



An ordinary company with extraordinary agility

**DAT-ESBD-001**  
**L'Essentiel pour Comprendre et Exploiter**  
**une Base de Données**

# Introduction au module

# Présentation des modules



Data - Base de données



- Données et Famille de Données
- Structure d'une BDD
- Manipulation de données
- Caractéristique d'une BDD
- Bonnes pratiques



- Analyse
- Logique
- Rigueur
- Synthèse

A la fin de la formation, les stagiaires seront familiarisés avec les bases de données et **auront les bases pour les utiliser afin de construire des analyses**

# Introduction aux bases de données

# Introduction aux bases de données

Avant même de vous parler de données, j'aimerais vous parler du système d'information :

Un système d'information est un ensemble de méthode et outils permettant de collecter, stocker et analyser les informations.

Avant même l'informatique, nous parlions déjà de système d'information et on peut avoir un système d'information fonctionnel sans informatique.

Moi-même j'ai commencé lorsque j'étais magasinier auto avec des fiches **Bristol** (fig1) classées par ordre alphabétique, que l'on rangeait dans plusieurs **boîtes à casiers** (fig2).

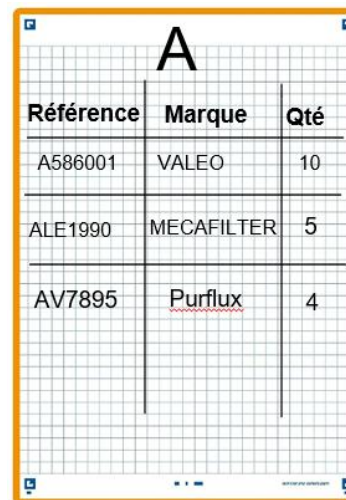
Chaque fiche Bristol était constituée de cellules (intersection entre les lignes et les colonnes) cela permettait de spécifier la référence, la marque ainsi que la quantité.

Il fallait une fiche pour chaque lettre soit 26 fiches au minimum car pour certaines lettres selon la quantité de données une fiche Bristol ne suffisait pas et 10 de plus pour les chiffres (0-9).

La quantité était notée au crayon de papier pour pouvoir la modifier selon une réception ou une vente.

Les mises à jour des fiches était fastidieuse.

(fig1)



Référence	Marque	Qté
A586001	VALEO	10
ALE1990	MECAFILTER	5
AV7895	<u>Purflux</u>	4

(fig2)



# Introduction aux bases de données



La technologie, l'informatique est juste un support de plus pour le système d'information. Aujourd'hui, l'aboutissement du stockage de l'information est le système de gestion de bases de données, le plus ancien, et je considère encore le plus robuste mais surtout le plus utilisé encore aujourd'hui sont les systèmes de gestions de bases de données relationnelles (SGBDR).

Un **Système de Gestion de Base de Données relationnelles** (SGBDR) est un logiciel qui permet de stocker des informations dans une base de données. Un tel système permet de lire, écrire, modifier, trier, transformer ou même imprimer les données qui sont contenus dans la base de données.

Parmi les logiciels les plus connus il est possible de citer : [Oracle](#), [MySQL](#), [MariaDB](#), [PostgreSQL](#) et [SQLite](#).

Une base de données est un ensemble de données structurées qui est constituée de **relations** appelées **Tables**.

On utilise le **SQL** qui est un langage de programmation pour interroger une base de données.

# Présentation des tables

Une application manipule des données et a donc besoin de les stockées, pour cela on utilise les bases de données.

Une base de données est constituée de tables que l'on peut comparer aux tableaux Excel.

Une table contient des enregistrements (lignes) et des champs (colonnes). Les champs ont des types de données différents, tels que du texte, des nombres, des dates et des liens hypertexte etc..

a_id	a_reference <small>Référence du code barre</small>	a_désignation	a_marque <small>Marque du produit</small>	a_quantite
1	7CK215G	Filtre à air	MAN FILTER	120
2	KT0WFQA	Filtre pollen	MECAFILTER	52
3	UICK6Z7	Filtre pollen	MAN FILTER	11

1. Un enregistrement (en vert) : contient des données spécifiques, telles que des informations sur un filtre à air.
2. Un champ (en rouge) : contient des données concernant un aspect du sujet de la table, par exemple la référence, la désignation.
3. Une valeur de champ (en jaune) : chaque enregistrement a une valeur de champ. Par exemple,

## Modèle relationnel

Afin de créer des bases de données, il a été défini un modèle relationnel permettant d'établir des contraintes de construction de celles-ci. Ces contraintes ont pour but de limiter le plus possible les problèmes à gérer plus tard. On parle de règles d'intégrité structurelle.

- Une contrainte d'intégrité est une règle qui définit la cohérence d'une donnée ou d'un ensemble de données de la BDD.
- Il existe deux types de contraintes :
- sur une colonne unique, ou sur une table lorsque la contrainte porte sur une ou plusieurs colonnes.
- Les contraintes sont définies au moment de la création des tables.

## Contrainte d'intégrité sur une colonne

- PRIMARY KEY : définit l'attribut comme la clé primaire
- UNIQUE : interdit que deux tuples de la relation aient la même valeur pour l'attribut.
- REFERENCES <nom table> (<nom colonnes>) : contrôle l'intégrité référentielle entre l'attribut et la table et ses colonnes spécifiées.

### Exemple :

Pour chaque information d'une table A qui fait référence à une information d'une table B, l'information référencée existe dans la table B,

## Modèle relationnel

### Contrainte d'intégrité sur une colonne

- **PRIMARY KEY** : Définit l'attribut comme clé primaire, garantissant l'unicité de chaque entrée.
- **UNIQUE** : Assure que les valeurs de la colonne soient uniques dans la table.
- **REFERENCES** : Maintient l'intégrité référentielle en s'assurant que les données référencées existent dans la table cible.
- **CHECK** : Vérifie la validité des données selon une condition, par exemple en restreignant les salaires à une certaine plage.

### Contrainte d'intégrité sur une table

- **PRIMARY KEY** : Définit plusieurs attributs comme clés primaires si nécessaire.
- **UNIQUE** : Assure que l'ensemble des valeurs pour une liste d'attributs est unique.
- **FOREIGN KEY** : Maintient l'intégrité entre les tables en s'assurant que les valeurs correspondent aux clés primaires dans la table référencée.

# Conseils et conventions

## Conseils et conventions

Pour le nom des tables, des champs ou des bases de données n'utilisez jamais d'espaces ou d'accent.

Pour remplacer un espace mettre un Underscore (touche 8).

Ex: lieu\_de\_naissance

Rester cohérent par exemple mettre toutes les tables au singulier ou toutes au pluriel.

Donner des noms qui ont du sens cela vous simplifiera la tâche par la suite.

Dans MySQL tous les noms (attributs) doivent être en minuscules.

Lorsque l'on écrit une requête SQL, les mots clés sont écrit en majuscules pour bien les différencier.

## Noms de tables

Préfixer les noms des tables permet d'éviter d'utiliser accidentellement des mots réservés.

Permet d'éviter les conflits lorsqu'il y a plusieurs logiciels similaires sur une même base de données (par exemple, si 2 logiciels utilisent chacun une table intitulée "utilisateur").

Utile pour séparer facilement les tables associées à un système ou à un autre. Par exemple si un blog WordPress et une boutique e-commerce Prestashop sont placés sur une même base de données, le blog aura des tables commençant par "wp\_" tandis que la boutique aura des tables commençant par "ps\_".

## Conseils et conventions

### Noms de tables

- C'est plus simple pour ré-installer un backup. Par exemple, pour réinstaller une sauvegarde du blog, il est possible d'ajouter des tables commençant par "wp2013\_" puis de modifier le code de l'application pour tout migrer d'un coup.
- Sur des gros projets ça peut être pratique pour que toutes les tables associées aux utilisateurs commencent par exemple par "user\_", toutes celles concernant les produits par "product\_" et ainsi de suite.

### Noms de colonnes

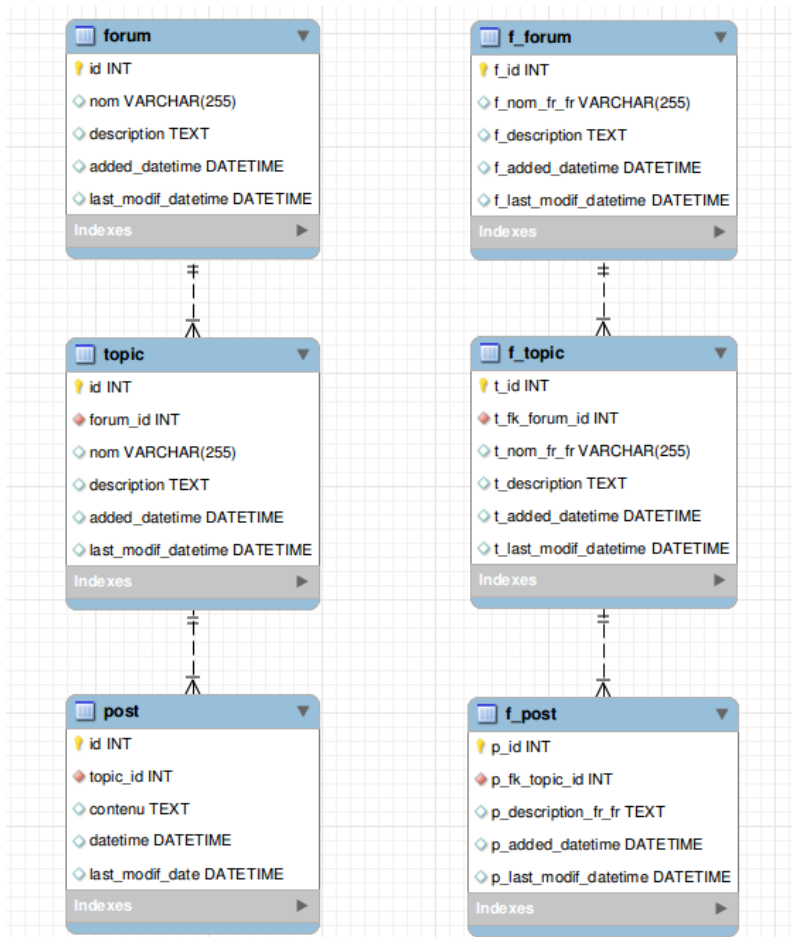
**Préfixer toutes les colonnes** de la même façon pour chaque table. C'est beaucoup plus pratique lorsqu'il convient d'effectuer des jointures.

Lorsqu'une clé étrangère est utilisée il est pratique de l'indiquer dans le nom de la colonne. La colonne peut contenir le préfixe, puis "fk" pour Foreign Key, puis le nom de la table et enfin se terminer par "id". Ainsi, une colonne pourrait s'intituler "wp\_fk\_user\_id" (cf. préfixe "wp", foreign key sur la table utilisateur de la colonne "id").

# Conseils et conventions

## Conseils et conventions

Imaginons les tables d'un forum. Il peut y avoir 3 tables : une pour les forums, une autre pour les questions et une dernière pour les messages. Ces tables sont liées entre elles grâce à des clé étrangères.



L'exemple que je recommande est l'exemple de droite. Cet exemple utilise des préfixes et respecte une bonne convention de nommage pour savoir ce que peut contenir les colonnes.

## Conseils et conventions

### Requêtes

Voici l'exemple d'une requête avec le mauvais nommage :

```
SELECT `post`.`id` AS post_id, `post`.`contenu` AS post_contenu, `topic`.`id` AS topic_id, `topic`.`nom` AS topic_nom  
FROM `post`  
INNER JOIN `topic` ON `topic`.`topic_id` = `post`.`id`
```

Cette requête est un peu longue et compliquée. Il faut par exemple utiliser le nom de la table ou un alias pour éviter des erreurs où SQL ne sais pas différencier quelle colonne est appelée. C'est indispensable pour éviter l'erreur : “nom de colonne ambigu”.

Cette requête SQL peut être simplifiée en utilisant des règles de nommages:

```
SELECT `p_id`, `p_description_fr_fr`, `t_id`, `t_nom_fr_fr`  
FROM `f_post`  
INNER JOIN `f_topic` ON `t_id` = `p_fk_topic_id`|
```

## Type de données

### Choisir le meilleur type possible afin de ne pas entraîner :

- un gaspillage de mémoire (ex. : si vous stockez de toutes petites données dans une colonne faite pour stocker de grosses quantités de données) ;
- des problèmes de performance (ex. : il est plus rapide de faire une recherche sur un nombre que sur une chaîne de caractères) ;
- un comportement contraire à celui attendu (ex. : trier sur un nombre stocké comme tel, ou sur un nombre stocké comme une chaîne de caractères ne donnera pas le même résultat) ;
- l'impossibilité d'utiliser des fonctionnalités propres à un type de données (ex. : stocker une date comme une chaîne de caractères vous prive des nombreuses fonctions temporelles disponibles).