

**Chapitre 8 Oracle PL/SQL**

Marc Philippe Parent Database Systems 6e de Connolly et Begg

# Base de données

**IFT-2004**

**Département d’informatique et de génie logiciel**

# Chapitre 8 - Objectifs

* Utiliser le langage de programmation SQL
* Utiliser les curseurs SQL
* Créer des procédures stockés (stored procedures)
* Créer des déclencheurs (triggers)
* Utiliser des déclencheurs pour implanter les contraintes d’intégrités
* Avantages et inconvénients des déclencheurs
* Requêtes récursives.

Connolly et Begg Database systems 6th 2

Département d’informatique et de génie logiciel – Marc Philippe Parent

|8:31| : apprendre a coder coté serveur,

# Langage de programmation SQL

* Conflit
  + Différents \_\_paradigmes\_\_\_\_\_\_\_\_ de programmation
  + SQL  langage déclaratif
  + Langages de haut-niveau  procéduraux ou objet
  + \_\_\_\_Structures\_\_\_\_\_\_ de données différentes
* SQL/PSM (Persistent Stored Modules)
* PL/SQL (\_\_\_\_Procedural\_\_\_\_\_\_ Language/SQL)

|8:34| : PL SQL = Oracle

* + Extension procédurale d’Oracle pour SQL
* Microsoft \_\_\_\_\_Transact SQL\_\_\_\_\_\_\_
* |8:35| : Code de serveur microsoft de BD,
* MySQL\_\_...SQL\_\_\_\_
* |8:36| : (MariaDB)

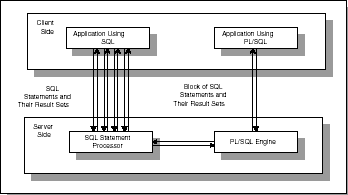
Connolly et Begg Database systems 6th 3

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Introduction PL/SQL

# |8:40| : A gauche je fais juste du SQL, mon site web, python php etc.

# |8:40| : L’idée du bloc PL SQL, on rassemble les requetes et ont les fait tous d’un seul coup.

****

Connolly et Begg Database systems 6th 4

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Déclarations

* Structure d’un programme PL/SQL

# \_\_\_declare\_\_\_\_

**section déclaration variable**

**\_\_begin\_\_\_**

**section exécution du code**

**\_\_\_\_exception\_\_\_\_\_**

**section Gestion d’exceptions end;**

**/**

* Exemple.

Connolly et Begg Database systems 6th 5

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Déclaration et types de données

# |8:46| : || C’est une concaténation

# |8:46| : ‘ ‘ séparation des strings.

**|8:48| : le select (attributs) into (variables), transfert les données dans des variables**

**|8:50| : Il faut que la requête renvoi un seul enregistrement**

**V\_NOM\_VARIABLE [CONSTANT] data\_type [not null] [:= (or default) value];**

**|8:51| :(:=) C’est un signe d’assignation pour donner une valeur à la variable.**

* Portée d’une variable
* Exemples

**V\_NUM number(2);**

**V\_NEXT\_TAX\_FILING\_DATE constant date := '15-APR-00'; V\_NOM ETUDIANT.NOM%type;**

**|8:52| : Met la variable NOM du même type du type NOM de la table.**

Connolly et Begg Database systems 6th 6

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Déclaration et types de données

* Type de données de bases SQL (rappel)

**number(précision, virgule)ou number(précision) ou number**

**char (taille), -Longueur fixe**

**varchar2 (taille),-moins de 4000 caractères DATE**

* Type de données PL/SQL
  + Types \_SQL\_\_
  + \_\_\_boolean\_\_\_\_  TRUE ou FALSE
  + binary\_integer  Entiers signés
  + \_%type\_\_\_\_\_ Même type que le champ mentionné.

Connolly et Begg Database systems 6th 7

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Opérateurs PL/SQL

* \_\_\_\_\_Arithmétiques\_\_\_\_\_\_\_\_

\*\*, \* , / , +, -

|8:55| : exposant, multiplier, division, + , -

* \_\_\_\_\_Comparaison\_\_\_\_\_\_

=, <>, !=, <, >, <=, >=, like, in, between, is null

* \_\_\_Logiques\_ and, or, not
* Chaînes de caractères

||, like

Connolly et Begg Database systems 6th 8

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Structures de contrôle (conditionnelles)

**if boolean\_exp1 then**

**<Instructions> [\_\_elsif\_\_\_ boolean\_exp2 then**

**<Instructions>]**

**...**

**[\_else\_\_\_**

**<Instructions>]**

**end if; Exemple**

Connolly et Begg Database systems 6th 9

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Structures de contrôle (boucles)

**loop**

**[ exit [when condition]];**

**<instructions>** **end loop;**

**while condition**

**loop**

**<instructions> end loop;**

**for loop\_index in**

**loop**

**<instructions> end loop;**

* Exemples

**[reverse] borne\_inf .. Borne\_sup**

Connolly et Begg Database systems 6th 10

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Exercice

* Quelles sont les sections d'un bloc PL/SQL?

A- exception, define, debut, end B- error, declare, debut, end

C- end, error, declare, begin D- exception, end, declare, begin E- define, begin, exception, end F- aucune de ces réponses

* Quelle déclaration est valide?

**A-V\_NB\_CLIENTS number(3) not null default 0; B-V\_AGE\_RETRAITE CONSTANT number(2) := 65;**

**C-V\_DATE\_NAISSANCE not null date;**

**|9:14| : on peut pas donner un not null sans donner une valeur.**

**D-V\_NOM\_PER PERSONNEL.NOM\_PER%type; E-A, C**

**F-A, B, C**

**G-A, B, D H-B, D**

**I-Toutes ces réponses**

Connolly et Begg Database systems 6th 11

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Gestion des exceptions

* Conditions \_\_\_d’erreur\_\_\_\_\_\_\_ exceptions
* Section gestion des exceptions
  + \_\_\_\_Optionnelle\_\_\_\_\_\_\_
* Vérifier erreurs pas \_\_après\_\_\_ chaque instruction
* Indique comment gérer conditions
* 2 catégories
  + \_\_\_\_Prédéfinies\_\_\_\_\_\_\_
  + \_\_\_Définies\_\_\_\_\_ par le programmeur.

|9:18| : (user-defined)

Connolly et Begg Database systems 6th 12

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Gestion de condition d’exception

* Définir \_\_\_gestionnaire\_\_\_\_\_\_\_\_\_
  + type
  + Conditions d’exception
  + Action de résolution
* Gestionnaire \_\_activité\_\_\_\_
  + Gestionnaire le plus approprié pour la condition lancée
* Exemple.

|9:20| : on doit mettre double apostrophe pour mettre une apostrophe dans le SQL.

Connolly et Begg Database systems 6th 13

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Gestion des exceptions

* Exception globales et locales

# declare

**………………… begin**

**……………………… begin**

**…………………………**

**\_\_\_exception\_\_\_\_\_\_**

**|9:21| : exception locale**

**………………………… end;**

**\_\_\_exception\_\_\_\_\_\_**

**|9:21| : exception globale**

**……………………………… end;**

* Exemple.

Exception locale

Connolly et Begg Database systems 6th 14

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Gestion des exceptions

* Exceptions prédéfinies
  + INVALID\_NUMBER
  + DUP\_VAL\_ON\_INDEX
  + NO\_DATA\_FOUND
  + VALUE\_ERROR
  + TOO\_MANY\_ROWS
  + |9:36| : select into sa prend une seule ligne, sinon: TOO many rows.
  + OTHERS.

Connolly et Begg Database systems 6th 15

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Gestion des exceptions

* Exceptions \_\_définies\_\_\_\_\_\_ par le programmeur
* Déclaration (dans la partie déclarative)

# declare

**E\_EXCEPTION\_NOM \_\_exception\_\_\_\_\_\_\_; begin**

**……………….. exception**

**when \_\_\_\_\_\_E\_EXCEPTION\_NOM\_\_\_\_\_\_\_\_\_ then …………….**

**end;**

Connolly et Begg Database systems 6th 16

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Gestion des exceptions

* Exception utilisateur déclenchée (lancée) explicitement

**declare**

**E\_EXCEPTION\_NOM exception ;**

**begin**

**…**

**if *CONDITION* then**

**raise E\_EXCEPTION\_NOM;**

**else**

**…**

**end if;**

**exception**

**when E\_EXCEPTION\_NOM then**

***instructions de traitement d’erreur*; end;**

* Exemple.

Connolly et Begg Database systems 6th 17

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Curseur

**|9:46| : erreur que je vois, on demande des requete pour de operations, plusieur etudiants font des curseurs, alors que souvent tu peux le faire avec une requete. Si vou mettez un curseur la ou vous netes pas supposé c’est pas bon. C’est bcp plus lourd. C’est a faire en dernier recours.**

* select into
  + Requête retourne \_un\_ enregistrement
* Curseurs
  + 1+ enregistrements
* Récupère tous les enregistrements
* Accède à \_un\_ à la fois
* Avance de 1 = l'enregistrement suivant
* Déclaré et \_\_\_ouvert\_\_\_ avant utilisation
* Fermé après utilisation.

Connolly et Begg Database systems 6th 18

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Curseur

* Ouvert
  + Enregistrements accédés un à la fois avec \_\_fetch\_\_\_
* Défini dans declare
* \_\_Ouvert\_\_\_\_ dans execution
* Parcouru dans une boucle avec fetch into
* Fin du curseur
  + NOM\_DU\_CURSEUR\_\_\_\_%notfound\_\_\_\_\_
* Fermé après la boucle
* Exemple.
* |9:52| : NE FAITE PAS: V\_MA\_VARIABLE := select
* c’est select … into… , fetch… into....

Connolly et Begg Database systems 6th 19

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Curseurs simplifiés

* Curseur for \_loop\_\_\_
* Syntaxe

for NOM\_VARIABLE\_ENREGISTREMENT in NOM\_CURSEUR

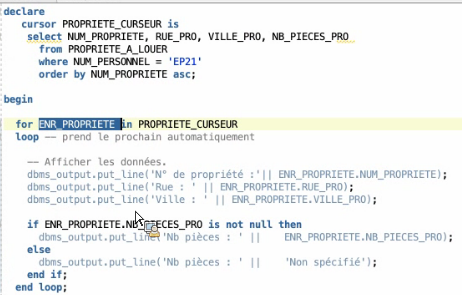
loop

--Opération avec enregistrement et attribut requête correspondante

… NOM\_VARIABLE\_ENREGISTREMENT.NOM\_ATTRIBUT

end loop;

* Exemple.



Connolly et Begg Database systems 6th 20

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Exercice

* Laquelle n'est pas une exception prédéfinie?

A- DUPLICATE\_INDEX B- NO\_DATA\_FOUND

C- OTHERS D- TOO\_MANY\_ROWS

E- VALUE\_ERROR F- aucune de ces réponses

* Quelles sont les étapes d'utilisation d'un curseur?

A-Ouverture dans boucle, fermeture après boucle, parcours dans boucle, défini dans declare

B-Ouverture avant boucle, fermeture après boucle, parcours dans boucle, défini dans declare

C-Ouverture dans declare, fermeture après boucle, parcours dans boucle, défini dans declare

D- Ouverture avant boucle, fermeture après boucle, parcours dans boucle, défini avant boucle

E-Aucune de ces réponses

Connolly et Begg Database systems 6th 21

Département d’informatique et de génie logiciel – Marc Philippe Parent

|8:33| : PL/SQL permet de faire de le programmation si on apprend Oracle on peut facilement appliquer dans un autre SGBD.

|8:34| : Les procédures et fonctions sert stocker des curseurs, des fonctions dans la base de données.

# Procédures et Fonctions

* Création de procédures et fonctions
  + Code enregistrable et réutilisable dans le SGBD
* \_\_\_Procédure\_\_\_\_\_\_

(ex Fonction avec un return VOID)

* |8:35| : Faire un traitement de donnée sans valeur de retour.
  + Traitement (pas de retour)
* \_\_\_Fonction\_\_\_\_\_
* |8:35| : Renvoi une valeur.
  + Traitement (avec retour).

Connolly et Begg Database systems 6th 22

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Procédures

**create [or replace] procedure SP\_*NOM\_PROCEDURE***

**[(ARGUMENT\_1...[,ARGUMENT\_N])] is**

**[*Section de déclaration de variable*]**

**|8:36| : braquette carée pour arguments optionel.**

**BEGIN**

***Section exécutable***

**[*Section\_gestion\_exception*] end [*SP\_NOM\_PROCEDURE*];**

**|8:37| : Obligé de mettre le SP\_NOM\_PROCEDURE selon la norme du cours. (END SP\_NOM\_PROCEDURE)**

**/**

**|8:37| : Mettez le slash pour executer la requete dun coup, super important sinon sa va planter.**

# ARGUMENT\_1...[,ARGUMENT\_N]

***P[U|I]\_NOM\_ARGUMENT* [in|out] *type* [{:=|default} valeur]**

**|8:38| : LE U et le I c’est pour dire in ou out (dans la défition du nom de l variable de facon explicite, et après le in out c’est le paramètre d’oracle c’est pour faire des passage par valeur ou reference. In c’est par valeur out c’est par référence.**

**|8:40| : et le default c’est pour crée une valeur par defaut avec assignation, on a déjà vu en C++**

* Pour \_\_éxecuter\_\_\_\_\_\_ une procédure

**execute nom\_procedure(….);**

* Exemple
* |8:44| : Faire tres attention, si vous mettez une erreur, sa dit compilé quand vous créer la procedure,
* |8:46| : Set SERVEROUPUT on; pour afficher dbms\_output.put\_line()
* |8:47| : execute SP\_SALUT\_LE\_MONDE.

|8:48| : Faire excessivement attention une erreur qui revient souvent,

|8:48| : la on arrive on fait des paramètres on met in ou out et met le type qui est number mais ironiquement ne mettez pas a cet endroit de precision ou de grandeur, sa va planter.

Create or replace function FCT\_BALANCE(P\_I\_NO\_COMPTE in number) return number (NE METTEZ PAS DES PRECISIONS A CET ENDROIT.

Ex (P\_I\_NO\_COMPTE in number(3)) --- FAITE PAS ca

Connolly et Begg Database systems 6th 23

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Fonctions

* Syntaxe

**create [or replace] function**

**FCT\_*NOM\_FUNCTION***

**[(ARGUMENT\_1...[,ARGUMENT\_N])] *return <type> is***

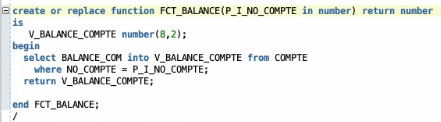
**[*Section de déclaration de variable*]**

***Section exécutable***

**[*Section\_gestion\_exception*] end [FCT\_*NOM\_FUNCTION*];**

**/**

* Exemple.



* |8:52| : La table DUAL (from DUAL) est une fausse table, elle permet d’executer une fois la fonction, elle permet de tester toutes les fonctions. Sa renvoit la valeur.

Connolly et Begg Database systems 6th 24

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Packages

* \_\_\_\_Collection\_\_\_\_\_\_ de procédures, fonctions, variables et requêtes SQL regroupées et stockées dans une seule unité
* \_\_\_\_\_\_Spécification\_\_\_\_\_\_\_
  + Déclare objets publics
* \_\_\_Corps\_\_
  + Défini objets (public et privé) du package
  + |8:55| : Sa se peut que vous voulez utiliser vous devez le nom du package
* Utiliser un objet d’un package

nomDuPackage.NomDeLObjet();

Connolly et Begg Database systems 6th 25

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Les déclencheurs

# |8:57| : le prof parle de gachette

* Trigger
  + Bloc PL/SQL
  + \_\_\_sauvegardé\_\_\_\_\_\_\_ dans la BD
  + Exécuté implicitement si événement survient
  + |8:58| : ‘est un peu de la programmation evenemenciel, on peut pas arriver et dire trigger execute toi maintenant! c’est automatique.
* \_\_\_Évenements\_\_\_\_\_\_\_
  + insert, delete, update
* Implanter règles de gestion \_\_\_\_non définies\_\_\_\_\_\_\_\_\_par CIR
* Maintenir règles de \_\_\_sécurité\_\_\_\_\_
* Générer valeurs des champs calculés
* Prévenir accès \_\_\_\_incorrects\_\_\_\_\_\_.
* |9:0| : permet de rajouter des contraintes.
* |9:0| : cest pas toujours utlisé tu peu faire des vérification au niveau de lapplication web, et a d’autre niveau.

Connolly et Begg Database systems 6th 26

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Les déclencheurs

**create [or replace] \_\_trigger\_\_\_\_\_ TRG\_XXX\_NOM\_TRIGGER**

**{before|after|instead of} *evenement\_declenchant on***

## \_NOM\_TABLE\_\_\_\_\_\_\_\_

**[for each row] [when *condition*]**

**|9:2| : Un trigger est toujours défini sur une et une seule table.**

***Bloc\_pl/sql (vu précédemment)* end;**

## Où evenement\_declenchant 1+

## |9:2| : 1 ou plusieurs séparé par une virgule.

* + insert
  + update [of LISTE\_NOM\_COLONNE]
  + delete.

Connolly et Begg Database systems 6th 27

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Les déclencheurs

* \_\_\_Before\_\_\_
  + Exécuter \_\_avant\_\_\_ l’action
    - Champs à \_\_calculer\_\_\_\_\_\_ avant
    - Vérifications à effectuer sans considérer nouvelle valeur
* \_\_\_After\_\_
  + Exécuté \_\_après\_\_\_ une action BD
  + Considère nouvelle valeur dans le calcul
* Exemples

|9:7| : exemple.

|9:7| : IF UPDATING THEN, UPDATING est une variable d’Oracle.

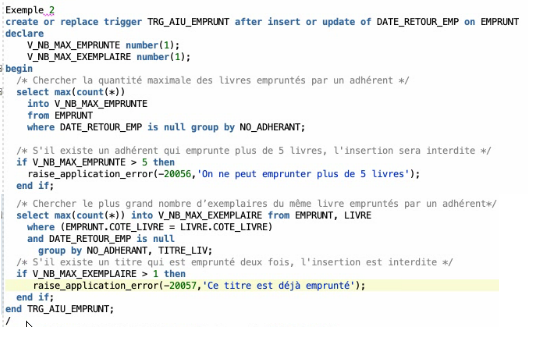
|9:14| : une daudit est une table qui permet de voir qui a mise a jour une table ou fait une supressoin dans une table c’est un log.

|9:21| : Raise\_application\_error SA ANNULE L’INSERTION, C’EST LE FUN.

|9:24| :

|9:24| : Les raise errro au dessus de -20000 c’est des messages d’erreur personnalisé.

|9:26| : Si je fais un insertion de plusieurs lignes, jai 2 choix, je peux le faire 1 fois au début des insertions ou aprèes les insertions. Ça cause un problème



2e exemple :



Connolly et Begg Database systems 6th 28

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Les déclencheurs

* :OLD.<colonne>
  + \_\_\_Ancienne\_\_\_\_\_ valeur
  + |9:48| : Sa permet de garder la valeur avant la mise a jour.
* :NEW.<colonne>
  + \_\_\_Nouvelle\_\_\_\_\_ valeur
* Seulement déclencheurs for each row
  + Erreur potentiel pour maj multiple
  + |9:49| : peut mener a une erreur de mutation pk? Si tu modifie 5 ligne et tu fais 5 vérifications dans une table, tu essai de selectionner des informations dans la table alors quelle de fait mettre a jour, on ne peut pas vérifier cela. C’est pourquoi il faut le faire ligne par ligne.
  + |9:50| : on vérifie toujours tout les abonnés.
* Utilisés comme suit
  + \_\_update\_\_\_\_ :NEW.<colonne>, :OLD.<colonne>
  + \_\_insert\_\_\_\_ :NEW.<colonne>
  + \_\_delete\_\_\_\_ :OLD.<colonne>
* Exemples.
* |10:2| : On vois sa souvent, des logs. (audit de systdate) (exemple d’étudiant)
* |10:3| : ex potentiel de probleme de mutation.
* |10:6| : si on fait des modifications sur plusieurs ligne en meme temps, on ne peux pas utiliser for each row sa fait des erreur de mutation. Les Delete from c’est commun

Connolly et Begg Database systems 6th 29

Département d’informatique et de génie logiciel – Marc Philippe Parent

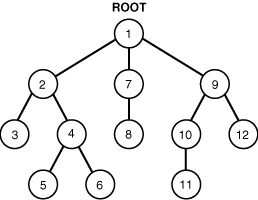
# Déclencheurs

* Avantages
  + Élimination code \_\_\_\_redondant\_\_\_\_\_
  + Simplifie maintenance
  + Augmente \_\_sécurité\_\_\_\_\_\_
  + Intégrité améliorée
  + Approprié pour client-serveur
* Inconvénients
  + \_\_\_Surcharge\_\_\_\_\_\_ performance
  + Cascade
  + |10:8| : un déclencheur peut déclencher plusieurs déclencheurs.
  + Ne peuvent être \_\_etre cédulé\_\_\_\_\_
  + Moins portable.

Connolly et Begg Database systems 6th 30

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Récursivité

* Difficile
* Requêtes entre relation et \_\_\_elle-même\_\_
* Directement ou indirectement
* |10:10| : table qui peut faire reference a une table

B et une table A.

* Permet de résoudre des \_\_hiérarchie\_\_\_\_\_\_\_\_\_
* Cas typique
  + Employé et son gestionnaire

PERSONNEL(**NUM\_PERSONNEL**, NOM\_PER, PRENOM\_PER, \_\_NUM\_PERSONNEL\_GESTIONNAIRE\_\_\_\_\_\_)

|10:11| : CE TRUC EST DANS LE TP2!.

|10:12| : Sa veut dire que l’employé 5 son gestionnaire c’est 4. et l’employé 4 son gestionnaire c’est 1. Et si on ferais des jointure direct pour trouver un patron de quelqun on ne trouvera pas facilement le patron.

|10:13| : Exemple.

|10:22| :



Connolly et Begg Database systems 6th 31

Département d’informatique et de génie logiciel – Marc Philippe Parent

# Exercice

* Laquelle est une bonne définition de paramètre?

A- P\_I\_NO\_CLIENT in number(5):= 0 B- P\_I\_NO\_CLIENT out number := 0 C- P\_I\_NO\_CLIENT in number := 0 D- P\_I\_NO\_CLIENT in number(5) = 0 E- Toutes ces réponses F- Aucune de ces réponses

* Laquelle est fausse? Les déclencheurs sont

A-Exécutés automatiquement lorsque survient un événement précis B-Exécutés automatiquement à une certaine fréquence

C-Capable d'assurer l'intégrité des données

D-Capable de modifier les données en cours de modification E-Aucune de ces réponses

* Laquelle est vraie? Les requêtes récursives

A-S'appelle elle-même B-Est l'union d'une table avec elle-même C-Est l'union et la jointure d'une table avec elle-même

D-Toutes ces réponses E-Aucune de ces réponses

Connolly et Begg Database systems 6th 32

Département d’informatique et de génie logiciel – Marc Philippe Parent