

Bases de données

Expressions Sélection simple

SQL_03b

v313c

2025-01-26

Christina.Khnaisser@USherbrooke.ca
Luc.Lavoie@USherbrooke.ca

© 2018-2021, Myrius (<http://info.usherbrooke.ca/lavoie>)
CC BY-NC-SA 4.0 (<https://creativecommons.org/licenses/by-nc-sa/4.0/>)

Plan

- **Historique**
- **SELECT (version simplifiée)**
 - Jointure (et renommage)
 - Restriction
 - Projection, extension et renommage
 - Union, intersection et différence
 - Contexte
- **Synthèse**
- **Exercices**
- **Références**



Le langage SQL

- L'histoire du SELECT
- Correspondance opérateurs relationnels
- Jointure (et renommage)
- Restriction
- Projection et extension
- Union, intersection et différence
- Contexte

Le langage SQL

L'histoire du SELECT

- La séquence d'opérations *jointure-restriction-projection-extension* contribue en pratique à une proportion importante des requêtes, dans la mesure d'un *renommage* occasionnel des termes.
- En réunissant cette séquence dans une même instruction, les concepteurs du langage SQL voulaient
 - donner une allure procédurale au langage, jugeant qu'elle faciliterait l'adhésion des programmeurs;
 - offrir un raccourci notationnel aux programmeurs;
 - faciliter l'optimisation conjointe des opérations.
- Les opérateurs ensemblistes (*union*, *différence*, *intersection*) seront intégrés par la suite, d'une façon moins commode.
- Au fil du temps, plusieurs opérateurs (ou pseudo-opérateurs) seront ajoutés – certains d'entre eux seront couverts dans le module suivant.

Le langage SQL

Algèbre relationnelle - opérateurs de base du SELECT

Restriction

$R \sigma_{\text{cond}}$

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

Projection

$R \pi \{A, C\}$

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

Jointure naturelle

$R \bowtie S$

A	B	B	C
a1	b1	b1	c1
a2	b1	b2	c2
a3	b3	b3	c3
a4	b4	b3	c4

Extension (augmentation)

$R \alpha C:f$

A	B	A	B	C
a1	b1	a1	b1	f(a1,b1)
a2	b1	a2	b1	f(a2,b1)
a3	b3	a3	b3	f(a3,b3)
a4	b4	a4	b4	f(a4,b4)

Renommage

$R \rho A:C$

A	C
...	...

Note : Le symbole de projection π est souvent omis, à l'instar de la multiplication dans les polynômes.

Opérateurs relationnels propres : 3

Opérateurs ensemblistes : 4

Opérateur structuel : 1 (renommage)

IGE 487

Le produit est-il vraiment nécessaire ?

Et le renommage ?

Quel est l'ensemble de base minimal (nécessaire et suffisant) ?

Cet ensemble est-il unique ?

Voir

DATE, C. J. ; DARWEN, H.

Databases, types, and the relational model: The third manifesto.

3rd ed., Addison-Wesley Inc., 2008.

ISBN 0-321-39942-0

Le langage SQL

Algèbre relationnelle - opérateurs ensemblistes du SELECT

Intersection

$$R \cap S$$

A	B
R	
S	

Union

$$R \cup S$$

A	B
R	
S	

Différence

$$R - S$$

A	B
R	
S	

Note : Le symbole de projection π est souvent omis, à l'instar de la multiplication dans les polynômes.

Opérateurs relationnels propres : 4 (réductible à 3 – l'extension se définit grâce aux autres)

Opérateurs ensemblistes : 3 (réductibles à 2, l'intersection se définit entre terme d'union et de différence)

Opérateur structurel : 1 (renommage)

IGE 487

Le produit est-il vraiment nécessaire ?

Et le renommage ?

Quel est l'ensemble de base minimal (nécessaire et suffisant) ?

Cet ensemble est-il unique ?

Voir

DATE, C. J. ; DARWEN, H.

Databases, types, and the relational model: The third manifesto.

3rd ed., Addison-Wesley Inc., 2008.

ISBN 0-321-39942-0

Le Langage SQL

SELECT (syntaxe simplifiée)

requête ::=

```
[ Contexte ]
SELECT opMode { * | projection-extension }
FROM listeDeJointures
[ Restriction ]
[ Groupement ]
[ OpComplémentaire ]
[ Ordonnancement ]
[ Divers ]
```

Les **catégories en vert**
sont traitées au module
suivant

opMode ::=

```
[ DISTINCT | ALL ]
```

- Le mode indique si l'expression retourne une relation (**DISTINCT**) ou une collection (**ALL**).
- L'étoile indique que tous les attributs de l'expression sont retenus (sans projection).

7

```
[ WITH [ RECURSIVE ] requête_with [, ...] ]
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    * | expression [ [ AS ] nom_d_affichage ] [, ...]
[ FROM éléments_from [, ...] ]
[ WHERE condition ]
[ GROUP BY expression [, ...] ]
[ HAVING condition [, ...] ]
[ WINDOW nom_window AS ( définition_window ) [, ...] ]
[ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select ]
[ ORDER BY expression [ ASC | DESC | USING opérateur ] [ NULLS { FIRST |
LAST } ] [, ...] ]
[ LIMIT { nombre | ALL } ]
[ OFFSET début ] [ ROW | ROWS ] ]
[ FETCH { FIRST | NEXT } [ total ] { ROW | ROWS } ONLY ]
[ FOR { UPDATE | SHARE } [ OF nom_table [, ...] ] [ NOWAIT ] [...]
```

avec *éléments_from* qui peut être :

```

[ ONLY ] nom_table [ * ] [ [ AS ] alias [ ( alias_colonne [, ...] ) ] ]
( select ) [ AS ] alias [ ( alias_colonne [, ...] ) ]
nom_requête_with [ [ AS ] alias [ ( alias_colonne [, ...] ) ] ]
nom_fonction ( [ argument [, ...] ] ) [ AS ] alias [ ( alias_colonne [, ...] |
définition_colonne [, ...] ) ]
nom_fonction ( [ argument [, ...] ] ) AS ( définition_colonne [, ...] )
éléments_from [ NATURAL ] type_jointure éléments_from [ ON condition_jointure |
USING ( colonne_jointure [, ...] ) ]

```

et *requête_with* est :

```

nom_requête_with [ ( nom_colonne [, ...] ) ] AS ( select | valeurs | insert | update |
delete )

```

```

TABLE [ ONLY ] nom_table [ * ]

```


Le Langage SQL

SELECT (séquence d'exécution)

Les clauses de l'énoncé SELECT sont exécutées dans l'ordre suivant:

1. Contexte
2. FROM ...
3. Restriction
4. Groupement
5. SELECT ...
6. OpComplémentaire
7. Ordonnancement
8. Divers

Nous suivrons cet ordre de présentation, sauf pour le contexte qui sera présenté en dernier.

Le Langage SQL

SELECT (syntaxe simplifiée) - listeDeJointures

listeDeJointures ::=

denotationTable [[**AS**] *alias*] [*jointure* ...]

denotationTable ::=

nomTable | (*requête*) | *autresDénnotations*

jointure ::=

jointure_naturelle | *produit* | *jointure_qualifiée* |
jointure_externe

Les catégories en gris
ne sont pas traitées dans
le présent cours.

- Une *listeDeJointures* permet de faire plusieurs jointures à la suite.
- Chaque terme est représenté par une *dénotationTable*, en pratique un nom de table ou une requête entre parenthèses.
- Le mot **AS** est superfétatoire, mais son utilisation est recommandée.
- L'identifiant *alias* permet de (re)nommer la *dénotationTable* en même temps.
- Rappel : une requête n'est pas autre chose qu'une expression **SELECT**.

Le Langage SQL

SELECT (syntaxe simplifiée) – jointure naturelle

jointure_naturelle ::=

NATURAL [**INNER**] **JOIN** *denotationTable* [[**AS**] *alias*]

- C'est la jointure de la théorie relationnelle.
- Le mot **INNER** est superfétatoire (sic).
- Le mot **AS** est superfétatoire, mais son utilisation est recommandée.
- L'identifiant *alias* permet de (re)nommer la *dénotationTable* en même temps.

Le Langage SQL

SELECT (syntaxe simplifiée) - produit

produit ::=

opProduit *denotationTable* [[**AS**] *alias*]

opProduit ::=

CROSS JOIN | ,

- C'est le produit de la théorie relationnelle.
- Le mot **AS** est superfétatoire, mais son utilisation est recommandée.
- L'identifiant *alias* permet de (re)nommer la *dénotationTable* en même temps.

Le Langage SQL

SELECT (syntaxe simplifiée) – jointure qualifiée

jointure_qualifiée ::=

[**INNER**] **JOIN** *denotationTable* [[**AS**] *alias*] *qualificationJ*

qualificationJ ::=

ON *conditionJ*

 | **USING** (*listeNomCol*)

- Le mot **INNER** est superfétatoire et son utilisation non recommandée.
- Le mot **AS** est superfétatoire, mais son utilisation est recommandée.
- L'identifiant *alias* permet de (re)nommer la *dénotationTable* en même temps.
- La *qualificationJ* permet de renommer les attributs de jointures en même temps.

Le langage SQL

Jointures, simplifions!

- Les jointures qualifiées sont les jointures de prédilection
 - **USING** (a_1, \dots, a_n)
lorsque les identifiants des attributs de jointure sont les mêmes dans les deux tables (on désigne souvent cette jointure comme étant *restreinte*).
 - **ON** $a_1=b_1$ **AND** ... **AND** $a_n=b_n$
lorsque les identifiants des attributs de jointure ne sont pas les mêmes (on désigne souvent cette jointure comme étant *conditionnelle*).

Note

- La variante ON est beaucoup plus riche qu'indiqué, mais il existe de nombreuses bonnes raisons pour ne pas utiliser cette richesse. L'exposé de ces raisons dépasse toutefois la portée du cours. En conséquence, nous nous limiterons à la seule forme recommandable.

Évaluation – Exemple de requêtes (inner join)

1. Quels sont les types d'évaluation ?
2. Quels sont les types d'évaluation ayant des notes ?
3. Quels sont les personnes étudiantes inscrites ?
4. Quels sont les personnes étudiantes non inscrites ?

1.

```
SELECT * FROM TypeEvaluation;
```

2.

```
SELECT * FROM TypeEvaluation JOIN Resultat ON (TE =code);
```

3.

```
SELECT * FROM Etudiant JOIN Resultat USING(matricule);
```

-- V2 intersect

```
SELECT matricule FROM Etudiant  
INTERSECT  
SELECT matricule FROM Resultat;
```

4.

```
SELECT * FROM Etudiant LEFT JOIN Resultat USING(matricule);
```

-- V2 différence

```
SELECT matricule FROM Etudiant  
EXCEPT  
SELECT matricule FROM Resultat;
```

Voir le code :

Exemples/Evaluation/Evaluation_req.sql

Évaluation – Exemple de données (inner join)

Étudiant

matricule	nom	adresse
15113150	Paul	>Δ ^ς σ ^ς ᵇ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Activité

sigle	titre
IFT159	Analyse et programmation
IFT187	Éléments de bases de données
IMN117	Acquisition des médias numériques
IGE401	Gestion de projets
GMQ103	Géopositionnement

TypeÉvaluation

code	description
IN	Examen intra
FI	Examen final
TP	Travail pratique
PR	Projet

Résultat

matricule	TE	activité	trimestre	note
15113150	TP	IFT187	20133	80
15112354	FI	IFT187	20123	78
15113150	TP	IFT159	20133	75
15112354	FI	GMQ103	20123	85
15110132	IN	IMN117	20123	90
15110132	IN	IFT187	20133	45
15112354	FI	IFT159	20123	52

On remarque l'omission de la dénotation des types des attributs.

C'est fréquemment le cas dans les représentations graphiques, afin de les alléger.

Cela ne porte pas à conséquence dans la mesure où une définition textuelle les accompagne, ce qui est le cas ici.

Nous verrons bientôt d'autres représentations graphiques plus complètes.

Note : >Δ^ςσ^ςᵇ se prononce (approximativement) Puvirnitug

Exercice au tableau

Comment les jointures de SQL gèrent-elles les attributs du résultat ?

La jointure naturelle (celle de la théorie relationnelle et le NATURAL JOIN de SQL) opère sur tous les attributs de même nom et élimine (automatiquement) la redondance des attributs de leur résultat, par exemple :

```
select *  
from Resultat natural join Etudiant
```

aura pour entête, puisque matricule est le seul attribut commun :

```
matricule, TE, activité, trimestre, note, nom, adresse
```

En conséquence, la référence à l'attribut matricule dans la clause SELECT ne pose aucun problème, par exemple

```
select matricule, nom, TE, activité, trimestre, note  
from Resultat natural join Etudiant
```

Comment les jointures de SQL gèrent-elles les attributs du résultat ?

Par contre, si on utilise la jointure conditionnelle (JOIN ... ON), ça ne sera pas le cas, ainsi :

```
select *  
from Resultat join Etudiant  
on Resultat.matricule = Etudiant.matricule
```

aura pour entête :

```
Resultat.matricule, TE, activité, trimestre, note,  
Etudiant.matricule, nom, adresse
```

En conséquence, la référence à l'attribut matricule dans la clause SELECT devra préciser lequel des deux est souhaité, même s'ils sont rigoureusement égaux! Par exemple :

```
select  
Etudiant.matricule, nom, TE, activité, trimestre, note  
from Resultat join Etudiant  
on Resultat.matricule = Etudiant.matricule
```

Comment les jointures de SQL gèrent-elles les attributs du résultat ?

Qu'en est-il alors de la jointure restreinte (JOIN ... USING) ?

Son comportement est analogue à celui de la jointure naturelle pour les attributs communs retenus par la clause USING et analogue à la jointure conditionnelle pour les autres attributs communs.

:-)

Comme quoi une tentative de simplification du renommage peut aussi entraîner parfois une augmentation de la complexité.

Encore un dernier exemple relatif au renommage dans les jointures

- Soit les tables A (x, y, z) et B (x, y, w)
- Pour une jointure naturelle :
 - `select * from A natural join B` $\rightarrow (x, y, z, w)$
- Pour une jointure sur le seul attribut x :
 - `with`
 A1 as (select x, y as a_y, z from A),
 B1 as (select x, y as b_y, w from B)
 select * from A1 natural join B1 $\rightarrow (x, a_y, z, b_y, w)$
- Ainsi que
 - `select * from A join B on A.x=B.x` $\rightarrow (A.x, A.y, z, B.x, B.y, w)$

Jointures externes

- SQL définit également des jointures dites « externes ».
- Nous y reviendrons dans le module relatif au traitement des informations manquantes.

Le Langage SQL

SELECT (syntaxe simplifiée) - Restriction

Restriction ::=

WHERE *condition*

- C'est la restriction de la théorie relationnelle.
- La *condition* suit la syntaxe présentée antérieurement. Les attributs y sont limités aux seuls attributs obtenus par la jointure de la clause **FROM** qui précède.

Évaluation – Exemple de requêtes (restriction)

1. Quels sont les étudiants inscrits en IFT 187?
2. Quels sont les étudiants inscrits à une activité d'informatique à l'automne 2013?
3. Quels sont les étudiants en situation d'échec au final à l'automne 2012?
4. Produire le relevé de notes d'Éliane.

Voir

Exemples/Evaluation/Evaluation_req.sql

Exercices

Lister les résultats des TP

Lister les résultats des TP où la note est < 50

Lister les résultats des TP où la note entre 70 et 100

Liste les activités d'informatique

Évaluation – Exemple de données (restriction)

Étudiant

matricule	nom	adresse
15113150	Paul	>Δ ^ς σ ^ς ᵇ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Activité

sigle	titre
IFT159	Analyse et programmation
IFT187	Éléments de bases de données
IMN117	Acquisition des médias numériques
IGE401	Gestion de projets
GMQ103	Géopositionnement

TypeÉvaluation

code	description
IN	Examen intra
FI	Examen final
TP	Travail pratique
PR	Projet

Résultat

matricule	TE	activité	trimestre	note
15113150	TP	IFT187	20133	80
15112354	FI	IFT187	20123	78
15113150	TP	IFT159	20133	75
15112354	FI	GMQ103	20123	85
15110132	IN	IMN117	20123	90
15110132	IN	IFT187	20133	45
15112354	FI	IFT159	20123	52

On remarque l'omission de la dénotation des types des attributs.

C'est fréquemment le cas dans les représentations graphiques, afin de les alléger.

Cela ne porte pas à conséquence dans la mesure où une définition textuelle les accompagne, ce qui est le cas ici.

Nous verrons bientôt d'autres représentations graphiques plus complètes.

Note : >Δ^ςσ^ςᵇ se prononce (approximativement) Puvirnitug

Exercice au tableau

Le Langage SQL

SELECT (syntaxe simplifiée) – projection et extension

projection-extension ::=

{ *expression* [[**AS**] *nomCol*] ... , }

- Tout retenir (par **SELECT ***) est peu flexible.
- La clause *projection-extension* permet de ne retenir que les expressions désirées et de les nommer.
- Une *expression* suit la syntaxe qui a été présentée antérieurement. Les attributs y sont limités aux seuls attributs obtenus par la jointure de la clause **FROM** qui suit.
- Lorsqu'une expression se réduit à un identificateur, il s'agit en fait simplement d'une projection de la théorie relationnelle.
- Le mot **AS** est superfétatoire, mais son utilisation est recommandée.
- L'identifiant *alias* permet de nommer l'*expression* en même temps.

Évaluation – Exemple de requêtes (projection et extension)

1. Lister le sigle et le titre des activités d'informatique.
2. Lister le sigle des activités ayant au moins un étudiant inscrit.
3. Étendre la table résultat pour ajouter la cote 'A' si la note est au moins de 90 et la cote 'B' si la note au moins de 70 mais inférieure à 90.
4. Lister les notes pour tous les étudiants. Afficher '0' si l'étudiant n'a pas de note pour l'activité.

```
SELECT sigle, titre FROM Activite WHERE SUBSTR(sigle, 1, 3) IN ('IFT', 'IGE');
```

```
SELECT activite FROM Resultat;
```

```
SELECT DISTINCT activite FROM Resultat;
```

```
SELECT *,
  CASE
    WHEN note >= 90 THEN 'A'
    WHEN note BETWEEN 70 AND 89 THEN 'B'
    ELSE 'C'
  END AS cote
FROM Resultat;
```

```
SELECT matricule, te, activite, trimestre, COALESCE(note, 0)
FROM Etudiant LEFT JOIN Resultat USING(matricule);
```

Voir

Exemples/Evaluation/Evaluation_req.sql

Évaluation – Exemple de données (projection et extension)

Étudiant

matricule	nom	adresse
15113150	Paul	>Δ ^ς σ ^ς ᵇ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Activité

sigle	titre
IFT159	Analyse et programmation
IFT187	Éléments de bases de données
IMN117	Acquisition des médias numériques
IGE401	Gestion de projets
GMQ103	Géopositionnement

TypeÉvaluation

code	description
IN	Examen intra
FI	Examen final
TP	Travail pratique
PR	Projet

Résultat

matricule	TE	activité	trimestre	note
15113150	TP	IFT187	20133	80
15112354	FI	IFT187	20123	78
15113150	TP	IFT159	20133	75
15112354	FI	GMQ103	20123	85
15110132	IN	IMN117	20123	90
15110132	IN	IFT187	20133	45
15112354	FI	IFT159	20123	52

On remarque l'omission de la dénotation des types des attributs.

C'est fréquemment le cas dans les représentations graphiques, afin de les alléger.

Cela ne porte pas à conséquence dans la mesure où une définition textuelle les accompagne, ce qui est le cas ici.

Nous verrons bientôt d'autres représentations graphiques plus complètes.

Note : >Δ^ςσ^ςᵇ se prononce (approximativement) Puvirnitug

Exercice au tableau

2025-01-26

SQL_03b : Expressions, fonctions simples (v31.3c) © 2018-2025, MPTG — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

Le Langage SQL

SELECT (syntaxe simplifiée) - OpComplémentaire

OpComplémentaire ::=

- INTERSECT opMode requête*
- | *UNION opMode requête*
- | *EXCEPT opMode requête*
- | *jointure [opComplémentaire]*

- Attention à *opMode*!
- En pratique, la deuxième forme de jointure est inutile (surtout quand nous aurons le *Contexte*).

Évaluation – Exemple de requêtes (U, I, D)

1. Lister les étudiants qui ont des notes.
2. Lister les étudiants qui n'ont pas eu de note.
3. ...

Évaluation – Exemple de données (U, I, D)

Étudiant

matricule	nom	adresse
15113150	Paul	>Δ ^ς σ ^ς ᵇ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Activité

sigle	titre
IFT159	Analyse et programmation
IFT187	Éléments de bases de données
IMN117	Acquisition des médias numériques
IGE401	Gestion de projets
GMQ103	Géopositionnement

TypeÉvaluation

code	description
IN	Examen intra
FI	Examen final
TP	Travail pratique
PR	Projet

Résultat

matricule	TE	activité	trimestre	note
15113150	TP	IFT187	20133	80
15112354	FI	IFT187	20123	78
15113150	TP	IFT159	20133	75
15112354	FI	GMQ103	20123	85
15110132	IN	IMN117	20123	90
15110132	IN	IFT187	20133	45
15112354	FI	IFT159	20123	52

On remarque l'omission de la dénotation des types des attributs.

C'est fréquemment le cas dans les représentations graphiques, afin de les alléger.

Cela ne porte pas à conséquence dans la mesure où une définition textuelle les accompagne, ce qui est le cas ici.

Nous verrons bientôt d'autres représentations graphiques plus complètes.

Note : >Δ^ςσ^ςᵇ se prononce (approximativement) Puvirnitug

Le Langage SQL

SELECT (syntaxe simplifiée) - contexte

Contexte ::=

WITH [**RECURSIVE**] { *requêteCont* ... , }

requêteCont ::=

nomReqCont [({ *nomCol* ... , })] **AS** (*opDefTable*)

opDefTable ::=

requête | *valeurs* | *insertion* | *miseAJour* | *retrait*

- Ceci est une adaptation la notation mathématique usuelle permettant de définir des sous-expressions (*opDefTable*) en leur donnant un nom (*nomReqCont*) utilisable dans l'expression qui suit afin d'en faciliter l'écriture... et la lecture!
- Nous l'utiliserons abondamment.

Gérer la complexité des requêtes

Emboîtement vs WITH

- Au sein d'une requête, le plus souvent, moyennant quelques contorsions syntaxiques, il est possible d'utiliser une expression SELECT à la place d'une référence à une table.
- Assez rapidement, la lisibilité en souffre.
- L'énoncé WITH devient alors l'instrument privilégié de décomposition syntaxique (diviser pour régner).
- Le seul moment où il est nécessaire de recourir à l'emboîtement : lorsque le SELECT fait référence à une variable liée définie dans le contexte englobant.

Évaluation – Exemple de requêtes (with)

1. Quels étudiants ne sont inscrits à aucune activité en 2013 ?
2. Lister la note des examens finaux des activités d'informatique du trimestre 20123 pour tous les étudiants. Afficher '0' si l'étudiant n'a pas de note pour l'activité.

WITH

```
Inscrit2013 AS -- (Résultat  $\sigma$  ( $20131 \leq \text{trimestre} \leq 20133$ ))  $\pi$  {matricule}
(
  SELECT matricule
  FROM Resultat
  WHERE trimestre BETWEEN '20131' AND '20133'
),
NonInscrit2013 AS -- Étudiant – (Étudiant  $\bowtie$  Inscrits2013)  $\pi$  {matricule}
(
  SELECT *
  FROM Etudiant
  EXCEPT
  SELECT *
  FROM Etudiant JOIN Inscrit2013 USING (matricule)
)
SELECT *
FROM NonInscrit2013
;
```

-- Très créatif :-)

WITH ActiviteInfo **AS**

(
 SELECT sigle **as** activite
 FROM Activite
 WHERE *SUBSTR*(sigle, 1, 3) **IN** ('IFT', 'IGE', 'IMN')

),

EtudiantInfo **AS**

(
 SELECT matricule, activite, '20123' **AS** trimestre, 'FI' **AS** te
 FROM Etudiant **CROSS JOIN** ActiviteInfo

),

etudiantNote **AS**

(
 SELECT matricule,
 activite,
 trimestre,
 te,
 coalesce(note, 0) **AS** note
 FROM Resultat **RIGHT JOIN** EtudiantInfo
 USING(matricule, activite, trimestre, te)

)

SELECT * **FROM** etudiantNote

WHERE te = 'FI' **AND** trimestre = '20123';

Évaluation – Exemple de données (with)

Étudiant

matricule	nom	adresse
15113150	Paul	>Δ ^ς σ ^ς ᵇ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Activité

sigle	titre
IFT159	Analyse et programmation
IFT187	Éléments de bases de données
IMN117	Acquisition des médias numériques
IGE401	Gestion de projets
GMQ103	Géopositionnement

TypeÉvaluation

code	description
IN	Examen intra
FI	Examen final
TP	Travail pratique
PR	Projet

Résultat

matricule	TE	activité	trimestre	note
15113150	TP	IFT187	20133	80
15112354	FI	IFT187	20123	78
15113150	TP	IFT159	20133	75
15112354	FI	GMQ103	20123	85
15110132	IN	IMN117	20123	90
15110132	IN	IFT187	20133	45
15112354	FI	IFT159	20123	52

On remarque l'omission de la dénotation des types des attributs.

C'est fréquemment le cas dans les représentations graphiques, afin de les alléger.

Cela ne porte pas à conséquence dans la mesure où une définition textuelle les accompagne, ce qui est le cas ici.

Nous verrons bientôt d'autres représentations graphiques plus complètes.

Note : >Δ^ςσ^ςᵇ se prononce (approximativement) Puvirnitug


```
WITH
Inscrit2013 AS -- (Résultat  $\sigma$  ( $20131 \leq \text{trimestre} \leq 20133$ ))  $\pi$  {matricule}
(
  SELECT matricule
  FROM Resultat
  WHERE trimestre BETWEEN '20131' AND '20133'
),
NonInscrit2013 AS -- Étudiant – (Étudiant  $\bowtie$  Inscrits2013)  $\pi$  {matricule}
(
  SELECT *
  FROM Etudiant
  EXCEPT
  SELECT *
  FROM Etudiant JOIN Inscrit2013 USING (matricule)
)
SELECT *
FROM NonInscrit2013
;
```

Synthèse

Le Langage SQL SELECT (exemples simples)

Norme ISO

Dialecte

PostgreSQL

Soit les relations (tables)

- A (x, y, z)
- B (v, w)
- C (v, w, x)
- D (x, y, z)

- Renommage : $A \rho \{x:q\}$
 - SELECT x AS q, y, z FROM A
- Restriction : $A \sigma \text{cond}$
 - SELECT * FROM A WHERE cond
- Projection : $A \pi \{y, z\}$
 - SELECT DISTINCT y, z FROM A
- Produit : $A \times B$
 - SELECT * FROM A, B
- Jointure : $A \bowtie C$
 - SELECT * FROM A NATURAL JOIN C
- Union : $A \cup D$
 - SELECT * FROM A UNION DISTINCT
SELECT * FROM D
- Intersection : $A \cap D$
 - SELECT * FROM A INTERSECT DISTINCT
SELECT * FROM D
- Différence : $A - D$
 - SELECT * FROM A EXCEPT DISTINCT
SELECT * FROM D

Le Langage SQL SELECT (exemples simples) Dialecte Oracle

Soit les relations (tables)

- A (x, y, z)
- B (v, w)
- C (v, w, x)
- D (x, y, z)

- Renommage : $A \rho \{x:q\}$
 - SELECT x AS q, y, z FROM A
- Restriction : $A \sigma \text{cond}$
 - SELECT * FROM A WHERE cond
- Projection : $A \pi \{y, z\}$
 - SELECT DISTINCT y, z FROM A
- Produit : $A \times B$
 - SELECT * FROM A, B
- Jointure : $A \bowtie C$
 - SELECT * FROM A NATURAL JOIN C
- Union : $A \cup D$
 - SELECT * FROM A UNION DISTINCT
SELECT * FROM D
- Intersection : $A \cap D$
 - SELECT * FROM A INTERSECT DISTINCT
SELECT * FROM D
- Différence : $A - D$
 - SELECT * FROM A MINUS DISTINCT
SELECT * FROM D

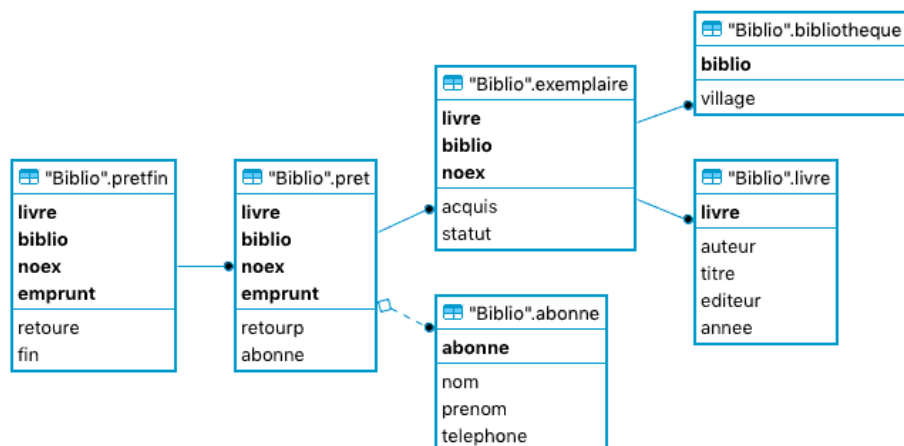
Exercices

- Bibliovik
- ... et les autres

Exercice

Bibliovik – Diagramme relationnel

- Proposer un schéma SQL correspondant au diagramme relationnel suivant :



Exercices

Bibliovik - Requêtes

- Lister les abonnés.
- Lister les livres de l'auteur « Michel Tremblay », afficher le titre et l'année seulement.
- Lister les livres acquis en 2019 (le livre doit apparaître une seule fois).
- Quels sont les livres apparus en 2020 qui sont commandés ?
- Quels sont les livres prêtés à l'abonné A123456 ?
- Quels sont les emprunts en retard ?
- Quels sont les livres qui n'existent pas dans la bibliothèque ?

Références

- Elmasri et Navathe (4^e ed.), chapitre 7
- Elmasri et Navathe (6^e ed.), chapitre 4
- [Loney2008]
Loney, Kevin ;
Oracle Database 11g: The Complete Reference.
Oracle Press/McGraw-Hill/Osborne, 2008.
ISBN 978-0071598750.
- [Date2012]
Date, Chris J. ;
SQL and Relational Theory : How to Write Accurate SQL Code.
2nd edition, O'Reilly, 2012.
ISBN 978-1-449-31640-2.
- Le site d'Oracle (en anglais)
 - http://docs.oracle.com/cd/E11882_01/index.htm
- Le site de PostgreSQL (en français)
 - <http://docs.postgresqlfr.org>
 - plus particulièrement la partie VI - Référence, rubrique SELECT
<https://doc.postgresql.fr/15/sql-select.html>

