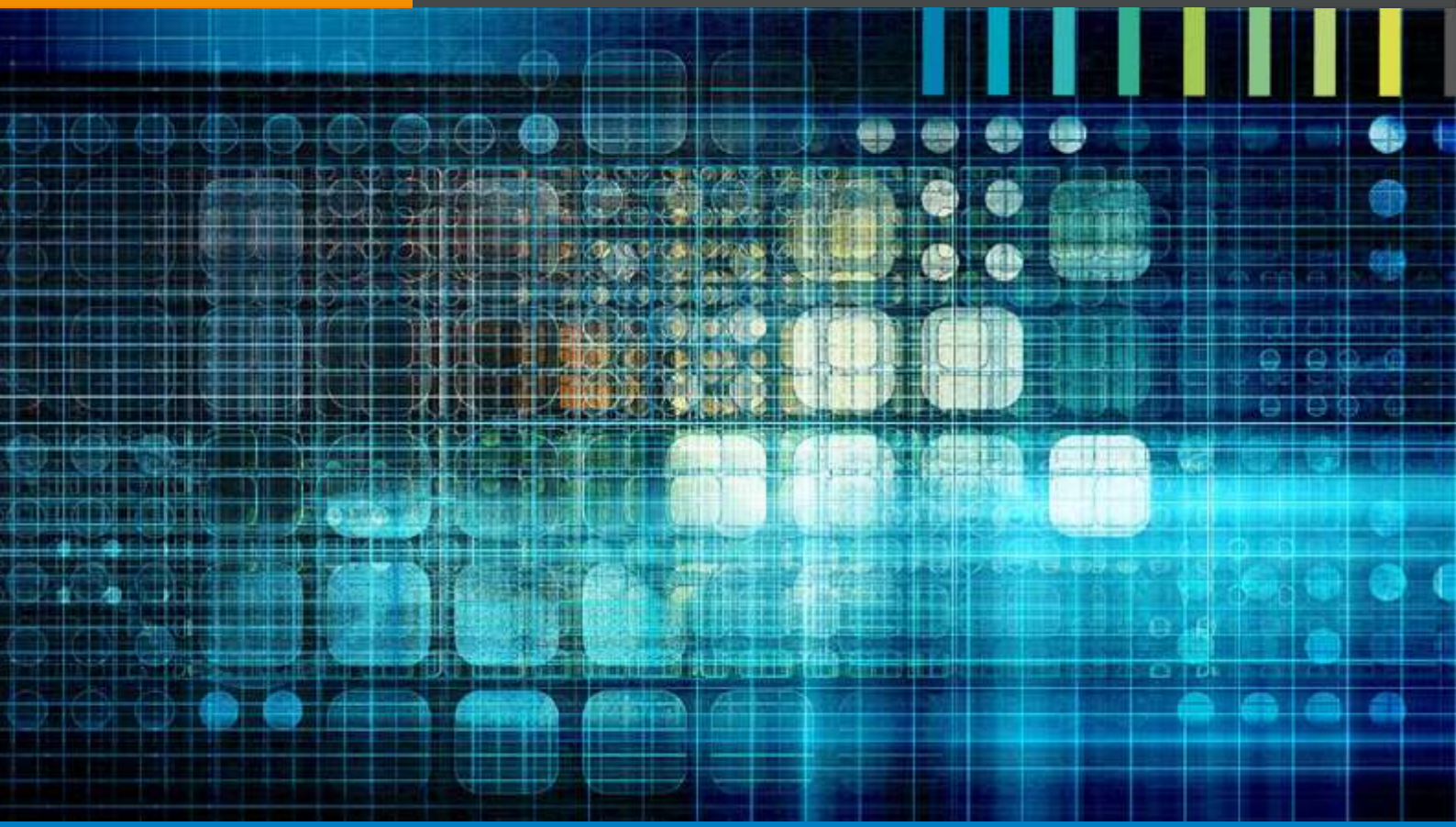
RÉPUBLIQUE DU NIGER
Fraternité - Travail - Progrès
MINISTÈRE DU PLAN
INSTITUT NATIONAL DE LA STATISTIQUE
PLATEFORME NATIONALE D'INFORMATION POUR LA NUTRITION

NIGER
GUIDE MÉTHODOLOGIQUE
AVRIL 2021

NUTRITION



GUIDE FORMATION EN GESTION ET SÉCURISATION DES BASES DE DONNÉES EN LIGNE

SOFRECO

HC3N

Institut National de la Statistique NIGER

AVANT-PROPOS

Dans le cadre du programme de la Plateforme Nationale d'Information pour la Nutrition (PNIN) et du programme de formation de l'Assitance Technique, il est prévu de renforcer les capacités de l'Institut National de la statistique (INS), du Haut-Commissariat à l'I3N (HC3N) et des Directions Statistiques (DS) des Ministères Sectoriels (Santé, Éducation, Agriculture et Élevage, Hydraulique et Assainissement, Environnement) dans le domaine de la gestion et la sécurisation des bases de données en ligne.

Un Système de Gestion de Base de Données (SGBD) est un logiciel système servant à stocker, à manipuler, à gérer, à partager des informations dans une base de données. Un SGDB doit garantir la qualité, la pérennité et la confidentialité des informations, tout en cachant la complexité des opérations, de manière transparente pour les utilisateurs. Utilisés pour de nombreuses applications informatiques et de plus en plus par les producteurs d'informations statistiques pour la mise en place de bibliothèques numériques, les SGDB permettent de sortir rapidement des informations suite à des requêtes spécifiques.

De plux en plus de Directions Statistiques (DS) centralisent leurs données sur des SGDBR (Système de Gestion de Base de Données Relationnelles) en lignes afin de faciliter l'accès à leurs données aux utilisateurs de plus en plus nombreux. Beaucoup d'initiatives poussées par les autorités nigériennes et les Partenaires Techniques et Financiers (PTFs) visent à accroitre l'accès aux informations statistiques. Dans ce cadre, il importe de renforcer la connaissance théorique et pratique des SGDB afin de sensibiliser les administrateurs de bases de données à mettre en place des SGDB.

Le présent Guide a été élaboré pour la formation en gestion et sécurisation des bases de données en ligne effectuée par l'Assitance Technique de la PNIN du 19 avril au 23 avril 2021. En effet, la mise en place de bases de données en ligne doit s'accompagner de mesure pour la sécurisation des données. Clées de voûte de beaucoup d'institutions, les bases de données reposent souvent sur des produits clés en main dont les vulnérabilités sont souvent mal connues des différents services qui les utilisent. Ainsi, la sécurité et la bonne maîtrise du Système de Gestion de Base de Données (SGBD) sont devenues indispensables. L'accès aux micro-données est une activité en croissance au niveau de l'Institut National de la Statistique (INS) et du Système Statistique National (SSN). Au sein même des institutions, la mise en place des bases de données en réseau accroit les problèmes de sécurisation des bases de données. La connexion d'un SGBD avec un progiciel, qui nécessite une méthode d'interconnexion spécifique, peut avoir pour effet d'abaisser le niveau de sécurité. Il importe de trouver un accord entre les personnes responsables de la sécurité des données et ceux responsables de l'intégration des données.

Avoir une connaissance sur la garantie de la traçabilité, l'intégrité, l'auditabilité, la confidentialité et la sauvegarde des données en fonction de ce que l'on souhaite garantir et des besoins identifiés est une première étape au renforcement de la sécurisation des données.

Guillaume **POIREL**

Expert Technique International, Chef de mission de l'Assistance Technique de la Plateforme Nationale d'Information pour la nutrition

SIGNALÉTIQUE

OURS

Unité responsable : Plateforme Nationale d'Information pour la Nutrition (PNIN)

Directeur du projet : **ALCHINA KOURGUENI Idrissa**, Directeur Général de l'INS

Chargée du suivi du projet : Mme OMAR Haoua Ibrahim, Secrétaire Générale de l'INS

Coordonnateur : **MAMAN HASSAN Moussa,** Coordonnateur de la Plateforme Nationale d'Information pour la Nutrition (PNIN), Institut National de la Statistique (INS)

Auteur :

Spécialiste des Systèmes d'Information et bases de données, AT-PNIN, SOFRECO : **Alexis CAPO-CHICHI**

Contributeur :

Chef d'Équipe, Statisticien-Analyste, Assistant Technique PNIN (AT/PNIN) : **POIREL Guillaume**

Photos Illustration : PNIN.

Éditeur de la publication : Plateforme Nationale d'Information pour la Nutrition / INS

SIGLES ET ABRÉVIATIONS

AC	Autorité de Certification
BLOB	Binary Large Object
CRUD	Create Read Update Delete (ou Destroy)
CRUD	Create, Read ou Retrieve, Update, Delete ou Destroy
DCIP	Disponibilité Intégrité Confidentialité Preuve
DCL	Langage de Contrôle des Données
DDL	Langage de Définition des Données
DML	Langage de Manipulation des Données
DOS	Denial Of Service
DS	Direction des Statistiques
HC3N	Haut-Commissariat à l'Initiative 3N « les Nigéreins Nourrissent les Nigériens »
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
INS	Institut National de la Statistique
LCD	Langage de Contrôle de Données
LCT	Langage de Contrôle des Transactions
LDD	Langage de Définition de Données
LMD	Langage de Manipulation de Données
OWASP	Open Web Application Security
PIN	Code d'Identification Personnel
PNIN	Plateforme Nationale d'Information pour la Nutrition
PTFs	Partenaires Techniques et Financiers
SGBD	Système de Gestion des Bases de Données
SGBDR	Système de Gestion des Bases de Données Relationnelles
SI	Système d'Information
SLA	Service Level Agreement
SQL	Structured Query Language
SSL	Secure Sockets Layer
SSN	Système Statistique National
TLS	Transport Layer Security
URI	Uniform Resource Identifier

SOMMAIRE

LISTE DES TABLEAUX

LISTE DES FIGURES

MODULE 1 : LES SGBD, DÉFINITIONS, PRINCIPES ET ARCHITECTURE

Dans ce premier module, vous commencerez par approfondir quelques définitions et principes indispensables liés aux Systèmes de Gestion des Bases de Données (SGBD) et aux bases de données (BD), pour ensuite installer un SGBD sur votre ordinateur.

À la fin de ce module, vous devriez :

- Savoir ce qu'est un SGBD, une base de données ;
- Connaitre les objectifs, les fonctionnalités d'un SGDB et les types de SGDB existants ;
- Savoir les caractéristiques des données : types de données, structure des données contenues dans une base de données, règles de cohérence et relations.

1. BASE DE DONNÉES ET SGBD

1.1 QU'EST-CE QU'UNE BASE DE DONNÉES

Une base de données informatique est un **ensemble de données** qui ont été stockées sur un support informatique et **organisées** et **structurées** de manière à pouvoir facilement consulter et modifier leur contenu.

Prenons l'exemple d'un site web avec un système de news et de membres. On utilise par exemple, une base de données MySQL pour stocker toutes les données du site : les news (avec la date de publication, le titre, le contenu, éventuellement l'auteur, etc.), les membres (noms, emails, …). Tout ces informations vont constituer notre base de données pour le site. Mais il ne suffit pas que la base de données existe. Il faut aussi pouvoir la **gérer, interagir avec cette base**. Il faut pouvoir envoyer des messages à MySQL (messages qu'on appellera « **requêtes »**) afin de pouvoir ajouter des news, modifier des membres, supprimer des informations et tout simplement afficher des éléments de la base.

Une base de données seule ne suffit donc pas, il est nécessaire d'avoir également :

- Un système permettant de gérer cette base ;
- Un langage pour transmettre des instructions à la base de données (par l'intermédiaire du système de gestion).

1.2 QU'EST-CE QU'UN SYSTÈME DE GESTION DE BASE DE DONNÉES

Un Système de Gestion de Base de Données (SGBD) est un logiciel (ou un ensemble de logiciels) permettant de manipuler les données d'une base de données. Manipuler, c'est à-dire sélectionner et afficher des informations tirées de cette base, modifier des données, en ajouter ou en supprimer (ce groupe de quatre (4) opérations étant souvent appelé « CRUD », pour Create, Read, Update, Delete). MySQL est par exemple un système de gestion de bases de données.

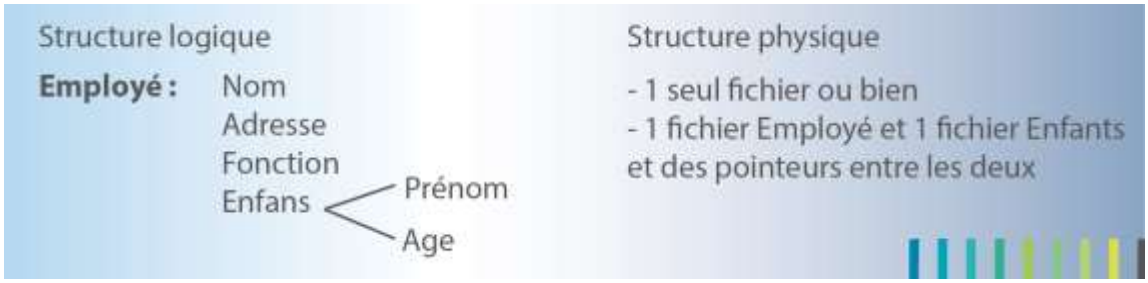
Figure 1 : Principe d'une Base de données

Source : Plateforme Nationale d'Information pour la Nutrition

1.2.1 LES OBJECTIFS D'UN SYSTÈME DE GESTION DES BASES DE DONNÉES

Un système de Gestion de Bases de données doit permettre de masquer les aspects de stockage des données et respecter le **principe d'indépendance physique.** Plus psécifiquement, la manipulation des données doit se faire indépendamment de leur structure de stockage : représentation abstraite des données, structure logique ou conceptuelle, structure physique. On peut ainsi modifier la structure physique sans que les utilisateurs soient affectés et sans modification des programmes.

Figure 2 : Structure logique et principe d'indépendance



Principe d'indépendance logique

Une application travaille sur une représentation logique des seules données qu'elle manipule : le SGBD assure la correspondance avec les données réelles. Les programmes ne doivent pas être revus lors de la modification de données qu'ils n'utilisent pas.

Exemple : Les données d'un hôpital : médecins, malades, chambres, …

- Application n°1 : Suivi des malades (Nom, N°SS, N°Chambre, Médecin) ;
- Application n°2 : Données d'un médecin (Nom, N°SS, N°Chambre, Thérapie) ;
- Application n°3 : Gestion du personnel : les médecins (Nom, Grade, Spécialité, Salaire).

Gérer efficacement les données

Offrir un stockage de données efficace par rapport à un enregistrement conventionnel dans des fichiers.

Optimiser les traitements de données

- Faciliter l'extraction et l'ajout données ;
- Obtenir et de modifier rapidement des données ;
- Garantie l'absence de plusieurs copies de la même donnée (redondance) ;
- Vérification des données pour assurer que les données introduites soient correctes (intervalle admis, format correct).

Assurer la sécurité des données

Éviter les conflits lors d'exploitation partagée :

- Plusieurs utilisateur/logiciels peuvent accéder simultanément aux données ;
- Des outils pour éviter les éventuels conflits de modification.

1.2.2 LES FONCTIONS D'UN SYSTÈME DE GESTION DES BASES DE DONNÉES

Les fonctions d'un SGBD permettent **d'enregistrer des données, de rechercher, de modifier et de créer automatiquement des comptes rendus** du contenu de la base de données, de spécifier les **types** de données, la **structure** des données ainsi que des **règles de cohérence** ou d'**intégrité** (exemple : l'absence de redondance, unicité de clé, etc.).

Les **contraintes d'intégrité** sont des **règles sémantiques** permettant de garantir la cohérence des données lors des mises à jour de la base. Ces règles peuvent être **déclarées explicitement** et mémorisées dans la base de données ou plus discrètement **implicitement**.

Un SGBD est équipé par des mécanismes qui effectuent des vérifications afin d'assurer la réussite des transactions, éviter des problèmes dus aux accès concurrents et assurer la sécurité des données.

Les **transactions** sont des groupes des opérations unitaires qui transforme le contenu de la BD d'un état cohérent « A » vers un état cohérent « B ». À la fin de chaque transaction ou plus après chaque mise à jour, il est nécessaire de contrôler qu'aucune règle d'intégrité ne soit violée. En cas de panne survenue durant des opérations de modification de la BD, le SGBD remet la BD dans l'état ou elle était au début de la transaction (état « A »).

La **concurrence** correspond à la manipulation simultanée d'une BD par plusieurs personnes afin que cela n'aboutisse pas à des incohérences. Pour un logiciel de réservation par exemple, le SGBD vérifie que chaque place est réservée au maximum par une personne, même si des réservations sont effectuées simultanément.

La **sécurité** permet d'interdire l'accès à des données par des **listes de contrôle d'accès.**

Les **caractéristiques des données** dans la base de données sont par exemple les relations, les règles de cohérence et les listes de contrôle d'accès. Ces caractéristqiues sont enregistrées dans un catalogue qui se trouve à l'intérieur de la base de données et manipulé par le SGBD.

Ces **fonctions** peuvent être exprimées sous forme de requêtes (anglais « query ») dans un langage informatique de manipulation et de communication de données reconnu par le SGBD : SQL (Structured Query Language).

Pour résumer, les fonctions dun SGBD permettent :

- La définition des données : on parle de Langage de définition des données (DDL) - (conforme à un modèle de données) ;
- La manipulation des données : interrogation, mise à jour (insertion, suppression, modification). On parle de Langage de manipulation des données (DML) - (langage de requête déclaratif) ;
- Le contrôle des données : contraintes d'intégrité, contrôle des droits d'accès, gestion de transactions. On parle de Langage de contrôle des données (DCL).

2. TYPOLOGIES DES SYSTSÈME DE GESTION DE BASES DE DONNÉES

2.1 LES TYPES DE SYSTÈMES DE GESTION DES BASES DE DONNÉES

Il y a différentes façons de classer les SGBD, en voici une reposant sur des aspects pratiques :

- Le SGBD relationnel (**SGBDR**) est le type de SGDB le plus fréquent actuellement. Ce type de SGDB repose sur la théorie mathématique des ensembles. Ce système fiable et robuste est dominé depuis trente (30) ans par de grands acteurs, comme Oracle, Microsoft et IBM. Il garantit de bonnes performances en ce qui concerne le stockage, les accès et la sécurité des données, mais avec l'avènement du Big Data, il s'avère rigide et limité, ce qui remet en cause

sa prédominance.

- Le SGBD NoSQL, plus flexible, ne demande pas de définir chaque donnée pour chaque entité, ce qui en fait une option intéressante pour gérer des bases de données importante, avec beaucoup de données éparses et dont la structure est susceptible d'évoluer au fil du temps. Il regroupe quatre (4) catégories de fonctions principales : 1/ clé-valeurs ; 2/ documents ; 3/ colonnes ; 4/ graphiques.
- Le SGBD in-Memory dont l'intérêt principal est de se passer de disque dur et de stocker les données dans sa mémoire. Il est beaucoup plus rapide pour traiter les opérations mais sa capacité et ses fonctionnalités sont limitées.
- Il existe d'autres systèmes, comme le SGBD pour les données XML/RDF, peu avantageux par rapport aux autres. Le système se caractérise par une base de données en colonnes, optimisé et rapide en lecture, mais pas en écriture. Il existe également des SGBD orientés objets, populaires dans les années 1990 ou enfin des systèmes hiérarchiques ou les systèmes de réseaux antérieurs aux SGBD relationnels.

Selon leur construction et les possibilités qu'ils offrent, les SGBD peuvent être dit **hiérarchiques**, **réseaux**, **relationnels**, **orientés objet**, **objet-relationnels**, **XML/RDF** ou **mixtes**. Ils peuvent être **distribués**, **centralisés** ou **embarqués** et peuvent être **spatiaux** :

- **Relationnel**. Selon ce modèle, les données sont placées dans des tables avec lignes et colonnes. N'importe quelle donnée contenue dans la base de données peut être retrouvée à l'aide du nom de la table, du nom de la colonne et de la clé primaire. Le modèle relationnel est destiné à assurer l'indépendance des données et à offrir les moyens de contrôler la cohérence et d'éviter la redondance. Il permet de manipuler les données comme des ensembles en effectuant des opérations de la théorie des ensembles. Les règles de cohérence qui s'appliquent aux bases de données relationnelles sont **l'absence de redondance ou de nul des clés primaires** et **l'intégrité** référentielle.
- **Hiérarchique.** Une base de données hiérarchique est une base de données dont le système de gestion lie les enregistrements dans une structure arborescente où chaque enregistrement n'a qu'un seul possesseur. Elle a été utilisée dans les premiers systèmes de gestion de base de données de type « mainframe ». Ce système de gestion a été inventé par la NASA.
- **Orienté objet** et **objet-relationnel.** Les SGBD orientés « objet » sont un sujet de recherche depuis 1980 lorsque sont apparus les premiers langages de programmation orientés « objet ». Ils sont destinés à offrir les fonctionnalités des SGBD à des langages orientés « objet » et permettre le stockage persistant des objets. Les objets sont manipulés en utilisant les possibilités natives des langages orientés « objet ». Une interface de programmation permet d'exploiter les fonctionnalités du SGBD. Celui-ci est équipé des mécanismes nécessaires pour permettre l'utilisation des possibilités d'encapsulation, d'héritage et de polymorphisme des langages de programmation orientée « objet ». Les SGBD objet-relationnel offrent à la fois les possibilités des SGBD orientés « objet » et ceux des SGBD relationnels.
- À base de **XML** ou **RDF.** Une base de données XML Native (NXD en anglais) est une base de données qui s'appuie sur le modèle de données fourni par XML. Elle utilise typiquement des langages de requête XML comme XPath ou XQuery. Une extension possible est une base RDF avec le langage d'interrogation SPARQL.
- **Mixte.** De tels SGBD utilisent les différents paradigmes évoqués ci-dessus.
- **Centralisé** ou **distribué.** Un SGBD est dit centralisé lorsque le logiciel contrôle l'accès à une base de données placée sur un ordinateur unique. Il est dit distribué lorsqu'il contrôle l'accès à des données qui sont dispersées entre plusieurs ordinateurs. Dans cette construction, un logiciel est placé sur chacun des ordinateurs et les différents ordinateurs utilisent des moyens de communication pour coordonner les opérations. Le fait que les informations sont

dispersées est caché à l'utilisateur et celles-ci sont présentées comme si elles se trouvaient à une seule place.

- **Embarqué.** Une base de données embarquée (anglais embedded) est un SGBD sous forme de composant logiciel qui peut être incorporé dans un logiciel applicatif. Contrairement à un SGBD client-serveur dans lequel un processus traite les requêtes, un modèle embarqué se compose de bibliothèques logicielles liées par liaison dynamique avec le logiciel qui utilise le SGBD. Dans ce type de SGBD, la base de données est souvent composée d'un fichier unique dont le format est identique quelles que soient les caractéristiques de l'ordinateur utilisé. Bien que le SGBD offre de nombreux avantages par rapport à un enregistrement sur fichier, ces derniers sont souvent préférés aux SGBD qui ont la réputation d'être des logiciels lourds, encombrants et compliqués à installer.
- **Spatial.** Les applications informatiques telles que les systèmes d'information géographiques et les outils de conception assistée par ordinateur utilisent des SGBD de type spatial. Ce type de logiciel permet le stockage d'informations géométriques telles que des points, des lignes, des surfaces et des volumes. Ils comportent des fonctions permettant de retrouver une information sur la base de caractéristiques géométriques telles que les coordonnées ou la dimension. Le langage de requête du SGBD permet la manipulation d'informations de géométrie tels que lignes, points ou polygones. Ce SGBD met en œuvre les algorithmes et les structures de fichiers nécessaires.

2.2 QUELQUES SYSTÈMES DE GESTION DE BASES DE DONNÉES

De nombreux Systèmes de Gestion de Base de Données différents sont disponibles. Vous trouverez dans le tableau ci-après les SGBD les plus populaires et les plus courants.

Tableau 1 : SGBD les plus populaires et les plus courants

Nom SGBD	Année	Éditeur	Caractéristiques	SQL	Licence
Microsoft Access	1992	Microsoft	Relationnel, pour particuliers et groupes de travail	Oui	Propriétaire
Microsoft SQL Server	1989	Microsoft	Entreprises, groupes de travail, particuliers, relationnel, distribué	Oui	Propriétaire
MySQL	1995	Oracle Corporation et MySQL AB	Centralisé, embarqué, distribué, pour entreprises, groupes de travail et particuliers, relationnel	Oui	GPL
Oracle Database	1979	Oracle Corporation	Entreprises, groupes de travail, particuliers, relationnel, spatial, distribué	Oui	Propriétaire
Oracle NoSQL Database	2011	Oracle Corporation	NoSQL	Non	Propriétaire
CouchDB	2010	CouchBase	NoSQL orientée documents	Non	Apache 2.0
Db2 (IBM)	1983	IBM	Pour entreprises, groupes de travail, particuliers	Oui	Propriétaire
IBM Informix	1981	IBM	Pour entreprises, groupes de travail, distribué	Oui	Propriétaire
MariaDB	2009	Monty Program AB	Pour entreprises, groupes de travail, particuliers	Oui	GPL
Sybase ASE	1987	Sybase Inc	Pour entreprises, groupes de travail, Relationnel	Oui	Propriétaire
MongoDB	2007	MongoDB	NoSQL orientée documents	Non	
PostgreSQL	1985	PostgreSQL Global Development Group	Entreprises, groupes de travail, particuliers, relationnel, distribué, Objets, Spatial	Oui	BSD
Firebird	1981	Firebird Corporation	Relationnel, centralisé, embarqué, pour groupes de travail et entreprises	Oui	MPL-1.1 (d) et IBM Public License
SQLite	2000	D. Richard Hipp	Système de gestion de base de données embarqué, moteur de base de données relationnelle accessible par le langage SQL	Oui	Domaine Publique

2.3 LE PARADIGME CLIENT / SERVEUR

Quand la base de données se trouve sur un serveur qui ne sert qu'à cela, et pour interagir avec cette base de données, il faut utiliser un logiciel « client » qui va interroger le serveur et transmettre la réponse que le serveur lui aura donnée. Le serveur peut être installé sur une machine différente du client ; c'est souvent le cas lorsque les bases de données sont importantes. Ce n'est cependant pas obligatoire.

La plupart des SGBD sont basés sur un modèle Client / serveur.

3. STRUCTURE DES BASES DE DONNÉES RELATIONNELS

Maintenant nous savons qu'une base de données est une collection de données relatives à un ou plusieurs domaines, il importe de savoir comment cette base de données est structurée (objet de cette partie).

La façon selon laquelle une base de données est structurée ou organisée **dépend du modèle de données utilisé**. Précédemment nous avons vu qu'il y a différents modèles de données (hiérarchique, réseau, relationnel et objet). Le modèle relationnel est le plus utilisé aujourd'hui et c'est à ce modèle que nous allons nous intéresser.

Selon le modèle relationnel, une base de données est composée de :

- Tables ;
- Colonnes ;
- Lignes ;
- Clés primaires ;
- Clés étrangères ;
- Contraintes d'intégrité.

Il est important de comprendre ces concepts, puis de découvrir les deux (2) formalismes utilisés pour représenter la structure d'une base de données : schéma d'une base de données.

3.1 DIFFÉRENTS ELEMENTS CONSTITUITIFS DE LA STRUCTURE D'UNE BASE DE DONNÉES

3.1.1 NOTION DE TABLE

Une table est un ensemble de données relatives à un même sujet (ou entité). Ainsi une table est structurée sous forme de tableau.

Comme l'indique la figure 3, une table est composée horizontalement d'un ensemble de lignes et verticalement d'un ensemble de colonnes. Les colonnes décrivent les propriétés relatives au sujet représentées par la table et les lignes correspondent aux occurrences du sujet.

Un autre terme utilisé également pour désigner une table est celui de la « relation ». Son utilisation se justifie par le fait que dans une table (ou relation), on trouve des données qui sont en relation avec un sujet donné.

Dans une base de données relationnelle, la table constitue la structure la plus importante car la manipulation des données se fait toujours à travers les tables : création, sélection, modification et suppression.

Exemple. La table « Article » regroupe les données relatives aux articles commercialisés dans un magasin. Chaque article est décrit par :

- **Code article** : code attribué de façon unique à chaque article ;
- **Désignation article** : nom courant de l'article ;

- **Prix unitaire** : prix de vente de l'article ;
- **Quantité stock** : quantité actuellement disponible pour un article.

Ainsi, la table « Article » peut être représentée selon la figure 3.

Figure 3 : Exemple de table

Code article	Désignation article	Prix unitaire	Quantité en stock
V10	Vis 50x3	40	2500
V20	Vis 20x2	20	1300
B100	Boulon 90x15	450	100
C60	Clou 60x2	5	5000

Remarques

- Il ne faut pas confondre « fichier » et « table ». Les données d'un fichier sont stockées dans un même et seul fichier alors que les données d'une table peuvent être stockées sur un ou plusieurs fichiers, comme on peut regrouper dans un même fichier les données de plusieurs tables.
- Il y a une indépendance entre la structure d'une table et son implémentation physique sur les supports de stockage (disque). C'est le Système de Gestion de Base de Données (SGBD) qui assure cette indépendance.
- Il y a un lien très étroit entre la notion de table et la notion d'ensemble en mathématiques. Une table est considérée comme un ensemble de lignes. Certains opérateurs ensemblistes sont applicables à tables telles que l'union, l'intersection et le produit cartésien.

3.1.2 NOTION DE COLONNE

Dans une table, une colonne correspond à une propriété élémentaire de l'objet décrit par cette table. Une colonne est décrite par :

- **Un nom.** C'est le nom de la colonne. Il est sous forme de code et il est généralement soumis aux mêmes règles de nommage des variables dans les langages de programmation.
- **Un type de données**. C'est le type de données prises par cette colonne. Les types de données les plus connus sont : numérique, chaîne de caractères (ou texte), date et booléen. Certains Systèmes de Gestion de Base de Bonnées supportent des types de données multimédias telles que les images, le son et la vidéo.
- **Une taille éventuelle**. Pour certains types de données tels que le type numérique ou chaînes de caractères, la taille indique la longueur maximale que peut prendre la colonne.
- **Un indicateur de présence obligatoire**. Cet indicateur de présence indique si cette colonne doit être toujours renseignée ou peut être vide dans certains cas. Lorsque la colonne n'est pas renseignée, on dit qu'elle contient une valeur nulle. Il est à noter que la valeur nulle est différente de zéro pour les colonnes de type numérique et de la chaîne vide pour les chaînes de caractères.
- **Une valeur par défaut éventuelle.** Il s'agit d'attribuer une valeur par défaut lorsqu'aucune valeur n'a été attribuée à cette colonne.
- **Une règle éventuelle indiquant les valeurs autorisées**. Dans certains cas, une colonne peut être soumise à certaines règles. Par exemple, les valeurs attribuées à cette colonne doivent être inférieures à une certaine valeur, supérieures à une certaine valeur ou bien comprises entre deux valeurs.

Un autre terme utilisé également pour désigner une colonne est celui « **d'attribut** » ou de « **champ** ».

La qualité de la description d'une table dépend du choix des colonnes et de la précision dans la description de ces colonnes.

Exemple. Nous avons vu dans l'exemple précédent que la table « Article » (figure 3) regroupe les quatre (4) colonnes suivantes : code article, désignation article, prix unitaire et quantité en stock. La description détaillée de chacune de ces colonnes à travers le tableau 2 permet de bien comprendre la notion de colonne. Il est à noter que le nom de la colonne est un code et que nous avons rajouté une colonne pour donner une description de chaque colonne.

Tableau 2 : Notion de colonne

Nom de la table						
Description : Détail des articles commercialisés						
Nom colonne	**Description**	**Type de données**	**Taille**	**Obligatoire**	**Valeur par défaut**	**Valeurs automatisées**
Code_art	Code de l'article	Chaîne de caractères	20	Oui		
Des_art	Désignation de l'article	Chaîne de caractères	50	Oui		
PU	Prix unitaire de l'article	Numérique	8,3	Non		> 0
Qte_stok	Quantité en stock	Numérique	4	Non	0	≥ 0

3.1.3 NOTION DE LIGNE

Une table est composée horizontalement d'un ensemble de lignes. Une ligne correspond à une occurrence du sujet représenté par la table. On dit aussi qu'elle correspond à un objet du monde réel.

Une table est initialement vide lorsqu'elle est créée, c'est-à-dire qu'elle ne contient aucune ligne. L'exploitation d'une table consiste à y insérer de nouvelles lignes, à modifier certaines lignes ou à consulter certaines lignes en fonction de critères bien déterminés. Une ligne peut être supprimée lorsqu'on estime qu'on n'en a plus besoin.

Le nombre de lignes d'une table peut être limité à quelques dizaines comme il peut atteindre des milliers, voire des millions de lignes.

Un autre terme utilisé également pour désigner une ligne est celui de l'« **enregistrement** » ou de « **n-uplet** ».

3.1.4 NOTION DE CLÉ PRIMAIRE

La clé primaire d'une table est une colonne ou un groupe de colonnes permettant **d'identifier de façon unique chaque ligne de la table**. Autrement dit, la connaissance de la valeur de la clé primaire permet de connaître sans aucune ambiguïté les valeurs des autres colonnes de la table.

Exemple. Dans la table « Article », la colonne « Code article » peut être utilisée comme clé primaire car la codification des articles est faite de telle sorte qu'on ne peut pas avoir deux articles qui ont le même code. Connaissant le code article, on peut déterminer de façon unique sa désignation, son prix unitaire et sa quantité en stock.

Remarques

- Chaque table doit comporter une et **une seule clé primaire**.
- Dans certains cas, dans une même table, on peut avoir deux ou plusieurs colonnes qui peuvent jouer le rôle de clé primaires. Dans ce cas on doit choisir une colonne parmi toutes ces colonnes. Par exemple, dans la table « article » si on suppose que la désignation est unique pour tous les articles, nous pouvons utiliser comme clé primaire, indifféremment, le code ou la désignation de l'article.
- Les colonnes qui constituent la clé primaire sont obligatoires.
- Pour distinguer une colonne qui fait partie de la clé primaire des autres colonnes, on la souligne ou on la met en **gras**.

3.1.5 LIENS ENTRE TABLES – CLE ETRANGERE

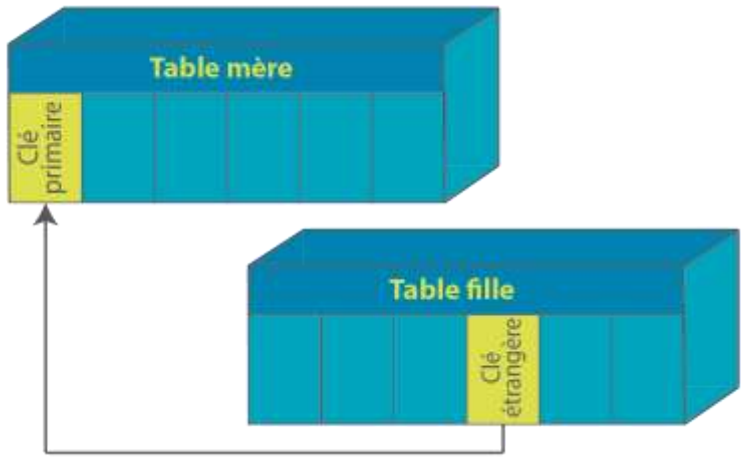
Une base de données est la représentation d'un ou plusieurs domaines composés chacun d'un ensemble de sujets (ou entité). Les différents sujets de chaque domaine sont généralement inter-reliés par des liens (ou associations). Pour que la base de données constitue une représentation fidèle du ou des domaines concernés, les liens entre les sujets du monde réel doivent se retrouver dans la base de données.

Si nous avons une table représentant les élèves et une autre table représentant les lycées, la phrase suivante « Un élève est inscrit dans un lycée » correspond à un lien (ou association) entre ces deux (2) tables.

Un lien entre deux tables « A » et « B » est représenté par l'ajout dans la table « B » d'une nouvelle colonne correspondant à la clé primaire de la table « A ». Cette nouvelle colonne est appelée **clé étrangère**.

Un lien entre deux (2) tables est orienté : il part de la table contenant la clé étrangère et arrive vers la table contenant la clé primaire. La table cible (celle contenant la clé primaire) s'appelle **table mère** et **la table source** (celle contenant la clé étrangère) s'appelle **table fille**. On dit aussi que la table fille se **réfère** à la table mère.

Figure 4 : Liens entre table, la clé étrangère



Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition

Supposons qu'en plus de la table « Article », nous avons une table « Commande » qui décrit les commandes reçues par le magasin. Dans cet exemple, nous admettons l'hypothèse qu'une commande ne concerne qu'un seul article. La figure 5 représente ainsi ces deux tables avec le lien qui existe entre elles.

Figure 5 : Liens entre deux tables

Articles

Code article	Désignation article	Prix unitaire	Quantité en stock
V10	Vis 50x3	40	200
V20	Vis 20x2	20	1300
B100	Boulon 90x15	450	100
C60	Clou 60x2	5	5000

N° Commande	Date de commande	Code article#	Quantité commandée
100	01/03/2006	V10	500
101	15/04/2006	B100	30
102	17/04/2006	V10	120

Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition

3.1.6 INDEX

Un index permet **d'accélérer l'accès aux données d'une table**. Le but principal d'un index est d'éviter de parcourir une table séquentiellement du premier enregistrement jusqu'à celui visé que l'on recherche dans une table non indexée. Le principe d'un index est l'association de l'adresse de chaque enregistrement avec la valeur des colonnes indexées.

3.1.7 INTÉGRITÉ RÉFÉRENTIELLE – CONTRAINTE D'INTÉGRITÉ

Assurer la cohérence et l'intégrité des données est l'une des principales fonctions du Système de Gestion de Base de Données (SGBD). Cela consiste à garantir que les données stockées dans une base de données sont cohérentes entre elles, c'est-à-dire qu'elles respectent toutes les règles exigées par le concepteur de la base.

La cohérence et l'intégrité des données sont **assurées à l'aide d'un ensemble de règles dites contraintes d'intégrité**. Une contrainte d'intégrité est une règle appliquée à une colonne ou à une table et qui doit être toujours vérifiée.

Les principaux types de contraintes d'intégrité sont :

- **Les contraintes de domaines**. Ce sont des contraintes appliquées à des colonnes. Elles permettent de fixer le caractère obligatoire ou pas d'une colonne et les règles de validité des valeurs qui peuvent être prises par cette colonne. Par exemple, la note obtenue dans une matière doit être comprise entre zéro (0) et vingt (20). Pour notre exemple, la quantité commandée dans la table « Commande » est obligatoire et doit être positive et supérieure à zéro.
- **Les contraintes d'intégrité de tables.** Elles permettent d'assurer que chaque table a une clé primaire.
- **Les contraintes d'intégrité référentielles.** Elles permettent de s'assurer que les valeurs introduites dans une colonne figurent dans une autre colonne en tant que clé primaire. Ces contraintes sont représentées sous forme de liens entre tables.

3.2 REPRÉSENTATION D'UN SCHÉMA DE BASE DE DONNÉES

Après avoir décrit les différentes structures qui constituent une base de données, il reste à donner un formalisme permettant de représenter de façon homogène tous ces concepts.

Un schéma de base de données représente la configuration logique de tout ou partie d'une base de données relationnelle. Il peut se présenter à la fois sous la forme d'une représentation visuelle et d'un ensemble de formules, appelées « contraintes d'intégrité », qui régissent une base de données. Ces formules sont exprimées dans un langage de définition des données, tel que SQL. Dans le cadre d'un dictionnaire de données, un schéma de base de données indique comment les entités qui composent la base de données se rapportent les unes aux autres, notamment les tables, les vues et les procédures stockées.

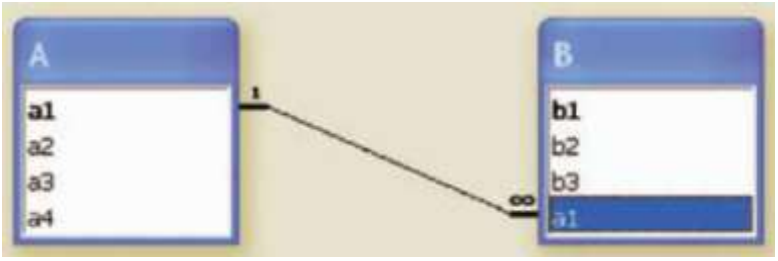
La structure d'une base de données peut être représentée selon deux formalismes, la représentation textuelle et la représentation graphique (figures 6 et 7).

Figure 6 : Représentation textuelle d'une structure de Base de Données

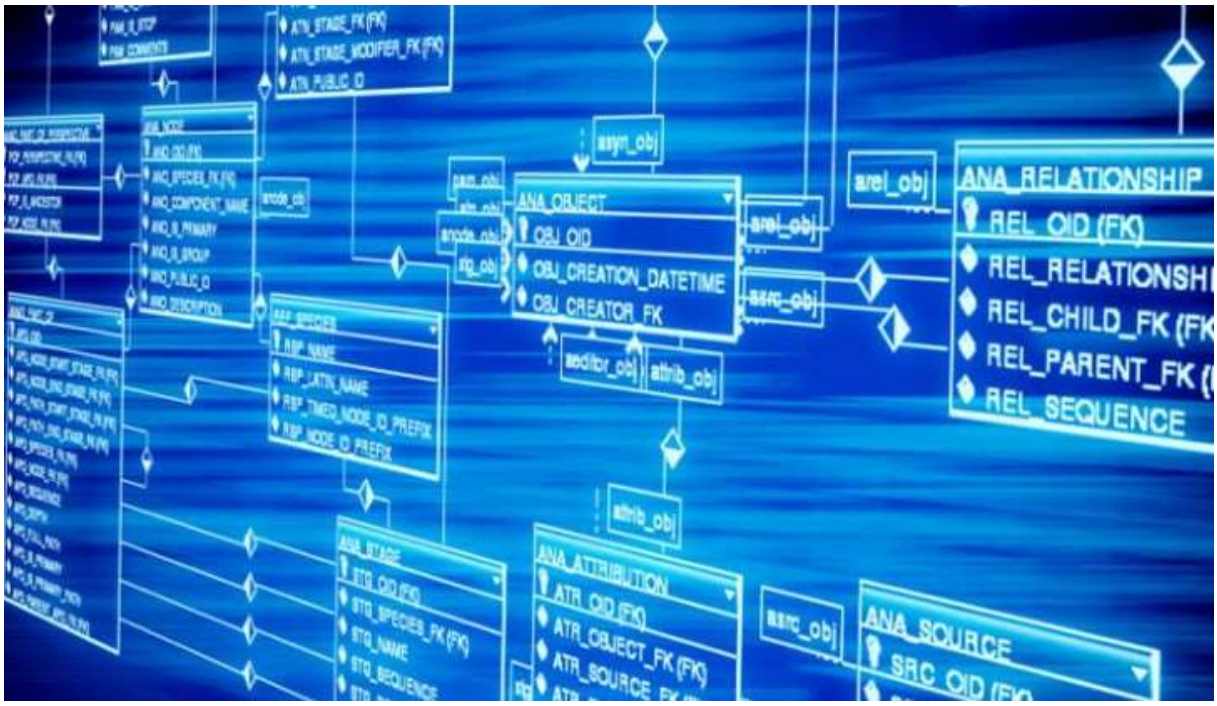
A (a1, a2, a3, a4)
B (b1, b2, b3, a1#)

Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition

Figure 7 : Représentation graphique d'une structure de Base de Données



Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition



MODULE 2 : SQL POUR MYSQL

Ayant abordé les notions essentielles et caractéristiques d'un Système de Gestion de Base de Données (SGBD), ce module met l'accent sur l'installation de SGBDR MySQL sur un poste d'ordinateur et son utilisation. Plus spécifiquement le module prévoit :

- Des travaux dirigés d'installation et de configuration de MySQL ;
- La connexion et déconnexion au client MySQL ;
- La céation d'un utilisateur ;
- Les bases de la syntaxe du langage SQL ;
- Les types de données offertes par le SGBDR MySQL ;
- Les différents types de requêtes SQL CRUD (Create, Read, Update, Delete) ;
- La gestion des utilisateurs ;
- Les privilèges ;
- Les accès à distant.

A titre de rappel, il existe plusieurs manières d'utiliser MySQL. La première utilisation recommandée est l'utilisation en ligne de commande.

4. TRAVAUX DIRIGÉS D'INSTALLATION ET DE CONFIGURATION DE MYSQL

4.1 INSTALLATION ET CONFIGURATION DE MYSQL

Pour télécharger MySQL, vous pouvez vous rendre sur le site suivant :

http://dev.mysql.com/downloads/mysql/#downloads

Sélectionnez l'OS sur lequel vous travaillez. Généralement, l'OS Windows est le plus courant. Téléchargez MySQL avec l'installeur (MSI Installer), puis exécutez le fichier téléchargé. L'installeur démarre et vous guide lors de l'installation.

L'installeur Windows nécessite d'avoir le framework.NET 4 installé. Si vous ne l'avez pas, l'installeur vous le signalera. Vous pouvez alors télécharger ce framework sur le site de Microsoft. Une fois celui-ci installé et votre ordinateur redémarré, vous pouvez relancer l'installeur MySQL.

Sélectionnez « Install MySQL products » et acceptez la licence. L'installeur vous propose plusieurs types d'installations possibles. Si vous laissez l'option par défaut, cela installera une flopée de programmes, non nécessaires pour ce cours. Il est conseillé donc de choisir l'installation « Custom ». Puis, à l'écran suivant, sélectionnez « MySQL Server 5.7.32 » (ou autre version). La documentation peut être utile si vous avez un jour besoin de la consulter hors ligne.

Figure 8 : Installation Windows - Choix du type d'installation

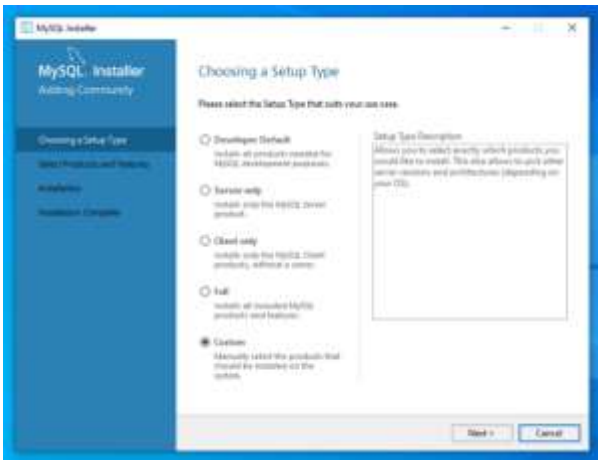
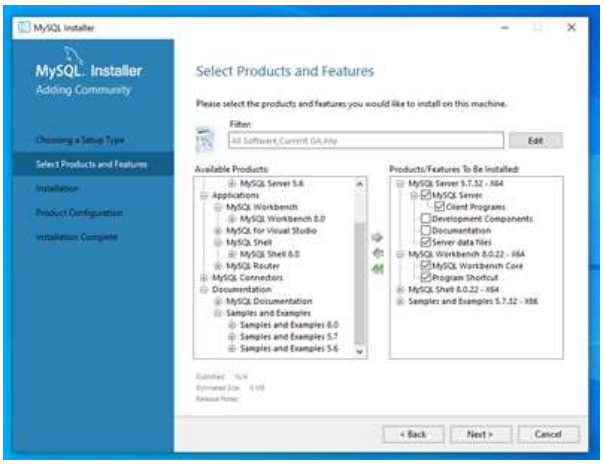


Figure 9 : Installation Windows - Éléments à installer



Lancez l'installation ! Une fois l'installation terminée, vous passez à l'étape de configuration. Laissez les options par défaut. La seconde étape de configuration vous permet de choisir un mot de passe pour l'utilisateur par défaut de MySQL, l'utilisateur « root ». Vous pouvez laisser le mot de passe vide, mais c'est déconseillé.

L'installation est maintenant terminée. Ouvrez maintenant le programme de ligne de commande « Command Prompt » (ou « Invite de commande »).

Pour pouvoir utiliser les programmes clients de MySQL, il faut que l'invite de commande sache où se trouvent ces programmes. On va donc ajouter le chemin vers MySQL aux dossiers explorés par l'invite de commande.

Utilisez bien l'invite de commande et pas l'utilitaire « MySQL 5.7 Command Line Client », installé en même temps que MySQL. Nous verrons bientôt ce qu'est cet utilitaire et comment l'utiliser.

Vérifiez le chemin vers MySQL dans votre explorateur, ce devrait être à peu près C:\Program Files\MySQL\MySQL Server 5.7\bin (le « bin » est important, c'est là que se trouvent les programmes clients).

Exécutez la commande (figure 10) dans l'invite de commande.

Figure 10 : Commande dans l'invite de commande MySQL

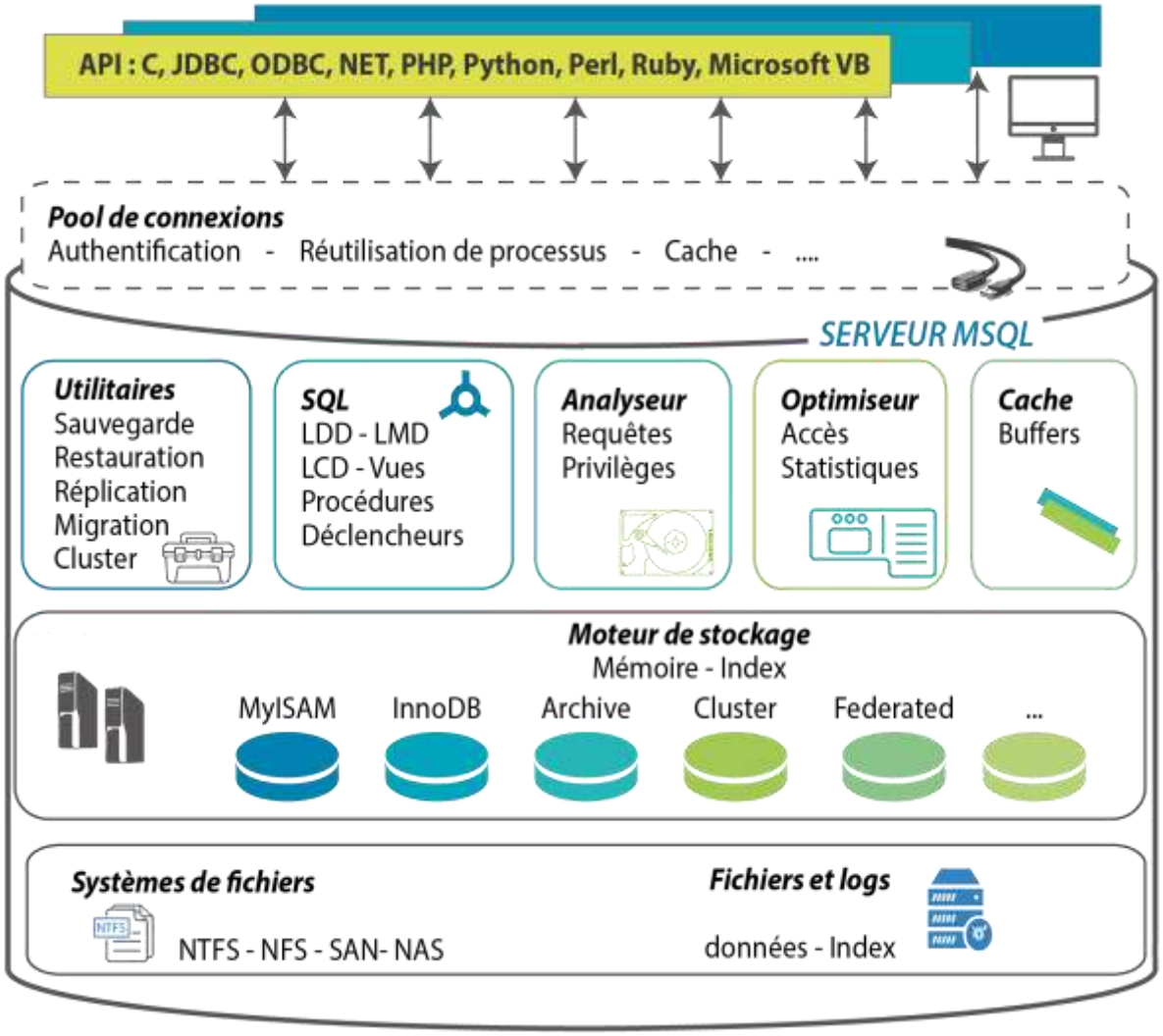
```
Microsoft Windows [version 10.0.19042.867]
(c) 2020 Microsoft Corporation. Tous droits réservés.

C:\Users\Godsent> set PATH=%PATH% ; ''C : \Program Files\MySQL\MySQL    Server 5.7\bin"
```

Fermer la fenetre de l'invite de commande

4.2 FONCTIONNALITÉS DU MOMENT DE MYSQL (VERSION 5.7)

La figure suivante présente la majeure partie des fonctionnalités de MySQL qui se positionnent au sein du serveur de données (SGBD).

Figure 11 : Fonctionnalités de MySQL au sein du serveur de données



Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition

4.3 INTERFACE DE COMMANDE

Il existe plusieurs manières d'utiliser MySQL :

- Utilisation en **ligne de commande.** Il s'agit d'une fenêtre toute simple (invite de commande de Windows) dans laquelle toutes les instructions sont tapées à la main. Pas de bouton, pas de zone de saisie. Juste votre clavier.
- Utilisation **d'interface graphique.** L'interface graphique permet d'exécuter pas mal de choses simples de manière intuitive sur une base de données. Comme interface graphique pour MySQL, on peut citer MySQL Workbench, PhpMyAdmin (souvent utilisé pour créer un site web en combinant MySQL et PHP) ou MySQL Front par exemple.

Pourquoi utiliser la ligne de commande ?

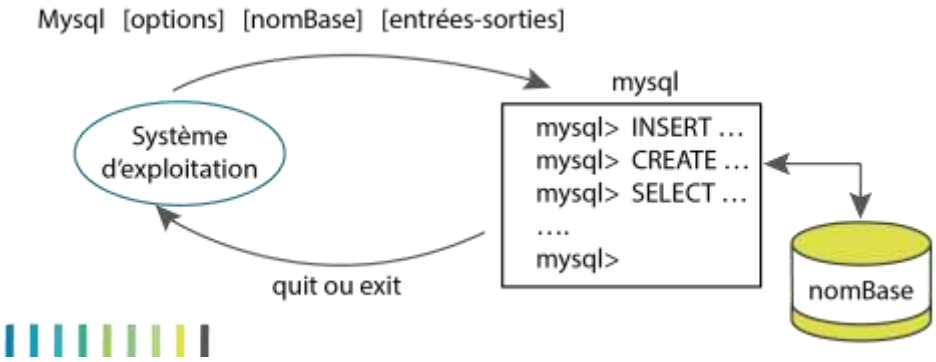
Deux raisons expliquent l'utilisation de la ligne de commande :

- Premiérement, parce que l'on **souhaite que vous maîtrisiez vraiment les commandes**. En effet, les interfaces graphiques permettent de faire pas mal de choses, mais une fois que vous serez bien lancés, vous vous mettrez à faire des choses subtiles et compliquées et il ne serait pas étonnant qu'il vous soit obligatoire d'écrire vous-même vos requêtes ;

- Ensuite, parce qu'il est fort probable que **vous désiriez utiliser MySQL en combinaison avec un autre langage de programmation** (si ce n'est pas votre but immédiat, cela viendra probablement un jour). Or, dans du code PHP (ou Java, ou Python, etc.), on ne va pas écrire « Ouvre PhpMyAdmin et clique sur le bon bouton pour que je puisse insérer une donnée dans la base ». On va devoir écrire en « dur » les requêtes. Il faut donc que vous sachiez comment faire.

Le principe général de l'interface de commande en ligne est le suivant : après une connexion locale ou distante, des instructions sont saisies et envoyées à la base qui retourne des résultats affichés dans la même fenêtre de commande.

Figure 12 : Principe général de l'interface de commande en ligne



Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition

4.4 SYNTAXE SQL ET PREMIÈRES COMMANDES

Maintenant que vous savez vous connecter, vous allez enfin pouvoir discuter avec le serveur MySQL (en langage SQL évidemment). Donc, reconnectez-vous si vous êtes déconnecté.

4.4.1 LE LANGAGE SQL

4.4.1.1 Définition

Le langage SQL (Structured Query Language) peut être considéré comme le langage d'accès standard et normalisé, destiné à interroger ou à manipuler une base de données relationnelle.

Le langage SQL a fait l'objet de plusieurs normes ANSI/ISO. Les instructions SQL sont regroupées en catégories en fonction de leur utilité et des entités manipulées. Nous pouvons distinguer cinq (5) catégories qui permettent :

- Un langage de définition de données (LDD) : langage orienté au niveau de la structure de la base de données. Le LDD permet de créer modifier, supprimer des objets ;
- Un langage de manipulation de données (LMD) : l'ensemble des commandes concernant la manipulation des données dans une base de données. Le LMD permet l'ajout, la suppression et la modification de lignes ;
- Un langage de contrôle de données (LCD) : l'ensemble des commandes pour contrôler l'accès aux données d'une base de données ;
- Un langage de contrôle des transactions (LCT) : l'ensemble des commandes utilisées pour le contrôle transactionnel dans une base de données, c'est-à-dire les caractéristiques des transactions, la validation et l'annulation des modifications ;
- Et d'autres modules destinés notamment à écrire des routines (procédures, fonctions ou déclencheurs) et interagir avec des langages externes.

4.4.1.2 CRUD : La base de la gestion de données

Le terme « CRUD » est étroitement lié avec la gestion des données. Plus précisément, CRUD est un acronyme des noms des quatre (4) opérations de base de la gestion de la persistance des données et applications :

- Create (créer) ;
- Read ou Retrieve (lire) ;
- Update (mettre à jour) ;
- Delete ou Destroy (supprimer).

Plus simplement, le terme « CRUD » résume les fonctions qu'un utilisateur a besoin d'utiliser pour créer et gérer des données. Divers processus de gestion des données sont basés sur « CRUD ». Cependant les opérations sont spécifiquement adaptées aux besoins des systèmes et des utilisateurs, que ce soit dans la gestion des bases de données ou pour l'utilisation des applications. Ainsi, les opérations sont des outils d'accès classiques et indispensables avec lesquels les experts, peuvent par exemple, vérifier les problèmes de base de données. Tandis que pour un utilisateur, « CRUD » signifie la création d'un compte (create), l'utilisation à tout moment (read), la mise à jour (update) ou encore la suppression (delete). En fonction de l'environnement de langage informatique, les opérations « CRUD » sont exécutées différemment, comme indiqué dans le tableau suivant/

Tableau 3 : les opérations « CRUD » en fonction de l'environnement de langage informatique

CRUD-Operation	SQL	RESTful HTTP	XQuery
Create	INSERT	POST, PUT	insert
Read	SELECT	GET, HEAD	copy/modify/return
Update	UPDATE	PUT, PATCH	replace, rename
Delete	DELETE	DELETE	delete

4.4.2 SYNTAXE

Avant d'aller plus loin, voici quelques règles générales à retenir concernant le SQL qui, comme tout langage informatique, **obéit à des règles syntaxiques très strictes**.

Fin d'une instruction

Pour signifier à MySQL qu'une instruction est terminée, il faut mettre le caractère « ; ». Tant qu'il ne rencontre pas ce caractère, le client MySQL pense que vous n'avez pas fini d'écrire votre commande et attend gentiment que vous continuiez. Par exemple, la commande suivante devrait afficher 100. Mais tant que MySQL ne recevra pas de « ; », il attendra simplement la suite.

Commentaires

Les commentaires sont des parties de code qui ne sont pas interprétées. Ils servent principalement à vous repérer dans votre code. En SQL, les commentaires sont introduits par -- (deux tirets). Cependant, MySQL déroge un peu à la règle SQL et accepte deux syntaxes : # : tout ce qui suit ce caractère sera considéré comme commentaire. -- : la syntaxe normale est acceptée uniquement si les deux tirets sont suivis d'une espace au moins. Afin de suivre au maximum la norme SQL, ce sont les -- qui seront utilisés tout au long de ce cours.

Chaînes de caractères

Lorsque vous écrivez une chaîne de caractères dans une commande SQL, il faut absolument l'entourer de guillemets simples (donc des apostrophes).

MySQL permet également l'utilisation des guillemets doubles, mais ce n'est pas le cas de la plupart des SGBDR. Pour ne pas prendre de mauvaises habitudes, il est conseillé donc de n'utiliser que les guillemets simples pour délimiter vos chaînes de caractères.

Par ailleurs, si vous désirez utiliser un caractère spécial dans une chaîne, il vous faudra l'échapper avec « \ ». Par exemple, si vous entourez votre chaîne de caractères de guillemets simples mais voulez utiliser un tel guillemet à l'intérieur de votre chaîne :

```
SELECT 'Salut l'ami'; -- Pas bien !
SELECT 'Salut l\'ami'; -- Bien !
```

Tableau 4 : Quelques autres caractères spéciaux de la syntaxe MySQL

\n	retour à la ligne
\t	tabulation
\\	antislash (eh oui, il faut échapper le caractère d'échappement…)
%	pourcent (vous verrez pourquoi plus tard)
_	souligné (vous verrez pourquoi plus tard aussi)

Cette manière d'échapper les caractères spéciaux (avec \) est propre à MySQL. D'autres SGBDR demanderont qu'on leur précise quel caractère sert à l'échappement ; d'autres encore demanderont de doubler le caractère spécial pour l'échapper. Soyez donc prudent et renseignez-vous si vous n'utilisez pas MySQL.

Pour échapper un guillemet simple (et uniquement ce caractère), vous pouvez également l'écrire deux fois. Cette façon d'échapper les guillemets correspond d'ailleurs à la norme SQL. Nous vous encourageons par conséquent à essayer de l'utiliser au maximum.

```
SELECT 'Salut l'ami';  -- ne fonctionne pas !
SELECT 'Salut l\'ami'; -- fonctionne !
SELECT 'Salut l''ami'; -- fonctionne aussi et correspond à la norme !
```

4.4.3 CONVENTIONS

Mots-clés

Une convention largement répandue veut que les commandes et mots-clés SQL soient écrits complètement **en majuscules**. Nous allons respecter cette convention et nous vous encourageons à le faire également. Il est plus facile de relire une commande de 5 lignes lorsque l'on peut différencier au premier coup d'œil les commandes SQL, des noms de tables et de colonnes.

Noms de bases, de tables et de colonnes

Comme nous venons de le dire, les mots-clés SQL seront écrits en majuscule pour les différencier du reste, donc évidemment, les noms de bases, tables et colonnes seront écrits en **minuscules**.

Toutefois, par habitude et par clarté, il est préférable de mettre une majuscule à la première lettre de noms de tables (et uniquement pour les tables : ni pour la base de données, ni pour les colonnes). Notez que MySQL n'est pas nécessairement sensible à la casse en ce qui concerne les noms de tables et de colonnes. En fait, il est très probable que si vous travaillez sous Windows, MySQL ne soit pas sensible à la casse pour les noms de tables et de colonnes. Sous Mac et Linux, par contre, c'est le contraire qui est le plus probable.

Quoi qu'il en soit, il est conseillé d'utiliser des majuscules pour la première lettre des noms de tables, mais libre à vous de daire ce choix ou non.

Options facultatives

Lorsque vous allez voir les commandes SQL et commencer à les utiliser pour interagir avec votre base de données, vous verrez que certaines commandes ont des options facultatives. Dans ces cas-là, nous utliserons des crochets [ ] pour indiquer ce qui est facultatif. La même convention est utilisée dans la documentation officielle MySQL (et dans beaucoup d'autres documentations d'ailleurs). La requête suivante signifie donc que vous pouvez commander votre glace vanille toute seule ou avec du chocolat ou avec de la chantilly ou avec du chocolat ET de la chantilly.

```
COMMANDE glace vanille [avec chocolat] [avec chantilly]
```

4.4.4 CRÉATION D'UN UTILISATEUR

Il n'est pas très conseillé de travailler en tant que « root » dans MySQL, à moins d'en avoir spécifiquement besoin. En effet, « root » a tous les droits. Ceci signifie que vous pouvez faire n'importe quelle bêtise dans n'importe quelle base de données. Pour éviter cela, il est préférable de créer maintenant un utilisateur MySQL.

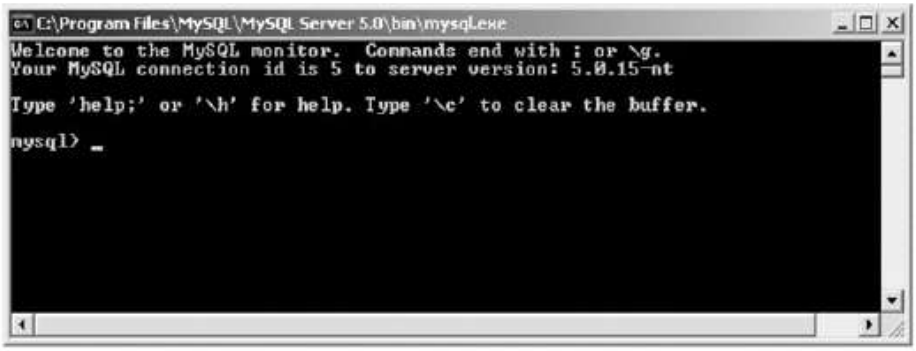
Ouvrez le fichier « *premierPas.sql* » qui se trouve dans le répertoire Exemples SQL, à l'aide du bloc-notes (ou d'un éditeur de texte de votre choix). Changez « util » par le nom de l'utilisateur à créer (modifiez aussi le nom de la base). Vous pouvez changer le mot de passe si vous voulez. Enregistrez ce fichier dans un de vos répertoires.

4.4.5 CONNEXION AU SERVEUR ET AUTRES COMMANDES DE BASE

Dans une fenêtre de commande Windows, lancez l'interface en ligne en connectant l'utilisateur « root » avec le mot de passe que vous avez donné lors de l'installation.

Figure 13 : Lancement de l'interface en ligne dans une fenêtre de commande Windows

```
mysql --user=root -p
```



Une fois connecté, par copier-coller (en effectuant un clic droit dans la fenêtre de commande MySQL), exécutez une à une les différentes instructions (création de la base, de l'utilisateur, des privilèges et déconnexion de root). Nous étudierons les notions élémentaires de droits et de sécurité (partie 9). Les lignes précédées de « -- » sont des commentaires.

Voilà, votre utilisateur (util) est créé. Il peut se connecter et il possède toutes les prérogatives sur la base (bdutil) pour exécuter les instructions décrites dans ce document. Pour tester votre connexion, lancez la commande suivante qui se connecte au serveur sur la base bdutil, sous l'utilisateur util.

```
mysql --user=util --host=localhost -p --database=bdutil
```

Vérification de la version

```
mysql --version
```

En ce qui concerne les principales options au lancement de MySQL et les principales options de l'invite de commandes, nous vous invitons à les consulter dans la documentation officielle sur le site de MySQL.

5. TYPES DE DONNÉES DANS MYSQL

Pour décrire les colonnes d'une table, MySQL fournit les types prédéfinis suivants :

- Caractères (CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT) ;
- Valeurs numériques (TINYINT, SMALLINT, MEDIUMINT, INT, INTEGER, BIGINT, FLOAT, DOUBLE, REAL, DECIMAL, NUMERIC, et BIT) ;
- Date/heure (DATE, DATETIME, TIME, YEAR, TIMESTAMP) ;
- Données binaires (BLOB, TINYBLOB, MEDIUMBLOB, LONGBLOB) ;
- Énumérations (ENUM, SET).

5.1 CARACTÈRES

Le type **CHAR** permet de stocker des chaînes de caractères de taille fixe. Les valeurs sont stockées en ajoutant, s'il le faut, des espaces à concurrence de la taille définie. Ces espaces ne seront pas considérés après extraction à partir de la table.

Le type **VARCHAR** permet de stocker des chaînes de caractères de taille variable. Les valeurs sont stockées sans l'ajout d'espaces à concurrence de la taille définie. Depuis la version 5.0.3 de MySQL, les éventuels espaces de fin de chaîne seront stockés et extraits en conformité avec la norme SQL. Des caractères Unicode (méthode de codage universelle qui fournit une valeur de code unique pour chaque caractère quels que soient la plate-forme, le programme ou la langue) peuvent aussi être stockés.

Les types **BINARY** et **VARBINARY** sont similaires à CHAR et VARCHAR, excepté par le fait qu'ils contiennent des chaînes d'octets sans tenir compte d'un jeu de caractères en particulier. Les quatre (4) types permettant aussi de stocker du texte sont **TINYTEXT**, **TEXT**, **MEDIUMTEXT**, et **LONGTEXT**. Ces types sont associés à un jeu de caractères. Il n'y a pas de mécanisme de suppression d'espaces de fin, ni de possibilité d'y associer une contrainte DEFAULT.

Tableau 5 : Les types de caractères en langage MySQL

Type	Description	Commentaire pour une colonne
`CHAR(n) [BINARY | ASCII | UNICODE]`	Chaîne fixe de *n* octets ou caractères.	Taille fixe (maximum de 255 caractères).
`VARCHAR(n) [BINARY]`	Chaîne variable de *n* caractères ou octets.	Taille variable (maximum de 65 535 caractères).
`BINARY(n)`	Chaîne fixe de *n* octets.	Taille fixe (maximum de 255 octets).
`VARBINARY(n)`	Chaîne variable de *n* octets.	Taille variable (maximum de 255 octets).
`TINYTEXT(n)`	Flot de *n* octets.	Taille fixe (maximum de 255 octets).
`TEXT(n)`	Flot de *n* octets.	Taille fixe (maximum de 65 535 octets).
`MEDIUMTEXT(n)`	Flot de *n* octets.	Taille fixe (maximum de 16 mégaoctets).
`LONGTEXT(n)`	Flot de *n* octets.	Taille fixe (maximum de 4,29 gigaoctets).

5.2 DATES ET HEURES

Les types suivants permettent de stocker des moments ponctuels (dates, dates et heures, années, et heures). Les fonctions **NOW()** et **SYSDATE()** retournent la date et l'heure courantes. Dans une procédure ou un déclencheur **SYSDATE** est réévaluée en temps réel, alors que **NOW** désignera toujours l'instant de début de traitement.

Tableau 6 : Les types dates et heures en langage MySQL

Type	Description	Commentaire pour une colonne
`DATE`	Dates du 1er janvier de l'an 1000 au 31 décembre 9999 après J.-C.	Sur 3 octets. L'affichage est au format `'YYYY-MM-DD'`.
`DATETIME`	Dates et heures (de 0 h de la première date à 23 h 59 minutes 59 secondes de la dernière date).	Sur 8 octets. L'affichage est au format `'YYYY-MM-DD HH:MM:SS'`.
`YEAR[(2|4)]`	Sur 4 positions : de 1901 à 2155 (incluant 0000). Sur 2 positions : de 70 à 69 (désignant 1970 à 2069).	Sur 1 octet ; l'année est considérée sur 2 ou 4 positions (4 par défaut). Le format d'affichage est `'YYYY'`.
`TIME`	Heures de -838 h 59 minutes 59 secondes à 838 h 59 minutes 59 secondes.	L'heure au format `'HHH:MM:SS'` sur 3 octets.
`TIMESTAMP`	Instants du 1er Janvier 1970 0 h 0 minute 0 seconde à l'année 2037.	Estampille sur 4 octets (au format `'YYYY-MM-DD HH:MM:SS'`) ; mise à jour à chaque modification sur la table.

5.3 VALEURS NUMÉRIQUES

De nombreux types de valeurs numériques sont proposés par MySQL pour définir des valeurs exactes (entiers ou décimaux, positifs ou négatifs : **INTEGER** et **SMALLINT**) et des valeurs à virgule fixe ou flottante (**FLOAT**, **DOUBLE** et **DECIMAL**). En plus des spécifications de la norme SQL, MySQL propose les types d'entiers restreints (**TINYINT**, **MEDIUMINT** et **BIGINT**). Le tableau suivant décrit ces types.

Tableau 7 : Les types valeurs numériques en langage MySQL

Type	Description
`BIT[(n)]`	Ensemble de *n* bits. Taille de 1 à 64 (par défaut 1).
`TINYINT[(n)] [UNSIGNED] [ZEROFILL]`	Entier (sur un octet) de -128 à 127 signé, 0 à 255 non signé.
`BOOL` et `BOOLEAN`	Synonymes de `TINYINT(1)`, la valeur zéro est considérée comme fausse. Le non-zéro est considéré comme vrai. Dans les prochaines versions, le type *boolean*, comme le préconise la norme SQL, sera réellement pris en charge.
`SMALLINT[(n)] [UNSIGNED] [ZEROFILL]`	Entier (sur 2 octets) de – 32 768 à 32 767 signé, 0 à 65 535 non signé.
`MEDIUMINT[(n)] [UNSIGNED] [ZEROFILL]`	Entier (sur 3 octets) de – 8 388 608 à 8 388 607 signé, 0 à 16 777 215 non signé.
`INTEGER[(n)] [UNSIGNED] [ZEROFILL]`	Entier (sur 4 octets) de –2 147 483 648 à 2 147 483 647 signé, 0 à 4 294 967 295 non signé.
`BIGINT[(n)] [UNSIGNED] [ZEROFILL]`	Entier (sur 8 octets) de – 9 223 372 036 854 775 808 à 9 223 372 036 854 775 807 signé, 0 à 18 446 744 073 709 551 615 non signé.
`FLOAT[(n[,p])] [UNSIGNED] [ZEROFILL]`	Flottant (de 4 à 8 octets) *p* désigne la précision simple (jusqu'à 7décimales) de $-3.4\ 10^{+36}$ à $-1.1\ 10^{-38}$, 0, signé, et de $1.1\ 10^{-38}$ à $3.4\ 10^{+38}$ non signé.
`DOUBLE[(n[,p])] [UNSIGNED] [ZEROFILL]`	Flottant (sur 8 octets) *p* désigne la précision double (jusqu'à 15 décimales) de $-1.7\ 10^{+308}$ à $-2.2\ 10^{-308}$, 0, signé, et de $2.2\ 10^{-308}$ à $1.7\ 10^{+308}$ non signé.
`DECIMAL[(n[,p])] [UNSIGNED] [ZEROFILL]`	Décimal à virgule fixe, *p* désigne la précision (nombre de chiffres après la virgule, maximum 30). Par défaut *n* vaut 10, *p* vaut 0.

5.4 DONNÉES BINAIRES

Les types **BLOB** (Binary Large OBject) permettent de stocker des données non structurées comme le multimédia (images, sons, vidéo, etc.). Les quatre (4) types de colonnes BLOB sont :**TINYBLOB**, **BLOB**, **MEDIUMBLOB** et **LONGBLOB**. Ces types sont traités comme des flots d'octets sans jeu de caractère associé.

Tableau 8 : Les types de données binaires en langage MySQL

Type	Description	Commentaire pour une colonne
`TINYBLOB(n)`	Flot de *n* octets.	Taille fixe (maximum de 255 octets).
`BLOB(n)`		Taille fixe (maximum de 65 535 octets).
`MEDIUMBLOB(n)`		Taille fixe (maximum de 16 mégaoctets).
`LONGBLOB(n)`		Taille fixe (maximum de 4,29 gigaoctets).

5.5 ÉNUMÉRATION

Deux types de collections sont proposés par MySQL.

- Le type **ENUM** définit une liste de valeurs permises (chaînes de caractères) ;
- Le type **SET** permettra de comparer une liste à une combinaison de valeurs permises à partir d'un ensemble de référence (chaînes de caractères).

Tableau 9 : Les types d'énumération en langage MySQL

Type	Description
`ENUM('valeur1','valeur2',...)`	Liste de 65 535 valeurs au maximum.
`SET('valeur1','valeur2',...)`	Ensemble de référence (maximum de 64 valeurs).

6. LANGAGE DE DÉFINITION DES DONNÉES

Ce chapitre décrit **les instructions SQL** qui constituent l'aspect LDD (langage de définition des données). À cet effet, nous verrons notamment comment déclarer une table avec ses éventuels index et contraintes.

6.1 CRÉATION D'UNE TABLE

Pour pouvoir créer une table dans votre base, il faut que vous ayez reçu le privilège **CREATE**.

La syntaxe SQL simplifiée est la suivante :

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] [nomBase.]nomTable
    ( colonne1 type1
            [NOT NULL | NULL] [DEFAULT valeur1] [COMMENT 'chaine1']
    [, colonne2 type2
            [NOT NULL | NULL] [DEFAULT valeur2] [COMMENT 'chaine2'] ]
    [CONSTRAINT nomContrainte1 typeContrainte1] ...)
    [ENGINE= InnoDB | MyISAM | ...];
```

- **TEMPORARY** : pour créer une table qui n'existera que durant la session courante (la table sera supprimée à la déconnexion). Deux (2) connexions peuvent ainsi créer deux tables temporaires de même nom sans risquer de conflit. Il faut posséder le privilège CREATE TEMPORARY TABLES.
- **IF NOT EXISTS** : permet d'éviter qu'une erreur se produise si la table existe déjà (si c'est le cas, elle n'est aucunement affectée par la tentative de création).

- **nomBase** : (jusqu'à 64 caractères permis dans un nom de répertoire ou de fichier sauf « / », « \ » et « . ») s'il est omis, il sera assimilé à la base connectée. S'il est précisé, il désigne soit la base connectée, soit une autre base (dans ce cas, il faut que l'utilisateur courant ait le droit de créer une table dans l'autre base). Nous aborderons ces points dans le chapitre « Contrôle des données » et nous considérerons jusque-là que nous travaillons dans la base courante (ce sera votre configuration la plupart du temps).
- **nomTable** : mêmes limitations que pour le nom de la base.
- **colonnei type i** : nom d'une colonne (mêmes caractéristiques que pour les noms des tables) et son type (INTEGER, CHAR, DATE…). La directive DEFAULT fixe une valeur par défaut. La directive NOT NULL interdit que la valeur de la colonne soit nulle.
- **COMMENT** : (jusqu'à 60 caractères) permet de commenter une colonne. Ce texte sera ensuite automatiquement affiché à l'aide des commandes SHOW CREATE TABLE et SHOW FULL COLUMNS (voir partie 5).
- **nomContraintei typeContraintei** : nom de la contrainte et son type (clé primaire, clé étrangère, etc.). Nous allons détailler par la suite les différentes contraintes possibles.
- **ENGINE** : définit le type de table (par défaut InnoDB, bien adapté à la programmation de transactions et adopté dans ce guide). MyISAM correspond au type par défaut des versions 3, parfaitement robuste, mais ne supportant pas pour l'heure l'intégrité référentielle. D'autres types existent, citons MEMORY pour les tables temporaires, ARCHIVE, etc.
- **« ; »** : symbole par défaut qui termine une instruction MySQL en mode ligne de commande (en l'absence d'un autre délimiteur).

Exemple. Le tableau 10 décrit l'instruction SQL qui permet de créer la table « Compagnie » illustrée par la figure 14, dans la base bdsoutou (Attention bdutil a été renommé en bdsoutou précédemment, l'absence du préfixe « bdsoutou. » conduirait au même résultat si bdsoutou était la base connectée lors de l'exécution du script).

Figure 14 : Table « Compagnie » (Exemple)

comp	nrue	ville	nomComp

Tableau 10 : Exemple d'Instruction SQL pour la création d'une table « Compagnie »

Instruction SQL	Commentaires

`CREATE TABLE bdsoutou.Compagnie`
`(comp CHAR(4),`
`nrue INTEGER(3),`
`rue CHAR(20),`
`ville CHAR(15) DEFAULT 'Paris'`
`COMMENT 'Par defaut : Paris',`
`nomComp CHAR(15) NOT NULL);`
	La table contient cinq colonnes (quatre chaînes de caractères et un numérique de trois chiffres). La colonne `ville` est commentée.
	La table inclut en plus deux contraintes :
	• `DEFAULT` qui fixe *Paris* comme valeur par défaut de la colonne `ville` ;
	• `NOT NULL` qui impose une valeur non nulle dans la colonne `nomComp`.

6.2 CONTRAINTES

Les contraintes ont pour but de programmer des règles de gestion au niveau des colonnes des tables. Les contraintes peuvent alléger un développement côté client (si on déclare qu'une note doit être comprise entre 0 et 20, les programmes de saisie n'ont plus à tester les valeurs en entrée mais seulement le code retour après connexion à la base ; on déporte les contraintes côté serveur).

Les quatre (4) types de contraintes les plus utilisées sont les suivants :

```
CONSTRAINT nomContrainte
UNIQUE (colonne1 [,colonne2]...)
PRIMARY KEY (colonne1 [,colonne2]...)
FOREIGN KEY (colonne1 [,colonne2]...)
REFERENCES nomTablePere [(colonne1 [,colonne2]...)]
              [ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
              [ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
CHECK (condition)
```

1. La contrainte **UNIQUE** impose une valeur distincte au niveau de la table (les valeurs nulles font exception à moins que NOT NULL soit aussi appliquée sur les colonnes).
2. La contrainte **PRIMARY KEY** déclare la clé primaire de la table. Un index est généré automatiquement sur la ou les colonnes concernées. Les colonnes clés primaires ne peuvent être ni nulles ni identiques (en totalité si elles sont composées de plusieurs colonnes).
3. La contrainte **FOREIGN KEY** déclare une clé étrangère entre une table enfant (child) et une table père (parent).
4. La contrainte **CHECK** impose un domaine de valeurs ou une condition simple ou complexe entre colonnes (exemple : **CHECK** (note BETWEEN 0 AND 20), CHECK (grade='Copilote' OR grade='Commandant')).

6.3 CRÉATION D'UN INDEX

Dans une table volumineuse la recherche séquentielle d'un enregistrement est une opération pénalisante en termes de temps de recherche.

On a intérêt à créer des index pour les attributs sur lesquels seront effectués de nombreuses recherches.

Pour pouvoir créer un index dans sa base, la table à indexer doit appartenir à la base. Si l'utilisateur a le privilège INDEX, il peut créer et supprimer des index dans sa base. Un index est créé par l'instruction **CREATE INDEX** et supprimé par **DROP INDEX**.

La syntaxe de création d'un index est la suivante :

```
CREATE [UNIQUE | FULLTEXT | SPATIAL] INDEX nomIndex
    [USING BTREE | HASH]
    ON nomTable (colonne1 [(taille1)] [ASC | DESC],...) ;
```

- **UNIQUE** permet de créer un index qui n'accepte pas les doublons.
- **FULLTEXT** permet de bénéficier de fonctions de recherche dans des textes (flot de caractères).
- **SPATIAL** permet de profiter de fonctions pour les données géographiques.
- **ASC** et **DESC** précisent l'ordre (croissant ou décroissant).

Créons deux index sur la table « Pilote ».

Instruction SQL	Commentaires
`CREATE UNIQUE INDEX idx_Pilote_nom3 USING BTREE ON Pilote (nom(3) DESC);`	Index *B-tree*, ordre décroissant sur les trois premiers caractères du nom des pilotes.
`CREATE INDEX idx_Pilote_compa USING BTREE ON Pilote (compa);`	Index *B-tree*, ordre croissant sur la colonne clé étrangère `compa`.

6.4 AFFICHER LA STRUCTURE D'UNE TABLE

DESCRIBE (écriture autorisée DESC) est une commande qui vient de SQL*Plus d'Oracle et qui a été reprise par MySQL. Cette écriture permet d'extraire la structure brute d'une table ou d'une vue.

```
DESCRIBE [nomBase.] nomTableouVue [colonne];
```

Si la base n'est pas indiquée, il s'agit de celle en cours d'utilisation. Retrouvons la structure des tables « Compagnie » et « Pilote » précédemment créées. Le type de chaque colonne apparaît :

Résultat

```
mysql> DESCRIBE Pilote;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| brevet | char(6)     | NO   | PRI |         |       |
| nom    | char(15)    | YES  | UNI | NULL    |       |
| nbHVol | double(7,2) | YES  |     | NULL    |       |
| compa  | char(4)     | YES  | MUL | NULL    |       |
+--------+-------------+------+-----+---------+-------+

mysql> DESCRIBE Compagnie;
+---------+----------+------+-----+---------+-------+
| Field   | Type     | Null | Key | Default | Extra |
+---------+----------+------+-----+---------+-------+
| comp    | char(4)  | NO   | PRI |         |       |
| nrue    | int(3)   | YES  |     | NULL    |       |
| rue     | char(20) | YES  |     | NULL    |       |
| ville   | char(15) | YES  |     | Paris   |       |
| nomComp | char(15) | NO   |     |         |       |
+---------+----------+------+-----+---------+-------+
```

Commentaires

Les clés primaires sont NOT NULL (désignées par PRI dans la colonne Key).
Les unicités sont désignées par UNI dans la colonne Key.
Les occurrences multiples possibles sont désignées par MUL dans la colonne Key.
Les contraintes NOT NULL nommées (définies via les contraintes CHECK) n'apparaissent pas.
La colonne Extra indique notamment les séquences (AUTO_INCREMENT).

6.5 EXERCICES

L'objectif de ces exercices est de créer des tables, leur clé primaire et des contraintes de vérification.

Exercice 1 : Présentation de la base de données

Une entreprise désire gérer son parc informatique à l'aide d'une base de données. Le bâtiment est composé de trois (3) étages. Chaque étage possède son réseau (ou segment distinct) Ethernet. Ces réseaux traversent des salles équipées de postes de travail. Un poste de travail est une machine sur laquelle sont installés certains logiciels. Quatre (4) catégories de postes de travail sont recensées (stations Unix, terminaux X, PC Windows et PC NT). La base de données devra aussi décrire les installations de logiciels.

Les noms et types des colonnes sont les suivants :

Colonne	Commentaire	Type
indIP	trois premiers groupes IP (exemple : 130.120.80)	VARCHAR(11)
nomSegment	nom du segment	VARCHAR(20)
etage	étage du segment	TINYINT(1)
nSalle	numéro de la salle	VARCHAR(7)
nomSalle	nom de la salle	VARCHAR(20)
nbPoste	nombre de postes de travail dans la salle	TINYINT(2)
nPoste	code du poste de travail	VARCHAR(7)
nomPoste	nom du poste de travail	VARCHAR(20)
ad	dernier groupe de chiffres IP (exemple : 11)	VARCHAR(3)
typePoste	type du poste (UNIX, TX, PCWS, PCNT)	VARCHAR(9)
dateIns	date d'installation du logiciel sur le poste	dateTIME
nLog	code du logiciel	VARCHAR(5)
nomLog	nom du logiciel	VARCHAR(20)
dateAch	date d'achat du logiciel	dateTIME
version	version du logiciel	VARCHAR(7)

typeLog	type du logiciel (UNIX, TX, PCWS, PCNT)	VARCHAR(9)
prix	prix du logiciel	DECIMAL(6,2)
numIns	numéro séquentiel des installations	INTEGER(5)
dateIns	date d'installation du logiciel	TIMESTAMP
delai	intervalle entre achat et installation	SMALLINT
typeLP	types des logiciels et des postes	VARCHAR(9)
nomType	noms des types (Terminaux X, PC Windows...)	VARCHAR(20)

Exercice 2 : Création des tables

Écrire puis exécuter le script SQL (que vous appellerez « creParc.sql ») de création des tables avec leur clé primaire (en gras dans le schéma suivant) et les contraintes suivantes :

- Les noms des segments, des salles et des postes sont non nuls ;
- Le domaine de valeurs de la colonne « ad » s'étend de 0 à 255 ;
- La colonne « prix » est supérieure ou égale à 0 ;
- La colonne « dateIns » est égale à la date du jour par défaut.

Segment

indIP	nomSegment	etage

Salle

nSalle	nomSalle	nbPoste	indIP

Poste

nPoste	nomPoste	indIP	ad	typePoste	nSalle

Logiciel

nLog	nomLog	dateAch	version	typeLog	prix

Installer

nPoste	nLog	**numIns**	dateIns	delai

Types

typeLP	nomType

Exercice 3 : Structure des tables

Écrire puis exécuter le script SQL (que vous appellerez « descParc.sql ») qui affiche la description de toutes ces tables (en utilisant des commandes **DESCRIBE**). Comparer le résultat obtenu avec le schéma ci-dessus (voir exercice 2).

Exercice 4 : Destruction des tables

Écrire puis exécuter le script SQL de destruction des tables (que vous appellerez « dropParc.sql »). Lancer ce script puis celui de la création des tables à nouveau.

7. LANGAGE DE MANIPULATION DES DONNÉES

Ce chapitre décrit l'aspect LMD (langage de manipulation des données) de MySQL. Nous verrons que SQL propose trois (3) instructions pour manipuler des données :

1. L'insertion d'enregistrements : **INSERT** ;
2. La modification de données : **UPDATE** ;
3. La suppression d'enregistrements : **DELETE** (et **TRUNCATE**).

7.1 INSERTIONS D'ENREGISTREMENTS

Pour pouvoir insérer des enregistrements dans une table, il faut que vous ayez reçu le privilège **INSERT**. Il existe plusieurs possibilités d'insertion : 1/ l'insertion mono ligne qui ajoute un enregistrement par instruction et l'insertion multiligne qui insère plusieurs enregistrements par une requête.

La syntaxe simplifiée de l'instruction INSERT mono ligne est la suivante.

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] [nomBase.] { nomTable | nomVue } [(nomColonne,...)]
    VALUES ({expression | DEFAULT},...),(...),...
    [ON DUPLICATE KEY UPDATE nomColonne = expression,...]
```

- **DELAYED** indique que l'insertion est différée (si la table est modifiée par ailleurs, le serveur attend qu'elle se libère pour y insérer périodiquement de nouveaux enregistrements si elle redevient active entre-temps).
- **LOW_PRIORITY** indique que l'insertion est différée à la libération complète de la table (option à ne pas utiliser sur des tables MyISAM).
- **HIGH_PRIORITY** annule l'option low priority du serveur.
- **IGNORE** indique que les éventuelles erreurs déclenchées suite à l'insertion seront considérées en tant que warnings.
- **ON DUPLICATE KEY UPDATE** permet de mettre à jour l'enregistrement présent dans la table, qui a déclenché l'erreur de doublon (dans le cas d'un index UNIQUE ou d'une clé primaire). Dans ce cas le nouvel enregistrement n'est pas inséré, seul l'ancien est mis à jour.

7.2 MODIFICATIONS DES DONNÉES DES COLONNES

L'instruction **UPDATE** permet la mise à jour des colonnes d'une table. Pour pouvoir modifier des enregistrements d'une table, il faut que cette dernière soit dans votre base ou que vous ayez reçu le privilège UPDATE sur la table.

```
UPDATE [LOW_PRIORITY] [IGNORE] [nomBase.] nomTable
SET col_name1=expr1 [, col_name2=expr2 ...]
      SET     colonne1 = expression1 | (requête_SELECT) | DEFAULT
            [,colonne2 = expression2...]
      [WHERE (condition)]
      [ORDER BY listeColonnes]
      [LIMIT nbreLimite]
```

- **LOW_PRIORITY** indique que la modification est différée à la libération complète de la table (option à ne pas utiliser sur des tables MyISAM).
- **IGNORE** signifie que les éventuelles erreurs déclenchées suite aux modifications seront considérées en tant que warnings.
- La clause **SET** actualise une colonne en lui affectant une expression (valeur, valeur par défaut, calcul ou résultat d'une requête).
- La condition du **WHERE** filtre les lignes à mettre à jour dans la table. Si aucune condition n'est précisée, tous les enregistrements seront actualisés. Si la condition ne filtre aucune ligne, aucune mise à jour ne sera réalisée.
- **ORDER BY** indique l'ordre de modification des colonnes.
- **LIMIT** spécifie le nombre maximum d'enregistrements à changer (par ordre de clé primaire croissante).

7.3 SUPPRESSIONS D'ENREGISTREMENTS

Les instructions **DELETE** et **TRUNCATE** permettent de supprimer un ou plusieurs enregistrements d'une table. Pour pouvoir supprimer des enregistrements dans une table, il faut que cette dernière soit dans votre base ou que vous ayez reçu le privilège **DELETE** sur la table.

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM [nomBase.] nomTable
       [WHERE (condition)]
       [ORDER BY listeColonnes]
       [LIMIT nbreLimite]
```

7.4 INSERTIONS À PARTIR D'UN FICHIER

L'importation de données (au format fichier texte) dans une table peut être programmée à l'aide de la directive **LOAD DATA INFILE**. Un tel mécanisme lit un fichier dans un répertoire du serveur et insère tout ou partie des informations dans une table.

La syntaxe simplifiée de cette directive est la suivante :

```
LOAD DATA INFILE 'nomEtCheminFichier.txt'
     [REPLACE | IGNORE] INTO TABLE nomTable
     [FIELDS [TERMIATED BY 'string']
             [[OPTIONALLY] ENCLOSED BY 'char']
             [ESCAPED BY 'char' ] ]
     [LINES [STARTING BY 'string'] [TERMINATED BY 'string'] ]
     [IGNORE number LINES];
```

7.5 EXERCICES

Les objectifs de ces exercices sont :

- D'insérer des données dans les tables du schéma Parc Informatique ;
- De créer une séquence et d'insérer des données en utilisant une séquence ;
- De modifier des données.

Exercice 5 : Insertion de données

Écrire puis exécuter le script SQL (que vous appellerez « insParc.sql ») afin d'insérer les données dans les tables suivantes :

Table Segment

INDIP	NOMSEGMENT	ETAGE
130.120.80	Brin RDC	
130.120.81	Brin 1er étage	
130.120.82	Brin 2e étage	

Table Salle

NSALLE	NOMSALLE	NBPOSTE	INDIP
s01	Salle 1	3	130.120.80
s02	Salle 2	2	130.120.80
s03	Salle 3	2	130.120.80
s11	Salle 11	2	130.120.81
s12	Salle 12	1	130.120.81
s21	Salle 21	2	130.120.82
s22	Salle 22	0	130.120.83
s23	Salle 23	0	130.120.83

Table Poste

NPOSTE	NOMPOSTE	INDIP	AD	TYPEPOSTE	NSALLE
p1	Poste 1	130.120.80	01	TX	s01
p2	Poste 2	130.120.80	02	UNIX	s01
p3	Poste 3	130.120.80	03	TX	s01
p4	Poste 4	130.120.80	04	PCWS	s02
p5	Poste 5	130.120.80	05	PCWS	s02
p6	Poste 6	130.120.80	06	UNIX	s03
p7	Poste 7	130.120.80	07	TX	s03
p8	Poste 8	130.120.81	01	UNIX	s11
p9	Poste 9	130.120.81	02	TX	s11
p10	Poste 10	130.120.81	03	UNIX	s12
p11	Poste 11	130.120.82	01	PCNT	s21
p12	Poste 12	130.120.82	02	PCWS	s21

Table : Logiciel

NLOG	NOMLOG	DATEACH	VERSION	TYPELOG	PRIX
log1	Oracle 6	1995-05-13	6.2	UNIX	3000
log2	Oracle 8	1999-09-15	8i	UNIX	5600
log3	SQL Server	1998-04-12	7	PCNT	2700
log4	Front Page	1997-06-03	5	PCWS	500
log5	WinDev	1997-05-12	5	PCWS	750
log6	SQL*Net		2.0	UNIX	500
log7	I. I. S.	2002-04-12	2	PCNT	810
log8	DreamWeaver	2003-09-21	2.0	BeOS	1400

Table : Types

TYPELP	NOMTYPE
TX	Terminal X-Window
UNIX	Système Unix
PCNT	PC Windows NT
PCWS	PC Windows
NC	Network Computer

Exercice 6 : Gestion d'une séquence

Dans ce même script, gérer la séquence associée à la colonne « numIns » commençant à la valeur « 1 » de manière à insérer les enregistrements suivants :

Table : Installer

NPOSTE	NLOG	NUMINS	DATEINS	DELAI
p2	log1	1	2003-05-15	
p2	log2	2	2003-09-17	
p4	log5	3		
p6	log6	4	2003-05-20	
p6	log1	5	2003-05-20	
p8	log2	6	2003-05-19	
p8	log6	7	2003-05-20	
p11	log3	8	2003-04-20	
p12	log4	9	2003-04-20	
p11	log7	10	2003-04-20	
p7	log7	11	2002-04-01	

Exercice 7 : Modification de données

Écrire le script « modification.sql » qui permet de modifier (avec UPDATE) la colonne « etage » (pour l'instant nulle) de la table « Segment », afin d'affecter un numéro d'étage correct (« 0 » pour le segment 130.120.80, « 1 » pour le segment 130.120.81, « 2 » pour le segment 130.120.82).

Diminuer de 10 % le prix des logiciels de type « PCNT ». Puis, vérifier :

- SELECT * FROM Segment;
- SELECT nLog, typeLog, prix FROM Logiciel.

7.6 RENOMMER UNE TABLE

L'instruction **RENAME** renomme une ou plusieurs tables ou vues. Il faut posséder le privilège **ALTER** et **DROP** sur la table d'origine et **CREATE** sur la base.

```
RENAME [nomBase.]ancienNomTable TO [nomBase.] nouveauNomTable
       [,[nomBase.] ancienNom2 TO [nomBase.]nouveauNom2]];
```

Les contraintes d'intégrité, index et prérogatives associés à l'ancienne table sont automatiquement transférés sur la nouvelle. En revanche, les vues et procédures cataloguées sont invalidées et doivent être recréées.

Il est aussi possible d'utiliser l'option **RENAME TO** de l'instruction **ALTER TABLE** pour renommer une table existante. Le tableau 11 suivant décrit comment renommer la table « Pilote » sans perturber l'intégrité référentielle.

Tableau 11 : Exemple d'Instruction SQL pour renommer la table « Pilote »

Commande RENAME	Commande ALTER TABLE
`RENAME Pilote TO Naviguant;`	`ALTER TABLE Pilote RENAME TO Naviguant;`

7.7 MODIFICATIONS STRUCTURELLES D'UNE TABLE

Considérons la table suivante que nous allons faire évoluer :

```
CREATE TABLE Pilote
   (brevet VARCHAR(4), nom VARCHAR(20));
INSERT INTO Pilote
   VALUES ('PL-1', 'Agnès Labat');
```

Pilote

brevet	nom
PL-1	Agnès Labat

7.7.1 AJOUT DE COLONNE

La directive **ADD** de l'instruction **ALTER TABLE** permet d'ajouter une nouvelle colonne à une table. Cette colonne est initialisée à NULL pour tous les enregistrements (à moins de spécifier une contrainte **DEFAULT**, auquel cas tous les enregistrements de la table sont mis à jour avec une valeur non nulle).

Le script suivant ajoute trois (3) colonnes à la table « Pilote ». La première instruction insère la colonne « nbHVo1 » en l'initialisant à NULL pour tous les pilotes (ici il n'en existe qu'une seule).

La deuxième commande ajoute deux (2) colonnes initialisées à une valeur non nulle. La colonne « ville » ne sera jamais nulle.

```
ALTER TABLE Pilote ADD (nbHVol DECIMAL(7,2));
ALTER TABLE Pilote
      ADD (compa VARCHAR(4)  DEFAULT 'AF',
           ville VARCHAR(30) DEFAULT 'Paris' NOT NULL);
```

La table est désormais la suivante :

Pilote

brevet	nom	nbHVol	compa	ville
PL-1	Agnès Labat		AF	Paris

7.7.2 RENOMMER DES COLONNES

Il faut utiliser l'option **CHANGE** de l'instruction **ALTER TABLE** pour renommer une colonne existante. Le nouveau nom de colonne ne doit pas être déjà utilisé dans la table. Le type (avec une éventuelle contrainte) doit être reprécisé. La position de la colonne peut aussi être modifiée en même temps. La syntaxe générale de cette option est la suivante :

```
ALTER TABLE [nomBase].nomTable CHANGE [COLUMN] ancienNom
       nouveauNom typeMySQL [NOT NULL | NULL] [DEFAULT valeur]
           [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
                 [COMMENT 'chaine'] [REFERENCES ...]
        [FIRST|AFTER nomColonne];
```

7.7.3 MODIFIER LE TYPE DES COLONNES

L'option **MODIFY** de l'instruction **ALTER TABLE** modifie le type d'une colonne existante sans pour autant la renommer. La syntaxe générale de cette instruction est la suivante (tabelau 12), les options sont les mêmes que pour **CHANGE**.

Tableau 12 : Présentation des différentes modifications de colonnes

Instructions SQL	Commentaires
ALTER TABLE Pilote **MODIFY** compa VARCHAR(6) DEFAULT 'SING'; INSERT INTO Pilote (brevet, nom) VALUES ('PL-2', 'Laurent Boutrand');	Augmente la taille de la colonne compa et change la contrainte de valeur par défaut puis insère un nouveau pilote.
ALTER TABLE Pilote **MODIFY** compa CHAR(4) NOT NULL;	Diminue la colonne et modifie également son type de VARCHAR en CHAR tout en le déclarant NOT NULL (possible car les données contenues dans la colonne ne dépassent pas quatre caractères).
ALTER TABLE Pilote **MODIFY** compa CHAR(4);	Rend possible l'insertion de valeur nulle dans la colonne compa.

La table est désormais la suivante :

CHAR(4)
NULL possible
Défaut : 'SING'

Pilote

brevet	nom	nbHVol	adresse	compa
PL-1	Agnès Labat		Paris	AF
PL-2	Laurent Boutrand		Paris	SING

7.7.4 SUPPRIMER DES COLONNES

L'option **DROP** de l'instruction **ALTER TABLE** permet de supprimer une colonne (aussi un index ou une clé que nous verrons plus loin). La possibilité de supprimer une colonne évite aux administrateurs d'exporter les données, de recréer une nouvelle table, d'importer les données et de recréer les éventuels index et contraintes. Lorsqu'une colonne est supprimée, les index qui l'utilisent sont mis à jour, voire éliminés si toutes les colonnes qui le composent sont effacées.

La syntaxe de ces options est la suivante :

```
ALTER TABLE [nomBase].nomTable DROP
  { [COLUMN] nomColonne | PRIMARY KEY
    | INDEX nomIndex | FOREIGN KEY nomContrainte }
```

7.8 SUPPRESSION D'UNE TABLE

Pour pouvoir supprimer une table dans une base, il faut posséder le privilège **DROP** sur cette base. L'instruction **DROP TABLE** entraîne la suppression des données, de la structure, de la description dans le dictionnaire des données, des index, des déclencheurs associés (triggers) et la récupération de la place dans l'espace de stockage.

```
DROP [TEMPORARY] TABLE [IF EXISTS]
    [nomBase.] nomTable1 [,[nomBase2.]nomTable2,...]
    [RESTRICT | CASCADE]
```

7.9 MODIFICATIONS COMPORTEMENTALES

Nous étudions dans cette section les mécanismes d'ajout, de suppression, d'activation et de désactivation de contraintes.

Faisons évoluer le schéma suivant. Les seules contraintes existantes sont les clés primaires nommées « pk_Compagnie », pour la table « Compagnie » et « pk_Avion » pour la table « Avion ».

Compagnie

comp	nrue	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
SING	7	Camparols	Singapour	Singapore AL

Affreter

compAff	immat	dateAff	nbPax
AF	F-WTSS	2003-05-13	85
SING	F-GAFU	2003-02-05	155
AF	F-WTSS	2003-05-15	82

Avion

immat	typeAvion	nbHVol	proprio
F-WTSS	Concorde	6570	SING
F-GAFU	A320	3500	AF
F-GLFS	TB-20	2000	SING

7.9.1 AJOUT DE CONTRAINTES

Jusqu'à présent, nous avons créé les contraintes en même temps que les tables. Il est possible de créer des tables sans contraintes.

```
ALTER TABLE [nomBase].nomTable ADD
  { INDEX [nomIndex] [typeIndex] (nomColonne1,...)
  | CONSTRAINT nomContrainte typeContrainte }
```

Trois (3) types de contraintes sont possibles :

- UNIQUE (*colonne1* [,*colonne2*]...)
- PRIMARY KEY (*colonne1* [,*colonne2*]...)
- FOREIGN KEY (*colonne1* [,*colonne2*]...)

```
    REFERENCES nomTablePère (col1 [,col2]...)
      [ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
      [ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}]
```

Unicité

Ajoutons la contrainte d'unicité portant sur la colonne du nom de la compagnie. Un index est automatiquement généré sur cette colonne à présent :

```
ALTER TABLE Compagnie ADD CONSTRAINT un_nomC UNIQUE (nomComp);
```

Clé étrangère

Ajoutons la clé étrangère (indexée) à la table « Avion » au niveau de la colonne « proprio » en lui assignant une contrainte NOT NULL :

```
ALTER TABLE Avion ADD INDEX (proprio);
ALTER TABLE Avion ADD CONSTRAINT fk_Avion_comp_Compag
                  FOREIGN KEY(proprio) REFERENCES Compagnie(comp);
ALTER TABLE Avion MODIFY proprio CHAR(4) NOT NULL;
```

Clé primaire

Ajoutons à la table « Affreter » en une seule instruction, sa clé primaire et deux (2) clés étrangères (une vers la table « Avion » et l'autre vers « Compagnie ») :

```
ALTER TABLE Affrete ADD (
      CONSTRAINT pk_Affreter        PRIMARY KEY (compAff, immat, dateAff),
      CONSTRAINT fk_Aff_na_Avion    FOREIGN KEY(immat) REFERENCES
                                    Avion(immat),
      CONSTRAINT fk_Aff_comp_Compag FOREIGN KEY(compAff)
                                    REFERENCES Compagnie(comp));
```

Les tables contiennent à présent les contraintes suivantes :

Compagnie *referenced / parent*

comp	nrue	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
SING	7	Camparols	Singapour	Singapore AL

Affreter *dependent / child*

compAff	immat	dateAff	nbPax
AF	F-WTSS	2003-05-13	85
SING	F-GAFU	2003-02-05	155
AF	F-WTSS	2003-05-15	82

Avion *dependent / child* NOT NULL

immat	typeAvion	nbHVol	proprio
F-WTSS	Concorde	6570	SING
F-GAFU	A320	3500	AF
F-GLFS	TB-20	2000	SING

referenced / parent

7.9.2 SUPPRESSION DE CONTRAINTES

Contrainte NOT NULL

Il faut utiliser la directive **MODIFY** de l'instruction **ALTER TABLE** pour supprimer une contrainte **NOT NULL** existant sur une colonne. Dans notre exemple, détruisons la contrainte NOT NULL de

la clé étrangère proprio dans la table « Avion ».

```
ALTER TABLE Avion MODIFY proprio CHAR(4) NULL;
```

Contrainte UNIQUE

Il faut utiliser la directive **DROP INDEX** de l'instruction **ALTER TABLE** pour supprimer une contrainte d'unicité. Dans notre exemple, effaçons la contrainte portant sur le nom de la compagnie. L'index est également détruit.

```
ALTER TABLE Compagnie DROP INDEX un_nomC;
```

Clé étrangère

L'option **DROP FOREIGN KEY** de l'instruction **ALTER TABLE** permet de supprimer une clé étrangère d'une table. La syntaxe générale est la suivante :

```
ALTER TABLE [nomBase].nomTable DROP FOREIGN KEY nomContrainte;
```

Si la contrainte a été définie sans être nommée, son nom est généré par le moteur InnoDB et l'instruction **SHOW CREATE TABLE** [nomBase].nomTable permet de le découvrir.

Détruisons la clé étrangère de la colonne proprio.

```
ALTER TABLE Avion DROP FOREIGN KEY fk_Avion_comp_Compag;
```

Clé primaire

L'option **DROP PRIMARY KEY** de l'instruction **ALTER TABLE** permet de supprimer une clé primaire.

Si la colonne clé primaire à supprimer contient des clés étrangères, il faut d'abord retirer les contraintes de clé étrangère. Si la clé primaire à supprimer est référencée par des clés étrangères d'autres tables, il faut d'abord ôter les contraintes de clé étrangère de ces autres tables.

Ainsi, pour supprimer la clé primaire de la table « Affreter », il faut d'abord enlever les deux (2) contraintes de clé étrangère concernant des colonnes composant la clé primaire.

```
ALTER TABLE Affreter DROP FOREIGN KEY fk_Aff_na_Avion;
ALTER TABLE Affreter DROP FOREIGN KEY fk_Aff_comp_Compag;

ALTER TABLE Affreter DROP PRIMARY KEY;
```

La figure suivante illustre les contraintes qui restent actives : les clés primaires des tables « Compagnie » et « Avion » et une contrainte de non nullité.

Compagnie

comp	nrue	rue	ville	nomComp
AF	10	Gambetta	Paris	Air France
SING	7	Camparols	Singapour	Singapore AL

Affreter

compAff	immat	dateAff	nbPax
AF	F-WTSS	2003-05-13	85
SING	F-GAFU	2003-02-05	155
AF	F-WTSS	2003-05-15	82

Avion

NULL

immat	typeAvion	nbHVol	proprio
F-WTSS	Concorde	6570	SING
F-GAFU	A320	3500	AF
F-GLFS	TB-20	2000	SING

7.9.3 DÉSACTIVATION DES CONTRAINTES

La désactivation des contraintes référentielles peut être intéressante pour accélérer des procédures de chargement d'importation et d'exportation massives de données externes. Ce mécanisme améliore aussi les performances de programmes batchs qui ne modifient pas des données concernées par l'intégrité référentielle ou pour lesquelles on vérifie la cohérence de la base à la fin. En effet les index ne sont pas mis à jour pour chaque insertion ou modification et aucun ordre n'est requis pour insérer ou supprimer des enregistrements.

L'instruction **SET FOREIGN_KEY_CHECKS=0** permet de désactiver temporairement (jusqu'à la réactivation) toutes les contraintes référentielles d'une base.

7.9.4 RÉACTIVATION DES CONTRAINTES

L'instruction **SET FOREIGN_KEY_CHECKS=1** permet de réactiver toutes les contraintes référentielles d'une base.

7.9.5 CONTRAINTES DIFFÉRÉES

Les contraintes que nous avons étudiées jusqu'à maintenant sont des contraintes immédiates (immediate) qui sont contrôlées après chaque instruction. Une contrainte est dite « différée » (deferred) si elle déclenche sa vérification seulement à la fin de la transaction (première instruction commit rencontrée). Pour l'heure, MySQL avec InnoDB ne propose pas ce mode de programmation.

7.10 EXERCICES

Exercice 8 : Ajout de colonnes

Écrire le script « évolution.sql » qui contient les instructions nécessaires pour ajouter les colonnes suivantes (avec **ALTER TABLE**). Le contenu de ces colonnes sera modifié ultérieurement.

Table	Nom, type et signification des nouvelles colonnes
`Segment`	`nbSalle TINYINT(2)` : nombre de salles par défaut = 0. `nbPoste TINYINT(2)` : nombre de postes par défaut = 0.
`Logiciel`	`nbInstall TINYINT(2)` : nombre d'installations par défaut = 0.
`Poste`	`nbLog TINYINT(2)` : nombre de logiciels installés par défaut = 0.

Vérifier la structure et le contenu de chaque table avec `DESCRIBE` et `SELECT`.

Exercice 9 : Modification de colonnes

Dans ce même script, rajouter les instructions nécessaires pour :

- Augmenter la taille dans la table « Salle » de la colonne « nomSalle » (passer à VARCHAR(30)) ;
- Diminuer la taille dans la table « Segment » de la colonne « nomSegment » à VARCHAR(15) ;
- Tenter de diminuer la taille dans la table « Segment » de la colonne « nomSegment » à VARCHAR(14).

Pourquoi la commande n'est-elle pas possible ?

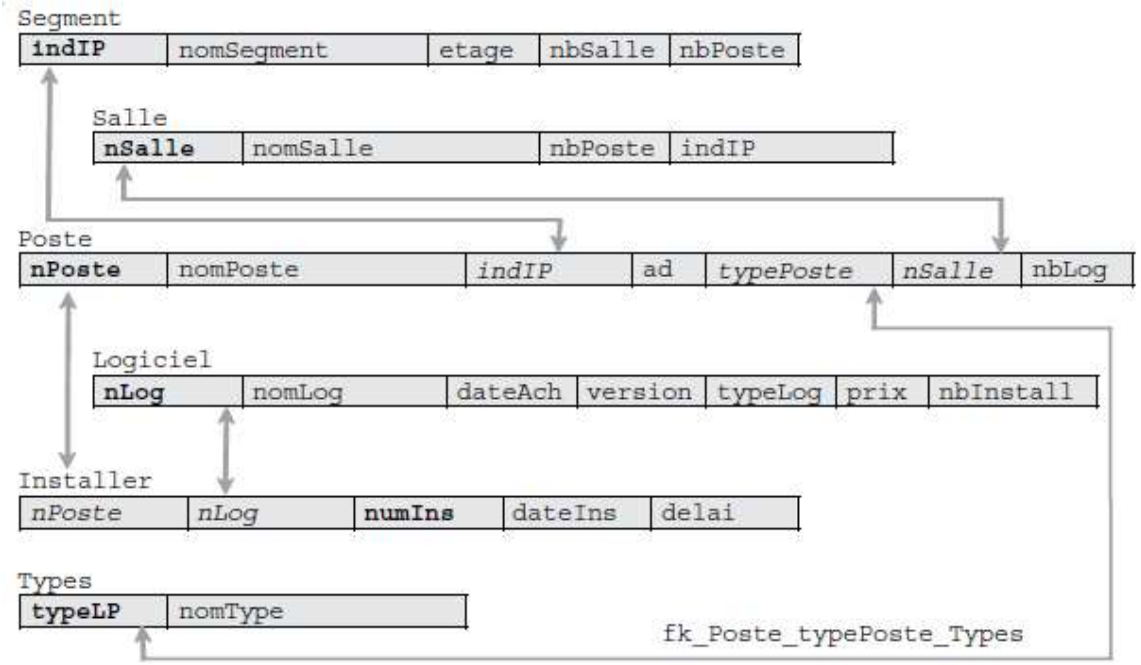
Vérifier la structure et le contenu de chaque table avec DESCRIBE et SELECT.

Exercice 10 : Ajout de contraintes

Ajouter la contrainte afin de s'assurer qu'on ne puisse installer plusieurs fois le même logiciel sur un poste de travail donné.

Ajouter les contraintes de clés étrangères pour assurer l'intégrité référentielle (avec ALTER TABLE… ADD CONSTRAINT…) entre les tables suivantes.

Si l'ajout d'une contrainte référentielle renvoie une erreur, vérifier les enregistrements des tables « pères » et « fils » (notamment au niveau de la casse des chaînes de caractères, « Tx » est différent de « TX » par exemple).



Modifier le script SQL de destruction des tables (dropParc.sql) en fonction des nouvelles contraintes.

Lancer ce script puis tous ceux écrits jusqu'ici.

8. LANGAGE D'INTERROGATION DES DONNÉES

Ce chapitre traite de l'aspect le plus connu du langage SQL qui concerne l'extraction des données par requêtes (nom donné aux instructions **SELECT**). Une requête permet de rechercher des données dans une ou plusieurs tables ou vues, à partir de critères simples ou complexes. Les instructions SELECT peuvent être exécutées dans l'interface de commande (voir les exemples de ce chapitre) ou au sein d'un programme SQL (procédure cataloguée), PHP, Java, C, etc.

8.1 SYNTAXE (SELECT)

Pour pouvoir extraire des enregistrements d'une table, il faut que celle-ci soit dans votre base ou que vous ayez reçu le privilège SELECT sur la table.

La syntaxe SQL simplifiée de l'instruction SELECT est la suivante :

```
SELECT [ { DISTINCT | DISTINCTROW } | ALL ] listeColonnes
 FROM nomTable1 [,nomTable2]...
 [ WHERE condition ]
 [ clauseRegroupement ]
 [ HAVING condition ]
 [ clauseOrdonnancement ]
 [ LIMIT [rangDépart,] nbLignes ] ;
```

Les options seront détaillées au fur et à mesure.

8.2 EXERCICES

Les objectifs de ces exercices sont :

- De créer dynamiquement des tables et leurs données ;
- D'écrire des requêtes monotables et multitables ;
- De réaliser des modifications synchronisées ;
- De composer des jointures et des divisions.

Exercice 5 :

Écrire le script « créaDynamique.sql » permettant de créer les tables « Softs » et « PCSeuls » suivantes (en utilisant la directive **AS SELECT** de la commande **CREATE TABLE**). Vous ne poserez aucune contrainte sur ces tables. Pensez à modifier le nom des colonnes.

Softs

nomSoft	version	prix

PCSeuls

nP	nomP	seg	ad	typeP	salle

La table « Softs » sera construite sur la base de tous les enregistrements de la table « Logiciel » que vous avez créée et alimentée précédemment. La table « PCSeuls » doit seulement contenir les enregistrements de la table « Poste », qui sont de type « PCWS » ou « PCNT ». Vérifier :

- SELECT * FROM Softs ;
- SELECT * FROM PCSeuls.

Exercice 6 :

Écrire le script « requêtes.sql » permettant d'extraire, à l'aide d'instructions **SELECT** et les données suivantes :

1. Type du poste « p8 » ;
2. Noms des logiciels « UNIX » ;
3. Noms, adresses IP, numéros de salle des postes de type « UNIX » ou « PCWS » ;
4. Même requête pour les postes du segment « 130.120.80 » triés par numéros de salles décroissants ;
5. Numéros des logiciels installés sur le poste « p6 » ;
6. Numéros des postes qui hébergent le logiciel « log1 » ;
7. Noms et adresses IP complètes (exemple : « 130.120.80.01 ») des postes de type « TX » (utiliser la fonction de concaténation).

Exercice 7 :

8. Pour chaque poste, le nombre de logiciels installés (en utilisant la table Installer) ;
9. Pour chaque salle, le nombre de postes (à partir de la table « Poste ») ;
10. Pour chaque logiciel, le nombre d'installations sur des postes différents ;
11. Moyenne des prix des logiciels « UNIX » ;
12. Plus récente date d'achat d'un logiciel ;
13. Numéros des postes hébergeant 2 logiciels ;
14. Nombre de postes hébergeant 2 logiciels (utiliser la requête précédente en faisant un

SELECT dans la clause FROM).

Exercice 8 :

Opérateurs ensemblistes

15. Types de postes non recensés dans le parc informatique (utiliser la table Types) ;
16. Types existant à la fois comme types de postes et de logiciels ;
17. Types de postes de travail n'étant pas des types de logiciels ;
18. Jointures procédurales ;
19. Adresses IP complètes des postes qui hébergent le logiciel « log6 » ;
20. Adresses IP complètes des postes qui hébergent le logiciel de nom « Oracle 8 » ;
21. Noms des segments possédant exactement trois (3) postes de travail de type « TX » ;
22. Noms des salles où l'on peut trouver au moins un poste hébergeant le logiciel « Oracle 6 » ;
23. Nom du logiciel acheté le plus récent (utiliser la requête 12).

Jointures relationnelles

Écrire les requêtes 18, 19, 20, 21 avec des jointures de la forme relationnelle. Numérotez ces nouvelles requêtes de 23 à 26.

27. Installations (nom segment, nom salle, adresse IP complète, nom logiciel, date d'installation) triées par segment, salle et adresse IP).

Jointures SQL2

Écrire les requêtes 18, 19, 20, 21 avec des jointures SQL2 (JOIN, NATURAL JOIN, JOIN USING).

Numérotez ces nouvelles requêtes de 28 à 31.

Exercice 9 :

Écrire le script « modifSynchronisées.sql » pour ajouter les lignes suivantes dans la table. Installer :

Installer

nPoste	nLog	numIns	dateIns	delai
...	...	...	...	...
p2	log6	séquence...	SYSDATE()	NULL
p8	log1		SYSDATE()	NULL
p10	log1		SYSDATE()	NULL

Écrire les requêtes UPDATE synchronisées de la forme suivante :

```
UPDATE table1 alias1
  SET colonne = (SELECT COUNT(*)
                 FROM  table2 alias2
                 WHERE alias2.colonneA = alias1.colonneB...);
```

Pour mettre à jour automatiquement les colonnes rajoutées :

- « nbSalle » dans la table « Segment » (nombre de salles traversées par le segment) ;
- « nbPoste » dans la table « Segment » (nombre de postes du segment) ;
- « nbInstall » dans la table « Logiciel » (nombre d'installations du logiciel) ;
- « nbLog » dans la table « Poste » (nombre de logiciels installés par poste).

Vérifier le contenu des tables modifiées (Segment, Logiciel et Poste).

Exercice 10 :

Rajouter au script « requêtes.sql », les instructions **SELECT** pour extraire les données suivantes : Sous-interrogation synchronisée.

32 Noms des postes ayant au moins un logiciel commun au poste « p6 » (on doit trouver les postes p2, p8 et p10).

Divisions

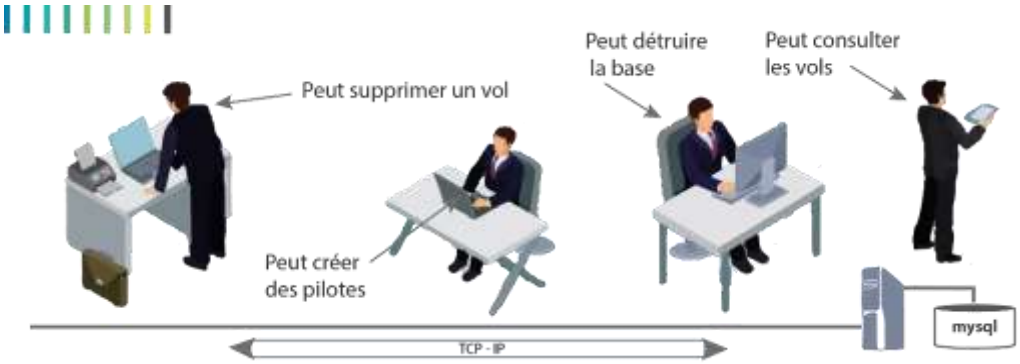
33 Noms des postes ayant les mêmes logiciels que le poste « p6 » (les postes peuvent avoir plus de logiciels que « p6 »). On doit trouver les postes « p2 » et « p8 » (division inexacte).

34 Noms des postes ayant exactement les mêmes logiciels que le poste « p2 » (division exacte), on doit trouver « p8 ».

9. LANGAGE DE CONTRÔLE DES DONNÉES

Comme dans tout système multi-utilisateur, l'usager d'un SGBD doit être identifié avant de pouvoir utiliser des ressources. **Les accès** aux informations et à la base de données **doivent être contrôlés à des fins de sécurité et de cohérence**. La figure 15 suivante illustre un groupe d'utilisateurs dans lequel existe une classification entre ceux qui peuvent consulter, mettre à jour, supprimer des enregistrements, voire les tables.

Figure 15 : Exemple de groupes d'utilisateurs selon des droits d'accès



Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition

Dans cette section les aspects du langage SQL qui concernent le contrôle des données et des accès sont abordés et plus spécifiquement :

- La gestion des utilisateurs qui manipuleront des bases de données dans lequelles se trouvent des objets tels que des tables, index, séquences, vues, procédures, etc. ;
- La gestion des privilèges qui permettent de donner des droits sur la base de données (privilèges système) et sur les données de la base (privilèges objet) ;
- La gestion des vues ;
- L'utilisation du dictionnaire des données (base de données « information_schema »).

9.1 GESTION DES UTILISATEURS

Un utilisateur (user) est identifié par MySQL par son nom et celui de la machine à partir de laquelle il se connecte. Cela fait, il pourra accéder à différents objets (tables, vues, séquences, index, procédures, etc.) d'une ou de plusieurs bases sous réserve d'avoir reçu un certain nombre de privilèges.

9.1.1 CLASSIFICATION

Les types d'utilisateurs, leurs fonctions et leur nombre peuvent varier d'une base à une autre. Néanmoins, pour chaque base de données en activité, on peut classifier les utilisateurs de la manière suivante :

- **Le DBA (DataBase Administrator)**. Il en existe au moins un (1). Une petite base peut n'avoir qu'un seul administrateur. Une base importante peut en regrouper plusieurs qui se partagent les tâches suivantes :
  - ✓ Installation et mises à jour de la base et des outils éventuels ;
  - ✓ Gestion de l'espace disque et des espaces pour les données ;
  - ✓ Gestion des utilisateurs et de leurs objets (s'ils ne les gèrent pas eux-mêmes) ;
  - ✓ Optimisation des performances ;
  - ✓ Sauvegardes, restaurations et archivages ;
  - ✓ Contact avec le support technique.
- **L'administrateur réseau** (qui peut être le DBA) se charge de la configuration des couches client pour les accès distants.
- **Les développeurs** qui conçoivent et mettent à jour la base. Ils peuvent aussi agir sur leurs objets (création et modification des tables, index, séquences, etc.). Ils transmettent au DBA leurs demandes spécifiques (stockage, optimisation, sécurité).
- **Les administrateurs d'application** qui gèrent les données manipulées par la ou les applications. Pour les petites et les moyennes bases, le DBA joue ce rôle.
- **Les utilisateurs** qui se connectent et interagissent avec la base à travers les applications ou à l'aide d'outils (interrogations pour la génération de rapports, ajouts, modifications ou suppressions d'enregistrements).

Tous seront des utilisateurs (au sens MySQL) avec des privilèges différents.

9.1.2 CRÉATION D'UN UTILISATEUR

Pour pouvoir créer un utilisateur, vous devez posséder le privilège **CREATE USER** ou **INSERT** sur la base système mysql (car c'est la table « mysql.user » qui stockera l'existence de ce nouvel arrivant). La syntaxe de création d'un utilisateur est la suivante :

CREATE USER *utilisateur*
[IDENTIFIED BY [PASSWORD] '*motdePasse*']
[,*utilisateur2* [IDENTIFIED BY [PASSWORD] '*motdePasse2*'] ...];

IDENTIFIED BY motdePasse permet d'affecter un mot de passe (16 caractères maximum, sensibles à la casse) à un utilisateur (16 caractères maximum sensibles aussi à la casse).

Le tableau suivant décrit la création d'un utilisateur (à exécuter en étant connecté en local en tant que root) :

Instruction SQL	Résultat
`CREATE USER soutou@localhost IDENTIFIED BY 'iut';`	*soutou* est déclaré « utilisateur à accès local », il devra se connecter à l'aide de son mot de passe.

Par défaut, les utilisateurs, une fois créés, n'ont aucun droit sur aucune base de données (à part en lecture écriture sur la base test et en lecture seule sur la base « information_schema »). La section « Privilèges » étudie ces droits.

Liste des utilisateurs

À propos de root, on le retrouve dans la table « user » de la base mysql (mysql.user). L'extraction des colonnes « User » et « Host » restitue la liste des utilisateurs connus du serveur. Si root n'avait pas sélectionné la base mysql, la commande à exécuter aurait été « SELECT User,Host FROM mysql.user ; ».

L'utilisateur vide ' ' correspond à une connexion anonyme.

La machine désignée par « % » indique que la connexion est autorisée à partir de tout site (en supposant qu'un client MySQL est installé et qu'il est relié au serveur par TCP-IP).

La machine désignée par « localhost » spécifie que la connexion est autorisée en local.

Ici, la table fait état que l'accès anonyme (restreint toutefois à la base test, voir la section Table mysql.db) est permis en local et à partir de tout site, et que soutou comme root ne peuvent se connecter qu'en local.

9.1.3 MODIFICATION D'UN UTILISATEUR

Le mot de passe d'un utilisateur peut être modifié sans parler de privilèges. Nous verrons plus tard qu'il est possible de restreindre le nombre de requêtes (SELECT), de modifications (UPDATE), de connexions par heure et de connexions simultanées à un serveur.

Puisqu'il n'existe pas de commande ALTER USER, pour changer un mot de passe, il faut donc modifier la table « user » par la seule commande SQL capable de le faire : **UPDATE**.

L'instruction suivante modifie le mot de passe de l'utilisateur « soutou » pour l'accès en local.

Notez l'utilisation de la fonction **PASSWORD()** qui code le mot de passe à affecter à la colonne **authentication_string** de la table « user ». Il est plus prudent d'utiliser ensuite **FLUSH PRIVILEGES** qui recharge les tables système de manière à rendre la manipulation effective sur l'instant.

```
UPDATE mysql.user
SET authentication_string=PASSWORD('Test')
WHERE User= 'soutou'
AND Host='localhost' ;
FLUSH PRIVILEGES ;
```

Chaque utilisateur peut changer son propre mot de passe à l'aide de cette instruction s'il en a le privilège.

Mais attention ! Le fait de lui donner ce droit implique également qu'il puisse aussi modifier les mots de passe de ses copains, ainsi que celui du root !

La syntaxe SET PASSWORD ... = PASSWORD ('auth_string') est obsolète dans MySQL 5.7 et est supprimée dans MySQL 8.0.

9.1.4 RENOMMER UN UTILISATEUR

Pour pouvoir renommer un utilisateur, vous devez posséder le privilège **CREATE USER** (ou le privilège **UPDATE** sur la base de données mysql). La syntaxe SQL est la suivante : **RENAME USER** utilisateur TO nouveauNom.

Penser à spécifier l'accès complet à renommer (user@machine). Les privilèges et le mot de passe ne changent pas. Le tableau suivant décrit trois (3) opérations de renommage d'utilisateurs (qui reviennent d'ailleurs à l'état initial).

Tableau 13 : Opérations de renommage d'utilisateurs

Instruction SQL	Commentaire
`RENAME USER soutou@localhost TO christiansoutou@localhost;`	L'accès *soutou* en local est renommé *christiansoutou* en local.
`RENAME USER christiansoutou@localhost TO christiansoutou@194.53.227.12;`	L'accès *christiansoutou* en local est renommé *christiansoutou* en accès distant.
`RENAME USER christiansoutou@194.53.227.12 TO soutou@localhost;`	L'accès est renommé complètement.

9.1.5 SUPPRESSION D'UN UTILISATEUR

Pour pouvoir supprimer un utilisateur, vous devez posséder le privilège **CREATE USER** (ou le privilège **DELETE** sur la base de données mysql). La syntaxe SQL est la suivante : **DROP USER** utilisateur [,utilisateur2 ...].

Il faut spécifier l'accès à éliminer (user@machine). Tous les privilèges relatifs à cet accès sont détruits. Si l'utilisateur est connecté dans le même temps, sa suppression ne sera effective qu'à la fin de sa (dernière) session.

Pour supprimer le compte soutou en local, la commande à lancer est :

```
DROP USER soutou@localhost;
```

9.2 GESTION DES BASES DE DONNÉES

9.2.1 CRÉATION D'UNE BASE

Pour pouvoir créer une base de données, vous devez posséder le privilège **CREATE** sur la nouvelle base (ou au niveau global pour créer toute table).

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] nomBase
  [ [DEFAULT] CHARACTER SET nomJeu ]
      [ [DEFAULT] COLLATE nomCollation ];
```

IF NOT EXISTS évite une erreur dans le cas où la base de données existe déjà (auquel cas elle ne sera pas remplacée).

nomBase désigne le nom de la base (64 caractères maximum, caractères compris par le système de gestion de fichier du système d'exploitation, notamment respectant les règles de nommage des répertoires). Les caractères « / », « \ », ou « . » sont proscrits.

CHARACTER SET indique le jeu de caractères associé aux données qui résideront dans les tables de la base.

COLLATE définit la collation1 du jeu de caractères en question. La collation, dans le jargon informatique, permet de définir la position des caractères dans le jeu. Par exemple, il sera possible de différencier « à » de « a » ou pas (sensibilité diacritique). Le but étant de s'adapter aux différentes règles et langues de notre petite planète.

Une fois créée, vous constaterez l'existence d'un répertoire portant le nom de votre nouvelle base (par défaut sous C:\Program Files\MySQL\MySQL Server 5.n\data\nouvelleBase dans le cas de Windows). Ce répertoire contiendra les données des tables qui seront constituées dans la nouvelle base. Si vous concevez manuellement un répertoire (mkdir rep1 par exemple) dans le répertoire de data de MySQL, rep1 sera considéré comme une base de données avec le jeu de caractères par défaut (visible avec « SHOW DATABASES; »).

N'effacez pas le fichier db.opt qui stocke les caractéristiques de la base. Vous pouvez l'ouvrir avec un éditeur de texte pour connaître le jeu de caractères par défaut que MySQL affectera à vos bases

en l'absence de clause **CHARACTER SET**. Souhaitons que ce ne soit pas gb2312 associé par défaut à la collation gb2312_chinese_ci qui vous ferait dire que je vous parle chinois !

Le tableau suivant décrit la création de deux (2) bases de données :

Instruction SQL	Résultat
`CREATE DATABASE bdnouvelle DEFAULT CHARACTER SET ascii COLLATE ascii_general_ci;`	La base `bdnouvelle` est créée, le jeu de caractères par défaut est ASCII.
`CREATE DATABASE bdnouvelle2 DEFAULT CHARACTER SET gb2312;`	La base `bdnouvelle2` est créée pour les Chinois.

Le jeu de caractères par défaut est défini dans **my.ini** à l'aide de la variable *defaultcharacter-set*. Il est donc possible de créer des bases de données associées à différents jeux de caractères au sein d'un même serveur. Le jeu de caractères d'une base définit celui des tables qui seront constituées dedans, à moins que la table ne soit combinée à un autre jeu (créé avec la directive [DEFAULT] CHARACTER SET jeu [COLLATE nomCollation]).

Notons enfin qu'il est même possible d'affecter un jeu de caractères à une colonne d'une table. L'exemple suivant construit la table « testChap5 » dans la base « bdnouvelle2 » (par défaut chinoise) en spécifiant que la colonne « col1 » sera associée au jeu « cp850 : DOS West European », tandis que le reste de la table (pour l'instant de portée col2) sera appliqué au jeu « latin1 : cp1252 West European ». Insérons une ligne.

```
CREATE TABLE bdnouvelle2.testChap5
(col CHAR(5) CHARACTER SET cp850, col2 CHAR(4))
 CHARACTER SET latin1;
INSERT INTO bdnouvelle2.testChap5 VALUES ('GTR','IUT');
```

9.2.2 SÉLECTION D'UNE BASE DE DONNÉES

Pour MySQL, **USE** sélectionne une base de données qui devient active dans une session.

```
USE nomBase;
```

Si vous désirez travailler simultanément dans différentes bases de données, faites toujours préfixer le nom des tables par celui de la base par la notation pointée (nomBase.nomTable).

L'exemple suivant exécute une jointure sur deux (2) tables situées dans deux bases distinctes :

Instruction SQL	Résultat
`CREATE TABLE bdnouvelle.testUSE (col3 CHAR(5), col4 CHAR(4)) CHARACTER SET latin1;` `INSERT INTO bdnouvelle.testUSE VALUES ('ACTMP','IUT');`	Création d'une table dans la base. Insertion d'une ligne.
`USE bdnouvelle2;`	Sélection de la base `bdnouvelle2`.

`SELECT col, bdnouvelle.testUSE.col3 FROM testChap5, bdnouvelle.testUSE WHERE col2 = bdnouvelle.testUSE.col4;`
`+------+-------+`
`| col | col3 |`
`+------+-------+`
`| GTR | ACTMP |`
`+------+-------+`
	Jointure de la table `testChap5` située dans la base active (`bdnouvelle2`) avec `testUSE` située dans la base `bdnouvelle`.

9.2.3 MODIFICATION D'UNE BASE DE DONNÉES

ALTER DATABASE vous permet de modifier le jeu de caractères par défaut d'une base de données. Pour pouvoir changer ainsi une base, vous devez avoir le privilège **ALTER** sur la base de données en question.

```
ALTER {DATABASE nomBase
  [ [DEFAULT] CHARACTER SET nomJeu ]
     [ [DEFAULT] COLLATE nomCollation ];
```

L'instruction suivante modifie la base « chinoise » en lui affectant le jeu de caractères de type DOS.

```
ALTER DATABASE bdnouvelle2 DEFAULT CHARACTER SET cp850;
```

9.2.4 SUPPRESSION D'UNE BASE DE DONNÉES

Pour pouvoir supprimer une base de données, vous devez posséder le privilège **DROP** sur la base (ou au niveau global pour effacer toute base). Cette commande détruit tous les objets (tables, index, etc.) et le répertoire contenu dans la base.

```
DROP {DATABASE | SCHEMA} [IF EXISTS] nomBase;
```

- `IF EXISTS` évite une erreur dans le cas où la base de données n'existerait pas.
- Cette instruction retourne le nombre de tables qui on été supprimées (fichiers à l'extension « `.frm` »).

Disons à présent adios à la base « chinoise » :

```
DROP DATABASE bdnouvelle2;
```

9.3 GESTION DES PRIVILÈGES

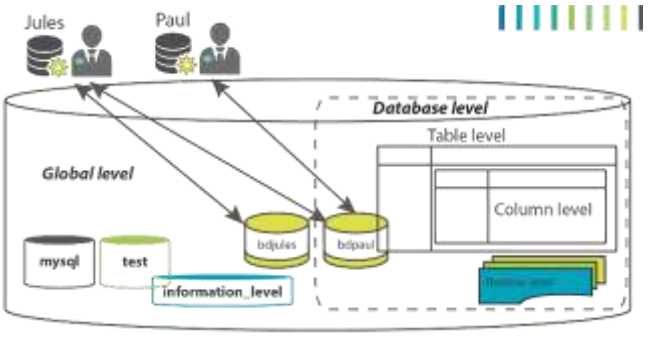
Un **privilège** (sous-entendu utilisateur) **est un droit d'exécuter une certaine instruction SQL** (on parle de privilège système) ou un droit relatif aux données des tables situées dans différentes bases (on parle de privilège objet). La connexion, par exemple, sera considérée comme un privilège système bien que n'étant pas une commande SQL.

Les privilèges système diffèrent sensiblement d'un SGBD à un autre. Chez Oracle, il y en a plus d'une centaine, MySQL est plus modeste en n'en proposant qu'une vingtaine. En revanche, on retrouvera les mêmes privilèges objet (exemple : autorisation de modifier la colonne « nomComp » de la table « Compagnie ») qui sont attribués ou retirés par les instructions **GRANT** et **REVOKE**.

9.3.1 LES NIVEAUX DE PRIVILÈGES

La figure suivante illustre les différents niveaux de privilèges que l'on peut rencontrer :

Figure 16 : Les différents niveaux de privilèges possibles



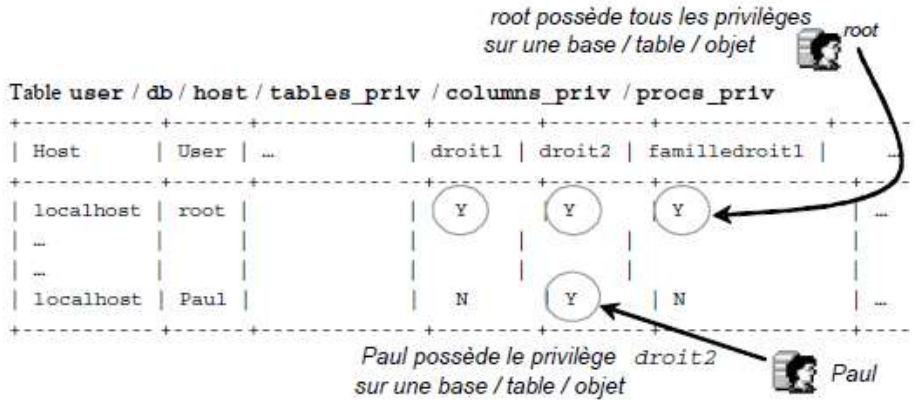
Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition

- **Global level** : privilèges s'appliquant à toutes les bases du serveur. Ces privilèges sont stockés dans la table « mysql.user » (exemple d'attribution d'un privilège global : GRANT CREATE ON *.* …).
- **Database level** : privilèges s'appliquant à tous les objets d'une base de données en particulier. Ces privilèges sont stockés dans les tables « mysql.db » et « mysql.host » (exemple d'attribution d'un privilège database : GRANT SELECT ON bdpaul.* …).
- **Table level** : privilèges s'appliquant à la globalité d'une table d'une base de données en particulier. Ces privilèges sont stockés dans la table « mysql.tables_priv » (exemple d'attribution d'un privilège table : GRANT INSERT ON bdpaul.Avion …).
- **Column level** : privilèges s'appliquant à une des colonnes d'une table d'une base de données en particulier. Ces privilèges sont stockés dans la table « mysql.columns_priv » (exemple d'attribution d'un privilège column : GRANT UPDATE(nomComp) ON bdpaul.Compagnie …).
- **Routine level** : privilèges globaux ou au niveau d'une base (CREATE ROUTINE, ALTER ROUTINE, EXECUTE, et GRANT) s'appliquant aux procédures cataloguées. Ces privilèges sont stockés dans la table « mysql.procs_priv » de la base « mysql » (exemple d'attribution d'un privilège routine : GRANT EXECUTE ON PROCEDURE bdpaul.sp1…).

9.3.2 LES TABLES DE LA BASE MYSQL

Cinq (5) tables de la base de données mysql suffisent à MySQL pour stocker les privilèges (système et objet) de tous les utilisateurs. La figure suivante illustre comment MySQL déduit toutes ces prérogatives toujours en fonction des accès (couple utilisateur, machine).

Figure 17 : Illustration des prérogatives de stockage des privilèges d'accès



Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition

La colonne « Db » est en plus dans les tables « host », « tables_priv » et « columns_priv », car elle est nécessaire pour désigner la base de données sur laquelle portera le droit ou la famille de droits.

Supposons, pour nos exemples, que l'utilisateur Paul (accès en local) et la base de données « bdpaul » soient créés.

```
CREATE DATABASE bdpaul;
CREATE USER Paul@localhost IDENTIFIED BY 'iut';
```

9.3.3 TABLE MYSQL.USER

Cette table est composée de 37 colonnes qui décrivent les privilèges au niveau global du serveur.

Privilèges objet (LMD) sur toutes les bases de données

La requête suivante extrait les prérogatives de Paul (et des autres). Pour l'instant, le caractère « N » étant dans toutes les colonnes, il ne peut ni interroger une table (Select_priv), ni insérer dans une table (Insert_priv), ni en modifier (Update_priv), ni en supprimer (Delete_priv) et ce quelle que soit la base de données (excepté les bases système test et information_schema) sur laquelle il voudra se connecter.

```
SELECT Host, User, Select_priv, Insert_priv, Update_priv, Delete_priv
       FROM mysql.user;
+-----------+------+-------------+-------------+-------------+-------------+
| Host      | User | Select_priv | Insert_priv | Update_priv | Delete_priv |
+-----------+------+-------------+-------------+-------------+-------------+
| localhost | root | Y           | Y           | Y           | Y           |
| localhost |      | Y           | Y           | Y           | Y           |
| %         |      | N           | N           | N           | N           |
| localhost | Paul | N           | N           | N           | N           |
+-----------+------+-------------+-------------+-------------+-------------+
```

Vous pouvez, par analogie, pour cet exemple et pour les suivants, découvrir les prérogatives des autres accès (ici root et anonyme).

Privilèges objet (LDD) sur toutes les bases de données

La requête suivante extrait les prérogatives à propos des instructions LDD. Pour l'instant, le caractère « N » étant dans toutes les colonnes, Paul ne peut ni créer une table ou une base (Create_priv), ni en supprimer (Drop_priv), ni créer ou supprimer un index (Index_priv), ni modifier la structure d'une table, la renommer ou modifier une base (Alter_priv), et ce quelle que soit la base de données (excepté les bases système test et information_schema).

```
SELECT Host, User, Create_priv, Drop_priv,Index_priv, Alter_priv
       FROM mysql.user;
+-----------+------+-------------+-----------+------------+------------+
| Host      | User | Create_priv | Drop_priv | Index_priv | Alter_priv |
+-----------+------+-------------+-----------+------------+------------+
| localhost | root | Y           | Y         | Y          | Y          |
| localhost |      | Y           | Y         | Y          | Y          |
| %         |      | N           | N         | N          | N          |
| localhost | Paul | N           | N         | N          | N          |
+-----------+------+-------------+-----------+------------+------------+
```

Privilèges système (LCD) sur toutes les bases de données

La requête suivante extrait les prérogatives à propos des instructions LCD. Pour l'instant, le caractère « N » étant dans toutes les colonnes, Paul ne peut ni créer un utilisateur (Create_user_priv), ni transmettre des droits qu'il aura lui-même reçus (Grant_priv), ni lister les bases de données existantes (Show_db_priv), et ce quelle que soit la base de données.

```
SELECT Host,User, Create_user_priv, Grant_priv, Show_db_priv FROM mysql.user;
+-----------+------+-----------------+------------+--------------+
| Host      | User | Create_user_priv | Grant_priv | Show_db_priv |
+-----------+------+-----------------+------------+--------------+
| localhost | root | Y               | Y          | Y            |
| localhost |      | N               | Y          | Y            |
| %         |      | N               | N          | N            |
| localhost | Paul | N               | N          | N            |
+-----------+------+-----------------+------------+--------------+
```

Privilèges à propos des vues sur toutes les bases de données

La requête suivante extrait les prérogatives à propos des instructions relatives aux vues (views détaillées dans la section suivante). Pour l'instant, le caractère « N » étant dans toutes les colonnes, Paul ne peut ni créer une vue (Create_view_priv), ni lister les vues existantes (Show_view_priv), et ce quelle que soit la base de données.

```
SELECT Host,User, Create_view_priv, Show_view_priv FROM mysql.user;
+-----------+------+------------------+-----------------+
| Host      | User | Create_view_priv | Show_view_priv |
+-----------+------+------------------+-----------------+
| localhost | root | Y                | Y               |
| localhost |      | N                | N               |
| %         |      | N                | N               |
| localhost | Paul | N                | N               |
+-----------+------+------------------+-----------------+
```

Privilèges à propos des procédures cataloguées sur toutes les bases de données

La requête suivante extrait les prérogatives à propos des procédures cataloguées. Pour l'instant, le caractère « N » étant dans toutes les colonnes, Paul ne peut ni créer une procédure (Create_routine_priv), ni en modifier (Alter_routine_priv), ni en exécuter (Execute_priv), et ce quelle que soit la base de données.

```
SELECT Host,User, Create_routine_priv, Alter_routine_priv, Execute_priv
       FROM mysql.user;
+-----------+------+---------------------+--------------------+--------------+
| Host      | User | Create_routine_priv | Alter_routine_priv | Execute_priv |
+-----------+------+---------------------+--------------------+--------------+
| localhost | root | Y                   | Y                  | Y            |
| localhost |      | N                   | N                  | Y            |
| %         |      | N                   | N                  | N            |
| localhost | Paul | N                   | N                  | N            |
+-----------+------+---------------------+--------------------+--------------+
```

Privilèges à propos des restrictions d'utilisateur

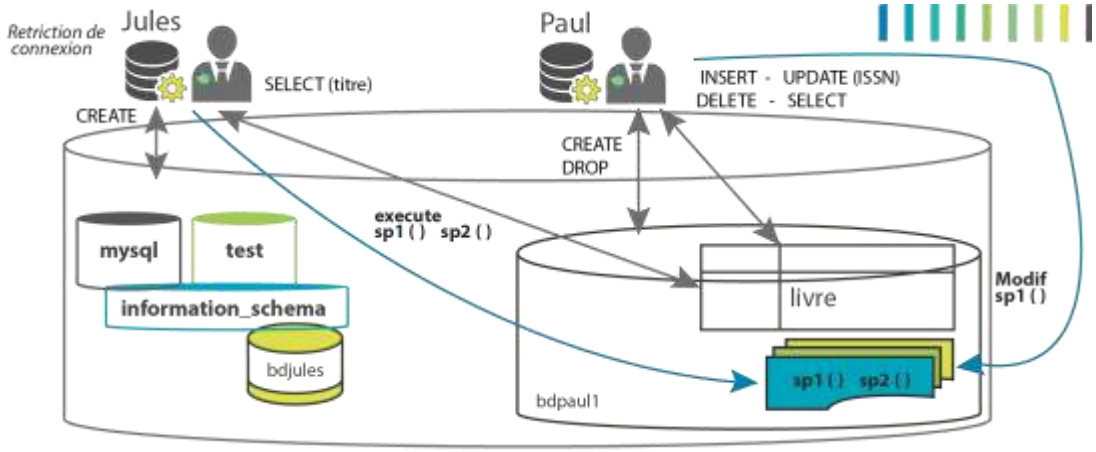
La requête suivante extrait les prérogatives à propos des restrictions qu'on peut définir par accès. Pour l'instant, le chiffre étant à « 0 » dans toutes les colonnes, aucun accès (utilisateur) n'est limité concernant le nombre de requêtes (max_questions), de modifications (max_updates), de connexions par heure (max_connections) et de connexions simultanées (max_user_connections) à un serveur.

```
SELECT Host, User, max_questions "Requetes", max_updates "Modifs" ,
       max_connections "Connexions", max_user_connections "Cx simult."
       FROM mysql.user;
+-----------+------+----------+--------+------------+------------+
| Host      | User | Requetes | Modifs | Connexions | Cx simult. |
+-----------+------+----------+--------+------------+------------+
| localhost | root |        0 |      0 |          0 |          0 |
| localhost |      |        0 |      0 |          0 |          0 |
| %         |      |        0 |      0 |          0 |          0 |
| localhost | Paul |        0 |      0 |          0 |          0 |
+-----------+------+----------+--------+------------+------------+
```

9.3.4 ATTRIBUTION DE PRIVILÈGES (GRANT)

La figure suivante illustre le contexte qui va servir d'exemple à l'attribution de prérogatives.

Figure 18 : Exemple d'un contexte d'attribution de prérigatives



Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition

L'instruction **GRANT** permet d'attribuer un (ou plusieurs) privilège(s) à propos d'un objet à un (ou plusieurs) bénéficiaire(s). L'utilisateur qui exécute cette commande doit avoir reçu lui-même le droit de transmettre ces privilèges (reçu avec la directive GRANT OPTION). Dans le cas de root, aucun problème, car il a implicitement tous les droits.

```
GRANT privilège [ (col1 [, col2...])] [,privilège2 ... ]
ON [ {TABLE | FUNCTION | PROCEDURE} ]
{nomTable | * | *.* | nomBase.*}
TO utilisateur [IDENTIFIED BY [PASSWORD] 'password']
[,utilisateur2 ...]
[ WITH [ GRANT OPTION ]
[ MAX_QUERIES_PER_HOUR nb ]
[ MAX_UPDATES_PER_HOUR nb2 ]
[ MAX_CONNECTIONS_PER_HOUR nb3 ]
[ MAX_USER_CONNECTIONS nb4 ] ];
```

- **Privilège** : description du privilège (exemple : SELECT, DELETE, etc.), voir le tableau suivant.
- **Col** : précise la ou les colonnes sur lesquelles se portent les privilèges SELECT, INSERT ou UPDATE (exemple : UPDATE(typeAvion) pour n'autoriser que la modification de la colonne typeAvion).
- **GRANT OPTION** : permet de donner le droit de retransmettre les privilèges reçus à une tierce personne.

Le tableau suivant résume la signification des principaux privilèges à accorder ou à révoquer.

Tableau 14 : Signification des principaux privilèges à accorder ou à révoquer

privilege	Commentaire
ALL [PRIVILEGES]	Tous les privilèges
ALTER	Modification de base/table
ALTER ROUTINE	Modification de procédure
CREATE	Création de base/table
CREATE ROUTINE	Création de procédure
CREATE USER	Création d'utilisateur
CREATE VIEW	Création de vue
DELETE	Suppression de données de table
DROP	Suppression de base/table
EXECUTE	Exécution de procédure
INDEX	Création/Suppression d'index
INSERT	Insertion de données de table
SELECT	Extraction de données de table
SHOW DATABASES	Lister les bases
SHOW VIEW	Lister les vues d'une base
SUPER	Gestion des déclencheurs
UPDATE	Modification de données de table
USAGE	Synonyme de « sans privilège », USAGE est utilisé pour conserver les privilèges précédemment définis tout en les restreignant avec des options

Exemples

Le tableau suivant décrit l'affectation de quelques privilèges en donnant les explications associées.

Tableau 15 : Description des affectations de privilèges

Instruction faite par `root`	Explication
`GRANT CREATE, DROP ON bdpaul.* TO 'Paul'@'localhost';`	Privilège système *database level* : `Paul` (en accès local) peut créer ou supprimer des tables dans la base `bdpaul`.
`GRANT INSERT, SELECT, DELETE, UPDATE(ISBN) ON bdpaul.Livre TO 'Paul'@'localhost';`	Privilège objet *table level* : `Paul` peut insérer, extraire, supprimer et modifier la colonne `ISBN` de la table `Livre` contenue dans la base `bdpaul`.
`GRANT ALTER ON bdpaul.Livre TO 'Paul'@'localhost';`	Privilège système *table level* : `Paul` peut modifier la structure ou les contraintes de la table `Livre` contenue dans la base `bdpaul`.
`GRANT SELECT(titre) ON bdpaul.Livre TO 'Jules'@'localhost' WITH GRANT OPTION;`	Privilège objet *column level* : `Jules` peut extraire seulement la colonne titre de la table `Livre` contenue dans la base `bdpaul`. Il pourra par la suite retransmettre éventuellement ce droit.
`GRANT CREATE ON *.* TO 'Jules'@'localhost';`	Privilège système *global level* : `Jules` peut créer des bases de données.
`GRANT USAGE ON bdpaul.* TO 'Jules'@'localhost' WITH MAX_QUERIES_PER_HOUR 50 MAX_UPDATES_PER_HOUR 20 MAX_CONNECTIONS_PER_HOUR 6 MAX_USER_CONNECTIONS 3;`	Privilège système *database level* : `Jules` ne peut lancer, chaque heure, que 50 `SELECT`, 20 `UPDATE`, se connecter 6 fois (dont 3 connexions simultanées) sur la base de données `bdpaul`.

Tout ce que vous avez le droit de faire doit être explicitement autorisé par la commande GRANT. Ce qui n'est pas dit par **GRANT** n'est pas permis. Par exemple, Jules peut créer des bases, mais pas en détuire, Paul peut modifier le numéro ISBN d'un livre mais pas son titre, etc.

Voir les privilèges

La commande **SHOW GRANTS FOR** liste les différentes instructions GRANT équivalentes à toutes les prérogatives d'un utilisateur donné. C'est bien utile quand vous avez attribué un certain nombre de privilèges à un utilisateur sans avoir pensé à les consigner dans un fichier de commande.

```
SHOW GRANTS FOR utilisateur;
```

Utilisons cette commande pour extraire les profils de Jules, de Paul et de root (accès en local).

9.3.5 RÉVOCATION DE PRIVILÈGES (REVOKE)

La révocation d'un ou de plusieurs privilèges est réalisée par l'instruction **REVOKE**. Pour pouvoir révoquer un privilège, vous devez détenir (avoir reçu) au préalable ce même privilège avec l'option **WITH GRANT OPTION**.

Dans la syntaxe suivante, les options sont les mêmes que pour la commande GRANT.

```
REVOKE privilège [ (col1 [, col2...])] [,privilège2 ... ]
ON [ {TABLE | FUNCTION | PROCEDURE} ]
{nomTable | * | *.* | nomBase.*}
FROM utilisateur [,utilisateur2 ... ];
```

Exemples

Le tableau suivant décrit la révocation de certains privilèges acquis des utilisateurs Paul et Jules.

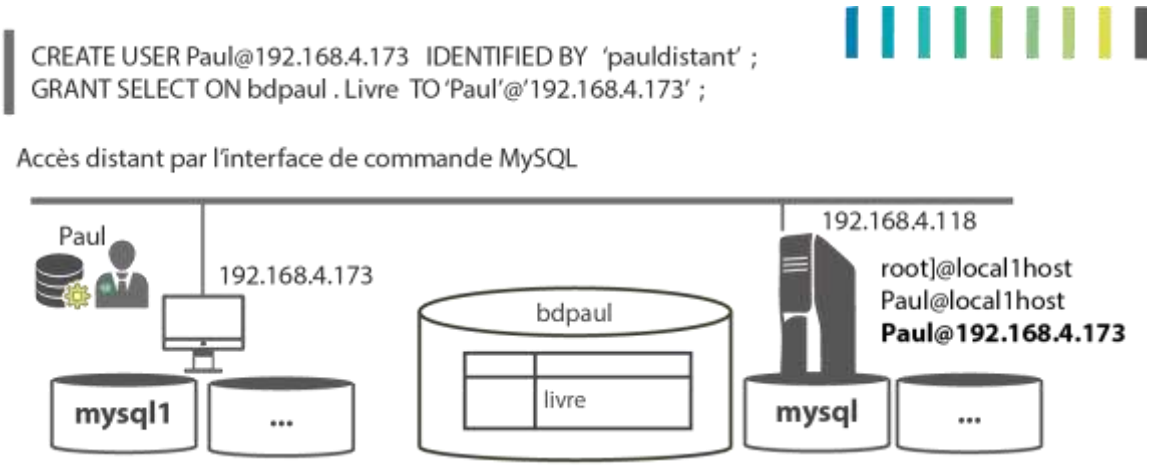
Figure 19 : Exemple de révocation de privilèges

Instruction faite par `root`	Explication
`REVOKE CREATE ON bdpaul.* FROM 'Paul'@'localhost';`	Privilège système *database level* : `Paul` (en accès local) ne peut plus créer de tables dans la base `bdpaul`.
`REVOKE ALTER, INSERT, UPDATE(ISBN) ON bdpaul.Livre FROM 'Paul'@'localhost';`	Privilège objet *table level* : `Paul` ne peut plus modifier la structure (ou les contraintes), insérer et modifier la colonne `ISBN` de la table `Livre` contenue dans la base `bdpaul`.
`GRANT USAGE ON bdpaul.* TO 'Jules'@'localhost' WITH MAX_QUERIES_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0;`	Privilège système *database level* : `Jules` n'est plus limité en requêtes `SELECT` et `UPDATE` sur la base de données `bdpaul`. Ici c'est un `GRANT` qu'il faut faire, car il s'agit plus d'une restriction de connexion que d'une instruction SQL.

9.4 ACCÈS DISTANTS

La figure suivante illustre la configuration du test : Un client est en 192.168.4.173 sur lequel sont installées les couches MySQL. Un serveur est en 192.168.4.118 équipé de MySQL Complete Package. Sur le serveur, root crée un accès à Paul, en précisant l'adresse de la machine client et lui attribue un droit d'extraction de la table « Livre » sur la base « bdPaul ».

Figure 20 : Exemple de configuration du test



Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition

Connexion par l'interface de commande

Sur le client, Paul se connecte au serveur dans une fenêtre de commande, en précisant l'adresse de la machine serveur. Ensuite, il donne son mot de passe distant. Pensez à enlever les pares-feu

Windows sur le client et le serveur (bloquant le port 3306).

```
mysql -h 192.168.4.118 -u Paul -p
```

Paul peut, à présent seulement, interroger la table distante, comme le montre la copie d'écran suivante :

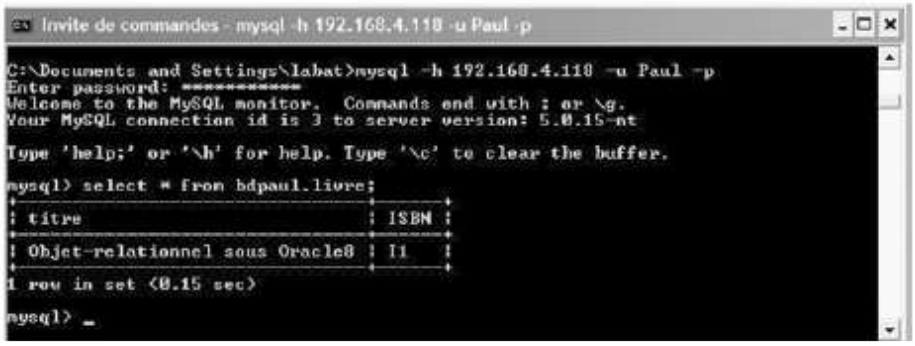


Table mysql.host

La table « mysql.host » est utilisée conjointement avec « mysql.db » et concerne les accès distants (plusieurs machines). Cette table n'est employée que pour les prérogatives au niveau database, indépendamment des utilisateurs. La structure est la même que celle de « mysql.db », à l'exception de la colonne « User » qui n'est pas présente. Le couple de colonnes (Host, Db) est unique.

Tableau 16 : Table de prérogatives au niveau datadase

Caractère	Signification pour `mysql.db`		Signification pour `mysql.host`	
	colonne `Host`	colonne `Db`	colonne `Host`	colonne `Db`
`%`	toute machine	toute base	toute machine	toute base
`' '` (chaîne vide)	consultez la table `mysql.host`	toute base	toute machine	toute base

La table « mysql.host » n'est mise à jour ni par GRANT, ni par REVOKE. Il faudra directement insérer (par INSERT), modifier (par UPDATE) ou supprimer (par DELETE) les lignes de cette table.

Elle n'est pas utilisée par la plupart des serveur MySQL car elle est dédiée à des usages très spécifiques (pour gérer un ensemble de machines à accès sécurisé, par exemple). Elle peut aussi être utilisée pour définir un ensemble de machines à accès non sécurisé.

9.5 LES VUES

Pour pouvoir créer une vue dans une base, vous devez posséder le privilège **CREATE VIEW** et les privilèges en **SELECT** des tables présentes dans la requête de définition de la vue. La syntaxe SQL de création d'une vue est la suivante :

```
CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    VIEW [nomBase.]nomVue [(listecolonnes)]
    AS requêteSELECT
    [WITH [CASCADED | LOCAL] CHECK OPTION];
```

Les vues peuvent également servir pour simplifier l'écriture de requêtes complexes, renforcer la confidentialité et une certaine intégrité. Vous devez posséder le privilège **DROP** sur une vue pour pouvoir la supprimer. La suppression d'une vue n'entraîne pas la destruction des données qui résident toujours dans les tables.

9.6 DICTIONNAIRE DES DONNÉES

Le dictionnaire des données (metadata ou data dictionary) est une partie majeure d'une base de données MySQL qu'on peut assimiler à une structure centralisée.

Pour MySQL, 16 vues sont issues de tables système non visibles. Ces vues, qui sont appelées « tables » par abus de langage (dans la documentation officielle, dans les livres et sur les forums) et qui sont situées dans la base INFORMATION_SCHEMA, suffisent à stocker toutes les informations décrivant tous les objets contenus dans toutes les bases de données. MySQL se rapproche davantage de DB2 et de SQL Server que d'Oracle (qui possède 600 vues). Les noms des vues sont similaires et se rapprochent de la spécification ANSI/ISO SQL:2003 standard Part 11 Schemata.

Le dictionnaire des données contient :

- La définition des tables, vues, index, séquences, procédures, fonctions et déclencheurs ;
- La description de l'espace disque alloué et occupé par chaque objet ;
- Les valeurs par défaut des colonnes (DEFAULT) ;
- La description des contraintes d'intégrité référentielle (de vérification à venir) ;
- Le nom des utilisateurs de la base ;
- Les privilèges pour chaque utilisateur ;
- Des informations d'audit (accès aux objets) et d'autre nature (commentaires, par exemple).

9.7 EXERCICES

Les objectifs de ces exercices sont :

- De créer des vues monotables et multitables ;
- D'afficher les Bases de données du serveur ;
- D'extraire la composition d'une base de données ;
- D'avoir le détail de stockage d'une base de données ;
- De déterminier les trivilèges des utilisateurs d'une base de données.

Exercice 11 :

Écrire le script « vues.sql », permettant de créer :

- La vue LogicielsUnix qui contient tous les logiciels de type « UNIX » (toutes les colonnes sont conservées). Vérifier la structure et le contenu de la vue (DESCRIBE et SELECT).
- La vue « Poste_0 » de structure (nPos0, nomPoste0, nSalle0, TypePoste0, indIP, ad0) qui contient tous les postes du rez-de-chaussée (etage=0 au niveau de la table Segment). Faire une jointure procédurale, sinon la vue sera considérée comme une vue multitable. Vérifier la structure et le contenu de la vue.

MODULE 3 : SÉCURISATION DES BASES DE DONNÉES EN LIGNE

1. MOTIVATIONS ET INTRODUCTION

La sécurité des bases de données se définit comme l'ensemble des mécanismes et dispositions protégeant la base de données contre les effets des menaces accidentelles ou intentionnelles.

La base de données est au cœur du Système d'Information (SI). Toute attaque contre les données met en danger le SI lui-même et par là, l'organisation toute entière.

L'accès aux bases de données via le web augmente considérablement les menaces (de 100 utilisateurs internes identifiés à de centaines de millions d'utilisateurs anonymes et incontrôlés).

Cinq (5) dangers spécifiques existent :

1. Perte d'intégrité des données ;
2. Non disponibilité des données ;
3. Perte de confidentialité des données ;
4. Perte de protection de données privées (privacy) ;
5. Vol et fraudes.

La sécurité des bases de données est un domaine complexe, polymorphe, pour lequel il n'existe pas encore d'étude intégrée et homogène. Il n'existe pas encore de solution globale ni de recettes clé-sur-porte.

Les objectifs de ce troisième module sont de :

- Présenter des concepts de base ;
- Présenter la méthode et la nature des accès à une base de données en ligne afin de définir une politique de sécurité adaptée ;
- Analyser des risques de sécurité en amont ;
- Identifier et descrire quelques failles les plus fréquemment rencontrées ;
- Présenter quelques solutions adéquates afin d'assurer la sécurisation des SGDB ;
- Superviser la sécurité et connaître les outils d'écoute et d'analyse du trafic réseau ;
- Présenter différents protocoles de chiffrement ainsi que leurs forces et faiblesse.

1.1 LES CRITÈRES FONDAMENTAUX DE LA SÉCURITÉ DE L'INFOMATION

Quatre (4) critères sont retenus et concernent des caractéristiques que le propriétaire ou le gestionnaire de l'information veut voir réalisées afin de s'assurer que la sécurité est au rendez-vous. Ils sont connus sous le nom de **D.C.I.P.** :

Disponibilité : Propriété ***d'accessibilité au moment voulu*** des biens par les personnes autorisées (i.e. le bien doit être disponible durant les plages d'utilisation prévues).

Intégrité : Propriété d'***exactitude*** et de ***complétude*** des biens et informations (i.e. une modification illégitime d'un bien doit pouvoir être détectée et corrigée).

Confidentialité : Propriété des biens de n'être accessibles qu'aux personnes autorisées.

Preuve : Propriété d'un bien permettant de retrouver, avec ***une confiance suffisante***, les circonstances dans lesquelles ce bien évolue.

Cette propriété englobe notamment :

- La traçabilité des actions menées ;
- L'authentification des utilisateurs ;
- L'imputabilité du responsable de l'action effectuée.

1.2 MÉCANISMES DE SÉCURITÉ POUR ATTEINDRE LES BESOINS DICP

Un Système d'Information a besoin de mécanismes de sécurité qui ont pour objectif d'assurer de garantir les propriétés DICP sur les biens de ce SI. Voici quelques exemples de mécanismes de sécurité participant à cette garantie :

Tableau 17 : Mécanismes de sécurité pour assurer les propriétés DICP

Critères		D	I	C	P
Anti-virus	Mécanisme technique permettant de détecter toute attaque virale qui a déjà été identifiée par la communauté sécurité	√	√	√	
Cryptographie	Mécanisme permettant d'implémenter du chiffrement et des signatures électroniques		√	√	√
Pare-feu	Équipement permettant d'isoler des zones réseaux entre-elles et de n'autoriser que le passage que de certains flux seulement	√		√	
Contrôles d'accès logiques	Mécanismes permettant de restreindre l'accès en lecture / écriture / suppression aux ressources aux seules personnes dument habilitées		√	√	√
Sécurité physique des équipements et locaux	Mécanismes de protection destinés à protéger l'intégrité physique du matériel et des bâtiments/bureaux	√	√	√	
Capacité d'audit	Mécanismes organisationnels destinés à s'assurer de l'efficacité et de la pertinence des mesures mises en œuvre. Participe à l'amélioration continue de la sécurité du SI	√	√	√	√
Clauses contractuelles avec les partenaires	Mécanismes organisationnels destinés à s'assurer que les partenaires et prestataires mettent en œuvre les mesures nécessaires pour ne pas impacter la sécurité des S.I. de leurs clients	√	√	√	√
Formation et sensibilisation	Mécanismes organisationnels dont l'objectif est d'expliquer aux utilisateurs, administrateurs, techniciens, PDG, clients, grand public, etc. en quoi leurs actions affectent la sécurité des SI Diffusion des bonnes pratiques de sécurité	√	√	√	√

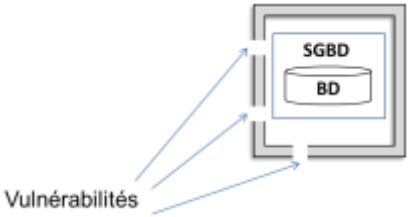
2. VULNÉRABILITÉ, MENACE, ATTAQUE

2.1 NOTIONS DE VULNÉRABILITÉ, MENACE, ATTAQUE

2.1.1 VULNÉRABILITÉ

Une vulnérabilité est une faiblesse au niveau d'un bien (au niveau de la conception, de la réalisation, de l'installation, de la configuration ou de l'utilisation du bien).

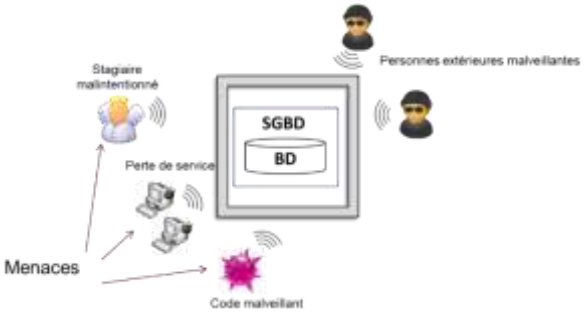
Figure 21 : Vulnérabilité



2.1.2 MENACE

Une menace est une cause potentielle d'un incident, qui pourrait entrainer des dommages sur un bien si cette menace se concrétisait.

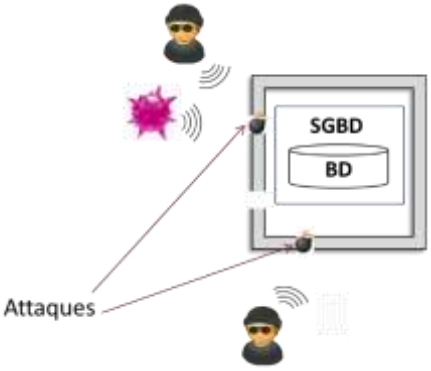
Figure 22 : Menace

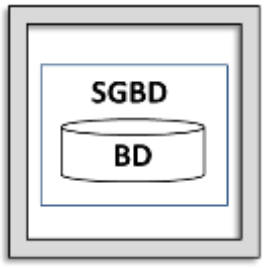


2.1.3 ATTAQUE

Une attaque est une action malveillante destinée à porter atteinte à la sécurité d'un bien. Une attaque représente la concrétisation d'une menace, et nécessite l'exploitation d'une vulnérabilité.

Figure 23 : Attaque



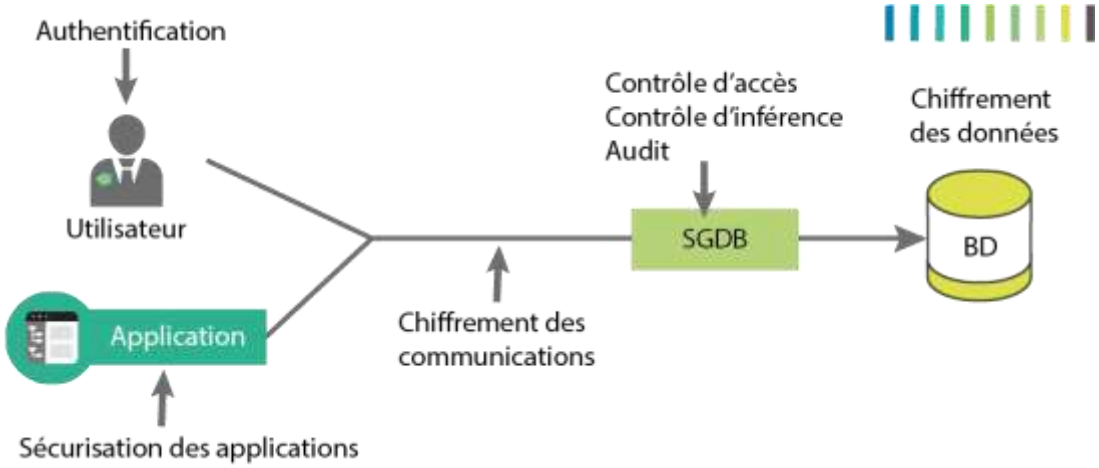


Une attaque ne peut donc avoir lieu (et réussir) que si le bien est affecté par une vulnérabilité.

Ainsi, tout le travail des experts « sécurité » consiste à s'assurer que le SI ne possède aucune vulnérabilité. Dans la réalité, l'objectif est en fait d'être en mesure de maitriser ces vulnérabilités plutôt que de viser un objectif « zéro » inatteignable.

Les protections contre les attaques se situent à plusieurs niveaux selon la figure suivante :

Figure 24 : Niveaux de protection contre les attaques



Source : Assistance Technique - Plateforme Nationale d'Information pour la Nutrition

2.2 FAILLES DE SÉCURITÉ LES PLUS FRÉQUEMMENT RENCONTRÉES

2.2.1 POUR LES APPLICATIONS WEB

L'Open Web Application Security (OWASP) est un organisme à but non lucratif mondial qui milite pour l'amélioration de la sécurité des logiciels. L'objectif est d'informer les individus ainsi que les entreprises sur les risques liés à la sécurité des Systèmes d'Information (SI). L'organisation fonctionne comme une communauté de professionnels qui partagent la même vision des choses. Tout le monde est libre de rejoindre la communauté qui compte aujourd'hui plus de 45 000 membres.

Chaque année OWASP publie un classement qui recense les failles de sécurité les plus critiques. Les failles de sécurité les plus critiques sont présentées ci-dessous (2017).

1. Injection

Des failles d'injection, telles que l'injection SQL, NoSQL, OS et LDAP, se produisent lorsque des données non approuvées sont envoyées à un interpréteur dans le cadre d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent inciter l'interpréteur à exécuter des commandes involontaires ou à accéder aux données sans autorisation appropriée.

2. Authentification cassée

Les fonctions applicatives liées à l'authentification et à la gestion de session sont souvent mal implémentées, ce qui permet aux attaquants de compromettre les mots de passe, les clés ou les jetons de session ou d'exploiter d'autres failles d'implémentation pour assumer temporairement ou définitivement l'identité des autres utilisateurs.

3. Exposition de données sensibles

De nombreuses applications Web et API ne protègent pas correctement les données sensibles, telles que les données financières, les soins de santé et les informations personnelles. Les attaquants peuvent voler ou modifier ces données faiblement protégées pour mener une fraude par carte de crédit, un vol d'identité ou d'autres crimes. Les données sensibles peuvent être compromises sans protection supplémentaire, tel que le cryptage au repos ou en transit et nécessitent des précautions particulières lorsqu'elles sont échangées avec le navigateur.

4. Entités externes XML (XXE)

De nombreux processeurs XML plus anciens ou mal configurés évaluent les références d'entités externes dans les documents XML. Les entités externes peuvent être utilisées pour divulguer des fichiers internes à l'aide du gestionnaire d'URI (Uniform Resource Identifier, en francais : identifiant uniforme de ressource) de fichier, des partages de fichiers internes, de l'analyse des ports internes, de l'exécution de code à distance et des attaques par déni de service.

5. Contrôle d'accès cassé

Les restrictions sur ce que les utilisateurs authentifiés sont autorisés à faire ne sont souvent pas correctement appliquées. Les attaquants peuvent exploiter ces failles pour accéder à des fonctionnalités et / ou des données non autorisées, comme accéder aux comptes d'autres utilisateurs, afficher des fichiers sensibles, modifier les données d'autres utilisateurs, modifier les droits d'accès, etc.

6. Mauvaise configuration de la sécurité

Une mauvaise configuration de la sécurité est le problème le plus courant. Cela est généralement le résultat de configurations par défaut non sécurisées, de configurations incomplètes ou ad hoc, d'un stockage en nuage ouvert, d'en-têtes HTTP mal configurés et de messages d'erreur détaillés contenant des informations sensibles. Non seulement tous les systèmes d'exploitation, frameworks, bibliothèques et applications doivent être configurés de manière sécurisée, mais ils doivent également être corrigés / mis à niveau en temps opportun.

7. Scripts intersites (XSS)

Des failles XSS se produisent chaque fois qu'une application inclut des données non approuvées dans une nouvelle page Web sans validation ou échappement appropriée ou met à jour une page Web existante avec des données fournies par l'utilisateur à l'aide d'une API de navigateur qui peut créer du HTML ou du JavaScript. XSS permet aux attaquants d'exécuter des scripts dans le navigateur de la victime qui peuvent détourner les sessions des utilisateurs, dégrader des sites Web ou rediriger l'utilisateur vers des sites malveillants.

8. Désérialisation non sécurisée

Une désérialisation non sécurisée conduit souvent à l'exécution de code à distance. Même si les failles de désérialisation n'entraînent pas l'exécution de code à distance, elles peuvent être utilisées pour effectuer des attaques, y compris des attaques de relecture, des attaques par injection et des attaques par élévation de privilèges.

La sérialisation est un processus qui convertit l'objet en un format de données de structure spécifique, tel que la conversion de la classe Java Entity au format JSON pour l'envoi via la communication avec d'autres services ou clients.

La désérialisation est un processus qui convertit le format de données en objet tel que le client envoie des demandes au format de données JSON et le service back-end le convertit en classe d'entité Java.

9. Utilisation de composants avec des vulnérabilités connues

Les composants, tels que les bibliothèques, les frameworks et d'autres modules logiciels, s'exécutent avec les mêmes privilèges que l'application. Si un composant vulnérable est exploité, une telle attaque peut faciliter de graves pertes de données ou une prise de contrôle du serveur. Les applications et les API utilisant des composants avec des vulnérabilités connues peuvent saper les défenses des applications et permettre diverses attaques et impacts.

10. Journalisation et surveillance insuffisantes

Une journalisation et une surveillance insuffisante, associées à une intégration manquante ou inefficace avec la réponse aux incidents, permettent aux attaquants d'attaquer davantage les systèmes, de maintenir la persistance, de basculer vers plus de systèmes et de falsifier, extraire ou détruire les données. La plupart des études sur les violations montrent que le délai de détection d'une violation est supérieur à 200 jours, généralement détecté par des parties externes plutôt que par des processus ou une surveillance interne.

2.2.2 LES 10 PRINCIPALES MENACES DE SÉCURITÉ DES BASES DE DONNÉES

1. Abus de privilège excessif

Lorsque les utilisateurs (ou les applications) ont des privilèges d'accès à une base de données excédant les exigences de leur fonction professionnelle, ils peuvent abuser de ces privilèges à des fins malveillantes. L'utilisateur d'une base de données peut se retrouver avec des privilèges excessifs pour la simple raison que les administrateurs de bases de données n'ont pas le temps de définir ni de mettre à jour des mécanismes de contrôle granulaire (acces limité dans le temps et non global) des privilèges d'accès pour chaque utilisateur. Par conséquent, tous les utilisateurs ou les grands groupes d'utilisateurs ont des privilèges d'accès génériques définis par défaut qui excèdent largement les exigences de leur fonction spécifique.

2. Abus de privilège légitime

Les utilisateurs peuvent également abuser de privilèges légitimes d'accès à une base de données à des fins non autorisées.

3. Elévation de privilège

Les auteurs d'attaques peuvent profiter des vulnérabilités des logiciels plate-forme de bases de données pour transformer les privilèges d'accès d'un utilisateur ordinaire en ceux d'un administrateur. Les vulnérabilités peuvent se trouver dans les procédures enregistrées, les fonctions intégrées, les implémentations de protocoles, voire même dans les données SQL.

4. Exploitation de failles des bases de données vulnérables ou mal configurées

Les bases de données sont souvent vulnérables, non corrigées ou disposent de comptes et d'une configuration toujours définis par défaut. L'auteur d'une attaque qui tente d'exploiter la base de données teste généralement les systèmes sur ces vulnérabilités, ce qui peut entraîner une violation de sécurité. Alors que les fournisseurs développent des packs de correction pour corriger les systèmes au niveau d'une vulnérabilité spécifique, les bases de données des entreprises demeurent librement exploitables. Lorsqu'un correctif est lancé, il n'est pas disponible immédiatement. Il existe différents aspects à prendre en considération au moment d'appliquer un correctif sur une base de données. Tout d'abord, l'organisation doit au préalable évaluer la procédure de correction du système avec le correctif en question, en tentant de comprendre comment le correctif affecterait le système. Parfois, un correctif peut être en contradiction avec un code déjà existant ou il peut impliquer d'autres opérations. Ensuite, le système subit un temps d'arrêt lorsque le serveur de bases de données ne parvient pas à fournir aux utilisateurs un service afin de le corriger. Enfin, les grandes entreprises ayant des dizaines voire des centaines de bases de données doivent prévoir un plan de correction, en traitant en priorité les bases de données, celles-ci devant être corrigées en premier. Les accès aux bases de données, les administrateurs système et les administrateurs informatiques, les développeurs, tous jouent un rôle dans la procédure de correction. Alors que les ressources et le temps sont limités, les serveurs demeurent vulnérables pendant des mois après le lancement d'un correctif.

5. Injection SQL

Dans une attaque par injection SQL, l'auteur insère généralement (ou « injecte ») des informations de bases de données non autorisées dans une chaîne de données SQL vulnérable. Généralement les chaînes de données visées incluent les procédures enregistrées et les paramètres d'entrée des applications Web. Ces informations injectées sont ensuite envoyées vers la base de données où elles sont exécutées. En utilisant l'injection SQL, les auteurs d'attaques peuvent obtenir l'accès illimité à l'ensemble d'une base de données.

6. Faiblesse de l'audit natif

Un enregistrement automatique de toutes les transactions de bases de données sensibles et / ou inhabituelles devrait être la base sous-jacente de tout déploiement de base de données. Une faible règle d'audit des bases de données représente un sérieux risque organisationnel à plusieurs niveaux.

7. Déni de service

Le déni de service (DOS - Denial Of Service) est une catégorie d'attaque générale par laquelle l'accès aux applications réseau est refusé à certains utilisateurs. Les conditions de déni de service peuvent être créées par de nombreuses techniques, parmi lesquelles beaucoup sont liées aux vulnérabilités mentionnées précédemment. Par exemple, le déni de service peut être obtenu en profitant de la vulnérabilité d'une plate-forme de bases de données pour faire tomber un serveur. D'autres techniques communes de déni de service incluent la corruption de données, l'engorgement du réseau et la surcharge en ressources du serveur (mémoire, unité centrale, etc.). La surcharge des ressources est une technique très commune dans les environnements de bases de données.

8. Vulnérabilités des protocoles de communication des bases de données

Un nombre croissant de vulnérabilités de sécurité sont identifiées dans les protocoles de communication des bases de données conçus par tous les fournisseurs de bases de données. Quatre (4) des sept (7) correctifs de sécurité inclus dans les deux (2) derniers packs de correction IBM DB2 traitent des vulnérabilités des protocoles de type 1. De la même façon, 11 vulnérabilités des 23 vulnérabilités de bases de données corrigées dans le dernier correctif trimestriel d'Oracle ont un rapport avec les protocoles. Les activités frauduleuses prenant pour cible ces vulnérabilités peuvent aller de l'accès aux données non autorisées, à la corruption de données, en passant par le déni de service. Le vers informatique SQL Slammer 2, par exemple, a profité d'une faille sur le protocole du serveur Microsoft SQL pour forcer un déni de service.

9. Copies non autorisées de données sensibles

De nombreuses entreprises s'efforcent de localiser et maintenir de manière appropriée un inventaire de toutes leurs bases de données. De nouvelles bases de données peuvent être créées sans que l'équipe responsable de la sécurité soit au courant. Les données sensibles copiées dans ces bases de données peuvent être exposées si les contrôles nécessaires ne sont pas appliqués. Ces bases de données « cachées » peuvent contenir des données potentiellement sensibles telles que les détails des transactions ainsi que les coordonnées des clients et des employés. Cependant, si les personnes chargées de la sécurité des données ne connaissent pas le contenu de ces bases de données, il est difficile de s'assurer que les contrôles nécessaires ont été appliqués. Que ce soit de manière intentionnelle ou non, les employés ou les pirates informatiques peuvent alors accéder illégalement aux données sensibles.

10. Exposition de données de sauvegarde

Les dispositifs de sauvegarde de bases de données auxiliaires ne sont généralement pas protégés contre d'éventuelles attaques. Par conséquent, plusieurs violations de sécurité importantes ont

vu le jour, y compris le vol de disques durs et de bandes de sauvegarde de bases de données.

3. SOLUTIONS ADÉQUATES AFIN D'ASSURER LA SÉCURISATION DES SYSTÈMES DE GESTION DE BASES DE DONNÉES

Les architectures 3-tiers, majoritairement employées pour les services en ligne, s'articulent autour de trois (3) composants : 1/ le navigateur du client ; 2/ le serveur applicatif (serveurs web + moteur applicatif) et ; 3/ la base de données. Une grande partie de la sécurité est assurée par des moteurs d'application mais les vulnérabilités de ces derniers, couplées aux failles issues parfois du développement, peuvent entrainer des attaques de type « Injection SQL ».

Voici donc quelques conseils de première nécessité pour sécuriser votre base de données. Ceux-ci doivent néanmoins être accompagnés de mesures de sécurisation de votre système d'exploitation et de votre moteur d'application.

1. Changer le mot de passe par défaut

Lorsque nous avons terminé l'installation de notre base de données, il faut systématiquement modifier les mots de passe par défaut.

2. Refuser les connexions distantes

Pour ne pas tenter le diable, il faut également nous assurer que le service de base de données n'est pas exposé sur une adresse IP publique. Par défaut, nous devrons nous assurer qu'il est en écoute sur l'interface loopback (127.0.0.1) ou une socket système.

Astuce : si votre interface d'administration (où est installé PhpMyAdmin ou SQL Entreprise Manager par exemple) n'est pas sur le même serveur, faites communiquer les composants via des adresses IP privées sans NAT vers l'extérieur de votre réseau.

3. Supprimer les comptes inutiles

Nous allons ensuite supprimer les accès aux données qui nous paraissant inutiles. Cest assez simple, nous supprimons tous les comptes d'accès configurés après l'installation, à l'exception bien entendu du compte d'administration principal (root, sa, …).

4. Supprimer la base de données d'exemple

Suivant la règle « tout ce qui ne nous sert pas doit être supprimé », nous allons faire de même avec les bases ou tables de données livrées à l'installation pour servir d'exemples. Attention, tous les moteurs ne sont pas livrés avec ce type de données.

Il faut être particulièrement attentifs aux bases ou tables dites « système » car elles contiennent par exemple les identifiants des comptes d'accès et les droits associés. Une lecture de la documentation de votre éditeur (Oracle, Microsoft, Mysql, Postgres, …) est un pré-requis avant de trancher dans le vif.

5. Activer les logs et les externaliser

L'activation des fichiers de logs doit être un réflexe de la première heure. En cas d'interruption du service, d'incidents logiciels ou matériels ou d'intrusion sur le système, les journaux d'activités permettront de mieux analyser la situation et de prendre les mesures nécessaires.

L'externalisation des fichiers logs en temps-réel (via utilisation de syslog par exemple) est parfois nécessaire pour répondre à des obligations légales ou règlementaires. Les données externalisées doivent être stockées en lecture seule et archivées sur support non-réinscriptible si besoin.

Astuce : si votre serveur base de données ne produit pas des fichiers logs compatibles avec syslog, copiez-les à intervalles réguliers sur un serveur distant qui n'autorisera le compte de copie qu'en écriture et non en lecture et suppression.

6. Exécuter le service avec un compte de service

Les moteurs de bases de données sont généralement conçus pour n'utiliser que des droits limités sur le système d'exploitation. Les répertoires et les ports d'écoute utilisés en standard sont en effet au sein d'un contexte utilisateur et non celui du noyau.

Si nous sommes victimes d'une attaque de type « dépassement de tampon », le code susceptible d'être exécuté le sera avec des droit limités et les dégâts seront moins importants.

Astuce : pour plus de sécurité, il est recommandé, lorsque ceci est possible, d'isoler l'exécution du moteur de base de données dans un environnement clos.

7. Restreindre les privilèges des utilisateurs

Dans une démarche toujours pragmatique, il faut nous assurer que les privilèges fournis aux utilisateurs de la base de données sont en adéquation avec le besoin fonctionnel. Le compte utilisé pour se connecter à la base et y consulter des données au travers d'un script PHP, n'aura nullement besoin de droits d'écriture ou de suppression.

Vous devez donc identifier les bases et leurs tables de données auxquelles un utilisateur devra avoir accès. Dans un second temps, il est nécessaire de déterminer les droits d'accès associés : l'utilisateur doit-il pouvoir lire, créer, modifier ou supprimer des enregistrements ?

Astuce : nous pouvons agir dans le respect de deux (2) règles simples, « tout ce qui n'est pas expressément nécessaire est interdit » et « si nous ne savons pas à quoi correspond un droit d'accès, c'est que nous n'en avons pas besoin ».

8. Chiffrer les données stockées

Les moteurs les plus récents proposent tous en standard ou via un module additionnel, des options de chiffrement des données. Ne nous gênons pas pour activer l'option si nos bases de données sont constituées de données sensibles.

Les étapes de chiffrement et déchiffrement pourront alors être réalisés par la couche d'accès aux données. Ainsi, une attaque de type « Injection SQL » ne donnera accès qu'à des données chiffrées à l'attaquant.

Astuce : une solution hybride consiste à ne chiffrer les données que lors de phases particulières d'écriture (stockage de données sensibles de l'utilisateur comme son numéro de sécurité sociale). N'oublions pas non plus de sécuriser les clés de chiffrement en les stockant hors du serveur de base de données.

9. Appliquer les patchs de l'éditeur

Cela semble relever de l'évidence même, mais une fois les phases d'installation, de configuration et de recette terminées, beaucoup d'entre nous oublient que nos logiciels s'inscrivent dans un cycle de vie parfois mouvementé.

Outre les correctifs de sécurité applicables à notre système d'exploitation, nous devons veiller à la parution de correctifs pour notre moteur de bases de données.

Astuce : nous suivrons de près l'ensemble des correctifs proposés par l'éditeur de notre base de données, qu'ils s'agissent des correctifs de sécurité ou qu'ils s'agissent des correctifs en termes d'évolutions. En effet, certains éditeurs s'imposent un rythme de correctifs soutenus et intègrent les correctifs de sécurité dans les paliers évolutifs.

4. SUPERVISION DE LA SÉCURITÉ ET LES OUTILS D'ÉCOUTE DU TRAFIC RÉSEAU

Les administrateurs réseau font face à deux (2) problématiques majeures : 1/ fournir un accès rapide aux applications et aux données ; 2/ tout en protégeant les actifs numériques. Les performances et la sécurité représentent la double mission que les administrateurs doivent relever chaque jour.

La publication par Gartner en début d'année d'un premier guide du marché des outils d'analyse du trafic réseau montre à quel point la surveillance du réseau est devenue essentielle. La pression vient également des cadres dans l'entreprise.

4.1 QU'EST CE QUE LA SURVEILLANCE DU RÉSEAU EXACTEMENTS ?

La supervision réseau fait référence à la surveillance du bon fonctionnement des réseaux informatiques et des services informatiques connectés sur ces réseaux.

La surveillance du réseau porte plus spécifiquement sur :

- La qualité (bande passante) ;
- La sécurité de la connexion Internet ;
- Mais aussi, par extension, à l'état des services et matériels connectés : serveurs, imprimantes, postes de travail, etc.

La supervision réseau est l'un des 3 types de supervision informatique avec la supervision système (bas niveau) et la supervision applicative.

4.2 LES ENJEUX DE LA SURVEILLANCE RÉSEAU

Garantir un niveau de service maximal

La surveillance continue de son réseau informatique est **indispensable** dans toute entreprise à partir d'une dizaine de personnes. C'est une taille critique à partir de laquelle une panne de réseau peut entraîner une perte d'activité préjudiciable pour l'entreprise. À titre d'exemple, si votre réseau est indisponible ou difficilement accessible 1 % du temps alors c'est 87 h de travail qui sont impactées (à multiplier par le nombre d'employés). Il est donc très important de pouvoir détecter les risques d'incidents ou les optimisations à réaliser avant de faire face à un problème technique grave.

Le monitoring de l'activité du réseau permet par ailleurs d'être correctement informé en cas de problème et de mettre en place le bon plan d'action rapidement.

Protéger les systèmes d'informations

L'autre volet de la supervision réseau concerne la sécurité de celui-ci. En effet, la cybercriminalité, les virus, les phishings et autres intrusions causent des dommages qui s'élèvent à quasiment 80 millions d'euros rien qu'en France (chiffres 2017).

Pour lutter efficacement contre ces attaques en constante augmentation (particulièrement sur les réseaux mobiles), la DSI doit se munir de technologies lui permettant de connaître ses forces, ses faiblesses mais aussi de pouvoir réagir vite lors d'une tentative.

Le rôle de la supervision est donc de garantir un niveau de qualité continu souvent nommé SLA pour Service Level Agreement pour prévenir les risques de rupture de services/d'intrusion et réagir vite aux incidents.

4.3 COMMENT ÇA MARCHE ?

La supervision réseau peut être mise en œuvre sur la base d'analyse de logs, de résultats de commandes et de scripts locaux mais c'est surtout sur la base de protocoles standards comme le protocole **snmp** que le monitoring des réseaux informatiques fonctionne.

De nombreux logiciels existent pour ce faire. La communauté du logiciel libre (open source) est particulièrement active dans le domaine, raison pour laquelle on trouve beaucoup de logiciels open source disponibles comme ZENOSS ou CACTI.

Les logiciels permettent d'assister le technicien grâce aux alertes SMS, email mais aussi en proposant des solutions concrètes pour résoudre ou anticiper un problème. Ils offrent une couverture fonctionnelle plus ou moins large mais s'adaptent, la plupart du temps, à tout type d'entreprise.

Les fonctionnalités peuvent aller de la simple analyse de log, la surveillance des routeurs et des firewalls à une gestion plus poussée du système d'information en temps réel, de l'administration réseau, des bases de données et des serveurs.

4.4 QUELQUES LOGICIELS DE SUPERVISION RÉSEAU

4.4.1 CACTI

Cacti permet de collecter des données de quasiment n'importe quel élément du réseau, notamment des systèmes de routage et de commutation ainsi que des pare-feux, puis de placer ces données dans des graphiques rigoureux.

4.4.2 SPICEWORKS

Spiceworks offre de nombreux outils de gestion informatique gratuits, notamment pour la gestion de l'inventaire, le flux de travail du support technique, voire la surveillance Cloud en plus de la solution de surveillance de réseau qui nous intéresse ici. Reposant sur des techniques sans agent telles que WMI (pour les machines Windows) et SNMP (pour les systèmes réseau et *nix), cet outil gratuit peut fournir des informations sur de nombreux problèmes de performances réseau. Vous pouvez également configurer des notifications personnalisables et redémarrer des services depuis l'application même.

4.5 OUTILS D'ACOUTE ET D'ANALYSE DU TRAFIC RÉSEAU

Il existe d'autres outils qui ne sont pas des solutions de surveillance proprement dites mais sont incroyablement utiles pour les professionnels de la surveillance.

4.5.1 WIRESHARK

Wireshark® est un outil d'analyse de paquets open source qui utilise libpcap (*nix) ou winpcap (Windows) pour capturer les paquets et les afficher dans son interface graphique. Il fournit de bonnes fonctions de filtrage, de regroupement et d'analyse. Il permet aux utilisateurs de capturer le trafic en temps réel ou de lire des données de lots de paquets et d'analyser les détails microscopiques. Wireshark prend en charge la plupart des protocoles et offre des fonctionnalités de filtre qui s'appuient sur le type de paquet, la source, la destination, etc. Il peut entre autres analyser les appels VoIP, dessiner des graphiques d'E/S pour l'intégralité du trafic d'une interface, décrypter de nombreux protocoles ou encore exporter les résultats.

Wireshark offre des opportunités illimitées d'analyse de paquets, ce qui en fait une option solide pour les administrateurs réseau, système et de sécurité.

4.5.2 NMAP

Nmap utilise une fonctionnalité de détection des hôtes sur le réseau qui permet de créer une carte du réseau. Les administrateurs réseau l'apprécient car il permet de collecter des informations des hôtes relatives au système d'exploitation, aux services ou aux ports exécutés ou ouverts, des informations d'adresse MAC, des noms DNS inversés, etc.

L'évolutivité est la principale raison pour laquelle les administrateurs réseau aiment Nmap. Il peut analyser un seul hôte ou un réseau entier constitué de « centaines de milliers » de machines. Lorsque vous avez besoin de mapper rapidement les hôtes de votre réseau, optez pour Nmap.

5. LES PROTOCOLES DE CHIFFREMENT

5.1 QU'EST-CE QUE LE CHIFFREMENT

Le chiffrement est un terme technique qui désigne **la méthode par laquelle les communications** (SMS, courriels, appels téléphoniques et vidéo) **sont sécurisées** afin d'empêcher toute personne autre que le destinataire prévu d'y accéder. Le chiffrement est la manipulation mathématique des informations dans le but de les rendre lisibles uniquement par le ou les destinataires prévus.

Le chiffrement est apparu bien avant Internet. Nous utilisons quotidiennement l'une des trois (3) méthodes de chiffrement dès que nous utilisons des services connectés :

- **Le chiffrement complet des données d'un disque dur ou d'un appareil.** Il s'agit d'un procédé par lequel toutes les données stockées sur un ordinateur ou un smartphone sont chiffrées lorsqu'elles se trouvent sur l'appareil. Si vous utilisez certains smartphones ou ordinateurs récents fonctionnant sous la dernière version disponible et mise à jour du système d'exploitation, le chiffrement des données de l'appareil sera activé par défaut, ce qui signifie que vous utilisez peut-être cette technologie sans même le savoir. Sur d'autres appareils, il peut être nécessaire de suivre une procédure spécifique pour activer le chiffrement des données de l'appareil. Avec le chiffrement des données d'un appareil, les données stockées sur l'appareil ne seront généralement pas lisibles par une personne qui ne possède pas votre code d'identification personnel (PIN) ou votre mot de passe.
- **Le chiffrement de bout en bout** garantit que les communications transmises entre l'expéditeur et le destinataire ne peuvent pas être déchiffrées ou lues par une tierce personne ou par un fournisseur de services. Lorsque le chiffrement de bout en bout est utilisé, tout appareil ou fournisseur de service intermédiaire ayant accès aux communications électroniques ou toute personne voulant intercepter ces communications est incapable de lire leur contenu.
- **Le chiffrement du transport ou chiffrement de la couche transport (dont l'implémentation la plus courante est le HTTPS avec le protocole TLS ou SSL)** est le moyen par lequel les communications entre les sites Internet que vous consultez (par exemple le moteur de recherche Google, les boutiques en ligne telles qu'Amazon.com, vos services de banque en ligne ou encore les messageries comme Outlook) et votre navigateur sont chiffrées.

5.2 QUELQUES TERMES UTILISÉS EN CRYPTOGRAPHIE

- Le **chiffrement** est la transformation d'une information en clair en une information chiffrée, incompréhensible, mais que l'on peut déchiffrer avec une clé pour obtenir l'information en clair originale.
- Un **système de chiffrement** (ou *cryptosystème*, ou encore *chiffre*) est composé d'algorithmes de chiffrement et déchiffrement et d'une clé de chiffrement.

- Un **algorithme de chiffrement** est une fonction qui prend en entrée le texte clair et la clé de chiffrement, transforme le texte par des opérations et fournit en sortie un texte chiffré.
- **L'algorithme de déchiffrement** est la fonction inverse qui prend en entrée le texte chiffré et la clé de déchiffrement, puis transforme ce texte par des opérations et fournit en sortie le texte clair d'origine.
- La **clé de chiffrement** (ou cryptovariable) est l'information qui permet de transformer un texte clair en texte chiffré en utilisant un algorithme de chiffrement. De même, la clé de déchiffrement est l'information qui permet de transformer un texte chiffré en son texte clair d'origine. **L'espace de clé** est l'ensemble des valeurs possibles de la clé, c'est une notion importante pour la sécurité d'un algorithme. Si la clé de chiffrement et la clé de déchiffrement sont identiques, on parle de clé secrète et de chiffrement symétrique.
- **Le terme crypter n'existe pas**. En langage informatique le terme crypter n'est pas utilisé car il vient de l'anglicisme. On utilise le terme **chiffrer**.
- Le **mot décrypter existe.** Il est à utilisé dans l'opération de déchiffrement lorsque l'on ne possède **PAS** la clé de déchiffrement. Alors que le terme **déchiffrer** est le mécanisme de déchiffrement en utilisant la clé de déchiffrement.

5.3 TYPES DE CHIFFREMENTS

Trois (3) types de chiffrements sont possibles :

1. **Le chiffrement symétrique** (méthode la plus ancienne). Symétrique par le fait d'utiliser une clé identique avec le même algorithme de chiffrement pour le chiffrement et le déchiffrement.
2. **Le chiffrement asymétrique.** Asymétrique par le fait d'utiliser une clé pour chiffrer (clé publique) et une autre clé pour déchiffrer (clé privée) avec le même algorithme de chiffrement.
3. **Le chiffrement hybride.** Cette méthode utilise un combiné des deux (2) modes de chiffrement précédents. L'idée est de chiffrer de manière symétrique le message puis de chiffrer de manière asymétrique la clé précédemment utilisée.

5.4 ALGORITHMES DE CHIFFREMENT ET DE HACHAGE

Les algorithmes symétriques les plus connus sont :

- Le chiffre de césar (Méthode par décalage) ;
- DES et (Triple DES) ;
- AES.

Les algorithmes asymétriques les plus connus sont :

- RSA ;
- El-Gamal ;
- Courbes elliptiques.

Les algorithmes hybrides les plus connus sont :

- PGP :
- GnuPG ;
- TLS.

Concernant les algorithmes de hachage, les plus connus sont :

- MD5 ;
- SHA-1 ;
- SHA-256.

5.5 FORCES ET FAIBLESSES

Algorithme	Forces	Faiblesses
Symétrique	Facilité d'intégration Plus performant	Moins sécurisé (Par le fait que la clé secrète est facilement transmissible)
Asymétrique	Clé privée connue que d'un seul acteur Moins performant (Couteux en ressource, temps de calcul plus élevé) (RSA demande une clé minimale de 1024/2048 bits) (Les courbes elliptiques ne nécessitent qu'une clé de 128 bits) Plus sécurisé	Complexité à gérer (Utilisation d'une PKI)
Hybride	Plus performant Plus sécurisé	Échange de deux informations (clé symétrique chiffré et message chiffré)

5.6 HTTPS, SSL ET TLS

La barre d'adresse d'un navigateur web fournit des informations sur le niveau de sécurité des sites visités. On connaît notamment le petit cadenas qui s'affiche à côté de l'URL, signe que le détenteur du site a adopté le protocole de sécurité HTTPS.

Cette sécurisation des serveurs (et donc des sites web) passe par des algorithmes de chiffrement. Ceux-ci consistent en la génération d'une clé cryptographique qui permet :

- D'assurer la confidentialité des données échangées entre un poste client et un serveur. Dès qu'il est activé, seules ces deux entités peuvent décrypter les informations qui circulent entre elles ;
- De garantir l'intégrité des données ;
- D'authentifier le serveur web avec lequel l'utilisateur communique. Car une simple clé de chiffrement ne garantit aucunement l'identité de son détenteur !

Pour bénéficier de cette protection, un site web utilise un certificat de chiffrement – un certificat SSL ou TLS – relié au protocole HTTPS. Il est délivré par une **Autorité de Certification (AC)**, chacune proposant des niveaux différents de fiabilité.

Le protocole HTTP (pour HyperText Transfer Protocol) permet le transfert des données sur le web. C'est, en quelque sorte, le mode d'emploi d'une requête envoyée par le navigateur au serveur, préalable indispensable aux échanges d'informations. Le hic, c'est que ce type de connexion n'offre aucune sécurité. N'importe qui peut intercepter les données.

C'est là qu'intervient le HTTPS (avec le « S » de Secure en plus), qui adjoint au HTTP classique un protocole SSL / TLS. Celui-ci assure le chiffrement des données grâce à une clé de cryptage asymétrique, rendant les informations échangées illisibles pour une personne tierce et sécurisant la connexion. Il prouve également l'identité du détenteur du certificat SSL / TLS correspondant.

L'activation du protocole HTTPS fait apparaître un cadenas à côté de l'URL dans la barre d'adresse. Lorsqu'un site web est protégé par un certificat SSL ou un certificat TLS ; il affiche un cadenas à côté de l'URL.

SSL signifie Secure Sockets Layer. En bref, il s'agit d'une technologie standard destinée à la sécurité de la connexion Internet et à la protection des données sensibles qui sont transmises entre deux systèmes, empêchant les criminels de lire et de modifier les informations transférées, y compris d'éventuelles informations personnelles. Les deux systèmes peuvent être un serveur et un client (par exemple, un site d'achat et un navigateur) ou un serveur et un autre serveur (par exemple, une application avec des informations personnelles identifiables ou des informations de paie).

TLS (Transport Layer Security) est simplement une nouvelle version, plus sûre, de SSL.

Ces différents termes et sigles (certificats, SSL, TLS, HTTPS) englobent une seule et même chose, à savoir des protocoles de sécurisation des échanges entre les internautes et les sites web.

DOCUMENTS UTILES

- Chantal Gribaumont, Administrez vos bases de données avec MySQL, édition du 12 mai 2020
- MySQL 5.7 Reference Manual
- Eyrolles Initiation a sql cours exercices corriges
- Maurice Libes , Administration et exploitation du SGBDR MySQL, octobre 2004
- CyberEdu, Sensibilisation et initiation à la cybersécurité, 05 novembre 2015
- https://dev.mysql.com/doc/
- https://mysql.developpez.com/
- https://fadace.developpez.com/

FICHIERS DE FORMATION

Les documents pour les exercices sont disponibles en cliquant sous les liens tout commes les présentations PowerPoint.

Exercices	liens
Exercice 1	creParc.sql
Exercice 2	creParc.sql
Exercice 3	descParc.sql
Exercice 4	dropParc.sql
Exercice 5	insParc.sql
Exercice 6	insParc.sql
Exercice 7	modification.sql
Exercice 8	evolution.sql
Exercice 9	evolution.sql
Exercice 10	evolution.sql
Exercice 11	creaDynamique.sql
Exercice 12	requetes.sql
Exercice 13	requetes.sql
Exercice 14	requetes.sql
Exercice 15	modifSynchronisees.sql
Exercice 16	requetes.sql
Exercice 17	vues.sql

Présentations	liens
Module 1	Module 1 .pptx
Module 2	Module 2 .pptx
Module 3	Module 3 .pptx

ANNEXES

1. BIBLIOGRAPHIE ET WEBOGRAPHIE

- Database Journal (www.databasejournal.com)
- F. BROUARD, C. SOUTOU, SQL, Pearson Education, 2005.
- E. DASPET, C.P. DE GEYER, PHP 5 avancé, Eyrolles, 2005.
- P. DELMAL, SQL2-SQL3, Applications à Oracle, De Boeck Université, 2000.
- M. KOFLER, MySQL 5, Eyrolles, 2005.
- C. SOUTOU, De UML à SQL, Eyrolles, 2002.
- MySQL Administrator, MySQL Query Browser : http://www.mysql.com/products/tools/
- phpMyAdmin : http://www.phpmyadmin.net/home_page/index.php
- Navicat : http://www.navicat.com/
- TOAD : http://www.toadsoft.com/toadmysql/toad_mysql.htm
- EMS SQL Manager : http://www.sqlmanager.net/
- Centre Canadien pour la Cybersécurité, Guide sur le chiffrement des services infonuagiques, ITSP.50.106 Mai 2020 : série praticiens
- Stéphane Jacob, Protection cryptographique des bases de données : conception et cryptanalyse, HAL Id: tel-00738272
- Luc Bouganim, Sécurisation du Contrôle d'Accès dans les Bases de Données, 31 Jul 2008
- Guillaume HARRY l Contenu sous licence Creative Commons CC-BY-NC-ND, Principales failles de sécurité des applications Web : Principes, parades et bonnes pratiques
- Blue Coat Systems Inc., Trafic SSL/TLS, février 2020
- Alexandre Viardin, Guide pratique pour la sécurisation et supervision des réseaux , Eyrolles, 2015
- Pierre-Yves Saumont, Stratégies anti-hackers, Écoutes sur un réseau informatique, avril 2009.

182, rue de la Sirba - BP 13416 Niamey - NIGER • Tél. : (227) 20 72 35 60 • Fax : (227) 20 72 21 74 • http://www.ins.ne •
http://www.pnin.org • E-mail : ins@ins.ne • E-mail : pnin@ins.ne