Chapitre 5

SQL

Data Query Language

Requêtes simples

Plan

- Sélectionner toute la table
- Sélectionner des colonnes
- Expression arithmétique
- Alias de nom de colonne
- Éviter les duplicatas
- Sélectionner des lignes
- BETWEEN … AND …
- IN (…)
- LIKE
- IS NULL
- NOT
- Fonctions
- Trier les lignes à l'affichage
- Résumé des requêtes simples

Sélectionner toute la table

Clause **SELECT** ⇨ identifie les colonnes à afficher

Clause **FROM** ⇨ identifie la table sur laquelle porte la requête

Sélection de toutes les colonnes et de toutes les lignes

```
SELECT *
FROM <nom_table> ;
```

Exemple

```
select * from section ;
```

code	libelle
DA	Développement d'applications
IA	Intelligence artificielle
IR	Sécurité des systèmes
TI	Technologies de l'informatique

Sélectionner des colonnes

⇨ Préciser les noms des colonnes dans la clause SELECT

```
SELECT <nom_colonne>, …
FROM <nom_table> ;
```

Exemple

```
select prenom, nom, email
from etudiant;
```

prenom	nom	email
François	Dupuis	f.dupuis@gmail.com
Caroline	Grant	car.grant@hotmail.be
Marie	Leblanc	mblanc@gmail.com
Luc	Petit	NULL

Expression arithmétique

Opérateurs arithmétiques qui peuvent être utilisés sur des colonnes numériques ou de type date : **+ - * /**

```
SELECT <nom_colonne> | <expression_arithmétique> , …
FROM <nom_table> ;
```

Dans la syntaxe des instructions SQL, la barre verticale | exprime un choix : tu peux utiliser soit l'expression à droite soit l'expression à gauche de la barre |. Attention, tu ne dois pas écrire | dans ton instruction SQL !

Dans ce cas-ci, tu peux choisir via la clause select de sélectionner aussi bien des colonnes que des expressions arithmétiques.

Expression arithmétique

Prix TVA comprise

Exemple

```
select libelle, prix * 1.21
from produit;
```

libelle	(Aucun nom de colonne)
Banane	1.5730
Carotte	1.0890
Crème fraîche	5.5176
Gâteau au chocolat	28.9190
Jambon de Parme	10.7690

Notons qu'il n'y aucun nom de colonne affiché pour les colonnes correspondant aux expressions arithmétiques.

Mais on peut choisir un nom de colonne lors de l'affichage via un alias. Notons que le nom de la colonne n'est pas modifié dans la base de données si on précise un alias.

Alias de nom de colonne

SELECT *<nom_colonne>* | *<expression_arithmétique>* [alias] , …
FROM *<nom_table>* ;

Exemple

```
select libelle, prix * 1.21 "Prix TVAc", poids / 1000 "Poids (en kg)"
from produit;
```

libelle	Prix TVAc	Poids (en kg)
Banane	1.5730	1.000000
Carotte	1.0890	0.500000
Crème fraîche	5.5176	NULL
Gâteau au chocolat	28.9190	NULL
Jambon de Parme	10.7690	0.100000

La colonne poids est facultative.
Une expression arithmétique contenant une valeur NULL est évaluée à NULL.

Éviter les duplicatas

SELECT [DISTINCT] ***<nom_colonne>*** **|** ***<expression_arithmétique>*** **[alias] , …**
FROM ***<nom_table>*** **;**

Exemples

```
select *
from localite;
```

numero	code_postal	nom
1	5000	Namur
2	1000	Bruxelles
3	4000	Liège
4	6000	Charleroi
5	5020	Daussoulx
6	5020	Malonne
7	5020	Champion

```
select code_postal
from localite;
```

code_postal
5000
1000
4000
6000
5020
5020
5020

```
select distinct code_postal
from localite;
```

code_postal
1000
4000
5000
5020
6000

Sélectionner des lignes

⇨ Spécifier les conditions que les lignes doivent respecter pour être sélectionnées

SELECT [DISTINCT] ***<nom_colonne>*** **|** ***<expression_arithmétique>*** **[alias] , …**
FROM ***<nom_table>***
[WHERE ***<condition(s)>*** **]**

Exemples

```
select prenom, nom
from etudiant
where matricule = 'BA23';
```

prenom	nom
François	Dupuis

La colonne matricule est l'identifiant ⇨ Une seule lignée sélectionnée !

```
select nom
from localite
where code_postal = 5020;
```

nom
Champion
Daussoulx
Malonne

Sélectionner des lignes

Opérateurs de comparaison

Opérateur	Signification
=	Égal
>	Plus grand
>=	Plus grand ou égal
<	Plus petit
<=	Plus petit ou égal
<>	Différent
BETWEEN ... AND …	Entre deux valeurs (bornes incluses)
IN (…)	Égal à au moins un élément de la liste
LIKE	Correspond à un modèle (chaîne de caractères)
IS NULL	Est une valeur inconnue

+ Combinaisons des conditions : AND et OR

Sélectionner des lignes

Exemples

```
select matricule, prenom, nom
from etudiant
where date_naissance < '02/10/2000';
```

```
select *
from etudiant
where nom <> 'Dupond';
```

```
select reference, libelle
from produit
where prix < 15.5 and nb_calories > 10;
```

```
select prenom, nom
from etudiant
where matricule = 'BA23' or matricule = 'KA3';
```

BETWEEN … AND …

- Permet d'exprimer des conditions basées sur des intervalles de valeurs
- Sur des colonnes de type nombre, chaîne de caractères et date

Exemples

```
select libelle
from produit
where prix between 1 and 5;
```

```
select *
from etudiant
where nom between 'Grant' and 'Smith';
```

```
select *
from etudiant
where date_naissance between '01/01/1998' and '10/08/2000';
```

Attention, les bornes sont comprises dans l'intervalle de valeurs permises

IN (…)

- Permet d'exprimer des conditions d'appartenance à une liste de valeurs
- Sur des colonnes de type nombre, chaîne de caractères et date

Exemples

```
select nom
from localite
where code_postal in (1000, 4000, 5000);
```

```
select libelle
from section
where code in ('DA','TI','IR','IA');
```

```
select *
from etudiant
where date_naissance in ('02/10/2000','01/01/1999');
```

LIKE

- Permet de faire des recherches sur base de modèle (pattern):
  - % remplace n'importe quelle chaîne de caractères
  - _ remplace n'importe quel caractère
- Sur des colonnes de type nombre, chaîne de caractères et date

Exemples

```
select code
from section
where code like '_A';
```

⇨ Sélection des sections dont le code contient "A" en seconde position (ex: DA , IA …)

```
select prenom, nom
from etudiant
where email like '%gmail.com';
```

⇨ Sélection des étudiants qui ont une adresse gmail

```
select *
from etudiant
where nom like '_u%t';
```

⇨ Sélection des étudiants dont la deuxième lettre du nom est "u" et dont le nom se termine par "t" (ex: Dupont)

IS NULL

- Permet de rechercher des valeurs inconnues (NULL)
- Sur des colonnes de type nombre, chaîne de caractères et date

Exemples

```
select *
from produit
where nb_calories is null;
```

⇨ Sélection des produits dont le nombre de calories est inconnu

```
select *
from etudiant
where date_naissance is null;
```

⇨ Sélection des étudiants dont la date de naissance est inconnue

NOT

- NOT(condition)
- NOT BETWEEN … AND …
- NOT IN (…)
- NOT LIKE
- IS NOT NULL

Exemples

```
select libelle
from produit
where prix not between 1 and 5;
```

```
select libelle
from section
where code not in ('DA','TI','IR','IA');
```

```
select prenom, nom
from etudiant
where email not like '%gmail.com';
```

```
select *
from produit
where nb_calories is not null;
```

```
select reference, libelle
from produit
where not (prix < 15.5 and nb_calories > 10);
```

Fonctions

<nom_fonction> **[(*<argument>*, …)]**

- Peuvent être utilisées dans les clauses SELECT ou WHERE
- Quelques fonctions disponibles :
  - **LOWER**

    LOWER('HENALLUX Partner') ⇨ henallux partner

    Exemple

```
select   *
from     etudiant
where  lower(nom) = 'leblanc';
```

  - **UPPER**

    UPPER('HENALLUX Partner') ⇨ HENALLUX PARTNER

    Exemple

```
select upper(nom)
from  localite ;
```

Fonctions

- **CONCAT**
  CONCAT('Hello','World') ⇨ HelloWorld
- **LEN**
  LEN('Welcome') ⇨ 7
- **REPLACE**
  REPLACE('Hellk wkrld','k','o') ⇨ Hello world
- **ROUND**
  ROUND(123.456,2) ⇨ 123.46
- **LEFT**
  LEFT('Welcome',3) ⇨ Wel
- **LTRIM**
  LTRIM(' Welcome') ⇨ Welcome
- … et bien d'autres

Trier les lignes à l'affichage

Affiche les lignes dans un certain ordre

Rappel : pas de notion d'ordre dans une BD relationnelle

⇨ Le tri à l'affichage ne modifie pas l'ordre des lignes dans la table

```
SELECT   *   |   [ DISTINCT ]   <expression>   [ <alias> ] , ...
FROM     <nom_table>
[ WHERE  <condition(s)> ]
[ ORDER BY <expression> | <alias> [ ASC | DESC ] , …  ] ;
```

Ordre ascendant (par défaut)

Ordre descendant

Exemple

```
select *
from produit
order by prix;
```

Trier les lignes à l'affichage

Exemples

```
select prenom, nom
from etudiant
order by date_naissance desc;
```

→ Étudiants affichés triés par date de naissance, en commençant par les plus jeunes

```
select *
from localite
order by code_postal, nom;
```

→ Localités affichées triées par code postal et en cas de code postal identique, par ordre alphabétique sur le nom

```
select *
from etudiant
where date_naissance is not null
order by nom, prenom;
```

→ Étudiants affichés triés par ordre alphabétique sur le nom et en cas de nom identique, par ordre alphabétique sur le prénom

Résumé des requêtes simples

Toutes les colonnes

Évite les duplications

- Colonne
- Expression arithmétique (+ - * /)
  (si contient null ⇨ vaut null)
- Fonction

Renomme une colonne à l'affichage

```
SELECT  *  |  [ DISTINCT ]  <expression>  [ <alias> ] , ...
FROM    <nom_table>
[ WHERE  <condition(s)> ]
[ ORDER BY <expression> | <alias> [ ASC | DESC ] , …  ] ;
```

= < <= > >= <>
BETWEEN … AND …
IN (…)
LIKE …
IS (NOT) NULL
NOT
AND
OR

Trie les lignes à l'affichage :

ASC : par ordre ascendant

DESC : par ordre descendant