

# Gestion des données

## Bases de données relationnelles

Olivier Schwander  
<olivier.schwander@sorbonne-universite.fr>

Master Statistiques  
Sorbonne Université

2022-2023

# SGBD - Systèmes de Gestion de Bases de Données

## Serveur

- ▶ Stocke les données
- ▶ Reçoit et interprète les requêtes des clients
- ▶ Gère le contrôle d'accès
- ▶ Gère les accès concurrents

## Client

- ▶ Application qui utilise la base de donnée

## Application

- ▶ Programme classique, contenant un client
- ▶ Bibliothèques dans des langages de programmation
- ▶ Fournit une interface avec la base

# Abstraction du stockage physique

## Serveur de bases de données

- ▶ Reçoit les requêtes par le réseau

## Moteur de base de données

- ▶ Stockage concret des données
- ▶ Indexation pour accélérer les requêtes
- ▶ Journalisation des écritures

# Langage standardisé et normalisé

## Théorie

- ▶ Une application conçue pour un SGBD doit marcher avec un autre

## Pratique

- ▶ Délicat...

## Pourquoi ?

- ▶ Extensions de la syntaxe
- ▶ Parties de la norme pas implantées
- ▶ Performances, passage à l'échelle

# 12 règles de Codd

Tentative de définition d'un SGBD soit relationnel

## Règle 1 - Unicité

Chaque information est représentée d'une seule manière.

## Règle 2 - Accès non-ambigu

(clé primaire, base/table/colonne) donne une valeur

## Règle 3 - Traitement des valeurs manquantes

Valeur NULL distincte de toutes les autres valeurs (y compris de NULL)

## Règle 7 - Insertion, mise à jour, et effacement de haut niveau

Traitement par lots

## 12 règles de Codd - Indépendances

### Règle 8 - Indépendance physique

On peut changer la façon de stocker les données

### Règle 9 - Indépendance logique

On peut changer la structure de la base

### Règle 10 - Indépendance d'intégrité

On peut exprimer des contraintes sur les données indépendamment de l'application

### Règle 11 - Indépendance de distribution

Distribution sur plusieurs machines transparente

# Groupes d'opérations

## Exemple

- ▶ Étape 1: débiter un compte bancaire
- ▶ Étape 2: créditer un autre compte

Que se passe-t-il si le système plante entre les étapes 1 et 2 ?

## Transaction

- ▶ Marquer une suite d'opération comme *atomique*
- ▶ Pouvoir revenir à l'étape antérieur en cas de problème

# ACID



# ACID

## Atomicité

- ▶ Une transaction se fait complètement ou pas du tout

## Cohérence

- ▶ Le système passe toujours d'un état valide à un autre

## Isolation

- ▶ Indépendance entre les transactions

## Durabilité

- ▶ Une transaction effectuée l'est de façon durable

# Algèbre relationnelle

## Motivation

- ▶ Formaliser les opérations sur les bases de données
- ▶ Basé sur la théorie des ensembles

## Optimisation des requêtes

- ▶ Transformer des requêtes en requêtes équivalentes
- ▶ Choisir la plus efficace

L'optimisation des requêtes est un point critique pour la performance d'un SGBD.

# Opérations ensemblistes

## Union

$$R \cup S = \{t : t \in R \text{ ou } t \in S\}$$

## Intersection

$$R \cap S = \{t : t \in R \text{ et } t \in S\}$$

## Différence

$$R - S = \{t : t \in R \text{ et } t \notin S\}$$

## Produit cartésien

$$R \times S = \{(r, s) : r \in R \text{ et } s \in S\}$$

# Opérations relationnelles

## Sélection

$$\sigma_F(R) = \{r \in R \mid F(r)\}$$

## Projection

$$\pi_A(R)$$

Éléments de  $R$  en ne considérant que les attributs listés dans  $A$ .

## Renommage

$$\rho_{a/b}(R)$$

Attribut  $b$  rebaptisé en  $a$ .

# Tables

## Tableaux à deux dimensions

- ▶ Attributs  $\times$  éléments
- ▶ Une ligne n'est pas forcément présente une et une seule fois

## Clé primaire

- ▶ Garantir l'unicité
- ▶ Un attribut ou plusieurs attributs naturellement uniques
- ▶ Un attribut rajouté artificiellement dans ce but

# Jointures

## Références entre tables

- ▶ Produit cartésien
- ▶ Sélection

## Clé étrangère

- ▶ Attribut identifiant un élément d'une autre table
- ▶ Clé primaire de l'autre table

# Sélection et projection

```
SELECT * FROM table WHERE ...;
```

## Projection

```
SELECT a,d,e FROM table;
```

## Filtre

```
SELECT * FROM table WHERE table.name = 'machin';
```

## Produit cartésien

```
SELECT t1.a, t2.a, t2.b FROM table1 t1, table2 t2;
```

# Jointure

## Deux opérations

- ▶ Produit cartésien
- ▶ Sélection

---

### Personnes

---

id	(clé primaire)
nom	
prenom	
age	
adresse_id	(clé étrangère)

---

---

### Adresses

---

id	(clé primaire)
rue	
ville	

---

## SQL

```
SELECT * FROM Personnes, Adresses WHERE  
Personnes.adresse_id = Adresses.id
```



# Opérations non-ensemblistes

## Tri

```
SELECT * FROM Personnes ORDER BY nom;
```

## Aggrégation

- ▶ Comptage: `SELECT count(*) FROM table;`
- ▶ Somme `SELECT sum(age) FROM personnes;`
- ▶ Autre: max, min, avg, fonctions statistiques

## Suppression de réponses identiques

- ▶ `SELECT DISTINCT`

# Contraintes

## Propriétés sur les colonnes

- ▶ Strictement positif
- ▶ Pas NULL
- ▶ etc

## Intégrité référentielle

- ▶ Garantir que les clés étrangères correspondent à des éléments existants

# Description du schéma

## Bibliothèque universitaire

- ▶ Des étudiants
- ▶ Des livres
- ▶ Des emprunts

## Étudiants

- ▶ Empruntent des livres
- ▶ 0 ou plus
- ▶ Nom, prénom, numéro unique d'étudiant

## Livres

- ▶ Sont empruntés
- ▶ 0 ou plus
- ▶ Titre, auteur, et numéro ISBN unique

# Entité-Association

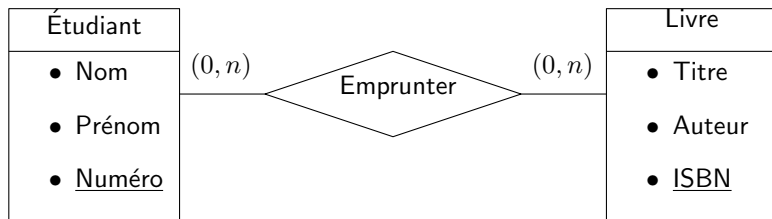


Figure 1: image

# Tables

## Étudiant

- ▶ Clé primaire
- ▶ Attributs

## Livre

- ▶ Clé primaire
- ▶ Attributs

## Lien Étudiant-Livre

- ▶ Clé étrangère vers Étudiant
- ▶ Clé étrangère vers Livre

## Exemple de requête

Liste des livres empruntés par un étudiant

```
SELECT etudiant.nom, livre.titre  
FROM etudiant, livre, emprunter_etudiant_livre emprunter  
WHERE etudiant.numero = emprunter.id_etudiant  
AND livre.isbn = emprunter.id_livre  
AND etudiant.numero = 123513223;
```

# Types numériques

<http://docs.postgresql.fr/9.3/datatype.html#datatype-numeric>

---

boolean	1 octet	état vrai ou faux
smallint	2 octets	entier de faible étendue
integer	4 octets	entier habituel
bigint	8 octets	grand entier
decimal	variable	valeur exacte
numeric	variable	valeur exacte
real	4 octets	valeur inexacte
double precision	8 octets	valeur inexacte

---

# Types caractères

<http://docs.postgresql.fr/9.3/datatype-character.html>

---

<code>varchar(n)</code>	Longueur variable avec limite
<code>char(n)</code>	longueur fixe, complété par des espaces
<code>text</code>	longueur variable illimitée

---



## Types date/heure

<http://docs.postgresql.fr/9.3/datatype-datetime.html>

---

timestamp	8 octets	date et heure (sans fuseau horaire)
timestamp with time zone	8 octets	date et heure, avec fuseau horaire
date	4 octets	date seule (pas d'heure)
time	8 octets	heure seule (pas de date)
time with time zone	12 octets	heure seule, avec fuseau horaire
interval	16 octets	intervalles de temps

---

- ▶ 04:05:06, 04:05:06-8
- ▶ 1999-01-08 04:05:06 -8:00, January 8 04:05:06 1999 PST

## Autres types

<http://docs.postgresql.fr/9.3/datatype.html>

- ▶ Données binaires
- ▶ Types géométriques
- ▶ Types personnalisés
- ▶ Intervalles
- ▶ Énumérations

## Création de tables

```
CREATE TABLE etudiant (  
    numero char(8) PRIMARY KEY,  
    nom varchar(40),  
    prenom varchar(40),  
);
```

```
CREATE TABLE emprunter (  
    id_etudiant char(8),  
    id_livre char(13),  
    CONSTRAINT pk_emprunter  
        PRIMARY KEY (id_etudiant, id_livre),  
    CONSTRAINT fk_emprunter_etudiant  
        FOREIGN KEY (id_etudiant) REFERENCES Etudiant(numero),  
    CONSTRAINT fk_emprunter_livre  
        FOREIGN KEY (id_livre) REFERENCES Livre(isbn),  
);
```

# Autres opérations

## Suppression

- ▶ DROP

## Modification

- ▶ ALTER