

Bases de données actives

Déclencheurs



Thierry DELOT, Marie THILLIEZ

UPHF

Institut des Sciences et Techniques de Valenciennes

`Prenom.nom@uphf.fr`

Introduction

- Maintien de la cohérence des données = 1 des objectifs majeurs des SGBD
- Comment préserver cette cohérence des données stockées dans la base ?

Définitions (rappels)

■ Contrainte d'intégrité :

- Assertion qui doit être vérifiée par des données à des instants donnés

■ Base de données cohérente :

- Base de données dont l'ensemble des contraintes d'intégrité (implicites ou explicites) est respecté par les données de la base

Types de contraintes

- Définition de types permet d'exprimer des CI (portant sur une seule donnée)
 - Contraintes sur le domaine de variation
Ex : la donnée est représentée par un nombre entier
 - Contraintes sur la plage de valeurs
Ex: la donnée qté doit être comprise entre 1 et 1000
- Contraintes peuvent également porter sur plusieurs données
 - Contraintes sur les clés étrangères

Vérification de contraintes

■ CHECK

```
CREATE DOMAIN COULEURSVIN CHAR(10)  
CONSTRAINT ICCV  
CHECK (VALUE IN ( 'Rouge' , 'Rose' , 'Blanc' )
```

```
CREATE DOMAIN VILLES CHAR(35)  
CONSTRAINT ICVILLES  
CHECK (VALUE IS NOT NULL)
```

```
CONSTRAINT QTEPOSITIVE CHECK (Abus.Qte > 0)
```

Principe

■ Base de Données Active

- réagit aux changements d ' état de la base de données

■ Déclencheur = Événement-Condition-Action

- Événement dans la base
- Condition
- Déclenchement d ' une action

■ Trigger SQL

- Événement
= INSERT, DELETE, UPDATE dans une relation
- Action = un ou plusieurs ordres SQL, SQL procédural

Motivations

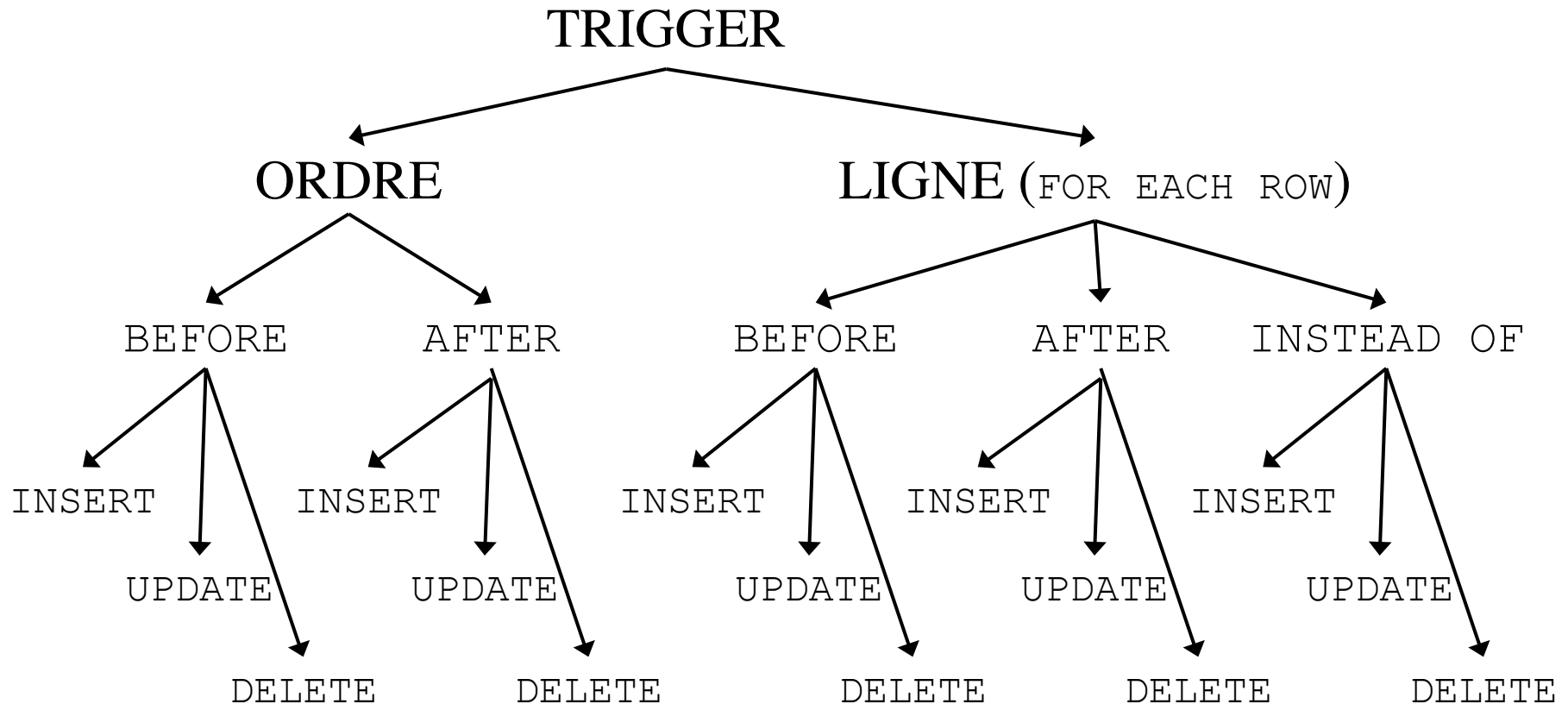
■ Pourquoi faire ?

- valider les données entrées
- créer un audit de la base de données
- dériver des données additionnelles
- maintenir des règles d 'intégrité complexes
- implanter des règles de métier
- supporter des alertes (envoi de e-mails par exemple)

■ Gains

- développement plus rapide
 - les triggers sont stockées dans la base
- maintien global des règles d 'intégrité

Différents triggers SQL



Remarques

- Différence entre produits et standards
- Différent des contraintes et des assertions SQL2
 - l'événement est programmable
 - Pas de condition dans les CHECK.

Trigger Ordre

```
CREATE TRIGGER nom_du_trigger  
    type_interposition    type_ordre  
    ON nom_de_la_table
```

```
action
```

- l' action est exécutée une seule fois
 avant (type_interposition=BEFORE) ou après (=AFTER)
 l' exécution d' un ordre
 sur la relation nom_de_la_table
- le type de l' ordre type_ordre peut être
 INSERT, UPDATE, DELETE,
 type_ordre OR type_ordre,
 UPDATE OF liste_de_colonnes

Exemple de Trigger Ordre

Vente(gencod, qte, prix) VolumeAffaire(total,date)

```
CREATE TRIGGER tg_modifVolume AFTER INSERT ON Vente
```

```
  DECLARE s number;
```

```
  BEGIN
```

```
    select sum(prix*qte) into s from Vente;
```

```
    insert into VolumeAffaire values (s,current);
```

```
  END;
```

```
CREATE TRIGGER tg_modifInterdit
```

```
  AFTER UPDATE OF prix, qte ON Vente
```

```
  BEGIN raise_application_error(-9998, ' Modification interdite ');
```

```
  END;
```

Trigger Ligne (FOR EACH ROW)

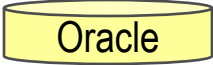
```
CREATE TRIGGER nom_du_trigger  
    type_interposition    type_ordre  
    ON nom_de_la_table  
FOR EACH ROW  
    action
```

- l' action est exécutée
 avant (type_interposition=BEFORE) ou après (=AFTER)
 l' opération réalisée sur chaque ligne
 de la relation nom_de_la_table

Variables de Transition des Triggers Ligne

- l' action du trigger ligne peut utiliser deux variables de transition (old et new) contenant les valeurs de la ligne avant et après l' événement

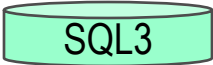
- Variables implicites dans Oracle



Oracle

:old et :new

- Déclaration explicite dans Informix et SQL3



SQL3

REFERENCING OLD AS variable_avant **NEW AS** variable_apres

■ Remarque

- la variable avant n'a pas de sens dans un INSERT
- la variable après n'a pas de sens dans un DELETE

Exemple de Trigger Ligne

Vente(gencod, qte, prix) Stock(gencod, qte)

```
CREATE TRIGGER tg_nouvVente AFTER INSERT ON Vente  
FOR EACH ROW
```

```
BEGIN
```

```
    if :new.qte > (select qte from Stock where gencod = :new.gencod)
```

```
    then    raise_application_error(-9997, ' Stock insuffisant ');
```

```
    else    update Stock set qte := Stock.qte - :new.qte
```

```
            where gencod = :new.gencod;
```

```
END;
```

Exemple de Trigger Ligne

Commande(gencod, qte, prix)

```
CREATE TRIGGER tg_nouvCmd AFTER UPDATE ON Commande  
FOR EACH ROW
```

```
BEGIN
```

```
    if :new.qte < :old.qte
```

```
    then      raise_application_error(-9996,
```

```
                ' Impossible de diminuer la commande ' );
```

```
    else      ...
```

```
END;
```

Condition de déclenchement WHEN

```
CREATE TRIGGER nom_du_trigger  
    type_interposition    type_ordre  
    ON nom_de_la_table  
FOR EACH ROW  
    WHEN (attribut condition_SQL valeur)  
  
    action
```

- l 'attribut est une colonne de la ligne avant (:old.colonne) ou après (:new.colonne) l 'événement
- la condition est une condition SQL
=,!=, >, <, IN, BETWEEN
- la valeur ne peut être le résultat d 'un ordre SELECT

Exemple de Condition sur le déclenchement d ' un trigger

Oracle

Commande(gencod, typeprod, qte, prix)

```
CREATE TRIGGER tg_nouvCmd AFTER UPDATE ON Commande  
FOR EACH ROW
```

```
WHEN (new.qte < old.qte AND new.typeprod IN ( 'Viande' ,  
        'Poisson' ))
```

```
BEGIN
```

```
    raise_application_error(-9996,
```

```
        ' Impossible de diminuer la commande pour ce type de produit' );
```

```
END;
```

Limite de la clause WHEN

```
Commande(gencod, qte, prix)      Produit(gencod, descr, typeprod)
CREATE TRIGGER tg_nouvCmd AFTER UPDATE ON Commande
FOR EACH ROW
WHEN (:new.gencod IN
      (select gencod from Produit where typeprod in
      ( 'Viande ', 'Poisson' )
)
BEGIN
    raise_application_error(-9995,
        ' Impossible de diminuer la commande pour ce type de produit' );
END;
-- NE FONCTIONNE PAS
```

Traitements différenciés

- les prédicats INSERTING, UPDATING, DELETING sont utilisables en PL/SQL pour différencier le type de l'ordre qui a déclenché le trigger.

```
CREATE TRIGGER tg_modifCmd
AFTER UPDATE OR DELETE ON Commande FOR EACH ROW
BEGIN
    if DELETING
    then raise_application_error(-9995,
                                ' Impossible d ' ' annuler la
commande' );
    else if UPDATING and :new.qte < :old.qte
    then raise_application_error(-9996,
                                ' Impossible de diminuer la commande' );
```

Limitations

- Pas d ' ordre (statement) dans l ' action
 - de transaction COMMIT, ROLLBACK, SAVEPOINT
 - de connection ou de session
 - mais l ' action peut lever une exception
 - raise_application_error
- Attention aux tables en mutation

Action sur une table en mutation

Produit(gencod, descr, typeprod)

```
CREATE TRIGGER tg_prod
```

```
BEFORE INSERT OR UPDATE OR DELETE ON Produit
```

```
FOR EACH ROW DECLARE n integer
```

```
BEGIN
```

```
    select gencod into n from Produit where typeprod= ' Viande ' ; ...
```

```
END;
```

- Suppression

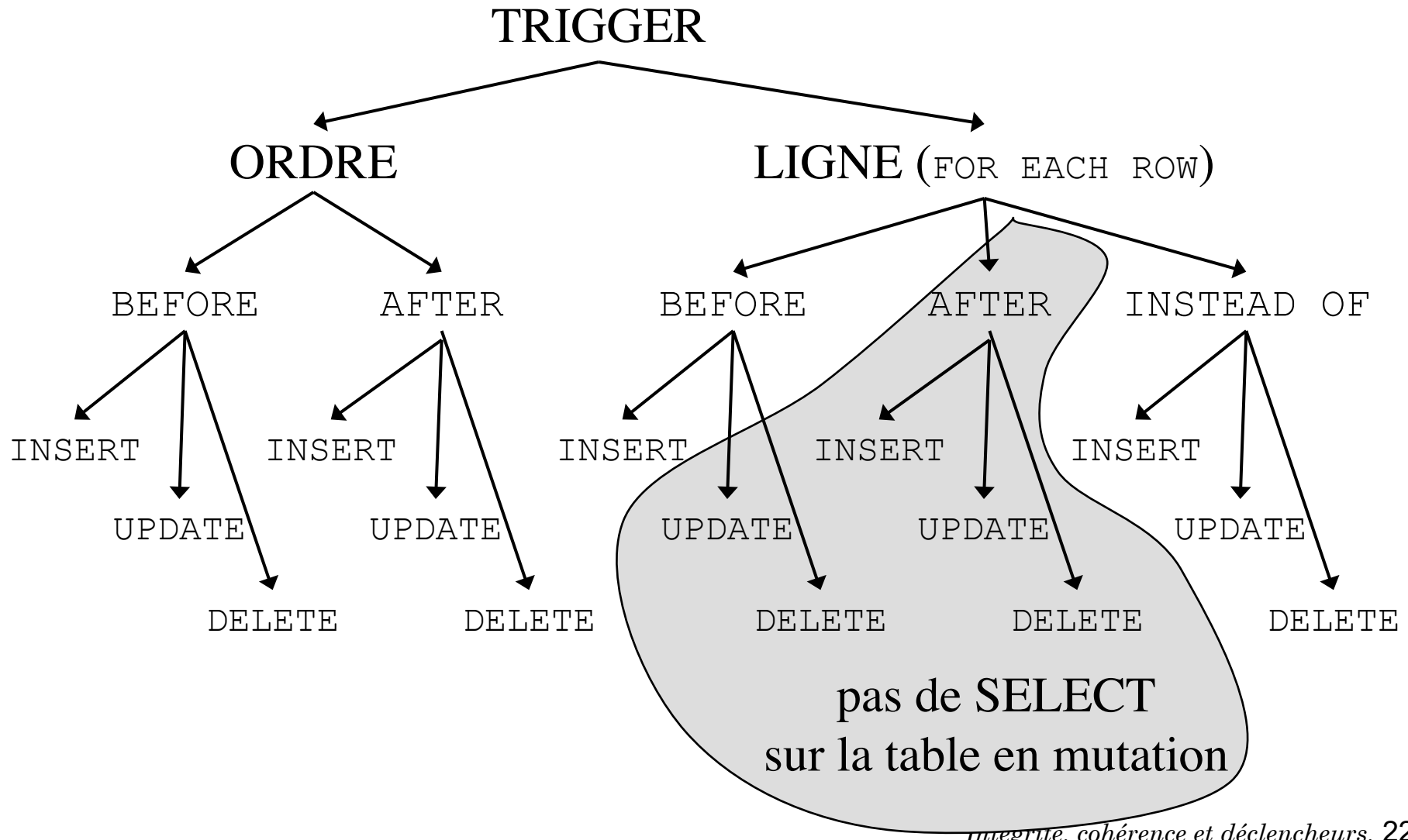
```
DELETE FROM Produit;  ERROR ... table DEMO.Produit is mutating
```

- Insertion

```
INSERT INTO Produit VALUES(8736, 'Lentilles' , 'Epicerie' );  OK
```

Action sur une table en mutation

SELECT autorisé



Trigger **INSTEAD OF**

```
CREATE TRIGGER nom_du_trigger  
    INSTEAD OF type_ordre  
    ON nom_de_la_table  
FOR EACH ROW  
    action
```

- La modification sur la **table** (ou sur la **vue**) est remplacée par l' action
- les variables :old et :new sont utilisées dans l' action comme si l' événement avait lieu

■ Usage

- permet les modifications sur une vue
 - par exemple une vue Objet Relationnelle d' une base Relationnelle

Exemple de Trigger INSTEAD OF (i)

Oracle

■ Base relationnelle

```
create table Pers( numss number, nom varchar2(10), prenom varchar2(10));  
create table Tel(numss number, numtel varchar2(10));
```

■ Vue Objet

```
create type listtel_t AS as varray(10) of varchar2(10);  
create type ovPers_t as object ( numss number, nom varchar2(10), prenom  
    varchar2(10), listtel listtel_t);  
create view ovPers of ovPers_t with objectoid(numss) as  
    select    numss, nom, prenom, cast( multiset(  
        select numtel from Tel t where t.numss=p.numss) as listtel_t)  
    from Pers p;
```


Exemple de Trigger INSTEAD OF (ii)

Oracle

■ Trigger INSTEAD OF

```
create trigger tg_ins_ovPers instead of insert on ovPers
for each row declare i: integer;
begin
    insert into Pers values(:new.numss, :new.nom, :new.prenom);
    if :new.listtel is not null and :new.listtel.count > 0 then
        for i in :new.listtel.first ... :new.listtel.last loop
            insert into Tel values (:new.numss, :new.listtel(i));
        end loop;
    end if; end;

> insert into ovPers values ( 1390120989, ' Dupont ', ' Jean ',
                             listtel_t( ' 0327141234 ', ' 0320445962 ' ) );
```

Gestion des Triggers

- **CREATE TRIGGER nom_trigger ...**
 - le trigger nom_trigger est créé et activé.
- **CREATE OR REPLACE TRIGGER nom_trigger ...**
 - le trigger nom_trigger est modifié.
- **DROP TRIGGER nom_trigger**
 - le trigger nom_trigger est supprimé de la base.
- **ALTER TRIGGER nom_trigger DISABLE**
 - le trigger nom_trigger est désactivé.
- **ALTER TRIGGER nom_trigger ENABLE**
 - le trigger nom_trigger est réactivé.

```
CREATE TRIGGER incendie_entrepot AFTER INSERT ON Vente  
  BEGIN raise_application_error(-9999, « Vente impossible »); END;  
ALTER TRIGGER incendie_entrepot DISABLE;
```

Interactions

■ Cascade de triggers

- l' action d' un trigger peut déclencher d' autres triggers

■ Interactions avec les contraintes

- l' action d' un trigger peut causer la vérification des contraintes
- les actions des contraintes référentielles peuvent déclencher des triggers
 - les triggers DELETE sont déclenchés par
ON DELETE CASCADE
 - les triggers UPDATE sont déclenchés par
ON DELETE SET NULL|DEFAULT, ON UPDATE CASCADE,
ON UPDATE SET NULL|DEFAULT

Ordonnancement de triggers multiples

■ Plusieurs triggers peuvent être définis pour le même événement

- La date de création d ' un trigger est conservée dans la base
- Les triggers sont activés dans l ' ordre ascendant de leur date de création

■ Remarque pour Oracle

- Un seul trigger sur la même table avec les mêmes événements de déclenchement.
- Oracle est un produit, SQL3 du papier

Conclusion

- SGBD Actif
- Contrôle dynamique et évolutif des manipulations dans la base
- Duplication contrôle d ' information
- Etendre les mécanismes de contrôle d ' intégrité
 - palier aux limites des contraintes

Bibliographie

- Nelson Mattos, "An Overview of the SQL3 Standard", presentation foils, Database Technology Institute, IBM Santa Teresa Lab., San Jose, CA, July 1996, ftp://jerry.ece.umassd.edu/isowg3/dbl/BASEdocs/descriptions/SQL3_foils.ps
- Scott Urman , « Oracle8 PL/SQL Programming », ed Osborne-McGraw-Hill, Oracle Press Series, ISBN 0-07-882305-6.
- " Using Oracle8™ Object Views: An Example, An Oracle Technical White Paper, October 1997,
<http://www.oracle.com/st/o8collateral/html/objview2.html>
- Steven Feuerstein, « Oracle PL/SQL Programming », 2nd Edition, ed O'Reilly. ISBN 1-56592-335-9.

Exercices

Oracle

■ Soit la base

Vente(gencod, qte, prix) VolumeAffaire(total,date)

■ Donner un trigger ligne qui modifie le Volume d ' Affaire en cas d ' insertion, de modification et du suppression dans Vente.

- Version 1 : VolumeAffaire ne contient qu ' une ligne qui sera modifiée
- Version 2 : VolumeAffaire contient une ligne par mise à jour dans Vente
- Version 3 : VolumeAffaire contient une ligne par journée de Vente