Cours de Système d'Information

Chapitre 7 - MLD et MPD

Tarek Boutefara

Version 0.3, 2020-12-20

Table of Contents

1. Introduction

Plusieurs modèles logiques et physiques ont été proposés pour les données. Dans ce chapitre, nous allons introduire les choix les plus courants des entreprises :

- Le modèle relationnel comme modèle logique,
- Les bases de données relationnelles comme niveau physique.

2. Niveau Logique : Le modèle relationnel

Le modèle relationnel se base sur un ensemble de concepts :

2.1. Relation :

Un sous-ensemble du produit cartésien d'une liste de domaines. Une relation n-aire sur D1, D2, …, Dn est un sous-ensemble du produit cartésien D1 x D2 x … X Dn.

2.2. Domaine d'attribut :

C'est un ensemble de valeurs possibles pour un attribut. Cela se ramène, au niveau physique, à un type de données. A ce type de données, une valeur particulière est ajoutée : la valeur "NULL". Cette dernière correspond à la valeur "inconnu" (différente de 0 et différente de la chaîne vide).

2.3. Attribut :

Un attribut (nommé généralement Ai) d'une relation est un identificateur (nom) associé à un domaine (Di) de la relation. Ce nom est généralement une propriété au niveau conceptuel.

2.4. Schéma relationnel :

Un schéma de relation définit le nom de la relation et l'ensemble de ses attributs. Chaque attribut appartient à un domaine spécifique. Pour chaque relation, un sous-ensemble de ses attributs constitue la clé primaire; il est souligné dans le schéma.

Dans une base de données relationnelle, les données sont perçues comme des Tables. Le schéma relationnel (schéma de la base de données) est obtenu à partir du modèle Entité/Association (niveau conceptuel) en appliquant les règles suivantes :

- Toute entité devient une relation,
- Toute association de type (n:m) devient une table,
- Les associations de type (1:n) se traduisent par des clés étrangères (la clé primaire du père migre vers le fils),
- Chaque relation qui ne contient qu'un seul attribut sera ignoré.

3. Niveau Physique : Langage SQL

Le niveau physique est influencé principalement par les caractéristiques de la machine (de l'ordinateur). Parmi ces limites de la machine par rapport aux abstractions faites au niveau sémantique (conceptuel) et logique, nous citons :

- L'implémentation des domaines,
- Le temps d'exécution :
  - Accès disque dure,
  - Complexité des algorithmes,
- L'espace mémoire :
  - La taille de la RAM vs. La taille de la base des données.

Le langage SQL repose sur les notions d'algèbre relationnel et ajoute la dimension physique liée à la machine et ses limites. SQL est un langage déclaratif (non procédural) : on précise le "Quoi" sans le "Comment". Ce qui permet d'obtenir cette possibilité est le fait que nous n'avons que quatre opérations de base sur les données (CRUD) :

- Créer (Créate),
- Lire (Read),
- Update (Mettre à jour),
- Delete (Supprimer).

SQL définit plusieurs requêtes :

- **LMD :** Langage de manipulation de données :
  - Select,
  - Insert,
  - Update,
  - Delete.
- **LDD :** Langage de définition de données :
  - Create,
  - Alter,
  - Drop.

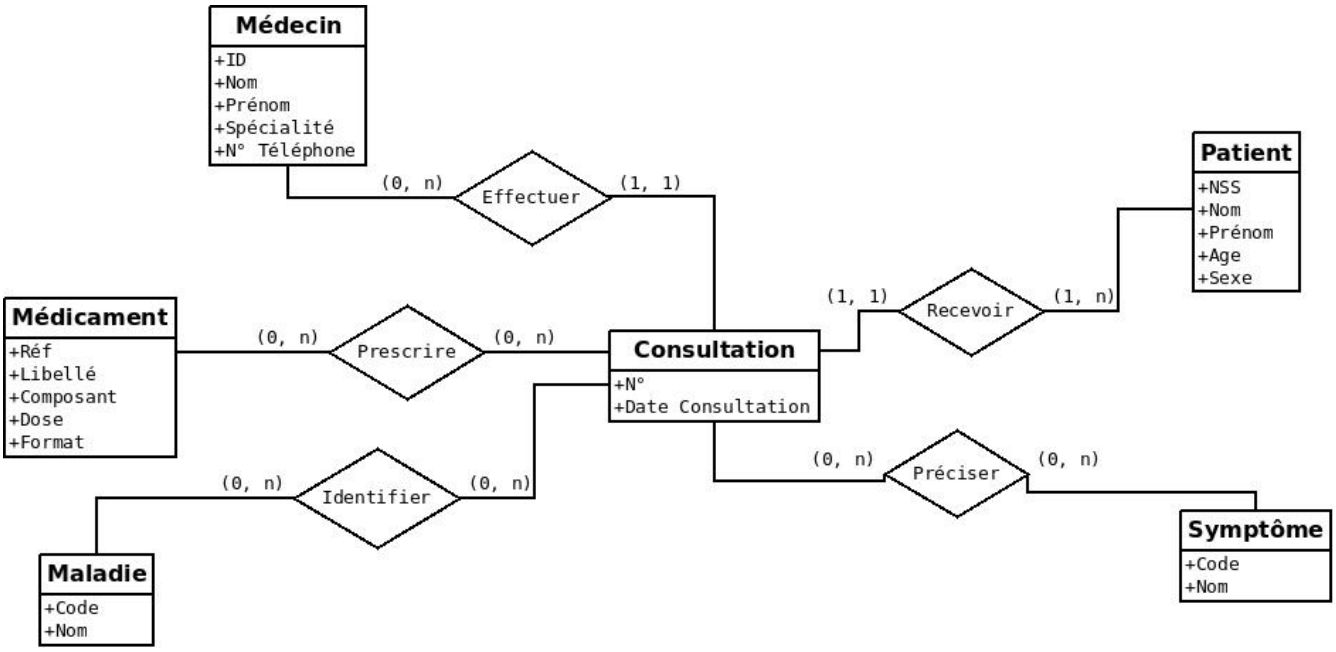
4. Exemple Complet

Soit la description suivante :

"Dans une clinique, plusieurs médecins exercent leur activité. Chaque médecin reçoit un malade pour chaque consultation. Pendant l'examen clinique, il remplit un rapport en précisant tous les symptômes (température, mal de tête, fréquence cardiaque, ect.). Il conclut son rapport par préciser

une ou plusieurs maladies ou pathologies. En se basant sur ces résultats, le médecin prescrit au patient plusieurs médicaments."

4.1. Le MCD



4.2. Le Modèle Relationnel

En appliquant les quatre règles de transformation, nous obtenons le schéma relationnel suivant :

Médecin (ID, Nom, Prénom, Spécialité, N° Téléphone)

Patient (NSS, Nom, Prénom, Age, Sexe)

Médicament (Réf, Libellé, Composant, Dose, Format)

Symptôme (Code, Nom)

Maladie (Code, Nom)

Consultation(N, Date Consultation, #ID Médecin, #NSS Patient)

Préciser (#Code Symptôme, #N Consultation)

Identifier (#Code Maladie, #N Consultation)

Prescrire (#Ref Médicament, #N Consultation)

4.3. Requêtes SQL

Par la traduction du schéma relationnel (ci-dessus) en langage SQL, nous pouvons obtenir les requêtes LDD Create suivantes :

```
Create Table Medecin (
        id Integer Primary Key,
```

```
        nom Char(100),
        prenom Char(100),
        specialite Char(100),
        n_telephone Char(20)
);

Create Table Patient (
        nss Char(20) Primary Key,
        nom Char(100),
        prenom Char(100),
        age Integer,
        sexe Char(1)
);

Create Table Medicament (
        ref Char(13) Primary Key,
        libelle Char(100),
        composant Char(100),
        dose Char(20),
        format Char(20)
);

Create Table Symptome (
        code Char(20) Primary Key,
        nom Char(100)
);

Create Table Maladie (
        code Char(20) Primary Key,
        nom Char(100)
);

Create Table Consultation (
        n Integer Primary Key Auto_Increment,
        date_consultation Date,
        id_medecin Integer References Medecin(id),
        id_patient Char(20) References Patient(nss)
);

Create Table Preciser (
        code_symptome Char(20) References Symptome(code),
        n_consultation Integer References Consultation(n),
        Primary Key (code_ symptome, n_consultation)
);

Create Table Identifier (
        code_maladie Char(10) References Maladie(code),
        n_consultation Integer References Consultation(n),
        Primary Key (code_ maladie, n_consultation)
);
```

```
Create Table Presecrire (
        ref_medicament Char10) References Medicament(ref),
        n_consultation Integer References Consultation(n),
        Primary Key (ref_medicament, n_consultation)
);
```

5. La structure physique de la base de données

5.1. Définition

Le niveau physique désigne l'organisation des données de la base de données sur le disque dur. C'est à dire, comment les données sont-elles rangées physiquement sur le disque dur. C'est le niveau le plus bas du SGBD.

Pour pouvoir comprendre l'organisation physique d'une base de données, il faut d'abord comprendre le fonctionnement du disque dur lui-même. Il faut aussi garder en esprit les types de données et leur taille (en octet).

Les disques durs, comme étant des mémoires secondaires, trouvent leur intérêt par rapport à la RAM, le cache processeur et les registres, comme étant des mémoires principale, à cause de :

- Ils sont moins chers : les bases de données sont généralement volumineuses.
- Ils sont considérés comme des mémoires longs-termes; ce qui les rend nécessaires pour les bases de données.

Leur inconvénient majeur est leur lenteur. Les disques durs sont très lents (temps d'accès très grand) par rapport aux mémoires principales. Durant ce cours, nous allons nous intéresser aux disques durs de type HDD et non pas SSD. Les premiers sont encore les plus utilisés au moment de rédaction de ce cours.

5.2. Rappel : fonctionnement du disque dur

Structure générique

Un disque dur de type HDD respecte l'architeture suivante :

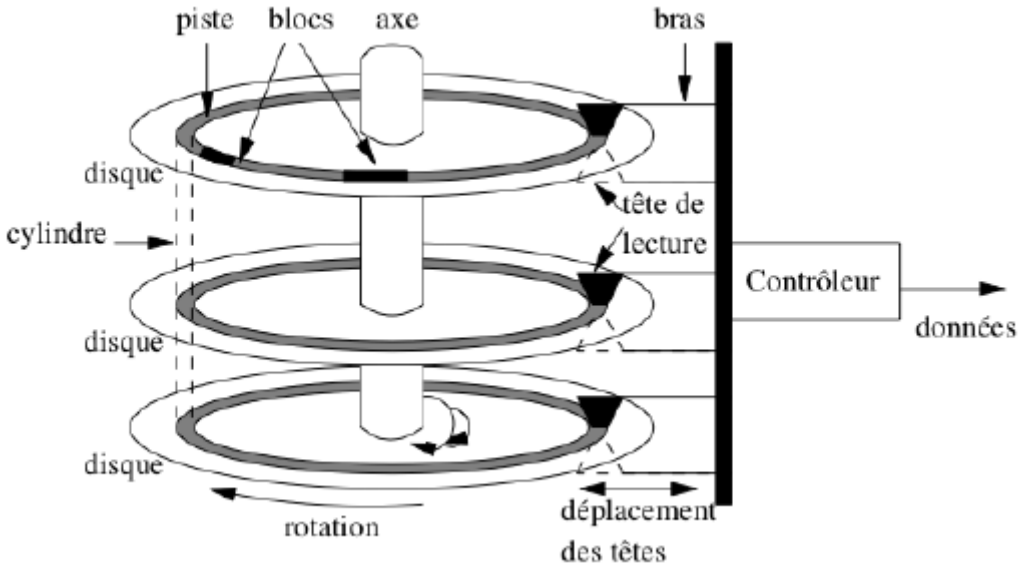


Figure 1. Structure d'un disque dur

Nous reprenons les définitions suivantes :

- Le disque dur se compose de plusieurs disques ou plateaux.
- Sur chaque disque, on peut écrire sur une surface (une face seulement) ou bien sur les deux surfaces (les deux faces). Dans le deuxième cas, chaque face dispose sa propre tête de lecture/écriture.
- Une surface est divisée en plusieurs pistes,
- Une piste est divisée en plusieurs blocs, c'est l'unité la plus petite qu'on peut lire.
- Tous les disques sont identiques et ils sont fixés (tous) à un axe qui permet la rotation.
- Les têtes de lecture/écriture sont fixées à des bras, ces derniers sont à leur tour fixés dans un deuxième axe. Cela permet, en association avec la rotation des plateaux, aux têtes de L/E de balayer toute la surface des disques (des plateaux).
- Le cylindre peut être obtenu en prennant une piste avec les pistes identiques à elle et qui se trouvent sur les autres plateaux. Cette notion est importante parce qu'elle représente la taille maximale qu'on peut lire ou écrire sur le disque dur sans déplacer les têtes de lecture/écriture.
- Le disque dur possède un contrôleur qui traduit une adresse à un mouvement qui permet de récupérer l'information (les données) désirées.
- Le disque dur permet un accès direct. C'est-à-dire, nous pouvons récupérer une information directement en utilisant son adresse sans être obligé de lire toutes les informations qui la précèdent.

Temps de lecture d'un bloc

La lecture n'est pas instantanée; elle prend un temps considérable par rapport aux autres mémoires primaires. Nous pouvons calculer ce temps de lecture en utilisant les formules suivantes (issues du principe du fonctionnement) :

Temps d'accès à un bloc = temps de recherche + temps de latence + temps de transfert

- **Temps de recherche :** C'est le temps nécessaire au positionnement de la tête sur la bonne piste. Il est de l'ordre de 7 à 10 microsecondes (3 à 8 msecs sur les serveurs),
- **Temps de latence :** délai de rotation des disques pour que le bloc soit sous la tête de lecture/écriture. On peut le calculer en sachant la vitesse de rotation (fournie généralement par le fabriquant). Pour une vitesse de 15000 rpm, le temps de latence sera environ 2 msecs (en moyenne, 4 msecs pour une rotation)
- **Temps de transfert :** le temps de lecture et de transfert du bloc. Généralement c'est le temps de passage du bloc sous la tête de lecture écriture vu que le temps de transfert de la tête vers le controleur est beaucoup plus court.

Le temps d'accès à un bloc peut être minimal. Cela arrive si la tête de lecture/écriture est déjà sur la bonne piste et le bloc ciblé se trouve sous la tête de lecture/écriture. Dans ce cas, la fonction prend sa valeur minimale :

Temps d'accès à un bloc (min) = temps de transfert

Ce temps d'accès peut aussi prendre une valeur maximale. Cela arrive lorsque la tête de lecture/écriture doit se déplacer entre toutes les pistes et le disque dur doit faire une rotation complète pour positionner le bloc sous la tête de lecture/écriture. La valeur maximale sera :

Temps d'accès à un bloc (max) = temps d'une rotation complète + temps de déplacement entre deux pistes * nombre de pistes + temps de transfert

Généralement, nous calculons le temps moyen vu que le positionnement est complètement aléatoire.

5.3. Organisation primaire et Organisation secondaire

La sauvegarde des données d'une base de données repose sur deux organisations à gérer :

- **Organisation primaire :** "Elle détermine la façon dont les enregistrements sont placés physiquement sur le disque". Cette organisation concerne les données concrètes de la base de données (toutes les données). Elle influence les algorithmes de recherche, insertion, modification et suppression.
- **Organisation secondaire :** Elle définit une structure d'accès auxiliaire qui permet d'accéder rapidement aux enregistrements. Elle n'implique pas toutes les données mais seulement les "clés" (champs) utilisés pour la recherche. Sa valeur ajoutée pour les autres opérations est remarquable vu que la suppression et la modification nécessitent une recherche avant.

Dans les deux cas, le fichier est constitué d'un ensemble d'enregistrements rangés dans des blocs. Cela représente la rubrique de base. La différence réside dans la taille et le type des enregistrements.

5.4. Les enregistrements

Un enregistrement peut être défini comme "une collection de valeurs ou éléments de données apparentés, chaque valeur (constituée de plusieurs octets, selon le type des données et selon le système) correspond à un champ particulier de l'enregistrement".

5.5. Taille de l'enregistrement

Pour la table suivante :

Exemple :

```
Create Table Employe (
        nom char(20),
        nss char(16),
        code_service integer,
        code_poste integer,
        ville char(20),
        date_recrutement date
);
```

La taille de l'enregistrement sera :

$$R = 20 + 16 + 4 + 4 + 20 + 8 = 72 \text{ octets.}$$

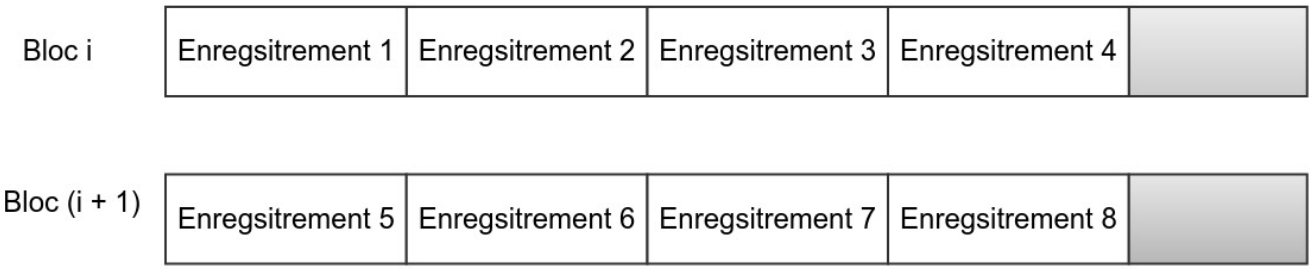
5.6. Rangement des enregistrements

Ayant étudié les enregistrements, il est temps de voir leur rangement dans les blocs.

Bloc de données

Puisque le bloc est l'unité la plus petite de lecture/écriture, un fichier est inévitablement composé de plusieurs blocs. Pour une meilleure gestion, il est important de garder des données sur le bloc (type, type contenu, pointeur vers le bloc suivant, etc.). Cette partie du bloc ne sera pas utilisée pour stocker des données. On appelle cet espace "Entête du bloc".

Pendant le rangement, un enregistrement n'est pas réparti sur plusieurs blocs; il appartient en entier à un seul bloc. En rangeant les enregistrements (en réservant l'espace de l'entête), un espace vide reste dans la fin de chaque bloc.



5.7. Taille du fichier

Fonctions à retenir

- **La fonction plancher :**

Elle permet d'arrondir x à la valeur entière inférieure.

Par exemple : plancher(3,7) = 3.

- **La fonction plafond :**

Elle permet d'arrondir x à la valeur entière supérieure.

Par exemple : plafond(3,7) = 4.

L'utilisation de ces fonctions d'arrondissement est nécessaire vu que :

- Les enregistrements ne sont pas répartis, ainsi, le nombre d'enregistrement dans un bloc est un nombre entier.
- Les blocs sont l'unité de base de sauvegarde, ainsi, un fichier contient un nombre entier de blocs.

Calcul de la taille du fichier

Soient :

- La taille d'un bloc : B octets,
- Nous gardons en esprit que nous réservons un espace au début de chaque bloc pour garder des données sur son contenu (en-tête).
- La taille d'enregistrement : R octets,
- r : le nombre d'enregistrement

On calcule :

- bfr : le facteur de blocage, le nombre d'enregistrements par bloc.

$$bfr = plancher((B - entête) / R);$$

- b : le nombre de blocs du fichier

$$b = plafond(r / bfr);$$

6. Conclusion

Dans ce chapitre, nous avons vu une introduction à la représentation logique et physique des données. Nous avons introduit aussi la dimension physique d'une base de données. Il existe plusieurs approches, techniques et structures de données pour l'organisation physique d'une base de données. Ce que nous avons vu n'est qu'une introduction minimale au domaine.