

# Bases du Langage SQL

- Le modèle relationnel est basé sur les propriétés de l'algèbre relationnelle
  - => - sémantique du langage « claire »
  - optimisations à l'exécution
- Concepts très simple : vision tabulaire des données

PERSONNE		Relation ou Table (PERSONNE)	
NO_SS	NOM	PRENOM	ADRESSE
12345122	Durand	Jean	Lyon
27234563	Dupond	Lise	St Etienne

← N-uplet ou tuple  
(12345122, Durand, Jean, Lyon)

↑ Attribut : ADRESSE

- Le type des attributs (domaine) : types simples (entier, date, chaîne, ...) par opposition à types structurés

# Création de TABLE

- CREATE TABLE <nom> (<attribut>, ...)
- 
- <attribut> : <nom> <type> [ NOT NULL]

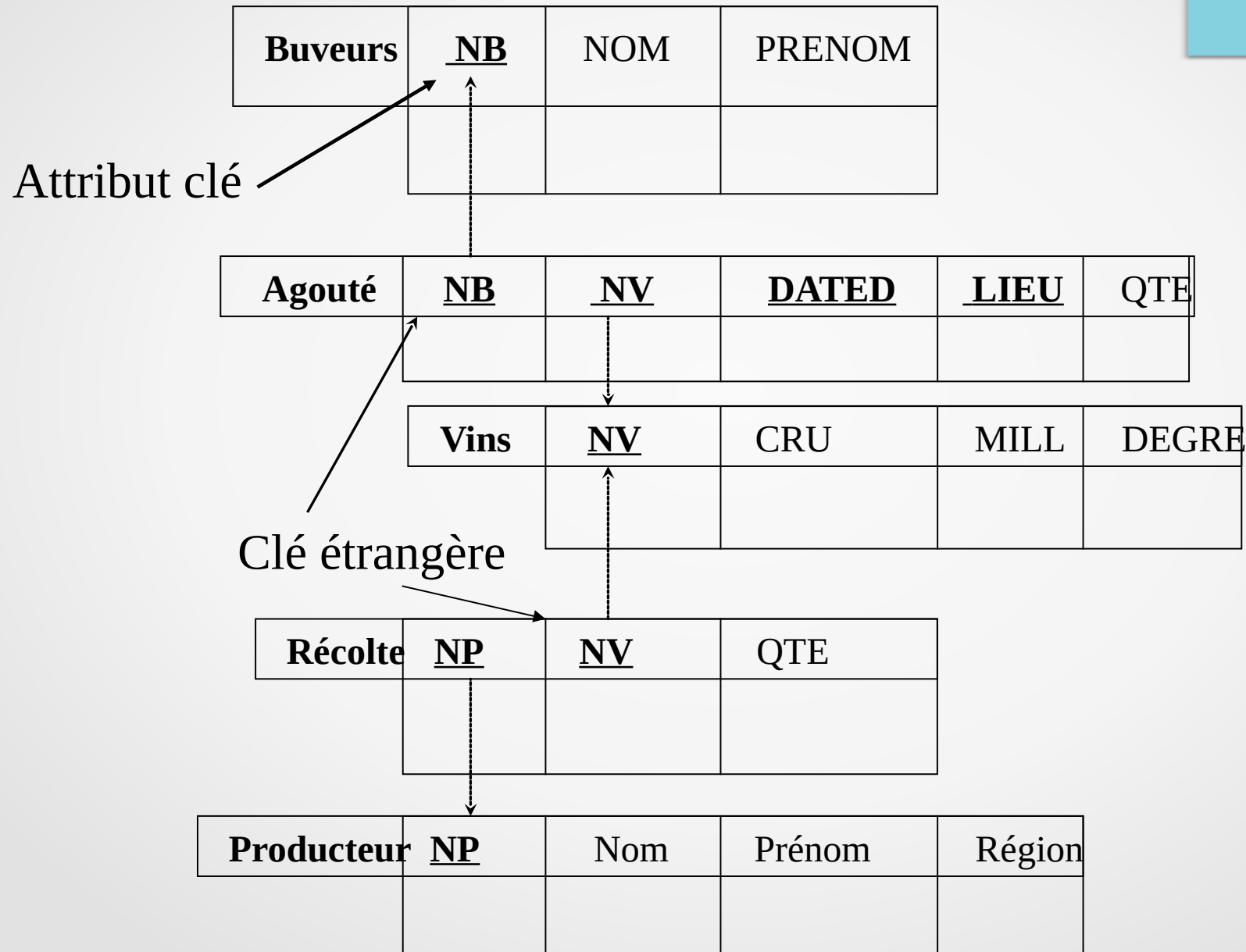
The diagram illustrates the relationship between the attribute name and the data type in a table creation statement. A vertical arrow points from the attribute name in the statement above to the list of data types below. Another vertical arrow points from the word 'Contrainte' to the attribute name in the statement above.

↓

↑  
Contrainte

CHAR (n)  
NUMBER [(n)]  
NUMBER (n, décimales)  
DATE  
LONG

# Exemple : schéma relationnel



# Exemple : ordres SQL de creation

```
CREATE TABLE BUVEURS (NB NUMBER NOT NULL,  
                        NOM CHAR (20),  
                        PRENOM CHAR (20))
```

```
CREATE TABLE AGOUTE (NB NUMBER NOT NULL,  
                      NV NUMBER NOT NULL,  
                      DATED DATE NOT NULL,  
                      LIEU CHAR (20) NOT NULL,  
                      QTE NUMBER)
```

```
CREATE TABLE VINS (  NV NUMBER NOT NULL,  
                     CRU CHAR (20),  
                     MILL NUMBER,  
                     DEGRE NUMBER (3, 1))
```

```
CREATE TABLE RECOLTE (NP NUMBER NOT NULL,  
                      NV NUMBER NOT NULL,  
                      QTE NUMBER)
```

```
CREATE TABLE PRODUCTEUR (NP NUMBER NOT NULL,  
                          NOM CHAR (20), PRENOM CHAR(20),  
                          REGION CHAR (20))
```

# Suppression de table

DROP TABLE <nom de la table>

Détruit les données de la table et la définition de celle-ci

Exemple :

DROP TABLE **PRODUCTEUR**

# MODIFICATION D'UNE TABLE

- On ne peut que :
  - Ajouter une colonne
  - Agrandir la taille d'une colonne
  - Modifier un attribut Null/Not Null
- ALTER TABLE <Nom de la table>
  - ADD (<attribut>, ...)
  - MODIFY (<attribut>, ...)
- **Exemple :**
  - ALTER TABLE **BUVEURS** ADD  
(**TYPEBUVEUR** CHAR (10))
  - ALTER TABLE **VINS** MODIFY  
(**DEGRE** NUMBER (4, 2))
  - ALTER TABLE **AGOUTE** MODIFY  
(**QTE** NUMBER NOT NULL)

*Valable que si tous les n-uplets contiennent des valeurs non nulles*

# Langage de Manipulation de données (LMD)

- Insertion de n-uplets : INSERT
- Suppression de n-uplets : DELETE
- Mise à jour de n-uplets : UPDATE
- Sélection de n-uplets : SELECT

# Insertion de valeurs : **INSERT**

- INSERT INTO <nom de table>  
                                  [(<nom attribut>, ...)]  
                  VALUES  
                  (<valeur>, ...)  
          ou     <Requête de sélection>
- **Exemples :**
- INSERT INTO **VINS** VALUES (**10**, **'BEAUJOLAIS'**, **1990**, **12**)
- INSERT INTO **VINS** VALUES (**11**, **'COTES DU R'**, **NULL**, **13**)
- INSERT INTO **VINS**  
          (**NV**, **CRU**, **DEGRE**) VALUES  
          (**11**, **'COTES DU R'**, **13**)





# Suppression de n-uplets : **DELETE**

- DELETE FROM <nom de table>  
[WHERE <condition>]
- Exemples :
- DELETE FROM **VINS**  
WHERE **CRU = 'BEAUJOLAIS'**  
AND **MILL < 1985**
- DELETE FROM **VINS**  
détruit tous les n-uplets, mais  
ne détruit pas la définition de  
la table.

# Mise à jour de valeurs : **UPDATE**

- UPDATE <nom de table>  
    SET <attribut> = <expression>, ...  
    [WHERE <condition>]
- UPDATE <nom de table>  
    SET (<attribut>, ...) =  
        (<requête de sélection>)  
    [WHERE <condition>]
- 
- **Exemple :**
  - UPDATE VINS  
    SET **DEGRE = 12.5**  
    WHERE **MILL = 1990 AND**  
        **CRU = 'BEAUJOLAIS'**



SELECT [ALL|DISTINCT] <expression>, ...  
FROM <table> [alias], ...  
[WHERE <condition>]

[GROUP BY <expression>, ...  
[HAVING <condition>]]

[UNION | INTERSECT | MINUS  
SELECT ...]

[ORDER BY <expression|position> [ASC|DESC], ...]

# Clause FROM : FROM table1, table2, table3

table1

A	B	C
A	D	E
F	B	E

A1      A2      A3

table2

C	G
E	H

A4      A5

table3

Z
---

A6

Produit cartésien

Table temporaire

A	B	C	C	G	Z
A	B	C	E	H	Z
A	D	E	C	G	Z
A	D	E	E	H	Z
F	B	E	C	G	Z
F	B	E	E	H	Z

A1                  A2                  A3                  A4                  A5                  A6

# Clause Select (projection)

SELECT [ALL | DISTINCT] <expression>, ...

<expression> :

- \* tous les attributs
- fonctions agrégate (COUNT, SUM, AVG, MAX, MIN)
- expressions arithmétiques
- Renommage d'attribut
- Fonction d'affichage

Exemple : *Donner les crus des vins*

```
SELECT * FROM VINS
```

*Donner les crus des vins sans double*

```
SELECT DISTINCT CRU  
FROM VINS
```

# Clause WHERE

## WHERE <expression>

- <expression> :
  - expressions liées par AND ou OR
  - opérateurs de comparaison
    - = != < > <= >=
    - BETWEEN LIKE NULL IN ALL ANY EXIST
  - Opérande : Constante, nom d 'attribut, fonction, résultat d 'un SELECT imbriqué
- **Exemple:** *Donner le nombre de Beaujolais de 1994 à 1997 ainsi que le degré moyen.*

```
SELECT    COUNT (*), AVG (DEGRE)  
FROM      VINS  
WHERE     CRU = 'BEAUJOLAIS'  
          AND    MILL BETWEEN 1994 AND 1997
```

# Clause Group By

## GROUP BY <attribut>, ....

- Permet de fragmenter la relation temporaire après l'application de la clause WHERE en un ensemble de sous-relations dont les n-uplets ont la même valeur pour les attributs spécifiés. On utilise cela pour faire des calculs (fonctions d'agrégation) sur chaque sous-relation
- **Exemple** : *Donner le nom et le degré moyen de chaque vins.*

```
SELECT      CRU, AVG (DEGRE)
FROM        VINS
GROUP BY    CRU
```

# Clause Having

## HAVING <expression>

- Permet de supprimer les n-uplets ou les sous-relation qui ne répondent pas à l'expression
- **Exemple** : *Donner le nom et le degré moyen des crus. Ne lister que les crus de degré moyen  $\geq 12$*

```
SELECT  CRU, AVG (DEGRE)
FROM    VINS
GROUP BY CRU
HAVING  AVG (DEGRE)  $\geq$  12
```



# Clause Order By

**ORDER BY <attribut|position>  
[ASC|DESC], ...**

- <position> : utilisé lorsque le résultat est produit en utilisant INTERSECT, MINUS, UNION
- **Exemple** : *Donner le nom des vins classés par ordre alphabétique et par année décroissante.*

```
SELECT      DISTINCT CRU, ANNEE
FROM        VINS
ORDER BY    CRU, ANNEE DESC
```

- Produit cartésien restreint aux n-uplets qui vérifient l'expression d'égalité d'attributs.
- Elles s'expriment dans la clause Where.
- Elles permettent de faire le rapprochement de relations sur des attributs dont la sémantique est la même.

- **Exemple** : *Donner les producteurs de Beaujolais*

```
SELECT  NOM, PRENOM
FROM    VINS, RECOLTE, PRODUCTEUR
WHERE   VINS.NV = RECOLTE.NV
        AND PRODUCTEUR.NP = RECOLTE.NP
        AND CRU = 'BEAUJOLAIS'
```

# Exemples (1)

*Donner les buveurs ayant dégusté du vin dont le cru commence par ' B ', de degré inconnu ou compris entre 11 et 13.*

```
SELECT    BUVEURS_*
FROM      BUVEURS_B, VINS, AGOUTE_A
WHERE     (B.NB = A.NB)
          AND (VINS.NV = A.NV)
          AND (VINS.CRU LIKE ' B% ')
          AND ((VINS.DEGRE BETWEEN 11 AND 13)
            OR (VINS.DEGRE IS NULL))
```

## Exemples (2)

*Donner le nom des buveurs ayant dégusté en 1996 une quantité d'alcool supérieure à 10*

```
SELECT    NOM
FROM      BUVEURS
WHERE     NB IN
          (SELECT NB
            FROM   AGOUTE, VINS
            WHERE  (DATE > '1996-01-01')
                  AND (DATE < '1996-12-31')
            GROUP BY NB
            HAVING SUM (QTE * DEGRE / 100) > 10
          )
```