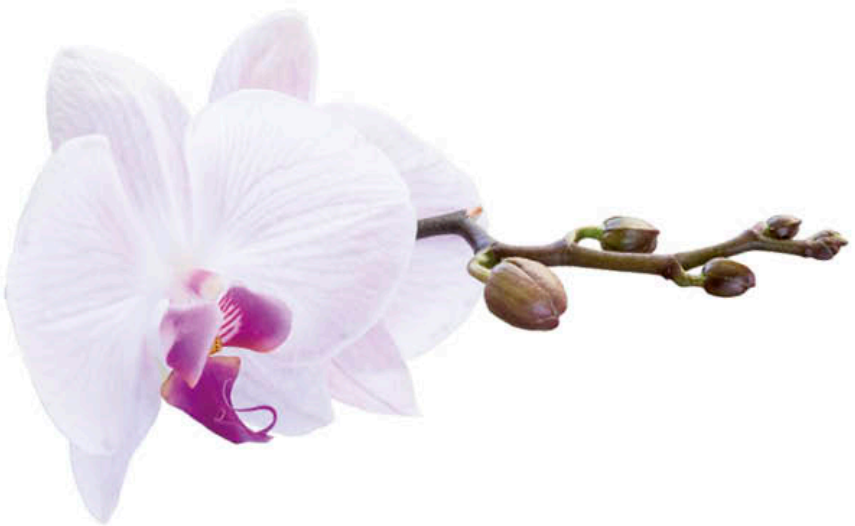
Sylvain Berger
Cédric Cassagne
Cédric Chaissac
René Rampnoux

2e édition

SQL par l'exemple

La pratique professionnelle des bases de données



ellipses

1 Un peu d'histoire

La vitesse des innovations technologiques fait oublier que les concepts évoluent plus lentement. L'idée SQL « existe » depuis un demi-siècle. L'innovation fut de se focaliser sur l'extraction des données selon les besoins des utilisateurs sans s'occuper de l'organisation physique du stockage des données.

1.1 SQL - Origines

Le stockage des informations et leur restitution est l'activité majeure des systèmes d'informations. D'abord, on automatise naturellement cette action à partir de fichiers et de programmes dédiés, habitudes des organisations. En 1968, le mathématicien hollandais D. L. Childs démontre que pour répondre à une question comme : « Quels sont les coureurs de plus de 50 ans qui ont participé au marathon de Bordeaux ? », il suffit d'utiliser trois notions ensemblistes : la « sélection », la « relation » et le « regroupement ». SQL est lancé.

Pendant que Childs publie ses travaux, Edgar Frank Codd, membre du centre de recherche d'IBM©, travaille sur un système de stockage des données plus performant que les fichiers indexés ou les bases de données structurées sous forme hiérarchique, comme le produit phare d'IBM© d'alors, *Information Management System* mis en service en 1966 pour le programme Apollo. Au lieu d'une structure pyramidale comme IMS, où ne peuvent exister que des liens hiérarchiques, Codd pense « relations », c'est-à-dire des tableaux normalisés, des ensembles, associés à un *Universal Data Sublanguage* fondé sur une algèbre relationnelle.

En 1974, c'est chose faite : Donald D. Chamberlin et Raymond 'Ray' Boyce publient *SEQUEL : a Structured English QUEry Language* (« Langage d'interrogation structuré en anglais »), renommé ultérieurement SQL pour cause de conflit de marque déposée. Ce « prototype » nommé SYSTEM/R sait bien réaliser sélection/regroupement sur une relation avec les requêtes utilisant les mots SELECT, FROM, WHERE, GROUP BY. Mais IBM© se désintéresse de ce projet. Immense erreur stratégique dont l'équivalent est la négligence du microordinateur, à la même époque.

C'est Larry Ellison qui, travaillant pour une base de données pour la CIA qu'il baptise « ORACLE© », voit toute l'opportunité des idées de Codd et fonde Software Development Laboratories qui commercialise le premier SGBD relationnel en 1979 écrit en C/assembleur. SQL est définitivement lancé sur le marché professionnel. Il arrive dans l'environnement Windows© en 1988, avec dBASE IV.

1.2 La normalisation

SQL a été normalisé plusieurs fois, de 1986 à 2011 par l'ISO, l'Organisation internationale de normalisation. Le mot norme est inapproprié car rien d'obligatoire n'existe en la matière ; ce ne sont que des recommandations ce qui explique que les SQL implémenté dans chaque entreprise gardent des spécificités telles qu'il est impossible de réaliser la moindre base de données compatibles entre

ORACLE©, Sybase, SQL Server (Microsoft©) et DB2 (IBM©), les poids lourds du marché. La dernière norme en date est référencée ISO/IEC 9075 en vente au format .pdf sur *iso.org*.

SQL est un langage simple et concis qui permet l'accès aux tables mais aussi leur description, la mise à jour des données et le contrôle des utilisateurs. Il est de type déclaratif, par opposition à procédural. Le moteur relationnel seul décide de la méthode d'exécution de la requête à l'aide de ses statistiques. C'est donc un Langage de Manipulation de Données (LMD, *Data Manipulation Language*), un Langage de Description de Données (LDD, *Data Definition Language*), un langage de contrôle des droits d'usage (LCD *Data Control Language*), un langage de contrôle de l'exécution des transactions (*Transaction Control Language*) et un langage d'intégration aux langages procéduraux (*Embedded SQL*). Il se réfère à une vue ensembliste des données. Il peut être utilisé directement grâce à un utilitaire Interactive SQL (ISQL) ou dans un langage hôte (VB, PHP ...).

1.3 Le marché

Il existe aujourd'hui quatre types d'informatique autour des SGBDR :

- Les bases de données personnelles dominées par MS-Access ; citons MySQL également.
- Les solutions de groupe (Workgroup). On y trouve ORACLE©Workgroups pour Nt et SQL server de Microsoft.
- Les applications départementales et les systèmes d'entreprise qui ont les mêmes réponses en termes de produits autour de DB2 (IBM©), ORACLE©, Ingres, MySql.
- Et l'internet.

L'internet est un environnement qui n'est pas du tout conçu pour l'utilisation de base de données. Il a été pensé pour de simples consultations, pas pour permettre un accès sécurisé à des données où la notion de transaction est nécessaire. Une transaction implique une connexion continue et fiable pendant un laps de temps nécessaire aux échanges. Elle commence puis se termine par une déconnexion.

À l'inverse, l'internet est une suite de connexions/déconnexions, le temps mobilisé étant très court à chaque fois. Il faut donc indiquer à chaque envoi vers le serveur internet, qui et pourquoi on sollicite ou on fournit des données. De plus, le nombre d'utilisateurs et les volumes de données explosent. D'où l'apparition de nouveaux SGBD dits NoSQL qui ne mettent pas en œuvre certaines fonctionnalités classiques des SGBDR en vue d'obtenir la puissance de calcul et la capacité d'évoluer facilement.

Il existe deux types d'implémentation physique des SGBDR :

- Ceux qui utilisent un service de fichiers associés à un protocole de réseau afin d'accéder aux données. Une unité de stockage est partagée qui

contient des fichiers partageables. Préférer une forte granularité des fichiers (dBase, Foxpro, Paradox).

- Ceux qui utilisent une application centralisée dite serveur de données qui assure une relative indépendance entre les données et les demandes de traitement venues des applications. Les droits des utilisateurs sont gérés d'une façon indépendante du système d'exploitation (ORACLE©, DB2 (IBM©), SQL/Server…).

1.4 NoSQL (*Not Only SQL*)

Les géants du Net stockent chaque jour plusieurs terabytes de données sur leurs utilisateurs. Plutôt que de mettre à jour sans cesse leur SDBDR pour en accroître les performances, ils ont développé, pour leurs besoins propres, des bases de données qui s'éloignent du modèle relationnel. En 2008, Facebook rend open source sa base de données non-relationnelle écrite en Java qui devient Apache Cassandra. ORACLE© annonce ORACLE© NoSQL Database en décembre 2015.

Cette technologie associe des données plus ou moins complexe à un identifiant, par exemple celui d'un utilisateur, et des informations comme son adresse, sa photo… C'est ce qui existe dans un fichier traditionnel : une clef d'identification associée à des données. On agrège des données comme une grosse molécule autour d'une clef informatique, quand le modèle relationnel décompose au maximum pour arriver à l'atome logique. La redondance était une grave lacune dans une structure relationnelle. Ce n'est pas le cas pour l'environnement NoSQL. Les agrégats sont réalisés ensuite, selon les besoins de l'utilisateur, par des requêtes SQL.

Cet abandon du modèle relationnel et de ses contraintes de normalisation, est-ce une régression ? Intellectuelle certainement car le retour à des données sur un couplage primaire clef/valeur sans cohérence des données assurée n'est pas une fulgurance conceptuelle. Mais la puissance des machines pour piocher dans d'immenses entrepôts de données en vrac permet de s'exonérer de la recherche de bonnes structures de données. Nous en reparlerons avec l'examen du Big Data dans le chapitre 21 : « Pour aller plus loin : le Big Data ».

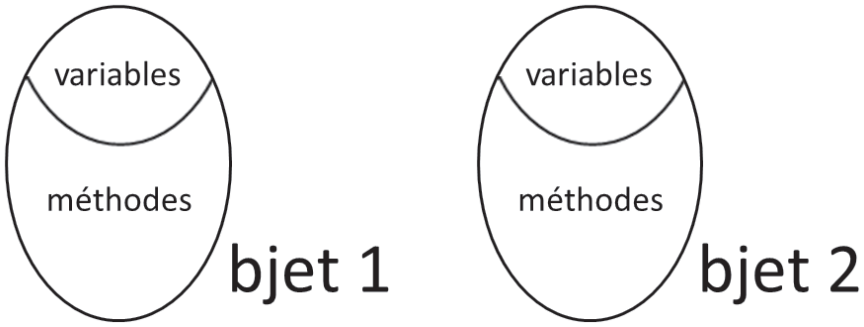
« Il est difficile de définir ce qu'est une base de données NoSQL car il n'y a pas de normes aujourd'hui à ce sujet. Il y a littéralement des centaines de produits qui prétendent être des bases de données NoSQL ou avoir des capacités NoSQL » (Dave Segleau, directeur de Produits chez ORACLE©, *digora.com*).

1.5 Les bases de données orientées objets

C'est une tentative d'unir le poisson/donnée et l'oiseau/traitement. Les SGBDR ont été pensés pour séparer les données, richesse et stabilité dans les systèmes d'information, des traitements très évolutifs. Les tables se pensent d'abord d'une façon cohérente, les restitutions viennent ensuite, que l'on découvre chemin faisant. SQL, par sa souplesse, répond à toute demande aisément si les données sont bien présentes dans la base et organisées. Cette dichotomie, poussée à

l'extrême ne reflète pas le monde réel de l'utilisateur qui mélange intimement ce qu'il fait (actions) et ce sur quoi il agit (données). D'où l'idée de concevoir des solutions informatiques et de les mettre en œuvre avec le concept d'objet qui a :

- Ses attributs, appelés variables.
- Ses méthodes décrivant son comportement.



Seules les méthodes de l'objet peuvent manipuler ses attributs.

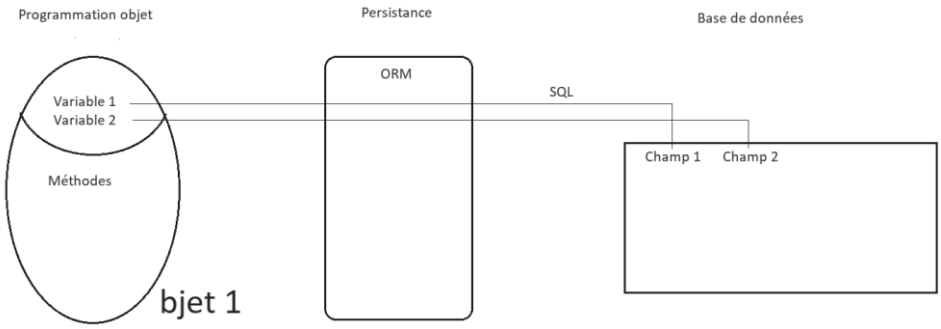
L'objet est d'abord apparu en programmation où il va s'imposer avec Java comme fer de lance. Les SGBD objets vont être un échec sauf dans des domaines spécifiques. L'exception est ORACLE© 8 qui glisse vers le concept relationnel-objet. Il permet de définir des types de données à objets (exemple : on se crée un type Adresse avec 4 lignes, 1 code postal, 1 ville). Un objet peut disposer de méthodes écrites en PL/SQL, en C ou en Java.

1.6 Cas d'utilisation des bases de données en programmation objet

Lorsqu'il travaille, un développeur peut s'appuyer sur des outils (Framework, librairies …) qui lui permettent de ne pas avoir besoin de coder certaines fonctionnalités génériques. Des Framework comme Hibernate remplissent la fonction d'enregistrement des données des objets. Ces enregistrements peuvent se faire sous forme de fichiers ou dans une base de données relationnelle.

De manière générique on parle de **persistance des données**. Les objets utilisés en programmation n'existant que durant l'exécution d'un programme, la persistance consiste à sauvegarder les propriétés des objets.

Lorsqu'on utilise une base de données relationnelle, il est nécessaire de déclarer les liens entre les propriétés des objets et les propriétés des relations (tables) de la base. On parle alors **d'Objet Relational Mapping**. Le schéma ci-après présente les liens décrits. La couche de persistance est alors prise en charge par le Framework.



L'utilisation de tels systèmes a pour principal avantage d'affranchir le développeur de coder le stockage des informations et donc de produire plus rapidement une application.

Il y a cependant des inconvénients comme celui de ne pas respecter les règles de l'art en termes de modélisation et d'utilisation d'une base de données relationnelle. De plus, cette utilisation de Framework ne permet pas de s'appuyer sur certaines fonctionnalités des SGBD relationnel comme les requêtes complexe, les procédures stockées … Elle rend la base de données abstraite.