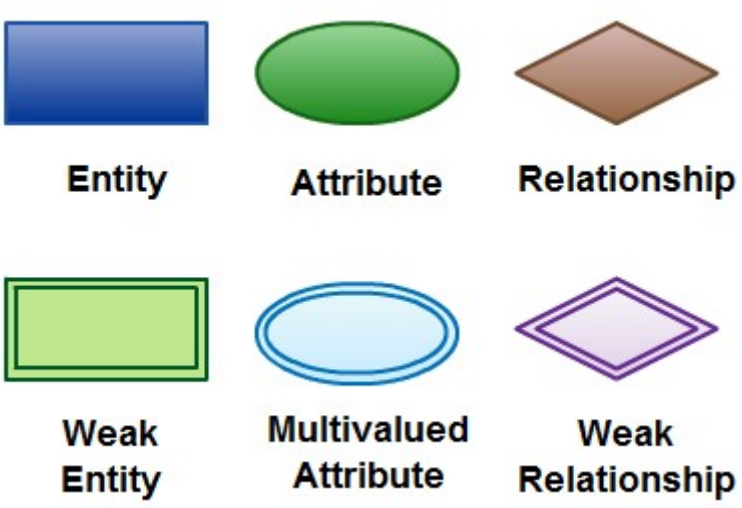
Entité Faible : Note
- Clé Étrangère vers Étudiant (Matricule)
- Autres Attributs : Note, Date

Relation Faible : Assister
- Clé Étrangère vers Étudiant (Matricule)
- Clé Étrangère vers Cours (CodeCours)
- Autres Attributs : Présence

- **Attributs** : Les attributs sont les propriétés ou caractéristiques des entités. ***Par exemple, un livre peut avoir des attributs tels que "Titre", "ISBN", "Auteur" et "Date de publication", et un auteur a des attributs tels que "Nom", "Date de naissance" et"Nationalité".***

*Un **attribut multivalué** est un attribut qui peut avoir plusieurs valeurs associées à une seule occurrence de l'entité à laquelle il appartient. Il est représenté par **un ovale connecté à l'entité par une ligne**. **EXEMPLE** : Une entité "Personne" a un attribut "Numéros de Téléphone". Une personne peut avoir plusieurs numéros de téléphone, donc cet attribut est multivalué.*

*Un **attribut dérivé** est un attribut dont la valeur peut être déduite à partir d'autres attributs dans la même entité. Il n'est pas stocké en tant que tel dans la base de données, mais plutôt calculé ou dérivé dynamiquement lorsque nécessaire. Il est représenté par **un ovale connecté à l'entité par une ligne pointillée**. **EXEMPLE : U**ne entité "Personne" a des attributs "Date de Naissance" et "Âge". L'âge est un attribut dérivé car il peut être calculé à partir de la date de naissance sans avoir besoin d'être stocké séparément.*



Cardinalité et Ordinalité :

Ces deux définitions définissent les relations entre les entités en plaçant la relation dans le contexte des nombres. Il existe un certain nombre de notations utilisées pour présenter la cardinalité dans les diagrammes ER. Dans un diagramme UML (Unified Modeling Language), l'information sur la cardinalité et l'ordinalité est généralement incluse dans les relations.

La cardinalité d'une relation entre deux entités indique **le nombre d'occurrences d'une entité qui peuvent être associées à une seule occurrence de l'autre entité dans une relation**. Elle exprime la quantité ou le nombre de liens entre les entités. La cardinalité peut être indiquée en utilisant des chiffres près des connecteurs de l'association entre les classes.

- **Un-à-Un (1:1)** : Une entité d'un côté de la relation peut être associée à une seule entité de l'autre côté, et vice versa. *Exemple : Un livre est associé à un seul exemplaire physique dans la bibliothèque, et chaque exemplaire est lié à un seul livre.*

- **Un-à-Plusieurs (1:N)** : Une entité d'un côté de la relation peut être associée à plusieurs entités de l'autre côté, mais chaque entité de l'autre côté est associée à une seule entité de l'autre côté. *Exemple : Un auteur peut avoir écrit plusieurs livres, mais chaque livre est écrit par un seul auteur.*

- **Plusieurs-à-Un (N:1)** : Plusieurs entités d'un côté de la relation peuvent être associées à une seule entité de l'autre côté, mais chaque entité du côté "plusieurs" est associée à une seule entité de l'autre côté. *Exemple : Plusieurs exemplaires du même livre peuvent être présents dans la bibliothèque, mais chaque exemplaire est lié à un seul livre.*

- **Plusieurs-à-Plusieurs (N:N)** : Plusieurs entités d'un côté de la relation peuvent être associées à plusieurs entités de l'autre côté, et vice versa. *Exemple : Plusieurs étudiants peuvent emprunter plusieurs livres, et chaque livre peut être emprunté par plusieurs étudiants.*

- **Zéro-à-Un (0..1)** : Une entité d'un côté de la relation peut être associée à zéro ou une entité de l'autre côté, et vice versa. *Exemple : Un livre peut ou non être réservé, mais s'il l'est, il n'est réservé que par un seul lecteur.*

- **Zéro-à-Plusieurs (0..N)** : Une entité d'un côté de la relation peut être associée à zéro, une ou plusieurs entités de l'autre côté, et vice versa. *Exemple : Un lecteur peut emprunter zéro, un ou plusieurs livres, et chaque livre peut être emprunté par zéro, un ou plusieurs lecteurs.*

- **Un-à-Zéro ou Un (1..0 or 1)** : Une entité d'un côté de la relation peut être associée à une entité de l'autre côté, ou aucune (zéro). *Exemple : Un genre peut être attribué à zéro ou plusieurs livres, mais un livre peut également ne pas avoir de genre spécifique.*

- **Zéro-à-Zéro (0..0)** : Aucune entité n'est associée à une entité de l'autre côté de la relation. *Exemple : Un livre peut ou non avoir une note de critique, et une critique peut ou non être associée à un livre.*

L'ordinalité indique si l**a participation d'une entité à une relation est obligatoire (1) ou facultative (0 ou n)**. L'ordinalité n'est pas explicitement détaillée dans UML. Cependant, on peut utiliser la multiplicité pour exprimer la contrainte de participation.

- **Obligatoire-Optionnelle (1:N)** : *Un livre (1) est écrit par au moins un auteur (N). Chaque livre a au moins un auteur, mais un auteur peut écrire zéro ou plusieurs livres.*

- **Optionnelle-Obligatoire (N:1)** : *Un auteur (N) peut écrire au moins un livre (1). Chaque auteur a la possibilité d'écrire un ou plusieurs livres, mais chaque livre est écrit par au moins un auteur.*

- **Obligatoire-Obligatoire (1:1)** : *Un livre (1) est référencé dans au moins une catégorie (1). Chaque livre doit être classé dans une catégorie spécifique, et chaque catégorie doit contenir au moins un livre.*

- **Optionnelle-Optionnelle (0..1:0..N)** : *Un livre (0..1) peut être réservé par au moins un lecteur (0..N). Chaque livre peut être réservé par zéro ou un lecteur, et chaque lecteur peut ou non avoir réservé un livre.*

- **Obligatoire-Obligatoire (N:N)** : *Un étudiant (N) peut emprunter au moins un livre (N), et chaque livre doit être emprunté par au moins un étudiant.*

- **Optionnelle-Optionnelle (0..N:0..N)** : *Un auteur (0..N) peut avoir écrit au moins un livre (0..N), et chaque livre peut ou non avoir été écrit par un auteur. Cela permet des situations où certains livres n'ont pas d'auteur, et certains auteurs n'ont pas écrit de livre.*

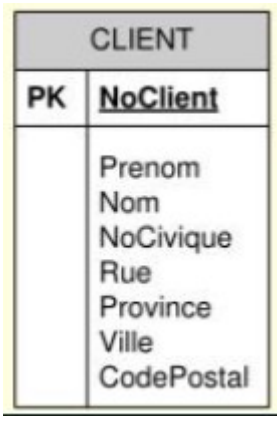
https://www.youtube.com/playlist?list=PLB9AbbTDeBzQ5oDCi3NfHivXYrLQyC3Lu

2) Modèle Physique de Données (MPD)

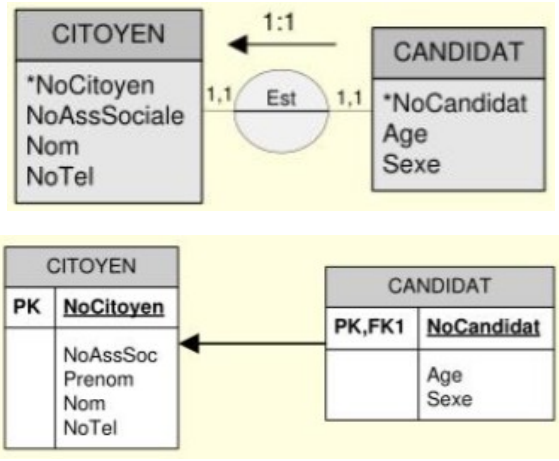
Dans la méthodologie Merise, le MPD (Modèle Physique des Données) fait suite au MCD. En s'appuyant sur des règles simples (et qui fonctionnent à tous les coups), l'analyste fait évoluer sa modélisation de haut niveau pour la transformer en un schéma plus proche des contraintes des logiciels de bases de données. Il s'agit de préparer l'implémentation dans un SGBDR.

Concrètement, cette étape permet de construire la structure finale de la base de données avec les différents liens entre les éléments qui la composent. Le vocabulaire change :

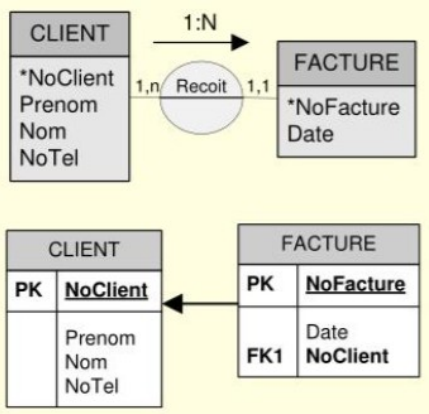
1. Les **entités** se transforment en **tables.** *Mise en forme : une table s'écrit en lettres majuscules, sans espace et sans accent)*
2. Les **identifiants** se transforment en **clés primaires** des tables. *Mise en forme : la clé primaire est la première colonne de la table, elle est identifiée par les lettres **PK**. Le nom de la clé primaire est écrit dans la deuxième colonne en **CamelCase** et **souligné***. Chaque table dispose d'au minimum 1 clé dite primaire ;
3. Les **attributs** se transforment en **champs** (ou attributs). *Mise en forme : les champs sont ajoutés dans la deuxième colonne de la table, et décomposés.*



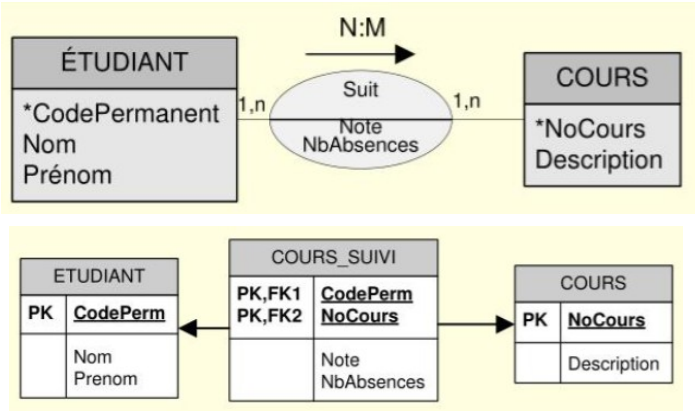
4. Les **relations de type 1:1** deviennent des **clés étrangères**. Une clé étrangère est une clé primaire provenant d'une autre table, qui permet de faire le lien entre deux tables.



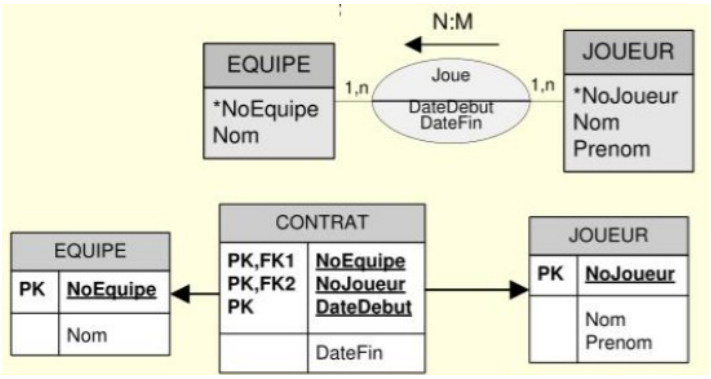
5. Les **relations de type 1:N** deviennent des **clés étrangères**. Une des deux tables reçoit, comme clé étrangère, la clé primaire de l'autre table. La table qui contient la clé étrangère est celle pour laquelle la clé étrangère ne reçoit qu'une seule valeur et qui correspond à l'entité dont la **cardinalité maximum est 1**. La point de la flèche est dirigée vers la table qui fournit la clé étrangère à l'autre table.

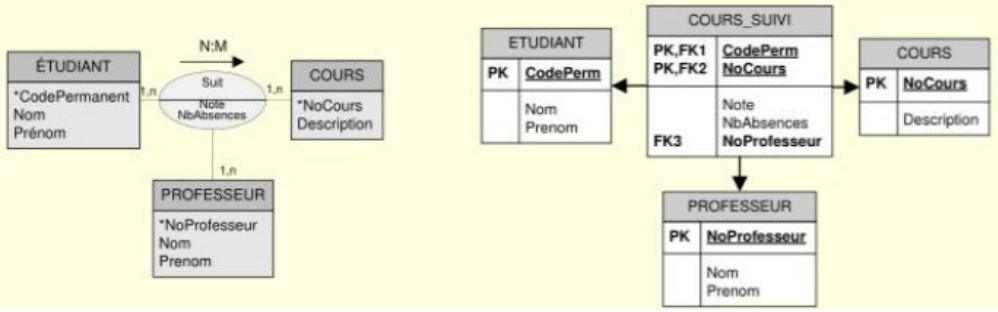


6. Les **relations de type N:N** deviennent des **tables supplémentaires**. Le nom de cette nouvelle table peut être la combinaison des noms des deux tables d'origine. La clé primaire d'une table supplémentaire est composée des clés primaires de chacune des tables à l'origine de la relation. Les attributs de la relation deviennent des colonnes de la nouvelle table.



Parfois la combinaison des deux clés n'est pas suffisante alors il faut ajouter autant d'attributs que nécessaires das la clé pour la rendre unique.





3) Normaliser le MCD

- **Première forme normale (1NF)**: Chaque **table** doit avoir une **clé primaire** et chaque **attribut** doit être **atomique**, c'est à dire ne doit contenir qu'**une seule information** à la fois *(il n'est donc ni multivalué ni composite)*. Elle est un pré-requis aux deux formes normales suivantes. Par exemple, la colonne "Nom_complet" doit être divisée en deux colonnes distinctes, "Prénom" et "Nom", ou

Exemple non 1NF

Nom	Numéros de téléphone
Alice	123-456-7890, 987-654-3210
Bob	555-123-4567
Carol	888-999-0000, 111-222-3333

Exemple 1NF

Nom	Numéro de téléphone
Alice	123-456-7890
Alice	987-654-3210
Bob	555-123-4567
Carol	888-999-0000
Carol	111-222-3333

- **Deuxième forme normale (2NF)**: Une table est en 2NF si elle est déjà en 1NF et si tous ses attributs non clés dépendent entièrement de la clé primaire. En d'autres termes, chaque colonne non clé doit dépendre de la totalité de la clé, pas seulement d'une partie.

Exemple non 2NF

Num_Commande	Client	Produit	Quantité
1	Alice	Stylo	5
2	Bob	Cahier	3
3	Alice	Gomme	2

Exemple 2NF

Num_Commande	Client
1	Alice
2	Bob
3	Alice

Num_Commande	Produit	Quantité
1	Stylo	5
2	Cahier	3
3	Gomme	2

Si on est pas en 2NF, la table « Commandes » a comme clé primaire "Num_Commande" mais la colonne "Client" dépend seulement partiellement de la clé primaire, car le client peut être

déterminé uniquement en regardant la "Num_Commande". Pour être en 2NF, il faut diviser cette table en deux : une table « Commandes » et une table « Détails_Commande », ce qui rend la colonne "Client" entièrement dépendante de la clé primaire dans la table « Commandes », et la table « Détails_Commande » devient également 1NF.

> *Une **clé candidate** est un ensemble minimal d'attributs qui peut être utilisé pour **identifier de manière unique** chaque enregistrement dans une table. Chaque enregistrement dans une table doit être unique par rapport à cette clé candidate, qui peut être constituée d'un ou plusieurs attributs. Elle peut être choisie comme clé primaire pour définir l'identifiant principal de l'entité, ou clé étrangère dans d'autres tables pour établir des relations.*

Pour être en 2NF, chaque attribut non clé d'une table doit dépendre de la totalité de la clé candidate, et non seulement d'une partie de celle-ci. Cela signifie que chaque attribut non clé doit être fonctionnellement dépendant de la clé candidate complète.

Dans la table « Détails_Commande », la clé candidate est l'ensemble d'attributs {Num_Commande, Produit}. Chaque attribut non clé (comme "Quantité") dépend de la clé candidate complète, car la quantité d'un produit dans une commande est spécifiée uniquement en fonction du numéro de commande et du produit.

Si la clé candidate était seulement "Num_Commande", alors "Produit" serait partiellement dépendant de la clé (puisque plusieurs produits peuvent être associés à une même commande), et cela ne satisferait pas la 2NF. En choisissant {Num_Commande, Produit} comme clé candidate, chaque attribut non clé dépend de la clé candidate complète, respectant ainsi la 2NF.

- **Troisième forme normale (3NF)**: Aucun attribut non-clé doit dépendre d'un autre attribut non-clé. Pour être en 3NF, une table doit être en 2NF et ne doit pas contenir de dépendances transitives, ce qui signifie que les attributs non clés ne doivent pas dépendre d'autres attributs non clés.

« Commandes »

Num_Commande	Client_ID
1	A1
2	B2
3	A1

« Clients »

Client_ID	Prénom	Nom	Adresse	Téléphone
A1	Alice	Dupont	123 Rue Principale	555-1234
B2	Bob	Martin	456 Rue Secondaire	555-5678

La clé étrangère "Client_ID" dans la table 'Commandes' était initialement liée à "Num_Commande", ce qui créait une **dépendance transitive**. Cependant, "Num_Commande" est la clé primaire de la table « Commandes », et cela impliquait une dépendance indirecte de "Client_ID" à la clé primaire "Num_Commande" de la table « Détails_Commande ».

Pour résoudre ce problème, il faur créer une troisième table « Clients » avec la clé primaire "Client_ID". En modifiant la table « Commandes » pour référencer directement "Client_ID", la dépendance transitive est éliminée. Ainsi, la structure de la base de données est conforme à la Troisième Forme Normale (3NF)."

https://www.ionos.fr/digitalguide/hebergement/aspects-techniques/normalisation-base-de-

donnees/

MCD	Relation	SGBD
Entité	Relation	Table
Association	Relation	Table
Identifiant	Clé primaire	Clé primaire
	Clé étrangère	Clé étrangère
Attribut, propriété	Attribut	Champs, colonne
Occurence, instance	Occurence	Occurence, enregistrement