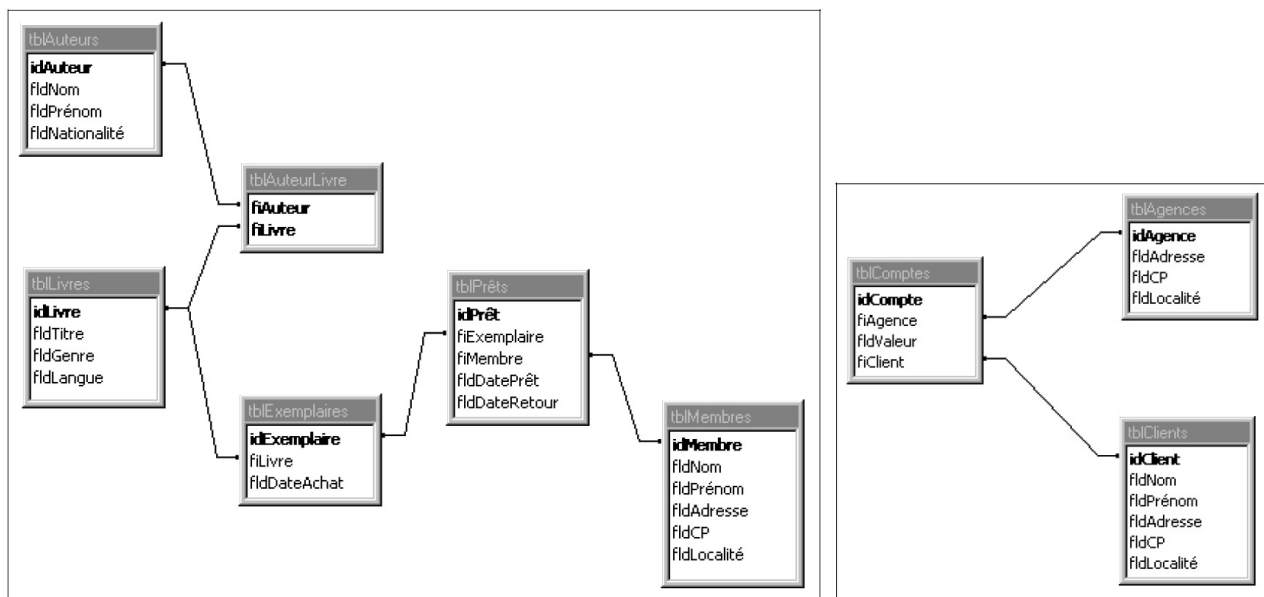


## Quelques remarques pour les exercices SQL

Certains exemples se basent sur les BD suivantes :



## Les requêtes SQL

### o Règles simples

- 1) Ne jamais renommer les noms des champs. Donc en aucun cas ajouter les préfixes *id*, *fi*, *fld* ou *tbl* s'ils ne font pas partie des noms des champs
- 2) L'ordre des clauses est : SELECT – FROM – WHERE – GROUP – HAVING – ORDER.
- 3) Quand on précise la table d'un champ on écrit d'abord la table, puis le champ, donc *tblClient.fldNom* et pas l'inverse. Même règle pour les alias : *Cl.fldNom*.
- 4) Pas de DISTINCT dans une requête groupée.

### o Procédé général

Dans la description ci-dessous, un **champ élémentaire** désigne un champ qui existe tel quel dans la base de donnée, p.ex. *fldNom*. Un **champ affiché** correspond à une colonne et peut être un champ élémentaire ou un **champ composé**. Un champ composé peut être un **champ calculé** (p.ex. *fldQuantité\*fldPrix*) ou un **champ d'agrégation** (un champ avec une fonction d'agrégation comme p.ex. *AVG(fldSalaire)*).

Pour déterminer la requête SQL selon un énoncé donné, on peut procéder comme suit :

#### ✓ La clause SELECT

Repérer les champs à afficher.

Ecrire la clause SELECT avec tous les champs affichés en respectant l'ordre de l'énoncé.

Tous les champs composés doivent être suivis d'un alias (utiliser AS).

Exemple : *SELECT fldDépartement, AVG(fldSalaire) AS [salaire moyen] FROM...*

L'énoncé peut demander d'indiquer un alias pour un champ élémentaire.

Exemple : *SELECT fldNom, fldBonClient AS [Code Spécial] FROM...*

#### ✓ La clause FROM

Pour établir une liste des tables concernées, repérer d'abord tous les champs élémentaires affichés et aussi les champs élémentaires des champs composés. Déterminer ensuite toutes les tables concernées par ces champs.

Exemple : Soit la requête

"Afficher par auteur le nombre de livres français empruntés au mois d'avril 2009".

Il faudra afficher trois champs : *fldNom* et *fldPrénom* de la table *tblAuteurs* et *COUNT(idLivre)* (puisque l'on compte les livres). On a donc pour la clause SELECT :  
*SELECT fldNom, fldPrénom, COUNT(idLivre) AS Nombre*. Les tables concernées par ces trois champs sont *tblAuteurs* et *tblLivres*.

Toutes les tables doivent en générale être liées par des jointures. Il se peut qu'il faut ajouter d'autres tables non concernées par les champs mais nécessaires pour que toutes les tables soient reliées par des jointures.

Même exemple : Il faut donc ajouter la table *tblAuteurLivre* pour que les tables *tblAuteurs* et *tblLivres* soient reliées.

Si la requête possède un filtre (WHERE) concernant des champs appartenant à des tables autres que celles déjà repérés, alors on a deux possibilités :

- a) Intégrer ces tables dans la clause FROM et utiliser la méthode par **jointure**
- b) Ne pas intégrer ces tables et procéder par **imbrication**

Ecrire la clause FROM. Les tables peuvent être suivies d'alias (AS pas nécessaire mais permis).

Même exemple : Nous avons 2 filtres : 1) les livres doivent être en langue française. Ceci concerne la table *tblLivres* qui est déjà dans notre liste. 2) les livres ont été empruntés en mis d'avril 2009. Ceci concerne le champ *fldDatePrêt* de la table *tblPrêts*. Nous avons donc 2 possibilités :

- a) jointure : intégrer cette table *tblPrêts* et avec elle aussi la table *tblExemplaires* (pour faire le lien) ce qui donne le résultat suivant *FROM tblAuteurs, tblAuteurLivre Al, tblLivres, tblExemplaires Ex, tblPrêts*. Ici nous avons ajouté 2 alias Al et Ex à cause de l'ambiguïté du champ *fiLivre* qui existe dans les tables *tblAuteurLivre* et *tblExemplaires*.
- b) imbrication : ne pas intégrer ces tables : *FROM tblAuteurs, tblAuteurLivre, tblLivres*

#### ✓ La clause WHERE

Ecrire les jointures nécessaires pour lier toutes les tables de la clause FROM. Il faut en générale (N-1) jointures pour N tables.

Même exemple : Nos deux possibilités : a) *WHERE idAuteur=fiAuteur AND Al.fiLivre=idLivre AND idLivre=Ex.fiLivre AND idExemplaire=fiExemplaire*  
et b) *WHERE idAuteur=fiAuteur AND fiLivre=idLivre*

Ecrire les autres filtres. Ils peuvent être simples (p.ex. *fldLocalité="Wiltz"*) ou imbriqués.

Même exemple : Pour le premier filtre : *AND fldLangue='FRA'* et pour le deuxième filtre nos deux possibilités : a) *AND fldDatePrêt BETWEEN #04/01/2009# AND #04/30/2009#*

b) (sous-requête imbriquée, nous avons ici encore une fois 2 possibilités a et b pour les mêmes raisons) *AND idLivre IN (SELECT DISTINCT fiLivre FROM tblExemplaires WHERE idExemplaire in (SELECT DISTINCT fiExemplaire FROM tblPrêts WHERE fldDatePrêt BETWEEN #04/01/2009# AND #04/30/2009#))* ou bien *AND idLivre IN (SELECT DISTINCT fiLivre FROM tblExemplaires, tblPrêts WHERE idExemplaire=fiExemplaire AND fldDatePrêt BETWEEN #04/01/2009# AND #04/30/2009#)*

Attention : une condition concernant une fonction d'agrégation doit obligatoirement figurer dans la clause HAVING.

#### ✓ La clause GROUP BY

Dès qu'une requête affiche à la fois des champs non-d'agrégation et des champs d'agrégation on est obligé d'ajouter une clause GROUP BY. Dans cette clause doivent figurer tous les champs non-d'agrégation et dans le même ordre que dans la clause SELECT.

Exemple : Dans une requête qui commence par *SELECT fldNom, SUM(fldValeur) AS Avoir* on est obligé de grouper par *fldNom*.

Il est permis de grouper par des champs qui ne figurent pas dans la clause SELECT.

Exemple : Dans la requête *SELECT fldNom, SUM(fldValeur) AS Avoir FROM tblClients, tblComptes WHERE fiClient=IdClient GROUP by idClient, fldNom*, nous avons groupé aussi par *idClient* pour éviter d'additionner les comptes de deux clients différents mais de même nom.

### ✓ La clause HAVING

La clause HAVING sert à noter les filtres sur les champs d'agrégation.

Il faut noter le champ comme il est indiqué dans la clause SELECT et non pas son alias.

Exemple : "Les noms des clients qui possèdent plus de 1 000 000€" : *SELECT fldNom, SUM(fldValeur) AS Avoir FROM tblClients, tblComptes WHERE fiClient=idClient GROUP by fldNom HAVING SUM(fldValeur)>1000000* et non pas *HAVING Avoir>1000000*.

Il est permis d'avoir un filtre sur un champ d'agrégation même si ce champ ne figure pas dans la clause SELECT.

Exemple : "Les noms des clients possédant plus d'un compte" : *SELECT fldNom FROM tblClients, tblComptes WHERE fiClient=idClient GROUP by fldNom HAVING COUNT(\*) > 1*, donc sans afficher le nombre de comptes.

### ✓ La clause ORDER BY

Syntaxe : *ORDER BY champ1 [ASC/DESC], champ2 [ASC/DESC], ...*

Si on trie par un champ composé, il faut noter le champ comme il est indiqué dans la clause SELECT et non pas son alias.

Exemple : "Les noms des clients de Wiltz triés par leur avoir (de leurs comptes)" : *SELECT fldNom, SUM(fldValeur) AS Avoir FROM tblClients, tblComptes WHERE fiClient=idClient GROUP by fldNom ORDER BY SUM(fldValeur)* et non pas *ORDER BY Avoir*.

Si on ne met ni ASC ni DESC, alors l'ordre de tri est par défaut l'ordre croissant (ASC)

Une indication ASC ou DESC ne se rapporte qu'à un seul champ.

Exemple : Si on tri par plusieurs champs dans l'ordre décroissant, il faut noter DESC pour chaque champ : *ORDER by DateDépart DESC, HeureDépart DESC*.

### ○ Autres remarques

#### ✓ DISTINCT

Il faut mettre distinct pour éviter des doublons.

DISTINCT suit presque toujours le mot SELECT, se rapporte à tous les champs qui suivent et n'utilise pas de parenthèses.

Exemple : "Afficher les clients qui ont un compte avec une valeur supérieure à 999999€".  
*SELECT DISTINCT fldNom, fldPrénom FROM tblClients, tblComptes WHERE idClient=fiClient AND fldValeur>999999*; Il est inutile d'afficher le même client plus d'une fois.

Exception : "Combien de villes différentes proposent une agence?" *SELECT COUNT(DISTINCT fldLocalité) FROM tblAgences*;

A l'examen, c'est souvent l'énoncé qui indique qu'il faut mettre DISTINCT. Exemple : "Affichez les villes des agences, en ne tenant compte qu'une seule fois de chaque ville."

DISTINCT se place aussi (souvent de manière facultative) dans les sous-requêtes précédés de IN quand celles-ci pourraient retourner des doublons. Voir exemple en haut (clause WHERE).

DISTINCT n'a pas de sens dans une requête groupée ou devant une fonction d'agrégation car ces requêtes n'affichent jamais des doublons.

#### ✓ Exercice d'exécution

Une des questions de l'examen demande de produire le résultat d'une requête donnée.

Il ne s'agit donc pas d'expliquer la requête, mais d'écrire le résultat, c'est-à-dire un tableau.

Même si on ne comprend pas la requête, il n'est pas difficile de produire au moins les titres du tableau. Ces titres sont les titres des colonnes et se trouvent dans la clause SELECT.

Exemple : *SELECT fldNom, fldPrénom, SUM(fldValeur) as Montant FROM...* aura comme titres

fldNom	fldPrénom	Montant
--------	-----------	---------