BDW - Opérateurs ensemblistes SQL

Fabien Duchateau

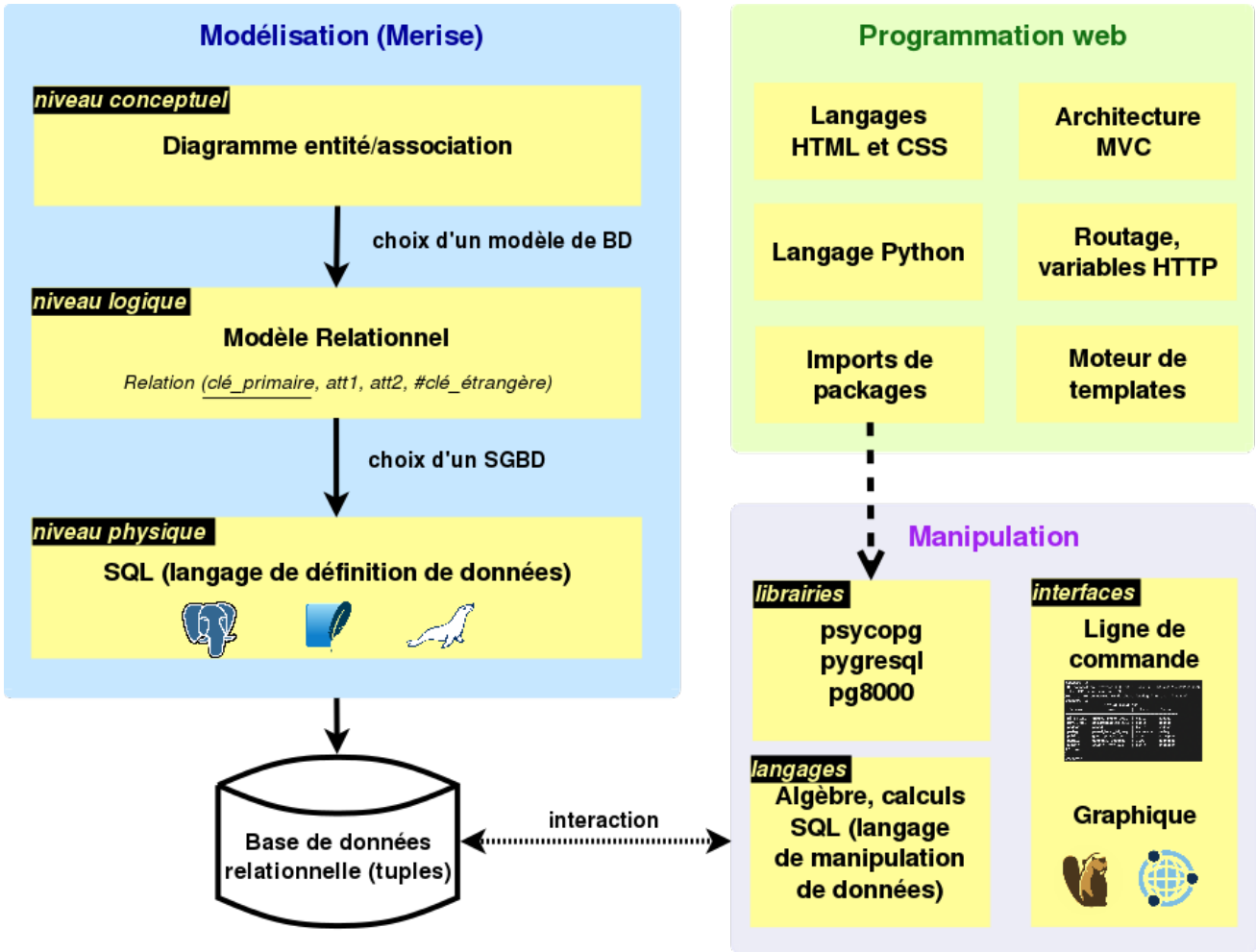
fabien.duchateau [at] univ-lyon1.fr

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)

Opérateurs ensemblistes

Comment utiliser les opérateurs ensemblistes en SQL ?

- Permettent de combiner les résultats de plusieurs SELECT
- Pas de doublons (sauf avec le mot-clé ALL)
- Les SELECT doivent contenir le même nombre d'attributs
- Le nom des attributs du résultat sont ceux du premier SELECT
  - C'est l'ordre des attributs qui compte
- Un opérateur ensembliste peut inclure un ORDER BY et un LIMIT de manière globale
  - dans ce cas, les deux SELECT sont mis entre parenthèses

Opérateurs ensemblistes (2)

Dans la norme SQL :

- $\cup$: UNION
- $\cap$: INTERSECT
- $-$: EXCEPT ou MINUS
- $\div$: pas d'opérateur (mais réalisable de différentes manières, voir cours *regroupements*)

Selon le SGBD utilisé, les attributs (des deux SELECT) ne sont pas forcément du même type : sous PostgreSQL, conversion implicite entre types compatibles (e.g., entier vers chaîne de caractères)

---

https://www.postgresql.org/docs/current/queries-union.html
https://www.postgresql.org/docs/current/typeconv-union-case.html
http://sqlpro.developpez.com/cours/divrelationnelle/

Plan

Union

Intersection

Différence

Opérateur union

(**SELECT** att_1, ..., att_n **FROM** ...)
UNION [**ALL**]
(**SELECT** att_1, ..., att_n **FROM** ...)
[**ORDER BY** att_i [, att_j, ...]]
[**LIMIT** n] ;

- Deux requêtes SELECT reliées par un opérateur UNION
- Le mot-clé ALL permet de conserver les doublons dans le résultat
- Les attributs att_i et att_j sont inclus dans att_1, ..., att_n

Exemple d'union

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 identifiant des élèves qui ont plus de 18 de moyenne ou qui sont accepté.e.s en informatique

```
SELECT idE, nomE FROM Eleve
WHERE moyenneLycee > 18
UNION
SELECT e.idE, nomE FROM Eleve e, Candidature c
WHERE e.idE = c.idE AND decision = 'O' AND
↪   departement = 'informatique';
```

idE	nomE
123	Ana
678	Farid
456	Damien
876	Irene
654	Ana
345	Chloe
898	Hector

Un moment de réflexion

Pourquoi ne peut-on pas faire une jointure entre les tables Candidature et Élève suivie d'une sélection avec les 3 conditions ?

```
SELECT e.idE, nomE
FROM Élève e NATURAL JOIN Candidature c
WHERE (décision = 'O' AND département = 'informatique')
        OR moyenneLycee > 18;
```

Un moment de réflexion

Pourquoi ne peut-on pas faire une jointure entre les tables Candidature et Élève suivie d'une sélection avec les 3 conditions ?

SELECT e.idE, nomE
FROM Élève e **NATURAL JOIN** Candidature c
WHERE (décision = 'O' **AND** département = 'informatique')
OR moyenneLycee > 18 **;**

⇒ La jointure fait "disparaître" des tuples : l'élève *(456, Damien, 19.5, 1000)* n'a pas candidaté et sera supprimé lors de la jointure malgré une moyenne de 19.5

Plan

Union

Intersection

Différence

Opérateur intersection

(**SELECT** att_1, ..., att_n **FROM** ...)
INTERSECT [**ALL**]
(**SELECT** att_1, ..., att_n **FROM** ...) **;**

- Deux requêtes SELECT reliées par un opérateur INTERSECT
- Le mot-clé ALL permet de conserver les doublons dans le résultat

Exemple d'intersection

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 identifiant des élèves qui ont candidaté à la fois en *bioinformatique* et en *informatique*

```
SELECT e.idE, nomE
FROM Eleve e JOIN Candidature USING(idE)
WHERE departement = 'bioinformatique'
INTERSECT
SELECT e.idE, nomE
FROM Eleve e JOIN Candidature USING(idE)
WHERE departement = 'informatique';
```

idE	nomE
345	Chloe

Un moment de réflexion

Pourquoi ne peut-on pas faire une jointure entre les tables Candidature et Élève suivie d'une sélection sur les 2 conditions ?

SELECT e.idE, nomE **FROM** Élève e **NATURAL JOIN** Candidature
WHERE département = 'bioinformatique' **AND** département = 'informatique' **;**

Un moment de réflexion

Pourquoi ne peut-on pas faire une jointure entre les tables Candidature et Élève suivie d'une sélection sur les 2 conditions ?

SELECT e.idE, nomE **FROM** Élève e **NATURAL JOIN** Candidature
WHERE département = 'bioinformatique' **AND** département = 'informatique' **;**

⇒Aucun tuple dans le résultat, car la valeur de *département* pour un même tuple ne peut pas prendre en même temps la valeur 'informatique' et la valeur 'bioinformatique'

Un moment de réflexion

Pourquoi ne peut-on pas faire une jointure entre les tables Candidature et Élève suivie d'une sélection sur les 2 conditions ?

SELECT e.idE, nomE **FROM** Élève e **NATURAL JOIN** Candidature
WHERE département = 'bioinformatique' **AND** département = 'informatique' **;**

⇒**Aucun tuple dans le résultat, car la valeur de *département* pour un même tuple ne peut pas prendre en même temps la valeur 'informatique' et la valeur 'bioinformatique'**

Par quoi peut-on remplacer une requête avec intersection ?

Un moment de réflexion

Pourquoi ne peut-on pas faire une jointure entre les tables Candidature et Élève suivie d'une sélection sur les 2 conditions ?

SELECT e.idE, nomE **FROM** Élève e **NATURAL JOIN** Candidature
WHERE département = 'bioinformatique' **AND** département = 'informatique' **;**

⇒**Aucun tuple dans le résultat, car la valeur de *département* pour un même tuple ne peut pas prendre en même temps la valeur 'informatique' et la valeur 'bioinformatique'**

Par quoi peut-on remplacer une requête avec intersection ?

⇒**Par une sous-requête avec IN ou EXISTS**

Plan

Union

Intersection

Différence

Opérateur différence

(**SELECT** att_1, ..., att_n **FROM** ...)
EXCEPT [**ALL**]
(**SELECT** att_1, ..., att_n **FROM** ...) **;**

- Deux requêtes SELECT reliées par un opérateur EXCEPT
- Le mot-clé ALL permet de conserver les doublons dans le résultat

Exemple de différence

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

L'identifiant des élèves qui ont candidaté en *informatique* mais pas en *bioinformatique*

```
SELECT idE FROM Candidature
WHERE departement = 'informatique'
EXCEPT
SELECT idE FROM Candidature
WHERE departement = 'bioinformatique';
```

idE
123
543
876
898

Un moment de réflexion

Par quoi peut-on remplacer une requête avec différence ?

Un moment de réflexion

Par quoi peut-on remplacer une requête avec différence ?

⇒ **Par une sous-requête avec NOT IN ou NOT EXISTS**

En résumé

- Opérateurs UNION, INTERSECT et EXCEPT dans la norme SQL et implémentés dans la majorité des SGBD
- Division avec des regroupements ou avec des sous-requêtes NOT EXISTS

```
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)