bases de données avancées

Bases de données Actives : Les déclencheurs (Trigger)

Présenté par:
Amal HALFAOUI (Epse GHERNAOUT)
amal.halfaoui@univ-tlemcen.dz
amal.halfaoui@gmail.com

SGBD actifs (Définition)

SGBD capable de **réagir** à des **événements** afin de **contrôler** l'intégrité, **gérer** des redondances, autoriser ou interdire des accès, **alerter** des utilisateurs, et plus généralement gérer le comportement réactif des applications. (Appelé aussi réactive ou intelligentes)

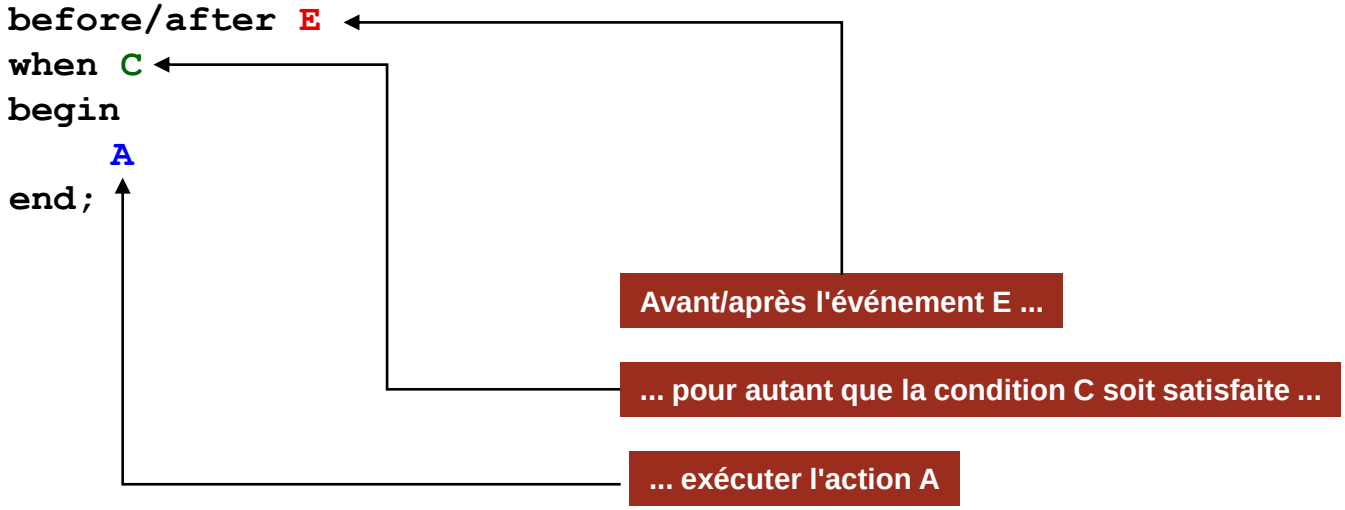
- Les mécanismes des **prédicats**, des **procédures** SQL et des **déclencheurs** permettent d'incorporer dans la base de données elle-même des composants actifs qui d'ordinaire sont inclus dans les programmes d'application.
- Elles permettent :

- de définir des contraintes d'intégrité complexes;
- de définir des comportements particuliers en cas de violation de contraintes;
- de contrôler la redondance ou de traduire des lois de comportement du domaine d'application en comportement des données;
- d' émettre des alertes.

Trigger (Définitions)

- Un déclencheur définit une action qui doit s'exécuter quand survient un évènement dans la base de données (séquence *Evénement-Condition-Action* (ECA))
- peuvent être vus comme des programmes résidents associés à un événement particulier (insertion, modification d'une ou de plusieurs colonnes, suppression) sur une table (ou une vue)

.



Trigger (Définitions)

- La majorité des déclencheurs sont programmés en PL/SQL (langage très bien adapté à la manipulation des objets., mais il est possible d'utiliser un autre langage (C ou Java par exemple).

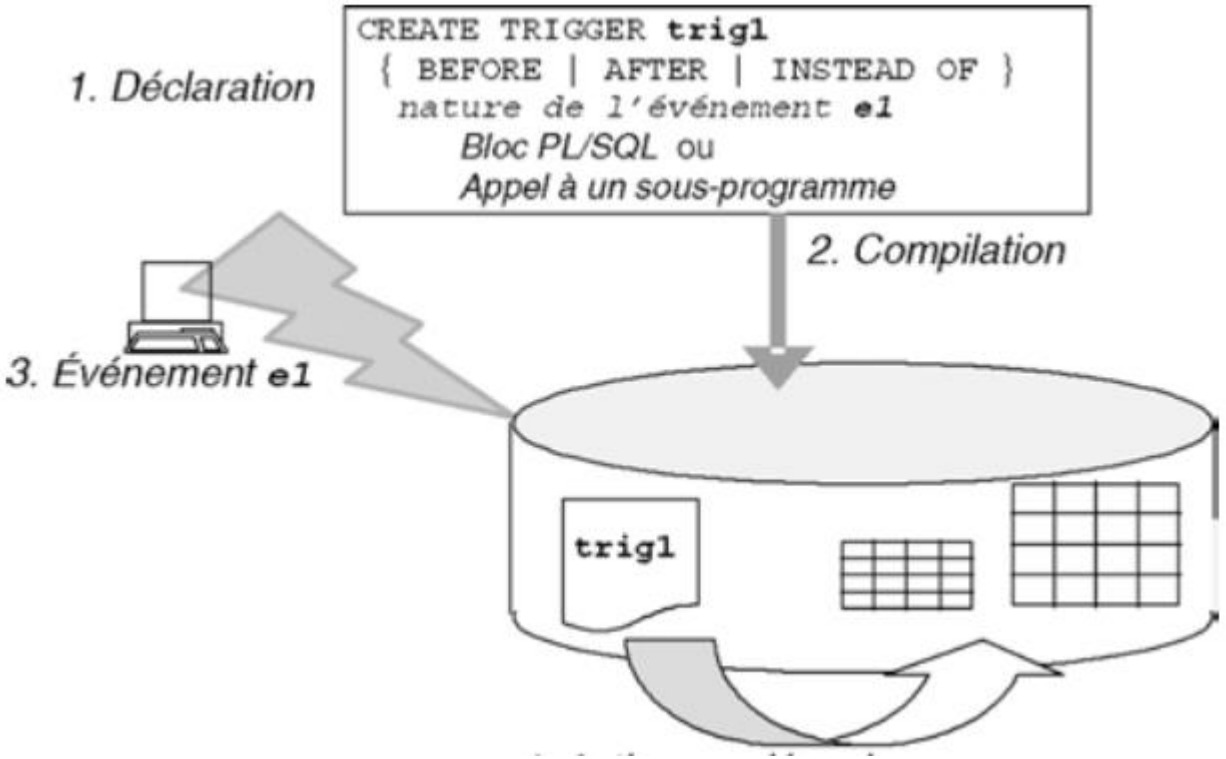
- Les événements déclencheurs peuvent être :

  - une instruction INSERT, UPDATE, ou DELETE sur une table (ou une vue) - **déclencheurs LMD**- ;
  - une instruction CREATE, ALTER, ou DROP sur un objet (table, index, séquence, etc.) -**déclencheurs LDD-** ;
  - le démarrage ou l'arrêt de la base (*startup ou shutdown), une erreur spécifique (NO_DATA*_FOUND, DUP_VAL_ON_INDEX, etc.), une connexion ou une déconnexion d'un utilisateur. - **déclencheurs d'instances-**.

- L'action associée à un déclencheur est un bloc PL/SQL enregistré dans la base. Un déclencheur est opérationnel jusqu'à la suppression de la table à laquelle il est lié.

Trigger (Définition)

- **les étapes à suivre pour mettre en œuvre un déclencheur**



Trigger (Syntaxe et principe)

bases de données avancées

```
CREATE OR REPLACE trigger <nom-trigger>
<quand> <événements> ON <table>
[FOR EACH ROW [WHEN <condition>]
BEGIN
<action>
END;
```

- Le **nom du trigger**
  - doit être unique dans un même schéma
  - peut être le nom d'un autre objet (table, vue, procédure) mais à éviter
- **quand** peut être BEFORE ou AFTER.
- **événements** spécifie DELETE, UPDATE ou INSERT séparés par des OR.
- **FOR EACH ROW** est optionnel.
- Si **UPDATE** on peut préciser les attributs concernés (UPDATE OF listeAttributs).

Trigger (Syntaxe et principe)

ORACLE propose deux types de triggers
1- les triggers **lignes** qui se déclenchent individuellement pour chaque ligne de la table affectée par le trigger,

2- les triggers **globaux** qui sont déclenchées une seule fois.

Si l'option FOR EACH ROW est spécifiée, c'est un trigger ligne, sinon c'est un trigger global.

Trigger (Syntaxe et principe)

Deux valeurs sont manipulées

- La nouvelle valeur est appelée **:new.colonne**

- L'ancienne valeur est appelée **:old.colonne**

Exemple : IF :new.salaire < :old.salaire .....

	:old	**:new**
insert	null	ligne insérée
delete	ligne supprimée	null
update	ligne avant modif	ligne après modif

Trigger (Syntaxe et principe)

• Pour les triggers lignes, on peut introduire une restriction sur les lignes à l'aide d'une expression logique SQL : *c'est la clause **WHEN** :*

- Cette expression est évaluée pour chaque ligne affectée par le trigger.
- Le trigger n'est déclenché sur une ligne que si l'expression WHEN est vérifiée pour cette ligne.
- L'expression logique ne peut pas contenir une sous-question.
- Par exemple, WHEN (:new.empno>0) empêchera l'exécution du trigger si la nouvelle valeur de EMPNO est 0, négative ou NULL.

Trigger (Syntaxe et principe)

BEFORE et AFTER:

BEFORE :

- Avant que les modifications ne soient prises de manière définitive par le SGBD
- On a toujours la possibilité de modifier les valeurs de **:new** dans le corps du déclencheur
- Avant que les contraintes statiques ne soient exécutées

AFTER:

- Les valeurs sont prises de manière définitive par le déclencheur. Pas de possibilité de modifier la valeur.
- Les contraintes statiques sont d'abord vérifiées

Trigger (Syntaxe et principe)

- **Exemple**

Enseignant(NUME, NOME, NDEPT*)
Departement (NDEPT, NOMDEPT, NBREMP)

- Mettre à jour le champ NbEmp automatiquement

Etapes :

1- sur quelle table mettre le déclencheur;

2- quel évènement;

4- déclencheur global ou ligne;

3- corps du déclencheur

Trigger (Syntaxe et principe)

- **Exemple**

Enseignant(NUME, NOME, NDEPT*)
Departement (NDEPT, NOMDEPT, NBREMP)

NUME	NOME	NDEPT
1	Allal	2
2	soufi	2
3	baghli	1

NDEPT	NOMDEPT	NBREMP
1	Informatique	1
2	Maths	2

- Mettre à jour le champ NbEmp automatiquement

Etapes :

1- sur quelle table mettre le déclencheur;

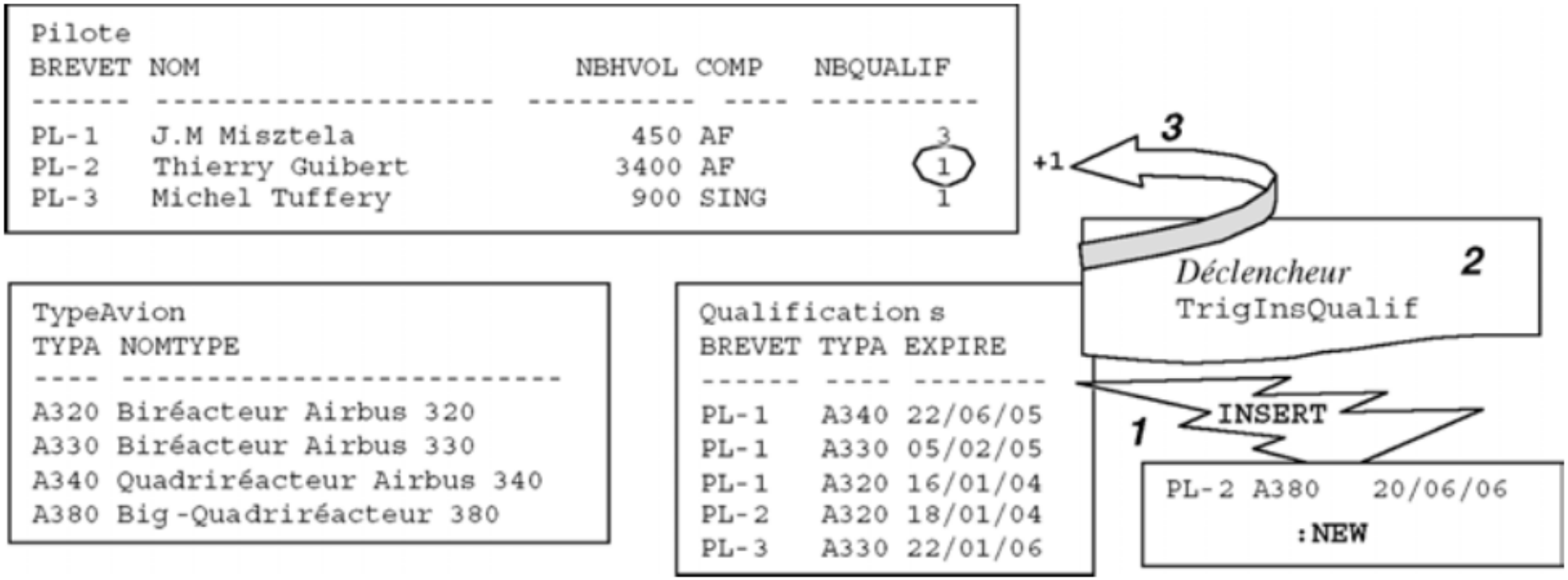
2- quel évènement;

4- déclencheur global ou ligne;

3- corps du déclencheur

Trigger (Syntaxe et principe)

- la directive :NEW



Trigger (Syntaxe et principe)

bases de données avancées

- **Exemple : TrigInsQualif**

Code PL/SQL	Commentaires

`CREATE TRIGGER TrigInsQualif`
`BEFORE INSERT ON Qualifications`
`FOR EACH ROW`
	Déclaration de l'événement déclencheur.

`DECLARE`
`v_compteur Pilote.nbHVol%TYPE;`
`v_nom Pilote.nom%TYPE;`
	Déclaration des variables locales.

`BEGIN`
`SELECT nbQualif, nom INTO v_compteur, v_nom`
`FROM Pilote WHERE brevet = :NEW.brevet;`
`IF v_compteur < 3 THEN`
`UPDATE Pilote SET nbQualif = nbQualif + 1`
`WHERE brevet = :NEW.brevet;`
`ELSE`
`RAISE_APPLICATION_ERROR(-20100, 'Le pilote '`
`|| v_nom || ' a déjà 3 qualifications!');`
`END IF;`
`END;`
`/`
	Corps du déclencheur. Extraction et mise à jour du pilote concerné par la qualification.
	
	Renvoi d'une erreur utilisateur.

Trigger (Syntaxe et principe)

bases de données avancées

- **Exemple :TrigInsQualif**

Événement déclencheur	Sortie SQL*Plus
SQL> **INSERT** INTO Qualifications VALUES (**'PL-2'**, 'A380', '20-06-2006');	1 ligne créée. SQL> SELECT * FROM Pilote; BREVET NOM NBHVOL COMP NBQUALIF ------ ---------- ------ ---- --------- PL-1 J.M Misztela 450 AF 3 **PL-2** Thierry Guibert 3400 AF **2** PL-3 Michel Tuffery 900 SING 1
SQL> **INSERT** INTO Qualifications VALUES (**'PL-1'**, 'A380', '20-06-2006');	ERREUR à la ligne 1 : ORA-20100: Le pilote J.M Misztela a déjà 3 qualifications! ORA-06512: à "SOUTOU.TRIGINSQUALIF", ligne 9 ORA-04088: erreur lors d'exécution du déclencheur 'SOUTOU.TRIGINSQUALIF'

Trigger (Syntaxe et principe)

- la directive :Old

- Exemple :TrigDelQualif

Code PL/SQL	Commentaires

`CREATE TRIGGER TrigDelQualif`
`AFTER DELETE ON Qualifications`
`FOR EACH ROW`
	Déclaration de l'événement déclencheur.

`BEGIN`
`UPDATE Pilote SET nbQualif = nbQualif - 1`
`WHERE brevet = :OLD.brevet;`
`END;`
`/`
	Corps du déclencheur. Mise à jour du pilote concerné par la suppression.

Trigger (Syntaxe et principe)

- la directive :Old

En considérant les données initiales des tables, le test de ce déclencheur sous SQL*Plus est le suivant :

Événement déclencheur	Sortie SQL*Plus

```
SQL> DELETE FROM Qualifications
           WHERE typa = 'A320';
```

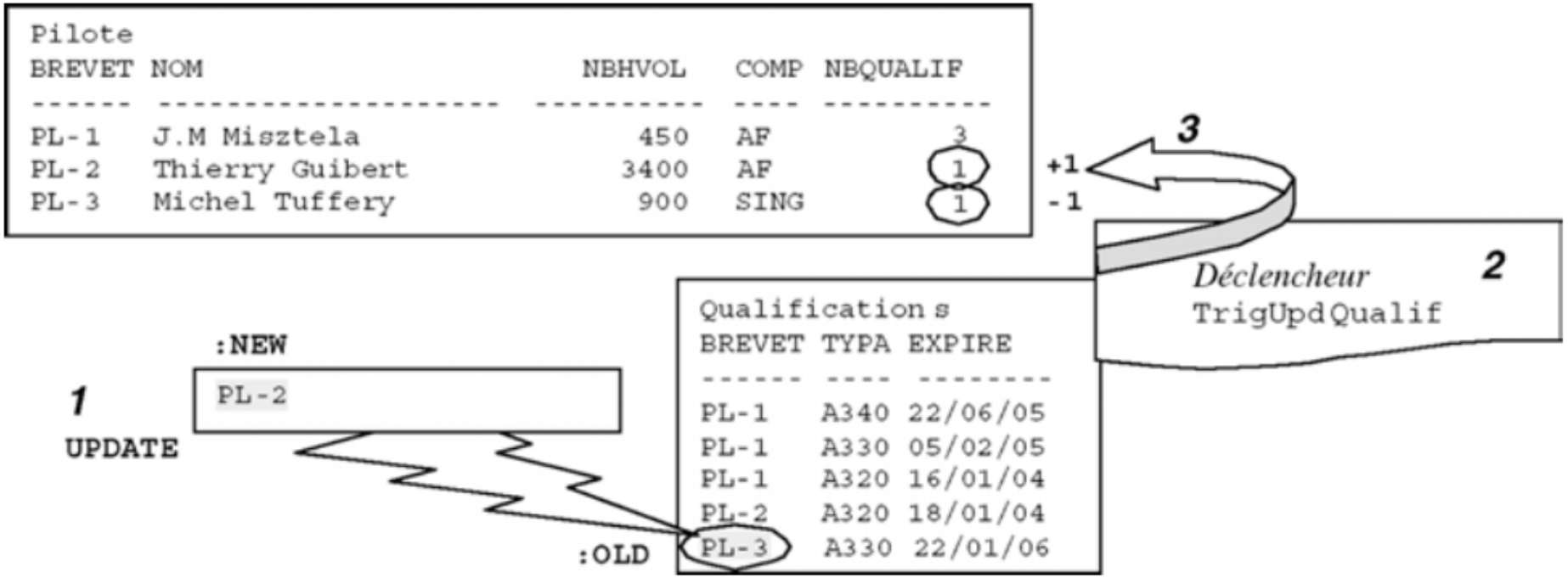
```
2 ligne(s) supprimée(s).
SQL> SELECT * FROM Pilote;
BREVET NOM              NBHVOL COMP  NBQUALIF
------ ---------------- ------ ----  -------
PL-1   J.M Misztela        450 AF          2
PL-2   Thierry Guibert    3400 AF          0
PL-3   Michel Tuffery      900 SING        1
```

Trigger (Syntaxe et principe)

- à la fois les directives :NEW et :OLD

Seuls les déclencheurs de type UPDATE FOR EACH ROW permettent de manipuler à la fois les directives :NEW et :OLD.

- Exemple : TrigUpdQualif



Trigger (Syntaxe et principe)

bases de données avancées

- **Exemple : TrigUpdQualif**

Code PL/SQL	Commentaires

`CREATE TRIGGER TrigUpdQualif`
`AFTER UPDATE OF brevet ON Qualifications`
`FOR EACH ROW`
	Déclaration de l'événement déclencheur.

`DECLARE`
`v_compteur Pilote.nbHVol%TYPE;`
`v_nom Pilote.nom%TYPE;`
	Déclaration des variables locales.

`BEGIN`
`SELECT nbQualif, nom INTO v_compteur, v_nom`
`FROM Pilote WHERE brevet = :NEW.brevet;`
`IF v_compteur < 3 THEN`
`UPDATE Pilote SET nbQualif = nbQualif + 1`
`WHERE brevet = :NEW.brevet;`
`UPDATE Pilote SET nbQualif = nbQualif - 1`
`WHERE brevet = :OLD.brevet;`
`ELSE`
`RAISE_APPLICATION_ERROR(-20100, 'Le pilote ' ||`
`:NEW.brevet || ' a déjà 3 qualifications!');`
`END IF;`
	Corps du déclencheur.
	
	Mise à jour des pilotes concernés par la modification de la qualification.
	
	Renvoi d'une erreur utilisateur.

`EXCEPTION`
`WHEN NO_DATA_FOUND THEN`
`RAISE_APPLICATION_ERROR(-20101, 'Pas de pilote`
`de code brevet ' || :NEW.brevet);`
`WHEN OTHERS THEN`
`RAISE;`
`END;`
`/`
	Renvoi d'une erreur utilisateur.
	
	Retour de l'erreur courante.

Trigger (Syntaxe et principe)

- **Exemple : TrigUpdQualif**

le test de ce déclencheur sous SQL*Plus est le suivant :

Événement déclencheur	Sortie SQL*Plus

```
SQL> UPDATE Qualifications
        SET   brevet = 'PL-2'
        WHERE brevet = 'PL-3'
        AND   typa   = 'A330';
```

```
1 ligne mise à jour.
SQL> SELECT * FROM Pilote;
BREVET NOM              NBHVOL COMP NBQUALIF
------ -------------- ------ ---- --------
PL-1      J.M Misztela       450 AF         3
PL-2      Thierry Guibert   3400 AF         2
PL-3      Michel Tuffery     900 SING       0
```

Trigger (Syntaxe et principe)

- Les prédicats conditionnels INSERTING, DELETING et UPDATING

Quand un trigger comporte plusieurs instructions de déclenchement (par exemple INSERT OR DELETE OR UPDATE), on peut utiliser des prédicats conditionnels (INSERTING, DELETING et UPDATING) pour exécuter des blocs de code spécifiques pour chaque instruction de déclenchement.

Syntaxe:
CREATE TRIGGER ...
BEFORE INSERT OR UPDATE ON nomTR
.....
BEGIN
.....
IF INSERTING THEN ..... END IF;
IF UPDATING THEN ..... END IF;
.....
END;

Trigger (Syntaxe et principe)

bases de données avancées

- **Les prédicats conditionnels INSERTING, DELETING et UPDATING**

Exemple : regroupement des deux triggers précédent en un seul

Code PL/SQL	Commentaires

`CREATE OR REPLACE TRIGGER TrigDelUpdQualif`
`AFTER DELETE OR UPDATE OF brevet ON Qualifications`
`FOR EACH ROW`
	Regroupement de deux événements déclencheurs.

`DECLARE`
`…`
`BEGIN`
`IF (DELETING) THEN`
`…`
	Bloc exécuté en cas de `DELETE`.

`ELSIF (UPDATING('brevet')) THEN`
`…`
`END IF;`
`END;`
	Bloc exécuté en cas de `UPDATE` de la colonne brevet.

Trigger (Syntaxe et principe)

- **Déclencheur LDD**

Ce sont des déclencheurs gérant les événements liés à la modification de la structure de la base et non plus à la modification des données de la base.

CREATE [OR REPLACE]TRIGGER [*schéma.] nomDéclencheur*
BEFORE | AFTER { ***actionStructureBase [OR actionStructureBase]…*** }
ON { **[*schéma.] SCHEMA* | *DATABASE*** } }
Bloc PL/SQL (variables BEGIN instructions END ;)
| CALL *nomSousProgramme(paramètres)* }

Trigger (Syntaxe et principe)

- **Déclencheur LDD**

Les principales actions sur la structure de la base prise en compte sont :

- ALTER pour déclencher en cas de modification d'un objet du dictionnaire (table, index, séquence, etc.).
- COMMENT pour déclencher en cas d'ajout d'un commentaire.
- CREATE pour déclencher en cas d'ajout d'un objet du dictionnaire.
- DROP pour déclencher en cas de suppression d'un objet du dictionnaire.
- GRANT pour déclencher en cas d'affectation de privilège à un autre utilisateur ou rôle.
- RENAME pour déclencher en cas de changement de nom d'un objet du dictionnaire.
- REVOKE pour déclencher en cas de révocation de privilège d'un autre utilisateur ou rôle.

Trigger (Syntaxe et principe)

- **Déclencheur LDD**

Le déclencheur suivant interdit toute suppression d'objet, dans le schéma *soutou*, *se produi*sant un lundi ou un vendredi.

- Exemple

Code PL/SQL	Commentaires
`CREATE TRIGGER surveilleDROPSoutou` `BEFORE DROP ON soutou.SCHEMA`	Événement déclencheur LDD.

`BEGIN`
`IF TO_CHAR(SYSDATE,'DAY') IN`
`('LUNDI ','VENDREDI') THEN`
`RAISE_APPLICATION_ERROR(-20104,'Désolé pas de destruction ce jour..') ;`
`END IF ;`
`END;`
`/`
	Corps du déclencheur.
	
	Retour d'une erreur.

Trigger (Syntaxe et principes)

bases de données avancées

- **Gestion des déclencheurs**

Un déclencheur est actif, comme une contrainte, dès sa création. Il est possible de le désactiver, de le supprimer ou de le réactiver à la demande grâce aux instructions ALTER TRIGGER (pour agir sur un déclencheur en particulier) ou ALTER TABLE (pour agir sur tous les déclencheurs d'une table en même temps).

SQL	Commentaires
`ALTER TRIGGER nomDéclencheur COMPILE;`	Recompilation d'un déclencheur.
`ALTER TRIGGER nomDéclencheur DISABLE;`	Désactivation d'un déclencheur.
`ALTER TABLE nomTable DISABLE ALL TRIGGERS;`	Désactivation de tous les déclencheurs d'une table.
`ALTER TRIGGER nomDéclencheur ENABLE;`	Réactivation d'un déclencheur.
`ALTER TABLE nomTable ENABLE ALL TRIGGERS;`	Réactivation de tous les déclencheurs d'une table.
`DROP TRIGGER nomDéclencheur;`	Suppression d'un déclencheur.

Trigger (Syntaxe et principe)

- **Gestion des déclencheurs**

Remarques

- Si le trigger est désactivé, le SGBD le stocke mais l'ignore.

- On peut désactiver un trigger si :
  - il référence un objet non disponible
  - on veut charger rapidement un volume de données important ou recharger des données déjà contrôlées.

Trigger (Syntaxe et principe)

- **Ordre d'exécution des déclencheurs**

Si plusieurs triggers sur une même table, l'ordre d'activation est :

1. tous les déclencheurs d'état BEFORE ;
2. analyse de toutes les lignes affectées par l'instruction SQL;
3. tous les déclencheurs de lignes BEFORE ;
4. verrouillage, modification et vérification des contraintes d'intégrité ;
5. tous les déclencheurs de lignes AFTER ;
6. tous les déclencheurs d'état AFTER.

Trigger (Syntaxe et principe)

bases de données avancées

- **Table Mutante**

- Il est, en principe, interdit de manipuler la table sur laquelle se porte le déclencheur dans le corps du déclencheur lui-même. Oracle parle de *mutating tables* (*erreur : ORA-04091table … en mutation, déclencheur/fonction ne peut la voir*).
- L'erreur n'est pas soulignée à la compilation mais est levée dès la première insertion.
- Cette restriction concerne les déclencheurs de lignes (FOR EACH ROW).

- Exemple :TrigMutant

Code PL/SQL	Trace SQL*Plus

`CREATE OR REPLACE TRIGGER TrigMutant1`
`AFTER INSERT ON Trace FOR EACH ROW`
`DECLARE`
`v_nombre NUMBER;`
`BEGIN`
`SELECT COUNT(*) INTO v_nombre`
`FROM Trace;`
`DBMS_OUTPUT.PUT_LINE`
`('Nombre de traces : '`
`|| v_nombre);`
`END;`
`/`
	`INSERT INTO Trace VALUES ('Insertion`
	`le ' || TO_CHAR(SYSDATE,'DD-MM-YYYY`
	`HH24:MI:SS'));`
	`ERREUR à la ligne 1 :`
	`ORA-04091: table SOUTOU.TRACE en`
	`mutation, déclencheur/fonction ne`
	`peut la voir`
	`ORA-06512: à "SOUTOU.TRIGMUTANT1",`
	`ligne 4`
	`ORA-04088: erreur lors d'exécution du`
	`déclencheur 'SOUTOU.TRIGMUTANT1'`

Trigger (Syntaxe et principe)

- **Table Mutante**

- Cette restriction concerne les déclencheurs de lignes (FOR EACH ROW).
- Oracle autorise des requêtes de sélection sur des déclencheurs BEFORE seulement au moment de L'INSERTION. Par contre, la ligne manipulée n'est pas prise en compte, il faudra donc rajouter les valeurs des champs de la ligne dans le cas des requêtes d'agrégations.

```
CREATE OR REPLACE TRIGGER TrigMutant1
BEFORE INSERT ON Trace For Each Row
DECLARE
V_nombre number;
BEGIN
SELECT COUNT(*) INTO v_nombre
FROM Trace;
DBMS_OUTPUT.PUT_LINE
('Nombre de trace : ' `|| v_nombre+1 );
END;
```

Trigger (Syntaxe et principe)

- **Gestion des exceptions**

Si une erreur se produit pendant l'exécution d'un trigger, toutes les mises à jour produites par le trigger ainsi que par l'instruction qui l'a déclenché sont défaites.

- On peut introduire des exceptions en provoquant des erreurs.

  • Une exception est une erreur générée dans une procédure PL/SQL.

  • Elle peut être prédéfinie ou définie par l'utilisateur.

  • Un bloc PL/SQL peut contenir un bloc EXCEPTION gérant les différentes erreurs possibles avec des clauses WHEN.

  • Une clause WHEN OTHERS THEN ROLLBACK; gère le cas des erreurs non prévues.

Trigger (Syntaxe et principe)

- **Gestion des exceptions**

  Exceptions prédéfinies – quelques exemples

  NO_DATA_FOUND
  cette exception est générée quand un SELECT INTO ne retourne pas de lignes

  DUP_VAL_ON_INDEX
  tentative d'insertion d'une ligne avec une valeur déjà existante pour une colonne à index unique

  ZERO_DIVIDE
  division par zéro
  etc

Trigger (Exemple)

bases de données avancées

- **Exemple**

employe(numemp,salaire,grade,...)
grille(grade,salmin,salmax)

```
/* vérifier le salaire d'un employé : */
/* s'assurer que le salaire est compris dans les bornes
correspondant au grade de l'employé */

CREATE TRIGGER verif_grade_salaire
AFTER INSERT OR UPDATE OF salaire, grade ON employe
FOR EACH ROW
DECLARE
minsal number;
maxsal number;
BEGIN
/* retrouver le salaire minimum et maximum du grade */
SELECT salmin,salmax
INTO minsal, maxsal
FROM grille
```

Trigger (Exemple)

- **Exemple**

```
WHERE grade= :new.grade;
/* s'il y a un problème, on provoque une erreur */
IF (:new.salaire<minsal OR  :new.salaire>maxsal)
THEN
        raise_application_error (-20300,'Salaire'//TO_CHAR
(:new.salaire)//
                'incorrect pour ce grade');
EXCEPTION
WHEN no_data_found THEN
        raise_application_error(-20301,'Grade incorrect');
END;
```

Déclencheurs sur les vues
Trigger INSTEAD OF

Déclencheurs sur les vues

Un déclencheur INSTEAD OF permet de mettre à jour une vue multitable qui ne pouvait être modifiée directement par INSERT, UPDATE ou DELETE.

Caractéristiques

- ne s'utilisent que sur des vues ;
- font intervenir la clause FOR EACH ROW ;
- ne font pas intervenir les options BEFORE et AFTER.
- Il n'est pas possible de spécifier une liste de colonnes dans un déclencheur INSTEAD OF UPDATE , le déclencheur s'exécutera quelle que soit la colonne modifiée.
- Il n'est pas possible d'utiliser la clause WHEN dans un déclencheur INSTEAD OF.

Déclencheurs sur les vues

Pilote

brevet	nom	nbHVol	compa
PL-1	Agnès Bidal	450	AF
PL-2	Aurélia Ente	900	AF
PL-3	Florence Périssel	1000	SING

Compagnie

comp	nrue	rue	ville	nomComp
AF	124	Port Royal	Paris	Air France
SING	7	Camparols	Singapour	Singapore AL

```
CREATE VIEW VueMultiCompPil
  AS SELECT c.comp,c.nomComp,p.brevet,p.nom,p.nbHVol
       FROM  Pilote p, Compagnie c
       WHERE p.compa = c.comp;
```

```
COMP NOMCOMP          BREVET NOM                NBHVOL
---- ---------------- ------ ---------------- ----------
AF   Air France       PL-1   Agnès Bidal          450
AF   Air France       PL-2   Aurélia Ente         900
SING Singapore AL     PL-3   Florence Périssel   1000
```

Déclencheurs sur les vues

Code PL/SQL	Commentaires

`CREATE TRIGGER TrigAulieuInsererVue`
`INSTEAD OF INSERT ON VueMultiCompPil`
`FOR EACH ROW`
`DECLARE`
`  v_comp NUMBER := 0;`
`  v_pil  NUMBER := 0;`
	Déclaration de la substitution de l'événement déclencheur.

`BEGIN`
`SELECT COUNT(*) INTO v_pil  FROM Pilote`
`  WHERE brevet = :NEW.brevet;`
`SELECT COUNT(*) INTO v_comp FROM Compagnie`
`  WHERE comp = :NEW.comp;`
`IF v_pil > 0 AND v_comp > 0 THEN`
`  RAISE_APPLICATION_ERROR(-20102, 'Le pilote`
`  et la compagnie existent déjà!');`
`ELSE`
`  IF v_comp = 0 THEN`
`  INSERT INTO Compagnie VALUES`
`  (:NEW.comp,NULL,NULL,NULL,:NEW.nomComp);`
`  END IF;`
`  IF v_pil = 0 THEN`
`  INSERT INTO Pilote VALUES`
`  (:NEW.brevet,:NEW.nom,:NEW.nbHVol,:NEW.comp);`
`  END IF;`
`END IF;`
`END;`
`/`
	Corps du déclencheur.
	Cas d'erreur.
	Ajout dans la table `Compagnie`.
	Ajout dans la table `Pilote`.

Reference

- Livre SQL pour Oracle, de C.Soutou.