Bases de données
Programmation PL/SQL

LP TOSPI, IUT Montluçon, Université Blaise Pascal
2010-2011
Laurent d'Orazio

Plan

I. Vue d'ensemble et principes de fonctionnement
II. Eléments de programmation
III. Curseurs
IV. Gestion des exceptions
V. Procédures et fonctions
VI. Paquetages
VII. Triggers

Plan

A. Définition

- Acronyme

  PL SQL = Procedural SQL

- Objectif

  Proposer des fonctionnalités procédurales performantes (traitements proches des données) pour contrôler les données (gestion de tableaux, boucles, etc.)

B. Exemple introductif (1)

Exemple

Si un client essaie de retirer plus d'argent qu'il n'en a, indiquer « fonds insuffisants », sinon mettre à jour le compte.

Implantation possible en SQL

```
ACCEPT personne PROMPT 'Nom du détenteur de compte'
ACCEPT montant PROMPT 'Montant du retrait'
UPDATE Comptes SET solde=solde–&montant
        WHERE detenteur='&personne' AND solde>&montant;
SELECT 'Fonds insuffisants' FROM Comptes
        WHERE detenteur='&personne' AND solde<=&montant;
```

Pas vraiment satisfaisant...

- Expressivité peu élégante
- Exécution des 2 requêtes (pas de structure conditionnelle en SQL)

B. Exemple introductif (2)

En PL/SQL : spécification

```
create or replace procedure retrait (pers in varchar, montant in real) is
        courant real;
        begin
        select solde into courant from comptes where detenteur=pers;
        if (montant > courant) then
                raise_application_error(-20101, 'Fonds insuffisants');
        else
                update comptes set solde=solde-montant
                where detenteur=personne;
                commit;
                end if;
        end retrait;
```

B. Exemple introductif (3)

En PL/SQL : utilisations

```
SQL> EXECUTE retrait('jean',10.2);
PL/SQL procedure successfully completed.

SQL> EXECUTE retrait('jean',10000);
ERROR at line 1;
OAR-20101: Fonds insuffisants
```

Plan

A. Structure d'un programme PL/SQL

- Structure d'un programme PL/SQL

```
[declare]
  /*variables, constantes, exceptions, curseurs*/
begin
  /*Ordres SQL, instructions PL/SQL, structures de contrôles*/
[exception]
  /*Traitement des exceptions*/
end;
```

- Remarque

  Syntaxe similaire aux langages ADA, Modula et Pascal

B. Types de données

- Types de données standards SQL
  char, date, number, etc.
- Types intégrés dans PL/SQL
  boolean, binary, integer
- Types structurés PL/SQL
  record, table
- Structures de données pour la gestion de résultats de requêtes
  cursor
- Types de données définis par les utilisateurs

C. Structures conditionnelles
1. Structure IF (1)

- Objectif

  Test de condition simple

- Syntaxe

  if <condition> then

  <instruction(s)>

  end if;

- Remarque

  Possibilité d'imbrication de plusieurs conditions

C. Structures conditionnelles
1. Structure IF (2)

Exemple 1

Augmentation du salaire de 15% si la date est postérieure au 17/11/2008

Code PL/SQL

```
if date>'17-NOVEMBRE-08' then
        salaire:=salaire*1,15;
end if;
```

C. Structures conditionnelles
1. Structure IF (3)

Exemple 2

Augmentation du salaire de 15% de l'employé Simpson si la date est postérieure au 17/11/2008

Code PL/SQL

```
if date>'17-NOVEMBRE-08' then
        if employé='Simpson' then
                salaire:=salaire*1,15;
        end if;
end if;
```

C. Structures conditionnelles
2. Structure IF-THEN-ELSE (1)

- Objectif

  Test de condition simple avec traitement de la condition opposée

- Syntaxe

```
if <condition> then
    <instruction(s)>
else
    <instruction(s)>
end if;
```

- Remarque

  Possibilité d'imbrication de plusieurs conditions

C. Structures conditionnelles
2. Structure IF-THEN-ELSE (2)

Exemple 1

Augmentation du salaire de 15% si la date est postérieure au 17/11/2008, de 5% sinon

Code PL/SQL

```
if date>'17-NOVEMBRE-08' then
        salaire:=salaire*1,15;
else
        salaire:=salaire*1,05;
end if;
```

C. Structures conditionnelles
2. Structure IF-THEN-ELSE (3)

Exemple 2

Augmentation du salaire de 15% de l'employé Simpson si la date est postérieure au 17/11/2008 ou 10% sinon. Augmentation pour les autres employés de 5%.

Code PL/SQL

```
if employé='Simpson' then
        if date>'17-NOVEMBRE-08' then
                salaire:=salaire*1,15;
        else
                salaire:=salaire*1,10;
        end if;
else
        salaire:=salaire*1,05;
end if;
```

C. Structures conditionnelles
3. Structure IF-THEN-ELSIF (1)

- Objectif

  Test de plusieurs conditions

- Syntaxe

```
if <condition> then
    <instruction(s)>
elsif <condition> then
    <instruction(s)>
elsif <condition> then
    <instruction(s)>
...
end if;
```

C. Structures conditionnelles
3. Structure IF-THEN-ELSIF (2)

Exemple

Augmentation du salaire de 15% de l'employé Simpson de 10% de l'employé Flanders et de 5% pour les autres.

Code PL/SQL

```
if employé='Simpson' then
        salaire:=salaire*1,15;
elsif employé='Flanders' then
        salaire:=salaire*1,10;
else
        salaire:=salaire*1,05;
end if;
```

C. Structures conditionnelles
4. Structure CASE (1)

- Objectif

  Test de plusieurs conditions

- Différences avec if-then-else
  - Test de plusieurs valeurs en une construction
  - Définition de la valeur d'une variable

- Syntaxe

```
case <variable>
  when <expression1> then <valeur1>
  when <expression2> then <valeur2>
  ...
  else <valeurn>
end;
```

C. Structures conditionnelles
4. Structure CASE (2)

Exemple

Affichage du club de football en fonction du nom de la ville (Grenoble – GF38 , Marseille – OM, …, autre – pas d'équipe)

Rappel

Affichage en SQL : dbms_output.put_line(<chaîne de caractères>)

Code PL/SQL

```
val:=case ville
    when 'Grenoble' then 'GF38'
    ...
    else 'Pas d'équipe'
end;
dbms_output.put_line(val);
```

D. Structures itératives
1. Structure LOOP (1)

- Objectif

  Exécution à plusieurs reprises d'un groupe d'instructions

- Syntaxe

```
loop
  <instruction1>;
  ...
end loop;
```

D. Structures itératives
1. Structure LOOP (2)

Exemple

Incrémenter jusqu'à la valeur 10 un nombre initialisé à 0

Code PL/SQL

```
val:=0;
loop
    val:=val+1;
    if(val=10) then
            exit;
    end if;
end loop;
```

D. Structures itératives
2. Structure WHILE (1)

- Objectif

  Exécution d'un groupe d'instructions jusqu'à vérification d'une condition

- Syntaxe

```
while <condition> loop
  <instruction1>;
  ...
end loop;
```

D. Structures itératives
2. Structure WHILE (2)

Exemple

Incrémenter jusqu'à la valeur 10 un nombre initialisé à 0

Code PL/SQL

```
val:=0;
while var < 10 loop
   val:=val+1;
end loop;
```

D. Structures itératives
3. Structure FOR (1)

- Objectif

  Itérations d'un groupe d'instructions un certain nombre de fois

- Syntaxe

  for <variable d'itération> in <borne inf>..<borne sup> loop
    <instruction1>;
    ...
  end loop;

D. Structures itératives
3. Structure FOR (2)

Exemple

Afficher les valeurs entières de 1 à 5

Code PL/SQL

```
for compteur in 1..5 loop
    dbms_output.put_line(compteur);
end loop;
```

D. Structures itératives
3. Structure FOR (3)

- Remarque

  Possibilité de parcourir en sens inverse

- Syntaxe

  for <variable d'itération> in reverse <borne inf>..<borne sup> loop
    <instruction1>;
    ...
  end loop;

D. Structures itératives
3. Structure FOR (4)

Exemple

Afficher les valeurs entières de 1 à 5 en sens inverse

Code PL/SQL

```
for compteur in reverse 1..5 loop
    dbms_output.put_line(compteur);
end loop;
```

D. Structures itératives
4. Instruction EXIT (1)

- Objectif

  Quitter une structure itératives

- Syntaxe

  <structure itérative> loop
  <instruction1>;
  ...
  exit [when <condition>]
  end loop;

D. Structures itératives
4. Instruction EXIT (2)

Exemple

Afficher les valeurs entières de 1 à 5 et interrompre la boucle si la valeur du compteur est égale à 3

Code PL/SQL

```
for compteur in reverse 1..5 loop
    dbms_output.put_line(compteur);
    exit when compteur=3;
end loop;
```

Plan

A. Objectif et définition

- Objectif

  Récupération d'un résultat de requête sous forme d'une collection

- Exemple

  select a,b into x,y from t where clé = 123

  - Récupération de l'unique ligne du résultat de la requête
  - Placement dans le couple de variables (x,y)

- Définition

  Variable permettant d'accéder à un résultat de requête SQL représentant une collection (ensemble de n-uplets).

B. Fonctionnement

Utilisation de curseurs

1. Déclaration du curseur
2. Remplissage en une seule fois par exécution de la requête
3. Récupération des lignes une par une
   (parcours séquentiel du curseur par un pointeur logique)
4. Libération de la zone qui devient inaccessible

C. Exemple

Exemple

Calculer la somme des salaires de tous les employés

Code PL/SQL

```
declare
        cursor c is select * from Employes where salaire > 1000;
        total integer := 0;
begin
        for emp in c loop
                total := total + emp.salaire;
        end loop;
end;
```

D. Attributs sur les curseurs

- %FOUND
  VRAI si un n-uplet est trouvé
- %ISOPEN
  VRAI si le curseur est actuellement actif
- %NOTFOUND
  VRAI après la lecture du dernier n-uplet
- %ROWCOUNT
  Retourne le nombre de n-uplets dans le curseur

E. Déclaration where current of (1)

- Objectif
  - Modification via SQL sur le n-uplet courant
- Syntaxe (pour un curseur c)
  - cursor nomCurseur... for update
  - ...
  - <opération> where current of nomCurseur
- Remarque
  - Ne pas oublier de déclarer for update

E. Déclaration where current of (2)

Exemple

Augmentation de salaire des employés peu rémunérés (employés dont le salaire est inférieur à 1500 €)

Code PL/SQL

```
declare
cursor c is select * from Employes for update;
begin
  for e in c loop
    if e.salaire < 500.0 then
      update Employes set salaire=salaire*1.2 where current of c;
    end if;
  end loop;
end;
```

Plan

I. Vue d'ensemble et principes de fonctionnement
II. Eléments de programmation
III. Curseurs
IV. Exceptions
    A. Définition
    B. Gestion des exceptions
    C. Déclaration d'exceptions
    D. Exceptions prédéfinies
V. Procédures et fonctions
VI. Paquetages
VII. Triggers

A. Définition

- Définition

  Avertissement ou erreur rencontré(e) lors de l'exécution

- Exemples
  - Erreur système (espace mémoire insuffisant)
  - Erreur causée par le programme utilisateur
  - Etc.

- Remarque

  Possibilité pour les utilisateurs de définir des exceptions

  Exemple d'exception utilisateur

  Lors de l'affichage de tous les articles, si l'un dépasse 10 000 euros, le dire et arrêter l'affichage.

B. Gestion des exceptions

- Objectif de la gestion des exceptions

  Associer un traitement à des « exceptions » survenues lors de l'exécution d'un bloc PL/SQL

- Exemples
  - Notification à l'utilisateur
  - Annulation de l'opération
  - Etc.

C. Déclaration d'exceptions (1)

Syntaxe

```
[declare]
        <NomException1> exception;
        ...
begin
        <instructions>
        …
        raise <NomException1>
exception
        when <NomException1> then <instruction1>;
        ...
end;
```

C. Déclaration d'exceptions (2)

Exemple

Afficher « prix élevé » si le prix d'un article dépasse à 10000 €

Code PL/SQL

```
declare
        cursor c is select * from produits;
        depassement exception;
begin
        for e in c loop
                if e.prix > 10000 then raise depassement;end if;
        end loop;
exception
        when depassement then dbms_output.put_line('prix élevé');
end;
```

D. Exceptions prédéfinies (1)

- ZERO_DIVIDE
  Division par 0
- INVALID_CURSOR
  Tentative d'accès à un curseur non ouvert
- INVALID_NUMBER
  Utilisation d'un type non numérique dans un contexte où un nombre est requis
- NO_DATA_FOUND
  SELECT... INTO ne retourne aucun résultat

D. Exceptions prédéfinies (2)

- NOT_LOGGED_ON
  Tentative d'exécution d'opération SQL sans être connecté à Oracle
- STORAGE_ERROR
  Erreur de stockage
- VALUE_ERROR
  Erreur de conversion arithmétique, troncature, contrainte de taille, etc.
- Etc.

Plan

I. Vue d'ensemble et principes de fonctionnement
II. Eléments de programmation
III. Curseurs
IV. Exceptions
V. Procédures et fonctions
   A. Définition
   B. Déclaration de procédure
   C. Exemple de procédure
   D. Déclaration de fonction
   E. Exemple de fonction
VI. Paquetages
VII. Triggers

A. Définition

- Définition
  - Code PL/SQL
    - Stocké dans la base
    - Appelable par les couches supérieures depuis
      - Bloc PL/SQL
      - SQL*Plus

B. Déclaration de procédure

Syntaxe

```
create [or replace] p (x1 in | out t1, x2 . . . , . . .) as
        -- Déclarations
begin
        -- Corps
[ exception ]
        -- Gestion des exceptions
end;
```

C. Exemple de procédure

- Déclaration

```
create procedure supprimer (num in char) as
  begin
  delete ... where ... num ... ;
  ...
  end;
```

- Appel sous SQL*Plus

```
SQL> execute supprimer('W');
```

- Appel depuis PL/SQL

```
declare x char;
begin . . . supprimer(x); . . . end;
```

D. Déclaration de fonction

Syntaxe

```
create [or replace] f (x1 in | out t1, ...) return <type> as
        -- Déclarations
begin
        -- Corps
[exception]
        -- Gestion des exceptions
end;
```

E. Exemple de fonction

Exemple

Ecrire la fonction factorielle

Code PL/SQL

```
create function fact (n in integer) return integer as
begin
        if n = 0 then
                return 1;
        else
                return n * fact(n – 1);
        end if;
end;
```

Plan

I. Vue d'ensemble et principes de fonctionnement
II. Eléments de programmation
III. Curseurs
IV. Exceptions
V. Procédures et fonctions
VI. Paquetages
    A. Définition
    B. Déclaration
VII. Triggers

A. Définition

- Définition
  - Ensemble de procédures et fonctions
- Composition
  - Deux parties
    - En-tête
    - Corps

B. Déclaration (1)

Syntaxe

```
create package <nomPaquetage> as
  procedure <nomProcédure1> (...);
  procedure <nomProcédure2> (...);
  function <nomFonction> (...) type;
end;

create package body <nomPaquetage> as
  procedure <nomProcédure1> (...) begin ... end;
  procedure <nomProcédure2> (...) begin ... end;
  function <nomFonction> (...) begin ... end;
end;
```

B. Déclaration (2)

Exemple

```
create package Hello as
  procedure Hello1;
  procedure Hello2;
end;

create package body Hello as
  procedure Hello1 begin dbms_output.put_line('Hello1') end;
  procedure Hello2 begin dbms_output.put_line('Hello2') end;
end;
```

Plan

I. Vue d'ensemble et principes de fonctionnement
II. Eléments de programmation
III. Curseurs
IV. Exceptions
V. Procédures et fonctions
VI. Paquetages
VII. Triggers
    A. Définition
    B. Déclaration

A. Définition

- Traduction
  - Déclencheur

- Définition
  - Bloc PL/SQL
    - Associé à une table
    - Exécuté lors d'un ordre LMD sur cette table

B. Déclaration (1)

Syntaxe

create [or replace] trigger <nom>
before | after
insert | update [of <nom_attribut>] | delete | insert or delete | ... or ...
on <table>
[for each row]
[when (...)]
-- bloc PL/SQL

B. Déclaration (2)

Exemple

Notifier la suppression d'un client (de la table Clients) avant l'exécution de l'opération

Code PL/SQL

```
create trigger avant_suppression
before delete on Clients
begin
        dbms_output.put_line('suppression d'un client');
end;
```

Bibliographie

- M.C. Fauvet, Database course, Université Joseph Fourier, Grenoble
- E. Waller, Database course, Ecole Polytechnique, Paris, 2007/2008