Chapitre V
Triggers

Les triggers

5.1 – Généralités

- « trigger » = gachette
- déclencheur
- « trigger »= traitement déclenché sur événement
- Usage
  - vérification de certaines contraintes
  - lancement de certains traitements (ex. alerte)

5.2 – Caractéristiques

- Exprimé sous forme d'une procédure PL/SQL
- Lancement sur événement, qqf conditions
- Associé à une seule table
- Peut être actif ou inactif
- Exécution avec succès ou échec
- Peut exister des lancements en cascade
- Triggers par ordre (une fois), soit pour chaque ligne (trigger-ligne)

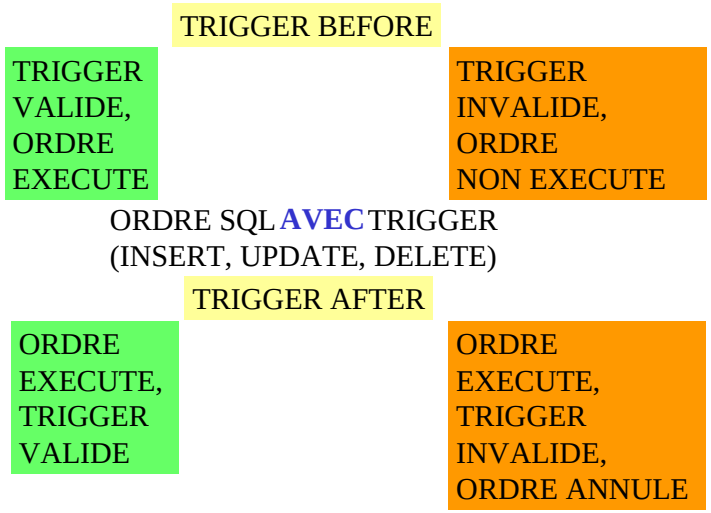
5.3 – Description d'un trigger

- Evénement
  - INSERT
  - UPDATE
  - DELETE
- Type : ligne : FOR EACH ROW
- Séquencement
  - BEFORE
  - AFTER

- Avec condition : WHEN
  - **WHEN INSERTING**
  - **WHEN DELETING**
  - **WHEN UPDATING**
- Ordres possibles
  - similaire à une procédure PL/SQL
  - en SQL: seuls **SELECT-INTO, INSERT, DELETE**

Exemple

```
• CREATE TRIGGER
  BEFORE INSERT OR UPDATE OR DELETE
    ON…..
  BEGIN
     IF INSERTING THEN ….. END IF;
     IF DELETING THEN ….. END IF;
     IF UPDATING THEN ….. END IF;
     ….
  END ;
```



5.4 – Déclencheur par ordre

- Création

```
CREATE [OR REPLACE] TRIGGER
  [schema.]nom_declencheur
sequence-trigger
événement [OR événement]
ON nom_table
Bloc_PL/SQL
```

- sequence-trigger : **BEFORE** ou **AFTER**
- événement : **INSERT** ou **UPDATE** ou **DELETE**

- Remarque
  - Si **UPDATE**, possibilité de limiter la MAJ sur certaines colonnes
- Exemple
  - **UPDATE OF** nom_colonne [, nom_colonne]

Trigger invalidé

- **RAISE_APPLICATION_ERROR (error_number, error_text)**

- Error_number : -20001 à -20999

Trigger : BEFORE

- Peut permettre de limiter l'exécution sous certaines conditions avec messages d'erreurs.

- Exemple :
  - petit contrôle de vraisemblance
  - limitation d'autorisation d'ajout d'enregistrement

```
CREATE TRIGGER ajout_pilote
BEFORE INSERT on pilote
BEGIN
  IF USER !=‘ DUPONT ’
     THEN RAISE_APPLICATION_ERROR
     (- 20001, ‘ Utilisateur non
     autorisé ’);
  ENDIF;
END;
```

Trigger : AFTER

- Validations a posteriori
- Propager des mises à jour
- Logs

- Exemple : vérifier que le nombre moyen d’heures de vol reste inférieur ou égal à 20 000

```
CREATE TRIGGER verif_nbvol
AFTER UPDATE on nbhvol OR INSERT on avion
DECLARE
  v_avg_nbhvol NUMBER
BEGIN
  SELECT AVG (nbhvol) INTO v_avg_nbhvol
  FROM avion ;
  IF v_avg_nbhvol > 20000
     THEN RAISE_APPLICATION_ERROR
     (- 20002, ‘ Le nombre d ’heures ’||
  TO_CHAR(v_avg_nbhvol)|| ’est trop
  élevé ’);
  ENDIF;
END;
```

5.5 – Déclencheur ligne

- Création

CREATE [OR REPLACE] TRIGGER
[schema.]nom_declencheur
sequence
événement [**OR** événement]
ON nom_table
REFERENCING {[OLD [AS] ancien] | **[NEW [AS]**
nouveau]}]
FOR EACH ROW
[WHEN condition
Bloc_PL/SQL

OLD et NEW

	OLD	NEW
INSERT	NULL	Valeur créée
DELETE	Valeur avant suppression	NULL
UPDATE	Valeur avant modification	Valeur après modification

OLD, NEW, :OLD, :NEW

- OLD et NEW dans les procédures
- :OLD et : NEW dans les triggers

5.6 – Gestion des triggers

- Création, exécution
  - **CREATE TRIGGER**
  - **ALTER TRIGGER**
- Information sur les triggers dans les tables
  - **USERS_TRIGGERS**
  - **ALL_TRIGGERS**
  - **DBA_TRIGGERS**

12 possibilités

BEFORE UPDATE ligne	AFTER UPDATE ligne
BEFORE DELETE ligne	AFTER DELETE ligne
BEFORE INSERT ligne	AFTER INSERT ligne
BEFORE UPDATE ordre	AFTER UPDATE ordre
BEFORE DELETE ordre	AFTER DELETE ordre
BEFORE INSERT ordre	AFTER INSERT ordre

Triggers en cascade

- Dans le corps d'un trigger, il peut y avoir **`INSERT, DELETE,`** ou **`UPDATE`**, mais **TABLES DIFFERENTES**
- Attention ne jamais modifier une colonne avec contraintes de type
  - **`PRIMARY KEY`**
  - **`UNIQUE KEY`**
  - **`FOREIGN KEY`**

Autres ordres

- Remplacement : `REPLACE`
- Suppression : `DROP TRIGGER nom_decl`
- Activation/Désactivation
  - ALTER TRIGGER nom_decl DISABLE
  - ALTER TRIGGER nom_decl ENABLE
- Activation/Désactivation sur une même table
  - ALTER TABLE DISABLE ALL TRIGGERS
  - ALTER TABLE ENABLE ALL TRIGGERS

Mutating trigger

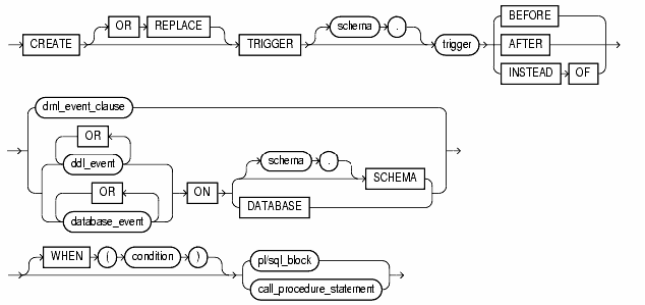
- Si trigger, la table est verrouillée :
  - impossible de la modifier ailleurs
- Si nécessité de modifier un autre enregistrement de cette table, alors rédiger une procédure

5.7 – Grammaire

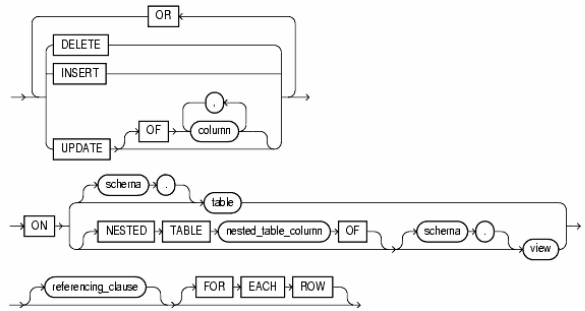
- Avoir le privilège de créer des triggers
- Triggers
  - sur un schéma
  - sur la base entière
- Les tables doivent être créées auparavant

CREATE TRIGGER

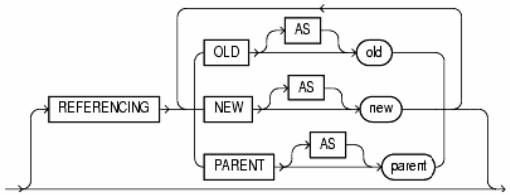
create_trigger::=



dml_event_clause::=



referencing_clause::=



Exemples

```
CREATE TRIGGER hr.emp_permit_changes
    BEFORE
    DELETE OR INSERT OR UPDATE
    ON hr.employees
       < pl/sql block >
```

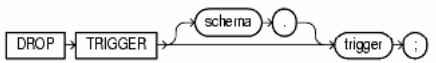
```
CREATE TRIGGER hr.salary_check
      BEFORE INSERT OR UPDATE OF salary, job_id ON hr.employees
      FOR EACH ROW
         WHEN (new.job_id <> 'AD_VP')
      < pl/sql_block >
```

```
CREATE TRIGGER hr.salary_check
   BEFORE INSERT OR UPDATE OF salary, job_id ON hr.employees
   FOR EACH ROW
   WHEN (new.job_id <> 'AD_VP')
   CALL check_sal(:new.job_id, :new.salary, :new.last_name);
```

```
CREATE TRIGGER log_errors AFTER SERVERERROR ON DATABASE
   BEGIN
      IF (IS_SERVERERROR (1017)) THEN
         <special processing of logon error>
      ELSE
         <log error number>
      END IF;
   END;
```

DROP TRIGGER

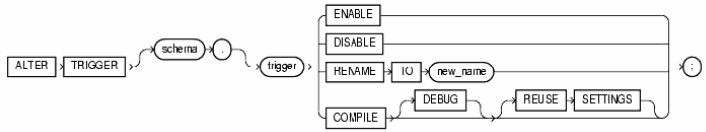
drop_trigger::=




```
DROP TRIGGER oe.order;
```

ALTER TRIGGER

alter_trigger::=



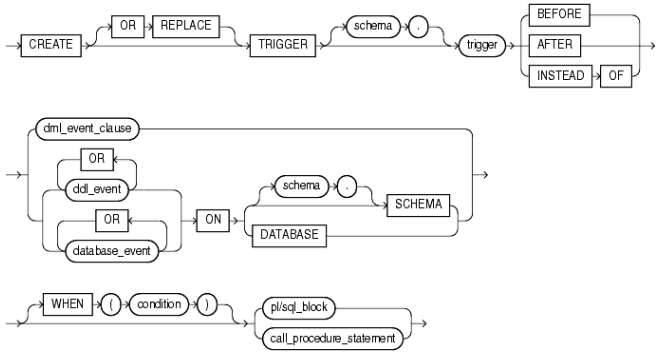

```
ALTER TRIGGER update_job_history DISABLE;
```

```
ALTER TRIGGER update_job_history ENABLE;
```

5.8 – Triggers `Instead_of`

- Mise à jour par l'intermédiaire de vues
- Possibilités de mettre à jour plusieurs tables (celles référencées dans la vue)
- Possibilité d'évaluer les états des tables avant et après (en évitant les mutating tables)

Grammaire



Exemple 1

```
CREATE VIEW order_info AS SELECT
  c.customer_id, c.cust_last_name,
  c.cust_first_name, o.order_id,
  o.order_date, o.order_status
FROM customers c, orders o
WHERE c.customer_id =
                      o.customer_id;
```

```
CREATE OR REPLACE TRIGGER order_info_insert
   INSTEAD OF INSERT ON order_info
DECLARE
  duplicate_info EXCEPTION;
  PRAGMA EXCEPTION_INIT (duplicate_info, -00001);
BEGIN
  INSERT INTO customers
    (customer_id, cust_last_name, cust_first_name)
  VALUES (
  :new.customer_id,
  :new.cust_last_name,
  :new.cust_first_name);
INSERT INTO orders (order_id, order_date, customer_id)
VALUES (
  :new.order_id,
  :new.order_date,
  :new.customer_id);
EXCEPTION
  WHEN duplicate_info THEN
    RAISE_APPLICATION_ERROR (
      num=> -20107,
      msg=> 'Duplicate customer or order ID');
END order_info_insert;
/
```

Exemple 2

```
SQL> SELECT * FROM Article WHERE noArticle = 10
  2  /

 NOARTICLE DESCRIPTION          PRIXUNITAIRE QUANTITÉENSTOCK
---------- -------------------- ------------ --------------
        10 Cèdre en boule              10,99             20

SQL> CREATE VIEW ArticlePrixPlusTaxe AS
  2   SELECT  noArticle, description, prixUnitaire * 1.15 AS prixPlusTaxe
  3   FROM  Article
  4  /

View created.

SQL> UPDATE ArticlePrixPlusTaxe
  2  SET prixPlusTaxe = 23
  3  WHERE noArticle = 10
  4  /
SET prixPlusTaxe = 23
    *
ERROR at line 2:
ORA-01733: virtual column not allowed here
```

Solution

```
SQL> CREATE OR REPLACE TRIGGER InsteadUpdate
  2  INSTEAD OF UPDATE ON ArticlePrixPlusTaxe
  3  REFERENCING
  4   OLD AS ligneAvant
  5   NEW AS ligneAprès
  6  FOR EACH ROW
  7  BEGIN
  8   UPDATE Article
  9   SET
 10  noArticle = :ligneAprès.noArticle,
 11  description = :ligneAprès.description,
 12  prixUnitaire = :ligneAprès.prixPlusTaxe / 1.15
 13   WHERE noArticle = :ligneAvant.noArticle;
 14  END;
 15  /

Trigger created.
```

Vérification

```
SQL> UPDATE ArticlePrixPlusTaxe
  2  SET prixPlusTaxe = 23
  3  WHERE noArticle = 10
  4  /

1 row updated.

SQL> SELECT * FROM Article WHERE noArticle = 10
  2  /

 NOARTICLE DESCRIPTION          PRIXUNITAIRE QUANTITÉENSTOCK
---------- -------------------- ------------ --------------
        10 Cèdre en boule                 20             20
```

= 23 – 15%

5.9 – Conclusion

- Trigger = outil puissant pour contrôle de cohérence et lancement automatique de traitement
- **Attention** aux enchaînements de triggers
- **Attention** débogage parfois difficile, notamment si « mutating trigger »
- Si limitations
  - =>écriture de procédures PL/SQL