Licence Professionnelle ISIL

Outils de développement des Bases de données

Pr. Lamia ZIAD
École Supérieure de Technologie d'Essaouira

Contents

Chapter 1

Transactions Oracle

Definition

Une **transaction** est une sequence d'operations SQL traitee comme une unite de travail atomique. Soit toutes les operations sont executees, soit aucune ne l'est.

1.1 Proprietes ACID des Transactions

- **A**tomicite : Tout ou rien
- **C**ohérence : Respect des contraintes
- **I**solation : Execution independante
- **D**urabilite : Resultats permanents

1.2 COMMIT

Important

COMMIT valide definitivement toutes les modifications effectuees depuis le debut de la transaction. Apres un COMMIT, les changements sont permanents et visibles par les autres utilisateurs.

Exemple

```
-- Transaction complete
UPDATE comptes SET solde = solde - 1000 WHERE id = 1;
UPDATE comptes SET solde = solde + 1000 WHERE id = 2;
COMMIT; -- Validation definitive
```

1.3 ROLLBACK

Important

ROLLBACK annule toutes les modifications effectuees depuis le debut de la transaction ou depuis le dernier SAVEPOINT.

Exemple

```
-- Exemple avec rollback
DELETE FROM employes WHERE departement = 'RH';
-- Realisation qu'on a fait une erreur
ROLLBACK; -- Annulation de la suppression
```

1.4 SAVEPOINT

Important

Les **SAVEPOINT** permettent de creer des points de restauration intermediaires dans une transaction.

Exemple

```
-- Utilisation des savepoints
INSERT INTO commandes (id, client, montant)
VALUES (1001, 'Dupont', 1500);
SAVEPOINT sp1;

UPDATE inventaire SET quantite = quantite - 1
WHERE produit_id = 50;
-- Verification de la disponibilite
IF quantite < 0 THEN
ROLLBACK TO sp1; -- Annule seulement l'update
END IF;

COMMIT; -- Valide l'insertion
```

1.5 Transactions Implicites et Explicites

- **Transaction implicite** : Tout DDL (CREATE, ALTER, DROP) execute un COMMIT automatique
- **Transaction explicite** : Debutee par le premier DML et terminee par COMMIT/ROLLBACK

Exemple

```
-- Transaction explicite
UPDATE produits SET prix = prix * 1.1;
SAVEPOINT avant_suppression;
DELETE FROM produits WHERE discontinued = 1;
-- Decision conditionnelle
IF SQL%ROWCOUNT > 100 THEN
ROLLBACK TO avant_suppression;
END IF;
COMMIT;
```

Chapter 2

Langage de Definition des Donnees (LDD)

Definition

Le **Langage de Definition des Donnees (LDD)** permet de creer, modifier et supprimer les objets de la base de donnees (tables, vues, index, etc.).

2.1 CREATE TABLE

Important

La commande **CREATE TABLE** cree une nouvelle table avec ses colonnes, types de donnees et contraintes.

Exemple

```
-- Creation d'une table complete
CREATE TABLE employes (
id NUMBER(6) PRIMARY KEY,
nom VARCHAR2(50) NOT NULL,
prenom VARCHAR2(50),
email VARCHAR2(100) UNIQUE,
salaire NUMBER(8,2) DEFAULT 0 CHECK(salaire >= 0),
date_embauche DATE DEFAULT SYSDATE,
departement_id NUMBER(4),
CONSTRAINT fk_emp_dept
FOREIGN KEY (departement_id)
REFERENCES departements(id)
);
```

2.2 Types de Donnees Oracle

Type	Description
NUMBER(p,s)	Numerique avec precision et echelle
VARCHAR2(n)	Chaine de caracteres variable
CHAR(n)	Chaine de caracteres fixe
DATE	Date et heure
TIMESTAMP	Date/heure avec precision nanoseconde
CLOB	Texte de grande taille
BLOB	Donnees binaires

2.3 ALTER TABLE

Important

ALTER TABLE modifie la structure d'une table existante.

Exemple

```sql
-- Ajouter une colonne
ALTER TABLE employes ADD (
telephone VARCHAR2(20),
date_naissance DATE
);

-- Modifier une colonne
ALTER TABLE employes MODIFY (
nom VARCHAR2(100) NOT NULL,
email VARCHAR2(150)
);

-- Supprimer une colonne
ALTER TABLE employes DROP COLUMN telephone;

-- Renommer une colonne
ALTER TABLE employes RENAME COLUMN date_embauche TO embauche_date;
```

2.4 DROP TABLE

Important

DROP TABLE supprime definitivement une table et ses donnees.

Exemple

```sql
-- Suppression simple
DROP TABLE employes;

-- Suppression avec verification
DROP TABLE employes CASCADE CONSTRAINTS;

-- Suppression si existe (Oracle 23c+)
DROP TABLE IF EXISTS employes;
```

2.5 CTAS : CREATE TABLE AS SELECT

Important

CTAS cree une nouvelle table a partir du resultat d'une requete SELECT.

Exemple

```
-- Copie structure et donnees
CREATE TABLE employes_backup AS
SELECT * FROM employes;

-- Creation avec filtre
CREATE TABLE employes_rh AS
SELECT id, nom, prenom, salaire
FROM employes
WHERE departement_id = 10;

-- Creation avec agregation
CREATE TABLE stats_departements AS
SELECT departement_id,
COUNT(*) as nb_employes,
AVG(salaire) as salaire_moyen,
MAX(salaire) as salaire_max
FROM employes
GROUP BY departement_id;
```

2.6 TRUNCATE TABLE

Important

TRUNCATE TABLE vide une table rapidement sans journalisation detaillee. Operation DDL (COMMIT automatique).

Exemple

```
-- Vider une table
TRUNCATE TABLE employes_temp;

-- Vider avec reinitialisation des sequences
TRUNCATE TABLE employes_temp REUSE STORAGE;
```

Chapter 3

Langage de Manipulation des Donnees (LMD)

Definition

Le **Langage de Manipulation des Donnees (LMD)** permet d'ajouter, modifier et supprimer des donnees dans les tables.

3.1 INSERT

Important

INSERT ajoute de nouvelles lignes dans une table.

Exemple

```sql
-- Insertion simple
INSERT INTO employes (id, nom, prenom, salaire, departement_id)
VALUES (1001, 'Martin', 'Pierre', 3500, 10);

-- Insertion avec valeurs par defaut
INSERT INTO employes (id, nom, prenom, date_embauche)
VALUES (1002, 'Dubois', 'Marie', DEFAULT);

-- Insertion multiple (Oracle 23c+)
INSERT ALL
INTO employes VALUES (1003, 'Leclerc', 'Paul', 3200, 20)
INTO employes VALUES (1004, 'Moreau', 'Sophie', 3800, 10)
INTO employes VALUES (1005, 'Garcia', 'Luc', 4000, 30)
SELECT 1 FROM DUAL;

-- Insertion a partir d'une requete SELECT
INSERT INTO employes_archive (id, nom, prenom, salaire)
SELECT id, nom, prenom, salaire
FROM employes
WHERE date_embauche < DATE '2020-01-01';
```

3.2 UPDATE

Important

UPDATE modifie des donnees existantes dans une table.

Exemple

```
-- Augmentation generale de salaire
UPDATE employes
SET salaire = salaire * 1.05;

-- Augmentation ciblee
UPDATE employes
SET salaire = salaire * 1.10,
date_promotion = SYSDATE
WHERE departement_id = 10
AND performance = 'EXCELLENT';

-- Mise a jour avec sous-requete
UPDATE employes e
SET salaire = (
SELECT AVG(salaire) * 1.15
FROM employes
WHERE departement_id = e.departement_id
)
WHERE e.date_embauche < DATE '2018-01-01';

-- Mise a jour conditionnelle avec CASE
UPDATE produits
SET prix = CASE
WHEN categorie = 'LUXE' THEN prix * 1.20
WHEN categorie = 'STANDARD' THEN prix * 1.10
ELSE prix * 1.05
END,
date_maj = SYSDATE;
```

3.3 DELETE

Important

DELETE supprime des lignes d'une table selon une condition.

Exemple

```sql
-- Suppression conditionnelle
DELETE FROM employes
WHERE date_embauche < DATE '2010-01-01';

-- Suppression avec sous-requete
DELETE FROM employes
WHERE departement_id IN (
SELECT id
FROM departements
WHERE ville = 'Lyon'
);

-- Suppression de doublons
DELETE FROM employes e1
WHERE ROWID > (
SELECT MIN(ROWID)
FROM employes e2
WHERE e1.email = e2.email
);
```

3.4 Merge (UPSERT)

Important

MERGE combine INSERT, UPDATE et DELETE en une seule operation.

Exemple

```
-- Synchronisation de deux tables
MERGE INTO employes_courant c
USING employes_nouveaux n
ON (c.id = n.id)
WHEN MATCHED THEN
UPDATE SET c.nom = n.nom,
c.salaire = n.salaire,
c.departement_id = n.departement_id
DELETE WHERE n.statut = 'INACTIF'
WHEN NOT MATCHED THEN
INSERT (id, nom, salaire, departement_id)
VALUES (n.id, n.nom, n.salaire, n.departement_id);
```

3.5 Retour d'Information avec RETURNING

Important

RETURNING permet de recuperer des valeurs apres INSERT, UPDATE ou DELETE.

Exemple

```
-- Recuperation apres INSERT
DECLARE
new_id NUMBER;
new_nom VARCHAR2(50);
BEGIN
INSERT INTO employes (id, nom, salaire)
VALUES (seq_emp.NEXTVAL, 'Durand', 4000)
RETURNING id, nom INTO new_id, new_nom;

text
DBMS_OUTPUT.PUT_LINE('Nouvel employe: ' || new_id || ' - ' || new_nom
    );
END;
/

-- Recuperation apres UPDATE
UPDATE employes
SET salaire = salaire * 1.10
WHERE departement_id = 20
RETURNING id, salaire BULK COLLECT INTO tab_ids, tab_salaires;
```

Chapter 4

Introduction Generale au SQL et au SGBDR

4.1 Qu'est-ce que SQL ?

Definition

SQL (Structured Query Language) est le langage standardise permettant d'interagir avec un systeme de gestion de base de donnees relationnelle (SGBDR).

Composant	Fonction
DDL	Definition des objets (CREATE, ALTER, DROP)
DML	Manipulation des donnees (INSERT, UPDATE, DELETE)
DQL	Interrogation des donnees (SELECT)
TCL	Controle des transactions (COMMIT, ROLLBACK)
DCL	Controle des acces (GRANT, REVOKE)

4.2 Architecture d'Oracle Database

Important

Oracle utilise une architecture client-serveur avec instance et base de donnees.

- **Instance** : SGA + processus background
- **Base de donnees** : Fichiers physiques (.dbf, .log, .ctl)
- **Schema** : Collection d'objets appartenant a un utilisateur

- **Tablespace** : Espace de stockage logique

Exemple

```
-- Connexion a la base
CONNECT username/password@hostname:port/service_name

-- Voir l'instance
SELECT * FROM v$instance;

-- Voir les tablespaces
SELECT tablespace_name, status, contents
FROM dba_tablespaces;
```

4.3 Structure des Tables Relationnelles

Important

Une table est une collection de lignes et colonnes organisees relationnellement.

Exemple

```
-- Description d'une table
DESC employes;

-- Structure d'une ligne typique
-- +----+----------+-----------+---------+----------+
-- | ID | NOM | PRENOM | SALAIRE | DEPT_ID |
-- +----+----------+-----------+---------+----------+
-- | 1 | Martin | Pierre | 3500 | 10 |
-- | 2 | Dubois | Marie | 4200 | 20 |
-- +----+----------+-----------+---------+----------+
```

4.4 Environnements de Developpement

- **SQLPlus** : Interface ligne de commande
- **SQL Developer** : IDE graphique gratuit
- **iSQLPlus** : Interface web (depreciee)

Chapter 5

Instruction SELECT : Projection, Filtrage et Expressions

5.1 Syntaxe Generale Complete

Important

Ordre d'execution : FROM -> WHERE -> GROUP BY -> HAVING -> SELECT -> ORDER BY

Exemple

```
SELECT [DISTINCT] colonnes, expressions
FROM tables
[WHERE conditions]
[GROUP BY colonnes]
[HAVING conditions_agregation]
[ORDER BY colonnes [ASC|DESC]];
```

5.2 Projection des Colonnes

Exemple

```sql
-- Projection specifique
SELECT nom, prenom, salaire FROM employes;

-- Toutes les colonnes
SELECT * FROM employes;

-- Colonnes calculees
SELECT nom, salaire, salaire * 12 AS salaire_annuel
FROM employes;

-- Expressions complexes
SELECT nom,
salaire,
salaire * 1.10 AS augmentation,
(salaire * 12) / 365 AS salaire_journalier
FROM employes;
```

5.3 Alias et Formatage

Important

Les alias ameliorent la lisibilite des resultats.

Exemple

```sql
-- Alias simples
SELECT nom AS "Nom de famille",
prenom AS "Prenom",
salaire AS "Salaire Mensuel"
FROM employes;

-- Alias avec calculs
SELECT nom || ' ' || prenom AS "Nom Complet",
salaire * 12 AS "Remuneration Annuelle",
ROUND(salaire / 160, 2) AS "Taux Horaire"
FROM employes;

-- Utilisation de DUAL pour les calculs
SELECT 10 * 5 AS calcul,
SYSDATE AS aujourdhui,
USER AS utilisateur
FROM DUAL;
```

5.4 Concatenation et Chaines

Exemple

```sql
-- Concatenation basique
SELECT prenom || ' ' || nom AS "Nom Complet"
FROM employes;

-- Concatenation avec formatage
SELECT 'Employe: ' || prenom || ' ' || nom ||
' - Salaire: ' || TO_CHAR(salaire, '99,999.00')
AS "Fiche Employe"
FROM employes;

-- Utilisation de CONCAT (limite a 2 parametres)
SELECT CONCAT(prenom, CONCAT(' ', nom)) AS nom_complet
FROM employes;
```

5.5 DISTINCT et UNIQUE

Important

Elimine les doublons du resultat.

Exemple

```sql
-- Valeurs distinctes simples
SELECT DISTINCT departement_id FROM employes;

-- Combinaisons distinctes
SELECT DISTINCT departement_id, poste
FROM employes;

-- DISTINCT avec expressions
SELECT DISTINCT UPPER(poste)
FROM employes;

-- Equivalent avec UNIQUE
SELECT UNIQUE departement_id FROM employes;
```

Chapter 6

Filtrage des Donnees : La Clause WHERE

6.1 Operateurs de Comparaison

Operateur	Description
=	Egal a
<> ou !=	Different de
	Superieur a
<	Inferieur a
=	Superieur ou egal
<=	Inferieur ou egal

Exemple

```
-- Egalite
SELECT * FROM employes WHERE salaire = 3000;

-- Inegalite
SELECT * FROM employes WHERE departement_id != 10;

-- Comparaisons numeriques
SELECT * FROM employes WHERE salaire > 4000;
SELECT * FROM employes WHERE salaire BETWEEN 3000 AND 5000;
```

6.2 BETWEEN et NOT BETWEEN

Important

Inclusif : BETWEEN A AND B inclut A et B.

Exemple

```
-- Plage inclusive
SELECT * FROM employes
WHERE salaire BETWEEN 3000 AND 5000;

-- Plage de dates
SELECT * FROM employes
WHERE date_embauche BETWEEN DATE '2020-01-01' AND SYSDATE;

-- En dehors d'une plage
SELECT * FROM employes
WHERE salaire NOT BETWEEN 2000 AND 4000;
```

6.3 IN et NOT IN

Exemple

```
-- Liste de valeurs
SELECT * FROM employes
WHERE departement_id IN (10, 20, 30);

-- Liste de textes
SELECT * FROM employes
WHERE poste IN ('DEVELOPPEUR', 'ANALYSTE', 'MANAGER');

-- Exclusion par liste
SELECT * FROM employes
WHERE departement_id NOT IN (10, 90);

-- IN avec sous-requete
SELECT * FROM employes
WHERE departement_id IN (
SELECT id FROM departements WHERE ville = 'Paris'
);
```

6.4 LIKE et Recherche de Modeles

Caractere	Description
_	Exactement un caractere

Exemple

```
-- Commence par
SELECT * FROM employes WHERE nom LIKE 'D%';

-- Termine par
SELECT * FROM employes WHERE nom LIKE '%son';

-- Contient
SELECT * FROM employes WHERE nom LIKE '%art%';

-- Modele precis
SELECT * FROM employes WHERE nom LIKE 'D_b%';

-- Recherche insensible a la casse
SELECT * FROM employes WHERE UPPER(nom) LIKE UPPER('%martin%');

-- Echappement du caractere _
SELECT * FROM produits WHERE code LIKE 'A_%' ESCAPE '';
```

6.5 Gestion des NULL

Important

NULL represente une valeur inconnue. Les comparaisons avec NULL donnent toujours NULL.

Exemple

```sql
-- Valeurs NULL
SELECT * FROM employes WHERE commission_pct IS NULL;

-- Valeurs non NULL
SELECT * FROM employes WHERE commission_pct IS NOT NULL;

-- Attention aux pieges avec NOT IN et NULL
SELECT * FROM employes
WHERE departement_id NOT IN (10, 20, NULL); -- Retourne 0 lignes!

-- Solution
SELECT * FROM employes
WHERE departement_id NOT IN (10, 20)
OR departement_id IS NULL;
```

6.6 Operateurs Logiques et Priorite

Important

Priorite : NOT -> AND -> OR. Utilisez les parentheses pour clarifier.

Exemple

```
-- AND a priorite sur OR
SELECT * FROM employes
WHERE departement_id = 10
AND salaire > 3000
OR departement_id = 20; -- Equivaut a (dept=10 AND salaire>3000) OR
    dept=20

-- Clarification avec parentheses
SELECT * FROM employes
WHERE departement_id = 10
AND (salaire > 3000 OR commission_pct > 0.1);

-- Combinaison complexe
SELECT * FROM employes
WHERE (departement_id IN (10, 20) AND salaire BETWEEN 3000 AND 6000)
OR (departement_id = 30 AND date_embauche > DATE '2022-01-01');
```

6.7 ORDER BY et Tri

Exemple

```sql
-- Tri simple
SELECT * FROM employes ORDER BY nom;

-- Tri multiple
SELECT * FROM employes
ORDER BY departement_id ASC, salaire DESC;

-- Tri par expression
SELECT nom, salaire, salaire * 12 AS annuel
FROM employes
ORDER BY annuel DESC;

-- Tri par position (deconseille)
SELECT nom, prenom, salaire
FROM employes
ORDER BY 3 DESC, 1 ASC;

-- Tri avec NULLS FIRST/LAST
SELECT nom, commission_pct
FROM employes
ORDER BY commission_pct NULLS LAST;
```

Chapter 7

Fonctions SQL : Texte, Nombres, Dates, Conversions

7.1 Fonctions de Chaines de Caracteres

Fonction	**Description**
UPPER/LOWER/INITCAP	Changement de casse
LENGTH/LENGTHB	Longueur en caracteres/octets
SUBSTR	Extraction de sous-chaine
INSTR	Position d'une sous-chaine
REPLACE	Remplacement
TRIM/LTRIM/RTRIM	Suppression d'espaces
LPAD/RPAD	Remplissage
CONCAT	Concatenation

Exemple

```sql
-- Manipulation de texte
SELECT
nom,
UPPER(nom) AS majuscule,
LOWER(nom) AS minuscule,
INITCAP(nom) AS initiale_maj,
LENGTH(nom) AS longueur,
SUBSTR(nom, 1, 3) AS trois_premieres,
INSTR(nom, 'a') AS position_a,
REPLACE(nom, 'a', '4') AS remplace
FROM employes;

-- Nettoyage de donnees
SELECT
TRIM(' Hello ') AS trim_exemple,
LTRIM('xxxHello', 'x') AS ltrim_exemple,
RPAD(nom, 20, '.') AS nom_rempli
FROM employes;
```

7.2 Fonctions Numeriques

Fonction	Description
ROUND	Arrondi
TRUNC	Troncature
CEIL/CEILING	Plus petit entier superieur
FLOOR	Plus grand entier inferieur
MOD	Modulo
POWER	Puissance
SQRT	Racine carree
ABS	Valeur absolue
SIGN	Signe

Exemple

```sql
-- Calculs numeriques
SELECT
salaire,
ROUND(salaire, -2) AS arrondi_100,
TRUNC(salaire, -2) AS tronque_100,
CEIL(salaire/100)*100 AS plafond_100,
FLOOR(salaire/100)*100 AS plancher_100,
MOD(salaire, 1000) AS modulo_1000,
POWER(2, 3) AS puissance,
SQRT(salaire) AS racine_salaire
FROM employes;

-- Expressions complexes
SELECT
salaire,
ROUND(salaire * 1.15, 2) AS augmentation_15pct,
MOD(ROUND(salaire), 100) AS centimes_arrondis
FROM employes;
```

7.3 Fonctions de Date

Important

Oracle stocke les dates avec le composant heure.

Exemple

```sql
-- Dates systeme
SELECT
SYSDATE AS maintenant,
CURRENT_DATE AS date_session,
CURRENT_TIMESTAMP AS timestamp_session
FROM DUAL;

-- Manipulation de dates
SELECT
date_embauche,
ADD_MONTHS(date_embauche, 6) AS date_revue,
MONTHS_BETWEEN(SYSDATE, date_embauche) AS mois_anciennete,
NEXT_DAY(date_embauche, 'LUNDI') AS prochain_lundi,
LAST_DAY(date_embauche) AS fin_mois,
EXTRACT(YEAR FROM date_embauche) AS annee_embauche
FROM employes;

-- Calculs de duree
SELECT
nom,
date_embauche,
TRUNC(MONTHS_BETWEEN(SYSDATE, date_embauche)/12) AS annees,
TRUNC(MOD(MONTHS_BETWEEN(SYSDATE, date_embauche), 12)) AS mois
FROM employes;
```

7.4 Fonctions de Conversion

Exemple

```
-- Conversion de dates
SELECT
date_embauche,
TO_CHAR(date_embauche, 'DD/MM/YYYY') AS date_fr,
TO_CHAR(date_embauche, 'Day, Month YYYY') AS date_longue,
TO_CHAR(date_embauche, 'YYYY-MM-DD HH24:MI:SS') AS date_complete
FROM employes;

-- Conversion numerique
SELECT
salaire,
TO_CHAR(salaire, '99,999.00') AS salaire_format,
TO_CHAR(salaire, 'C999G999D00') AS salaire_monnaie,
TO_NUMBER('1234.56') AS nombre_converti
FROM employes;

-- Conversion date depuis chaine
SELECT
TO_DATE('15/03/2023', 'DD/MM/YYYY') AS date_from_string,
TO_TIMESTAMP('2023-03-15 14:30:25', 'YYYY-MM-DD HH24:MI:SS') AS
   timestamp_from_string
FROM DUAL;
```

7.5 Gestion des NULL

Exemple

```sql
-- NVL : remplace NULL par valeur
SELECT
nom,
commission_pct,
NVL(commission_pct, 0) AS commission_sans_null,
salaire * (1 + NVL(commission_pct, 0)) AS salaire_total
FROM employes;

-- NVL2 : valeur differente selon NULL ou non
SELECT
nom,
commission_pct,
NVL2(commission_pct, 'Avec commission', 'Sans commission') AS
    statut_commission
FROM employes;

-- COALESCE : premier non-NULL
SELECT
nom,
COALESCE(commission_pct, prime_annuelle, 0) AS remuneration_variable
FROM employes;

-- DECODE et CASE pour logique conditionnelle
SELECT
nom,
salaire,
CASE
WHEN salaire < 3000 THEN 'Junior'
WHEN salaire BETWEEN 3000 AND 6000 THEN 'Intermediaire'
ELSE 'Senior'
END AS categorie_salaire
FROM employes;
```

Chapter 8

Les Sous-interrogations SQL

Definition

Une **sous-requete** (subquery) est une requete SQL imbriquee dans une autre requete. Elle peut etre utilisee dans SELECT, FROM, WHERE, et HAVING.

8.1 Sous-requete Scalaire

Important

Retourne une seule valeur (une ligne, une colonne). Utilisable partout ou une expression est attendue.

Exemple

```
    -- Employes avec salaire superieur a la moyenne
    SELECT nom, salaire
    FROM employes
    WHERE salaire > (SELECT AVG(salaire) FROM employes);

    -- Pourcentage du salaire par rapport au maximum
    SELECT nom, salaire,
    (salaire / (SELECT MAX(salaire) FROM employes)) * 100 as
   pourcentage_max
    FROM employes;

```

8.2 Sous-requetes Multivaluees avec IN

Important

Retourne plusieurs valeurs dans une seule colonne. Utilisee avec IN, NOT IN.

Exemple

```
    -- Employes des departements bases a Paris
    SELECT nom, prenom
    FROM employes
    WHERE departement_id IN (
    SELECT id
    FROM departements
    WHERE ville = 'Paris'
    );

    -- Employes n'ayant jamais ete managers
    SELECT nom, prenom
    FROM employes
    WHERE id NOT IN (
    SELECT DISTINCT manager_id
    FROM employes
    WHERE manager_id IS NOT NULL
    );

```

8.3 Sous-requetes avec ANY/ALL

Important

ANY : au moins une valeur satisfait la condition **ALL** : toutes les valeurs satisfont la condition

Exemple

```
    -- Employes avec salaire > au minimum des managers
    SELECT nom, salaire
    FROM employes
    WHERE salaire > ANY (
    SELECT salaire
    FROM employes
    WHERE poste = 'MANAGER'
    );

    -- Employes avec salaire > a tous les developpeurs
    SELECT nom, salaire
    FROM employes
    WHERE salaire > ALL (
    SELECT salaire
    FROM employes
    WHERE poste = 'DEVELOPPEUR'
    );

```

8.4 Sous-requetes Corrélées

Important

La sous-requete corrélée reference des colonnes de la requete externe. Executee une fois pour chaque ligne de la requete principale.

Exemple

```
    -- Employes avec salaire superieur a la moyenne de leur
departement
    SELECT e.nom, e.salaire, e.departement_id
    FROM employes e
    WHERE e.salaire > (
    SELECT AVG(salaire)
    FROM employes
    WHERE departement_id = e.departement_id
    );

    -- Dernier employe embauche dans chaque departement
    SELECT d.nom_dept, e.nom, e.date_embauche
    FROM departements d
    JOIN employes e ON d.id = e.departement_id
    WHERE e.date_embauche = (
    SELECT MAX(date_embauche)
    FROM employes
    WHERE departement_id = d.id
    );

```

8.5 Vue en Ligne (Inline View)

Important

Une sous-requete dans la clause FROM, traitee comme une table temporaire.

Exemple

```
-- Top 3 des departements avec le salaire moyen le plus eleve
SELECT *
FROM (
SELECT departement_id,
AVG(salaire) as salaire_moyen,
COUNT(*) as nb_employes
FROM employes
GROUP BY departement_id
ORDER BY salaire_moyen DESC
)
WHERE ROWNUM <= 3;

-- Employes avec leur classement salarial
SELECT e.nom, e.salaire, s.classement
FROM employes e
JOIN (
SELECT id,
RANK() OVER (ORDER BY salaire DESC) as classement
FROM employes
) s ON e.id = s.id
WHERE s.classement <= 10;

```

8.6 EXISTS et NOT EXISTS

Important

Verifie l'existence de lignes dans une sous-requete. Tres efficace pour les relations.

Exemple

```
    -- Departements ayant au moins un employe
    SELECT nom
    FROM departements d
    WHERE EXISTS (
    SELECT 1
    FROM employes e
    WHERE e.departement_id = d.id
    );

    -- Departements sans employes
    SELECT nom
    FROM departements d
    WHERE NOT EXISTS (
    SELECT 1
    FROM employes e
    WHERE e.departement_id = d.id
    );

    -- Clients ayant commande un produit specifique
    SELECT nom_client
    FROM clients c
    WHERE EXISTS (
    SELECT 1
    FROM commandes cmd
    JOIN details_commande det ON cmd.id = det.commande_id
    WHERE cmd.client_id = c.id
    AND det.produit_id = 1001
    );

```

Chapter 9

Les Jointures SQL

9.1 Inner Join Classique

Important

Retourne seulement les lignes avec correspondance dans les deux tables.

Exemple

```sql
-- Jointure simple
SELECT e.nom, e.prenom, d.nom_dept
FROM employes e
INNER JOIN departements d ON e.departement_id = d.id;

-- Jointure multiple
SELECT e.nom, d.nom_dept, p.nom_projet
FROM employes e
JOIN departements d ON e.departement_id = d.id
JOIN projets p ON e.id = p.responsable_id;

-- Jointure avec conditions supplementaires
SELECT e.nom, d.nom_dept
FROM employes e
JOIN departements d ON e.departement_id = d.id
WHERE d.ville = 'Paris'
AND e.salaire > 4000;
```

9.2 Outer Joins

Important

Inclut les lignes sans correspondance.

Exemple

```sql
-- LEFT JOIN : tous les employes meme sans departement
SELECT e.nom, d.nom_dept
FROM employes e
LEFT JOIN departements d ON e.departement_id = d.id;

-- RIGHT JOIN : tous les departements meme sans employes
SELECT e.nom, d.nom_dept
FROM employes e
RIGHT JOIN departements d ON e.departement_id = d.id;

-- FULL OUTER JOIN : tous les employes et tous les departements
SELECT e.nom, d.nom_dept
FROM employes e
FULL OUTER JOIN departements d ON e.departement_id = d.id;

-- Identification des lignes sans correspondance
SELECT e.nom, d.nom_dept
FROM employes e
LEFT JOIN departements d ON e.departement_id = d.id
WHERE d.id IS NULL; -- Employes sans departement
```

9.3 Self Join

Important

Jointure d'une table avec elle-meme, utile pour les hierarchies.

Exemple

```
-- Hierarchie employe-manager
SELECT e.nom AS employe, m.nom AS manager
FROM employes e
LEFT JOIN employes m ON e.manager_id = m.id;

-- Structure hierarchique complete
SELECT
LPAD(' ', (LEVEL-1)*2) || e.nom AS organigramme,
e.poste
FROM employes e
CONNECT BY PRIOR e.id = e.manager_id
START WITH e.manager_id IS NULL;
```

9.4 JOIN avec USING

Important

Simplifie la syntaxe quand les colonnes de jointure ont le meme nom.

Exemple

```
-- Avec USING
SELECT e.nom, d.nom_dept
FROM employes e
JOIN departements d USING (departement_id);

-- Equivalent a ON
SELECT e.nom, d.nom_dept
FROM employes e
JOIN departements d ON e.departement_id = d.departement_id;
```

9.5 Natural Join

Important

Jointure automatique sur toutes les colonnes de meme nom. A utiliser avec prudence.

Exemple

```
-- Natural JOIN (dangereux!)
SELECT e.nom, d.nom_dept
FROM employes e
NATURAL JOIN departements d;

-- Mieux vaut etre explicite
SELECT e.nom, d.nom_dept
FROM employes e
JOIN departements d ON e.departement_id = d.id;
```

9.6 Jointures Non-Equivalentes

Exemple

```
-- Jointure sur une plage de valeurs
SELECT e.nom, e.salaire, g.niveau_grade
FROM employes e
JOIN grades_salaires g ON e.salaire BETWEEN g.salaire_min AND g.
   salaire_max;

-- Jointure croisee (CROSS JOIN)
SELECT e.nom, p.nom_projet
FROM employes e
CROSS JOIN projets p
WHERE e.departement_id = p.departement_id;
```

Chapter 10

GROUP BY, Agregations et HAVING

10.1 Fonctions d'Agregation de Base

Fonction	**Description**
COUNT	Nombre de lignes
SUM	Somme des valeurs
AVG	Moyenne des valeurs
MIN	Valeur minimale
MAX	Valeur maximale

Exemple

```sql
-- Statistiques de base
SELECT
COUNT(*) AS total_employes,
AVG(salaire) AS salaire_moyen,
MIN(salaire) AS salaire_min,
MAX(salaire) AS salaire_max,
SUM(salaire) AS masse_salariale
FROM employes;

-- COUNT avec DISTINCT
SELECT
COUNT(DISTINCT departement_id) AS nb_departements,
COUNT(DISTINCT poste) AS nb_posts
FROM employes;

-- Gestion des NULL dans les agregations
SELECT
AVG(NVL(commission_pct, 0)) AS commission_moyenne,
COUNT(commission_pct) AS nb_avec_commission -- Ignore les NULL
FROM employes;
```

10.2 GROUP BY Simple

Important

GROUP BY permet d'appliquer des fonctions d'agregation par groupe.

Exemple

```sql
-- Statistiques par departement
SELECT
departement_id,
COUNT(*) AS nb_employes,
AVG(salaire) AS salaire_moyen,
MIN(salaire) AS salaire_min,
MAX(salaire) AS salaire_max
FROM employes
GROUP BY departement_id;

-- Groupement multiple
SELECT
departement_id,
poste,
COUNT(*) AS nb_employes,
AVG(salaire) AS salaire_moyen
FROM employes
GROUP BY departement_id, poste
ORDER BY departement_id, poste;

-- Groupement avec expressions
SELECT
EXTRACT(YEAR FROM date_embauche) AS annee_embauche,
COUNT(*) AS nb_embauches
FROM employes
GROUP BY EXTRACT(YEAR FROM date_embauche)
ORDER BY annee_embauche;
```

10.3 HAVING vs WHERE

Important

WHERE filtre les lignes avant regroupement, HAVING filtre apres regroupement.

Exemple

```sql
-- HAVING pour filtrer les groupes
SELECT
departement_id,
AVG(salaire) AS salaire_moyen,
COUNT() AS nb_employes
FROM employes
GROUP BY departement_id
HAVING AVG(salaire) > 5000 AND COUNT() >= 5;

-- WHERE + HAVING
SELECT
departement_id,
AVG(salaire) AS salaire_moyen
FROM employes
WHERE date_embauche > DATE '2020-01-01' -- Filtre avant regroupement
GROUP BY departement_id
HAVING AVG(salaire) > 4000; -- Filtre apres regroupement

-- Comparaison WHERE vs HAVING
-- WHERE sur colonne non agregee
-- HAVING sur resultat d'agregation
```

10.4 Agregations Avancees

Exemple

```sql
-- ROLLUP pour les sous-totaux
SELECT
departement_id,
poste,
COUNT(*) AS nb_employes,
SUM(salaire) AS total_salaire
FROM employes
GROUP BY ROLLUP(departement_id, poste);

-- CUBE pour toutes les combinaisons
SELECT
departement_id,
poste,
COUNT(*) AS nb_employes
FROM employes
GROUP BY CUBE(departement_id, poste);

-- GROUPING SETS pour des regroupements specifiques
SELECT
departement_id,
poste,
COUNT(*) AS nb_employes
FROM employes
GROUP BY GROUPING SETS (
(departement_id, poste),
(departement_id),
(poste),
()
);
```

10.5 Fonctions Analytiques (Window Functions)

Important

Permettent des calculs sur un ensemble de lignes sans regrouper.

Exemple

```sql
-- Classement des salaires par departement
SELECT
nom,
departement_id,
salaire,
RANK() OVER (PARTITION BY departement_id ORDER BY salaire DESC) AS
   rang_salaire,
AVG(salaire) OVER (PARTITION BY departement_id) AS moyenne_dept
FROM employes;

-- Cumul des salaires
SELECT
nom,
salaire,
SUM(salaire) OVER (ORDER BY salaire) AS cumul_salaire,
ROUND(salaire * 100.0 / SUM(salaire) OVER (), 2) AS pourcentage_total
FROM employes
ORDER BY salaire;
```