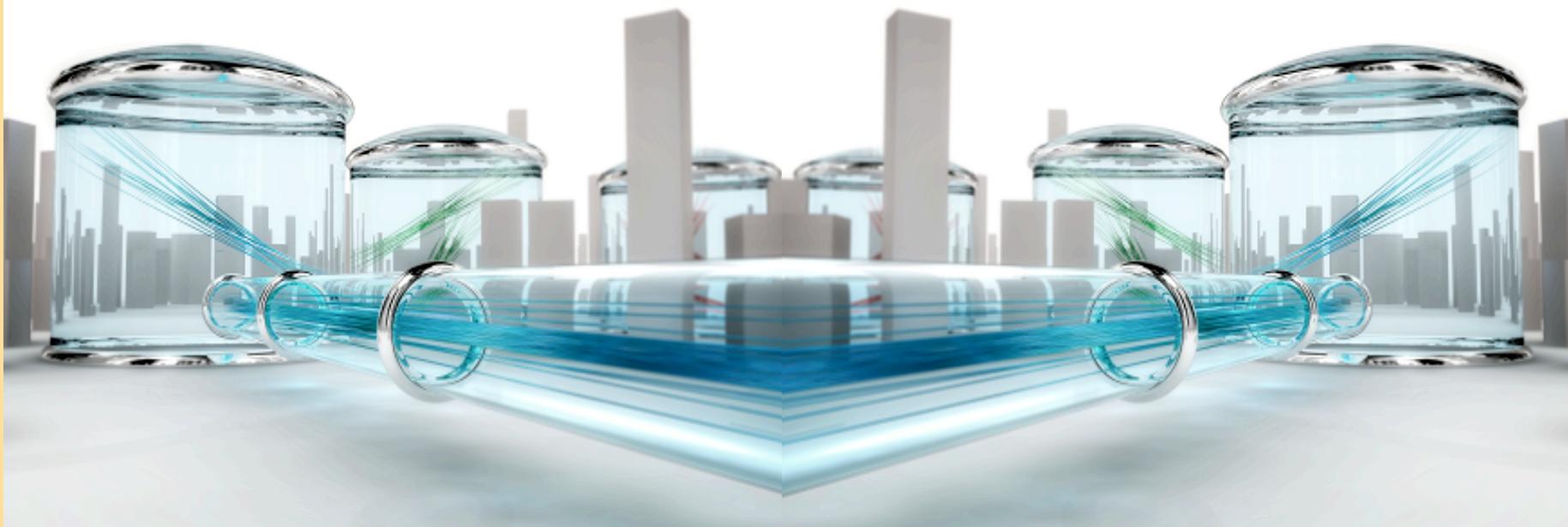


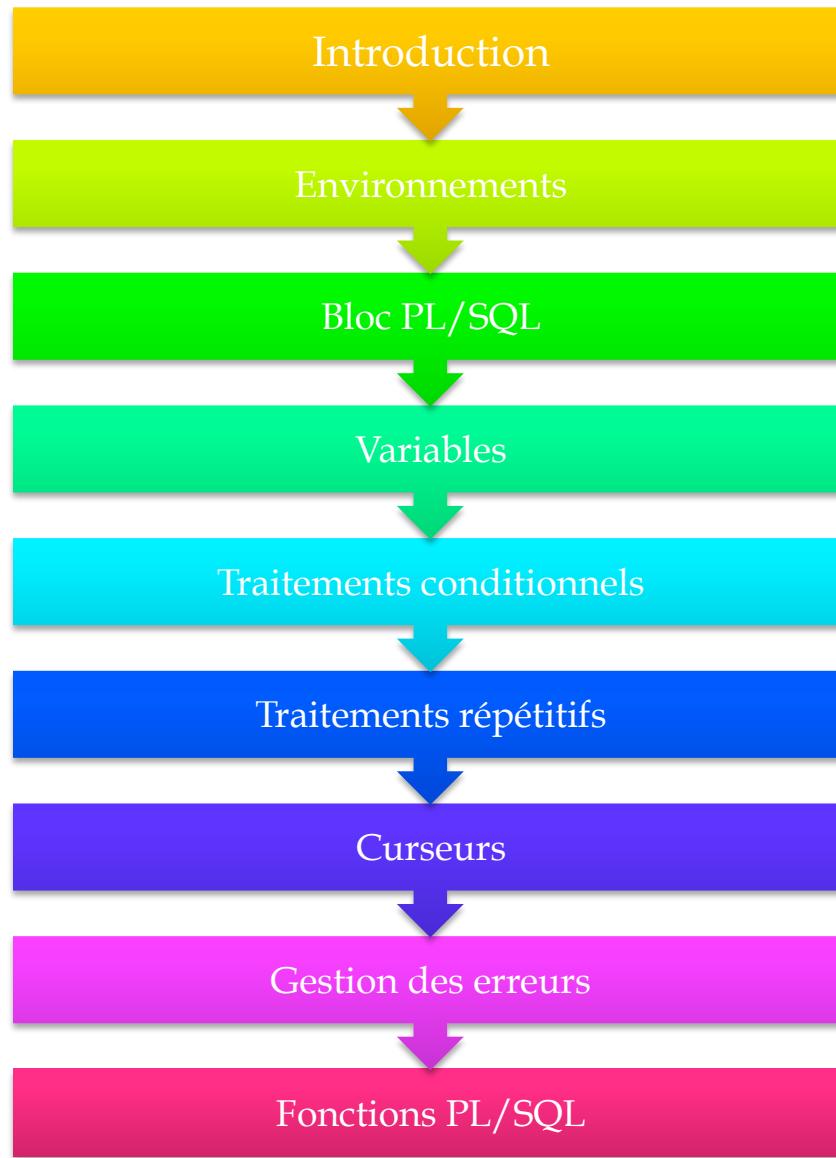
LABORATOIRE D'INFORMATIQUE DE
L'UNIVERSITE DE PAU ET DES PAYS
DE L'ADOUR

BDD avancées PLSQL



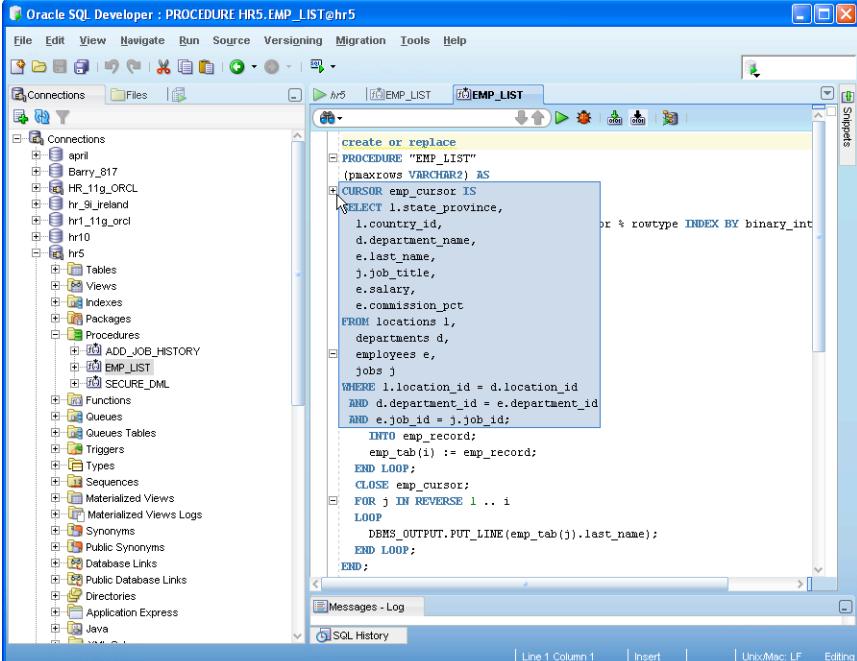
Richard Chbeir, Ph.D.

Plan



Introduction

- PL/SQL est
 - Un langage procédural
 - Utilisable dans les outils Oracle
 - Forms
 - Menu
 - SQL*Plus
 - SQL*DBQ
 - **SQL*Developer**
 - Etc.



The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : PROCEDURE HR\$.emp_list@hr\$". The left pane displays a database schema tree for the "hr\$" schema, including connections, tables, views, indexes, packages, procedures, functions, queues, triggers, types, sequences, materialized views, synonyms, public synonyms, database links, directories, application express, and Java. The right pane contains the PL/SQL code for the "EMP_LIST" procedure:

```

create or replace
PROCEDURE "EMP_LIST"
(pmaxrows VARCHAR2) AS
  CURSOR emp_cursor IS
    SELECT l.state_province,
           l.country_id,
           d.department_name,
           e.last_name,
           j.job_title,
           e.salary,
           e.commission_pct
    FROM locations l,
         departments d,
         employees e,
         jobs j
   WHERE l.location_id = d.location_id
     AND d.department_id = e.department_id
     AND e.job_id = j.job_id;
  emp_record;
  emp_tab(i) := emp_record;
END LOOP;
CLOSE emp_cursor;
FOR j IN REVERSE 1 .. i
LOOP
  DBMS_OUTPUT.PUT_LINE(emp_tab(j).last_name);
END LOOP;
END;
  
```

Below the code editor, there are tabs for "Messages - Log" and "SQL History". At the bottom, status bars show "Line 1 Column 1", "Insert", "Unix/Mac LF", and "Editing".

Introduction

- Pourquoi
 - SQL est ensembliste et non procédurale
 - PL/SQL est procédural qui intègre des ordres SQL
 - LID
 - SELECT
 - LMD
 - INSERT, UPDATE, DELETE
 - Transactions
 - COMMIT, ROLLBACK, SAVEPOINT, etc.
 - Fonctions
 - TO_CHAR, TO_DATE, UPPER, SUBSTR, ROUND, etc.

Fonctionnalités

- Définition de variables
- Traitements conditionnels
- Traitements répétitifs
- Traitements des curseurs
- Traitements des erreurs

Environnement PL/SQL

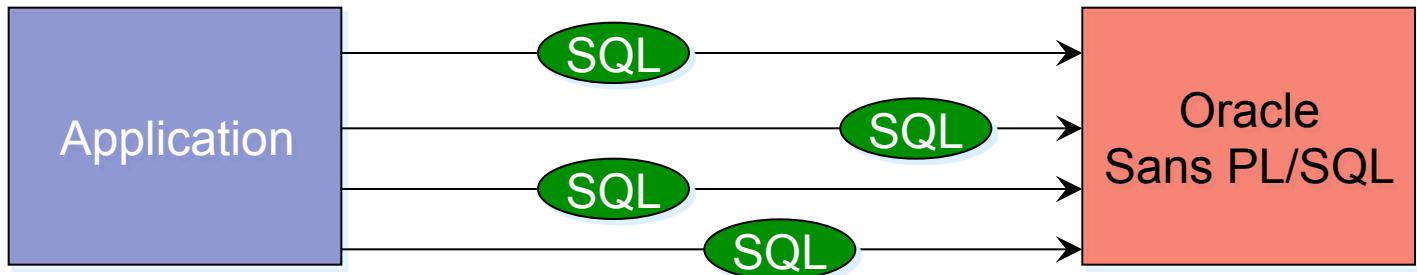
- Fonctionnement SQL et PL/SQL
- Moteur PL/SQL

Fonctionnement SQL et PL/SQL

- Moteurs
 - Moteur SQL
 - Nommé SQL STATEMENT EXECUTOR
 - Se trouve toujours avec le noyau du SGBD
 - Moteur PL
 - Nommé PROCEDURAL STATEMENT EXECUTOR
 - Peut se trouver
 - Avec le noyau du SGBD
 - Avec l'outil

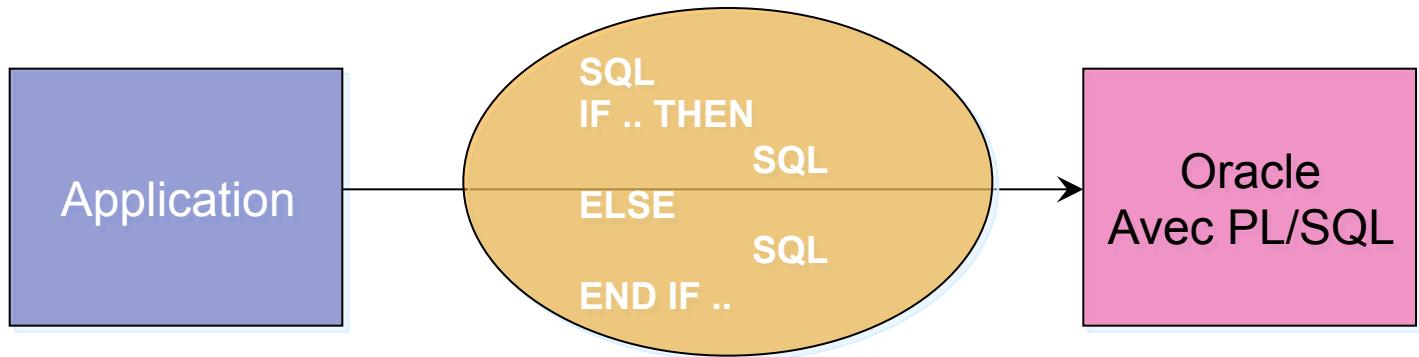
Moteur SQL

- Interprète les commandes une à une



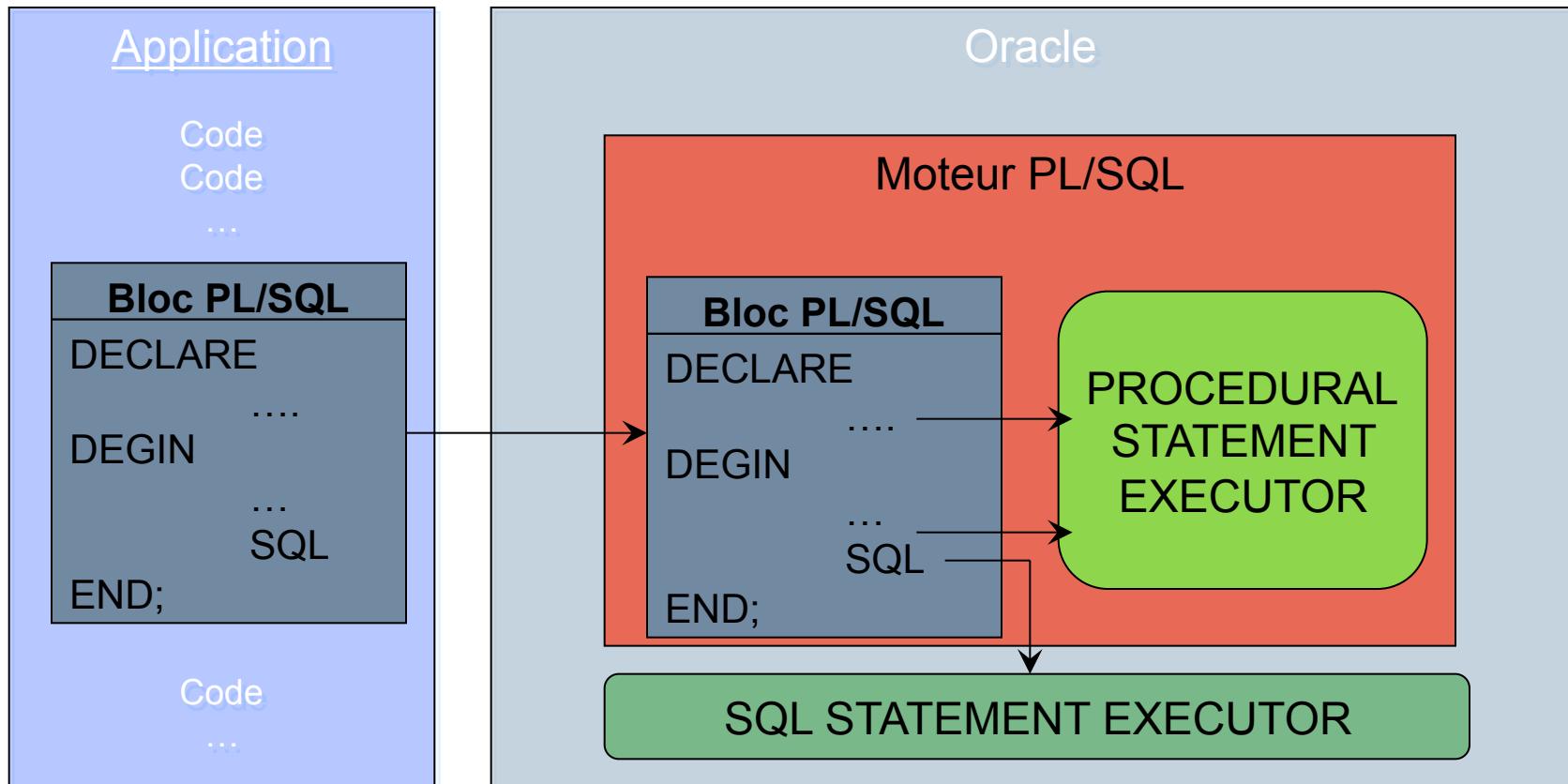
Moteur PL/SQL

- Interprète des blocs de commandes



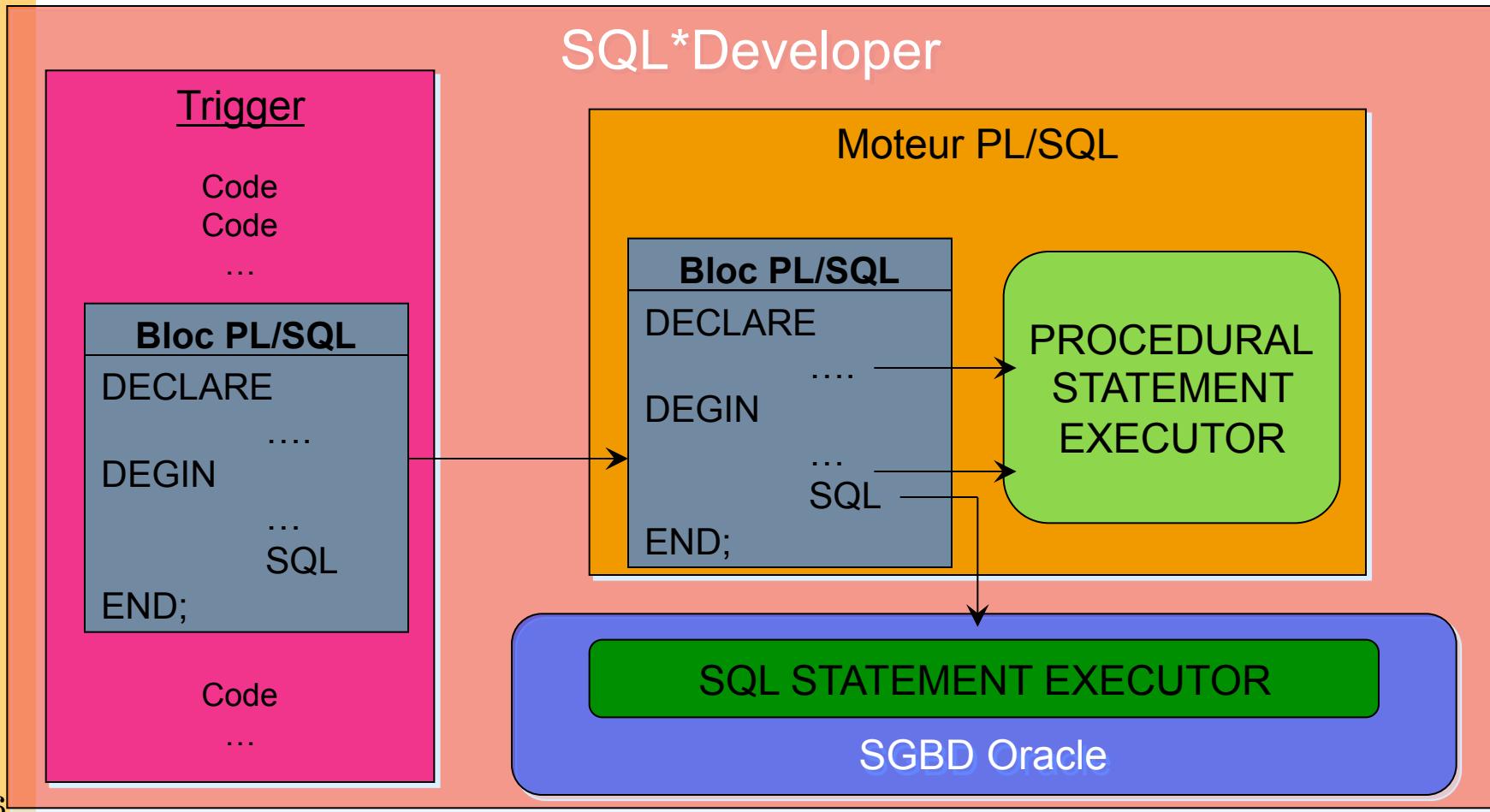
Moteur PL/SQL

- À l'intérieur du SGBD



Moteur PL/SQL

- Dans les outils



Bloc PL/SQL

- PL/SQL n'interprète pas une commande mais un ensemble de commandes contenu dans un **bloc PL/SQL**
- 2 types de blocs
 - Anonyme (DECLARE)
 - Nommé (Procédure, fonction, ...)

Bloc PL/SQL

- Exécution
 - Copier/coller
 - RUN ou /
 - Start ou @

Bloc PL/SQL

- Structure

[DECLARE]

 déclaration de variables, constantes, exceptions,
 curseurs

BEGIN [*nom_bloc*]

 Instructions SQL et PL/SQL

[EXCEPTION]

 traitement des exceptions et des erreurs

END [*nom_bloc*];

Bloc PL/SQL

- Structure

The diagram illustrates the syntax of a PL/SQL block. It features a vertical column of keywords in bold black font on the left, each with a red arrow pointing to its corresponding description in blue text on the right. The keywords are [DECLARE], BEGIN [nom_bloc], [EXCEPTION], and END [nom_bloc];. The descriptions are déclaration de variables, constantes, exceptions, curseurs; Instructions SQL et PL/SQL; and traitement des exceptions et des erreurs; respectively. A large red box labeled "Facultatives" encloses the descriptions of [DECLARE] and BEGIN [nom_bloc].

[DECLARE]
déclaration de variables, constantes, exceptions,
curseurs

BEGIN [nom_bloc]
Instructions SQL et PL/SQL;

[EXCEPTION]
traitement des exceptions et des erreurs

END [nom_bloc];

Facultatives

Bloc PL/SQL

- Structure
 - Chaque instruction est terminée par un ‘;’
 - Pour placer des commentaires
 - -- commentaire sur une ligne
 - /* .. */ Commentaire sur plusieurs lignes

Bloc PL/SQL

- Structure
 - Dans la section BEGIN, on peut imbriquer plusieurs sous blocs

Exemple

```

PROMPT nom du salarié désiré
ACCEPT nom_emp

...
DECLARE
    salaire          NUMBER(5);
    commission       NUMBER(5);
    dif_sal          NUMBER(5);
    message          CHAR(100);
BEGIN
    SELECT sal,comm INTO salaire,commission FROM Emp WHERE ename = '&nom_emp';
    dif_sal := salaire-commission;
    IF dif_sal > 100 THEN
        UPDATE Emp SET comm = comm/100 WHERE ename = '&nom_emp';
        message := 'commission modifiée de ' || '&nom_emp' || ' le ' || SYSDATE;
    ELSE
        message := 'commission non modifiée de ' || '&nom_emp' ;
    END IF;
    dbms_output.put_line('salaire ' || salaire); --Il faut faire "set serveroutput on" avant
    dbms_output.put_line('comm ' || commission);
    commit;
END;
/

```

Variables

- Types de variables utilisés en PL/SQL
- Déclaration de variables
- Initialisation et visibilité des variables

Variables

- Types de variables utilisés en PL/SQL
 - Variables locales
 - De type Oracle
 - Faisant référence aux schémas de la BDD
 - Variables de l'environnement extérieur à PL/SQL
 - Champs d'écran en SQL*Forms
 - Variables définies en langage hôte prefixées de « : »
 - Variables définies dans SQL*Plus prefixées de « & »

Variables

- Déclaration de variables
 - Les variables locales se définissent dans la partie DECLARE d'un bloc PL/SQL
 - Syntaxe

```
NomVariable [CONSTANT]
{type | variable%TYPE | | nom_table.nom_colonne%TYPE | nom_table
%ROWTYPE }
[NOT NULL]
[{:= | DEFAULT} expression_PLSQL]
```

Variables

- Déclaration de variables
 - Les variables locales se définissent dans la partie DECLARE d'un bloc PL/SQL
 - Syntaxe

NomVariable **[CONSTANT]**
{type | variable%TYPE | | nom_table.nom_colonne%TYPE |
nom_table%ROWTYPE }
[NOT NULL]
[{:| DEFAULT} expression_PLSQL]

Fige l'affectation

Variables

- Déclaration de variables
 - Les variables locales se définissent dans la partie DECLARE d'un bloc PL/SQL
 - Syntaxe

Rend obligatoire d'initialiser la variable

```
NomVariable [CONSTANT]
{type | variable%TYPE | | nom_table.nom_colonne%TYPE |
nom_table%ROWTYPE }
[NOT NULL]
[{:= | DEFAULT} expression_PLSQL]
```

Variables

- Déclaration de variables
 - Variables de type Oracle
 - Nom_var CHAR; --longueur maxi 225
 - Nom_var NUMBER; -- longueur maxi 38
 - Nom_var DATE; -- par défaut le format est DD-MON-YY
 - Exemple

```
DECLARE
    nom CHAR(15);
    numero NUMBER;
    Date_jour DATE;
    Salaire NUMBER(7,2);
BEGIN
    ...
END;
```

Variables

- Déclaration de variables
 - Variables de type Booléen
 - Nom_var BOOLEAN; --valeur (TRUE, FALSE, NULL)
 - Exemple

```
DECLARE
    reponse BOOLEAN;
BEGIN
    ...
END;
```

Variables

- Déclaration de variables
 - Variables faisant références au dictionnaire de données
 - Même type qu' une colonne dans une table
 - Syntaxe:
 - Nom_var table.nom_colonne%TYPE;
 - Exemple

```
DECLARE
    nom_salarie emp.ename%TYPE ;
BEGIN
...
END ;
```

Variables

- Déclaration de variables
 - Variables faisant références au dictionnaire de données
 - Même structure qu' une ligne dans une table
 - Syntaxe:
 - Nom_var table%ROWTYPE;
 - Exemple

```
DECLARE
    enreg emp%ROWTYPE;
BEGIN
...
END ;
```



ENREG . EMPNO	ENREG . ENAME	...	ENREG . DEPNO
---------------	---------------	-----	---------------

Variables

- Déclaration de variables
 - Variables de même type qu’ une autre variable précédemment définie
 - Syntaxe:
 - Nom_var nom_var_ancienne%TYPE;
 - Exemple

```
DECLARE
    commi NUMBER (7,2);
    salaire commi%TYPE;
BEGIN
    ...
END ;
```

Variables

- Initialisation des variables
 - Elle peut se faire par :
 - L'opérateur `:=` dans les sections **DECLARE**, **BEGIN**, et **EXCEPTION**

```
DECLARE
    nom CHAR(10) := 'CHBEIR';
    salaire NUMBER(7,2) := 1500;
    Réponse BOOLEAN := TRUE;
    tva CONSTANT NUMBER(7,2) := 3.14; -- la variable devient constante
    début NUMBER NOT NULL := 1000; -- interdit les valeurs non saisies
BEGIN
    ...
END;
```

Variables

- Initialisation des variables
 - Elle peut se faire par :
 - L'ordre **SELECT ... INTO ...** dans la section **BEGIN**
 - La clause **INTO** est obligatoire

```
DECLARE
    nom_emp CHAR(15);
    salaire emp.sal%TYPE;
    commission emp.comm%TYPE;
    nom_depart CHAR(15);

BEGIN
    SELECT ename, sal, comm, dname
    INTO nom_emp, salaire, commission, nom_depart -- La clause INTO est obligatoire
    FROM emp, dept
    WHERE ename = (SELECT max(ename) FROM emp) AND emp.deptno = dept.deptno;
    ...
END;
```