Loric Informatique

Oracle Database (SQL)

Version 1.0 (créé le 10/08/2023, modifié le 28/01/2024)

SQL (Structured Query Language) est un langage servant à exploiter des bases de données relationnelles.

Oracle Database est souvent utilisé pour les bases de données de sites web qui utilisent le langage PHP, et très souvent utilisé pour les bases de données des applications utilisant le langage Java.

Toutes les options facultatives seront représentées par des [].

Outils nécessaires :

- SQL Developer (contenant Oracle si vous avez un serveur simple hébergé)

Table des matières :

I. Bases
- **I.1. Syntaxe**
- **I.2. Types de variables**
  - **I.2.1. Chaînes de caractères**
  - **I.2.2. Numériques**
  - **I.2.3. Autres types de variables**
- **I.3. Conditions**
- **I.4. Commentaires**

II. Les fonctions
- **II.1. Les fonctions arithmétiques**
- **II.2. Les fonctions pour les chaînes de caractères**
- **II.3. Les fonctions pour les dates**
  - **II.3.1. Les masques pour les dates**
  - **II.3.2. Les fonctions de manipulation des dates**
- **II.4. Autres fonctions**
- **II.5. Fonctions d'agrégat (pour des sélections uniquement, ne concernent pas les conditions)**

III. Requêtes d'initialisation de la base de données
- **III.1. Créer une table (tableau de valeurs)**
- **III.2. Les contraintes**
  - **III.2.1. Ajouter une contrainte (exemple avec une clé primaire)**
  - **III.2.2. Supprimer une contrainte (exemple avec une clé primaire)**
  - **III.2.3. Activer/Désactiver une contrainte d'intégrité (exemple avec une clé primaire)**
- **III.3. Renommer une table**
- **III.4. Supprimer une table**
- **III.5. Copier une table**

IV. Requêtes de mise à jour de la table
IV.1. Insérer un enregistrement (ligne)
IV.2. Mettre à jour un ou plusieurs enregistrement(s)
IV.3. Supprimer un ou plusieurs enregistrement(s)
IV.4. Modifier la structure de la table
IV.4.1. Ajouter une colonne
IV.4.2. Supprimer une colonne
IV.4.3. Modifier le type d'une colonne
IV.4.4. Modifier le nom d'une colonne
V. Requêtes de sélection de données
V.1. Sélection de tous les enregistrements de la table
V.2. Sélection des enregistrements de la table respectant la condition
V.3. Sélection des enregistrements de la table dans un ordre spécifique
V.4. Sélection des n premiers enregistrements de la table
V.5. Sélection des attributs de la table avec des enregistrements respectant la condition
V.6. Sélection d'un attribut de la table sans répétitions d'enregistrements respectant la condition
V.7. Les jointures (sélection des attributs de la table avec une ou plusieurs jointure(s) à une autre table)
V.7.1. Jointure interne simple
V.7.2. Jointure externe gauche (avec des enregistrements de la table de gauche non présents dans la table de droite)
V.7.3. Jointure externe droite (avec des enregistrements de la table de droite non présents dans la table de gauche)
V.7.4. Jointure externe entière (avec des tous les enregistrements non présents dans une autre table)
V.7.5. Jointure naturelle (pour deux tables avec une colonne du même nom)
V.7.6. Produit cartésien (retourne chaque ligne d'une table avec chaque ligne d'une autre table)
V.8. Utilisation d'une fonction d'agrégat
V.9. Les alias
V.9.1. Alias sur les tables (pour faciliter les jointures)
V.9.2. Alias sur une fonction d'agrégat (pour nommer une colonne sans nom)
V.10. Les groupes
V.11. Sélection des enregistrements d'un attribut présents dans la première table ou la deuxième ou les deux (union)
V.12. Sélection des enregistrements d'un attribut présents dans la première table, mais pas la deuxième (différence)
V.13. Sélection des enregistrements d'un attribut présents dans la première table et la deuxième (intersection)
VIII. Autres requêtes de données
VIII.1. Requête de description d'une table
VIII.2. Les transactions
VII. PL/SQL (sur Oracle uniquement)
1. Structure
2. Déclarations (dans le bloc DECLARE)
A. Variables de types simples
B. Variables constantes
C. Variables de types d'une colonne
D. Variables de types d'une ligne entière d'une table
E. Curseurs explicites
a. Curseur sans paramètres (c_curseur)
b. Curseur avec paramètres (c_curseur(parametres))
c. Curseurs avec verrouillage des données (pour une mise à jour de la table notamment)
F. Exceptions
a. Exception simple
b. Exception créée à l'aide d'un numéro d'exception

3. Opération d'affectation
A. Affectation simple
B. Affectation avec une requête de sélection (ne fonctionne que si la sélection ne renvoie qu'une seule valeur)
4. Conditions
5. Boucles
6. Les curseurs implicites et explicites
A. Les curseurs implicites
B. Les curseurs explicites
7. Les procédures
A. Créer une procédure
B. Retourner une valeur de la procédure
C. Faire un appel à la procédure
D. Supprimer une procédure
8. Les fonctions
A. Créer une fonction
B. Retourner une valeur de la fonction
C. Faire un appel à la fonction
D. Supprimer une fonction
9. Les vues
A. Créer une vue
B. Faire un appel à la vue
C. Supprimer une vue
10. Les CTE (Common Table Expression)
11. Les exceptions (dans le bloc EXCEPTION)
A. Exception prédéfinies
B. Provoquer une erreur personnalisée
C. Gestion des erreurs (dans le bloc EXCEPTION)
12. Les instructions de base

I. Bases

I.1. Syntaxe

```
requête1;
requête2;
requête3;
```

I.2. Types de variables

I.2.1. Chaînes de caractères

Type	Description
CHAR(X)	Longueur fixée à X caractères (maximum : 255)
VARCHAR2(X)	Longueur inférieure ou égale à X caractères (maximum : 255)

I.2.2. Numériques

Type	Description
NUMBER(X)	Longueur inférieure ou égale à X caractères
NUMBER(X, Y)	Longueur inférieure ou égale à X chiffres dans la partie entière et inférieure ou égale à Y chiffres après la virgule

I.2.3. Autres types de variables

Type	Description
BOOLEAN	0 = FALSE, 1 = TRUE
DATE	Format : 'DD/MM/YYYY' (utiliser TO_DATE() pour une meilleure compatibilité. Voir les fonctions pour les dates)

I.3. Conditions

Opérateurs de comparaisons	= != (ou ‹›) ‹ ‹= › ›=
Connecteurs logiques	OR, AND
Opérateur de négation	NOT
Opérateurs mathématiques	+ - * /
Comparaison logique	IS [NOT] {TRUE|FALSE|UNKNOWN}
Comparaison avec valeur	IS [NOT] NULL
Intervalle	*valeur* BETWEEN *borne_basse* AND *borne_haute*
Comparaison à une liste de valeurs	*valeur* [NOT] IN (*liste*)
Comparaison à toutes les valeurs de la liste	ALL (*liste*)
Comparaison à une valeur contenue dans la liste	ANY (*liste*)
Comparaison avec une chaîne de caractère en MAJUSCULE	UPPER(*valeur*) = '*CHAINE*'
Comparaison avec une chaîne de caractère en minuscule	LOWER(*valeur*) = '*chaine*'

Comparaison avec une chaîne de caractère avec une majuscule au début et le reste en minuscules	`INITCAP(valeur) = 'Chaine'`

I.4. Commentaires

```
-- Commentaire tenant sur une ligne

/* Commentaire pouvant être sur une ou plusieurs lignes */
```

II. Les fonctions

II.1. Les fonctions arithmétiques

Fonction	Utilité
`ROUND(n, d)`	Arrondit *n* au réel à *d* chiffres après la virgule ou à l'entier si *d* n'est pas renseigné
`TRUNC(n, d)`	Tronque *n* à *d* chiffres après la virgule ou à 0 si *d* n'est pas renseigné
`POWER(n, m)`	Renvoie *n* à la puissance *m* (si *n* est négatif, *m* doit être un entier)
`CEIL(n)`	Renvoie un entier directement supérieur ou égal à *n*
`FLOOR(n)`	Renvoie un entier directement supérieur ou égal à *n* (partie entière de *n*)
`ABS(n)`	Renvoie la valeur absolue de *n*
`MOD(n, m)`	Renvoie le reste de la division de *n* par *m*
`SQRT(n)`	Renvoie la racine carrée
`SIGN(n)`	Renvoie -1 si *n* est négatif, 1 si *n* est positif et 0 si *n* égal à 0

II.2. Les fonctions pour les chaînes de caractères

Fonction	Utilité
`LENGTH(chaine)`	Renvoie la longueur de la chaine

SUBSTR(*chaine*, *debut*[, *longueur*])	Renvoie la position (en commençant à une certaine position et allant jusqu'à la longueur fixée ou à la fin)
UPPER(*chaine*)	Convertit en majuscule
LOWER(*chaine*)	Convertit en minuscule
INITCAP(*chaine*)	Met en majuscule la première lettre et en minuscule les autres
TRANSLATE(*chaine*, *c1*, *c2*)	Remplace chaque caractère *c1* par *c2* dans la chaîne
REPLACE(*chaine*, *ch1*, *ch2*)	Remplace chaque chaine *ch1* par *ch2* dans la chaîne

II.3. Les fonctions pour les dates

II.3.1. Les masques pour les dates

Masque	Description
'CC'	Siècle (ex : 21)
'YYYY'	Année (ex : 2024)
'Q'	Numéro du trimestre dans l'année (ex : 3)
'WW'	Numéro de la semaine dans l'année
'MM'	Numéro du mois (ex : 05)
'DDD'	Numéro du jour dans l'année (ex : 185)
'DD'	Numéro du jour dans le mois (ex : 27)
'D'	Numéro du jour dans la semaine (ex : 5) avec dimanche égal à 0
'HH' ou 'HH12'	Heure sur 12 heures (ex : 08)
'HH24'	Heure sur 24 heures (ex : 20)
'MI'	Minutes (ex : 05)
'S'	Secondes (ex : 30)

`'YEAR'`	Année en lettres
`'MONTH'`	Mois en lettres
`'MON'`	Mois abrégé
`'DAY'`	Jour en lettres
`'DY'`	Jour abrégé

II.3.2. Les fonctions de manipulation des dates

Fonction	Utilité
`SYSDATE`	Renvoie la date actuelle
`ADD_MONTHS(`*`date, nombre`*`)`	Ajoute ou soustrait un certain nombre de mois
`LAST_DAY(`*`date`*`)`	Expression de type date qui a pour valeur la date du dernier jour du mois contenant la date
`NEXT_DAY(`*`date, jour`*`)`	Expression de type date qui a pour valeur la date du prochain jour de la semaine spécifié dans le jour
`ROUND(`*`date, masque`*`)`	Arrondit la date à la précision spécifiée
`TRUNC(`*`date, masque`*`)`	Tronque la date à la précision spécifiée
`EXTRACT(YEAR | MONTH | DAY FROM` *`date`*`)`	Extrait le jour, le mois ou l'année à partir d'une date
`TO_DATE(`*`chaine, masque`*`)`	Convertit une chaîne de caractères en dates (ex : `TO_DATE('10/12/2024', 'DD/MM/YYYY')`

II.4. Autres fonctions

Fonction	Utilité
`CHR(`*`n`*`)`	Retourne le caractère dont le code (ASCII ou EBCDIC) est égal à l'expression numérique entré en paramètre
`GREATEST(`*`exp1, exp2…`*`)`	Retourne la plus grande des valeurs des expressions arguments

LEAST(*exp1, exp2…*)	Retourne la plus petite des valeurs des expressions arguments
COALESCE(*exp1, exp2…*)	Retourne la première valeur différente de NULL des expressions arguments, s'il y en a une, et la valeur NULL s'il n'y en a pas

II.5. Fonctions d'agrégat (pour des sélections uniquement, ne concernent pas les conditions)

Fonction	Utilité
AVG(*expression*)	Moyenne des valeurs d'une colonne
SUM(*expression*)	Somme des valeurs d'une colonne
MIN(*expression*)	La plus petite des valeurs d'une colonne
VARIANCE(*expression*)	La plus grande des valeurs d'une colonne
STDDEV(*expression*)	Écart-type ou déviation standard
COUNT(*)	Nombre de lignes
COUNT(*expression*)	Nombre de lignes ayant pour valeur non nulle pour une expression
COUNT(DISTINCT *expression*)	Nombre de lignes ayant des valeurs distinctes non nulles pour une expression

III. Requêtes d'initialisation de la base de données

III.1. Créer une table (tableau de valeurs)

```
CREATE TABLE [IF NOT EXISTS] table1 (
    attribut1 type1 [[CONSTRAINT nomContrainte1] contrainte1] [GENERATED AS IDENTITY],
    attribut2 type2 [[CONSTRAINT nomContrainte2] contrainte2],
    attribut3 type3 [[CONSTRAINT nomContrainte3] contrainte3],
    [CONSTRAINT nomContrainte4] PRIMARY KEY (attribut1),
    [[CONSTRAINT nomContrainte5] FOREIGN KEY (attribut3) REFERENCES table2(attribut1)]
);
```

ou

```
CREATE TABLE table1 (
    attribut1 type1 [CONSTRAINT nomContrainte1] PRIMARY KEY [GENERATED AS IDENTITY],
    attribut2 type2 [[CONSTRAINT nomContrainte2] contrainte2],
    attribut3 type3 [[CONSTRAINT nomContrainte3] REFERENCES table2(attribut1)]
);
```

`GENERATED AS IDENTITY` permet de créer automatiquement les valeurs des clés primaires si le type est un nombre entier.

III.2. Les contraintes

Contrainte	Initiales nom contrainte	Description
`NOT NULL`	*nn_nomContrainte*	Permet de rendre obligatoire l'entrée d'une valeur dans un attribut
`CHECK(`*condition*`)`	*ck_nomContrainte*	Permet de vérifier la condition avant d'insérer un enregistrement
`UNIQUE`	*uq_nomContrainte*	Permet d'éviter l'insertion de plusieurs valeurs identiques
`DEFAULT` *valeurParDefaut*	*df_nomContrainte*	Permet de remplacer une valeur non renseignée par une autre
`PRIMARY KEY (`*attribut1*`)`	*pk_nomContrainte*	Permet de créer une clé primaire. Il s'agit généralement d'un identifiant, et sa valeur est unique.
`FOREIGN KEY (`*attribut3*`) REFERENCES` *table2*`(`*attribut7*`)`	*fk_nomContrainte*	Permet de créer une clé étrangère en relation avec une autre table (déjà existante)

III.2.1. Ajouter une contrainte (exemple avec une clé primaire)

```
ALTER TABLE table1
ADD CONSTRAINT pk_cle PRIMARY KEY(attribut1);
```

III.2.2. Supprimer une contrainte (exemple avec une clé primaire)

```
ALTER TABLE table1
DROP CONSTRAINT pk_cle;
```

III.2.3. Activer/Désactiver une contrainte d'intégrité (exemple avec une clé primaire)

```
ALTER TABLE table1
ENABLE/DISABLE pk_cle;
```

Il est fortement conseillé d'ajouter une contrainte en même temps que la création d'une table. Ces commandes peuvent être utile si une contrainte a été oubliée ou n'aurait pas dû être insérée.

III.3. Renommer une table

```
RENAME ancienNom TO nouveauNom;
```

III.4. Supprimer une table

```
DROP TABLE [IF EXISTS] table1 [CASCADE CONSTRAINTS];
```

III.5. Copier une table

```
CREATE TABLE table2 AS (SELECT * FROM table1);
```

IV. Requêtes de mise à jour de la table

IV.1. Insérer un enregistrement (ligne)

```
INSERT INTO table1
VALUES (valeur1, valeur2…);
```

ou

```
INSERT INTO table1 (attribut1, attribut2…)
VALUES (valeur1, valeur2…);
```

IV.2. Mettre à jour un ou plusieurs enregistrement(s)

```
UPDATE table1
SET attribut1 = valeur1 [, attribut2 = valeur2…]
WHERE condition;
```

IV.3. Supprimer un ou plusieurs enregistrement(s)

```
DELETE FROM table1
WHERE condition;
```

IV.4. Modifier la structure de la table

IV.4.1. Ajouter une colonne

```
ALTER TABLE table1
ADD attribut type;
```

IV.4.2. Supprimer une colonne

```
ALTER TABLE table1
DROP [COLUMN] attribut;
```

IV.4.3. Modifier le type d'une colonne

```
ALTER TABLE table1
MODIFY attribut nouveauType;
```

IV.4.4. Modifier le nom d'une colonne

```
ALTER TABLE table1
RENAME ancienNom TO nouveauNom;
```

V. Requêtes de sélection de données

V.1. Sélection de tous les enregistrements de la table

```
SELECT * FROM table1;
```

V.2. Sélection des enregistrements de la table respectant la condition

```
SELECT * FROM table1
WHERE condition;
```

V.3. Sélection des enregistrements de la table dans un ordre spécifique

```
SELECT * FROM table1
[WHERE condition]
ORDER BY attribut1 [DESC], attribut2 [DESC]…;
```

V.4. Sélection des *n* premiers enregistrements de la table

```
SELECT * FROM table1
[WHERE condition]
[ORDER BY attribut1 [DESC], attribut2 [DESC]…]
FETCH FIRST n ROW[S] ONLY;
```

V.5. Sélection des attributs de la table avec des enregistrements respectant la condition

```
SELECT attribut1, attribut2 FROM table1
WHERE condition;
```

V.6. Sélection d'un attribut de la table sans répétitions d'enregistrements respectant la condition

```
SELECT DISTINCT(attribut1) FROM table1
WHERE condition;
```

ou

```
SELECT DISTINCT attribut1 FROM table1
WHERE condition;
```

V.7. Les jointures (sélection des attributs de la table avec une ou plusieurs jointure(s) à une autre table)

V.7.1. Jointure interne simple

```
SELECT table1.attribut2, table2.attribut3 FROM table1
[INNER] JOIN table2 ON table1.attribut1 = table2.attribut4
WHERE condition;
```

Autre solution (à utiliser uniquement pour deux attributs identiques de deux tables différentes) :

```
SELECT table1.attribut2, table2.attribut3 FROM table1
[INNER] JOIN table2 USING(attribut1)
WHERE condition;
```

V.7.2. Jointure externe gauche (avec des enregistrements de la table de gauche non présents dans la table de droite)

```
SELECT table1.attribut1, table2.attribut3 FROM table1
LEFT [OUTER] JOIN table2 ON table1.attribut1 = table2.attribut2
WHERE condition;
```

V.7.3. Jointure externe droite (avec des enregistrements de la table de droite non présents dans la table de gauche)

```
SELECT table1.attribut1, table2.attribut3 FROM table1
RIGHT [OUTER] JOIN table2 ON table1.attribut1 = table2.attribut2
WHERE condition;
```

V.7.4. Jointure externe entière (avec des tous les enregistrements non présents dans une autre table)

```
SELECT table1.attribut1, table2.attribut3 FROM table1
FULL [OUTER] JOIN table2 ON table1.attribut1 = table2.attribut2
WHERE condition;
```

V.7.5. Jointure naturelle (pour deux tables avec une colonne du même nom)

```
SELECT table1.attribut1, table2.attribut3 FROM table1
NATURAL JOIN table2
WHERE condition;
```

V.7.6. Produit cartésien (retourne chaque ligne d'une table avec chaque ligne d'une autre table)

```
SELECT table1.attribut1, table2.attribut3 FROM table1
CROSS JOIN table2
WHERE condition;
```

V.8. Utilisation d'une fonction d'agrégat

```
SELECT maFonction(attribut1, attribut2…) FROM table1
WHERE condition;
```

V.9. Les alias

V.9.1. Alias sur les tables (pour faciliter les jointures)

```
SELECT t1.attribut2, t2.attribut3 FROM table1 t1
JOIN table2 t2 ON table1.attribut1 = table2.attribut4
WHERE condition;
```

V.9.2. Alias sur une fonction d'agrégat (pour nommer une colonne sans nom)

```
SELECT COUNT(*) [AS] nomColonne FROM table1
WHERE condition;
```

V.10. Les groupes

Exemple : Calculer le nombre d'enregistrements de la table pour chaque enregistrement d'attribut du même nom respectant la condition 1 avant le groupement et la condition 2 après le groupement

```
SELECT attribut1 [, attribut2…], COUNT(*) AS variable FROM table1
WHERE condition1
GROUP BY attribut1 [, attribut2…]
[HAVIING condition2]
[ORDER BY attribut1 [DESC], attribut2 [DESC]…];
```

V.11. Sélection des enregistrements d'un attribut présents dans la première table ou la deuxième ou les deux (union)

```
SELECT attribut1 FROM table1
UNION
SELECT attribut1 FROM table2;
```

V.12. Sélection des enregistrements d'un attribut présents dans la première table, mais pas la deuxième (différence)

```
SELECT attribut1 FROM table1
MINUS
SELECT attribut1 FROM table2;
```

V.13. Sélection des enregistrements d'un attribut présents dans la première table et la deuxième (intersection)

```
SELECT attribut1 FROM table1
INTERSECT
SELECT attribut1 FROM table2;
```

VIII. Autres requêtes de données

VIII.1. Requête de description d'une table

```
DESC table1;
```

ou

```
DESCRIBE table1;
```

VIII.2. Les transactions

Valider une transaction : `COMMIT;`

Annuler une transaction : `ROLLBACK;`

VII. PL/SQL (sur Oracle uniquement)

1. Structure

```
DECLARE
    declarations;
BEGIN
    instructions;
EXCEPTION
    gestion_des_erreurs;
END;
```

Note : Le bloc DECLARE peut être retiré si aucune déclaration n'a été faite. Le bloc EXCEPTION permet de gérer les erreurs du programme, il n'est donc pas obligatoire. Il est possible de faire plusieurs blocs imbriqués.

2. Déclarations (dans le bloc DECLARE)

A. Variables de types simples

```
v_variable1 type1 := valeur;
v_variable2 type2;
v_variable3 type3;
```

B. Variables constantes

```
v_variable1 CONSTANT type1 := valeur;
```

C. Variables de types d'une colonne

```
v_variable1 table1.attribut1%TYPE;
```

D. Variables de types d'une ligne entière d'une table

```
v_variable1 table1%ROWTYPE;
```

E. Curseurs explicites

a. Curseur sans paramètres (*c_curseur*)

```
CURSOR c_curseur IS (SELECT * FROM table1 WHERE condition [ORDER BY attribut1 [DESC]]);
```

b. Curseur avec paramètres (*c_curseur*(*parametres*))

```
CURSOR c_curseur(v_attribut1 typeSansParentheses) IS (SELECT * FROM table1 WHERE attribut1
= v_attribut1 [ORDER BY attribut1 [DESC]]);
```

c. Curseurs avec verrouillage des données (pour une mise à jour de la table notamment)

```
CURSOR c_curseur IS (SELECT * FROM table1 WHERE condition [ORDER BY attribut1 [DESC]]) FOR
UPDATE;
```

Note : Pour désigner l'élément sélectionné par le curseur dans une condition, entrez : `CURRENT OF c_curseur`

F. Exceptions

a. Exception simple

```
NOM_EXCEPTION_PERSONNALISEE EXCEPTION;
```

b. Exception créée à l'aide d'un numéro d'exception

```
NOM_EXCEPTION_PERSONNALISEE EXCEPTION;
PRAGMA EXCEPTION_INIT(NOM_EXCEPTION_PERSONNALISEE, numero_exception);
```

Exemple :

```
VIOLATION_CONTRAINTE_CHECK EXCEPTION;
PRAGMA EXCEPTION_INIT(VIOLATION_CONTRAINTE_CHECK, -2290);
```

3. Opération d'affectation

A. Affectation simple

```
v_variable := valeur;
```

B. Affectation avec une requête de sélection (ne fonctionne que si la sélection ne renvoie qu'une seule valeur)

```
SELECT attribut1 INTO v_variable FROM table1 WHERE condition;
```

4. Conditions

Instruction	Description

IF *condition1* THEN
  instruction1;
END IF;
	Si *condition1* est vraie, alors on exécute *instruction1*

IF *condition1* THEN
  instruction1;
ELSE
  instruction2;
END IF;
	Si *condition1* est vraie, alors on exécute *instruction1*, sinon, on exécute *instruction2*

IF *condition1* THEN
  instruction1;
}
ELSIF *condition2* THEN
  instruction2;
ELSE
  instruction3;
END IF;
	Si *condition1* est vraie, alors on exécute *instruction1*, sinon, si *condition2* est vraie, on exécute *instruction2*, sinon, on exécute *instruction3*

5. Boucles

Instruction	Description

`FOR i IN 0..n LOOP`
`    instruction1;`
`END LOOP;`
	On répète *n* fois *instruction1* pour *i* allant de 0 à *n* compris (pas besoin de déclarer la variable, elle est déclarée implicitement et est détruite à la fin de la boucle)

`FOR elt IN (SELECT * FROM table1) LOOP`
`    instruction1;`
`END LOOP;`
	On parcours la requête de sélection (ou une chaine de caractère) pour *elt* prenant tous les enregistrements de la requête (pas besoin de déclarer la variable, elle est déclarée implicitement et est détruite à la fin de la boucle)

`WHILE condition LOOP`
`    instruction1;`
`END LOOP;`
	On répète jusqu'à ce que *condition* soit fausse (peut ne pas être répété)

`LOOP`
`    instruction1;`
`    EXIT WHEN condition;`
`    instruction2;`
`END LOOP;`
	On répète jusqu'à ce que *condition* soit fausse (*instruction1* est forcément répété une fois)

`EXIT;`	Permet de sortir d'une boucle sans la terminer

6. Les curseurs implicites et explicites

A. Les curseurs implicites

Curseur	Utilité
`SQL%ROWCOUNT`	Nombre de lignes traitées par la requête précédentes de mise à jour de la table
c_curseur`%NOTFOUND`	Vérifier que le curseur est vide ou la dernière ligne est déjà atteinte
c_curseur`%FOUND`	Vérifier que le curseur n'est pas vide ou la dernière ligne n'est pas encore atteinte
c_curseur`%ISOPEN`	Vérifier que le curseur est ouvert

B. Les curseurs explicites

```
OPEN c_curseur;
LOOP
```

```
    FETCH c_curseur INTO v_variable;
    EXIT WHEN c_curseur%NOTFOUND;
    instructions;
END LOOP;
CLOSE c_curseur;
```

ou

```
FOR v_variable IN c_curseur LOOP
    instructions;
END LOOP;
```

7. Les procédures

A. Créer une procédure

```
CREATE OR REPLACE PROCEDURE myprocedure [(parametreEntree1 IN type…, parametreSortie1 OUT
type…)]
IS
    [déclarations]
BEGIN
    instructions;
EXCEPTION
    gestion_des_erreurs;
END;
```

Note : Les paramètres d'entrée et de sortie ne sont pas obligatoires. Une variable en paramètre peut être à la fois une entrée et une sortie (avec IN OUT). La gestion des exceptions n'est pas obligatoire.

B. Retourner une valeur de la procédure

```
parametreSortie1 := valeur;
```

C. Faire un appel à la procédure

```
myprocedure[(variable1, variable2)];
```

ou en dehors d'un bloc :

```
EXECUTE myprocedure[(variable1, variable2)];
```

D. Supprimer une procédure

```
DROP PROCEDURE myprocedure;
```

8. Les fonctions

A. Créer une fonction

```
CREATE OR REPLACE FUNCTION myfontion [(parametre1 type…, parametre2 type…)]
RETURN type
IS
    [déclarations]
BEGIN
    instructions;
```

```
EXCEPTION
    gestion_des_erreurs;
END;
```

B. Retourner une valeur de la fonction

```
RETURN variable;
```

C. Faire un appel à la fonction

```
variable := myfonction[(variable1, variable2)];
```

ou en dehors d'un bloc :

```
SELECT myfonction[(variable1, variable2)] FROM dual;
```

D. Supprimer une fonction

```
DROP FUNCTION myfonction;
```

9. Les vues

A. Créer une vue

```
CREATE OR REPLACE VIEW myvue
AS requete;
```

Note : Il est possible de modifier une vue avec des requêtes de mises à jour de la table.

B. Faire un appel à la vue

```
SELECT * FROM myvue;
```

C. Supprimer une vue

```
DROP VIEW myvue;
```

10. Les CTE (Common Table Expression)

Les CTE sont des vues provisoires, la syntaxe est la suivante :

```
WITH
myCTE1 AS (requete1),
myCTE2 AS (requete2)…
requete;
```

11. Les exceptions (dans le bloc EXCEPTION)

A. Exception prédéfinies

Nom exception	Erreur Oracle	Numéro exception

(Violation de contrainte unique)	ORA-00001	-803
(Violation de contrainte not null)	ORA-01400	-407
NO_DATA_FOUND	ORA-01403	100
TOO_MANY_ROWS	ORA-01422	-1422
ZERO_DIVIDE	ORA-01476	-1476
(Violation de contrainte check)	ORA-02290	-2290
STORAGE_ERROR	ORA-06500	-6500
VALUE_ERROR	ORA-06502	-6502
CURSOR_ALREADY_OPEN	ORA-06511	-6511
COLLECTION_IS_NULL	ORA-06531	-6531
(Type de donnée invalide)	ORA-00902	-104 ou -199
OTHER (Autres erreurs)	ORA-42612	-84

B. Provoquer une erreur personnalisée

```
RAISE NOM_EXCEPTION_PERSONNALISEE;
```

C. Gestion des erreurs (dans le bloc EXCEPTION)

```
WHEN NOM_EXCEPTION1 THEN
    instructions1;
WHEN NOM_EXCEPTION2 THEN
    instructions2;
```

12. Les instructions de base

Instruction	Description
SET SERVEROUTPUT ON;	Activer les commandes d'affichage de messages dans la conole (à exécuter 1 fois par fichier au démarrage de SQL Developper)
DBMS_OUTPUT.PUT_LINE('*texte*');	Affiche un texte dans la console avec un retour à la ligne