

# Aide-mémoire SQL

## 1 Structure des requêtes

La structure générale d'une requête SQL est (seuls SELECT et FROM sont requis) :

SELECT *attributs, expressions...*

FROM *table ou jointure*

WHERE *condition(s) sur les lignes*

GROUP BY *attributs, expressions...*

HAVING *condition(s) sur les groupes*

ORDER BY *attribut, expression... [DESC]*

LIMIT *nombre OFFSET nombre*

Note : HAVING n'a de sens que s'il y a création de groupes avec GROUP BY

## 2 Fonctions d'agrégation

SUM(*attribut*) : somme des valeurs

MIN(*attribut*) : plus petite valeur

MAX(*attribut*) : plus grande valeur

COUNT(\*) : nombre de lignes

COUNT(DISTINCT \*) :

nombre de lignes différentes

COUNT(DISTINCT *attribut*) :

nombre de valeurs différentes de l'attribut

En l'absence de GROUP BY, les calculs sont effectués sur toute la table (après filtre éventuel avec WHERE).

En présence d'un GROUP BY, les calculs sont effectués sur chaque groupe.

## 3 Projection

On peut projeter sur n'importe quel attribut disponible dans la (ou les) table(s) sélectionnées, mais aussi sur des expressions :

SELECT *attr 1 + attr 2* FROM ...

On peut obtenir tous les attributs disponibles avec \* :

SELECT \* FROM ...

Pour supprimer les doublons dans les résultats :

SELECT DISTINCT *attributs* FROM ...

## 4 Restriction du résultat

Pour ne conserver que les 5 premières « lignes » du résultat de la requête :

LIMIT 5

Pour obtenir les septième, huitième et neuvième « lignes » (trois lignes, après avoir sauté les six premières) :

LIMIT 3 OFFSET 6

La restriction du nombre de lignes retournées n'a généralement de sens qu'après un tri des résultats avec ORDER BY.

Pour obtenir les derniers enregistrements, on triera les résultats dans le sens inverse (ORDER BY ... DESC) et on prendra les premiers.

## 5 Opérateurs booléens

On dispose des opérateurs de comparaison et booléens usuels :

< <= > >= = <> AND OR NOT

Pour vérifier si une valeur (attribut, expression) fait partie d'une liste de valeurs :

*attribut* IN *liste de valeurs*

Pour vérifier si une valeur (attribut, expression) est définie :

*attribut* IS NOT NULL

Pour tester si une chaîne de caractères ressemble à un modèle (% remplace toute séquence de caractères) :

*attribut* LIKE *modèle de chaîne*

## 6 Sous-requêtes

Le résultat de toute requête placée entre parenthèses peut être utilisée dans une autre requête.

Les requêtes retournant une unique valeur peuvent être utilisées dans les expressions et les comparaisons.

Les requêtes retournant une « liste » (projection sur un unique attribut ou une unique expression par exemple) peuvent être utilisées avec IN.

Les requêtes retournant une table peuvent être utilisées comme tables avec FROM.

## 7 Joindre des tables

Produit cartésien :

*table1, table2*

Jointure (interne) :

*table1 JOIN table2 ON expr. booléenne*

Les jointures internes correspondent à un produit cartésien des deux tables, suivi d'une sélection des lignes résultantes de ce produit pour lesquelles l'expression booléenne est vraie.

En cas d'ambiguité entre des attributs de deux tables différentes, on précise le nom de la table avant l'attribut en écrivant *table.attribut*

Pour spécifier un alias pour une table :

*table1 AS alias JOIN ...*

Pour effectuer des jointures multiples :

*table1 JOIN table2 ON expr. booléenne*

JOIN *table3 ON expr. booléenne...*

## 8 Combiner les requêtes

Union (sans doublon) de requêtes :

*requête 1 UNION requête 2*

Intersection de requêtes :

*requête 1 INTERSECT requête 2*

Différence (non symétrique) :

*requête 1 MINUS requête 2*