مخبر هندسة الأنظمة المعقدة
LABORATOIRE D'INGÉNIERIE DES SYSTÈMES COMPLEXES

جامعة باجي مختار-عنابة
Université Badji Mokhtar-Annaba

Administration des bases de données avancées

Master I: Ingénierie des Logiciels Complexes (ILC)

Dr Kamilia MENGHOUR
Laboratoire d'Ingénierie des Systèmes Complexes
Université Badji Mokhtar-Annaba
K_menghour@yahoo.fr

Année Universitaire : 2024- 2025

Chapitre 2

Les modèles avancés de bases de données

Plan du cours

Chapitre 2 : *Les modèles avancés de bases de données (20%)*

- Les bases de données orientées objet.
- Bases de données actives.
- Bases de données déductives.
- Bases de données géographiques.
- Les entrepôts de données.

Les bases de données passives

- Toutes les actions sur les données sont des invocations explicites de programme d'application. Le SGBD fait ce que le programme lui dit.
- **Exemples de problèmes:**
- ***Contrôle d'inventaire*** : commander des produits lorsque le stock est sous un seuil prédéfini. Ce comportement peut être implémenté avec une base de données passive par deux méthodes:
  1. Ajouter une vérification d'une condition dans chaque programme modifiant l'inventaire. —> Une mauvaise solution de point de vue de Génie Logiciel.
  2. Le programme d'application doit vérifier l'inventaire périodiquement.
     - Fréquence de vérification très élevée —> manque d'efficacité
     - Fréquence de vérification faible - —> manque de fiabilité du pgm.

D'où l'idée d'ajouter des fonctionnalités au SGBD pour vérifier la situation précédente.

Les bases de données actives

- La notion de **SGBD actif** s'oppose à celle de **SGBD passif**, qui subit sans réagir des opérations de modification et interrogation de données..
- **SGBD actif :** 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.
- La notion de ***base de données active*** va permettre de déplacer une partie de la sémantique des applications au sein du SGBD.
- Comment réagit-il ?
  - Il pourra déclencher une opération subséquente à un événement donné (par exemple, une mise à jour),
  - interdire une opération en annulant la transaction qui l'a demandée,
  - envoyer un message à l'application, voire sur Internet.

Les bases de données actives: Objectifs

- 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 métier
- Supporter des alertes (envoi de e-mails par exemple)

Les bases de données actives: Avantages

- Simplification du code des applications : une partie de programme peut être programmée avec les règles actives de SGBD.
- Sécurité renforcée par l'automatisation : les actions sont déclenchées automatiquement et sans intervention de l'utilisateur.
- Les déclencheurs sont stockées dans la base.
- Cohérence globale des déclencheurs.
- Augmenter la fiabilité de données : plus d'action de vérification et de réparation, mieux aider à la décision.
- Les bases de données actives permettent de créer des systèmes d'information: plus performants, plus rapides, et à meilleur coût que dans le passé.

Modèle de règle ECA

- La notion de réagir à des situations dans les systèmes actifs est supportée par des règles actives de la forme: **Évènement-Condition-Action.**
- La sémantique d'une règle ECA est;

Lorsqu'un événement E se produit

Si la condition C est satisfaite

Alors exécuter l'action A

1. Les ***événement***s qui déclenchent la règle.
2. La ***condition*** qui détermine l'action définie dans la règle qui doit être exécutée. Si aucune condition n'est spécifiée, l'action est exécutée dés que l'événement se produit. Si une condition est spécifiée, elle est d'abord évaluée et si le résultat du test est vrai, l'action est exécutée.
3. L'***action*** à exécutée est généralement constituée d'une séquence d'instruction SQL, mais il peut également s'agir d'une transaction ou de l'appel d'un programme externe qui sera lancé automatiquement

Modèle de règle ECA

Les règles peuvent exprimer des diverses aspects liés aux sémantiques d'application :

1. ***Des contraintes statiques*** : ex, les cardinalités, les contraintes d'intégrités référentielles, restriction de valeurs…. ;

   Un encadreur ne peut pas encadrer plus de quatre groupes.

   Le salaire d'un employé ne peut pas dépasser le salaire de son manager.

2. ***Des contraintes sur la gestion workflow***.

   Si la commande est acceptée, une facture proformat est rédigée.

3. ***Des données historiques***

   les données mensuelles sur les bons de commandes sont transférées à l'entrepôt des données

Modèle de règle ECA

4. ***Implémentation des relations génériques***

   Un lecteur est soit un enseignant ou un étudiant mais pas les deux en même temps.

5. ***Données dérivées***

   Le nombre des étudiants enregistrés dans un cours doit être parmi les attributs du cours.

6. ***Contrôle d'accès***

   Un employé ne peut consulter que les données de son département.

Déclencheurs « Triggers »

- **Trigger (déclencheur)** :Routine déclenchée automatiquement par des événements liés à des actions sur la base. Les triggers complètent les contraintes d'intégrité en permettant des contrôles et des traitements plus complexes:
  - Génération automatique de valeurs manquantes (ex. Valeur dérivée, par défaut)
  - Éviter des modifications invalides (C: test, A: abort)
  - Implantation de règles applicatives (« business rules »)
  - Génération de traces d'exécution, statistiques, …
  - Maintenance de répliques
  - Propagation de mises-à-jour sur des vues vers les tables
  - Intégrité référentielle entre des données distribuées
  - Interception d'événements utilisateur / système (LOGIN, STARTUP, …)

Triggers: Syntaxe

Syntaxe de Trigger d'Oracle:

{ CREATE | REPLACE } TRIGGER <nom_trigger>
// Événement
{BEFORE | AFTER}
{INSERT | DELETE | UPDATE [OF] < attribut, …>}
ON <relation>
[REFERENCING [NEW AS <new-value-tuple-name>]
[OLD AS < old-value-tuple-name >]]
[FOR EACH ROW] // ROW TRIGGER
// Condition
[WHEN (<condition SQL>)]
// Action
DECLARE
BIGEN
<PL/SQL bloc> //Procedure SQL
END

Triggers: Syntaxe

- Before | after *permet d'indiquer quand le trigger va être exécuté*
- Insert | delete | update [of <attribut>] *indique l'événement déclencheur*
- On <relation> *indique le nom de la table qui doit être surveillée*
- [referencing old as <var> , new as <var>] *en SQL3, oracle utilise :new et :old*
- For each row *précise si l'action est exécuté 1 fois par tuple touché ou pour toute la table .*
- [when <condition>] *permet d'indiquer une condition pour l'exécution du trigger*
- DECLARE *déclaration de variables pour le bloc PL/SQL*
- BEGIN *bloc PL/SQL contenant le code de l'action à exécuter*
- ***<PL/SQL bloc>*** *dans SLQ3, on peut indiquer une suite de commande SQL*

Triggers: Exemple 1

Emp (**Eno**, Ename, Title, City)

- Création automatique d'une valeur de clé (autoincrément) :

```
CREATE TRIGGER SetEmpKey
BEFORE INSERT ON Emp
REFERENCING NEW AS N
FOR EACH ROW
BEGIN
    N.Eno := SELECT COUNT(*)
    FROM Emp
END;
/* le premier Eno sera 0 */
```

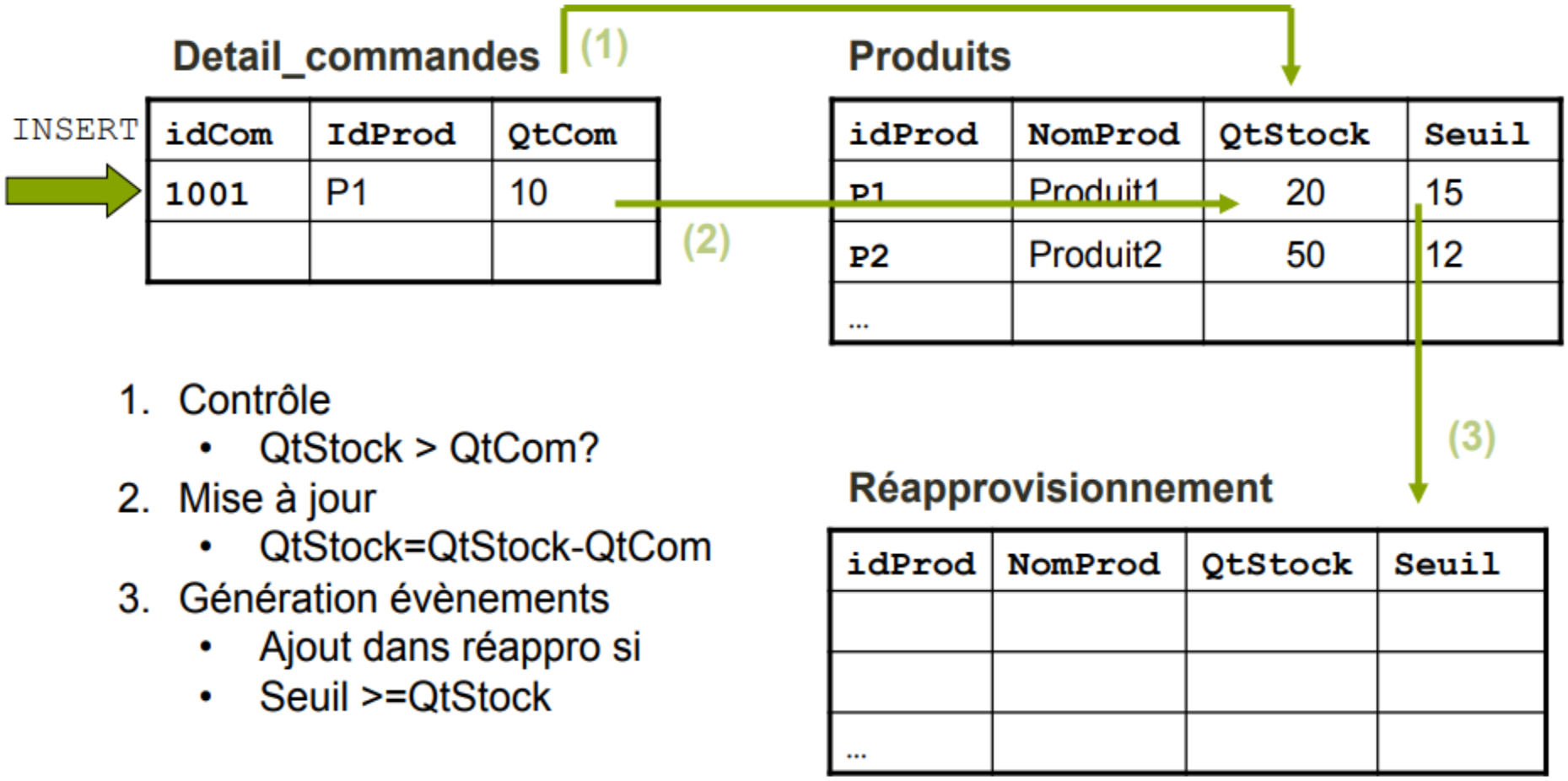
Triggers: Exemple 2

Pay(**Title**, Salary, Raise)

- Maintenance des augmentations (raise) de salaire:

```
CREATE TRIGGER UpdateRaise
AFTER UPDATE OF Salary
ON Pay
REFERENCING OLD AS O, NEW AS N
FOR EACH ROW
BEGIN
    UPDATE Pay SET Raise = N.Salary - O.Salary
    WHERE Title = N.Title;
END
```

Triggers: Exemple 3



1. Contrôle
   - QtStock > QtCom?
2. Mise à jour
   - QtStock=QtStock-QtCom
3. Génération évènements
   - Ajout dans réappro si
   - Seuil >=QtStock

Triggers: Exemple 3

```
CREATE TRIGGER t_b_i_detail_commandes
BEFORE INSERT ON detail_commandes
FOR EACH ROW
DECLARE
vqtstock NUMBER;
BEGIN
SELECT qtstock INTO vqtstock FROM produits
WHERE idprod = :NEW.idprod;
IF vqtstock < :NEW.qtcom THEN
RAISE_APPLICATION_ERROR(-20001, 'stock insuffisant');
END IF;
END;
/
```

RAISE_APPLICATION_ERROR *est une instruction Oracle qui permet de déclencher une erreur en lui associant un message et un numéro (compris entre-20000 et -20999).*

Triggers: Exemple 3

```
CREATE TRIGGER t_a_i_detail_commandes
AFTER INSERT ON detail_commandes
FOR EACH ROW
BEGIN
UPDATE Produits p
SET p.qtstock = p.qtstock - :NEW.qtcom
WHERE idprod= :NEW.idprod;
END;
/
```

Triggers: Exemple 3

```
CREATE TRIGGER t_a_u_produits
AFTER UPDATE OF qtstock ON produits
FOR EACH ROW
BEGIN
IF :NEW.qtstock <= : NEW.seuil THEN
INSERT INTO reapprovisionnement VALUES
(:NEW.idprod, :NEW.nomprod, :NEW.qtstock, :NEW.seuil);
END IF;

END;
/
```

Restriction de Triggers

- Il est possible d'ajouter une clause WHEN pour restreindre les cas où le trigger est exécuté.
- WHEN est suivi de la condition nécessaire à l'exécution du trigger.
- Cette condition peut référencer la nouvelle et l'ancienne valeur d'une colonne de la table (new et old ne doivent pas être préfixés par _ :_ comme à l'intérieur du code du trigger).

Exemple :

```
CREATE OR REPLACE TRIGGER modif_salaire_trigger
BEFORE UPDATE OF sal ON emp
FOR EACH ROW
WHEN (new.sal < old.sal)
BEGIN
raise_application_error(-20001, 'Interdit de baisser le salaire !
  (' || :old.name || ')');
END;
```

Problèmes de conception de BDD Active

- L'une des difficultés À la généralisation des règles actives , réside dans le fait qu'il s'agit de techniques relativement difficiles À utiliser en termes de conception.
- Par exemple il n'est pas évident de vérifier la cohérence d'un ensemble de règles , autrement dit de s'assurer que deux ou plusieurs règles ne sont pas conflictuelles . Il n'est pas simple de garantir qu'un ensemble de règles se terminera dans toutes les circonstances.

Exemple :

```
R1 : create trigger T1
After insert on table1
For each row
Update table2 Set attribut1=..;
R2 : create trigger T2
After update attribut1
on table T2
For each Row
Insert into table1 values(.....);
```

Opérations sur les triggers

➢ 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 nom trigger) et on le crée à nouveau.

➢ Activation de triggers

- Un trigger peut être activé ou désactivé.
- S'il est désactivé, ORACLE le stocke mais l'ignore.
- Par défaut, un trigger est activé dès sa création.

Opérations sur les triggers

➢ Désactivation de triggers

✓ 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
```

Merci de votre attention

Des Questions

K_menghour@yahoo.fr