



PL/SQL- BAS663 - Triggers et procédures stockées avec Oracle –

BIENVENUE !



LES OBJECTIFS

-
- Savoir utiliser le langage de développement PL/SQL
- Savoir écrire des procédures stockées, des fonctions et des triggers.

1. Installer Oracle v 19c

BIENVENUE !



LES OBJECTIFS

Être en mesure d'installer une instance Oracle.

Savoir installer le client Sql Developer.

Savoir créer un schéma de base de données

- Créer un compte utilisateur
- Donner des droits

Initiation à l'administration

- Créer de tablespaces
- gérer l'espace d'une base



PLAN

1. Installer Oracle v 19c
2. Installer Sql Developer
3. Mettre en place HR : la *base de données pour formation Oracle*
4. *Rappel sur SQL*
5. *Procédures stockées*
6. *Fonctions*
7. *Trigger*
8. *Projet*

PLAN DE COURS



OBJECTIFS PÉDAGOGIQUES

- Connaître l'utilisation de PL/SQL sous oracle

PROGRAMME

INTRODUCTION

- Qu'est-ce qu'un SGBDR ?
- Le SQL
- Le SQL adapté à Oracle

SQL PLUS : LES BASES

- Structure générale d'une requête
- Les différentes clauses
- Requête à plusieurs relations et sous-requêtes
- Opérations ensemblistes et multi-ensembles



PLAN DE COURS



DES OUTILS ORACLE DANS LA MODÉLISATION

- Les tables
- Les séquences
- Les Index
- Les vues et les vues matérialisées
- Les tables partitionnées

LE BLOC PL/SQL

- Définition d'un bloc PL/SQL
- Interagir avec la base de données
- Les curseurs
- Le traitement des erreurs
- La gestion des transactions

LE PL/SQL STOCKÉ EN BASE

- Les triggers
- Les fonctions et procédures
- Les package

Partie 1

PRÉPARER SON ENVIRONNEMENT Oracle Entreprise V 19c



LES OBJECTIFS



Être autonome sur l'installation

PLAN DE COURS



1. **Procédure d'installation d'Oracle version 19c**
 - 1.1. Télécharger et décompresser d'Oracle version 19c pour Windows
 - 1.2. Lancement de l'installation et Paramétrage de l'installation
 - 1.3. SE CONNECTER À L'INTERFACE Oracle Enterprise Manager
 - 1.4. Se connecter à la base via **sqlplus** en mode **Admin**
 - 1.4.1. Retrouver son **SID**
2. **INSTALLER le CLIENT Sql Developer**
 - 2.1. Procédure : télécharger, décompresser puis installer
3. Connecter Sql Developer à une base Oracle
4. Diagnostiquer des Pb de connexion à la base oracle
 - 4.1. Vérifier l'état du listener
 - 4.2. Explorer les fichiers tnsnames.ora et listener.ora
5. **Mettre en place HR : la *base de données pour formation Oracle***
 - 5.1. Vérifier que le schéma optionnel HR -**base de données de formation Oracle**- est installé
 - 5.1.1. via l'interface : -SQL Developer
 - 5.1.2. via l'interface : -SQL Plus
 - 5.2. Activer la connexion HR
 - 5.3. Créer une nouvelle connexion pour le compte HR
 - 5.4. Tester l'accès aux tables

Procédure d'installation d'Oracle version 19c




Ask "What cloud developer tools do you have?"

Database / Technical Details /
Oracle Database 19c Download for Microsoft Windows x64 (64-bit)

Oracle Database 19c (19.3)

Oracle Database 19c (19.3) for Microsoft

Download

 WINDOWS.X64_193000_db_home.zip


Directions
Installation guides and general Oracle Database 19c documentation are [here](#).

×

You must accept the [Oracle License Agreement](#) to download this software.

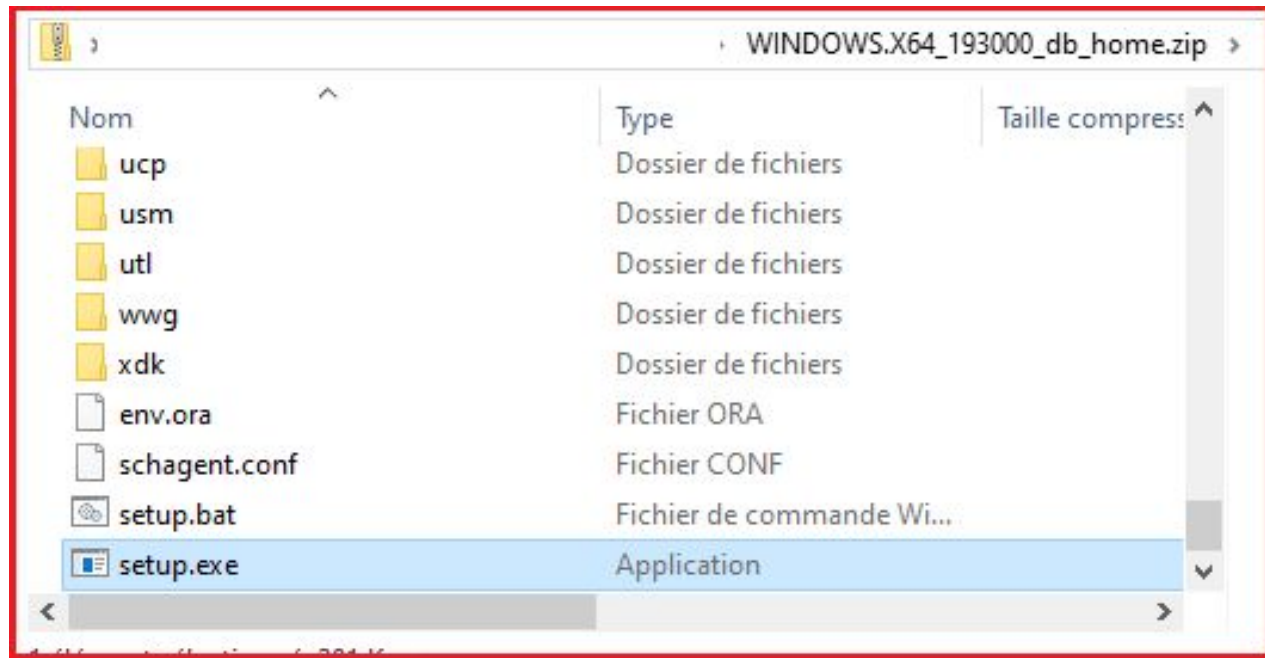
☒ I reviewed and accept the Oracle License Agreement

Download WINDOWS.X64_193000_db_home.zip



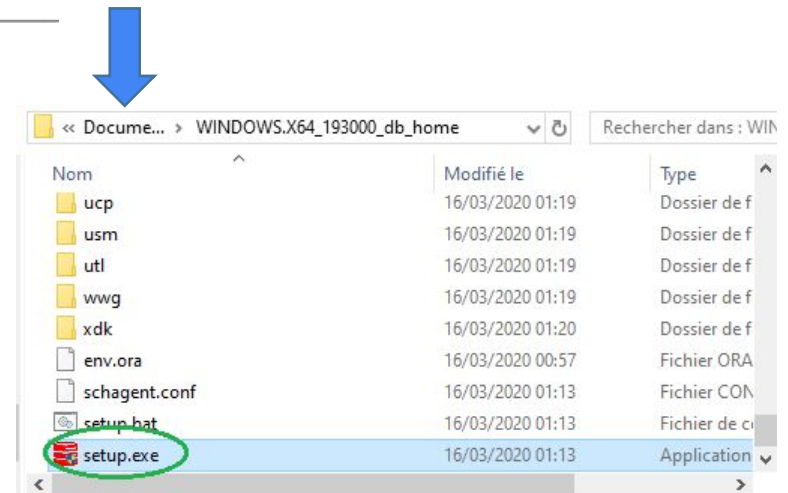
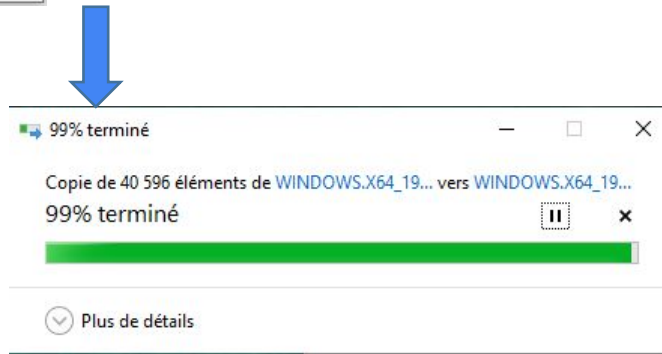
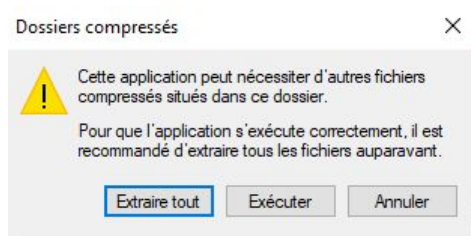
Télécharger d'Oracle version 19c pour Windows 64.

Pour télécharger le SGBD Oracle, vous devez avoir un compte chez Oracle, ce compte est gratuit



Télécharger d'Oracle version 12c pour Windows 64.

Extraire dans un dossier de votre choix :



Lancement de l'installation



Programme d'installation d'Oracle Database 19c - Etape 1 sur 16

Sélectionner une option de configuration

19c ORACLE Database

Sélectionnez l'une des options d'installation suivantes.

☒ Créer et configurer une base de données mono-instance
Cette option crée une base de données de départ.

☐ Configurer le logiciel uniquement

Remarque 1 : pour l'installation RAC, exécutez 'Configurer le logiciel uniquement', puis exécutez DBCA (Assistant Configuration de base de données) à partir du répertoire de base Oracle.
Remarque 2 : pour la mise à niveau d'une instance Oracle Database, exécutez 'Configurer le logiciel uniquement', puis exécutez DBUA (Assistant Mise à niveau de base de données) à partir du répertoire de base Oracle.

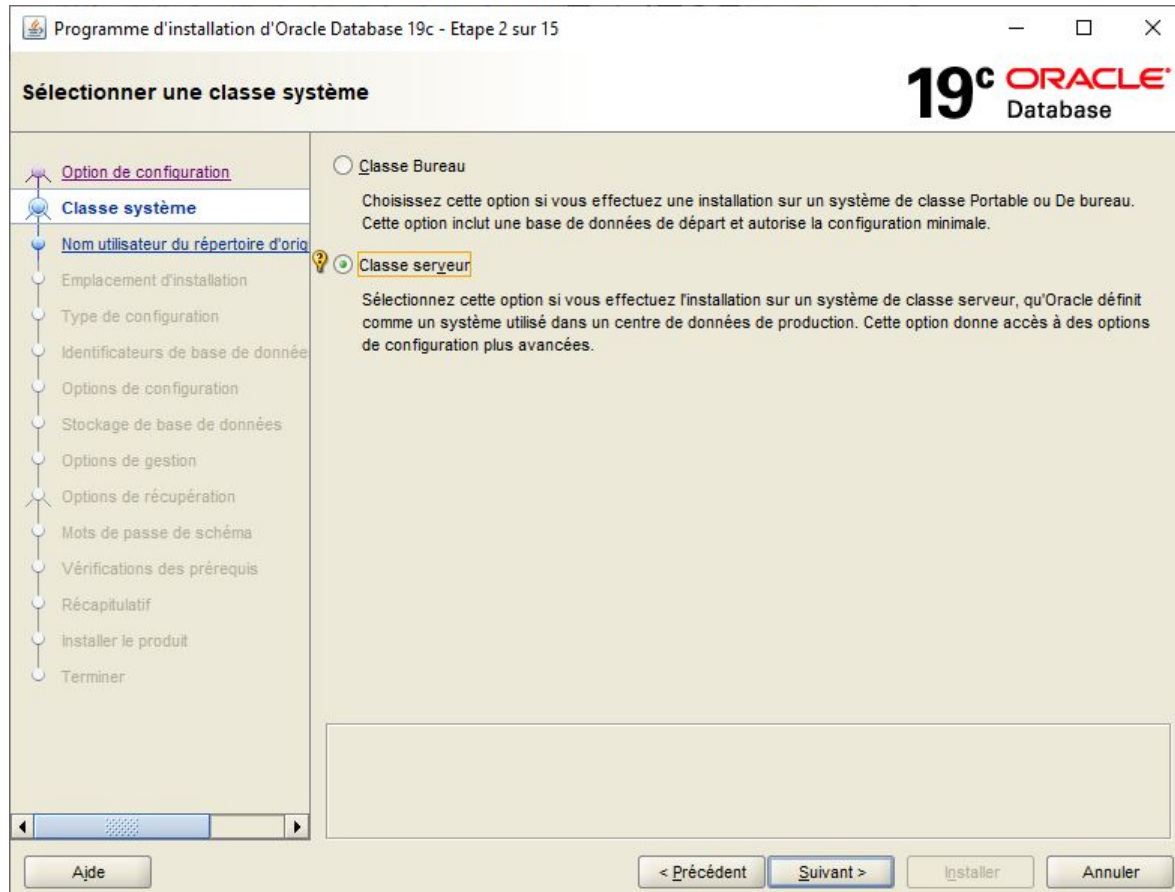
Option de configuration

- Options d'installation de base de d
- Type d'installation
- Nom utilisateur du répertoire d'orig
- Emplacement d'installation
- Type de configuration
- Identificateurs de base de donnée
- Options de configuration
- Stockage de base de données
- Options de gestion
- Options de récupération
- Mots de passe de schéma
- Vérifications des prérequis
- Récapitulatif
- Installer le produit
- Terminer

Aide < Précédent Suivant > Installer Annuler

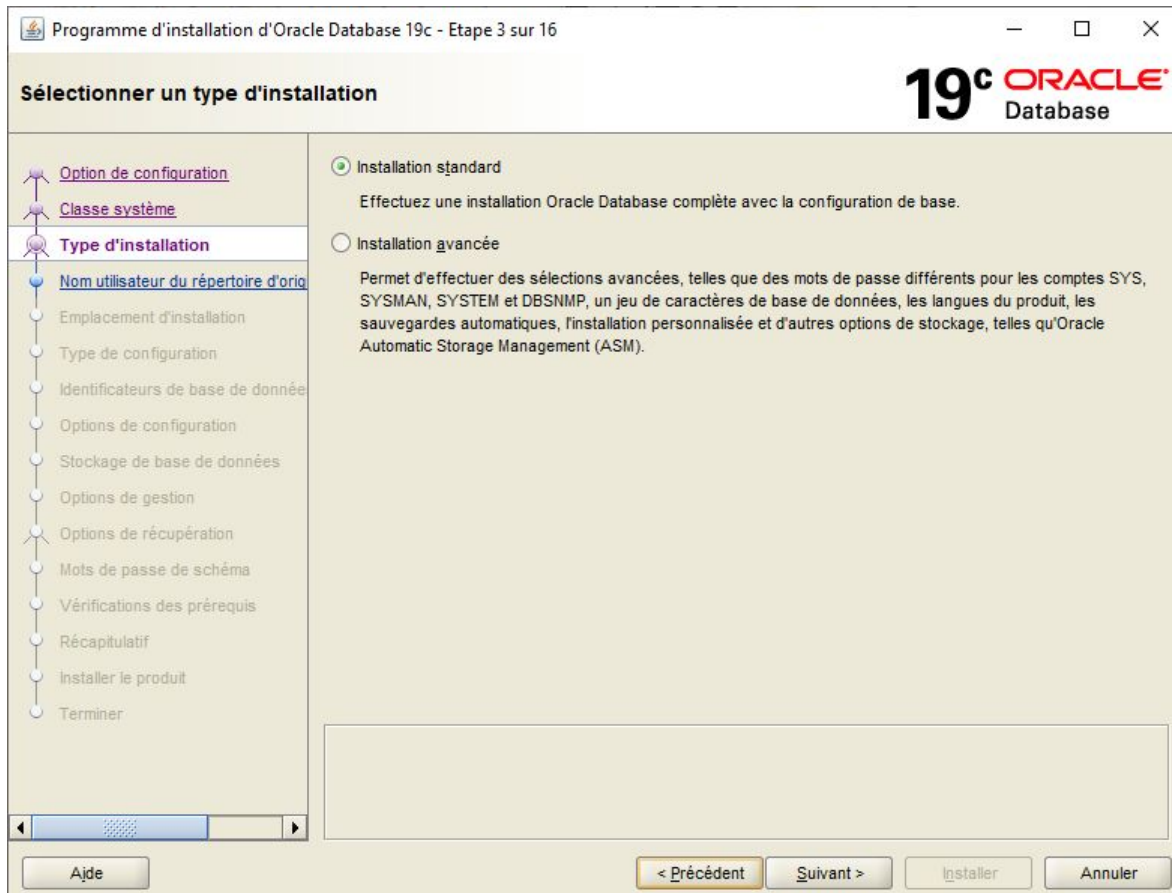
Lancement de l'installation

Sélectionner la classe Server



Lancement de l'installation

Choisir une installation standard



paramétrage de l'installation

Le mot de passe -choisir : oracle. C'est le mot de passe l'utilisateur **sys**, administrateur du SGBD

Programme d'installation d'Oracle Database 19c - Etape 5 sur 9

Configuration d'installation standard **19c ORACLE Database**

Effectuez l'installation complète de la base de données avec la configuration de base.

Répertoire de base Oracle Base : Parcourir...

Software location: C:\WINDOWS.X64_193000_db_home

Type de stockage :

Emplacement des fichiers de base de données : Parcourir...

Edition de base de données :

Nom global de base de données :

Mot de passe : Confirmer le mot de passe :

Nom du service :

☐ Créer en tant que base de données de conteneur

Nom de la base de données pluggable :

Messages :

⚠ Mot de passe : [INS-30011] Le mot de passe ADMIN entré n'est pas conforme aux normes recommandées par Oracle.

Partie 2

- Rappel sur SQL
- PL/SQL



LES OBJECTIFS

- Rappel sur SQL, les formes normales
- Connaître l'utilisation de PL/SQL sous oracle

PLAN DE COURS



OBJECTIFS PEDAGOGIQUES

- Connaître l'utilisation de PL/SQL sous oracle

PROGRAMME

INTRODUCTION

- Qu'est-ce qu'un SGBDR ?
- Le SQL
- Le SQL adapté à Oracle

SQL PLUS : LES BASES

- Structure générale d'une requête
- Les différentes clauses
- Requête à plusieurs relations et sous-requêtes
- Opérations ensemblistes et multi-ensembles

SQL Live

- Sql Live



LES OBJECTIFS

- savoir travailler sans une installation

Première partie
LES SYSTÈMES DE GESTION DE BASE DE DONNÉES

LES SYSTÈMES DE GESTION DE BASE DE DONNÉES

PROGRAMME

INTRODUCTION

- **Qu'est-ce qu'un SGBDR ?**
- Le SQL
- Le SQL adapté à Oracle

SQL PLUS : LES BASES

- Structure générale d'une requête
- Les différentes clauses
- Requête à plusieurs relations et sous-requêtes
- Opérations ensemblistes et multi-ensembles

- Définition d'un SGBD
- Propriétés essentielles d'un SGBDR
- Le modèle relationnel
- Le schéma de base de données
- Base de données normalisée
- SGBD ACID
- Les formes normales

Une **base de données** est un ensemble de **données**

- structurées
- et stockées sur un support permanent.

Un **système de gestion de bases de données (SGBD)** est :

- un **logiciel de haut niveau d'abstraction**
- qui permet de **stocker** et **manipuler ces données**
- le stockage se fait soit en suivant :
 - un schéma **relationnel (SGBDR**-support les opérations ensemblistes de l'algèbre relationnelle-) **ou**
 - un système de key-value (NoSQL) pour les SGBD non relationnel.

Propriétés essentielles d'un SGBDR

Un Système de gestion de base de données relationnel, (SGBDR comme oracle) peut être:

- **Transactionnel** : garantie l'intégrité des données.

- Les insertions, modification sont fréquents.
- Les accès sont concurrents.

ET

- **Analytic** OLAP (Data Warehouse : Recherche - projection - reporting).

- Pas de mise à jour fréquent de la base.
- Recherche très fréquent

LE MODÈLE RELATIONNEL

Définition : un schéma de relations : l'ensemble des relations (**tables**) de la base de données.

Exemple pour la base **Mini-compta**, le schéma de relation est composé des relations (tables) suivantes :

- ❑ **factures** (numéro, ref_client, date)
- ❑ **clients**(numéro, raison_sociale, adresse),
- ❑ **fournisseur** (numero, adresse),
- ❑ **paiement**(numero, date, montant, ref_fact)

LE MODÈLE RELATIONNEL

Définitions (Vocabulaire et éléments de langage).

Relation : Selon le modèle relationnel, une relation est une **table** SQL. **Relation = Table** .

Une relation est donc identifiée par un **nom** et comporte une ou plusieurs **attributs** (colonnes ou champs).

Un nom est associé à chaque colonne.

Exemple la **relation** ou **table client** (**nom**, **prenom**, **adresse**). Donc avec les **colonnes** ou **champs** *nom*, *prenom* et *adresse*.

Attribut : L'attribut est un champs dans la relation, il est caractérisé par un nom.

Tuple ou n-uplet : Un tuple est une **enregistrement**. C'est une **ligne** de relation, elle contient des valeurs d'attributs. Une **tuple** est aussi appelé **row**.

Domaine : ensemble des valeurs que peut prendre un attribut. Par exemple un attribut peut être numérique, une chaîne de caractères...

Autrement dit c'est le type du champs de la table (entier, date, string ...)

Les types peuvent être int, char.

Comment élaborer un schéma de bd : les méthodologies de conception.

Les méthodologies de conception permettent d'élaborer un **schéma de base de données (ou schéma de relations = Model)**.

Dans tous les cas une conception doit aboutir à un schéma de relations doit être **normalisé (et donc qui respecte au moins les 3 premières formes normales)** .

base de données normalisée

En effet, la **normalisation** est utile :

- pour **limiter les redondances** de données (doublon),
- pour limiter les pertes de données (**cif : contraintes d'intégrités fonctionnelles** (éviter des orphelins : articles sans catégorie),
- pour limiter les **incohérences** au sein des données (commandes sans clients)
- pour améliorer les performances des traitements.

schéma relationnel et contraintes d'intégrités fonctionnelles : CIF

- Les SGBD-Relationnel garantissent l'implémentation des schémas de bases de données et veillent au respect des CIF- et ce parce qu'ils sont ACID.

Un SGBD est **ACID** :

Atomic : **indivisible** : l'ensemble des modifications sur une transaction est **unitaire** : c'est tout ou rien. Pour une succession d'instructions de **DELETE**, **UPDATE** et **d'INSERT** exécutées dans le contexte d'une transaction.

Consistant : **cohérente**. La transaction s'assure de garder la base cohérente (cif).

Isolated : verrouillé. Les modifications d'une donnée dans le contexte d'une transaction sont isolées par rapport à d'autres transactions sur cette même donnée.

Durable : **persistante**. Son état est conservé même en cas de crash système. Elle peut ainsi être restaurée -Existence de solutions HA(Réplication, LogShipping, Mirroring, Mise en cluster (fail-over : basculement automatique), ...).

Conception de la base de données : la normalisation

- La **conception de la base de données** doit aboutir à schéma d'une **base de données normalisée**
- Le but essentiel de la normalisation est de **corriger** les anomalies transactionnelles qui peuvent découler d'une mauvaise modélisation des données.
- La normalisation des modèles de données permet de vérifier la robustesse de leur conception pour améliorer la modélisation (approche récursive).

Définitions :

- Une clé primaire définie de manière unique un enregistrement d'une table

- Une clé primaire peut être composite (constituée de plusieurs attributs) ou simple

- Une **clé étrangère** est une **clé primaire** d'une autre table

Les formes normales

- Elles s'emboîtent les unes dans les autres :
- Dans le modèle relationnel, les trois premières sont plus utilisées

Première forme normale

Définition :

Une relation est en première forme normale **si tous ses attributs sont atomiques**. (indivisible)

Première forme normale

- Un **attribut atomique** n'est pas :
 - **multivalué** (liste de valeurs d'autres attributs) **OU**
 - **composé** (structuré en sous-attributs)

Relation	
ID Livre	Livres
1	« Les raisins de la colères», « Pour qui sonne le glas »



Livre	
ID	Titre
1	« Les raisins de la colères»
2	« Pour qui sonne le glas »
...	

Dans le premier exemple, l'ID 1 désigne une liste de 2 livres. Ce qui n'est pas correcte. Chaque livre doit avoir son ID

Relation : client

Adresse

12 rue Planchat

12 bis rue Planchat

...

L'adresse étant ici un champs libre, il sera difficile de le contrôler. S'il y a **besoin** de vérifier l'adresse mieux vaut le décomposer en numéro, rue ...

Deuxième forme normale

Définition

Une relation est en **deuxième forme normale** ssi :

- - elle est en première forme normale
- **tout attribut non clé dépend de la totalité des clés**

Exemple

R(cle1, cle2, cle3, attrib1, attrib2, attrib3)

Si attrib3 ne dépend que de cle1 alors la relation n'est pas en 2FN

Exemple

R1 (cle1, attrib3)

R2 (cle1, cle2, cle3, attrib1, attrib2)

Emprunt			
<u>IdAbonne</u>	<u>Id_Livre</u>	<u>Nom Abonne</u>	date emprunt
1	1	2	aujourd'hui
...			

Abonne			
<u>IdAbonne</u>	Nom Abonne		
1	2		

Troisième forme normale

Objectif : élimination des redondances dues aux dépendances fonctionnelles déduites par transitivité.

Définition

Une relation est en troisième forme normale ssi :

- elle est en deuxième forme normale
- **il n'existe aucune DF entre attributs non clé**

Remarque : le concepteur peut introduire des cas de dénormalisation pour raison de perf

Commande				
<i>Idcommande</i>	<i>Id_client</i>	Prix	Totallc	Qté
1	1	2	20	10
...				

Les objets de la base de données

Tables

Les tables sont les objets qui contiennent effectivement les données dans la base. Elles peuvent être de ***deux types*** :

Les tables systèmes

Contiennent les informations permettant le bon fonctionnement de la base de données

Les tables utilisateurs

Contiennent les données des utilisateurs

Vues

Une vue est « une table virtuelle » dont les données ne sont pas stockées dans une table de la base de données, et dans laquelle il est possible de rassembler des informations provenant de plusieurs tables.

Une vue est le résultat d'une requête.

Procédures stockées

Une procédure stockée est un ensemble d'instructions précompilées et exécutées sur demande ou bien encore de façons automatique (déclenché par un trigger).

Trigger

Lorsqu'une action a été exécutée sur une table ou sur le serveur.
Cela déclenche un événement.
Vous pouvez utiliser cet événement pour prendre certaines mesures.

Index

L'index permettra de faire des recherches par dichotomie et d'accélérer les tris sur le champ concerné.

Un index est une table de correspondance où les données seront classées .

Deuxième partie

Le langage SQL

Le langage SQL

Structured Query Language (SQL)

- Langage de requête structurée.

Comme tout langage qui sera traduit pour être exécuté par la machine, il a un **LEXIQUE**, une **SYNTAXE** et une **SÉMANTIQUE** (et possède un métalangage qu'on appelle aussi grammaire du langage).

- **Requête** : instruction demandant une action sur la base de données.
- SQL: Langage **standardisé** pour **interroger**, **manipuler** et **définir** des données, et pour fournir un contrôle des accès dans les BDDR
- Développé chez IBM fin des années 79 avec la forte participation d'Oracle standard SQL ANSI.

Les ensembles du langage SQL

Il y a 16 commandes SQL classées en 4 sous-ensembles d'utilisation :

- Le Langage de Définition de Données (**DDL**) pour créer et supprimer des objets dans la base (schéma de la BDD) :
■ CREATE ■ ALTER ■ DROP ■ RENAME ■ TRUNCATE ■ COMMENT
- Le Langage de Contrôle de Données (**DCL**) pour gérer les droits sur les objets de la base:
■ GRANT ■ REVOKE
- Le Langage de Manipulation de Données (**DML**) pour la recherche, l'insertion, la mise à jour et la suppression de données
■ SELECT ■ INSERT ■ UPDATE ■ DELETE ■ MERGE
- Le Langage de Contrôle de Transaction (**TCL**) pour la gestion des transactions de la base
■ COMMIT ■ ROLLBACK ■ SAVEPOINT

Figure 1-2 Block Structure

```
SET SERVEROUTPUT ON;  -- activer l'affichage

/* Structure d'un block PL/SQL
un bloc PL/SQL peut être constitué de 3 sections : */
```

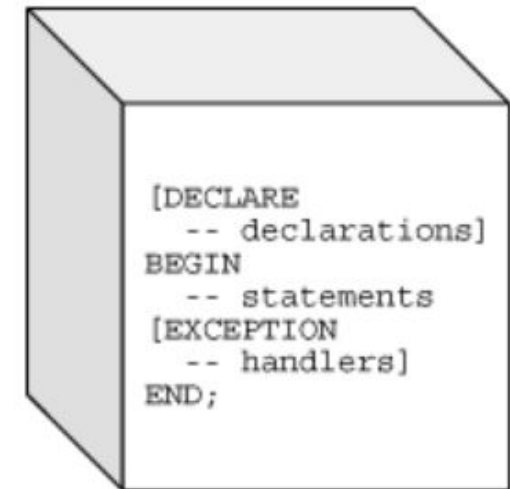
1- Section déclare (en commentaire car non obligatoire)

```
-- DECLARE
-- déclarations et initialisation -- facultative
```

2- Section BEGIN qui contient la section EXCEPTION

```
BEGIN
-- instructions exécutables -- obligatoire
DBMS_OUTPUT.PUT_LINE('hello');
-- EXCEPTION
-- interception des erreurs -- facultative
END;
```

ce block est appelé block anonyme car non stocké dans la base sous forme de procédure ou de fonction.



DBMS_OUTPUT.PUT_LINE(argument);
Pour afficher un résultat

:= pour valoriser une variable

Block imbriqué sous PL/SQL

```
DECLARE
    val1 int ; val2 int ;
BEGIN
    val1 := 15; val2 := 10;
```

```
declare
    resultat int :=val1+val2;
begin
    DBMS_OUTPUT.put_line ( 'la somme de '||val1 || ' et '||val2 ||' est : ' ||resultat);
end;
```

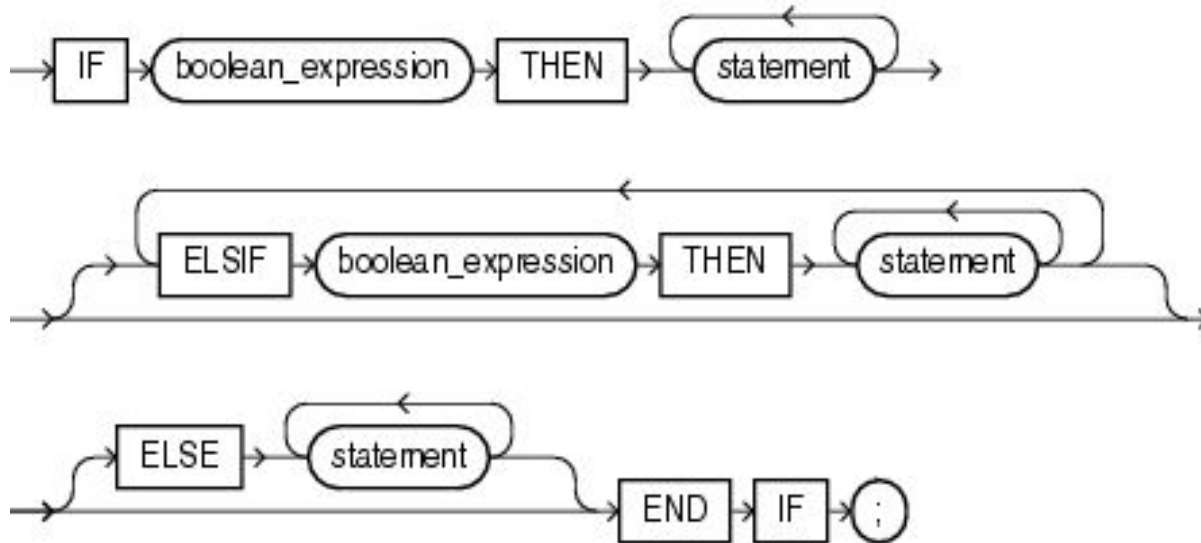
```
EXCEPTION
    WHEN OTHERS
    THEN
        DBMS_OUTPUT.put_line (SQLERRM);
END;
```

Section exception du bloc PL/SQL

```
set serveroutput on;  
DECLARE  
    l_message    varchar2(2) ;  
BEGIN  
    l_message := 'hlllo';  
    DBMS_OUTPUT.put_line (l_message);  
  
EXCEPTION  
    WHEN OTHERS  
    THEN  
        DBMS_OUTPUT.put_line ('erreur '||DBMS_UTILITY.format_error_stack||' --- '||sqlerrm);  
END;
```

Référence : DOCUMENTATION OFFICIELLE ORACLE

https://docs.oracle.com/cd/B19306_01/appdev.102/b14261/if_statement.htm



Déclaration de variable sous PL/SQL

Déclaration de variable sous PL/SQL

Déclare pour déclarer des variables;

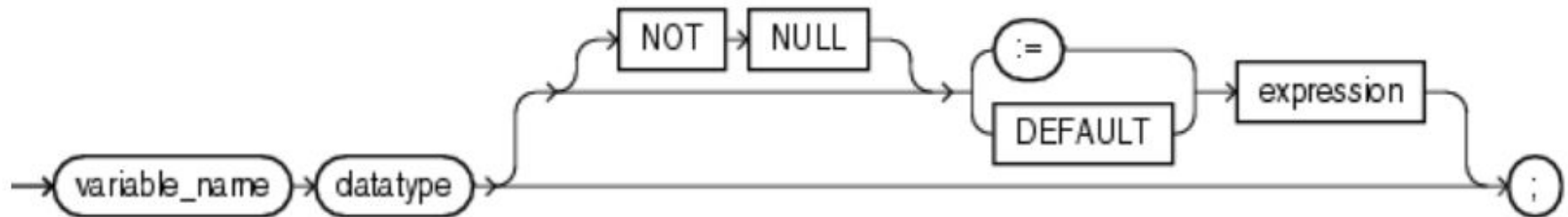
:= pour valoriser une variable

Begin

-- liste des instructions

End;

variable declaration ::=



Utilisation de variables sous PL/SQL

```
DECLARE
    val1 int ; val2 int ;
BEGIN
    val1 := 15; val2 := 10;
    DBMS_OUTPUT.put_line ( 'la somme de ' || val1 || ' et ' || val2 || ' est : ' || (val1+val2));
EXCEPTION
    WHEN OTHERS
    THEN
        DBMS_OUTPUT.put_line (SQLERRM);
END;
```

Utilisation de variables sous PL/SQL

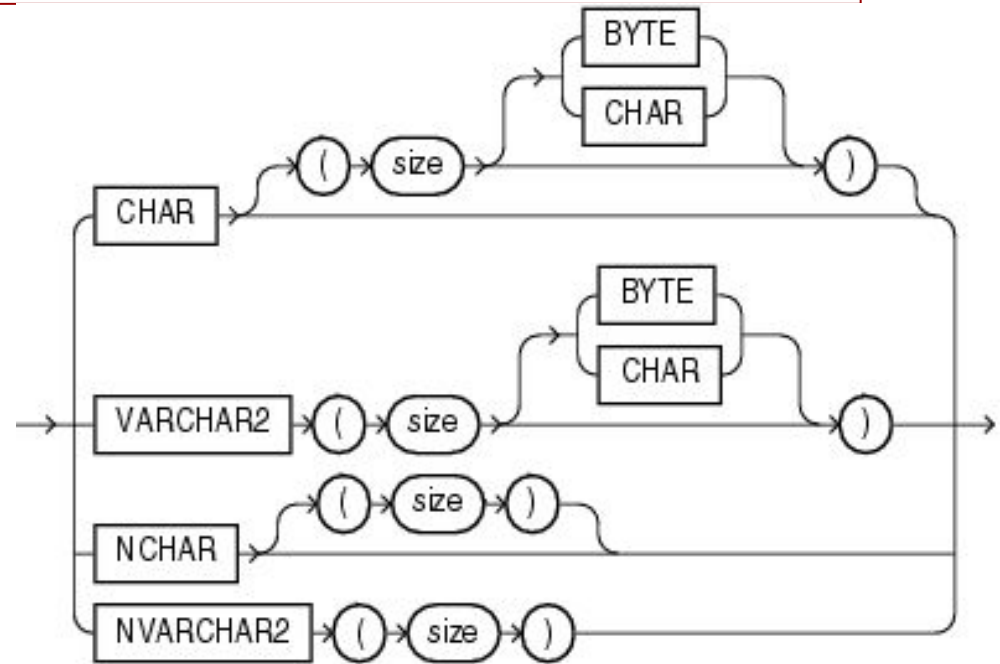
```
DECLARE
    wages          NUMBER;
    hours_worked   NUMBER := 40;
    hourly_salary  NUMBER := 22.50;
    bonus          NUMBER := 150;
    country        VARCHAR2(128);
    counter        NUMBER := 0;
    done           BOOLEAN;
    valid_id       BOOLEAN;
    emp_rec1       employees%ROWTYPE;
    emp_rec2       employees%ROWTYPE;
    TYPE commissions IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
    comm_tab       commissions;
BEGIN
    wages := (hours_worked * hourly_salary) + bonus;
    country := 'France';
    country := UPPER('Canada');
    done := (counter > 100);
    valid_id := TRUE;
    emp_rec1.first_name := 'Antonio';
    emp_rec1.last_name := 'Ortiz';
    emp_rec1 := emp_rec2;
    comm_tab(5) := 20000 * 0.15;
END;
```

Les types de données sous Oracle

- les types de données sous oracle
- **char** vs **varchar2**
- les nombres sous oracle
- a la découverte du type **number(p,s)**
- constantes
- type de données **booléen**
- les types **datetime**
- quelques fonctions de conversions : la fonction **to_char()** - la fonction **to_date()** la fonction **extract()** ...

Les types de données sous Oracle CHAR, VARCHAR2

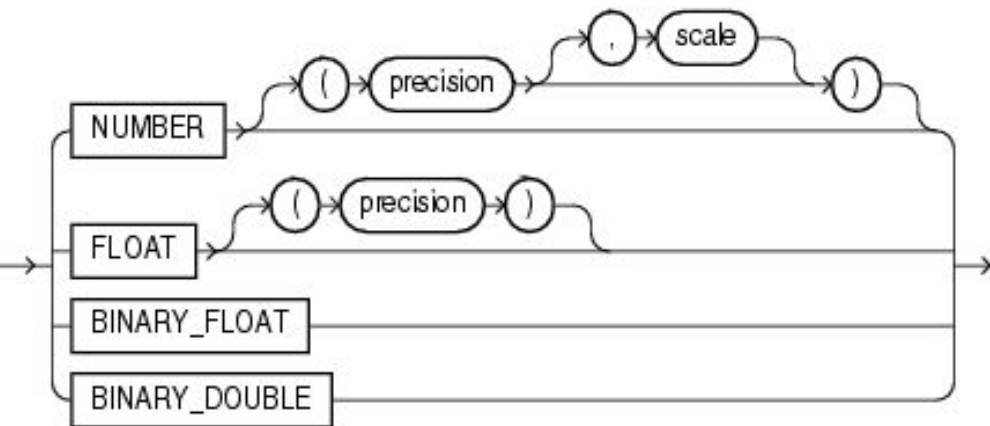
Types	Description	Size
VARCHAR2(n)	Variable-length character string.	From 1 byte to 4KB.
NVARCHAR2(size)	Variable-length Unicode character string having maximum length size characters.	Maximum size is determined by the national character set definition, with an upper limit of 4000 bytes. You must specify size for NVARCHAR2.
Char	Fixed length	



```

{ CHAR [ (size [ BYTE | CHAR ]) ]
| VARCHAR2 (size [ BYTE | CHAR ])
| NCHAR [ (size) ]
| NVARCHAR2 (size)
}
  
```

LES NOMBRES SOUS ORACLE



$x \text{ number } (6,2) \longleftrightarrow \overbrace{1234.56}^{p = 6}$
 $\underbrace{\hspace{1.5cm}}_{s = 2}$

Un nombre de longueur **p** dont **s** décimaux (au **s**-ième près).

```

{ NUMBER [ (precision [, scale ]) ]
| FLOAT [ (precision) ]
| BINARY_FLOAT
| BINARY_DOUBLE
}
    
```

$x = 1234.56 \rightarrow \text{OK}$

$x = 1234.567 \rightarrow \text{OK sera arrondi à } 1234.57$

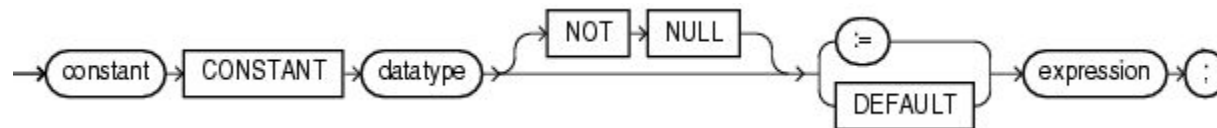
Exercices 1

```
CREATE TABLE Ora_Type_demo ( number_value NUMERIC(13, 2));  
--  
insert into ora_type_demo (number_value) values (12345678901.23); -- s < p  
select * from ora_type_demo;  
-----  
insert into ora_type_demo (number_value) values (12345678901.23); -- s < p  
select * from ora_type_demo;  
-----  
alter table ora_type_demo add number_ps_neg number(5,-2);  
insert into ora type demo (number ps neg) values (12345);
```

Exercice 2 dirigé : A la découverte du type Number(p,s)

```
DECLARE
x NUMBER(4,2);
BEGIN
  -- x := 23.45;
  -- x := 23.456;
  x := 123.456;
  DBMS_OUTPUT.put_line(' x = '||x);
END;
```

CONSTANTES



Exemples :

```
credit_limit      CONSTANT REAL      := 5000.00;
max_days_in_year  CONSTANT INTEGER   := 366;
urban_legend      CONSTANT BOOLEAN   := FALSE;
hours_worked      INTEGER := 40;
employee_count    INTEGER := 0;
pi                CONSTANT REAL := 3.14159;
```

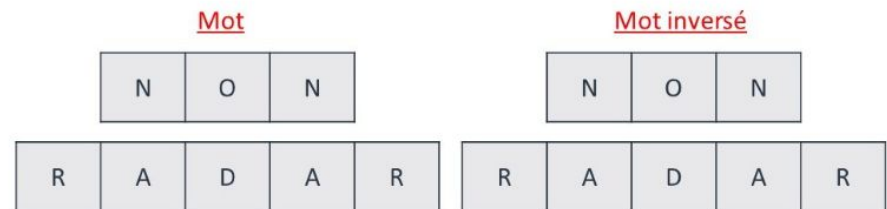

TYPE DE DONNÉES BOOLÉEN

Le type de données PL/SQL BOOLEAN stocke des valeurs logiques, qui sont **TRUE** , **FALSE** ou **NULL**.

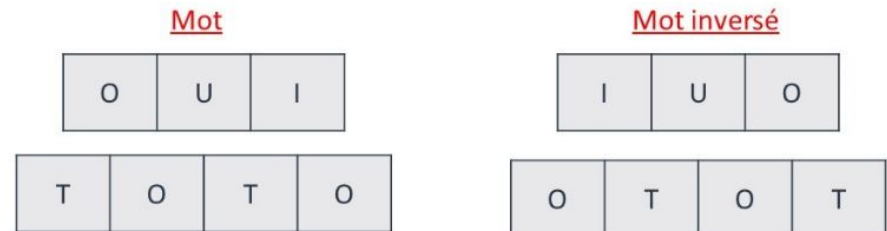
La syntaxe pour déclarer une variable BOOLEAN est :

nom_variable BOOLEAN

Exercice EstPalindrome



Exemple de mots qui ne sont pas des palindromes :



Exercice EstPalindrome, PSEUDO-CODE

```
FONCTION estPalindrome (mot)
VARIABLES
    motInverse : CHAINE DE CARACTERES
DEBUT
    motInverse ← « »
    POUR i ALLANT DE taille(mot) - 1 A 0 PAR PAS DE -1 FAIRE
        motInverse ← motInverse + mot[i]
    FINPOUR

    POUR i ALLANT DE 0 A taille(mot) - 1 PAR PAS DE 1 FAIRE
        //vérification lettre à lettre
    FINPOUR

FIN
```

TYPE DE DONNÉES BOOLÉEN

```
declare
c1 varchar2(250);
c1_inverse varchar2(250) := '';
reponse boolean := true;
compteur integer ;
reponse_afficher varchar2(50) := ' est un palindrome ';
begin
    dbms_output.put_line('saisir un mot :');
    c1 := '&c1';
    dbms_output.put_line(length(c1));
    for compteur IN REVERSE 1 .. length(c1)
    loop
        c1_inverse := c1_inverse||substr(c1,compteur,1);
        dbms_output.put_line(substr(c1,compteur,1));

    end loop;
    for compteur IN 1 .. length(c1)
    loop
        reponse := (substr(c1,compteur,1) = substr(c1_inverse,compteur,1));
        if (reponse = false) then
            reponse_afficher := ' n'est pas un palindrome';
        end if;
    end loop;

    dbms_output.put_line(c1_inverse || ' --- ' || c1 || reponse_afficher);

end;
```

LES TYPES DATETIME

Les type datetimes sont :

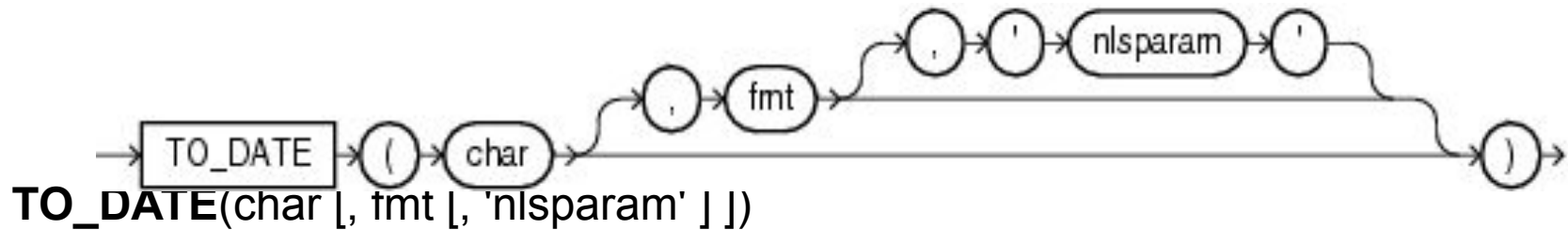
- ❑ DATE,
- ❑ TIMESTAMP,
- ❑ TIMESTAMP WITH TIME ZONE,
- ❑ et TIMESTAMP WITH LOCAL TIME ZONE.

Exemple :

--https://docs.oracle.com/cd/B19306_01/server.102/b14200/sql_elements004.htm#i34924

LA FONCTION TO_DATE()

Elle convertie en date, un string fourni en entrée suivant un format spécifique.



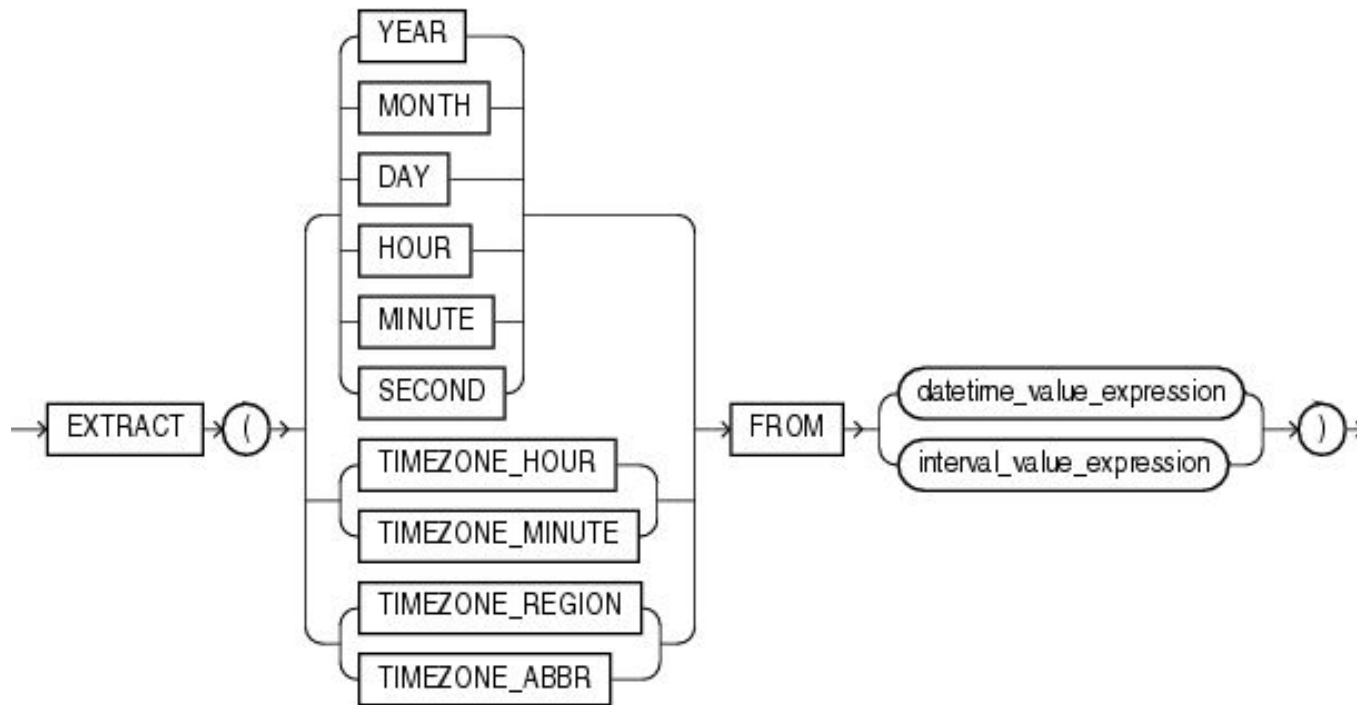
fmt : DD - DDTH - DDSP - DDSPTH - DAY (jours)

MM - MON - MONTH (mois)

YY - YYYY - RR - RRRR - YEAR - (année)

nlsparam : français,

LA FONCTION EXTRACT()



```

EXTRACT( { { YEAR      | MONTH      | DAY      | HOUR      | MINUTE      | SECOND      }
          | { TIMEZONE_HOUR | TIMEZONE_MINUTE }
          | { TIMEZONE_REGION | TIMEZONE_ABBR } }
FROM { datetime_value_expression | interval_value_expression } )
  
```

LES TYPES DATETIME

```
DECLARE
d1 DATE := sysdate;
d2 TIMESTAMP with time zone := sysdate;
BEGIN
--dbms_output.put_line (d1);
dbms_output.put_line (to_char(d1, 'DAY, DD MM (MONTH) YYYY HH:Mi:SS "de lan" YEAR '));
dbms_output.put_line (d2);
dbms_output.put_line (to_char(d2, 'DAY, DD MM (MONTH) YYYY HH:Mi:SS:ff YEAR TZR TZD TZh TZM '));
END.
--ALTER SESSION SET TIME_ZONE = '-07:00';
alter session set time_zone='00:00';
```

LA FONCTION EXTRACT()

```
declare
d1 date ;
begin

-- PROMPT  "saisir la date de rendez-vous"

d1 := to_date('&d1', 'DD/MM/YYYY');
  dbms_output.put_line('nous sommes :'||to_char(d1,'DAY'));
  dbms_output.put_line('dans 3jours, nous seront :'||to_char((d1+3),'DAY'));
dbms_output.put_line('l''année :'|| extract(YEAR from d1));

end;
```


EXERCICE

/*

Afficher **SECOND** et MILLIÈME de seconde à partir de CURRENT_TIMESTAMP

*/

LES BLOCS PL SQL : LES VARIABLES et TYPE ATTRIBUTS

```
SET SERVEROUTPUT ON
DECLARE
v_name EMPLOYEES.first_name%TYPE;
v_depno departments.department_name%TYPE;
BEGIN
DBMS_OUTPUT.PUT_LINE(NVL(v_name, 'Sans nom ') ||
' Appartient au departement ' || NVL(v_depno, ': Aucun' ));
END;
```

Exercices

- Ecrire un bloc anonyme pour afficher la somme, le produit de 2 nombres
- Ecrire un bloc anonyme pour calculer le temps qu'il faut à votre pc pour réaliser une addition de 2 nombres
-

Exercices

```
----- LES VARIABLES et initialisation -----  
  
SET SERVEROUTPUT ON  
DECLARE  
v_valeur1 int := 10;  
v_valeur2 int := 20;  
  
BEGIN  
DBMS_OUTPUT.PUT_LINE ('La somme est :'|| (v_valeur1 + v_valeur2) );  
END;
```

Exercices : puissance

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
v_valeur1 int := 10;
```

```
v_valeur2 int := 20;
```

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE ('Le produit est : ' || power(v_valeur1, v_valeur2) );
```

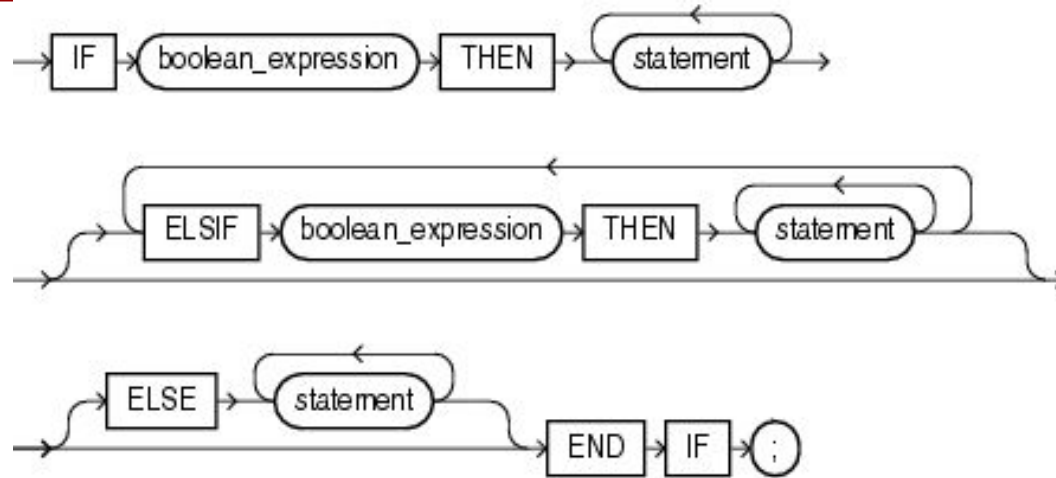
```
END;
```

IF THEN : forme simple

```
IF CONDITION THEN  
    STATEMENT 1;  
ELSE  
    STATEMENT 2;  
END IF;
```

```
IF CONDITION THEN  
    BEGIN  
        STATEMENT 1;  
        STATEMENT 2;  
    END;  
ELSE  
    STATEMENT 3;  
END IF;
```

STRUCTURE DE CONTRÔLE : IF - THEN - ELSE



```
IF boolean_expression THEN
    statement [statement]...
    [ELSIF boolean_expression THEN
        statement [statement]...]
    [ELSIF boolean_expression THEN
        statement [statement]...]...
    [ELSE statement [statement]...]
END IF;
```

EXERCICE

/*

EXERCICE 1: Demander à l'utilisateur de saisir un nombre et lui dire si ce nombre est SUP ou INF à 100

EXERCICE 2:

Demandez à l'utilisateur de saisir l'id d'un employé

si :

- le job_id = **IT_PROG** alors **afficher** son salaire + une augmentation de 0.8%
- le job_id = **FI_ACCOUNT** alors **afficher** son salaire + une augmentation de 0.5%
- le job_id = **ST_CLERK** alors **afficher** son salaire + une augmentation de 0.3%
- si non 0.2%

*/

EXERCICE preparation

```
select * from employees;
desc employees;
select job_id from employees;
select distinct job_id from employees
where job_id in ('AC_ACCOUNT', 'FI_ACCOUNT', 'FI_MGR');
---
select EMPLOYEE_ID from employees
where job_id in ('AC_ACCOUNT', 'FI_ACCOUNT', 'FI_MGR');
-- Augmentation de salaire 0.9 pour AC_ACCOUNT
-- 0.08 pour FI_ACCOUNT
-- 0.07 pour FI_MGR
```

EXERCICE preparation

```
DECLARE
    EMPID      EMPLOYEES.EMPLOYEE_ID%TYPE ;
    JOBID      EMPLOYEES.JOB_ID%TYPE ;
    ENAME      EMPLOYEES.FIRST_NAME%TYPE;
    SAL_RAISE   NUMBER(3,2);
BEGIN
    EMPID := '&empid';
    DBMS_OUTPUT.PUT_LINE ('pour l''employé :'||EMPID);
    SELECT JOB_ID, FIRST_NAME INTO JOBID, ENAME FROM EMPLOYEES WHERE EMPLOYEE_ID = EMPID;
    IF JOBID = 'AC_ACCOUNT' THEN SAL_RAISE := .09;
    ELSIF JOBID = 'FI_ACCOUNT' THEN SAL_RAISE := .08;
    ELSIF JOBID = 'FI_MGR' THEN SAL_RAISE := .07;
    ELSE SAL_RAISE := 0;
    END IF;
    DBMS_OUTPUT.PUT_LINE (ENAME||' sera augmenté de '||SAL_RAISE||' job id'||JOBID);
END;
```

IF THEN ELSE

```
DECLARE
v_num NUMBER := &sv_user_num;
BEGIN
    IF MOD(v_num,2) = 0 THEN
        DBMS_OUTPUT.PUT_LINE (v_num||' est un nombre paire');
    ELSE
        DBMS_OUTPUT.PUT_LINE (v_num||' un nombre impaire');
    END IF;
DBMS_OUTPUT.PUT_LINE (' Fin');
END;
```

IF THEN ELSE

```
SET SERVEROUTPUT ON
DECLARE
v_date DATE := TO_DATE('&sv_user_date', 'DD/MM/YYYY');
v_day VARCHAR2(15);
BEGIN
v_day := TO_CHAR(v_date, 'DAY');
DBMS_OUTPUT.PUT_LINE (' ce sera un :'||v_day);
IF v_day IN ('SAMEDI', 'DIMANCHE') THEN
DBMS_OUTPUT.PUT_LINE (v_day||', week end!');
END IF;
-- control resumes here
DBMS_OUTPUT.PUT_LINE ('Done...');
END;
```

UTILISATION DU ELSIF

```
DECLARE
v_num NUMBER := &sv_num;
BEGIN
IF v_num < 0 THEN
DBMS_OUTPUT.PUT_LINE (v_num||' nombre négatif');
ELSIF v_num > 0 THEN
DBMS_OUTPUT.PUT_LINE (v_num||' nombre positif');
END IF;
DBMS_OUTPUT.PUT_LINE ('Done...');
END;
```

CASE NULLIF ET COALESCE

```
DECLARE
v_num NUMBER := &sv_user_num;
v_num_flag NUMBER;
BEGIN
v_num_flag := MOD(v_num,2);
CASE v_num_flag
WHEN 0 THEN DBMS_OUTPUT.PUT_LINE (v_num||' EST PAIRE');
ELSE DBMS_OUTPUT.PUT_LINE (v_num||' EST IMPAIRE');
END CASE;
DBMS_OUTPUT.PUT_LINE ('Done');
END;
```