

Révision Chapitre 2



Contrainte d'intégrité dynamique

- Instruction:

Create or replace Trigger <Nom> {Before/ After}
{evenement [Or ...]} On <Table> [For [Each] {Row/
Statement}] 

Execute Procedure <NomP> (Argument);

Contrainte d'intégrité dynamique

- Instruction:

Create or replace Trigger <Nom> {Before/ After}
{evenement [Or ...]} On <Table> [For [Each] {Row/
Statement}]

Execute Procedure <NomP> (Argument);



- Description:

Create trigger: crée un nouveau déclencheur associé à la table spécifiée et exécute les actions précisées notamment dans <NomP>, quand certains évènements surviennent.

Contrainte d'intégrité dynamique

- Instruction:

Create or replace Trigger <Nom> {Before/ After}
{**evenement** [Or ...]} On <Table> [For [Each] {Row/
Statement}] 

Execute Procedure <NomP> (Argument);

- Description:

Evenement: Insert, Update ou Delete

Contrainte d'intégrité dynamique

- Instruction:

Create or replace Trigger <Nom> {Before/ After}
{**evenement** [Or ...]} On <Table> [For [Each] {Row/
Statement}] 

Execute Procedure <NomP> (Argument);

- Description:

Before/ After: L'appel du déclencheur peut avoir lieu avant le traitement de l'opération ou après.

Si **Before:** Le déclencheur peut ignorer l'opération sur la ligne courante ou modifier la ligne en cours d'insertion (insert, update)

Contrainte d'intégrité dynamique

- Instruction:

Create or replace Trigger <Nom> {Before/ After}
{evenement [Or ...]} On <Table> [For [Each] {Row/
Statement}]



Execute Procedure <NomP> (Argument);


- Description:

Before/ After: L'appel du déclencheur peut avoir lieu avant le traitement de l'opération ou après.

Si **After:** Toute modification, dont la dernière insertion, m à j ou suppression est visible par le déclencheur.

Contrainte d'intégrité dynamique

- Instruction:

Create or replace Trigger <Nom> {Before/ After}
{evenement [Or ...]} On <Table> [**For [Each] {Row/**
Statement}] 


Execute Procedure <NomP> (Argument);

- Description:

For Each Row: Pour chaque ligne que l'opération modifie

Contrainte d'intégrité dynamique

- Instruction:

Create or replace Trigger <Nom> {Before/ After}
{evenement [Or ...]} On <Table> [**For [Each]** {Row/
Statement}] 


Execute Procedure <NomP> (Argument);

- Description:

For Each Statement: s'exécute une seule fois pour une opération donnée (Une seule fois pour chaque instruction SQL)

Contrainte d'intégrité dynamique

- Instruction:

Create or replace Trigger <Nom> {Before/ After}
{evenement [Or ...]} On <Table> [For [Each] {Row/
Statement}] 

Execute Procedure <NomP> (Argument);

- Description

- Procédure: Create [or replace] Procedure <NomP>
[(Paramètre [In |out| In Out] typeSQL [:= | default])]
Is Bloc PL/SQL.

Contrainte d'intégrité dynamique

- Instruction:

Create or replace Trigger <Nom> {Before/ After}
{evenement [Or ...]} On <Table> [For [Each] {Row/
Statement}]



Execute Procedure <NomP> (Argument);

- Remarques

- Si plusieurs déclencheurs sont définies pour le même évènement, ils sont déclenchés suivant l'ordre alphabétique de leurs noms.
- Select ne modifie aucune ligne donc aucun déclencheur ne peut être défini.

Manuel PL/SQL

BLOC PL/SQL :

[**DECLARE** ... déclarations et initialisation]
BEGIN
 ... instructions exécutables
 [**EXCEPTION** ... interception des erreurs]
END;

Exemples de déclarations et initialisations

DECLARE
 mot CHAR(5); note NUMBER (4,2) := 10;
 x NUMBER(4) := 0;
 v1 table%**ROWTYPE**; (*type du tuple d'une table*)
 v2 table.attribut%**TYPE**; (*type d'un attribut d'une table*)

si-alors


IF condition
 THEN instructions;
 END IF;

si-alors-sinon

IF condition
 THEN instructions;
 ELSE instructions;
 END IF;

Imbrications de conditions

IF condition1 THEN instructions;
 ELSIF condition2 THEN instructions;
 ELSIF ;
 ELSE instructions;
 END IF;


CASE variable

WHEN expr1 THEN instructions1;
 WHEN expr2 THEN instructions2; ...
 WHEN exprN THEN instructionsN;
 [ELSE instructionsN+1;]
 END CASE;

CASE

WHEN condition1 THEN instructions1;
 WHEN condition2 THEN instructions2; ...
 WHEN conditionN THEN instructionsN;
 [ELSE instructionsN+1;]
 END CASE;

Boucle Pour :

FOR i IN 1..10
 LOOP instructions;
 END LOOP;

Boucle Tant que :

WHILE condition
 LOOP instructions;
 END LOOP;

Boucle Répéter :

LOOP instructions;
 EXIT WHEN condition;
 END LOOP;


Affichage à l'écran :

DBMS_OUTPUT.PUT_LINE
 (... || ||);

Exercice d'application



Chapitre 2

Un institut de sondage a réalisé une enquête sur les choix électoraux des Algériens. Les résultats de cette enquête sont modélisés dans une base de données contenant les relations suivantes (les attributs clés primaires sont soulignés et * signifie l'existence d'une clé étrangère) : 

PERSONNE(numPers, Nom, Age, Sexe, numCat*)

CATEGORIE(numCat, Intitule)

QUESTION(numQ, Description)

AVIS(numQ*, numPers*, Reponse)

Partie I : Fonctions générales de SGBD

1. Dans quels cas une insertion dans la table PERSONNE est rejetée par le SGBD ?

PERSONNE(numPers, Nom, Age, Sexe, numCat*)



Partie I : Fonctions générales de SGBD

1. Dans quels cas une insertion dans la table PERSONNE est rejetée par le SGBD ?

PERSONNE(numPers, Nom, Age, Sexe, numCat*)



Réponse :

- Problème de clé primaire : insert into PERSONNE values(1,'Ahmed Hania', 53,'feminin',3); insert into PERSONNE values(1,'Aissa Nadia', 21,'feminin',2);
- Problème de clé étrangère : insert into PERSONNE values(2,'Labdi Lina,21,'feminin',123); la catégorie 123 n'existe pas.

PERSONNE(numPers, Nom, Age, Sexe, numCat*)
AVIS(numQ*,numPers*, Reponse)

2. L'administrateur veut interdire les personnes mineurs (moins de 18 ans) à remplir le questionnaire.



a. Proposer une solution au niveau de la table
Personne

PERSONNE(numPers, Nom, Age, Sexe, numCat*)
AVIS(numQ*, numPers*, Reponse)

2. L'administrateur veut interdire les personnes mineurs (moins de 18 ans) à remplir le questionnaire.



a. Proposer une solution au niveau de la table Personne

Réponse:

Contrainte d'intégrité de type check sur l'attribut age de la table PERSONNE :

- **ALTER TABLE PERSONNE ADD CONSTRAINT CHECK_AGE CHECK(AGE>18);**


PERSONNE(numPers, Nom, Age, Sexe, numCat*)
AVIS(numQ*, numPers*, Reponse)

2. L'administrateur veut interdire les personnes mineurs (moins de 18 ans) à remplir le questionnaire.



b. Proposer une autre solution au niveau de la table AVIS :

PERSONNE(numPers, Nom, Age, Sexe, numCat*)
AVIS(numQ*,numPers*, Reponse)

2. L'administrateur veut interdire les personnes mineurs (moins de 18 ans) à remplir le questionnaire. 

b. Proposer une autre solution au niveau de la table AVIS :

- Un trigger avant l'ajout dans avis, il vérifie que l'âge de personne est supérieur à 18, sinon, il invalide la mise à jour :

PERSONNE(numPers, Nom, Age, Sexe, numCat*)
AVIS(numQ*,numPers*, Reponse)

b. Proposer une autre solution au niveau de la table AVIS :



```
CREATE OR REPLACE TRIGGER CHECK_AGE_TRIGGER
BEFORE INSERT ON AVIS
FOR EACH ROW
AGE_PERSONNE PERSONNE.AGE%TYPE; /* VARIABLE POUR RECUPERER L'AGE */
BEGIN
/* RECHERCHER L'AGE DE LA PERSONNE */
SELECT AGE TO AGE_PERSONNE FROM PERSONNE WHERE NUMPERS=:NEW.NUMPERS;
IF AGE_PERSONNE <= 18 THEN RAISE_APPLICATION_ERROR(-20001, 'VOUS N'ETES PAS AUTORISE A
REPONDRE A CE QUESTIONNAIRE');
ENDIF
END
```

PERSONNE(numPers, Nom, Age, Sexe, numCat*)
CATEGORIE(numCat, Intitule)
QUESTION(numQ, Description)
AVIS(numQ*, numPers*, Reponse)

3. Proposer une vue permettant, pour chaque question, de calculer et d'afficher le nombre de personnes qui y ont répondues.



PERSONNE(numPers, Nom, Age, Sexe, numCat*)
CATEGORIE(numCat, Intitule)
QUESTION(numQ, Description)
AVIS(numQ*, numPers*, Reponse)

3. Proposer une vue permettant, pour chaque question, de calculer et d'afficher le nombre de personnes qui y ont répondues.



Réponse:

```
CREATE OR REPLACE VIEW TOTAL_REPONSES_QUESTION
AS
(SELECT COUNT(NUMPERS) AS TOTAL_REPONSES, Q.NUMQ AS NUM_QUESTION
FROM QUESTION Q, AVIS A
WHERE Q.NUMQ=A.NUMQ
GROUP BY Q.NUMQ
);
```

PERSONNE(numPers, Nom, Age, Sexe, numCat*)

CATEGORIE(numCat, Intitule)

QUESTION(numQ, Description)

AVIS(numQ*, numPers*, Reponse)

4. L'administrateur veut sauvegarder dans la BD, pour chaque question, le nombre de personnes qui y ont répondu.
- a. Proposer une solution en supposant que toutes les tables sont encore vides Donner la requête SQL en donnant les catalogues modifiés par cette solution.



PERSONNE(numPers, Nom, Age, Sexe, numCat*)

CATEGORIE(numCat, Intitule)

QUESTION(numQ, Description)

AVIS(numQ*, numPers*, Reponse)

4. L'administrateur veut sauvegarder dans la BD, pour chaque question, le nombre de personnes qui y ont répondu.
- a. Proposer une solution en supposant que toutes les tables sont encore vides Donner la requête SQL en donnant les catalogues modifiés par cette solution.

Réponse:



- a.1) ALTER TABLE QUESTION ADD TOTAL_REPONSE INTEGER default 0 ;

PERSONNE(numPers, Nom, Age, Sexe, numCat*)

CATEGORIE(numCat, Intitule)

QUESTION(numQ, Description)

AVIS(numQ*, numPers*, Reponse)

4. L'administrateur veut sauvegarder dans la BD, pour chaque question, le nombre de personnes qui y ont répondu.
- a. Proposer une solution en supposant que toutes les tables sont encore vides Donner la requête SQL en donnant les catalogues modifiés par cette solution.

Réponse:



- a.1) ALTER TABLE QUESTION ADD TOTAL_REPONSE INTEGER default 0 ;
- **a.2) La création d'un nouvel attribut a pour conséquences:**
 - Dans le catalogue attribut : l'insertion d'une ligne décrivant l'attribut (Nom attribut : TOTAL_REPONSE, type : integer,)
 - Dans le catalogue relation : Modification de la cardinalité de la relation attribut : card = card + 1
 - Modification de la relation question: Degré = Degré + 1, Taille d'un tuple = Taille d'un tuple + Longueur d'un entier, Version = version +1.

PERSONNE(numPers, Nom, Age, Sexe, numCat*)

CATEGORIE(numCat, Intitule)

QUESTION(numQ, Description)

AVIS(numQ*, numPers*, Reponse)

4. L'administrateur veut sauvegarder dans la BD, pour chaque question, le nombre de personnes qui y ont répondu. 📢
- b. Ecrire le script permettant de mettre à jour automatiquement ce nombre.

PERSONNE(numPers, Nom, Age, Sexe, numCat*)

CATEGORIE(numCat, Intitule)

QUESTION(numQ, Description)

AVIS(numQ*, numPers*, Reponse)

4. L'administrateur veut sauvegarder dans la BD, pour chaque question, le nombre de personnes qui y ont répondu.

b. Ecrire le script permettant de mettre à jour automatiquement ce nombre. 

```
CREATE OR REPLACE TRIGGER MAJ_NBREPONSES_QUESTION_TRIGGER
AFTER INSERT OR DELETE ON AVIS
FOR EACH ROW
BEGIN
IF INSERTING THEN UPDATE QUESTION SET TOTAL_REPONSE=TOTAL_REPONSE+1
WHERE NUMQ=:NEW.NUMQ
ELSEIF DELETING THEN UPDATE QUESTION SET TOTAL_REPONSE=TOTAL_REPONSE-1
WHERE NUMQ=:OLD.NUMQ
END IF
END
```

PERSONNE(numPers, Nom, Age, Sexe, numCat*)

CATEGORIE(numCat, Intitule)

QUESTION(numQ, Description)

AVIS(numQ*, numPers*, Reponse)

5. Comment adapter la solution donnée en 4.a pour qu'elle soit applicable au cas où toutes les tables contiennent déjà des données.



PERSONNE(numPers, Nom, Age, Sexe, numCat*)

CATEGORIE(numCat, Intitule)

QUESTION(numQ, Description)

AVIS(numQ*, numPers*, Reponse)

5. Comment adapter la solution donnée en 4.a pour qu'elle soit applicable au cas où toutes les tables contiennent déjà des données.

Réponse :



Après la création de la colonne, utiliser la vue définie dans 4 pour initialiser la colonne

```
CREATE OR REPLACE PROCEDURE INITIALISER_TOTAL_REPONSE(NUM_QUESTION.NUMQ%TYPE)
AS
BEGIN
UPDATE QUESTION SET TOTAL_REPONSE:=SELECT TOTAL_REPONSES FROM TOTAL_REPONSES_QUESTION
WHERE NUM_QUESTION=NUM_QUESTION;
END
```