Cours SQL

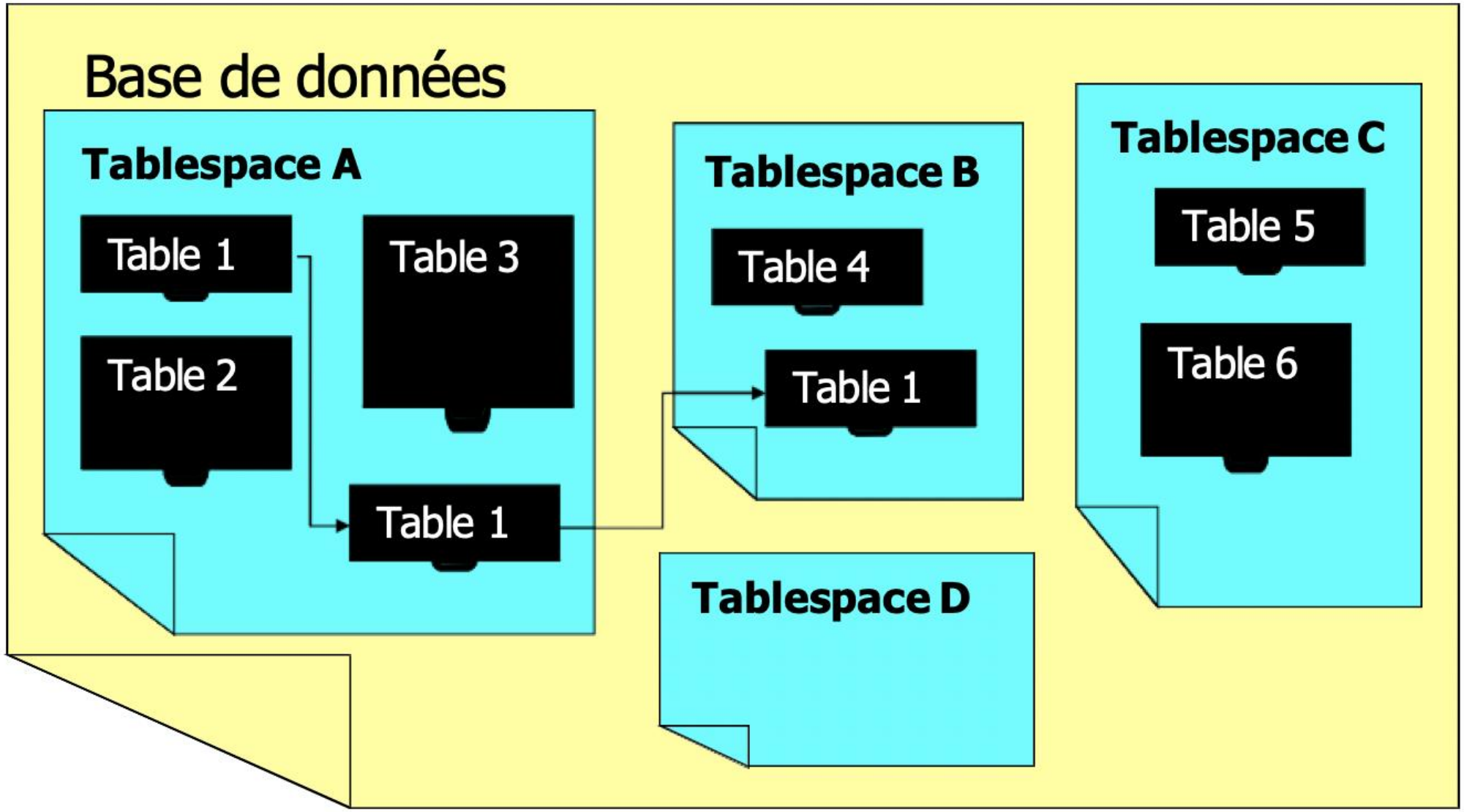
Structure du cours

- Les bases de données relationnelles
- La structure d’une base de données oracle et postgresql
- L’interrogation des données
- La mise à jour des données
- Le langage de définition des données
- Le langage de programmation PL/SQL

Les bases de données relationnelles

- **Le SQL : Structured Query Language**
  - **Le langage SQL est un langage simple qui permet de travailler sur une base de données**
    - LID
      - Langage d'interrogation des données
      - SELECT
    - LDD
      - Langage de définition des données
      - ALTER, CREATE, DROP
    - LMD
      - Langage de manipulation des données
      - UPDATE, INSERT, DELETE
    - LCD
      - Langage de contrôle des données
      - GRANT, REVOKE

La structure d’une base de données PostgreSQL



L'interrogation des données La syntaxe du SELECT

- **Permet d'interroger de manière simple des données**

- **SELECT [DISTINCT] { * | [nom_table.]nom_col1[,nom.col2....] | [,expression] }**
  FROM { [nom_créateur1.]nom_table1[,nom_créateur2.]nom_table2...]}
  [WHERE prédicat]
  [GROUP BY nom_coli[,nom_colj.....] [HAVING prédicat]]
  [ORDER BY nom_colp [DESC] [,nom_colq [ASC] ...] ;
- **On peut introduire un commentaire dans un ordre SQL en l'encadrant par /* Commentaire*/**
- **Syntaxe :**
  - entre crochets des arguments facultatifs
  - entre accolades des arguments obligatoire
  - les choix possibles sont séparés par des barres verticales
- **SELECT : choix des colonnes à afficher ➔ Projection**
- **FROM : choix des tables de travail ➔ Relations à traiter**
- **WHERE : condition sur les données à sélectionner ➔ Qualification**
- **GROUP BY/ HAVING/ ORDER BY : vu plus tard**

L’interrogation des données

- **La syntaxe du SELECT**
  - **SELECT * : permet de choisir toutes les colonnes d’une table**
  - **SELECT col1 : permet de choisir une colonne d’une table**
  - **SELECT distinct : permet de choisir des valeurs distinctes d’une table**
  - **Exemple : Les prénoms différents des employés**

EXERCICES

Simples sélections

- **1 : Sélectionner toutes les colonnes de la table SERV.**
- **2 : Sélectionner d’une autre manière ces colonnes.**
- **3 : Sélectionner les colonnes SERVICE et NOSERV de la table SERV.**
- **4 : Sélectionner toutes les colonnes de la table EMP.**
- **5 : Sélectionner les emplois de la table EMP.**
- **6 : Sélectionner les différents emplois de la table EMP.**

SELECT : Les valeurs nulles

- **Une valeur nulle est différente de la valeur zéro. Elle correspond à une valeur non renseignée.**
- **Il faut comprendre cette valeur comme l'ensemble vide et imaginer les calculs en découlant :**
  - NULL + 3 donne NULL.
  - NULL = 0 est faux.
  - NULL n'est pas inférieur à 3 et NULL n'est pas supérieur à 3.
- **Pour détecter une clause NULL, il faut utiliser le prédicat IS NULL. L'inverse étant IS NOT NULL.**
- **Pour transformer la clause Null SUR ORACLE, on peut utiliser la fonction NVL (Null VaLue):**

  NVL(champ, 0) donne la valeur 0 si le champ est nul sinon donne sa valeur.
  - **Pour transformer la clause Null sur postgreSQL, on peut utiliser la fonction COALESCE :**

    COALESCE(champ, 0) donne la valeur 0 si le champ est nul sinon donne sa valeur.
  - Important pour protéger les colonnes pouvant être nulles.

EXERCICES

SELECT AVEC CLAUSE WHERE : Prédicats simples

- **7 : Sélectionner les employés du service N°3.**
- **8 : Sélectionner les noms, prénoms, numéro d'employé, numéro de service de tous les techniciens.**
- **9 : Sélectionner les noms, numéros de service de tous les services dont le numéro est supérieur à 2.**
- **10 : Sélectionner les noms, numéros de service de tous les services dont le numéro est inférieur ou égal à 2.**
- **11 : Sélectionner les employés dont la commission est inférieure au salaire. Vérifiez le résultat jusqu'à obtenir la bonne réponse.**

EXERCICES

SELECT : Travailler avec les nuls

- **12 : Sélectionner les employés qui ne touchent jamais de commission.**
- **13 : Sélectionner les employés qui touchent éventuellement une commission et dans l'ordre croissant des commissions.**
- **14 : Sélectionner les employés qui ont un chef.**
- **15 : Sélectionner les employés qui n'ont pas de chef.**

SELECT : Les opérateur logiques AND, OR et NOT

- **Rappels logiques**
  - VRAI AND VRAI donne VRAI
  - VRAI AND FAUX donne FAUX
  - FAUX AND FAUX donne FAUX
  - VRAI OR VRAI donne VRAI
  - VRAI OR FAUX donne VRAI
  - FAUX OR FAUX donne FAUX
- **AND prioritaire sur le OR : il faut donc <u>parenthéser le OR</u>**
- **NOT inverse le résultat logique de l'opération**

EXERCICES

SELECT AVEC CLAUSE WHERE : and, or, not

- **16 : Sélectionner les noms, emploi, salaire, numéro de service de tous les employés du service 5 qui gagnent plus de 20000 €.**
- **17 : Sélectionner les vendeurs du service 6 qui ont un revenu mensuel supérieur ou égal à 9500 €.**
- **18 : Sélectionner dans les employés tous les présidents et directeurs. Attention, le français et la logique sont parfois contradictoires.**
- **19 : Sélectionner les directeurs qui ne sont pas dans le service 3.**
- **20 : Sélectionner les directeurs et « les techniciens du service 1 ».**
- **21 : Sélectionner les « directeurs et les techniciens » du service 1.**
- **22 : Sélectionner les employés du service 1 qui sont directeurs ou techniciens.**
- **23 : Sélectionner les employés qui ne sont ni directeur, ni technicien et travaillant dans le service 1.**

Les différents types de données(ORACLE)

- **Trois types principaux de données**
  - **Numérique :** NUMBER(precision, scale)
    - NUMBER(5,2) signifie 5 chiffres dont 2 après la virgule (par ex : 432,36)
  - **Caractère : VARCHAR2(longueur)**
    - VARCHAR2(10) signifie une chaine de 10 caractères maximum (ex : 'Chien')
    - Attention différence entre les minuscules et majuscules
  - **Date : DATE**
    - Stockage interne sous forme d'un nombre en jours à partir d'une date de référence
    - Les minutes et secondes sont stockés en fraction de journée
    - Format d'affichage par défaut suivant le paramétrage de la base

Les différents types de données (POSTGRESQL)

- **Trois types principaux de données**
  - **Numérique :** NUMERIC(precision, scale)
    - NUMERIC(5,2) signifie 5 chiffres dont 2 après la virgule (par ex : 432,36)
    - (VOIR https://www.postgresql.org/docs/9.1/static/datatype-numeric.html)
  - **Caractère : VARCHAR(longueur)**
    - VARCHAR(10) signifie une chaine de 10 caractères maximum (ex : 'Chien')
    - Attention différence entre les minuscules et majuscules
    - (VOIR https://www.postgresql.org/docs/9.3/static/datatype-character.html)
  - **Date : DATE**
    - Stockage interne sous forme d'un nombre en jours à partir d'une date de référence
    - Les minutes et secondes sont stockés en fraction de journée
    - Format d'affichage par défaut suivant le paramétrage de la base
    - (VOIR https://www.postgresql.org/docs/9.1/static/datatype-datetime.html)

Les opérateurs

- **Les symboles relationnels**

égal	=
inférieur	<
supérieur	>
inférieur ou égal	<=
supérieur ou égal	>=
différent	<> (!= ou ^=)

(VOIR https://www.postgresql.org/docs/9.6/static/functions-comparison.html)

- **L'ordre**

TYPE DE DONNEE	*ORDRE CORRESPONDANT*
NUMERIQUE	ORDRE NATUREL DES REELS
DATE	ORDRE CHRONOLOGIQUE
CARACTERE	ORDRE LEXICOGRAPHIQUE

Les opérateurs : LIKE, IN, BETWEEN

- **LIKE (https://www.postgresql.org/docs/9.0/static/functions-matching.html#FUNCTIONS-LIKE)**
  - Syntaxe : **expr LIKE chaîne modèle**
  - chaîne est une chaîne de caractères pouvant contenir l'un des caractères jokers suivants:
    - '_' underscore remplace 1 caractère exactement.
    - '%' remplace une chaîne de caractères de longueur quelconque, y compris de longueur nulle.
- **IN (VOIR https://www.postgresql.org/docs/9.0/static/functions-comparisons.html#AEN16964)**
  - **expr1 IN (expr2, expr3 . . .)**
  - Vrai si expr1 est égale à une des expressions de la liste entre parenthèses.
  - NOT IN pour inverser cette condition
- **BETWEEN (VOIR https://www.postgresql.org/docs/9.6/static/functions-comparison.html)**
  - expr1 BETWEEN expr2 AND expr3
  - Vrai si expr1 est compris entre expr2 et expr3, bornes incluses.

EXERCICES

• Prédicats construits avec IN, BETWEEN, LIKE

- **24 : Sélectionner les employés qui sont techniciens, comptables ou vendeurs.**
- **25 : Sélectionner les employés qui ne sont ni technicien, ni comptable, ni vendeur.**
- **26 : Sélectionner les directeurs des services 2, 4 et 5.**
- **27 : Sélectionner les subalternes qui ne sont pas dans les services 1, 3, 5.**
- **28 : Sélectionner les employés qui gagnent entre 20000 et 40000 euros, bornes comprises.**
- **29 : Sélectionner les employés qui gagnent moins de 20000 et plus de 40000 euros.**
- **30 : Sélectionner les employés qui ne sont pas directeur et qui ont été embauchés en 88.**
- **31 : Sélectionner les directeurs des services 2 ,3 , 4, 5 sans le IN.**
- **32 :Sélectionner les employés dont le nom commence par la lettre M.**
- **33 : Sélectionner les employés dont le nom se termine par T.**
- **34 : Sélectionner les employés ayant au moins deux E dans leur nom.**
- **35 : Sélectionner les employés ayant exactement un E dans leur nom.**
- **36 : Sélectionner les employés ayant au moins un N et un O dans leur nom.**
- **37 : Sélectionner les employés dont le nom s'écrit avec 6 caractères et qui se termine par N.**
- **38 : Sélectionner les employés dont la troisième lettre du nom est un R.**
- **39 : Sélectionner les employés dont le nom ne s'écrit pas avec 5 caractères.**

SUR POSTGRESQL

- ET AUSSI https://www.postgresql.org/docs/9.6/static/functions.html

SELECT : ORDER BY

- **Les critères de tri sont spécifiés dans une clause ORDER BY figurant après la clause FROM et la clause WHERE**
- **La syntaxe est la suivante :**
  - ORDER BY nom_col1 | num_col1 [DESC], [nom_col2 | num_col2 [DESC], ...
- **Le tri se fait selon la colonne 1, les lignes ayant la même valeur dans la colonne 1 sont triées selon la colonne 2, etc...**
- **Pour chaque colonne le tri peut être ascendant (ASC option par défaut) ou descendant (option DESC)**
- **Pour simplifier, il n'est pas indispensable de nommer explicitement la colonne mais en donnant sa position dans le Select.**

EXERCICES :Maître de l’ordre

- **40 : Trier les employés (nom, prénom, n° de service, salaire) du service 3 par ordre de salaire croissant.**
- **41 : Trier les employés (nom, prénom, n° de service , salaire) du service 3 par ordre de salaire décroissant.**
- **42 : Idem en indiquant le numéro de colonne à la place du nom colonne.**
- **43 : Trier les employés (nom, prénom, n° de service, salaire, emploi) par emploi, et pour chaque emploi par ordre décroissant de salaire.**
- **44 : Idem en indiquant les numéros de colonnes.**
- **45 : Trier les employés (nom, prénom, n° de service, commission) du service 3 par ordre croissant de commission.**
- **46 : Trier les employés (nom, prénom, n° de service, commission) du service 3 par ordre décroissant de commission, en considérant que celui dont la commission est nulle ne touche pas de commission.**

LES OPERATEURS ENSEMBLISTES LA JOINTURE

- **C'est une opération qui permet d'obtenir dans une même ligne des informations provenant de plusieurs tables.**

  SELECT ...

  FROM table1, table2, table3, ..., tablen

- **Dans le cas de deux tables ayant une colonne en commun, on sélectionne les couples ayant une valeur commune dans cette colonne, on dit alors que l'on réalise une JOINTURE des deux tables, bien entendu cette sélection est exprimée à l'aide d'une clause WHERE.**
- **Lorsqu'il y a ambiguïté sur la table à laquelle appartient une colonne, il faut préfixer cette colonne avec le nom de la table (ou son alias).**
- **Exemple: afficher la liste des employés avec la ville dans laquelle ils travaillent.**
- **Pour une jointure d'une table avec elle-même, il faut utiliser des alias différents pour cette table.**

EXERCICES : Make a join !

- **47 : Sélectionner le nom, le prénom, l'emploi, le nom du service de l'employé pour tous les employés.**
- **48 : Sélectionner le nom, l'emploi, le numéro de service, le nom du service pour tous les employés.**
- **49 : Idem en utilisant des alias pour les noms de tables.**
- **50 : Sélectionner le nom, l'emploi, suivis de toutes les colonnes de la table SERV pour tous les employés.**
- **51 : Sélectionner les nom et date d'embauche des employés suivi des nom et date d'embauche de leur supérieur pour les employés plus ancien que leur supérieur, dans l'ordre nom employés, noms supérieurs.**

LES OPERATEURS ENSEMBLISTES LA JOINTURE EXTERNE

- **Permet de lier une table 1 à une table 2 même s'il n'y a pas d'enregistrement correspondant dans la table 2.**

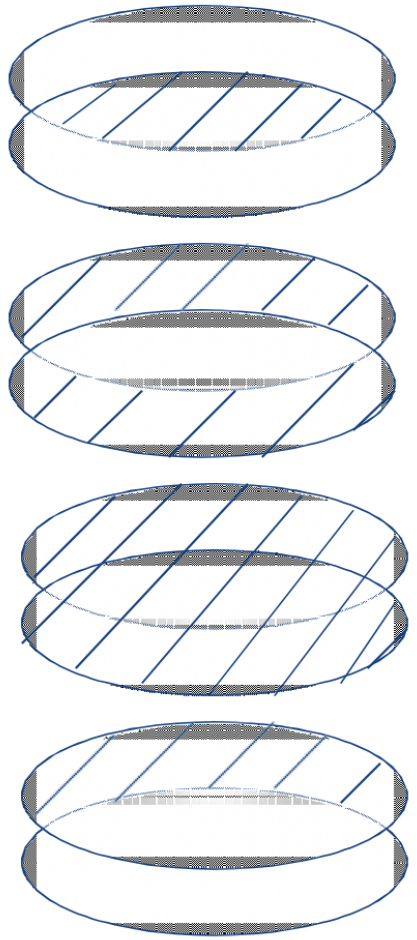
```
SELECT expr
FROM table1, table2
WHERE table1.champ1 = table2.champ2(+)
```

- **S'il n'y a pas de lien, la valeur du champ en jointure externe sera nulle.**
- **Le signe (+) indique la table en jointure externe.**
- **Exemple : Sélectionner les services n'ayant pas d'employés.**

```
SELECT serv.*
FROM emp, serv
WHERE serv.noserv = emp.noserv(+)  AND noemp is null;
```

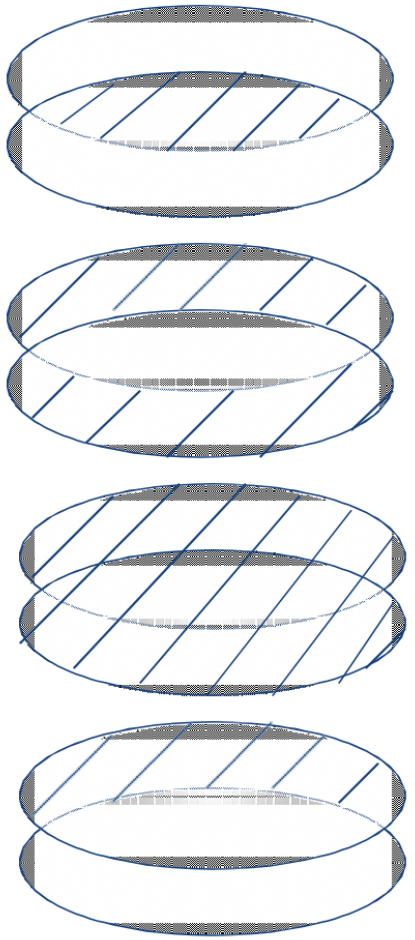
LES OPERATEURS ENSEMBLISTES UNION et MINUS (ORACLE)

- **INTERSECT**
  - Elle donne pour résultat les lignes communes entre la première et la seconde requête
- **UNION**
  - Elle donne pour résultat les lignes de la première requête suivies de celles de la deuxième requête **sans** les doublons.
- **UNION ALL**
  - Elle donne pour résultat les lignes de la première requête suivies de celles de la deuxième requête **avec** les doublons.
- **MINUS**
  - Elle donne pour résultat les lignes de l'union des deux requêtes qui ne sont pas dans la deuxième requête.



LES OPERATEURS ENSEMBLISTES UNION et MINUS (POSTGRESQL)

- **INTERSECT**
  - Elle donne pour résultat les lignes communes entre la première et la seconde requête
- **UNION**
  - Elle donne pour résultat les lignes de la première requête suivies de celles de la deuxième requête **sans** les doublons.
- **UNION ALL**
  - Elle donne pour résultat les lignes de la première requête suivies de celles de la deuxième requête **avec** les doublons.
- **MINUS ➔ EXCEPT**
  - Elle donne pour résultat les lignes de l'union des deux requêtes qui ne sont pas dans la deuxième requête.

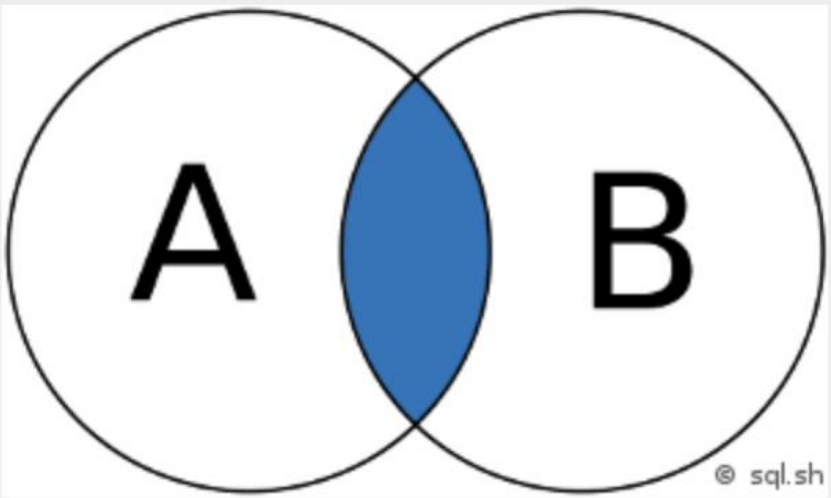


AUTRES SYNTAXE

- Il y a plusieurs méthodes pour associer 2 tables ensemble. Voici la liste des différentes techniques qui sont utilisées :
- **INNER JOIN :** jointure interne pour retourner les enregistrements quand la condition est vrai dans les 2 tables. C'est l'une des jointures les plus communes.
- **CROSS JOIN :** jointure croisée permettant de faire le produit cartésien de 2 tables. En d'autres mots, permet de joindre chaque lignes d'une table avec chaque lignes d'une seconde table. Attention, le nombre de résultats est en général très élevé.
- **LEFT JOIN (ou LEFT OUTER JOIN) :** jointure externe pour retourner tous les enregistrements de la table de gauche (LEFT = gauche) même si la condition n'est pas vérifié dans l'autre table.
- **RIGHT JOIN (ou RIGHT OUTER JOIN) :** jointure externe pour retourner tous les enregistrements de la table de droite (RIGHT = droite) même si la condition n'est pas vérifié dans l'autre table.
- **FULL JOIN (ou FULL OUTER JOIN) :** jointure externe pour retourner les résultats quand la condition est vrai dans au moins une des 2 tables.
- **SELF JOIN :** permet d'effectuer une jointure d'une table avec elle-même comme si c'était une autre table.
- **NATURAL JOIN :** jointure naturelle entre 2 tables s'il y a au moins une colonne qui porte le même nom entre les 2 tables SQL
- **UNION JOIN :** jointure d'union

Exemples de jointures

INNER JOIN

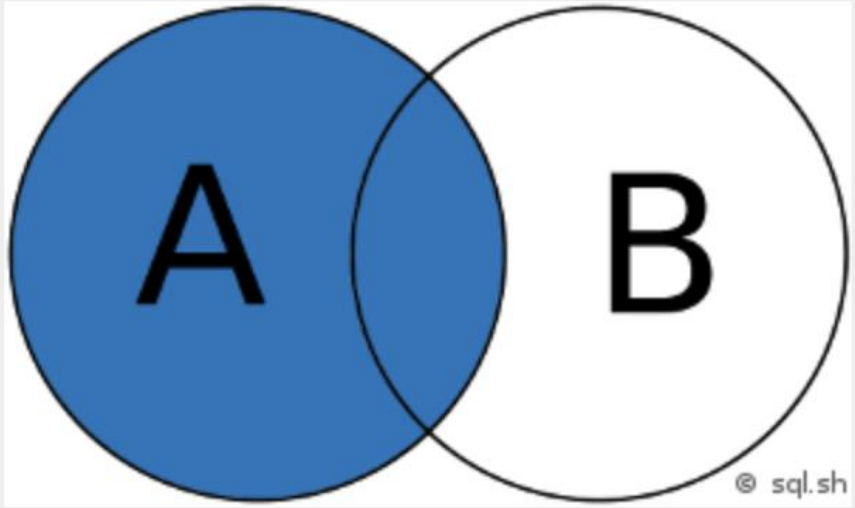


— Intersection de 2 ensembles

```
SELECT *
FROM A
INNER JOIN B ON A.key = B.key
```

Exemples de jointures

LEFT JOIN

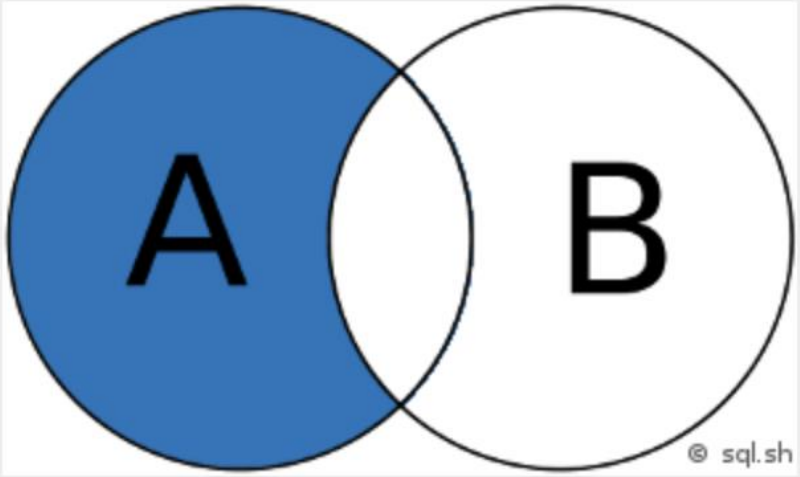


— Jointure gauche (LEFT JOINT)

```
SELECT *
FROM A
LEFT JOIN B ON A.key = B.key
```

Exemples de jointures

LEFT JOIN (sans l'intersection de B)

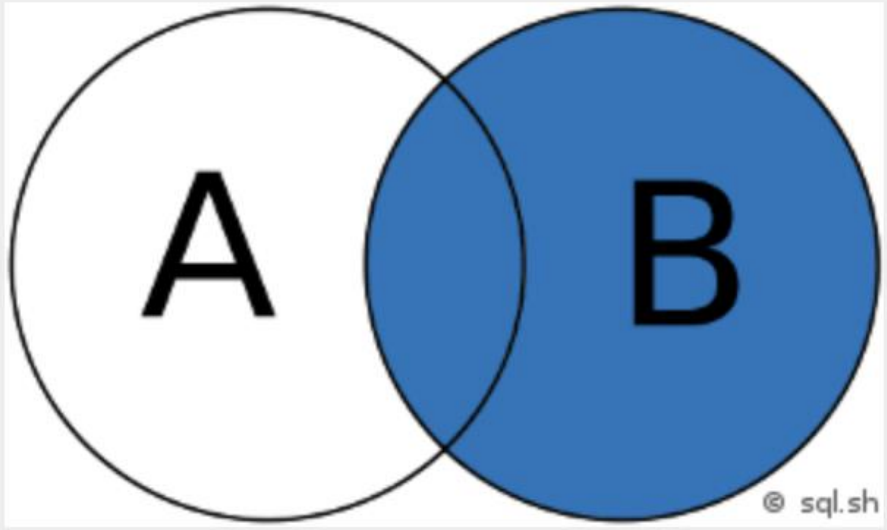


— Jointure gauche (LEFT JOINT sans l'intersection B)

```
SELECT *
FROM A
LEFT JOIN B ON A.key = B.key
WHERE B.key IS NULL
```

Exemples de jointures

RIGHT JOIN

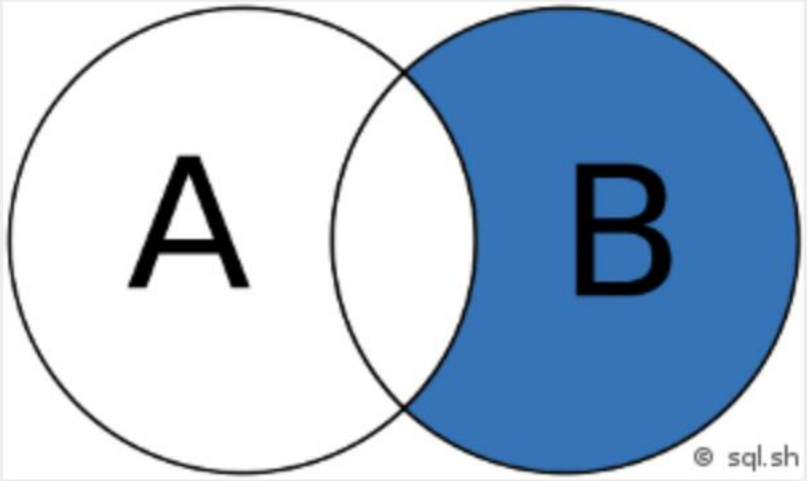


— Jointure droite (RIGHT JOINT)

```
SELECT *
FROM A
RIGHT JOIN B ON A.key = B.key
```

Exemples de jointures

RIGHT JOIN (sans l'intersection de A)

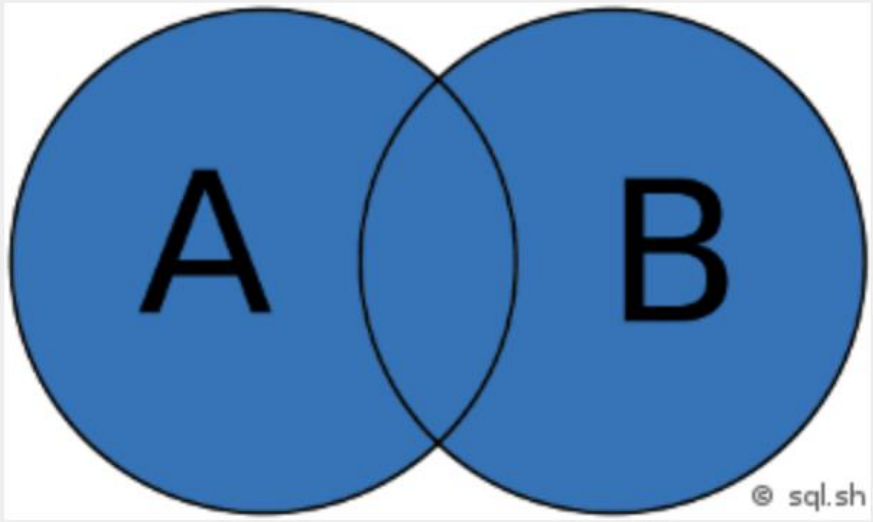


— Jointure droite (RIGHT JOINT sans l'intersection A)

```
SELECT *
FROM A
RIGHT JOIN B ON A.key = B.key
WHERE B.key IS NULL
```

Exemples de jointures

FULL JOIN

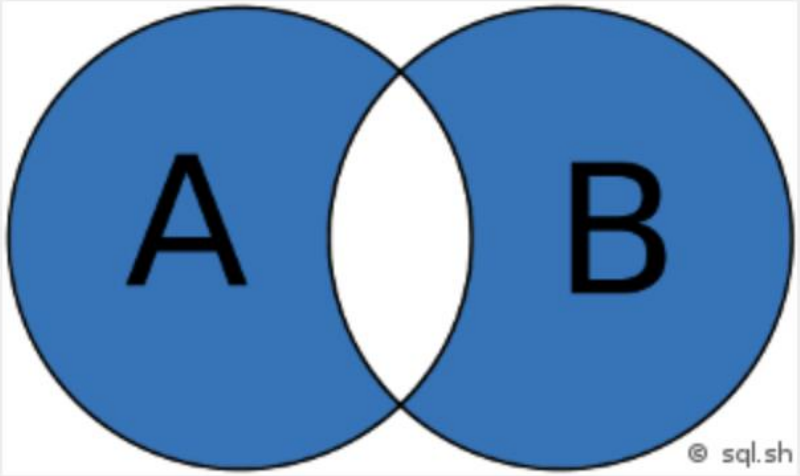


— Union de 2 ensembles

```
SELECT *
FROM A
FULL JOIN B ON A.key = B.key
```

Exemples de jointures

FULL JOIN (sans intersection)

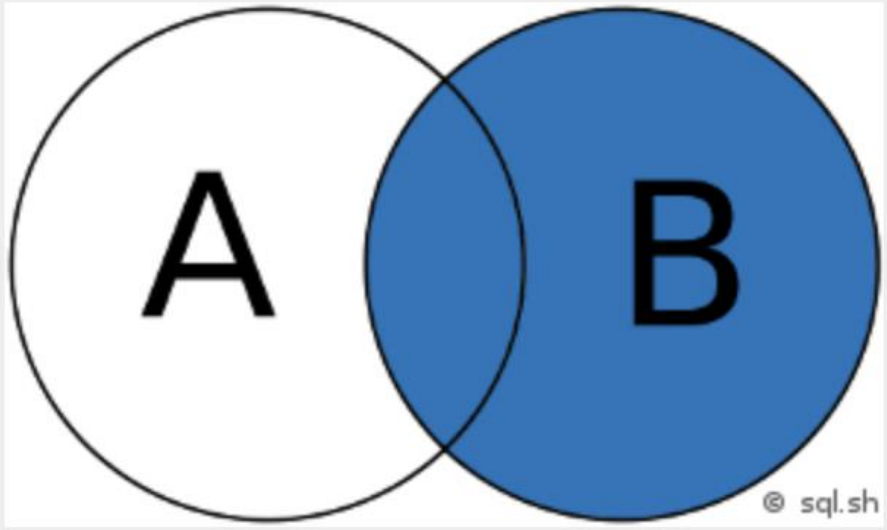


— Jointure pleine (FULL JOINT sans intersection)

```
SELECT *
FROM A
FULL JOIN B ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

Exemples de jointures

RIGHT JOIN



— Jointure droite (RIGHT JOINT)

```
SELECT *
FROM A
RIGHT JOIN B ON A.key = B.key
```

EXERCICES

- **Jointures externes, Union et Minus**
  - **52 : Sélectionner sans doublon les prénoms des directeurs et « les prénoms des techniciens du service 1 » avec un UNION.**
  - **53 : Sélectionner les numéros de services n’ayant pas d’employés sans une jointure externe.**
  - **54 : Sélectionner les services ayant au moins un employé.**
  - **54 bis : Sélectionner les numéros de services ayant des employés sans une jointure externe**

Sous interrogations avec une ou des valeurs

- **Sous interrogations comparant une valeur**

```
SELECT champ1, ...  FROM table1, ...
WHERE champA = (SELECT champ
FROM tableY, ... )
```

  - L'opérateur de comparaison peut être n'importe lequel

- **Sous interrogations comparant des valeurs**

```
SELECT champ1, ...  FROM table1, ...
WHERE (champA,champB,champC) = (SELECT champX, champY, champZ
FROM tableY, ... )
```

Sous interrogations (REQUÊTES IMBRIQUÉES)

- **avec un ensemble de valeurs**
  - **IN**
    - Sous interrogation comparant un ensemble de valeurs

```
SELECT champ1, ...  FROM table1, ...
WHERE champA IN (SELECT champX, ...
FROM tableY, ... )
```

  - **EXISTS et NOT EXISTS**
    - Permets de construire un prédicat qui est vrai si la sous interrogation qui suit ramène au moins une ligne.

```
SELECT champ1, ...  FROM table1, ...
WHERE EXISTS (SELECT 'toto' FROM tableY, ... )
```

EXERCICES

- **Interrogation sur les sous interrogations**
  - **55 : Sélectionner les employés qui travaillent à LILLE.**
  - **56 : Sélectionner les employés qui ont le même chef que DUBOIS, DUBOIS exclu.**
  - **57 : Sélectionner les employés embauchés le même jour que DUMONT.**
  - **58 : Sélectionner les nom et date d'embauche des employés plus anciens que MINET, dans l'ordre des embauches.**
  - **59 : Sélectionner le nom, le prénom, la date d'embauche des employés plus anciens que tous les employés du service N°6. (Attention MIN)**
  - **60 : Sélectionner le nom, le prénom, le revenu mensuel des employés qui gagnent plus qu'au moins un employé du service N°3, trier le résultat dans l'ordre croissant des revenus mensuels.**
  - **61 : Sélectionner les noms, le numéro de service, l'emplois et le salaires des personnes travaillant dans la même ville que HAVET.**
  - **62 : Sélectionner les employés du service 1, ayant le même emploi qu'un employé du service N°3.**
  - **63 : Sélectionner les employés du service 1 dont l'emploi n'existe pas dans le service 3.**

EXERCICES

- **Interrogation sur les sous interrogations**
  - **64 : Sélectionner nom, prénom, emploi, salaire pour les employés ayant même emploi et un salaire supérieur à celui de CARON.**
  - **65 : Sélectionner les employés du service N°1 ayant le même emploi qu'un employé du service des VENTES.**
  - **66 : Sélectionner les employés de LILLE ayant le même emploi que RICHARD, trier le résultat dans l'ordre alphabétique des noms.**
  - **67 : Sélectionner les employés dont le salaire est plus élevé que le salaire moyen de leur service (moyenne des salaires = avg(sal)), résultats triés par numéros de service.**
  - **68 : Sélectionner les employés du service INFORMATIQUE embauchés la même année qu'un employé du service VENTES.**

- **(année : to_char(embauche,'YYYY'))**
  - **69 : Sélectionner le nom, l'emploi, la ville pour les employés qui ne travaillent pas dans le même service que leur supérieur hiérarchique direct.**
  - **70 : Sélectionner le nom, le prénom, le service, le revenu des employés qui ont des subalternes, trier le résultat suivant le revenu décroissant.**

Les fonctions : Pour les numériques

ABS (n)	valeur absolue de n.
CEIL (n)	partie entière plus 1.
COS (n), SIN(n), TAN(n)	cosinus, sinus et tangente
COSH (n), SINH(n), TANH(n)	cosinus hyperbolique, sinh et tanh
EXP(n)	exponentielle de n
FLOOR (n)	partie entière.
GREATEST (x, y, z .....)	le plus grand
LEAST (x ,y ,z ,....)	le plus petit
LN(n)	logarithme népérien de n
LOG(m,n)	logarithme à base m de n
MOD (a, b)	reste de la division de l'entier a par l'entier b.
POWER (m, n)	m puissance n
ROUND (n[, m])	n arrondi à 10^{-m} ; par défaut m = 0
SIGN (n)	1 si n>0, 0 si n=0, -1 si n<0
SQRT (x)	racine carrée de x.
TRUNC (n [, d])	n tronqué à 10^{-m} ; par défaut m = 0

Les fonctions Pour les chaînes

ASCII (chaîne)	code ASCII du premier caractère de la chaîne.
CHR (n)	caractère dont n est le code ASCII.
CONCAT(ch1, ch2) ou ch1 || ch2	concaténation de ch1 et ch2
DECODE(exp,v1,r1,[v2,r2,]def)	si exp n'est pas dans (v1,v2,...) donne def, si exp=vn donnern.
INITCAP (chaîne)	caractère n°1 en majuscules.
INSTR(ch1,ch2, [n, [m]])	Position de la $m^{ième}$ occurrence de ch2 dans ch1 à partir du $n^{ième}$ caractère (par défaut n = m = 1).
LENGTH (chaîne)	longueur de la chaîne.
LOWER (chaîne)	conversion en minuscules.
LPAD(ch1, n [,ch2])	chaîne ch1 complétée sur la gauche jusqu'à une longueur total de n par ch2 (par défaut ch2 = '') ou tronquée si n < longueur ch1.
LTRIM(ch, [,set])	caractères supprimés à partir de la gauche de ch tant qu'ils appartiennent à l'ensemble de caractères de set (par défaut set = '').
RPAD(ch1, n [,ch2])	chaîne ch1 complétée sur la droite jusqu'à une longueur total de n par ch2 (par défaut ch2 = '') ou tronquée si n <longueur ch1.
RTRIM(ch, [,set])	caractères supprimés à partir de la droite de ch tant qu'ils appartiennent à l'ensemble de caractères de set (par défaut set = '').
SOUNDEX (chaîne)	calcul de la valeur phonétique
SUBSTR (chaîne, m[,n])	extraire une sous chaîne à partir de m, de longueur n (par défaut jusqu'à la fin de la chaîne).
TO_CHAR (expression, format)	convertir un nombre ou une date en chaîne de caractères.
TO_NUMBER (chaîne)	convertir une chaîne en nombre.
TRANSLATE(ch,from,to)	remplacement dans ch de chaque caractère de from par le caractère occupant la même position dans to.
TRUNC (expression, précision)	Tronquer.
UPPER (chaîne)	conversion en majuscules.

Les fonctions Pour les dates

ADD_MONTHS (date, nombre)	ajoute ou soustrait des mois à une date.
LAST_DAY(date)	dernier jour du mois de la date.
MONTHS_BETWEEN (date1, date2)	nombre de mois entre date1 et date2
NEXT_DAY(date, nom jour)	date du prochain nom de jour spécifié.
SYSDATE	est la date système du jour
TO_DATE (chaîne, format)	convertir une chaîne de caractères en date

EXERCICES : Prenez vos fonctions

- **71 :Sélectionner le nom, l'emploi, le revenu mensuel (nommé Revenu) avec deux décimales pour tous les employés, dans l'ordre des revenus décroissants.**
- **72 : Sélectionner le nom, le salaire, commission des employés dont la commission représente plus que le double du salaire.**
- **73 : Sélectionner nom, prénom, emploi, le pourcentage de commission (deux décimales) par rapport au revenu mensuel (renommé "% Commissions") , pour tous les vendeurs dans l'ordre décroissant de ce pourcentage.**
- **74 : Sélectionner le nom, l'emploi, le service et le revenu annuel (à l'euro près) de tous les vendeurs.**
- **75 : Sélectionner nom, prénom, emploi, salaire,commissions, revenu mensuel pour les employés du service dont le numéro sera saisi au clavier.**
- **76 : Idem en remplaçant les noms des colonnes : SAL par SALAIRE, COMM par COMMISSIONS, SAL+NVL(COMM,0) par GAIN_MENSUEL.**
- **77 : Idem en remplaçant GAIN_ MENSUEL par GAIN MENSUEL**
- **78 : Afficher le nom, l'emploi, les salaires journalier et horaire pour les employés du service dont le numéro sera saisi au clavier (22 jours/mois et 8 heures/jour), sans arrondi, arrondi au centime près.**
- **79 : Idem sans arrondir mais en tronquant.**

EXERCICES : Prenez vos fonctions

- **80 : Concaténer les colonnes Service et Ville en les reliant par " ----> ", les premières lettres des noms de villes doivent se trouver sur une même verticale.**
- **81 : Sélectionner nom, emploi pour les employés en ajoutant une colonne "CODE EMPLOI", trier le résultat sur ce code.**

PRESIDENT	1
DIRECTEUR	2
COMPTABLE	3
VENDEUR	4
TECHNICIEN	5

0 signifie que le code emploi n'existe pas.

- **82 : Sélectionner les employés en remplaçant les noms par '*****' dans le service n°1, trier le résultat suivant le N° de service.**
- **83 : Sélectionner les noms des services en affichant que les 5 premiers caractères.**
- **84 : Sélectionner les employés embauchés en 1988.**
- **85 : Sélectionner les noms des employés sur 3 colonnes la première en majuscules, la seconde avec l'initiale en majuscule et le reste en minuscules, la troisième en minuscules.**
- **86 : Sélectionner les positions des premiers M et E dans les noms des employés**
- **87 : Afficher le nombre de lettres qui sert à écrire le nom de chaque service.**
- **88 : Tracer un Histogramme des salaires avec nom, emploi, salaire triés dans l'ordre décroissant (max de l'histogramme avec 30 caractères).**

EXERCICES : Le choix des dates

- **89 : Sélectionner nom, emploi, date d'embauche des employés d'un service dont le numéro sera saisi au clavier.**
- **90 : Même chose en écrivant la colonne embauche sous la forme 'dd-mm-yy', renommée embauche.**
- **91 : Même chose en écrivant la colonne embauche sous la forme 'fm day dd month yyyy'**
- **92 : Même chose en écrivant la colonne embauche sous la forme 'day dd month yyyy'**
- **93 : Même chose en écrivant la colonne embauche sous la forme 'dy mon yy'**
- **94 : Même chose en écrivant la colonne embauche sous la forme 'fm Day dd Month yyyy'**
- **95 : Sélectionner les employés avec leur ancienneté en jours dans l'entreprise.**
- **96 : Sélectionner les employés avec leur ancienneté en mois dans l'entreprise.**
- **97 : Sélectionner toutes les dates d'embauche majorées de 12 ans.**
- **98 : Sélectionner les employés ayant plus de 12 ans d'ancienneté.**

- **Plus amusant …**
- **99 : Depuis combien de jours êtes-vous nés ?**
- **100 : Quand allez-vous avoir 10.000 jours ? (Pour les plus vieux, mettez la question au passé !)**

Les fonctions de groupe

AVG (col)	la moyenne.
SUM (col)	la somme.
MIN (col)	la plus petite valeur.
MAX (col)	la plus grande valeur.
COUNT (*)	le nombre de lignes satisfaisants à la condition WHERE qui suit.
VARIANCE (col)	la variance.
STDDEV (col)	l'écart type.

- **Exemples :**
  - Salaire moyen de tous les employés
  - Nombre d’employés
  - Nombre de prénoms différents

Les fonctions de groupe : L'utilisation : GROUP BY, HAVING

- **GROUP BY**
  - Sert à regrouper ensemble les colonnes n'étant pas dans des fonctions de regroupement
  - En général, il contient la liste des champs sans fonction de regroupement
- **HAVING**
  - Porte une/des conditions sur une/des fonctions de groupe

```
SELECT …
FROM …
[WHERE prédicat]
[GROUP BY nom_coli[,nom_colj.....]  [HAVING prédicat] ]
[ORDER BY nom_colp [DESC] [,nom_colq [ASC] ...];
```

- **Exemple**
  - Salaire moyen par service
  - Service dont le salaire moyen est supérieur à 19000 €

EXERCICES : Restez groupir !

- **101 : Afficher la moyenne des revenus (avec commission) des vendeurs.**
- **102 : Paramétrer la requête qui précède sur l'emploi.**
- **103 : Afficher la somme des salaires et la somme des commissions des vendeurs.**
- **104 : Afficher le plus haut salaire, le plus bas salaire, la différence entre les deux.**
- **105 : Compter les employés embauchés chaque trimestre.**
- **106 : Afficher le nombre de lettres du service dont le nom est le plus court.**
- **107 : Sélectionner nom, emploi, revenu mensuel de l'employé qui gagne le plus.**
- **108 : Déterminer le nombre d'employés du service 3 qui reçoivent éventuellement une commission.**
- **109 : Déterminer le nombre d'emploi différents du service N°1.**
- **110 : Déterminer le nombre d'employés du service N°3.**

Le Langage de Manipulation de Données
Mettre à jour les données

- **UPDATE nom table**

```
  SET col1 = { expr1 |(select...)}
    [ col2 = { expr2 |(select...)} ... ]  [WHERE
prédicat];
```

- **Les champs col1, col2 ....sont mis à jour dans toutes les lignes vérifiant le prédicat.**
- **En l'absence de clause WHERE toutes les lignes sont mises à jours.**
- **dans expr1 on peut utiliser col1, col2..**

Le Langage de Manipulation de Données
Insérer des données

- **Pour insérer une ligne**
  - `INSERT INTO nom_table [(col1[,col2[,.......)] VALUES (val1,[val2,.....]);`
  - La liste des noms de colonnes n'est pas obligatoire à condition de ranger les valeurs dans l'ordre de création des colonnes de la table.
  - Les colonnes non spécifiées auront la valeur NULL.

- **Pour insérer plusieurs lignes (postgresql)**
  - `INSERT INTO nom_table [(col1,....)] SELECT .....;`
- **Pour insérer plusieurs lignes (oracle)**
  - `INSERT INTO nom_table [(col1,....)] Values (requêtes);`
  - Le select ne doit pas contenir de clause ORDER BY.

Le Langage de Manipulation de Données
Supprimer des données

- **`DELETE FROM nom_table`**
  `[WHERE prédicat];`

- **Toutes les lignes qui vérifient le prédicat de la clause WHERE sont détruites, sans clause WHERE toutes les lignes sont détruites il ne reste que la définition de la table.**

Le Langage de Manipulation de Données
Notion de transaction

- Une transaction débute lors du LOGIN, d'un COMMIT, d'un ROLLBACK.
- La commande `COMMIT`: toutes les modifications faites depuis le début de la transaction sont écrites dans la base et deviennent visibles aux autres utilisateurs, une nouvelle transaction commence.
- La commande `ROLLBACK`: toutes les modifications faites depuis le début de la transaction sont annulées, une nouvelle session commence.
- Point de sauvegarde
  - la commande SAVEPOINT savepoint_nom; insère un SAVEPOINT dans la transaction en cours.
  - ROLLBACK savepoint_nom; annule toutes les modifications effectuées depuis la création du SAVEPOINT savepoint_nom et efface le SAVEPOINT savepoint_nom.
  - COMMIT et ROLLBACK sans paramétres effacent tous les SAVEPOINT

EXERCICES : Pratiquons la manipulation

- Pour ces exercices, vous devez vous connecter sous votre identifiant et faire une copie des tables EMP et SERV.
- 121 : Augmenter de 10% ceux qui ont une salaire inférieur au salaire moyen. Valider.
- 122 : Insérez vous comme nouvel employé embauché aujourd'hui au salaire que vous désirez. Valider.
- 123 : Effacer les employés ayant le métier de SECRETAIRE. Valider.
- 124 : Insérer le salarié dont le nom est MOYEN, prénom Toto, no 1010, embauché le 12/12/99, supérieur 1000, pas de comm, service no 1, salaire vaut le salaire moyen des employés. Valider.
- 125 : Supprimer tous les employés ayant un A dans leur nom. Faire un SELECT sur votre table pour vérifier cela. Annuler les modifications et vérifier que cette action s'est bien déroulée.
- /*126 : Les verrous. Supprimer l'employé créé à l'exercice 122 de votre voisin. Ne pas valider. Vérifiez tous les deux le contenu de la table. Demander à votre voisin d'augmenter son propre salaire de 1000 €. Valider la suppression. Chacun vérifie l'action. Refaire l'exercice en échangeant les rôles.*/

Le langage de définition de données Créer une table

- Deux façons de créer une table
- Pour créer une table à partir de rien
- `CREATE TABLE nom table (col1 type1 [NOT NULL], col2 type2 [NOT NULL],.....)`

`[TABLE SPACE nomtbs]`

- Pour créer une table à partir d'une autre table
- `CREATE TABLE nom table [AS select .....];`

Le langage de définition de données Créer une table (oracle)

- EX: Création de la table EMP sous SQL.
-
```
create table emp
 ( noemp     number(4) not null,
   nom varchar2(20),
   prenom    varchar2(20),
    emploi    varchar2(20),
    supnumber(4),
    embauche date,
     sal number(9,2),
     comm number(9,2),
     Noserv number(2))
```

Le langage de définition de données
Modifier une table

- **Ajouter une colonne.**
  - **`ALTER TABLE nom_table  ADD (col1 type1,`**
    - **`col2 type2, ..);`**
  - La nouvelle colonne est placée à droite des anciennes.
  - Tous les champs de la nouvelle colonne sont à NULL.
- **Modifier une colonne.**
  - **`ALTER TABLE nom_table  MODIFY (nom_col nouv_type );`**
  - le nouveau type doit être compatible avec l'ancien, on peut augmenter la taille maximale d'une colonne VARCHAR2(n) par exemple.
  - on peut déclarer NOT NULL un colonne qui ne contient pas de valeur NULL.
  - Sur postgresql (https://www.postgresql.org/docs/9.1/static/sql-altertable.htm)

Le langage de définition de données
Renomme et supprimer une table

- la commande DROP TABLE
  - Permet de supprimer la table.
  - Aucune récupération possible, la place est toute de suite réutilisée
- la commande RENAME
  - Permet de renommer une table
  - EX: **`RENAME serv to services;`**
  - Permet de renommer une colonne
  - EX : **`ALTER TABLE EMP`**

    `RENAME COLUMN EMPLOI TO EMPLOI1`

EXERCICES : Créons !

- 127 : Créer les tables EMP et SERV comme copie des tables EMP et SERV.
- 128 : Vérifier que la table PROJ n'existe pas.
- 129 : Créer une table PROJ avec les colonnes suivantes:
  - ➢ numéro de projet (noproj), type numérique 3 chiffres, doit contenir une valeur.
  - ➢ nom de projet (nomproj), type caractère, longueur = 10
  - ➢ budget du projet (budget), type numérique, 6 chiffres significatifs et 2 décimales.
  - Vérifier l'existence de la table PROJ.
  - Faire afficher la description de la table PROJ.
- 130 : Insérer trois lignes de données dans la table PROJ:
  - numéros des projets = 101, 102, 103
  - noms des projets = alpha, beta, gamma
  - budgets = 250000, 175000, 950000
  - Afficher le contenu de la table PROJ.
  - Valider les insertions faites dans la table PROJ.

EXERCICES : Créons !

- 131 :
  - Modifier la table PROJ en donnant un budget de 1.500.000 Euros au projet 103.
  - Modifier la colonne budget afin d'accepter des projets jusque 10.000.000.000 d'Euros
  - Retenter la modification demandée 2 lignes au dessus.
- 132 :
  - Ajouter une colonne NOPROJ (type numérique) à la table EMP.
  - Afficher le contenu de la table EMP.
- 133 : Affecter les employés du service 2 et les directeurs au projet 101.
- 134 : Affecter les employés dont le numéro est supérieur à 1350 au projet 102, sauf ceux qui sont déjà affectés à un projet.
- 135 : Affecter les employés n'ayant pas de projet au projet 103.
- 136 : Sélectionner les noms d'employés avec le nom de leur projet et le nom de leur service.
- 137 : La liste des tables de l'utilisateur se trouve dans la table système user_tables. Ecrire une requête qui affiche les noms de toutes vos tables.

Le langage de définition de données Les vues : la création

- Le résultat d'une commande select est une table virtuelle, cette table créée dynamiquement peut être vue comme une table réelle par les utilisateurs par les biais d'une vue.
- `CREATE VIEW nom_vue [(col1, AS SELECT col2..)] AS SELECT .....;`
- Il n'est pas obligatoire de préciser les noms de colonnes, par défaut les colonnes de la vue ont les mêmes noms que les colonnes du select qui a défini cette vue, les colonnes résultat d'une expression doivent êtres renommées dans le select.
- EX: créer la vue SUBA de la table employée limitée au non directeurs.

```
create view SUBA
    as select * from emp
    where emploi <> 'DIRECTEUR';
```

- Généralement une vue est créée pour empêcher l'accès à certaines colonnes.

Le langage de définition de données Les vues : l'utilisation

- Elle s'interroge comme une table

- Pour détruire une vue
  - `DROP VIEW nom_vue;`

- Pour renommer une vue
  - `RENAME ancien_nom TO nouveau_nom;`

EXERCICES : Ajustez votre vue

- 138 : Créer la vue EMP1 de la table EMP contenant les colonnes: numéro d'employé, nom et emploi et limitée aux employés du service numéro 1,
- 139 : Sélectionner cette vue EMP1 entièrement,
- 140 : Sélectionner les vendeurs de la vue emp1.
- 141 : Modifier la table EMP en transformant l'emploi de NYS en comptable.
- 142 : Sélectionner la vue EMP1.

Le langage de définition de données Les contraintes

- **Il y a trois types de contraintes :**
  - Déclaration de clef primaire
    - `constraint pk_serv primary key (noserv)`
  - Contrainte d'appartenance à un domaine
    - `constraint ck_sal check (SAL between 1 and 9999999)`
    - `constraint ck_comm check (COMM >= 0)`
  - Contrainte référentielle (Clef étrangère)
    - `constraint fk1_emp foreign key (SUP) references EMP(NOEMP)`
    - `constraint fk2_emp foreign key (NOSERV) references SERV(NOSERV)`
- **La clef peut être multiple, les champs participant à la clef sont alors séparés par des virgules.**

Le langage de définition de données Les contraintes

- **La déclaration des contraintes se placent :**
  - Les contraintes de domaine : après la déclaration du champ
    - **create table serv**

```
(noserv number(4) not null,

constraint ck_noser check(NOSERV between 1 and 9999), .............);
```

  - Les autres contraintes :
    - Soit à la fin du create table :
    - **Create table emp (.........,**

```
constraint pk_serv primary key (noserv),
constraint fk1_emp foreign key (SUP) references EMP(NOEMP),
constraint fk2_emp foreign key (NOSERV) references SERV(NOSERV));
```

  - Soit dans un alter table :
    - **Alter table emp**

```
add constraint fk1_emp foreign key (SUP)  references
EMP(NOEMP);
```

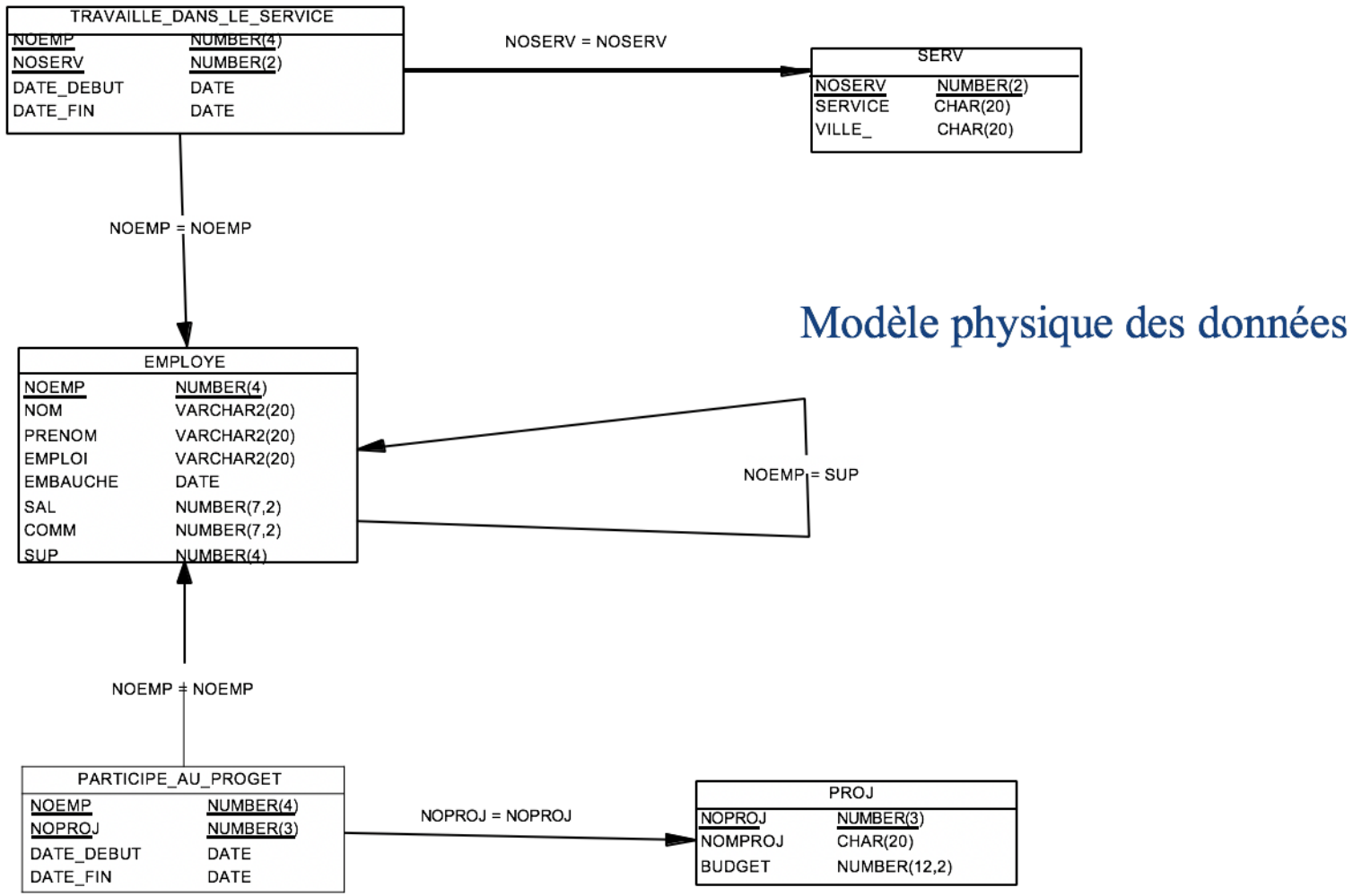
- **Généralement la contrainte de clef primaire se trouve dans le create table, celle de clef étrangère dans un alter table, les "create table" sont regroupés en début de script.**

Exemple / Exercice : Création d'une base de données

- **On se propose de définir correctement la mini base de donnée constituée des entités qui suivent :**
- **Liste des entités**

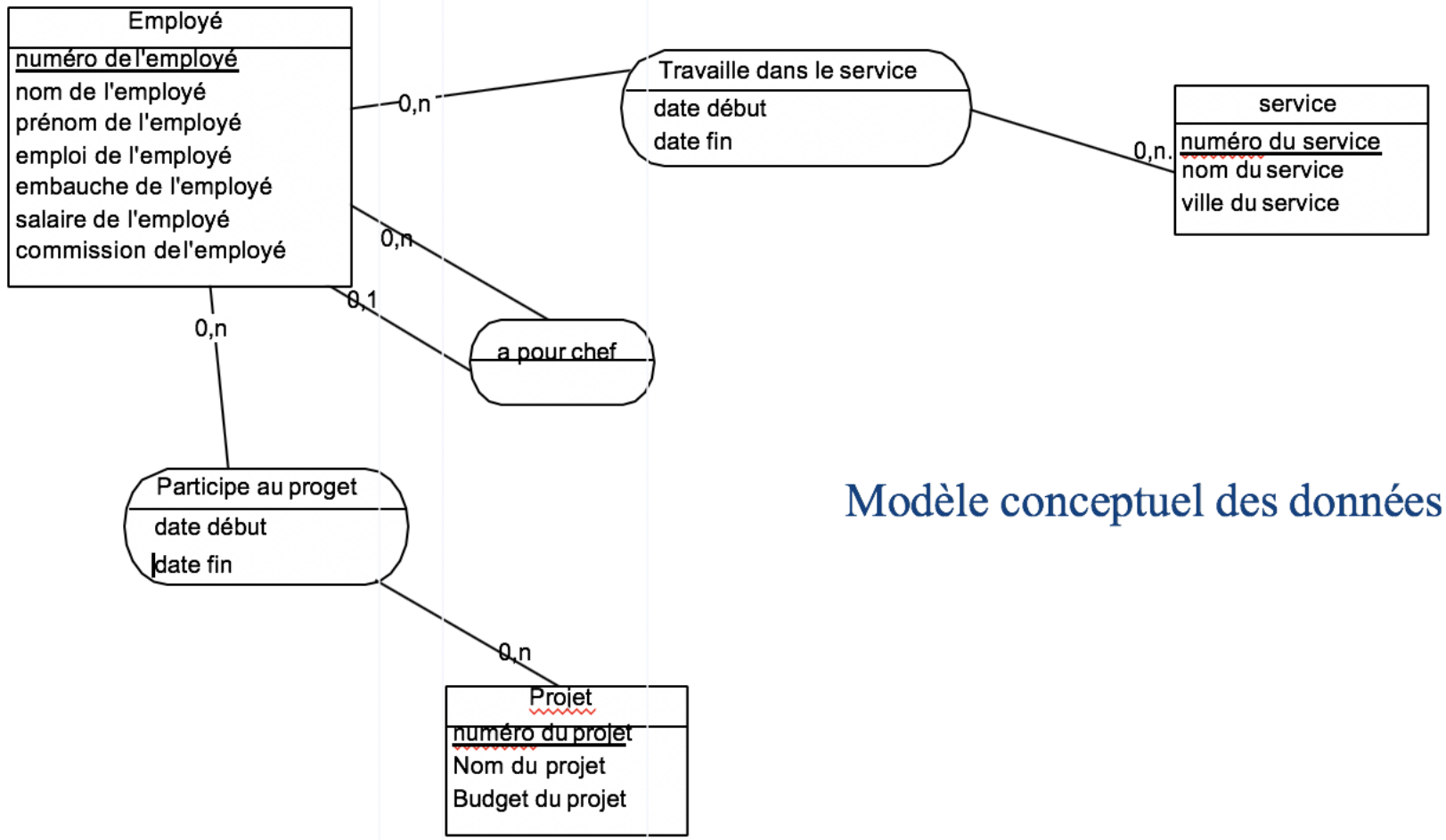
- **Règles de gestion**
  - Un employé peut travailler pour plusieurs services,
  - Un employé peut travailler sur plusieurs projets,
  - Un employé a au plus un chef.
- **Questions**
  - Dessiner le modèle conceptuel de données
  - Dessiner le modèle physique de données
  - Ecrire le script de création de la base de données.
  - Utiliser les données de sysadmin.emp pour documenter la nouvelle base.

Exemple / Exercice
Création d'une base de données



Modèle physique des données

Exemple / Exercice
Création d'une base de données



Modèle conceptuel des données

Exercices de contraintes

- Contrainte d'unicité de la table proj (NOPROJ)
- Contrainte clé étrangère table EMP (numéro de service)
- Contrainte de domaine table PROJ (budget > 20 000)
- Contrainte clé étrangère table EMP (numéro de supérieur)
- Contrainte de domaine table EMP (numéro de supérieur entre 0 et 9999
- Contrainte de domaine table SERV (numéro de service < 10)

Le langage de définition de données : Les index

- **Quand on définit une table, on ne défini pas de colonne clé d'accès, le prédicat de la clause where peut porter sur toute colonne, la recherche sur cette colonne est alors séquentielle.**
- **Quand l'accès par une colonne est fréquent, on a intérêt pour accélérer la recherche à créer un index sur cette colonne.**
- **Un index sur une colonne est une table qui comporte deux champs, le premier reprend la valeur rangée dans la colonne, le second pointe la ligne de la table où est rangée la valeur en question, cette table étant classée (dynamiquement) suivant le premier champ, Oracle fait des recherches dichotomiques sur les colonnes indexées.**
- **Création d'un index**
  - **`CREATE INDEX nom_index ON nom_table (col1[,col2,....]);`**
  - Un index porte sur une ou plusieurs colonnes.
  - On peut créer plusieurs index indépendants sur la même table.
  - L'utilisation et la mise a jours des index est automatique dans oracle.
- **L'option UNIQUE indique que chaque valeur de l'index doit être unique dans la table.**
  - **`CREATE UNIQUE INDEX numemp ON emp (noemp);`**
  - Pour cela il faut que tout n° soit unique dans les lignes existantes.
  - Par la suite l'insertion d'un N° existant dans la table sera interdite.
  - on peut dire alors que noemp est une clef d'accès.
- **Détruire un index**
  - **`DROP INDEX nom_index;`**