



Global Knowledge.®



Support de cours



Global Knowledge®

GKSQL Les bases du SQL Partie 1

WORLDWIDE LOCATIONS

BELGIUM CANADA COLOMBIA DENMARK EGYPT FRANCE IRELAND JAPAN KOREA MALAYSIA MEXICO NETHERLANDS NORWAY QATAR
SAUDI ARABIA SINGAPORE SPAIN SWEDEN UNITED ARAB EMIRATES UNITED KINGDOM UNITED STATES OF AMERICA

Objectifs de la formation

Comprendre : - la conception des bases de données relationnelles
 - la structure des tables, les vues, les index
 - les clés primaires et étrangères du modèle relationnel.

Être capable de faire des consultations, à l'aide du langage SQL,

- sur 1 table
- sur plusieurs tables d'une base de données relationnelle,
- d'effectuer des statistiques.

Effectuer des mises à jour de données (création, modification et suppression de lignes), des créations de vues.

Contenu de la formation

Jour 1

- Les systèmes de gestions de bases de données – Historique, vocabulaire
- Principes et concepts du modèle relationnel
- Conception d'une base relationnelle
- Le langage SQL – Requêtes sur une table

Jour 2

- Suite du langage SQL – Requêtes sur plusieurs tables - Jointure
- Les jointures externes
- Sous-requête

Jour 3

- Les vues
- Les fonctions d'agrégation
- Présentation des ordres de mises à jour – Conséquences des intégrités

I – Les bases du SQL

Les systèmes de gestion de base de données



Les systèmes de gestion de base de données - Définition

Base de données définition du JO :

Ensemble de données organisées, en vue de leur utilisation par des programmes correspondant à des applications distinctes et de manière à faciliter l'évolution indépendante des données et des programmes.

Cette définition :

Exclut les bases de 1^{ère} génération.

Indique l'accès aux données que via des programmes.

Ne parle pas du SGBD.

Une base n'est rien sans son SGBD.

Le SGBD est un

Outil logiciel assurant l'interface Utilisateur/Base de données.

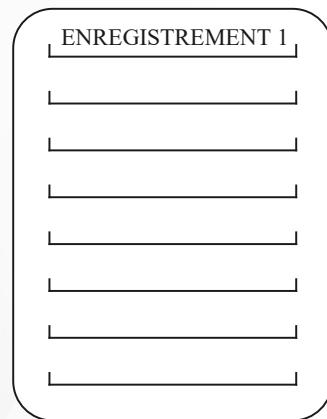
Les systèmes de gestions de bases de données - Historique

Dans les bases de données il y a deux informations :

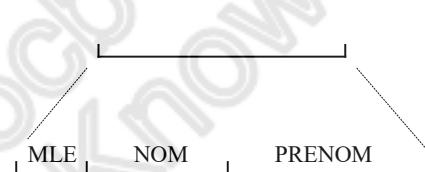
Le **STOCKAGE** des données et **l'INTERROGATION**.

Première génération : Fichier – Enregistrement - rubrique

FICHIER DES EMPLOYES



ENREGISTREMENT 1



Ex : Répertoire téléphonique

| Clé | Valeur |
|--------|--------|
| Alain | 01671 |
| Albert | 07123 |
| Aline | 06839 |
| Carine | 06789 |
| Claude | 02901 |

Les systèmes de gestion de fichiers : fichiers **séquentiels**
fichiers séquentiels **indexés**

→ On a conservé le terme d'index.

Les systèmes de gestion de base de données - Historique

Les bases de données ont connu des évolutions majeures :

- hiérarchiques (IBM - 1960),
- réseaux (CODASYL - 1970),
- relationnelles (IBM - 1980).



IMS/DL1 1968
IDMS 1972 - BULL IDS2

Langages
spécifiques à
chaque SGBD.

Deuxième génération : LA THEORIE DES ENSEMBLES ET LE MODELE RELATIONNEL

- Indépendance totale entre les données et les traitements.
- Les données ne sont plus reliées physiquement.
- **Langage assertional** (SQL par exemple)

SQL

Les systèmes de gestion de base de données - Historique

Troisième génération : LES BASES DE DONNEES UNIVERSELLES

SQL

Stockage de données de type: image, son, vidéo, ...

Stockage de traitements: procédures stockées, Triggers

Ce sont les bases de données relationnelles deuxième génération qui évoluent vers la troisième génération.

DB2 – UDB IBM

Oracle

SQLServer Microsoft

OpenSource :

MySQL MariaDB

PostgreSQL

Vers une quatrième génération avec les bases orientées objets ?

Vers de nouveaux concepts, avec le Big Data et le NOSQL Not Only SQL

Les systèmes de gestion de base de données– Fonctions du SGBD/R

Description du Modèle logique des données dans la base (DDL)

Manipulation des données (DML)

Garantie d'Intégrité des données (pas de clés en double, etc)

Gestion de la Confidentialité des données

Gestion des Concurrences d'accès aux données

Sécurité de fonctionnement et possibilités de restauration en cas d'incident

Outils d'Analyse des performances

Gestion de la Documentation concernant les données

Interface de développement

Les systèmes de gestion de base de données – Fonctions du SGBD/R

Cette liste de fonctions varie selon les éditeurs, les utilisateurs, etc, d'où la difficulté d'établir une définition des SGBD.

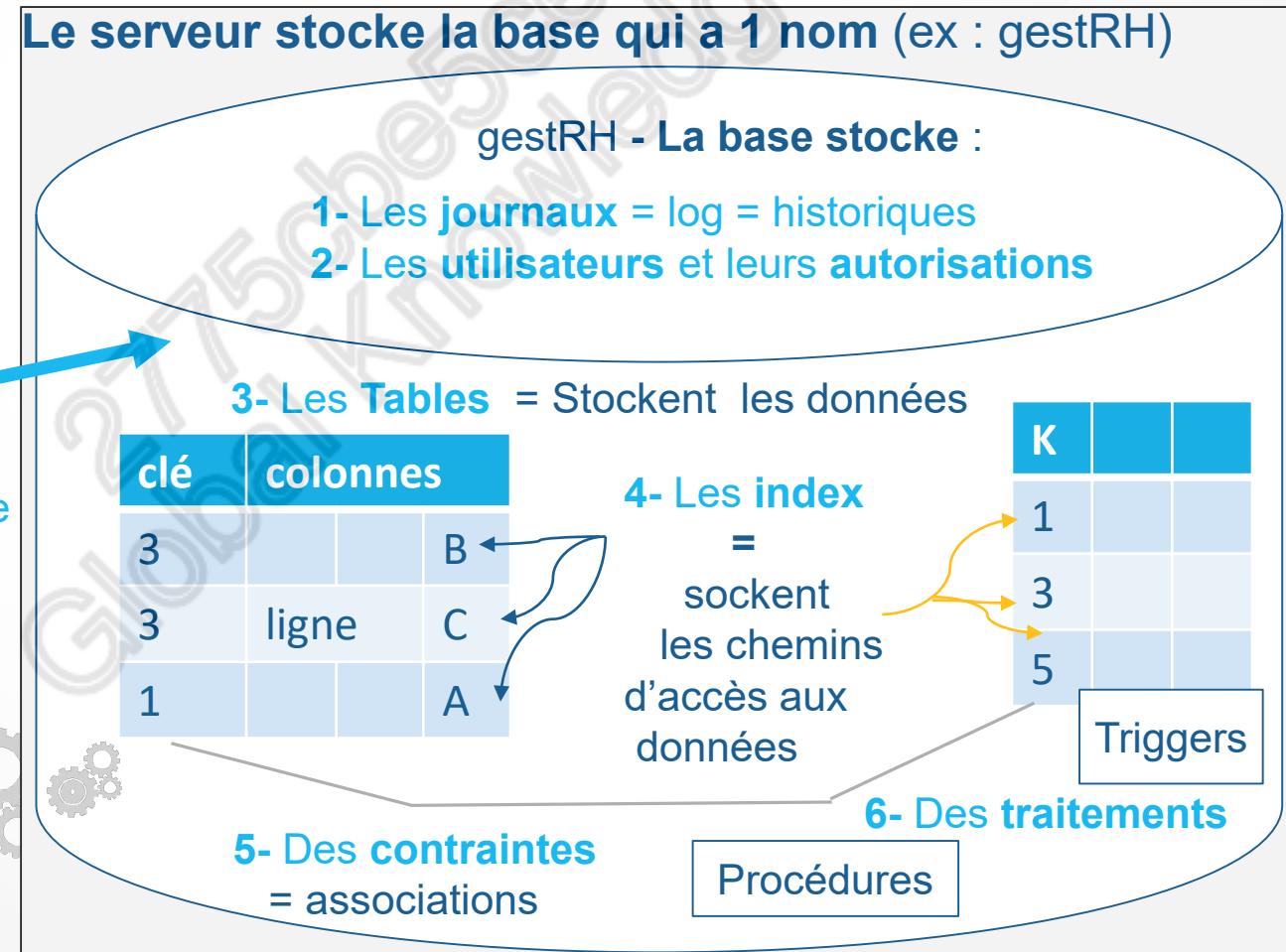
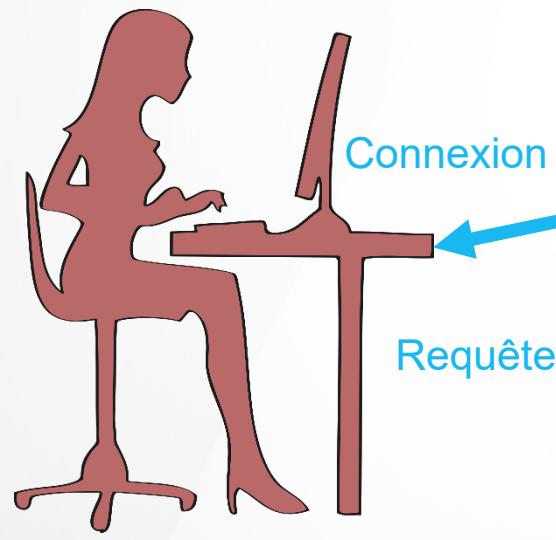
Dès 1970 Edward CODD, chercheur chez IBM, a défini un modèle d'organisation des bases de données fondé sur des bases mathématiques, connu depuis sous le nom de modèle relationnel.

En 1985, Codd a édicté 13 règles permettant de définir ce que devrait être un SGBD conforme au Modèle Relationnel. Aujourd'hui, ce sont 333 fonctionnalités.

Bien qu'aucun des SGDB actuellement disponibles sur le marché ne satisfasse ces 13 règles, certains des plus diffusés tentent d'en satisfaire le maximum.

Les systèmes de gestion de base de données – Terminologie du Relationnel

SGBDR = Logiciel pouvant gérer plusieurs bases de données dans 1 entreprise,
Interface entre l'homme et les données



II – Les bases du SQL

Principes et concepts du modèle
relationnel



Principes et concepts du modèle relationnel - Domaine

Ensemble de valeurs sémantiquement homogènes :

DOM_PRÉNOMS = {Paul, Pierre, André}

DOM_FONCTIONS = {P, CDP, AP}

Même si dans les bases ces domaines sont stockés sous forme de chaîne de caractères,
il est impossible de comparer ces deux domaines.

→ On utilise les types de données :

Numérique

Alphanumérique

Date

int

Date et heure

LOB

Double

Principes et concepts du modèle relationnel - Relation

Relation = Sous-ensemble du produit cartésien d'une liste de domaines, tel que chaque ligne satisfait un critère de vérité du modèle à mettre en œuvre.

Produit cartésien de domaines :
DOM_PRÉNOMS = {Paul, Pierre, André}
DOM_FONCTIONS = {P, CDP, AP}
Relation(DOM_PRÉNOMS - DOM_FONCTIONS)
Relation Salarié (le réel)

| Prénom | Fonction |
|--------|----------|
| Paul | CDP |
| Pierre | AP |
| André | P |

| Prénom | Fonction |
|--------|----------|
| Paul | P |
| Paul | CDP |
| Paul | AP |
| Pierre | P |
| Pierre | CDP |
| Pierre | AP |
| André | P |
| André | CDP |
| André | AP |

Principes et concepts du modèle relationnel - Relation

| SALARIE | Prénom | Fonction |
|--------------------|--------|----------|
| | Paul | CDP |
| | Pierre | AP |
| Ligne (ou tuple) ➔ | André | P |

↑
Colonne ou Attribut

La Relation n'est pas un synonyme de **table**. La table est une implémentation **physique** de la relation, on parlera d'autres relations nommées des **vues**.

Degré d'une relation

: nombre d'attributs



Peu utilisé
Différent en modélisation

Cardinalité d'une relation

: nombre de lignes

Schéma d'une relation

: nom de la relation, suivi de la liste de ses attributs.

Schéma de la relation SALARIÉ : SALARIÉ (Prénom, fonction)

L'ordre des lignes et des colonnes est indifférent .

Principes et concepts du modèle relationnel – Première FORME NORMALE

Pour une ligne et un attribut donnés, on ne peut trouver au plus qu'une valeur atomique

Table Personnage

Table NON 1NF →

Table 1NF →

| Nom | Prénom_des_enfants |
|-----------|----------------------|
| MAHEU | Catherine, Jeanlin |
| VILLEMEUR | Jean, Élise, Thierry |

| Nom | Prénom_des_enfants |
|-----------|--------------------|
| MAHEU | Catherine |
| MAHEU | Jeanlin |
| VILLEMEUR | Jean |
| VILLEMEUR | Élise |
| VILLEMEUR | Thierry |

Principes et concepts du modèle relationnel – Valeurs nulles

"Nulle" ne veut pas dire égale à zéro, mais non connue :

Soit l'attribut **n'a pas de sens (I-MARK)** soit l'attribut **est inconnu (A-MARK)**

| PERSONNAGE | Nom | Prénom_enfant | DATE_NAISS | DATE_BAC |
|------------|-----------|---------------|------------|------------|
| | MAHEU | Catherine | 16-05-1870 | I-mark |
| | MAHEU | Jeanlin | 23-07-1875 | 30-06-1895 |
| | VILLEMEUR | Jean | 17-03-1928 | A-mark |
| | VILLEMEUR | Élise | A-mark | 28-06-1949 |
| | VILLEMEUR | Thierry | 05-04-1933 | 03-07-1953 |

En SQL, on a qu'un seul NULL testé avec l'opérateur IS et IS NOT

| Prénom | Fonction | Salaire | Prime |
|--------|----------|---------------|---------------|
| Paul | CDP | 4000 | 3000 |
| Pierre | AP | Null (A-Mark) | Null (A-Mark) |
| André | P | 2000 | Null (I-MARK) |

Intérêts:

Les valeurs par défaut peuvent varier au cours du temps et au gré des humeurs.
Les calculs sur les colonnes ignorent les « null » et évitent des résultats faux

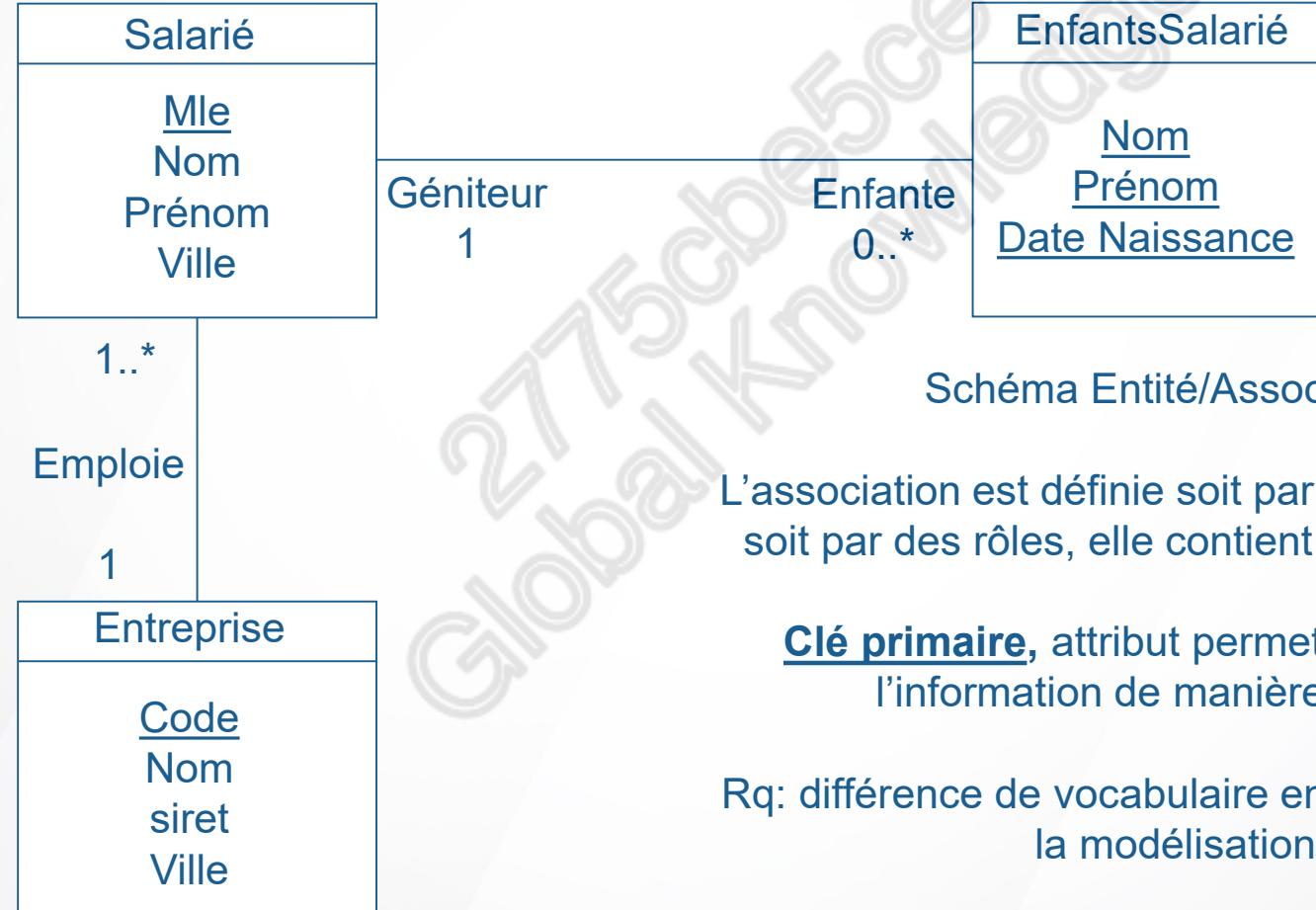
Principes et concepts du modèle relationnel – Modélisation - Clé primaire

Intérêt : La clé primaire permet d'identifier de manière unique une ligne dans la table

| Matricule | Prénom | Fonction | Salaire |
|-----------|--------|----------|---------|
| 00001 | Paul | CDP | 4000 |
| 00002 | Pierre | AP | 3000 |
| 00003 | André | P | 2000 |

Principes et concepts du modèle relationnel – Modélisation

La modélisation permet d'avoir un langage commun

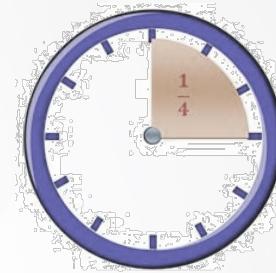


Principes et concepts du modèle relationnel – Exercice de modélisation

Gestion des Services d'hospitalisation d'un hôpital.

On a :

- Des services, ils ont un numéro, un nom et un nombre de lits.
- Des patients, ils ont un nom et un numéro d'entrée dans le service.
- Des médecins, ils ont un numéro d'ordre et un nom.
- Les patients sont hospitalisés dans un service et sont suivis par des médecins.



Représenter via des Entités et des Relations cette gestion, choisir soit un verbe pour les relations, soit des rôles.

Indiquer : les cardinalités,

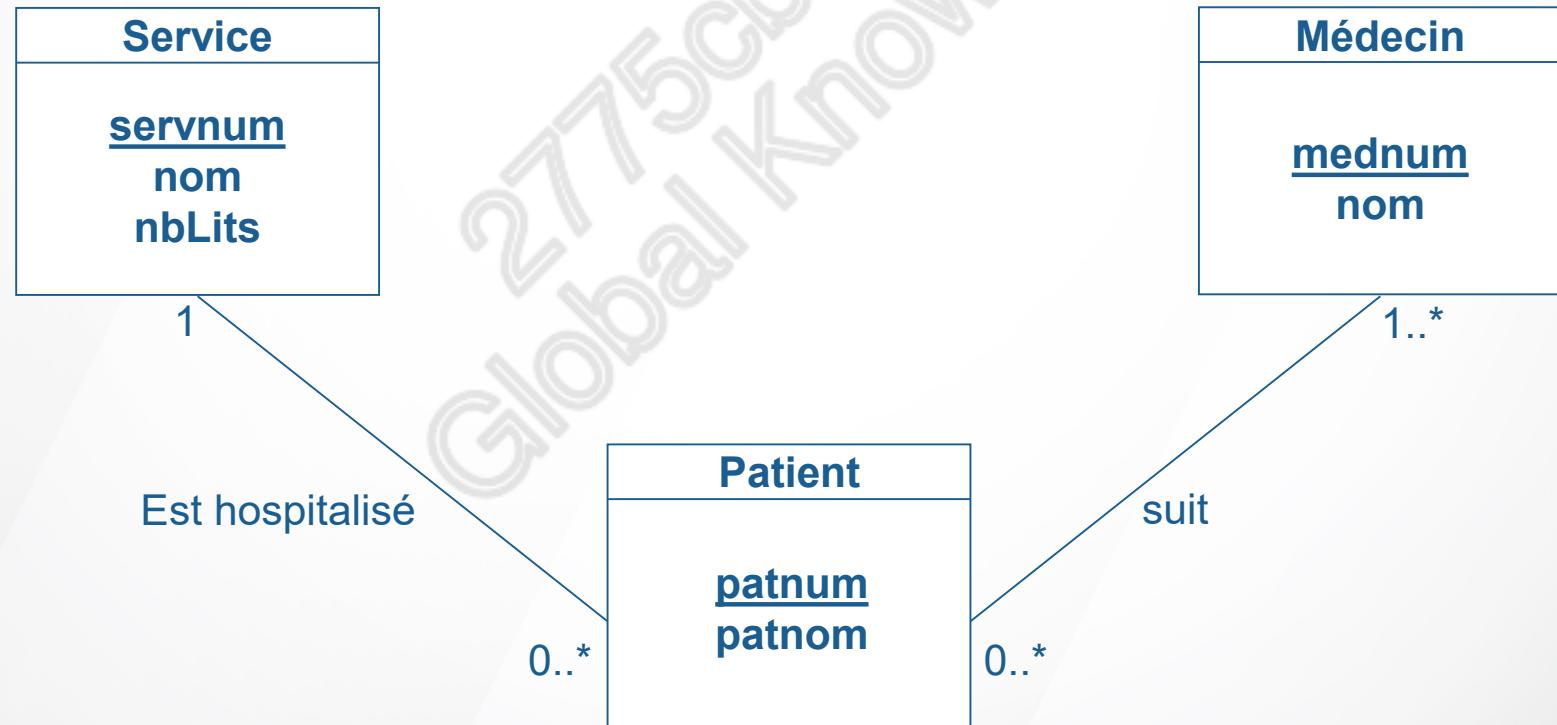
les clés primaires pour chaque entité, ainsi que leurs attributs.

Principes et concepts du modèle relationnel – Modélisation Correction

Gestion des Services d'hospitalisation d'un hôpital

Rq : on est tenté de faire une relation entre Service/Médecin mais on n'a aucune info, donc on ne fait rien.

Service/Patient si on met 1 cardinalité 1..* cela veut dire que le dernier patient doit rester dans le service donc 0..n. On n'aborde pas la notion de temps, ni du nombre de lits.



Principes et concepts du modèle relationnel – Modélisation Correction

Schéma Entité/Relation :

Chaque Entité donne lieu à une table. Pour la relation ayant une cardinalité 1 on parle de clé étrangère.

Schéma relationnel:

Service(servnum, nom, nblits)

Médecin(**mednum**, nom)

Patient(patnum, patnom, servnum)

→ servnum est clé étrangère.

Comment représenter la relation « suit » entre Médecin/Patient ?

Principes et concepts du modèle relationnel – Modélisation Correction

Comment représenter la relation « suit » entre Médecin/Patient :

Clé étrangère impossible les cardinalités sont * - *

On crée donc une nouvelle table

SUIT(mednum,patnum)

Quelle est la clé primaire ?

Principes et concepts du modèle relationnel – Modélisation

La modélisation permet d'avoir un langage commun et permet d'avoir une vision globale des tables

| Salarié | |
|----------|---|
| Mle | |
| Nom | |
| Prénom | |
| Ville | |
| Géniteur | 1 |

| EnfantsSalarié | |
|----------------|------|
| Nom | |
| Prénom | |
| Date Naissance | |
| Enfante | 0..* |

| Mle | Nom | Prénom | Ville |
|-----|-----------|--------|-------|
| 10 | MAHEU | Jean | Paris |
| 15 | Martin | Iea | Lyon |
| 20 | VILLEMEUR | Paul | Brest |

| Nom | Prénom_des_enfants | Date Naiss | Mle parent |
|-----------|--------------------|------------|------------|
| MAHEU | Catherine | 10-04-1995 | 10 |
| MAHEU | Jeanlin | 25-04-1999 | 10 |
| VILLEMEUR | Jean | 21-08-1989 | 20 |
| VILLEMEUR | Élise | 21-08-1989 | 20 |
| VILLEMEUR | Thierry | 10-04-1995 | 20 |

Quelle clé primaire ?

Principes et concepts du modèle relationnel – Dépendance fonctionnelle

L'attribut X détermine l'attribut Y

(A chaque valeur de X correspond au + 1 valeur de Y)

| PERSONNAGE | Nom | Prénom enfant | DATE NAISS | DATE BAC |
|------------|-----------|---------------|------------|----------|
| MAHEU | Catherine | 16-05-1870 | "nulle" | |
| MAHEU | Jeanlin | 23-07-1875 | 30-06-1895 | |
| VILLEMEUR | Jean | 17-03-1928 | "nulle" | |
| VILLEMEUR | Élise | "nulle" | 28-06-1949 | |
| VILLEMEUR | Thierry | 05-04-1933 | 03-07-1953 | |

```
graph TD; A[Attribut X détermine l'attribut Y] --> B[PERSONNAGE]; A --> C[Nom]; A --> D[Prénom enfant]; A --> E[DATE NAISS]; A --> F[DATE BAC]; G[PERSONNAGE] --> H[Row];
```

Dans cet exemple, les dépendances fonctionnelles sont ←

Elémentaire = Le plus petit nombre d'attributs d'une relation qui détermine l'ensemble.

Principes et concepts du modèle relationnel – Les clés

Clé candidate:

Ensemble minimum d'attributs qui détermine tous les autres.

Elle peut être simple (mono-attribut) ou multi-attributs.

Clé primaire :

Clé choisie parmi les clés candidates.

→ Aucun des attributs de la clé primaire ne peut prendre la valeur nulle.

Principes et concepts du modèle relationnel - La normalisation

Utilité de la normalisation → Eviter les redondances, car coûteuses en cas de māj

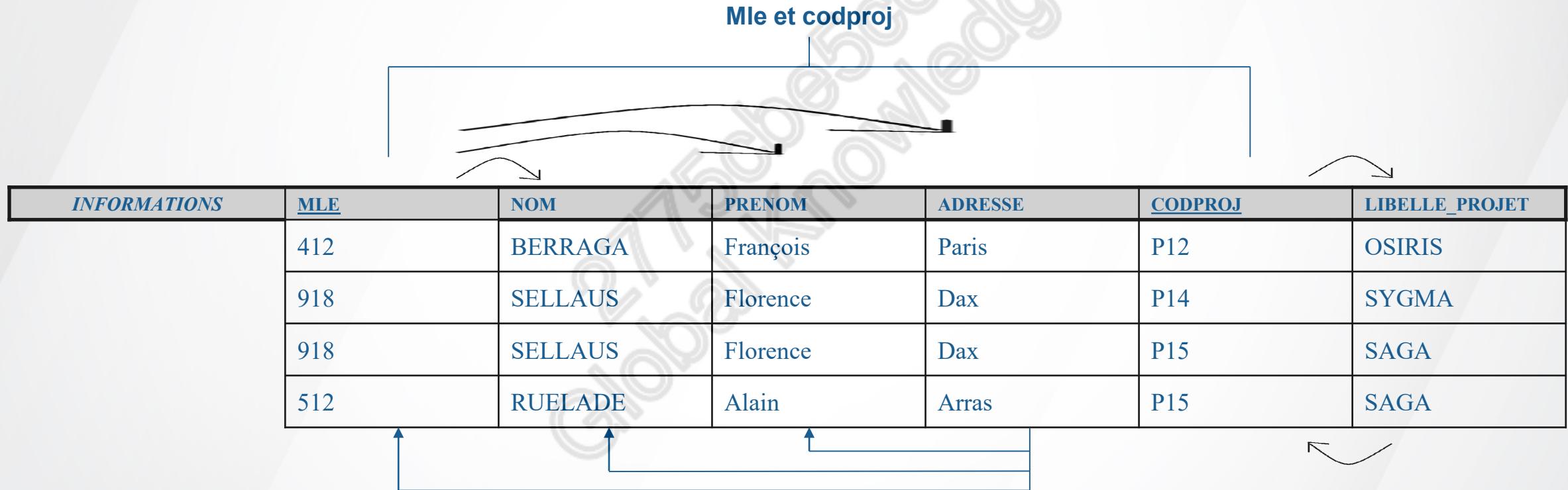
Pourquoi est-ce en première forme normale ?

| INFORMATIONS | MLE | NOM | PRENOM | ADRESSE | CODPROJ | LIBELLE_PROJET |
|--------------|-----|---------|----------|---------|---------|----------------|
| | 412 | BERRAGA | François | Paris | P12 | OSIRIS |
| | 918 | SELLAUS | Florence | Dax | P14 | SYGMA |
| | 918 | SELLAUS | Florence | Dax | P15 | SAGA |
| | 512 | RUELADE | Alain | Arras | P15 | SAGA |

Quelles sont les dépendances fonctionnelles élémentaires ?

Principes et concepts du modèle relationnel - La normalisation

1NF : à l'intersection d'une ligne et d'une colonne on ne trouve qu'1 valeur au plus.



Mle et codproj détermine l'ensemble des autres attributs

Principes et concepts du modèle relationnel – La normalisation

La normalisation consiste à éliminer les redondances induites par le respect de la première forme normale

=

décomposer une relation en plusieurs plus petites.



6 étapes de Normalisation

Nous irons jusqu'à la troisième forme normale.

Principes et concepts du modèle relationnel – Deuxième forme normale 2NF

Une relation est en 2FN si :

Elle est en 1FN

et si tout attribut non clé est en dépendance fonctionnelle élémentaire avec la totalité de la clé primaire.

| INFORMATIONS | MLE | NOM | PRENOM | ADRESSE | CODPROJ | LIBELLE_PROJET |
|--------------|-----|---------|----------|---------|---------|----------------|
| | 412 | BERRAGA | François | Paris | P12 | OSIRIS |
| | 918 | SELLAUS | Florence | Dax | P14 | SYGMA |
| | 918 | SELLAUS | Florence | Dax | P15 | SAGA |
| | 512 | RUELADE | Alain | Arras | P15 | SAGA |

MLE et codproj détermine l'ensemble des autres attributs

| SALARIÉ | MLE | NOM | PRENOM | ADRESSE |
|---------|---------|----------------|--------|---------|
| | | | | |
| PROJET | CODPROJ | LIBELLE_PROJET | | |
| | | | | |

| AFFECTATION | MLE | CODPROJ |
|-------------|-----|---------|
| | | |

On a une relation supplémentaire

Principes et concepts du modèle relationnel – Troisième forme normale 3NF

Une relation est en 3FN si :

Elle est en 2FN

et si aucun attribut non clé ne dépend d'un autre attribut non clé primaire.

A un grade donné ne correspond qu'un salaire. Dépendance fonctionnelle ?

| SALARIÉ | MLE | NOM | PRENOM | ADRESSE | GRADE | SALAIRE |
|---------|-----|-----------|----------|-----------|-------|---------|
| | 412 | BERRAGA | François | Paris | G3 | 100 |
| | 918 | SELLAUS | Florence | Dax | G3 | 100 |
| | 512 | RUELADE | Alain | Arras | G6 | 110 |
| | 492 | COURTNISE | Sophie | Lille | G3 | 100 |
| | 178 | GUENGUE | Serge | Rochefort | G3 | 100 |
| | 264 | GYRON | Yves | Paris | G6 | 110 |

Respect de la 3NF → Décomposition en deux tables

| SALARIÉ | MLE | NOM | PRENOM | ADRESSE | GRADE |
|---------|-----|-----|--------|---------|-------|
| | | | | | |

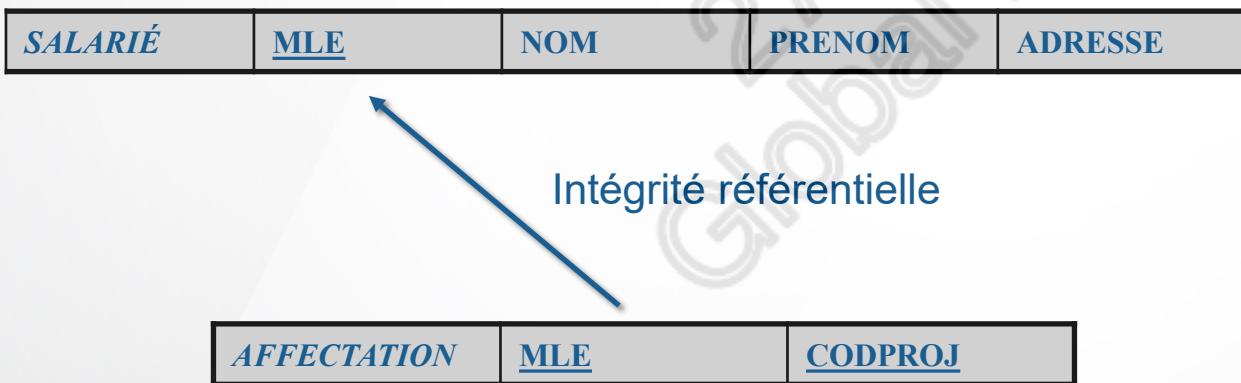
| RÉMUNÉRATION | GRADE | SALAIRE |
|--------------|-------|---------|
| | | |

Principes et concepts du modèle relationnel – Récapitulatif de la normalisation

Une relation est en 3FN quand :

- 1- Elle a une clé primaire
- 2- Tout attribut non clé dépend

de la clé
uniquement de la clé (2NF)
rien que de la clé (3NF).



Principes et concepts du modèle relationnel – Intégrité référentielle



Tables parentes

Clé primaire de salarié : **MLE**

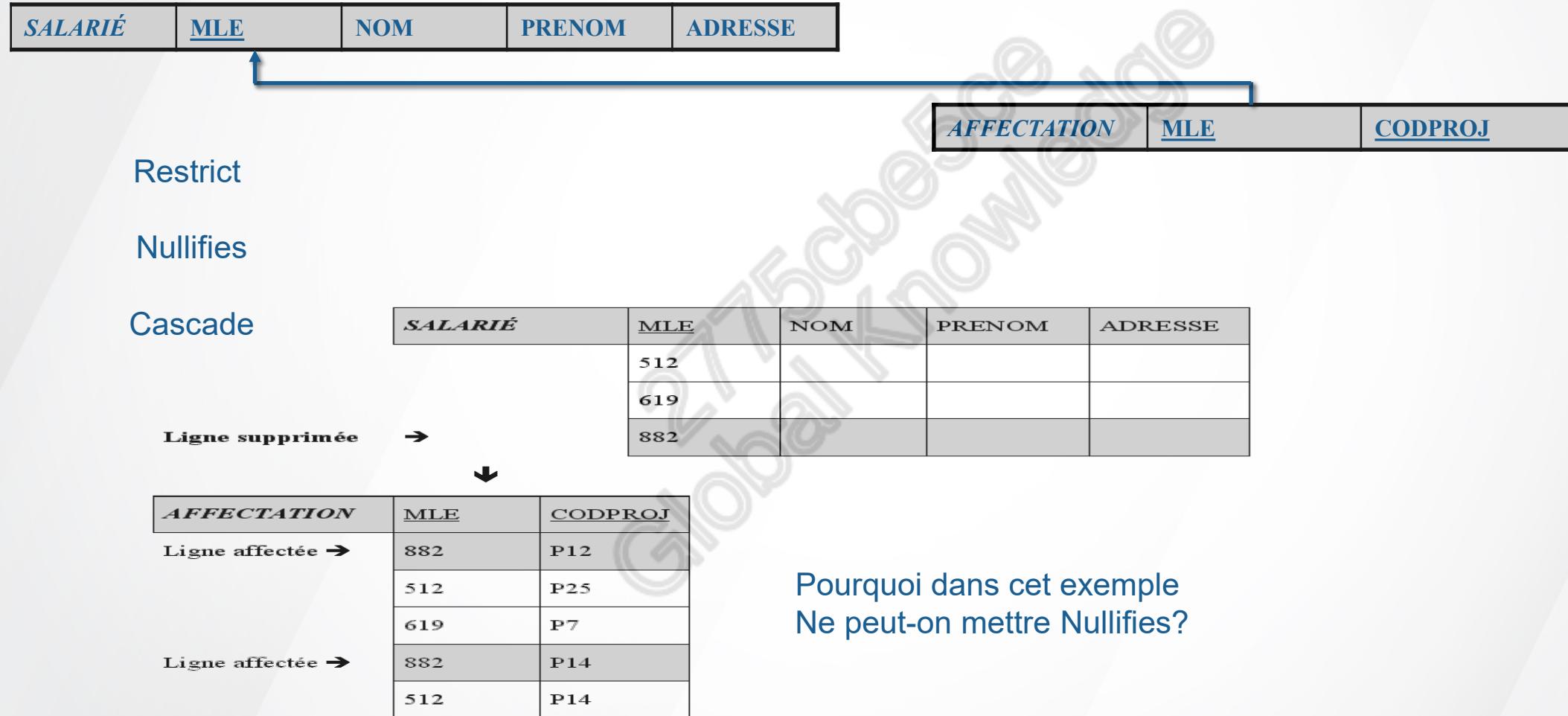
Clé primaire de projet : **Codproj**

Table dépendante

Clé primaire d'affectation : **MLE et Codproj**

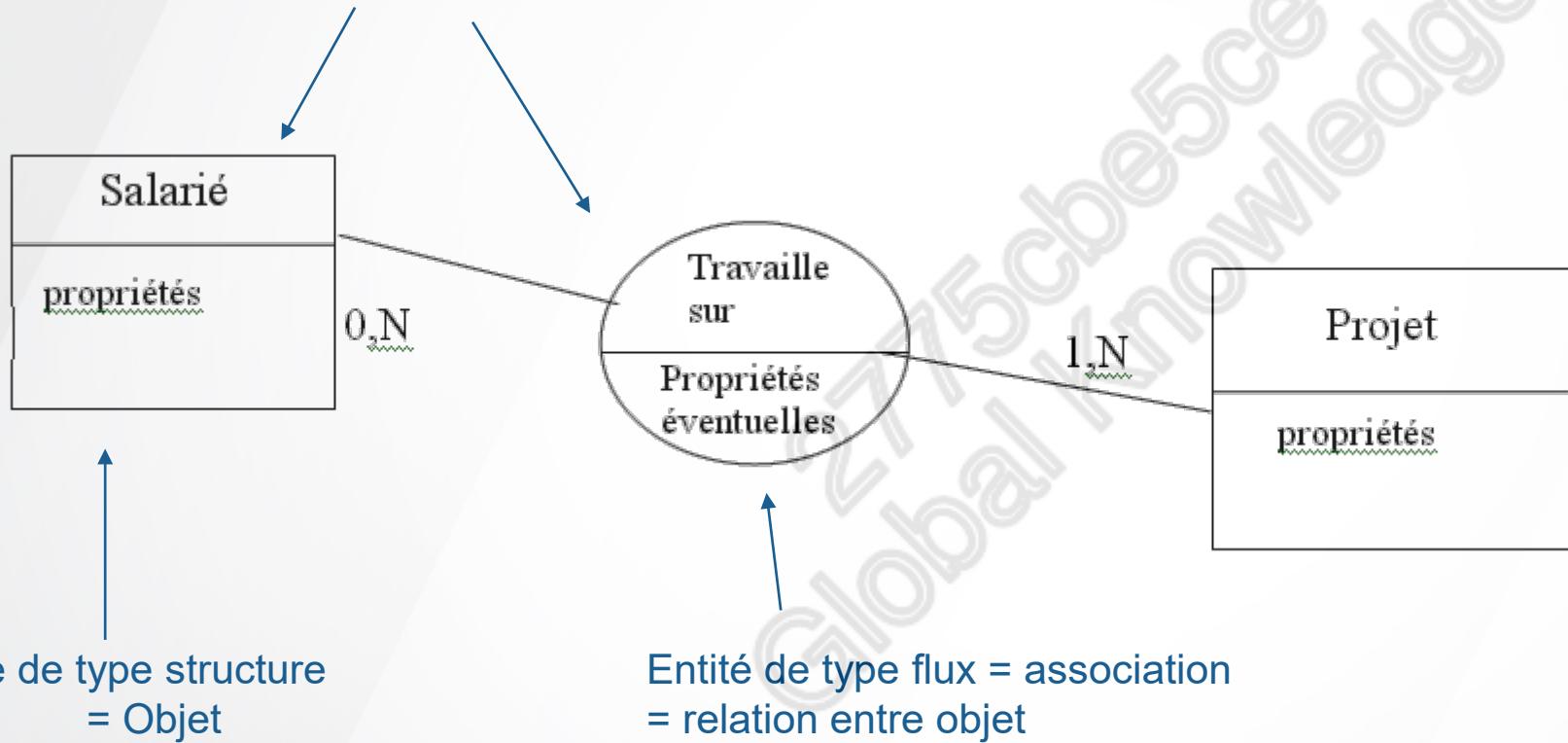
MLE d'affectation est une clé étrangère
Codproj aussi

Principes et concepts du modèle relationnel – Intégrité référentielle - Métabolisme

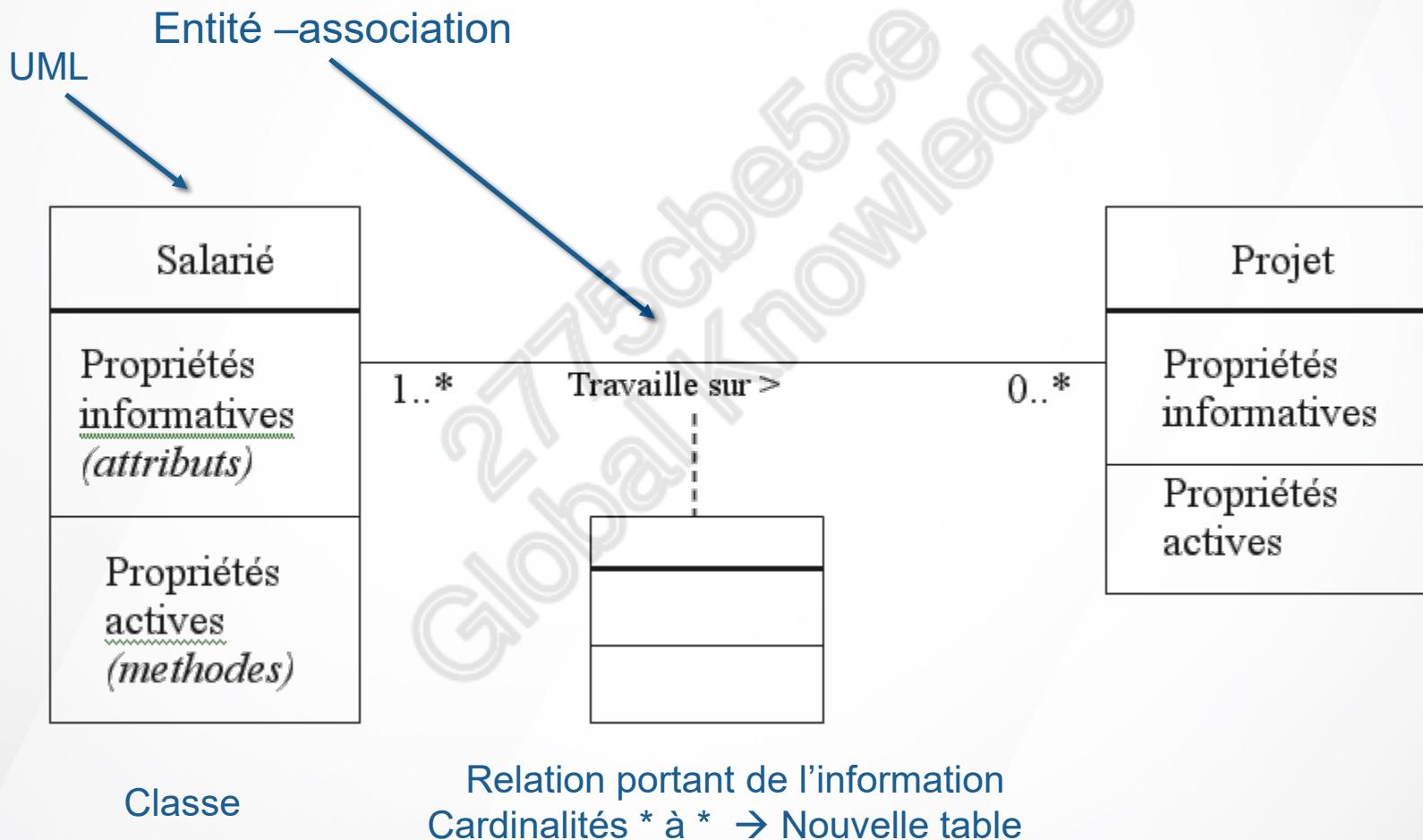


Principes et concepts du modèle relationnel – Représentation

Représentation entité-association avec Merise – MCD



Principes et concepts du modèle relationnel – Modélisation



Principes et concepts du modèle relationnel - Exercice

Conception de tables – énoncé :

Pour assurer la gestion de son service informatique, une société met en œuvre une base de données relationnelle. On a recensé les règles de gestion suivantes :

1) A chaque département (ou société) est rattaché un ensemble d'ingénieurs.

Un ingénieur, à une date donnée, est affilié à un département et un seul.

On met en historique le passage dans les différents départements.

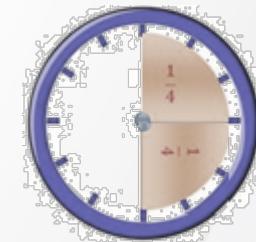
2) Les ingénieurs sont engagés dans les projets, selon une fonction et un prix de vente donné, où ils exercent certaines de leurs compétences.

3) Un ingénieur ne peut pas démarrer deux projets à la même date.

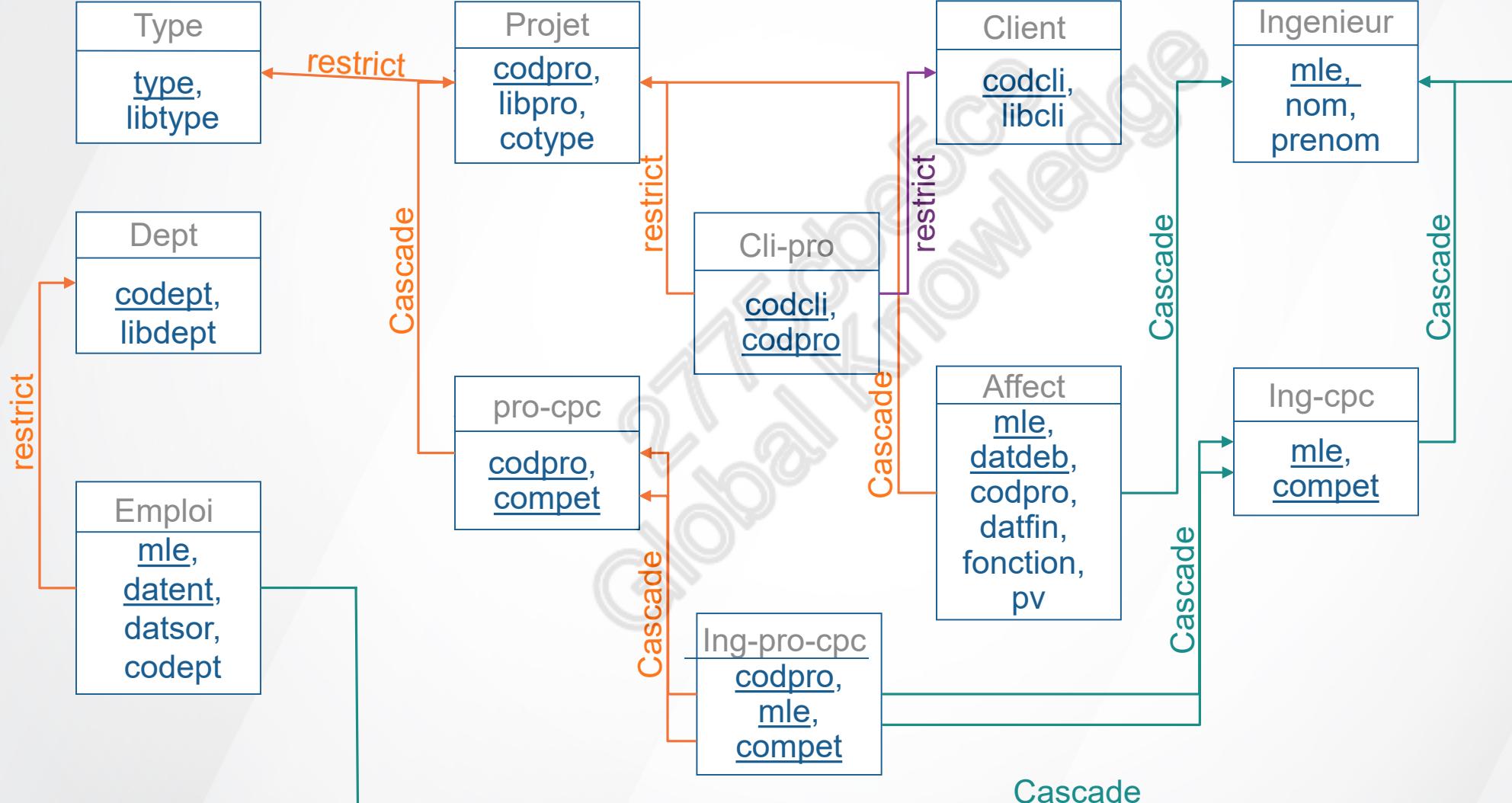
4) Chaque projet associe un, voire plusieurs clients, et inversement.

5) Chaque projet est rattaché à un seul type de projet.

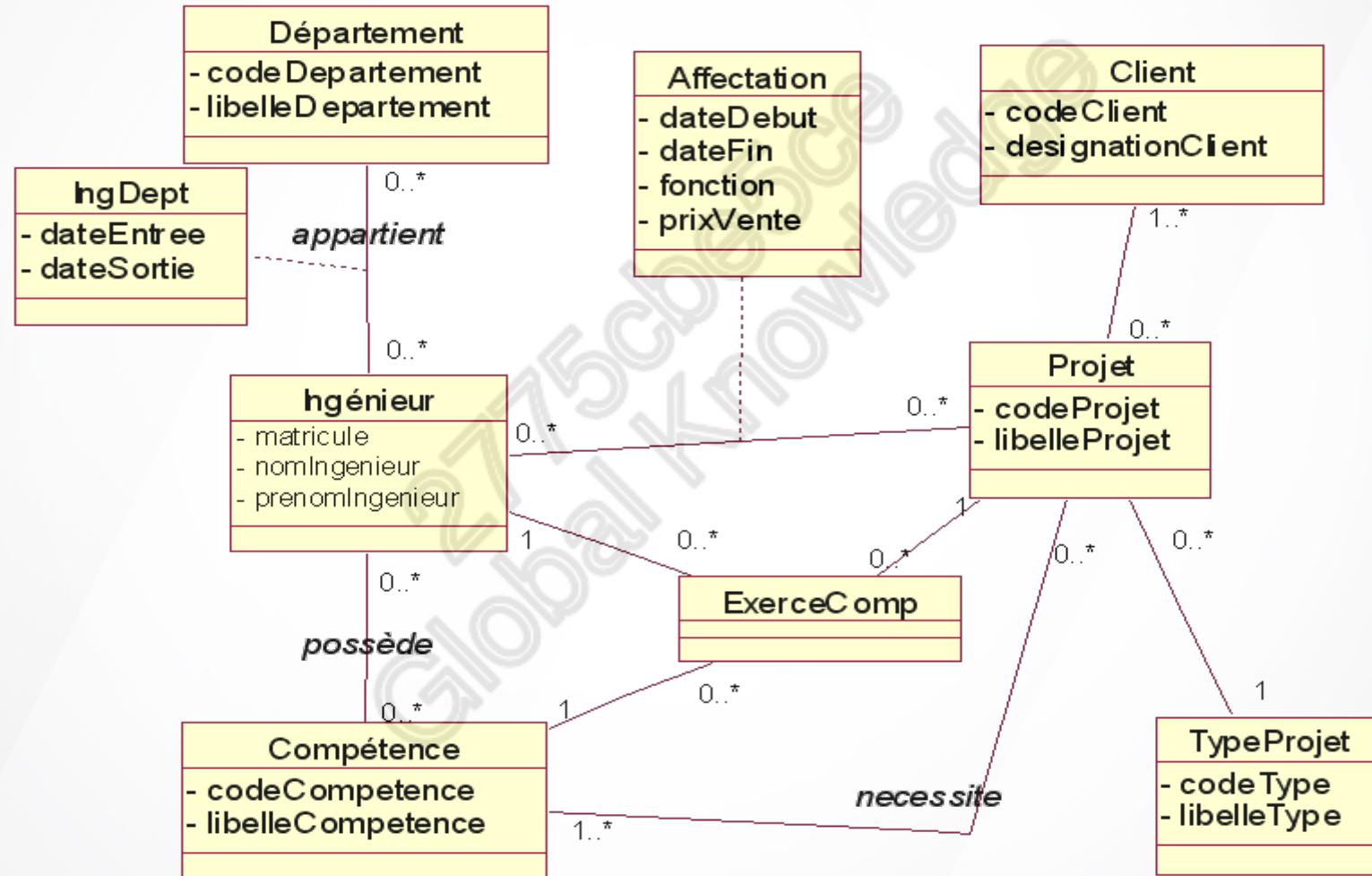
6) Les projets requièrent des compétences.



Principes et concepts du modèle relationnel - Correction



Principes et concepts du modèle relationnel – Correction Diagramme de Classes



Ajout d'une douzième table Compétence



Global Knowledge®

GKSQL Les bases du SQL Partie 2

WORLDWIDE LOCATIONS

BELGIUM CANADA COLOMBIA DENMARK EGYPT FRANCE IRELAND JAPAN KOREA MALAYSIA MEXICO NETHERLANDS NORWAY QATAR
SAUDI ARABIA SINGAPORE SPAIN SWEDEN UNITED ARAB EMIRATES UNITED KINGDOM UNITED STATES OF AMERICA

III - Les requêtes avec le langage SQL

1. Présentation de SQL



Les requêtes avec le langage SQL – Présentation de SQL

SQL : est un langage non procédural de manipulation des données (DML).

est adapté à la consultation des bases relationnelles.

ne respecte pas totalement l'algèbre relationnelle.

fait l'objet d'une normalisation depuis 1986.

La norme SQL-2 définie en 1992 a été la plus couramment employée.

De nouvelles normes lui ont succédé, pas forcément implémentées par tous les éditeurs, ni implantées de la même manière.

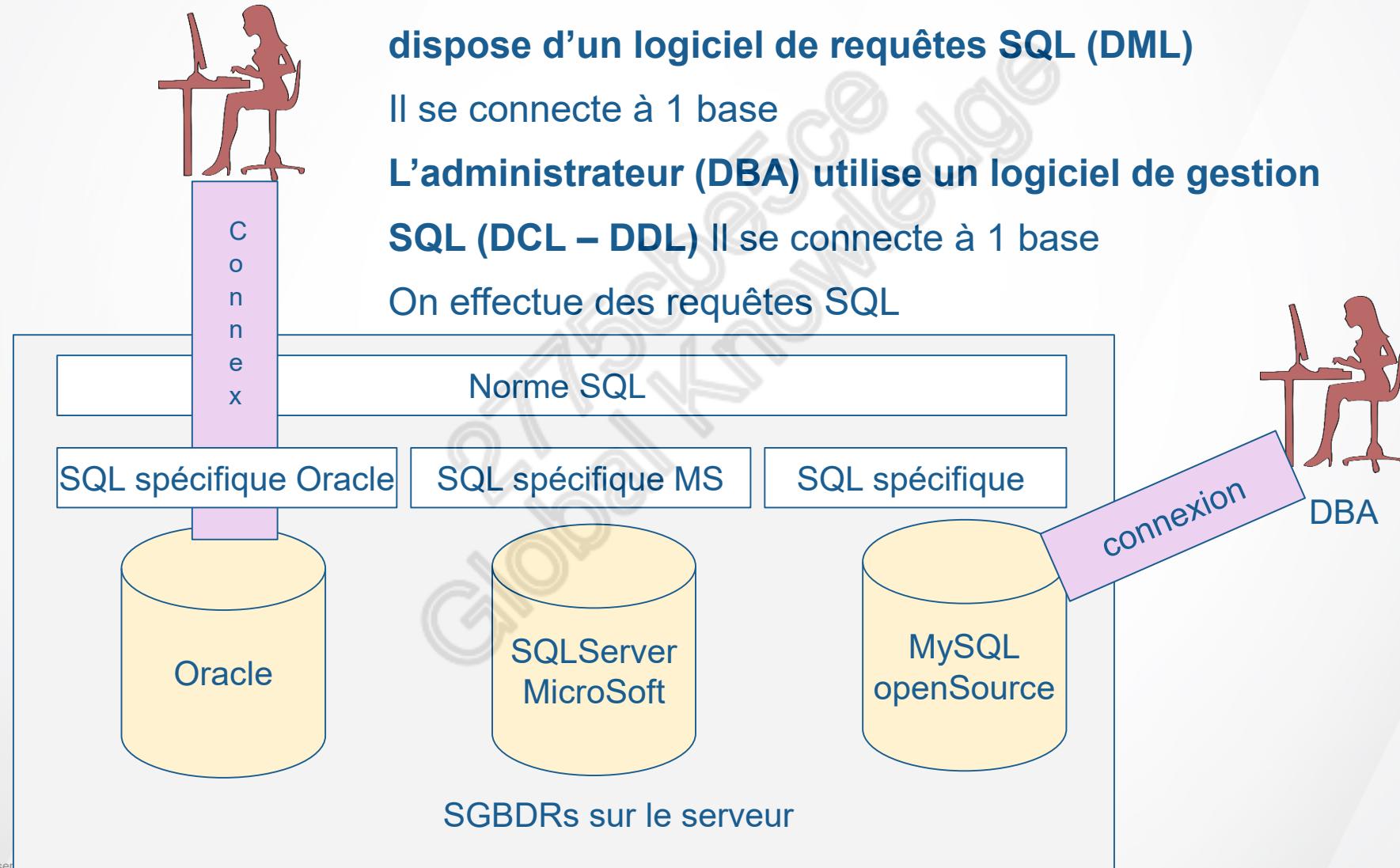
La norme ISO SQL: 2006 (prévue en 2003) permet de gérer les fichiers XML.

La norme ISO SQL: 2011 est la 7^{ème} révision, permet l'historisation auto.

Seules seront présentées ci-après les syntaxes de consultation, qui répondent le mieux à la norme, adaptées à tous les SGBDR.

Les requêtes avec le langage SQL – Présentation de SQL

L'utilisateur



III - Les requêtes avec le langage SQL

2. Requêtes simples



Les requêtes avec le langage SQL – Structure générale du Select

Table des ingénieurs
définie sur le SGBDR

| basesql.ingenieur | |
|-------------------|--------------|
| MLE | smallint(5) |
| NOM | varchar(20) |
| PRENOM | varchar(20) |
| NOSS | varchar(50) |
| DATNAIS | datetime |
| NOTEL | varchar(14) |
| ADRESSE | varchar(30) |
| CODEPOST | varchar(5) |
| VILLE | varchar(20) |
| SEXE | varchar(1) |
| SITUATION | varchar(1) |
| CPT | double(15,5) |

L'ordre Select va fournir 1 résultat sous forme de
ligne/colonne

```
SELECT col1, col2, ..., coln  
FROM nom-table
```

On indique donc quelles colonnes on souhaite et sur quelle
table on travaille (from)

Exemple

```
SELECT mle, prenom, nom  
FROM ingenieur
```

| MLE | PRENOM | NOM |
|-----|------------|-----------|
| 1 | 53 NICOLAS | BLASQUEZ |
| 2 | 54 ANTOINE | HERNANDEZ |
| 3 | 55 JACQUES | MARCIER |
| 4 | 56 BRUNO | GALLET |
| 5 | 100 SYLVIE | DUCHE |

Résult

Les requêtes avec le langage SQL – Requêtes simples - Select (projection)

Sélection de certaines colonnes :

```
SELECT      col1, col2  
FROM        nom-table
```

Exemple :

Table Ingenieur

| | | | | | | | | | | | |
|-----|-----|--------|------|---------|-------|---------|----------|-------|------|-----------|-----|
| MLE | NOM | PRENOM | NOSS | DATNAIS | NOTEL | ADRESSE | CODEPOST | VILLE | SEXE | SITUATION | CPT |
|-----|-----|--------|------|---------|-------|---------|----------|-------|------|-----------|-----|

Liste des ingénieurs (Matricule, Nom, Prénom, ville)

```
SELECT      MLE, NOM, PRENOM, VILLE  
FROM        INGENIEUR
```

Sélection de toutes les colonnes – A éviter en programmation

```
SELECT      *  
FROM        nom-table
```

Les requêtes avec le langage SQL – Select - Démonstration

Tous ensemble, si distance, on accède aux machines de Global Knowledge :

1- Navigateur autre que IE pour accéder au SERVEUR GK :

- <https://msg.gk-labs.fr>
- User : s99xxxx
- et mdp : s99xxxx

2- Serveur GK, puis Machine virtuelle Le formateur vous attribue 1 numéro

- mdp : P@ssw0rd (zéro et non O)
- Chaque machine virtuelle a son SGBDR et sa base nommée basesql

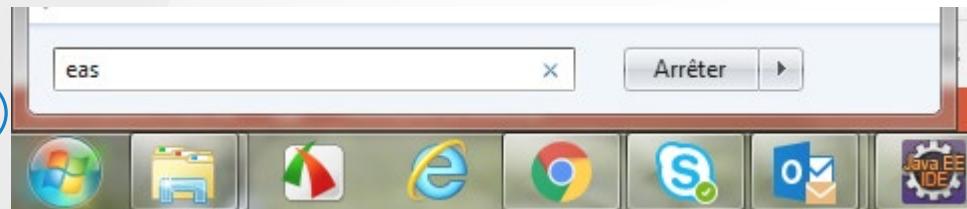


Les requêtes avec le langage SQL – Select - Démonstration

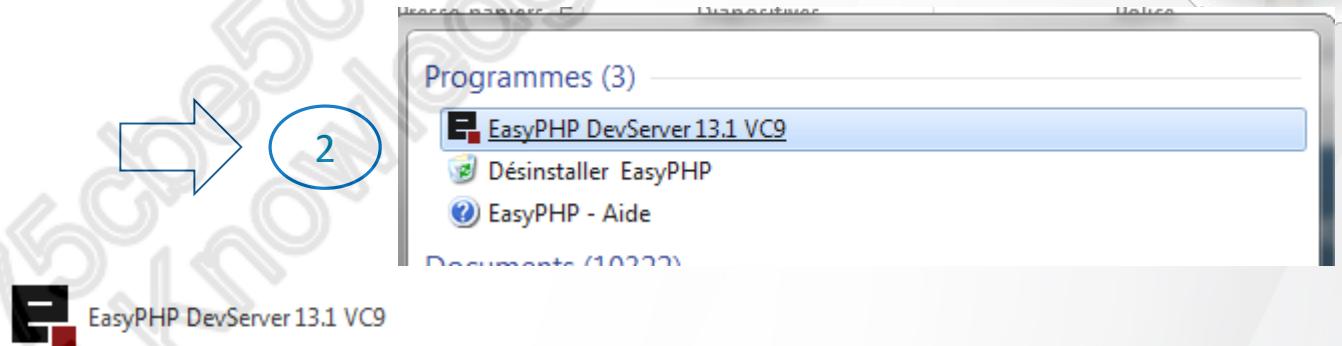


Nous allons utiliser le logiciel **EasyPHP**, nous faisons ensemble une première prise en main.

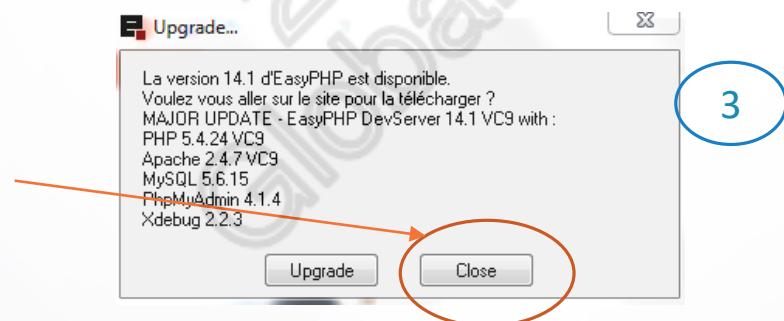
1. Menu Windows « Démarrer ». Commencer à taper « eas »



2. Dans la liste des programmes, sélectionner pour exécuter le logiciel.

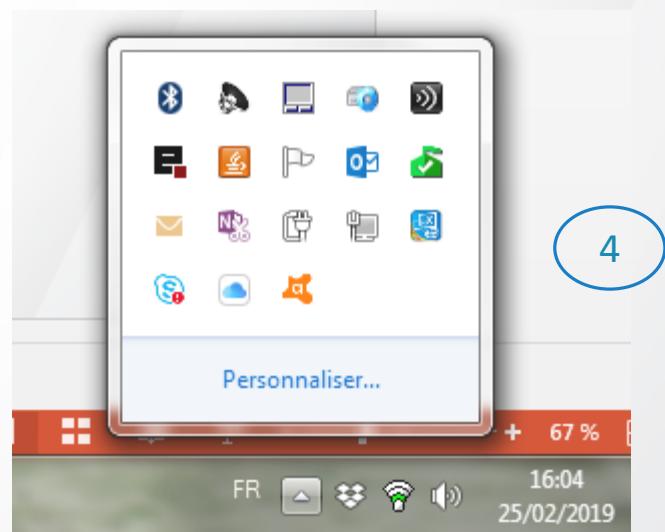


3. Ne pas accepter l'upgrade



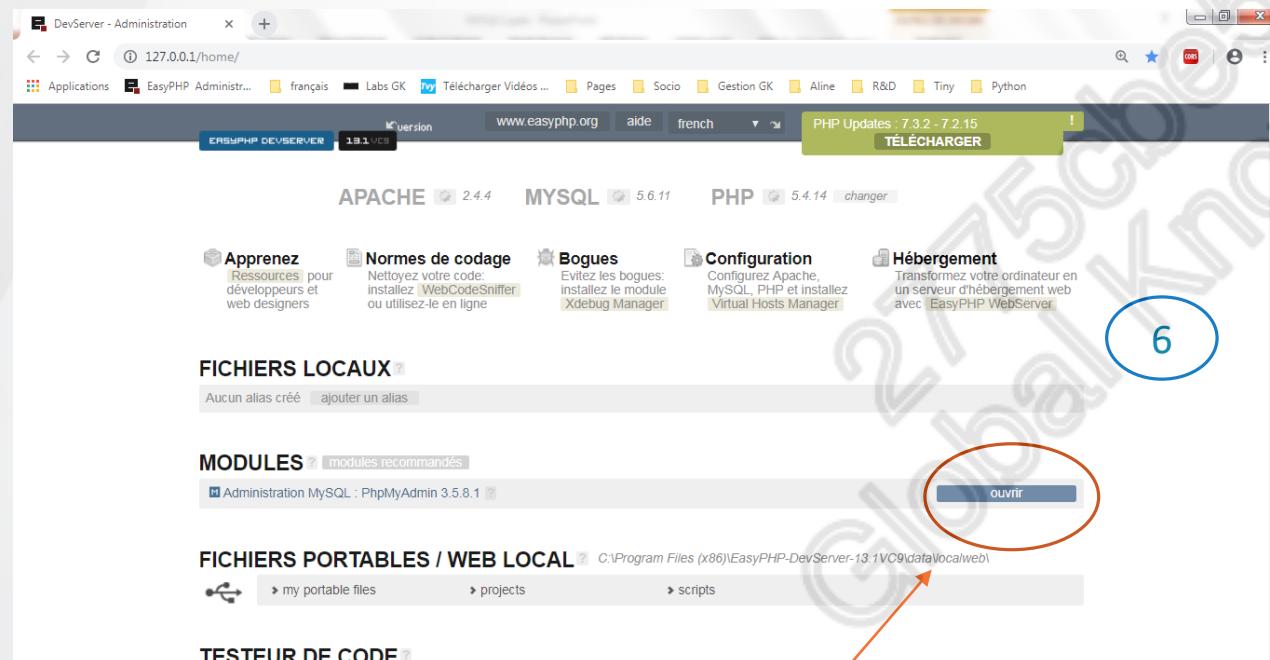
Soit dans la barre de tâches soit en icônes cachées :

4. Vous devez avoir l'icône d'EasyPHP

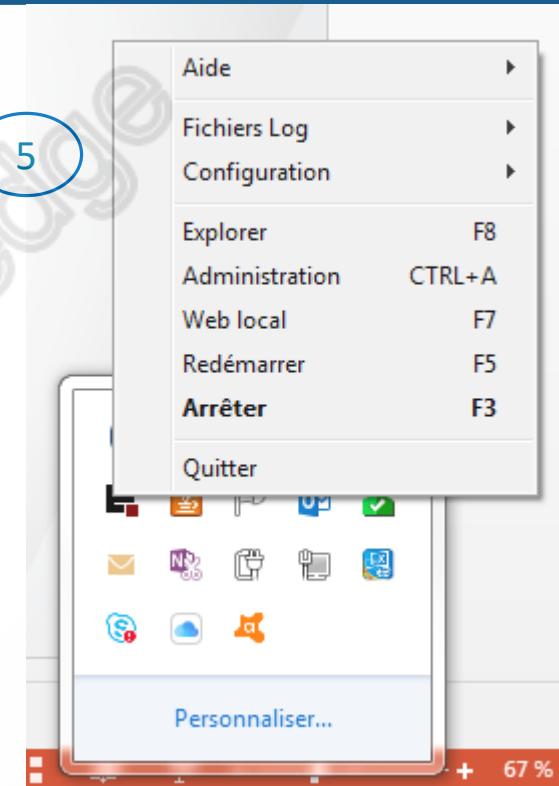


Les requêtes avec le langage SQL – Select - Démonstration

5. Sélectionner l'icône d'EasyPHP puis click droit,
Choisir **Administration**, cela va ouvrir un navigateur :



6. Dans Modules, choisir « ouvrir »



Les requêtes avec le langage SQL – Select - Démonstration



127.0.0.1 / 127.0.0.1 | phpMyAdmin 3581x150615172349

Applications EasyPHP Administr... français Labs GK Télécharger Vidéos ... Pages Socio Gestion GK Aline R&D Tiny Python

phpMyAdmin

127.0.0.1

Bases de données SQL État Utilisateurs Exporter Importer Paramètres plus

Paramètres généraux

Interclassement pour la connexion au serveur : utf8_general_ci

Paramètres d'affichage

Langue - Language : Français - French Thème : pmahomme Taille du texte : 82% Plus de paramètres

Serveur de base de données

- Serveur: 127.0.0.1 via TCP/IP
- Logiciel: MySQL
- Version du logiciel: 5.6.11-log - MySQL Community Server (GPL)
- Version du protocole: 10
- Utilisateur: root@localhost
- Jeu de caractères du serveur: UTF-8 Unicode (utf8)

7

8

127.0.0.1 / 127.0.0.1 | basesql | pma 3581x150615172349

Applications EasyPHP Administr... français Labs GK Télécharger Vidéos ... Pages Socio Gestion GK Aline R&D Tiny Python

phpMyAdmin

127.0.0.1 / basesql

Structure SQL Rechercher Requête Exporter Importer Opérations plus

Table Action

| | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | Lignes | Type | Interclassement | Taille | Per |
|-------------|----------|-----------|------------|---------|-------|-----------|--------|--------|-------------------|--------|-----|
| affect | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 56 | InnoDB | latin1_swedish_ci | 32 Kio | |
| client | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 9 | InnoDB | latin1_swedish_ci | 16 Kio | |
| cli_pro | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 11 | InnoDB | latin1_swedish_ci | 32 Kio | |
| dept | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 17 | InnoDB | latin1_swedish_ci | 16 Kio | |
| emploi | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 54 | InnoDB | latin1_swedish_ci | 32 Kio | |
| filles | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 21 | Vue | --- | - | |
| ingenieur | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 44 | InnoDB | latin1_swedish_ci | 16 Kio | |
| ing_cpc | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 158 | InnoDB | latin1_swedish_ci | 16 Kio | |
| ing_pro_cpc | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 64 | InnoDB | latin1_swedish_ci | 32 Kio | |
| parisiens | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 15 | Vue | --- | - | |
| projet | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 16 | InnoDB | latin1_swedish_ci | 32 Kio | |
| pro_cpc | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 60 | InnoDB | latin1_swedish_ci | 16 Kio | |
| type | Afficher | Structure | Rechercher | Insérer | Vider | Supprimer | 6 | InnoDB | latin1_swedish_ci | 16 Kio | |

13 tables Somme

7. Sélectionner la base basesql

A gauche, la liste des tables,
à droite des onglets pour travailler avec cette base.

8. Onglet « SQL » pour taper les requêtes SQL.

Les requêtes avec le langage SQL – Select - Démonstration



9

Exécuter une ou des requêtes SQL sur la base basesql:

```
1 select *
2 From ingénieur
3 |
```

Vider

Conserver cette requête SQL dans les signets:

[Délimiteur :] Afficher à nouveau la requête après exécution Conserver la boîte de requêtes

Exécuter

9

Montrer zone SQL

Affichage des lignes 0 - 29 (44 total, Traitement en 0.000 sec)

SELECT *
FROM ingénieur
LIMIT 0 , 30

Tout afficher > >> Afficher : Ligne de départ: 30 Nombre de lignes: 30 En-têtes à chaque 100 lignes

Profilage [En ligne] [Modifier] [Expliquer SQL] [Créer source PHP] [Actualiser]

| | MLE | NOM | PRENOM | NOSS | DATNAIS | NOTEL | ADRESSE | CODE |
|-----------------------------------|-----|----------|-----------|--------------|---------------------|----------------|---------------------|-------|
| <input type="checkbox"/> Modifier | 25 | LORENT | CATHERINE | 260050000000 | 1960-05-15 00:00:00 | 01 30 64 19 31 | 8 rue de Samar | 78180 |
| <input type="checkbox"/> Modifier | 53 | BLASQUEZ | NICOLAS | 150060000000 | 1950-06-30 00:00:00 | 01 19 04 56 78 | Allée des Verdurins | 84300 |

9. Taper votre requête, puis exécuter.

Le résultat s'affiche en bas de l'écran, vous pouvez utiliser « modifier » pour reprendre votre requête.

Vous pouvez aussi ouvrir plusieurs onglets sur le navigateur.

Les requêtes avec le langage SQL – Select - Elimination des doubles

Les lignes sont présentées dans l'ordre dans lequel elles sont stockées dans la table.

SQL n'élimine pas les doubles - Elimination des doublons avec DISTINCT :

```
SELECT      DISTINCT col1, col2  
  FROM nom-table  
;
```

Remarque :
; non obligatoire

Exemple : Liste des différents prénoms d'ingénieur

```
SELECT DISTINCT PRENOM  
  FROM INGENIEUR
```

Une seule clause DISTINCT par requête.

```
SELECT DISTINCT VILLE, PRENOM  
  FROM INGENIEUR
```



| VILLE | PRENOM |
|---------|-----------------|
| LE THOR | ANTOINE |
| RENNES | FLORENCE |
| RENNES | MYRIAM |
| RENNES | NADIA |
| PARIS | MARIE DOMINIQUE |

Les requêtes avec le langage SQL – Select Qualifié (Restriction - Filtre)

**SELECT
FROM
WHERE
;**

Exemples :

**SELECT
FROM
WHERE**

MLE, NOM, PRENOM
INGENIEUR
CODPOST = 40300

**col1, col2
nom-table
qualification**

→ on restreint les lignes du résultat

Ingénieurs habitant à Peyrehorade

Numérique

**SELECT
FROM
WHERE**

MLE, NOM, PRENOM
INGENIEUR
SEXЕ = 'F'

Ingénieres

Alphanumérique

**SELECT
FROM
WHERE**

MLE, NOM, PRENOM, NOSS, SEXE
INGENIEUR
NOTEL IS NULL AND NOSS > 200000000000

Plusieurs filtres



Les requêtes avec le langage SQL – Select Qualifié (Restriction - Filtre)

Ingénieurs qui sont nés après le premier janvier 1980

Regarder avec votre logiciel comment est présentée la date avant de la saisir !

```
SELECT NOM, PRENOM, SEXE, DAT  
      FROM INGENIEUR  
 WHERE DATNAIS >= '1980-01-01'
```

Mysql accepte
DATNAIS > '1980'

Ingénieurs habitant Les Landes

```
SELECT MLE, NOM, PRENOM  
      FROM INGENIEUR  
 WHERE CODPOST BETWEEN 40000 and 40999
```

Ingénieurs dont le nom contient LE

```
SELECT MLE, NOM, PRENOM  
      FROM INGENIEUR  
 WHERE NOM LIKE '%LE%'
```

Ingénieurs ayant le téléphone

```
SELECT MLE, NOM, PRENOM  
      FROM INGENIEUR  
 WHERE NOTEL IS NOT NULL
```

Les requêtes avec le langage SQL – Select Qualifié (Restriction - Filtre)

Départements de l'entreprise (libellé et ville d'implantation)

SELECT DESIGN, VILLE FROM DEPT

WHERE PDG = 'SOKAL' OR PDG = 'AUGUIN'
AND DESIGN LIKE 'F%' AND NOTEL IS NULL
AND CODPOST BETWEEN 50000 AND 50999

OR est prioritaire !

Pour avoir les mêmes résultats il faut des parenthèses.

;
SELECT DESIGN, VILLE FROM DEPT

WHERE PDG IN ('SOKAL', 'AUGUIN')
AND DESIGN LIKE 'F%' AND NOTEL IS NULL
AND CODPOST BETWEEN 50000 AND 50999

;
Est-ce que ce sont les mêmes requêtes ?

Les requêtes avec le langage SQL – Select - Classement

Possibilité de ranger (trier) le résultat selon plusieurs critères,
croissant, décroissant

```
SELECT      col1, col2  
FROM        nom-table  
WHERE       Qualification  
ORDER BY    col1 ASC, col2 DESC
```

Par défaut ASC,
Non obligatoire

Exemple : Liste des prénoms des salariés féminins dans l'ordre de leurs dates de naissance.

```
SELECT      PRENOM, DATNAIS  
FROM        INGENIEUR  
WHERE       SEXE = 'F'  
ORDER BY    DATNAIS
```

;

Equivalent à ORDER BY 2

ORDER BY DATNAIS DESC, PRENOM

| | PRENOM | DATNAIS |
|---|-----------------|------------|
| 1 | MARIE DOMINIQUE | 1956-04-10 |
| 2 | RITA | 1957-05-10 |
| 3 | LINDA | 1958-01-27 |
| 4 | SYLVIE | 1959-09-12 |
| 5 | GLADYS | 1960-09-09 |
| 6 | MARIA | 1961-10-17 |
| 7 | PATRICIA | 1962-04-10 |
| 8 | MYRIAM | 1964-08-15 |
| 9 | ARMINE | 1965-05-14 |

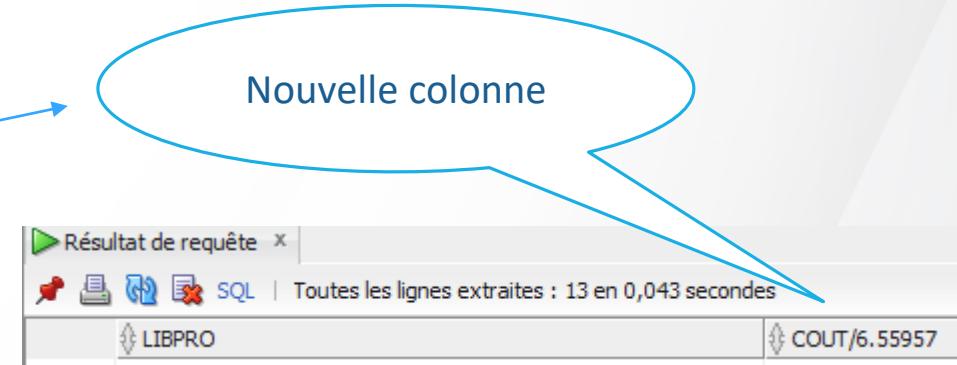
Les requêtes avec le langage SQL – Select - Opérateurs arithmétiques

Les opérateurs arithmétiques sont utilisables pour la projection (SELECT) ou pour la restriction (WHERE)

Symboles : + - * /

Exemple : Liste des libellés de projet avec leur coût en EURO si celui-ci est supérieur à 100.000, les coûts étant enregistrés en francs dans la base.

```
SELECT LIBPRO, COUT/6.55957  
FROM PROJET  
WHERE COUT > 655957
```



Remarques : "null" est un élément absorbant.

Les opérations effectuées sur une valeur "null" sont "null".

Les requêtes avec le langage SQL – Select – Alias et Constante

```
SELECT LIBPRO, COUT/6.55957 AS COUT_EURO  
FROM PROJET  
WHERE COUT > 655957
```

L'alias est visible en résultat :

The screenshot shows a SQL query results window. The query is:

```
Réseau de requête x  
SQL | Toutes les lignes extraites : 13 en 0 secondes
```

| LIBPRO | COUT_EUROS |
|------------------------------|---|
| 1 GOLDENSCORE REASAU | 182938,8206848924548407898688481104706559 |
| 2 ELF SERVICE JURIDIQUE | 1250081,94134676510807873077046208821615 |
| 3 03E | 182938,8206848924548407898688481104706559 |
| 4 CARREFOUR DU DEVELOPPEMENT | 114336,7629280577842754936680300690441599 |
| 5 LA MONDIALE | 320905,1812847488478665522282710604506088 |
| 6 BISCUITERIE BELGE | 1522965,68220172968654957565816051966821 |

```
SELECT 'Monsieur', nom, prenom  
FROM INGENIEUR  
WHERE sexe = 'M'
```

The screenshot shows a SQL query results window. The query is:

```
Réseau de requête x  
SQL | Toutes les lignes extraites : 24 en 0 secondes
```

| 'MONSIEUR' | NOM | PRENOM |
|-----------------------|---------|--------|
| 1 Monsieur BLASQUEZ | NICOLAS | |
| 2 Monsieur HERNANDEZ | ANTOINE | |
| 3 Monsieur MARCIER | JACQUES | |
| 4 Monsieur GALLET | BRUNO | |
| 5 Monsieur SOUPINACIO | JESUS | |

Les requêtes avec le langage SQL - Récapitulatif

```
SELECT    col1, col2, col3 + col4 / 100 AS calcul, 'constante'  
FROM      table  
WHERE     (col2 is null AND col3 * 100 > 200 ) OR  
          (col3 = 0 and col1 like 'A% ' )  
ORDER BY  col2 DESC, col1  
;
```

Les requêtes avec le langage SQL – Exercices Requêtes Simples

Support « Exercices et résultats attendus » :

Nous vous avons mis le **schéma** de la base Orasql en point 1

Nous effectuons les **exercices du point 2 : Requêtes Simples**

Vous avez la question, nous vous fournissons le résultat attendu, afin que vous puissiez vérifier l'exactitude de votre requête et avancer de manière autonome

Ne restez pas bloqué → **Sollicitez le formateur**

Faire la série 1 à son rythme,

Prévenez le formateur quand cette série est terminée

Faire la série 2 - Prévenez dès la fin des exercices « Accès une table »

Vous aurez besoin de la date système, cette information n'a pas été normalisée par SQL

Date système:

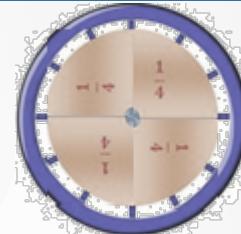
sous DB2 : **CURRENT DATE**

sous SQLServer : **getdate()**

sous Oracle: **SYSDATE et CURRENT_DATE**

sous MySQL: **CURRENT_DATE**

sous PostgreSQL : **now**

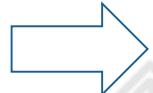


Les requêtes avec le langage SQL – Correction (série 1)

10) Donner la liste des matricules de salariés, compris entre 501 et 1000, intervenant **actuellement** au moins sur l'un des projets suivants :

17, 22 ou 120

```
select distinct mle, codpro  
from affect  
where codpro in (17, 22, 120)  
and mle between 501 and 1000  
and datfin is null  
;
```



On obtient 7 lignes.
Cependant, étant donné qu'il peut y avoir des dates dans le futur,
il fallait écrire :



```
select distinct mle, codpro  
from affect  
where codpro in (17, 22, 120)  
and mle between 501 and 1000  
and (datfin is null or datfin >= sysdate)  
and datdeb <= sysdate  
;
```

Les requêtes avec le langage SQL – Correction (série 2)

1. Liste des ingénieurs de la société. (Mle, Nom, Prénom).

```
SELECT mle , nom, prenom
```

```
FROM ingenieur ;
```

2. Liste des couples (PDG,DG) de tous les départements avec élimination des doubles.

```
SELECT distinct pdg, dg
```

```
FROM dept ;
```

Les requêtes avec le langage SQL – Correction

3. Liste des MLE des Ingé actuellement dans le département 7, triée sur les MLE croissants.

```
SELECT mle  
      FROM emploi  
     where codept = 7  
       and (datsor is null or datsor > current_date)  
       and datent < current_date  
   order by mle  
;
```

Les requêtes avec le langage SQL - Correction

4. Liste des prénoms rangée par sexe et prénom.

```
SELECT distinct sexe, prenom  
FROM ingenieur  
ORDER BY sexe, prenom  
;
```

← A définir sous MySQL, les autres SGBD le font automatiquement avec Distinct

5. Liste des départements (code département, désignation, PDG) situés rue de Londres à Paris (chercher 'LONDRES' et 'PARIS' en majuscules).

```
SELECT codept, design, pdg  
FROM dept  
where adresse like '%LONDRES%' and ville like 'PARIS%'  
;
```

Les requêtes avec le langage SQL - Correction

6. Liste des ingénieurs n' ayant pas le téléphone.

```
SELECT mle, nom, notel  
FROM ingenieur  
WHERE notel is null  
;
```

III - Les requêtes avec le langage SQL

3. Jointures



Les jointures – Jointure 2 tables – Critère simple

SALARIÉ et AFFECTATION sont jointes par leurs colonnes MLE qui doivent faire référence au même domaine pour que la jointure ait un sens

| SALARIE | MLE | NOM | PRÉNOM | FONCTION | GRADE |
|---------|--------|-------|--------|----------|-------|
| 915 | KANTOR | Georg | | DP | G7 |
| 628 | CODD | Ted | | CDP | G5 |
| 732 | DATE | Chris | | AP | G5 |

colonnes MLE

| AFFECTATION | MLE | CODPROJ | DAT_DEB | DAT_FIN |
|-------------|------|---------|------------|------------|
| 915 | P212 | | 02-03-2002 | 12-11-2003 |
| 628 | P911 | | 01-05-2002 | 30-06-2003 |
| 732 | P911 | | 10-09-2001 | 03-02-2002 |
| 915 | P911 | | 01-12-2003 | "null" |
| 732 | P212 | | 01-03-2003 | "null" |

Lever les ambiguïtés

SELECT SALARIE.MLE, NOM, PRENOM, CODPROJ, DAT_DEB

From/Join
noms de Tables

FROM SALARIE
JOIN AFFECTATION

ON noms de colonnes

ON SALARIE.MLE = AFFECTATION.MLE

Résultat

| MLE | NOM | PRÉNOM | CODPROJ | DAT_DEB |
|-----|--------|--------|---------|------------|
| 915 | KANTOR | Georg | P212 | 02-03-2002 |
| 628 | CODD | Ted | P911 | 01-05-2002 |
| 732 | DATE | Chris | P911 | 10-09-2001 |
| 915 | KANTOR | Georg | P911 | 01-12-2003 |
| 732 | DATE | Chris | P212 | 01-03-2003 |

Les jointures – Jointure 2 tables - Critère simple

SALARIE et AFFECTATION sont jointes par leurs colonnes MLE

Lever les ambiguïtés les colonnes qui portent le même nom dans les 2 tables doivent être préfixées par le nom de la table à laquelle elles appartiennent.

Ancienne écriture de jointure ← Pour la maintenance

```
SELECT      SALARIE.MLE, SALARIE.NOM, PRENOM,  
            AFFECTATION.CODPROJ, DAT_DEB  
        FROM      SALARIE, AFFECTATION  
        WHERE     SALARIE.MLE = AFFECTATION.MLE
```

Erreur possible
Si on mélange
jointure et filtre

Remarques L'omission du prédicat de jointure → produit cartésien

Ecriture normalisée

```
SELECT      SALARIE.MLE, SALARIE.NOM, PRENOM,  
            AFFECTATION.CODPROJ, DAT_DEB  
        FROM      SALARIE  
        JOIN     AFFECTATION  
        ON       SALARIE.MLE = AFFECTATION.MLE
```

From : 1 seul nom
Join Autre nom
+
ON noms de colonnes

Les jointures – Jointure 2 tables - Critère simple

Résultat = Equijointure - opérateur égal dans le prédicat de jointure

= jointure naturelle - Clé primaire / Clé étrangère

= projection - on ne sélectionne pas toutes les colonnes

On peut ajouter une restriction avec **WHERE** (d'où une possible erreur avec l'ancienne écriture)

```
SELECT      SALARIE.MLE, SALARIE.NOM, SALARIE.PRENOM,  
           AFFECTATION .CODPROJ, AFFECTATION .DAT_DEB  
FROM        SALARIE  
INNER JOIN AFFECTATION  
          ON SALARIE.MLE = AFFECTATION.MLE  
WHERE       SALARIE.MLE >= 100
```



Les jointures – Jointure 2 tables - Multi-critères

SALARIÉ et **REMUNERATION** sont jointes par leurs colonnes **FONCTION** et **GRADE** afin de récupérer le salaire de chacun des employés

Ancienne écriture

```
SELECT      SALARIE.MLE, SALARIE.NOM, SALARIE.PRENOM,  
            REMUNERATION.SALAIRE  
        FROM    SALARIE, REMUNERATION  
       WHERE SALARIE.GRADE = REMUNERATION.GRADE  
             AND SALARIE.FONCTION = REMUNERATION.FONCTION
```

Remarques : Pour que la jointure entre ces 2 tables ait un sens, il faut travailler sur grade ET fonction

Ecrire cette jointure en utilisant la syntaxe normalisée

Les jointures – Jointures 2 tables - Multi-critères

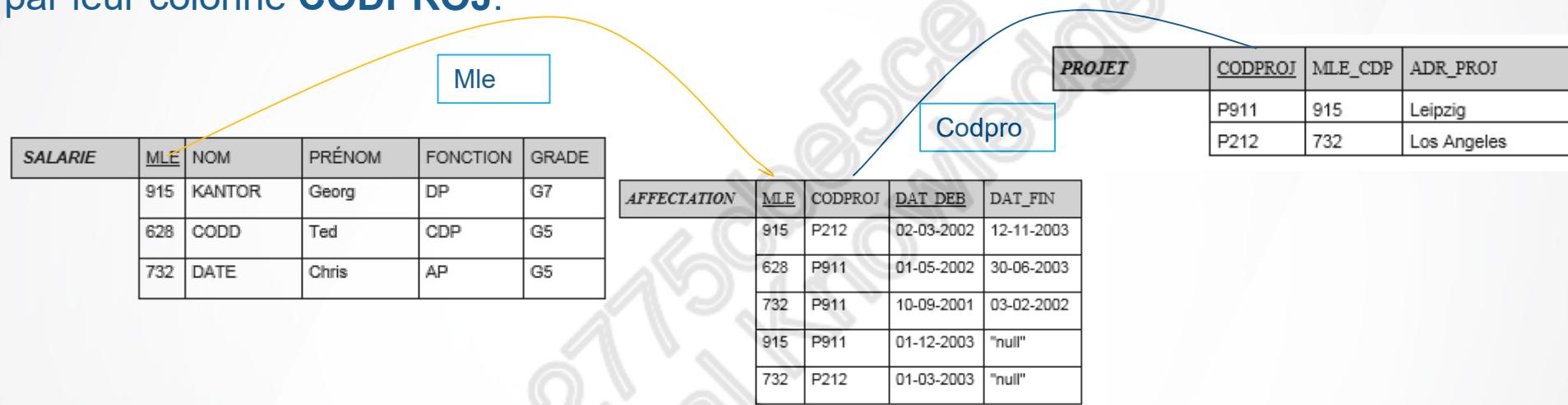
Ecriture normalisée à privilégier

```
SELECT      SALARIE.MLE, SALARIE.NOM, SALARIE.PRENOM,  
           REMUNERATION.SALAIRE  
  FROM        SALARIE  
 INNER JOIN  REMUNERATION  
    ON        SALARIE.GRADE = REMUNERATION.GRADE  
    AND       SALARIE.FONCTION = REMUNERATION.FONCTION
```

Si une table parente a une clé primaire constituée de plusieurs colonnes, une equijointure sur une de ses tables dépendantes doit contenir dans sa clause ON toutes les colonnes qui réfèrentent la clé étrangère.

Les jointures – Jointure de plus de 2 tables

Les tables **SALARIÉ** et **AFFECTATION** sont jointes par leur colonne **MLE**, les tables **AFFECTATION** et **PROJET** par leur colonne **CODPROJ**.



```
SELECT SALARIE.MLE, SALARIE.NOM, SALARIE.PRENOM,  
       PROJET.CODPROJ, PROJET.ADR_PROJ,  
       AFFECTATION.DAT_DEB  
FROM SALARIE  
JOIN AFFECTATION  
      ON SALARIE.MLE = AFFECTATION.MLE  
JOIN PROJET  
      ON AFFECTATION.CODPROJ = PROJET.CODPROJ
```

Les jointures - Nom de corrélation

```
SELECT S.MLE, S.NOM, S.PRENOM,  
       P.CODPROJ, P.ADR_PROJ, A.DAT_DEB  
  
FROM SALARIE S  
      INNER JOIN AFFECTION A  
      ON S.MLE = A.MLE  
      INNER JOIN PROJET P  
      ON A.CODPROJ = P.CODPROJ
```

Résultat

| MLE | NOM | PRÉNOM | CODPROJ | ADR_PROJ | DAT_DEB |
|-----|--------|--------|---------|-------------|------------|
| 915 | KANTOR | Georg | P212 | Los Angeles | 02-03-2002 |
| 628 | CODD | Ted | P911 | Leipzig | 01-05-2002 |
| 732 | DATE | Chris | P911 | Leipzig | 10-09-2001 |
| 915 | KANTOR | Georg | P911 | Leipzig | 01-12-2003 |
| 732 | DATE | Chris | P212 | Los Angeles | 01-03-2003 |

Les jointures - Compléments – Auto-jointure

Jointure d'une table sur elle-même - Les noms de corrélations sont obligatoires

Exemple : Liste des matricules d'ingénieurs affectés le même jour sur un projet.

A1

| MLE | CODPROJ | DAT_DEB | DAT_FIN |
|-----|---------|------------|------------|
| 915 | P212 | 01-05-2002 | 12-11-2003 |
| 628 | P911 | 01-05-2002 | 30-06-2003 |
| 732 | P911 | 01-05-2002 | 03-12-2002 |
| 915 | P911 | 01-12-2003 | "null" |
| 732 | P212 | 01-03-2003 | "null" |

A2

| MLE | CODPROJ | DAT_DEB | DAT_FIN |
|-----|---------|------------|------------|
| 915 | P212 | 01-05-2002 | 12-11-2003 |
| 628 | P911 | 01-05-2002 | 30-06-2003 |
| 732 | P911 | 01-05-2002 | 03-12-2002 |
| 915 | P911 | 01-12-2003 | "null" |
| 732 | P212 | 01-03-2003 | "null" |



| Mle (A1) | Mle (A2) |
|----------|----------|
| 915 | 915 |
| 915 | 628 |
| 915 | 732 |
| 628 | 915 |
| 628 | 628 |
| 628 | 732 |
| 732 | 915 |
| 732 | 628 |
| 732 | 732 |

Evite
les doublons
(matx,matx) et les
tautologies
(matx, maty) et
(maty, matx)

```
SELECT A1.MLE, A2.MLE, A1.DAT_DEB
FROM AFFECTION A1 JOIN AFFECTION A2
ON A1.DAT_DEB = A2.DAT_DEB
AND A1.MLE > A2.MLE
```

Les jointures - Compléments – Theta-jointure

L'opérateur de comparaison sur la colonne de jointure est quelconque

Exemple : Liste des projets ayant un coût supérieur à celui du projet 2

(auto-thétajointure !)

```
SELECT      A.CODPRO, A.LIBPRO, A.COUT  
FROM        PROJET A  
INNER JOIN  PROJET B  
              ON A.COUT > B.COUT  
WHERE       B.CODPRO = 2
```

Les jointures - Compléments – Semi-jointure

Soient R1 et R2 deux tables de colonne de jointure COL1

La semi-jointure est l'ensemble des lignes de R1 tel que la valeur de COL1 existe dans R2.

Exemple : Liste des emplois dont le code département est valide.

```
SELECT      EMPLOI.*
FROM        EMPLOI
INNER JOIN DEPT
ON          EMPLOI.CODEPT = DEPT.CODEPT
```

Les jointures - Exercices

Support « Exercices et résultats attendus » :

Nous effectuons les **exercices du point 3 : Jointures**

On débute par la Série 1,

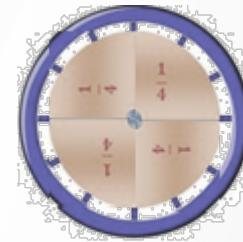
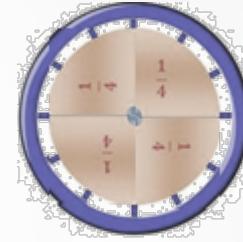
on prévient quand on a fini, avant de faire la série 2

Date système:

sous DB2 : **CURRENT DATE**

sous Oracle: **SYSDATE et CURRENT_DATE**

sous MySQL: **CURRENT_DATE**



Les jointures - Corrigés

1. Liste des ingénieurs (NOM, PRENOM, DATENT) appartenant au départ. 7.

```
SELECT NOM, PRENOM, DATENT  
      FROM INGENIEUR I  
INNER JOIN EMPLOI E  
      ON I.MLE = E.MLE  
 WHERE E.CODEPT = 7  
AND (E.DATSOR IS NULL OR E.DATSOR > CURRENT_DATE)  
AND E.DATENT < CURRENT_DATE
```

2. Idem mais la liste doit être triée par date d'entrée.

```
          ORDER BY DATENT
```

Les jointures - Corrigés

3. Liste des affectations en cours (NOM, LIBPRO, PV) du département 7.

```
SELECT NOM, LIBPRO, PVENTE  
FROM INGENIEUR I  
INNER JOIN EMPLOI E      ON I.MLE = E.MLE  
INNER JOIN AFFECT A     ON I.MLE = A.MLE  
INNER JOIN PROJET P    ON A.CODPRO = P.CODPRO
```

```
WHERE E.CODEPT = 7  
AND (E.DATSOR IS NULL OR E.DATSOR > CURRENT_DATE)  
AND (A.DATFIN IS NULL OR A.DATFIN > CURRENT_DATE)
```

4. Idem mais la liste doit être triée par CODPRO descendant.

```
ORDER BY P.CODPRO DESC
```

```
;
```

Les jointures - Corrigés

1. Liste des ingénieurs (MLE,NOM,PRENOM) présents dans le département 7 au moins 1 jour en 2000

```
SELECT I.MLE, I.NOM, I.PRENOM  
FROM INGENIEUR I inner JOIN EMPLOI E ON I.MLE = E.MLE  
WHERE E.CODEPT = 7  
AND E.DATENT <= '2000-12-31'  
AND (E.DATSOR >= '2000-01-01'  
OR E.DATSOR IS NULL)  
;
```

Les jointures - Corrigés

2. Liste des départements ayant le même PDG, avec élimination des lignes où le PDG n'est pas valorisé

```
SELECT A.DESIGN, B.DESIGN, A.PDG  
FROM DEPT A INNER JOIN DEPT B  
    ON      A.PDG    = B.PDG  
        AND A.CODEPT < B.CODEPT  
WHERE A.PDG IS NOT NULL  
;
```

Les jointures - Corrigés

3. Liste des ingénieurs participant actuellement au projet : 22
(Nom, Prénom, Fonction, Libellé du projet).

```
SELECT I.NOM , I.PRENOM , A.FONCTION , P.LIBPRO
FROM INGENIEUR I INNER JOIN AFFECT A ON I.MLE    = A.MLE
                           INNER JOIN PROJET P  ON A.CODPRO = P.CODPRO
WHERE P.CODPRO = 22
AND (A.DATFIN >= CURRENT_DATE OR A.DATFIN IS NULL)
AND A.DATDEB < CURRENT_DATE
;
```

Les jointures - Corrigés

4. Liste des ingénieurs participant actuellement à un projet dont l'un au moins des clients habite la même ville que l'ingénieur.

```
SELECT DISTINCT I.NOM , C.CODCLI  
FROM INGENIEUR I INNER JOIN AFFECT A ON I.MLE = A.MLE  
INNER JOIN CLI_PRO CP ON A.CODPRO = CP.CODPRO  
INNER JOIN CLIENT C CP.CODCLI = C.CODCLI  
  
WHERE  
I.VILLE = C.VILLE  
AND (A.DATFIN >= CURRENT DATE OR A.DATFIN IS NULL)  
AND A.DATDEB <= CURRENT DATE  
;
```

Les jointures - Corrigés

5. Quels sont les ingénieurs (MLE) ayant participé à des projets situés à Paris ?

```
SELECT A.MLE  
FROM AFFECT A INNER JOIN PROJET P ON A.CODPRO = P.CODPRO  
WHERE A.DATFIN <= CURRENT_DATE  
AND VILLE = 'PARIS'  
;
```

Les jointures - Jointure Externe

OUTER JOIN = Jointure externe

| SALARIÉ_EXPAT | MLE | NOM | PRÉNOM | SALAIRE | GRADE |
|---------------|-----|--------|--------|---------|-------|
| | 915 | CANTOR | Georg | 17.000 | CDP |
| | 628 | CODD | Ted | 20.000 | DP |
| | 545 | CURIE | Marie | 18.000 | CD |
| | 732 | DATE | Chris | 15.000 | CDP |

RIGHT, LEFT ou les deux FULL

| PROJET | CODPROJ | MLE_CDP | ADR_PROJ |
|--------|---------|---------|----------|
| | P911 | 915 | Leipzig |
| | P417 | 1000 | Dublin |
| | P212 | 732 | Madrid |

SELECT s.mle, nom, prenom, salaire, grade, codproj, p.mle, ville

FROM salaries s LEFT OUTER JOIN projet p on s.mle=p.mle

| MLE | NOM | PRÉNOM | SALAIRE | GRADE | CODPROJ | MLE_CDP | ADR_ROJ |
|-----|--------|--------|---------|-------|---------|---------|---------|
| 915 | CANTOR | Georg | 17.000 | CDP | P911 | 915 | Leipzig |
| 628 | CODD | Ted | 20.000 | DP | "null" | "null" | "null" |
| 545 | CURIE | Marie | 18.000 | CD | "null" | "null" | "null" |
| 732 | DATE | Chris | 15.000 | CDP | P212 | 732 | Madrid |

SELECT s.mle, nom, prenom, salaire, grade, codproj, p.mle, ville

FROM salaries s RIGHT OUTER JOIN projet p on s.mle=p.mle

| MLE | NOM | PRÉNOM | SALAIRE | GRADE | CODPROJ | MLE_CDP | ADR_ROJ |
|--------|--------|--------|---------|--------|---------|---------|---------|
| 915 | CANTOR | Georg | 17.000 | CDP | P911 | 915 | Leipzig |
| "null" | "null" | "null" | "null" | "null" | P417 | 1000 | Dublin |
| 732 | DATE | Chris | 15.000 | CDP | P212 | 732 | Madrid |

RESULTAT

Liste de tous des salariés expatriés et éventuellement leur projet

RESULTAT

Liste de tous les projets et éventuellement le salarié

Les jointures - Jointure Externe

OUTER JOIN = Jointure externe

| SALARIÉ_EXPAT | <u>MLE</u> | NOM | PRÉNOM | SALAIRE | GRADE |
|---------------|------------|--------|--------|---------|-------|
| | 915 | CANTOR | Georg | 17.000 | CDP |
| | 628 | CODD | Ted | 20.000 | DP |
| | 545 | CURIE | Marie | 18.000 | CD |
| | 732 | DATE | Chris | 15.000 | CDP |

RIGHT, LEFT ou les deux FULL

| PROJET | <u>CODPROJ</u> | MLE_CDP | ADR_PROJ |
|--------|----------------|---------|----------|
| | P911 | 915 | Leipzig |
| | P417 | 1000 | Dublin |
| | P212 | 732 | Madrid |

SELECT s.mle, nom, prenom, salaire, grade, codproj, p.mle, ville

FROM salaries s FULL OUTER JOIN projet p on s.mle=p.mle

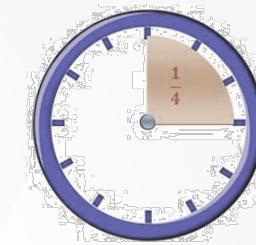
| MLE | NOM | PRÉNOM | SALAIRE | GRADE | CODPROJ | MLE_CDP | ADR_ROJ |
|--------|--------|--------|---------|--------|---------|---------|---------|
| 915 | CANTOR | Georg | 17.000 | CDP | P911 | 915 | Leipzig |
| 628 | CODD | Ted | 20.000 | DP | "null" | "null" | "null" |
| "null" | "null" | "null" | "null" | "null" | P417 | 1000 | Dublin |
| 545 | CURIE | Marie | 18.000 | CD | "null" | "null" | "null" |
| 732 | DATE | Chris | 15.000 | CDP | P212 | 732 | Madrid |

RESULTAT

Liste de tous des salariés expatriés et avec leur projet
 Et liste de tous les projets avec les salariés
 = Union des 2 Résultats précédents

Les jointures - Jointure Externe – Exercices

**Support « Exercices et résultats attendus » :
Jointures – Série 3**



Les jointures - Jointure Externe – Corrigés – Série 3

Liste des projets (codpro, libpro, type) n'ayant pas de compétences requises

```
SELECT a.codpro, libpro, type  
FROM projet a  
LEFT JOIN pro_cpc b ON a.codpro = b.codpro  
WHERE b.codpro is null
```

Liste des départements dont l'emploi a débuté avant le 1^{er} janvier 1991 ou pour lesquels il n'y a que le PDG/DG (sans employé), triée par désignation

```
SELECT a.codept, design, datent, pdg, dg  
FROM dept a LEFT JOIN emploi b ON a.codept = b.codept  
WHERE ( datent is null or datent < '1991-01-01') and pdg is not null  
ORDER BY a.design
```

III - Les requêtes avec le langage SQL

4. Sous-requêtes



Sous-Requête

Il est possible d'utiliser dans une qualification d'une requête le résultat d'un autre ordre SELECT.

Cette possibilité d'imbriquer les SELECT est à l'origine du terme SQL.

→ structuré et imbriqué ne sont pas synonymes...

Liste des ingénieurs du département 7.

```
SELECT NOM  
      FROM INGENIEUR  
     WHERE MLE IN  
           (SELECT MLE  
                 FROM EMPLOI  
                WHERE CODEPT = 7)
```

*Ensemble des
mle du département 7.
(933 286 ...)*

```
SELECT NOM  
      FROM INGENIEUR  
     WHERE MLE IN  
           ('933', '286',....)
```

Cette écriture peut faire croire que l'interpréteur évalue d'abord la requête imbriquée, créant une table de matricules résultat (933, 286,...).

En fait, les SGBDR évolués transforment cette requête en jointure.

Sous-Requête corrélée

QUE FAIT CETTE REQUÊTE ?

```
SELECT NOM  
      FROM INGENIEUR A  
     WHERE 7 IN  
           (SELECT CODEPT  
             FROM EMPLOI  
            WHERE MLE = A.MLE)
```

→ sous-requête corrélée.

On cherche les CODEPT
Où le mle « du dessus »
a travaillé

Exemple:

| Ingenieur | Emploi (Mle codept) | |
|-----------|---------------------|---|
| 600 | 600 | 7 |
| 200 | 200 | 3 |
| 300 | 300 | 3 |
| | 300 | 7 |

Exécution

On part d'Emploi on sélectionne le dépt du 600 – On regarde s'il appartient à 7 – On prend le 600

| | | | | | | |
|---|---|-----|---|---|---|-----------------|
| ” | ” | 200 | ” | ” | ” | On ne prend pas |
| ” | ” | 300 | ” | ” | ” | On ne prend pas |
| ” | ” | 300 | ” | ” | ” | On prend le 300 |

Sous-Requête corrélée

Sous-requête corrélée:

La sous-requête est exécutée pour chaque ligne de la table ingénieur.

L'opérande reliant les deux niveaux est : **IN**. Il peut être tout autre :

=, ^= ,<>, >, <, >=, <= Dans ce cas, la sous requête ne doit fournir qu'une seule valeur.

ALL ou ANY avec les opérandes précédents, la sous-requête peut fournir plusieurs valeurs

| | | |
|-------|--|--------------|
| > ALL | supérieur à toutes les valeurs renvoyées par la sous-requête. | (Tous) |
| = ANY | égal à n'importe quelle valeur renvoyée par la sous-requête. est similaire à IN | (Au moins 1) |

Sous-Requête

Liste des ingénieurs ayant le même numéro de téléphone que 'SOKAL'

```
SELECT NOM  
      FROM INGENIEUR  
     WHERE NOTEL =  
           (SELECT NOTEL  
              FROM INGENIEUR  
             WHERE NOM = 'SOKAL')
```

Il faut être sûr que la sous-requête n'amène qu'une ligne
sinon erreur à l'exécution

Sous-Requête - EXISTS

Cette requête liste également les ingénieurs du département 7

```
SELECT      NOM  
  FROM INGENIEUR A  
 WHERE EXISTS  
   (SELECT      'bidon'  
     FROM EMPLOI  
    WHERE A.MLE = MLE  
    AND CODEPT = 7 )
```

On veut savoir si le résultat de la sous-requête renvoie ou pas 1 information
Ensemble vide ou pas

Toute requête utilisant IN peut être traduite avec un EXISTS

→ Corrélation

La sous-requête est évaluée pour chaque ligne sélectionnée dans le premier SELECT

Sous-Requête NOT EXISTS

Liste des ingénieurs n'ayant pas travaillé pour le département 7.

```
SELECT          NOM
               FROM INGENIEUR A
              WHERE NOT EXISTS
                    (SELECT 'x'
                           FROM EMPLOI
                          WHERE A.MLE = MLE
                            AND CODEPT = 7
                     )
```

Sous-Requête - Exercice

Sous-Requête

Liste des ingénieurs (Masculins) domiciliés dans une ville où ne se trouve aucun projet

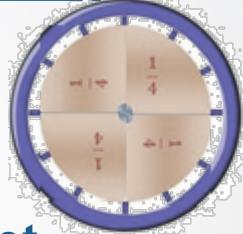
Plus complexe :

Liste des ingénieurs ayant toujours travaillé dans leur ville de résidence actuelle

Indications :

On prend les Ingénieurs affectés à 1 projet

On sélectionnera les villes dans lesquelles a travaillé un ingénieur



Sous-Requête - Correction

Liste des ingénieurs (Masculins) domiciliés dans une ville où ne se trouve aucun projet

Select nom, prenom, ville

from ingenieur

where sexe = 'M' and

ville not in (select ville from projet)

;

Autre possibilité

select i.mle, i.nom

from ingenieur i

left join projet p

on i.ville = p.ville

where p.codpro is null and sexe = 'M'

;

| | NOM | PRENOM | VILLE |
|---|-----------|---------|----------------------|
| 1 | GALLET | BRUNO | ORANGE |
| 2 | HERNANDEZ | ANTOINE | LE THOR |
| 3 | GREBOTO | CLAUDE | MONTPELLIER |
| 4 | GORBACHEV | MICHAIL | FLEURY MEROGIS CEDEX |
| 5 | MESRINE | RENE | FLEURY MEROGIS CEDEX |
| 6 | TURINCEV | SACHA | FLEURY MEROGIS CEDEX |
| 7 | MARCIER | JACQUES | BEAUMES DE VENISE |

Sous-Requête - Correction

Nom du dernier ingénieur qu'il est prévu d'embaucher.

Sans fonction récapitulative

2 noms sont trouvés en résultat Rebja et Lopez

→ 1 seul ingénieur Mle = 805 – Nom = Rebja

Sous-Requête - Correction

Nom du dernier ingénieur qu'il est prévu d'embaucher.

```
SELECT NOM  
FROM INGENIEUR I  
INNER JOIN EMPLOI E  
ON I.MLE = E.MLE  
WHERE DATENT >= ALL
```

On prend les Ingénieurs employés

pour lesquels leur date d'entrée est
supérieure ou égale à l'ensemble

des dates d'entrée de la table des emplois

(SELECT datent FROM EMPLOI)

;



2 ingénieurs : cependant Maria Lopez est déjà là et sera mutée ! Il ne faut pas la prendre en compte

Sous-Requête – Correction avec corrélation

Nom du dernier ingénieur qu'il est prévu d'embaucher.

```
SELECT NOM  
FROM INGENIEUR I  
INNER JOIN EMPLOI E ON I.MLE = E.MLE  
WHERE  
DATENT >= ALL  
(SELECT datent FROM EMPLOI )  
AND NOT EXISTS  
  
(SELECT 'X' FROM EMPLOI  
where mle = e.mle and datsor is not null)
```

On prend les Ingénieurs employés

pour lesquels leur date d'entrée est supérieure ou égale à l'ensemble des dates d'entrée de la table des emplois

Et pour lesquels il n'existe pas dans l'ensemble

le matricule dans la table Emploi ayant une date de sortie non nulle

Sous-Requête – Correction avec fonctions récapitulatives et regroupement

Nom du dernier ingénieur qu'il est prévu d'embaucher.

```
SELECT NOM  
FROM INGENIEUR I  
INNER JOIN EMPLOI E ON I.MLE = E.MLE  
WHERE DATENT =  
( SELECT max(datent)  
FROM EMPLOI )  
AND I.MLE IN  
(SELECT MLE FROM EMPLOI  
GROUP BY MLE  
HAVING COUNT(*) = 1)
```

On prend les Ingénieurs employés pour lequel sa date d'entrée est égale au Maximum de la date d'entrée de la table des emplois

Et pour lequel le matricule dans la table Emploi n'a qu'un département

Fonctions que l'on va aborder

Sous-Requête - Exercice

Liste des ingénieurs ayant toujours travaillé dans leur ville de résidence actuelle

→ 1 seul ingénieur Mle = 53
ville = Carpentras

Sous-Requête - Exercice

Liste des ingénieurs ayant toujours travaillé dans leur ville de résidence actuelle

Indications: On prend les Ingénieurs affectés à 1 projet

On sélectionnera les villes dans lesquelles a travaillé 1 Ing

Sous-Requête - Correction

Liste des ingénieurs ayant toujours travaillé dans leur ville de résidence actuelle

SELECT NOM, I.VILLE

FROM INGENIEUR I

INNER JOIN AFFECT A1

ON I.MLE = A1.MLE

WHERE I.VILLE = ALL

(SELECT VILLE

FROM PROJET O

INNER JOIN AFFECT A2

ON A2.CODPRO = O.CODPRO

WHERE A2.MLE = I.MLE)

;

On prend les Ingénieurs affectés à 1 projet

Pour lesquels leur ville appartient à l'ensemble

On sélectionne toutes les villes des projets
dans lesquelles a travaillé

l'ingénieur donné

III - Les requêtes avec le langage SQL

5. Vues



Vues – Définition – Crédit - Intérêts

Une vue est une **relation virtuelle** – Non physique (table = relation physique)

C'est le résultat de l'exécution de la requête **de création de la vue**

On peut consulter une vue avec 1 select

On peut mettre à jour le contenu d'une table via 1 vue

```
CREATE      VIEW PARISIENNES
AS          SELECT *
FROM        INGENIEUR
WHERE       SEXE = 'F'
AND         CODPOST BETWEEN 75000 AND 75999
```

La clause ORDER BY est interdite dans une vue

On aura les mêmes colonnes
que la table
Ingenieur

Intérêts

Projection →
Confidentialité,
Indépendance
dans les Apps,
Simplicité,
Ajout de colonnes

Vues – Consultation, utilisation

```
SELECT      *
FROM    PARISIENNES
```

```
SELECT      NOM, PRENOM
FROM    PARISIENNES
WHERE DATNAIS > '1980-02-01'
```

Equivaut à

```
SELECT      NOM, PRENOM
FROM    INGENIEUR
WHERE SEXE = 'F'
AND    CODPOST BETWEEN 75000 AND 75999
AND    DATNAIS > '1980-02-01'
```

III - Les requêtes avec le langage SQL

6 - Fonctions Récapitulatives - Partitionnement



Fonctions récapitulatives sur les colonnes

Les fonctions retournent q'une valeur unique pour un groupe de lignes

COUNT : Dénombrement des lignes de la table résultat.

Nombre de départements existants :

```
SELECT      COUNT (*)  
  FROM      DEPT
```

comment peut-on obtenir ce renseignement autrement ?

Nombre de prénoms féminins dans la table Ingénieur

```
SELECT      COUNT (DISTINCT PRENOM)  
  FROM      INGENIEUR  
 WHERE     SEXE = 'F'
```

Fonctions récapitulatives sur les colonnes

MAX : Maximum.

Coût le plus important de tous les projets :

```
SELECT MAX (COUT)  
FROM PROJET
```

MIN : Minimum.

Fonctions récapitulatives sur les colonnes

Pour les colonnes numériques uniquement

SUM : Somme des éléments .

Coût total de tous les projets situés à PARIS :

```
SELECT      SUM (COUT)
  FROM        PROJET
  WHERE       VILLE='PARIS'
```

AVG : Calcul de la moyenne .

Remarque : Les « fonctions de fonctions » Moyenne de la somme ne sont pas réalisables chez tous les SGBDr.

Partitionnement logique

GROUP BY: Permet de partitionner la table résultat du **SELECT** selon la valeur d'une colonne.

Liste des matricules avec le nombre de projets sur lesquels ils ont travaillé

**SELECT MLE, COUNT(*)
FROM AFFECTATION
GROUP BY MLE**

Résultat:

| MLE | COUNT(*) |
|-----|----------|
| 628 | 1 |
| 732 | 2 |
| 915 | 2 |

MLE

| AFFECTATION | MLE | CODPROJ | DAT_DEB | DAT_FIN |
|-------------|-----|---------|------------|------------|
| | 915 | P212 | 02-03-1995 | 12-11-1996 |
| | 628 | P911 | 01-05-1995 | 30-06-1996 |
| | 732 | P911 | 10-09-1994 | 03-02-1995 |
| | 915 | P911 | 01-12-1996 | "null" |
| | 732 | P212 | 01-03-1996 | "null" |

Le résultat est trié sur le MLE, car le SGBD doit trier la table pour gérer les ruptures sur le matricule.

Partitionnement logique

GROUP BY

Les expressions apparaissant dans le **SELECT** ne peuvent être que :

- Des colonne(s) du **GROUP BY**.
- Des constantes.
- Une fonction de colonne telle que **COUNT, SUM, MAX, MIN, AVG**, s'appliquant sur chaque partition.

Partitionnement logique

HAVING : est au groupe de lignes issu du **GROUP BY** ce que **WHERE** est à la ligne élimine les groupes de lignes ne répondant pas au critère.

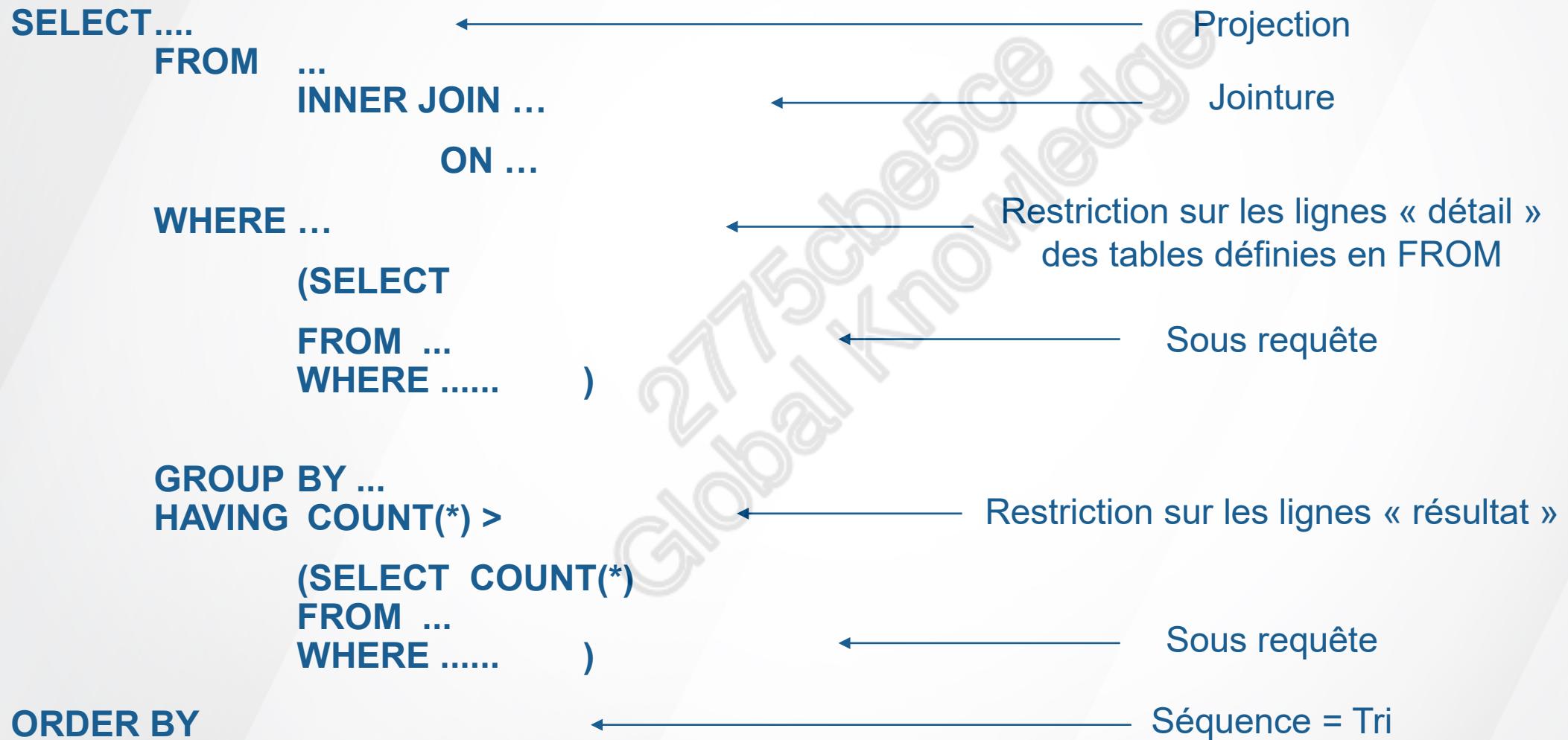
Liste des ingénieurs ayant travaillé sur plus de 2 projets

```
SELECT MLE  
      FROM AFFECTATION  
     GROUP BY MLE  
    HAVING COUNT(*) > 2
```

Les expressions du **HAVING** ne peuvent porter que sur :

- des fonctions de colonne (**COUNT**, **SUM**, **MAX**...).
- une ou plusieurs colonne(s) du **GROUP BY**
- l'autre opérande peut être le résultat d'une sous-requête

Partitionnement logique – Syntaxe du select



Fonctions récapitulatives Partitionnement logique - Exercices

Fonctions d'agrégation – Partitionnement

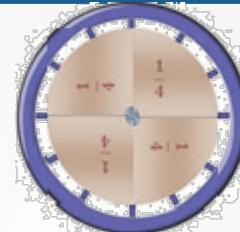
Deux séries

- 1. Moyenne des prix de vente des ingénieurs par projet (affectation en cours).**

- 2. Liste des ingénieurs (mle, nom, codproj, pvente) dont le prix de vente est supérieur à la moyenne des prix de vente tous projets confondus (Affectations en cours uniquement).**

- 3. Nombre d'ingénieurs par département (actuellement).**

Liste des projets occupant actuellement deux ingénieurs.



Fonctions récapitulatives Partitionnement logique - Corrections

1. Moyenne des prix de vente des ingénieurs par projet (affectation en cours).

```
SELECT CODPRO, AVG(PVENTE)
FROM AFFECT
WHERE (DATFIN IS NULL OR DATFIN > CURRENT DATE)
AND DATDEB < CURRENT DATE
GROUP BY CODPRO
;
```

Fonctions récapitulatives Partitionnement logique - Corrections

2. Liste des ingénieurs (mle, nom, codproj, pvente) dont le prix de vente est supérieur à la moyenne des prix de vente tous projets confondus (Affectations en cours uniquement).

```
SELECT I.MLE, I.NOM, A.CODPRO, A.PVENTE  
FROM INGENIEUR I  
INNER JOIN AFFECT A ON I.MLE = A.MLE  
WHERE (A.DATFIN IS NULL OR A.DATFIN >= CURRENT DATE)  
AND A.PVENTE >  
(SELECT AVG(PVENTE)  
FROM AFFECT  
WHERE DATFIN IS NULL OR DATFIN >= CURRENT DATE)  
;
```

Fonctions récapitulatives Partitionnement logique - Corrections

3. Nombre d'ingénieurs par département (actuellement).

```
SELECT DESIGN, COUNT(*)  
FROM EMPLOI  
    INNER JOIN DEPT D  
        ON E.CODEPT = D.CODEPT  
WHERE (E.DATSOR IS NULL OR E.DATSOR > CURRENT DATE)  
AND E.DATENT < CURRENT DATE  
GROUP BY D.CODEPT, DESIGN  
;
```

Fonctions récapitulatives Partitionnement logique - Corrections

4. Liste des projets occupant actuellement deux ingénieurs.

```
SELECT CODPRO, LIBPRO  
FROM PROJET  
WHERE CODPRO IN (    ← Sous-requête  
    SELECT CODPRO  
    FROM AFFECT  
    WHERE (DATFIN IS NULL OR DATFIN > CURRENT DATE)  
        AND DATDEB < CURRENT DATE  
    GROUP BY CODPRO  
    HAVING COUNT(*) = 2    )
```

Jointure →

```
SELECT p.CODPRO, p.LIBPRO, COUNT(a.MLE)  
FROM PROJET p  
    INNER JOIN AFFECT a ON p.CODPRO=a.CODPRO  
WHERE (DATFIN IS NULL OR DATFIN > CURRENT DATE)  
    AND DATDEB < CURRENT DATE  
GROUP BY p.CODPRO  
HAVING COUNT(a.MLE) = 2
```

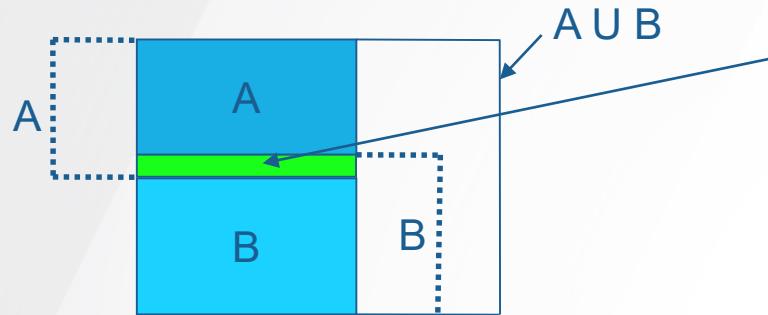
III - Les requêtes avec le langage SQL

7. Autres opérateurs ensemblistes

A photograph of two people in an office setting. A man with glasses and a denim shirt is gesturing with his hands while speaking. A woman with glasses and a yellow top is listening. They are sitting at a desk with papers and charts. A large watermark "Global 275CCE" is diagonally across the image.

opérateurs ensemblistes

Opérateurs ensemblistes - UNION



Liste des matricules dont le prénom est 'MAX' ou ayant travaillé au département 7

```
SELECT MLE  
      FROM INGENIEUR  
     WHERE PRENOM = 'MAX'
```

UNION

```
SELECT MLE  
      FROM EMPLOI  
     WHERE CODEPT = 7
```

ORDER BY 1

Union élimine les doubles

Lignes appartenant à l'une ou à l'autre des 2 relations

REMARQUES

Le nombre de SELECT est quelconque.

Les colonnes sélectionnées doivent être :

- De même description.
- Dans le même ordre.

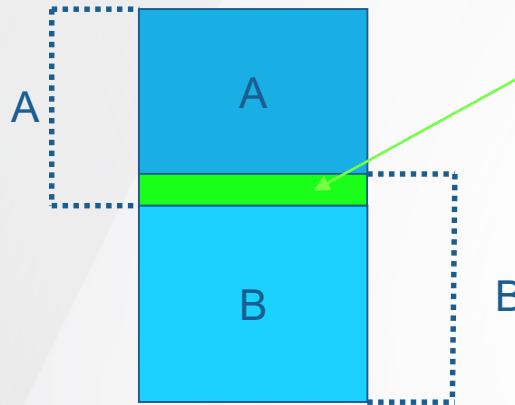
ORDER BY selon un rang de colonne (et non un nom).

UNION est interdit dans une sous-requête.

Les matricules redondants n'apparaissent qu'une fois.

Certains SGBDR permettent d'écrire UNION ALL, pour obtenir les matricules redondants.

Opérateurs ensemblistes - INTERSECTION



A n B Intersection de A et B

Lignes appartenant à la fois aux 2 relations

Liste des clients associés à un projet

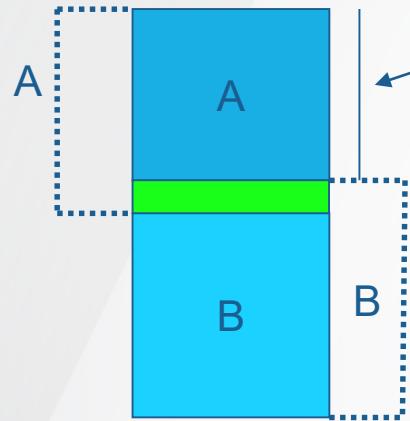
```
SELECT      CODCLI  
FROM       CLIENT
```

INTERSECT

```
SELECT      CODCLI  
FROM       CLIPRO
```

Certains SGBDR ne permettent pas cet opérateur

Opérateurs ensemblistes - DIFFERENCE



A - B Différence

Lignes appartenant à la première relation mais pas à la seconde

Liste des ingénieurs n'ayant jamais eu de promotion

```
SELECT      MLE  
FROM       INGENIEUR
```

EXCEPT (ou MINUS)

```
SELECT      MLE  
FROM       PROMOTION
```

Certains SGBDR ne permettent pas cet opérateur

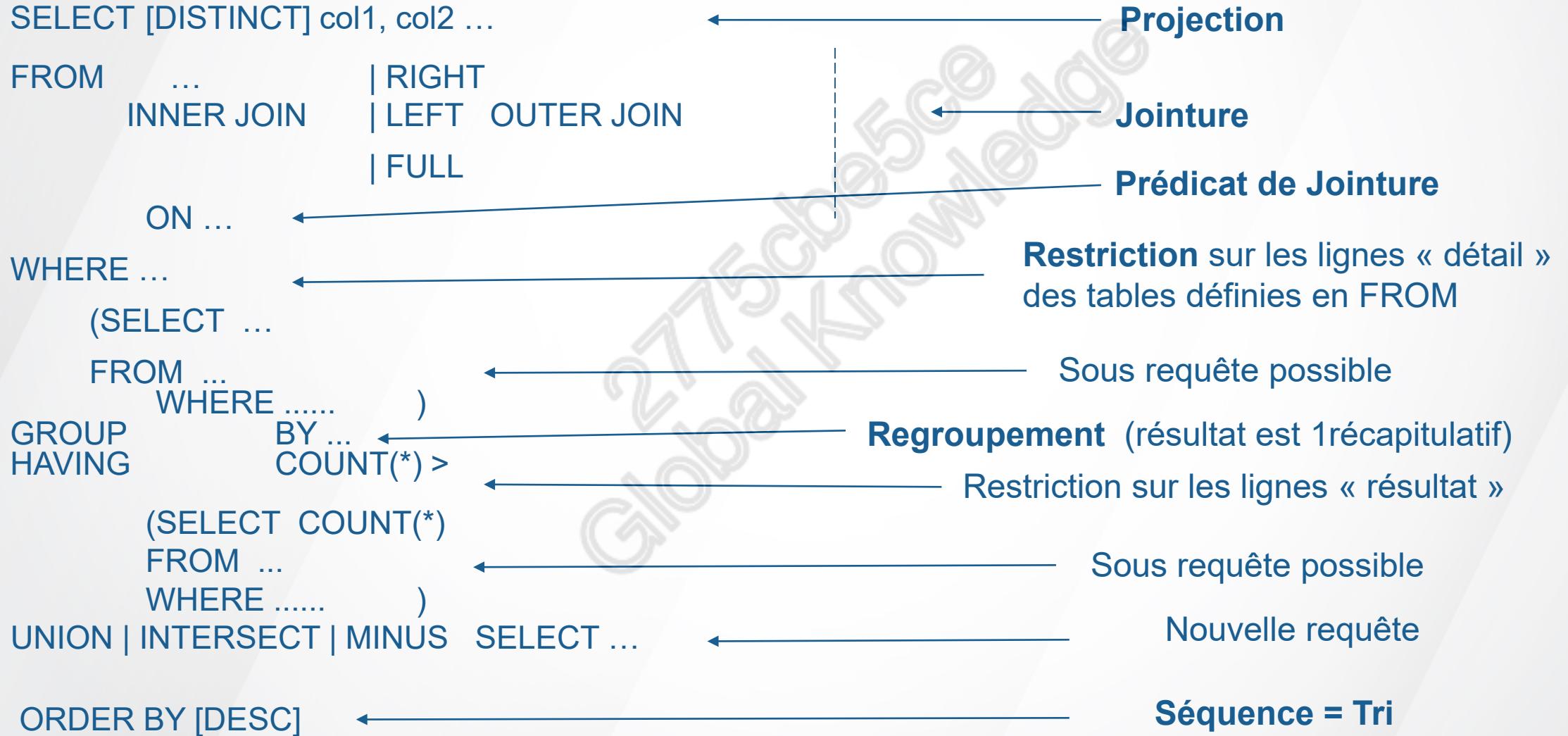
III - Les requêtes avec le langage SQL

8. Récapitulatif

A photograph of two people in an office setting. A man with glasses and a denim shirt is gesturing with his hands while speaking. A woman with glasses and a yellow top is listening. There are papers and a pen on the desk between them.

opérateurs ensemblistes

Résumé - Syntaxe du select



Opérateurs ensemblistes Exercices



III - Les requêtes avec le langage SQL

9. Mises à jour avec SQL

A photograph of two people in an office setting. A man with glasses and a denim shirt is gesturing with his hands while speaking. A woman with glasses and a yellow top is listening. They are sitting at a desk with papers and charts. A large watermark "2775Global" is diagonally across the image.

opérateurs ensemblistes

Mises à jour - INSERT

Syntaxe :

```
INSERT INTO nom_table  
[(colonne1, colonne2,...)]  
VALUES (valeur1, valeur2,...)
```

Exemples :

```
INSERT INTO DEPT  
(codept, design)  
VALUES  
(20, 'GESTION ')
```

Les autres colonnes ne sont pas à préciser : sauf si elles sont définies avec NOT NULL

Sans renseigner les colonnes

```
INSERT INTO EMPLOI  
VALUES  
(8910, '1984-10-03' , 7, NULL)
```

Toutes les colonnes sont à renseigner et dans l'ordre de la table

Mises à jour - INSERT

Syntaxe :

UPDATE nom_table

Set col1=valeur, col2=valeur

WHERE condition

Exemples :

UPDATE dept

SET pdg = 'DUPONT '

WHERE codept = 1

Mise à jour d'une ligne

Plusieurs lignes :

UPDATE projet

Set cout = 1.2 * cout

WHERE ville = 'PARIS '

Mises à jour - DELETE

Syntaxe :

DELETE FROM nom_table
WHERE condition

Exemples :

DELETE FROM
Ingénieur
WHERE mle = 933

Mise à jour d'une ligne

Plusieurs lignes

DELETE FROM ingenieur
WHERE ville = 'PARIS'

Toutes les lignes

DELETE FROM ingenieur

Mises à jour - Principes de la journalisation

| MLE | NOM | PRÉNOM | FONCTION | GRADE |
|-----|--------|--------|----------|-------|
| 915 | KANTOR | Georg | DP | G7 |
| 628 | CODD | Ted | CDP | G5 |
| 732 | DATE | Chris | AP | G5 |

Sauvegarde externe
de la base

Mises à jour de la base par les Users

| MLE | NOM | PRÉNOM | FONCTION | GRADE |
|-----|--------|--------|----------|-------|
| 915 | KANTOR | Georg | DP | G7 |
| 628 | CODD | Ted | CDP | G5 |
| 732 | DATE | Chris | AP CDP | G5 |
| 10 | Dupont | Pierre | DP | G7 |

Mises à jour dans le journal par le système

| Table | Action | Valeurs de la ligne | | |
|---------|-------------|---------------------|--------|--------------|
| Salarié | Insert | 10 | Dupont | Pierre DP G7 |
| Salarié | Modif Avant | 732 | DATE | Chris AP G5 |
| Salarié | Modif Après | 732 | DATE | Chris CDP G5 |

Point validation = COMMIT

.../...

Point d'invalidation = ROLLBACK

Mises à jour - Principes de la journalisation

| MLE | NOM | PRÉNOM | FONCTION | GRADE |
|-----|--------|--------|----------|-------|
| 915 | KANTOR | Georg | DP | G7 |
| 628 | CODD | Ted | CDP | G5 |
| 732 | DATE | Chris | AP | G5 |

Sauvegarde externe
de la base

Mises à jour de la base par les Users

Plantage

| MLE | NOM | PRÉNOM | FONCTION | GRADE |
|-----|--------|--------|----------|-------|
| 915 | KANTOR | Georg | DP | G7 |
| 628 | CODD | Ted | CDP | G5 |
| 732 | DATE | Chris | AP | G5 |
| 10 | Dupont | Pierre | DP | G7 |

| MLE | NOM | PRÉNOM |
|-----|--------|--------|
| 915 | KANTOR | Georg |
| 628 | CODD | Ted |
| 732 | DATE | Chris |
| 10 | Dupont | Pierre |

Mises à jour dans le journal par le système

| Table | Action | Valeurs de la ligne | | |
|---------|-------------|---------------------|--------|--------------|
| Salarié | Insert | 10 | Dupont | Pierre DP G7 |
| Salarié | Modif Avant | 732 | DATE | Chris AP G5 |
| Salarié | Modif Après | 732 | DATE | Chris CDP G5 |

Point validation = COMMIT

.../...

Point d'invalidation = ROLLBACK

Restauration de la base / Ré implantation des mises à jour à partir du journal
→ Récupération de la base

Mises à jour - Transactions

Transaction : unité logique de travail

Comprend un ensemble d'ordres SQL indissociables, exécutés par un seul utilisateur.

Les propriétés ACID d'une transaction :

- **Atomicité** des instructions, considérées comme indissociables les unes des autres. Soit elles arrivent toutes à terme, soit elles sont toutes annulées.
- **Cohérence** de la base, qui passe d'un état cohérent à un autre état cohérent.
- **Isolation** des opérations, par le mécanisme des verrous.
- **Durabilité** des mises à jour, une fois qu'une validation est faite, il n'est plus possible de demander l'annulation des mises à jour.

Conclusion

Merci

aline.cassar@globalknowledge.fr



Conclusion

- Renseigner le formulaire d'évaluation
- Chacun choisit 3 mots qui décrivent le mieux ses sentiments sur la formation :

A l'aise, absorbé, abattu, ahuri, agacé, allégé, agité, amusé, animé, attentif, apathique, de bonne humeur, bloqué, calme, captivé, centré, charmé, concentré, concerné, confiant, confortable, content de soi, curieux, détaché, déconcerté, détendu, déçu, emballé, embrouillé, enchanté, encouragé, ennuyé, étonné, éveillé, éreinté, étourdi, fier, fatigué, gai, galvanisé, hilare, impatient, impliqué, informé, inquiet, insouciant, indifférent, intéressé, joyeux, libre, nourri, optimiste, paisible, rassuré, ravi, satisfait, sceptique, sensibilisé, soulagé, stimulé, vindicatif, valorisé, zen

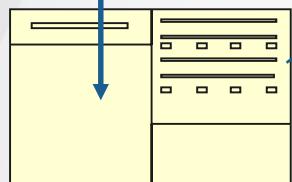
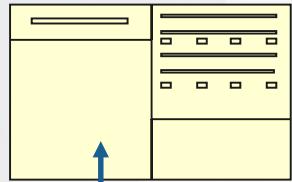
A photograph of two people in an office setting. A man with glasses and a denim shirt is gesturing with his hands while speaking. A woman in a yellow top is partially visible on the right, looking towards him. They are seated at a desk covered with several papers containing charts and graphs.

Aller plus loin dans la data

opérateurs ensemblistes

Evolution des SGBD – Base de données réparties

Serveur 1



Serveur 2

Gestion décentralisée



Une base répartie permet :

- Une plus grande fiabilité
- De meilleures performances
- De faciliter l'accroissement du volume de données, du nombre d'utilisateurs,.. (scalability)

Gestion de la répartition par:
réPLICATION ou la FRAGMENTATION

La réPLICATION est une copie de chaque relation.

La FRAGMENTATION est la réPARTITION d'une relation sur plusieurs sites. Fragmentation horizontale ou Fragmentation verticale

Evolution des SGBD – NoSQL

Not only SQL, 4 types de bases NoSQL :

➤ Clé/valeur

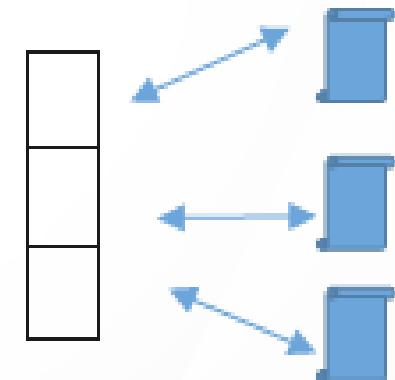
≈ simple tableau associatif, 1 dimension,
des millions, voire des milliards d'entrées
Pas de structure

DynamoDB
Aerospike
Redis
Riak

| Clé | Valeur |
|-----|--------|
| | |
| | |
| | |
| | |
| | |
| | |
| | |

➤ Orienté Document

Clé / Document



DynamoDB
CouchDB
MongoDB

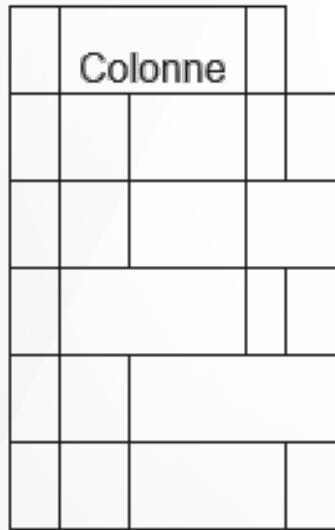
Document soit XML
 soit Jason

Evolution des SGBD – NoSQL

➤ Orienté colonnes

≈ SGBDR les lignes peuvent avoir des colonnes différentes

Cassandra
Hbase
Accumulo



➤ Orienté Graphe

Nœuds et arcs orientés

InfiniteGraph
Neo4j

