Vue d’ensemble de SQL

1. SQL une norme

Notons que SQL sera normalisé à quatre reprises : 1986 (SQL 86 - ANSI), 1989 (ISO et ANSI) et 1992 (SQL 2 - ISO et ANSI) et enfin 1999 (SQL:1999 - ISO) souvent apellé à tort SQL 3. A ce jour, aucun SGBD n'a implémenté la totalité des spécifications de la norme actuellement en vigueur. Mais je dois dire que le simple (?) SELECT est argumenté de quelques 300 pages de spécifications syntaxiques dans le dernier document normatif...

Néanmoins, la version SQL 2 (1992) est la version vers laquelle toutes les implémentations tendent. C'est pourquoi nous nous baserons sur SQL 2.

A noter, la future norme SQL:2003, sortira en juillet 2003.

2. SQL un standard

Même si SQL est considéré comme le standard de toutes les bases de données relationnelles et commercialisées, il n'en reste pas moins vrai que chaque éditeur tend à développer son propre dialecte, c'est à dire à rajouter des éléments hors de la norme. Soit fonctionnellement identiques mais de syntaxe différentes (le LIKE d'Access en est l'exemple le plus parlant !) soit fonctionnellement nouveau (le CONNECT BY d'Oracle pour la récursivité par exemple). Il est alors très difficile de porter une base de données SQL d'un serveur à l'autre. C'est un moindre mal si l'on a respecté au maximum la norme, mais il est de notoriété absolue que lorsque l'élément normatif est présent dans le SGBD avec un élément spécifique ce sera toujours l'élément spécifique qui sera proposé et documenté au détriment de la norme !

Exemple, la fonction CURRENT_TIMESTAMP, bien présente dans MS SQL Server, n'est pas spécifié dans la liste des fonctions temporelle de l'aide en ligne !!!

De plus, le plus petit dénominateur commun entres les 4 poids lourds de l'édition que sont Oracle, Sybase (ASE), Microsoft (SQL Server) et IBM (DB2) est tel qu'il est franchement impossible de réaliser la moindre base de données compatible entre ces différentes implémentations, sauf à n'y stocker que des données numériques !!!

Pour chaque SGBD, on parle alors de dialecte. Il y a donc le dialecte SQL d'Oracle, le dialecte SQL de Sybase...etc.

Dans un excellent article, Peter Gulutzan fait le point sur l'essentiel de ce qui est normatif chez les principaux éditeurs de SGBD : http://www.dbazine.com/gulutzan3.html
Et fustige leur comportement, similaire à celui des constructeurs d'ordinateurs en matière d'OS UNIX incompatible et et qui maintenant pleurent à chaudes larmes devant le rouleau compresseur "Linux" !
Une autre étude, par Michael M. Gorman montre que la volonté de ces éditeurs de SGBD et leur intérêts mercantiles à courte vue est incompatible avec le respect des standards...
A lire donc : http://www.tdan.com/i016hy01.htm

Nous avons aussi démontré qu'une même table créée sur différents SGBD avec les mêmes données n'avait par forcément le même comportement au regard du simple SELECT du fait des jeux de caractères et collations par défaut. Or tous les SGBD ne sont pas paramétrables à ce niveau et ceux qui le sont, ne présentent pas en général les mêmes offres en cette matière.
A lire : Une question de caractères...

Enfin, pour tous ceux qui veulent connaître la norme comparée au dialecte de leur SGBD favori, il suffit de lire le tableau de comparaison des fonctions de SQL : Toutes les fonctions de SQL

3. Remarques préliminaires sur les SGBD

Avant tout, nous supposerons connues les notions de "bases de données", "table", "colonne", "ligne", "clef", "index", "intégrité référentielle" et "transaction", même si ces termes, et les mécanismes qu'ils induisent seront plus amplement décrits au fur et à mesure de votre lecture.

Voici quelques points qu'il convient d'avoir à l'esprit lorsque l'on travaille sur des bases de données :

- Il existe une grande **indépendance** entre la couche abstraite que constitue le SGBD et la couche physique que sont le ou les fichiers constituant une base de données. En l'occurrence il est déraisonnable de croire que les fichiers sont arrangés dans l'ordre des colonnes lors de la création des tables et que les données sont rangées dans l'ordre de leur insertion ou de la valeur de la clef.
- Il n'existe **pas d'ordre spécifique** pour les tables dans une base ou pour les colonnes dans une table, même si le SGBD en donne l'apparence en renvoyant assez généralement l'ordre établi lors de la création (notamment pour les colonnes). Par conséquent les tables, colonnes, index... doivent être repérés par leur nom et uniquement par cet identifiant.
- Les **données sont systématiquement présentées sous forme de tables**, et cela quel que soit le résultat attendu. Pour autant la table constituée par la réponse n'est pas forcément une table persistante, ce qui signifie que si vous voulez conserver les données d'une requête pour en faire usage ultérieurement, il faudra créer un objet dans la base (table ou vue) afin d'y placer les données extraites.
- La logique sous-jacente aux bases de données repose sur l'algèbre relationnel lui-même basé sur la théorie des ensembles. Il convient donc de penser en terme d'ensemble et de logique et non en terme d'opération de nature atomique. *A ce sujet, il est bon de se rappeler l'usage des patates (ou diagrammes de Wen) lorsque l'on "sèche" sur une requête.*
- Du fait de l'existence d'optimiseurs, la manière d'écrire une requête à peu d'influence en général sur la qualité de son exécution. Dans un premier temps il vaut mieux se consacrer à la résolution du problème que d'essayer de savoir si la clause "bidule" est plus gourmande en ressource lors de son exécution que la clause "truc". Néanmoins l'optimisation de l'écriture des requêtes sera abordée dans un article de la série.

4. A propos des SGBD

Il existe à ce jour, deux types courant d'implémentation physique des SGBD relationnels. Ceux qui utilisent un service de fichiers associés à un protocole de réseau afin d'accéder aux données et ceux qui utilisent une application centralisée dite serveur de données. Nous les appellerons SGBD "fichier" et SGBD client/serveur (ou C/S en abrégé).

4.1. SGBD "fichier"

Le service est très simple à réaliser : il s'agit de placer dans une unité de stockage partagée (en général un disque d'un serveur de réseau) un ou plusieurs fichiers partageables. Un programme présent sur chaque poste de travail assure l'interface pour traiter les ordres SQL ainsi que le va et vient des fichiers de données sur le réseau.
Il convient de préférer des SGBD à forte granularité au niveau des fichiers. En effet plus il y a de fichiers pour une même base de données et moins la requête encombrera le réseau, puisque seuls les fichiers nécessaires à la requête seront véhiculés sur le réseau.

Ces SGBD "fichier" ne proposent en général pas le contrôle des transactions, et peu fréquemment le DDL et le DCL. Ils sont, par conséquent, généralement peu ACID !

Les plus connus sont ceux qui se reposent sur le modèle XBase. Citons parmi les principaux SGBD "fichier" : dBase, Paradox, Foxpro, BTrieve, MySQL, … Généralement basés sur le modèle ISAM de fichiers séquentiels indexés.

Avantage : simplicité du fonctionnement, coût peu élevé voire gratuit, format des fichiers ouverts, administration quasi inexistante.

Inconvénient : faible capacité de stockage (quoique certains, comme Paradox, acceptent 2 Go de données par table !!!), encombrement du réseau, rarement de gestion des transactions, faible nombre d'utilisateurs, faible robustesse, cohérence des données moindre.

Access se distingue du lot en étant assez proche d'un serveur SQL : pour une base de données, un seul fichier et un TCL. Mais cela présente plus d'inconvénients que d'avantages : en effet pour interroger une petite table de quelques enregistrements au sein de base de données de 500 Mo, il faut rapporter sur le poste client, la totalité du fichier de la base de données... Un non sens absolu, que Microsoft pali en intimant à ses utilisateurs de passer à SQL Server dès que le nombre d'utilisateurs dépasse 10 !

4.2. SGBD "Client/Serveur"

Le service consiste à faire tourner sur un serveur physique, un moteur qui assure une relative indépendance entre les données et les demandes de traitement de l'information venant des différentes applications : un poste client envoie à l'aide d'un protocole de réseau, un ordre SQL (une série de trames réseau), qui est exécuté, le moteur renvoie les données. De plus le SGBD assure des fonctions de gestions d'utilisateurs de manière indépendante aux droits gérés par l'OS.

A ce niveau il convient de préférer des SGBD C/S qui pratiquent : le verrouillage d'enregistrement plutôt que le verrouillage de page (évitez donc SQL Server...), et ceux qui tournent sur de nombreuses plates-formes système (Oracle, Sybase...). Enfin certains SGBD sont livrés avec des outils de sauvegarde et restauration.

Les SGBD "C/S" proposent en général la totalité des services du SQL (contrôle des transactions, DDL et DCL). Ils sont, par conséquent, pratiquement tous ACID. Enfin de plus en plus de SGBD orientés objets voient le jour. Dans ce dernier cas, ils intègrent la plupart du temps le SQL en plus d'un langage spécifique d'interrogation basé sur le concept objet (O^2, ObjectStore, Objectivity, Ontos, Poet, Versant, ORION,GEMSTONE...)
Les serveurs SQL C/S les plus connus sont : Oracle, Sybase, Informix, DB2, SQL Server, Ingres, InterBase, SQL Base...

Avantage : grande capacité de stockage, gestion de la concurrence dans un SI à grand nombre d'utilisateurs, haut niveau de paramétrage, meilleure répartition de la charge du système, indépendance vis à vis de l'OS, gestion des transactions, robustesse, cohérence des données importante. Possibilité de montée en charge très importante en fonction des types de plateformes supportées.

Inconvénient : lourdeur dans le cas de solution "monoposte", complexité du fonctionnement, coût élevé des licences, administration importante, nécessité de machines puissantes.

NOTA : Pour en savoir plus sur le sujet, lire l'étude comparative sur les SGBD à base de fichier et ceux utilisant un moteur relationnel, intitulée Quand faut-il investir sur le client/serveur ? On y discute aussi des différents modes de verrouillage ...

5. Les langages du SQL

Le SQL comporte 5 grandes parties, qui permettent : **la définition des éléments d'une base de données** (tables, colonnes, clefs, index, contraintes...), **la manipulation des données** (insertion, suppression, modification, extraction…), **la gestion des droits d'accès aux données** (acquisition et révocation des droits), **la gestion des transactions** et enfin le **SQL intégré**. La plupart du temps, dans les bases de données "fichier" (dBase, Paradox...) le SQL n'existe qu'au niveau de la manipulation des données, et ce sont d'autres ordres spécifiques qu'il faudra utiliser pour créer des bases, des tables, des index ou gérer des droits d'accès.
Certains auteurs ne considèrent que 3 subdivisions incluant la gestion des transactions au sein de la manipulation des données... Cependant ce serait restreindre les fonctionnalités de certains SGBD capable de gérer des transactions comprenant aussi des ordres SQL de type définition des données ou encore gestion des droits d'accès !

5.1. DDL : " Data Definition Language "

C'est la partie du SQL qui permet de créer des bases de données, des tables, des index, des contraintes…
Elle possède les commandes de base suivante :

CREATE, ALTER, DROP
Qui permettent respectivement de créer, modifier, supprimer un élément de la base.

5.2. DML : " Data Manipulation Language "

C'est la partie du SQL qui s'occupe de traiter les données.
Elle comporte les commandes de base suivantes :

INSERT, UPDATE, DELETE, SELECT
Qui permettent respectivement d'insérer, de modifier de supprimer et d'extraire des données.

5.3. DCL : " Data Control Language "

C'est la partie du SQL qui s'occupe de gérer les droits d'accès aux tables.
Elle comporte les commandes de base suivantes :

GRANT, REVOKE
Qui permettent respectivement d'attribuer et de révoquer des droits.