Analyse et Conception de Base de données

I. Notion d'analyse et de conception :

Une **méthode d'analyse et de conception** a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client. Pour ce faire, on part d'un énoncé informel (le besoin tel qu'il est exprimé par le client, complété par des recherches d'informations auprès des experts du domaine fonctionnel, comme par exemple les futurs utilisateurs d'un logiciel), ainsi que de l'analyse de l'existant éventuel (c'est-à-dire la manière dont les processus à traiter par le système se déroulent actuellement chez le client).

La phase d'analyse permet de lister les résultats attendus, en termes de fonctionnalités, de performance, de robustesse, de maintenance, de sécurité, d'extensibilité, etc.

La phase de conception permet de décrire de manière non ambiguë, le plus souvent en utilisant un langage de modélisation, le fonctionnement futur du système, afin d'en faciliter la réalisation.

A. Merise :

La méthode merise ou méthode d'étude et de réalisation informatique du système d'entreprise permet d'analyser (modéliser) un système information d'entreprise (SIE). Il en France vers les années 70. Elle propose l'analyse séparée entre les données et les traitements. Cette analyse se fait è 3 niveaux différents : **conceptuel, logique ou organisationnel et physique ou opérationnel.**

Analyser un système d'information avec merise consiste à proposer un certain nombre de modèles pour les données et les traitements dans différents niveaux.

Les différents modèles utilisés pour analyser un système d'information en merise sont :

- Le modèle conceptuel de communication (MCC)
- Le modèle conceptuel des données(MCD)
- Le modèle conceptuel des traitements(MCT)
- Le modèle logique des données(MLD)
- Le modèle physique des données(MPD)
- Le modèle opérationnel des traitements (M0pT)
- Le modèle organisationnel des traitements (M0T)

Ces différents modèles sont répartis dans deux systèmes différents à savoir :

- Le système d'information informatisé (SII) : qui gère l'aspect technique du système.
- Le système d'information organisationnel (SIO) : qui gère les procédures.

B. UML (Langage de modélisation unifié) :

Le langage UML (Unified Modeling Language, ou langage de modélisation unifié) a été pensé pour être un langage de modélisation visuelle commun, et riche **sémantiquement et syntaxiquement**. Il est destiné à l'architecture, la conception et la mise en œuvre de systèmes logiciels complexes par leur structure aussi bien que leur comportement. L'UML a des applications qui vont au-delà du développement logiciel, notamment pour les flux de processus dans l'industrie. Il ressemble aux plans utilisés dans d'autres domaines et se compose de différents types de diagrammes. Dans l'ensemble, les diagrammes UML décrivent la limite, la structure et le comportement du système et des objets qui s'y trouvent.
L'UML n'est pas un langage de programmation, mais il existe des outils qui peuvent être utilisés pour générer du code en plusieurs langages à partir de diagrammes UML. L'UML a une relation directe avec l'analyse et la conception orientées objet.

- **UML et son rôle dans la modélisation et la conception orientées objet**
  Il existe de nombreux modèles et paradigmes de résolution de problèmes en science informatique, qui est l'étude des algorithmes et des données. On dénombre quatre catégories de modèles de résolution de problèmes : les langages **impératifs, fonctionnels, déclaratifs et orientés objet (LOO)**. Dans le cas des langages orientés objet, les algorithmes sont exprimés en définissant des objets et en les faisant interagir les uns avec les autres. Ces objets sont des éléments que l'on peut manipuler et qui existent dans le monde réel. Il peut s'agir d'immeubles, de widgets sur un ordinateur ou encore d'êtres humains. Les langages orientés objet dominent le monde de la programmation parce qu'ils modélisent des objets du monde réel. L'UML combine plusieurs notations orientées objet : Object-Oriented Design (conception orientée objet), Object Modeling Technique (technique de modélisation objet) et Object-Oriented Software Engineering (génie logiciel orienté objet). L'UML utilise les points forts de ces trois approches pour présenter une méthodologie plus cohérente et plus facile à utiliser. Il représente les meilleures pratiques de création et de documentation des différents aspects de la modélisation des systèmes logiciels et d'entreprise.

- **Histoire et origines de l'UML**

  Les Three Amigos du génie logiciel, comme on les appelait alors, avaient élaboré d'autres méthodologies. Ils se sont associés pour apporter plus de clarté aux programmeurs en créant de nouvelles normes. La collaboration entre Grady, Booch et Rumbaugh a renforcé les trois méthodes et a amélioré le produit final.

  Les efforts de ces penseurs ont abouti à la publication des documents UML 0.9 et 0.91 en 1996. Il est rapidement devenu évident que des sociétés comme Microsoft, Oracle et IBM voyaient l'UML comme un

élément critique pour leur développement futur. Elles ont donc mis en place des ressources, accompagnées en cela par de nombreuses autres sociétés et personnes, permettant de développer un langage de modélisation complet. Les Three Amigos ont publié The Unified Modeling Language User Guide en 1999, qui fut suivi d'une mise à jour comportant des informations sur l'UML 2.0 en 2005.

- **Utilité de l'UML selon l'OMG**

  L'OMG définit les objectifs de l'UML comme suit :

  Fournir aux concepteurs de systèmes, ingénieurs logiciels et développeurs de logiciels des outils pour l'analyse, la conception et la mise en œuvre de systèmes logiciels, ainsi que pour la modélisation de processus métier et d'autres processus similaires.

  Faire progresser l'industrie en permettant l'interopérabilité des outils de modélisation visuelle orientés objet. Toutefois, pour permettre un échange significatif d'informations de modèles entre outils, il est nécessaire de trouver un accord sur la sémantique et la notation.

  UML répond aux exigences suivantes :

  Fixer une définition formelle d'un métamodèle basé sur une norme Meta-Object Facility (MOF) commune qui spécifie la syntaxe abstraite de l'UML. La syntaxe abstraite définit l'ensemble des concepts de modélisation UML, leurs attributs et leurs relations, ainsi que les règles permettant d'associer ces concepts afin de créer des modèles UML partiels ou complets.

  Fournir une explication détaillée de la sémantique de chaque concept de modélisation UML. La sémantique définit, d'une façon indépendante de la technologie, comment les concepts UML doivent être mis en œuvre par les ordinateurs.

  Spécifier des éléments de notation lisibles par l'homme pour représenter chaque concept de modélisation UML, ainsi que les règles pour les combiner au sein d'une grande variété de diagrammes correspondant à différents aspects des systèmes modélisés.

  Définir des moyens grâce auxquels les outils UML peuvent être mis en conformité avec cette spécification. Ceci est pris en charge (dans une spécification distincte) par une spécification XML des formats d'échange de modèles correspondants (XMI) qui doivent être réalisés par des outils conformes.

- **UML et la modélisation de données**

  L'UML est prisé par les programmeurs, mais n'est généralement pas utilisé par les développeurs de bases de données. Cela s'explique notamment par le fait que les créateurs de l'UML ne se sont tout simplement pas intéressés aux bases de données. Malgré cela, l'UML est efficace pour la modélisation générale de données et peut être utilisé dans différents types de diagrammes UML. Vous trouverez des informations sur l'application d'un modèle de classe orienté objet sur une base de données relationnelle dans cet article sur **la modélisation de bases de données en UML.**

C. Etude Comparative :

MERISE (Méthode d'Etude et de Réalisation Informatique pour les Systèmes d'Entreprise) est une méthode d'analyse et de réalisation des systèmes d'information qui est élaborée en plusieurs étapes: schéma directeur, étude préalable, étude détaillée et la réalisation.
Alors que UML (Unified Modeling Langage), est un langage de modélisation des systèmes standard, qui utilise des diagrammes pour représenter chaque aspect d'un système : statique, dynamique, en s'appuyant sur la notion d'orienté objet qui est un véritable atout pour ce langage.

. Niveaux d'abstraction

L'approche Merise : Le cycle d'abstraction permet de sérier les niveaux de Préoccupations lors de la description ou de l'analyse du système. Les trois niveaux retenus correspondent à des degrés de stabilité et d'invariance de moins en moins élevés

–Le niveau conceptuel,

–le niveau logique,

–le niveau physique.

L'approche UML propose différentes notions (cas d'utilisation, paquetage, classe, composant, nœud) et différents diagrammes pour modéliser le système aux différents niveaux d'abstraction.

. Approche fonctionnelle

L'approche Merise propose une approche descendante où le système réel est décomposé en activités, elles-mêmes déclinées en fonctions. Les fonctions sont composées de règles de gestion, elles-mêmes regroupées en opérations. Ces règles de gestion au niveau conceptuel génèrent des modules décomposés en

modules plus simples et ainsi de suite jusqu'à obtenir des modules élémentaires... Les limites d'une telle approche résident dans le fait que les modules sont difficilement extensibles et exploitables pour de nouveaux systèmes.

L'approche UML: Les fonctions cèdent la place aux cas d'utilisation qui permettent de situer les besoins de l'utilisateur dans le contexte réel. A chaque scénario correspond des diagrammes d'interaction entre les objets du système et non pas un diagramme de fonction...

. Dualité des données -traitements

L'approche Merise propose de considérer le système réel selon deux points de vue: un point de vue statique (les données), un point de vue dynamique (les traitements). Il s'agit d'avoir une
vision duale du système réel pour bénéficier de l'impression de relief qui en résulte, et donc consolider et valider le système final.
L'approche UML : L'approche objet associe les informations et les traitements. De cette façon, elle assure un certain niveau de cohérence.

Tableau 1 : tableau comparatif UML - Merise

Merise	UML
méthode d'analyse et de conception de système d'information	langage de représentation d'un système d'information.
méthode de modélisation de données et traitements orienté bases de données relationnelles.	système de notation orienté objet.
relationnel	objet.
Franco-français	International
schéma directeur, étude préalable, étude détaillée et la réalisation.	langage de modélisation des systèmes standard, qui utilise des diagrammes pour représenter chaque aspect d'un systèmes ie: statique, dynamique,....en s'appuyant sur la notion d'orienté objet
plus adapté à une approche théorique	plus orientée vers la conception
du "bottom up" de la base de donnée vers le code	du "top down" du modèle vers la base de donnée.

II. Concepts et Analyse de concepts :

Dans la méthodologie Merise destinée à créer des bases de données, il y a des outils dédiés aux traitements et aux données. Le **MCD** et le **MLD** sont des outils majeurs concernant les données.

A. Le MCD :

Le **MCD** est une représentation graphique de haut niveau qui permet facilement et simplement de comprendre comment les différents éléments sont liés entre eux à l'aide de diagrammes codifiés dont les éléments suivants font partie :

- Les entités (1 rectangle = 1 objet) ;

- Les propriétés (la liste des données de l'entité) ;
- Les relations qui expliquent et précisent comment les entités sont reliées entre elles (les ovales avec leurs « pattes » qui se rattachent aux entités) ;
- Les cardinalités (les petits chiffres au-dessus des « pattes »).

Utilisé assez tôt en conception de base de données, le **MCD** évoluera ensuite vers les autres outils de Merise, à savoir le MPD et le MLD.

Cet outil **permet d'échanger entre informaticiens et non-informaticiens** sur l'outil à informatiser. On peut ainsi à partir d'un MCD **valider et préciser des règles** qui s'appliqueront à la future base de données.

Avant de concevoir le **MCD**, il est conseillé de dresser un dictionnaire de des données.

1. **Dictionnaire des données**

   C'est un tableau contenant la liste des données. Pour chaque donnée on s'intéresse à son code, son libellé, son type et sa taille. Cependant il faut éviter de mettre dans le dictionnaire les données calculées, les synonymes ou les redondances.

   Exemple :

Client

Code	Libellé	Type	Taille
Pcl	Prénom Client	Texte	100
Ncl	Nom Client	Texte	40
Tcl	Téléphone	Numérique	20
Icl	Id Client	Numérique	20

2. **Les concepts de base du MCD**

   - **Entités :**

     Une entité correspond à un **ensemble homogène d'informations** qui correspondent au même « objet » à informatiser. Cette **entité** a un nom unique afin de la manipuler facilement. Plus tard dans l'analyse, l'entité se transforme en table et devient concrètement une table lors de la réalisation effective de la base de données. Cet ensemble d'informations, **l'entité**, partage les mêmes caractéristiques et peut être manipulé au sein du système d'information mais aussi en discutant entre informaticiens et personnes du métier. L'entité « *contrat* » et l'entité « *client* » sont typiques des bases de données de gestion. Isoler, comprendre le rôle et les

informations qui se trouvent dans ces coquilles est une des premières étapes de l'analyse. Une **entité** reste cependant peu utile si elle n'est pas remplie avec des informations calibrées et si elle ne discute pas avec les autres entités de la base de données : on parle alors de propriétés et d'associations ce qui a donné le nom de modèle entités-associations que l'on croise parfois.

Une entité est tout objet indentifiable et pertinent à l'application. Une entité a une existence propre. Une entité est couramment représentée par un rectangle formé de 2 parties : la partie du haut contient le nom de l'entité, la partie du bas contient les attributs. Un ou plusieurs **attributs** doivent former un identifiant permettant de distinguer chaque **occurrence** d'entité ou élément de l'ensemble.

- **Attributs :**

  Ce sont les caractéristiques de l'Entité. Un attribut est une donnée atomique associée à un type de donnée. Un attribut porte un nom et un type de donnée. Le nom et le type de donnée sont séparés par le signe **":"**.

  Exemple : PrenomClient, NomClient, Telephone, IdClient…

- **Occurrences :**

  Une occurrence d'entité, dans le contexte d'un modèle conceptuel de données, est un élément de l'ensemble représenté par l'entité.

- **Identifiants :**

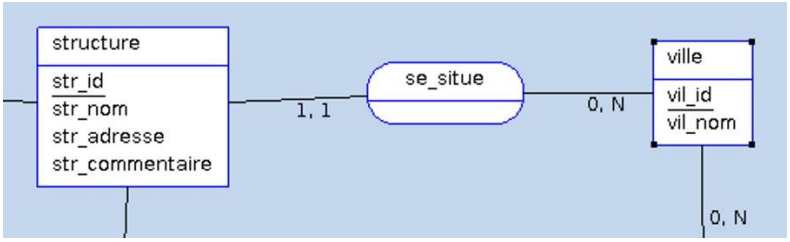
  Avec la méthodologie Merise, on utilise les deux termes « **identifiant** » et « **clé primaire** ». Il s'agit de **la même chose** mais pas utilisé dans le même outil. On utilise le terme « **identifiant** » à l'étape du MCD. On utilise le terme « clé primaire » aux étapes du MPD, MLD et lorsque la base est créée .Dans chaque entité, il existe des propriétés. La première des propriétés, celle **dont la valeur est unique** est **l'identifiant**. C'est grâce à l'identifiant que l'on peut identifier de manière unique les données de chaque propriété de l'entité sans se tromper. On la représente soulignée. Il n'y a qu'un seul identifiant par entité.

  Dans la modélisation Merise, les entités peuvent être reliées entre elles. Les relations et les dépendances fonctionnelles reposent sur les identifiants : ce sont eux qui permettent de relier les bonnes valeurs entre elles

- **Associations** :

Les bases de données sont constituées de différentes données homogènes qui discutent entre elles (les entités). Lorsque l'on représente les liaisons et les interactions entre ces différentes données cohérentes, on parle d'**association**. En conception de base de données, on dit qu'**une association précise et explicite le lien entre plusieurs entités**. Une association se fait entre 2 entités minimum mais il n'y a pas de maximum.

Exemple :



- **Cardinalités :**

Lorsque l'on conçoit une base de données avec le MCD de Merise, on obtient un schéma avec des entités et des associations. Pour préciser au mieux les associations, on utilise des cardinalités. Les cardinalités sont des caractères (0,1, n) qui fonctionnent par couple et qui sont présents de chaque côté d'une association (sur chaque « patte »). Ils donnent des indications très intéressantes et permettent par la suite de construire la base de données :

- ✓ avec la création de clés étrangères dans le cas d'une CIF
- ✓ avec la création d'une table intermédiaire dans le cas d'une CIM

Les cardinalités possibles sont :

- ✓ 0,1 : au minimum 0, au maximum 1 seule valeur (CIF) ;
- ✓ 1,1 : au minimum 1, au maximum 1 seule valeur (CIF) ;
- ✓ 0, n : au minimum 0, au maximum plusieurs valeurs ;
- ✓ 1, n : au minimum 1, au maximum plusieurs valeurs.

Les cardinalités maximales (le côté droit ou le dernier caractère) définissent si la relation est une CIM ou une CIF.

Dans l'image, on a « 1,1 » en cardinalité du côté gauche de l'association et une cardinalité « 0, n » du côté droit. Concrètement, il faut lire les cardinalités ainsi :

- ✓ Une structure se situe dans 1 et 1 seule ville ;
- ✓ Dans une ville peut se situer 0 ou plusieurs (n) structures.

Les cardinalités sont importantes car elles traduisent les règles de gestion, elles doivent être validées avec l'utilisateur final.

- **Formalisme :**

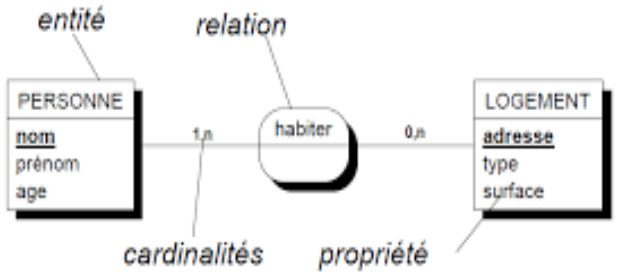


Figure 7.2 : Les concepts du formalisme entité - relation

B. Le MLD

Le modèle logique de données permet de faire une représentation du niveau logique des données. Il permet de modéliser la structure selon laquelle les données seront stockées dans la base de données. Il est adapté à la famille de système de gestion de base de données(**SGBD**) dit **SGBD** relationnel. Il permet donc d'implémenter la base de données dans un **SGBD** donné. Il est obtenu à partir du **MCD** par application des règles dites règles de passages. Ces règles utilisent en général les cardinalités pour désigner les relations ainsi que les migrations.

- **MLD-R :**

  Historiquement, le terme relationnel s'appliquait à la notion de structure tabulaire ; il mettait en évidence les relations existantes entre les colonnes d'une table. Afin d'éviter toute erreur d'interprétation, dans la suite de ce cours, le terme de table sera utilisé en lieu et place de relation et le terme relation sera réservé aux "liens" entre tables. Un modèle logique de données relationnel (MLD-R) est la représentation des données d'un système d'information réalisé en vue d'une mise en œuvre au sein d'un système de gestion de base de données relationnel (SGBD-R). Les données sont représentées sous forme de tables et de relations entre tables.

- **Modèle Relationnel :**

  Le modèle relationnel est basé sur une organisation des données sous forme de tables. La manipulation des données se fait selon le concept mathématique de relation de la théorie des ensembles, c'est-à-dire l'algèbre relationnelle. L'algèbre relationnelle a été inventée en 1970 par E.F. Codd, le directeur de recherche du centre IBM de San José. Elle est constituée d'un ensemble d'opérations formelles sur les relations. Les

opérations relationnelles permettent de créer une nouvelle relation (table) à partir d'opérations élémentaires sur d'autres tables (par exemple l'union, l'intersection, ou encore la différence).

- **Règles de passages** :

Règle numéro 1 :

a) Une entité du MCD devient une relation, c'est à dire une table. Dans un SGBD (Système de Gestion de base de données) de type relationnel, une table est une structure tabulaire dont chaque ligne correspond aux données d'un objet enregistré (d'où le terme enregistrement) et où chaque colonne correspond à une propriété de cet objet. Une table contiendra donc un ensemble d'enregistrements. Une ligne correspond à un enregistrement. Une colonne correspond à un champ. La valeur prise par un champ pour un enregistrement donné est située à l'intersection ligne-colonne correspondant à enregistrement-champ. Il n'y a pas de limite théorique au nombre d'enregistrements que peut contenir une table. Par contre, la limite est liée à l'espace de stockage.

b) Son identifiant devient la clé primaire de la relation. La clé primaire permet d'identifier de façon unique un enregistrement dans la table. Les valeurs de la clé primaire sont donc uniques. Les valeurs de la clé primaire sont obligatoirement non nulles. Dans la plupart des SGBDR (Système de Gestion de Base de Données Relationnelle), le fait de définir une clé primaire donne lieu automatiquement à la création d'un index. Un index est un fichier interne au SGBD. L'utilisateur standard n'a pas besoin d'y accéder. L'index a pour but d'accélérer les traitements de recherche, de tri, de filtre et notamment sur les tables avec de nombreux enregistrements. La contrepartie est que l'index nécessite de l'espace mémoire et surtout, les temps d'insertion, de suppression d'enregistrements sont plus importants car il faut mettre à jour à la fois la table et l'index. c) Les autres propriétés deviennent les attributs de la relation.

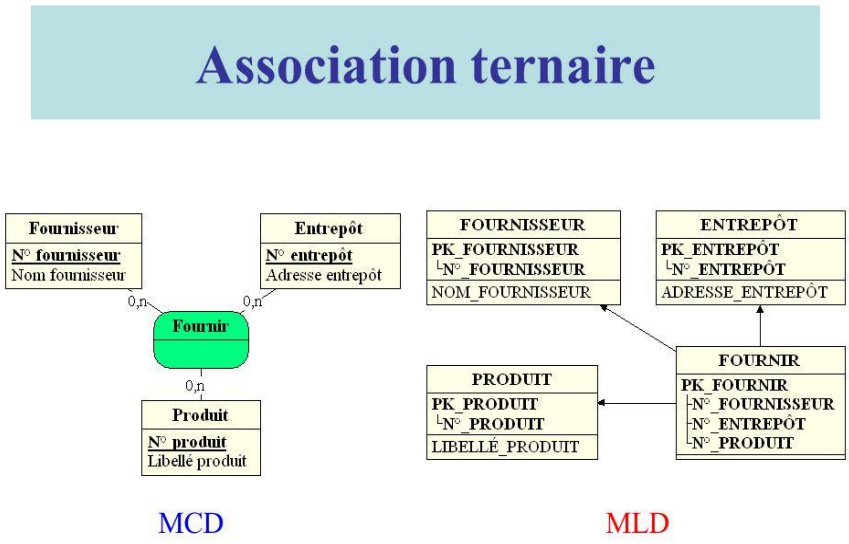
Règle numéro 2 :

Une association de type 1:N (c'est à dire qui a les cardinalités maximales positionnées à « 1 » d'une côté de l'association et à « n » de l'autre côté) se traduit par la création d'une clé étrangère dans la relation correspondante à l'entité côté « 1 ». Cette clé étrangère référence la clé primaire de la relation correspondant à l'autre entité.

Règle numéro 3 :

Une association de type N : N (c'est à dire qui a les cardinalités maximales positionnées à « N » des 2 côtés de l'association) se traduit par la création d'une table dont la clé primaire est composée des clés étrangères référençant les relations correspondant aux entités liées par l'association. Les éventuelles propriétés de l'association deviennent des attributs de la relation

- **Formalisme :**



III. SQL :

SQL est un langage déclaratif qui permet d'interroger une base de données sans se soucier de la représentation interne (physique) des données, de leur localisation, des chemins d'accès ou des algorithmes nécessaires.

Langage de définition des données (LDD) :

est un langage de programmation et un sous-ensemble de SQL pour manipuler les structures de données d'une base de données, et non les données elles-mêmes.

Il permet de définir le domaine des données, c'est-à-dire l'ensemble des valeurs que peut prendre une donnée : nombre, chaîne de caractères, date, booléen. Il permet aussi de regrouper les données ayant un lien conceptuel au sein d'une même entité. Il permet également de définir les liens entre plusieurs entités de nature différente. Il permet enfin d'ajouter des contraintes de valeur sur les données.

CREATE,

Une syntaxe de création d'une table est la suivante:

_ Create table nom_table (

◦ Nom_col1 type_donnees,

◦ Nom_col2 type_donnees,

◦ Nom_coln type_donnees)

Dans cette requête, 3 colonnes ont été définies. Le mot-clé « type_donnees » sera à remplacer par un mot-clé pour définir le type de données (INT, DATE, TEXT …). Pour chaque colonne, il est également possible de définir des options telles que (liste non-exhaustive) :

- **NOT NULL** : empêche d'enregistrer une valeur nulle pour une colonne.
- **DEFAULT** : attribuer une valeur par défaut si aucune donnée n'est indiquée pour cette colonne lors de l'ajout d'une ligne dans la table.
- **PRIMARY KEY** : indiquer si cette colonne est considérée comme clé primaire pour un index.
- **FOREIGN KEY (A) REFERENCES R (B)**

L'attribut A est une clé primaire de la table et B est la clé primaire de R.

- **CHECK**: contrainte sur la valeur

DROP,

Supprimer une table avec

◦ **Drop table nom_table**;

ALTER,

Modifier la structure d'une table

Synthaxe : ALTER TABLE nom_table instruction

◦ **ALTER TABLE nom_table ADD nom_colonne type_donnees ;**

RENAME ;

La commande Rename permet de renommer une ou plusieurs tables (Première syntaxe) ou de la transférer vers une autre base de données (seconde syntaxe)

RENAME TABLE table1 TO nouveau nom table1 [, table2 TO nouveau nom table2, ...]

RENAME TABLE base_de_données1.table1 TO base_de_données2.table2

ALTER DATABASE ancienbdname RENAME TO newbdname;

Pour renommer une colonne, il convient d'indiquer l'ancien nom de la colonne et le nouveau nom de celle-ci.

ALTER TABLE nom_table CHANGE colonne_ancien_nom colonne_nouveau_nom type_donnees ;

TRUNCATE

la commande TRUNCATE permet de supprimer toutes les données d'une table sans supprimer la table en elle-même.

TRUNCATE TABLE ` nom_table ` ;

La commande TRUNCATE s'avère être similaire à la commande DELETE sans le Where, lorsqu'elle est utilisée de la façon suivante :

DELETE FROM ` nom_table ` ;

✓ Langage de manipulation de données (LMD) : est la partie du langage SQL qui permet d'effectuer des opérations sur le contenu (LIGNES) d'une ou plusieurs tables.

INSERT,

L'insertion de données dans une table s'effectue à l'aide de la commande INSERT INTO. Cette commande permet au choix d'inclure une seule ligne à la base existante ou plusieurs lignes d'un coup.

La syntaxe pour remplir une ligne en spécifiant toutes les colonnes avec cette méthode est la suivante :

INSERT INTO nom_table VALUES ('valeur 1', 'valeur 2', ...)

Insérer une ligne en spécifiant seulement les colonnes souhaitées Cette deuxième solution est très similaire, excepté qu'il faut indiquer le nom des colonnes avant « VALUES ». La syntaxe est la suivante :

INSERT INTO nom_table (nom_colonne_1, nom_colonne_2, ... VALUES ('valeur 1', 'valeur 2', ...)

UPDATE ;

La commande UPDATE permet d'effectuer des modifications sur des lignes existantes. Très souvent cette commande est utilisée avec WHERE pour spécifier sur quelles lignes doivent porter la ou les modifications.

La syntaxe basique d'une requête utilisant UPDATE est la suivante :

UPDATE nom_table SET nom_colonne_1 = 'nouvelle valeur' WHERE condition

Exemple : Il est possible d'effectuer une modification sur toutes les lignes en omettant d'utiliser une clause conditionnelle. Il est par exemple possible de mettre la valeur "FRANCE" dans la colonne "pays" pour toutes les lignes de la table, grâce à la requête SQL ci-dessous.

UPDATE client SET pays = 'FRANCE'

DELETE ;

La commande DELETE en SQL permet de supprimer des lignes dans une table. En utilisant cette commande associée à WHERE il est possible de sélectionner les lignes concernées qui seront supprimées.

La syntaxe pour supprimer des lignes est la suivante :

DELETE FROM `table` WHERE condition

✓ Langage d'interrogation de données (LID) : SELECT ;

Le langage d'interrogation de données (LID) permet d'établir une combinaison d'opérations portant sur des tables (relation). Le résultat de cette combinaison d'opérations est lui-même une table dont l'existence ne dure qu'un temps.

L'utilisation la plus courante de SQL consiste à lire des données issues de la base de données. Cela s'effectue grâce à la commande SELECT, qui retourne des enregistrements dans un tableau de résultat. Cette commande peut sélectionner une ou plusieurs colonnes d'une table.

SELECT nom_colonne FROM nom_du_table

Jointure

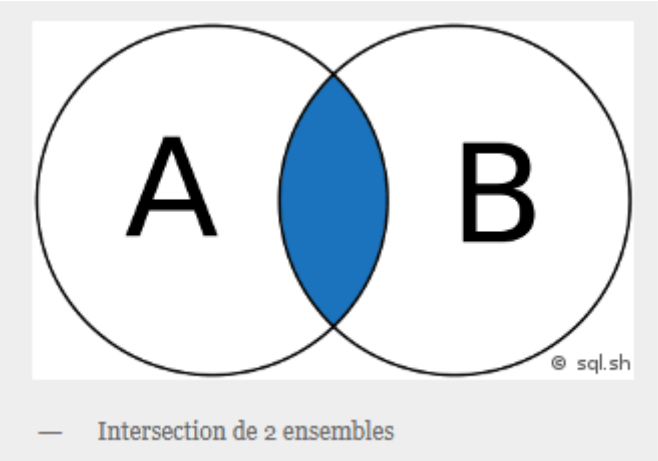
Les jointures en SQL permettent d'associer plusieurs tables dans une même requête. Cela permet d'exploiter la puissance des bases de données relationnelles pour obtenir des résultats qui combinent les données de plusieurs tables de manière efficace.

Types de jointures

Il y a plusieurs méthodes pour associer 2 tables ensemble. Voici la liste des différentes techniques qui sont utilisées :

INNER JOIN **:** jointure interne pour retourner les enregistrements quand la condition est vraie dans les 2 tables. C'est l'une des jointures les plus communes.

SELECT * FROM A INNER JOIN B ON A.key = B.key



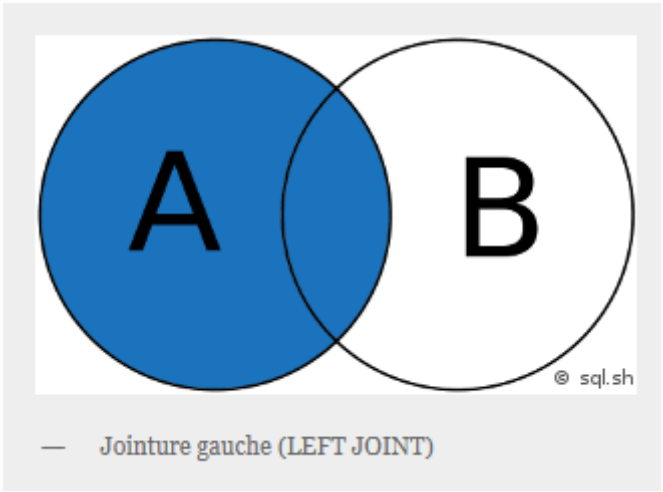
— Intersection de 2 ensembles

CROSS JOIN **:** jointure croisée permettant de faire le produit cartésien de 2 tables. En d'autres mots, permet de joindre chaque ligne d'une table avec chaque ligne d'une seconde table. Attention, le nombre de résultats est en général très élevé.

SELECT * FROM table1 CROSS JOIN table2

LEFT JOIN **(ou LEFT OUTER JOIN) :** jointure externe pour retourner tous les enregistrements de la table de gauche (LEFT = gauche) même si la condition n'est pas vérifié dans l'autre table.

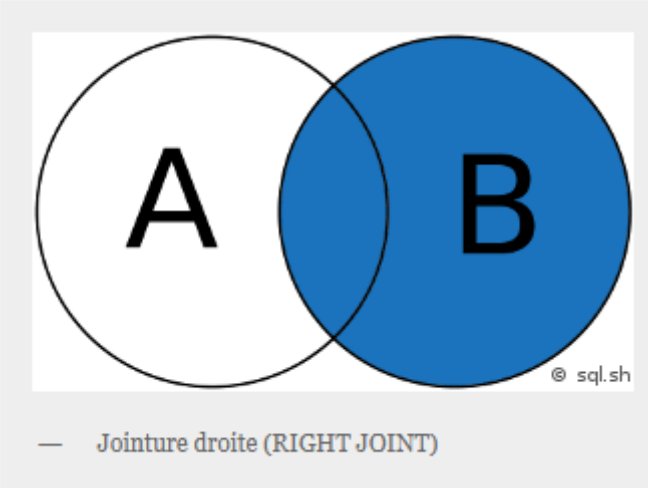
SELECT * FROM A LEFT JOIN B ON A.key = B.key



— Jointure gauche (LEFT JOINT)

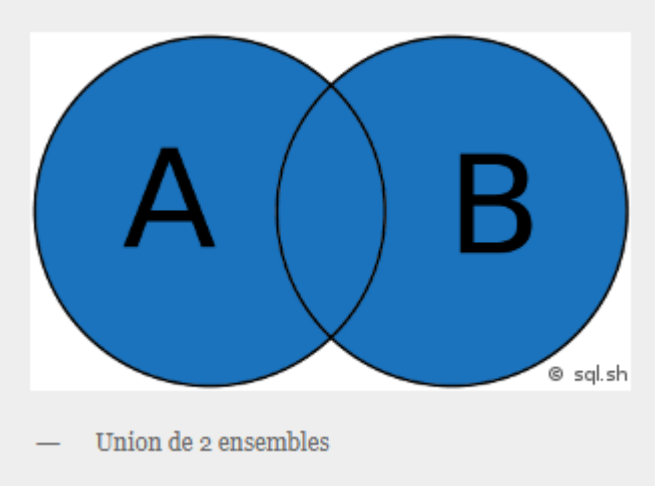
RIGHT JOIN **(ou RIGHT OUTER JOIN) :** jointure externe pour retourner tous les enregistrements de la table de droite (RIGHT = droite) même si la condition n'est pas vérifié dans l'autre table.

SELECT * FROM A RIGHT JOIN B ON A.key = B.key



FULL JOIN **(ou FULL OUTER JOIN) :** jointure externe pour retourner les résultats quand la condition est vrai dans au moins une des 2 tables.

SELECT * FROM A FULL JOIN B ON A.key = B.key



SELF JOIN **:** permet d'effectuer une jointure d'une table avec elle-même comme si c'était une autre table.

NATURAL JOIN **:** jointure naturelle entre 2 tables s'il y a au moins une colonne qui porte le même nom entre les 2 tables SQL

SELECT * FROM table1 NATURAL JOIN table2

UNION JOIN : jointure d'union

Projection

Une projection est une instruction permettant de sélectionner un ensemble de colonnes dans une table.

La sélection de toutes les colonnes de la table se fait par l'instruction :

SELECT * FROM nom_table

La sélection des colonnes Modèle et Série de la table se fait par l'instruction :

SELECT Modèle, Série FROM nom_table

La sélection des colonnes Modèle et Série en éliminant les doublons se fait par l'instruction :

SELECT DISTINCT Modèle, Série FROM nom_table

Sous requête

Dans le langage SQL une sous-requête (aussi appelé “requête imbriquée” ou “requête en cascade”) consiste à exécuter une requête à l’intérieur d’une autre requête. Une requête imbriquée est souvent utilisée au sein d’une clause WHERE ou de HAVING pour remplacer une ou plusieurs constante.

Exemple :

Requête imbriquée qui retourne un seul résultat

SELECT * FROM `table` WHERE `nom_colonne` = (SELECT `valeur` FROM `table2` LIMIT 1)

Requête imbriquée qui retourne une colonne

SELECT * FROM `nom_table` WHERE `nom_colonne` IN (SELECT `colonne` FROM `table2` WHERE `cle_etrangere` = 36)

Alias (AS)

Dans le langage SQL il est possible d’utiliser des **alias** pour renommer temporairement une colonne ou une table dans une requête. Cette astuce est particulièrement utile pour faciliter la lecture des requêtes.

-Alias sur une colonne

La syntaxe pour renommer une colonne de **colonne1** à **c1** est la suivante :

SELECT colonne1 AS c1, colonne2 FROM `table`

Cette syntaxe peut également s’afficher de la façon suivante :

SELECT colonne1 c1, colonne2 FROM `table`

Alias sur une table

La syntaxe pour renommer une table dans une requête est la suivante:

SELECT * FROM `nom_table` AS t1

Cette requête peut également s’écrire de la façon suivante:

SELECT * FROM `table`t1

Commande

- UNION

La commande UNION de SQL permet de mettre bout-à-bout les résultats de plusieurs requêtes utilisant elle-même la commande SELECT. C’est donc une commande qui permet de concaténer les résultats de 2 requêtes ou plus. Pour l’utiliser il est nécessaire que chacune des requêtes à concaténer retournes le même nombre de colonnes, avec les mêmes types de données et dans le même ordre.

Syntaxe

La syntaxe pour unir les résultats de 2 tableaux sans afficher les doublons est la suivante:

SELECT * FROM table1 UNION SELECT * FROM table2

- GROUP BY

La commande GROUP BY est utilisée en SQL pour grouper plusieurs résultats et utiliser une fonction de totaux sur un groupe de résultat. Sur une table qui contient toutes les ventes d’un magasin, il est par exemple possible de lister, regrouper les ventes par clients identiques et d’obtenir le coût total des achats pour chaque client.

SELECT colonne1, fonction (colonne2) FROM table GROUP BY colonne1

- ORDER BY

La **commande ORDER BY** permet de trier les lignes dans un résultat d'une requête SQL. Il est possible de trier les données sur une ou plusieurs colonnes, par ordre ascendant ou descendant.

SELECT colonne1, colonne2 FROM table ORDER BY colonne1

Par défaut les résultats sont classés par ordre ascendant, toutefois il est possible d'inverser l'ordre en utilisant le suffixe DESC après le nom de la colonne.

SELECT colonne1, colonne2, colonne3 FROM table ORDER BY colonne1 DESC, colonne2 ASC

- HAVING

La condition HAVING en SQL est presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que SUM (), COUNT (), AVG (), MIN () ou MAX ().

SELECT colonne1, SUM (colonne2) FROM nom_table GROUP BY colonne1 HAVING fonction (colonne2) operateur valeur

Important : HAVING est très souvent utilisé en même temps que GROUP BY bien que ce ne soit pas obligatoire.

-Function

Les fonctions SQL permettent d'effectuer des requêtes plus élaborées, par exemple adaptant les résultats pour qu'une chaîne soit affichée en majuscule ou bien pour enregistrer une chaîne avec la date actuelle.

- DISTINCT

L'utilisation de la commande SELECT en SQL permet de lire toutes les données d'une ou plusieurs colonnes. Cette commande peut potentiellement afficher des lignes en doubles. Pour éviter des redondances dans les résultats il faut simplement ajouter DISTINCT après le mot SELECT.

SELECT DISTINCT ma_colonne FROM nom_du_tableau

- COUNT

En SQL, la fonction d'agrégation COUNT () permet de compter le nombre d'enregistrement dans une table. Connaître le nombre de lignes dans une table est très pratique dans de nombreux cas, par exemple pour savoir combien d'utilisateurs sont présents dans une table ou pour connaître le nombre de commentaires sur un article.

SELECT COUNT (*) FROM table SELECT COUNT (nom_colonne) FROM table

- LIMIT

La clause LIMIT est à utiliser dans une requête SQL pour spécifier le nombre maximum de résultats que l'ont souhaité obtenir. Cette clause est souvent associée à un OFFSET, c'est-à-dire effectuer un décalage sur le jeu de résultat. Ces 2 clauses permettent par exemple d'effectuer des systèmes de pagination (exemple : récupérer les 10 articles de la page 4).

SELECT * FROM table LIMIT 10

- LIKE %% (B%A)

L'opérateur LIKE est utilisé dans la clause WHERE des requêtes SQL. Ce mot-clé permet d'effectuer une recherche sur un modèle particulier. Il est par exemple possible de rechercher les enregistrements dont la valeur d'une colonne commence par telle ou telle lettre. Les modèles de recherches sont multiples.

SELECT * FROM table WHERE colonne LIKE modèle
LIKE '%a' : le caractère "%" est un caractère joker qui remplace tous les autres caractères. Ainsi, ce modèle permet de rechercher toutes les chaines de caractère qui se termine par un "a".
LIKE 'a%' : ce modèle permet de rechercher toutes les lignes de "colonne" qui commence par un "a".
LIKE '%a%' : ce modèle est utilisé pour rechercher tous les enregistrements qui utilisent le caractère "a".
LIKE 'pa%on' : ce modèle permet de rechercher les chaines qui commence par "pa" et qui se terminent par "on", comme "pantalon" ou "pardon".

- IN & NOT IN
L'opérateur logique IN dans SQL s'utilise avec la commande WHERE pour vérifier si une colonne est égale à une des valeurs comprise dans set de valeurs déterminées. C'est une méthode simple pour vérifier si une colonne est égale à une valeur OU une autre valeur OU une autre valeur et ainsi de suite, sans avoir à utiliser de multiple fois l'opérateur OR.

SELECT nom_colonne FROM table WHERE nom_colonne IN (valeur1, valeur2, valeur3, ...)
Cette syntaxe peut être associée à l'opérateur NOT pour recherche toutes les lignes qui ne sont pas égales à l'une des valeurs stipulées.

- BETWEEN
L'opérateur BETWEEN est utilisé dans une requête SQL pour sélectionner un intervalle de données dans une requête utilisant WHERE. L'intervalle peut être constitué de chaînes de caractères, de nombres ou de dates. L'exemple le plus concret consiste par exemple à récupérer uniquement les enregistrements entre 2 dates définies.

SELECT * FROM table WHERE nom_colonne BETWEEN 'valeur1' AND 'valeur2'
La requête suivante retournera toutes les lignes dont la valeur de la colonne "nom_colonne" sera comprise entre **valeur1** et **valeur2**.
- AVG
La fonction d'agrégation AVG () dans le langage SQL permet de calculer une valeur moyenne sur un ensemble d'enregistrement de type numérique et non nul.

SELECT AVG (nom_colonne) FROM nom_table
- MIN
La fonction d'agrégation MIN () de SQL permet de retourner la plus petite valeur d'une colonne sélectionnée. Cette fonction s'applique aussi bien à des données numériques qu'à des données alphanumériques.

SELECT MIN (nom_colonne) FROM table
SELECT colonne1, MIN (colonne2) FROM table GROUP BY colonne1

- MAX

Dans le langage SQL, la fonction d'agrégation MAX () permet de retourner la valeur maximale d'une colonne dans un set d'enregistrement. La fonction peut s'appliquée à des données numériques ou alphanumériques. Il est par exemple possible de rechercher le produit le plus cher dans une table d'une boutique en ligne.

SELECT MAX (nom_colonne) FROM table SELECT colonne1, MAX (colonne2) FROM table GROUP BY colonne1

- SUM

Dans le langage SQL, la fonction d'agrégation SUM () permet de calculer la somme totale d'une colonne contenant des valeurs numériques. Cette fonction ne fonction que sur des colonnes de types numériques (INT, FLOAT …) et n'additionne pas les valeurs NULL.

SELECT SUM (nom_colonne) FROM table