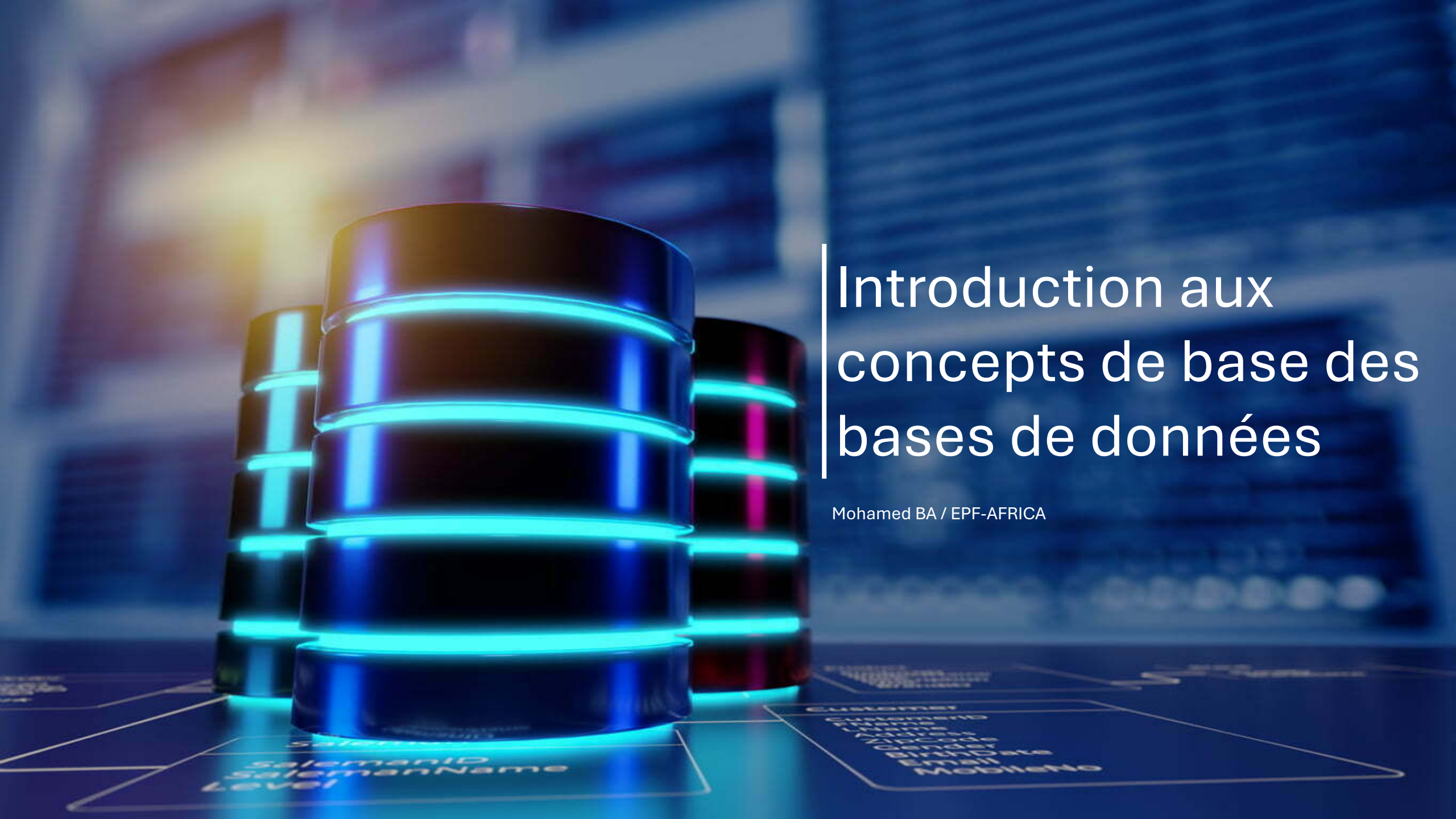
Microsoft®
SQL Server®



Introduction aux concepts de base des bases de données

Mohamed BA / EPF-AFRICA

SOMMAIRE

I. INTRODUCTION GENERALE

❖ **Notions intuitives**

- Base de données

Ensemble structuré de données apparentées qui modélisent un univers réel.
Une BD est faite pour enregistrer des faits, des opérations au sein d'un organisme (administration, banque, université, hôpital, ...).

- Système de Gestion de Base de Données (SGBD)

DATA BASE MANAGEMENT SYSTEM (DBMS)
Système qui permet de gérer une BD partagée par plusieurs utilisateurs simultanément.

I. INTRODUCTION GENERALE

❖ **Objectifs et avantages des SGBD**

- Que doit permettre un SGBD ?

Décrire les données (indépendamment des applications - de manière intrinsèque).

⇒ langage de définition des données (Data Definition Language - DDL).

- Manipuler les données

Interroger et mettre à jour les données via un langage de requêtes déclaratif
Exemple : quels sont les noms des produits de prix < 100F ?

⇒ langage de manipulation des données (Data Manipulation Language - DML).

I. INTRODUCTION GENERALE

❖ Objectifs et avantages des SGBD

- Contrôler les données
  - Intégrité

Vérification de contraintes d'intégrité.

Exemple : Le salaire doit être compris entre 400.000F et 2.000.000F

  - Confidentialité

Contrôle des droits d'accès et autorisations.

⇒ Langage de contrôle des données (Data Control Language - DCL).

- Partage

Une base de donnes est partagée entre plusieurs utilisateurs en même temps.

⇒ Contrôle des accès concurrents

I. INTRODUCTION GENERALE

- Objectifs et avantages des SGBD

  - Sécurité

Reprise après une panne, journalisation.

  - Performances d'accès

index (hashage, arbres balancés ...)

  - Indépendance physique

Pouvoir modifier les structures de stockage ou les index sans que cela n'ait de répercussion au niveau des applications.

  - Indépendance logique

Permettre aux différentes applications d'avoir des vues différentes des mêmes données.

II. CONCEPTS FONDAMENTAUX

- Une base de données relationnelle, est un ensemble structuré de données organisées en tables, qui peuvent être traitées à l'aide d'un langage de requête structuré. C'est le contraire d'une base de données NoSQL.

- Chaque ligne d'une table représente une entité de données et chaque colonne définit un champ d'information (attribut) spécifique.

- Les données contenues dans ces tables peuvent être consultées, gérées, modifiées, mises à jour, contrôlées et organisées à l'aide d'un langage conçu pour interagir avec la bases de données, SQL.

- De nombreux systèmes de bases de données adoptent un langage de programmation comme SQL mais intègrent souvent des extensions propriétaires uniques spécifiques à leurs plateformes. Toutefois, les commandes SQL fondamentales, comme « SELECT », « INSERT » et « CREATE », sont universellement applicables à la plupart des opérations de base de données.

II. CONCEPTS FONDAMENTAUX

- Le langage SQL permet d'exécuter des fonctions essentielles de gestion des données, mais il peut également servir à construire des requêtes complexes afin de transformer les données brutes disponibles en informations utiles et contextuelles.

En programmant une requête en langage SQL, les utilisateurs peuvent :

- Effectuer des requêtes ciblées dans une base de données.
- Extraire ou modifier des enregistrements et des données existants dans une base de données.
- Ajouter de nouvelles entrées à une base de données.
- Supprimer les enregistrements obsolètes de la base de données.
- Créer de nouvelles bases de données ou de nouvelles tables dans une base de données existante pour améliorer l'organisation et l'optimisation des données.
- Développer des procédures stockées et des vues dans une base de données.
- Attribuer des droits d'accès aux procédures, vues, tables et divers ensembles de données d'une base de données.

II. CONCEPTS FONDAMENTAUX

Les bases de données SQL restent l'un des systèmes les plus populaires et les plus utilisés pour le stockage et le traitement de données hautement structurées.

Elles sont au cœur de nombreux systèmes backend d'entreprise et indispensables au bon déroulement de l'ère électronique.

Marketing

Dans le domaine du marketing, la programmation SQL est un outil précieux pour l'analyse des données. Par exemple, les responsables marketing comme Anne s'en servent pour naviguer dans les vastes bases de données clients. Il s'agit notamment d'extraire des informations clés, comme l'historique des achats ou les données démographiques des clients, pour affiner les stratégies marketing et cibler les campagnes plus efficacement.

Finance

Le secteur financier s'appuie sur SQL pour l'analyse des données de vente et les prévisions. Les professionnels de la finance comme John utilisent ce langage pour identifier les tendances et les anomalies dans les ventes afin d'élaborer des stratégies et des plans financiers en connaissance de cause.

Services médicaux

La programmation SQL est également appliquée dans le secteur des soins de santé, notamment pour la gestion des données des patients. Ce langage permet aux organisations de suivre l'évolution des maladies et d'allouer les ressources de manière efficace. Il facilite également la gestion des soins aux patients et l'analyse des soins de santé, en plus d'améliorer les résultats pour les patients et l'efficacité des services.

Développement d'applications

Pour le développement mobile et web, l'utilisation des instructions SQL est essentielle au traitement des données et permet de garantir un stockage sécurisé. Ce langage favorise l'intégration des bases de données dans les applications, ce qui contribue au bon fonctionnement et à la sécurité de ces plateformes.

Analyse

L'analyse de données repose en grande partie sur le SQL pour extraire et analyser de grands ensembles de données. Ainsi, les data analysts parviennent à découvrir des tendances et des informations qui éclairent les stratégies et la prise de décision de l'entreprise. Les business analysts utilisent les instructions SQL pour la collecte et l'analyse des données, ainsi que pour la création de tableaux de bord de gestion. L'intégration de SQL à des outils comme Tableau et PowerBI est cruciale pour présenter des données complexes dans un format accessible.

Data science

En data science, le langage SQL est essentiel pour des tâches telles que l'extraction de données, le prétraitement et l'analyse exploratoire des données. Il prend en charge des applications en temps réel, notamment la tarification dynamique et la détection de la fraude, en traitant efficacement de grands volumes de données.

III. DONNEES, ATTRIBUTS, TABLES ET BASES DE DONNEES

❖ Qu’est-ce qu’une donnée

- Définition

Unité élémentaire d’information, capturée et stockée dans des systèmes informatiques.

- Types

BINARY, INTEGER, DATETIME, DECIMAL, IMAGE, TEXT, …

❖ Qu’est-ce qu’un attribut

- Définition

Caractéristique d'une entité susceptible d'être enregistrée dans la base de données.

Les colonnes nom, adresse, telephone sont des attributs d’une entité.
Vous ne pouvez pas avoir de table sans attributs.
Vous pouvez créer une table vide qui a des attributs définis, mais aucun enregistrement.

III. DONNEES, ATTRIBUTS, TABLES ET BASES DE DONNEES

❖ **Qu'est-ce qu'une table**

- Définition

Les tables sont des objets essentiels dans une base de données, car ils conservent toutes les informations ou données.

Exemple : Une base de données d'une entreprise peut contenir une table *Contacts* qui stocke les noms des fournisseurs, leurs adresses de messagerie et numéros de téléphone.

❖ **Qu'est-ce qu'une base de données**

- Définition

Collection organisée d'objets structurés ou bruts (NoSQL).
Permet de stocker et de retrouver des données.

Une collection de bases de données constitue une banque de données.

III. DONNEES, ATTRIBUTS, TABLES ET BASES DE DONNEES

❖ **Qu'est-ce qu'une entité**

- Définition

Notion en rapport avec le domaine d'activité pour lequel la base de données est utilisée.

Exemple : Des personnes, des produits, des commandes, des réservations, ...

❖ **Cardinalité**

- Définition

Entre 2 entités A et B, la cardinalité est le nombre de A pour lesquelles il existe un B et inversement.

- Types

un-à-un, un-à-plusieurs ou plusieurs-à-plusieurs.

Un compte bancaire appartient à un seul client, et un client peut avoir plusieurs comptes.

III. DONNEES, ATTRIBUTS, TABLES ET BASES DE DONNEES

❖ **Les contraintes**

- NULL

La valeur null indique que la valeur de l'attribut est absente, non disponible ou inapplicable.

- NOT NULL

Indique que la valeur de l'attribut ne peut pas être nulle.

- UNIQUE

Les valeurs des attributs sont différentes.

- PRIMARY KEY

Combinaison de NOT NULL et UNIQUE. Identifie chaque enregistrement de façon unique.

- FOREIGN KEY

Une clé étrangère met en relation deux tables.

III. DONNEES, ATTRIBUTS, TABLES ET BASES DE DONNEES

❖ Les contraintes

- CHECK

Utilisée pour s'assurer que la valeur d'un attribut respecte une condition.

Exemple :

```
        Age int,
    CHECK (Age>=18)
        );
```

- DEFAULT

Permet d'assigner une valeur par défaut a un attribut.

Exemple :

```
        Age int,
        Ville varchar(255) DEFAULT Dakar'
);
```

III. DONNEES, ATTRIBUTS, TABLES ET BASES DE DONNEES

❖ Les contraintes

- CREATE INDEX

Utilisée pour accélérer la récupération de données.

❖ Cardinalité

- Définition

Entre 2 entités A et B, la cardinalité est le nombre de A pour lesquelles il existe un B et inversement.

- Types

un-à-un, un-à-plusieurs ou plusieurs-à-plusieurs.

Un compte bancaire appartient à un seul client, et un client peut avoir plusieurs comptes.

IV. LE MODELE RELATIONNEL

Le modèle relationnel est un modèle dans lequel les données sont organisées sous forme de tables qui sont associées entre elles. Une table correspond à une relation.
Ce modèle permet d'indiquer toutes les caractéristiques d'une table de données, avec notamment les liens entres les différentes relations.

❖ Principes

- Schéma d'une relation

Une relation est une table qui contient des données, chaque colonne de cette table va correspondre à un attribut de celle-ci, qui permet d'identifier une information stockée dans cette relation. Une ligne de la relation est appelée un enregistrement.
On appelle schéma d'une relation la formule : NOM_RELATION(NOM_ATTRIBUT1, ...).

Exemple : Si on considère un ensemble de livres, on peut construire la relation LIVRES avec comme attributs l'ISBN, le titre, le nom de l'auteur et l'année de publication. Le schéma de cette relation est :
LIVRES(ISBN, titre, nom_auteur, annee_publication).

IV. LE MODELE RELATIONNEL

❖ **Règles**

Voici les règles que la relation doit respecter pour éviter des erreurs que l'on nomme incohérences dans le langage des bases de données.

Règle n° 1

Un attribut ne peut prendre à un instant donné qu'une seule et unique valeur. On dit qu'il y a atomicité de l'attribut.

Règle n° 2

Il n'y a qu'un nombre fini d'enregistrements, on ne peut par exemple pas avoir la liste des nombres entiers.

IV. LE MODELE RELATIONNEL

Règle n° 3
Lorsque la base de données est définie, on définit le domaine de chacun des attributs, c'est-à-dire de quel type doivent être les données ou à quel intervalle les valeurs doivent appartenir.
On appelle cela l'intégrité de domaine.

Règle n° 4
Les enregistrements doivent tous être différents, c'est-à-dire qu'il ne peut pas y avoir deux fois la même ligne.

Exemple : LIVRES(Identifiant numerique, Titre, Auteur, Annee)
Le schéma de cette relation peut être représenté par la table suivante.

Relation	LIVRES			
Attribut	Identifiant numerique	Titre	Auteur	Annee
Valeur	1	MS SQL Server	Python	2020

IV. LE MODELE RELATIONNEL

Règle n° 5
Toute relation a une clé primaire.
Pour signaler la clé primaire dans le schéma de la relation associée, on peut souligner l'attribut qui correspond à cette clé primaire. On parle d'intégrité de clé.

Exemple : Dans la relation LIVRES précédente, le seul attribut qui correspond à une clé primaire est l'identifiant numérique, puisque des livres différents peuvent avoir un même auteur, une même année de publication et même un même titre.
On a donc le schéma de relation :
LIVRES(Identifiant numerique, Titre, Auteur, Annee).

IV. LE MODELE RELATIONNEL

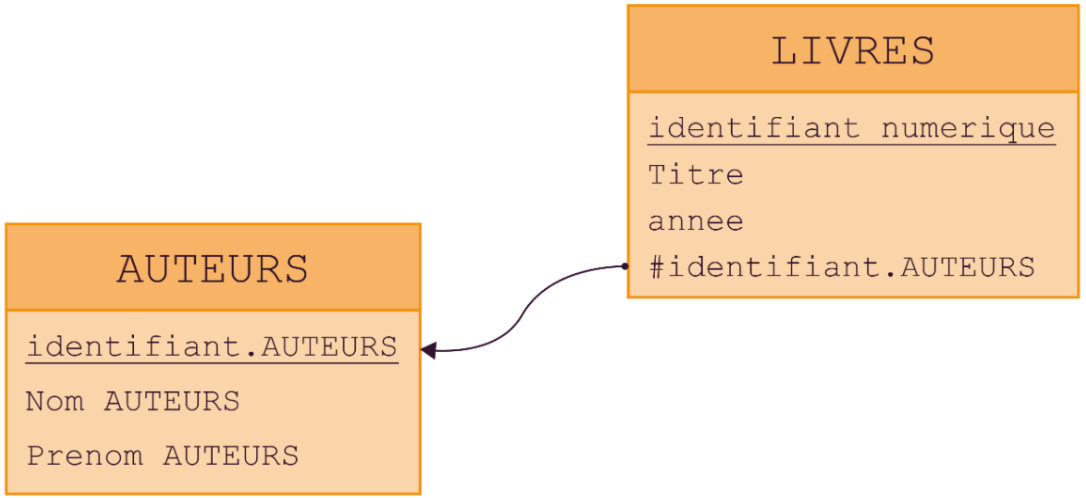
Règle n° 6
Une clé étrangère (Foreign Key en anglais) est un attribut qui n'est pas primaire pour la relation étudiée mais qui est la clé primaire d'une autre relation. On parle d'intégrité référentielle.
Pour signaler la clé étrangère dans le schéma de la relation associée, on peut soit placer un # devant l'attribut qui correspond à cette clé étrangère, soit surligner cet attribut en gris.

Exemple : On considère les deux schémas de relation suivants
LIVRES(identifiant numerique, Titre, #identifiant.AUTEURS, annee)
AUTEURS(identifiant.AUTEURS, Nom AUTEURS, Prenom AUTEURS)
identifiant.AUTEURS est une clé primaire de la relation AUTEURS et une clé étrangère de la relation LIVRES.

IV. LE MODELE RELATIONNEL

❖ Schéma relationnel

Le schéma relationnel correspond à l'ensemble des relations présentes dans une base de données. Le schéma relationnel d'une base de données est composé de l'ensemble des schémas de relation de cette base.



V. ALGEBRE RELATIONNEL

❖ Définition

On peut voir l'algèbre relationnelle comme un langage de programmation très simple qui permet d'exprimer des requêtes sur une base de données relationnelle.

❖ Opérateurs

L'algèbre relationnel se compose d'un ensemble d'opérateurs, parmi lesquels 6 sont nécessaires et suffisants et permettent de définir les autres par composition.

V. ALGEBRE RELATIONNEL

- La projection, π

La projection $\pi_{A_1, A_2, \ldots, A_k}(R)$ s'applique à une relation R, et construit une relation contenant tous les nuplets de R, dans lesquels seuls les attributs A_1, A_2, A_k sont conservés. La requête suivante construit une relation avec le nom des logements et leur lieu.

$$\pi_{nom,lieu}(logement)$$

En SQL, le projection s'exprime avec le select suivi de la liste des attributs à projeter.
Exemple :

```
select nom, lieu from Logement;
```

Si on souhaite conserver tous les attributs, on peut éviter d'en énumérer la liste en la remplaçant par le caractère *.
Exemple :

```
select * from Logement;
```

V. ALGEBRE RELATIONNEL

○ La sélection, σ

La sélection $\sigma_F(R)$ s'applique à une relation, R, et extrait de cette relation les nuplets qui satisfont un critère de sélection, F. Ce critère peut être :

- La comparaison entre un attribut de la relation, A, et une constante a. Cette comparaison s'écrit $A\theta a$, où θ appartient à {=,<, >, ≤ ,≥}.
- La comparaison entre deux attributs A_1et A_2, qui s'écrit avec les mêmes opérateurs de comparaison que précédemment.

Exemple : exprimer la requête qui donne tous les logements a Dakar.

$$\sigma_{lieu='Dakar'}(logement)$$

En SQL, les critères de sélection sont exprimés par la clause where.

```
select * from Logement where lieu = 'Dakar';
```

V. ALGEBRE RELATIONNEL

- Le produit cartésien, $\times$

Le premier opérateur binaire, et le plus utilisé, est le produit cartésien, $\times$. Le produit cartésien entre deux relations R et S se note R $\times$ S, et permet de créer une nouvelle relation où chaque nuplet de R est associé à chaque nuplet de S.

En SQL, le produit cartésien est un opérateur cross join intégré à la clause from.

Exemple :

```
select * from R cross join S;
```

V. ALGEBRE RELATIONNEL

- Le renommage

Quand les schémas des relations R et S sont complètement distincts, il n'y a pas d'ambiguïté sur la provenance des colonnes dans le résultat. Par exemple on sait que les valeurs de la colonne A dans $R \times S$ viennent de la relation R. Il peut arriver (il arrive de fait très souvent) que les deux relations aient des attributs qui ont le même nom. On doit alors se donner les moyens de distinguer l'origine des colonnes dans la relation résultat en donnant un nom distinct à chaque attribut.

En SQL, le renommage est obtenu avec le mot-clé as.

Exemple :

```
select * from R as R1 cross join R as R2;
```

V. ALGEBRE RELATIONNEL

- L'union, ∪

Il s'agit d'un operateur binaire moins fréquemment utilisé.
L'expression R ∪ S crée une relation comprenant tous les nuplets existant dans l'une ou l'autre des relations R et S. Il existe une condition impérative : les deux relations doivent avoir le même schéma, c'est-à-dire même nombre d'attributs, mêmes noms et mêmes types.

En SQL, le renommage est obtenu avec le mot-clé union.

Exemple :
select lieu **from** Logement **union select** région **as** lieu **from** Voyageur;

V. ALGEBRE RELATIONNEL

- La différence, –

Comme l'union, la différence s'applique à deux relations qui ont le même schéma. L'expression R – S a alors pour résultat tous les nuplets de R qui ne sont pas dans S.

En SQL, la différence est obtenue avec except.

Exemple :
select A, B **from** R **except select C as** A, D **as** B **from** S;

La contrainte d'identité des schémas rend cet opérateur très peu pratique à utiliser, et on lui préfère le plus souvent la construction logique du SQL « déclaratif », not exists.

V. ALGEBRE RELATIONNEL

- La jointure

En principe, on pourrait donc s'en contenter. En pratique, il existe d'autres opérations, très couramment utilisées, qui peuvent se construire par composition des opérations de base. La plus importante est la jointure.
Initialement, SQL ne proposait pour effectuer la jointure que la version déclarative.

```
select * from Logement as l, Activité as a where l.code=a.codeLogement;
```

> Il existe ici une alternative.
> En 1992, la révision de la norme a introduit l'opérateur algébrique qui, comme le produit cartésien, et pour les mêmes raisons, prend place dans le from.

```
select * from Logement join Activité on (code=codeLogement);
```