**- Introduction Base de Données :**

1. Définition:

**Base de données :** Ensemble structuré et partagé de données en relation logique qui permet de répondre aux besoins d’une organisation.

**SGBD :** [*Système de Gestion de Bases de Données*], logiciel qui permet de gérer des bases de données.

*Exemple: SQLServer / MariaDB / Access / Oracle / ...*

**B.D. relationnelles :** Bases de données ou les données sont organisées dans des tables (*ou relations*) liées entre elles, selon le modèle introduit par Edgar F. Codd en 1970.

**SGBDR :** Système de Gestion de Bases de Données Relationnelles.

**Clé étrangères:** Attribut qui est clé primaire dans une autre table.

**Fonctions d’un SGBD :**

- ll permet de créer une base de données grâce à un langage de définitions de données (LDD)

- Il permet de mettre à jour et retrouver les données grâce à un langage de manipulation de données (LMD)

- Il assure la sécurité des données.

- Il permet l’accès partagé de données.



**B.D.D.**





*Pourquoi une base de données relationnelle ?*

- Avant les bases de données, il y avait les fichier cependant il y a quelques problèmes avec ceci :

1. **Redondance** d’information (*une même donnée est répétée plusieurs fois.*).
2. Risque **d’incohérence**.
3. Accès aux données difficile et long.
4. Problèmes de mise à jour.

- Maintenant, nous utilisons les bases de données pour éviter tous ces problèmes.

1. Structure d’une table:

| Narbonne | Aude |
| --- | --- |
| Toulouse | Midi P |
| Montpellier | Hérault |
| Carcassonne | Aude |

- **Tables:** tableau de données.

- **Attributs:** Colonnes du tableau.

- **Tuples:** Ligne du tableau.

- **Domaine des attributs:** un attribut peut prendre un nombre fini de valeurs. L’ensemble de ces valeurs est appelé le domaine.

1. Structure de la base de données relationnelle (*SR ou MRD*):

3.1/ Définition:

- Afin de décrire les données dans la base, tous ces logiciels se basent sur le Schéma Relationnel SR => Représentation de la structure de chaque table de la base de données.

JOUEURS (Code license, Nom, Prénom, Ville naissance)

VILLES (Nom ville, Département ville)

3.2/ Règles de modélisation:

Règle n°1:

- Tous les attributs d’un S.R. sont **atomiques** (*ex: pas de second prénom dont il faut un autre attribut*).

Règle n°2:

- Chaque table possède une **clé primaire**.

Une clé primaire est un attribut, ou un *ensemble d’attributs*,qui permet d’identifier chaque tuple de la table de facon unique. La clé primaire ne peut pas être nulle.

*Ex: Code license*

*Remarque: Dans une table deux tuples ne peuvent pas avoir la même clé primaire.*

JOUEURS (Code licence, Nom, Prénom, ville naissance)

1 Pipo

2 Pipo

~~1 Marcel~~

VILLES (Nom ville, Département ville)

Montpellier Hérault

~~Montpellier Midi-Pyrénées~~

Béziers Hérault

Extension de la règle n°2:

- Lorsqu’on a le choix entre plusieurs clés candidates on privilégiera la clé la plus stable et la plus simple.

3.3/ Liens entre les différentes tables:

- les liens entre les tables sont réalisés par les **clé étrangères**.

**Clé étrangères:** Attribut qui est clé primaire dans une autre table.

*La clé étrangères peut-être nulle.*

*Exemple:*

Lien | Ville naissance - Nom bille | => Représenté par #

JOUEURS (Code licence, Nom, Prénom, ville naissance #)

VILLES (Nom ville, Département ville)

Les clés étrangères permettent de représenter deux types d’associations:

- Association de type *un-plusieurs.*

- Association de type *plusieurs-plusieurs*.

Association de type *un-plusieurs*:

JOUEURS (Code license, Nom, Prénom, ville naissance #)

VILLES (Nom ville, Département ville)

*Un joueur a au maximum une ville de naissance.*

*Dans une ville, plusieurs joueurs peuvent naître.*

PROFS (N° prof, Nom prof)

MATIERES (Code matière, libellé matière, N° prof #)

*Un prog enseigne plusieurs matière*

*Une matière peut être enseignée que par un profs au maximum*

Association de type *plusieurs-plusieurs*:

PROFS (N° prof, Nom prof)

MATIERES (Code matière, libellé matière)

ENSEIGNER (N° prof #, Code matières #)

*Un prof peut enseigner plusieurs matières*

*Une matière peut être enseignée par plusieurs profs*

3.4/ Représentation graphique: MRD





1. Les contraintes d’intégrité:

- Les contraintes d’intégrité sont des règles qui garantissent la cohérence des données pendant toute la durée de vide de la base de données. On peut classifier ces contraintes selon trois grandes catégories:

**Les contraintes d’intégrité de domaine (*ou de donnée*):**

Elles concernent la valeur d’un seul attribut:

- Règle de gestion

- Valeur obligatoire ‘NOT NULL’

- Format

- Type de données: *numérique, ...*

**Les contraintes d’intégrité de relation:**

Les valeurs d’un tuple sont des valeurs des autres tuples dans la même table.

**Les contraintes d’intégrité référentielle (CIR):**

Cette contrainte permet d’assurer la cohérence des clés étrangères d’une base de données, notamment dans le cas d’une insertion, d’une mise à jour ou bien d’une suppression (*dilemme de l’interdiction ou de la suppression en cascade*).

La valeur de la clé étrangère doit exister en tant que clé primaire dans la table reliée.

*Exemple:* Tous les prof de la table ENSEIGNER doivent obligatoirement exister dans la table PROFESSEURS.

**Mise à jour en cascade :** Il est impossible de changer une valeur clé primaire si ce tuple possède un tuple relié

**Suppression en cascade :** Il est impossible de supprimer un tuple dans la table primaire s’il existe des tuples reliés.

=> Les SGBDR permettent de gérer les contraintes d’intégrité de base de données relationnelle.