Le langage SQL (deuxième partie)

© Olivier Caron

Polytech'Lille

Les requêtes de consultation

√ Représente la majorité des requêtes

Les requêtes de consultation

- √ Représente la majorité des requêtes
- √ Encapsule complètement l'algèbre relationnel

Les requêtes de consultation

- √ Représente la majorité des requêtes
- √ Encapsule complètement l'algèbre relationnel
- √ Une seule commande !

Syntaxe 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 ]]
```

Etat 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

Etat 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

Etat 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

Consultation simple d'une table

√ Syntaxe :

```
select col1, col2, ..., coln from nomTable
```

Consultation simple d'une table

- √ Syntaxe :

  ```
  select col1, col2, ..., coln from nomTable
  ```

- √ Variante usuelle : `select * from nomTable`

Consultation simple d'une table

- Syntaxe :

  `select col1, col2, ..., coln from nomTable`

- Variante usuelle : `select * from nomTable`
- Exemple :

```
biblio=# select * from utilisateur ;
 num_u |    nom      |   prenom
-------+-------------+------------
     1 | Caron       | Olivier
     2 | Janot       | Stéphane
     3 | Seynhaeve   | Franck
     4 | Duthilleul  | Jean-Michel
(4 rows)
```

Expression d'une projection

$\surd$ Syntaxe : `select` $\texttt{col}_{i_1}$`,` $\texttt{col}_{i_p}$`, ..,` $\texttt{col}_{i_p}$ `from nomTable`

Expression d'une projection

- √ Syntaxe : `select` $\texttt{col}_{i_1}$`,` $\texttt{col}_{i_p}$`, ..,` $\texttt{col}_{i_p}$ `from nomTable`
- √ Exemple :

```
biblio=# select nom, prenom from utilisateur ;
    nom      |   prenom
-------------+-------------
 Caron       | Olivier
 Janot       | Stéphane
 Seynhaeve   | Franck
 Duthilleul  | Jean-Michel
```

- √ On peut inverser l'ordre de présentation (aucun impact sur le calcul)

Expression d'une projection

- √ Syntaxe : `select` $\texttt{col}_{i_1}$`,` $\texttt{col}_{i_p}$`, ...,` $\texttt{col}_{i_p}$ `from nomTable`
- √ Exemple :

```
biblio=# select nom, prenom from utilisateur ;
    nom      |   prenom
-------------+-------------
 Caron       | Olivier
 Janot       | Stéphane
 Seynhaeve   | Franck
 Duthilleul  | Jean-Michel
```

- √ On peut inverser l'ordre de présentation (aucun impact sur le calcul)
- √ Algèbre relationnel pur : clause distinct

Expression d'une restriction

√ Introduction clause `WHERE`

Expression d'une restriction

- √ Introduction clause `WHERE`
- √ Utilisation des opérateurs booléens : `and, or, not`

Expression d'une restriction

- √ Introduction clause `WHERE`
- √ Utilisation des opérateurs booléens : `and, or, not`
- √ Comparaison de chaînes, dates, d'entiers,. . .

Expression d'une restriction

- √ Introduction clause `WHERE`
- √ Utilisation des opérateurs booléens : `and, or, not`
- √ Comparaison de chaînes, dates, d'entiers,. . .
- √ Exemple :

```
biblio=# select * from livre where auteur=2 ;
 num_l |        titre         | auteur
-------+----------------------+--------
     2 | Les misérables       |      2
     3 | Notre dame de Paris  |      2
```

Traitement de chaînes (1/2)

$\surd$ Accès très simple : ch1 = ch2

Traitement de chaînes (1/2)

- $\surd$ Accès très simple : ch1 = ch2
- $\surd$ Opérateur `LIKE`

Traitement de chaînes (1/2)

- √ Accès très simple : ch1 = ch2
- √ Opérateur `LIKE`
- √ Caractère spéciaux : '%' (remplace de 0 à plusieurs caractères) et '_' (remplace exactement un caractère).

Traitement de chaînes (1/2)

- √ Accès très simple : ch1 = ch2
- √ Opérateur `LIKE`
- √ Caractère spéciaux : '%' (remplace de 0 à plusieurs caractères) et '_' (remplace exactement un caractère).
- √ Exemple :

```
biblio=#  select distinct titre from livre where titre like 'H%' ;
                  titre
-------------------------------------
 Harry Potter à l'école des sorciers
 Harry Potter et la chambre des secrets
(2 rows)
```

Traitement de chaînes (2/2)

$\surd$ Opérateur de comparaison '$>$', '$<$', . . .(ordre lexicographique)

Traitement de chaînes (2/2)

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

Traitement de chaînes (2/2)

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

```
biblio=# select upper(nom || ' ' || prenom) as nom_prenom
         from utilisateur ;
       nom_prenom
------------------------
 CARON OLIVIER
 JANOT STÉPHANE
 SEYNHAEVE FRANCK
 DUTHILLEUL JEAN-MICHEL
(4 rows)
```

Comparaison de valeurs

√ Introduction clause `BETWEEN`

Comparaison de valeurs

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

```
select nom from utilisateur where nom between 'A%' and 'F%' ;
    nom
------------
 Caron
 Duthilleul
```

Comparaison de valeurs

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

```
select nom from utilisateur where nom between 'A%' and 'F%' ;
    nom
------------
 Caron
 Duthilleul
```

- √ Note : l'exemple suivant est identique :

```
select nom from utilisateur
  where nom >= 'A%' and nom < 'F%' ;
```

Présentation des données (1/2)

√ Ordre d'affichage des colonnes

Présentation des données (1/2)

- √ Ordre d'affichage des colonnes
- √ Clause `distinct`, évite les doublons

Présentation des données (1/2)

- √ Ordre d'affichage des colonnes
- √ Clause `distinct`, évite les doublons
- √ Ordre d'affichage des lignes, clause `Order By`
- √ Ordre des lignes multi-critères

Présentation des données (1/2)

- √ Ordre d'affichage des colonnes
- √ Clause `distinct`, évite les doublons
- √ 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 ] [, ...]
```

Présentation des données (2/2)

√ Syntaxe :

```
ORDER BY expression [ ASC | DESC |
USING operator ] [, ...]
```

√ Exemple :

```
biblio=# 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

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

Opérations de calcul

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

```
biblio=# 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
```

Opérations de calcul

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

```
biblio=# 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 standards : count, min, max, avg, sum

Fonctions de calcul - exemples

```
biblio=# select count(*) as nombre from livre ;
 nombre
-------
     5
```

```
biblio=# 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`

Calcul sur des groupes de lignes (1/2)

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

```
biblio=# select auteur, count(*) as nbre_par_auteur
biblio-# from livre group by auteur ;
 auteur | nbre_par_auteur
--------+-----------------
      1 |               1
      2 |               2
      3 |               2
```

- √ Note : Toutes les colonnes figurant dans un `group by` doivent apparaître dans la clause `select`.

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`

Calcul sur des groupes de lignes (1/2)

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

```
biblio=# select auteur, count(*) as nbre_par_auteur
biblio-# from livre group by auteur ;
 auteur | nbre_par_auteur
--------+-----------------
      1 |               1
      2 |               2
      3 |               2
```

- √ Note : Toutes les colonnes figurant dans un `group by` doivent 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 :

```
biblio=# select auteur, count(*) as nbre_par_auteur
biblio-# 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 :

```
biblio=# select auteur, count(*) as nbre_par_auteur
biblio-# from livre group by auteur having count(*)>1 ;
 auteur | nbre_par_auteur
--------+-----------------
      2 |               2
      3 |               2
```

- √ Ne pas confondre avec clause `where`

Expression d'un produit cartésien

√ Utilisation clause `from`

Expression d'un produit cartésien

- √ Utilisation clause `from`
- √ Déclaration de variables (ou utiliser le nom de la table)

Expression d'un produit cartésien

- √ Utilisation clause `from`
- √ Déclaration de variables (ou utiliser le nom de la table)
- √ Produit cartésien : pas de sens en soi !

Expression d'un produit cartésien

- √ Utilisation clause `from`
- √ Déclaration de variables (ou utiliser le nom de la table)
- √ Produit cartésien : pas de sens en soi !
- √ Exemples :

```
select * from utilisateur, livre ;
... (20 lignes)

select distinct e.nom as nomEditeur , a.nom as NomAuteur
  from editeur e, auteur a;
```

Expression d'une jointure

√ Relier avec cohérence deux (ou plus) tables.

Expression d'une jointure

- √ Relier avec cohérence deux (ou plus) tables.
- √ Relier les clés étrangères avec les clés primaires correspondantes

Expression d'une jointure

- √ Relier avec cohérence deux (ou plus) tables.
- √ Relier les clés étrangères avec les clés primaires correspondantes
- √ Exemple :

```
select titre, nom from auteur, livre
  where auteur.num_a=livre.auteur ;
                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
```

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, livre l2
    where l1.auteur=l2.auteur
      and l1.titre > l2.titre ;
                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 (ex. Oracle)

Les jointures à la SQL/2

- √ Nouvelles possibilités d'expression de jointures
- √ Non encore implémenté par tous les SGBD (ex. Oracle)
- √ 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

Les jointures à la SQL/2

- √ Nouvelles possibilités d'expression de jointures
- √ Non encore implémenté par tous les SGBD (ex. Oracle)
- √ 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 titre, nom
  from livre right 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
                                         | Paltoquet
(6 rows)
```

Jointure externe complète

```
select titre, nom
  from livre full 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                            |
                                             | Paltoquet
(7 rows)
```

Multi-jointures SQL/2

```
select  nom, 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)

- √ INNER et OUTER sont toujours facultatifs
- √ LEFT, RIGHT et FULL impliquent une jointure externe

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

- √ INNER et OUTER sont toujours facultatifs
- √ 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 (1/2)

√ 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.a=t2.a`

√ `NATURAL` effectue une comparaison de toutes les colonnes de même nom dans les deux tables.

Expression d'une union

√ Les deux tables doivent avoir la même structure

Expression d'une union

- √ Les deux tables doivent avoir la même structure
- √ Exemple :

```
select nom from auteur union select nom from editeur ;
      nom
---------------
 Albert-René
 Albert Uderzo
 Folio
 Gallimard
 J.K. Rowling
 Victor Hugo
```

Expression d'une intersection

√ Les deux tables doivent avoir la même structure

Expression d'une intersection

- √ Les deux tables doivent avoir la même structure
- √ Exemple :

```
select nom from auteur intersect select nom from editeur ;
      nom
---------------
(0 rows)
```

Expression d'une différence

- √ Les deux tables doivent avoir la même structure
- √ Exemple :

```
select nom from auteur except select nom from editeur ;
      nom
---------------
 Albert Uderzo
 J.K. Rowling
 Victor Hugo
```

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`
- √ 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 (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, emprunte where
  emprunte.num_l=livre.num_l and
  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) ;
```

- √ (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 il a écrit des livres

```
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.
- On peut tester aussi la valeur `null`.

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) ;
```

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

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. . .

√ Stockage d'une requête dans des tables

```
select ... into table nomNouvelleTable from ...
```

Mais aussi. . .

- √ Stockage d'une requête dans des tables

  ```
  select ... into table nomNouvelleTable from ...
  ```

- √ Insertion de plusieurs lignes

  ```
  insert into table nomTable select ...
  ```

En résumé

√ Le langage est simple !

En résumé

√ Le langage est simple !

√ Répond à la majorité des requêtes

En résumé

- √ Le langage est simple !
- √ Répond à la majorité des requêtes
- √ Langage en constante évolution

En résumé

- √ Le langage est simple !
- √ Répond à la majorité des requêtes
- √ Langage en constante évolution
- √ Le langage peut être très puissant (requêtes corrélatives)

En résumé

- √ Le langage est simple !
- √ 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

- √ Le langage est simple !
- √ 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`
- √ 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
```

- √ 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 la puissance de SQL