Notes de cours

Antoine Gagné

5 février 2016

Contents

Ce document regroupe l'ensemble de mes notes pour le cours **Modèles et langages de bases de données**.

Chapter 1

Introduction

Ce chapitre est une introduction aux bases de données et aux système de gestion de bases de données. Il expliquera leur raison d'exister, les fonctions typiques, les composants majeurs, etc.

1.1 L'ancêtre des bases de données

Le prédécesseur des bases de données est l'approche par fichiers. Cette approche utilisait un système décentralisé où chaque système avait ses propres données.

1.1.1 Problèmes

L'approche par fichiers comprend cinq problèmes majeurs.

1.1.1.1 Séparation et isolation des données

Dans une approche par fichiers, les données sont séparées et isolées. Cela fait en sorte qu'il est plus difficile d'accéder aux données. Lorsqu'on veut accéder à des données, on doit synchroniser notre traitement des fichiers pour s'assurer que les bonnes données sont extraites.

1.1.1.2 Duplication des données

Un autre problème de cette approche est la duplication des données. Chaque département a sa propre version des données. Cela fait en sorte que les départements vont avoir des données identiques entre eux dans certains cas. C'est donc du gaspillage de temps et d'argent, car les données être entrées plusieurs fois

par des personnes différentes. De plus, ça veut également dire qu'il y a plus d'espace de stockage qui est pris. Finalement, il y a plus de risques d'avoir des informations incorrectes, car il peut y arriver que la synchronisation entre les différents fichiers des départements ne s'est pas bien faite.

1.1.1.3 Dépendences des données

Les programmes dépendent d'un format de fichier particulier. Si le format change, toutes les applications connectées à ce fichier doivent changer aussi. Le format des données est donc assez rigide, car il n'y a pas d'abstraction quant à la disposition de celles-ci. De plus, changer une adresse de données va nécessiter la création d'un programme pour pouvoir changer tous les champs de manière à ce qu'ils soient conformes à la nouvelle structure.

1.1.1.4 Format de fichiers

Comme la structure des fichiers est dépendant du langage de programmation, il est plus difficile d'utiliser différents langages pour les différentes applications. En effet, un fichier généré dans un langage va être différent du même fichier généré à partir d'un autre langage.

1.1.1.5 Requêtes fixes et prolifération de programmes

Les besoins de nouvelles requêtes peuvent changer à tous moments. Étant donné que les requêtes sont écrites par les développeurs d'application, il était très difficile d'ajouter des nouvelles requêtes rapidement. Il n'y avait aucun moyen de créer des requêtes non planifiées pour aller voir les types de données disponibles. Le fait que ce soit les développeurs d'application qui devaient écrire toutes les requêtes faisait en sorte que leur charge de travail était énorme et, donc, dans certain cas, ils étaient obligés de couper dans certaines fonctionnalités. Par exemple, les fonctionnalités suivantes étaient souvent omises:

- Aucune mesure de sécurité ou de vérification de l'intégrité des données
- La récupération des données dans le cas d'un disfonctionnement logiciel ou matériel était limité ou inexistant
- Accès aux fichier était restreint à un utilisateur à la fois

1.1.1.6 Résumé

Tous les facteurs limitants de l'approche par fichiers peuvent être attribués à deux facteurs:

- La définition des données est incrustée dans les applications au lieu d'être enregistré séparemment et indépendemment
- Il n'y a aucun contrôle sur les accès et manipulations des données hormis celui imposé par les programmes

1.2 L'approche par base de données

L'approche par fichiers ne répondant pas aux besoins des entreprises, une nouvelle solution fut inventée: l'approche par base de données.

1.2.1 Les bases de données

1.2.1.1 Définition

Une **base de données** est une collection de données liées logiquement et sa description, conçu dans le but de répondre aux besoins d'information d'une organisation.

1.2.1.2 Avantages

Les bases de données ont plusieurs avantages par rapport à l'approche par fichiers. Par exemple:

- Une base de données a le minimum de duplication possible.
- Au lieu d'être liée à un seul département, elle est partagée à toute l'organisation complète.
- Elle peut être accédée par plusieurs utilisateurs en même temps.
- En plus de contenir les données, elle contient aussi une description de ces données.
- La définition des données est cachée aux utilisateurs qui ne voient que leurs définitions externes ce qui permet de protéger les programmes des modifications à la base de données.

1.2.2 Les systèmes de gestion de bases de données (DBMS ou SGBD)

1.2.2.1 Définition

Un **système de gestion de base de données** est un logiciel qui permet aux utilisateurs de définir, créer, maintenir et contrôler l'accès à la base de données.

1.2.2.2 Fonctionnalités fournies

Les systèmes de gestion de bases de données donnent accès à des fonctionnalités. Par exemple:

- Permet aux utilisateurs de définir la base de données avec un **langage de définition de données (DDL)**. Ce langage permet de spécifier les types, les structures et les contraintes à appliquer sur les données qui vont être entreposées dans la base de données.
- Permet aux utilisateurs de d'insérer, de mettre à jour, d'effacer et d'aller chercher des données dans la base de données avec un **langage de manipulation de données (DML)**. Le langage de manipulation de données permet de faire ces opérations avec un **langage de requête** comme le **langage structuré de requête (SQL)**.
- Permet un accès contrôlé à la base de données. Par exemple:
  - Système de sécurité qui empêche les accès non autorisés à la base de données
  - Système d'intégrité qui maintient la consistence des données
  - Système de contrôle d'accès multiple qui permet l'accès partagé à la base de données
  - Système de contrôle de récupération qui permet de remettre la base de données à un état antérieur après un problème matériel ou logiciel
  - Catalogue usager qui contient des descriptions des données dans la base de données

1.2.2.3 Avantages

Il y a plusieurs avantages aux systèmes de gestion de bases de données. Parmi ces avantages, on dénombre le contrôle de la redondance des données, la cohérence des données, plus d'informations avec le même nombre de données, le partage des données, une meilleure intégrité des données, une sécurité accrue, l'application des standards, des économies de taille, la balance de besoins contradictoires, l'amélioration de l'accessibilité et de la réactivité des données, une productivité accrue, maintenance améliorée grâce à l'indépendence des données, amélioration de l'accès simultané et de meilleurs services de restauration des données.

1.2.2.3.1 Contrôle de la redondance des données L'approche par bases de données essaie d'éliminer les redondances en intégrant les fichiers de sorte que des copies des mêmes données ne sont pas enregistrées. Toutefois, l'approche par bases de données n'élimine pas toutes les redondances, elle ne fait que contrôler la quantité de redondance. Dans certains cas, il est nécessaire de dupliquer des données clés pour modéliser des relations. Dans d'autres cas, il est désirable de dupliquer quelques données pour améliorer les performances.

1.2.2.3.2 Cohérence des données En éliminant ou en controllant les redondances, on réduit le risques d'incohérences. S'il n'y a qu'une copie d'une données dans la base de données, n'importe quelle mise à jour de sa valeur ne va devoir être fait qu'une fois. S'il y a plusieurs copies et que le système est au courant, il peut s'assurer que toutes les copies sont consistentes.

1.2.2.3.3 Plus d'informations avec le même nombre de données Comme toutes les informations sont accessibles par tous les départements d'une organisation, il est possible d'avoir plus d'information avec les mêmes données.

1.2.2.3.4 Partage des données Comme la base de données est commune à toute l'organisation, elle peut être accéder par tout le monde.

1.2.2.3.5 Meilleure intégrité des données L'intégrité d'une base de données réfère à la validité et à la cohérence des données enregistrées. L'intégrité est exprimé en termes de contraintes que la base de données ne peut violer.

1.2.2.3.6 Sécurité accrue La sécurité des bases de données est la protection de la base de données contre les utilisateurs non autorisés. L'intégraton permet à l'administrateur de la base de données de définir la sécurité de la base de données. Cette sécurité peut prendre la forme d'identifiants et de mots de passe.

1.2.2.3.7 Application des standards L'intégraton permet à l'administrateur de la base de données d'appliquer les standards nécessaires. Ceux-ci peuvent prendre la forme de format de données, format de la documentation, les procédures de mise à jour et règles d'accès.

1.2.2.3.8 Économies de taille Combiner toutes les données d'une organisations dans une base de données et créer un ensemble d'application qui marche avec cette seule source de données peut permettre d'économiser beaucoup d'argent. En effet, au lieu que chaque département ait leur propre budget pour la maintenant et le développement de leur système basé sur une approche fichier, les efforts peuvent être combinés pour être concentré vers la base de données uniques pouvant ainsi sauver de l'argent.

1.2.2.3.9 Balance de besoins contradictoires Chaque utilisateur ou département a des besoins qui peuvent être en conflit avec ceux des autres utilisateurs. Comme la base de données est sous le contrôle de l'administrateur de bases de données, celui-ci peut prendre des décisions à propos de la conception et de l'utilisation de la base de données qui vont tirer le maximum des ressources disponibles de l'organisation.

1.2.2.3.10 Amélioration de l'accessibilité et de la réactivité des données En raison de l'intégration, les données qui sont partagées à travers les frontières des différents départements sont directement accessibles aux utilisateurs. Cela donne un système qui a potentiellement beaucoup plus de fonctionnalités.

1.2.2.3.11 Productivité accrue Un système de gestion de gestion de bases de données fourni par défaut la plupart des fonctionnalités d'une approche par fichiers. Ainsi, le programmeur n'a pas à se concentrer sur les aspects de base de la base de données et peut se concentrer sur les aspects plus spécifiques à sa propre application.

1.2.2.3.12 Maintenance améliorée grâce à l'indépendence des données Un système de gestion de bases de données sépare la description des données des applications, rendant ainsi les applications immunisées contre les changements de descriptions des données.

1.2.2.3.13 Amélioration de l'accès simultané La plupart des systèmes de gestion de bases de données contrôlent les accès simultanés à la base de données rendant ainsi les problèmes d'accès simultanés impossible.

1.2.2.3.14 Meilleurs services de restauration et de sauvegarde des données Les systèmes de gestion de bases de données ont des mesures pour minimiser les dégats en cas de défaillance.

1.2.2.4 Désavantages

Il y a aussi quelques désavantages aux systèmes de gestion de bases de données tels que une plus grande complexité, une plus grande taille, le coût des systèmes de gestion de bases de données, les coûts additionnels de matériel, le coût de conversion, la performance et un plus gros impact en cas de défaillance.

1.2.2.4.1 Complexité La plupart des fonctionnalités dont nous nous attendons d'un système de gestion de bases de données font en sorte que c'est un logiciel très complexe. Une incompréhension du système peut amener des mauvaises décisions de conception qui peuvent avoir des conséquences sérieuses pour l'organisation.

1.2.2.4.2 Taille La complexité et toutes les fonctionnalités des systèmes de gestion de bases de données font en sorte que c'est un logiciel extrêmement lourd qui demande beaucoup d'espace disque et qui demande beaucoup de mémoire pour l'exécuter.

1.2.2.4.3 Coût des systèmes de gestion de bases de données Le coût des systèmes de gestion de bases de données varie en fonction des fonctionnalités et de l'environnement fournis. Également, il y a des frais annuels de maintenance qui s'imposent.

1.2.2.4.4 Coûts additionnels de matériel Les besoins en espace de disque pour la base de données et le système de gestion de bases de données peut demander à acheter du nouvel espace disque. De plus, pour atteindre les performances requises, il peut être nécessaire d'acheter une machine dédiée pour exécuter la base de données.

1.2.2.4.5 Coût de conversion Dans certains cas, le coût de conversion des applications existants pour qu'elles roulent sur le nouveau système peut être très grand.

1.2.2.4.6 Performance Une approche par fichiers permet de créer des applications et des formats de fichier qui ont un but spécifique. Cela fait en sorte que leur performance est souvent très bonne. Par contre, dans le cas d'un système de gestion de bases de données, on essaie d'être plus général dans le but de desservir plus d'applications. Cette généralité entraîne souvent un coût au niveau de la performance.

1.2.2.4.7 Plus gros impact en cas de défaillance. La centralisation des données augmente la vulnérabilité du système. Comme tout le monde dépend de la base de données, sa défaillance peut amener plusieurs opérations à s'arrêter.

1.2.2.5 Mécanisme de vision

En plus d'offrir toutes les fonctionnalités précédentes, les système de gestion de base de données offrent un **mécanisme de vue** qui permet aux utilisateurs d'avoir leur propre vue de la base de données. Par exemple, une vue pourrait permettre de ne voir que certaines entités de la base de données

1.2.2.5.1 Bénéfices Les vues offrent plusieurs bénéfices tels que:

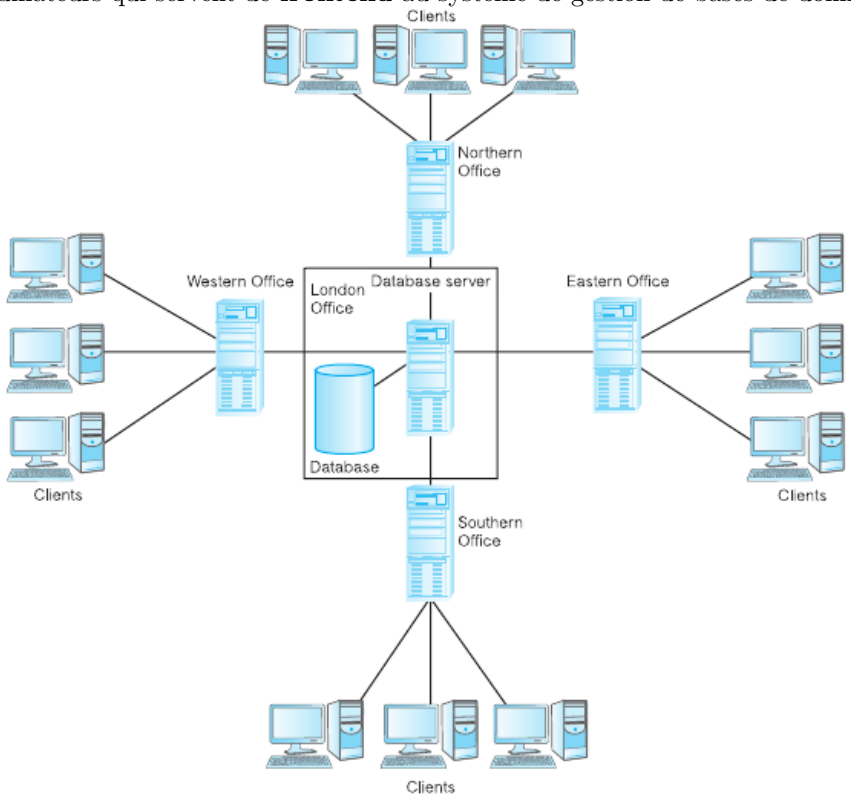
- Un niveau de sécurité supplémentaire, car on peut exclure des données que certains utilisateurs ne devraient pas voir
- Un mécanisme pour paramétrer l'apparence de la base de données
- Permet de présenter une vision immutable de la structure de la base de données même si la base de données en dessous est changée, car la vue va seulement présentée les données qui intéressent l'utilisateur. Donc, même si

les autres champs changent, tant que les champs qui intéressent l'utilisateur n'ont pas changé, la vue de l'utilisateur ne sera pas affectée.

1.2.2.6 Composants d'un système de gestion de bases de données

Un système de gestion de bases de données est composées de plusieurs choses.

1.2.2.6.1 Matériel Un système de gestion de bases de données a besoin de matériel pour fonctionner. Ce matériel peut aller d'un simple ordinateur personnel à un réseau d'ordinateurs. Quelques systèmes de gestion de bases de données nécessite un système d'exploitation particulier ou du matériel spécifique alors que d'autres, non. Les système de gestion de bases de données ont besoin d'un minimum de mémoire et d'espace disque pour fonctionner, mais ce minimum ne garantit pas une performance adéquate. Un exemple d'**architecture client-serveur** se retrouve à la figure suivante. Dans cette figure, on remarque le serveur central qui sert de **backend** au système de gestion de bases de données et les différents ordinateurs qui servent de **frontend** au système de gestion de bases de données.



1.2.2.6.2 Logiciel La composante logiciel comprend le logiciel de système de gestion de bases de données lui-même, les programmes d'application, le système d'exploitation et le logiciel de réseau si le système de gestion de bases de données est utilisé par un réseau.

1.2.2.6.3 Données Les données sont la partie la plus importante d'un système de gestion de bases de données. Elles servent de pont entre les humains et les machines. La base de données contient les données opérationnelles et les metadatas. La structure de la base de données est appelée **schéma**. Les schémas contiennent des **tables**. Les champs dans ces tables sont appelés des **attributs**.

1.2.2.6.4 Procédures Les procédures sont les instructions et les règles qui régissent la forme et l'utilisation de la base de données. Ces procédures sont destinées aux utilisateurs du système et au personnel qui est responsable de la base de données.

1.2.2.6.5 Humains Les personnes impliquées avec le système.

1.2.2.7 Conception de bases de données

La structure d'une base de données est déterminée lors du **design de la base de données**.

1.2.2.8 Rôles dans un environnement de bases de données

Il y a quatre types de personnes dans un environnement de bases de données. Il y a les administrateurs des données et de la base de données, les concepteurs de bases de données, les développeurs d'application et les utilisateurs.

1.2.2.8.1 Administrateurs des données et de la base de données Les administrateurs des données et de la base de données sont des rôles qui sont chargés du contrôle et de la gestion du système de gestion de la base de données et de ses données.

- L'**administrateur des données (DA)** est responsable de gérer les ressources de données, incluant la planification de la base de données, le développement et la maintenance des standards, politiques et procédures. Il doit aussi s'occuper du design conceptuel/logique de la base de données.
- L'**administrateur de la base de données (DBA)** est responsable de la réalisation physique de la base de données, incluant le design physique de la base de données et l'implantation, la sécurité, le contrôle d'intégrité, la maintenance du système opérationnel et d'assurer une performance satisfaisante des applications des utilisateurs.

1.2.2.8.2 Concepteurs de bases de données Il y a deux types de concepteurs de bases de données. Il y a les **concepteurs de bases de données logiques** et les **concepteurs de bases de données physiques**.

- **Concepteurs de bases de données logiques**: Responsables d'identifier les données (ou plutôt les entités et les attributs), les relations entre les données et les contraintes sur les données qui vont être enregistrées dans la base de données. Il doit avoir une bonne compréhension des données de l'organisation et des contraintes qui s'appliquent sur celles-ci.
- **Concepteurs de bases de données physiques**: Décident comment la conception logique de la base de données va être physiquement implantée. Cela consiste à:
  - Transformer le design logique de la base de données dans un ensemble de tables et de contraintes d'intégrité.
  - Sélectionner des structures d'entreposage et des méthodes d'accès aux données pour obtenir une bonne performance.
  - Établir les mesures de sécurité requises pour les données.

1.2.2.8.3 Développeurs d'application Les développeurs d'application sont responsables de développer des programmes qui vont fournir les fonctionnalités requises aux usagers. Ces applications vont interagir avec la base de données.

1.2.2.8.4 Utilisateurs Les utilisateurs sont ceux qui ont besoin des informations de la base de données. On distingue deux types d'utilisateurs, les **utilisateurs naïfs** et les **utilisateurs sophistiqués**.

- **Utilisateurs naïfs**: Ignorent tout du système de gestion de bases de données. Ils accèdent à la base de données par des programmes spécifiques. Ils utilisent les opérations de bases de données en appuyant sur des boutons ou en entrant des commandes simples.
- **Utilisateurs sophistiqués**: Familiés avec la structure de la base de données et des installations fournies par le système de gestion de bases de données. Ils sont capable d'utiliser un langage de requêtes comme le *SQL* pour exécuter les opérations requises. Ils sont parfois même capables d'écrire des applications pour leur propre usage.

Chapter 2

Modèle relationnel

Dans ce chapitre, nous aborderons le modèle relationnel et ses différents éléments.

2.1 Structure des données relationnelles

La structure des données relationnelles peut être décomposée dans les éléments suivants:

Relation (table ou fichier) Table avec des lignes et des colonnes.

Attribut (colonne ou champ) Colonne nommée d'une relation.

Domaine L'ensemble des valeurs allouées pour un ou plusieurs attributs.

Tuple (ligne ou archive) Nom donné à une ligne dans une relation (ou table).

Degré Le nombre d'attributs que la relation contient.

Une relation de degré un est appelée une relation unaire. Une relation de degré deux est appelée une relation binaire, une de degré trois est appelée une relation tertiaire et, après ça, le terme n-tiaire est utilisé.

Cardinalité Le nombre de tuples que la relation contient.

Base de données relationnel Une collection de relations normalisées avec des noms distincts de relations.

2.2 Relations mathématiques

Une relation mathématique est définie comme étant le **produit cartésien** de plusieurs ensembles. Le **produit cartésien** est défini comme suit:

Soit $D_1, D_2, \ldots D_n$, n ensembles. Leur produit cartésien est défini comme:

$$D_1 \times D_2 \times \cdots \times D_n = \{(d_1, d_2, \ldots, d_n) \mid d_1 \in D_1, d_2 \in D_2, \ldots d_n \in D_n\}$$

qui peut être réécrit comme

$$\prod_{i=1}^{n} D_i$$

2.3 Relations des bases de données

En applicant les notions vues à la section précédente, nous pouvons désormais définir un schéma de relation. Soit $A_1, A_2, \ldots, A_n$ des attributs avec comme domaines $D_1, D_2, \ldots, D_n$. Alors, l'ensemble $\{A_1 : D_1, A_2 : D_2, \ldots, A_n : D_n\}$ est un schéma de relation. Une relation R définie par le schéma de relation S est un ensemble qui met en relation les noms des attributs avec leurs domaines respectifs. Ainsi, la relation R est un ensemble de n-tuples:

$$(A_1 : d_1, A_2 : d_2, \ldots, A_n : d_n) \text{ tel que } d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n$$

Chaque élément du n-tuple consiste d'un attribut et d'une valeur pour cet attribut. Finalement, nous pouvons définir le **schéma relationnel de bases de données**. Soit $R_1, R_2, \ldots, R_n$ des ensembles de schémas de relation, alors nous pouvons écrire le schéma relationnel de bases de données comme:

$$R = \{R_1, R_2, \ldots, R_n\}$$

2.4 Propriétés des relations

Une relation a les propriétés suivantes:

- elle a un nom qui est distinct de tous les autres noms dans le schéma relationnel;
- chaque cellule contient exactement une valeur atomique;

- chaque attribut a un nom distinct;
- les valeurs d'un attribut proviennent tous d'un même domaine;
- chaque tuple est distinct; il n'y a pas de doublons;
- l'ordre des attributs n'a pas d'importance;
- l'ordre des tuples n'a pas d'importance, en théorie.

2.5 Clés relationnelles

Comme il n'y a pas de tuples identiques dans une relation, nous devons être capables d'identifier un ou plusieurs attributs qui identifient chaque tuple dans la relation. Ces attributs sont appelés des **clés relationnelles**.

2.5.1 Superclé

Une **superclé** est un attribut, ou un ensemble d'attributs, qui identifient de manière unique un tuple dans une relation.

2.5.2 Clé candidate

Une **clé candidate** est une superclé tel qu'aucun sous-ensemble propre est une superclé dans la relation. Une clé candidate K d'une relation R possède deux propriétés:

- *Unicité*: Pour chaque tuple de R, les valeurs de K identifient de manière unique ce tuple.
- *Irréductibilité*: Aucun sous-ensemble de K a la propriété d'unicité.

2.5.3 Clé composée

Une **clé composée** est une clé candidate qui possède plus qu'un attribut.

2.5.4 Clé primaire

Une **clé primaire** est la clé candidate choisie pour identifier chaque tuple de manière unique dans la relation.

2.5.5 Clé alternative

Les clés candidates qui ne sont pas choisies pour être la clé primaire sont dites **clés alternatives**.

2.5.6 Clé étrangère

Une **clé étrangère** est un attribut, ou un ensemble d'attributs, à l'intérieur d'une relation qui correspond à la clé candidate d'une relation (possiblement la même).

2.6 Contraintes d'intégrité

Comme chaque attribut à un domaine associé, il y a des contraintes (appelées **contraintes de domaine**) qui forment des restrictions sur l'ensemble des valeurs permises pour les attributs des relations. Il y a aussi des **règles d'intégrité** qui sont des contraintes qui s'appliquent sur toutes les instances de la base de données. Il y a deux règles d'intégrité importantes: l'**intégrité d'entité** et l'**intégrité référentielle**. Il existe également d'autres contraintes d'intégrité telles que la **multiplicité** et les **contraintes générales**.

2.6.1 Null

Les nulls représentent une valeur pour un attribut qui est présentement inconnu ou pas applicable pour ce tuple. Les nulls peuvent poser des problèmes d'implantation, car le modèle relationnel est basé le calcul de prédicat de premier ordre, qui permet seulement deux valeur: vrai ou faux.

2.6.2 Intégrité d'entité

Cette règle s'applique aux clé primaires des **relations de base**. Une relation de base est une relation qui correspond à une entité dans le schéma conceptuel. Cette règle stipule que, dans une relation de base, aucun attribut de la clé primaire ne peut être nul. En effet, une clé primaire est l'identifiant minimal qui est utilisé pour identifier les tuples de manière unique. Si on inclut les nulls, nous disons que tous les attributs ne sont pas nécessaires pour identifier de manière unique les tuples. Il y a donc une contradiction.

2.6.3 Intégrité référentielle

Cette règle s'applique seulement aux clés étrangères. Elle stipule que, si une clé étrangère existe dans une relation, soit la valeur de la clé étrangère correspond à la valeur d'une clé candidate d'un tuple dans sa relation maison ou la valeur de la clé étrangère doit être complètement nulle.

2.6.4 Contraintes générales

Les contraintes générales sont des règles additionnelles spécifiées par les utilisateurs ou les administrateurs de bases de données de la base de données qui définissent ou contraignent des aspects de l'entreprise.

2.7 Vues

Dans le modèle relationnel, une vue n'est pas la structure de la base de données telle que l'aperçoit un utilisateur particulier. Une vue est plutôt une **relation virtuelle** ou une **relation dérivée**, c'est-à-dire une relation qui n'existe pas par elle-même, mais qui peut être dérivée dynamiquement d'une ou plusieurs relations de base.

2.7.1 Définition

Une vue est le résultat dynamique d'une ou plusieurs opérations opérant sur les relations de base afin de produire une autre relation. Une vue est une *relation virtuelle* qui n'existe pas nécessairement dans la base de données mais qui peut être produite sur demande par un utilisateur donné au moment de la requête. Les vues semblent exister pour l'utilisateur et peuvent être manipulées, mais elles ne sont pas nécessairement enregistrées dans la base de données comme une relation de base. Toutes les opérations sur les vues sont automatiquement traduites en opération sur les relations de base duquelles celles-ci sont dérivées. De plus, les vues sont **dynamiques** ce qui impliquent que les changements aux relations de base influeront immédiatement les vues.

2.7.2 Utilités

Le mécanisme de vues est désirable pour plusieurs raisons:

- Fourni un mécanisme puissant et flexible pour cacher des parties de la base de données à certains utilisateurs. Les utilisateurs ne sont pas au courant des attributs cachés par la vue.
- Permet aux utilisateurs d'accéder aux données d'une façon adaptée à leurs besoins. De cette façon, les même données peuvent être vues par différentes personnes de différentes façons.
- Peut simplifier des opérations complexes sur les relations de base.

Une vue devrait être développée de sorte qu'elle supporte le modèle externe que l'utilisateur trouve familier. Les vues fournissent une indépendence des données logiques.

2.7.3 Mise à jour des vues

Toutes les mises à jour à une relation de base devraient immédiatement être reflétées dans les vues qui font références à cette relation de base. De façon similaire, si une vue est mise à jour, la relation de base sur laquelle elle est basée devrait être mise à jour aussi. Toutefois, il y a des restrictions sur les types de modification qui peuvent être faits sur les vues. Les conditions sous lesquelles la mise à jour à partir de la vue est possible sont les suivantes:

- Les mises à jour sont permises sur une vue définie en utilisant une requête simple qui implique une seule relation de base et qui contient ou bien la clé primaire ou une clé candidate de la relation de base.
- Les mises à jour ne sont pas permises par des vues impliquant de multiples relations de base.
- Les mises à jour ne sont pas permises par des vues impliquant l'aggrégation ou le groupement d'opérations.

Chapter 3

Normalisation

Dans ce chapitre, nous allons parler de la normalisation et de ses buts. Nous allons aussi aborder le concept de dépendances fonctionnelles.

3.1 Redondance de données et anomalies de mises à jour

En éliminant la redondance dans les bases de données, on en retire les bénéfices suivants:

- les mises à jour des données sont faites avec un nombre minimal d'opérations réduisant ainsi les risques d'incohérences dans la base de données;
- le nombre d'espace de stockage requis pour les relation est diminué réduisant ainsi les coûts.

Les relations qui ont des données redondantes peuvent avoir des problèmes appelés des **anomalies de mise à jour**. Il y a trois types d'anomalies de mise à jour: les **anomalies d'insertion**, les **anomalies de suppression** et les **anomalies de modification**.

3.1.1 Anomalies d'insertion

Si on ajoute des nouvelles données à la base de données, il peut y avoir des incohérences avec les champs déjà existants. De plus, si la table est mal faite, on peut être forcé d'ajouter des *nulls* pour pouvoir ajouter de nouveaux tuples.

3.1.2 Anomalies de suppression

Si on efface un attribut d'un tuple qui contient la dernière information sur un des attributs, cette information sera perdu lorsque l'attribut sera enlevé.

3.1.3 Anomalies de modification

Si on modifie la valeur d'un attribut, il faut s'assurer que dans la table, toutes les tuples qui contiennent cet attribut sont modifier. Sinon, on risque d'avoir une anomalie de modification. On peut éviter ces anomalies en décomposant les relations en plus petites relations. Cette façon de faire possède deux propriétés intéressantes:

- la propriété du **raccord sans pertes** qui assure que toutes les instances de la relation originale peuvent être identifiées avec les relations plus petites.
- la propriété de **préservation des dépendances** qui assure qu'une contrainte de la relation originale sera maintenue en maintenant la contraintes sur les relations plus petites.

3.2 Dépendances fonctionnelles

Les **dépendances fonctionnelles** jouent un rôle important dans la normalisation des bases de données.

3.2.1 Définition

Assumons un schéma relationnel possédant les attributs $(A, B, C, \ldots, Z)$ et une base de données décrite par une **relation universelle** $R = (A, B, C, \ldots, Z)$. Cela veut dire que tous les attributs dans la base de données ont un nom unique. Une **dépendance fonctionnelle** décrit le lien entre les attributs d'une relation. Par exemple, si A et B sont des attributs de la relation R, B est fonctionnellement dépendant de A (dénoté $A \mapsto B$), si chaque valeur de A est associée à seulement une valeur de B (A et B peuvent consister de plus d'un attributs).

3.2.2 Déterminant

Le terme **déterminant** réfère à l'attribut, ou le groupe d'attributs, du côté gauche de la flèche d'une dépendance fonctionnelle.

3.2.3 Dépendance fonctionnelle complète

Une **dépendance fonctionnelle complète** est une dépendance fonctionnelle dans laquelle le déterminant à le nombre minimal d'attribut pour avoir une dépendance fonctionnelle avec les attributs à droite de la flèche.

3.2.4 Dépendance fonctionnelle partielle

Une dépendance fonctionnelle est dite dépendance fonctionnelle partielle dans laquelle un des attributs peut être enlevé du déterminant et la dépendance fonctionnelle est encore valide.

3.2.5 Dépendance transitive

Si A, B et C sont des attributs d'une relation tel que si $A \mapsto B$ et $B \mapsto C$, alors C est dépendant transitif de A via B (si A n'est pas fonctionnellement dépendant de B ou C).

3.2.6 Résumé

Les dépendances fonctionnelles qui nous intéressent pour la normalisation possèdent les propriétés suivantes:

- Il y a une relation *un à un* entre les attributs du déterminant et ceux du côté droit d'une dépendance fonctionnelle.
- Elles sont toujours valides
- Le déterminant a le minimum d'attributs nécessaires pour maintenir une dépendance fonctionnelle avec le côté droit. Il doit donc y avoir une dépendance fonctionnelle complète entre les attributs du côté gauche avec ceux du côté droit.

3.3 Normalisation et formes normales

Dans cette section, nous verrons la normalisation ainsi que les différentes formes normales ainsi que les techniques pour les obtenir.

3.3.1 Définition de la normalisation

La **normalisation** est une technique pour produire un ensemble de relations avec des propriétés désirables selon les besoins de données de l'entreprise. Les caractéristiques d'un bon ensemble de relations sont les suivantes:

- le *nombre minimal d'attributs* pour supporter les besoins en données de l'entreprise;
- les attributs avec des relations logiques (décrites comme des **dépendances fonctionnelles**) sont dans les mêmes relations;
- il y a un *minimum de redondance*, avec chaque attribut qui se retrouve seulement une fois dans la base de données excepté les attributs qui sont ou qui font parties de clés étrangères qui sont essentielles pour lier des relations.

3.3.2 Forme non normalisée (UNF) à première forme normale

Dans cette section, nous verrons comment passer de la forme non normalisée à la première forme normale.

3.3.2.1 Définition de la forme non normalisée (UNF)

Une table qui contient une ou plusieurs répétitions de groupe.

3.3.2.2 Définition de la première forme normale (1NF)

Une relation dans laquelle l'intersection de chaque ligne et colonne contient seulement une valeur.

3.3.2.3 Technique de normalisation

Il y a deux approches pour passer de la forme non normalisée à la première forme normale.

3.3.2.3.1 Mise à plat Remplir les colonnes de lignes vides avec les données non dupliquées.

3.3.2.3.2 Séparer les données dupliquées Placer les données dupliquées avec une copie de l'attribut clé dans une relation différente.

Chapter 4

SQL

4.1 Exemple

```
    create table PROPRIETE_A_LOUER
    (NUM PROPRIETE varchar(5) not null,
        PIECES number(2) not null default 4,
        LOCATION number(6,2) not null default 600,
        NUM_PROPRIETAIRE varchar(5) not null,
        NUM_PERSONNEL varchar(5)
        NUM_FILIALE char(4) not null,
        constraint PK_PROPRIETE_A_LOUER primary key
        (NUM_PROPRIETE),
        constraint FK_PROPRIETE_NUM_PERSONNEL foreign key
        (NUM PERSONNEL) references PERSONNEL on delete set null,
        constraint CK_PIECES_RANGE check (PIECES between 1 and 15),
        constraint CK_LOCATION_RANGE check(LOCATION between 0
and
        9999.99);
```

Chapter 5

Annexe

Application de base de données Une application qui interagit avec la base de données à un certain point dans son exécution.

Abstraction des données Nom donné à l'approche qui cache la définition interne des données aux utilisateurs de la base de données et expose seulement la définition externe.

Administrateur des données (DA) Est responsable de gérer les ressources de données, incluant la planification de la base de données, le développement et la maintenance des standards, politiques et procédures. Il doit aussi s'occuper du design conceptuel/logique de la base de données.

Administrateur de la base de données (DBA) Est responsable de la réalisation physique de la base de données, incluant le design physique de la base de données et l'implantation, la sécurité, le contrôle d'intégrité, la maintenance du système opérationnel et d'assurer une performance satisfaisante des applications des utilisateurs.

Anomalies de mise à jour Problèmes qui peuvent survenir lorsqu'on met à jour une base de données qui comporte de la redondance.

Anomalies de modification Anomalies de mise à jour qui peuvent survenir lors de la modification d'un attribut.

Anomalies de suppression Anomalies de mise à jour qui peuvent survenir lors de la suppression d'un tuple.

Anomalies d'insertion Anomalies de mise à jour qui peuvent survenir lors de l'insertion d'un nouveau tuple.

Attribut (colonne ou champ) Définition 1: Propriété qui décrit un aspect de l'objet que nous souhaitons enregistrer.

Définition 2: Colonne nommée d'une relation.

Définition 3: Décrit des propriétés des données ou des relations entre les données qui sont importantes pour l'entreprise.

Base de données (BD) C'est une collection de données liées logiquement et sa description, conçu dans le but de répondre aux besoins d'information d'une organisation.

Base de données relationnel Une collection de relations normalisées avec des noms distincts de relations.

Cardinalité Le nombre de tuples que la relation contient.

Catalogue système (dictionnaire de données ou metadata) Le nom donné à la description des données dans une base de données.

Clé alternative (AK) Clé candidate qui n'a pas été choisie comme la clé primaire.

Clé candidate (CK) Une superclé tel qu'aucun sous-ensemble propre est une superclé dans la relation.

Clé composée Une clé candidate qui possède plus qu'un attribut.

Clé étrangère Un attribut, ou un ensemble d'attributs, à l'intérieur d'une relation qui correspond à la clé candidate d'une relation (possiblement la même).

Clé primaire (PK) Clé candidate choisie pour identifier chaque tuple de manière unique dans la relation.

Concepteurs de bases de données logiques Responsables d'identifier les données (ou plutôt les entités et les attributs), les relations entre les données et les contraintes sur les données qui vont être enregistrées dans la base de données. Il doit avoir une bonne compréhension des données de l'organisation et des contraintes qui s'appliquent sur celles-ci.

Concepteurs de bases de données physiques Décident comment la conception logique de la base de données va être physiquement implantée.

Contraintes Règles de cohérence que la base de données ne peut enfreindre.

Contraintes d'intégrité Restrictions sur l'ensemble des valeurs permises pour les attributs des relations.

Contraintes générales Règles additionnelles spécifiées par les utilisateurs ou les administrateurs de bases de données de la base de données qui définissent ou contraignent des aspects de l'entreprise.

Déterminant L'attribut, ou le groupe d'attributs, du côté gauche de la flèche d'une dépendance fonctionnelle.

Degré Le nombre d'attributs que la relation contient.

Dépendance fonctionnelle (DF) Décrit le lien entre les attributs d'une relation.

Dépendance fonctionnelle complète Définition 1: Une dépendance fonctionnelle dans laquelle le déterminant à le nombre minimal d'attribut pour avoir une dépendance fonctionnelle avec les attributs à droite de la flèche.

Définition 2: Indique que si A et B sont des attributs d'une relation, B est complètement fonctionnellement dépendant de A si B est fonctionnellement dépendant de A, mais pas aucun sous-ensemble de A.

Définition 3: Soit A et B, deux attributs d'une relation et $C \subset A$, alors $(A \mapsto B) \wedge \neg(C \mapsto B)$.

Dépendance partielle (DP) Définition 1: Une dépendance fonctionnelle dans laquelle un des attributs peut être enlevé du déterminant et la dépendance fonctionnelle est encore valide.

Définition 2: Soit A et B, deux attributs d'une relation et $C \subseteq A$, alors $C \mapsto B$.

Dépendance transitive (DT) Définition 1: Si A, B et C sont des attributs d'une relation tel que si $A \mapsto B$ et $B \mapsto C$, alors C est dépendant transitif de A via B (si A n'est pas fonctionnellement dépendant de B ou C).

Définition 2: Soit A, B et C, des attributs d'une relation tel que $(A \mapsto B) \wedge (B \mapsto C) \Rightarrow (A \mapsto C)$, si $\neg(B \mapsto A) \vee \neg(C \mapsto A)$

Domaine L'ensemble des valeurs allouées pour un ou plusieurs attributs.

Entité Objet distinct (une personne, un endroit, une chose, un concept ou un événement) dans l'organisation qui doit être représenté dans la base de données.

Forme non normalisée (UNF) Une table qui contient une ou plusieurs répétitions de groupe.

Forme normale (NF) :

Indépendance des données La séparation de la description des données des applications rendant ainsi les applications immunisées aux changement de la description des données.

Intégrité d'entité Règle d'intégrité qui stipule que, dans une relation de base, aucun attribut de la clé primaire ne peut être nul.

Intégrité référentielle Règle d'intégrité qui stipule que, si une clé étrangère existe dans une relation, soit la valeur de la clé étrangère correspond à la valeur d'une clé candidate d'un tuple dans sa relation maison ou la valeur de la clé étrangère doit être complètement nulle.

Langage de définition des données (DDL) Permet aux utilisateurs de spécifier les types, les structures et les contraintes à appliquer sur les données qui seront entreposées dans la base de données.

Langage de manipulation des données (DML) Permet aux utilisateurs d'insérer, de mettre à jour, d'effacer et d'aller chercher des données dans la base de données.

Langage de requête Permet de faire les opérations des langages de manipulation des données. Un exemple de langage de requête est le *SQL*.

Normalisation Définition 1: Une technique pour produire un ensemble de relations avec des propriétés désirables selon les besoins de données de l'entreprise.

Définition 2: Une technique formelle pour analyser les relations basée sur leur clé primaire (ou clés candidates) et les dépendances fonctionnelles.

Null Représente une valeur pour un attribut qui est présentement inconnu ou pas applicable pour ce tuple.

Première forme normale (1NF) Une relation dans laquelle l'intersection de chaque ligne et colonne contient seulement une valeur.

Programme d'application Un programme informatique qui interagit avec la base de données en envoyant des requêtes (la plupart du temps des instructions *SQL*) au système de gestion des bases de données.

Règles d'affaires Les contraintes d'une organisation sur les données.

Règles d'intégrité Contraintes ou restrictions qui s'appliquent sur toutes les instances de la base de données.

Relation (table ou fichier) Définition 1: Une association entre plusieurs entités.

Définition 2: Table avec des lignes et des colonnes.

Relation de base Définition 1: Relation qui correspond à une entité dans le schéma conceptuel.

Définition 2: Relation nommée qui correspond à une entité dans le schéma conceptuel, dont les tuples sont entreposés physiquement dans la base de données.

Relation virtuelle (relation dérivée) Relation qui n'existe pas par elle-même, mais qui peut être dérivée dynamiquement d'une ou plusieurs relations de base.

Schéma Structure de la base de données.

Schéma conceptuel (modèle conceptuel) L'ensemble de tous les schémas de la base de données.

Schéma de relation Une relation nommée définie par un ensemble d'attributs et de paires de noms de domaines.

Schéma de base de données relationnel Un ensemble de schémas de relation avec chacun un nom distinct.

Superclé Attribut, ou un ensemble d'attributs, qui identifient de manière unique un tuple dans une relation.

Système de base de données L'ensemble des applications qui interagissent avec la base de données et le système de gestion de base de données.

Système de gestion de base de données (DBMS ou SGBD (en français)) Un logiciel qui permet aux utilisateurs de définir, maintenir et contrôler l'accès à la base de données.

Tuple (ligne ou archive) Nom donné à une ligne dans une relation (ou table).

Utilisateurs naïfs Ignorent tout du système de gestion de bases de données. Ils accèdent à la base de données par des programmes spécifiques. Ils utilisent les opérations de bases de données en appuyant sur des boutons ou en entrant des commandes simples.

Utilisateurs sophistiqués Familiés avec la structure de la base de données et des installations fournies par le système de gestion de bases de données. Ils sont capable d'utiliser un langage de requêtes comme le *SQL* pour exécuter les opérations requises. Ils sont parfois même capables d'écrire des applications pour leur propre usage.

Vue Résultat dynamique d'une ou plusieurs opérations opérant sur les relations de base afin de produire une autre relation. Une vue est une *relation virtuelle* qui n'existe pas nécessairement dans la base de données mais qui peut être produite sur demande par un utilisateur donné au moment de la requête.