Concepteur Développeur en Informatique

Assurer la persistance des données

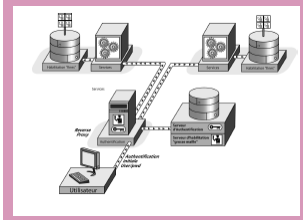
Normalisation des données

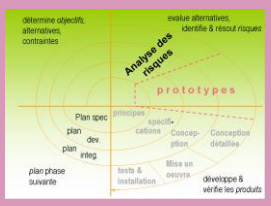
Accueil

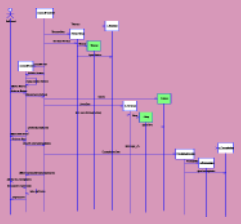
Apprentissage

PAE

Evaluation







Localisation : U02-E01-S03

Sommaire

1 Introduction

Nous allons aborder dans ce document les principes de normalisation des données. Nous parlerons de modèles de données conforment à la 1ère, 2ème ou 3ème forme normale.

Dans une base de données relationnelle, une forme normale désigne un type de relation particulier entre les entités. Comme son nom l'indique, normale décrit la norme.

La normalisation des modèles de données permettra donc de s'assurer de la robustesse de la conception de la base pour améliorer la modélisation et optimiser la mémorisation des données en évitant la redondance et les problèmes sous-jacents de mise à jour ou de cohérence.

La normalisation s'applique à toutes les entités et aux relations porteuses de propriétés.

On dira d'un modèle qu'il est en première, deuxième ou troisième forme normale. Il s'agit là des règles de normalisation les plus importantes, d'autres règles pouvant être appliquées mais de manière non systématique.

La forme normale vient après la simple validité d'un modèle relationnel, c'est-à-dire que les valeurs des différents attributs soient bien en dépendance fonctionnelle avec la clé primaire (complètement déterminés par la clé primaire).

En pratique, la première et la deuxième forme normale sont nécessaires pour avoir un modèle relationnel juste. Les formes normales supplémentaires ont leurs avantages et leurs inconvénients.

Les avantages sont :

- de limiter les redondances de données
- de limiter les incohérences de données qui pourraient les rendre inutilisables
- d'éviter les processus de mise à jour récurrents

Les inconvénients sont des temps d'accès qui peuvent être plus longs si les requêtes sont trop complexes et mettent en œuvre de très nombreuses tables.

La troisième forme normale reste généralement une des meilleures solutions d'un point de vue architecture de bases de données. Toutefois, pour des bases de données très volumineuses, cela n'est pas toujours le cas. Il s'agit de choisir l'équilibre entre deux options :

- la génération dynamique des données via les jointures entre tables
- l'utilisation statique de données correctement mises à jour

La normalisation des modèles de données a été popularisée principalement par la méthode Merise. La principale limite de la normalisation est que les données doivent se trouver dans une même base de données (dans un seul schéma relationnel).

Il convient d'être prudent lorsqu'on renonce à la forme normale. Il n'est pas garanti qu'une forme dé-normalisée améliore les temps d'accès.

En effet, la redondance peut entrainer une explosion des volumes de données qui peuvent écrouler les performances ou saturer les disques durs

2 La première forme normale

Notée 1FN - première forme normale :

Respecte la première forme normale, la relation dont tous les attributs :

- contiennent une valeur atomique (les valeurs ne peuvent pas être divisées en plusieurs sous-valeurs dépendant également individuellement de la clé primaire). Ainsi, le nom et le prénom d'une personne seront des propriétés distinctes.
- contiennent des valeurs non répétitives (le cas contraire consiste à mettre une liste dans un seul attribut). On pourrait citer pour exemple les enfants d'un salarié. Ces propriétés doivent être définies au sein d'une entité spécifique qui sera associée à l'entité salarié.
- Les attributs qui composent l'identifiant (clé primaire) conservent une valeur constante. Ainsi, ils ne sont pas affectés par des aspects temporels (utiliser par exemple la date de naissance plutôt que l'âge) ou des événements particuliers (changement nom usage pour la femme lors de son mariage en France).
- L'identifiant est irréductible : aucun sous-ensemble de l'identifiant ne permet de garantir l'unicité (clé primaire)
- L'identifiant est une valeur **Unique** qui permet de discriminer deux tupples

Le non respect des deux premières conditions de la 1FN rend la recherche parmi les données plus lente parce qu'il faut analyser le contenu des champs.

3 La deuxième forme normale

Notée 2FN - deuxième forme normale

Respecte la seconde forme normale, la relation respectant la première forme normale et dont :

- Tout attribut de l'association dépend de la totalité de la clé primaire et non d'une partie de celle-ci.

Considérons la relation ExercerActivité entre Salarié vers Service.

Salarie(matricule, nom)
Service(codeservice, désignation)
ExercerActivité(codeservice, matricule, fonction, adresse…)

La propriété « fonction » peut être acceptée pour l'association si cette propriété est bien en dépendance fonctionnelle totale avec l'identifiant de l'association ici (codeservice,matricule) La propriété « Fonction » correspond à la fonction occupée par un salarié donné dans le service considéré".

La propriété « adresse » devra être rejetée car chaque service est implanté à une adresse précise : la propriété « adresse »n'est alors en dépendance fonctionnelle qu'avec une partie de l'identifiant.

Le non respect de la 2FN entraine une redondance des données qui encombrent alors inutilement la mémoire et l'espace disque.

4 La troisième forme normale

Notée 3FN - troisième forme normale

Respecte la troisième forme normale, la relation respectant la seconde forme normale et dont :

- tous les attributs n'appartenant pas à la clé ne dépendent pas d'un attribut non-clé. En d'autre terme, la dépendance fonctionnelle est directe. Dans le cas d'une adresse, la ville dépend du code postal et non de l'identifiant adresse. De la même façon, une adresse est située dans un pays mais un pays ne dépend pas de l'adresse.
- La relation suivante n'est pas en 3FN :

VéhiculeLoué(*numVéhicule, numClient*, codeCategorie, catégorie, kilométrage)

- Elle doit être décomposée en :
- VéhiculeLoué(numVéhicule, numClient, codeCategorie, kilométrage)
- Catégorie(codeCategorie, catégorie)

5 Règles de vérification

5.1 Règles de vérification Propriété et Entité type

5.1.1 Règle de non répétitivité :

Chaque propriété rattachée à un individu type doit vérifier la règle suivante :

A toute occurrence de l'individu type, il ne peut y avoir au plus qu'une valeur de la propriété dans le système d'information. Si cela n'est pas le cas, alors la propriété concernée n'appartient pas à l'individu type. Par exemple, ici la propriété n° véhicule:

ASSURE
N° Sociétaire Nom … *n° véhicule*

5.1.2 Règle d'identification :

On doit pouvoir faire référence distinctement à chaque occurrence d'un individu type. Pour cela, tout individu type doit être pourvu d'un identifiant. L'identifiant peut être :

- Une propriété naturelle. Par exemple le nom d'un pays pour l'entité Pays
- Une propriété artificielle. Numéros, références chronologiques, …

- Une propriété composée. Identifiant composite : N° INSEE par exemple Sexe+Annee+Mois+département+commune+chrono

Un identifiant doit être :

- Univalué : à une occurrence correspond une seule valeur
- Discriminant : à une valeur correspond une seule occurrence
- Stable : la valeur de l'identifiant d'une occurrence demeure inchangée jusqu'à la destruction de l'occurrence.

5.1.3 Règle de distinction :

Les occurrences d'un individu type doivent être distinguables. C'est la pertinence du choix de l'identifiant qui permettra de vérifier cette règle.

5.1.4 Règle d'homogénéité :

Il est souhaitable que les propriétés rattachées à un individu type aient un sens pour toutes les occurrences de celui-ci.

5.1.5 Règles de vérification relation type

La relation type modélise des associations entre deux ou plusieurs occurrences d'individus type.

On appelle collection la liste des individus en relation.

Il est souhaitable d'utiliser un verbe à l'infinitif pour exprimer la relation.

Une relation type n'a pas d'identifiant propre. Son identifiant est composé des valeurs des identifiants des individus types en relation.

La relation type peut être porteuse de propriétés. Les valeurs de ces propriétés dépendent des occurrences de sa collection.

5.1.6 Règle d'unicité

Il ne peut y avoir plus d'une valeur d'une propriété d'une relation type pour une occurrence d'une relation type.

5.1.7 Règle de normalisation

Une propriété d'une relation type doit dépendre de la totalité des individus types qui participent à la relation.

6 Exemple de normalisation avec SQL Server

La conception logique de la base de données, comprenant les tables et les relations qui les unissent, constitue le cœur d'une base de données relationnelle. *L'exemple donné ici même avec SQL Server vaut tout autant avec une autre base de données cible, Oracle ou DB2. Les règles à appliquer sont rigoureusement les mêmes.*

Avec une conception logique bien pensée, vous obtiendrez une base de données et des performances d'application optimales. À l'inverse, une mauvaise conception peut détériorer les performances de tout le système.

Normaliser la conception logique d'une base de données implique d'utiliser des méthodes formelles pour répartir les données sur plusieurs tables reliées les unes aux autres.

Vous aurez fait appel pour cela à des méthodes de conception, telle Merise, afin de vous assurer de la conformité de votre conception aux règles de l'art.

Une base de données normalisée se caractérise souvent par un grand nombre de tables étroites (comportant moins de colonnes). À l'inverse, une base de données non normalisée se caractérise par un grand nombre de tables larges (comportant plus de colonnes).

La normalisation, dans des limites raisonnables, apportera souvent une amélioration des performances.

Les avantages de la normalisation sont entre autres :

- Un tri et une création d'index plus rapides.
- Un nombre important d'index organisés en clusters.
- Des index plus étroits et compacts.
- Un nombre plus réduit d'index par table, ce qui améliore les performances des instructions de mise à jour telles INSERT, UPDATE et DELETE.
- Un nombre plus réduit de valeurs NULL et des risques d'incohérences plus faibles.

Lorsque la normalisation augmente, le nombre et la complexité des jointures nécessaires pour récupérer les données augmentent aussi. Or des jointures relationnelles trop complexes entre des tables trop nombreuses peuvent avoir un effet néfaste sur les performances.

Dans la théorie des bases de données relationnelles, des règles de normalisation identifient les attributs qui doivent être présents (ou absents) dans une base de données bien conçue.

Vous trouverez ci-après quelques règles qui pourront vous aider à bien concevoir votre base de données :

- Une table doit avoir un identificateur.

La théorie de la conception des bases de données pose comme loi fondamentale que chaque table doit avoir un identificateur de ligne unique (enregistrement), une colonne ou un jeu de colonnes pouvant servir à distinguer un enregistrement d'une table de tous les autres. Chaque table doit avoir une colonne d'identification.

Par ailleurs, deux enregistrements ne peuvent pas partager la même valeur d'identification. La ou les colonnes faisant office d'identificateur de ligne unique pour une table constituent la clé primaire de cette table.

Cet identificateur doit être stable. Sa valeur ne sera pas modifiée tout au long de sa vie (depuis sa création jusqu'à sa suppression).

- Une table ne doit stocker que les données d'un unique type d'entités.

En essayant de stocker trop d'informations dans une table, vous risquez de nuire à l'efficacité et la fiabilité de la gestion des données de cette table.

Dans l'exemple de la base de données pubs, les informations concernant le titre et l'éditeur étaient stockées dans deux tables distinctes.

Bien qu'il soit possible de disposer de colonnes pour le titre et pour son éditeur au sein de la table des titres, une telle conception n'est pas sans poser quelques problèmes.

En effet, les informations concernant l'éditeur doivent être ajoutées et stockées pour chaque livre publié par cet éditeur. La base de données exige un espace de stockage supplémentaire.

Si l'adresse de l'éditeur vient à changer, la modification devra être effectuée pour chaque titre. Quand le dernier livre d'un éditeur donné est supprimé de la table des titres, les informations concernant cet éditeur sont perdues.

La base de données pubs stocke donc les informations concernant les titres et les éditeurs dans deux tables distinctes. Les informations concernant un éditeur n'ont à être entrées qu'une seule fois pour être ensuite liées à chaque livre. Quand les informations sur un éditeur sont modifiées, la modification n'a à être entrée qu'une seule fois ; de plus, les informations concernant un éditeur sont conservées et cela même si cet éditeur n'a pas de livre dans la base de données.

Il s'agit là d'un principe fondamental de la normalisation des bases de données. Une propriété dépend nécessairement de l'identifiant (toute valeur d'une colonne dépend de la clé primaire). On évite ainsi la redondance d'informations.

- Il faut éviter d'avoir dans les tables des colonnes acceptant des valeurs nulles.

Les tables peuvent comporter des colonnes définies comme tolérant les valeurs nulles. Une valeur nulle indique qu'il n'y a pas de valeur. S'il est parfois utile d'autoriser les valeurs nulles dans certains cas isolés, il est préférable de les utiliser avec parcimonie, dans la mesure où elles requièrent une prise en charge particulière qui augmente la complexité des opérations effectuées sur les données.

Si vous avez plusieurs lignes présentant des valeurs nulles dans des colonnes qui tolèrent ces valeurs, il peut être dès lors préférable de placer ces colonnes dans une autre table liée à la première. Quand le stockage est effectué dans deux tables distinctes, la table primaire reste simple de conception tout en étant capable de répondre aux demandes occasionnelles de stockage de ces informations.

Attention toutefois car cette approche risque de multiplier le nombre de tables de votre base de données.

- Une table ne doit pas comporter de valeurs ni de colonnes répétitives.

La table relative à un élément de la base de données ne doit pas contenir une liste de valeurs pour une information spécifique.

Par exemple, un livre de la base de données pubs ne peut pas être attribué à deux auteurs. S'il existe une colonne dans la table titles pour le nom d'auteur, un problème se pose.

Une solution, à proscrire absolument, consisterait à stocker le nom des deux auteurs dans la colonne, mais dans ce cas il devient difficile d'afficher une liste pour chaque auteur.

Une autre solution consiste à modifier la structure de la table de façon à ajouter une autre colonne pour le nom du second auteur, mais cette solution ne permet d'entrer que deux noms d'auteur. Il faudra donc entrer une colonne supplémentaire si le livre a trois auteurs.

Si vous avez besoin de stocker une liste de valeurs dans une seule colonne, ou si vous avez plusieurs colonnes pour une même donnée (au_lname1, au_lname2, etc.), envisagez de placez les données dupliquées dans une autre table en créant un lien vers la table primaire.

La base de données pubs comporte une table pour les informations concernant les livres et une autre qui ne stocke que les identificateurs des livres et des auteurs. Une telle conception permet d'avoir plusieurs auteurs pour un même livre sans qu'il soit nécessaire de modifier la définition de la table ; elle présente en outre l'avantage de ne pas allouer d'espace de stockage superflu aux livres qui n'ont qu'un seul auteur.

6.1 Mise en place des contrôles d'intégrité des données

L'application de l'intégrité des données garantit la qualité des données stockées dans la base. Par exemple, si un employé est entré avec un matricule « 123 », la base de données ne doit pas autoriser qu'un autre employé ait le même matricule.

Si vous avez une colonne note_employé destinée à accueillir des valeurs comprises dans une fourchette de 1 à 5, la base de données ne doit pas autoriser une valeur de 6.

Si la table contient une colonne réf_service qui stocke le numéro de service de l'employé, la base de données ne doit accepter que des valeurs correspondant bien aux références de service de la société.

Lors de la planification des tables, deux étapes importantes :

- Identifier les valeurs valides pour une colonne
- Décider de la façon d'appliquer l'intégrité des données dans cette colonne.

L'intégrité des données se répartit en quatre catégories :

- Intégrité de l'entité
- Intégrité du domaine
- Intégrité référentielle
- Intégrité définie par l'utilisateur

6.1.1 Intégrité d'entité :

L'intégrité d'entité définit une ligne comme étant une entité unique pour une table particulière. L'intégrité des entités met en application l'intégrité de la ou les colonnes d'identification ou de la clé primaire d'une table : Contrainte PRIMARY KEY, contrainte UNIQUE KEY (sans doublon) ou propriété IDENTITY (Compteur ordinal géré par le système).

6.1.2 Intégrité de domaine

L'intégrité de domaine fait référence à la fourchette (au domaine) des entrées valables pour une colonne donnée. Vous pouvez mettre en application l'intégrité de domaine en restreignant le type (à l'aide des types de données), le format (à l'aide des contraintes CHECK et des règles) ou la fourchette des valeurs possibles (à l'aide des contraintes FOREIGN KEY et CHECK, des définitions DEFAULT et NOT NULL, et des règles).

6.1.3 Intégrité référentielle

L'intégrité référentielle préserve les relations définies entre plusieurs tables lorsque des enregistrements sont entrés, modifiés ou supprimés. L'intégrité référentielle garantit la cohérence des valeurs de clés entre les tables.

Lorsque vous mettez en application l'intégrité référentielle, SQL Server interdit aux utilisateurs de :

- Ajouter des enregistrements à une table connexe lorsqu'il n'y a aucun enregistrement associé dans la table primaire.
- Changer des valeurs dans une table primaire qui engendreraient des enregistrements « orphelins » dans une table connexe.
- Effacer des enregistrements d'une table primaire si des enregistrements liés correspondants existent.

Par exemple, avec les tables sales et titles de la base de données pubs, l'intégrité référentielle est basée sur la relation entre la clé étrangère (title_id) de la table sales et la clé primaire (title_id) de la table titles.

6.1.4 Intégrité définie par l'utilisateur

L'intégrité définie par l'utilisateur vous permet de définir des règles personnelles qui n'entrent dans aucune des autres catégories d'intégrité. Un des exemples souvent usité pourrait être le codepostal qui obéit en France à des règles précises : il est alphanumérique et doit comporter 5 chiffres de 0 à 9. Ainsi, nous aurons 01000 et non 1000.