- Between (entre) : date BETWEEN #01/01/2000# AND #31/01/2000#
- Like (comme) : nom LIKE "A*" (qui commence par A) ; nom LIKE "?A*" (qui contient un A en deuxième place)
- IN (dans) dept IN (67, 68, 90, 88, 57) (qui est dans la liste donnée).
- IN idem mais la liste peu être le résultat d'une requête :

  WHERE dept IN (SELECT deptNum from Departement)

Remarque, on ne peut pas :

- Utiliser de champs qui ne sont pas dans la liste des champs contenu dans les tables de la clause FROM !!!!
- Utiliser de fonction de regroupement ici.

Post-condition : une post-condition est une condition, comme ci-dessus, mais qui peut intégrer des fonctions de regroupement.

La post-condition permet de faire une sélection/restriction dans les lignes affichées par la requête, au niveau du résultat de la requête, après les calcule et regroupements.

Remarque : on ne peut pas y utiliser d'alias, il faut y recalculer les valeurs.

LMD : Modification des données contenues dans les tables

Trois clauses : DELETE FROM, INSERT INTO, UPDATE

Ces trois clauses n'utilisent QU'UNE SEULE TABLE à la fois

Syntaxes :

DELETE FROM table [WHERE condition] ;

WHERE sert à cibler les enregistrements à supprimer. Sans WHERE, on vide la table !!

INSERT INTO table [(liste_de_champs)] VALUES (liste_de_valeurs) ;

La liste de valeurs doit être dans le même ordre que la liste de champs.

Si la liste de champs est omise, la requête utilise la liste de champ complète de la table, selon l'ordre défini dans la requête de création de celle-ci.

UPDATE table SET affectation [, affectation ...] [WHERE condition]

Une affectation est un champ de la table qui est affecté d'une valeur résultante d'une expression :

champ= expression

Une expression peut être une constante, un calcul, même fait avec les champs de l'enregistrement à modifier.

WHERE sert à ciber les enregistrements à modifier. Sans WHERE, on modifie toutes les lignes

LDD : Modifier la structure des tables, de la base de données

CREATE {TABLE | VIEW} table_ou_vue {(liste de définitions de champs ou de contraintes) | AS requete sélection} ;

Annexe 2 : Les Liens sympa

Pour aller plus loin ou en cas de doute, visiter les liens des sites suivants, il ont des exemples, des tutos ou des cours, des forums sur différents langages dont le SQL.

Developpez.com est un site plein d'info très bien présentées et utile comme site de référence.

http://sqlpro.developpez.com/cours/sqlaz/fonctions/

http://sqlpro.developpez.com/cours/sqlaz/ddl/?page=partie2

http://www.developpez.net

Le site du zéro moins pertinent en SQL, il reste une référence en développement

http://www.siteduzero.com

Comment ça marche est un site qui était très bien mais s'est diversifié (sans critique) et amodifié son interface sans trop de bonheur. La navigation sur le site abouti plus à des forums qu'à des articles de référence. Dommage : www.commentcamarche.net

Annexe 1 : Pseudo Mémento SQL

Les données entre [] sont optionnels.
Le signe ...] signifie que le contenu des crochets peut être répété.
Les caractères { et } encadrent une option, une valeur qui comporte une alternative d'écriture. Le caractère | signifie qu' l'on peut utiliser l'indication qui se trouve à gauche OU l'indication qui se trouve à droite (/ex. {*|champ} = écrire * ou un nom de champ)

Une requête peut être écrite indifféremment sur une ou plusieurs lignes. Il est meilleur de mettre un ; pour indiquer la fin de la requête. Aujourd'hui, les moteurs SQL tolèrent son absence.

LID : Sélection (Restriction et projection)

```
SELECT [DISTINCT] {* | liste_de_champs_affichés}
FROM liste_de_tables
[WHERE pré-condition]
[GROUP BY liste_de_champs_de_regroupement]
[HAVING post-condition]
[ORDER BY liste_de_tri]
```

Distinct indique qu'il faut masquer les lignes en double.

liste_de_champs_affichés est une liste d'expressions séparées par une virgule.
Une expression est un nom de champ, un calcul, une constante, etc. ..., une fonction d'agrégat (regroupement, voir la liste ci-dessous)
On peut y renommer l'expression en lui donnant un alias avec la commande AS.
Syntaxe : expr. [AS alias][, expr. [AS alias] ...]
Exemple : numero_adherent, nom_adherent, (cotisationMensuelle*12) AS cotisationAnnuelle, avg(cotisationMensuelle) AS moyenneMensuelle
Remarque : éviter de mettre des espaces dans les alias (quels qu'ils soient)

liste_de_tables est une liste de tables, séparées par une virgule.
On peut renommer une table en ajoutant l'alias à la suite du nom de la table (sans AS)
Syntaxe : table alias [, table alias ...]
Exemple : Adherent Adh, Emprunter Emp, Ouvrage Ouv
Remarque : Il est interdit d'ouvrir deux fois la même table dans la clause FROM. Pour le faire, il faut impérativement lui donner un alias
Exemple : Adherent **Adh1**, Emprunter, Adherent **Adh2**

liste_de_champs_de_regroupement est la liste de **tous** les champs qui ne contiennent pas de fonction d'agrégat dans son expression.
Cette liste est obligatoire dès qu'il y a un mix de champs agrégés (regroupés) et non agrégés.
Syntaxe : GROUP BY champs [,champ ...]
Exemples :

```
SELECT numero_adherent, count(*) FROM emprunter
GROUP BY numero_adherent ;
```

```
SELECT numero_adherent, nom-adherent, count(*) FROM emprunter
GROUP BY numero_adherent, nom_adherent ;
```

liste_de_tri : Liste de champs (n'importe lesquels, même des alias), suivi du sens de tri à réaliser
Syntaxe : ORDER BY champ [ASC | DESC] [,champ [ASC | DESC] ...]
Exemple : ORDER BY nom_Adherent, prenom_Adherent DESC
Remarques : si le sens est omis, il est ascendant, à partir du moment où n utilise des fonctions d'agrégats ou GROUP BY, le résultat est trié par ordre croissant.

Pré-condition : c'est une condition de sélection des enregistrement AVANT regroupements.
Une condition (pré ou post) est une suite de **comparaison** ou de fonctions booléennes (= dont le résultat est VRAI/FAUX) et reliées par des opérateurs logiques (ET, OU = AND, OR).
Exemple : prix>15.3 AND poidsBrut<=poidsNet*1.20 ; montant>10 AND (Type="A" OR type="B")

Comparaison : Une comparaison est une égalité ou inégalité entre deux **expressions**.
Une **expression** est un nom de champ, un calcul ou une constante.
Les **opérateurs de calcul** sont les opérateurs classiques : + -, *, / et les fonctions : SQRT, COS, SIN, YEAR(date), MONTH(date), etc. ...
Les **opérateurs de comparaison** sont les opérateurs classiques : <, >, <=, >=, = et certains opérateur spécifiques :

57. Supprimer la vue ListeCAClient que vous venez de créer.

6 Conclusion

Le langage SQL est compliqué mais plein de ressources et sa connaissance avancée permet de décharger avantageusement la programmation de l'application qui utilise le SGBD.

Nom du champ	Type et longueur
clinum	Entier de 8
cliNom	Chaine de caractères alphanumériques de 50
cliAdresse	Texte de longueur indéfinie
cliCP	Chaine de 15 caractères
cliVille	Chaine de 30 caractères
cliTel	Chaine de 15 caractères
artCode	Chaine de 8 caractères
artDesc	Chaine de 250 caractères
artPrixHT	Réel de 15 chiffres dont 5 décimales
artTyp	Chaîne de 8 caractères

5.3 Modifier : ALTER TABLE

La seule modification est l'ajout ou la suppression d'une contrainte de clé étrangère.
Syntaxe :

```
ALTER TABLE table {ADD|DROP|ALTER} contrainte
```

Une contrainte indique des conditions à respecter. Nous n'utiliserons que les contraintes de clé étrangère, les contraintes de référence.
Syntaxe d'une contrainte de clé étrangère :

```
CONSTRAINT [nom_contrainte] FOREIGN KEY 'liste_de_champs1' REFERENCES nom_table ('liste_de_colonne2')
```

La liste de champs 1 donne ceux qui FONT référence A, la liste de champs 2 sont ceux qui SONT référence DE. Les champs doivent être dans le même ordre.
Exemple :

```
CREATE TABLE Commande (
       cdeNum INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
       cdeDate DATE NOT NULL,
       cliNum INT(11)
);
ALTER TABLE Commande ADD CONSTRAINT FOREIGN KEY cliNum REFERENCES Client(cliNum) ;

CREATE TABLE LigCdeArt (
       cdeNum INT(11) NOT NULL,
       artCode VARCHAR(15),
       PRIMARY KEY (cdeNum, artCode)
);
ALTER TABLE LigCdeArt Add Constraint Foreign Key cdeNum References Commande(cdeNum) ;
ALTER TABLE LigCdeArt Add Constraint Foreign Key artCode References Article(artCode) ;
```

Dans MySQL, il faut mettre des `(apostrophes inversés) autour des champs, comme suit :

```
ALTER TABLE `LigCdeArt` Add Constraint Foreign Key `artCode` References Article(`artCode`) ;
```

53. Créer la clé étrangère dans la table Article (artTyp référence typeArtCode dans TypeArticle).

5.4 Les Vues : CREATE|DROP VIEW

Une vue est une requête qui a été enregistrée dans la base de données. Elle sert à être exécutée souvent sans avoir à être réécrite en permanence. Elle sert aussi à faciliter l'écriture de certaines requêtes compliquées (du type calcul du CA des clients ...).

On va donc créer une vue comme le résultat d'une requête : Syntaxe :

```
CREATE VIEW vue AS SELECT … le reste de la requête ;
```

La difficulté de la vue réside dans la requête SELECT qui lui sert de base. C'est tout.

Il est possible de modifier une vue, ce n'est pas très utile en général. Autant la supprimer et la recréer (à moins qu'elle soit la cible de contrainte de référence ... ce qui n'est pas très pertinent !).

On supprime une vue comme une table : Syntaxe :

```
DROP VIEW vue ;
```

54. Reprendre la requête n° 35 et en faire une vue, le champ calculé sera nommé caTotal (le chiffre d'affaire en € HT). La vue sera nommée ListeCAClient
55. Utiliser la vue pour afficher les clients qui ont un caTotal supérieur à 1000.
56. Reprendre les questions 35 et 38, comparez, quelle requête est la plus facile à utiliser ?

46. Le client précédent passe une commande de 3 articles "Clés USB 16Go InfoTech". Rédiger les requêtes nécessaires.
    Pour la date de la commande, utiliser la (ou l'une des) fonction :
    - (Access et MySql) : NOW()
    - (MySql) CURDATE(),CURRENT_DATE(), SYSDATE ➔ date du jour au format 'AAAA-MM-JJ' (texte) ou AAAAMMJJ (numérique), selon le contexte ;

4.3 Modifier : UPDATE

```
UPDATE table SET affectation [, affectation …]
[WHERE pré-condition] ;
```

Une affectation est un transfert du résultat d'une valeur dans un champ de la table.
Voici différents exemples :

```
SET prix=3, qte=qte+1, codeArt = upper(substr(descArt, 0, 6)), nbr=(select
count(*) from Commande)
```

47. Modifier l'article "ABC123" et changer la description pour "Article ABC"
48. Le client N° 3528, change d'adresse, il est maintenant au 234 Rue de L'industrie, 67390 Marckolsheim.
49. Les articles sont revalorisé, augmentez les prix de 3%
50. (difficile) un nouveau champ a été ajouté à la table Client. Il contient le CA brut HT du client. Recalculez le CA du client n°3528.

5 LDD SQL : Créer, modifier et supprimer des objets de la base

Les clauses CREATE, ALTER et DROP peuvent agir sur différents objets du SGBD : Des tables, des vues mais aussi des utilisateurs ou des bases de données, entre autres.
Nous n'observerons que les tables et les vues. Les autres objets (bases, utilisateurs, procédures, triggers) ne sont pas de notre ressort.

5.1 Supprimer : DROP TABLE

```
DROP TABLE table ;
```

Attention, la table ne doit pas être référencée par aucune clé étrangère dans une autre table. Auquel cas, il faut d'abord détruire la table dépendante.

51. Supprimer la table test et la table essai

5.2 Créer : CREATE TABLE

```
CREATE TABLE table (Liste de définition de champs et contraintes) ;

CREATE [{ GLOBAL | LOCAL } TEMPORARY] TABLE nom_table
(colonne | contrainte_de_table [{ , colonne | contrainte_de_table }...] )
```

La liste de définition de champs et contraintes sont des lignes séparées par des virgules.

Elles on la syntaxe suivante :
nomChamp *type* [NOT] NULL [Default valeurParDéfaut] [AUTO_INCREMENT] [PRIMARY KEY],

Type : les types usuels sont : (pour MySQL / Access)
- Texte : VARCHAR(longueur)
- Entier : INTEGER(longueur) ou INT(longueur)
- Réel : FLOAT(mantisse, décimales) : mantisse= nombre de chiffres significatifs.
- Date : DATE : date au format aaaa-mm-jj (année, mois, jour)

Quelques types moins fréquents mais pratiques : (pour MySQL / Access)
- Texte : TEXT : texte de longueur indéfinie
- Entier : LONG(Longueur) (Access) : entier de plus grande longueur
- Réel : DOUBLE(mantisse, décimales) : réel de plus grande précision (longueur)
- Heure : TIME : heure au format HH:MM:SS:NNN (heures, minutes, secondes, millisecondes)
- Enuméré : ENUM(liste de valeurs) : la valeur de ce champ doit appartenir à la liste donnée. Utile pour des valeurs comme sexe qui prend la valeur F ou M, idéal pour des listes de 5 10 valeurs au maxi, qui ne changent jamais.

Contrainte : la seule contrainte qu l'on voit ici est celle de clé primaire.
Soit on écrit PRIMARY KEY derrière le champ concerné, soit on écrit une ligne spéciale qui commence par PRIMARY KEY et est suivi de la liste des champs qui composent la clé primaire. Cette contrainte est toujours indiquée lors de la création de la table, /ex :

```
PRIMARY KEY cliNum
```

52. Créer les tables Client puis Article. On utilisera les définitions de champ suivantes :

25. Quelle est la liste des Clients qui n'ont passés aucune commande ?
26. Quels sont les articles qui ont étés commandés (ce sont ceux qui sont dans la liste des articles de la table LigneCdeArticle)

2 SQL avec agrégat de lignes

Un agrégat est un regroupement de lignes afin de faire un calcul (comptage, somme ou moyenne) ou d'extraire une valeur extrême (maximum ou minimum)
Lorsque l'on demande des regroupement par quelquechose (n° de client, code article, référence, nom, ville, etc...) il faut indiquer la liste des champs de regroupement (tous ceux qui sont dans le SELECT mais ne sont pas dans une fonction de regroupement) dans la clause GROUP BY, placée à la fin de la requête.

27. Compter le nombre de lignes de la table Commande
28. Combien y a-t-il de clients
29. Combien y a-t-il de commandes pour le client numéro 1
30. Combien y a-t-il de commande par numéro de client ?
31. Combien y a-t-il d'articles par type article (afficher le code type, la description du type et le nombre d'article, renommer la colonne calculée en nbreArticles)
32. Faire la liste des noms de client et du nombre de commande qu'ils ont passés.
33. Revoir la requête de la question 21 et faire la somme des montants hors taxes pour la commande 1.
34. Reprendre la requête précédente et faire la somme des montants hors taxes par commande. (c'est presque la requête la plus compliquée de la série). Elle permet de calculer le chiffre d'affaire (CA) par commande.
35. Reprendre la requête précédente et faire la somme des montants hors taxes par client au lieu de commande. Que permet de calculer cette requête ?

3 SQL avec agrégat et Having

La clause HAVING se place après la clause GROUP BY. Elle sert à ajouter une nouvelle condition de sélection (comme dans WHERE) mais en plus, elle permet d'utiliser les champs qui ont été calculés, ce que ne peut pas faire la clause WHERE.

36. Reprendre la question 30 et ne garder que les clients qui ont plus de 5 commandes
37. Reprendre la question 31 et ne garder que les types qui ont moins de 5 articles, trier par ordre croisant de nombre d'articles.
38. Reprendre la question 35 et ne garder que les clients qui ont un total (un chiffre d'affaire en € HT) supérieur à 1000 (c'est la requête la plus compliquée de la série)

4 LMD SQL : insérer, modifier, supprimer des enregistrements

4.1 Supprimer : DELETE FROM

```
DELETE FROM table [WHERE pré_condition] ;
```

39. Supprimer l'article dont le code est "ABC123"
40. Supprimer la commande numéro 123

Attention, dans certains cas, la suppression est impossible : on ne peut pas supprimer une commande s'il existe des lignes de commande article. Ainsi, il faut faire deux requêtes pour supprimer une commande :

41. Supprimer les lignes de la commande n° 1 puis la commande n°1

Remarque, on peut utiliser tout type de condition dans le WHERE :

42. Supprimer les commandes (qui sont encore vides) du mois de Février 2013.
43. Supprimer le mauvais client numéro 2 qui a déjà passé plusieurs commandes.

4.2 Insérer : INSERT INTO

```
INSERT INTO table [(listeDeChamps)] VALUES (listeDeValeurs) ;
```

La liste de valeurs doit être dans le même ordre que la liste de champs et, à défaut, que la liste des champs de la table. Une valeur particulière peut être produite par une requête.
La liste complète des valeurs peut aussi être produite par une requête dont les colonnes correspondent exactement aux champs insérés.

44. Insérer l'article (code = "ABC123", description="Article test", prix HT= 17,25)
45. Insérer le client dont la fiche est :
    N° client est 3528, le nom= abc info, l'adresse = 15 rue des Iris, le CP= 67000, la ville=Strasbourg.

Note : lorsqu'un champ (souvent la clé) est auto-incrémenté, il ne faut pas lui donner de valeur. Le SGBD se charge de le remplir automatiquement avec le dernier numéro+1.
Un champ qui n'est pas valorisé recevra 0, vide ou la valeur par défaut indiquée à la construction de la table.

Avant de commencer, analyser le schéma relationnel avec les questions suivantes :

- Comment sait-on à qui appartient une commande ?
- Tous les clients ont-ils passé une commande ? Expliquer pourquoi la réponse est non.
- Un client peut-il passer deux commandes le même jour ?
- Comment savoir si un article a été commandé ou non ?
- Combien d'articles peut-il y avoir dans une commande ?
- Pourquoi la quantité est-elle notée dans la relation LigneCdeArticle et pas dans la relation Commande ni dans la relation Article ?

SELECT ... FROM uneTable ;

1. Quelle est la liste des articles
2. Quelle est la liste des clients
3. Quelle est la liste des noms de ville où sont implantés les clients
4. Quelle est la liste des Codes postaux et noms de ville où sont implantés les clients, triée par code postal
5. Même question, sans afficher deux fois la même réponse

SELECT ... FROM ... WHERE condition simple ;
Une condition simple est une comparaison d'un champ avec une valeur. Voir les opérateurs de comparaison dans le mémento en annexe.

6. Quelle est la liste des clients de Strasbourg
7. Quelle est la liste des articles qui coûtent plus de 150€
8. Quelle est la liste des articles dont le code commence par "A"
9. Quelle est la liste des articles qui coûtent plus de 150€ mais moins de 300€
10. Quels sont les noms des clients n° 1, 2, 4

SELECT ... FROM ... WHERE condition composée ;
Une condition composée est une condition à plusieurs critères, une condition composée de plusieurs comparaisons reliées par des opérateurs logiques (ET ou OU ➔ AND ou OR). Voir les nouveaux opérateurs de comparaison dans le mémento en annexe.

11. Quelle est la liste des clients de Strasbourg dont le nom commence par "A"
12. Quelle est la liste des commandes du 01/01/2012 du client n°1 et du client n°3
13. Quelle est la liste des clients alsaciens

SELECT ... FROM deux tables WHERE ...;

Quel est le nom du client de la commande n°3 ?

```
SELECT nomCli FROM Commande, Client
WHERE cdeNum=3
AND     cde.noCli=client.noCli ;
```

14. Quelle est la liste des commandes du client nommé "Toto"
15. Quelle est la liste des codes article de plus de 1000€, commandés au moins une fois.
16. Quelle est la liste des codes article qui sont dans la commande n°1

SELECT ... FROM n tables WHERE ...; (n >= 3)

17. Quels sont les codes articles et leur prix qui ont étés commandés le 01/01/2013
18. Quels sont les codes article et leur prix hors taxes qui ont étés commandés par le client n°1
19. Quels sont les codes articles et leur prix hors taxes qui ont étés commandés par le client nommé Alfred

a) sans le prix :

```
SELECT artCode , artDesc FROM Client, Commande, ligneCdeArticle, Article
WHERE nomCli="Mélanie"
AND     Client.noCli=Cde.noCli
AND     Cde.noCde=ligneCodArticle.noCde
AND     ligneCdeArticle.codeArt=Article.codeArt
```

SELECT ... FROM n tables WHERE ...; avec des calculs

20. Faire la liste des codes article, quantité et prix hors taxes de la commande n°1
21. Faire la même liste en calculant en plus, pour chaque ligne, le montant hors taxes
22. Faire la même liste que ci-dessus pour le client n°1
23. Faire la même liste que ci-dessus pour le client nommé Alfred entre le 1er et le 31 Janvier 2013

Utilisation de listes

24. Que signifie la requête suivante :
    SELECT * FROM Client WHERE cliNum IN (SELECT cliNum FROM Commande) ;

Clé primaire : cdeNum , artCode
Clé étrangère : cdeNum est en dépendance de référence avec cdeNum dans la table Commande.
Clé étrangère : artCode est en dépendance de référence avec artCode dans la table Article.

1.2.2 Base Bibliothèque

Adherent(adhNum, adhNom, adhPrenom, adhAdresse, adhTel, adhCarteNum)
Clé primaire : adhNum
Auteur(autNum, autNom, autPrenom, autPseudo)
Clé primaire : autNum
Editeur(edtNum, edtNom, edtAdresse)
Clé primaire : edtNum
Theme(thmNum, thmLibelle)
Clé primaire : thmNum
Ouvrage(ouvNum, ouvtitre, ouvdate, edtNum, autNum)
Clé primaire : ouvNum
Clé étrangère : autNum est en dépendance de référence avec autNum dans la table Auteur.
Appartenir(ouvNum, thmNum)
Clé primaire : ouvNum, thmNum
Clé étrangère : ouvNum est en dépendance de référence avec ouvNum dans la table Ouvrage.
Clé étrangère : thmNum est en dépendance de référence avec thmNum dans la table Theme.
Emprunter(ouvNum, adhNum, empDebDate, empFinDate)
Clé primaire : ouvNum, adhNum, empDebDate
Clé étrangère : ouvNum est en dépendance de référence avec ouvNum dans la table Ouvrage.
Clé étrangère : adhNum est en dépendance de référence avec adhNum dans la table Adherent.
Reserver(ouvNum, adhNum, resaDate)
Clé primaire : ouvNum, adhNum, resaDate
Clé étrangère : ouvNum est en dépendance de référence avec ouvNum dans la table Ouvrage.
Clé étrangère : adhNum est en dépendance de référence avec adhNum dans la table Adherent.

1.2.3 Base Tintin (extraits)

Album(noAlb, titreAlb, dateAlb)
Clé primaire : cdeNum , artCode
Personnage(noPers, nomPers, prenomPers, gentilPers)
Clé primaire : cdeNum , artCode
Juron(noJuron, nomJuron)
Clé primaire : cdeNum , artCode
Participer(noAlb, noPers)
Clé primaire : (noAlb, noPers)
Clé étrangère : noAlb est en dépendance de référence avec noAlb dans la table Album.
Clé étrangère : noPers est en dépendance de référence avec NoPers dans la table Personnage.
Prononcer(noAlb, noPers, noJuron, noPage)
Clé primaire : (noAlb, noPers, noPers, noPage)
Clé étrangère : noAlb est en dépendance de référence avec noAlb dans la table Album.
Clé étrangère : noPers est en dépendance de référence avec NoPers dans la table Personnage.
Clé étrangère : noJuron est en dépendance de référence avec noJuron dans la table Juron.
Clé étrangère : (noAlb, noPers) est en dépendance de référence avec (noAlb, noPers) dans la table Participer.

La base Tintin remplie est disponible sous MS-Access 2003.

1.3 Requêtes SQL simples (base Gestion commerciale)

Les requêtes sont faites avec un langage nommé **Structured Query Language** ou **SQL**.
Il contient plusieurs parties dont :

- Le Langage d'Interrogation de Données ou **LID** (SELECT …),
- Le langage de Manipulation des Données ou **LMD** (INSERT INTO, UPDATE, DELETE)
- Le langage de Définition des Données ou **LDD** (CREATE TABLE, CREATE VIEW, DROP, ALTER),
- Le Langage de Contrôle des Données ou **LCD** (Non étudié ; CREATE USER, GRANT, REVOKE).

SQL, Exercices

Ce document contient des exercices progressifs de langage SQL. Ils nécessitent des connaissances minimum sur le langage d'interrogation des données (LID : SELECT) ainsi que des connaissances sur la construction des conditions
Note : un memento SQL est placé en fin de document (Annexe 1)

1 Révisions de 1ère

1.1 Le schéma relationnel

Une relation est un peu comme une feuille d'un tableur, avec des colonnes = des propriétés et des lignes de données = des enregistrements ou tuples.
L'ensemble des feuilles de données, le classeur, qui représentent l'ensemble des données traitées est une base de données.
La description des feuilles : nome de la feuille et liste des noms de colonnes utilisées est équivalent au schéma relationnel.

Définitions :
Le SR représente la liste des champs contenus dans les relations de la base de données sous la forme d'une liste comme suit :

Client (id, nom_Client, prenom_Client, adresse_Client, tel_Client)

Les propriétés (synonyme de champs) ne contiennent qu'une et une seule valeur.
On peut écrire prenom_Client="Geronimo"
On NE peut PAS écrire prenom_Client="Geronimo, Zidane, Alfred"

Toute relation possède OBLIGATOIREMENT un champ dont la valeur est unique pour chaque enregistrement et qui permet de distinguer chaque ligne de toutes les autres. Ce champ est nommé "Clé primaire".
On écrira :

Client (id, nom_Client, prenom_Client, adresse_Client, tel_Client)
Clé primaire : id

Pour éviter, dans une feuille/relation "Commande", de répéter des informations déjà présentes dans une autre relation (le nom, prénom, adresse du client) à chaque enregistrement, on utilise juste la clé primaire de cette autre relation.
Ces relations entre tables sont des dépendances de référence.
Dans notre relation ce champ prendra le nom de "Clé étrangère".
Exemple :

Commande(id, date_Commande, numero_Client)
Clé primaire : id
Clé étrangère : numero_Client en référence à id de la table Client.

Notez bien la forme de la phrase, c'est cette forme qui est utilisée sur les documents comme les devoirs.

1.2 Bases d'exemples

Dans la suite, nous utiliserons plusieurs schémas relationnels
Notes :

Nous respecterons exactement l'orthographe utilisée pour les noms des champs et tables,
Les champs clés sont normalement nommés **id** (comme identifiant), mais pour des raisons d'homonymie, nous donnerons ici des noms significatifs.

1.2.1 Base Gestion Commerciale :

Client(clinum, cliNom, cliAdresse, cliCP, cliVille, cliTel)
Clé primaire : cliNum
TypeArticle(CodeTypeArt, descTypeArt)
Clé primaire : codeTypeArt
Article(artCode, artDesc, artPrixHT, codeTyp)
Clé primaire : artCode
Clé étrangère : codeType est en dépendance de référence avec codeTypeArt de TypeArticle.
Commande(cdeNum, cdeDate, cliNum)
Clé primaire : cdeNum
Clé étrangère : cliNum est en dépendance de référence avec cliNum dans la table Client.
LigneCdeArticle(cdeNum, artCode, quantite)