

SQL

Structured Query Language

SQL, une concrétisation de la théorie relationnelle

You speak english .. So you speak SQL

Plan



1. Introduction

2. BD exemple

3. Recherche

4. Mises à jour

Présentation de SQL

- Fonctionnalités
 - Définition (LDD)
 - Manipulation (LMD) de données au format relationnel

- 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

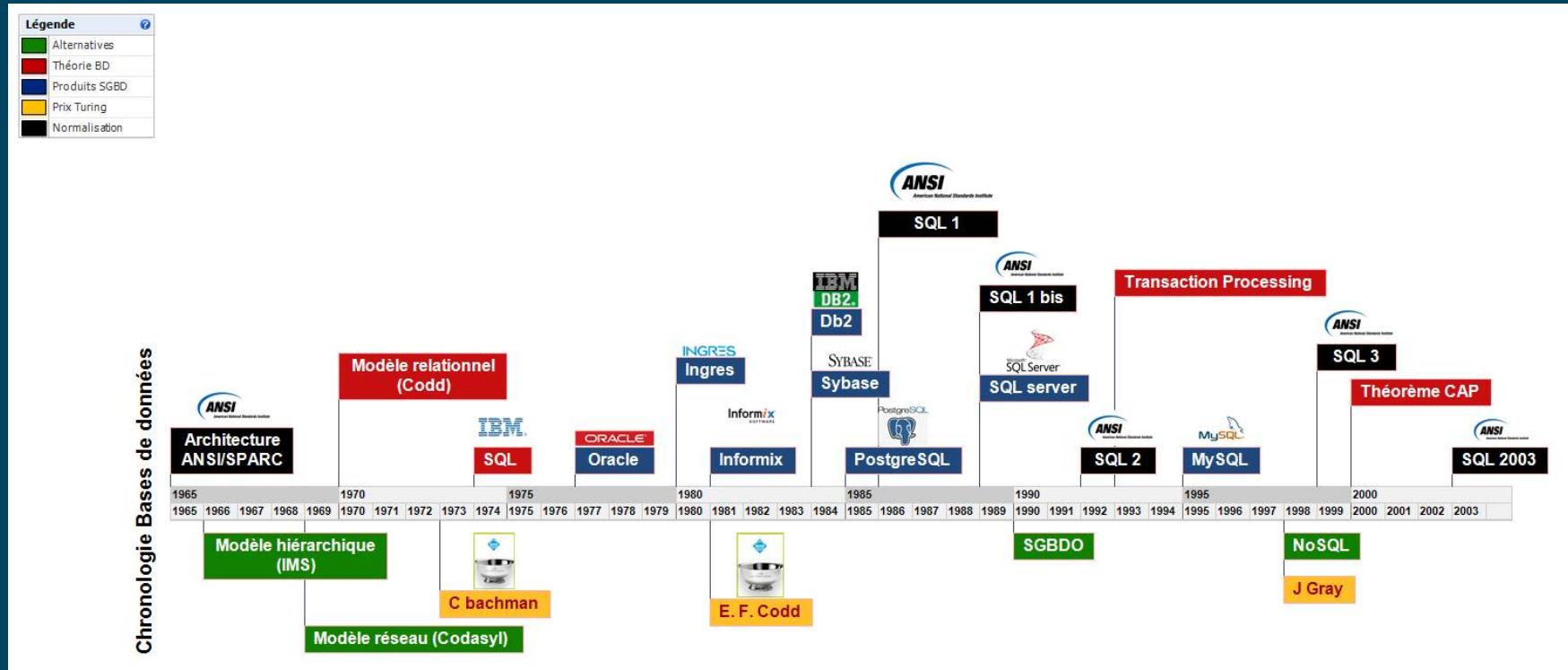


Requête SQL (sans fonctions et tri)



Composition d'opérations de l'algèbre relationnelle

Présentation de SQL (2)



Plan



1. Introduction

2. BD exemple

3. Recherche

4. Mises à jour

BD Exemple : les vins



Vins (num, cru, année, degré) V

Recoltes (nvin, nprod, quantité) R

Producteurs (num, nom, prénom, région) P

Clients (num, nom, prénom, ville) Cl

Commandes (ncde, date, ncli, nvin, qte) C

Livraisons (ncde, no_ordre, qteLivree) L

Plan



1. Introduction
2. BD exemple
3. Recherche
 1. Les incontournables SELECT ... FROM ... WHERE
 2. Relier est indispensable mais jamais sans condition
 3. C'est l'histoire de deux requêtes qui voulaient être reliées (opérateurs ensemblistes)
 4. Des calculs sur une relation
 5. Des calculs par groupe
1. Mises à jour

Syntaxe générale de recherche

Syntaxe

SELECT <liste
d'attributs
projétés>

FROM <liste de
relations>

[WHERE] <liste des
critères de
restriction
et de
jointure>]

Comment remplir les clauses ?



❸ Quel résultat souhaite voir
l'utilisateur (schéma du résultat) ?

❶ Dans quelles relations sont les
attributs dont j'ai besoin ?

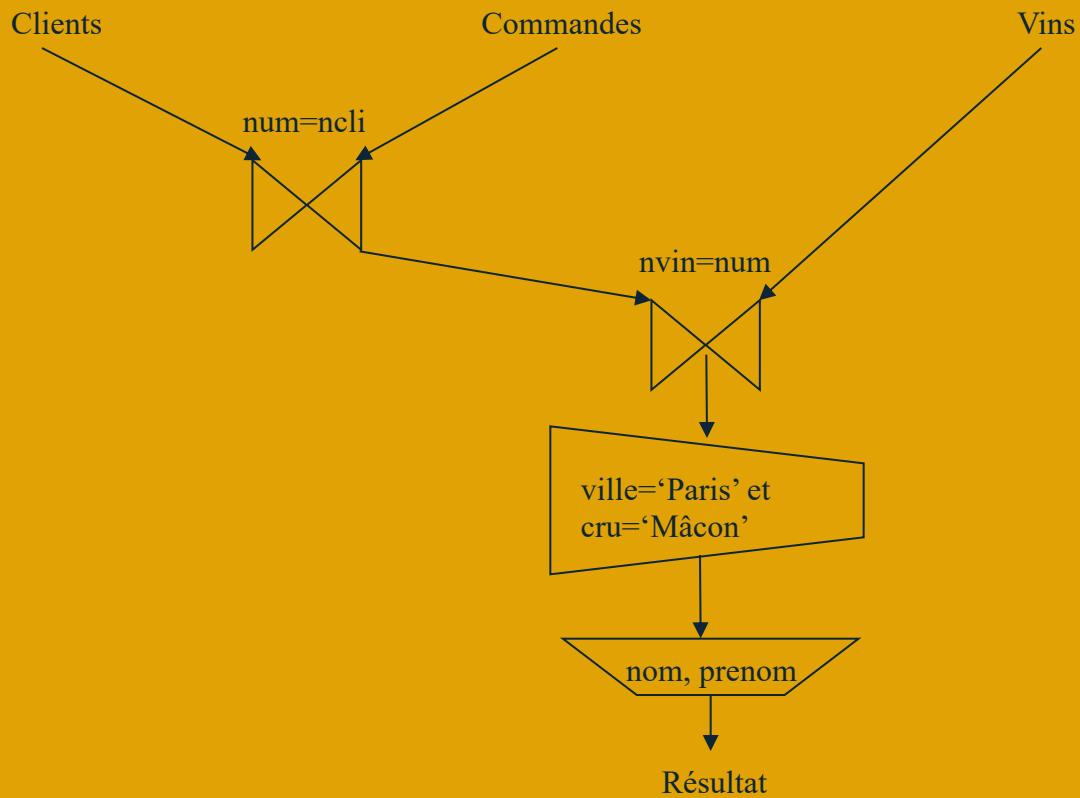
❷ Y-a t-il

- 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) ?



Présentation de SQL

Arbre algébrique



SQL

SELECT

nom, prenom

FROM

Clients Cl,

Commandes C,

Vins V

WHERE

Cl.num=C.ncli

AND V.num=C.nvin

AND ville='Paris'

AND cru='Macon'

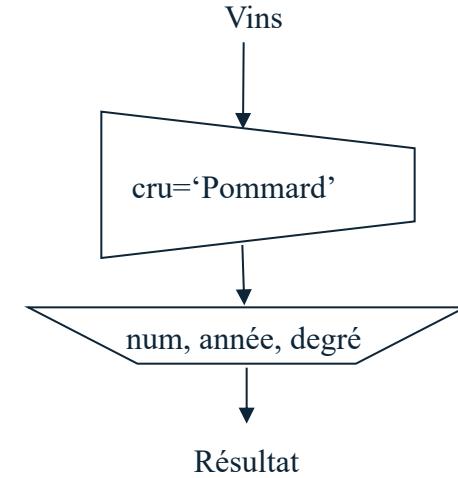
Restriction (σ) et projection (Π)

■ "Donner les vins de cru Pommard"

Schéma

Vins (num, cru, année, degré)

```
SELECT num, année, degré  
FROM   Vins  
WHERE  cru = 'Pommard';
```



NUM	ANNEE	DEGRE
5	1976	11.70
23	1972	12.00

Nombre de tuples accédés : 2

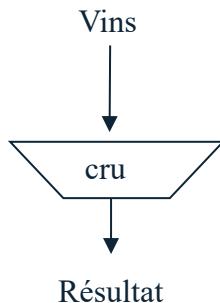
Projection (Π)

- ## "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 degré = 12
ORDER BY cru [DESC];
```

Plan



1. Introduction
2. BD exemple
3. Recherche
 1. Les incontournables SELECT ... FROM ... WHERE
 2. Relier est indispensable mais jamais sans condition
 3. C'est l'histoire de deux requêtes qui voulaient être reliées (opérateurs ensemblistes)
 4. Des calculs sur une relation
 5. Des calculs par groupe
 6. Synthèse
4. Mises à jour

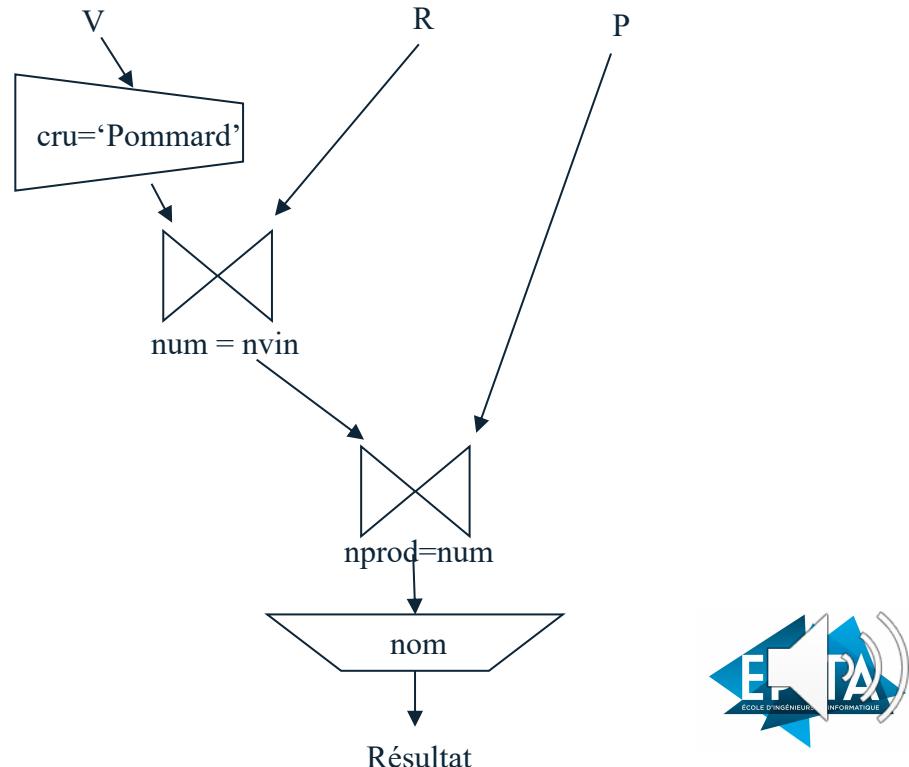
Jointure

■ « *Donner les noms des producteurs de Pommard* »

Schéma

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

Vins(num, cru, annee, degre) V
Recoltes(nvin, nprod, quantite) R
Producteurs(num, nom, prenom, region) P



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 produisant le même vin* »

```
SELECT P1.num, P2.num  
FROM Producteurs P1, Producteurs P2, Recoltes R1,  
      Recoltes R2  
WHERE P1.num = R1.nprod AND P2.num = R2.nprod  
AND R1.nvin = R2.nvin  
AND P1.num > P2.num ;
```

Plan



1. Introduction
2. BD exemple
3. Recherche
 1. Les incontournables SELECT ... FROM ... WHERE
 2. Relier est indispensable mais jamais sans condition
 3. C'est l'histoire de deux requêtes qui voulaient être reliées (opérateurs ensemblistes)
 4. Des calculs sur une relation
 5. Des calculs par groupe
 6. Synthèse
4. Mises à jour
5. Traitement d'une requête

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 ;
```

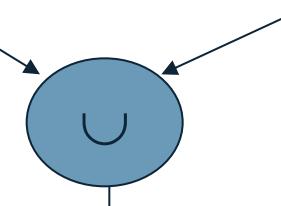
■ Différence (norme SQL2 !)

```
SELECT num FROM Clients
```

EXCEPT (ou MINUS)

```
SELECT num FROM Producteurs ;
```

SELECT SELECT

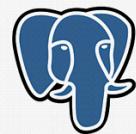


Schéma

Producteurs(num, nom, prenom, region)

Clients(num, nom, prenom, ville)

Postgre**SQL**



Plan



1. Introduction
 2. BD exemple
 3. Recherche
 1. Les incontournables SELECT ... FROM ... WHERE
 2. Relier est indispensable mais jamais sans condition
 3. C'est l'histoire de deux requêtes qui voulaient être reliées (opérateurs ensemblistes)
 4. Des calculs sur une relation
 5. Des calculs par groupe
 6. Synthèse
-
4. Mises à jour

Les fonctions par l'exemple

Vins	num	cru	degre	annee
	1	Pommard	12	2010
	2	Tavel	11	2009
	3	Pommard		2010
	4	Tavel	12	
	5	Tavel	13	2012

Nb vins 5

Nb cru 2

Nb annee 3

Degre moyen 12

Degre min 11

Degre max 13

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 partitionnement** (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 ;
```



{1, 3, 6, 10, 5, 7, 2, 8 ,11}	12,3
----------------------------------	------

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 Courtois"

Schéma

Clients(num, nom, prenom, ville)
Commandes(ncde, date, ncli, nvin, qte)

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

Exemples avec fonctions (2)

■ "Nombre de crus différents"

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

■ "Nombre de vins"

```
SELECT COUNT(num) 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
);
```

Schéma

Commandes(ncde, date, ncli, nvin, qte)
Livraisons(ncde, no_ordre, qteLivree)

Plan



1. Introduction
2. BD exemple
3. Recherche
 1. Les incontournables SELECT ... FROM ... WHERE
 2. Relier est indispensable mais jamais sans condition
 3. C'est l'histoire de deux requêtes qui voulaient être reliées (opérateurs ensemblistes)
 4. Des calculs sur une relation
 5. Des calculs par groupe
 6. Synthèse
4. Mises à jour

Les fonctions par groupe - exemple

Vins	num	cru	degre	annee
	1	Pommard	12	2010
	2	Tavel	11	2009
	3	Pommard	13	2010
	4	Tavel	12	
	5	Tavel	13	2012

Stat cru	cru	Moyenne degre
	Pommard	12,5
	Tavel	11

Partitionnement

■ Principe : « je veux des stats par ??? »

- 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  AVG(degre) 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 ;
```

Conditions sur fonctions

Calcul de la partition

1. Trier la relation selon les attributs de groupement
2. Créer une sous-relation pour chaque paquet ayant même valeur sur l'ensemble des attributs de groupement, ici « cru »
3. Calculer les fonctions (clause SELECT ou HAVING ou ORDER BY) sur chaque partition (dans notre exemple la valeur de cru et la moyenne des degrés sur la partition)
4. Appliquer la restriction du HAVING
5. Unifier les résultats

initiale

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

1 et 2

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

3 4 5

Vins	cru	degre
	Pommard	12,5
	Tavel	12

Exemple de requête erronée

■ Requête erronée

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

■ Résultat « attendu »

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 conforme au modèle relationnel)

Plan



1. Introduction
2. BD exemple
3. Recherche
 1. Les incontournables SELECT ... FROM ... WHERE
 2. Relier est indispensable mais jamais sans condition
 3. C'est l'histoire de deux requêtes qui voulaient être reliées (opérateurs ensemblistes)
 4. Des calculs sur une relation
 5. Des calculs par groupe
 6. Synthèse
4. Mises à jour

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

```
SELECT      nom,  SUM(qte)
FROM Clients Cl, Commandes C
WHERE Cl.num = C.ncli
      AND ville = 'Bordeaux'
GROUP BY   Cl.num , nom
HAVING     MIN(qte) >= 20
ORDER BY   SUM(qte)
```

Schéma

Clients(num, nom, prenom, ville)
Commandes(ncde, date, ncli, nvnr, qte)

Plan



1. Introduction

2. BD exemple

3. Recherche

4. Mises à jour

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 'Courtois' augmente toutes ses commandes de 10 unités »

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

