



Université des sciences et de la Technologie Houari Boumediene
USTHB – Alger

Département d'Informatique

MASTER SYSTÈMES INFORMATIQUES INTELLIGENTS

MASTER INFORMATIQUE VISUELLE

ARCHITECTURE ET ADMINISTRATION DES BASES DE DONNÉES

CONTRAINTES D'INTÉGRITÉ

- ❖ Les données de la base doivent rester conformes à la réalité qu'elle représente
- ❖ La notion de cohérence est humaine
- ❖ Il faut la transférer au SGBD

Solution : Définition des contraintes d'intégrité (CI).

Deux types

- ❑ **Contraintes d'intégrité statiques** : celles qui correspondent à un prédicat sur l'état courant de la base.
- ❑ **Contraintes d'intégrité dynamiques** : celles qui concernent le passage d'un état à un autre.
- ❖ A tout instant de l'existence d'une base de données, on doit pouvoir définir une nouvelle contrainte d'intégrité.

CONTRAINTES STATIQUES

☐ Unicité de la clé

☐ CI individuelles

- **Plage de valeurs** : $4.000 \leq \text{salaire} \leq 20.000$,
- **Liste de valeurs** : Couleur dans [bleu, rouge, vert, jaune]
- **Contraintes entre constituants** : $\text{QTE-STOCK} \geq \text{QTE-COMMANDE}$
- **Contraintes de format** : NOM CHAR(20)

☐ Contraintes d'intégrité Statiques

☐ Contraintes d'intégrité dynamiques

L'INSTRUCTION ASSERT

❑ **ASSERT** <nom assertion> [on relation] : prédicat ;

ASSERT CI1 ON PIECE : MATERIAU IN ('fer', 'bronze', 'zinc', 'plomb') ;

❑ **SQL 99**

CREATE ASSERTION <Constraint name>

CHECK (search condition)

[<constraint attributes>]

❑ **Exemple 1**

CREATE ASSERTION constraint_1

CHECK ((SELECT AVG (Salaire) FROM
Employe >40 000))

CREATE ASSERTION constraint_1

CHECK (0 <> (SELECT COUNT(*) FROM
Table_1));

❑ **Exemple 2**

Create Assertion CityCheck Check

(NOT EXISTS (

Select *

From customer

Where customer_city is null));

ASSERTIONS VS. TRIGGERS

- ❑ Les assertions ne modifient pas les données mais vérifient seulement des conditions
- ❑ Les assertions ne sont pas liées à des tables spécifiques dans la BD ni à certains évènements.
- ❑ Les triggers vérifient des conditions et modifient les données
- ❑ Les triggers sont liés à des tables spécifiques et des évènements déclencheurs
- ❑ Toutes les assertions peuvent être implémentées par des Triggers, le contraire n'est pas toujours possible.
- ❑ Les assertions ne sont pas supportées dans tous les SGBD (Oracle par exemple)

LES TRIGGERS

Trigger = action spontanée

Exemple

- Lorsqu'un nouvel employé est inséré dans la base, le nombre d'employés figurant dans la relation rayon doit augmenter de un pour le rayon du nouvel employé

DEFINE TRIGGER EMPINS

AFTER INSERT ON Employé

(UPDATE RAYON

SET Nb-Employés = Nb-Employés + 1

WHERE Nom-Rayon = NEW Employé.Nom-Rayon);

CRÉATION DE TRIGGERS

```
CREATE TRIGGER Nom_Trigger  
  
BEFORE DELETE OR INSERT OR UPDATE ON Nom_Table  
  
FOR EACH ROW  
  
WHEN (new.empno>0)  
  
DECLARE ..... <<<<déclarations>>>>  
  
BEGIN .....  
  
<<<< bloc d'instructions PL/SQL>>>>  
  
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
- ❑ Option BEFORE/AFTER
- ❑ Elle précise le moment de l'exécution du trigger

DÉFINITION DU TRIGGER

Elle comprend le type d'instruction SQL qui déclenche le trigger :

- ❑ DELETE, INSERT, UPDATE On peut en avoir une, deux ou les trois.
- ❑ Pour UPDATE, on peut spécifier une liste de colonnes.
 - ❑ Dans ce cas, le trigger ne se déclenchera que si l'instruction UPDATE porte sur l'une au moins des colonnes précisée dans la liste.
 - ❑ S'il n'y a pas de liste, le trigger est déclenché pour toute instruction UPDATE portant sur la table.

TYPES DE TRIGGERS

❑ **Le type d'un trigger détermine :**

- ❑ quand ORACLE déclenche le trigger,
- ❑ combien de fois ORACLE déclenche le trigger.

❑ **Le type du trigger est défini par l'utilisation de l'une ou l'autre des options suivantes :**

❑ **BEFORE, AFTER, FOR EACH ROW**

❑ **ORACLE propose deux types de triggers**

- ❑ les triggers lignes qui se déclenchent individuellement pour chaque ligne de la table affectée par le trigger,
- ❑ les triggers globaux qui sont déclenchés une seule fois.

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

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.

CORPS DU TRIGGER

- ❑ **Le corps du trigger est un bloc PL/SQL**
 - ❑ Il peut contenir du SQL et du PL/SQL.
 - ❑ Il est exécuté si l'instruction de déclenchement se produit et si la clause de restriction WHEN, le cas échéant, est évaluée à vrai.

NOMS DE CORRÉLATION

- ❑ La nouvelle valeur est appelée :new.colonne
- ❑ L'ancienne valeur est appelée :old.colonne
- ❑ Exemple : IF :new.salaire < :old.salaire ...
- ❑ Si un trigger ligne BEFORE modifie la nouvelle valeur d'une colonne, un éventuel trigger ligne AFTER déclenché par la même instruction voit le changement effectué par le trigger BEFORE.

L'OPTION REFERENCING

❑ Si une table s'appelle NEW ou OLD, on peut utiliser REFERENCING pour éviter l'ambiguïté entre le nom de la table et le nom de corrélation.

❑ Exemple

```
CREATE TRIGGER nomtrigger BEFORE UPDATE ON  
new REFERENCING new AS newnew  
  
FOR EACH ROW BEGIN  
  
:newnew.colon1:= TO_CHAR(:newnew.colon2);  
  
END;
```

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.

Exemple

```
CREATE TRIGGER ... BEFORE INSERT OR UPDATE ON employe .....
```

```
BEGIN .....
```

```
IF INSERTING THEN ..... END IF;
```

```
IF UPDATING THEN ..... END IF;
```

```
END;
```

OPTION UPDATING

❑UPDATING peut être suivi d'un nom de colonne

```
CREATE TRIGGER ... BEFORE UPDATE OF salaire, commission ON  
employe .....  
  
BEGIN .....  
  
IF UPDATING ('salaire') THEN .....  
  
END IF;  
  
..... END;
```

ORDRE DE TRAITEMENT DES LIGNES

- ❑ On ne peut pas gérer l'ordre des lignes traitées par une instruction SQL.
- ❑ On ne peut donc pas créer un trigger qui dépend de l'ordre dans lequel les lignes sont traitées.
- ❑ Triggers en cascade
 - ❑ Un trigger peut provoquer le déclenchement d'un autre trigger.
 - ❑ ORACLE autorise jusqu'à 32 triggers en cascade à un moment donné.

CONDITIONS NÉCESSAIRES POUR CRÉER UN TRIGGER

- ❑ Il faut avoir le privilège CREATE TRIGGER
- ❑ Il faut soit posséder la table sur laquelle on veut définir un trigger, soit posséder le privilège ALTER sur la table sur laquelle on veut définir le trigger, soit posséder le privilège ALTER ANY TABLE
- ❑ Modification de triggers
 - ❑ Pour modifier un trigger, on refait une instruction CREATE TRIGGER suivie de OR REPLACE ou bien on supprime le trigger (DROP TRIGGER nomtrigger) et on le crée à nouveau.

ACTIVATION D'UN TRIGGER

- ❑ **Un trigger peut être activé ou désactivé.**
 - ❑ S'il est désactivé, ORACLE 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.
- ❑ **Par défaut, un trigger est activé dès sa création.**
- ❑ **Pour désactiver un trigger, on utilise l'instruction**
 - ❑ ALTER TRIGGER avec l'option DISABLE : **ALTER TRIGGER nomtrigger DISABLE;**
- ❑ **On peut désactiver tous les triggers associés à une table avec la commande :**
 - ❑ **ALTER TABLE nomtable DISABLE ALL TRIGGERS;**
- ❑ **A l'inverse on peut réactiver un trigger :**
 - ❑ **ALTER TRIGGER nomtrigger ENABLE;** ou tous les triggers associés à une table :
 - ❑ **ALTER TABLE nomtable ENABLE ALL TRIGGERS;**

RECHERCHE D'INFORMATION SUR LES TRIGGERS

❑ Les définitions des triggers sont stockées dans les tables de la métabase, notamment dans les tables

❑ USER_TRIGGERS

❑ ALL_TRIGGERS

❑ DBA_TRIGGERS

GESTION D'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.

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

QUELQUES EXEMPLES

```
employe(numserv,....) service(numserv,...)
```

```
/* vérifier que le service de l'employé existe bien */
```

```
CREATE TRIGGER verif_service BEFORE INSERT OR UPDATE OF numserv ON employe
```

```
FOR EACH ROW
```

```
WHEN (new.numserv is not null)
```

```
DECLARE noserv integer;
```

```
    BEGIN noserv:=0;
```

```
        SELECT numserv INTO noserv
```

```
        FROM SERVICE WHERE numserv=:new.numserv;
```

```
        IF (noserv=0)
```

```
        THEN raise_application_error(-20501, 'N° de service non correct');
```

```
        END IF;
```

```
    END;
```

EXAMPLES

Employe (salaire,....)

/* mettre une valeur par défaut si le champ ne contient rien */

/* affecter 5000 au salaire d'un employé qui n'en a pas */

CREATE TRIGGER smic BEFORE INSERT OR UPDATE OF salaire ON employe

FOR EACH ROW

WHEN (new.salaire is null)

BEGIN

SELECT 5000 INTO :new.salaire

FROM employe;

END;

EXEMPLES

`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`

`BEFORE 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`

`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;`