

**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,

# Dans le Langage de programmation SQL

* Il y a des Conflits car on travaille avec :
  + Différents \_\_paradigmes\_ de programmation en même temps
  + SQL est en premier lieu un langage déclaratif
  + Mais il contient aussi des langages de haut-niveau de type procéduraux ou objet
  + Il y a des aussi des\_Structures\_\_\_ de données différentes
* SQL/PSM veut dire: Persistent Stored Modules
* PL/SQL veut dire:\_\_\_Procedural\_ Language/SQL)
* Et est: L’ Extension procédurale d’Oracle
* Pour Microsoft c’est:\_\_\_\_\_Transact SQL\_\_\_\_\_\_\_
* MySQL\_\_...SQL\_\_\_\_
* |8:36| : (MariaDB)

Connolly et Begg Database systems 6th 3

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

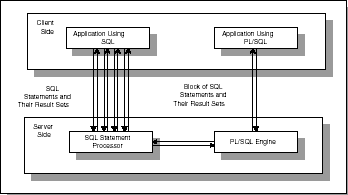
# Introduction PL/SQL

# A gauche du diagramme on fait seulement du langage : déclaratif.

# Une grande quantité de requetes venant d’une application cause : une surcharge sur le serveur.

# Le principe d’un bloc SQL est :

# De rassembler plusieurs requetes SQL et les executer de façon interne dans le serveur.

****

Connolly et Begg Database systems 6th 4

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

# Déclarations

La Structure d’un programme PL/SQL se compose de :

**Une section déclaration variable**

# \_declare\_

**une section exécution du code**

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

**une section Gestion d’exceptions**

**\_exception\_**

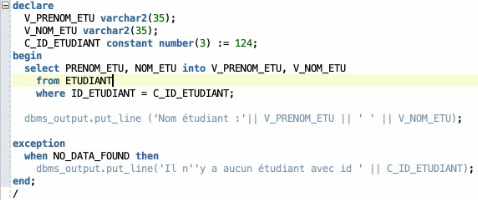
**Un programme PL/SQL se termine toujours avec :**

**end;**

**/**

**N’oubliez pas le slash!**

Completer ce code:



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];**

* Portée d’une variable
* Exemples

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

**V\_NEXT\_TAX\_FILING\_DATE constant date := '15-APR-00';**

**Mettre la Variable**

**nom du même type du champ NOM.**

**V\_NOM ETUDIANT.NOM%type;**

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**

* Qu’elles sont les Types de données dans PL/SQL
  + Tous les Types \_SQL\_déclaratif (number, char, varchar2 date, etc)
  + \_\_\_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\_\_\_\_\_\_\_\_

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

* \_\_\_\_\_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**

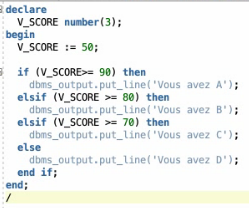
**<Instructions>]**

**...**

**[\_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**

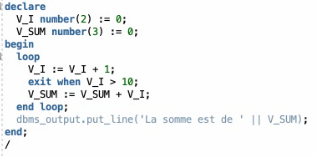
**<instructions> end loop;**

* Exemples

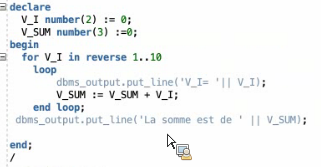
**[reverse] borne\_inf .. Borne\_sup**

Connolly et Begg Database systems 6th 10

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



Faite un decompte dans le loop de 10 a 1.



# 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 n’est pas 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;**

**D-V\_NOM\_PER PERSONNEL.NOM\_PER%type;**

**La réponse est C, un not null doit absolument définir une valeur.**

Connolly et Begg Database systems 6th 11

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

# La section Gestion des exceptions

* renferme des conditions \_d’erreur\_ des exceptions (try catch)
* La Section gestion des exceptions est

\_\_Optionnelle\_

* Elle Vérifie les erreurs pas \_\_après\_\_\_ chaque instruction
* Indique comment gérer conditions
* Il y a 2 catégories d’exception.
  + \_\_\_Prédéfinies\_\_
  + \_\_\_Définies\_\_\_\_\_ par le programmeur. (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

* Les Exceptions prédéfinies:
  + INVALID\_NUMBER quand une fonction to\_number ne peut convertir
  + DUP\_VAL\_ON\_INDEX valeur identique de clé alternative
  + NO\_DATA\_FOUND aucune donnée trouvée
  + VALUE\_ERROR rentrer un char(5) dans un char(1)
  + TOO\_MANY\_ROWS select into peut causer cette erreur.
  + 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**

**…**

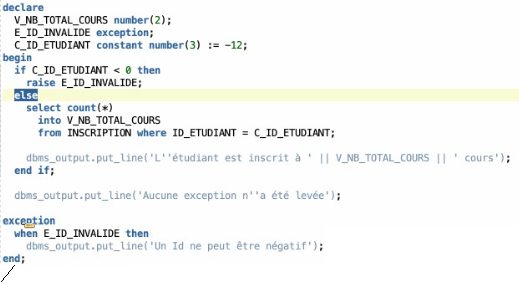
**end if;**

**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

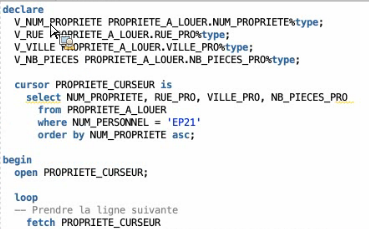
Département d’informatique et de génie logiciel – Marc Philippe Parent

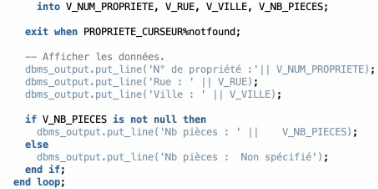
# Curseur

* 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

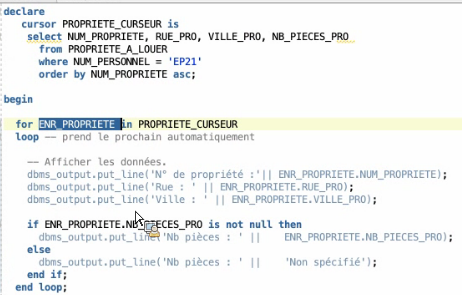
loop

--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 Pare

# Procédures et Fonctions

* Les procédures et fonctions sert à stocker des méthodes programmes dans la base de données
  + Ceci a pour but de rendre le Code enregistrable et réutilisable dans le SGBD

Une Procédure fait un traitement de données mais ne donne pas de valeur de retour

(ex Fonction avec un return VOID)

Une Fonction fait un traitement de données et envoi

une valeur de 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 optionnel.**

**BEGIN**

***Section exécutable***

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

**/**

**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]**

**LE P[U|I] c’est pour spécifier OUT ou IN dans LE NOM de la Procédure**

**IN permet de faire des passages par valeur (entrée en copie)**

**OUT permet de faire des passages par reference (adresse)**

* Pour \_éxecuter\_ une procédure il faut utiliser la commande :

**execute nom\_procedure(….);**

Faire tres attention, si vous mettez une erreur dans la procedure ou la fonction,

il est fort probable que : le tout va compiler quand même.

pour afficher dbms\_output.put\_line() Il faut utiliser la commande:

Set SERVEROUPUT on;

Il faut faire attention au typage dans les arguments :

Utiliser la précision : (P\_I\_NO\_COMPTE in number (3)) va causer une erreur.

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***

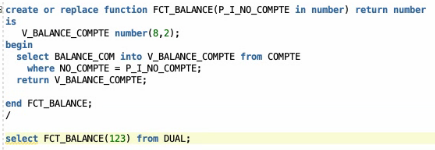
**[*Section de déclaration de variable*]**

***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

# Un Package

* Est une : Collection de procédures, fonctions, variables et requêtes SQL regroupées et stockées dans une seule unité
* Dans un package il y a une section: Spécification
  + Qui permet de : Déclarer des objets publics
* Il y a un: Corps
  + Qui permet de définir des objets (public et privé) du package
* Pour utiliser un objet d’un package vous devez écrire :

nomDuPackage.NomDeLObjet();

Connolly et Begg Database systems 6th 25

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

# Les déclencheurs / trigger / gachette

* Trigger
  + C’est un Bloc PL/SQL
  + Il est sauvegardé dans la BD
  + Le bloc est exécuté implicitement si un événement survient

Les Évenements possibles sont:

* + insert, delete, update

Le trigger permet d’implanter :

des règles de gestion non définies par CIR (contrainte d’intégrité reférenciel)

Il permet aussi de: maintenir des règles de sécurité

Il permet aussi de: générer valeurs par défaut sur des champs calculés

Il permet aussi de : prévenir accès incorrects

(faire des opérations invalides)

* |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*]**

***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

Quand on stipule la condition : Before

Le code est exécuté avant l’action (insert/update ou delete)

Les Champs sont calculé avant

Les Vérifications sont effectuées sans considérer les nouvelles valeurs.

* Quand on stipule la condition : After
  + Le code est exécuté après une action BD (insert/update ou delete)
  + On considère les nouvelles valeur dans le calcul

|9:7| : exemple.

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

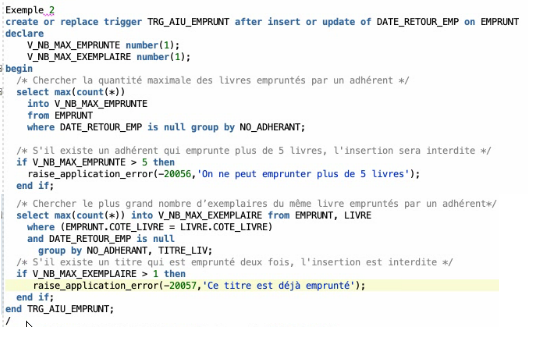
|9:14| : une daudit est une table qui permet de voir qui a mise a jour une table ou fait une supressoin dans une table c’est un log.

|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 Paren

# Les déclencheurs

La déclaration:OLD.<colonne>

* + Donne l’ancienne valeur

La déclaration :NEW.<colonne>

* + Donne la Nouvelle valeur
* Ces 2 déclarations sont disponible seulement si

for each row est spécifié.

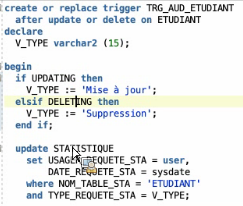
Qu’est-ce qu’un erreur de mutation?

C’est une erreur qui se déclenche lorsque le système doit mettre à jour une table en même temps qu’il doit rechercher des valeurs dans cet même table.

Pour éviter les erreurs de mutation on doit :

Spécifier la clause before et éviter de faire des requêtes qui affecte ou insert plusieurs lignes.

* 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

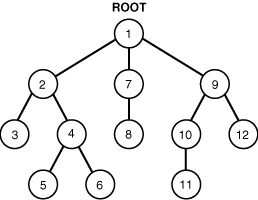
Département d’informatique et de génie logiciel – Marc Philippe Parent

# Les Déclencheurs

* Qu’elles sont les Avantages?
  + Élimination code \_redondant\_ car il est centralisé en BD
  + Simplifie maintenance car le tout est a un seul endroit.
  + Augmente la sécurité : permet de journaliser(audit) et contraindre.
  + Intégrité améliorée
  + Approprié pour client-serveur
* Qu’elles sont les Inconvénients?
  + \_Surcharge\_ performance car il y a toujours des vérifications.
  + Cascade : Plusieurs déclencheurs en déclencher d’autres.
  + Ils ne peuvent être etre cédulé
  + Moins portable car les triggers sont différent pour chaque systeme

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 (meme table)
* Directement ou indirectement
* La récursivité permet de résoudre des \_hiérarchies\_
* Cas typique
  + Employé et son gestionnaire

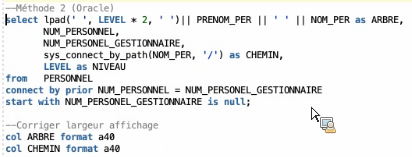
PERSONNEL(**NUM\_PERSONNEL**, NOM\_PER, PRENOM\_PER, \_\_NUM\_PERSONNEL\_GESTIONNAIRE\_\_\_\_\_\_)

|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