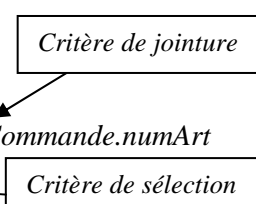

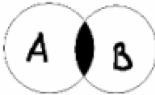



# Aide mémoire SQL

MOTS-CLES	COMMENTAIRES	EXEMPLES
SELECT	Opérateur de <i>projection</i> , permettant de choisir les attributs à afficher Suivi de noms d'attributs ou de * (indiquant tous les attributs)	Affiche le nom et le prénom de tous les élèves SELECT nomEleve, prenomEleve FROM Eleves;
FROM	Suivi d'une ou plusieurs noms de table. La présence de plusieurs tables réalise le <i>produit cartésien</i> de ces tables.	Affiche tous les articles (toutes les colonnes) SELECT * FROM Article;
WHERE	Opérateur de <i>sélection</i> , qui permet de choisir des n-uplets selon certains critères (séparés par AND ou OR)  Cet opérateur permet aussi de réaliser des jointures, le plus souvent par égalité des deux colonnes de sens identique dans les deux tables.	Affiche la désignation de tous les articles de la commande numéro 1258  SELECT Designation FROM Article, Commande WHERE Article.numArt = Commande.numArt AND numCom = 1258; 
DISTINCT	La clause DISTINCT permet d'éliminer les doublons : si dans le résultat plusieurs n-uplets sont identiques, un seul sera conservé.	Afficher toutes les villes où habite au moins un élève  SELECT DISTINCT villeEleve FROM Eleves;
ORDER BY	La clause ORDER BY permet de trier les résultats par ordre croissant (ASC) ou décroissant (DESC).  Le tri peut se faire sur un ou plusieurs attributs.  L'option ASC est prise par défaut pour chacune des expressions citées.	Afficher la liste des employés par salaire décroissant  SELECT nomEmpl, salaire FROM Employe ORDER BY salaire DESC;  Afficher les élèves par ordre alphabétique du nom puis du prénom  SELECT * FROM Eleves ORDER BY nomEleve, prenomEleve;
'	Les cotes sont obligatoires pour entourer les chaînes de caractères et les dates.	Recherche tous les articles de couleur rouges : SELECT numArt FROM Article WHERE couleur = 'ROUGE';  SELECT numEleve FROM Eleves WHERE dateNaissance = '12/10/1980';
IN	IN indique si la valeur est égale à l'une de celles qui se trouvent entre parenthèses (peuvent être les résultats d'une sous-requête). On peut utiliser aussi NOT IN (la valeur n'est égale à aucune de celles de la liste)	Liste des employés occupant la fonction de programmeur, analyste ou développeur  SELECT nomEmpl, numEmpl FROM Employe WHERE fonction IN ('programmeur', 'analyste', 'developpeur');
NULL	Indique une valeur non définie (non renseignée). <i>Attention</i> , 0 n'est pas une valeur NULL L'opérateur de comparaison est IS ou IS NOT	Afficher le nom des professeurs dont on ne connaît pas le prénom SELECT nomProf FROM Profs WHERE prenomProf IS NULL;
LIKE	Permet de comparer une valeur avec une chaîne non complète. % désigne plusieurs caractères _ désigne un seul caractère quelconque (ou aucun)	Recherche tous les noms commençant par DE :  SELECT nomCom, prenomCom, rueCom FROM Commerçants WHERE nomCom LIKE 'DE%';

UPPER LOWER	<p>UPPER transforme tous les caractères d'un champ en majuscule. Attention, il faut marquer le nom voulu en majuscule.</p> <p>LOWER les transforme en minuscule. Attention, il faut marquer le nom voulu en minuscule.</p>	<pre>SELECT numArt FROM Article WHERE UPPER (couleur) = 'BLEU';</pre> <pre>SELECT NumArt FROM Article WHERE LOWER (couleur) = 'blanc';</pre>
BETWEEN	Indique si une valeur est comprise entre deux valeurs. Les extrêmes sont inclus dans l'intervalle.	<p>Articles qui coûtent entre 10 et 20 euros inclus.</p> <pre>Select numArt, nomArt FROM Article WHERE prix BETWEEN 10 AND 20;</pre>
AS	Permet de renommer un attribut (seulement pour l'affichage de la réponse à la requête)	<p>Affiche la référence et la marge de tous les produits</p> <pre>SELECT refProd, (prixVente – prixAchat) AS marge FROM Produit;</pre>
Opérateurs de calcul (+ - * ...)	Permet d'afficher le résultat d'un calcul à partir d'un ou plusieurs attributs.	<p>Affiche la référence et la marge de tous les produits</p> <pre>SELECT refProd, (prixVente – prixAchat) FROM Produit;</pre>
EXISTS	<p>Permet de tester l'existence de données dans une sous-requête.</p> <p>Si la sous-requête renvoie au moins une ligne, même remplie de marqueurs NULL, le prédicat est vrai. Dans le cas contraire le prédicat à la valeur faux.</p> <p>Le prédicat EXISTS peut être combiné avec l'opérateur de négation NOT (NOT EXISTS) pour tester l'absence de données dans la sous-requête.</p>	<pre>SELECT nomAttr1 FROM table1 WHERE EXISTS (SELECT * FROM table2               WHERE condition);</pre> <pre>SELECT nomAttr1 FROM table1 WHERE NOT EXISTS (SELECT * FROM table2                   WHERE condition);</pre>
OPERATEURS ENSEMBLISTES		
UNION	<p>Il est possible de faire l'<i>union</i> des résultats de deux requêtes. L'union élimine les doublons.</p> <p>On a alors <math>A \cup B</math> (ou <math>A + B - A \cap B</math>).</p> 	<p>La liste des villes où habitent des élèves et des professeurs :</p> <pre>SELECT villeEleve FROM Eleves UNION SELECT villeProf FROM Profs;</pre>
INTERSECT	<p>Il est possible de faire l'<i>intersection</i> des résultats de deux requêtes.</p> <p>On a alors : <math>A \cap B</math>.</p> 	<p>La liste des villes où habitent à la fois des élèves et des professeurs :</p> <pre>SELECT DISTINCT VilleEleve FROM Eleves INTERSECT SELECT DISTINCT VilleProf FROM Profs;</pre>
MINUS	<p>Il est possible de faire la <i>différence</i> des résultats de deux requêtes.</p> <p>On a alors : <math>A - B</math>.</p> 	<p>La liste des villes où habitent seulement des élèves et pas de professeurs :</p> <pre>SELECT DISTINCT VilleEleve FROM Eleves MINUS SELECT DISTINCT VilleProf FROM Profs;</pre>

## OPERATEURS D'AGREGATION

COUNT(*)	Permet de compter le nombre de lignes (n-uplets) résultats.	Compte le nombre d'élève habitant Cachan et appelle le résultat NbCachanais. SELECT COUNT (*) AS NbCachanais
COUNT(attribut)	Permet de compter le nombre de lignes où l'attribut indiqué a une valeur non nulle.	FROM Eleves WHERE villeEleve = 'CACHAN';
SUM(attribut)	Permet d'additionner les valeurs d'une colonne numérique pour les n-uplets sélectionnés. (A ne pas confondre avec COUNT !)	Calcul le cumul (la somme) de tous les opérations de débit du compte 1259 le 09/01/04. SELECT SUM(montantOperation) FROM Operations WHERE compte = '1259' AND date = '09/01/04';
AVG(attribut)	Permet d'afficher la moyenne des valeurs d'une colonne numérique pour les lignes sélectionnées.	Afficher le salaire moyen des analystes SELECT AVG(salaire) FROM Employe WHERE fonction = 'analyste';
MAX(attribut) MIN(attribut)	Permet d'obtenir la valeur maximale (ou minimale) d'un attribut pour un ensemble de n-uplets sélectionnés.	Affiche le salaire de la secrétaire la mieux payée : SELECT MAX(salaire) FROM Employe WHERE fonction = 'secretaire';

## GROUP BY, HAVING

GROUP BY	Permet de subdiviser la table en groupes, chaque groupe étant l'ensemble des n-uplets ayant une valeur commune pour les expressions spécifiées. Les attributs de la clause SELECT doivent alors être des opérateurs d'agrégation ou des expressions figurant dans la clause GROUP BY.	GROUP BY <i>expr1, expr2, ...</i>  Affiche le nombre de professeurs par bureau SELECT bureau, COUNT(nomProf) FROM Profs GROUP BY bureau;
HAVING	Sert à préciser quels groupes doivent être sélectionnés. Cette clause se place après GROUP BY et le prédicat ne peut porter que sur des opérateurs d'agrégation ou des expressions figurant dans la clause GROUP BY.	HAVING <i>predicat</i>  Affiche les bureaux qui accueillent plus de 2 professeurs SELECT bureau FROM Profs GROUP BY bureau HAVING COUNT (*) >2;

## ANY, ALL

<p>Le critère de recherche employé dans une clause WHERE (l'expression à droite d'un opérateur de comparaison) peut être le résultat d'un SELECT.</p> <p>Dans le cas des opérateurs classiques, la sous-interrogation ne doit ramener qu'un n-uplet et un attribut. Elle peut ramener plusieurs n-uplets à la suite de l'opérateur [NOT] IN, ou à la suite des opérateurs classiques moyennant l'ajout d'un mot clé (ANY ou ALL).</p>		
ANY	La comparaison est vraie si elle est vraie pour au moins un élément de l'ensemble (donc fausse si l'ensemble des n-uplets est vide).	... WHERE <i>expr1</i> = / != / < / > / <= / >= <i>expr2</i> ANY (SELECT...);
ALL	La comparaison est vraie si elle est vraie pour tous les éléments de l'ensemble (donc vraie si l'ensemble des n-uplets est vide).	... WHERE <i>expr1</i> = / != / < / > / <= / >= <i>expr2</i> ALL (SELECT...);

		Affiche le bureau qui accueille le plus de professeurs SELECT bureau FROM Profs GROUP BY bureau HAVING COUNT (*) >= ALL(SELECT COUNT (*) FROM Profs GROUP BY bureau);
JOINTURES		
Jointure	C'un produit cartésien entre deux tables suivi d'une sélection selon le critère de jointure (n'importe quelle opération de comparaison entre un attribut de table1 avec un attribut de table2). Généralement, le "=" (équijointure).	SELECT attr1, attr2, ... FROM table1, table2 WHERE table1.attr1 = table2.attr2;
JOINTURES INTERNES		
Jointure interne	Il s'agit d'une jointure naturelle mettant deux fois en jeu la même table. Ceci permet de rassembler venant d'un n-uplet d'une table avec les informations venant d'un autre n-uplet. A noter qu'il est obligatoire de spécifier deux fois la table source dans la clause FROM, en attribuant un alias à l'une des occurrences.	SELECT table.expr1, alias.expr2 FROM table, table alias WHERE table.champ1 = alias.champ2;
JOINTURES EXTERNES		
LEFT / RIGHT / FULL OUTER JOIN	Lorsqu'on effectue une jointure naturelle sur deux tables, il est possible qu'un n-uplet d'une table n'ait pas de correspondant dans l'autre. Dans ce cas, le n-uplet en question n'est pas affiché. La jointure externe permet de résoudre ce problème par l'ajout de n-uplets fictifs, qui réalisent la correspondance avec les n-uplets de l'autre table n'ayant pas de correspondant réel (LEFT OUTER JOIN correspond à l'ajout de n-uplets fictifs dans table1).	SELECT expr1, expr2, ... FROM table1 LEFT OUTER JOIN table2 ON table1.champ1 = table2.champ2;  SELECT expr1, expr2, ... FROM table1 RIGHT OUTER JOIN table2 ON table1.champ1 = table2.champ2;
CREATION, MISE A JOUR, DESTRUCTION DE TABLES ET CONTRAINTES		
CREATE TABLE	<i>table</i> est le nom donné à la nouvelle table. <i>col1, col2,...</i> sont les noms des attributs. <i>type1, type2,...</i> sont les types des données contenues dans les attributs.	CREATE TABLE table (col1 type1, col2 type2...);
ALTER TABLE	Pour modifier la structure d'une base de données existante.	Ajoute les colonnes spécifiées à une table existante. ALTER TABLE table ADD (col1 type1, col2 type2, ...); Modifie la définition des colonnes spécifiées. ALTER TABLE table MODIFY (col1 type1, col2 type2...); Supprime la colonne col. ALTER TABLE table DROP col;

DROP TABLE	Supprime la définition d'une table et par conséquent l'ensemble des n-uplets qu'elle contient.	DROP TABLE table;
CONSTRAINT	Définit et nomme une contrainte sur un ou plusieurs attributs.	CONSTRAINT nomContrainte contrainte
PRIMARY KEY	Indique la clé primaire de la table. Aucun des attributs de cette clé ne doit avoir une valeur NULL.	PRIMARY KEY (col1, col2,...)
UNIQUE	Interdit qu'un attribut (ou la concaténation de plusieurs attributs) contienne deux valeurs identiques.	UNIQUE (col1, col2,...)
FOREIGN KEY	Indique que la concaténation de col1, col2,... est une clé étrangère faisant référence à la concaténation de col'1, col'2,... de table.  Si col'1,col'2,... sont omises, la clé primaire de table est prise par défaut.	FOREIGN KEY (col1, col2, ...) REFERENCES table [(col'1,col'2,...)]
CHECK	Donne une condition devant être vérifiée par un ou plusieurs attributs.	CHECK (condition)
PRINCIPAUX TYPES DE DONNEES		
CHAR(taille)	Chaînes de caractères de longueur fixe	CHAR(3)
VARCHAR(taillemax)	Chaînes de caractères de longueur variable	VARCHAR(10)
SMALLINT	Entier court (sur 16 bits : de -32768 à 32757)	
INTEGER	Entier long (sur 32 bits : de -2 147 483 648 à 2 147 483 647)	
NUMBER(n,[d])	Nombres décimaux qui ont <i>n</i> chiffres dont <i>d</i> chiffres après la virgule	Nombre à 5 chiffres avant la virgule et 2 après la virgule : NUMBER(7,2)
REAL	Nombre à virgule flottante (faible précision)	
FLOAT DOUBLE PRECISION	Nombre à virgule flottante (grande précision)	
DATE	Date (jour, mois, année)	10/01/2002
TIME	Heures, minutes et secondes (les secondes peuvent comporter un certain nombre de décimales)	10:12:42.85
TIMESTAMP	Moment précis : date avec heures, minutes et secondes (6 chiffres après la virgule, c'est-à-dire en microsecondes)	
CREATION, MISE A JOUR, DESTRUCTION DE N-UPLETS		
INSERT	Insère un nouvel n-uplet dans table. Si (col1, col2,...) est omis, l'ordre utilisé par défaut est celui spécifié lors de la création de la table. A l'inverse, si cette liste est donnée, les attributs n'y figurant pas auront la valeur NULL.	INSERT INTO table [(col1, col2,...)] VALUES (val1, val2,...)
UPDATE	Modifie les n-uplets de <i>table</i> qui vérifient le prédicat. Si <i>predicat</i> est omis, tous les n-uplets sont mis à jour.	UPDATE table SET col1 = expr1, col2 = expr2,... WHERE predicat
DELETE	Supprime les n-uplets de <i>table</i> qui vérifient le prédicat. Si <i>predicat</i> est omis, tous les n-uplets sont supprimés.	DELETE FROM table WHERE predicat
VUES		

CREATE VIEW	Permet de créer une vue en spécifiant une requête.  Remarque : il est impossible de modifier le schéma d'une vue. Il faut alors la supprimer et la recréer.	CREATE VIEW nomVue AS uneRequeteSelect;  CREATE VIEW BureauxProfs AS SELECT bureau, COUNT(nomProf) FROM Profs GROUP BY bureau;
DROP VIEW	Supprime une vue.	DROP VIEW nomVue;
GESTION DE PRIVILEGES		
GRANT	Permet au propriétaire de <i>table</i> de donner à <i>utilisateur</i> des droits d'accès sur celle-ci (droit de lecture, de modification, de suppression). Si l'option WITH GRANT OPTION est spécifiée, <i>utilisateur</i> pourra à son tour transmettre ces droits.	GRANT privilege ON table TO utilisateur [WITH GRANT OPTION]
REVOKE	Permet de reprendre un privilège à un utilisateur.	REVOKE privilege ON table FROM utilisateur