Le SQL (Structured Query Language – langage de requêtes structuré) est un langage informatique de dialogue avec une base de données relationnelle.

Une requête est une question posée à une base de données.

1 – Les requêtes d'interrogation

11 – La logique d'interrogation

Procédure	Opération relationnelle	Ordre SQL
Champs (colonnes) à afficher	Projection	SELECT
Tables concernées		FROM
Conditions permettant de restreindre les enregistrements (lignes) à afficher	Sélection	WHERE
	Jointure	JOIN
Classement des données affichées	Tri	ORDER BY

Toute requête SQL se termine par un **;** (point-virgule).

Par convention, les instructions SQL sont écrites en majuscule dans le code d'un programme afin de les distinguer du langage de programmation (ex : dans page web dynamique, les instructions SQL seront en majuscule et le code PHP en minuscule).

12 – Les opérations de base

121 – La projection

La projection est une opération permettant de n'afficher qu'une partie des champs (colonnes) d'une table.

Modèle relationnel

Client (Codecli, Nom, Prenom, Adresse, CP, Ville, Tel)

Clé primaire : Codecli

Requête 1 : Afficher toutes les informations concernant les client.

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Tous
Tables concernées	Client
Conditions	Aucune
Classement	Aucun

Requête SQL

```
SELECT *
FROM Client ;
```

Indique d'afficher tous les champs.

Requête 2 : Afficher les nom et prénom des clients.

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Nom, Prenom
Tables concernées	Client
Conditions	Aucune
Classement	Aucun

Requête SQL

```
SELECT Client.Nom, Client.Prenom
FROM Client ;
```

Le nom de table est séparé du nom du champ par un . (point).

Les champs sont séparés par une , (virgule).

Lorsque l'on est certain qu'il n'y a aucun risque d'homonymie dans les noms de champs utilisés dans la base de données, il est possible de ne pas mentionner le nom de la table devant le nom du champ.

122 – La sélection

La sélection est une opération permettant de n'afficher qu'une partie des enregistrements (lignes) d'une table.

La sélection peut être monocritère ou multicritère. Elle repose sur l'utilisation d'opérateurs de comparaison :

Opérateur	Signification
=	Est strictement égal à ; peut être utilisé avec tout type de données
>	Est strictement supérieur à ; est utilisé avec des données numériques
<	Est strictement inférieur à ; est utilisé avec des données numériques
>=	Est supérieur ou égal à ; est utilisé avec des données numériques
=<	Est inférieur ou égal à ; est utilisé avec des données numériques
LIKE	Comme ; peut être utilisé avec tout type de données
BETWEEN	Entre ; peut être utilisé avec tout type de données
AND	Et ; tous les critères de comparaison doivent être vrais
OR	Ou ; au moins un des critères de comparaison doit être vrai

Les valeurs de comparaison peuvent nécessiter une syntaxe particulière.

Syntaxe	Microsoft Access	Norme SQL
Remplacer une chaîne de caractères	*	%
Remplacer un seul caractère	?	_
Champ au format texte	"	'
Champ au format date	#mm/jj/aaaa#	'mm/jj/aaaa'
Valeur de comparaison saisie par l'utilisateur	[Texte à afficher]	

Pour ne pas afficher les doublons (deux enregistrements avec les mêmes données d'affichées), il faut ajouter l'opérateur DISTINCT après l'ordre SELECT. L'opérateur DISTINCT prendra en compte tous les champs mentionnés dans l'ordre SELECT.

Modèle relationnel

Client (Codecli, Nom, Prenom, Adresse, CP, Ville, Tel)

Clé primaire : Codecli

Requête 1 : Afficher les nom et prénom des clients qui habitent Lillers.

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Nom, Prenom
Tables concernées	Client
Conditions	Ville = Lillers
Classement	Aucun

Requête SQL

```
SELECT Client.Nom, Client.Prenom
FROM Client
WHERE Client.Ville= "Lillers";
```

Requête 2 : Afficher le nom et le numéro de téléphone des clients dont le prénom est Marc ou Lucie.

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Tel
Tables concernées	Client
Conditions	Prenom = Marc ou Prenom = Lucie
Classement	Aucun

Requête SQL

```
SELECT Client.Nom, Client.Tel
FROM Client
WHERE Client.Prenom LIKE "Marc" OR Client.Prenom LIKE "Lucie";
```

Requête 3 : Afficher le nom et le prénom des clients dont le numéro de téléphone commence par 032136 et dont la première lettre du nom est comprise entre A et F.

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Nom, Prenom
Tables concernées	Client
Conditions	Tel commence par 032136 et Nom compris entre A et F
Classement	Aucun

Requête SQL

```
SELECT Client.Nom, Client.Prenom
FROM Client
WHERE Client.Tel LIKE "032136%" AND Client.Nom BETWEEN "A" AND "F";
```

Requête 4 : Afficher les villes dans lesquelles on a des clients.

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Ville
Tables concernées	Client
Conditions	Aucun
Classement	Aucun

Requête SQL

```
SELECT DISTINCT Client.Ville
FROM Client ;
```

123 – Les jointures

La jointure est une opération qui met en relation plusieurs tables, par l'intermédiaire des liens qui existent entre la clé primaire de l'une et la clé étrangère de l'autre.

La jointure est une opération de sélection car elle permet de ne retenir que les enregistrements pour lesquels la valeur de la clé primaire d'une table correspond à la valeur de la clé étrangère d'une autre table.

Modèle relationnel

Client (Codecli, Nom, Prenom, Adresse, CP, Ville, Tel)

Clé primaire : Codecli

Commande (Num, Date, Codecli)

Clé primaire : Num

Clé étrangère : Codecli en référence à Codecli de Client

123-1 Jointure interne : elles sélectionnent les données d'une des tables qui respectent le critère de jointure.

Afficher numéros de commande de Sosthène Baheux.

Extrait des tables

Extrait de la table Client							Extrait de la table Commande		
Codecli	**Nom**	**Prenom**	**Adresse**	**CP**	**Ville**	**Tel**	**Num**	**Date**	**Codecli**
41100035	Bacquet	Adeline	10 rue platini	62170	Saint-Josse	0321459631	C20050215	23/03/2005	41100094
41100094	Baheux	Sosthène	296 rue des jumeaux	59500	Douai	0328627431	C20050183	26/06/2005	41100035
41100095	Baillieu	Anastasia	277 boulevard danton	59600	Maubeuge	0328639248	C20050341	18/11/2005	41100094
41100097	Mori	Jean	1 rue des Etournaux	32110	Moulins	489170469			
41100126	Renault	Marie	3 boulevard Victor Hug	59500	Douai	336373900			

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Num
Tables concernées	Client, Commande
Conditions	Codecli = Codecli et Nom = Baheux et Prenom = Sosthene
Classement	Aucun

Requête SQL ancienne version utilisation de WHERE

```
SELECT Commande.Num
FROM Client, Commande
WHERE Client.Codecli = Commande.Codecli AND Client.Nom LIKE "Baheux" AND Client.Prenom LIKE "Sosthène" ;
```

La requête précédente effectue une **jointure interne** entre deux tables : seules les lignes qui respectent la condition de la jointure sont affichées.

Afin d'améliorer la lisibilité des requêtes (notamment bien séparer ce qui relève du filtrage et ce qui relève de la jointure), et dans une certaine mesure de les optimiser, une nouvelle syntaxe a été introduite pour les jointures.

Requete SQL "nouvelle formule" avec JOIN et ON

```
SELECT Commande.Num
FROM Commande
INNER JOIN Client ON Client.codecli = Commande.Codecli
WHERE Client.Nom LIKE "Baheux"AND Client.Prenom LIKE "Sosthène"
```

Remarque la commande INNER est comprise par défaut

123-1 Jointure externe : elles sélectionnent les données d'une des tables qui ne respectent pas le critère de jointure.

- LEFT JOIN : jointure de gauche

Le LEFT JOIN (ou LEFT OUTER JOIN) implique que l'on sélectionne toutes les lignes respectant le critère de jointure, puis on ajoute toutes les lignes de la table "TableGauche" qui ont été rejetées car elles ne respectaient pas le critère de jointure.

Le principal intérêt est de pouvoir identifier les lignes qui ne correspondent pas au critère de jointure et d'obtenir un résultat avec un NULL.

Afficher numéros de commande de Sosthène Baheux et les autres. (Dans la table de droite suivant notre exemple)

```
SELECT DISTINCT Commande.Num, Client.Nom
FROM Commande
LEFT JOIN Client ON Client.codecli = Commande.Codecli
AND Client.Nom LIKE "Baheux" AND Client.Prenom LIKE "Sosthène"
```

Résultat :

Num	Nom
C20050215	Baheux
C20050341	Baheux
C20050183	*NULL*

124 – Le tri

Le tri est une opération qui consiste à classer les enregistrements en fonction d'un ou plusieurs critères. La clause ORDER BY dispose de deux instructions de tri :

ASC	Tri dans l'ordre croissant ; un tri dans l'ordre alphabétique est un tri croissant ; un tri dans l'ordre chronologique est un tri croissant
DESC	Tri dans l'ordre décroissant

Requête 1 : Afficher numéros de commande de Sosthène Baheux, du plus récent au plus ancien.

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Num
Tables concernées	Client, Commande
Conditions	Codecli = Codecli et Nom = Baheux et Prenom = Sosthene
Classement	Date dans l'ordre décroissant

Requête SQL

```
SELECT Commande.Num
FROM Client, Commande
WHERE Client.codecli = Commande.Codecli AND Client.Nom LIKE "Baheux" AND Client.Prenom LIKE "Sosthène"
ORDER BY Commande.Date DESC ;
```

Avec Jointure

```
SELECT Commande.Num
FROM Client
INNER JOIN Commande ON Client.codecli = Commande.Codecli
WHERE Client.Nom LIKE "Baheux"
AND Client.Prenom LIKE "Sosthène"
ORDER BY Commande.Date DESC
```

I

res données et éventuellement d'un calcul. Ces nouvelles données sont affichées dans un nouveau champ, créé pour l'occasion. Ce champ n'est créé que pour la requête. Il ne modifie en rien la structure de la table.

Modèle relationnel

Produit (Codeproduit, Designation, PU_HT)

Clé primaire : Codeproduit

Requête 1 : Afficher le prix unitaire TTC des Produit, avec un taux de TVA de 19,60%. Cette donnée sera affichée dans un champ nommé PU_TTC.

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Designation, PU_TTC
Tables concernées	Produit
Conditions	Aucune
Classement	Aucun

Requête SQL

```
SELECT Produit.Designation, Produit.PU_HT*1.196 AS PU_TTC
FROM Produit ;
```

L'opérateur AS permet de créer un nouveau champ pour afficher le résultat de l'opération.

Ce champ n'est pas créé dans la table. En conséquence, ne pas mettre de nom de table devant.

14 – Les fonctions d'agrégat

Les fonctions d'agrégat permettent d'effectuer des opérations mathématiques sur un ensemble d'enregistrements sélectionnés.

141 – Compter les enregistrements

Le décompte des enregistrements sélectionnés se fait à l'aide de la fonction COUNT.
Syntaxe : COUNT ()
Le résultat de l'opération sera affiché dans un nouveau champ.

Modèle relationnel

Produit (Codeproduit, Designation, PU_HT)

Clé primaire : Codeproduit

Requête 1 : Compter le nombre de Produit dont le prix unitaire hors taxes est supérieur à 20,00 €.

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Quantite_Produit
Tables concernées	Produit
Conditions	PU_HT > 20
Classement	Aucun

Requête SQL

```
SELECT COUNT (Produit.Codeproduit) AS Quantite_Produit
FROM Produit
WHERE Produit.PU_HT > 20 ;
```

N'importe quel champ de la table peut être retenu pour le décompte des enregistrements sélectionnés.

142 – Additionner les valeurs d'un champ numérique

L'addition des valeurs sélectionnées d'un champ numérique se fait à l'aide de la fonction SUM.
Syntaxe : SUM ()
Le résultat de l'opération sera affiché dans un nouveau champ.

Modèle relationnel

Comporter (Codeproduit, Num, Quantite)

Clé primaire : Codeproduit, Num

Requête 1 : Combien de Produit P81425 ont été commandés ?

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Quantite_Produit
Tables concernées	Comporter
Conditions	Codeproduit = P81425
Classement	Aucun

Requête SQL

```
SELECT SUM (Comporter.Quantite) AS Quantite_Produit
FROM Produit
WHERE Comporter.Codeproduit LIKE " P81425" ;
```

143 – Calculer la moyenne des valeurs d'un champ numérique

Le calcul de la moyenne des valeurs sélectionnées d'un champ numérique se fait à l'aide de la fonction AVG (Average).
Syntaxe : AVG ()
Le résultat de l'opération sera affiché dans un nouveau champ.

Modèle relationnel

Produit (Codeproduit, Designation, PU_HT)

Clé primaire : Codeproduit

Requête 1 : Quel est le prix moyen des Produit ?

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Prix_moyen
Tables concernées	Produit
Conditions	Aucun
Classement	Aucun

Requête SQL

```
SELECT AVG (Produit.PU_HT) AS Prix_moyen
FROM Produit ;
```

144 – Afficher la valeur minimale d'un champ numérique

L'affichage de la valeur minimale, parmi les valeurs sélectionnées d'un champ numérique, se fait à l'aide de la fonction MIN.

Syntaxe : MIN ()

Le résultat de l'opération sera affiché dans un nouveau champ.

Modèle relationnel

Produit (Codeproduit, Designation, PU_HT)

Clé primaire : Codeproduit

Requête 1 : Quel est le prix minimum des Produit ?

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Prix_minimum
Tables concernées	Produit
Conditions	Aucun
Classement	Aucun

Requête SQL

```
SELECT MIN (Produit.PU_HT) AS Prix_minimum
FROM Produit ;
```

145 – Afficher la valeur maximale d'un champ numérique

L'affichage de la valeur maximale, parmi les valeurs sélectionnées d'un champ numérique, se fait à l'aide de la fonction MAX.

Syntaxe : MAX ()

Le résultat de l'opération sera affiché dans un nouveau champ.

Modèle relationnel

Produit (Codeproduit, Designation, PU_HT)

Clé primaire : Codeproduit

Requête 1 : Quel est le prix maximum des Produit ?

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	Prix_maximum
Tables concernées	Produit
Conditions	Aucun
Classement	Aucun

Requête SQL

```
SELECT MAX (Produit.PU_HT) AS Prix_maximum
FROM Produit ;
```

15 – Les clauses de regroupement

Les clauses de regroupement permettent de réaliser des opérations sur des groupes d'enregistrements.

Procédure	Opération relationnelle	Ordre SQL
Champs (colonnes) à afficher	Projection	SELECT
Tables concernées		FROM
Conditions permettant de restreindre les enregistrements (lignes) à afficher	Sélection Jointure	WHERE
Regroupement des données affichées Hors programme	*Regroupement*	*GROUP BY*
Conditions (filtres) permettant de restreindre les données affichées par le regroupement Hors programme	*Sélection*	*HAVING*
Classement des données affichées	Tri	ORDER BY

151 – La clause GROUP BY

La clause GROUP BY permet de créer des groupes d'enregistrements sur lesquels pourront être utilisées les fonctions d'agrégat (voir point 14). Elle est nécessaire dès lors que l'on souhaite afficher des données issues des tables et des données issues de fonctions d'agrégat.

Syntaxe : GROUP BY Champ1, Champ2, …

Modèle relationnel

Comporter (Codeproduit, Num, Quantite)

Clé primaire : Codeproduit, Num

***Requête 1** : Indique pour chaque commande le nombre de références de Produit commandés ?*

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	*Num, Nb_ref*
Tables concernées	*Comporter*
Conditions	*Aucun*
Regroupement	*Num*
Classement	*Aucun*

Requête SQL

```
SELECT Comporter.Num, COUNT (Comporter.Codeproduit) AS Nb_ref
FROM Comporter
GROUP BY Comporter.Num ;
```

Tous les champs ne faisant pas partie d'une fonction d'agrégat de l'ordre SELECT doivent être repris dans la clause GROUP BY.

152 – La clause HAVING

La clause HAVING permet d'appliquer des sélections sur les regroupements créés à l'aide de la clause GROUP BY. Contrairement à l'ordre WHERE qui sélectionne les enregistrements, la clause HAVING sélectionne les résultats d'une fonction d'agrégat.

Syntaxe : HAVING critères de sélection

Modèle relationnel

Comporter (Codeproduit, Num, Quantite)

Clé primaire : Codeproduit, Num

***Requête 1** : Afficher les commandes dont le nombre de références de Produit commandés est supérieur à 5 ?*

Analyse préparatoire à l'élaboration de la requête

Champs à afficher	*Num, Nb_ref*
Tables concernées	*Comporter*
Conditions	*Aucun*
Regroupement	*Num*
Filtres	*Nb_Ref > 5*
Classement	*Aucun*

Requête SQL

```
SELECT Comporter.Num, COUNT (Comporter.Codeproduit) AS Nb_ref
FROM Comporter
GROUP BY Comporter.Num
HAVING COUNT (Comporter.Codeproduit) > 5 ;
```

Le filtre porte sur la fonction d'agrégat et non son résultat (le champ calculé créé.

2 – Les requêtes de modification

Les requêtes de modification ont pour objet la modification des données stockées dans les tables.

21 – Les requêtes d'insertion de données

Les requêtes d'insertion permettent d'ajouter un enregistrement dans une table.

Syntaxe :

INSERT INTO *table (champ1, champ2, ...)*

VALUES *('valeur1',' valeur2', ...)* ;

Indique le nom de la table et les champs où ajouter l'enregistrement.

Indique les données à ajouter. Elles suivent impérativement l'ordre des champs indiqué par l'ordre INSERT INTO.

Modèle relationnel

Produit (Codeproduit, Designation, PU_HT)

Clé primaire : Codeproduit

Requête 1 : Ajouter un nouveau produit dans la table Produit : code=P14689 ; Désignation=Tout ce que l'élève de terminale CFE a toujours voulu savoir sur la comptabilité et finance d'entreprise ; Prix unitaire hors taxes=22,00 €.

Analyse préparatoire à l'élaboration de la requête

Tables concernées	Produit
Champs concernés	Tous

Requête SQL

```
INSERT INTO Produit (Codeproduit, Designation, PU_HT)
VALUES ('P14689','Tout ce que l'élève de terminale CFE a toujours voulu savoir sur la comptabilité et finance d'entreprise','22') ;
```

22 – Les requêtes de mise à jour de données

Les requêtes de mise à jour permettent de modifier les données stockées dans les tables.

Syntaxe :

UPDATE *table*

SET *champs à modifier*

WHERE *sélection* ;

La modification peut résulter de la saisie ou du calcul d'une nouvelle valeur pour un champ donné.

Modèle relationnel

Produit (Codeproduit, Designation, PU_HT)

Clé primaire : Codeproduit

Requête 1 : Modifier la désignation du produit dont le code est P14689. La nouvelle désignation sera « La bible du bachelier STG CFE ».

Analyse préparatoire à l'élaboration de la requête

Tables concernées	Produit

Champs concernés	Designation
Condition	Codeproduit = P14689

Requête SQL

```
UPDATE Produit
SET Produit.Designation="La bible du BTS IRIS"
WHERE Produit.Codeproduit="P14689" ;
```

Requête 2 : Augmenter de 2% les prix de tous les Produit.

Analyse préparatoire à l'élaboration de la requête

Tables concernées	Produit
Champs concernés	PU_HT
Condition	Aucune

Requête SQL

```
UPDATE Produit
SET Produit.PU_HT= Produit.PU_HT*1.02 ;
```

23 – Les requêtes de suppression de données

Les requêtes de suppression permettent de supprimer des données stockées dans les tables. La suppression concerne l'intégralité de l'enregistrement.

Syntaxe :

DELETE FROM *table*
WHERE *sélection* ;

En l'absence de critère de sélection, tous les enregistrements de la table seront supprimés.

Modèle relationnel

Produit (Codeproduit, Designation, PU_HT)

Clé primaire : Codeproduit

Requête 1 : Retirer de la base de données le produit dont le code est P14689.

Analyse préparatoire à l'élaboration de la requête

Tables concernées	Produit
Condition	Designation = "lampadaire"

Requête SQL

```
DELETE FROM Produit
WHERE Produit.Designation ="lampadaire" ;
```