I.U.T. LUMIÈRE

UNIVERSITÉ LUMIÈRE LYON 2

Bases de Données Avancées – Module A

IUT Lumière, License CE-STAT
2006-2007
Pierre Parrend

# Memo : Fonctions SQL

## I. Sélectionner des données

| SQL | Opération | Syntaxe | Exemple |
|---|---|---|---|
| SELECT | Sélectionner toutes les colonnes de la table | SELECT * FROM table; | Afficher tout le contenu de la table stagiaires:<br>SELECT *<br>FROM stagiaires; |
| | **Séléctionner seulement certaines colonnes** | **SELECT colonne1, colonne2, ...<br>FROM table;** | **Afficher le nom et le prenom des stagiaires:<br>SELECT nom, prenom<br>FROM stagiaires;** |
| **DISTINCT** | **Eliminer les doublons** | **SELECT DISTINCT colonne1<br>FROM table;** | **afficher les différents code postal des profs:<br>SELECT DISTINCT code from prof;** |
| **WHERE** | **restreindre le nombre de lignes renvoyées avec une condition** | **SELECT *<br>FROM table<br>WHERE condition1<br>AND condition2 ...;** | **afficher la liste des stagiaires qui habitent Nice:<br>SELECT *<br>FROM stagiaires<br>WHERE ville ='Nice';** |
| **Les opérateurs de comparaison** | **Les conditions sont posées grâce aux opérateurs suivants:<br>= égal<br>!= différent<br>< inférieur<br>> supérieur<br><= inférieur ou égal<br>>= supérieur ou égal** | **SELECT *<br>FROM table<br>WHERE colonne1 opérateur colonne2;<br><br>ou<br><br>SELECT *<br>FROM table<br>WHERE colonne1 opérateur valeur;** | **Afficher les évaluations dont la note est supérieur à 15:<br>SELECT *<br>FROM evaluations<br>WHERE note > 15;** |

| | | | |
|---|---|---|---|
| **BETWEEN** | **Définir un intervalle d'application** | **SELECT * FROM table<br><br>WHERE colonne BETWEEN expression1 AND expression2;** | **Afficher les évalutions dont la note est comprise entre 10 et 15:<br>SELECT *<br>FROM évaluations<br>WHERE note<br>BETWEEN 10 AND 15;** |
| IN | **Chercher une valeur dans une liste de choix.** | **SELECT * FROM table<br><br>WHERE colonne IN ( expression1, expression2, ...);** | **Afficher les évaluations dont la note est comprise entre 10 et 15:<br>SELECT *<br>FROM évaluation<br>WHERE note IN(10, 11, 12, 13, 14, 15);** |
| **LIKE** | **Comparer les chaines de caractères où chaine peut contenir des caractères jokers:<br><br>_ remplace 1 caractère<br><br>% remplace une chaîne de caratère de longueur quelconque.** | **SELECT * from table<br>WHERE colonne LIKE 'chaine' ;<br><br>SELECT * FROM table<br>WHERE colonne LIKE 'ch_ine' ;<br>SELECT * FROM table<br>WHERE colonne LIKE 'c%' ;** | **Afficher la liste des stagiaires qui ont un nom commençant par "Du" et dont le code commence par 06 et finit par 00.<br>SELECT * FROM stagiaires WHERE nom LIKE 'Du%' AND code LIKE '06_00';** |
| **NULL** | **Savoir si le champs a été saisi ou non** | **SELECT * FROM table<br>WHERE colonne1 IS NULL<br>AND colonne2 IS NOT NULL ;** | **Afficher la liste des stagiaires dont le nom est saisi mais pas encore l'adresse:<br>SELECT * FROM stagiaires<br>WHERE nom IS NOT NULL<br>AND adresse IS NULL;** |

## II. Jointures

| SQL | Opération | Syntaxe | Exemple |
|---|---|---|---|
| INNER JOIN | Jointure interne (INNER JOIN): permet de regrouper les données présentes dans deux tables si elles ont un champ commun. | SELECT table1.colonne1, table2.colonne2, ... FROM table1 INNER JOIN table2 USING(colonne) WHERE condition;<br><br><=><br><br>SELECT table1.colonne1, table2.colonne2, ... FROM table1, table2 WHERE table1.colonne = table2.colonne [AND condition];<br><br>Rq: il n'est nécessaire de mettre le nom de la table devant le nom colonne que si les deux tables ont une colonne du même nom.<br><br>Rq: par défaut la jointure est une jointure interne, il suffit donc de mettre JOIN plutôt qu'INNER JOIN.<br><br>Rq: si la colonne qui sert de jointure ne porte pas le même nom dans les deux tables, on peut utiliser la syntaxe suivante:<br><br>SELECT table1.colonne1, table2.colonne2, ... FROM table1 JOIN table2 USING(colonne1 = colonne2) WHERE condition; | Afficher la liste des matières suivies du nom du professeur qui les enseigne:<br><br>SELECT titre, nom FROM matieres JOIN prof USING(id_prof);<br><br><=><br>SELECT titre, nom FROM matieres, prof WHERE matieres.id_prof=prof.id_prof; |
| CROSS JOIN | Jointure croisée: permet de faire le produit cartésien de deux tables. | SELECT table1.colonne1, table2.colonne2, ... FROM table1 CROSS JOIN table2 | Obtenir la liste de tous les couples professeur -matière possibles: SELECT nom, titre FROM prof CROSS |

| | | | |
|---|---|---|---|
| | | WHERE condition; | JOIN matieres; |
| LEFT OUTER JOIN | Jointure avec tous les éléments de la table de gauche même ceux qui n'ont pas de correspondance dans la table de droite. | SELECT table1.colonne1, table2.colonne2, ... FROM table1 LEFT OUTER JOIN table2 USING(colonne) WHERE condition; | Lister tous les professeurs suivi de la matière qu'ils enseignent ou d'u blanc s'ils n'enseigne tpas pour l'instant. SELECT nom, titre FROM prof LEFT OUTER JOIN matieres USING(id_prof); |
| RIGHT OUTER JOIN | Jointure avec tous les éléments de la table de droite même ceux qui n'ont pas de correspondance dans la table de gauche. | SELECT table1.colonne1, table2.colonne2, ... FROM table1 RIGHT OUTER JOIN table2 USING(colonne) WHERE condition; | lister les matieres avec le nom des professeurs qui les enseigne ainsi que les matiers qui sont pour l'instant sans enseignant. SELECT titre, nom FROM prof RIGHT OUTER JOIN matieres USING(id_prof); |
| FULL OUTER JOIN | Union des deux précédentes jointures: permet d'obtenir tous les éléments des deux tables même lorsqu'il n'y a pas de correspondance. | SELECT table1.colonne1, table2.colonne2, ... FROM table1 FULL OUTER JOIN table2 USING(colonne) WHERE condition; | Faire une liste complète des matièers et des professeurs. SELECT titre, nom FROM prof FULL OUTER JOIN matieres USING(id_prof); |
| JOINTURE PAR INEGALITE | Toutes les requêtes sur les jointures précédentes sont des jointures par égalité entre les deux colonnes qui servent de jointure. Les jointures par inégalité se font grâce à un des opérateurs suivants: <, >, <=, >=, <>, !=, BETWEEN, IN, LIKE | SELECT table1.colonne1, table2.colonne2, ... FROM table1 JOIN table2 USING(colonne1 op colonne2) WHERE condition;<br><br><=><br><br>SELECT table1.colonne1, table2.colonne2, ... FROM table1, table2 WHERE table1.colonne op table2.colonne [AND condition]; | Affichez la lite des professeurs dont l'indice de rémunération est supérieure à celle de STICCA Lucie.<br><br>SELECT prof.nom, prof.indice FROM prof JOIN prof p USING (prof.indice > p.indice) AND p.nom='STICCA' AND p.prenom='Lucie'; |
| AUTOJOINTURE | C'est la jointure d'une table sur elle-même, elle est utilisée lorsque l'on souhaite comprarer un ensemble d'enregistrement à un enregistrement de cette table. | SELECT table.colonne1, t.colonne2, ... FROM table, table t WHERE table.colonne op t.colonne [AND condition]; | |

## III.Requêtes Imbriquées

| SQL | Opération | Syntaxe | Exemple |
|---|---|---|---|
| Sous-requête renvoyant une seule valeur | Le résulat de la sous-requête est considéré comme une constante. | SELECT colonne1, colonne2<br>FROM table1<br>WHERE colonne =<br>(SELECT colonne3<br>FROM table2<br><br>WHERE condition); | Afficher la liste des profs qui ont un indice égal à celui de STICCA Lucie:<br>SELECT nom, indice<br>FROM prof<br>WHERE indice =<br>(SELECT indice<br>FROM prof<br>WHERE nom='STICCA'<br>AND prenom='Lucie'; |
| Sous-requête renvoyant un ensemble de valeurs | Permet de comparer la valeur d'une colonne avec l'ensemble de valeurs renvoyé par la sous-requête. Cette comparaison se fait avec les opérateurs IN, ANY ou ALL. | SELECT colonne1, colonne2<br>FROM table1<br>WHERE colonne IN<br>(SELECT colonne3<br>FROM table2<br><br>WHERE condition) | Comptez le nombre d'évaluations effectuées par des stagiaires qui sont des filles:<br>SELECT COUNT(*)<br>FROM evaluations<br>WHERE id_stagiaire IN<br>(SELECT id_stagiaire<br>FROM stagiaires<br>WHERE sexe='F'); |
| Sous-requête renvoyant plusieurs colonnes | Permet de comparer simultanément plusieurs colonnes avec le résultat de la sous-requête. | SELECT colonne1, colonne2<br>FROM table1<br>WHERE colonne1, colonne2 =<br>(SELECT colonne3, colonne4<br>FROM table2<br>WHERE condition) | Chercher sil y a un prof qui est aussi stagiaire:<br>SELECT nom, prenom<br>FROM prof<br>WHERE(nom, prenom) =<br><br>(SELECT nom, prenom<br>FROM stagiaires); |
| Sous-requête existentielle | Affiche le résultat de la première requête si la colonne existe dans la seconde requête. On utilise les mots clefs: EXISTS et NOT EXISTS | SELECT colonne1, colonne2<br>FROM table1<br>WHERE colonne<br>EXISTS<br>(SELECT colonne3<br>FROM table2<br>WHERE condition); | Affichez les noms et prenoms des profs qui ont le meilleur indice:<br>SELECT nom, prenom, indice<br>FROM prof p1<br>WHERE NOT EXISTS<br>( SELECT *<br>FROM prof p2<br>WHERE p2.indice >p1.indice); |

## IV.Opérateurs et Fonctions

| SQL | Opération | Syntaxe | Exemple |
|---|---|---|---|
| Les opérateurs logiques | AND: ET logique où la condition1 et la condition2 doivent être vérifiées<br><br>OR: OU logique où il suffit que l'une des conditions au moins soit vérifiée<br><br>NOT: NON logique qui permet de tester le contraire d'une condition | SELECT * FROM table WHERE condition1 AND condition2;<br><br>SELECT * FROM table WHERE condition1 OR condition2;<br><br>SELECT * FROM nom_table WHERE colonne1 NOT IN (expression1, expression2) ; | Afficher la liste des stagiaires dont le nom est saisi mais pas encore l'adresse ou qui ont un identifiant >150:<br><br>SELECT * FROM stagiaires<br>WHERE (nom IS NOT NULL AND adresse IS NULL)<br>OR id_stagiaire > 150; |
| Fonctions et opérateurs arithmétiques | Effectuer des calculs sur des attributs<br><br>+ addition<br>- soustraction<br>* multiplication<br>/ division<br><br>ABS (val) valeur absolue de val<br>ROUND ( val, nb) arrondi de val à nb chiffres après la virgule<br>TRUNC ( val, nb) renvoie val tronqué à nb chiffres après la virgule | | Ramener les notes sur 10 au lieu de 20 et les arrondir à un chiffre après la virgule:<br>SELECT ROUND(note / 2, 2)<br>FROM evaluations; |
| Fonctions et opérateurs sur les chaînes de caractères | chaîne1 \|\| chaîne2 : concatène les deux chaînes<br><br>CONCAT ( chaine1, chaine2 )<br><br>INITCAP ( chaine) met le premier caratère de chaque mot de chaîne en majuscule et le reste du mot en minuscules. | SELECT colonne1 '=>' colonne2<br>FROM table;<br><br>SELECT CONCAT(colonne1, colonne2)<br>FROM table;<br><br>SELECT INITCAP(nom_colonne)<br>FROM table; | Afficher la liste des noms et prenoms des stagiaires séparés par un .<br>SELECT nom '.' prenom<br>FROM stagiaires;<br><br>Afficher la liste des noms et prenoms des stagiaires collés l'un à l'autre.<br>SELECT CONCAT (nom, prenom)<br>FROM stagiaires;<br><br>Afficher la liste des noms des stagiaires avec la première lettre en majuscule.<br>SELECT INITCAP(nom) |

| | | | |
|---|---|---|---|
| | UPPER (chaine) met tous les caratères de la chaine en lettres majuscules.<br>LOWER (chaine) met tous les caractères de la chaine en lettres minuscules.<br>LENGHT(chaine) renvoie la longueur de chaine.<br>REPLACE (nom_champs, ancienne_chaine, nouvelle_chaine) remplace dans le champs nom_champs, les occurences d'ancienne_chaine par nouvelle_chaine. | SELECT UPPER(nom_colonne) FROM table;<br>SELECT LOWER(nom_colonne) FROM table;<br>SELECT LENGHT(colonne) FROM table;<br>SELECT REPLACE (colonne, 'ancienne_chaîne', 'nouvelle_chaîne') FROM stagiaires; | FROM stagiaires;<br>Afficher les noms des stagiaires en majuscules.<br>SELECT UPPER(nom) FROM stagiaires;<br>Afficher les noms des stagiaires en majuscules.<br>SELECT LOWER(nom) FROM stagiaires;<br>Afficher la longueur des noms des stagiaires.<br>SELECT LENGHT(nom) FROM stagiaires;<br>SELECT REPLACE (nom, 'Dupont', 'Durand') FROM stagiaires; |
| Fonctions et opérateurs sur les dates | =, !=, <, >, <=, >= permettent de comparer deux dates entre elles | SELECT * FROM table WHERE colonne_date < date;<br>SELECT date2 - date1 FROM table; | Lister les stagiaires dont la date de naissance est postérieure au 01/01/1980:<br>SELECT * FROM stagiaires WHERE datenaiss > '1980-01-01';<br>Afficher le nom et l'âge des stagiaires au 1er mars 2003:<br>SELECT nom, ('2003-03-01' - datenaiss)/365 AS "Age" FROM stagiaires; |

## V. Ordonner des données

| SQL | Opération | Syntaxe | Exemple |
|---|---|---|---|
| La clause ORDER BY | permet de classer le résultat d'un select par ordre croissant (ASC) ou décroissant (DESC) | SELECT colonne1, colonne2, ...<br>FROM table<br>ORDER BY colonne1 ; (ordre croissant par défaut)<br><br>SELECT colonne1, colonne2, ...<br>FROM nom_table<br>ORDER BY colonne1, colonne2 DESC ; | Faire une liste des élèves par ordre alphabétique:<br>SELECT nom, prenom<br>FROM stagiaires<br>ORDER BY nom, prenom ASC;<br><br>Faire une liste des matières par coefficients croissants:<br>SELECT titre, coeff<br>FROM matieres<br>ORDER BY coeff; |
| AS | Permet de renommer une en-tête de colonne du résultat.<br><br>Ce mot clef est très pratique pour donner un nom de colonne pour un résultat de calcul. | SELECT colonne1, colonne2 AS nouveau_nom_colonne<br>FROM nom_table; | Afficher les prix horaires nets, sachant qu'il y a 10% de différence entre la valeur brute et la valeur nette:<br>SELECT ROUND((prix_horaire*110/100),2) AS "Prix Nets"<br>FROM remuneration; |

## VI.Grouper les données

| SQL | Opération | Syntaxe | Exemple |
|---|---|---|---|
| GROUP BY | Permet de subdiviser une table en plusieurs groupes. Chaque groupe regroupe l'ensemble des lignes ayant la même valeur pour le champ spécifié | SELECT colonne1, count(colonne2)<br>FROM nom_table<br>GROUP BY nom_champs1; | Comptez le nombre d'étudiant par option:<br>SELECT option, count(*)<br>FROM stagiaires<br>GROUP BY option; |
| HAVING | Permet de préciser les conditions dans lesquelles le GROUP BY doit s'exécuter. Elle fonctionne de la même façon que la clause WHERE. | SELECT colonne1, count(colonne2)<br>FROM table<br>GROUP BY colonne1, colonne2, ...;<br>HAVING condition; | Comptez le nombre d'étudiant pour les options Bureautique et Infographie:<br>SELECT option, count(*)<br>FROM stagiaires<br>GROUP BY option<br>HAVING option in ('Bureautique', 'Infographie'); |
| Les Fonctions de groupe | AVG : calcule la | SELECT | Afficher la moyenne des |

| | | | |
|---|---|---|---|
| | moyenne des valeurs contenues dans champs_numérique<br><br>COUNT: compte le nombre de lignes du résultat de la requête<br><br>MIN : sélectionne la plus petite valeur contenu dans le champs nom_champs<br><br>MAX: sélectionne la plus grande valeur contenu dans le champs nom_champs<br><br>SUM: calcule la somme des valeurs de nom_champs<br><br>STDDEV: calcule l'écart-type des valeurs de nom_champ<br><br>VARIANCE: calcule la variance des valeurs de nom_champs | AVG(champs_numérique)<br>FROM nom_table;<br><br>SELECT COUNT(nom_colonne\|*)<br>FROM nom_table;<br><br>SELECT MIN( nom_champs)<br>FROM nom_table;<br><br>SELECT MAX( nom_champs)<br>FROM nom_table;<br><br>SELECT SUM( nom_champs)<br>FROM nom_table;<br><br>SELECT STDDEV( nom_champs)<br>FROM table;<br><br>SELECT VARIANCE(nom_champs)<br>FROM table; | notes des évaluations<br>SELECT AVG(note)<br>FROM évaluations;<br><br>Compter le nombre de stagiaires:<br>SELECT COUNT(*)<br>FROM stagiaires;<br><br>Afficher le coefficient minimum des matières:<br>SELECT MIN(coefficient) FROM matieres;<br><br>Afficher le coefficient maximum des matières:<br>SELECT MAX(coefficient) FROM matieres;<br><br>Calculer la somme des coefficients des matières:<br>SELECT SUM(coefficient FROM matieres);<br><br>Calculer l'écart-type des notes:<br>SELECT STDDEV(note)<br>FROM evaluations;<br><br>Calculer la variance des notes:<br>SELECT VARIANCE(note)<br>FROM evaluations; |

## VII.Opérateurs ensemblistes

| SQL | Opération | Syntaxe | Exemple |
|---|---|---|---|
| UNION | le résultat contiendra le résulat de requête1 + le résultat de requête | requête1 UNION requête2 | Afficher la liste de toutes les personnes présentes dans l'établissement:<br>SELECT nom, prenom<br>FROM prof<br>UNION<br>SELECT nom, prenom<br>FROM stagiaires; |
| INTERSECT | le résultat contiendra les lignes communes aux deux requêtes | requête1 INTERSECT requête2: | Afficher la liste des profs qui sont aussi stagiaires:<br>SELECT nom, prenom<br>FROM prof<br>INTERSECT<br>SELECT nom, prenom<br>FROM stagiaires; |
| EXCEPT | résultat de requête1 - résultat de requête2 | requete1 EXCEPT requête2 | Afficher la liste des profs qui ne sont pas stagiaires:<br>SELECT nom, prenom<br>FROM prof<br>EXCEPT<br>SELECT nom, prenom<br>FROM stagiaires; |

## VIII.Insérer des Données

| SQL | Opération | Syntaxe | Exemple |
|---|---|---|---|
| INSERT | Ajouter un nouvel enregistrement à une table. | INSERT INTO table<br>( colonne1, colonne2, ...)<br>VALUES<br>( valeur_colonne1, valeur_colonne2, ...);<br><br>Si l'on ne précise aucune colonne, par défaut la liste des colonnes sera la liste complète des champs de la table. Par contre dans ce cas-là, il faut bien donner une valeur pour tous les champs de la table dans le VALUES.<br><br>INSERT INTO table<br>VALUES<br>( valeur_colonne1, | Insérer un stagaire dont on connait seulement l'identifiant, le nom et le prénom:<br>INSERT INTO stagiaires<br>VALUES(100, 'STICCA','Lucie');<br><br>Insérer un stagiaire dont on connait tous les renseignements:<br>INSERT INTO stagiaires<br>VALUES(100, 'STICCA', 'Lucie', '62, Impasse du 8 Mai 1945','06700', 'Saint Laurent du Var', '04 98 08 44 93', 'F', '1976-07-24', 'Programmation'); |

| | | | |
|---|---|---|---|
| | | valeur_colonne2, ...); | |

## IX.Modifier des Données

| SQL | Opération | Syntaxe | Exemple |
|---|---|---|---|
| UPDATE | Modifier le contenu d'un ou plusieurs enregistrements. | UPDATE table<br>SET colonne1 = nouvelle_valeur1,<br>colonne2 = nouvelle_valeur2, ...<br>WHERE condition; | Modifier l'enregistrement de STICCA Lucie qui vient de déménager:<br>UPDATE stagiaires<br>SET adresse=' 32 bd Auguste Gal',<br>code='06300',<br>ville='Nice'<br>WHERE nom='STICCA' AND prenom='Lucie'; |

## X. Supprimer des Données

| SQL | Opération | Syntaxe | Exemple |
|---|---|---|---|
| DELETE | Supprimer une ou plusieurs lignes d'une table. Attention, si aucune condition n'est spécifiée, le contenu entier de la table sera effacé. | DELETE FROM table<br>WHERE condition; | Effacer la ligne de la table stagiaires concernant STICCA Lucie:<br>DELETE FROM stagiaires<br>WHERE nom='STICCA' et prenom='Lucie';<br><br>Effacer tout le contenu de la table stagiaires:<br>DELETE FROM stagiaire |