Programmation Web Dynamique 2

Cours 11 - Optimisation et normalisation des tables

Optimisation

- Une des façons très simples d'optimiser la performance de votre base de données est de réduire au maximum la taille des données et des index dans la base.
  - Plus les colonnes sont petites, plus les lectures sur le disque sont rapides et moins la mémoire centrale sera utilisée.
  - L'indexation de petites colonnes (pour en faire des clés, par exemple) prend beaucoup moins de ressources.

Optimisation (suite)

- Pour ce faire, on :
  - Utilise les types les plus efficaces et les plus petits possibles.
  - Utilisez les types d'entiers les plus petits possibles. MEDIUMINT est souvent préférable à INT.
    - MEDIUMINT varie de -8388608 à 8388608 alors que INT varie de -2147483648 à 2147483648. C'est donc souvent suffisant, et on sauve un octet.
  - Déclarer les colonnes comme étant NOT NULL accèlère les traitements.
  - La clé primaire doit être aussi courte que possible.
  - Ne pas créer trop d'indexes. Ceux-ci accélèrent les requêtes SQL, mais prennent de l'espace disque.

Optimisation des requêtes SELECT

- Pour optimiser les requêtes SELECT, il est important de créer des indexs pour toutes les colonnes qui seront fréquemment filtrées par une clause WHERE.

Normalisation

- La normalisation est un processus de vérification de base de données permettant d'assurer que les informations qui y sont contenues ne seront pas redondantes, et que les mises-à-jour y seront donc aisées.
- Ce processus implique de vérifier que la base de données respecte certaines formes, que l'on nomme les formes normales. Il y en a 8.
- Cependant, plus on va loin dans la normalisation, plus les requêtes permettant d'*obtenir* les infos de la bases sont potentiellement ralenties.
- Généralement, il est accepté sur le marché qu'une base de données respectant les 3 premières formes normales est une base de données que l'on peut dire *normalisée*.
- Nous verrons donc ici les 3 premières formes.

1ère forme normale

- Si toutes les tables de votre modèle :
  - Ne contiennent que des attributs atomiques (c'est-à-dire qu'ils sont indécomposables),
  - Sont constants dans le temps (Ex : on met la date de naissance d'un individu plutôt que son âge),
  - Ne contiennent pas une aggrégation de plusieurs valeurs dans une seule colonne,
  - Alors votre modèle respecte la 1ère forme normale.

2ème forme normale

- Si toutes les tables de votre modèle :
  - Ne contiennent que des colonnes dont la valeur dépend de l'identifiant de votre table
  - Et que la première forme normale est respectée,
  - Votre modèle respecte la seconde forme normale.
- En d'autres mots, les attributs n'appartenant pas à la clé primaire ne dépendent pas d'une partie de la clé.
- Ex de 2ème forme normale non-respectée :
  - Une table commande qui contient l'ID d'un produit (fait partie de l'identifiant), le nom du produit, l'ID du fournisseur (fait aussi partie de l'identifiant) de la commande ainsi que le nom et l'adresse de ce fournisseur, et le nombre d'éléments dans la commande.
    - Ex : Rangée - (56, « Windows 7 », 3, « Microsoft », « 1734 Main, Redmond, VA », 10)
    - Pourquoi? Parce que l'adresse du fournisseur dépend de l'ID du fournisseur et non pas de l'ID du produit. De plus, le nom du produit dépend de l'ID du produit. On devrait donc avoir une table produit qui contient le nom du produit, une table fournisseur qui contient le nom et l'adresse du fournisseur, et une table commandes qui contient l'ID du produit, l'ID du fournisseur et le nombre d'éléments.

3ème forme normale

- Si toutes les tables de votre modèle :
  - Ne contiennent que des colonnes dont la valeur dépend *directement* de l'identifiant de votre table (on dit aussi que tout attribut non clé ne dépend pas fonctionnellement d'un autre attribut non clé)
  - Et que la première et la seconde forme normale sont respectées,
  - Votre modèle respecte la troisième forme normale.
- Ex de 3ème forme normale non-respectée :
  - Une table fournisseurs qui contient l'ID d'un fournisseur, le nom du fournisseur, la ville du fournisseur et le code postal du fournisseur.
    - Parce que le code postal du fournisseur dépend directement de la ville du fournisseur. On devrait donc créer une nouvelle table ville, qui contient le code postal comme attribut, et mettre une référence à la ville dans la table fournisseurs.

Remarques

- Si votre MCD a bien été réalisé, il devrait techniquement mené à un modèle physique DÉJÀ normalisé.
- Les erreurs au MCD amènent donc une base de données non normalisée.
- Le passage à la troisième forme normale réduit considérablement le nombre de mises à jour qui devront être effectuées pour modifier une info dans la BD.
  - Ex : Changer le code postal d'un fournisseur qui se trouve dans une ville dont le code postal a changé nécéssiterait dans le format non normalisé de trouver tous les endroits où l'ancien code postal était utilisé et de modifier ces lignes.
  - Avec le modèle normalisé, il suffit de changer le code postal attribué à la ville du fournisseur, et il sera changé par relation pour tous les fournisseurs situés dans cette ville.