Chapitre 0 :

Retour sur les bases de données relationnelles

0. 1. Introduction

Un modèle de données définit un mode de représentation de l'information selon trois composantes :

1. Des structures de données.

2. Des contraintes qui permettent de spécifier les règles que doit respecter une base de données.

3. Des opérations pour manipuler les données, en interrogation et en mise à jour.

Les deux premières composantes relèvent du Langage de Définition de Données (DDL) dans un SGBD. Le DDL est utilisé pour décrire le schéma d'une base de données. La troisième composante (opérations) est la base du Langage de Manipulation de Données (DML) dont le représentant le plus célèbre est SQL.

Dans le contexte des bases de données, la principale qualité d'un modèle de données est d'être indépendant de la représentation physique. Cette indépendance permet de séparer totalement les tâches respectives des administrateurs de la base, chargés de l'optimisation de ses performances, et des développeurs d'application ou utilisateurs finaux qui n'ont pas à se soucier de la manière dont le système satisfait leurs demandes. Le modèle relationnel, venant après les modèles hiérarchique et réseau, offre une totale indépendance entre les représentations logique et physique.

0.2. Le schéma relationnel

Un des grands avantages du modèle relationnel est sa très grande simplicité. Il n'existe en effet qu'une seule structure, la **relation**. Une relation peut simplement être représentée sous forme de table, comme sur la figure suivante. Une relation a donc un nom (Film) et se compose d'un ensemble de colonnes désignées par un nom d'attribut.

Dans chaque colonne on trouve des valeurs d'un certain domaine (chaînes de caractères, nombres). Enfin on constate que chaque ligne (ou tuple) correspond à une entité (ici des films).

titre	année	genre
Alien	1979	Science-Fiction
Vertigo	1958	Suspence
Volte-face	1997	Thriller
Pulp Fiction	1995	Policier

Fig. 1. 1. Exemple d'une relation

Un schéma de relation est simplement un nom suivi de la liste des attributs, chaque attribut étant associé à son domaine. La syntaxe est donc : R(A1:D1,A2:D2, An:Dn)

où les Ai sont les noms d'attributs et les Di les domaines. L'arité d'une relation est le nombre de ses attributs. On peut trouver dans un schéma de relation plusieurs fois le même domaine, mais une seule fois un nom d'attribut. Le domaine peut être omis en phase de définition. Le schéma de la relation de la figure précédente est donc : *Film* (titre: string, année: number, genre : string)

L'instance d'une relation R , ou simplement relation se définit mathématiquement comme un sous ensemble fini du produit cartésien des domaines des attributs de R . Rappelons que le produit cartésien $D1 \times D2 \times \ldots Dn \times$ entre des domaines d1, D2, …Dn est l'ensemble de tous les tuples (v1, …vn) où $vi \in Di$. Un des fondements du modèle relationnel est la théorie des ensembles et la notion de relation dont le modèle correspond strictement au concept mathématique dans cette théorie. Une relation se représente sous forme de table, et on emploie le plus souvent ces deux termes comme des synonymes.

La clé d'une relation est le plus petit sous-ensemble des attributs qui permet d'identifier chaque ligne de manière unique. Comme on a vu que deux lignes sont toujours différentes, l'ensemble de tous les attributs est lui-même une clé mais on peut pratiquement toujours trouver un sous-ensemble qui satisfait la condition. Pour distinguer la clé, nous mettrons le (ou les) attribut(s) en gras. Film (**titre, année**, genre) : Le choix de la clé est très important pour la qualité du schéma.

Un tuple est une liste de n valeurs (v1, v2,..., vn) où chaque valeur vi est la valeur d'un attribut Ai de domaine Di. Exemple: ('Cyrano', 1992, 'Rappeneau') ; Un tuple est donc simplement une ligne dans la représentation d'une relation sous forme de table. En théorie, on connaît les valeurs de tous les attributs du tuple.

0.2.1. Base de données relationnelle

Une (instance de) base de données est un ensemble fini (d'instances) de relations. Le schéma de la base est l'ensemble des schémas des relations de cette base. La création d'un schéma de base de données est simple une fois que l'on a déterminé toutes les relations qui constituent la base. En revanche le choix de ces relations est un problème difficile car il détermine en grande partie les caractéristiques et qualités de la base: performances, exactitude, exhaustivité, disponibilité des informations, etc. Un des aspects importants de la théorie des bases de données relationnelles consiste précisément à définir ce qu'est un bon schéma et propose des outils formels pour y parvenir.

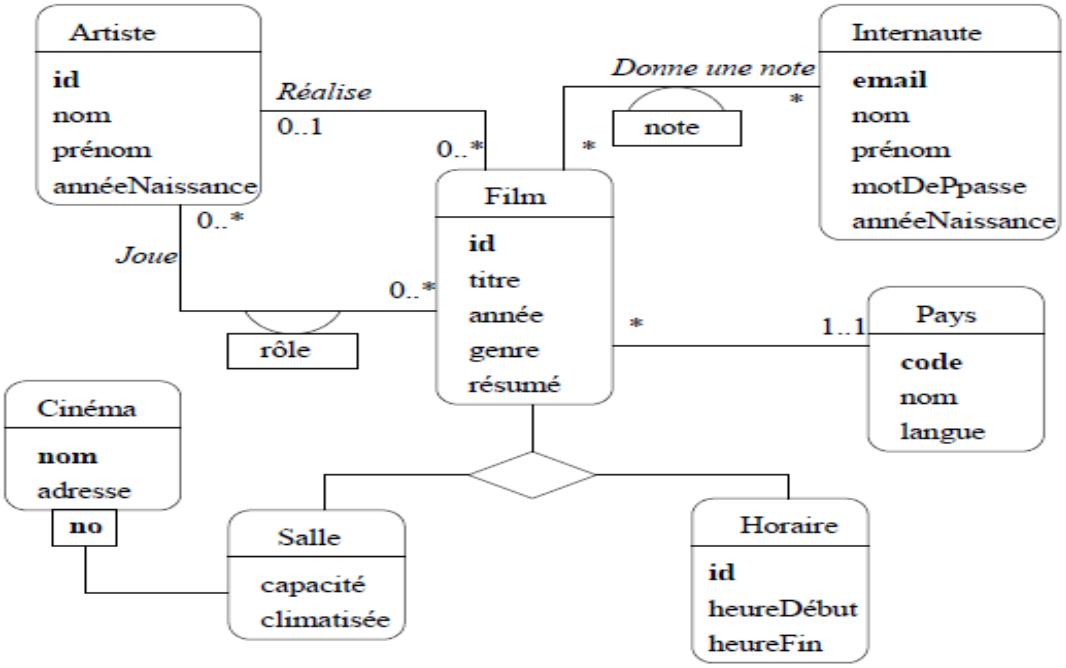

Fig.0.2. Le schéma de la base de données Films

0.2.2. Les règles d'intégrité

Les règles d'intégrité sont les assertions qui doivent être vérifiées par les données contenues dans une base. Le modèle relationnel impose les contraintes structurelles suivantes :

- **Intégrité de Domaine**
- **Intégrité de clé**
- **Intégrité référentielle**

La gestion automatique des contraintes d'intégrité est l'un des outils les plus importants d'une base de données. Elle justifie à elle seule l'usage d'un SGBD.

Exemple d'intégrité référentielle:

CLIENT (no_client, nom, adresse)
ACHAT (no_produit, no_client, date, qte)

Clé étrangère no_client dans ACHAT

- Insertion tuple no_client = X dans ACHAT
  - vérification si X existe dans CLIENT
- Suppression tuple no_client = X dans CLIENT
  - soit interdire si X existe dans ACHAT
  - soit supprimer en cascade tuple X dans ACHAT
  - soit modifier en cascade X = NULL dans ACHAT
- Modification tuple no_client = X en X' dans CLIENT
  - soit interdire si X existe dans ACHAT
  - soit modifier en cascade X en X' dans ACHAT

0.3. Le langage SQL

SQL (Structured Query Language) est le langage de manipulation des données relationnelles le plus utilisé aujourd'hui. Il est devenu un standard de fait pour les relationnels. Il possède des caractéristiques proches de l'algèbre relationnelle (jointure par emboîtement) et d'autres proches du calcul des tuples (variables sur les relations).

SQL est un langage de définition de données: SQL est un langage de définition de données (LDD), c'est-à-dire qu'il permet de créer des tables dans une base de données relationnelle, ainsi que d'en modifier ou en supprimer.

SQL est un langage de manipulation de données : SQL est un langage de manipulation de données (LMD), cela signifie qu'il permet de sélectionner, insérer, modifier ou supprimer des données dans une table d'une base de données relationnelle.

SQL est un langage de protections d'accès : Il est possible avec SQL de définir des permissions au niveau des utilisateurs d'une base de données. On parle de DCL (Data Control Language).

0.3.1. Définition des données

Les exemples dans cette partie du cours s'appuient sur la base de données relative aux fournisseurs (F), produits (P), usines (U) et livraisons (PUF), décrite par le schéma suivant:

F (NF, nomF, statut, ville)
P (NP, nomP, poids, couleur)
U (NU, nomU, ville)
PUF (NP, NU, NF, qt)

Pour créer une table, on utilise le couple de mots-clés CREATE TABLE.
La syntaxe est la suivante :

```
CREATE TABLE NomTable (
NomColonne1 TypeDonnée1,
NomColonne2 TypeDonnée2, ... );
CREATE TABLE U (
NU integer,
NomU char(30)
Ville char(30)
);
```

Les informations pouvant être mises à jour par les instructions suivantes :

- Ajouter des n-uplets : INSERT INTO
  - Insérer une seule ligne :

```
INSERT INTO NomTable(colonne1,colonne2,colonne3,...)
VALUES (Valeur1,Valeur2,Valeur3,...)
```

  - Insérer plusieurs lignes :

```
INSERT INTO NomTable(colonne1,colonne2,...)
SELECT colonne1,colonne2,... FROM NomTable2
WHERE qualification
```

(les valeurs inconnues prennent la valeur NULL)

- Modifier des n-uplets existants : UPDATE...SET...WHERE...
  ```
  UPDATE NomTable
  SET Colonne = ValeurOuExpression
  WHERE qualification
  ```

- Supprimer des n-uplets : DELETE FROM...WHERE...
  ```
  DELETE FROM NomTable
  WHERE qualification
  ```

0.3.2. Interrogation d'une base de données

La commande SELECT permet d'interroger une BD. La syntaxe est la suivante :

```
SELECT [ALL|DISTINCT] NomColonne1,... | *
FROM NomTable1,...
WHERE Condition
```

Exemples :

1. Nom et poids des produits rouges.

```
SELECT nomP, poids
FROM P
WHERE couleur = "rouge"
```

2. Tous les renseignements sur tous les fournisseurs.

```
SELECT NF, nomF, statut, ville
FROM F
```

ou

```
SELECT * /* l'étoile signifie: tous les attributs*/
FROM F
```

3. Liste des couleurs qui existent (sans doublons)

```
SELECT DISTINCT couleur
FROM P
```

0.3.3. Concurrence d'accès

Plusieurs utilisateurs doivent pouvoir accéder en même temps aux mêmes données. Le SGBD doit savoir :

– Gérer les conflits si les deux font des mises-à-jour.

– Offrir un mécanisme de retour en arrière si on décide d'annuler des modifications en cours.

– Donner une image cohérente des données si l'un fait des requêtes et l'autre des mises-à-jour.

Le but étant d'éviter les blocages, tout en empêchant des modifications anarchiques.

Une transaction est une unité logique de traitement qui est soit complètement exécutée, soit complètement abandonnée. Une transaction fait passer la BD d'un état cohérent à un autre état cohérent. Une transaction est terminée soit par **COMMIT,** soit par **ROLLBACK.**

Exemple du Banquier : Le transfert d'une somme S d'un compte C1 vers un compte C2

(1) début-transaction
(2) lire C1
(3) C1 := C1 - S
(4) écrire C1
(5) lire C2
(6) C2 := C2 + S
(7) écrire C2
(8) fin-transaction

Cette transaction est constituée d'un ensemble d'actions élémentaires, mais elle doit être traitée comme une seule opération. Autrement dit le gestionnaire des transactions doit assurer que **toutes** les actions de la transaction sont exécutées, ou bien qu'**aucune** ne l'est.

Un système de gestion transactionnel doit garantir les propriétés suivantes (résumées par le vocable **ACID**) :

Atomicité : Une transaction doit effectuer toutes ses mises à jour ou ne rien faire du tout

Cohérence : La transaction doit faire passer la base de données d'un état cohérent à un autre

Isolation : Les résultats d'une transaction ne doivent être visibles aux autres transactions qu'une fois la transaction validée

Durabilité : Dès qu'une transaction valide ses modifications, le système doit garantir que ces modifications seront conservées en cas de panne.

De nombreuses solutions ont été proposées pour traiter le problème des accès concurrents. Un exemple important est le protocole appelé verrouillage à deux phases qui est un des plus utilisés.