Memento SQL

Tout est relation : on traite **le quoi, pas le comment**.

A la fin de chaque requête SQL, on écrit un point-virgule ;

**Création de tables**

CREATE TABLE nomTable(

nomAttribut1 typeAttribut1 PRIMARY KEY,

nomAttribut2 typeAttribut2 REFERENCES Table(nomAttribut2),

nomAttribut3 typeAttribut3 … ) ;

⋆ typeAttribut : INT, REAL, TXT, VARCHAR(17) (17 caractères max)

⋆ Il peut y avoir plusieurs clés étrangères et les clés étrangères peuvent être déclarées à part : FOREIGN PRIMARY KEY Table(attr)

⋆ La clé primaire peut être non unitaire, dans ce cas, on est obligé de la déclarer à part en rajoutant : PRIMARY KEY (nomAttribut1, nomAttribut2)

⋆ note REAL CHECK(0<=note<=20) ok VARCHAR(1) CHECK(ok IN ('O','o')

⋆ Pour renommer : ALTER TABLE Table1 RENAME TO Table2

⋆ Pour effacer : DROP TABLE IF EXISTS nomTable

⋆ Pour activer la vérification des clés étrangères : PRAGMA FOREIGN\_KEY=1

⋆ INSERT INTO Table VALUES (attribut1, attribut2, attribut3…) ;

INSERT INTO Table SELECT… ;

**Fonctions d'agrégation :**

MIN(a) MAX(a) AVG(a) SUM(a) COUNT(a) LENGTH(a) COALESCE(a,b,0)

*Coalesce renvoie la valeur du premier attribut non NULL donc, ici, au pire 0 !*

**NULL :** Attention, les opérateurs de comparaison renvoient NULL et non pas vrai ou faux, quand l'une des entrées est NULL.

**Ordre requête : SELECT/FROM/WHERE/GROUP BY/HAVING/ ORDER BY y**

**Projection : SELECT attributs *(sélection de colonnes…)***

avec fct d’agreg et/ou AS et/ou DISTINCT et/ou \* et/ou FROM

SELECT nom, 2\*AVG(note) , MAX(moyenne) *séparés par virgule*

SELECT AVG(note ) AS moy *renommage*

SELECT DISTINCT nom *enlève doublon*

SELECT \* FROM classe *tous les attributs de classe*

**Sélection : WHERE conditions *(sélection de lignes…)***

AND, OR, NOT, IN, BETWEEN, <, <=, >, >=, =, <>, LIKE, \_ et %

SELECT nom, note

FROM classe

WHERE (note > 15) OR (note IN (10,11)) AND (nom LIKE 'A%')

**Jointure : FROM table1 JOIN table2 ON condition / USING attribut**

**JOIN = INNER JOIN, NATURAL JOIN si attributs communs**

*Pour lever toute ambigüité, préfixer par le nom de la table (auto-jointures).*

*Beaucoup de syntaxes possibles…*

FROM T1 JOIN T2 USING(attr) FROM T1 JOIN T2 JOIN T3

FROM T1 NATURAL JOIN T3 ON T1.a1 = T2.a2 AND T3.a3 = T2.b2

FROM T1 JOIN T2 USING(a) JOIN T3 ON a=b

Pour avoir TOUS les éléments de T1, même s’ils n’ont pas de correspondance avec T2 (NULL) : FROM T1 LEFT JOIN T2 USING(attr) *jointure externe*

**Classement : ORDER BY attributs … (ASC ou DESC)**

ORDER BY note *croissant par défaut* ORDER BY note DESC *décroissant*

ORDER BY note DESC, nom ASC *tri selon note decr. puis selon nom alphab.*

**Groupement :** GROUP BY attr + HAVING condition ***(sélection de tuples…)***

*Attention, les attributs doivent être communs à tout le groupe !*

SELECT anonymat FROM eleve NATURAL JOIN DS

GROUP BY anonymat HAVING SUM(note1+note2) BETWEEN 15.0 AND 20.0

**Requêtes imbriquées : SELECT + FROM + WHERE + … SELECT…**

*Attention parenthèses obligatoires ! L'indentation permet d'y voir clair.*

SELECT nom FROM eleve

WHERE note = (

SELECT MAX(note) FROM eleve)

**Opérations ensemblistes : Combinent résultats de plusieurs SELECT**

UNION ou INTERSECT ou EXCEPT ou UNION ALL (sans doublon)

SELECT nom FROM classe1 *Les SELECT ont le même nombre*

UNION / INTERSECT / EXCEPT *d’attributs, de types compatibles*

SELECT nom FROM classe2 *toutes orthographes de dupon\**

WHERE LOWER(nom) LIKE 'dupon\_' *insensible à la casse*

**Calcul :**

SELECT *Le \*1. permet de forcer le*

(SELECT SUM(note\*coeff) FROM table ) *type float pour* effectuer

/ *effectuer une division*

(SELECT SUM(coeff\*1.) FROM table) *décimale !*