# Introduction aux Bases de données

# Cours 4 : Introduction à SQL

## UFR 919 – Licence

# Plan

- Introduction
- Sélection et projection
- Tri des résultats
- Opérations ensemblistes
- Fonctions numériques, de caractères et de dates

# SQL : Structured Query Language

- Langage d'interrogation pour les BD relationnelles
- Développé chez IBM (1970-80)
- Devenu une norme (ANSI/ISO) en 1986
- Malgré ça, implantations différentes selon SGBD
- Langage déclaratif (basé sur calcul relationnel de tuple)

# L'évolution des sta ndards SQL

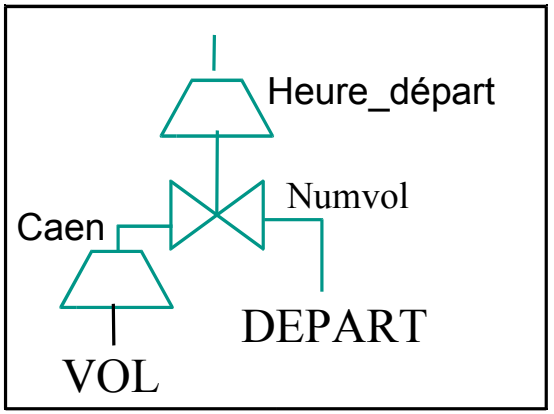
- Début : SQL86
- SQL89 ou SQL1
- SQL92 ou SQL2
- SQL99 ou SQL3 (ajout récursivité, triggers, fonctions OO, types binaires,...)
- SQL2003 (ajout manipulation XML, auto-incrément...)
- SQL2008 (ajout des fonctions de fenêtrage, limite du nombre de résultats, ...)

# Principaux rôles de SQL

1) Définir et modifier le schéma d'une BD
2) Manipuler les données (ajout, suppression, modification)
3) Interroger les données

# D'où vient SQL ?

***THEORIE DES BD***




{V.Heure_Départ/ Vol (V)
et V.Ville_arrivée = 'Caen'
et ∃ D / Départ (D)
et D.Numvol = V.Numvol
et D.Date = '19-12-95'}

**Algèbre relationnelle**

**Autre**

**Calcul de tuples**

**SQL**

# Syntaxe simplifée de SQL

SELECT liste-colonnes
FROM *Table*
[WHERE condition] ;

Retourne

- Les attributs de liste-colonnes
- Des enregistrements de la table *Table*
- qui vérifient condition

La clause where est facultative mais très utile

# Exemples

**Schéma de la BD**

**Emp** (Eno, Ename, Title, City)
**Project**(Pno, Pname, Budget, City)
**Pay**(Title, Salary)
**Works**(Eno, Pno, Resp, Dur)

## Noms de tous les employés

```
SELECT
FROM      ;
```

## Noms et budgets des projets

```
SELECT
FROM      ;
```

# Remarques 1/2

- La clause from déclare les variables (calcul)
  - § Par défaut nom de la relation : from R, S
  - § on peut renommer : from R v1, S v2...
- Pour retourner toutes les colonnes
  - § Select *
- Sémantique « multi-ensembliste »:
  - § Possibilité d'avoir des doublons

    (parce que les éliminer coûte cher, parce qu'on peut vouloir les compter,...)
  - § Les éliminer avec le mot clé **distinct**

    select DISTINCT

# Exemples

**Schéma de la BD**

**Emp** (Eno, Ename, Title, City)
**Project**(Pno, Pname, Budget, City)
**Pay**(Title, Salary)
**Works**(Eno, Pno, Resp, Dur)

Toutes les informations sur les employés

```
SELECT
FROM     ;
```

Toutes les villes où vivent des employés

```
SELECT
FROM     ;
```

L'ensemble des villes où vivent des employés

```
SELECT
FROM     ;
```

# Remarques 2/2

- Possibilité d'exprimer des opérations arithmétiques
  - (att1+att2, att*1.5, etc)
- Possibilité de retourner des chaines entre ' '
- Possibilité de préfixer les attributs par le nom de la table ou une variable
  - Lever les ambiguïtés de noms d'attributs
- Possibilité de renommer une colonne dans le SELECT avec le mot-clé AS
  - Lisibilité des résultats

# Exemples

**Schéma de la BD**

**Emp** (Eno, Ename, Title, City)
**Project**(Pno, Pname, Budget, City)
**Pay**(Title, Salary)
**Works**(Eno, Pno, Resp, Dur)

Salaires mensuel par titre(considérer que Salary est pour un an)

```
SELECT
        FROM      ;
```

Toutes les villes où vivent des employés

```
        SELECT
        FROM     ;
```

Noms et budgets des projets

```
SELECT
        FROM      ;
```

# Exemple

PROJ

| PNO | PNAME | BUDGET |
|---|---|---|
| P1 | Instrumentation | 150000 |
| P2 | Database Develop. | 135000 |
| P3 | *CAD/CAM* | 250000 |
| P4 | Maintenance | 310000 |
| P5 | *CAD/CAM* | 500000 |

SELECT PNO, BUDGET
FROM PROJ :

| PNO | BUDGET |
|---|---|
| P1 | 150000 |
| P2 | 135000 |
| P3 | 250000 |
| P4 | 310000 |
| P5 | 500000 |

SELECT PNAME FROM PROJ :

| PNAME |
|---|
| Database Develop. |
| Instrumentation |
| *CAD/CAM* |
| Maintenance |
| *CAD/CAM* |

SELECT DISTINCT PNAME
FROM PROJ :

| PNAME |
|---|
| Maintenance |
| *CAD/CAM* |
| Database Develop. |
| Instrumentation |

# WHERE : Prédicats

Prédicats simples :

- Expression1 $\theta$ Expression2
  - où Expression1 peut être un attribut ou une expression arithmétique impliquant des attributs, $\theta \in \{<, >, =, <=, >=, <>\}$ et Expression2 une expression ou une valeur de domaine
- Exemples :
  - R.Name = 'J. Doe'
  - (S.Age + 30) >= 65
  - R.A = S.B

Prédicats composés :

- prédicats simples combinés avec les connecteurs logiques AND, OR, NOT

# Exemple de sélection

EMP

| ENO | ENAME | TITLE |
|---|---|---|
| E1 | J. Doe | Elect. Eng. |
| E2 | M. Smith | Syst. Anal. |
| E3 | A. Lee | Mech. Eng. |
| E4 | J. Miller | Programmer |
| E5 | B. Casey | Syst. Anal. |
| E6 | L. Chu | Elect. Eng. |
| E7 | R. Davis | Mech. Eng. |
| E8 | J. Jones | Syst. Anal. |

```
SELECT * FROM EMP
WHERE TITLE = 'Elect. Eng.' ;
```

| ENO | ENAME | TITLE |
|---|---|---|
| E1 | J. Doe | Elect. Eng |
| E6 | L. Chu | Elect. Eng. |

# Requêtes avec prédicats

**Emp** (Eno, Ename, Title, City)
**Pay**(Title, Salary)

**Project**(Pno, Pname, Budget, City)
**Works**(Eno, Pno, Resp, Dur)

Professions qui gagnent plus de 50 000 € par an ?

```
SELECT
FROM
WHERE    ;
```

Numéros des managers d'un projet pendant plus de 17 mois?

```
SELECT
FROM
WHERE    ;
```

# IN, BETWEEN, LIKE

- Appartenance à un ensemble de valeurs :

  *Att IN (Const1, Const2, ...)*

- Appartenance à un intervalle de valeurs :

  *Att BETWEEN Constante1 AND Constante2*

- Ressemblance à un motif :

  Att LIKE 'MOTIF'

  § où MOTIF combine des chaînes et des joker
  - % pour une chaîne quelconque (y compris vide)
  - _ pour un caractère quelconque et un seul

# Requêtes avec prédicats (2)

**Emp** (Eno, Ename, Title, City)
**Pay**(Title, Salary)

**Project**(Pno, Pname, Budget, City)
**Works**(Eno, Pno, Resp, Dur)

Nom des projets de Paris, Lyon ou Nantes ?

```
SELECT Pname
FROM   Project
WHERE  City  (...) ;
```

*Comment le faire sans IN ?*

Nom des projets ayant un budget compris entre 5M et 10M euros?

```
SELECT Pname
FROM   Projet
WHERE  Budget  ;
```

*Comment le faire sans BETWEEN ?*

# Requêtes avec prédicats (3)

**Emp** (Eno, Ename, Title, City)
**Pay**(Title, Salary)

**Project**(Pno, Pname, Budget, City)
**Works**(Eno, Pno, Resp, Dur)

Nom des employés commençant pas C?

```
SELECT Ename
FROM   Emp
WHERE  Ename LIKE ... ;
```

Nom des employés dont le 2ème numéro est un 5?

```
SELECT Ename
FROM   Emp
WHERE  Eno LIKE ... ;
```

*Nom des employés habitant une ville composé de 2 mots (ex : Chatenay Malabry)?*

# Valeurs nulles

- La valeur de certains attributs peut
  - ne pas être connue (ex. : année de construction du Louvre)
  - ou ne pas avoir de sens (ex. : nom de jeune fille pour un homme)

on parle alors de valeurs nulles (mot-clé NULL)

- NULL n'est pas une valeur mais une absence de valeur! Les opérations ou les comparaisons ne peuvent lui être appliqué

- Toute opération (+,-,/,*,substr, to_char,... ) appliquée à NULL donne NULL

- Toute comparaison avec NULL donne ni vrai ni faux, mais INCONNU

  Notions de sémantique Tri-Valuée abordées en cours 8 : compléments SQL

# Syntaxe du tri

```
SELECT liste-colonnes
FROM nomtable
WHERE condition
ORDER BY liste-colonnes ;
```

- Dans la clause ORDER BY, on peut avoir des :
  - des noms de colonnes
  - des expressions avec noms de colonnes
  - des numéros de position des colonnes dans la clause SELECT.
- On précise le sens : ASC (par défaut) ou DESC
- Les valeurs nulles sont à la fin par ordre croissant, au début par ordre décroissant.

# Exemple de tri

**Emp** (Eno, Ename, Title, City)
**Pay**(Title, Salary)

**Project**(Pno, Pname, Budget, City)
**Works**(Eno, Pno, Resp, Dur)

Noms, budgets et villes des projets de budget supérieur à 250 000 euros, en ordonnant le résultat par ordre décroissant de budget puis par nom par ordre alphanumérique croissant ?

```
SELECT Pname, Budget, City
  FROM   Project
  WHERE Budget > 250000
  ORDER BY ... ;
```

# Exemple de tri (2)

**Emp** (Eno, Ename, Title, City)
**Pay**(Title, Salary)

**Project**(Pno, Pname, Budget, City)
**Works**(Eno, Pno, Resp, Dur)

Noms, budgets TTC (TVA 20%) et villes des projets, en ordonnant le résultat par ordre décroissant de budget TTC ?

```
SELECT Pname, Budget*1.2, City
 FROM   Project
 WHERE  ...
 ORDER BY 2 DESC ;
```

Noms, budgets et villes des projets en ordonnant le résultat par ordre décroissant de budget TTC ?

```
SELECT Pname, Budget, City
 FROM   Project
 WHERE  Budget > 250000
 ORDER BY ... ;
```

# Opérations ensemblistes

On peut réaliser des opérations ensemblistes sur les clauses SELECT.

*3 opérations ensemblistes*

| | |
|---|---|
| UNION | union de deux ensembles |
| INTERSECT | intersection de deux ensembles |
| MINUS | différence de deux ensembles (norme : EXCEPT) |

# Principe

Pour les opérations ensemblistes :

- Pas de lien entre les objets sélectionnés dans les 2 requêtes
- Même schéma dans les SELECT des deux requêtes : c'est à dire même nombre d'attributs et chacun du même type (par forcément le même nom)
- Le schéma en sortie correspond au schéma de la première requête
- Par défaut, les opérations ensemblistes éliminent les doublons (ensemble). Pour garder les doublons (multi-ensemble), il faut ajouter ALL après l'opérateur : UNION ALL, EXCEPT ALL, INTERSECT ALL

# UNION

**Emp** (Eno, Ename, Title, City)
**Pay**(Title, Salary)

**Project**(Pno, Pname, Budget, Town)
**Works**(Eno, Pno, Resp, Dur)

Noms des villes où habitent des employés ou où sont localisés des projets?

```
SELECT City
  FROM   Emp
...
SELECT Town
 FROM Projet ;
```

# INTERSECTION

**Emp** (Eno, Ename, Title, City)
**Pay**(Title, Salary)

**Project**(Pno, Pname, Budget, Town)
**Works**(Eno, Pno, Resp, Dur)

Noms des villes où habitent des employés et où sont localisés des projets?

```
SELECT City
  FROM   Emp
...
SELECT Town
 FROM Projet ;
```

# DIFFERENCE

**Emp** (Eno, Ename, Title, City)
**Pay**(Title, Salary)

**Project**(Pno, Pname, Budget, Town)
**Works**(Eno, Pno, Resp, Dur)

Noms des villes où habitent des employés mais où n'est localisé aucun projet?

```
SELECT City
  FROM   Emp
...
SELECT Town
 FROM Projet ;
```

# Fonctions

De nombreuses fonctions existent pour :

- Manipuler les dates et les temps
- Manipuler les chaînes de caractères
- Manipuler les chiffres

Cependant, bien qu'une norme existe, elles diffèrent souvent d'un SGBD à l'autre...

Quelques exemples compatibles avec la syntaxe H2 →

# Dates

- Les dates ont un format de stockage optimisé et un format d'affichage/saisie par défaut qui dépend du SGBD
- Sous H2
  - Création à partir d'une chaine de caractères
  - Restitution sous forme d'une chaine de caractères
  - Extraction de composants (jour, mois, année, …)
  - Manipulation (ajout d'un nombre de jour/mois/annee/..., différence)

# Création d'une date

PARSEDATETIME('chaîne', 'masque')

- Fournir une chaine et un masque
- Masque précise le format :
  - composants (année, mois, ….), suivent une syntaxe fixée
    - y : année, M : mois, d : jour, H : heure, m : minute, s : seconde
    - Respecter la casse !
  - Séparateurs (/, -, …) choisis par l'utilisateur

Exemple : parsedatetime('01-02-2022', 'dd-MM-yyyy')

par défaut heure 00:00:00

# Création d'une date

DATE 'chaîne'

- Pas d'heure, prend en considération le format système

  DATE '2022-02-01'
  produit la date du 1er février 2022

# Restitution en chaine de caractères

FORMATDATETIME(date, 'masque')

- Fournir une date et un masque
- Masque défini comme pour la création
- Exemple
  - FORMATDATETIME(CURRENT_DATE,'dd-MM-yyyy')

    retourne la date courante au format indiqué

# Extraction de composants

- Utiliser EXTRACT(composant FROM date)
- Ou utiliser les fonctions suivantes (plus simple)

| Composant | Fonction | Type | Exemple |
|---|---|---|---|
| Année | YEAR | nombre | YEAR(current_date) → 2022 |
| Mois | MONTH | nombre | MONTH(current_date) → 2 |
| Jour du mois | DAY_OF_MONTH | nombre | DAY_OF_MONTH(current_date) → 8 |
| Jour de la semaine * | DAY_OF_WEEK | nombre | DAY_OF_WEEK(current_date) → 3 |
| Nom du jour | DAYNAME | texte | DAYNAME(current_date) → Tuesday |
| Semaine de l'an | WEEK | Nombre | DAY_OF_WEEK(current_date) → 7 |

* le 1er jour est dimanche

Quizz : retrouver le jour de la semaine du 05-02-1992

# Manipulation de date

- Ajout d'un nombre de jour/semaines/mois/année

  DATEADD(datetimeField, addIntLong, dateAndTime)

  Ex. DATEADD(MONTH, 1, DATE '2022-02-01') → 2022-03-01

- Différence entre deux dates en terme de années/mois/jours/heures/minutes/secondes

  DATEDIFF(datetimeField, aDateAndTime, bDateAndTime) *si a uit chronologiquement b, retourne un nombre négatif*

  Ex. call DATEDIFF(day, DATE '2022-02-01', current_date) → 7

# Aller plus loin

- Diverses familles de fonctions
  - Dimensions spatiales et temporelles
  - Manipulation de textes, nombres, …
  - Consulter la doc. pour une liste exhaustive
- Possibilité de définir des fonctions utilisateurs → utilisateurs avancés
- Programmation en lien avec SQL : PL/SQL
  - Supporté par certains systemes (Oracle, postgresql,...)
  - dernier cours