BDW - Bases du SQL

Fabien Duchateau

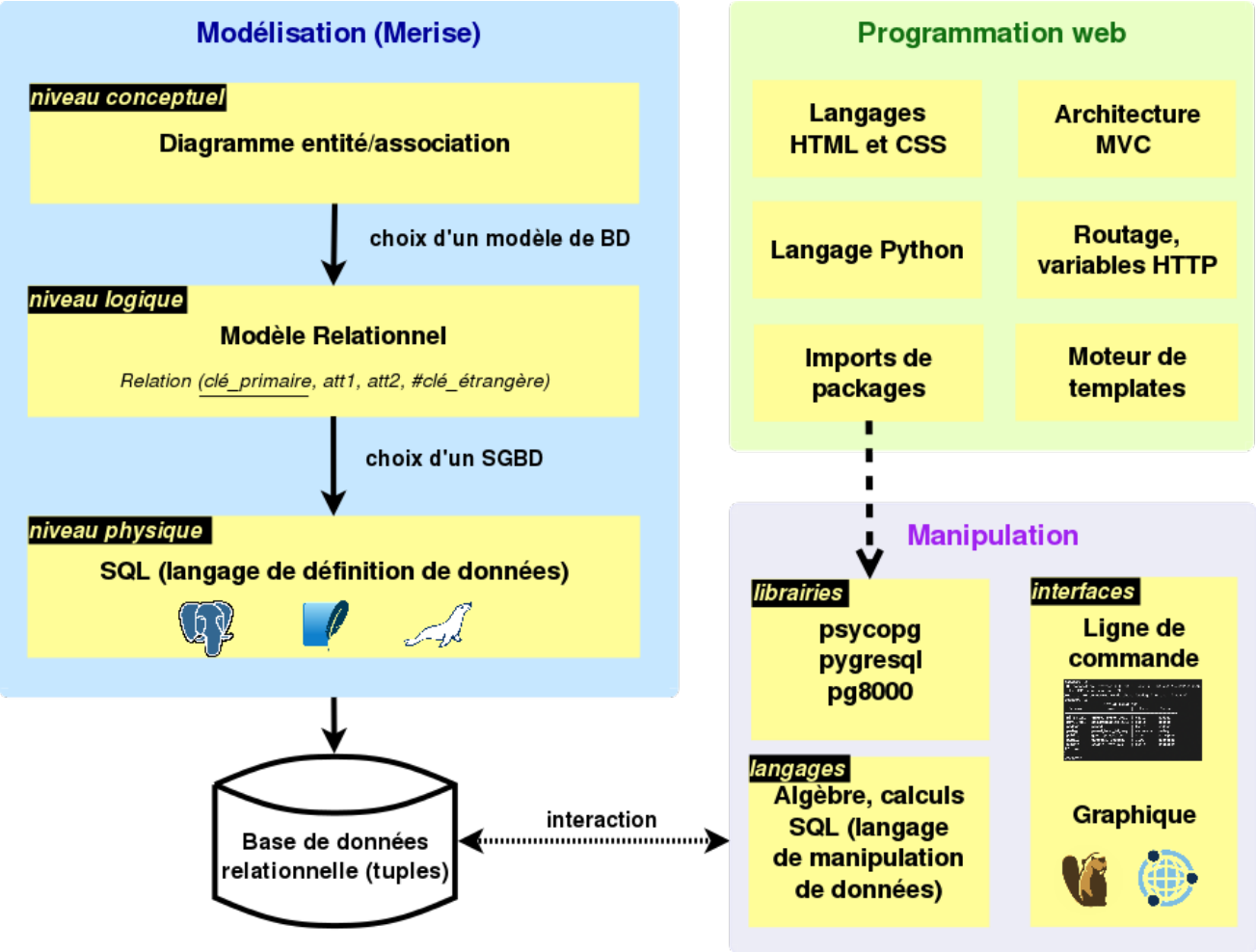
fabien.duchateau [at] univ-lyon1.fr

Université Claude Bernard Lyon 1

2024 - 2025

https://perso.liris.cnrs.fr/fabien.duchateau/BDW/

Positionnement dans BDW



Ces diapositives utilisent le genre féminin (e.g., chercheuse, développeuses) plutôt que l'écriture inclusive (moins accessible, moins concise, et pas totalement inclusive)

Rappels de cours

Calcul Relationnel à Variable Tuple (CRVT) :

- Une variable est un tuple (une ligne)
- Accès aux attributs par *t.attribut* où *t* est un tuple

Algèbre Relationnelle (AR) :

- Combinaison d'opérateurs (projection, sélection, jointure, etc.)

Le langage SQL est basé sur le CRVT
Un SGBD traduit une requête SQL en opérateurs de l'AR

Rappel du jeu de données

Élève (idE, *nomE*, *moyenneLycée*, *effectifLycée*)
Candidature (#*idE*, #*nomU*, département, *décision*)
Université (nomU, *ville*, *effectif*)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Dans ces diapositives, les tables sont abrégées en *E*, *C* et *U*

Jeu de données francisé inspiré du cours Databases de Stanford

Plan

Le langage SQL

Projection (SELECT ... FROM)

Sélection (WHERE)

Tri, limite (ORDER BY, LIMIT)

Qu'est ce que SQL ?

SQL = Structured Query Language

Un langage concret pour interagir avec le modèle relationnel :

- Un langage de manipulation de données
- Un langage de description de données
- Un langage pour administrer la base, gérer les contrôles d'accès

Langage de manipulation de données **déclaratif** ⇒ description du résultat escompté

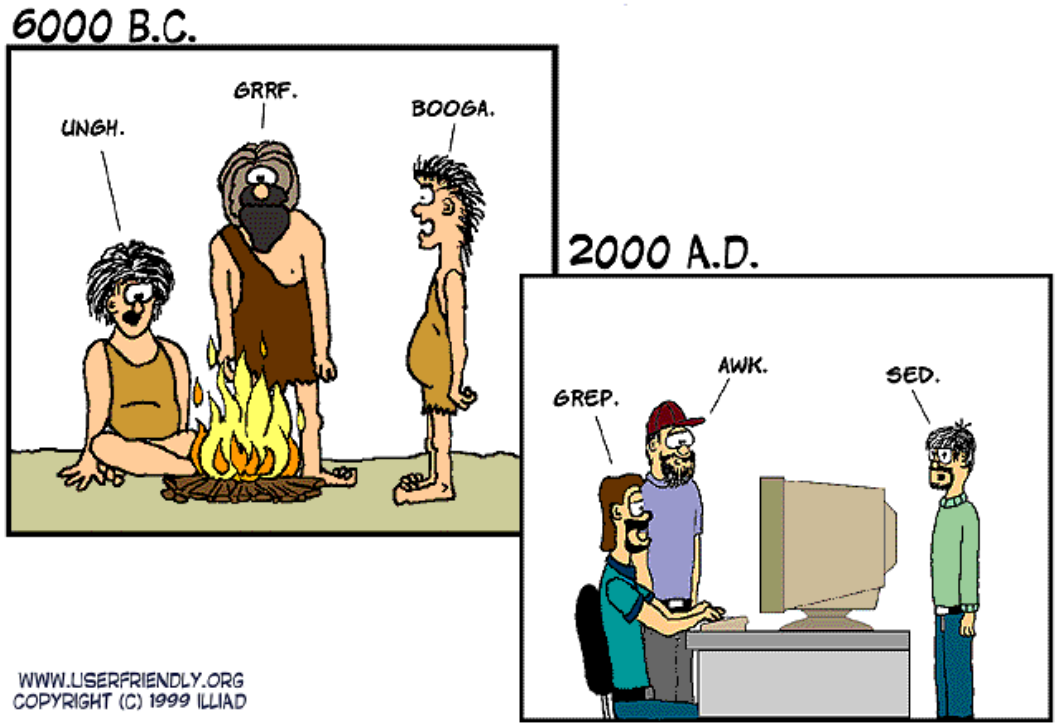
Cours et tutoriels SQL sur http://sql.sh
Voir aussi http://modern-sql.com/ et http://use-the-index-luke.com/

Historique de SQL

Origine : IBM en 1974

Versions de la norme :

- SQL-87 : 1987 (ISO)
- SQL-2 : 1992
- SQL : 1999
- ...
- SQL-2019
- SQL-2023



http://fr.wikipedia.org/wiki/SQL

SQL dans les SGBD

De nombreux SGBD utilisent le langage SQL :

- PostgreSQL
- MySQL / MariaDB
- Oracle
- SQL Server
- LibreOffice Base, Access
- ...

Chaque SGBD implémente plus ou moins respectueusement le standard SQL (différents niveaux de conformité)

https://tdan.com/is-sql-a-real-standard-anymore/4923

Différences entre la théorie et SQL

- Possibilité de doublons
- Possibilité d'ordonner le résultat des requêtes
- Notion de valeur non définie
- Absence de certains opérateurs ensemblistes

La casse en SQL

Sont insensibles à la casse :

- Les clauses d'une requête (SELECT, WHERE, ...)
- Les noms de tables et colonnes (parfois selon l'OS)

Sont sensibles à la casse :

- Les valeurs des constantes (e.g., dans les conditions)

Exemple :
SELECT nomU **FROM** Université **WHERE** nomU = 'UCB' ;
≡ **select** nomu **from** université **where** nomu = 'UCB' ;
mais
≢ **SELECT** nomU **FROM** Université **WHERE** nomU = 'Ucb' ;

En résumé

Le langage SQL :

- Basé sur le CRVT
- Différences entre la théorie (monde ensembliste) et la norme SQL
- Différences entre la norme SQL et son implémentation dans les SGBD
- Différences entre les implémentations de SQL (propre à chaque SGBD)

Dans ce cours, la syntaxe SQL est celle implémentée dans le SGBD PostgreSQL

Plan

Le langage SQL

Projection (SELECT ... FROM)

Sélection (WHERE)

Tri, limite (ORDER BY, LIMIT)

Syntaxe

Syntaxe minimale d'une requête SQL :

SELECT att_1, att_2, ...
FROM *nom_table* **;**

- SELECT et FROM sont des clauses SQL
- Projection : récupération des valeurs contenues dans la table `nom_table`, en ne gardant que les attributs `att1, att2, ...`
- On peut remplacer att_1, att_2, ... par * pour utiliser tous les attributs des tables listées dans la clause FROM

Équivalences

- En SQL :
  SELECT att_1, att_2, ...
  FROM *nom_table* **;**

- En algèbre relationnelle :
  $\pi_{att_1, att_2, \ldots}(nom_table)$

- En calcul relationnel tuple :
  $\{t.att_1, t.att_2, \cdots \mid nom_table(t)\}$

Exemple de projection

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Le nom et l'effectif des universités

```
SELECT nomU, effectif
FROM Université;
```

nomU	effectif
INSA	36000
UCB	15000
UJF	10000
UJM	21000

Exemple de projection (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les universités

```
SELECT nomU, ville, effectif
FROM Université;
```

ou

```
SELECT * FROM Université;
```

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Suppression des doublons

Mot clé DISTINCT pour supprimer les n-uplets en doublon :

SELECT DISTINCT att_1, att_2, ...
FROM *nom_table* **;**

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Etienne	21000
UJF	Grenoble	10000

En rouge, un exemple de n-uplet en doublon dans la table (les trois valeurs des 2 n-uplets étant identiques)

Suppression des doublons (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les villes où se trouvent des universités

```
SELECT ville FROM Université;
```

ville
Lyon
Lyon
Grenoble
Saint-Étienne

```
SELECT DISTINCT ville FROM Université;
```

ville
Lyon
Grenoble
Saint-Étienne

Renommage et alias

Mot-clé AS pour renommer un attribut :

SELECT att_1 **AS** att'_1, att_2 **AS** att'_2, ...
FROM *nom_table* **;**

- L'attribut att_1 sera renommé en att'_1, etc.

Alias de table comme moyen de désambiguïsation ou pour utiliser deux fois la même table :

SELECT DISTINCT $t.att_1$, att_2, ...
FROM *nom_table* [**AS**] t **;**

- L'alias de la table est t, accès à l'attribut att_1 par $t.att_1$

Renommage et alias (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Le nom et la ville des universités, avec renommage

```
SELECT nomU AS nom-univ, ville AS ville-univ FROM Université;
```

nom-univ	ville-univ
INSA	Lyon
UCB	Lyon
UJF	Grenoble
UJM	Saint-Étienne

Renommage et alias (3)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Le nom des universités avec alias de table

```
SELECT u.nomU, u.ville FROM Université AS u;
```

nomU	ville
INSA	Lyon
UCB	Lyon
UJF	Grenoble
UJM	Saint-Étienne

En résumé

- Clause SELECT ≡ projection (d'attributs)
- Clause FROM ≡ liste des tables utilisées
- Renommage d'attribut (AS) et alias de table
- Syntaxe d'une requête SQL (crochets = option) :

SELECT att_1 [, att_2 [**AS** att'_2], ...]
FROM nom_table_1 [, nom_table_2 [alias] ...] **;**

Plan

Le langage SQL

Projection (SELECT ... FROM)

Sélection (WHERE)

Tri, limite (ORDER BY, LIMIT)

Syntaxe

Clause WHERE pour la sélection :

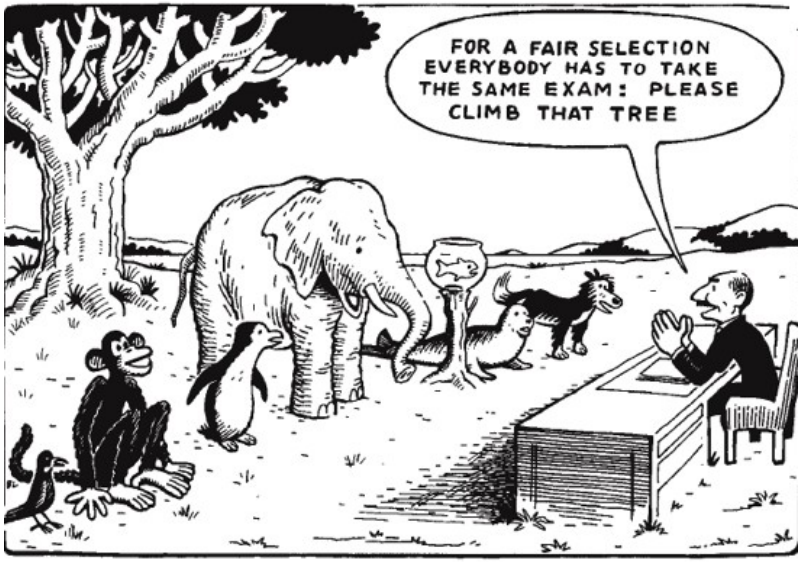
SELECT att_1, att_2, ...
FROM *nom_table*
WHERE *condition* **;**

- La clause WHERE permet de sélectionner les lignes en filtrant celles qui ne remplissent pas la condition (i.e., éliminées du résultat)

Équivalences

- En calcul relationnel tuple :
  $\{t.att_1, t.att_2, \cdots \mid nom_table(t) \wedge condition\}$

- En algèbre relationnelle :
  $\pi_{att_1,att_2,\ldots}(\sigma_{condition}(nom_table))$



Conditions

Condition du WHERE : une combinaison d'expressions connectées par **AND** ($\wedge$) ou **OR** ($\vee$)

Une expression effectue une opération entre un attribut et une constante ou entre deux attributs

Expressions simples :

- Opérateurs de comparaison (`=`, `!=`, `<`, `<=`, `>`, `>=`)
- Différents types de données utilisés pour les constantes :
  - nombres : *1, 36000, 1.5*
  - chaînes de caractères : *'UCBL', 'Grenoble'*
  - dates : *'1986-04-26'* (le formatage des dates peut varier selon le SGBD)

Exemple de sélection

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Nom et effectif des universités avec un effectif inférieur à 20000

```
SELECT nomU, effectif
FROM Université
WHERE effectif < 20000;
```

nomU	effectif
UCB	15000
UJF	10000

Exemple de sélection (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Informations sur les universités de Lyon avec un effectif inférieur à 20000

```
SELECT *
FROM Université
WHERE ville = 'Lyon'
AND effectif < 20000;
```

nomU	ville	effectif
UCB	Lyon	15000

Autres opérateurs

- Opérateur **IN** (val_1, val_2, ...) :
  - spécifie un ensemble de valeur possibles

- Opérateur **BETWEEN** val_1 **AND** val_2 :
  - spécifie un intervalle de valeurs (bornes val_1 et val_2 incluses)
  - attention à ne pas confondre le AND du **BETWEEN** avec celui qui correspond au $\wedge$

Exemple de sélection (3)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les universités de Lyon ou Grenoble

```
SELECT *
FROM Université
WHERE ville IN ('Lyon', 'Grenoble');

SELECT *
FROM Université
WHERE ville = 'Lyon' OR ville = 'Grenoble';
```

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000

Exemple de sélection (4)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les universités avec un effectif entre 12000 et 25000

```
SELECT * FROM Université
WHERE effectif BETWEEN 12000 AND 25000;
```

nomU	ville	effectif
UCB	Lyon	15000
UJM	Saint-Étienne	21000

```
SELECT * FROM Université
WHERE effectif >= 12000 AND effectif <=
↪  25000;
```

Valeurs non définies

En pratique, il est possible d'avoir des valeurs non définies :

- Elles sont représentées par le mot clé NULL
- On peut tester si une valeur n'est pas définie grâce à la condition **IS NULL** (ou son contraire **IS NOT NULL**)

Exemple de sélection (5)

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	**NULL**
UJM	Saint-Etienne	21000

Les informations sur les universités avec un effectif non défini

```
SELECT * FROM Université
WHERE effectif IS NULL;
```

nomU	ville	effectif
UJF	Grenoble	NULL

En résumé

- Clause WHERE ≡ sélection (d'instances)
- Condition = combinaison (AND, OR) d'expressions (`=`, `!=`, `<`, `<=`, `>`, `>=`)
- Opérateurs spécifiques (IN, BETWEEN), valeur non définie (NULL)
- Syntaxe d'une requête SQL (crochets = option) :

SELECT att_1 [, att_2 [**AS** att'_2], ...]
FROM nom_table_1 [, nom_table_2 [alias], ...]
[**WHERE** *condition*] **;**

Plan

Le langage SQL

Projection (SELECT ... FROM)

Sélection (WHERE)

Tri, limite (ORDER BY, LIMIT)

Syntaxe du tri

Clause ORDER BY pour trier le résultat d'une requête :

SELECT att_1, att_2, ...
FROM *nom_table*
WHERE *condition*
ORDER BY att_i, att_j, . . . **;**

- Le résultat de la requête est trié selon l'ordre naturel croissant de l'attribut att_i
- En cas d'égalité entre deux lignes au niveau de l'attribut att_i, on utilise l'attribut suivant att_j, etc.
- Le nom d'un attribut peut être suivi par ASC ou DESC pour indiquer un ordre croissant (défaut) ou décroissant

Exemple de tri

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Les informations sur les universités, triées par effectif croissant

```
SELECT * FROM Université
ORDER BY effectif ASC;
```

nomU	ville	effectif
UJF	Grenoble	10000
UCB	Lyon	15000
UJM	Saint-Étienne	21000
INSA	Lyon	36000

Exemple de tri (2)

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table ÉLÈVE

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table UNIVERSITÉ

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table CANDIDATURE

Les informations sur les universités de plus de 12000 étudiant-e-s, avec un tri par ville décroissante puis par effectif croissant

```
SELECT * FROM Université
WHERE effectif > 12000
ORDER BY ville DESC, effectif ASC;
```

nomU	ville	effectif
UJM	Saint-Étienne	21000
UCB	Lyon	15000
INSA	Lyon	36000

Syntaxe de la limitation

Clause LIMIT pour conserver un nombre restreint de résultats :

SELECT att_1, att_2, ...
FROM *nom_table*
WHERE *condition*
ORDER BY att_i, att_j, ...
LIMIT n **;**

- Les n premières instances (après le tri) sont conservés dans le résultat
- Pas un top-K, qui récupère toutes les instances avec les n meilleures valeurs

Dans la norme SQL, la syntaxe est FETCH FIRST n ROWS ONLY

Exemple de limitation

idE	nomE	moyenneLycée	effectifLycée
123	Ana	19.5	1000
234	Bob	18	1500
345	Chloé	17.5	500
456	Damien	19.5	1000
543	Chloé	17	2000
567	Éléonore	14.5	2000
654	Ana	19.5	1000
678	Farid	19	200
765	Joana	14.5	1500
789	Gisèle	17	800
876	Irène	19.5	400
898	Hector	18.5	800

Table Élève

nomU	ville	effectif
INSA	Lyon	36000
UCB	Lyon	15000
UJF	Grenoble	10000
UJM	Saint-Étienne	21000

Table Université

idE	nomU	département	décision
123	INSA	informatique	O
123	UCB	électronique	N
123	UCB	informatique	O
123	UJM	électronique	O
234	INSA	biologie	N
345	UJF	bioinformatique	O
345	UJM	bioinformatique	N
345	UJM	électronique	N
345	UJM	informatique	O
543	UJF	informatique	N
678	UCB	histoire	O
765	UCB	histoire	O
765	UJM	histoire	N
765	UJM	psychologie	O
876	UCB	informatique	N
876	UJF	biologie	O
876	UJF	biologie marine	N
898	INSA	informatique	O
898	UCB	informatique	O

Table Candidature

Deux universités qui ont les effectifs les plus élevés

```
SELECT nomU, effectif FROM Université
ORDER BY effectif DESC LIMIT 2;
```

nomU	effectif
INSA	36000
UJM	21000

En résumé

- Clause ORDER BY ≡ tri du résultat
- Clause LIMIT ≡ troncature du résultat aux premières instances
- Syntaxe d'une requête SQL (crochets = option) :

SELECT att_1 [, att_2 [**AS** att'_2], ...]
FROM nom_table_1 [, nom_table_2 [**AS** alias]]
[**WHERE** *condition*]
[**ORDER BY** att_i [, att_j, ...]]
[**LIMIT** n] **;**

Démo PostgreSQL

```
fabien=# select * from universite ;
 nomu |      ville      | effectif
------+-----------------+----------
 UCB  | Lyon            |    15000
 INSA | Lyon            |    36000
 UJF  | Grenoble        |    10000
 UJM  | Saint-Etienne   |    21000
(4 lignes)
```

Démo avec PostgreSQL (scripts SQL en ligne)