Bases de données
SQL

SQL
LMD – partie 4
Synthèse

BD107
v310d
2020-10-31

Département d'informatique
Faculté des sciences

UNIVERSITÉ DE SHERBROOKE

Christina.Khnaisser@USherbrooke.ca
http://info.USherbrooke.ca/ckhnaisser
Luc.Lavoie@USherbrooke.ca
http://info.USherbrooke.ca/llavoie

PLAN

- Quantification
- Jointures externes
- Retour sur le « modèle » SQL
- Retour sur la notion d'intégrité
- Références

QUANTIFICATION

- Motivation
- Quantification générale
- Quantification de comparaison

QUANTIFICATION
MOTIVATION

- Compléter l'équivalence avec la logique du premier ordre.
- Caractériser les cas où il est nécessaire d'imbriquer les SELECT (par oppostion aux cas où il est suffisant, et préférable, d'utiliser WITH ou GROUP BY).

LE LANGAGE SQL
SELECT - QUANTIFICATION GÉNÉRALE (1)

- $\exists\ x.\ p(x)$:
  - *ensemble* $\neq$ *vide*
  - 0 <> (SELECT count(*) FROM *ensemble*)
  - EXISTS (*ensemble*)
- $\neg\ \exists\ x.\ p(x)$:
  - *ensemble* = *vide*
  - 0 = (SELECT count(*) FROM *ensemble*)
  - NOT EXISTS (*ensemble*)

- Pourquoi alors définir un nouvel opérateur ?
  - ... plus simple, plus expressif et souvent plus efficace

0 le nombre de tuple.

https://explainextended.com/2009/09/16/not-in-vs-not-exists-vs-left-join-is-null-postgresql/

LE LANGAGE SQL
SELECT - QUANTIFICATION GÉNÉRALE (2)

- Quantification universelle absente ?
  - Il n'y pas d'opérateur « FORALL »
- Pas un problème, puisque
  - $\forall x.\ p(x) \equiv \neg\exists x.\ \neg p(x)$
- Donc
  - FORALL (cond) ≡
    NOT EXISTS (NOT cond)
  - FORALL (requête [cond]) ≡
    NOT EXISTS (requête [NOT cond])

LE LANGAGE SQL
SELECT (SYNTAXE SIMPLIFIÉE) - QUANTIFICATION

restriction ::=
  [WHERE *condition*]
condition ::=
  quantification
 | ...
quantification ::=
  [NOT] EXISTS (*requête*)

- En conséquence,
  - il y aura un emboitement de SELECT
  - une « variable » devra lier le SELECT intérieur au SELECT extérieur.

- Et plus, si affinités ☺

C'est le fameux « x » de la quantification.

En pratique, on utilise souvent un produit cartésien restreint sur le modèle de l'équivalence entre la jointure et le produit cartésien restreint.

EXEMPLE DE DONNÉES

Étudiant

matricule	nom	adresse
15113150	Paul	>ᐃᖏᓂᖅ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

TypeÉvaluation

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

Activité

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

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 : >ᐃᖏᓂᖅ se prononce (approximativement) Puvirnituq

EXEMPLE DE QUANTIFICATION 1

Quels sont les étudiants qui n'ont aucune évaluation à ce jour ?

```
SELECT *
FROM Etudiant
WHERE NOT EXISTS
  (
    SELECT 1
    FROM Resultat
    WHERE Etudiant.matricule =
            Resultat.matricule
  )
```

Note

« SELECT 1 » car, peu importe ce qu'on sélectionne, l'inexistence sera invalidée.

Étudiant

matricule	nom	adresse
15113150	Paul	>ᐃᖃᓗᑐᖅ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

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

	matricule ▲ 1	nom	adresse
1	15113870	Mohamed	Tadoussac

EXEMPLE DE QUANTIFICATION 2

Quelles sont les activités n'ayant aucun étudiant ?

```
SELECT *
FROM activite
WHERE NOT EXISTS
  (
    SELECT 'toto'
    FROM resultat
    WHERE resultat.activite =
          activite.sigle
  )
```

Activité

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

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

	sigle	titre
1	IGE401	Gestion de projets

EXEMPLE DE QUANTIFICATION 3

Quels sont les étudiants sans résultats en 2013 ?

```
SELECT *
FROM Etudiant
WHERE NOT EXISTS
 (
 SELECT matricule
 FROM Resultat
 WHERE Etudiant.matricule =
          Resultat.matricule
   AND NOT
    (trimestre BETWEEN
     '20131' and '20133')
 )
```

FAUX!

Étudiant

matricule	nom	adresse
15113150	Paul	>ᐃᖅᓂᑐᖅ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

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

matricule	nom	adresse
15113150	Paul	>ᐃᖅᓂᑐᖅ
15113870	Mohamed	Tadoussac

On obtient les étudiants dont TOUS les résultats sont en 2013!

(y compris Mohamed qui n'a… acun réultat, bien sûr)

EXEMPLE DE QUANTIFICATION 3 (BIS)

Quels sont les étudiants sans résultats en 2013 ?

```
SELECT *
FROM Etudiant
WHERE NOT EXISTS
 (
 SELECT matricule
 FROM Resultat
 WHERE Etudiant.matricule =
          Resultat.matricule
   AND
    (trimestre BETWEEN
     '20131' and '20133')
 )
```

JUSTO!

Étudiant

matricule	nom	adresse
15113150	Paul	>ᐃᖃᓗᐃᑦ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

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

	matricule	nom	adresse
1	15112354	Éliane	Blanc-Sablon
2	15113870	Mohamed	Tadoussac

On obtient les étudiants la bonne réponse.

(y compris Mohamed qui n'a… aucun résultat en 2013, bien sûr)

SELECT - QUANTIFICATION DE COMPARAISON

- Des raccourcis simples et efficients ciblant les opérateurs relationnel simples :
  - $\exists\ x \in ensemble.\ expression\ opComp\ x$
    - se traduit en SQL par : *expression opComp* ANY *ensemble*
  - $\forall\ x \in ensemble.\ expression\ opComp\ x$
    - se traduit en SQL par : *expression opComp* ALL *ensemble*
  - *opComp* ::=
    = | <> | < | <= | > | >=
  - *ensemble* ::=
    (*requête*) | (*listeDeValeurs*)

EXEMPLE DE QUANTIFICATION 4

Quels sont les étudiants dont toutes les notes sont supérieurs ou égales à 60 ?

```
SELECT matricule
FROM Etudiant
WHERE 60 <= ALL
  (
  SELECT note
  FROM Resultat
  WHERE Etudiant.matricule =
    Resultat.matricule
  )
```

Étudiant

matricule	nom	adresse
15113150	Paul	>ᐃᖏᓂᑐᖅ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

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

matricule	nom	adresse
15113150	Paul	>ᐃᖏᓂᑐᖅ
15113870	Mohamed	Tadoussac

On obtient la bonne réponse.

(y compris Mohamed qui n'a… aucun résultat en 2013, bien sûr)

JOINTURES EXTERNES

- Rappel du contexte
- Syntaxe des jointures externes
- Sémantique des jointures externes

LE LANGAGE SQL
SELECT (SYNTAXE SIMPLIFIÉE) - LISTEDEJOINTURES

ListeDeJointures ::=
  DenotationTable [[AS] *alias*] [*Jointure* ...]
DenotationTable ::=
  nomTable | (*Requête*) | *autresDénotations*
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 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 EXTERNE

Jointure_externe ::=

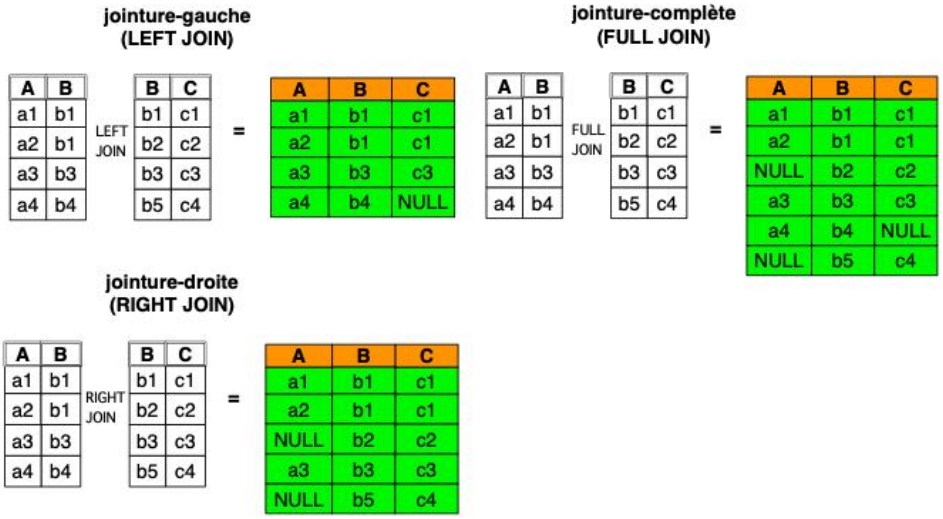
jExterne JOIN *DenotationTable* [[AS] *alias*] *qualificationJ*

jExterne ::=

LEFT [OUTER] | RIGHT [OUTER] | FULL [OUTER]

- Une jointure externe permet de ne pas « perdre » les données des tuples sans correspondant en affectant des NULL aux attributs de valeur inconnue.
- Le mot OUTER est superfétatoire (sic).
- Le mot AS permet de (re)nommer la *dénotationTable* en même temps.

LE LANGAGE SQL
ILLUSTRATION JOINTURE EXTERNE




```
CREATE TABLE R (A CHAR(2) PRIMARY KEY, B CHAR(2));
CREATE TABLE S (B CHAR(2) PRIMARY KEY, C CHAR(2));
INSERT INTO R VALUES
  ('a1', 'b1'),
  ('a2', 'b1'),
  ('a3', 'b3'),
  ('a4', 'b4');
INSERT INTO S VALUES
  ('b1', 'c1'),
  ('b2', 'c2'),
  ('b3', 'c3'),
  ('b5', 'c4');
SELECT * FROM R LEFT JOIN S USING(B);
SELECT * FROM R RIGHT JOIN S USING(B);
SELECT * FROM R FULL JOIN S USING(B);
```

```
SELECT C.nom, T.tel
FROM
  C RIGHT JOIN T ON C.id = T.id
```

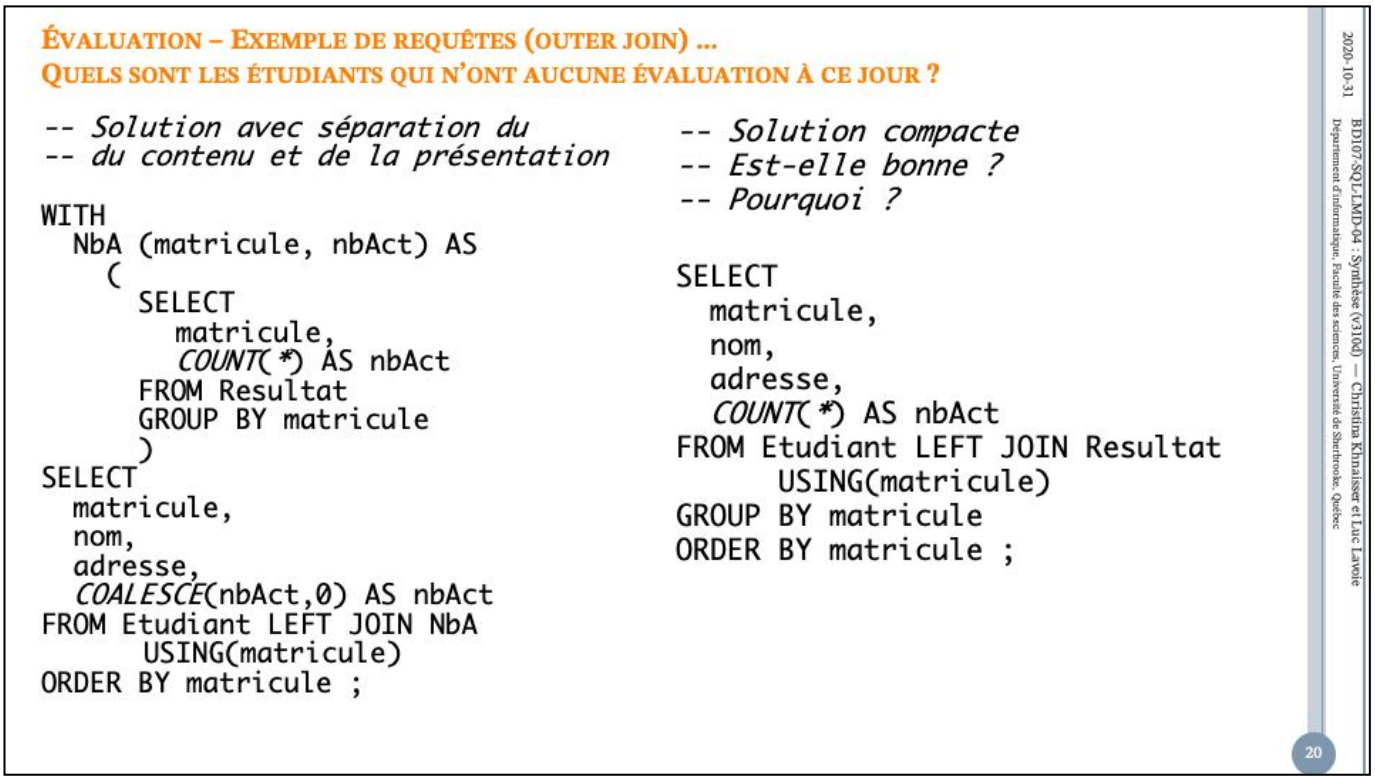
est équivalent à :

```
SELECT C.nom, T.tel
FROM
  T LEFT JOIN C ON C.id = T.id
```

```
SELECT C.nom, T.tel
FROM
  C FULL JOIN T ON C.id = T.id
```

est équivalent à :

```
SELECT C.nom, T.tel
FROM
  C RIGHT JOIN T ON C.id = T.id
UNION
SELECT C.nom, T.tel
FROM
  C LEFT JOIN T  ON C.id = T.id
```



ÉVALUATION – EXEMPLE DE REQUÊTES (OUTER JOIN) ...
QUELS SONT LES ÉTUDIANTS QUI N'ONT AUCUNE ÉVALUATION À CE JOUR ?

```
-- Solution avec séparation du
-- du contenu et de la présentation

WITH
  NbA (matricule, nbAct) AS
    (
      SELECT
        matricule,
        COUNT(*) AS nbAct
      FROM Resultat
      GROUP BY matricule
      )
SELECT
  matricule,
  nom,
  adresse,
  COALESCE(nbAct,0) AS nbAct
FROM Etudiant LEFT JOIN NbA
      USING(matricule)
ORDER BY matricule ;
```

```
-- Solution compacte
-- Est-elle bonne ?
-- Pourquoi ?

SELECT
  matricule,
  nom,
  adresse,
  COUNT(*) AS nbAct
FROM Etudiant LEFT JOIN Resultat
      USING(matricule)
GROUP BY matricule
ORDER BY matricule ;
```



```
-- Solution stricte
WITH
  NbA AS
    (
      SELECT matricule, COUNT(*) AS nbAct
      FROM Resultat
      GROUP BY matricule
      )
SELECT matricule, nom, adresse, COALESCE (nbAct,0) AS nbAct
FROM Etudiant LEFT JOIN NbA USING(matricule)
ORDER BY matricule ;

-- solution avec induction, mais est-elle bonne ? Pourquoi ?
SELECT matricule, nom, adresse, COUNT(*) AS nbAct
FROM Etudiant LEFT JOIN Resultat USING(matricule)
GROUP BY matricule
ORDER BY matricule ;
```

```
-- solution avec induction, mais est-elle bonne ? Pourquoi ?
SELECT matricule, nom, adresse, COUNT(activite) AS nbAct
FROM Etudiant LEFT JOIN Resultat USING(matricule)
GROUP BY matricule
ORDER BY matricule ;
```

SUITE ...

```
-- Solution compacte
-- Est-elle bonne ?
-- Pourquoi ?

SELECT
  matricule,
  nom,
  adresse,
  COUNT(activite) AS nbAct
FROM Etudiant LEFT JOIN Resultat
      USING(matricule)
GROUP BY matricule
ORDER BY matricule ;
```

```
-- Solution compacte
-- Étonnante ?
-- Est-elle bonne ?
-- Pourquoi ?

SELECT
  matricule,
  MAX(nom) AS nom,
  MAX(adresse) AS adresse,
  COUNT(activite) AS nbAct
FROM Etudiant LEFT JOIN Resultat
      USING(matricule)
GROUP BY matricule
ORDER BY matricule ;
```

FIN

-- Simplici myrto

```
SELECT
  matricule,
  nom,
  adresse,
  COUNT(activite) AS nbAct
FROM Etudiant LEFT JOIN Resultat USING(matricule)
GROUP BY matricule, nom, adresse
ORDER BY matricule ;
```

EXEMPLE DE DONNÉES

Étudiant

matricule	nom	adresse
15113150	Paul	>ᕿᖅᓂᑐᖅ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

TypeÉvaluation

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

Activité

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

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 : >ᕿᖅᓂᑐᖅ se prononce (approximativement) Puvirnituq

RETOUR SUR LE « MODÈLE » SQL

- La complexité apparente des expressions relationnelles
- La multiplicité des jointures
- L'annulabilité
- Les doublons
- L'ordre des attributs
- Les noms d'attribut

RETOUR SUR LE « MODÈLE » SQL
LA COMPLEXITÉ APPARENTE DES EXPRESSIONS RELATIONNELLES (DÉBUT...)

Requête ::=
[*Contexte*]
SELECT *OpMode* { * | *Projection-extension* }
FROM *ListeDeJointures*
[*Restriction*]
[***Groupement***]
[*OpComplémentaire*]
[***Ordonnancement***]
[***Divers***]

OpComplémentaire ::=
INTERSECT *OpMode Requête*
| UNION *OpMode Requête*
| EXCEPT *OpMode Requête*
| *Jointure*

ListeDeJointures ::=
DenotationTable [[AS] *alias*] [*Jointure* ...]

DenotationTable ::=
nomTable | (*Requête*) | *autresDénotations*

OpMode ::=
[DISTINCT | ALL]

RETOUR SUR LE « MODÈLE » SQL
LA COMPLEXITÉ APPARENTE DES EXPRESSIONS RELATIONNELLES (... PRESQUE LA FIN)

Jointure ::=
  Jointure_naturelle | *Produit* | *Jointure_qualifiée* | *Jointure_externe*
Jointure_naturelle ::=
  NATURAL [INNER] JOIN *DenotationTable* [[AS] *alias*]
Produit ::=
  OpProduit *DenotationTable* [[AS] *alias*]
OpProduit ::=
  CROSS JOIN | ,
Jointure_qualifiée ::=
  [INNER] JOIN *DenotationTable* [[AS] *alias*] *QualificationJ*
QualificationJ ::=
  ON *conditionJ*
 | USING (*listeNomCol*)
Jointure_externe ::=
  OpJointExterne JOIN *DenotationTable* [[AS] *alias*] *qualificationJ*
OpJointExterne ::=
  LEFT [OUTER] | RIGHT [OUTER] | FULL [OUTER]

LE LANGAGE SQL
LA MULTIPLICITÉ DES JOINTURES

- Jointure naturelle
  - Égalité de tous les attributs de mêmes noms.
  - Risqué relativement à l'évolution des tables.
- Jointures externes
  - Utilisation des NULL pour ne pas « perdre » la perte d'information!
  - Discuté et discutable.
- Le CROSS JOIN
  - C'est le produit cartésien!
- Les mots INNER et OUTER
  - Superfétatoires!

RETOUR SUR LE « MODÈLE » SQL
L'ANNULABILITÉ

- Que se passe-t-il quand la condition d'une contrainte est UNKNOWN ?
  - Elle est réputée satisfaite!
- Que se passe-t-il quand la condition d'une restriction (WHERE) est UNKNOWN ?
  - Elle est réputée non satisfaite!!!

On ne peut donc s'assurer simplement du respect d'une contrainte avec une clause where équivalente, il faut s'assurer de traiter adéquatement les cas unknown explicitement

RETOUR SUR LE « MODÈLE » SQL
LES DOUBLONS

- Quand sont-ils insérés ou préservés ?
  - *Implicitement*, par une projection lors d'un SELECT (par élimination des attributs discriminant uniquement la jointure FROM)
  - *Explicitement*, lors d'une opération ensembliste (union, intersection, différence)
- Comment peut-on les enlever ?
  - En spécifiant (*généralement*) DISTINCT après SELECT, UNION, INTERSECT et EXCEPT
  - En ne spécifiant pas (*jamais*) ALL après SELECT, UNION, INTERSECT et EXCEPT

Note bien que SQL offre pas moins de 18 forme d'unions (pas toute présentées ici), il n'offre PAS l'union de collection (multiset, bag) !!!

LE LANGAGE SQL
SELECT - JOINTURES, SIMPLIFIONS!

- Jointure naturelle
  - Égalité de tous les attributs de mêmes noms
  - Risqué relativement à l'évolution des tables
- Jointures externes
  - Utilisation des NULL pour ne pas « perdre » la perte d'information (sic).
  - Discuté et discutable
- Le CROSS JOIN
  - C'est le produit cartésien!
- INNER et OUTER
  - Superfétatoires!

RETOUR SUR LE « MODÈLE » SQL
L'ORDRE DES ATTRIBUTS

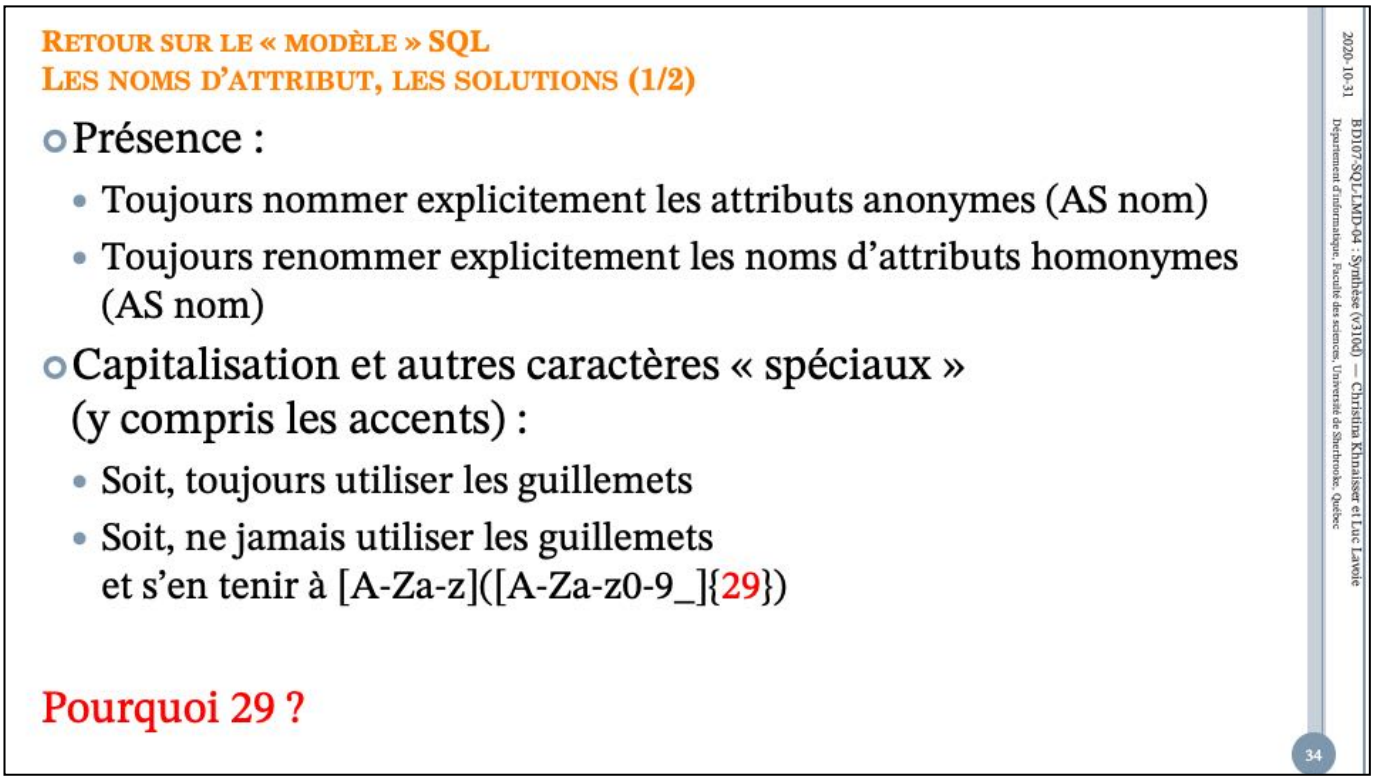
- Comment est-il déterminé ?
  - Par l'ordre des champs du CREATE TABLE.
  - Par l'ordre des expressions projection-extension du SELECT.
  - Par l'ordre des opérandes de jointure (FROM) (qui n'est donc plus commutatif).
  - Selon des règles propres aux attributs de jointure (règles à voir ultérieurement).
- Quand importe-t-il ?
  - Dans les INSERT.
  - Dans toutes les instructions portant la mention CORRESPONDING (comme l'union).

RETOUR SUR LE « MODÈLE » SQL
L'ORDRE DES ATTRIBUTS, LES SOLUTIONS

- INSERT :
  - Toujours indiquer explicitement la liste des attributs.
  - Ne pas utiliser la clause DEFAULT
    (pour cela et pour d'autres raisons).
- CORRESPONDING :
  - Ne jamais utiliser cette mention
    (qui n'est d'ailleurs pas présentée dans ce cours).

RETOUR SUR LE « MODÈLE » SQL
LES NOMS D'ATTRIBUT

- Comment peuvent-ils ne pas être présents ?
  - Expression non atomique dans la projection-extension du SELECT
  - Redondance dans les noms des opérandes des jointures
- Que se passe-t-il quand ils ne sont pas présents?
  - On ne sait pas!!!
- Qu'en est-il de la capitalisation?
  - Variable selon les SGBD




```
voir A003_identificateurs-Discipulus-SQL
        ------------------------------------------------------
--------------
        dialecte    longueur    start           extend
        ------------------------------------------------------
---------------
        DB2          30*        a, @, #, $      a, n, _, @,
#, $
        MS-SQL      116         a, _, @, #      a, n, _, @,
#, $
        mySQL        64         "tout ce qui n'est pas ambigu
!"
        Oracle       30         a               a, n, _, $, #
        PostgreSQL   31         a, _, $         a, n, _, $
        ------------------------------------------------------
--------------
```

RETOUR SUR LE « MODÈLE » SQL
LES NOMS D'ATTRIBUT, LES SOLUTIONS (2/2)

dialecte	longueur	start	extend
DB2	30	a, @, #, $	a, n, _, $, #, @
MS-SQL	116	a, _, @, #	a, n, _, $, #, @
Oracle	30	a	a, n, _, $, #
PostgreSQL	31	a, _, $	a, n, _, $
mySQL	64	*"tout ce qui n'est pas ambigu !"*	

- Note : Le $ est souvent réservé aux seules tables du catalogue.
- Conclusion : [A-Za-z]([A-Za-z0-9_]{29})

•*DB2 : aussi peu que 8, selon le SE et la nature de la table!!!

RETOUR SUR LA NOTION D'INTÉGRITÉ

- Le moment de la vérification
- La nécessité de la propagation

RETOUR SUR LA NOTION D'INTÉGRITÉ
LE MOMENT DE LA VÉRIFICATION

- Quand un contrainte est-elle vérifiée ?
  - À la fin de l'instruction ou à la fin de la transaction, selon le « mode de contrainte »
- Qu'est-ce qu'une transaction ?
  - Une suite d'instructions indivisible terminée par un COMMIT ou un ROLLBACK (plus de détails ultérieurement)
- Comment modifier le mode ?
  - SET CONSTRAINTS [MODE] IMMEDIATE
  - SET CONSTRAINTS [MODE] DEFERRED

EN GUISE DE BILAN DU SELECT

- Jointures
- Doublons
- Syntaxe

LE LANGAGE SQL
SELECT - JOINTURES, SIMPLIFIONS!

- Jointure naturelle
  - Égalité de tous les attributs de mêmes noms
  - Risqué relativement à l'évolution des tables
- Jointures externes
  - Utilisation des NULL pour ne pas « perdre » la perte d'information (sic).
  - Discuté et discutable
- Le CROSS JOIN
  - C'est le produit cartésien!
- INNER et OUTER
  - Superfétatoires!

LE LANGAGE SQL
LES DOUBLONS DU SELECT *ALL*

- Le SELECT *ALL* par défaut revient à privilégier une réponse inexacte par défaut.
- Rien ne peut justifier cela!

LE LANGAGE SQL
SELECT – REGROUPER LES OPÉRATEURS ET LEUR IMPOSER UN ORDRE

- A posteriori, cela apparait comme une fausse bonne idée en regard de la souplesse et de la clarté des expressions de l'algèbre relationnelle.
- De plus, l'optimisation des requêtes nécessite de les « reconvertir » en expressions relationnelles de toute façon!
- Pour en savoir plus, voir IGE487.

RÉFÉRENCES

- Elmasri et Navathe (4e ed.), chapitre 7
- Elmasri et Navathe (6e 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