



Introduction base de données

▼ Version	V1
≡ Type	Technique
📅 Date de création	@23 juillet 2023 23:12
📅 Dernière modification	@6 mars 2024 15:49

👋 Introduction

Qu'est-ce qu'une base de données ?

Une base de données est un ensemble d'informations (données) qui ont été stockées sur un support informatique de manière organisée et structurée afin de pouvoir facilement consulter et modifier leur contenu.

Une base de données est fort utile pour stocker les clients et les informations liées à ces derniers comme les coordonnées et les contacts. Cette même base devra pouvoir stocker des devis, des factures, etc. en lien avec ces clients. Cet ensemble de données constitue une base de données. On peut faire le parallèle avec une boutique e-commerce. La base de données d'un tel site doit pouvoir stocker toutes les informations du catalogue produits ainsi que celles liées aux clients et aux commandes.

Un SGBD est un logiciel permettant d'interagir avec les informations d'une base de données. On entend par interagir, sélectionner, ajouter, modifier et supprimer des données de la base. On regroupe généralement ces opérations sous l'acronyme **CRUD**.

📄 En bref :

- Les Systèmes de Gestion de Base de Données (SGBD) permettent de gérer les bases de données.
- MySQL est un SGBD.
- Le langage SQL est utilisé pour dialoguer avec MySQL.

Bases de données relationnelles

Les bases de données relationnelles stockent les données dans des tables structurées qui sont reliées entre elles par des relations. Chaque table est composée de lignes (aussi appelées enregistrements ou tuples) et de colonnes (attributs ou champs).

Le modèle relationnel utilise le langage SQL (Structured Query Language) pour la création, la manipulation et l'interrogation des données. Ce modèle est idéal pour assurer l'intégrité des données et faciliter les requêtes complexes grâce à des opérations comme les jointures de tables.

Les SGBD relationnels tels que MySQL, PostgreSQL et Oracle sont largement utilisés dans les applications d'entreprise pour gérer des données transactionnelles et assurer la cohérence et la fiabilité des informations.

Bases de données non relationnelles

Contrairement aux bases de données relationnelles qui stockent les données dans des tables structurées, les bases de données non relationnelles (NoSQL) utilisent une variété de modèles de stockage comme les

documents, les graphes, les clés-valeurs, et les colonnes larges. Elles sont conçues pour une grande scalabilité et une flexibilité des schémas de données.

Type	Système de Gestion de Base de Données (SGBD)
Relationnel	MySQL
Relationnel	PostgreSQL
Relationnel	Oracle DB
Relationnel	Microsoft SQL Server
Non Relationnel	MongoDB
Non Relationnel	Cassandra
Non Relationnel	Redis
Non Relationnel	Neo4j
Relationnel/Non Relationnel	SQLite

Organisation d'une BDD

Une base de données MySQL est organisée avec différentes tables. Une base de données contient **une ou plusieurs tables**, dont les noms doivent être uniques au sein de la base de la données. **Une table contient des colonnes. Les colonnes contiennent les données.**

id	prenom	nom	email	ville
1	Marine	Leroy	mleeroy@example.com	Paris
2	Jean	René	jrene@example.com	Lyon
3	Ted	Bundy	tbundy@example.com	Miami



Conception

Merise

 [Documentation MERISE](#)


 [Vidéo YouTube à propos de la conception de base de données](#) (les 5 premières vidéos sont suffisantes)


MERISE est une méthode française née dans les années 70, développée initialement par Hubert Tardieu. Elle fut ensuite mise en avant dans les années 80, à la demande du ministère de l'Industrie.

Modèle conceptuel des données (MCD)

Il s'agit de l'élaboration du modèle conceptuel des données (MCD) qui est une représentation graphique et structurée des informations. Le MCD est basé sur deux notions principales : **les entités et les associations**.

Élaboration

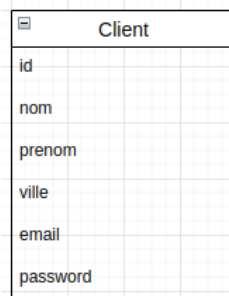
 [Draw.io](#) outil pour les conceptions de base de données simplement et rapidement.

 [MySQLWorkbench](#) super outil pour la conception de base de données complètes et professionnelles.

Les entités

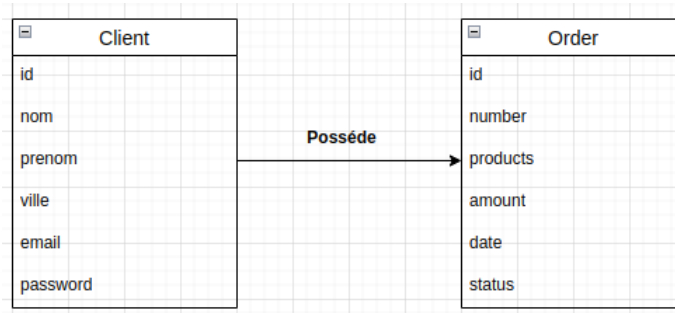
Chaque entité est unique et est décrite par un ensemble de propriétés. Une des propriétés de l'entité est l'identifiant (**id**). Cette propriété doit posséder des occurrences uniques. Bien souvent, on utilise une donnée de type entier qui s'incrémente pour chaque occurrence.

Ainsi, si on nous pouvons schématiser par exemple une entité « Client » comme ceci :



Les relations

Une association définit un lien sémantique entre une ou plusieurs entités. Généralement le nom de l'association est un verbe définissant le lien entre les entités qui sont reliées par cette dernière. Par exemple :



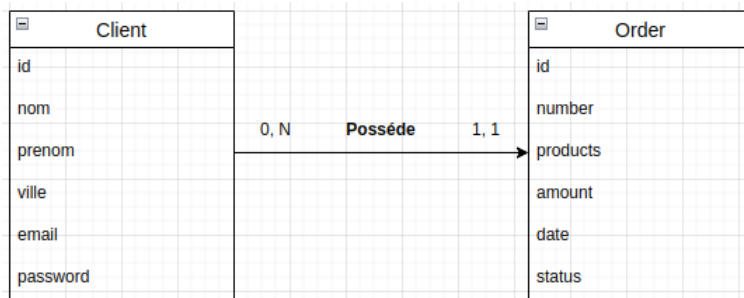
On définit des règles d'association à l'aide de cardinalités. Les cardinalités sont des caractères (0,1, n) qui fonctionnent par couple et qui sont présents de chaque côté d'une association. C'est grâce aux cardinalités que l'on est ensuite capable de créer soit des clés étrangères soit des tables intermédiaires.

- Les cardinalités minimales sont soit 0 soit 1. Elles servent surtout à déterminer si une occurrence est obligatoirement associée à une autre ou bien si elle est facultative.
- La cardinalité maximale est N, elle indique qu'une occurrence peut être associée à plusieurs autres.

Comprendre l'exemple suivant :

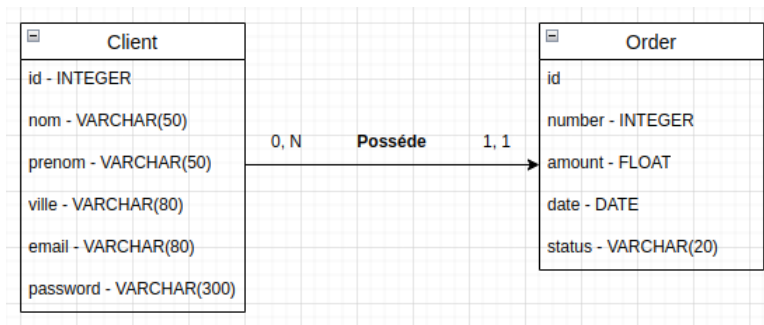
La cardinalité du côté gauche (0,N) indique que le client peut ne pas posséder de commande (**0,N**) ou en posséder plusieurs (**0,N**).

La cardinalité du côté droit (1,1) indique qu'une commande doit appartenir à minimum 1 client (**1,1**) et au maximum à un client (**1,1**).



🔥 Aller plus loin

Ajouter les types des données.



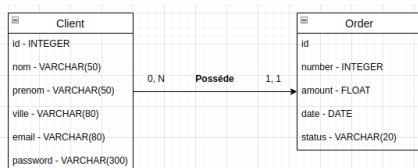
Modèle logique des données (MPD)

L'étape de création du **MPD** est presque **une formalité comparée à la création du MCD**. Il s'agit de faire évoluer sa modélisation de haut niveau pour la transformer en un schéma plus proche des contraintes des logiciels de bases de données.

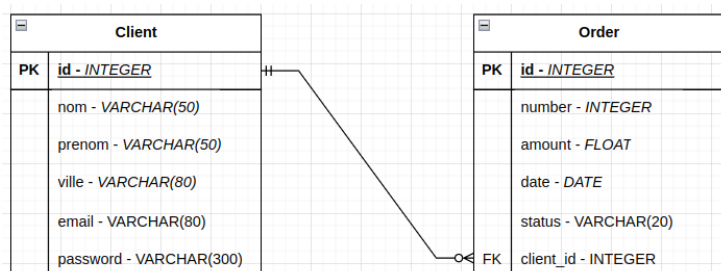
Il s'agit de préparer l'implémentation dans un SGBDR (système de gestion de base de données relationnelles).

- Chaque table dispose d'au minimum 1 clé dite **primaire**, il s'agit de l'identifiant unique dans la majorité des cas.
- Les relations et les cardinalités évoluent : il s'agit de créer des « **clés étrangères** » reliées à une « clé primaire » dans une autre table.

🎲 On passe de ça



🎉 À ça



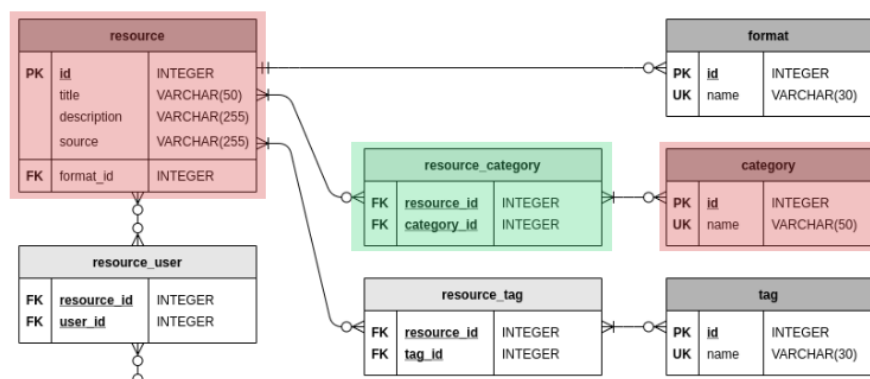
- Notre propriété `id` est devenue un champ unique grâce à l'annotation `PK` signifiant `primary key`. C'est grâce à ce champ que l'on identifiera les enregistrements dans une table
- Notre relation est devenue un champ dans la table `Order` grâce à l'annotation `FK` pour `foreign key`. C'est grâce à ce champ que l'on pourra retrouver le client d'une commande
- Notre relation est symbolisé par un flèche bien spécial, dans cet exemple elle symbolise :
 - Le client à 0 a N order (voir fin de la flèche collé à l'entité `Order`)
 - La commande à 1 a 1 client (voir fin de la flèche collé à l'entité `Client`)

i Dans la cadre d'une entité qui peut avoir x ou plusieurs lien avec un autre type d'entité

C'est une relation `many to many`, ici une ressources peut avoir plusieurs catégories et une catégories peut appartenir à plusieurs articles.

Dans le MCD la table intermédiaire (en vert) n'est pas présente, on l'ajoute à notre schema lors du MPD.

Une relation `many to many` a **forcément** une table intermédiaire reprenant la clé primaire `PK` de chacune des tables



Les différentes relations

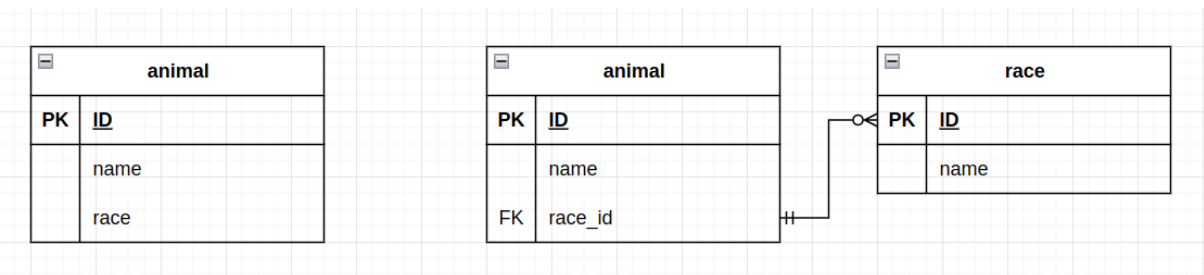
Les cardinalités présentent dans MCD vous indiquent qu'elle type de relation utiliser :

- **One to One (pas besoin de table intermédiaire)** : exemple un maire a une commune, une commune est dirigée par un maire
- **One to Many (pas besoin de table intermédiaire mais la table coté "one" stock la référence)** : exemple une commande à un seul propriétaire, un utilisateur peut avoir plusieurs commandes
- **Many to Many (table intermédiaire)** : exemple une école peut avoir plusieurs étudiants, un étudiants peut avoir plusieurs écoles

Normes qualitatives de votre BDD :

1. **Précision des données** : elle mesure à quel point les informations stockées dans la base de données reflètent fidèlement la réalité. Les erreurs, les doublons et les incohérences doivent être minimisés.
2. **Complétude** : la complétude évalue si toutes les données nécessaires sont présentes dans la base de données et si elles couvrent tous les aspects du domaine concerné. Des données manquantes peuvent compromettre l'efficacité de l'application ou de l'analyse basée sur la base de données.
3. **Cohérence** : une base de données doit être cohérente, c'est-à-dire que les informations similaires doivent être représentées de manière uniforme. Par exemple, si une même information est stockée à différents endroits, elle doit être identique partout.

4. **Intégrité des données** : l'intégrité des données concerne le respect des contraintes et des règles définies pour garantir la validité des données. Les contraintes d'intégrité (comme les clés primaires, les clés étrangères, etc.) doivent être correctement implémentées et respectées.
5. **Performance** : la performance évalue la rapidité avec laquelle la base de données répond aux requêtes et aux demandes d'accès aux données. Une base de données bien conçue doit être optimisée pour des performances élevées.
6. **Sécurité** : la sécurité des données est primordiale. Cela implique de mettre en place des mécanismes de protection pour éviter les accès non autorisés et prévenir la perte ou la corruption des données.
7. **Évolutivité** : une base de données de qualité doit être capable de s'adapter à l'évolution des besoins en données sans compromettre ses performances et sa fiabilité.



Par exemple : dans le cas d'une table "animal" contenant les informations d'un animal (par exemple pour une animalerie). On préfère avoir une table "animal" en relation avec une table "race" pour définir la race d'un animal.

Dans le cas où l'on stockerait la race sous forme de string dans la table "animal" si une race était amenée à changer de nom il faudrait modifier l'enregistrement de chaque animal utilisant la race à modifier. Cela provoque un problème d'évolutivité et de cohérence.