Système de gestion de bases de données (SGBD)

Introduction :

L'objectif de ce chapitre est d'identifier les enjeux que représentent les bases de données informatisées et les besoins qui découlent de l'exploitation de ces bases.
Dans un premier temps dans ce cours, nous nous pencherons sur les besoins à l'origine des bases de données informatiques. Puis nous déroulerons un bref historique concernant les techniques employées pour leur mise en œuvre. Enfin, nous détaillerons en quoi consiste un système de gestion de base de données ainsi que les différents moyens d'interagir avec celui-ci.

1 Qu'attend-on des bases de données informatisées ?

a. Des attentes au niveau des données elles-mêmes

Le but de regrouper un ensemble d'informations dans une base de données est de pouvoir les retrouver facilement, en faisant appel à des **critères de recherche**.

Exemple

Dans cette première sous-partie nous utiliserons l'exemple de données relatives aux recettes et adhérent·e·s d'un club de cuisine.

Le volume d'informations que les organisations sont amenées à brasser est toujours grandissant. Aussi, si l'on souhaite retrouver tous les adhérent·e·s de moins de trente ans dans cette base de données, celle-ci sera informatisée dans le but premier de **limiter le temps des recherches**.

- Toutefois, une base de données informatisée doit aussi pouvoir répondre à une grande variété de recherches, sans que l'on ait à les prévoir toutes au moment où elle est conçue.
- Il est également essentiel que cette base soit accessible par différents programmes, donc qu'elle soit **indépendante des traitements informatiques** que l'on peut élaborer sur les données qu'elle stocke.
- Enfin, on attend d'une base d'une part que ses données soient **cohérentes**, d'autre part que les informations qui y sont stockées ne soient **pas redondantes**.

Exemple d'incohérence

Dans la base recensant des recettes de cuisine, certaines recettes sont enregistrées avec l'ingrédient « yaourt », d'autres avec « yogourt ». Ainsi, en cherchant les recettes à bases de « yaourt », on ne retrouvera pas celles enregistrées à base de « yogourt ».

→ Bien que les deux termes désignent le même ingrédient, cette incohérence sur son nom biaise le résultat des recherches.

Exemple de redondance

Toujours dans notre base, on stocke le nom, le prénom, l'adresse de messagerie ainsi que l' adresse, le code postal et la commune où habite chaque adhérent·e du club. Cette information « code postal +ville » va se trouver dupliquée dans la base autant de fois qu'il y a d'auteur·e·s de recette qui habitent la même commune.

→ C'est, un gâchis de place qui peut s'avérer coûteux lorsqu'il ne s'agit plus de la base de recettes d'un petit club de cuisine, mais, par exemple, de celle d'un grand site international de partage de recettes qui compte les adresses des auteur·e·s pas millions.

À retenir

> C'est le rôle d'une base de données informatisée, que l'on aura conçue selon des règles précises, que de permettre d'éviter les incohérences et redondances.

b. Des attentes au niveau des performances

L'optimisation du temps pris par les opérations dans une base de données informatisée implique une réflexion sur l'architecture matérielle employée. On identifie trois grands rôles (niveaux) dans un système d'informations :

- **interactions avec l'utilisateur·rice** ;
- **calcul et traitements sur les données** ;
- et **gestion des données** (organisation, stockage et accès).

→ Selon l'architecture employée et les besoins en performance, ces rôles sont répartis différemment.

Les architectures peuvent comporter plusieurs couches, généralement appelées « tiers » (ex. : 1-tier, 2-tiers, … n-tiers).

Attention

> Le terme « tiers » est un anglicisme et un faux-ami signifiant en français « niveau », « couche » ou « étage ».

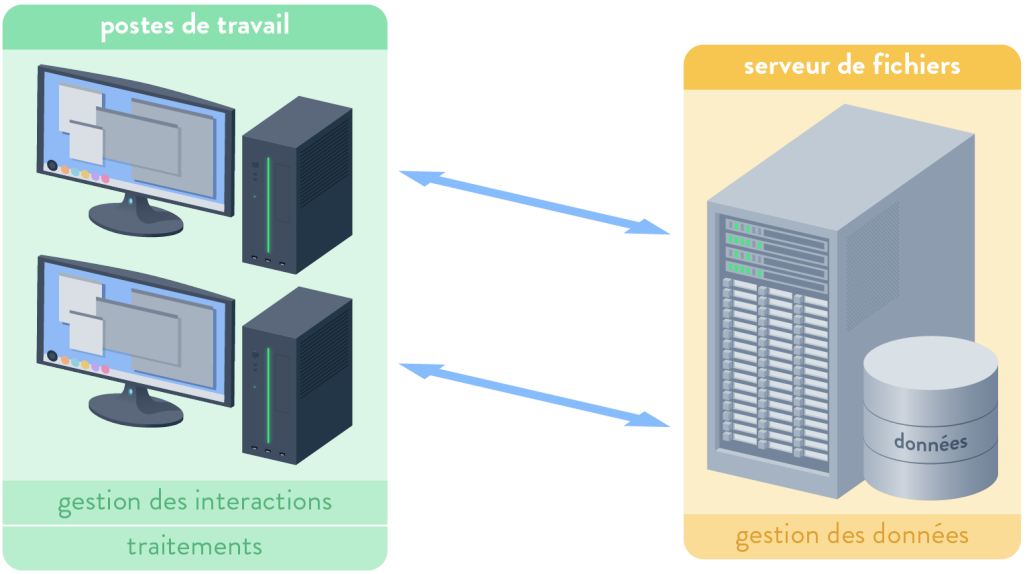
Rappel

> L'interaction client-serveur repose sur des échanges entre plusieurs programmes : l'un (le client) contacte l'autre (le serveur) et lui envoie des requêtes auxquelles ce dernier doit répondre.

1 Architecture 1-tiers : architecture correspondant à un client dit « autonome »

Cette architecture, aujourd'hui un peu dépassée, impliquait l'installation des applications sur chacun des postes de travail. Cette répartition rendait la gestion du parc matériel assez lourde. Les données étaient alors à l'époque plutôt réparties dans des fichiers individuels (et non dans un SGBD, tel qu'étudié plus loin).

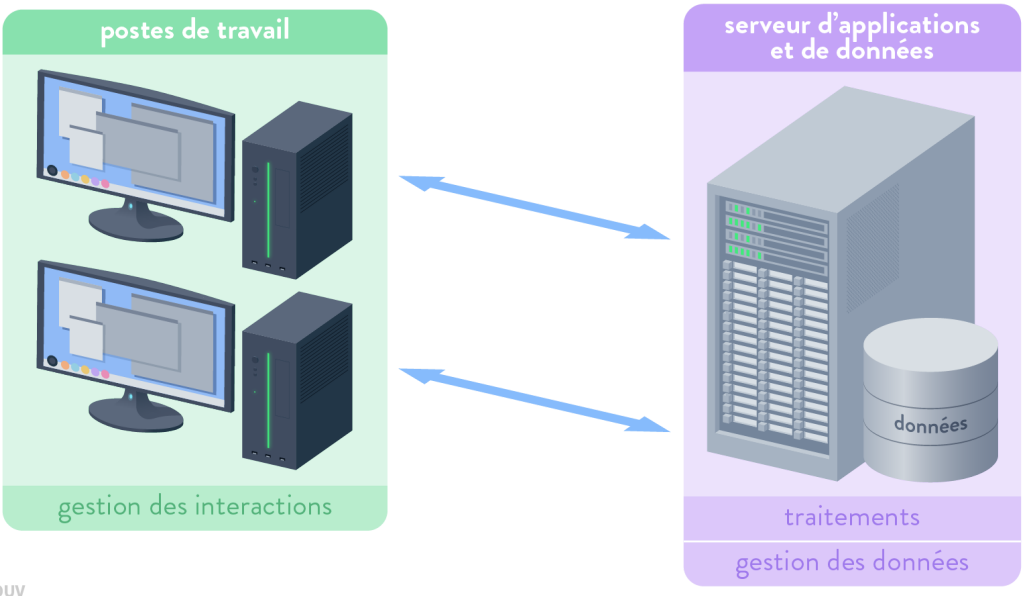
Architecture 1-tiers



2 Architecture 2-tiers : architecture correspondant à un client dit « lourd »

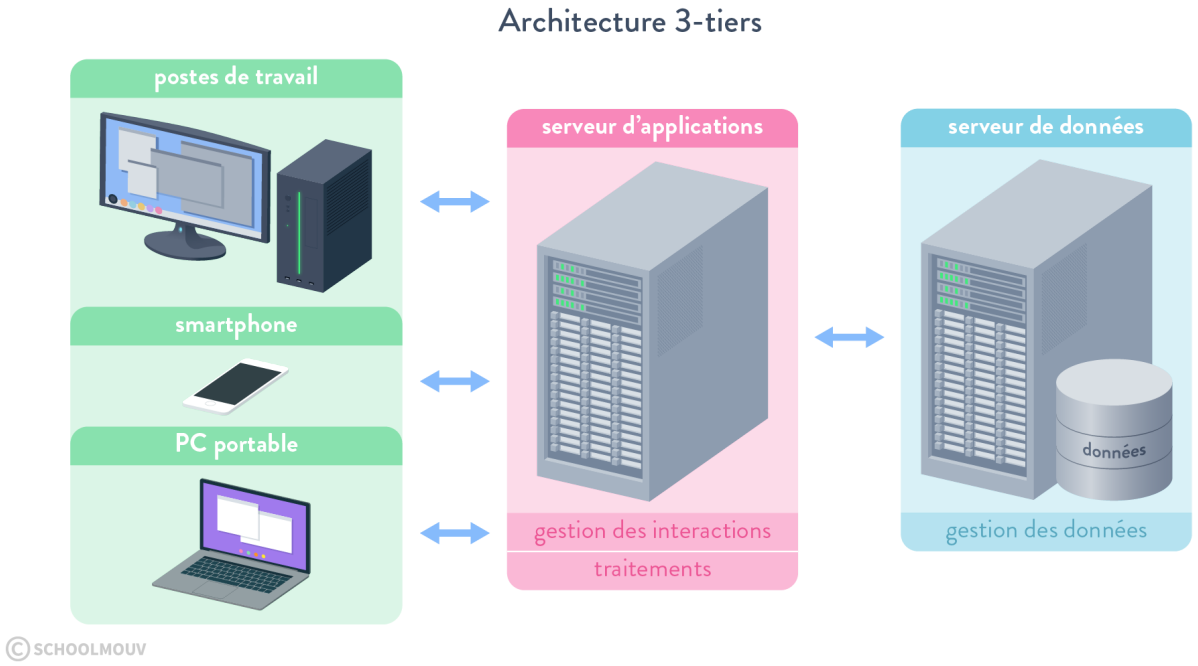
Cette architecture nécessite l'installation sur chaque poste de l'IHM particulière aux traitements. Les traitements eux-mêmes et la gestion des données sont, quant à eux, assurés par un serveur qui peut se trouver sur un réseau différent de celui du client.

Architecture 2-tiers



3 Architecture 3-tiers : architecture correspondant à un client dit « léger »

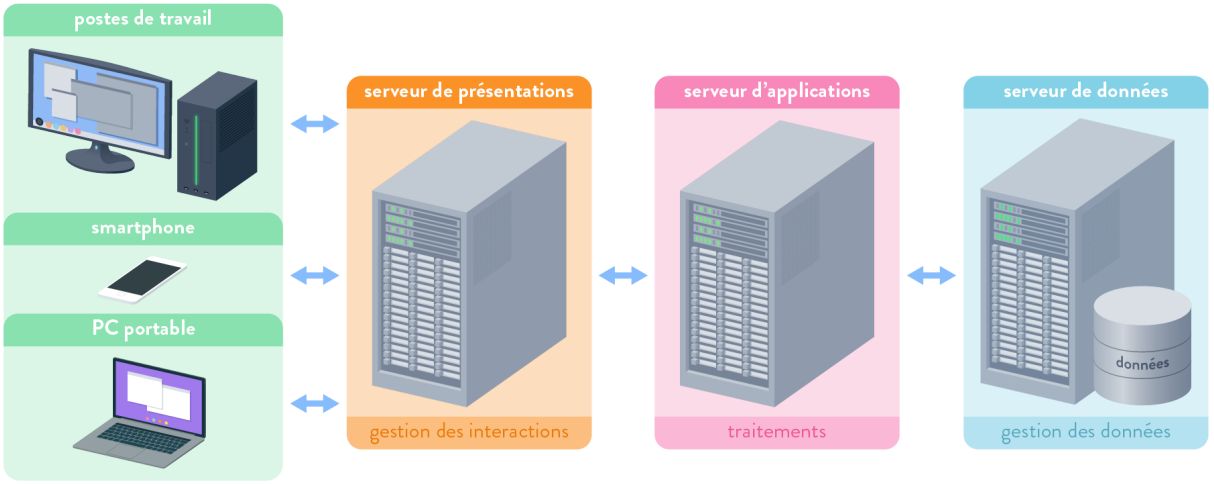
Cette architecture nécessite l'installation sur chaque poste d'une application légère de visualisation, telle qu'un **navigateur web** par exemple. Les traitements et la gestion des données sont chacun assurés par un serveur différent.



4 Architecture 4-tiers

Cette architecture répartit les trois rôles sur des serveurs différents.

Architecture 4-tiers



Astuce

> Notons que le serveur de données peut lui-même faire l'objet d'une répartition : pour des besoins de performance ou de sécurité, les bases de données peuvent être dupliquées ou réparties techniquement sur plusieurs serveurs.

→ On parle alors de **bases de données répliquées** dans le premier cas et de **bases de données réparties** dans le second.

Nous avons identifié les attentes vis-à-vis des bases de données informatisées, étudions à présent comment celles-ci se sont développées au fil des années.

2 | Plusieurs modèles de données

Le développement des bases de données a démarré à partir des années 1960, lorsque les ordinateurs se sont démocratisés et ont commencé à étendre, au-delà du calcul, leurs capacités à des traitements d'informations plus élaborées.
Plusieurs modèles de données ont émergé jusqu'à nos jours.

Définition

> **Modèle de données :**
>
> Un modèle de données définit la manière dont l'information est structurée dans la base de données. Il est une représentation, exploitable par l'informatique, d'informations du monde réel.

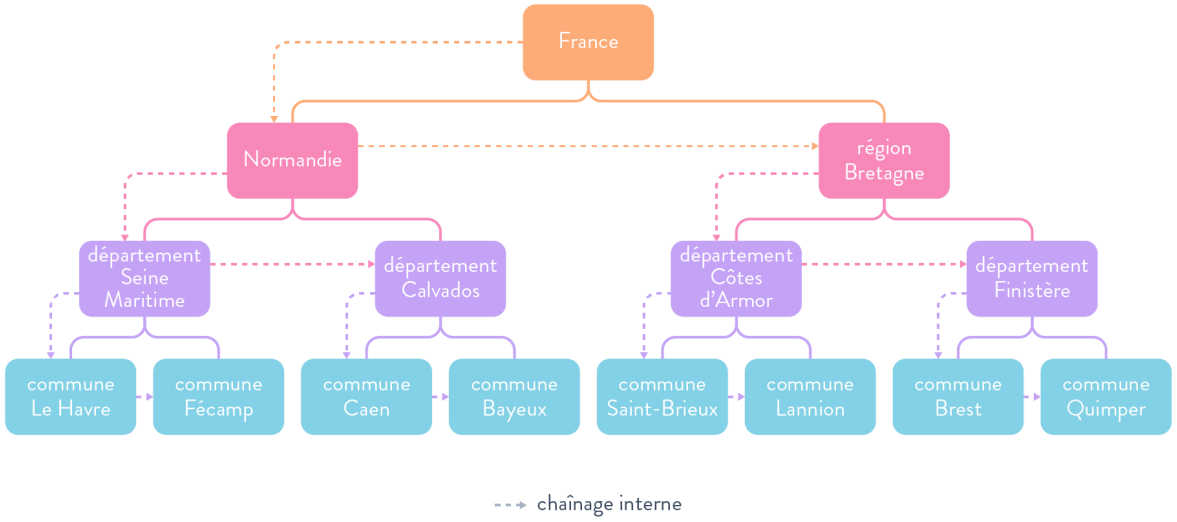
a. Le modèle hiérarchique

Dans le **modèle hiérarchique**, les éléments d'information sont reliés entre eux par une **arborescence** dans laquelle chacun d'eux n'est rattaché qu'à **un seul possesseur**.
La recherche d'une information est contrainte de suivre un parcours défini par un système de chaînage interne : à chaque étape du chaînage, un pointeur stocké avec la donnée indique où trouver la donnée suivante Pour accéder à une donnée, on part du niveau qui lui est supérieur et on parcourt séquentiellement, de proche en proche, grâce aux pointeurs, l'ensemble de ses entités filles, pour trouver la donnée cherchée.

→ Un tel système lie très fortement les données aux programmes qui les manipulent.

Exemple

Les régions, départements et communes de France



Le modèle hiérarchique a, entre autres projets, servi à la gestion des données relatives au projet Apollo de la NASA. Cependant, le monde réel n'est pas toujours organisé selon une hiérarchie rigoureuse au sein de laquelle les liaisons transversales ne sont pas toutes immédiates.
En effet, dans l'exemple du découpage du territoire français, pour récupérer la liste de tous les départements de France (approche transversale), il est nécessaire d'emprunter des liaisons transversales mais aussi verticales.

→ Pour retrouver tous les départements de France, on suit le parcours France → Normandie → Seine maritime → Calvados ; puis Bretagne → Côtes d'Armor → Finsitère.

⚠ Attention

Il n'est pas possible de parcourir l'arbre de façon uniquement transversale : Seine-Maritime, Calavados, Côtes d'Armor puis Finistère, car Calvados et Finistère n'ont pas le même possesseur.

→ Ce qui alourdit considérablement les recherches. D'où l'apparition du modèle réseau.

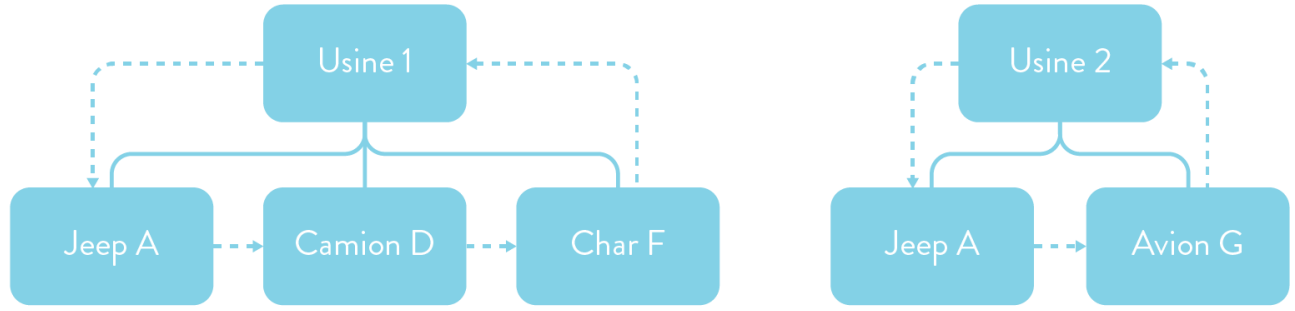
b. Le modèle réseau

Le **modèle réseau** lève certaines limitations du modèle hiérarchique, en particulier en autorisant des liens transversaux. Le système de chaînage interne, formant un graphe, s'en voit inévitablement, et malheureusement, plus compliqué. Une structure en graphe se substitue à la structure en arbre du modèle hiérarchique.

Exemple

> Pour mieux comprendre son principe voici l'exemple de deux usines : Usine A et Usine B produisent respectivement les engins Jeep A, Camion D et Char F pour l'une ; Jeep A et Avion G pour l'autre.

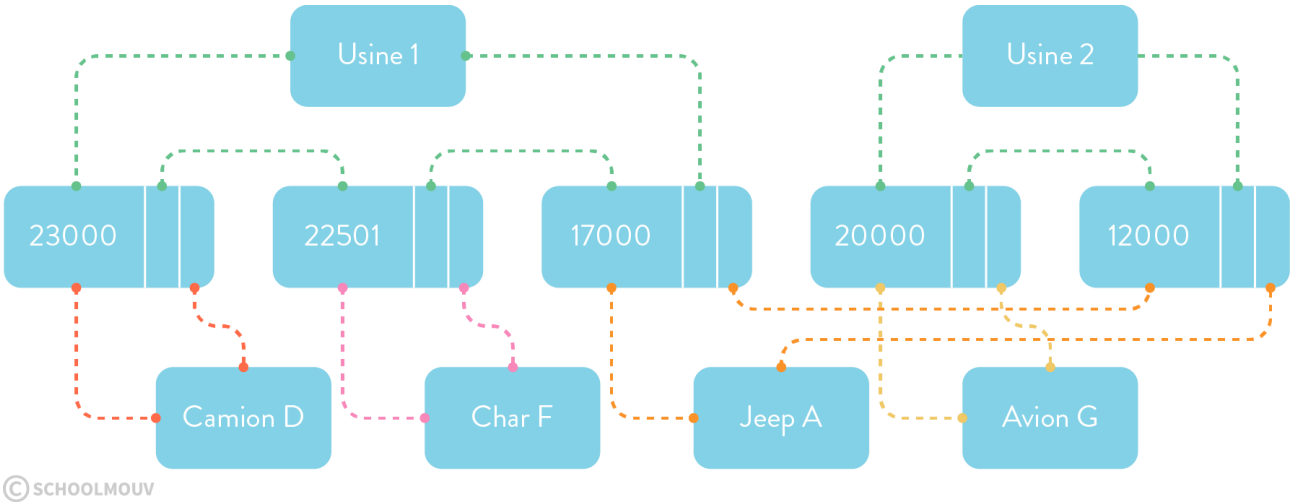
Si l'on traduit cette situation par un modèle hiérarchique, on obtient le résultat suivant.



Attention

> Avec l'utilisation du modèle hiérarchique, on constate un problème de redondance : Jeep A est renseignée deux fois.

Alors qu'avec l'utilisation d'un graphe structuré en modèle réseau, on obtient le résultat suivant.



Des données intermédiaires sont de nature technique et ne font donc pas partie des données issues du monde réel modélisé (ici identifiées 23000, 22501, 17000, etc...). Elles stockent divers pointeurs qui permettent d'établir des parcours chaînés entre les données.

→ On trouve ici un intérêt au graphe par rapport au modèle hiérarchique : la donnée Jeep A n'est pas dupliquée, et, pour autant, on n'a pas perdu l'information qu'elle était produite par l'usine 1 et l'usine 2.

La complexité de ce modèle s'accompagne d'une grande dépendance entre les données et les programmes qui les manipulent.

→ Cette complexité explique que ce modèle ait été abandonné au profit du modèle relationnel.

C. Le modèle relationnel

C'est en 1970 qu'a émergé le concept du **modèle relationnel**, proposé par l'informaticien britannique Edgar Frank Codd, alors directeur de recherche chez le constructeur IBM. Il faudra une dizaine d'années pour que l'idée débouche sur un produit abouti, nommé « Oracle », alors commercialisé par Relational Software. Il sera suivi plus tard par DB2, commercialisé par IBM.

Ce modèle a pour vocation première d'améliorer l'indépendance des données par rapport à leurs traitements : contrat qu'il remplit avec succès. Il s'appuie sur des tableaux à deux dimensions nommés « relations ». Ceux-ci sont liés à d'autres relations (ou « tables ») par un mécanisme

d'associations qui repose sur l'**algèbre relationnelle**. Ces associations définissent et précisent les liens entre les relations.

Les accès aux enregistrements ne reposent pas sur des pointeurs, tels que ceux utilisés dans les systèmes de chaînage évoqués dans les modèles précédents. Ils sont assurés par des clés primaires ou étrangères accessibles par des index.

Rappel

> L'index est une sorte de répertoire qui indique où trouver sur le disque l'enregistrement relatif à une clé donnée.

Par ailleurs, c'est grâce au processus de modélisation abordé dans les chapitres précédents que l'on parvient à définir les liaisons entre les tables.

d. Le modèle objet et relationnel-objet

Ces modèles intègrent la **notion d'objet**, c'est-à-dire celle employée par les langages de programmation orientée objet. Ce concept d'objet s'avère en particulier utile pour modéliser des informations complexes, comme par exemple celles de type multimédia.

Attention

> Ces modèles sont encore en phase de réflexion et de recherche. Nous n'en développerons pas plus le principe ici.

Ces différents modèles ont été conçus pour traiter des données fortement structurées et finement catégorisées. Ils sont en revanche peu adaptés aux besoins croissants de stockage et de traitements, souvent en temps réel, de données massives. Ces mégadonnées (communément appelées *"big data"*) sont alors gérées avec des SGDB de type NoSQL. Cette appellation NoSQL désigne collectivement une famille hétérogène de bases de données différentes du modèle relationnel : s'appuyant sur différents paradigmes (clés-valeurs, colonnes, documents, graphes, multi-modèles, etc.), ces SGBD NoSQL sont capables de traiter de très gros volumes de données possiblement hétérogènes.

Pour conclure sur cet inventaire, nous pouvons noter qu'il subsiste encore, en particulier dans le secteur bancaire, des bases de données hiérarchiques qui s'avèrent plus performantes pour la gestion de certaines informations que les bases de données relationnelles.

À retenir

> Mais retenons que le modèle relationnel est celui qui est le plus largement répandu de nos jours.

Il est temps maintenant de nous intéresser aux SGBD, systèmes qui permettent de mettre en œuvre un modèle de données.

3 Les missions d'un SGBD

Rappel

> Une base de données informatisée est un ensemble d'informations stockées sous forme de données sur un support numérique.

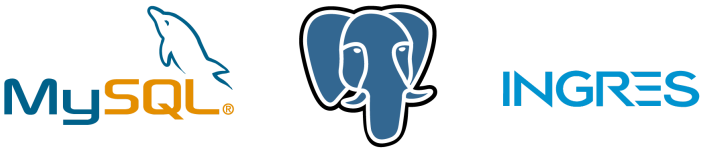
Définition

> **SGBD :**
>
> Un SGBD, sigle signifiant « Système de gestion de base de données », est un logiciel spécialisé pour manipuler des données stockées en base.

Parmi les SGBD relationnels les plus connus et utilisés, on peut citer :

- open-source ;

Exemples de SGBD open-source



- propriétaires.

Exemples de SGBD propriétaires

Définition

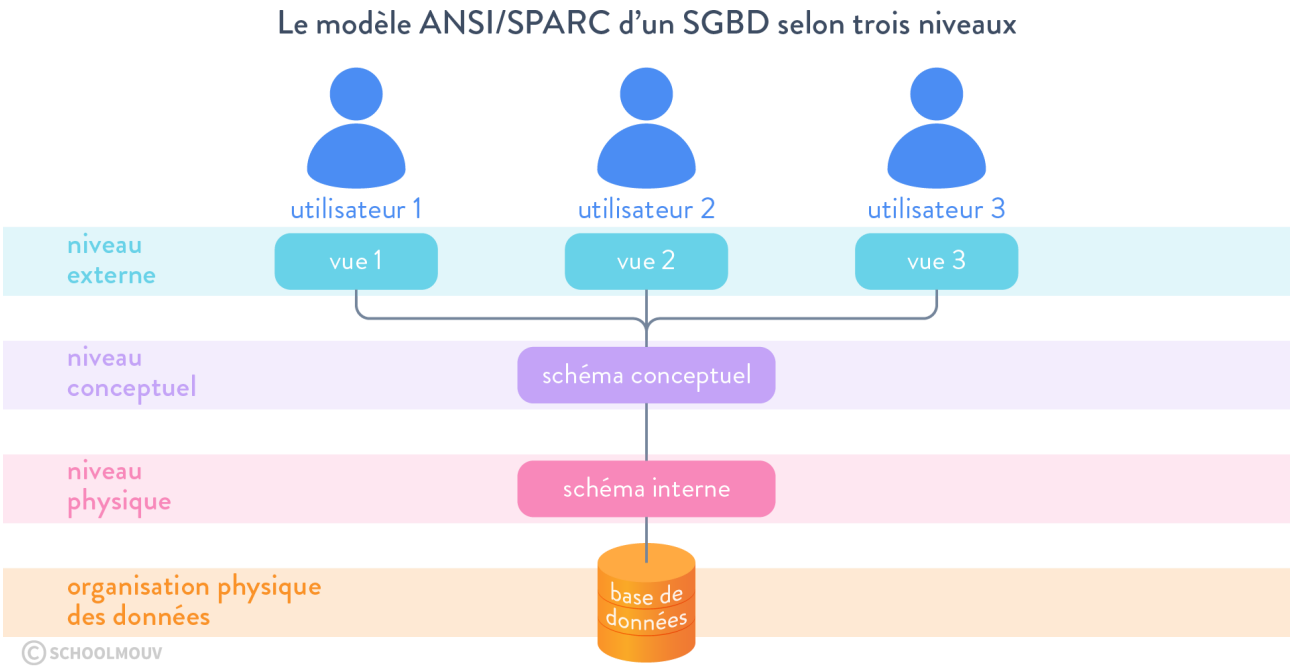
> **Système d'informations :**
>
> Un système d'informations est l'ensemble constitué par la base de données, le SGBD et les programmes d'applications qui lui sont associés.

L'architecture ANSI/SPARC spécifie l'architecture d'un SGBD selon un découpage en trois niveaux.

1. **Le niveau physique** correspond au niveau interne, dans lequel on décrit les données telles qu'elles sont stockées dans la machine. Ce niveau est dépendant du SGBD employé, utilisant un modèle physique de données qui lui est spécifique.
2. **Le niveau conceptuel** (ou logique) est utilisé pour décrire les éléments de la base qui traduisent, de manière structurée, les informations du monde réel. Ce

niveau, qui définit l'organisation des données, ignore totalement le niveau physique.

3. **Le niveau externe** sert à décrire les différentes vues des données qui doivent être proposées aux utilisateur·rice·s suivant leur catégorie. Il est indépendant du niveau conceptuel.



Voyons un peu plus en détail ce qu'apporte chacun de ces niveaux.

a. Le niveau physique

Vous trouverez listées ci-dessous les différentes missions du niveau physique.

- Gestion des données sur le support de stockage (fichiers).
- Représentation physique des valeurs de données et encodage (réel, entier, texte…).
- Partage des données et gestion des accès concurrents :

→ il s'agit de permettre à plusieurs utilisateur·rice·s d'accéder simultanément aux mêmes données, y compris lorsqu'il s'agit de les modifier.

- Résistance aux pannes :

→ il faut que le SGBD garantisse un état « sain » de la base après une panne intervenant au milieu d'une modification. Le but est que la modification alors interrompue soit achevée ou que les données soient ramenées dans l'état qui précédait la modification.

À retenir

> Notons que l'on peut mesurer la qualité des opérations de modifications d'un SGBD en vérifiant les propriétés ACID que voici.
>
> - **A**tomicité : une modification est soit exécutée entièrement, soit abandonnée.
> - **C**ohérence : une modification doit se faire d'un état cohérent de la base vers un état cohérent de la base.
> - **I**solement : deux modifications simultanées ne doivent pas interférer entre elles.
> - **D**urabilité : une modification est permanente, même en cas de panne.

b. Le niveau conceptuel

Rappel

> Le niveau conceptuel est indépendant du niveau physique.

Vous trouverez listées ci-dessous les différentes missions du niveau conceptuel.

- Définition de la structure de données à l'aide d'un langage de description de données (LDD)
- Consultation et mise à jour des données à l'aide d'un langage d'interrogation des données (LID) ainsi qu'un langage de manipulation des données (LMD)
- Cohérence de données assurée par la description de contraintes d'intégrités
- Gestion de la confidentialité

c. Le niveau externe

Rappel

> Le niveau externe est indépendant du niveau conceptuel.

→ Un ensemble de données peut être vu différemment selon le profil des utilisateur·rice·s.

Vous trouverez listées ci-dessous les différentes missions du niveau externe.

- Vues décrivant la partie des données qui présente un intérêt pour un·e utilisateur·rice ou un groupe d'utilisateur·rice·s
- Vues décrivant la partie des données qui présente un intérêt pour des programmeur·euse·s d'applications
- Interfaces d'utilisation
- Environnements de programmation

Nous connaissons dorénavant les missions d'un SGBD. Mais comment l'utiliser ? C'est ce que nous allons voir maintenant, en recensant les différents moyens d'interagir avec lui.

4 Modes d'interaction avec un SGBD

Pour **agir sur la base de données**, que ce soit dans le but de la créer, de la mettre à jour ou de la consulter, il est nécessaire de solliciter le SGBD. Selon les situations et les utilisateur·rice·s, plusieurs moyens permettent d'interagir avec le SGBD. Quels sont-ils ?

a. La ligne de commande

À l'aide de l'application « Terminal » ou « Console » (sous Linux ou Unix) ou de l'« invite de commandes » (sous Windows), il est possible de solliciter le SGBD via des commandes entrées en mode texte, donc au clavier.

Ce mode d'interaction, assez rudimentaire et peu ergonomique, est plutôt employé par les informaticien·ne·s. Ces dernier·e·s formulent leurs commandes en ayant recours au LDD, LMD et LID.

Astuce

> L'intérêt de la ligne de commande est qu'elle exige peu de ressources, ce qui peut être utile dans certains cas de situations dégradées.

Exemple

> Sous Windows avec MySQL : connexion à la base de données « recettes » et commande pour obtenir la liste des tables de la base de données.

b. L'interface graphique

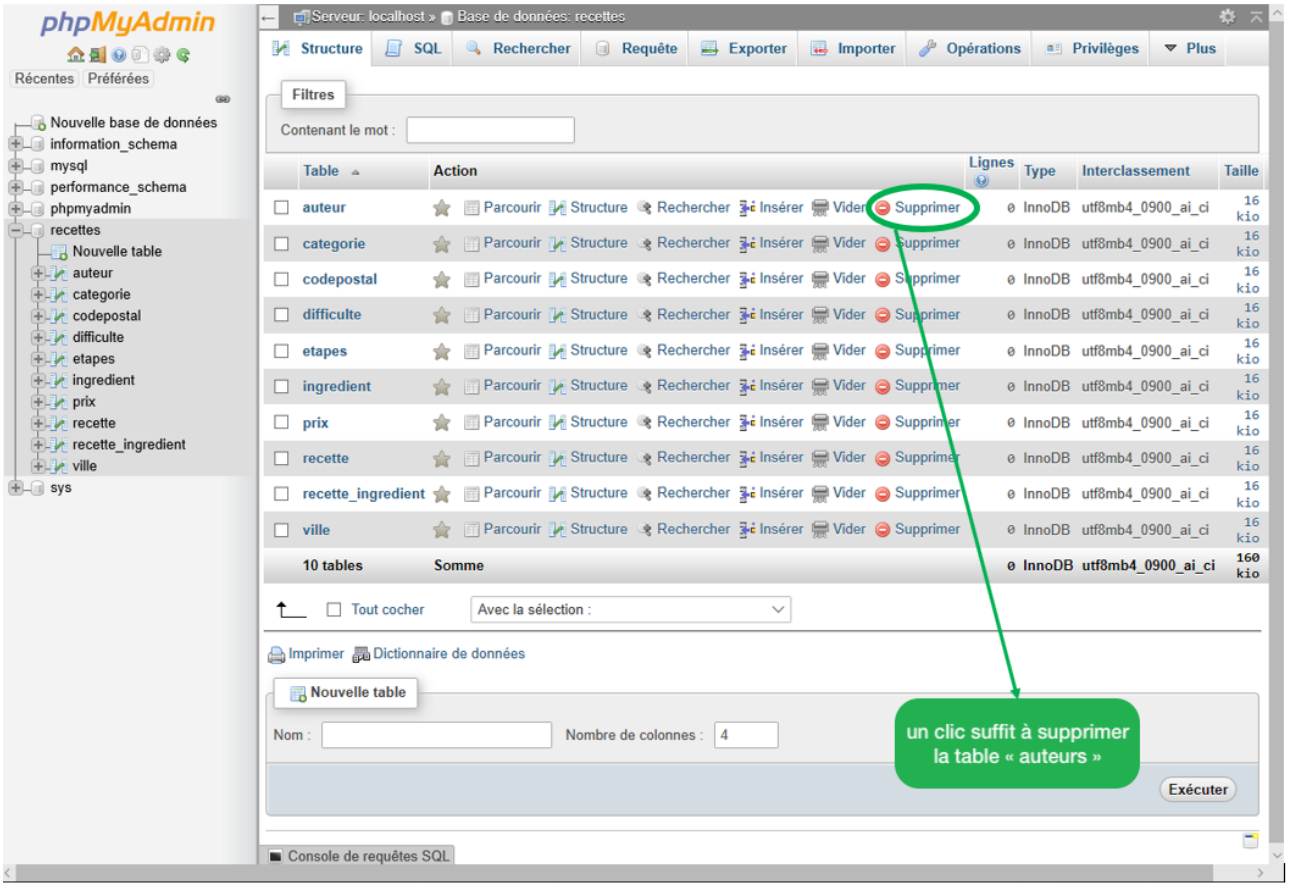
Les SGBD s'accompagnent d'une interface graphique d'administration. Une partie des instructions passées avec le LDD, le LMD ou le LID en ligne

de commande y sont substituées par de simples clics. Notons qu'il est nécessaire de se connecter au SGBD au préalable afin que les données et les opérations proposées par l'interface ne soient que celles qui sont autorisées.

Exemple

Par exemple, PhpMyAdmin est une interface d'administration très répandue de bases MySQL. Il en existe d'autres. PHPMyAdmin se présente sous la forme d'une application Web accessible *via* un navigateur. Dans l'exemple donné ci-dessous, un seul clic de souris dans l'interface PHPMyadmin suffit pour demander au SGBD la suppression de la table « auteur ».

→ Cela a eu pour effet de lui transmettre la commande « drop table auteur », empruntée au LDD.



c. Les bibliothèques spécialisées

Les bibliothèques spécialisées sont les éléments de code qui permettent à des programmes de solliciter un SGBD, que ce soit pour lui adresser des instructions de consultation, modification, suppression de donnée ou plus encore, de modification de la structure de la base de données elle-même.

Exemple

> L'exemple ci-dessous est un programme écrit en PHP qui effectue la même opération que celle effectuée via l'interface PHPMyAdmin dans l'exemple précédent : la suppression de la table « auteur ».

```
<!DOCTYPE html>
<html>
    <head>
        <title>Exemple d'usage de la bibliothèque spécialisée mysql
        <meta charset='utf-8'>
    </head>
    <body>
        <h1>Suppression de la table "auteur"
        <?php
            servname = "localhost" ;dbname = "recettes" ; user = "admin4289 » ;pass = "QiJArRlLste4qiy";

            /* connexion à la base de données */
            try{
                dbco = newPDO("mysql : host =servname ;dbname =dbname",user, pass) ;;
                $dbco->setAttribute(PDO : :ATTRERRMODE, PDO,: :ERRMODEEXCEPTION);

                /* soumission de l'instruction au SGBD
                $ sql = "DROP TABLEauteur";
                dbco− > exec(sql);
                echo 'Table bien supprimée';
            }

            catch(PDOException ]$e){
                echo "Erreur : " . $e->getMessage();
            }
        ?>
```

```
    </body>
</html>
```

Notons au passage que PHPMyAdmin n'est ni plus ni moins qu'une application PHP qui a été écrite en faisant appel aux bibliothèques spécialisées relatives au SGBD MySQL.

> Conclusion :
>
> Nous connaissons maintenant l'intérêt des bases de données informatisées ainsi que la richesse des SGBD. Il est temps d'apprendre à les exploiter. C'est ce que nous allons aborder dans les deux chapitres qui suivent. Nous y découvrirons SQL, et plus précisément les sous-ensembles LID et LMD de langage.