Memento SQL

> ++Crédits :++ d'après le *memento de Brigitte Mougeot (https://cloud-lyon.beta.education.fr/s/4qwaBm2FqT2yR69#pdfviewer)* :+1: et un document de Franz Ridde

Grands principes

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

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

:::success

Syntaxe générale d'une requête SQL (toujours dans cet ordre)

```
SELECT attributs, agrégats (AS...)
FROM table  -- commentaire :avec jointure éventuelle ... JOIN ... ON ... = ...
WHERE conditions
GROUP BY attributs
HAVING conditions
ORDER BY attributs (... ASC ou DESC) ;
```

:warning: La clause `WHERE` opère une sélection sur la table.
La clause `HAVING` opère une sélection sur les groupes. Elle n'a pas de sens sans `GROUP BY`[1].
:::

:::info

Fonctions d'agrégation

- `MIN(a)`
- `MAX(a)`
- `AVG(a)`
- `SUM(a)`
- `COUNT(a)`
- `COUNT(*)` *avec wildcard *, compte le nombre d'enregistrements tous les attributs*
  :::

:::warning

Projection

> ++**Définition :**++
> Si on se représente la **table/relation** comme un tableau, la **projection** consiste à ne conserver que certaines **colonnes/attributs**.

- Syntaxe de base

```
SELECT attribut1, attibut2, ... FROM table ;
```

:warning: SELECT correspond à une opération de **projection** et non pas de **sélection**.

- Si on veut projeter sur tous les attributs de la table, on peut utiliser le caractère joker *

```
SELECT * FROM table ;
```

- On peut utiliser des opérations arithmétiques ou des fonctions d'agrégation pour opérer sur des **colonnes/attributs** :

```
SELECT attribut1, 2 * AVG(attibut2) , MAX(attibut3) FROM table ;
```

- On peut renommer une **colonne/attribut** avec AS :

```
SELECT AVG(note) AS moyenne FROM table ;
```

- On peut dédoublonner les **lignes/enregistrements** avec DISTINCT :

```
SELECT DISTINCT nom FROM table ;
```

:::

:::warning

Sélection

> ++**Définition :**++
> Si on se représente la **table/relation** comme un tableau, la **sélection** consiste à ne conserver que les lignes qui vérifient une certaine condition.
> La **condition** après un WHERE doit avoir une valeur booléenne, elle peut résulter d'une opération de comparaison ou d'une opération booléenne.

- Syntaxe de base :

```
SELECT attribut1, attibut2, ... FROM table WHERE condition ;
```

- Opérateurs de comparaison :

Opération	Opérateur
égalité	=
différence	<>
inférieur (ou égal)	< ou <=
supérieur (ou égal)	> ou >=

comparaison de motif 'markdown' LIKE 'm_r%n'
dans/parmi 9 IN (6, 9)
entre binf et bsup BETWEEN binf AND bsup

- opérateurs booléens :

Opération	Opérateur
et	AND
ou	OR
non/négation	NOT

- Exemple :

```
SELECT nom, note

FROM classe

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

:::

:::warning

Jointure

> ++**Principe de la jointure :**++
> On combine deux **tables/relations** d'une même base en juxtaposant les **lignes/enregistrements** qui correspondent pour un ou plusieurs attributs communs ou de même domaine.

- Syntaxe d'une jointure entre deux tables :

```
SELECT *              -- ou choix des attibuts de projection
FROM
table1 JOIN table2    --  ou table1 INNER JOIN table2
ON table1.attibut1 = table2.attibut2  ;
```

:warning: *Pour lever toute ambigüité, en particulier dans le cas des auto-jointures, on peut préfixer par le nom de la table ou faire un renommage avec AS*

- **Jointure naturelle** si certains attibuts sont communs (mêmes noms et domaines) :

```
SELECT ...              -- choix des attibuts de projection
FROM table1 NATURAL JOIN table2  ;
```

- On peut enchaîner les jointures :

```
SELECT ...
FROM T1 JOIN T2 JOIN T3
ON T1.a1 = T2.a2 AND T3.a3 = T2.b2 ;
```

Plusieurs syntaxes sont possibles :

```
SELECT ...
FROM T1 JOIN T2
ON T1.a1 = T2.a2
JOIN T3
ON T3.a3 = T2.b2 ;
```

:::

:::info

Classement

> ++**Principe :**++
> On peut classer les **lignes/enregistrements** de la table obtenue par la requête en plaçant à la fin (et toujours à la fin) de la requête le mot clef ASC suivi des attributs dans l'ordre desquels la comparaison lexicographique sera effectuée. On peut préciser le sens pour chaque attibut avec ASC pour croissant et DESC pour décroissant.

- Syntaxe

```
SELECT
....
ORDER BY attribut1, attibut2 (... ASC ou DESC) ;
```

- Exemples :

```
SELECT
....
ORDER BY attribut1, attibut2 (ASC ou DES) ;

SELECT
....
ORDER BY note ;   -- croissant par défaut

SELECT
....
ORDER BY note DESC ; -- décroissant


SELECT
....
ORDER BY note DESC, nom ASC ; -- tri selon note decr. puis selon nom  croissant
```

:::

:::warning

Groupements / Agrégats

> ++**Principe :**++
> On regroupe certaines **lignes/enregistrements** qui ont un ou plusieurs attibuts communs. Les groupes obtenus ne peuvent avoir comme attributs que ceux communs à tout le groupe ou des **agrégats**, résultats de fonctions d'agrégation calculées sur toutes les **lignes/enregistrements** constituant le groupe.
> On peut faire une sélection sur les groupes avec la clause `GROUP BY`[1]

- Syntaxe :

```
SELECT attribut1, attribut2, agrégat(s)
FROM .....
WHERE ....
GROUP BY attribut1, attribut2 ;
```

:warning: La clause `WHERE` opère une sélection sur la table complète, elle doit donc être placée **avant** `GROUP BY`.

:warning: La clause `HAVING` opère une sélection sur la table après regroupement, elle doit donc être placée **après** `GROUP BY`.

- Exemple :

```
SELECT anonymat

FROM eleve NATURAL JOIN DS

GROUP BY anonymat

HAVING SUM(note1+note2) BETWEEN 15.0 AND 20.0 ;
```

:::

:::info
Parfois on a besoin de requêtes imbriquées, par exemple pour déterminer l'élève qui a eu la note maximale d'un devoir :

```
SELECT nom
FROM eleve
WHERE note = (
    SELECT MAX(note)
    FROM eleve
    )
;
```

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

:::warning

Opérations ensemblistes

> ++**Principe :**++
> Elles permettent de combiner les résultats de plusieurs tables de schémas compatibles avec les mots clefs `UNION` pour la réunion (ou `UNION ALL` pour la réunion sans doublon), `INTERSECT` pour l'intersection et `EXCEPT` (ou `MINUS`) pour la différence.

- Un exemple :

```
-- Les deux SELECT doivent avoir le même nombre d'attributs de même domaine
SELECT nom FROM classe1
UNION / INTERSECT / EXCEPT
SELECT nom FROM classe2
WHERE LOWER(nom) LIKE 'dupon_'
-- 'dupont' et 'dupond' acceptés mais pas 'dupontt' avec  le joker  '_' pour un caractère quelconque
```

:warning:*Ne pas mettre de parenthèses autour des tables sur lesquelles on opère avec `UNION`, `UNION ALL`, `EXCEPT` ou `INTERSECT`*
:::

:::info

Calcul

> ++**Astuce :**++ :bulb:
> Par défaut la division de deux entiers retourne le quotient entier de leur division euclidienne. Pour effectuer la division décimale, on peut multiplier le numérateur ou le dénominateur par 1.0 pour forcer sa conversion en flottant.

- Un exemple de calcul de moyenne pondérée :

```
SELECT
(SELECT SUM(note * coeff) FROM table )
/
(SELECT SUM(coeff * 1.0) FROM table)
```

:::

:::danger
:fire: Attention à l'évaluation des attributs en présence de `NULL` !

Les opérateurs de comparaison habituels renvoient `NULL` et non pas vrai ou faux, quand l'une des entrées est `NULL`.
:::

---

1. hors programme de terminale NSI ↩ ↩