Ascent FORMATION

Juin 2026

Le langage SQL

Lot 4 – Gestion et analyse des données – SGBD – Référence 4-069

Elodie Montmasson – contact@unidia.fr

Le langage SQL

Introduction

Modèle relationnel

Modèle relationnel

Manière de modéliser les relations existantes entre plusieurs informations et de les ordonner entre elles.

Données organisées sous forme de :

- Tables
- Lignes
- Colonnes
- Relations

Modèle relationnel - tables

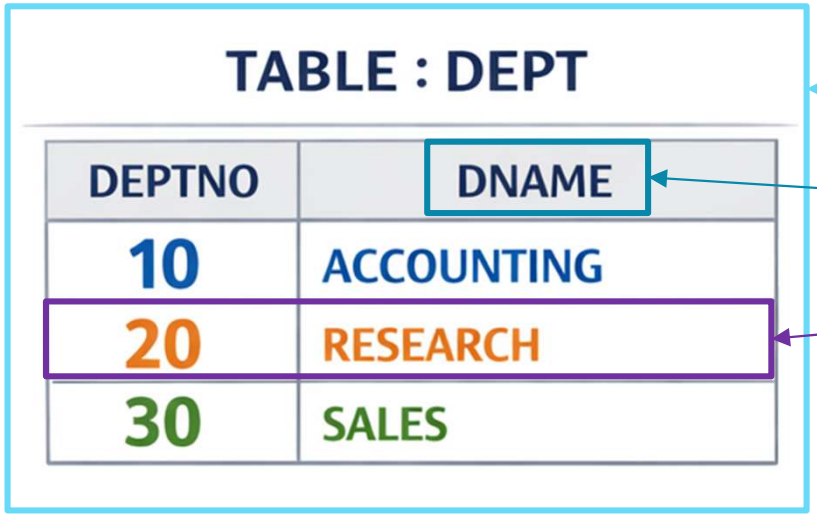


Table : stocke les données

Colonne : représente une caractéristique

Ligne : représente un enregistrement

Modèle relationnel

- Une table se représente de la manière suivante :
  NomTable (clé primaire, attribut1, …, attribut n, clé étrangère) .
- **Clé primaire :** un attribut ou un groupe d'attributs qui permet d'identifier de façon unique chaque ligne d'une table.
- **Attribut** : une colonne dans une table
- **Clé étrangère :** un attribut ou un groupe d'attributs dont la valeur fait référence à la clé primaire d'une autre table.

Elle permet de **lier les tables entre elles**.

Modèle relationnel - relations

TABLE : DEPT

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES

CLÉ PRIMAIRE *DEPT.DEPTNO*

TABLE : EMP

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7782	CLARK	MANAGER	10
7934	MILLER	CLERK	10
7566	JONES	MANAGER	20
7788	SCOTT	ANALYST	20
7902	FORD	ANALYST	20
7698	BLAKE	MANAGER	30
7499	ALLEN	SALESMAN	30

CLÉ ÉTRANGÈRE *EMP.DEPTNO*

Introduction

Principaux objets d’une base

Principaux objets d'une base

- **Tables**

  Une table est **l'objet principal d'une base de données**.

- **Vues**

  Une vue est **une table virtuelle basée sur une requête**.

  Elle ne stocke pas les données, elle les **affiche** à partir d'une ou plusieurs tables.
  Elle permet de **simplifier l'accès aux données** ou de **restreindre ce qu'on voit**.

Principaux objets d'une base

- **Index**

  Un index est un objet qui permet **d'accélérer les recherches** dans une table.

  Il fonctionne comme un **sommaire ou index de livre**.
  Il permet de retrouver les données plus rapidement.

- **Contraintes**

  Une contrainte est une règle appliquée sur une table pour **garantir la validité des données**.

  Elle empêche les erreurs (doublons, valeurs interdites, etc.).
  Exemples : clé primaire, clé étrangère, NOT NULL, UNIQUE.

Introduction

Environnement de formation

Environnement de formation

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17/12/80	800		20
7499	ALLEN	SALESMAN	7698	20/02/81	1600	300	30
7521	WARD	SALESMAN	7698	22/02/81	1250	500	30
7566	JONES	MANAGER	7839	02/04/81	2975		20
7654	MARTIN	SALESMAN	7698	28/09/81	1250	1400	30
7698	BLAKE	MANAGER	7899	01/05/81	2850		30
7782	CLARK	MANAGER		09/06/81	2450		10
7788	SCOTT	ANALYST	7566	09/12/82	3000		20
7839	KING	PRESIDENT		17/11/81	5000	0	10
7844	TURNER	SALESMAN	7698	28/09/81	1500		30
7876	ADAMS	CLERK	7788	12/01/83	1100		20
7900	JAMES	CLERK	7698	03/12/81	950		30
7902	FORD	ANALYST	7566	03/12/81			
7934	MILLER	CLERK	7782	23/01/82	10		

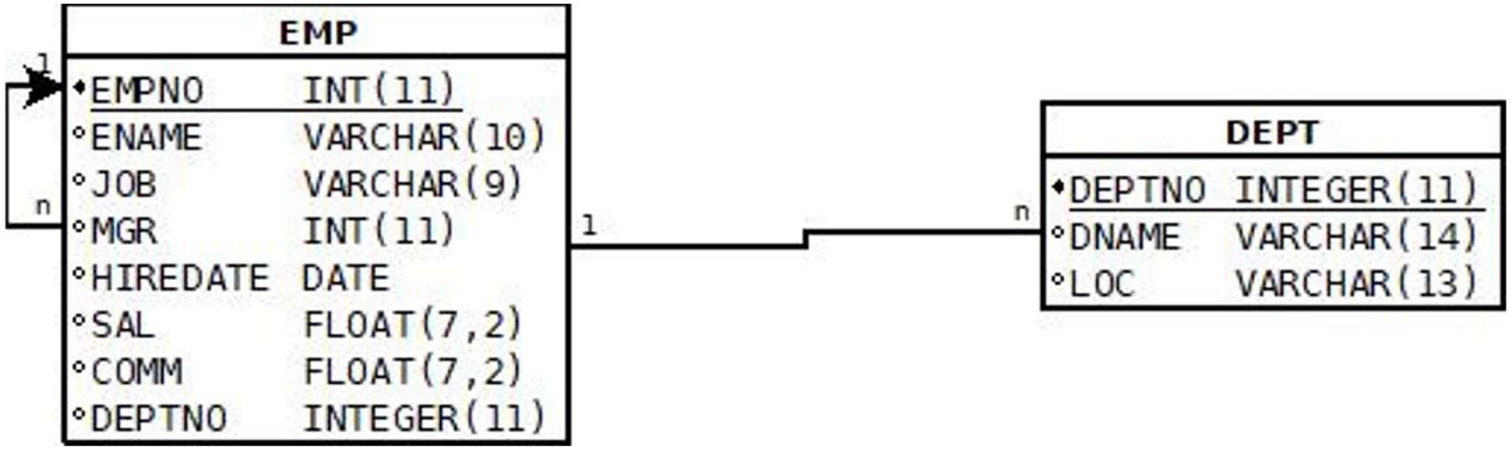
Table emp

Table dept

DEPTNO	DNAME	LOC
10	Accounting	NEW YORK
20	Research	DALLAS
30	Sales	CHICAGO
40	Operations	BOSTON
40	Miller	BOSTON

Environnement de formation

- Les exemples et exercices suivants seront illustrés au moyen d'une base de données composée de 2 tables dept et emp.
- Les liaisons entre ces tables sont présentées sur la figure ci-dessous, qui met en évidence les clés primaires et étrangères :



Définition des données

Introduction

Ordres de définition de structure

- **Quatre ordres de définition de structure :**
  - CREATE
  - ALTER
  - DROP
  - TRUNCATE

Définition des données

L'ordre create

L’ordre CREATE

- **Syntaxe de création d’une table :**

```
CREATE TABLE nom_de_la_table (
    colonne1 type_donnees,
    colonne2 type_donnees
    CONSTRAINT pk_nom_de_la_table_colonne1  PRIMARY KEY (colonne1),
    CONSTRAINT fk_nom_de_la_table_colonne2  FOREIGN KEY (colonne2),
    REFERENCES nom_de_la_table_référencée(nom_de_la_colonne_de_la_table_référencée));
```

L'ordre CREATE

- **Type de données :**
  - Numérique

Concept	Oracle	PostgreSQL	Cas d'utilisation
Entier petit	NUMBER(p)	SMALLINT	Âge, note sur 20, nombre d'enfants
Entier standard	NUMBER(p) ou INTEGER	INTEGER / INT	ID client, numéro de commande
Grand entier	NUMBER(p)	BIGINT	Nombre de vues YouTube, transactions bancaires
Décimal précis	NUMBER(p,s)	NUMERIC(p,s) / DECIMAL(p,s)	Prix produit, montant facture, taux TVA
Flottant	FLOAT(p) / BINARY_FLOAT	REAL	Coordonnées GPS approximatives, mesures scientifiques
Auto-incrément	SEQUENCE + NUMBER	SERIAL / BIGSERIAL	Clé primaire auto-générée

L'ordre CREATE

- **Type de données :**
  - Chaînes de caractères

Concept	Oracle	PostgreSQL	Cas d'utilisation
Chaîne fixe	`CHAR(n)`	`CHAR(n)`	Code pays (FR, US), code postal, code ISO
Chaîne variable	`VARCHAR2(n)`	`VARCHAR(n)`	Nom, prénom, adresse email
Texte illimité	`CLOB`	`TEXT`	Description produit, article de blog, commentaire
Texte Unicode illimité	`NCLOB`	`TEXT`	Articles en langues asiatiques, contenu multilingue

L'ordre CREATE

- **Type de données :**
  - Date

Concept	Oracle	PostgreSQL	Cas d'utilisation
Date seule	`DATE` *(inclut heure)*	`DATE`	Date de naissance, date d'embauche
Date + Heure	`TIMESTAMP`	`TIMESTAMP`	Date/heure de création d'un compte
Timestamp avec timezone	`TIMESTAMP WITH TIME ZONE`	`TIMESTAMPTZ`	Réservation d'hôtel internationale
Heure seule	*(non disponible)*	`TIME`	Horaires d'ouverture d'un magasin
Intervalle	`INTERVAL YEAR TO MONTH`	`INTERVAL`	Durée d'un abonnement, délai de livraison

L'ordre CREATE

- **Type de données :**
  - Spéciaux

Concept	Oracle	PostgreSQL	Cas d'utilisation
Booléen	`NUMBER(1)` par convention	`BOOLEAN`	Compte actif/inactif, case à cocher
UUID	*(non natif)*	`UUID`	Identifiant unique global (token, lien de partage)
Tableau	*(non disponible)*	`type[]`	Liste de tags, liste de numéros de téléphone
Énumération	*(non disponible)*	`ENUM`	Statut commande, rôle utilisateur, civilité
Géométrie	`SDO_GEOMETRY`	`POINT`, `LINE`, `POLYGON`	Localisation magasin, zone de livraison
Plage de valeurs	*(non disponible)*	`RANGE`	Période de validité, plage de prix

L’ordre CREATE

- **Options disponibles pour chaque colonne**
  - NOT NULL : empêche d’enregistrer une valeur nulle pour une colonne.
  - DEFAULT : attribuer une valeur par défaut si aucune donnée n’est indiquée pour cette colonne lors de l’ajout d’une ligne dans la table.

L’ordre CREATE

- **Clés primaires**
  - CONSTRAINT pk_nom_de_la_table_colonne1 PRIMARY KEY (colonne1)
- **Règles**
  - toute table doit disposer d'une clé primaire,
  - la norme SQL impose que toutes les colonnes d'une clé primaire soient obligatoires (la colonne est forcée à NOT NULL),
  - unicité des valeurs de la clé : les index uniques sont créés automatiquement à partir de la clé primaire.

L’ordre CREATE

- Clés étrangères
  - CONSTRAINT fk_nom_de_la_table_colonne2 FOREIGN KEY (colonne2)
  - REFERENCES nom_de_la_table_référencée(nom_de_la_colonne_de_la_table_référencée));
- Règles :
  - une colonne ou un groupe de colonnes qui référencent la clé primaire ou une des clés uniques d'une autre table
  - la clé étrangère doit exister dans la table référencée

Définition des données

L'ordre create – Exercice

Exercice 1

- Créer la table dept et emp avec les clés primaires sans les clés étrangères.

Définition des données

L'ordre alter table

L’ordre ALTER TABLE

- Permet des modifications sur la structure d'une table existante :
  - ajout/modification/suppression de colonnes
  - Ajout et suppression de clés primaires et étrangères et de références
- Ajout d’une colonne :
  - ALTER TABLE nom_de_table ADD nom_colonne type_colonne;
- Ajout d’une clé primaire :
  - ALTER TABLE nom_de_table ADD PRIMARY KEY (nom_colonne)

L’ordre ALTER TABLE

- Ajout d'un index
  - ALTER TABLE ADD KEY FK_nom_table_nom_colonne (nom_colonne);
- Ajout d’une clé étrangère
  - ALTER TABLE nom_de_table
  - ADD CONSTRAINT FK_nom_table_nom_colonne
  - FOREIGN KEY (nom_colonne)
  - REFERENCES autre_table(colonne);

L'ordre ALTER TABLE

- Modification d'une colonne
  - Oracle : ALTER TABLE nom_table MODIFY nom_colonne type_donnees;
  - PostgreSQL : ALTER TABLE nom_table ALTER COLUMN nom_colonne TYPE type_donnees;
- Suppression d'une colonne
  - Oracle : ALTER TABLE nom_table DROP nom_colonne;
  - PostgreSQL : ALTER TABLE nom_table DROP COLUMN nom_colonne;
- Suppression d'une clé primaire
  - Oracle : ALTER TABLE nom_table DROP PRIMARY KEY;
  - PostgreSQL : ALTER TABLE nom_table DROP CONSTRAINT nom_pk;
- Suppression d'une clé étrangère (la référence se supprime automatiquement)
  - ALTER TABLE nom_table DROP INDEX nom_clé_étrangère;

Définition des données

L'ordre alter table – Exercices

Exercice 2

- Créer les clés étrangères de la table emp.

Environnement de formation

Script à exécuter après la création des tables

```
INSERT INTO dept (DEPTNO, DNAME, LOC) VALUES
(10, 'ACCOUNTING', 'NEW YORK'),
(20, 'RESEARCH', 'DALLAS'),
(30, 'SALES', 'CHICAGO'),
(40, 'OPERATIONS', 'BOSTON');
INSERT INTO emp (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES
(7839, 'KING',   'PRESIDENT', NULL, '1981-11-17', 5000.00, NULL, 10),
(7566, 'JONES',  'MANAGER',   7839, '1981-04-02', 2975.00, NULL, 20),
(7698, 'BLAKE',  'MANAGER',   7839, '1981-05-01', 2850.00, NULL, 30),
(7782, 'CLARK',  'MANAGER',   7839, '1981-06-09', 2450.00, NULL, 10),
(7902, 'FORD',   'ANALYST',   7566, '1981-12-03', 3000.00, NULL, 20),
(7788, 'SCOTT',  'ANALYST',   7566, '1982-12-09', 3000.00, NULL, 20),
(7499, 'ALLEN',  'SALESMAN',  7698, '1981-02-20', 1600.00, 300.00,  30),
(7521, 'WARD',   'SALESMAN',  7698, '1981-02-22', 1250.00, 500.00,  30),
(7844, 'TURNER', 'SALESMAN',  7698, '1981-09-08', 1500.00, 0.00,    30),
(7654, 'MARTIN', 'SALESMAN',  7698, '1981-09-28', 1250.00, 1400.00, 30),
(7369, 'SMITH',  'CLERK',     7902, '1980-12-17',  800.00, NULL, 20),
(7876, 'ADAMS',  'CLERK',     7788, '1983-01-12', 1100.00, NULL, 20),
(7900, 'JAMES',  'CLERK',     7698, '1981-03-12',  950.00, NULL, 30),
(7934, 'MILLER', 'CLERK',     7782, '1982-01-23', 1300.00, NULL, 10);
```

Définition des données

L'ordre drop table et truncate

L’ordre DROP TABLE et TRUNCATE

- **Suppression d'une table :**
  - DROP TABLE nom_de_la_table

- **Pour le vidage complet d’une table, on utilise l’ordre TRUNCATE :**
  - TRUNCATE nom_de_table

Interrogation de données

Syntaxe de base

SELECT

- **La forme générale d'une interrogation SQL est la suivante :**

```
SELECT   liste d'attributs
FROM     nom de tables
WHERE    conditions;
```

- Les clauses **SELECT** et **FROM** sont **obligatoires**. Toutes les **autres** clauses ne servent qu'à affiner la sélection et sont donc **optionnelles**.
- **Consultation simple (complète) d'un objet (table ou vue)**

```
SELECT   *
FROM     dept;
```

SELECT

- **Sélection de colonnes d'une table**

Sélection de 2 colonnes de la table dept

```
SELECT deptno, loc
FROM  dept;
```

DISTINCT

- Permet **éliminer** les **valeurs dupliquées**.
- La fonction **DISTINCT** s'applique à **l'ensemble des colonnes ramenées**. Les lignes résultantes sont triées.

Exemple sans/avec DISTINCT :

```
SELECT  job
FROM   emp;
```

```
SELECT  distinct job
FROM  emp
```

Les attributs sélectionnés

- Colonne d'une table
- Constante : 1, 'ABCD', null
- Expression composite
  - Fonctions de chaînes : concat('Bonjour Monsieur ',Nom)
  - Calcul : salaire * 1.05
  - Fonction arithmétique : 4.0/3 * ACOS(-1) * POWER(5, 3)
  - Conditionnel : CASE WHEN salaire > 3000 THEN 'Haut' ELSE 'Bas' END as niveau_salaire
  - Agrégation : Sum(salaire)
  - Fonction utilisateur : ma_fonction(salaire)
- Pseudo colonnes : ROWNUM (Oracle)

Alias

- Permet de référencer une expression complexe dans une sélection.

```
SELECT  empno,ename,sal*1.1 + comm as salaire
FROM  emp
ORDER BY salaire;
```

En-tête de colonne

- Changement d'un titre de colonne à l'affichage
- Entre " " si le nom contient des séparateurs

```
SELECT sal "salaire mensuel", ename
FROM emp;
```

ORDER BY

- Trier les données à la sortie d'une requête d'interrogation
  - Par ordre ascendant (ASC) (par défaut)
  - Par ordre descendant (DESC),
- Une seule clause ORDER BY, toujours citée en dernier
- Les éléments cités dans la clause ne figurent pas nécessairement dans le SELECT
- On peut utiliser un alias dans une clause ORDER BY

Exemple : liste des salariés, leur emploi et leur salaire trié par emploi et salaire descendant.

```
SELECT  ename, job, sal
FROM  emp
ORDER BY  job, sal  DESC;
```

Interrogation de données

Syntaxe de base - Exercices

Exercice 3

- Afficher le contenu de la table EMP

empno	ename	job	mgr	hiredate	sal	comm	deptno
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Exercice 4

- Afficher la liste des départements (deptno) présents dans la table EMP, sans doublons

```
deptno
---------
10
20
30
```


46

Exercice 5

- Afficher le nom des employés (ename) et leur salaire (sal), en renommant les colonnes en "Nom" et "Salaire".

```
Nom        Salaire
--------   --------
SMITH      800
ALLEN      1600
WARD       1250
...
```


47

Exercice 6

- Afficher le nom des employés et leur salaire annuel (sal * 12).

```
Nom        salaire_annuel
--------   --------------
SMITH      9600
ALLEN      19200
WARD       15000
...
```

Exercice 7

- Afficher le nom et la date d'embauche de tous les employés triés par date d'embauche de la plus récente à la plus ancienne

```
ADAMS        12 jan 1983
SCOTT        09 dec 1982
MILLER       23 jan 1982
FORD         03 dec 1981
KING         17 nov 1981
MARTIN       28 sep 1981
TURNER       08 sep 1981
CLARK        09 jun 1981
BLAKE        01 may 1981
JONES        02 apr 1981
JAMES        12 mar 1981
WARD         22 feb 1981
ALLEN        20 feb 1981
SMITH        17 dec 1980
14 rows selected.
```

Interrogation de données

La restriction

WHERE

- La restriction permet d'appliquer une opération de filtrage sur les données.
- La clause **WHERE** sert simplement à limiter le nombre de lignes en ne gardant que celles qui correspondent aux critères de recherche.
- **Alias** non utilisable dans le **WHERE**

Exemple : Nom et localisation du département 20

```
SELECT  dname, loc
FROM  dept
WHERE  deptno = 20;
```

Liste des opérateurs

- La clause **WHERE** permet de définir une condition de sélection, appelée *prédicat*.
- Ce *prédicat* est une expression logique, construite à partir de plusieurs conditions reliées entre elles avec des opérateurs comme **AND**, **OR** ou **NOT**.
- Une expression logique peut prendre différentes formes :
  - Comparaison : vérifier une égalité ou une différence (=, !=, <>, >, >=, <, <=)
  - Intervalle de valeurs : vérifier si une valeur est comprise entre deux bornes (**BETWEEN**)
  - Recherche de texte : vérifier une correspondance avec des caractères (**LIKE** avec les jokers **_** et **%**)
  - Valeurs NULL : tester si une valeur est absente (**IS NULL**)
  - Appartenance à une liste : vérifier si une valeur est dans un ensemble (**IN, = ANY, = ALL**)
  - Négation : exprimer le contraire d'une condition (**NOT IN, NOT LIKE, NOT BETWEEN, IS NOT NULL**)

L’opérateur =

- Permet de vérifier si une valeur est **strictement** la **même** qu’une autre.

  Exemple :

  Quel est le numéro du matricule du président ?

```
SELECT  empno
FROM  emp
WHERE  job = 'PRESIDENT';
```

Les opérateurs < et >

- La notion d'ordre dépend du type de données utilisé :
  - Pour les nombres, on compare simplement les valeurs (plus grand, plus petit).
  - Pour les dates,
    - > signifie plus récent (postérieur)
    - < signifie plus ancien (antérieur)
- Il est nécessaire de convertir la chaîne à comparer en type DATE afin d'effectuer une compa-raison correcte.

Exemple : employés embauchés après le 31/12/82

Oracle :

```
SELECT empno, ename, hiredate
FROM emp
WHERE hiredate >
TO_DATE('1982-12-31', 'YYYY-MM-DD');
```

PostgreSQL :

```
SELECT empno, ename, hiredate
FROM emp
WHERE hiredate >
DATE '1982-12-31';
```

L'opérateur BETWEEN

- Compare les valeurs comprises entre (**BETWEEN**) n et (**AND**) m.
- Ce prédicat peut être précédé par l'opérateur logique unaire **NOT**.
- Les expressions peuvent être de type numériques, caractères ou date.

Exemples :

Employés ayant/n'ayant pas un salaire compris entre 1500 et 2000

(1)
```
SELECT  empno, ename, sal
FROM   emp
WHERE  sal
BETWEEN 1500 AND 2000;
```

(2)
```
SELECT empno, ename, sal
 FROM  emp
WHERE  sal
 NOT BETWEEN 1500 AND 2000;
```

L’opérateur IN

- Permet de comparer la valeur de l'expression située à gauche du mot clé **IN** à la liste de valeurs comprises entre parenthèses.
- Sensible à la casse
- Ce prédicat peut être précédé par l'opérateur logique unaire **NOT**.

Exemple : lister les numéros, noms, postes et salaires des analystes et employés de bureau :

```
SELECT  empno, ename, job, sal
FROM  emp
WHERE  job IN (‘ANALYST’,’CLERK’);
```

L’opérateur IN

Exemple :

Numéros et nom des employés ne travaillant pas dans les services 10 et 20

```
SELECT empno, ename
FROM  emp
WHERE  deptno NOT IN (10,20);
```

LIKE

- L'opérateur **LIKE** sert à rechercher du texte en se basant sur un modèle (un "motif").
- Il utilise des **caractères spéciaux** appelés jokers :
  - **%** : remplace *n'importe quelle suite de caractères* (0, 1 ou plusieurs)
  - **_** : remplace *un seul caractère*

Exemples :

(1) Liste des employés dont le nom commence par un A :

```
SELECT  ename
FROM  emp
WHERE  ename like 'A%';
```

(2) Liste des employés dont le nom contient un S en 5-ème position :

```
SELECT  ename
FROM  emp
WHERE  ename LIKE  '____S%';
```

Utilisation de NULL

- NULL ⇔ non défini

```
SELECT …
FROM  nom_objet
WHERE  coln  IS NULL ⇔ coln  NON renseignée
Ou
WHERE  coln  IS NOT NULL ⇔ coln  renseignée
```

- Coln = NULL, Coln ! = NULL => expression toujours FAUSSE
- NULL est NULL, mais NULL n'est pas égal à NULL

Exemples : employés sans/avec commisions

(1)
```
SELECT  empno, ename
FROM  emp
WHERE  comm IS NULL;
```

(2)
```
SELECT  empno, ename
FROM  emp
WHERE  comm is NOT NULL;
```

L'opérateur AND

- Liaison par l'opérateur logique AND
  - relie deux conditions ou prédicats,
  - ne renvoie des résultats que lorsque toutes les conditions sont vraies

Exemple :

Numéro du matricule, nom des employés, numéro de service et salaire pour les employés travaillant dans le service 30, ayant un salaire supérieur à 1500 et une commission

```
SELECT empno, ename, deptno, sal
FROM   emp
WHERE  deptno = 30
AND    sal > 1500
AND    comm  IS NOT NULL;
```

L'opérateur OR

- Liaison par l'opérateur logique OR
  - relie deux conditions,
  - renvoie des résultats lorsque l'une des conditions est vraie

Exemple :

Numéro du matricule, nom des employés, numéro de service et salaire pour les employés n'ayant pas de commission ou un salaire inférieur à 1500

```
SELECT empno, ename, deptno, sal
FROM   emp
WHERE  sal < 1500
OR     comm  IS NULL;
```

Table de vérité

AND	Faux	Vrai
Faux	Faux	Faux
Vrai	Faux	Vrai

OR	Faux	Vrai
Faux	Faux	Vrai
Vrai	Vrai	Vrai

Interrogation de données

La restriction – Exercices

Exercice 8

- Afficher le nom, le salaire et la commission des employés du département n° 30

ename	sal	comm
ALLEN	1600	300
WARD	1250	500
MARTIN	1250	1400
BLAKE	2850	
TURNER	1500	0
JAMES	950	

6 ligne(s) sélectionnée(s).

Exercice 9

- Afficher les employés, avec leur salaire, n'étant pas commissionnés

```
ename                 sal
---------- ---------
SMITH                 800
JONES                2975
BLAKE                2850
CLARK                2450
SCOTT                3000
KING                 5000
ADAMS                1100
JAMES                 950
FORD                 3000
MILLER               1300

10 rows selected.
```

Exercice 10

- Lister les employés dont leur salaire se situe entre 1100 et 3000 $

```
ename                 sal
---------- ---------
ALLEN                1600
WARD                 1250
JONES                2975
MARTIN               1250
BLAKE                2850
CLARK                2450
SCOTT                3000
TURNER               1500
ADAMS                1100
FORD                 3000
MILLER               1300

11 rows selected.
```

Exercice 11

- Lister les employés qui sont CLERK ou MANAGER

```
ename      job
---------- -------
SMITH      CLERK
JONES      MANAGER
BLAKE      MANAGER
CLARK      MANAGER
ADAMS      CLERK
JAMES      CLERK
MILLER     CLERK

7 rows selected.
```

Exercice 12

- Lister les employés embauchés après 1981

```
ename      hiredate
---------- ---------
SCOTT      09/12/82
ADAMS      12/01/83
MILLER     23/01/82

3 rows selected.
```

Exercice 13

- Lister les employés ayant une commission et dont le nom est terminé par la lettre N

```
ename                com
---------- ---------
ALLEN                300
MARTIN              1400

2 rows selected.
```

Exercice 14

- Lister les employés ayant une commission et dont le nom n' est pas terminé par la lettre N et ne contient pas la lettre A sur la deuxième position.

ename	comm
ALLEN	300
WARD	500
TURNER	0

3 rows selected.

Interrogation de données

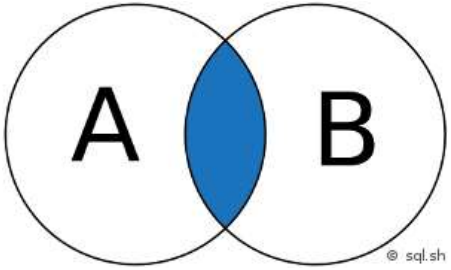
La jointure

Les jointures

- Lien entre deux tables disposant d'au moins une colonne commune sémantiquement.
- Liste des principales jointures :
  - INNER JOIN
  - NATURAL JOIN
  - LEFT JOIN
  - RIGHT JOIN

Les principales jointures

- INNER JOIN : intersection des deux ensembles.
  jointure interne pour retourner les enregistrements quand la condition est **strictement vraie** dans les 2 tables

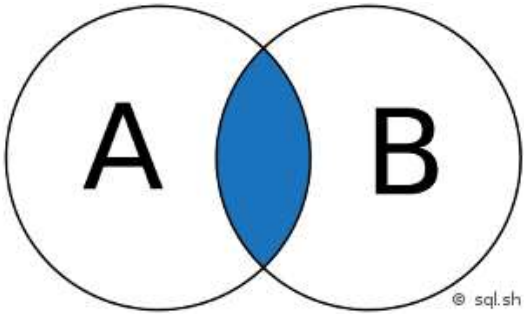


Syntaxe :

```
SELECT a.colonne_a_recuperer, b.colonne_a_recuperer
FROM   a
INNER JOIN b on a.id_commun = b.id_commun;
```

Les principales jointures

- 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.



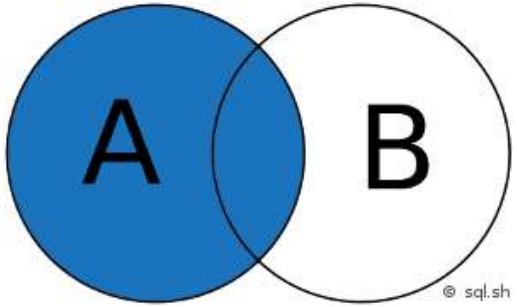
Syntaxe :

```
SELECT a.colonne_a_recuperer, b.colonne_a_recuperer
FROM   a
NATURAL JOIN b;
```

Les principales jointures

- LEFT JOIN : Jointure gauche.

  jointure externe pour retourner **tous les enregistrements** de la table de **gauche** (LEFT = gauche) même si la condition **n'est pas vérifiée** dans l'autre table.



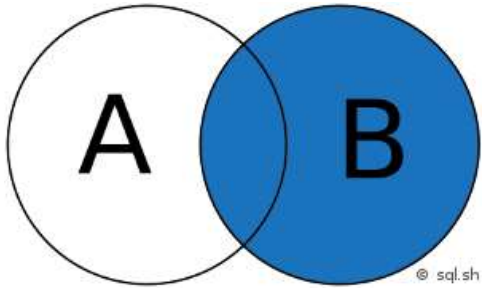
Syntaxe :

```
SELECT a.colonne_a_recuperer, b.colonne_a_recuperer
FROM   a
LEFT JOIN b on a.id_commun = b.id_commun;
```

Les principales jointures

- RIGHT JOIN : Jointure gauche.

  jointure externe pour retourner **tous les enregistrements** de la table de **droite** (RIGHT = droite) même si **la condition n'est pas vérifiée** dans l'autre table.



Syntaxe :

```
SELECT a.colonne_a_recuperer, b.colonne_a_recuperer
FROM   a
RIGHT JOIN b on a.id_commun = b.id_commun;
```

Interrogation de données

La jointure – Exercices

Exercice 15

- Lister les employés et le nom du département dans lequel ils se trouvent

```
ENAME      DNAME
---------- ----------
CLARK      ACCOUNTING
KING       ACCOUNTING
MILLER     ACCOUNTING
SMITH      RESEARCH
ADAMS      RESEARCH
FORD       RESEARCH
SCOTT      RESEARCH
JONES      RESEARCH
ALLEN      SALES
BLAKE      SALES
MARTIN     SALES
JAMES      SALES
TURNER     SALES
WARD       SALES

14 ligne(s)
sélectionnée(s).
```

Exercice 16

- Lister les employés et le nom de leur chef

ENAME	ENAME
SCOTT	JONES
FORD	JONES
ALLEN	BLAKE
WARD	BLAKE
JAMES	BLAKE
TURNER	BLAKE
MARTIN	BLAKE
MILLER	CLARK
ADAMS	SCOTT
JONES	KING
CLARK	KING
BLAKE	KING
SMITH	FORD

13 ligne(s) sélectionnée(s).

Exercice 17

- Même exercice en incluant les employés n'ayant pas de chef

ENAME	ENAME
SCOTT	JONES
FORD	JONES
ALLEN	BLAKE
WARD	BLAKE
JAMES	BLAKE
TURNER	BLAKE
MARTIN	BLAKE
MILLER	CLARK
ADAMS	SCOTT
JONES	KING
CLARK	KING
BLAKE	KING
SMITH	FORD
KING	

14 ligne(s) sélectionnée(s).

Exercice 18

- Depuis la table des employés, liste de tous les départements avec leurs employés ou vides

No dept	Nom dept	N° emp	Nom emp
10	ACCOUNTING	7782	CLARK
10	ACCOUNTING	7839	KING
10	ACCOUNTING	7934	MILLER
20	RESEARCH	7369	SMITH
20	RESEARCH	7566	JONES
20	RESEARCH	7788	SCOTT
20	RESEARCH	7876	ADAMS
20	RESEARCH	7902	FORD
30	SALES	7499	ALLEN
30	SALES	7521	WARD
30	SALES	7654	MARTIN
30	SALES	7698	BLAKE
30	SALES	7844	TURNER
30	SALES	7900	JAMES
40	OPERATIONS	*NULL*	*NULL*

Interrogation de données

Les opérateurs ensemblistes

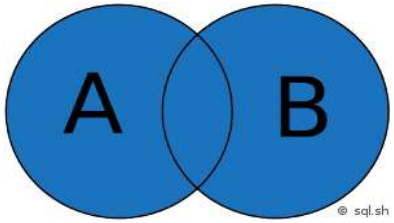
Les opérateurs ensemblistes

- Servent à **combiner les résultats de plusieurs requêtes** (SELECT).

- Règles :
  - Les doublons sont supprimés
  - Même structure obligatoire (même nombre de colonnes et des types compatibles)
  - Combinaison possible de plus de deux SELECT,
  - Ordres exécutés du haut vers le bas
  - Priorité aux parenthèses.

L’opérateur UNION

- Rassemble les résultats de deux requêtes **sans doublons**
  → on fusionne tout, mais chaque ligne apparaît une seule fois

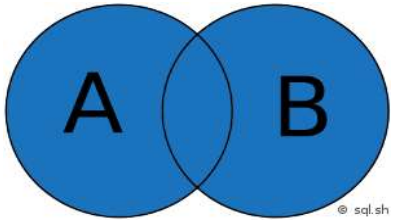


Exemple : Nom, composant (texte : salaire ou commission), montant (salaire ou commission) des salariés ayant une commission.

```
SELECT ename employe, 'salaire' composant, sal montant
FROM  emp WHERE comm  IS NOT NULL
UNION
SELECT ename, 'commission' composant, comm montant
 FROM  emp WHERE comm  IS NOT NULL;
```

L'opérateur UNION ALL

- le seul opérateur ensembliste qui ne fait pas de **DISTINCT implicite**.



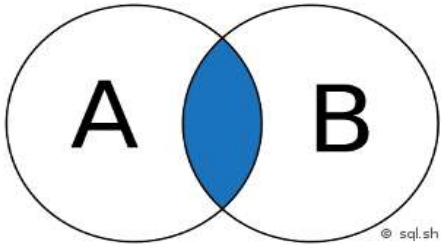
Exemple : Numéros de départements de tous les employés et de tous les départements

```
SELECT deptno
FROM emp
UNION ALL
SELECT deptno
FROM dept;
```

L'opérateur INTERSECT

- garde seulement les lignes communes aux deux requêtes
  → ce que les deux résultats ont en commun

  Exemple : trouver les managers qui gagnent plus de 2500.

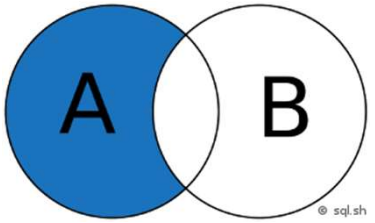



```
SELECT ename, job, sal
FROM emp
WHERE job = 'MANAGER'
INTERSECT
SELECT ename, job, sal
FROM emp
WHERE sal > 2500;
```

L'opérateur MINUS / EXCEPT

- enlève les lignes de la deuxième requête
  → on enlève un résultat à un autre

  Exemple : trouver les managers qui ne gagnent PAS plus de 2500



Oracle :

```
SELECT ename, job, sal
FROM emp
WHERE job = 'MANAGER'
MINUS
SELECT ename, job, sal
FROM emp
WHERE sal > 2500;
```

PostgreSQL :

```
SELECT ename, job, sal
FROM emp
WHERE job = 'MANAGER'
EXCEPT
SELECT ename, job, sal
FROM emp
WHERE sal > 2500;
```

Interrogation de données

Les opérateurs ensemblistes – Exercices

Exercice 19

- Liste des employés dont le nom :
  - commence par un ‘M’ ou
  - se termine par un ‘N’
- Avec doublons, puis sans doublons

```
SANS DOUBLONS

ename
----------
ALLEN
MARTIN
MILLER
```

```
AVEC DOUBLONS

ename
------
MARTIN
MILLER
ALLEN
MARTIN
```

Exercice 20

- Lister les employés qui :
  - ont un job CLERK
  - et travaillent dans le département 20

ename	job	deptno
ADAMS	CLERK	20
SMITH	CLERK	20

Exercice 21

- Lister les départements qui :
  - ont des employés
  - mais aucun employé avec salaire > 3000

```
deptno
-------
30
20
```

Interrogation de données

Regroupement et calculs

Les fonctions d'agrégation

- Les fonctions d'agrégat :
  - COUNT : Nombre de lignes/valeurs de colonne satisfaisant la requête
  - AVG : Moyenne des valeurs de la colonne satisfaisant la requête.
  - SUM : Somme des valeurs de la colonne satisfaisant la requête
  - MIN/MAX : Minimum/Maximum des valeurs de la colonne satisfaisant la requête.

- les valeurs "NULL" sont ignorées par les fonctions de groupe.
- si le groupe est un ensemble vide, la valeur retournée par les fonctions MIN, MAX, AVG, VARIANCE est la valeur NULL.

La fonction MIN

- Retourne la plus petite valeur pour le groupe concerné.
- S’applique à tous types de données.

La fonction MAX

- Retourne la plus grande valeur pour le groupe concerné.
- S’applique à tout type de données.

95

La fonction SUM

- Retourne la somme des valeurs non NULL pour le groupe concerné.

- NULL si toutes les valeurs du même groupe sont NULL

La fonction COUNT

- Permet de compter le nombre d’enregistrements dans une table.

Les fonctions d'agrégation

Exemple : Somme des commissions des employés n'étant pas président

```
SELECT  SUM(comm) total_commission
FROM  emp
WHERE  job <> 'PRESIDENT' ;
```

Le verbe GROUP BY

- permet de **regrouper les lignes** ayant les mêmes valeurs dans certaines colonnes
- Equivalent du **DISTINCT**
- Utilisé spécifiquement pour appliquer des calculs sur chaque groupe
- **Obligatoire** quand on utilise une fonction d'agrégation avec d'autres colonnes à afficher.

Le verbe GROUP BY

- Exemple : Moyenne des salaires par service

```
SELECT  deptno, AVG(sal)
FROM  emp
GROUP BY  deptno;
```

Le verbe HAVING

- La clause HAVING est l'équivalent du WHERE appliqué aux groupes.

Exemple :

Emploi, nombre d'employés exerçant l'emploi et moyenne de salaire de l'emploi, pour les emplois exercés par au moins 2 employés

```
SELECT  job, COUNT(*), ROUND(AVG(sal),2)
FROM  emp
GROUP BY  job
HAVING  count(*) > 1;
```

Interrogation de données

Regroupement et calculs - Exercices

Exercice 22

- Salaire moyen de chaque fonction

job	salaire moyen
ANALYST	3000,00
CLERK	1037,50
MANAGER	2758,33
PRESIDENT	5000,00
SALESMAN	1400,00

Exercice 23

- Nombre d'employés pour chacun des départements

dname	nb employés
ACCOUNTING	3
OPERATIONS	0
RESEARCH	5
SALES	6

Exercice 24

- Moyenne des salaires, dernière date d'embauche et première date d'embauche par département

deptno	sal_moy	maxdate	mindate
30	1567	1981-09-28	1981-02-20
10	2917	1982-01-23	1981-06-09
20	2175	1983-01-12	1980-12-17

Exercice 25

- Départements avec plus de 3 employés

```
Dname
-------
SALES
RESEARCH
```

Interrogation de données

Les sous-sélections

Les sous-sélections

- Principe

  Requête à l'intérieur d'une autre requête
- Pourquoi on l'utilise ?

  Pour préparer un résultat, puis l'utiliser dans une autre requête.
  Exemple : connaître les salariés dont le salaire est supérieur à la moyenne.
- Où peut-on utiliser une sous-requête ?
  - dans un WHERE → pour filtrer
  - dans un FROM → comme une table temporaire
  - dans un SELECT → pour ajouter une valeur calculée

Les sous-sélections

- Expression d'un prédicat à l'aide du résultat d'un select.

  SELECT ....
  FROM objet
  WHERE coln (*opérateur*) (SELECT .... FROM ... WHERE ...)

- Opérateurs :

  - EXISTS, NOT EXISTS
  - IN ou = ANY, NOT IN ou !=ALL
  - {=, !=, > ,>=, <, <=}{ANY, ALL}

Note : Si une requête est susceptible de renvoyer plus d'une valeur, utiliser IN au lieu de = .

Les sous-sélections

Exemple :
Employé avec son salaire le mieux payé

```
SELECT  e.empno, e.ename, e.sal
FROM    emp e
WHERE e.sal = (SELECT max(sal) FROM emp)
```

Les sous-sélections

- La sous-interrogation simple :

  Elle est indépendante de l'interrogation principale.

  La sous-interrogation est évaluée en premier, puis l'interrogation principale avec le résultat trouvé.

- La sous-interrogation synchronisée :

  Elle fait référence à une colonne de l'interrogation principale.

  La sous-interrogation est réévaluée pour chaque ligne de l'interrogation principale.

Les sous-sélections simples

- La sous-interrogation simple :

Exemple :

Salariés gagnant moins que Clark :

```
SELECT empno, ename, sal, job
FROM  emp
WHERE  sal <  (SELECT  sal
               FROM  emp
               WHERE  ename = 'CLARK');
```

Les sous-sélections synchronisées

- Une sous sélection synchronisée utilise dans le corps de sa requête une colonne de la sélection mère

Exemple :

Liste des employés ayant un plus gros salaire que la moyenne des salaires de leur service :

```
SELECT ename, sal
FROM emp e1
WHERE sal > ( SELECT AVG(sal) FROM emp e2 WHERE e1.deptno = e2.deptno);
```

Les sous-sélections synchronisées

Autre exemple :

Employés ne travaillant pas dans le même département que leur manager :

```
SELECT  empno, ename
FROM  emp e
WHERE  deptno <> (SELECT  deptno
                  FROM  emp
                  WHERE  e.mgr = empno)
AND  mgr is NOT NULL ;
```

La sous-interrogation est réévaluée pour chaque nouveau manager (e.mgr).

Les jointures et sous-sélections

- Les jointures et les sous sélections cohabitent parfaitement :

  Exemple :

  Lister les vendeurs et le nom de leur chef du département des ventes

```
SELECT e.ename nom, chef.ename chef
FROM emp e
INNER JOIN emp chef ON e.mgr = chef.empno
WHERE e.deptno = (SELECT  deptno FROM dept WHERE dname =
'SALES');
```

L’opérateur EXISTS

- Il envoie le booléen VRAI ou FAUX selon le résultat de la sous-requête.

- Si l'évaluation de la sous-requête donne lieu à une ou plusieurs ligne(s), la valeur retournée est VRAI.

  Dans le cas contraire, la valeur retournée est FAUX.

L’opérateur EXISTS

Exemple : Liste des noms de salariés avec une colonne calculée, « salaire avec prime » dont leur commission est supérieure à 0

```
SELECT  e.ename, e.sal * 1.05 as "salaire avec prime"
FROM  emp e
WHERE  EXISTS (SELECT 'x' FROM emp e2 WHERE e.empno = e2.empno AND e2.comm IS NOT NULL);
```

L’opérateur IN

- Permet de comparer la valeur de l'expression située à gauche du mot clé IN à la liste de valeurs comprises entre parenthèses. La condition de recherche est satisfaite quand l'expression de comparaison est comprise dans la liste de valeurs.

Exemple : Employés embauchés qui travaillent dans le département dont le nom est « SALES » ou « ACCOUNTING »

```
SELECT  e.empno, e.ename, e.job
FROM   emp e
WHERE  e.deptno  IN
            (SELECT  e2.deptno  FROM   dept e2
                  WHERE  e2.dname  IN  (‘SALES’,’ACCOUNTING’));
```

L'opérateur ANY

- Permet de comparer une valeur avec le résultat d'une sous-requête. Il est ainsi possible de vérifier si une valeur est "égale", "différente", "supérieur", "supérieure ou égale", "inférieure" ou "inférieur ou égale" pour au moins une des valeurs de la sous-requête.

Exemple : Employés non vendeurs, ayant un salaire plus élevé qu'au moins un des vendeurs.

```
SELECT  e1.empno, e1.ename, e1.sal, e1.job
FROM  emp e1
WHERE  e1.sal > ANY(SELECT  e2.sal FROM  emp e2 WHERE  e2.job = 'SALESMAN')
AND e1.job <> 'SALESMAN';
```

L’opérateur ALL

- La condition est vraie si la comparaison est vraie pour chacune des valeurs retournées.

  Permet de s’assurer qu’une condition est “égale”, “différente”, “supérieure”, “inférieure”, “supérieure ou égale” ou “inférieure ou égale” pour tous les résultats retournés par une sous-requête.

  Exemple : Employés non vendeurs, ayant un salaire plus élevé que chaque vendeur.

```
SELECT   e1.empno, e1.ename, e1.sal
FROM  emp e1
WHERE  e1.sal > ALL (SELECT  e2.sal FROM  emp e2 WHERE  e2.job =
'SALESMAN');
AND e1.job <> 'SALESMAN';
```

Plusieurs sous-colonnes sélectionnées

Exemple :

Sélection des noms, emplois et salaires des employés ayant les mêmes : emploi, salaire et n° département que « FORD »

```
SELECT ename, job, sal
FROM  emp
WHERE  (job, sal, deptno) = (SELECT  job, sal, deptno FROM  emp
WHERE  ename = 'FORD' )
AND  ename <> 'FORD' ;
```

La sous-sélection dans le FROM

- Au lieu de créer une vue temporairement, il est possible de positionner, à la place du nom de la table ou de la vue, une requête SQL indépendante. Elle sera alors considérée comme une vue, avec les contraintes associées (ORDER BY) :

Exemple : Employés dont le salaire est supérieur à 2000 euros et dont le nom commence par S.

```
SELECT  cadre.ename FROM
(
    SELECT  ename, sal
        FROM  emp
        WHERE  sal > 2000
    ) cadre
    WHERE  cadre.ename  LIKE  'S%';
```

La sous-sélection dans le SELECT

- Au lieu de faire une première requête et de réutiliser le résultat dans une autre requête, on peut tout faire en une grâce à la sous-sélection dans le SELECT.

  Elle ne doit cependant retourner qu'**une seule ligne** et **une seule colonne**.

Exemple : Afficher chaque employé avec le salaire moyen :

```
SELECT ename, sal,
 (SELECT AVG(sal) FROM emp) AS Moyenne
FROM emp;
```

La sous-sélection dans le SELECT

Exemple : Afficher la moyenne des salaires par département

```
SELECT ename, deptno, sal,
(SELECT AVG(sal)
FROM emp e2
WHERE e2.deptno = e1.deptno) AS moy_dept
FROM emp e1;
```

Interrogation de données

Les sous-sélections - Exercices

Exercice 26

- Liste des employés dont le salaire est supérieur à la moyenne des salaires de leur service

```
ENAME
----------
ALLEN
JONES
BLAKE
SCOTT
KING
FORD

6 ligne(s) sélectionnée(s).
```

Exercice 27

- Liste des employés étant dans le même département que SMITH

```
ENAME
----------
SMITH
JONES
SCOTT
ADAMS
FORD
```

Exercice 28

- Liste des employés dont le salaire est supérieur au salaire de leur chef

```
ENAME
----------
SCOTT
FORD
```

Exercice 29

- Liste des employés ayant la même fonction et étant dans le même service que SMITH

```
ENAME
-----
SMITH
ADAMS
```

Exercice 30

- Afficher chaque employé avec son nom, son salaire et le salaire maximum de l'entreprise

ename	sal	max_sal
KING	5000.00	5000.00
JONES	2975.00	5000.00
BLAKE	2850.00	5000.00
CLARK	2450.00	5000.00
FORD	3000.00	5000.00
SCOTT	3000.00	5000.00
ALLEN	1600.00	5000.00
WARD	1250.00	5000.00
TURNER	1500.00	5000.00
MARTIN	1250.00	5000.00
SMITH	800.00	5000.00
ADAMS	1100.00	5000.00
JAMES	950.00	5000.00
MILLER	1300.00	5000.00

Exercice 31

- Afficher les départements dont le salaire moyen est supérieur à 2500.

deptno	avg_sal
10	2917

Manipulation de données

L'ordre INSERT

L’ordre INSERT

- La commande INSERT sert à ajouter une ou plusieurs lignes de données dans une table déjà créée.
- insertion de données externes (saisies),
- insertion de données internes (depuis une autre table).

L'ordre INSERT

- Insertion sans nom de colonnes :

Exemple : INSERT INTO dept
VALUES (60, 'RH', 'PARIS');

Dans ce cas, le nombre de valeurs insérées doit correspondre au nombre de colonnes de la table.

- Insertion avec nom de colonne :

Exemple : INSERT INTO dept (deptno)
VALUES (70);

Insertion du résultat d'un SELECT

- L'ordre SELECT, avec toutes ses fonctionnalités, ORDER BY exclu, peut être utilisé dans un ordre INSERT.

Exemple : Insertion d'un nouveau département en prenant le dernier id et en lui ajoutant 10.

```
INSERT INTO dept
SELECT max(DEPTNO)+10 as DEPTNO, "new departement", "new location"
FROM dept;
```

Manipulation de données

L'ordre DELETE

L’ordre DELETE

- L’ordre DELETE sert à **supprimer des lignes dans une table**.
  On utilise WHERE pour **choisir quelles lignes supprimer**.

Exemple : Suppression du département 30

```
DELETE FROM  dept
WHERE deptno = 30;
```

L'ordre DELETE

- Pour supprimer toutes les lignes d'une table, il n'est pas nécessaire d'inclure une clause WHERE.

  Exemple : Suppression de tous les départements
  DELETE FROM dept;

Manipulation de données

L'ordre UPDATE

L’ordre UPDATE

- L’ordre UPDATE permet la modification d’un ou plusieurs champs d’une ou plusieurs lignes d’une table.

- La modification à faire peut contenir :
  - des valeurs fixes (saisies),
  - des noms de colonnes,
  - ou même des requêtes.

L'ordre UPDATE

Exemple :

Affecter l'analyste SCOTT (numéro d'employé 7788) aux Ventes (deptno : 30)

```
UPDATE emp
SET deptno = 30
WHERE empno = 7788;
```

L’ordre UPDATE

Exemple :

Augmenter les salaires de 10%

```
UPDATE  emp
SET sal = sal*1.1;
```

L'ordre UPDATE

Exemple :

Affecter les employés des ventes aux opérations avec une augmentation de 5%

```
UPDATE emp
SET deptno = 40,
        sal = sal*1.05
WHERE deptno = 30;
```

L'ordre UPDATE

- Une modification synchronisée permet d'utiliser le résultat d'une requête sur une table, dans un ordre UPDATE sur cette même table.

Exemple :

```
UPDATE emp e
SET e.sal = (SELECT AVG (e2.sal)
                FROM  emp e2
                WHERE e.deptno = e2.deptno);
```

Note : le select ne doit ramener qu'une seule ligne.

Interrogation de données

Les sous-sélections - Exercices

Exercice 32

- Insérer un nouvel employé de numéro 1234, avec votre prénom.
- Augmenter de 5% le salaire des vendeurs et diminuer de 10% ceux des CLERCKs,
- Supprimer l'enregistrement relatif à votre prénom.
- Supprimer le plus grand numéro de département où il n'y a aucun salarié rattaché.

Les fonctions conditionnelles

Les fonctions conditionnelles

- Les fonctions conditionnelles comme CASE et COALESCE agissent sur tous les types de données.

La fonction CASE

- Permet d'utiliser des conditions de type "si / sinon".

  CASE peut être utilisé dans n'importe quelle instruction ou clause, telle que SELECT, UPDATE, DELETE, WHERE, ORDER BY ou HAVING.

  Exemple : Salaires de tous les employés sauf du président

```
SELECT  empno, ename, job,
  CASE job
     WHEN 'PRESIDENT' THEN NULL
     ELSE sal
  END as sal
FROM  emp;
```

La fonction CASE

Exemple : Salaires de tous les employés sauf du président

```
SELECT  empno, ename, job,
  CASE
     WHEN job  = 'PRESIDENT' THEN NULL
     ELSE sal
  END as sal
FROM  emp;
```

La fonction COALESCE

- Permet de remplacer une valeur si elle est nulle.
- Syntaxe : COALESCE (expr, valeur)

résultat : « expr » si expr IS NOT NULL
« valeur » si expr IS NULL

Exemple : Salaire avec commission comprise de tous les salariés

```
SELECT empno, ename, sal + COALESCE((comm), 0) "sal et comm "
FROM emp;
```

Les fonctions conditionnelles

Exercices

Exercice 33

- Afficher les noms des employés puis une colonne statut. Quand il s'agit du président, on affiche "boss" sinon on affiche "employé"

Ename	job
KING	BOSS
JONES	EMPLOYE
CLARK	EMPLOYE
FORD	EMPLOYE
SCOTT	EMPLOYE
BLAKE	EMPLOYE
ALLEN	EMPLOYE
WARD	EMPLOYE
TURNER	EMPLOYE
SMITH	EMPLOYE
ADAMS	EMPLOYE
MILLER	EMPLOYE
JAMES	EMPLOYE
MARTIN	EMPLOYE

Exercice 34

- Afficher le commentaire selon la valeur de la commission

Ex : comm::varchar

ENAME	COMM
SMITH	Non Commissionné
ALLEN	300
WARD	500
JONES	Non Commissionné
MARTIN	1400
BLAKE	Non Commissionné
CLARK	Non Commissionné
SCOTT	Non Commissionné
KING	Non Commissionné
TURNER	Très mauvais vendeur
ADAMS	Non Commissionné
JAMES	Non Commissionné
FORD	Non Commissionné
MILLER	Non Commissionné

Les fonctions sur les dates

Les fonctions sur les dates

- Ajouter *n unités* à une date
  SELECT date '2024-01-01' + interval '5 day';
- Valeurs possibles :
  - Second
  - Minute
  - Hour
  - Day
  - Week
  - Month
  - year

Les fonctions sur les dates

- Dernier jour du mois :

  SELECT (date_trunc('month', date '2024-02-10') + interval '1 month - 1 day')::date;

Les fonctions sur les dates

- Formater une date :
  SELECT to_char(now(), 'YYYY-MM-DD');
  SELECT to_char(now(), 'DD/MM/YYYY');
  SELECT to_char(now(), 'YYYY-MM-DD HH24:MI:SS');
- Valeurs possibles :
  - YYYY → année
  - MM → mois
  - DD → jour
  - HH24 → heure
  - MI → minute
  - SS → seconde

Exercice 35

- Afficher la fin du mois de chaque date d'embauche de chaque employé

Les fonctions de chaînes de caractères

Les fonctions de chaînes de caractère

- Elles agissent sur les colonnes de type CHAR ou VARCHAR.
  - LOWER(chaine) : Retourne la chaîne de caractères "chaine" en minuscules.
  - UPPER(chaine) : Retourne la chaîne de caractères "chaine" en majuscules.
  - REPLACE(colonne_concerne,text_a_remplacer,nouveau_texte) : Retourne une chaîne de caractère dans laquelle toutes les occurrences de "chaine2" dans "chaine1" ont été remplacées par "chaine3".

Les fonctions de chaînes de caractère

- CONCAT(chaine1,chaine2,...) : retourne la chaîne de caractères issue de la concaténation des chaînes de caractères "chaine1","chaine2",etc.
- TRIM(chaine) : supprime les espaces de début et de fin de "chaine"

Exercice 36

- Afficher les jobs "ANALYST" en "Analyste"
- Concaténer le job et le nom de chaque employé

Corrigés d'exercices

Exercice 1

- Créer la table dept avec la clé primaire :

```
CREATE TABLE dept (
  DEPTNO INTEGER NOT NULL,
  DNAME varchar(14) DEFAULT NULL,
  LOC varchar(13) DEFAULT NULL,
  CONSTRAINT PK_DEPT_DEPTNO PRIMARY KEY (DEPTNO)
) ;
```

Exercice 1

- Créer la table emp avec la clé primaire

```
CREATE TABLE emp (
    EMPNO integer NOT NULL,
    ENAME varchar(10) DEFAULT NULL,
    JOB varchar(9) DEFAULT NULL,
    MGR integer DEFAULT NULL,
    HIREDATE date DEFAULT NULL,
    SAL decimal(7,2) DEFAULT NULL,
    COMM decimal(7,2) DEFAULT NULL,
    DEPTNO integer DEFAULT NULL,
    CONSTRAINT PK_EMP_EMPNO  PRIMARY KEY (EMPNO)
);
```

Exercice 2

- Créer les clés étrangères et références de la table emp

```
-- Création des index
CREATE INDEX FK_EMP_DEPTNO ON emp (DEPTNO);
CREATE INDEX FK_EMP_MGR ON emp (MGR);

-- Ajout des contraintes de clés étrangères
ALTER TABLE emp
ADD CONSTRAINT DEPT_FK_DEPTNO FOREIGN KEY (DEPTNO) REFERENCES
dept (DEPTNO),
ADD CONSTRAINT EMP_FK_EMPNO FOREIGN KEY (MGR) REFERENCES emp
(EMPNO);
```

Exercice 3

- Afficher le contenu de la table EMP

```
SELECT *
FROM emp
```


168

Exercice 4

- Afficher la liste des départements (deptno) présents dans la table EMP, sans doublons

```
SELECT DISTINCT(deptno)
FROM emp
ORDER BY deptno
```


169

Exercice 5

- Afficher le nom des employés (ename) et leur salaire (sal), en renommant les colonnes en "Nom" et "Salaire".

```
SELECT ename as Nom, sal "Salaire"
FROM emp
```


170

Exercice 6

- Afficher le nom des employés et leur salaire annuel (sal * 12).

```
SELECT ename as Nom, sal * 12 as salaire_annuel
FROM emp
```

Exercice 7

- Afficher le nom et la date d'embauche de tous les employés triés par date d'embauche de la plus récente à la plus ancienne

```
SELECT ename as Nom, hiredate
FROM emp
ORDER BY hiredate DESC
```

Exercice 8

- Afficher le nom, le salaire et la commission des employés du département n° 30

```
SELECT ename, sal, comm
FROM emp
WHERE deptno = 30;
```

Exercice 9

- Afficher les employés, avec leur salaire, n'étant pas commissionnés

```
SELECT ename, sal, comm
FROM emp
WHERE comm IS NULL;
```

Exercice 10

- Lister les employés dont leur salaire se situe entre 1100 et 3000 $

```
SELECT ename, sal
FROM emp
WHERE sal BETWEEN 1100 AND 3000 ;
```

Exercice 11

- Lister les employés qui sont CLERK ou MANAGER

```
SELECT ename, sal
FROM emp
WHERE job IN ('CLERK', 'MANAGER') ;

SELECT ename, sal
FROM emp
WHERE job = 'CLERK' OR job = 'MANAGER' ;
```

Exercice 12

- Lister les employés embauchés après 1981

```
SELECT ename, hiredate
FROM emp
WHERE hiredate  > DATE '31/12/81';
```

Exercice 13

- Lister les employés ayant une commission et dont le nom est terminé par la lettre N

```
SELECT ename, comm
FROM emp
WHERE comm  IS NOT NULL AND ename LIKE '%N';
```

Exercice 14

- Lister les employés ayant une commission et dont le nom n’ est pas terminé par la lettre N et ne contient pas la lettre A sur la deuxième position.

```
SELECT ename, comm
FROM emp
WHERE comm  IS NOT NULL AND NOT ename LIKE '_A%N';
```

Exercice 14

- Lister les employés ayant une commission et dont le nom n’ est pas terminé par la lettre N et ne contient pas la lettre A sur la deuxième position.

```
SELECT ename, comm
FROM emp
WHERE comm  IS NOT NULL AND NOT ename LIKE '_A%N';
```

Exercice 15

- Lister les employés et le nom du département dans lequel ils se trouvent

```
SELECT ename, dname
FROM emp e
INNER JOIN dept d ON e.deptno = d.deptno
ORDER BY dname
```

Exercice 16

- Lister les employés et le nom de leur chef

```
SELECT e.ename as employe, m.ename as mangager
FROM emp e
INNER JOIN emp m ON  e.mgr = m.empno
ORDER BY e.mgr
```

Exercice 17

- Même exercice en incluant les employés n'ayant pas de chef

```
SELECT e.ename AS employe, m.ename AS mangager
FROM emp e
LEFT JOIN emp m ON  e.mgr = m.empno
ORDER BY e.mgr
```

Exercice 18

- Depuis la table des employés, liste de tous les départements avec leurs employés ou vides

```
SELECT     d.deptno "No dept",
           d.dname  "Nom dept",
           e.empno  "N° emp",
           e.ename  "Nom emp"
FROM     emp e
RIGHT JOIN dept d ON e.DEPTNO = d.DEPTNO;
```

Exercice 19

- Liste des employés dont le nom :
  - commence par un ‘M’ ou
  - se termine par un ‘N’

- Avec doublons (1) et sans doublon (2)

(1)
```
SELECT ename
FROM emp
WHERE ename LIKE 'M%'
UNION ALL
SELECT ename
FROM emp
WHERE ename LIKE '%N';
```

(2)
```
SELECT ename
FROM emp
WHERE ename LIKE 'M%'
UNION
SELECT ename
FROM emp
WHERE ename LIKE '%N';
```

Exercice 20

- Lister les employés qui :
  - ont un job CLERK
  - et travaillent dans le département 20

```
SELECT ename, job, deptno
FROM emp
WHERE job = 'CLERK'
INTERSECT
SELECT ename, job, deptno
FROM emp
WHERE deptno = 20;
```

Exercice 21

- Lister les départements qui :
  - ont des employés
  - mais aucun employé avec salaire > 3000

```
SELECT d.deptno FROM dept d
INNER JOIN emp e ON e.deptno = d.deptno
EXCEPT
SELECT deptno
FROM emp
WHERE sal > 3000
```

Exercice 22

- Salaire moyen de chaque fonction

```
SELECT  job, ROUND(AVG(sal), 2) as "SALAIRE MOYEN"
FROM  emp
GROUP BY  job
ORDER BY job
```

Exercice 23

- Nombre d'employés pour chacun des départements

```
SELECT  dname, count(empno)
FROM  dept
LEFT JOIN emp ON emp.deptno = dept.deptno
GROUP BY dept.deptno
```

Exercice 24

- Moyenne des salaires, dernière date d'embauche et première date d'embauche par département

```
SELECT  deptno, AVG(sal) as sal_moy,
MAX(hiredate) maxdate,  MIN(hiredate)
mindate
FROM  emp
GROUP BY  deptno;
```

Exercice 25

- Départements avec plus de 3 employés

```
SELECT dname
FROM emp
INNER JOIN dept ON dept.deptno = emp.deptno
GROUP BY dept.deptno
HAVING COUNT(*) > 3
```

Exercice 26

- Liste des employés dont le salaire est supérieur à la moyenne des salaires de leur service

```
SELECT  e1.ename
FROM  emp e1
WHERE  e1.sal  >  (SELECT  AVG(sal)
                   FROM  emp
                   WHERE  e1.deptno = deptno
                   GROUP BY  deptno);
```

Exercice 27

- Liste des employés étant dans le même département que SMITH

```
SELECT  e.ename
FROM  emp e
WHERE  e.deptno = (SELECT  deptno
                   FROM  emp
                   WHERE  ename = 'SMITH');
```

Exercice 28

- Liste des employés dont le salaire est supérieur au salaire de leur chef

```
SELECT  e.ename
FROM  emp e
WHERE  e.sal > ANY (SELECT  e2.sal
                    FROM   emp e2
                    WHERE  e2.empno = e.mgr);
```

Exercice 29

- Liste des employés ayant la même fonction et étant dans le même service que SMITH

```
SELECT  e.ename
    FROM  dept d,  emp e
    WHERE  (e.job, d.dname) =
                                (SELECT  e1.job, d1.dname
                                 FROM  dept d1, emp e1
                                WHERE  e1.ename = 'SMITH'
                                 AND  d1.deptno = e1.deptno)
    AND  d.deptno = e.deptno ;
```

Exercice 30

- Afficher chaque employé avec son nom, son salaire et le salaire maximum de l'entreprise

```
SELECT ename, sal,
        (SELECT MAX(sal) FROM emp) AS max_sal
FROM emp;
```

Exercice 31

- Afficher les départements dont le salaire moyen est supérieur à 2500.

```
SELECT deptno, avg_sal
FROM ( SELECT deptno, AVG(sal) AS avg_sal
            FROM emp GROUP BY deptno )
tab_sal_moyen
WHERE avg_sal > 2500;
```

Exercice 32

- Insérer un nouvel employé de numéro 1234, avec votre prénom.

```
INSERT  INTO  emp  VALUES (1234,'Votre nom','DEV',7839,'2023/01/05',3900,null,10);
```

- Augmenter de 5% le salaire des vendeurs et diminuer de 10% ceux des CLERCKs,

```
UPDATE emp
SET   sal =
CASE
      WHEN job = 'CLERK' THEN sal*0.90
      ELSE sal*1.05
END
WHERE  job IN ('CLERK','SALESMAN');
```

Exercice 32

- Supprimer votre prénom.

```
UPDATE emp SET ENAME = NULL WHERE EMPNO = 1234;
```

- Supprimer le plus grand numéro de département où il n'y a aucun salarié rattaché.

```
DELETE FROM dept WHERE DEPTNO =
(SELECT MAX(DEPTNO)
FROM dept
WHERE DEPTNO NOT IN (SELECT DISTINCT DEPTNO FROM emp));
```

OU

```
DELETE FROM dept WHERE DEPTNO =
(SELECT max(d.DEPTNO)
FROM dept d
LEFT JOIN emp e ON d.DEPTNO = e.DEPTNO
WHERE e.DEPTNO IS NULL);
```

Exercice 33

- Afficher les noms des employés puis une colonne statut. Quand il s'agit du président, on affiche "boss" sinon on affiche "employé"

```
select ename,
CASE job
WHEN 'PRESIDENT' THEN 'BOSS'
ELSE 'EMPLOYE'
END AS statut
FROM emp
```

Exercice 34

- Afficher le commentaire selon la valeur de la commission

```
SELECT ename,
CASE
    WHEN comm IS NULL THEN 'Non commissioné'
    WHEN comm = 0 THEN 'très mauvais vendeur'
    ELSE comm::varchar
END AS comm
FROM emp
```

Exercice 35

- Afficher la fin du mois de chaque date d'embauche de chaque employé

```
SELECT (date_trunc('month', hiredate) +
interval '1 month - 1 day')::date from emp
```

Exercice 36

- Afficher les jobs "ANALYST" en "Analyste"

  SELECT REPLACE(job, 'ANALYST', 'Analyste') FROM emp

- Concaténer le job et le nom de chaque employé

  SELECT CONCAT(ename, ' ' , job) FROM emp