Chapitre 9 : Bases de données

Table des matières

1 Concepts élémentaires

1.1 Introduction aux bases de données

1.1.1 Introduction

De nombreuses applications informatiques manipulent de grandes quantités de données qui doivent être organisées et stockées de sorte qu'il est possible de les traiter efficacement pour en ajouter, en retirer, ou en extraire de l'information. Ce traitement doit pouvoir se faire de manière concurrente tout en préservant l'intégrité des données.

1.1.2 Systèmes de gestion de bases de données (SGBD) et paradigme logique

Préserver l'intégrité des données est une tâche complexe dans un contexte d'accès concurrents, donc le traitement des données sera confié à un outil appelé SGBD dont le rôle est de recevoir les requêtes des utilisateurs (modification des données ou extraction d'information à partir de ces données) et de les traduire en des opérations effectuées sur la base de données. C'est au SGBD de garantir la cohérence des données au fur et à mesure des opérations réalisées. Les requêtes prennent en général la forme d'une description du résultat attendu sans indication sur la manière de calculer. C'est le SGBD qui implémente la recherche du résultat.

C'est le principe du paradigme logique : un programme est une description des propriétés que doit satisfaire le résultat. Un résultat est un jeu de paramètres qui satisfait les propriétés. Il n'y a aucune indication sur la manière de calculer les résultats.

Pour que cela fonctionne, la plupart des SGBD s'appuient sur un modèle introduit dans les années 1970, appelé modèle relationnel.

1.1.3 Le modèle relationnel

• Le modèle relationnel est une modèle mathématique basé sur la théorie des ensembles et la logique des prédicats, et qui présente les bases de données comme des objets qui définissent des relations entre les blocs d'information. C'est un modèle abstrait qui s'exprime indépendamment des implémentations possibles et qui couvre donc de nombreux SGBD.

• Dans ce modèle, une base de données est vue comme un ensemble de relations. Ces relations sont aussi appelées *tables* car on peut les représenter par des tableaux à double entrée dont les colonnes correspondent à un type d'information particulier. Ces colonnes sont appelées les *attributs* de la relation.

Exemple : dans le système d'information d'une bibliothèque, on peut avoir une table Document dont les attributs sont le titre, l'auteur, le genre, la date de parution, le nombre de pages, ...

• Chaque attribut est associé à un domaine qui correspond à l'ensemble des valeurs possibles par l'attribut. Le domaine permet de choisir un type pour implémenter concrè-

tement la base de données.

Étant donné une relation R dont les attributs sont $A_1, \ldots, A_n$ associés aux domaines $D_1, \ldots, D_n$, on appelle *schéma relationnel* de R l'association des attributs et des domaines notée

$$R(A_1 : D_1, \ldots, A_n : D_n)$$

Exemple : pour la table Document :

- Le titre et l'auteur sont des données textuelles ;
- Le genre est tirée d'une énumération finie (roman, poésie, théâtre, ...) ;
- La date de parution est une date ;
- Le nombre de pages est un entier.

D'où le schéma relationnel :
Document(titre : texte, auteur : texte, genre : enum(roman, ...), date de parution : date, nombre de pages : entier)

Le domaine "texte" peut être par exemple associé au type `string` des chaînes de caractères et le domaine entier au type `int`.

• Les lignes d'une relation de schéma $R(A_1 : D_1, \ldots, A_n : D_n)$ correspondent aux éléments de la relation R, qui est un sous ensemble de $\prod_{k=1}^{n} D_k$. On appelle donc ces éléments des tuples ou des enregistrements.

Exemple pour la table Document :

Titre	Auteur	Genre	Date de parution	Nombre de pages
La cousine Bette	Honoré de Balzac	Roman	1846	240
De la guerre	Carl von Clausewitz	Traité	1832	240
Cyrano de Bergerac	Edmond Rostand	Théâtre	1857	280

Remarque : certains enregistrements peuvent coïncider pour certains attributs et on veut une manière efficace de les distinguer.

• On appelle *clé candidate* un ensemble minimal (pour l'inclusion) d'attributs permettant de caractériser de manière unique chaque enregistrement, *i.e* tel qu'il n'existe pas deux enregistrements qui coïncident sur tous les attributs de la clé.

Exemple : pour la table Document, {titre, auteur} devrait convenir.

Il peut y avoir plusieurs clés candidates et on doit en choisir une, appelée *clé primaire*. On souligne dans le schéma relationnel les attributs de la clé primaire pour les repérer efficacement.
Remarque : on choisit souvent d'ajouter un attribut entier pour numéroter les enregistrements, que l'on choisit comme clé primaire.

• Dans une relation R, on appelle *clé étrangère* un ensemble d'attributs qui constitue une clé candidate (souvent primaire) d'une autre relation.

Exemple : dans une table Emprunts décrivant les emprunts de la bibliothèque, on intègre la clé de la table Document pour identifier les documents empruntés.

1.1.4 Algèbre relationnelle

L'algèbre relationnelle est une théorie mathématique qui décrit des opérations que l'on peut réaliser sur une base de données du point de vue du modèle relationnel. Les propriétés qui découlent de cette théorie définissent un fondement rigoureux aux implémentations de SGBD en justifiant les optimisations des requêtes des utilisateurs. Cette théorie des H.P, mais nous l'étudierons *via* le langage de requêtes SQL (Structured Query Language).

1.2 Le langage SQL : requêtes élémentaires

1.2.1 Gestion du contexte

- Choix de la base de données : `USE <nom_base>;`
- Obtention du schéma relationnel d'une table : `DESCRIBE <nom_table>;`

Exemple :

```
USE bibliotheque;
DESCRIBE Document;
```

Les opérations de création / suppression / modification de bases de données / de tables sont H.P.

1.2.2 Opération de projection

Pour visualiser le contenu d'une table, on utilise la requête `SELECT * FROM <nom_table>;`

C'est une cas particulier de l'opération de projection qui permet de construire une table ne contenant que les valeurs des tuples que pour certains attributs.

Attention, la table construite par une requête est éphémère.

Syntaxe : `SELECT <attribut`$_1$`>, ..., <attribut`$_n$`> FROM <nom_table>;`

Exemple :

```
SELECT titre, date_de_parution FROM Document;
```

Remarque : cette requête peut créer une table contenant des doublons qui sont conservés par défaut. On utilise le mot-clé `DISTINCT` pour éliminer les doublons.

Exemple :

```
SELECT DISTINCT titre, date_de_parution FROM Document;
```

Les enregistrements du résultat d'une projection sont a *priori* rangés dans le même ordre que dans la table initiale. On peut choisir de les réordonner en utilisant le mot-clé `ORDER BY` suivi d'une liste d'attributs.

L'ordre associé à cette liste est l'ordre croissant lexicographique.

Exemple :

```
SELECT * FROM Document
ORDER BY auteur, date_de_parution;
```

Pour utiliser l'ordre décroissant selon l'un des attributs, on utilise le mot-clé `DESC` après le nom de l'attribut.

Exemple :

```
SELECT * FROM Document
ORDER BY date_de_parution DESC;
```

Attention, `DESC` ne porte que sur un seul attribut (le répéter si besoin).

Il est aussi possible de renommer les attributs du résultat d'une projection, ce qui peut être nécessaire dans des opérations plus complexes faisant intervenir des sous-requêtes ou plusieurs tables. On utilise pour cela le mot-clé `AS`.

Exemple :

```
SELECT titre, auteur, date_de_parution AS date
FROM Document;
```

On peut combiner tout cela :

```
SELECT titre, date_de_parution AS date
FROM Document
ORDER BY date DESC, titre;
```