Un livre de Wikilivres.

Programmation SQL

Introduction

SQL est l'acronyme de **S**tructured **Q**uery **L**anguage.

SQL est un langage d'interrogation de bases de données, supporté par la plupart des systèmes de gestion de bases de données relationnelles du marché, parmi lesquels Oracle, IBM DB2, Microsoft SQL Server, Sybase ASE, Microsoft Access, MySQL, PostgreSQL, Borland Interbase, FirebirdSQL, Informix, Ingres, 4eme Dimension (4D), ...

Historique

- 1974 : Création de SQL par IBM
- 1977 : IBM Sequel, première base de donnée utilisant ce système
- 1979 : Lancement d'Oracle SQL RDBMS
- 1986 : Normalisation SQL1 de l'ANSI (également appelée SQL-86)
- 1989 : Extension de la norme SQL1 (également appelée SQL-89)
- 1992 : Normalisation SQL2 de l'ANSI (également appelée SQL-92)
- 1999 : Normalisation SQL3 de l'ANSI (également appelée SQL-99)

Installation d'interfaces

Introduction

Afin de pouvoir manipuler des bases de données SQL, il est conseillé d'avoir un Système de Gestion de Base de Données (SGBD) installé sur votre ordinateur.

Logiciels à télécharger :

Logiciel	Description	Adresse	Licence
MySQL	Le Système de Gestion de Base de Donnée	http://dev.mysql.com/downloads/	La version "Community Server" est sous licence libre GPL
SQLyog MySQL GUI et ~~SQLyog Free Edition 5.02~~	Gestion de la base MySQL La distribution de la version gratuit est arrêtée à la version 5.02. qui ne supporte pas les dernières versions de MySQL.	http://www.webyog.com/	Logiciel propriétaire en version d'essai (30 jours) ou en version bridée
HeidiSQL	Gestion de la base MySQL	http://www.heidisql.com/	Logiciel libre
~~DBDesigner4~~	Modélisation de base de donnée MySQL Le développement de ce logiciel est arrêté à la version 4.0.5.6 Il ne supporte pas les dernières versions de MySQL Ce logiciel est replacé par DBDesigner Fork	http://www.fabforce.net /dbdesigner4/	Logiciel libre sous licence GPL.
DBDesigner Fork	Modélisation de base de donnée MySQL C'est le successeur de DBDesigner4	http://sourceforge.net/projects /dbdesigner-fork/	Logiciel libre sous licence GPL.
Toad for MySQL	Modélisation de base de donnée MySQL	http://www.toadworld.com	Logiciel propriétaire en version gratuite
Oracle	SGBD pour grandes bases de données	http://www.oracle.com /technetwork/database/express-edition/downloads/index.html	Logiciel propriétaire.

(Version Linux)

MySQL Windows

Tout-en-un

Des logiciels tout-en-un (serveur Web, base de donnée MySQL, et PHP) permettent de s'affranchir d'une installation fastidieuse et rédhibitoire pour le débutant :

1. EasyPHP téléchargement (http://www.easyphp.org) [archive] : n'a pas vocation à être installé pour de la production, mais pour le développement. Il stocke les bases de données dans C:\Program Files (x86)\EasyPHP\binaries\mysql\data.
2. WAMP téléchargement (http://www.wampserver.com) [archive] : est du même type qu'EasyPHP : ce logiciel installe facilement un serveur Web Apache, une base de données MySQL et PHP 4 et 5. Il a l'avantage de permettre de passer facilement de PHP 4 à PHP 5, sans avoir à refaire une installation ou une compilation. Tout comme EasyPHP, c'est un environnement de développement, et non un environnement de production. Attention : la résolution des noms d'hôtes se réalise séparément. Les installations WAMP servent à tester en local sur votre PC. Dans la plupart des cas, il suffit d'utiliser le fichier *Hosts* local, comme on le ferait sur une machine Linux, afin de lier des noms aux adresses IP. Dans Windows XP, Vista et 7, ce fichier se trouve dans le répertoire *systemroot\System32\Drivers\Etc*. Il peut se faire que le service ait déjà été configuré. Lorsque vous vous en doutez, contactez votre administrateur réseau. Remarque : vous trouverez une liste des possibilités de résolution de noms avec MS Windows sur Microsoft.com (http://www.microsoft.com/technet/prodtechnol/winxppro/reskit/c24621675.mspx) [archive].
3. XAMPP téléchargement (http://www.apachefriends.org/fr/xampp.html) [archive] : est du même type qu'EasyPHP ou WAMP, le deuxième P étant pour Perl. Son usage est recommandé avec PHPEclipse (http://www.phpeclipse.de/tiki-view_articles.php) [archive].
4. The Uniform Server téléchargement (http://www.uniformserver.com) [archive] : en anglais seulement avec Apache2, Perl5, PHP5, MySQL5, phpMyAdmin.

> **Attention !**
>
> Sur Windows 10 pro le serveur IIS est installé par défaut, et oblige Apache à changer de port (888 au lieu de 80) lors de l'installation. Pour résoudre cela il suffit de décocher *Internet Information Services* dans *Programmes et fonctionnalités, Activer ou désactiver des fonctionnalités Windows. Par ailleurs,* EasyPHP development server *n'y fonctionne pas (*il manque MSVCR110.dll*) mais* EasyPHP hosting server *tourne normalement.*
>
> De même, le port MySQL est susceptible de passer de 3306 à 3388.

Message d'erreur relatif à SSL

Pour l'instant, WAMP ne supporte pas encore le *Secure Socket Layer* (SSL). L'installation se finit par un message qui vous informe de ce fait. Afin de pouvoir travailler sans problèmes, éditez le fichier *c:\windows\php.ini*. Cherchez dans ce fichier la ligne qui commence avec *extension=php_openssl.dll*. Commentez cette ligne en la faisant précéder d'un point-virgule :

```
;extensions=php_openssl.dll
```

Si tout se passe bien, vous pouvez ouvrir la page de test dans votre navigateur.

Installation manuelle

- Apache est disponible sur le site Web de Apache Software Foundation apache.org (http://www.apache.org) [archive].

- PHP est téléchargeable sur le site officiel de php (http://www.php.net) [archive]. Choisissez le fichier au format ZIP.
- Enfin, vous trouverez MySQL sur mysql.com (http://www.mysql.com) [archive].

Installer Apache

Pour installer Apache, double-cliquez sur le fichier exécutable, et suivez les instructions d'installation automatique.

Si vous installez Apache sur un ordinateur de développement, renseignez le champ "nom de domaine" avec la valeur `localhost`.

Si vous installez un serveur de production et que vous disposez d'un nom de domaine, vous devriez disposer des informations nécessaires concernant votre nom de domaine, fournies par le *registrar*.

Une fois l'installation terminée, il faut encore indiquer à Apache qu'il doit fonctionner conjointement avec PHP, car il ne sait pas les traiter par défaut. Pour cela, il faut modifier les informations de configuration d'Apache, contenues dans le fichier `httpd.conf`, qui se trouve dans le dossier d'installation d'Apache, dans le sous-dossier `conf`.









Installer PHP

Une fois l'archive téléchargée, décompressez-la à la racine de votre disque dur et renommez le dossier en `'PHP'`. Dans le dossier `PHP`, vous trouverez deux fichiers: `php.ini-dist` et `php.ini-recommended`. Copiez `php.ini-recommended` dans votre dossier `C:\Windows` ou `C:\winnt` (le nom du dossier dépend de la version de votre système.
renommez-le en `php.ini`.

Ce fichier est le fichier de configuration qui contrôle les options dont vous disposerez.

MySQL

Télécharger et installer le .msi sur http://dev.mysql.com/downloads/gui-tools/5.0.html.

Pour arrêter, démarrer, démarrer automatiquement le serveur MySQL vous devez aller dans la gestion des services (Démarrer/Exécuter/services.msc).

MySQL Linux

LAMP

Logiciel tout-en-un pour Linux (Apache + MySQL + PHP), comme WAMP pour Windows.

commande nécessitant les privilèges root

```
# tasksel install lamp-server
```

Apache sur Debian / Ubuntu

commande nécessitant les privilèges root

```
# apt-get install apache2
```

Le service peut ne pas être lancé par défaut, mais même s'il l'est on peut quand-même essayer de l'activer avec :

commande nécessitant les privilèges root

```
# /etc/init.d/apache2 start
```

On peut ensuite tester le serveur, pour voir si une page s'affiche ou s'il refuse la connexion :

commande

```
$ lynx http://localhost/
```

Cette adresse est la boucle locale, elle peut aussi être rentrée directement dans tout navigateur web.

Si Apache était déjà installé vérifier le fichier pour indiquer le démarrage automatique d'Apache 2 **/etc/default**

/apache2 :

```
# vi /etc/default/apache2
...
NO_START=0
```

PHP

PHP peut-être installé avec toutes les déclinaisons de la distribution Debian (stable, testing, unstable). Il suffit pour cela d'insérer vos lignes préférées dans le fichier */etc/apt/sources.list* :

```
deb http://ftp.fr.debian.org/debian/ stable main non-free contrib
deb-src http://ftp.fr.debian.org/debian/ stable main non-free contrib
```

Ce qui suit suppose que vous ayez déjà installé votre serveur WEB (cf manuels pour l'installation d'Apache sous Debian). Par la suite, exécutez en tant que "root" les commandes suivantes :

```
apt-get update && apt-get install php5
```

Une fois ces commandes exécutées, vous devez redémarrer votre serveur WEB. Dans le cas d'Apache cela s'effectue avec la commande suivante :

```
/etc/init.d/apache restart
```

Si tout s'est bien passé, vous disposez maintenant d'un serveur WEB qui a la capacité d'exécuter des scripts PHP version 5 dans votre navigateur.

Apache sur Gentoo

Premièrement il faut installer Apache si ce n'est pas déjà fait :

```
emerge apache
```

Ensuite, il faut installer PHP :

```
emerge dev-lang/php
```

Puis il faut qu'apache utilise PHP dans sa configuration.

Code: Configuration de apache

```
# nano -w /etc/conf.d/apache2
APACHE2_OPTS="-D PHP5"
```

MySQL seul

MySQL est disponible sur http://dev.mysql.com/downloads/gui-tools/5.0.html au format :

1. .msi (Windows)
2. .dmg (Mac)
3. .rpm (Linux)
4. .tar

En l'absence de gestionnaire de paquets, utiliser le .tar ainsi :

```
shell> groupadd mysql
shell> useradd -r -g mysql mysql
shell> cd /usr/local
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
shell> chown -R root .
shell> chown -R mysql data
shell> bin/mysqld_safe --user=mysql &
```

Sur Gentoo :

```
emerge mysql
```

SQLyog

Dans cet exemple nous avons téléchargé `SQLyog502.exe`, vous pouvez installer la version de votre choix. Ce logiciel est une version d'essais limitée à 30 jours et certaines fonctions sont limitées. (Le logiciel HeidiSQL est son équivalent libre)

Installation Windows

Exécuter SQLyog502.exe.

Configuration

Créer une nouvelle connexion (new)
Lui donner un nom (Save connection)
Mettre le mot de passe (celui de l'utilisateur root)
Enregistrer (Save)
Se connecter (Connect)

Vous pouvez à présent :

- Créer une base
- Créer des tables
- Gérer les utilisateurs
- Exécuter des requêtes
- et beaucoup d'autre choses

DBDesigner4 & DBDesigner Fork

Les deux logiciels DBDesigner4 et son successeur DBDesigner Fork présentent la même interface

Installation

L'installation n'est pas automatisée.

- Télécharger le fichier DBDesignerFork-1.5-bin-i386-win32.zip (http://sourceforge.net/projects/dbdesigner-fork/files/dbdesigner-fork/r1.5/DBDesignerFork-1.5-bin-i386-win32-beta5.zip/download) [archive]
- Dans *C:\Program Files*, créer un répertoire *dbdesigner* et y décomprimer le contenu du fichier ZIP téléchargé.
- Cliquez-droit sur le fichier *DBDesignerFork.exe* et sélectionner "Créer un raccourci sur le bureau""

Avec HeidiSQL ou SQLyog

- Créer un utilisateur MySQL nommé DBDesigner@localhost et avec un mot de passe de votre choix (Représenté dans tous ce qui suit par ******)

```
CREATE USER 'DBDesigner'@'localhost' IDENTIFIED BY '******';
```

- Attribuer à cet utilisateur tous les droits

```
GRANT ALL PRIVILEGES
 ON *.*
 TO 'DBDesigner'@'localhost'
 IDENTIFIED BY '******'
 WITH GRANT OPTION MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR
```

- DBDesigner et DBDesignerFork ne supportent pas le nouveau cryptage de mot de passe. Le mot de passe doit être codé suivant l'ancien format

```
SET PASSWORD FOR DBDesigner@localhost = OLD_PASSWORD('******');
```

Dans DBDesigner

- Ouvrir le formulaire de connexion (menu [Database][Connect to database]) et y créer une nouvelle connexion avec les paramètres suivants
  - Driver: MySQL
  - Host Name : 127.0.0.1 ou LocalHost
  - User: DBDesigner
  - Password: le mot de passe que vous avez choisi

BD

Les bases de données relationnelles

Insuffisance des fichiers

En entreprise, les applications informatiques gèrent des volumes de données très importants : des informations sur les clients, sur les produits, sur les employés, sur la paye, sur la comptabilité, sur les stocks, etc. À l'origine, ces informations étaient stockées sous forme de fichiers. Un fichier dans ce contexte désigne une structure de données qui permet un accès séquentiel ou aléatoire à une suite d'octets. En d'autres termes les opérations que permet un fichier sont la lecture et l'écriture d'octets. Ce que représentent ces octets, le fichier ne l'impose pas; un fichier peut représenter aussi bien une image qu'un extrait sonore.

Rapidement, cette structuration montre ses limites : l'utilisation du fichier nécessite que l'on se préoccupe en permanence de la manière de stocker les données dans celui-ci. Supposons que l'on veuille stocker une liste de clients, chacun avec un numéro unique. Faut-il utiliser une ligne par client ? Comment délimiter le numéro du client et le nom du client de sorte que l'on puisse les extraire du fichier ? Quelle est la taille maximale du nom du client ? Que faire si le nom du client comporte un saut de ligne ?

Le but de la base de données relationnelle est de fournir à l'utilisateur une interface qui fasse abstraction de la manière dont sont stockées ces données. La structure de base est la relation (également appelée table).

Une relation est, pour schématiser, un ensemble de n-uplets de même nature. Le fait de pouvoir structurer les données dans un ensemble offre à l'utilisateur des outils de très haut niveau pour manipuler ces données : l'intersection, l'union, la projection, la sélection. La base de données relationnelle, même si elle va utiliser des fichiers pour le stockage. Plus besoin donc de se souvenir que l'octet 23 du fichier "clients.txt" est le numéro du client. Il suffit de parcourir la table clients et de ne retenir que l'attribut client. Comme les détails du stockage et de la persistance des informations sont pris en charge par la base de données, l'utilisateur peut se concentrer sur des problèmes bien plus fondamentaux, à savoir que faire des informations.

Le langage SQL est le langage qui permet d'interroger une base de données relationnelle pour en extraire et en modifier des informations.

Importance stratégique des bases de données

Notre société moderne est une société où l'information a une valeur marchande considérable : si une entreprise perd la liste de ses clients ou sa comptabilité, les conséquences financières peuvent être catastrophiques. La possibilité de gérer rapidement et automatiquement des commandes, le suivi de celles-ci, les stocks, la comptabilité,... permet à beaucoup d'entreprises d'augmenter considérablement leur compétitivité.

Introduction au modèle clients/serveur

Le modèle de base est le modèle clients/serveur : on installe le **Système de Gestion de Base de Données (SGBD)** sur un serveur et on installe un programme client sur chacune des machines de l'entreprise. Toutes les applications travaillent simultanément sur les données qui sont centralisées sur le serveur. Les applications clientes envoient des requêtes au SGBD (en général dans le langage SQL) et ensuite mettent en forme à l'écran la réponse à ces requêtes. Les applications clientes gèrent l'interaction avec l'utilisateur et le SGBD la manipulation des données.

Des besoins sans limites

Dans beaucoup d'entreprises, les bases de données apparaissent pour gérer une fonctionnalité très spécifique :

par exemple, on veut gérer la liste des commandes et on installe une base de données et une première application interrogeant la base de données. Très rapidement, il apparaît nécessaire d'utiliser une autre application gérant une nouvelle fonctionnalité : par exemple la liste des stocks. Les deux applications ne sont pas indépendantes et vont manipuler les mêmes données. Elles vont accéder à la même base de données. Au fur et à mesure de l'évolution de l'entreprise, le nombre de fonctionnalités informatisées va augmenter jusqu'à parfois englober la totalité du fonctionnement de l'entreprise. Pour permettre une telle évolution, il faut à tout prix que la base de données initiale soit correctement structurée : les tables doivent avoir une organisation très précise. Mieux la base est organisée, plus facile sera l'évolution de celle-ci lors de l'apparition de nouvelles applications. L'**analyse** est la science étudiant la structuration de la base de données. Une étude minutieuse de la base est absolument nécessaire avant le développement de toute application.

L'algèbre relationnelle

Les bases de données relationnelles tiennent leur utilité des caractéristiques suivantes :

1. La base théorique sur laquelle elles reposent, l'algèbre relationnelle, est à la fois relativement simple et éprouvée, ce qui garantit des résultats stables et prévisibles.
2. Le haut niveau d'abstraction qu'offre l'algèbre relationnelle permet de se concentrer sur les propriétés des données, plutôt que la manière d'accéder à celles-ci.

Ces deux caractéristiques permettent en outre aux auteurs de SGBD d'offrir des produits fiables.

Définitions

Quelques définitions

Le SGBD

Le SGBD (Système de Gestion des Bases de Données) est le serveur qui gère la ou les bases de données. Il est administré par un administrateur de bases de données. Celui-ci doit installer la base de données, gérer sa sauvegarde régulière, garantir sa sécurité et gérer les droits des utilisateurs de la base.

Les bases de données

Une base de données relationnelle est constituée d'un ensemble de tables. Une entreprise ne possède en général qu'une seule base de données. Toutefois, un SGBD peut gérer plusieurs bases de données : ceci permet par exemple de garantir que deux applications ne vont absolument pas interagir.

Les tables

Une base de données relationnelle est constituée d'un ensemble de tables. Une table est constituée de lignes et de colonnes. Chaque colonne correspond à un champ de données. Chaque ligne correspond à un enregistrement. Tous les enregistrements possèdent donc les mêmes champs.

Par exemple si vous avez une liste de clients définis par un nom, un prénom et une adresse, vous allez définir une table avec 3 colonnes : NOM, PRENOM et ADRESSE. Chaque client correspondra à une ligne dans la table.

Les colonnes

Une ligne comporte plusieurs colonnes. De manière analogique on peut représenter une table sous la forme d'un tableau : une valeur est située à chaque intersection d'une ligne et d'une colonne. Une colonne correspond à un champ de données. Ce champ possède un nom (par exemple : nom, prénom, adresse, âge), un type (entier, chaîne de caractères, ...), et des options (possibilité de valeur nulle, clé primaire, ...).

Le langage SQL

Les instructions essentielles SQL (*Structured Query Language*) se répartissent en plusieurs familles fonctionnellement distinctes parmi lesquelles : le LDD (Langage de Définition de Données) permet la description de la structure de la base : tables, vues ("views"), index, attributs… Le dictionnaire contient à tout moment le descriptif complet de la structure de données. Le LMD (Langage de Manipulation de Données) permet la manipulation des tables et des vues. Le LCD (Langage de Contrôle des Données) contient les primitives de gestion des transactions et des privilèges d'accès aux données.

Langage de définition de données

On appelle langage de définition de donnée la partie du langage SQL qui permet de créer les différentes bases et les différentes tables.

Langage de manipulation de données

Il s'agit de la partie du langage SQL permettant de modifier ou d'extraire des informations en provenance des différentes tables.

Langage de contrôle de données

Il s'agit de la partie du langage SQL permettant de gérer les droits des différents utilisateurs du SGBD.

Langage de définition de données

Le LDD comprend les commandes pour la définition des données, qui sont CREATE (créer), DROP (supprimer), et ALTER (modifier).

SCHEMA

Instruction CREATE SCHEMA

```
CREATE SCHEMA nom_base;
```

- **Signification :** crée une base de données appelée **nom_base**.
- **Rappel :** une base de données est un ensemble de tables. Un SGBD peut gérer plusieurs bases de données.
- **Exemple :**

```
CREATE SCHEMA `toto` ;
```

Cette commande permet de créer une base de données **toto**.

Sous MySQL on peut employer le terme DATABASE

```
CREATE {DATABASE | SCHEMA} nom_base;
```

Instruction DROP SCHEMA

```
DROP SCHEMA nom_base;
```

- **Signification :** détruit une base de données appelée **nom_base**.
- **remarque :** cette instruction est parfois désactivée pour des raisons de sécurité.
- **Exemple :**

```
DROP SCHEMA `toto` ;
```

Cette commande permet de détruire une base de données **toto**.

Sous MySQL on peut employer le terme DATABASE

```
DROP {DATABASE | SCHEMA} nom_base;
```

TABLE

Instruction CREATE TABLE

- Cette instruction permet de créer une table : il faut définir son nom, les différents champs de la table avec leur type ainsi que des caractéristiques comme par exemple des clés.

- Chaque champs d'une table possède un type :
  - CHAR
  - VARCHAR
  - INTEGER
  - NUMBER
  - DECIMAL
  - FLOAT
  - DOUBLE
  - DATE
  - TIME
  - TIMESTAMP
- **Exemple :**

```
CREATE TABLE client1 ("NOM" VARCHAR( 20 ) ,"PRENOM" VARCHAR( 20 ) ,"ADRESSE" VARCHAR( 200
```

Cette instruction crée une table intitulée **client1** contenant 3 champs :

- Un champ NOM : chaîne d'au plus 20 caractères.
- Un champ PRENOM : chaîne d'au plus 20 caractères.
- Un champ ADRESSE : chaîne d'au plus 200 caractères.

Instruction DROP TABLE

```
DROP TABLE nom_table;
```

- **Signification :** détruit la table appelée **nom_table**.
- **Exemple :**

```
DROP TABLE `client1` ;
```

Cette commande permet de détruire la table **client1**.

INDEX

Instruction CREATE INDEX

```
CREATE [UNIQUE] INDEX nom_de_l_index ON nom_de_table
```

- **Signification :** crée un index appelée **nom_de_l_index** sur la table **nom_de_table**.

Instruction DROP INDEX

```
DROP INDEX nom_de_l_index ON nom_de_table
```

- **Signification :** détruit l'index appelé **nom_de_l_index** de la table **nom_table**.

VIEW

Instruction CREATE VIEW

```
{CREATE | REPLACE} VIEW {nom_de_la_vue} AS
{Instruction SELECT}
WITH READ ONLY;
```

Avec Oracle :

- La création d'une vue nécessite les droits de création de vue.
- L'instruction `WITH READ ONLY` (lecture seule) est facultative.
- Une vue qui comporte certaines expressions (comme `GROUP BY`) est forcément en lecture seule.
- L'`{Instruction SELECT}` de la vue ne comportera pas d'`ORDER BY`.

Voir aussi

- MySQL/LDD, LMD et LCD

Langage de manipulation de données

Instruction SELECT

L'instruction SELECT est la base du LMD, elle permet de renvoyer une table contenant les données correspondantes aux critères qu'elle contient.

Seules les clauses "SELECT" et "FROM" sont obligatoires. La forme générale d'une instruction SELECT (on parle également de *phrase SELECT* ou *requête*) est :

```
SELECT [ALL] | [DISTINCT] * | <liste de champs ou d'instructions d'agrégation>
FROM <liste de tables>
WHERE <condition>
GROUP BY <champs de regroupement>
HAVING <condition>
ORDER BY <champs de tri> [DESC] | [ASC]
```

Une autre forme est

```
SELECT [ALL] | [DISTINCT] * | <liste de champs ou d'instructions d'agrégation>
FROM <table de base>
<liste de jointures>
GROUP BY <champs de regroupement>
HAVING <condition>
ORDER BY <champs de tri> [DESC] | [ASC]
```

Détails des clauses :

SELECT

La clause SELECT permet de spécifier les informations qu'on veut récupérer. Elle contient des champs provenant de tables spécifiées dans la clause FROM, ainsi que des instructions d'agrégation portant sur ces champs.
Le nom des champs ne doit pas être équivoque, ce qui veut dire que si des champs de tables différentes ont le même nom, les champs doivent être préfixés par le nom de la table : `nom_de_table.nom_de_champs`
Les noms des champs sont séparés par des virgules. Si la requête comporte la clause DISTINCT, les doublons (lignes de la table de résultat ayant exactement les mêmes valeurs dans tous les champs) seront éliminés, alors qu'avec la clause ALL, tous les résultats sont renvoyés.
SELECT * permet de renvoyer tous les champs de toutes les tables spécifiées dans FROM.

Instructions d'agrégation

Les instructions d'agrégation permettent des opérations comme le comptage ou les sommes.

Les différentes instructions d'agrégation sont :

- AVG(<champs>)
- COUNT(*)
- MAX (<champs>)
- MIN (<champs>)
- SUM (<champs>)

FROM

Clause obligatoire qui détermine sur quelles tables l'on fait la requête. Les noms des tables sont séparés par des virgules.

Exemple :

```
SELECT nom, prenom
FROM table_president ;
```

Résultat :

nom	prenom
Chirac	Jacques
Kaczynski	Lech
Napolitano	Giorgio
Bachelet	Michelle

Pour voir l'utilisation avec plusieurs tables voir plus bas le chapitre **jointure**.

Si vous voulez récupérer tous les enregistrements de la table, il suffit de remplacer le nom des champs par une "étoile".

Exemple :

```
SELECT *
FROM table_president;
```

Ou

```
SELECT table_president.*
FROM table_president ;
```

Résultat :

genre	nom	prenom	pays
m	Chirac	Jacques	France
m	Kaczynski	Lech	Pologne
m	Napolitano	Giogio	Italie
f	Bachelet	Michelle	Chili

Vous pouvez renommer les champs dans le résultat de la sélection.

Exemple :

```
SELECT nom AS "name", prenom AS "firstname"
FROM table_president ;
```

Résultat :

name	firstname
Chirac	Jacques
Kaczynski	Lech
Napolitano	Giorgio
Bachelet	Michelle

Vous pouvez adapter le résultat à certains critères.

Exemple :

```
SELECT
  CASE genre
    WHEN "m" THEN "Monsieur le président"
    WHEN "f" THEN "Madame la présidente"
    ELSE "Erreur"
  END AS "salutation", nom, prenom
FROM table_president ;
```

Résultat :

salutation	nom	prenom
Monsieur le président	Chirac	Jacques
Monsieur le président	Kaczynski	Lech
Monsieur le président	Napolitano	Giorgio
Madame la présidente	Bachelet	Michelle

Vous pouvez utiliser le "plus" pour concaténer les chaînes de caractères.

Exemple :

```
SELECT prenom + " " + nom + "(" + pays + ")" AS "Nom et pays"
FROM table_president ;
```

Résultat :

Nom et pays
Chirac Jacques (France)

Kaczynski Lech (Pologne)
Napolitano Giorgio (Italie)
Bachelet Michelle (Chili)

Vous pouvez aussi faire des opérations arithmétiques (+, -, *, /) sur les champs.

Exemple :

```
SELECT prix, taux_tva, remise, prix + (prix /100 * taux_tva) - remise AS "total"
FROM table_commande ;
```

Résultat :

prix	taux_tva	remise	total
50	10	7	48
200	25	0	250

Alias

Vous pouvez utiliser un alias, c'est-à-dire un renommage provisoire, pour le nom de la table. Il suit le nom de la table dont il est séparé par une espace. Ceci n'a aucun intérêt à ce stade, mais ça permettra d'abréger la rédaction de vos requêtes quand vous écrirez des conditions, et ce sera très utile pour écrire une auto-jointure. *Exemple :*

```
SELECT *
FROM table_president pres ;
```

Résultat :

genre	nom	prenom	pays
m	Chirac	Jacques	France
m	Kaczynski	Lech	Pologne
m	Napolitano	Giogio	Italie
f	Bachelet	Michelle	Chili

Plusieurs tables

Si vous invoquez plusieurs tables à la fois sans utiliser de clause "WHERE" ni de jointure (un peu de patience, voir ci-dessous), vous obtenez alors une "jointure croisée" (CROSS JOIN) ou "produit cartésien", qui devrait être expliqué plus bas un jour. Les noms des tables sont alors séparés par des virgules. *Exemple :*

```
SELECT *
FROM table_president, table_premier_ministre ;
```

WHERE

Le mot clef "`WHERE`" permet de mettre des conditions à la requête, qui permettent de sélectionner certaines lignes. Les conditions sont des comparaisons entre les champs.

Elément	Description
>	Plus grand que
>=	Plus grand ou égal à
<	Plus petit que
<=	Plus petit ou égal à
=	Égal à
<>	Différent de

Exemple :

```
SELECT *
FROM table_eleve ;
```

Résultat :

nom	age
Paul	8
Jean	7
Jacques	8
Sylvie	9
Steve	8
Julie	7

Pour avoir tous les élèves qui ont 8 ans et plus l'on fait ces requêtes.

```
SELECT *
FROM table_eleve
WHERE age >= 8 ;
```

ou

```
SELECT *
FROM table_eleve
WHERE age > 7 ;
```

Pour avoir les élèves qui ont exactement 8 ans.

```
SELECT *
FROM table_eleve
WHERE age = 8 ;
```

Pour avoir tous les élèves qui n'ont pas 9 ans.

```
SELECT *
FROM table_eleve
WHERE age <> 9 ;
```

Pour avoir les élèves qui se nomment Jean.

```
SELECT *
FROM table_eleve
WHERE nom = "Jean" ;
```

Attention, je dois écrire Jean correctement en faisant attention aux espaces. "Jean " ou "jean" ne fonctionnera pas (espace, minuscule).

Pour la comparaison de chaîne de caractères regarder le mot clef "`LIKE`".

AND et OR

Vous pouvez chaîner des conditions avec les mots clef "`AND`" (et), "`OR`" (ou inclusif, c'est-à-dire que si les 2 conditions sont vraies ensemble, la condition est vérifiée).

Sélectionner les élèves qui ont 7 ou 9 ans et dont le nom commence par J

Exemple :

```
SELECT *
FROM table_eleve
WHERE (age = 7
  OR age = 9)
  AND nom Like "J%" ;
```

Sélectionner les élèves qui ont 7 ans, ou qui ont un nom qui commence la lettre par J, ou les deux.

Exemple :

```
SELECT *
FROM table_eleve
WHERE age = 7 OR nom Like "J%" ;
```

IN et NOT IN

Vous pouvez aussi faire la comparaison avec une liste grâce à "`IN`" et "`NOT IN`".

Sélectionner uniquement les élèves qui ont 7 ans et ceux qui ont 9 ans.

Exemple :

```
SELECT *
FROM table_eleve
WHERE age IN ( 7 , 9 );
```

Résultat :

nom	age
Jean	7
Sylvie	9
Julie	7

Pour avoir les élèves qui ne se nomment pas Jean ou Steve ou Julie.

Exemple :

```
SELECT *
FROM table_eleve
WHERE nom NOT IN ("Jean" , "Steve" , "Julie");
```

Résultat :

nom	age
Paul	8
Jacques	8
Sylvie	9

BETWEEN

Avec "`BETWEEN`" vous pouvez sélectionner des valeurs dans un intervalle.

Pour sélectionner les élèves qui ont entre 6 et 8 ans :

Exemple :

```
SELECT *
FROM table_eleve
WHERE age BETWEEN 6 AND 8 ;
```

Et avec le "`NOT BETWEEN`" vous pouvez sélectionner les élèves qui ne sont pas dans la tranche d'âge 6-8 ans.

Exemple :

```
SELECT *
FROM table_eleve
```

```
WHERE age NOT BETWEEN 6 AND 8 ;
```

LIKE

Condition sur une chaîne de caractères.

- _ : Joker qui remplace exactement un caractère.
- % : Joker qui remplace une suite de caractère.

Critère : "J_l%"
Mots correspondant : Jules, Jel, Julie, Julien, Jolien

Pour sélectionner des courriels qui semblent valides.

Exemple :

```
SELECT *
FROM table_client
WHERE email LIKE "%@%.%" ;
```

IS NULL

`IS NULL` permet de sélectionner les valeurs nulles.

Exemple :

```
SELECT *
FROM table_client
WHERE adresse IS NULL ;
```

Le résultat sera l'ensemble des enregistrements de la table_client où les adresses sont nulles.

`IS NOT NULL` permet de sélectionner uniquement ceux qui n'ont pas la valeur nulle.

Exemple :

```
SELECT *
FROM table_client
WHERE adresse IS NOT NULL ;
```

Cette fois nous avons l'ensemble des enregistrements de la table_client qui ont une adresse.

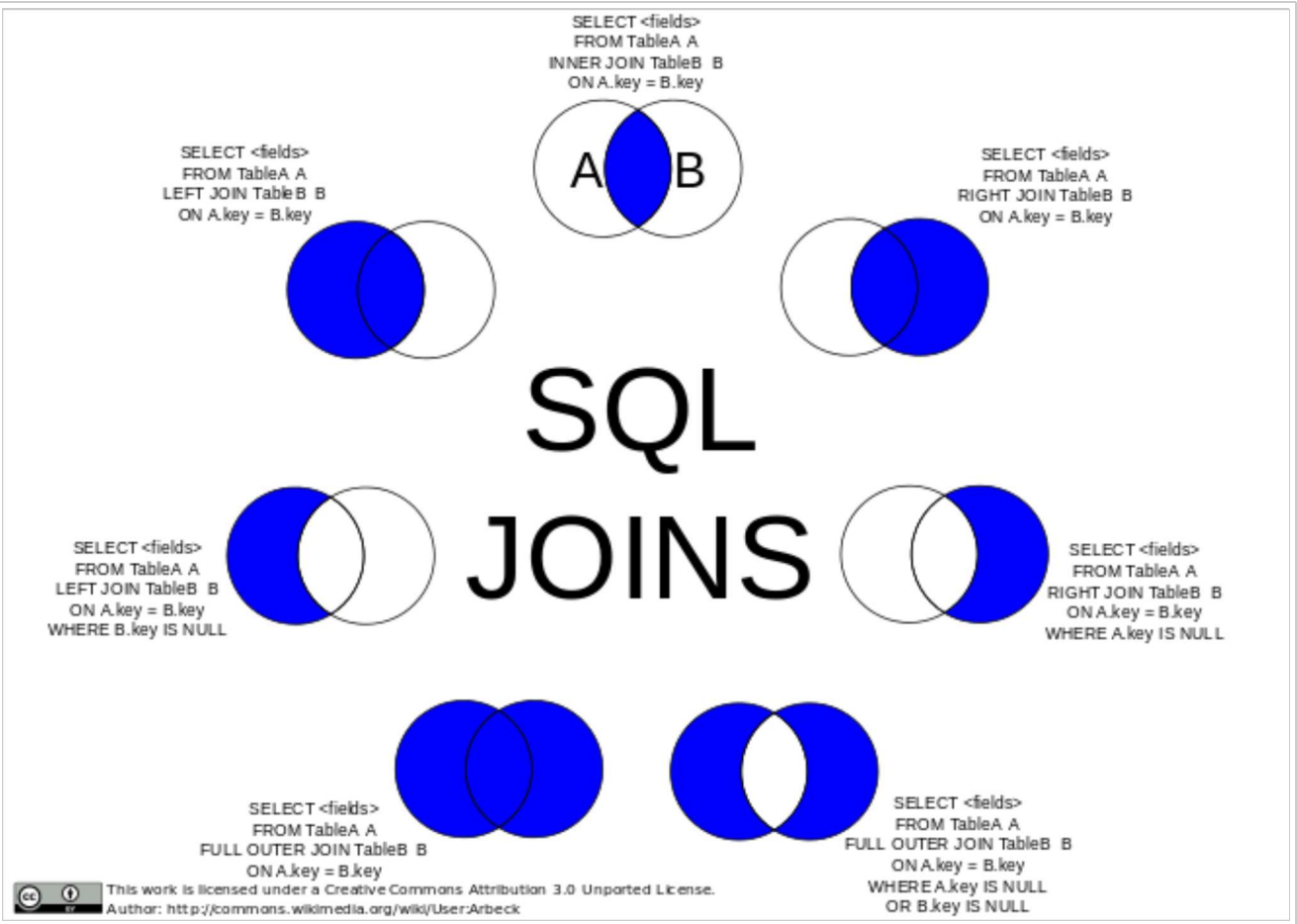
Jointures

Les jointures permettent d'assembler les champs de différentes tables.

Jointure interne

`INNER JOIN`

Exemple



Signification des différentes jointure en SQL

Nous avons une table "eleves" d'élèves :

id_eleve	nom	age
1	Paul	8
2	Jean	7
3	Jacques	8
4	Sylvie	9
5	Steve	8
6	Julie	7

une table "branches" de branches :

id_branche	nom
1	Francais
2	Histoire
3	Math

et une table "notes" de notes :

id_note	id_eleve	id_branche	note
1	1	1	8
2	2	1	10
3	4	1	10
4	5	1	8
5	6	1	4

Pour afficher chaque note avec l'élève et la branche correspondante, je dois faire une jointure.

```
SELECT notes.id_note, eleves.nom as eleve, branches.nom as branche, notes.note
FROM eleves, branches, notes
WHERE eleves.id_eleve = notes.id_eleve
  AND branches.id_branche = notes.id_branche ;
```

ou

```
SELECT notes.id_note, eleves.nom as eleve, branches.nom as branche, notes.note
FROM notes INNER JOIN eleves ON notes.id_eleve = eleves.id_eleve
  INNER JOIN branches ON notes.id_branche = branches.id_branche ;
```

id_note	eleve	branche	note
1	Paul	Francais	8
2	Jean	Francais	10
3	Sylvie	Francais	10
4	Stève	Francais	8
5	Julie	Francais	4

Jointure externe

Dans le cas ci-dessus nous ne pouvons pas voir l'élève Jacques (3) qui n'a pas participé à l'examen de français. La jointure externe permet de ne pas perdre d'information en conservant soit toutes les lignes de la table de gauche (LEFT OUTER JOIN) soit à droite.

LEFT OUTER JOIN

Ici, en conservant toutes les lignes de la table eleves :

```
SELECT notes.id_note, eleves.nom as eleve, branches.nom as branche, notes.note
FROM eleves LEFT OUTER JOIN notes ON notes.id_eleve = eleves.id_eleve
  INNER JOIN branches ON notes.id_branche = branches.id_ branche ;
```

RIGHT OUTER JOIN

Pour conserver toutes les lignes des deux tables. ***FULL OUTER JOIN***

Auto jointures

ORDER BY

Cette clause permet d'ordonner les résultats d'une requête. On peut les ordonner de deux manières : ascendante (ou dans l'ordre croissant) avec le mot clé ASC ou descendant (décroissant) avec le mot clé DESC.

Reprenons notre modèle logique :

élèves(**id_élève**, nom_élève, prénom_élève, adresse_élève, cp_élève, ville_élève, **# id_classe**)

Si l'on désire afficher les élèves par ordre alphabétique croissant sur le nom, on fait :

```
SELECT *
FROM élèves
ORDER BY nom_élève ASC;
```

La requête ci-dessous est équivalente :

```
SELECT *
FROM élèves
ORDER BY nom_élève
```

La plupart des SGBD trient par défaut de manière croissante.

Nous pouvons préciser le tri. Par exemple nous voulons trier sur le nom, puis le prénom puis l'adresse de manière décroissante.

```
SELECT *
FROM élèves
ORDER BY (nom_élève, prénom_élève, adresse_élève) DESC;
```

GROUP BY

L'instruction GROUP BY en sql permet de grouper un ensemble d'enregistrements issus d'une requête, de façon à ce que chaque groupe ainsi créé ne soit représenté que par une seule ligne.

Prenons par exemple le modèle logique suivant à 3 tables :

élèves(**id_élève**, nom_élève, prénom_élève, adresse_éléve, cp_élève, ville_élève, **# id_classe**)

Classes(**id_classe**, nom_classe, **# id_section**)

section(**id_section**, nom_section)

Lorsque l'on veut afficher la classe, la section de chaque élève on procède comme suit :

```
SELECT
  nom_élève    AS "nom",
```

```
  prenom_élève AS "prenom",
  nom_classe   AS "classe",
  nom_section  AS "section"
FROM
  élèves  e,
  classes c,
  section s
WHERE
  e.id_classe=c.id_classe   AND
  c.id_section=s.id_section
```

Si nous voulons obtenir les effectifs de chaque classe nous faisons :

```
SELECT id_classe, Count(*)
FROM élèves
GROUP BY id_classe
```

Remarque :

- La présence de la clause GROUP BY est nécessaire dès que la requête comporte simultanément dans la clause de sélection (SELECT) le filtre (WHERE), des jointures, des colonnes de tables hors calcul et des calculs d'agrégation.
- Dans ce cas, toutes les colonnes représentées hors des calculs d'agrégation doivent figurer dans la clause GROUP BY.

HAVING

Cette clause est liée à la clause GROUP BY. Elle permet de préciser une sélection. Lorsque l'on applique GROUP BY, on effectue une réunion. HAVING va nous permettre d'émettre une condition sur cette réunion. La clause HAVING est largement utilisée avec les fonctions d'agrégation.

Reprenons par exemple le modèle logique de GROUP BY ci-dessus. Si cette fois nous voulons filtrer les effectifs et n'afficher que les classes qui ont un effectif inférieur ou égal à 20 élèves, nous faisons :

```
SELECT id_classe, Count(*)
FROM élèves
GROUP BY id_classe
HAVING Count(*) <= 20
```

Attention la clause HAVING doit impérativement précéder la clause ORDER BY.

Exemple :

```
SELECT id_classe AS "identifiant", nom_classe AS "nom", Count(*) AS "effectif"
FROM classes c, élèves e
WHERE e.id_classe=c.id_classe
GROUP BY (identifiant, nom)
HAVING Count(*) BETWEEN 15 AND 30
ORDER BY nom ASC
```

Instruction INSERT

Cette requête sert à insérer des enregistrements dans les tables d'une base de données.

La syntaxe de la commande est la suivante (cas le plus simple, la table étant déjà créée) :

INSERT INTO nom_table VALUES ("Paul","Jean","Isabelle","Marie");

Dans un cas plus général, reprenons notre exemple :

élèves(**id_élève**, nom_élève, prénom_élève, adresse_éléve, cp_élève, ville_élève, **# id_classe**)

Pour ajouter un enregistrement à cette table, nous faisons :

```
INSERT INTO élèves (id_élève, nom_élève, prénom_élève, adresse_éléve, cp_élève, ville_é
  VALUES ('1','brindzingue','toto','trifoullie les oyes','66000','un coin perdu','12');
```

Remarque : Si l'on ajoute tous les champs de l'enregistrement, il n'est pas utile de préciser le nom des champs (ex: nom_élèves, ville_élève, ...), il suffit de saisir, dans l'ordre des champs de la base de données (de la gauche vers la droite) les valeurs associées aux enregistrements.

L'exemple suivant est donc equivalent à l'exemple précédent :

```
INSERT INTO élèves
  VALUES ('1','brindzingue','toto','trifoullie les oyes','66000','un coin perdu','12');
```

Pour n'ajouter que quelques données (données obligatoires par exemple, c'est-à-dire, celles qui ne doivent pas être nulles dans un enregistrement) on peut opèrer comme suit :

En imaginant que les données adresse_éléve, cp_élève et ville_élève soient optionnelles dans la base de données :

```
INSERT INTO élèves (id_élève, nom_élève, prénom_élève, id_classe)
  VALUES ('1','brindzingue','toto','12');
```

est une requête valide.

Instruction UPDATE

Cette requête sert à mettre à jour des enregistrements dans les tables d'une base de données. La syntaxe de la commande est la suivante :

```
UPDATE <nom_de_la_table> SET
  (<colonne_1>=<nouvelle_valeur> , <colonne_2>=<nouvelle_valeur>, …)
[ WHERE <condition> ];
```

Reprenons notre exemple :

élèves(id_élève, nom_élève, prénom_élève, adresse_éléve, cp_élève, ville_élève, # id_classe)

Actuellement la table ressemble à cela :

id_élève	nom_élève	prénom_élève	adresse_éléve	cp_élève	ville_élève	# id_classe
1	brindzingue	toto	trifoullie les oyes	66000	un coin perdu	12

Pour mettre à jour un enregistrement dans cette table, nous faisons :

```
UPDATE élèves SET (nom_élève='Brindzyngue')
WHERE id_élève='1';
```

Après éxécution de la requête, la table est équivalente à :

id_élève	nom_élève	prénom_élève	adresse_éléve	cp_élève	ville_élève	# id_classe
1	Brindzyngue	toto	trifoullie les oyes	66000	un coin perdu	12

Attention, l'absence de condition peut entrainer certains problèmes (modification des champs de tous les enregistrements par exemple) qui ne provoque pas d'erreur SQL mais qui falsifie toutes les données de la base (les anciennes "bonnes" données risque fort d'être définitivement perdues !!!).

Instruction DELETE

Cette instruction sert à effacer un enregistrement d'une table de la base de données. La syntaxe est la suivante :

```
DELETE FROM <nom_de_la_table>
[ WHERE <condition> ] ;
```

Reprenons notre exemple : **élèves(** **id_élève**, nom_élève, prénom_élève, adresse_éléve, cp_élève, ville_élève, **# id_classe**)

Actuellement la table ressemble à cela :

id_élève	nom_élève	prénom_élève	adresse_éléve	cp_élève	ville_élève	# id_classe
1	brindzingue	toto	trifoullie les oyes	66000	un coin perdu	12

Nous allons supprimer l'enregistrement en faisant :

```
DELETE FROM élèves
WHERE id_élève='1' ;
```

La table est désormais vide.

Attention à la condition. Si elle n'est pas présente, la requête s'applique à tous les enregistrements de la table. En d'autres termes, vous videz la table.

Langage de contrôle de données

Le LCD consiste à gérer les permissions et annulations des transactions, selon certaines utilisateurs, certaines requêtes ou la reprise sur panne.

GRANT

La commande GRANT est utilisée dans le menu "Privilèges" de PhpMyAdmin.

La commande GRANT permet d'attribuer un ou plusieurs privilèges à un utilisateur ou un rôle sur des instructions (CREATE DATABASE, CREATE TABLE, CREATE VIEW, etc.) ou divers objets tels que des tables, des vues, des colonnes ou encore des procédures stockées.

```
CREATE USER 'NomDelUtilisateur'@'localhost' IDENTIFIED BY '***';
GRANT SELECT ,
INSERT ,
UPDATE ;
```

Exemple MySQL :

```
GRANT ALL PRIVILEGES ON `Base1`.* TO 'NomDelUtilisateur'@'localhost' WITH GRANT OPTION;
```

DENY

Empêche certains utilisateurs d'effectuer une certaine opération.

```
DENY ALL
ON Base1.*
TO 'NomDelUtilisateur'@'localhost';
```

NB : cette commande existe en Transact-SQL[1] mais pas en MySQL.

REVOKE

La commande REVOKE permet de révoquer des privilèges qui ont été attribués à un utilisateur ou à un rôle sur des instructions ou divers objets tels que des tables, des vues, des colonnes ou encore des procédures stockées.

```
REVOKE ALL PRIVILEGES ON `Base1`.* FROM 'NomDelUtilisateur'@'localhost';
```

UPDATE

Autorise un utilisateur à mettre à jour des enregistrements.

READ

Empêche un utilisateur de modifier la base de données, il peut juste lire les données.

DELETE

Autorise un utilisateur à effacer des enregistrements.

COMMIT

Valide une transaction en cours dans la base.

ROLLBACK

Annule une transaction en cours.

Références

1. https://msdn.microsoft.com/fr-fr/library/ms188338.aspx?f=255&MSPPError=-2147217396

Exemples

Exemple d'applications

Une bibliothèque d'images

Nous allons construire une bibliothèque d'images qui permet à plusieurs utilisateurs de partager des images et de les annoter en associant des à chaque image, un ou plusieurs mot-clés. Le dialecte SQL utilisé dans cette exemple est celui de PostgreSQL (http://www.postgresql.org) [archive], pour des raisons de simplicité. Les types de données varient d'un SGBD à l'autre mais les concepts resteront les mêmes.

Modèle de données

Créons tout d'abord une représentation d'un utilisateur.

```
create table utilisateur
(
   sid serial primary key not null,
   prenom varchar(256) not null,
   nom varchar(256) not null,
   email varchar(256) not null,
   creation timestamp not null,
);
```

`sid` est un identifiant dont la particularité est d'être unique et strictement monotone grâce au fait qu'il est basé sur une `séquence` qui possède les mêmes propriétés. Une séquence est une fonction strictement monotone qui renvoie une liste d'entiers distincts à chaque fois qu'elle est appelée. `sid` se prête donc particulièrement à une utilisation comme clé primaire. Utiliser des clés primaires arbitraires, c'est-à-dire créer un identifiant spécifiquement à cet usage, plutôt que d'utiliser des attributs uniques existants (comme dans ce cas, l'adresse email) facilite les jointures.

Pour s'assurer de l'unicité de l'adresse email dans cette table, ajoutons une contrainte d'unicité sur l'attribut `email`

```
alter table utilisateur add constraint email_unique unique(email);
```

Créons à présent la table qui comportera toutes les images. Comme nous construisons une bibliothèque qui pourra contenir un grand nombre d'images, nous stockerons ces images en dehors du SGBD et ne ferons figurer que le chemin qui mène au fichier qui contient l'image. Il est possible de stocker les images elles-mêmes dans la base mais cela conduira à limiter la capacité du SGBD, qui, sous une charge importante, passera un certain temps à extraire les images de ses tables et limitera le trafic maximum.

```
create table image
(
   sid serial primary key not null,
   nom varchar(256) not null,
   chemin varchar(256) not null,
   creation timestamp not null
);
```

Associons maintenant images et utilisateurs.

```
create table image_utilisateur
(
   sid serial primary key not null,
   image_sid integer not null,
   utilisateur_sid integer not null
);
```

Pour être sûr que `image_sid` et `utilisateur_sid` pointent vers des objets existants, ajoutons 2 contraintes d'intégrité référentielle.

```
alter table image_utilisateur add constraint image_utilisateur_utilisateur_fk
  foreign key (utilisateur_sid) references utilisateur(sid);

alter table image_utilisateur add constraint image_utilisateur_image_fk
  foreign key (image_sid) references image(sid);
```

Ajoutons des mots-clés :

```
create table mot_cle
(
   sid serial primary key not null,
   nom varchar(256) not null,
   creation timestamp not null
);
```

Et associons-les aux images :

```
create table image_mot_cle
(
   sid serial primary key not null,
   image_sid integer not null,
   mot_cle_sid integer not null
);
```

Même traitement pour éviter les mots-clés inexistants attachés à des images inexistantes :

```
alter table image_mot_cle add constraint image_mot_cle_mot_cle_fk
 foreign key (mot_cle_sid) references mot_cle(sid);

alter table image_mot_cle add constraint image_mot_cle_image_fk
 foreign key (image_sid) references image(sid);
```

Dans cet exemple, nous omettons de créer des index puisque ceux-ci, quoique qu'indispensables pour obtenir une performance correcte lorsque la cardinalité des relations dépassent la dizaine ou la centaine, ne modifient en rien le résultat.

Exemples de requête

Nous avons besoin de données pour ces exemples.

```
insert into utilisateur (prenom, nom, email, creation) values ('Dupont', 'Dupond', 'dupo
insert into utilisateur (prenom, nom, email, creation) values ('Durant', 'Durand', 'dura
insert into image (chemin, nom, creation) values ('/mes/images/', 'image1.jpg', now());
```

```
insert into image (chemin, nom, creation) values ('/mes/images/', 'image2.jpg', now());
insert into image (chemin, nom, creation) values ('/mes/images/', 'image3.jpg', now());
insert into image_utilisateur (image_sid, utilisateur_sid) values (1, 1);
insert into image_utilisateur (image_sid, utilisateur_sid) values (3, 1);
insert into image_utilisateur (image_sid, utilisateur_sid) values (3, 2);
insert into image_utilisateur (image_sid, utilisateur_sid) values (2, 2);
insert into mot_cle (nom, creation) values ('wiki', now());
insert into mot_cle (nom, creation) values ('populaire', now());
insert into image_mot_cle (image_sid, mot_cle_sid) values (3, 1);
```

Combien d'images appartiennent-elles à l'utilisateur dont l'email est "dupont@dupond.fr" ?

```
select count(*)
   from utilisateur u
   join image_utilisateur iu on (u.sid = iu.utilisateur_sid)
 where u.email = 'dupont@dupond.fr';
```

Quelles sont les 10 images les plus partagées ?

```
select i.chemin, i.nom, count(*) as partage
  from image i
  join image_utilisateur ui on (i.sid = ui.image_sid)
 group by i.chemin, i.nom
 order by count(*) desc
 limit 10;
```

Quel est l'utilisateur avec le plus d'images ?

```
select u.prenom, u.nom
  from utilisateur u
   join image_utilisateur iu on (u.sid = iu.utilisateur_sid)
 group by u.prenom, u.nom
 order by count(*) desc
 limit 1;
```

Quelles sont les images qui n'appartiennent à personne ?

```
select i.chemin, i.nom
  from image i
  left outer join image_utilisateur iu on (i.sid = iu.image_sid)
 where iu.utilisateur_sid is null;
```

Ajoutons le mot-clé "wiki" à toutes les images de "dupont@dupond.fr".

```
insert into image_mot_cle (image_sid, mot_cle_sid)
select iu.image_sid, mc2.sid
 from utilisateur u
 join image_utilisateur iu on (u.sid = iu.utilisateur_sid)
 left outer join image_mot_cle imc on (iu.image_sid = imc.image_sid)
 left outer join mot_cle mc on (imc.mot_cle_sid = mc.sid),
      mot_cle mc2
where u.email = 'dupont@dupond.fr'
  and (imc.sid is null or mc.sid is null or mc.nom <> 'wiki')
  and mc2.nom = 'wiki'
group by mc2.sid, iu.image_sid;
```

Dans cette dernière requête, nous voulons éviter les images qui sont déjà associées au mot-clé "wiki", tout en conservant les images sans mot-clé ou associées à d'autres mot-clés, le tout pour l'utilisateur en question.

Comme vous pouvez le voir, SQL permet d'exprimer de manière succincte des conditions relativement précises.

Mots réservés

Mots usuels

Commande	Fonction
select	Mode sélection
from	Choisit les tables
where	Filtre le résultat
join	Joint des tables
natural join	Jointure naturelle
inner join	Jointure interne
full outer join	Jointure externe
left join	Jointure à gauche
right join	Jointure à droite
union	union
intersect	intersection
as	Renomme les colonnes du résultat
distinct	Filtre les doubles
between... and	Filtre dans une plage de nombre
in, any	Applique à certaines valeurs
all	Applique à toutes les valeurs
exists	Applique aux valeurs qui existent
like... %	Désigne une chaîne de caractère
like... _	Représente un caractère non vide
not	Exclut
having	Inclut
group by	Groupe dans le résultat
order by	Trie le résultat
top	Limite le résultat en dessous d'une ligne
limit... offset	Limite le résultat dans une plage de lignes
count	Compte le résultat
sum	somme
avg	moyenne
min	minimum
max	maximum
Modifications de la base	
create database	Crée une base de données
create table	Crée une table
alter table	Modifie la structure d'une table (ajout de colonne...)
drop table	Supprime une table
drop database	Supprime une base
update... set	Met à jour des enregistrements

Commande	Fonction
insert into... values	Insère des enregistrements
delete	Efface des enregistrements
on delete cascade	Supprimer en cascade
primary key	clé primaire
foreign key... references	clé étrangère

En 1992 ont été ajoutés : DATE, TIME, TIMESTAMP, INTERVAL, BIT, VARCHAR, CAST.

En 2003 : CREATE TABLE AS et CREATE TABLE LIKE. BIT a été retiré.

En 2008 : MERGE, DIAGNOSTIC, TRUNCATE TABLE, INSTEAD OF.

Voir aussi

- Mots réservés en SQL 92 (http://www-igm.univ-mlv.fr/~fpetit/download/BD1/cours/SQL-mots-reserves.pdf) [archive]
- Vérificateur de mot réservé (http://www.petefreitag.com/tools/sql_reserved_words_checker/) [archive]

Récupérée de « https://fr.wikibooks.org/w/index.php?title=Programmation_SQL/Version_imprimable&oldid=504468 »