

Implémentation de systèmes de gestion de bases de données

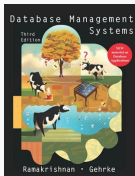
- introduction aux bases de données relationnelles
- transactions et propriétés ACID
- gestion de la concurrence des transactions
- reprise après une panne système

- enseignant responsable : Maxime Buron
- page Moodle (Implémentation des SGBDs) :
<https://ent.uca.fr/moodle/course/view.php?id=6527>
- 6 cours magistraux avec des TDs
- 5 séances de TP (essayer de se connecter à sa VM!)
- 1 examen écrit + 1/2 TPs notés

- **Database Systems : The Complete Book** Héctor García Molina, Jeffrey Ullman et Jennifer Widom



- **Database Management Systems** Raghu Ramakrishnan, Johannes Gehrke



Introduction aux base de données relationnelles

Histoire des bases de données



Tout était fait à la mano

Des employés récupéraient
les données physiquement

Figure 1: Base de données des empreintes du FBI en 1944 (23m de fiches, 110K requêtes par mois)

à la main

Histoire des bases de données

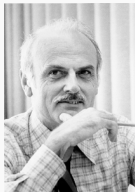
- 1970 : Edgar F. Codd introduit la notion de modèle relationnel chez **IBM**



Formalise l'idée de BDD relationnelle



Détient JAVA



Boîte d'informatique
émergente

- 1979 : Oracle sort le premier système de gestion de base de données (**SGDB**)



- 1980 à 2000 : De nombreux SGDBs sont créés et SQL devient un langage standard



Proche de la langue anglaise

Les systèmes de base de données aujourd'hui

Omniprésents

- navigateur
- site d'achat en ligne
- banque
- moteur de recherche (Google Search 50M de page, 3M de requêtes par jours)

(milliards)

Applications,
connections,
requêtes, ...

Bonnes propriétés

- fiabilité, ils sont très stables
- performance, ils permettent de gérer de larges bases de données
- durabilité, ils assurent la qualité des données

Stable, sécurisé (normalement hum), complexe



Permet de s'assurer que les données insérées ont un sens

Les bases de données relationnelles

Une base de données relationnelles comporte :

Il n'y a pas que les bases de données relationnelles (on a aussi graphiques, ...)

- un **schéma** : la structure des tables/reliations (le nom et le type des colonnes/attributs)
- un **contenu** : les lignes stockées dans chaque table
- des **contraintes** d'intégrité assurent la cohérence des données

Exemple

"La donnée est bien une date"

Une structure pré-définie

- schéma

Somme(client, total)

Contenu(client, produit)

Produit(produit, nom, prix)

- contenu

table		Panier		Produit		
Somme		client	produit	produit	nom	prix
client	total	0	0	0	pommes	5
0	10	0	1	1	poires	5
1	2	1	2	2	carottes	2

Ligne

Contraintes d'intégrité

Les contraintes d'intégrité permettent de définir des limitations sur le contenu de la base de données. Il y a deux types de contraintes :

- les **contraintes d'intégrité**, qui sont stockés dans le SGBD et peuvent être vérifiées automatiquement à chaque modification du contenu
- les **contraintes implicites**, qui sont dans la tête de l'administrateur de la base de données uniquement

Ex : 50 caractères max, un nombre, etc...

Peut être un calcul complexe

Exemple de contraintes

- une explicite : `Produit.prix >= 0`
- une implicite : le total d'un panier est la somme des prix des objets qu'il contient

Le serveur vérifie cette contrainte, alors que l'intégrité c'est la SGBD

Contraintes d'intégrité : les clés

Soit K un sous-ensemble des colonnes d'une table T ,

- K est une **super-clé** de T , si le nombre de ligne ayant les mêmes valeurs sur les colonnes de K est au plus 1 \longrightarrow On assure que la ligne est unique sur un/des colonne(s)
- K est une **clé candidate** de T , c'est une super-clé qui contient un nombre minimale de colonnes
- une **clé primaire** de T est une clé candidate choisie de T pour identifier chaque ligne de T

On peut identifier un étudiant grâce à son numéro étudiant

Produit			
	produit	nom	prix
0		pommes	5
1		poires	5
2		carottes	2

Un exemple de clé candidate

- Quels sont les **super-clés**, **clés candidates**?
- Quel est le choix naturel pour la clé primaire?

Un produit a une clé unique qui est associée à un nom et prix

Exemple de clé primaire

Panier		Produit		
client	produit	<u>produit</u>	nom	prix
0	0	0	pommes	5
0	1	1	poires	5
1	2	2	carottes	2

Panier utilise la clé primaire de Produit

1. Quelle clé primaire peut on choisir pour la table Panier ?

Exemple de clé primaire

Panier		Produit		
client	produit	<u>produit</u>	nom	prix
0	0	0	pommes	5
0	1	1	poires	5
1	2	2	carottes	2

1. Quelle clé primaire peut on choisir pour la table Panier ?
{client, produit}
2. Avec cette clé primaire, un client peut-il avoir plusieurs fois le même produit dans son panier ?
3. Comment résoudre ce problème ?

Exemple de clé primaire

Panier		Produit		
client	produit	<u>produit</u>	nom	prix
0	0	0	pommes	5
0	1	1	poires	5
1	2	2	carottes	2

Panier utilise la clé qui identifie un Produit

1. Quelle clé primaire peut on choisir pour la table Panier ?
{client, produit}
2. Avec cette clé primaire, un client peut-il avoir plusieurs fois le même produit dans son panier ?

On rajoute une colonne quantité pour éviter les duplications de ligne (ne pas dire : Pomme, Pomme, Pomme, ...)


3. Comment résoudre ce problème ?

Panier		
<u>client</u>	<u>produit</u>	quantité
0	0	2
0	1	1
1	2	4

Contraintes d'intégrité : les clés étrangères

Une **clé étrangère** est un ensemble de colonnes K d'une table faisant référence à la clé primaire d'une autre table. La clé étrangère est satisfaite si les tuples de valeurs présents sur les colonnes de K sont aussi présents sur les colonnes de la clé primaire.

Un produit a un nombre qui l'identifie, et panier l'utilise (clé étrangère)



Panier		Produit		
client	produit	produit	nom	prix
0	0	0	pommes	5
0	1	1	poires	5
1	2	2	carottes	2

Il y a une clé étrangère depuis la colonne produit de la table Panier vers la clé primaire de la table Produit. Cette contrainte assure que chaque produit présent dans un panier a un nom et un prix.

Cohérence d'une base de données

Une base de données est **cohérente** si son contenu vérifie :

- les **contraintes d'intégrité**
- les **contraintes implicites**

Si toutes les contraintes sont respectées, la bdd est cohérente (la SGBD s'occupe des contraintes d'intégrité, le serveur le reste)

On souhaite que tous les modifications apporter à une base de données conservent la cohérence de la base.

Exemple

Somme		Panier		Produit		
<u>client</u>	total	<u>client</u>	produit	<u>produit</u>	nom	prix
0	10	0	0	0	pommes	5
1	2	0	1	1	poires	5
		1	2	2	carottes	2

Contrainte implicite

L'unicité c'est le contrainte d'intégrité

Les opérations sur les bases de données

SQL est le langage standard pour accéder et modifier une base de données relationnelle :

- requêtes d'interrogation

```
SELECT client , SUM(prix*quantite)
FROM Panier NATURAL JOIN Contenu
GROUP BY client ;
```

“Récupère le client et la somme de ses dépenses dans Panier (on colle les tables Panier et Contenu en collant les id de clients)”

- requêtes de mise à jour ou d'insertion

```
UPDATE Produit SET prix = 1.1*prix WHERE nom = 'pommes' ;
```

- opérations de modification du schéma ou des contraintes

On augmente de 10% le prix des pommes

La prochaine fois ...

- les **transactions** ou de petits programmes en SQL, qui définissent des opérations atomiques de gestion de base de données
- les propriétés ACID des transactions