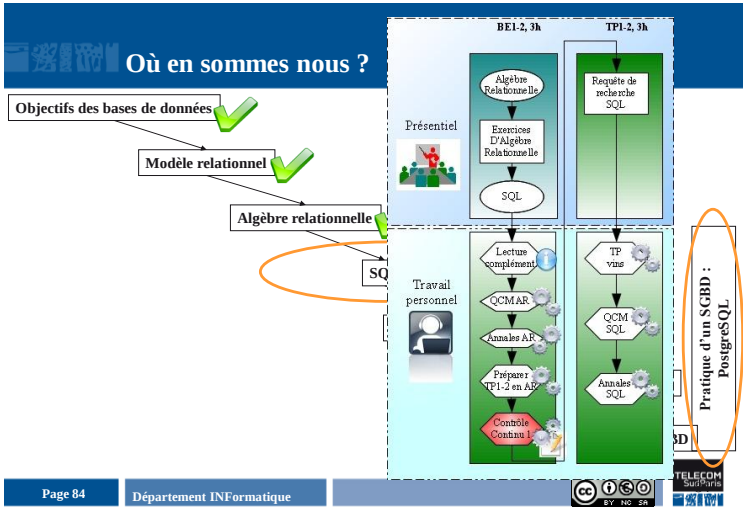
SQL
Structured Query Language

Département INFormatique

Où en sommes nous ?



Plan du document

Langages de requêtes

- **Langage de requêtes**
  - Langage de Définition de Données (LDD)
  - Langage de Manipulation de Données (LMD)
- **Langages de manipulation formels**
  - Algèbre relationnelle
  - Calcul relationnel (basé sur la logique des prédicats)
- **Langages de requêtes orientés utilisateur**
  - Structured Query Language (SQL)
  - QUEry Language (QUEL)
  - Query By Example (QBE)
- **Lien avec les langages de programmation**
  - Immersion d'un langage de requêtes dans un langage de programmation : approche Embedded SQL (**Java**, Pascal, C, ...)

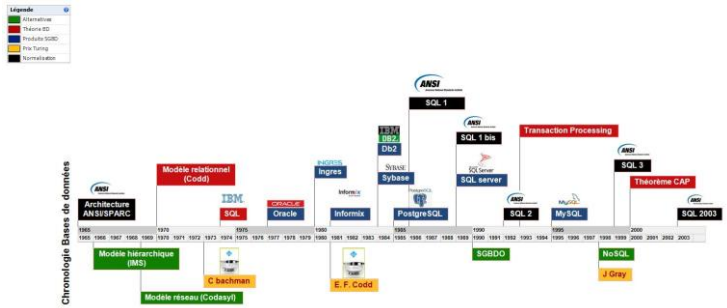
Présentation de SQL

- **Fonctionnalités**
  - Définition et manipulation de données au format relationnel
  - Contrôle des données
- **Le langage de manipulation**
  - Déclaratif, non procédural
  - Emprunté à l'algèbre relationnelle et au calcul relationnel de tuples
- **Puissance du langage de manipulation**

Algèbre Relationnelle
+
Fonctions-Agrégats
+
Tri

Requête SQL (sans fonctions et tri)
⇔
Composition d'opérations de l'algèbre relationnelle

Présentation de SQL (2)



Plan du document

BD Exemple : les vins

Vins(num, cru, annee, degre)	V
Recoltes(nvin, nprod, quantite)	R
Producteurs(num, nom, prenom, region)	P
Clients(num, nom, prenom, ville)	Cl
Commandes(ncde, date, ncli, nvin, qte)	C
Livraisons(ncde, no_ordre, qteLivree)	L

Plan du document

- Introduction
- BD Exemple : les vins
- **Langage de Définition des Données**
  - Fonctionnalités
  - Gestion du schéma d'une relation
  - LDD SQL vs. Modèle relationnel
- Langage de Manipulation des Données
  - Recherche
  - Mises à jour
- Traitement d'une requête
- Et moi que dois-je faire ?

Fonctionnalités

BE 5

- **Définition des schémas des relations**
- **Définition de contraintes d'intégrité**
- **Définition de vues relationnelles**
- **Définition de droits**

- **Définition du placement et des index**
  - (non normalisé => SGBD dépendant !!)

Gestion du schéma d'une relation (1)

- **Création minimale**

```
CREATE TABLE Vins (
   num    INTEGER ,
   cru    CHAR(40),
   annee  INTEGER);
```

- **Mise à jour**
  - Ajout d'un attribut (norme SQL2 !)

```
ALTER TABLE Vins

ADD [COLUMN] degre INTEGER ;
```

- **Suppression (norme SQL2 !)**

```
DROP TABLE Vins ;
```


Gestion du schéma d'une relation (2)

- **Domaines de base**
  - Numériques
    - Entier : INTEGER, SMALLINT
    - Décimal : DECIMAL (m,n), NUMBER(m,n)
    - Réel flottant : FLOAT, REAL
  - Chaîne de caractères : CHAR (n), VARCHAR(n)
  - Temporel : DATE (dans la norme SQL2 !)
  - Chaque SGBD possède d'autres domaines qui lui sont propres
- **Valeur NULL**
  - Absence de valeur
  - Ce n'est pas zéro !

LDD SQL vs. Modèle relationnel

- **Domaines limités**
- **Clé facultative :**
  - Doublons
  - Opérateur de projection différent (→ DISTINCT)
  - Relation n'est plus un ensemble

Plan du document

Syntaxe générale de recherche

Syntaxe		Comment remplir les clauses ?
SELECT	<liste d'attributs projetés>	❶ Quel résultat souhaite voir l'utilisateur (schéma du résultat) ?
FROM	<liste de relations>	❶ Où sont les attributs dont j'ai besoin ?
[WHERE	<liste des critères de restriction et de jointure>]	❷ Y-a t-il • des conditions sur les valeurs d'attributs exprimées dans ma requête ? • plusieurs relations dans ma clause FROM ? Si oui, quelles sont les conditions de jointure(s) ?

Restriction (σ) et projection (Π)

- *"Donner les vins de cru Pommard"*

```
SELECT num, annee, degre
FROM Vins
WHERE cru = 'Pommard';
```

Valeur des chaînes de caractères délimitées par des '

NUM	ANNEE	DEGRE
5	1976	11.70
23	1972	12.00

Nombre de tuples accédés : 2

Projection (Π)

- *"Donner tous les vins"*

```
SELECT *
FROM Vins ;
```

- *"Donner la liste de tous les crus, avec élimination des doublons"*

```
SELECT DISTINCT cru
FROM Vins;
```


Restriction

- *"Donner les vins de degré compris entre 8 et 12"*

```
SELECT *
FROM Vins
WHERE degre >=8 AND degre <=12 ;
```

```
SELECT *
FROM Vins
WHERE degre BETWEEN 8 AND 12 ;
```

```
SELECT *
FROM Vins
WHERE degre IN (8, 9, 10, 11, 12);
```


Restriction et tri

- *"Donner les vins dont le cru commence par p ou P"*

```
SELECT *
FROM Vins
WHERE cru LIKE 'p%' OR cru LIKE 'P%';
```

- *"Donner les crus des vins de millésime 1995 et de degré 12, triés par ordre croissant"*

```
SELECT cru
FROM Vins
WHERE annee=1995 AND degre = 12
ORDER BY cru ;
```


Jointure

- *"Donner les noms des producteurs de Pommard"*

```
SELECT nom
FROM Vins V, Recoltes R, Producteurs P
WHERE   V.num = R.nvin
AND     R.nprod = P.num
AND     cru = 'Pommard';
```

Produit cartésien

Conditions de jointures

Nom d'attribut non ambigu

Jointure (syntaxe SQL2)

- **Syntaxe plus proche de l'algèbre relationnelle (directement exprimée dans le FROM)**
- **Supportée dans plusieurs SGBD (>= Oracle 9, MySQL, SQLServer, …)**

```
SELECT nom
FROM Vins V JOIN Recoltes R ON (V.num = R.nvin)
  JOIN Producteurs P ON (R.nprod=P.num)
WHERE cru = 'Pommard' ;
```

- **Jointure « naturelle »**
  - L'égalité sur les attributs de même nom peut être remplacée par `NATURAL JOIN`, ou `JOIN … USING (attributs)`

Jointure « procédurale » ou ensembliste

```
SELECT nom
FROM Producteurs
WHERE num IN (
  SELECT nprod
  FROM Recoltes
  WHERE nvin IN (
    SELECT num
    FROM Vins
    WHERE cru = 'Pommard')) ;
```


Auto-jointure

- **Jointure d'une relation avec elle-même**

  → synonymes
- ***« Donner les couples de producteurs ayant le même nom. Préciser les régions »***

```
SELECT P1.nom, P1.region, P2.region
FROM Producteurs P1, Producteurs P2
WHERE P1.nom = P2.nom
AND P1.num > P2.num ;
```


Opérateurs ensemblistes

- **Union (norme SQL1)**
  - Élimination automatique des doublons

```
SELECT num FROM Producteurs
UNION
SELECT num FROM Clients ;
```

- **Intersection (norme SQL2 !)**

```
SELECT num FROM Producteurs
INTERSECT
SELECT num FROM Clients ;
```

- **Différence (norme SQL2 !)**

```
SELECT num FROM Clients
EXCEPT            (ou MINUS)
SELECT num FROM Producteurs ;
```

Fonctions

- **5 fonctions prédéfinies : `COUNT, SUM, MIN, MAX, AVG`**
- **Principe :**
  - S'applique à l'ensemble des valeurs d'un attribut d'une relation
  - Produit une valeur unique
  - Pour une requête **sans partionnement** (plus tard) :
    - uniquement dans le SELECT, jamais dans le WHERE
    - Ne pas mélanger dans le SELECT les fonctions et les attributs simples !

```
SELECT num, AVG(degre)
FROM Vins ;
```

num	AVG(degre)
{1, 3, 6, 10}	12,5
{5, 7}	12,0
{2, 8, 11}	11,0

Exemples avec fonctions

- **"Donner la moyenne des degrés de tous les vins"**

```
SELECT AVG(degre)
FROM Vins ;
```

- **"Donner la quantité totale commandée par le client de nom Bac"**

```
SELECT SUM(qte)
FROM Commandes, Clients
WHERE Clients.nom= 'Bac'
AND Clients.num=Commandes.ncli;
```


Exemples avec fonctions (2)

- ***"Nombre de crus différents"***

```
SELECT COUNT(DISTINCT cru)
FROM Vins ;
```

- ***"Nombre de vins"***

```
SELECT COUNT (*)
FROM Vins
```


Exemples avec fonctions (3)

- **" Vins dont le degré est supérieur à la moyenne des degrés des vins"**

```
SELECT *
FROM Vins
WHERE degre > (
   SELECT AVG(degre)
   FROM Vins) ;
```

- **" Numéros de commande où la quantité commandée a été totalement livrée"**

```
SELECT ncde
FROM Commandes C
WHERE qte = (
   SELECT SUM(L.qteLivree)
   FROM Livraisons L
   WHERE L.ncde =  C.ncde
        );
```

L'alias « C » est visible dans le bloc imbriqué

Partitionnement

- **Principe**
  - Partitionnement horizontal d'une relation, selon les valeurs d'un attribut ou d'un groupe d'attributs qui est spécifié dans la clause GROUP BY
  - Relation (logiquement) fragmentée en groupes de tuples, où tous les tuples de chaque groupe ont la même valeur pour l'attribut (ou le groupe d'attributs) de partitionnement
- **Fonctions sur les groupes**
- **Restrictions sur les groupes**
  - Application possible d'un critère de restriction sur les groupes obtenus
  - Clause HAVING

Exemples de partitionnement

- **" Donner, pour chaque cru, la moyenne des degrés des vins de ce cru …"**

```
SELECT cru, AVG(degre)
FROM Vins
GROUP BY cru ;
```

- **" … avec un tri par degré décroissant"**

```
SELECT cru, AVG(degre)
FROM Vins
GROUP BY cru
ORDER BY 2 DESC ;
```

- **" … uniquement si ce cru concerne plus de 3 vins"**

```
SELECT cru, AVG(degre)
FROM Vins
GROUP BY cru
HAVING COUNT(*)>=3
ORDER BY 2 DESC ;
```


Calcul de la partition

- ❶ **Trier la relation selon les attributs de groupement**
- ❷ **Créer une sous-relation pour chaque paquet ayant même valeur sur l'ensemble des attributs de groupement, ici « cru »**
- ❸ **Appliquer la clause SELECT sur chaque partition (dans notre exemple la valeur de cru et la moyenne des degrés sur la partition)**
- ❹ **Unifier les résultats**
- ❺ **Appliquer la restriction du HAVING**

initiale

Vins	cru	degre
	Pommard	12
	Tavel	11
	Pommard	13
	Tavel	12
	Tavel	13

❶ et ❷

Vins	cru	degre
	Pommard	12
	Pommard	13
	Tavel	11
	Tavel	12
	Tavel	13

❸ et ❹

Vins	cru	degre
	Pommard	12,5
	Tavel	12

Exemple de requête erronée

```
SELECT cru, num, AVG(degre)
FROM Vins
GROUP BY cru ;
```

- **Résultat « attendu »**

cru	num	AVG(degre)
Pommard	{1, 3, 6, 10}	12,5
Tavel	{5, 7}	12,0
Gamay	{2, 8, 11}	11,0

- **Problème**
  - Num est multivalué / cru
  - → il n'y a pas une valeur par case (pas en première forme normale)

Prédicats (1)

- **ALL**
  - Teste si la valeur d'un attribut satisfait un critère de comparaison avec **tous** les résultats d'une sous-requête

```
SELECT Cl.num, Cl.nom
FROM Clients Cl, Commandes C
WHERE Cl.num = C.ncli
AND C.qte >= ALL(
   SELECT qte
   FROM Commandes) ;
```


Prédicats (2)

- **ANY**
  - Teste si la valeur d'un attribut satisfait un critère de comparaison **avec au moins un** résultat d'une sous-requête

```
SELECT Cl.num, Cl.nom
FROM Clients Cl, Commandes C
WHERE Cl.num = C.ncli
AND C.qte > ANY (
   SELECT qte
   FROM Commandes ) ;
```


Prédicats (3)

- **EXISTS**
  - Teste si la **réponse** à une sous-requête est **non vide**
  - *« Producteurs ayant produit au moins un vin"*

```
SELECT P.*
FROM Producteurs P
WHERE EXISTS (
   SELECT R.*
   FROM Recoltes R
   WHERE P.num = R.nprod) ;
```

- **NOT EXISTS**
  - Teste si la **réponse** à une sous-requête est **vide**

Division avec prédicat EXISTS

"Quels sont les producteurs ayant produit tous les vins (ceux connus de la base de données) ?"

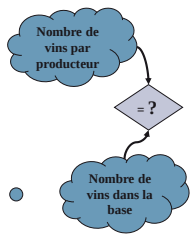
- Méthode : paraphraser en français avec **une double négation**

"Un producteur est sélectionné
s'il n'existe aucun
vin
pour lequel il n'y a aucune
récoltes
(sous entendu de ce producteur et de ce vin) "

```
SELECT P.*
FROM Producteurs P
WHERE NOT EXISTS (
   SELECT V.*
   FROM Vins V
   WHERE NOT EXISTS (
      SELECT R.*
      FROM Recoltes R
      WHERE P.num=R.nprod
      AND R.nvin=V.num )
) ;
```

Division avec prédicat (sans EXISTS)

```
SELECT P.*
FROM Producteurs P
WHERE P.num IN (
  SELECT nprod
  FROM Recoltes
  GROUP BY nprod
  HAVING COUNT(*) = (
    SELECT COUNT(*)
    FROM Vins) )
```



Synthèse

6	SELECT	<liste et/ou expressions attributs A_j et/ou fonctions sur Attributs A_p>	Projection de l'ensemble obtenu en (5) sur les A_j, calcul des expressions, calcul des fonctions (appliquées aux groupes s'il y en a) sur A_p
1	FROM	<liste de relations R_i>	Produit cartésien des relations R_i
2	WHERE	<Conditions sur les tuples> : C_1	Sélection des tuples de (1) respectant la condition C_1
3	GROUP BY	<liste attributs $A_k \supseteq A_j$>	Partitionnement de l'ensemble obtenu en (2) suivant les valeurs A_k
4	HAVING	<condition sur groupes - fonctions> : C_2	Sélection des groupes de (3) vérifiant C_2
5	ORDER BY	<liste d'attributs A_l ou n° ordre dans le SELECT>	Tri des tuples obtenus en (4) suivant les valeurs A_l

Synthèse (2)

- **Condition de recherche :**
  - WHERE (sélection de tuples), HAVING (sélection de groupes)
  - Compositions de conditions élémentaires (AND, OR, NOT)
  - Évaluée à Vrai ou Faux
- **Condition élémentaire :**
  - Évaluée à Vrai ou Faux
  - Prédicat :
    - Comparaison : =, <, <=, >, >=, <>
      - Attribut/valeur
      - Attribut/attribut
    - Intervalle :BETWEEN
    - Chaîne : LIKE
    - Nullité : IS NULL
    - Appartenance : IN
    - Quantification : EXISTS, ANY, ALL

Exemple complet

- ***"Donnez par ordre croissant le nom et la somme des quantités commandées par des Clients bordelais, uniquement si chaque commande est d'une quantité strictement supérieure à 20 litres."***

```
SELECT
FROM
WHERE
AND
GROUP BY
HAVING
ORDER BY
```

Plan du document

- Introduction
- BD Exemple : les vins
- Définition des données
- **Manipulation des données**
  - Recherche
  - Mises à jour
    - Insertion
    - Suppression
    - Modification
- Traitement d'une requête
- Et moi que dois-je faire ?

Insertion

- **Insertion d'un seul tuple**

```
INSERT INTO Vins VALUES (100, 'Jurançon', 1979, 12) ;

INSERT INTO Vins (num, cru) VALUES (200, 'Gamay') ;
```

- **Insertion d'un ensemble de tuples**

```
CREATE TABLE BORDEAUX(num INTEGER, annee INTEGER, degre
  NUMBER(4,2)) ;

INSERT INTO BORDEAUX
   SELECT num, annee, degre
   FROM Vins
   WHERE cru = 'Bordeaux' ;

CREATE TABLE BORDEAUX AS
   SELECT num, annee, degre
   FROM Vins
   WHERE cru = 'Bordeaux' ;
```


Suppression

- **"Supprimer tous les tuples de Vins"**

```
DELETE FROM Vins ;        ou      TRUNCATE TABLE Vins ;
```

- **"Supprimer le vin de numéro 150"**

```
DELETE FROM Vins
WHERE num = 150 ;
```

- **"Supprimer les vins de degré <9 ou >12"**

```
DELETE FROM Vins
WHERE degre < 9 OR degre > 12 ;
```

- **"Supprimer les commandes passées par Belaïd"**

```
DELETE FROM Commandes
WHERE ncli IN (
   SELECT num
   FROM Clients
   WHERE nom= 'Belaïd')  ;
```


Modification

- **« Le producteur 150 habite dans le sud ouest »**

```
UPDATE Producteurs
SET region = 'Sud Ouest'
WHERE num = 150  ;
```

- **« Les degrés des Gamays augmentent de 10 % »**

```
UPDATE Vins
SET degre = degre * 1.1
WHERE cru = 'Gamay'  ;
```

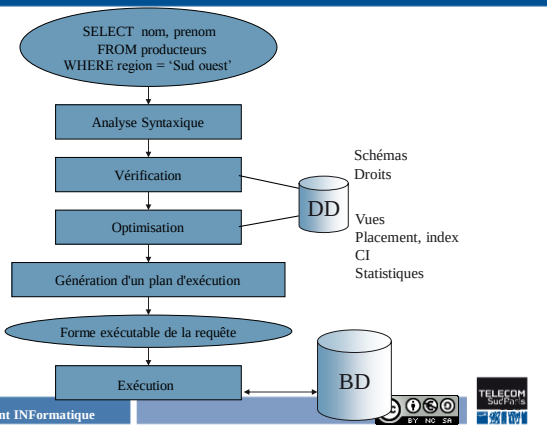
- **« Le client 'Bac' augmente ses commandes de 10 unités »**

```
UPDATE Commandes
SET qte = qte + 10
WHERE ncli IN (
   SELECT num
   FROM Clients
   WHERE nom='Bac') ;
```

Plan du document

- Introduction
- BD Exemple : les vins
- Définition des données
- Manipulation des données
  - Recherche
  - Mises à jour
- **Traitement d'une requête**
- Et moi que dois-je faire ?

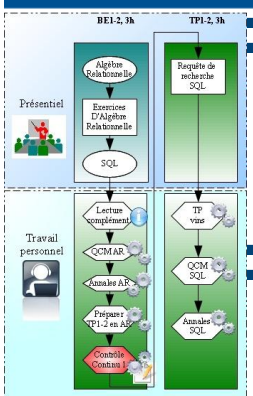
Traitement d'une requête SQL



Plan du document

- Introduction
- BD Exemple : les vins
- Définition des données
- Manipulation des données
  - Recherche
  - Mises à jour
- Traitement d'une requête
- **Et moi que dois-je faire ?**

Et moi que dois-je faire ?



- **Relire les transparents**
- **Lire la documentation complémentaire :**
  - Notre cours rédigé : http://www-inf.it-sudparis.eu/COURS/bd/index.php?idr=37
  - Cours rédigé de Télécom ParisTech : http://perso.telecom-paristech.fr/~talel/cours/inf225/wwwbd/polyv7/chap4.htm
  - Cours rédigé de l'IMAG : http://www-lsr.imag.fr/Les.Personnes/Herve.Martin/HTML/SQL.htm
- **Faire le QCM lié à ce cours sur moodle**
- **Pratiquer**
  - ⇨ préparer le TP1-2 (SQL) en algèbre relationnelle pour comprendre le lien AR⇔SQL
  - ⇨ faire les questions SQL des annales
  - ⇨ faire le TP en ligne (les vins)