Le modèle relationnel (partie II)

Informatique et Statistique 2A 3ème année

Olivier Caron[1]

[1]École d'ingénieurs Polytech Lille
Université de Lille

21 septembre 2023

Plan

Plan

Partie I

- Le modèle relationnel

Plan

Partie I

- Le modèle relationnel
- Le langage SQL, partie DDL (Data Definition Language)

Plan

Partie I

- Le modèle relationnel
- Le langage SQL, partie DDL (Data Definition Language)
- Passage d'un schéma conceptuel à un schéma relationnel

Plan

Partie I

- Le modèle relationnel
- Le langage SQL, partie DDL (Data Definition Language)
- Passage d'un schéma conceptuel à un schéma relationnel

Partie II

Plan

Partie I

- Le modèle relationnel
- Le langage SQL, partie DDL (Data Definition Language)
- Passage d'un schéma conceptuel à un schéma relationnel

Partie II

- L'algèbre relationnelle (base de requêtes)

Plan

Partie I

- Le modèle relationnel
- Le langage SQL, partie DDL (Data Definition Language)
- Passage d'un schéma conceptuel à un schéma relationnel

Partie II

- L'algèbre relationnelle (base de requêtes)
- Le langage SQL, partie requêtes et modification de données

Algèbre relationnelle

- Egalement définie par Codd (1970)

Algèbre relationnelle

- Egalement définie par Codd (1970)
- Basée sur des opérateurs algébriques simples

Algèbre relationnelle

- Egalement définie par Codd (1970)
- Basée sur des opérateurs algébriques simples
- Construire des requêtes par composition de ces opérateurs

La commande Select du langage SQL

- Supporte l'algèbre relationnelle de Codd via une seule commande

La commande Select du langage SQL

- Supporte l'algèbre relationnelle de Codd via une seule commande
- Plusieurs représentations possibles pour une même requête (évolutions successives de la norme SQL)

La commande Select du langage SQL

- Supporte l'algèbre relationnelle de Codd via une seule commande
- Plusieurs représentations possibles pour une même requête (évolutions successives de la norme SQL)
- En interne, le SGBD traduit la requête SQL en une composition **optimisée** des opérateurs algébriques.

Exemple base

Vins1	Numéro	Cru	Millésime	Degré
	100	Chablis	1974	12
	110	Mercurey	1978	13
	120	Meursault	1977	12

Vins2	Numéro	Cru	Millésime	Degré
	100	Chablis	1974	12
	200	Sancerre	1974	12

Consultation simple d'une table

```
-- requête usuelle :
select * from vins1 ;

-- définition explicites des colonnes de la table :
-- ordre des colonnes quelconque
select numero, cru, millesime, degre from vins1 ;

-- utilisation des noms de table
select vins1.* from vins1

-- utilisation d'une variable pour le nom de la table :
select tablevins.* from vins1 tablevins ;
```

Opérateur d'union

Définition

L'union de 2 relations R et S de même schéma est une relation T de même schéma contenant l'ensemble des tuples appartenant à R ou S

Opérateur d'union

Définition

L'union de 2 relations R et S de même schéma est une relation T de même schéma contenant l'ensemble des tuples appartenant à R ou S

- Notation texte : $T = R \cup S$ ou $T = union(R, S)$

Exemple Union

```
select * from vins1
union
select * from vins2 ;
```

Vins3 : Vins1 ∪ Vins2	Numéro	Cru	Millésime	Degré
	100	Chablis	1974	12
	110	Mercurey	1978	13
	120	Meursault	1977	12
	200	Sancerre	1974	12

Opérateur de différence

Définition

la différence *de 2 relations R et S de même schéma est une relation T de même schéma contenant l'ensemble des tuples appartenant à R et n'appartenant pas à S*

Opérateur de différence

Définition

la différence *de 2 relations R et S de même schéma est une relation T de même schéma contenant l'ensemble des tuples appartenant à R et n'appartenant pas à S*

- Notation texte : $T = R - S$ ou $T = minus(R, S)$

Exemple différence

```
select * from vins1
except
select * from vins2 ;
```

Vins4 : Vins1 − Vins2	Numéro	Cru	Millésime	Degré
	110	Mercurey	1978	13
	120	Meursault	1977	12

Produit cartésien

- Opérateur intermédiaire (**pas de sens en soi**)

Produit cartésien

- Opérateur intermédiaire (**pas de sens en soi**)
- Schéma quelconque de deux relations

Produit cartésien

- Opérateur intermédiaire (**pas de sens en soi**)
- Schéma quelconque de deux relations
- Notation texte : $T = R \times S$ ou $T = product(R, S)$

Exemple produit cartésien

Vignoble	nom	vin
	Nicolas	Bourgogne
	Martin	Bordeaux

```
select vins2.* vignoble.* from vins2, vignoble
```

Vins5 : Vins-2×Vignoble	Num.	Cru	Millé.	Deg.	nom	vin
	100	chablis	1974	12	Nicolas	Bourgogne
	200	Sancerre	1974	12	Martin	Bordeaux
	100	chablis	1974	12	Martin	Bordeaux
	200	Sancerre	1974	12	Nicolas	Bourgogne

- Opérateur très gourmand (espace disque)

Opérateur unaire de projection

Définition

la projection *d'une relation R de schéma $R(A_1, A_2, \ldots, A_n)$ sur les attributs $A_{i_1}, A_{i_2}, \ldots, A_{i_p})$ $(p < n)$ est une relation $R'(A_{i_1}, A_{i_2}, \ldots, A_{i_p})$ dont les tuples sont obtenus par élimination des valeurs des attributs de R n'appartenant pas à R' et par suppression des tuples en double.*

Opérateur unaire de projection

Définition

la projection *d'une relation R de schéma $R(A_1, A_2, \ldots, A_n)$ sur les attributs $A_{i_1}, A_{i_2}, \ldots, A_{i_p}$) $(p < n)$ est une relation $R'(A_{i_1}, A_{i_2}, \ldots, A_{i_p})$ dont les tuples sont obtenus par élimination des valeurs des attributs de R n'appartenant pas à R' et par suppression des tuples en double.*

- Notation texte : $T = \prod_{x_1,\ldots,x_n}(R)$ ou $T = project(R/x_1, \ldots, x_n)$

Exemple projection

Vins3 :	Numéro	Cru	Millésime	Degré
	100	Chablis	1974	12
	110	Mercurey	1978	13
	120	Meursault	1977	12
	200	Sancerre	1974	12

select distinct millesime , degre **from** vins3
— *distinct nécessaire pour algèbre relationnelle pure*

Vins6 : project(Vins3/Millésime,degré)	Millésime	Degré
	1974	12
	1978	13
	1977	12

Opérateur unaire de restriction ou sélection

Définition

La restriction (ou sélection) de la relation R par une qualification Q est une relation R' de même schéma dont les tuples sont ceux de R satisfaisant la qualification Q.

Opérateur unaire de restriction ou sélection

Définition

La restriction (ou sélection) de la relation R par une qualification Q est une relation R' de même schéma dont les tuples sont ceux de R satisfaisant la qualification Q.

- La qualification peut être exprimée à l'aide de constantes, comparateurs arithmétiques, opérateurs logiques

Opérateur unaire de restriction ou sélection

Définition

La restriction (ou sélection) de la relation R par une qualification Q est une relation R' de même schéma dont les tuples sont ceux de R satisfaisant la qualification Q.

- La qualification peut être exprimée à l'aide de constantes, comparateurs arithmétiques, opérateurs logiques
- Notation texte : $T = \sigma_Q(R)$ ou $T = restrict(R/Q)$

Exemple restriction

Vins3 :	Numéro	Cru	Millésime	Degré
	100	Chablis	1974	12
	110	Mercurey	1978	13
	120	Meursault	1977	12
	200	Sancerre	1974	12

```
select * from vins3
where degre=12 and millesime >1974
```

Vins7 :	Numéro	Cru	Millésime	Degré
restrict(Vins3/degré=12 and Millésime>1974)				
	120	Meursault	1977	12

Expression d'une restriction en SQL

- Introduction clause `Where` après clause `From`

Expression d'une restriction en SQL

- Introduction clause `Where` après clause `From`
- Ordre des clauses imposé

Expression d'une restriction en SQL

- Introduction clause `Where` après clause `From`
- Ordre des clauses imposé
- Utilisation des opérateurs booléens : `and`, `or`, `not`

Expression d'une restriction en SQL

- Introduction clause `Where` après clause `From`
- Ordre des clauses imposé
- Utilisation des opérateurs booléens : `and`, `or`, `not`
- Comparaison de chaînes, dates, d'entiers,...

Traitement de chaînes (1/2)

- Syntaxe très simple : `ch1 = ch2`
- ou bien opérateur `LIKE` avec caractères spéciaux :
  - '%' remplace de 0 à plusieurs caractères
  - '_' remplace exactement un caractère.
- Exemple :

```
vins=> select * from vins3 where cru like 'M%' ;
 numero |     cru     | millesime | degre
--------+-------------+-----------+-------
    120 | Meursault   |      1977 |    12
    110 | Mercurey    |      1978 |    13
(2 rows)
```

Traitement de chaînes (2/2)

- Opérateur de comparaison '>', '<', ...(ordre lexicographique)
- Opérateur de concaténation ||, fonctions prédéfinies (ex : upper)
- Exemple :

```
vins=> select upper(cru || ' ' || millesime) as nom
       from vins3 ;
          nom
-------------------------
 MEURSAULT 1977
 SANCERRE 1974
 CHABLIS 1974
 MERCUREY 1978
(4 rows)
```

Comparaison de valeurs

- Introduction clause `BETWEEN`
- Applicable à tout type (integer, chaîne, date, ...)

```
vins=> select * from vins3
       where millesime between 1974 and 1977 ;
 numero |     cru     | millesime | degre
--------+-------------+-----------+-------
    120 | Meursault   |      1977 |    12
    200 | Sancerre    |      1974 |    12
    100 | Chablis     |      1974 |    12
(3 rows)
```

Les opérateurs de base

- Simplicité : seulement 5 opérateurs !

Les opérateurs de base

- Simplicité : seulement 5 opérateurs !
- Composition de ces opérateurs possible : tout résultat est une relation

Les opérateurs de base

- Simplicité : seulement 5 opérateurs !
- Composition de ces opérateurs possible :
  tout résultat est une relation
- Ces opérateurs répondent à la majorité des requêtes (sauf calcul)

Les opérateurs de base

- Simplicité : seulement 5 opérateurs !
- Composition de ces opérateurs possible :
  tout résultat est une relation
- Ces opérateurs répondent à la majorité des requêtes (sauf calcul)
- Introduction d'opérateurs additionnels

L'intersection

Définition

L'intersection *de deux relations R et S de même schéma est une relation T de même schéma contenant les tuples appartenant à la fois à R et S*

- Notation texte : $T = (R \cap S)$ ou $T = inter(R, S)$
- Création opérateur : $R \cap S = R - (R - S) = S - (S - R)$

```
select * from R
intersect
select * from S ;
```

La jointure

Définition

La jointure de deux relations R et S selon une qualification multi-attributs Q est l'ensemble des tuples du produit cartésien $R \times S$ satisfaisant la qualification Q

La jointure

Définition

La jointure de deux relations R et S selon une qualification multi-attributs Q est l'ensemble des tuples du produit cartésien $R \times S$ satisfaisant la qualification Q

- Notation texte : $T = join(R, S, Q)$

La jointure

Définition

La jointure de deux relations R et S selon une qualification multi-attributs Q est l'ensemble des tuples du produit cartésien $R \times S$ satisfaisant la qualification Q

- Notation texte : $T = join(R, S, Q)$
- Création opérateur : $join(R, S, Q) = restrict(R \times S / Q)$

Exemple jointure (1/2)

- Tables Parent et Enfant :

Parent		**Enfant**		
nom	numéro	num_e	prénom	num_père
dupont	1	1	jérome	1
durant	2	2	alain	2
		3	pierre	1

Exemple jointure (1/2)

- Tables Parent et Enfant :

Parent		**Enfant**		
nom	numéro	num_e	prénom	num_père
dupont	1	1	jérome	1
durant	2	2	alain	2
		3	pierre	1

- join(Parent, Enfant, numéro=num_père) :

nom	numéro	num_e	prénom	num_père
dupont	1	1	jérome	1
durant	2	2	alain	2
dupont	1	3	pierre	1

Exemple jointure (2/2)

```
-- compatible norme SQL 2 et plus
select parent.*, enfant.*
from parent join enfant on numero=num_pere ;

-- compatible norme SQL 1 et plus
-- utilisation explicite du produit cartésien
-- avec opérateur de restriction
select parent.*, enfant.*
from parent, enfant              -- produit cartésien
where numero=num_pere ;          -- restriction
```

Important

privilégiez la première version !

Quelques remarques

- Pour un même résultat, plusieurs requêtes sont possibles

Quelques remarques

- Pour un même résultat, plusieurs requêtes sont possibles
- La jointure est l'opérateur le plus coûteux

Quelques remarques

- Pour un même résultat, plusieurs requêtes sont possibles
- La jointure est l'opérateur le plus coûteux
- Idéal de faire des sélections et restrictions avant la ou les jointures

Et des opérateurs de calcul

- Existent dans la plupart des S.G.B.D.

Et des opérateurs de calcul

- Existent dans la plupart des S.G.B.D.
- Donnent un résultat de type relation !

Et des opérateurs de calcul

- Existent dans la plupart des S.G.B.D.
- Donnent un résultat de type relation !
- Quelques illustrations :

Et des opérateurs de calcul

- Existent dans la plupart des S.G.B.D.
- Donnent un résultat de type relation !
- Quelques illustrations :
  - L'opérateur $count(R)$: : calcul du nombre de lignes d'une relation R

Et des opérateurs de calcul

- Existent dans la plupart des S.G.B.D.
- Donnent un résultat de type relation !
- Quelques illustrations :
  - L'opérateur $count(R)$: : calcul du nombre de lignes d'une relation R
  - L'opérateur $sum(R(A_i))$ calcul de la somme cumulée des valeurs d'un attribut.

Et des opérateurs de calcul

- Existent dans la plupart des S.G.B.D.
- Donnent un résultat de type relation !
- Quelques illustrations :
  - L'opérateur $count(R)$: : calcul du nombre de lignes d'une relation R
  - L'opérateur $sum(R(A_i))$ calcul de la somme cumulée des valeurs d'un attribut.
  - et aussi `avg`, `max`, `min`, ...

Syntaxe très partielle commande Select

```
SELECT [ ALL | DISTINCT [ ON (expression [, ...]) ]]
  * | expression [ AS output_name ] [, ...]
  [ FROM from_item [, ...] ]
  [ WHERE condition ]
  [ GROUP BY expression [, ...] ]
  [ HAVING condition [, ...] ]
  [ { UNION | INTERSECT | EXCEPT [ ALL ] } select ]
  [ ORDER BY expression [ ASC | DESC |
    USING operator ] [, ...] ]
  [ FOR UPDATE [ OF tablename [, ...] ] ]
  [ LIMIT { count | ALL } [ { OFFSET | , } start ]]
```

État de la base exemple (1/3)

Auteur :

num_a	nom
1	Albert Uderzo
2	Victor Hugo
3	J.K. Rowling

Editeur :

num_e	nom	ville
1	Albert-René	Bruxelles
2	Gallimard	Paris
3	Folio	Paris

État de la base exemple (2/3)

```
livre :
num_l |                  titre                   | auteur
------+------------------------------------------+--------
    1 | Le fils d'Astérix                        |      1
    2 | Les misérables                           |      2
    3 | Notre dame de Paris                      |      2
    4 | Harry Potter à l'école des sorciers      |      3
    5 | Harry Potter et la chambre des secrets   |      3

edite_par :  num_l | num_e | date_edition
            -------+-------+--------------
                1 |     1 | 1998-03-24
                2 |     3 | 1940-02-02
                3 |     2 | 1967-06-12
                4 |     2 | 1999-03-01
                5 |     2 | 2000-02-01
```

État de la base exemple (3/3)

utilisateur :

num_u	nom	prenom
1	Caron	Olivier
2	Janot	Stéphane
3	Seynhaeve	Franck
4	Duthilleul	Jean-Michel

emprunte :

num_l	num_u
1	1
2	4
4	1

reserve

num_l	num_u
1	2
4	2

Présentation des données (1/2)

- Renommage des colonnes et expressions à afficher (as `nomColonneResultat`)

Présentation des données (1/2)

- Renommage des colonnes et expressions à afficher (as `nomColonneResultat`)
- Clause `distinct`, évite les doublons

Présentation des données (1/2)

- Renommage des colonnes et expressions à afficher (as `nomColonneResultat`)
- Clause `distinct`, évite les doublons
- Ordre d'affichage des colonnes

Présentation des données (1/2)

- Renommage des colonnes et expressions à afficher (as `nomColonneResultat`)
- Clause `distinct`, évite les doublons
- Ordre d'affichage des colonnes
- Ordre d'affichage des lignes, clause `Order By`

Présentation des données (1/2)

- Renommage des colonnes et expressions à afficher (as `nomColonneResultat`)
- Clause `distinct`, évite les doublons
- Ordre d'affichage des colonnes
- Ordre d'affichage des lignes, clause `Order By`
- Ordre des lignes multi-critères

Présentation des données (1/2)

- Renommage des colonnes et expressions à afficher (as `nomColonneResultat`)
- Clause `distinct`, évite les doublons
- Ordre d'affichage des colonnes
- Ordre d'affichage des lignes, clause `Order By`
- Ordre des lignes multi-critères
- Aucun impact sur le traitement algébrique des requêtes

Présentation des données (2/2)

- Syntaxe :
  ORDER BY expression [ASC | DESC |
  USING operator] [, ...]
- Exemple :

```
select * from livre order by auteur DESC, titre ASC ;
```

num_l	titre	auteur
4	Harry Potter à l'école des sorciers	3
5	Harry Potter et la chambre des secrets	3
2	Les misérables	2
3	Notre dame de Paris	2
1	Le fils d'Astérix	1

Opérations de calcul

- Opérateurs arithmétiques : $+, -, \ldots$
- Exemple :

```
select now()-date_edition as duree, num_l
from edite_par ;
```

```
        duree          | num_l
-----------------------+-------
 1423 days 17:30:01    |     1
 22658 days 16:30:01   |     2
 12666 days 17:30:01   |     3
 1081 days 17:30:01    |     4
 744 days 17:30:01     |     5
```

- Expressions arithmétiques applicables dans la clause `where`

Fonctions de calcul

- Syntaxe : `nomFonction(nomColonne)` ou `nomFonction(*)`

Fonctions de calcul

- Syntaxe : `nomFonction(nomColonne)` ou `nomFonction(*)`
- le résultat est stocké dans une colonne correspondant au nom de la fonction (sauf si renommage)

Fonctions de calcul

- Syntaxe : `nomFonction(nomColonne)` ou `nomFonction(*)`
- le résultat est stocké dans une colonne correspondant au nom de la fonction (sauf si renommage)
- Toujours une ligne résultat.

Fonctions de calcul

- Syntaxe : `nomFonction(nomColonne)` ou `nomFonction(*)`
- le résultat est stocké dans une colonne correspondant au nom de la fonction (sauf si renommage)
- Toujours une ligne résultat.
- Fonctions standards : count, min, max, avg, sum

Fonctions de calcul - exemples

```
select count(*) as nombre from livre ;
```

```
 nombre
-------
     5
```

```
select min(num_l),max(num_l),avg(num_l),sum(num_l)
from livre;
```

```
 min | max |        avg        | sum
-----+-----+-------------------+-----
   1 |   5 | 3.0000000000      |  15
```

Calcul sur des groupes de lignes (1/2)

- Sélectionner des lignes pour appliquer un calcul

Calcul sur des groupes de lignes (1/2)

- Sélectionner des lignes pour appliquer un calcul
- Introduction clause `Group By`

```
select auteur, count(*) as nbre_par_auteur
from livre group by auteur ;
```

```
 auteur | nbre_par_auteur
--------+-----------------
      1 |               1
      2 |               2
      3 |               2
```

Calcul sur des groupes de lignes (1/2)

- Sélectionner des lignes pour appliquer un calcul
- Introduction clause `Group By`

```
select auteur, count(*) as nbre_par_auteur
from livre group by auteur ;
```

```
 auteur | nbre_par_auteur
--------+-----------------
      1 |               1
      2 |               2
      3 |               2
```

- Seules les colonnes figurant dans un `group by` peuvent apparaître dans la clause `select`.

Calcul sur des groupes de lignes (2/2)

- Imposer une condition aux groupes formés par la clause `Group By`

Calcul sur des groupes de lignes (2/2)

- Imposer une condition aux groupes formés par la clause `Group By`
- Introduction clause `Having`

Calcul sur des groupes de lignes (2/2)

- Imposer une condition aux groupes formés par la clause `Group By`
- Introduction clause `Having`
- Exemple :

```
select auteur, count(*) as nbre_par_auteur
from livre group by auteur having count(*)>1 ;
```

```
 auteur | nbre_par_auteur
--------+-----------------
      2 |               2
      3 |               2
```

Calcul sur des groupes de lignes (2/2)

- Imposer une condition aux groupes formés par la clause `Group By`
- Introduction clause `Having`
- Exemple :

```
select auteur, count(*) as nbre_par_auteur
from livre group by auteur having count(*)>1 ;
```

```
 auteur | nbre_par_auteur
--------+-----------------
      2 |               2
      3 |               2
```

- Ne pas confondre avec clause `where`

Jointures sur une même table

- Appelée également auto-jointure

Jointures sur une même table

- Appelée également auto-jointure
- Exemple : liste de couples de livres ayant le même auteur

```
select l1.titre, l2.titre
from livre l1 join livre l2 on l1.auteur=l2.auteur
where l1.titre > l2.titre ; -- diviser par 2
                              le nbre de couples
```

```
                titre          |                titre
-------------------------------+-----------------------------
 Notre dame de Paris           | Les misérables
 Harry Potter et la chambre ... | Harry Potter à l'école ...
```

Les jointures à la SQL/2

- Nouvelles possibilités d'expression de jointures

Les jointures à la SQL/2

- Nouvelles possibilités d'expression de jointures
- Non encore implémenté par tous les SGBD

Les jointures à la SQL/2

- Nouvelles possibilités d'expression de jointures
- Non encore implémenté par tous les SGBD
- Les expressions de jointures sont exprimés dans la clause `from`

Les jointures à la SQL/2

- Nouvelles possibilités d'expression de jointures
- Non encore implémenté par tous les SGBD
- Les expressions de jointures sont exprimés dans la clause `from`
- Distinction de jointures : inner join (défaut), left outer join, right outer join, full outer join

```
insert into livre values (6, 'Le livre inconnu', null) ;
insert into auteur values (4, 'Paltoquet') ;
```

Jointure SQL/2 classique

```
select titre , nom
from livre inner join auteur on livre.auteur=auteur.num_a
```

```
                titre                      |      nom
-------------------------------------------+---------------
 Le fils d'Astérix                         | Albert Uderzo
 Les misérables                            | Victor Hugo
 Notre dame de Paris                       | Victor Hugo
 Harry Potter à l'école des sorciers       | J.K. Rowling
 Harry Potter et la chambre des secrets    | J.K. Rowling
(5 rows)
```

Jointure externe gauche

```
select titre , nom
from livre left outer join auteur on livre.auteur=auteur.num_a
```

```
                titre                    |      nom
-----------------------------------------+---------------
 Le fils d'Astérix                       | Albert Uderzo
 Les misérables                          | Victor Hugo
 Notre dame de Paris                     | Victor Hugo
 Harry Potter à l'école des sorciers     | J.K. Rowling
 Harry Potter et la chambre des secrets  | J.K. Rowling
 le livre inconnu                        |
(6 rows)
```

Jointure externe droite

```
select l.titre, a.nom
from livre l right outer join auteur a on l.auteur=a.num_a
```

```
                titre                    |      nom
-----------------------------------------+---------------
 Le fils d'Astérix                       | Albert Uderzo
 Les misérables                          | Victor Hugo
 Notre dame de Paris                     | Victor Hugo
 Harry Potter à l'école des sorciers     | J.K. Rowling
 Harry Potter et la chambre des secrets  | J.K. Rowling
                                         | Paltoquet
(6 rows)
```

Jointure externe complète

```
select l.titre, a.nom
from livre l full outer join auteur a on l.auteur=a.num_a
```

```
                titre                    |      nom
-----------------------------------------+---------------
 Le fils d'Astérix                       | Albert Uderzo
 Les misérables                          | Victor Hugo
 Notre dame de Paris                     | Victor Hugo
 Harry Potter à l'école des sorciers     | J.K. Rowling
 Harry Potter et la chambre des secrets  | J.K. Rowling
 le livre inconnu                        |
                                         | Paltoquet
(7 rows)
```

Multi-jointures SQL/2

```
select  u.nom, l.titre
from utilisateur u join emprunte e on u.num_u= e.num_u
     join  livre l on e.num_l=l.num_l
```

```
    nom     |                titre
------------+-------------------------------------
 Caron      | Le fils d'Astérix
 Duthilleul | Les misérables
 Caron      | Harry Potter à l'école des sorciers
(3 rows)
```

Remarques sur la syntaxe des jointures SQL/2 (1/2)

- LEFT, RIGHT et FULL impliquent une jointure externe

Remarques sur la syntaxe des jointures SQL/2 (1/2)

- LEFT, RIGHT et FULL impliquent une jointure externe
- Jointures croisées :

Remarques sur la syntaxe des jointures SQL/2 (1/2)

- LEFT, RIGHT et FULL impliquent une jointure externe
- Jointures croisées :
  - Syntaxe : `from T1 cross join T2`

Remarques sur la syntaxe des jointures SQL/2 (1/2)

- LEFT, RIGHT et FULL impliquent une jointure externe
- Jointures croisées :
  - Syntaxe : `from T1 cross join T2`
  - Est équivalent à un produit cartésien

Remarques sur la syntaxe des jointures SQL/2 (1/2)

- LEFT, RIGHT et FULL impliquent une jointure externe
- Jointures croisées :
  - Syntaxe : `from T1 cross join T2`
  - Est équivalent à un produit cartésien
  - Est équivalent à `from T1 inner join T2 on true`

Remarques sur la syntaxe des jointures SQL/2 (2/2)

- Les syntaxes :

```
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } join T2
     on boolean_expression
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } join T2
     using liste_nom_colonne
T1 NATURAL { [INNER] | { LEFT | RIGHT | FULL } [OUTER] }
 join T2
```

Remarques sur la syntaxe des jointures SQL/2 (2/2)

- Les syntaxes :

```
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } join T2
     on boolean_expression
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } join T2
     using liste_nom_colonne
T1 NATURAL { [INNER] | { LEFT | RIGHT | FULL } [OUTER] }
 join T2
```

- `USING a` équivalent à `on t1.a = t2.a`

Remarques sur la syntaxe des jointures SQL/2 (2/2)

- Les syntaxes :

```
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } join T2
     on boolean_expression
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } join T2
     using liste_nom_colonne
T1 NATURAL { [INNER] | { LEFT | RIGHT | FULL } [OUTER] }
 join T2
```

- `USING a` équivalent à `on t1.a = t2.a`
- `USING (a,b)` équivalent à `on t1.a=t2.a and t1.b=t2.b`

Remarques sur la syntaxe des jointures SQL/2 (2/2)

- Les syntaxes :

```
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } join T2
     on boolean_expression
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] } join T2
     using liste_nom_colonne
T1 NATURAL { [INNER] | { LEFT | RIGHT | FULL } [OUTER] }
 join T2
```

- `USING a` équivalent à `on t1.a = t2.a`
- `USING (a,b)` équivalent à `on t1.a=t2.a and t1.b=t2.b`
- `NATURAL` effectue une comparaison de toutes les colonnes de même nom dans les deux tables.

Sous-requêtes non corrélatives (1/4)

- Cas 1 : une sous-requête retourne une valeur unique

Sous-requêtes non corrélatives (1/4)

- Cas 1 : une sous-requête retourne une valeur unique
- Cette valeur est exploitée dans une clause `where`

Sous-requêtes non corrélatives (1/4)

- Cas 1 : une sous-requête retourne une valeur unique
- Cette valeur est exploitée dans une clause `where`
- Exemple : liste des personnes dont le salaire est supérieur au salaire moyen du personnel

```
select nom from personne
where salaire >
      (select avg(salaire) from personne)
```

Sous-requêtes non corrélatives (1/4)

- Cas 1 : une sous-requête retourne une valeur unique
- Cette valeur est exploitée dans une clause `where`
- Exemple : liste des personnes dont le salaire est supérieur au salaire moyen du personnel

```
select nom from personne
where salaire >
      (select avg(salaire) from personne)
```

- Conformité des types (ici : `float`)

Sous-requêtes non corrélatives (2/4)

- Cas 2 : une sous-requête retourne des valeurs multiples avec la clause `in`

Sous-requêtes non corrélatives (2/4)

- Cas 2 : une sous-requête retourne des valeurs multiples avec la clause `in`
- Exemple : liste des livres empruntés et réservés

```
select titre from livre join emprunte using num_l
where livre.num_l in (select num_l from reserve) ;
```

```
                titre
-----------------------------------
 Le fils d'Astérix
 Harry Potter à l'école des sorciers
```

Sous-requêtes non corrélatives (3/4)

- Cas 3 : une sous-requête retourne des valeurs multiples avec la clause `all`

Sous-requêtes non corrélatives (3/4)

- Cas 3 : une sous-requête retourne des valeurs multiples avec la clause `all`
- Exemple : liste des personnes dont le salaire est supérieur à tous les salaires du personnel

```
select nom from personne
  where salaire >=
  ALL (select salaire from personne) ;
```

Sous-requêtes non corrélatives (3/4)

- Cas 3 : une sous-requête retourne des valeurs multiples avec la clause `all`
- Exemple : liste des personnes dont le salaire est supérieur à tous les salaires du personnel

```
select nom from personne
  where salaire >=
  ALL (select salaire from personne) ;
```

- (on peut faire plus simple...)

Sous-requêtes non corrélatives (4/4)

- Cas 4 : une sous-requête retourne des valeurs multiples avec la clause `[NOT] EXISTS`

Sous-requêtes non corrélatives (4/4)

- Cas 4 : une sous-requête retourne des valeurs multiples avec la clause `[NOT] EXISTS`
- Exemple : afficher le nom de l'auteur 1 si des livres qu'il a écrit sont dans la base.

```
select nom from auteur where num_a=1
  and exists(select * from livre where auteur=1) ;
```

Sous-requêtes non corrélatives (4/4)

- Cas 4 : une sous-requête retourne des valeurs multiples avec la clause `[NOT] EXISTS`
- Exemple : afficher le nom de l'auteur 1 si des livres qu'il a écrit sont dans la base.

```
select nom from auteur where num_a=1
  and exists(select * from livre where auteur=1) ;
```

- `exists` retourne vrai ou faux si la requête retourne des lignes ou pas.

Sous-requêtes corrélatives

- Requête et sous-requête sont liées

Sous-requêtes corrélatives

- Requête et sous-requête sont liées
- Exemple : Liste des auteurs n'ayant pas écrit de livres

```
select num_a, nom from auteur where
  not exists(select * from livre
             where auteur=auteur.num_a) ;
```

Sous-requêtes corrélatives

- Requête et sous-requête sont liées
- Exemple : Liste des auteurs n'ayant pas écrit de livres

```
select num_a, nom from auteur where
  not exists(select * from livre
             where auteur=auteur.num_a) ;
```

- Note : La requête supérieure fournit une à une les valeurs de auteur.num_a à la requête inférieure.

Expression d'une division (1/3)

- Pas de mot-clé "divide"

Expression d'une division (1/3)

- Pas de mot-clé "divide"
- Une division traduit un "pour tous/quel que soit"

Expression d'une division (1/3)

- Pas de mot-clé "divide"
- Une division traduit un "pour tous/quel que soit"
- Exemple : *"on veut connaître les utilisateurs qui ont emprunté tous les livres"*

Expression d'une division (1/3)

- Pas de mot-clé "divide"
- Une division traduit un "pour tous/quel que soit"
- Exemple : *"on veut connaître les utilisateurs qui ont emprunté tous les livres"*
- Un quel que soit en logique du premier ordre équivaut à 2 non ('not exists' en SQL).

Expression d'une division (1/3)

- Pas de mot-clé "divide"
- Une division traduit un "pour tous/quel que soit"
- Exemple : *"on veut connaître les utilisateurs qui ont emprunté tous les livres"*
- Un quel que soit en logique du premier ordre équivaut à 2 non ('not exists' en SQL).
- En langage naturel, la requête devient : "les utilisateurs pour lesquels il n'existe pas de livres, pour lesquels il n'existe pas de lien avec ces utilisateurs"

Expression d'une division (2/3)

- L'expression d'une division applique toujours le même schéma :.

```
select A1  from  R        -- ce que l'on veut en sortie :
where not exists
  select A2 from S        -- ce par quoi on divise :
  where not exists
    select * from Z       -- expressions des liens
    where Z.E1=R.A1 and Z.E2=S.A2
```

Expression d'une division (3/3)

- Illustration :

```
select * from Utilisateur u
where not exists
  select * from livre l
  where not exists
    select * from emprunte e
    where e.num_u=u.num_u and e.num_l=l.num_l
```

Alias de table

- Possibilité de donner un nom de variable à une table

Alias de table

- Possibilité de donner un nom de variable à une table
- Syntaxe : `FROM nom_table_tres_long [ AS ] nom_alias`

Alias de table

- Possibilité de donner un nom de variable à une table
- Syntaxe : `FROM nom_table_tres_long [ AS ] nom_alias`
- Permet aussi d'exprimer des sous-requêtes :
  Syntaxe : `FROM (select ...) as nom_alias`

Mais aussi...

- Tester les valeurs nulles des colonnes (syntaxe : `is null` ou `is not null`).

Mais aussi...

- Tester les valeurs nulles des colonnes (syntaxe : `is null` ou `is not null`).
- Stockage d'une requête dans des tables `select ... into table newTable from ...`

Mais aussi...

- Tester les valeurs nulles des colonnes
  (syntaxe : `is null` ou `is not null`).
- Stockage d'une requête dans des tables
  `select ... into table newTable from ...`
- Insertion de plusieurs lignes
  `insert into table nomTable select ...`

En résumé

- A la fois simple (pour une grande majorité de requêtes) et compliqué

En résumé

- A la fois simple (pour une grande majorité de requêtes) et compliqué
- Répond à la majorité des requêtes

En résumé

- A la fois simple (pour une grande majorité de requêtes) et compliqué
- Répond à la majorité des requêtes
- Langage en constante évolution

En résumé

- A la fois simple (pour une grande majorité de requêtes) et compliqué
- Répond à la majorité des requêtes
- Langage en constante évolution
- Le langage peut être très puissant (requêtes corrélatives)

En résumé

- A la fois simple (pour une grande majorité de requêtes) et compliqué
- Répond à la majorité des requêtes
- Langage en constante évolution
- Le langage peut être très puissant (requêtes corrélatives)
- La puissance est au détriment de la complexité

En résumé

- A la fois simple (pour une grande majorité de requêtes) et compliqué
- Répond à la majorité des requêtes
- Langage en constante évolution
- Le langage peut être très puissant (requêtes corrélatives)
- La puissance est au détriment de la complexité
- Une solution : le concept de vue

Les vues

- Indépendance logique des données

Les vues

- Indépendance logique des données
- Tables virtuelles issues de plusieurs tables

Les vues

- Indépendance logique des données
- Tables virtuelles issues de plusieurs tables
- La vue n'est pas enregistrée physiquement

Les vues

- Indépendance logique des données
- Tables virtuelles issues de plusieurs tables
- La vue n'est pas enregistrée physiquement
- A l'exécution d'une requête portant sur une vue, une opération de reconstitution de la vue est effectuée

Les vues

- Indépendance logique des données
- Tables virtuelles issues de plusieurs tables
- La vue n'est pas enregistrée physiquement
- A l'exécution d'une requête portant sur une vue, une opération de reconstitution de la vue est effectuée
- Une vue est considérée comme une table pour les utilisateurs

Création d'une vue

- Syntaxe :

  CREATE VIEW view **AS SELECT** query

Création d'une vue

- Syntaxe :

  CREATE VIEW view **AS SELECT** query

- Suivant les SGBD, il existe certaines restrictions sur la définition de la vue (ex : ORDER BY)

Un exemple

- Création d'une vue donnant la liste de tous les livres dont le prix est supérieur au prix moyen.

```
CREATE VIEW prix_sup_moyen
  AS SELECT n_livre, titre ,prix FROM LIVRE
     WHERE prix > (SELECT avg(prix) FROM livre) ;
```

Utilisation de la vue

- Se comporte comme une table :

```
select titre, prix from prix_sup_moyen
order by titre
```

Utilisation de la vue

- Se comporte comme une table :

```
select titre, prix from prix_sup_moyen
order by titre
```

- Suppression d'une vue :

```
drop view prix_sup_moyen ;
```

Utilisation de la vue

- Se comporte comme une table :

```
select titre, prix from prix_sup_moyen
order by titre
```

- Suppression d'une vue :

```
drop view prix_sup_moyen ;
```

- mise à jour possible dans certains cas (Oracle)

Et QBE

- Utilisé pour les SGBD simples

Et QBE

- Utilisé pour les SGBD simples
- Langage graphique, pas de syntaxe à retenir

Et QBE

- Utilisé pour les SGBD simples
- Langage graphique, pas de syntaxe à retenir
- Permet de répondre aux requêtes simples (selection, projection, jointure)

Et QBE

- Utilisé pour les SGBD simples
- Langage graphique, pas de syntaxe à retenir
- Permet de répondre aux requêtes simples (selection, projection, jointure)
- Ne dispose pas de toute la puissance de SQL