Principes d'une base de données

Les bases de données constituent de nos jours un élément incontournable pour la quasi-totalité des applications. Elles se substituent à l'utilisation de fichiers, trop lourde et trop contraignante pour le développeur. Elles permettent aussi de partager l'information plus simplement, tout en offrant une excellente performance.

L'utilisation de bases de données nécessite la connaissance de deux prérequis : certains éléments de terminologie fréquemment utilisés ainsi que quelques notions de manipulation des données à l'aide du langage SQL.

1. Terminologie

La connaissance des termes décrits ci-dessous est indispensable pour la bonne compréhension de ce chapitre.

Table

Une table est une unité logique de stockage correspondant à un type de donnée. Sa structure est définie par des champs (colonnes), et elle contient des enregistrements (lignes). L'équivalent logique d'une table en C# est une classe.

Enregistrement

Un enregistrement est un élément d'une table possédant une valeur pour chaque colonne de la table. L'équivalent C# d'un enregistrement serait un objet, donc une instance de classe.

Champ

Un enregistrement contient plusieurs champs. Chacun de ces champs représente une donnée composant l'enregistrement. L'équivalent C# d'un champ est une propriété.

Ces trois éléments sont souvent représentés sous la forme d'un tableau, ce qui permet de mieux s'approprier et manipuler ces concepts.

Identifiant	Nom	Prenom	DateNaissance
1	MARTIN	Henry	1954-01-25 00:00:00.000
2	MARTINEZ	Eric	1982-05-04 00:00:00.000
3	MARTIN	Michel	2004-05-28 00:00:00.000

Clé primaire

Une clé primaire est un **identifiant unique** pour chaque enregistrement d'une table. Elle est reconnue par la base de données comme pouvant être utilisée dans une autre table pour faire référence à un enregistrement.

Elle doit être définie explicitement, en général à la création de la table.

2. Le langage SQL

Les développeurs peuvent manipuler les données stockées dans une base de données relationnelle au travers du langage **SQL** (*Structured Query Language*).

Ce langage est fondé sur quatre types d'instructions qui correspondent aux opérations de lecture, d'ajout, de modification et de suppression de données.

Il définit aussi certaines instructions permettant de manipuler la structure de données : création de tables, manipulation d'index ou gestion des relations entre les tables.

L'objet de cet ouvrage n'étant pas la description exhaustive du langage SQL, les instructions de manipulation de la structure de données ne seront pas traitées.

a. Recherche d'enregistrements

La lecture des données est effectuée à l'aide du mot-clé SELECT. Celui-ci, couplé au mot-clé WHERE, permet de filtrer et récupérer des jeux de données issus d'une ou de plusieurs tables de la base.

La requête suivante permet de récupérer l'ensemble des noms et prénoms dans la liste des enregistrements de la table Client :

```
SELECT Nom, Prenom FROM Client
```

Il est aussi possible d'utiliser le symbole * comme raccourci syntaxique pour désigner l'ensemble des champs de la table :

```
SELECT * FROM Client
```

Dans la majorité des cas, il n'est pas pertinent de récupérer l'ensemble des données. Il est alors nécessaire d'utiliser le mot-clé WHERE pour restreindre le nombre d'enregistrements récupérés en fonction d'un ou de plusieurs prédicats.

Clause WHERE

La clause WHERE permet de spécifier une ou plusieurs conditions que les enregistrements doivent respecter pour pouvoir être retournés par la requête SQL.

La requête suivante récupère la liste des clients dont le nom est Dupond.

```
SELECT * FROM Client WHERE Nom = 'Dupond'
```

Pour filtrer à la fois sur le nom et sur le prénom, il faut combiner deux conditions à l'aide des mots-clés AND ou OR.

```
SELECT * FROM Client WHERE Nom = 'Dupond' AND Prenom = 'Michel'
```

Différents autres types de prédicats peuvent être utilisés dans les clauses WHERE pour vérifier notamment qu'une valeur fait partie d'un ensemble ou comparer partiellement des chaînes de caractères.

WHERE ... IN

L'utilisation de la clause WHERE ... IN permet de valider qu'une valeur fait partie d'un ensemble de valeurs spécifiques. Cet ensemble peut être prédéfini ou être le résultat d'une autre requête SELECT.

La liste des clients habitant les villes de Lyon, Toulouse et Nantes peut être récupérée de la manière suivante :

```
SELECT * FROM Client WHERE Ville IN ('Lyon', 'Toulouse, 'Nantes')
```

WHERE ... BETWEEN ... AND

Il est également possible de filtrer les enregistrements en comparant une valeur avec un ensemble spécifié par ses bornes.

La recherche des clients habitant le département du Rhône (69) peut être effectuée à l'aide de la requête suivante :

```
SELECT * FROM Client WHERE CodePostal BETWEEN 69000 AND 69999;
```

WHERE ... LIKE

Dans certains cas, il peut être intéressant de filtrer sur une valeur alphanumérique partielle, notamment pour l'implémentation de scénarios d'autocomplétion. Pour cela, il faut utiliser la clause WHERE ... LIKE.

```
SELECT * FROM Client WHERE Nom LIKE 'A%';
```

Ici, la requête recherche la liste des clients dont le nom commence par un A majuscule. Le caractère **%** dans ce contexte symbolise un ensemble quelconque de caractères.

b. Ajout d'enregistrements

L'ajout d'un enregistrement dans une base de données se fait à l'aide de la commande INSERTINTO. Pour l'exécuter, il faut spécifier la table dans laquelle ajouter les données, lister les champs impactés, et enfin fournir les valeurs à insérer dans ces champs.

La ligne suivante ajoute un enregistrement dans la table Client :

```
INSERT INTO Client (Nom, Prenom) VALUES ('DUPOND', 'Marc');
```

Ne pas spécifier la liste des champs à renseigner équivaut à lister la totalité des champs de la table. Dans ce cas, il faut donc passer une liste de valeurs correspondant à l'ensemble de ces champs. Il est possible de passer la valeur spéciale NULL pour les champs que l'on ne souhaite pas renseigner, à condition que la définition de ces champs autorise cette valeur.

En considérant que la table Client compte trois champs : Nom, Prenom et Adresse, on pourrait réécrire l'exemple précédent de la manière suivante :

```
INSERT INTO Client VALUES (’DUPOND’, ’Marc’, NULL);
```

c. Mise à jour d’informations

La modification d’informations dans la base de données se fait à travers l’utilisation d’une instruction UPDATE. Cette commande peut mettre à jour un ou plusieurs champs d’un ou de plusieurs enregistrements.

Pour cela, il faut indiquer le nom de la table qui doit être mise à jour, suivi du mot-clé SET et de la liste des affectations de valeurs à exécuter.

```
UPDATE Client SET Nom = ’DUPONT’;
```

En l’état, cette requête modifiera le champ Nom pour tous les enregistrements de la table Client, ce qui n’est généralement pas le comportement voulu. Afin de limiter le nombre d’enregistrements sur lesquels doit porter la mise à jour, il est possible d’utiliser une clause WHERE :

```
UPDATE Client SET Nom = ’DUPONT’ WHERE Nom = ’DUPOND’;
```

d. Suppression d’informations

Pour supprimer un ou plusieurs enregistrements d’une table, il est nécessaire d’utiliser la commande DELETE. On l’utilise d’une manière similaire à l’instruction SELECT.

La suppression des clients dont le nom est Dupond est effectuée de la manière suivante :

```
DELETE FROM CLIENT WHERE Nom = ’Dupond’
```