CONCEPTION DE BASES DE DONNEES

Table des matières

Introduction

Un système d'information est de plus en plus **complexes à mettre en place.**

Le nombre de participants à la conception du SI de plus en plus important

La durée de conception et de mise en œuvre d'un SI de plus en plus importante

L'importance des **enjeux financiers** et des **risques**

D'où la nécessité d'une méthode de conception et de développement des systèmes d'information.

Objectifs des méthodes de conception

Permettre la description des SI à l'aide de **modèles**, selon une **démarche** (étapes) et des moyens de **contrôle qualité**.

Aider à **réaliser le système informatisé** correspondant au système d'information.

Diminuer les **coûts** et les **risques** des projets d'informatisation.

Rendre l'activité de conception et de développement de SI une **activité d'ingénierie** au même titre que le génie mécanique, le génie civil,

Permettre à l'équipe de conception et de développement de disposer d'un **vocabulaire standard**.

I. Notion analyse et conception

• MERISE

MERISE est une méthode de conception, de développement et de réalisation de projets informatiques. Le but de cette méthode est d'arriver à concevoir un système d'information. La méthode MERISE est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques.
La séparation des données et des traitements assure une longévité au modèle. En effet, l'agencement des données n'a pas à être souvent remanié, tandis que les traitements le sont plus fréquemment.

• UML

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.

- Comparaison de MERISE à UML

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 Analyse et Conception

A. MCD

Un modèle conceptuel de données (MCD) est la représentation la plus abstraite des données d'un système d'information. Les données sont représentées sous forme d'entités et d'associations entre entité.

Ce modèle a pour but d'écrire de façon formelle les données qui seront utilisées par le système d'information. Il s'agit donc d'une représentation des données, facilement compréhensible, permettant de décrire le système d'information à l'aide d'entités.

Le MCD est l'élément le plus connu de MERISE et certainement le plus utile. Il permet d'établir une représentation claire des données du SI et définit les dépendances fonctionnelles de ces données entre elles.

- Entités

Une entité représente un objet du SI (acteur, document, concept, …), ou plus exactement un ensemble d'objets ayant les mêmes caractéristiques.

Dans une entité, on met les informations nécessaires et suffisantes pour caractériser cette entité. Ces informations sont appelées **propriétés**.

Une entité est un objet, une chose concrète ou abstraite qui peut être reconnue distinctement :

· ayant une existence propre,

· présentant un intérêt pour l'entreprise,

· traduisant une préoccupation de gestion.

Une propriété particulière, appelée identifiant, permet de distinguer sans ambiguïté toutes les occurrences de l'entité. L'identifiant est toujours souligné. L'identifiant est une propriété qui ne peut pas changer au cours du temps pour une occurrence.

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.

- ## Attributs

Un attribut, dans le contexte d'un modèle conceptuel de données, est une propriété d'une 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 " : ".

La propriété est définie comme étant une donnée élémentaire : ayant un sens, pouvant être utilisée de manière autonome.

- ## Association

C'est un lien entre deux entités (ou plus). On doit lui donner un nom, souvent un verbe, qui caractérise le type de relation entre les entités. Une association possède parfois des propriétés

- ## Occurrence

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

On peut donc représenter une entité comme un ensemble où les éléments sont les occurrences.

- ## Cardinalités

Une cardinalité est un couple de nombres qui exprime la participation des occurrences d'une entité à une association. Ce sont des expressions qui permettent d'indiquer combien de fois au minimum et au maximum le lien entre 2 entités peut se produire.

Pour une association de 2 entités, il y a 4 cardinalités à indiquer.

Il y a trois valeurs typiques : 0, 1 et N (plusieurs).

Les cardinalités traduisent des règles de gestion. Ce sont des règles propres au SI étudié, qui expriment des contraintes sur le modèle.

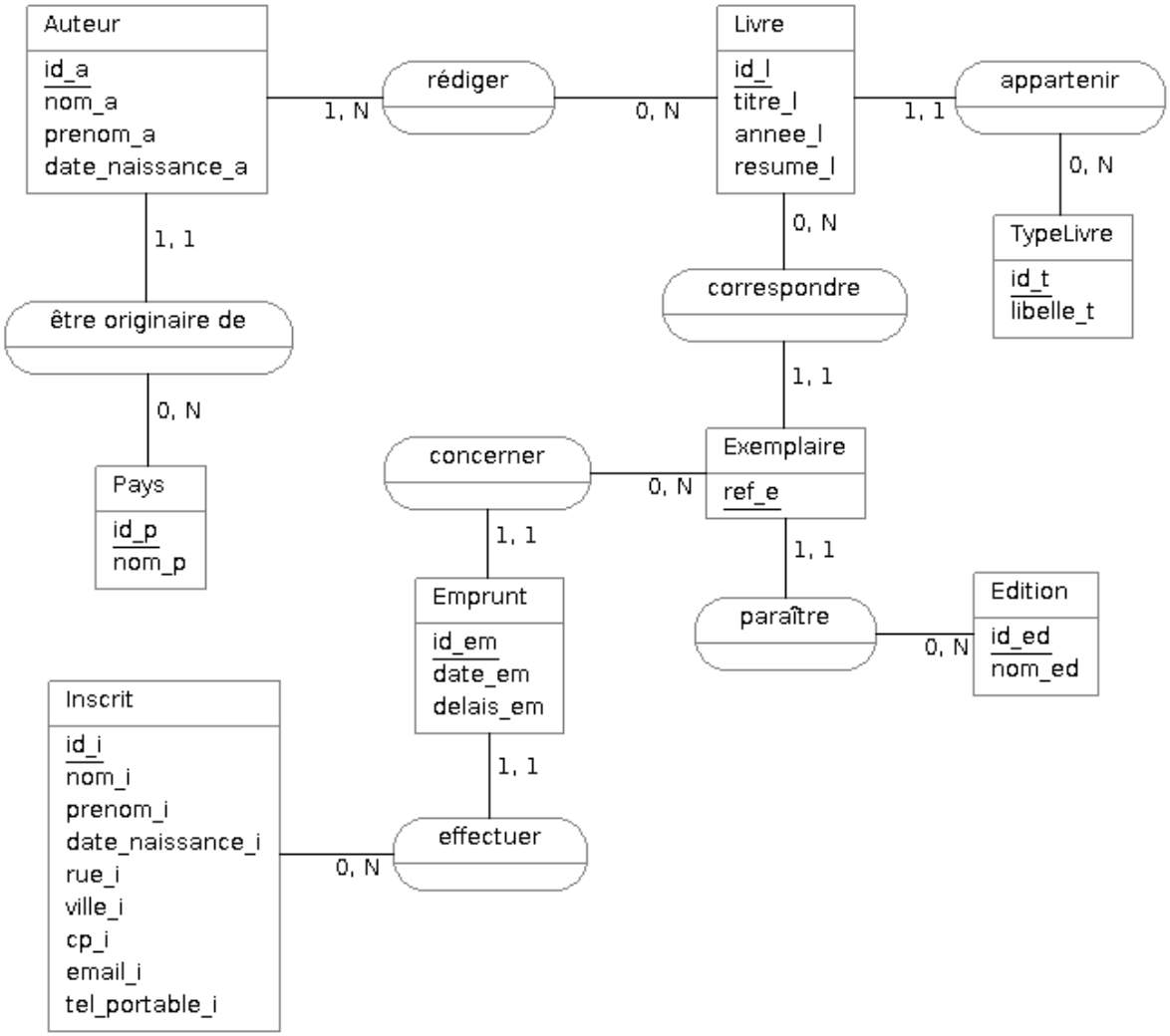
Les cardinalités caractérisent le lien entre une entité et une association. La cardinalité d'une association est le nombre de fois minimal et maximal qu'une occurrence d'une des entités associées peut intervenir dans l'association :

- ***Minimum*** : nombre minimum de fois qu'une occurrence d'une entité participe aux occurrences de l'association, généralement **0** ou **1**.

- ***Maximum*** : nombre maximum de fois qu'une occurrence d'une entité participe aux occurrences de l'association, généralement **1** ou **n**.

Les cardinalités maximales sont nécessaires pour la création de la base de données. Les cardinalités minimales sont nécessaires pour exprimer les contraintes d'intégrités.

- Formalisme de représentation :



B. MLD

- MLD-R

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.

- Le 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.

Les objectifs du modèle relationnel sont :

- Proposer des schémas de données faciles à utiliser ;
- Améliorer l'indépendance logique et physique ;
- Mettre à la disposition des utilisateurs des langages de haut niveau ;
- Optimiser les accès à la base de données ;
- Améliorer l'intégrité et la confidentialité ;
- Fournir une approche méthodologique dans la construction des schémas.

Au modèle relationnel est associée à la théorie de la normalisation des relations qui permet de se débarrasser des incohérences au moment de la conception d'une base de données relationnelle.

➢ Eléments du model relationnel

- **Définition 1 -attribut-** Un attribut est un identificateur (un nom) décrivant une information stockée dans une base.
- Exemples d'attribut : l'âge d'une personne, le nom d'une personne, le numéro de sécurité sociale.
- **Définition 2 -Domaine-** Le domaine d'un attribut est l'ensemble, fini ou infini, de ses valeurs possibles.
- **Définition 3 -relation-** Une relation est un sous-ensemble du produit cartésien de n domaines d'attributs (n > 0).
- Une relation est représentée sous la forme d'un tableau à deux dimensions dans lequel les *n* attributs correspondent aux titres des *n* colonnes.
- **Définition 4 -schéma de relation-** Un schéma de relation précise le nom de la relation ainsi que la liste des attributs avec leurs domaines.

• Règles de passage du MCD au MLD

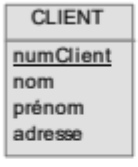
Règle numéro 1 :

Une entité du MCD devient une relation, c'est à dire une table.

Son identifiant devient la clé primaire de la relation

Les propriétés deviennent les attributs de la relation.

Exemple :



CLIENT(numClient , nom , prenom , adresse)
numClient : clé primaire de la table CLIENT

numClient	Nom	Prenom	adresse
1	Dupont	Pierre	5 rue de Paris 93000 Saint-Denis
2	Durand	Raymond	68 rue Alphonse Daudet 77540 Noisy le grand
3	Dupuis	Elisa	1, boulevard Louis Blériot 94800 Villejuif
4	Dubois	Raymonde	15bis, rue de la Gaité 75014 Paris
...	...	...	...

Règle numéro 2 :

Une association de type 1 : N 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.

Exemple :

CLIENT(numClient , nom , prenom , adresse)
numClient : clé primaire de la table CLIENT
COMMANDE(numCommande ,dateCommande , #numClient)
numCommande : clé primaire de la table COMMANDE
#numClient : clé étrangère qui référence numClient de la table CLIENT

Table CLIENT :

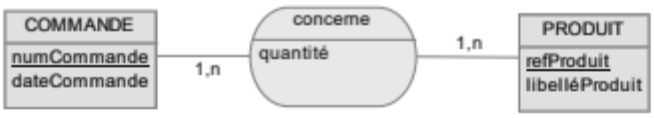
numClient	Nom	Prenom	adresse
1	Dupont	Pierre	5 rue de Paris 93000 Saint-Denis
2	Durand	Raymond	68 rue Alphonse Daudet 77540 Noisy le grand
3	Dupuis	Elisa	1, boulevard Louis Blériot 94800 Villejuif
4	Dubois	Raymonde	15bis, rue de la Gaité 75014 Paris
...	...	...	...

Table COMMANDE :

numCommande	dateCommande	numClient
11	1/02/2014	1
62	1/02/2014	3
423	2/02/2014	3
554	3/02/2014	2
...	...	...

Règle numéro 3 : Une association de type N :N 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.

Exemple :



COMMANDE(numCommande ,dateCommande)
numCommande : clé primaire de la table COMMANDE
PRODUIT(refProduit, libelleProduit)
refProduit : clé primaire de la table PRODUIT
CONCERNE(#numCommande , #refProduit , quantité)
#numCommande , #refProduit : clé primaire composée de la table CONCERNE
#numCommande : clé étrangère qui référence numCommande de la table COMMANDE
#refProduit : clé étrangère qui référence refProduit de la table PRODUIT

Si le nom du MCD n'est pas significatif, on peut renommer le nom de la table.
Dans notre exemple, plutôt que d'appeler la table « CONCERNE », on la nommera « LIGNE_DE_COMMANDE ».

LIGNE_DE_COMMANDE (#numCommande , #refProduit , quantité)
#numCommande , #refProduit : clé primaire composée de la table CONCERNE
#numCommande : clé étrangère qui référence numCommande de la table COMMANDE
#refProduit : clé étrangère qui référence refProduit de la table PRODUIT

Table COMMANDE :

numCommande	dateCommande
11	1/02/2014
62	1/02/2014
423	2/02/2014
554	3/02/2014
...	...

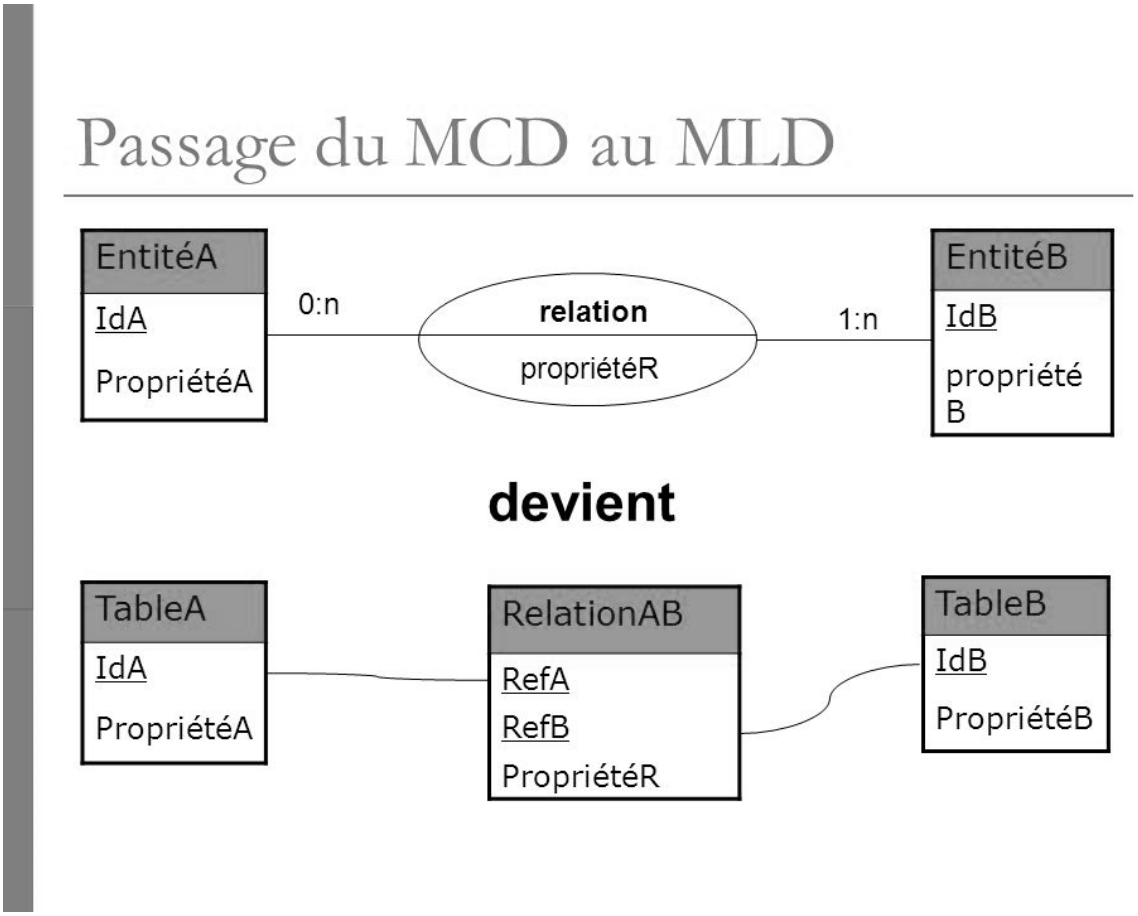
Table PRODUIT :

refProduit	libelleProduit
C24	Chocolat
B12	Bière
L22	Lait
...	...

TableLIGNE_DE_COMMANDE :

numCommande	refProduit	Quantite
11	C24	3
62	B12	3
62	C24	8
423	C24	8765
...	...	...

- Formalisme de représentation :



III. Le langage SQL

- Langage de définition de données :

- CREATE

La commande de création de tables la plus simple ne comportera que le nom et le type de chaque colonne de la table. À la création, la table sera vide, mais un certain espace lui sera alloué.

La syntaxe est la suivante : CREATE TABLE nom_table (nom_col1 TYPE1, nom_col2 TYPE2)

- DROP

Supprimer une table revient à éliminer sa structure et toutes les données qu'elle contient. Les *index* associés sont également supprimés.

La syntaxe est la suivante : DROP TABLE nom_table

- ALTER

Cette commande consiste à apporter des modifications à une table et à une partie des données qu'elle contient.

La syntaxe est la suivante : ALTER TABLE nom_table {ADD/MODIFY}([nom_colonne type])

- RENAME

Cette commande est utilisée pour définir un nouveau nom pour toute table existante.

Voici la syntaxe : RENAME TABLE old_table_name to new_table_name

- TRUNCATE

La commande supprime tous les enregistrements d'une table. Mais cette commande ne détruira pas la structure de la table. Lorsque nous utilisons cette commande sur une table, sa clé primaire (auto-incrémentée) est également initialisée.

Voici sa syntaxe : TRUNCATE TABLE table_name

✓ Langage de manipulation de données :

- INSERT

La commande insert est utilisée pour insérer des données dans une table.

Voici sa syntaxe générale : INSERT INTO table_name VALUES (data1, data2, ...)

- UPDATE

La commande est utilisée pour mettre à jour tout enregistrement de données dans une table.

Voici sa syntaxe générale : UPDATE table_name SET column_name = new_value WHERE some_condition ;

- DELETE

La commande est utilisée pour supprimer des données d'une table.

Voici sa syntaxe générale :

Toutes les données d'une table : DELETE FROM nom_table ;

Un enregistrement particulier : DELETE FROM nom_table WHERE une_condition ;

✓ Langage d'interrogation de données

- SELECT

La requête est utilisée pour récupérer les enregistrements d'une table. Nous pouvons spécifier les noms des colonnes que nous voulons dans l'ensemble de résultats.

Tous les enregistrements d'une table : SELECT * FROM nom_table ;

Un enregistrement particulier : SELECT col1, col2 FROM nom_table WHERE une_condition ;

- ## 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.

Quelques 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

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 vérifiée dans l'autre table.

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

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 vérifiée 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 vraie dans au moins une des 2 tables.

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

- ## SOUS REQUETE

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 constantes.

L'exemple ci-dessous est un exemple typique d'une sous-requête qui retourne une colonne à la requête principale.

SELECT * FROM `table`

WHERE `nom_colonne` IN (SELECT `colonne` FROM `table2` WHERE `cle_etrangere` = une_clé)

- ## 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.

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

SELECT colonne1 AS c1, colonne2 FROM `table`

- ## COMMANDES

➢ UNION

La commande UNION de SQL permet de mettre bout-à-bout les résultats de plusieurs requêtes utilisant elles-mêmes 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

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.

De façon générale, la commande GROUP BY s'utilise de la façon suivante

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.

Une requête où l'on souhaite filtrer l'ordre des résultats utilise la commande ORDER BY de la sorte :

SELECT colonne1, colonne2 FROM table ORDER BY colonne1

➢ 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().

Exemple : SELECT client, SUM (tarif) FROM achat GROUP BY client HAVING SUM (tarif) > 40

- ## FONCTION

➢ 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.

Exemple : SELECT DISTINCT prenom FROM client ;

➢ 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.

Exemple : Pour compter le nombre d'utilisateurs total depuis que le site existe, il suffit d'utiliser COUNT(*) sur toute la table : SELECT COUNT(*) FROM utilisateurs

➢ LIMIT

La clause LIMIT est à utiliser dans une requête SQL pour spécifier le nombre maximum de résultats que l'on souhaite obtenir.

La syntaxe commune aux principaux systèmes de gestion de bases de données est la suivante :
SELECT * FROM table LIMIT 10

➢ LIKE

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.

Exemple : Si l'on souhaite obtenir uniquement les clients des villes qui commencent par un "N", il est possible d'utiliser la requête suivante :

SELECT * FROM client WHERE ville LIKE 'N%'

➢ 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.

Exemple : Si l'on souhaite obtenir les membres qui se sont inscrit entre le 1 avril 2012 et le 20 avril 2012 il est possible d'effectuer la requête suivante :

```
SELECT * FROM

Utilisateurs WHERE date_inscription BETWEEN '2020-04-01' AND '2020-04-20'
```

➢ 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.

Exemple : Pour connaitre le montant moyen effectué par chaque client, il est possible d'utiliser une requête qui va utiliser :

```
SELECT client, AVG (tarif) FROM achat GROUP BY client ;
```

➢ 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.

Exemple : Pour extraire le prix du produit le moins cher de la catégorie "maison", il est possible d'effectuer la requête SQL ci-dessous :

```
SELECT MIN(prix) FROM `produits` WHERE `categorie` = 'maison' ;
```

➢ 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.

Exemple :

Pour extraire uniquement le tarif le plus élevé dans la table, il est possible d'utiliser la requête suivante : SELECT MAX(prix) FROM produit ;

➢ 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.

Exemple : Pour calculer le montant de la facture n°1 il est possible d'utiliser la requête SQL suivante : SELECT SUM(prix) AS prix_total FROM facture WHERE facture_id = 1

Sources :

https://www.cours-gratuit.com/cours-merise/introduction-au-modele-conceptuel-des-donnees-mcd

https://web.maths.unsw.edu.au/~lafaye/CCM/relation/relintro.htm

https://sites.google.com/site/pasunier/home/SIIE/MCD/mcd-e2

https://www.univ-orleans.fr/lifo/Members/Mirian.Halfeld/Cours/BD/iutA2-intro.pdf

https://laurent-audibert.developpez.com/Cours-BD/?page=bases-de-donnees-relationnelles

http://tony3d3.free.fr/files/Passage-du-MCD-au-MLD.pdf

https://www.studytonight.com/dbms/introduction-to-sql.php

https://sql.sh/