Introduction Aux SGBD et SGBDR

1 Introduction : L'importance des fondations

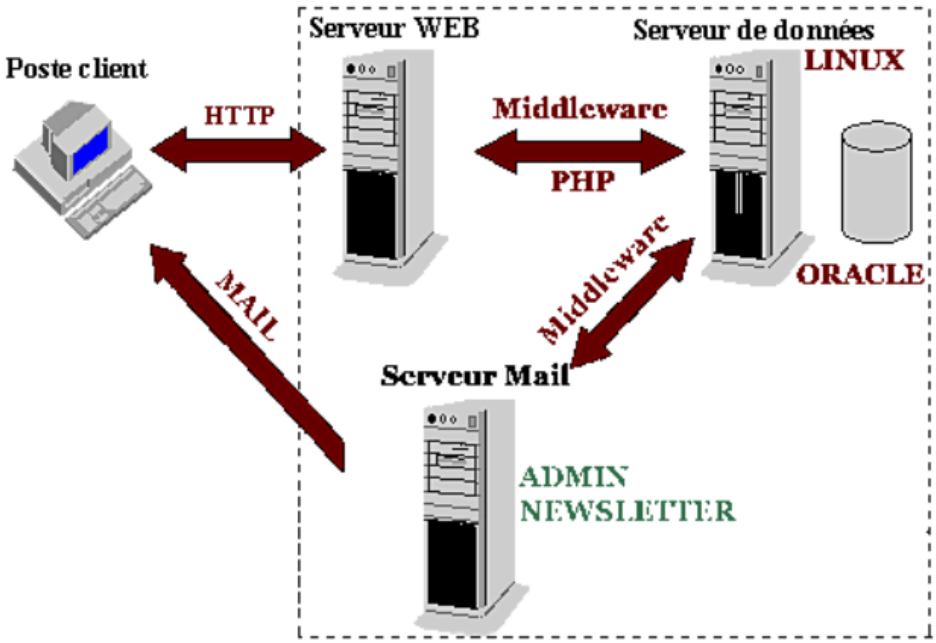
La conception d'une base de donnée est une étape cruciale dans le développement d'un site Web. Indépendamment des autres choix techniques, (Matériel, langages de programmations, système d'exploitation du serveur, etc...), son architecture et sa construction détermineront en grande partie la cohésion et le bon fonctionnement du site et du logiciel. Il s'agit de la partie servant à stocker les informations et cette fonctionnalité est primordiale dans la plupart des environnements modernes et traditionnels (boutiques, portails, ...). Sa fiabilité est décisive pour la réussite du projet dans lequel elle est impliquée, car c'est sur elle que repose l'ensemble du développement.

Son utilisation est de plus en plus présente et sa structure de plus en plus complexe, nécessitant des compétences professionnelles d'analyse de plus en poussé afin de maîtriser les divers aspects de sa conception et de son utilisation. Il est même possible d'affirmer que la majorité des techniques et problématiques du Client Serveur seront amenées à être transposées dans la conception Web. C'est donc très logiquement que les informaticiens du Web ont été amenés à développer leurs compétences et à organiser leur travail en conséquence afin d'entrer dans l'ère des sites complexes et du traitement de l'information.

2 L'architecture SGBD WEB : la descendance du Client-Serveur

Plusieurs personnes consultent et gèrent les même informations depuis plusieurs ordinateurs. L'information est gérée sur un ordinateur principal distant, chaque utilisateur accédant aux données communes depuis son poste de travail.

Définitions :
Le poste de consultation s'appelle le poste **CLIENT**.
Les Données sont gérées et transmises par un **SERVEUR**.
Dans le cas de l'Internet :
Le programme client est le **Navigateur**, qui n'est qu'un **interpréteur** HTML.
Le **logiciel** est situé sur le serveur Web, et gère les données via la base de donnée.
Exemple d'architecture Web :

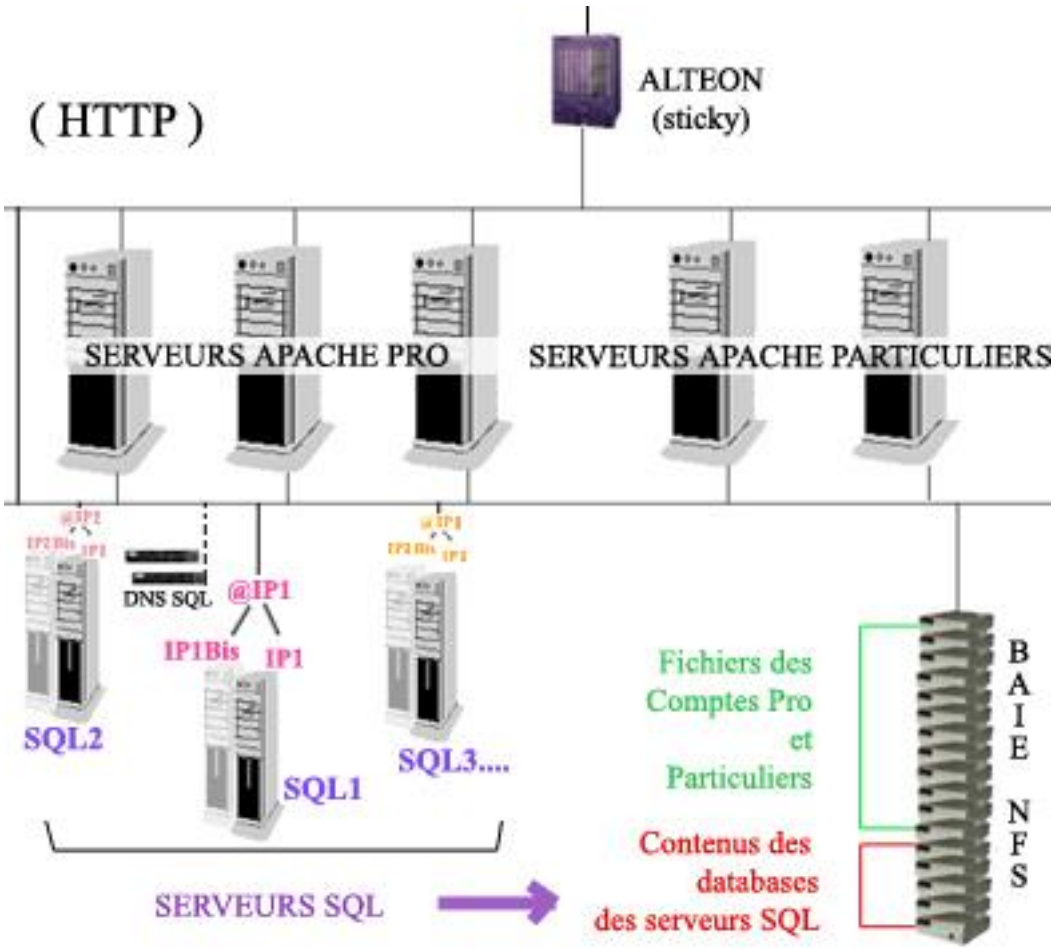


Concrètement, le **SGBD stocke les données et informations**. Le **Logiciel/Serveur d'Applications** s'occupe de **l'insertion et de l'extraction (lecture)** de ces données, et communique le **HTML contextuel** correspondant au **Serveur Web**, lequel renvoie ce HTML via un **flux HTTP** au Poste client qui consulte la page demandée grâce à son **Navigateur local**.

IMPORTANT :
Le poste client ne gère aucune donnée par lui-même.

Note :
- Il est courant que les 3 serveurs ci dessus (Web + SGBD + Mail) soient regroupés sur la même machine (cas des sites de petite et moyenne importance).
- Il est possible de déléguer certaines fonctions logicielles au serveur SGBD : Triggers, procédures stockées…. L'avantage est d'économiser la charge réseau du Middleware et la charge machine du serveur applicatif.
- Exemple de Middleware : *: ODBC, CORBA, HTTP…*
- L'architecture ClientServeur standard était traditionnellement à deux niveaux.
- Dans le cas d'un hébergement mutualisé, ce type d'architecture peut devenir nettement plus complexe :

Exemple d'étude pour une plate-forme d'hébergement mutualisé LAMP (Linux + Apache + MySql + PHP) commandée par SIRIS et réalisée par IBM :



Mais l'ensemble de cette structure, dans l'absolue et vue par l'internaute uniquement, peut être assimilé à un serveur unique…..

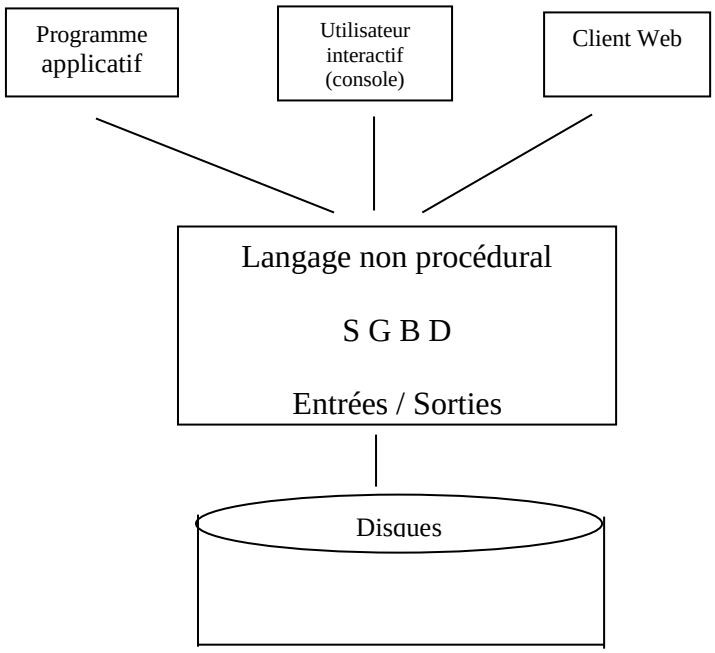
3 Définition du SGBD

Une **base de données** (Data base) est un ensemble de données
- Structuré, modélisant les objets du monde réel
- Servant de support à une application informatique
- Interrogeable par le contenu (on doit pouvoir retrouver des objets satisfaisant à certains critères).

Un **SGBD (**Système de Gestion de Bases de Données, Data base Management System) est un outil informatique offrant différentes couches de fonctions servant à gérer la **Base de données**. A cet effet, elle peut fournir divers outils : la gestion des données via le gestionnaire de fichiers interne (indexation), la cohérence des données (pour les SCGB**R ,** ou base de donnée relationnelle), la gestion du langage de requète utilisé, ...

De plus, les SGBD mettent à la disposition de nombreux utilisateurs les données de l'entreprise. Cette mise en commun ne va pas sans problèmes. Il a notamment fallu prévoir le partage et la sécurité des données : transactions, files d'attentes, authentification .....

Ce qui nous amène au schéma suivant :



D'un côté, le SGBD gère les entrées-sorties disques. D'un autre, les données sont interrogées et mises à jour soit par des programmes applicatifs, soit par des programmes utilitaires fournis avec le SGBD (éditeurs de rapport par exemple). Ces programmes sont écrits dans des langages de programmation traditionnels appelés langages de 3ème génération (C, Cobol, …) ou dans des langages plus avancés dits environnements de 4ème génération (Visual Basic, Delphi, ….).
Dans tous les cas, l'accès à la base de données se fait à l'aide d'un **langage unifié de description et de manipulation de données**, pour nous le langage SQL.

Les SGBD/R dit **relationnels** (la seconde génération des SGBD) ont été commercialisés dans les années 80 (Oracle, SyBase, Informix, SQL Server). Ils occupent une place essentielle en informatique. Les données y sont représentées sous forme de **table**, les recherches et mises en forme sont effectuées à l'aide d'un langage **non procédural**, **SQL**.

La troisième génération supporte le relationnel et l'objet : il est cependant nécessaire de noter que beaucoup d'entreprises ayant fait le choix technologique d'utiliser un SGBD Objet reviennent en arrière. Les compétences demandées pour administrer, voir même pour simplement utiliser ces SGBD étant tellement pointues, et le gain de performance si peu évident, que beaucoup d'entre elles sont repassées sur des outils plus traditionnels.

La base en fichiers textes...

Nombre d'entreprises ont développé en interne de petits systèmes dynamiques en C ou en PERL, généralement à base de fichiers textes. La construction de ces bases était bien souvent directement influencée par les requêtes désirées, dans un souci d'optimisation des temps de réponse. Des balises particulières étaient alors placées dans le HTML aux endroits ou l'on désirait faire apparaître les données dynamiques. Chaque fichier HTML était parcouru par un programme qui remplaçait les dites balises particulières par le contenu des fichiers textes avant de l'envoyer au serveur.

Pour éviter les manipulations des données, les développeurs prenaient en compte l'ordre d'insertion des enregistrements pour éviter les tris, très gourmands en mémoire et ressources car effectués en scripts plus ou moins bien optimisés. De même, un script recevait les formulaires afin d'écrire leur contenu au bon endroit dans des fichiers eux aussi au format texte. Ces systèmes ont rendu par le passé de fiers services pour de petits développements comprenant un nombre limité d'informations bien souvent très simples.

Néanmoins, ils ont montré leurs limites dès qu'il y eut besoin d'afficher des résultats de tris faisant intervenir plusieurs types d'informations. Par exemple, supposons que l'on ait besoin d'accéder aux "Titres des vidéos de sports dont la dernière commande date du 10 Novembre 2000". Il faut prendre en compte les catégories, la Date, la localisation géographique des clients, les Editeurs et toutes les vidéos référencées. Chaque requête de ce type demandait un nouveau développement spécifique. Et là, cela devenait très compliqué.

4 Méthodologie de conception d'un serveur SGBD

L'objectif majeur d'un SGBD est d'assurer **l'indépendance des programmes et des données**, c'est-à-dire de préserver la possibilité de modifier les schémas conceptuels (ajouter une nouvelle table, de nouveaux champs, etc.....) sans modifier les programmes d'applications et donc les schémas externes vus par ces programmes. De même, dans la mesure où les informations traités et les fonctionalités sont identiques, il devra être aussi possible de changer de logiciel, voir de langage, sans remettre en cause la base de donnée existante. Cette indépendance évite une maintenance coûteuse des programmes.

5 La généralisation du SQL

Le SQL est un langage naturel proche du discours humain, signifiant Structured Query Language (Langage d'Interrogation Structuré). Développé par le laboratoire de recherche d'IBM à San José en Californie à la fin des années 70, il est reconnu en tant que norme officielle de langage de requête relationnelle par l'institut ANSI et par l'organisme ISO. Il facilite grandement la manière d'indiquer ce que l'on désire obtenir à la machine. La commande principale est " SELECT ". Ainsi, la phrase "Les Titres des vidéos de sports dont la dernière commande date du 10 Novembre 2000" devient la requête " SELECT TitreVideos FROM VIDEOS WHERE Style='Sport' AND LastCommande='2000-11-10' ", ce qui est beaucoup plus facile à programmer qu'un tri de tableaux ou de fichiers.

Les informaticiens du Web ont alors développé des interpréteurs SQL pour mieux gérer leur développement. Ils ont aussi mis au point des systèmes de "Locking", c'est à dire de protection

dans le cas ou plusieurs utilisateurs accèderaient simultanément aux données. Comme très souvent dans l'Internet, il a surtout été question de faire vite, souple, efficace, facile à utiliser et répondant très vite. Très rapidement, les langages de programmation Internet ont été munis des librairies nécessaires (DBI) pour fonctionner avec ces bases de données. Devant la facilité et les gains de temps apportés par ces technologies, les développeurs se sont engouffrés dans l'utilisation du SQL qui s'est ainsi très largement généralisé.

Le meilleur exemple de cette évolution est MySql, une des bases de donnée les plus utilisées dans le Web actuellement. Rapide, fiable et efficace, elle est capable de supporter une très grande quantité d'enregistrement ainsi qu'une charge . Elle n'est actuellement pas relationnelle car cela ne correspondait pas aux besoins auquel elle était chargée de répondre.

6 **... l'avenir est à la base**

L'avenir sur Internet est aux sites dynamiques gérant un très grand nombre d'entrée/sortie d'informations. Une boutique avec une gestion des moyens de paiements, promotions, TVA, Catégories, produits, caddy, etc... comporte au minimum une vingtaine de tables. Un site gérant des compétitions sportives peut nécessiter plusieurs centaines de tables selon ce que l'on veut y gérer.

Dorénavant plus que jamais, on ne peut se permettre de bâcler un cahier des charges et de négliger la conception de sa base de données. Il n'est plus possible de se lancer à corps perdu dans la conception en créant ses tables au fur et à mesure, au petit bonheur la chance, sous peine de gros soucis de développements. Dans le cas contraire, les symptômes sont désormais classiques : évolution du cahier des charges au fur et à mesure du développement, absence de cahier technique, et donc : Non-respect des délais, projet se retrouvant rapidement très compliqué, données fausses ou incompréhensibles à l'affichage. Un des risques peut aussi être l'impossibilité de reprendre ou de modifier l'existant rapidement en l'absence du concepteur, compliquant alors singulièrement la maintenance. Mais surtout, tout ceci est généralement synonyme de dépassement de Budget...

Les techniques pour construire une base de donnée existent depuis maintenant plusieurs années, et ont fait leurs preuves dans des secteurs nécessitants la gestion de très gros Systèmes d'Informations. La plus connue est MERISE, qui provient directement des travaux des Français MOULIN, TARDIEU et TEBOUL, vers le début des années 70, et rendue célèbre par l'américain Peter CHEN en 1976. A ce jour, tous les spécialistes du domaine de l'analyse orientée base de données se servent de ce modèle comme outil d'analyse et de conception avec la plupart des logiciels de construction d'applications de bases de données comme PARADOX, ORACLE, SQL Server, Informix, Ingres, Sybase... Appliquer ces méthodes est un moyen efficace d'arriver à un développement fiable et évolutif, et donc à une réussite technique et commerciale.