Bases de données actives

Vu Tuyet Trinh
trinhvt@soict.hut.edu.vn

Departement de Systemes d'Information
Ecole de Technologies d'Information et de Communication
Institut Polytechnique de Hanoi

Plan

- Contraintes d 'intégrité
- SGBD actif
- Les règles ECA
- Règles en SQL3
- Les mécanismes d'exécution
- Principaux SGBD actifs

Contraintes d ’intégrité

- Garder les bases cohérentes
- Vérifier les données lors des chargements
- Vérifier les données lors des mises à jour
- Répercuter certaines mises à jour entre tables
- Gérer les références inter-tables

- C'est essentiel :
  - Une BD non cohérente est peu utile !

3

Contraintes structurelles

- Contrainte structurelle (Structural constraint)
  - Contrainte d'intégrité spécifique à un modèle exprimant une propriété fondamentale d'une structure de données du modèle.
- Les contraintes structurelles sont généralement statiques.
- Pour le modèle relationnel, elles permettent d'exprimer explicitement certaines propriétés des relations et des domaines des attributs.

4

Principales contraintes structurelles

- Unicité de clé
- Contrainte référentielle.
  - Elle spécifie que toute valeur d'un groupe de colonnes d'une table doit figurer comme valeur de clé dans une autre table.
- Contrainte de domaine.
  - Ce type de contrainte permet de restreindre la plage de valeurs d'un domaine.
- Contrainte de non nullité.
  - Une telle contrainte spécifie que la valeur d'un attribut ne peut être nulle.

5

Contrainte non structurelle

- Contrainte de comportement (Behavioral constraint)
  - Contrainte d'intégrité exprimant une règle d'évolution que doivent vérifier les données lors des mises à jour.
- Les dépendances généralisées :
  - Les dépendances fonctionnelles. On dit que X -> Y (X détermine Y) si pour toute valeur de X il existe une valeur unique de Y associée.
    - VINS (NV, CRU, REGION, PAYS)
  - Les dépendances multivaluées. On dit que X->> Y (X multidétermine Y) dans une relation R si pour toute valeur de X il existe un ensemble de valeur de Y, et ceci indépendamment des valeurs des autres attributs Z de la relation R.
    - BUVEURS (NOM, CRU, SPORT) NOM ->> CRU | SPORT

6

Autres dépendances

- Les dépendances d'inclusion.
  - Généralise les contraintes référentielles.
- Les contraintes temporelles.
  - Elles permettent de comparer l'ancienne valeur d'un attribut à la nouvelle après mise à jour.
- Les contraintes équationnelles.
  - Il s'agit là de comparer deux expressions arithmétiques calculées à partir de données de la base et de forcer l'égalité ou une inégalité.
  - Exemple : gestion de stocks.

7

Syntaxe en SQL1

- CREATE TABLE <nom de table>
  ({<Attribut> <Domaine> [<CONTRAINTE D'ATTRIBUT>]}+)
  [<CONTRAINTE DE RELATION>]
- <CONTRAINTE D'ATTRIBUT> ::=
  NOT NULL |
  UNIQUE | PRIMARY KEY
  REFERENCES <Relation> (<Attribut>) |
  CHECK <Condition>
- <CONTRAINTE DE RELATION> ::=
  UNIQUE (<Attribut>+) | PRIMARY KEY (<Attribut>+) |
  FOREIGN KEY (<Attribut>+)
  REFERENCES <Relation> (<Attribut>+) |
  CHECK <Condition>

8

Exemple

```
CREATE TABLE ABUS (
   NB INT NOT NULL,
   NV INT NOT NULL REFERENCES VINS(NV),
   DATE DEC(6) CHECK BETWEEN 010180 AND 311299,
   QUANTITE SMALLINT DEFAULT 1,
   PRIMARY KEY(NB, NV, DATE),
   FOREIGN KEY NB REFERENCES BUVEURS,
   CHECK (QUANTITE BETWEEN 1 AND 100)
)
```


Contrainte référentielle en SQL2

FOREIGN KEY (<Attribut>+)

REFERENCES <Relation> (<Attribut>+)

[ON DELETE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]

[ON UPDATE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]

Contrainte de comportement

CREATE ASSERTION <nom de contrainte>
[{ BEFORE COMMIT |
AFTER {INSERT|DELETE|UPDATE[OF(Attribut+)]} ON <Relation>
}...]
CHECK <Condition>
[FOR [EACH ROW OF] <Relation>]

Quelques exemples

CREATE ASSERTION MINDEGRÉ
BEFORE COMMIT
CHECK (SELECT MIN(DEGRÉ) FROM VINS > 10)
FOR VINS ;

CREATE ASSERTION SOMMEQUANTITÉBUE
BEFORE COMMIT
CHECK (SELECT SUM(QUANTITE)
FROM ABUS GROUP BY NB) < 100
FOR ABUS.

Analyse des contraintes

- Les contraintes sont-elles cohérentes entre-elles ?
- Les contraintes ne sont-elles pas redondantes ?
  - ce sont là des problèmes de logique (résolution !)
- Quelles contraintes doit-on vérifier suite à une insertion, une suppression, une mise à jour d'une table ?
  - détermination d ’étiquettes de vérification
- Est-il possible de simplifier les contraintes de sorte à faciliter le contrôle ?

Quand vérifier une contrainte ?

- Toute contrainte affirmant l'existence d'un tuple dans une relation R doit être étiquetée (R, DELETE).
- Toute contrainte vraie pour tous les tuples d'une relation R doit être étiquetée (R,INSERT).
- Toute contrainte étiquetée (R,DELETE) ou (R,INSERT) doit être étiquetée (R,MODIFY).

Utilisation relations différentielles

- La transaction t fait passer R de l'état R à l'état Rt comme suit :

  $Rt := (R - R^-) \cup R^+$.

- Test différentiel (Differential test)
  - Contrainte d'intégrité simplifiée à vérifier à priori après une opération sur R afin de garantir la satisfaction de la contrainte après application de l'opération.

15

Quelques tests différentiels

Type de contrainte	Insertion	Suppression	Mise à jour
Clé primaire K de R	Les clés de R^+ sont uniquent et ne figurent pas dans R.	Pas de vérification	Les clés de R^+ sont uniquent et ne figurent pas dans $R-R^-$.
Clé étrangère A de R Ref K de S	Les tuples de R^+ référence un tuple de S.	R : Pas de vérification S : Les clés K de S^- ne figurent pas dans A de R	Les tuples de R^+ référence un tuple de S.
Domaine A de R	Domaine A sur R^+	Pas de vérification	Domaine A sur R^+
Non nullité	Non nullité sur R^+	Pas de vérification	Non nullité sur R^+
Dépendance fonctionnelle A->B	A de R^+ = A de R implique B de R^+ = B de R	Pas de vérification	Pas de forme simplifiée
Contrainte temporelle sur attribut	Pas de vérification	Pas de vérification	Vérifier les tuples de R^+ par rapport à ceux de R^-

16

SGBD ACTIFS

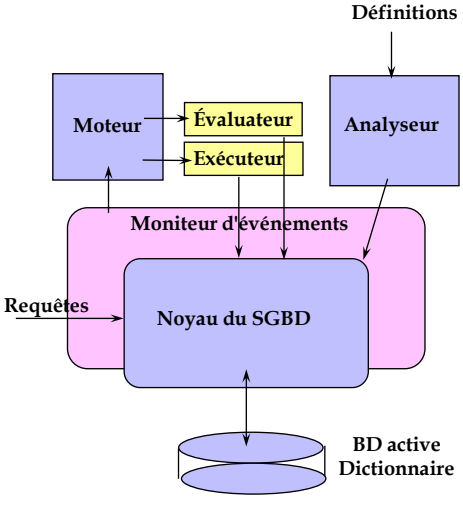
- SGBD capable de réagir afin de contrôler l'intégrité, gérer les redondances, autoriser, interdire, alerter, déclencher.
- Quand ?
  - lors d'opérations illicites
  - lors d'opérations licites
  - à un instant donné
  - sous certaines conditions
- Comment ?
  - en déclenchant une opération
  - en interdisant une opération
  - en annulant la transaction

17

Composants d'un SGBD actif

- Analyseur de règles
- Moniteur d'événements
- Exécuteur d'actions
- Évaluateur de conditions
- Moteur de règles
- Dictionnaire de règles

- Deux approches possibles
  - approche intégrée
  - approche sur-couche



18

Types de règles

- Règles de production (IA)
  - IF <Condition sur BD> THEN <Action sur BD>
  - Exécution jusqu'à saturation (point fixe)
- Déclencheur (Trigger) ECA
  - WHEN <Événement> IF <Condition sur BD> THEN <Action sur BD>
  - Modèle d'exécution variable (à préciser)
- Forme dégénérée EA
  - WHEN <Événement> IF <Condition sur BD> THEN <Action sur BD>
  - La condition peut toujours être dans l'action

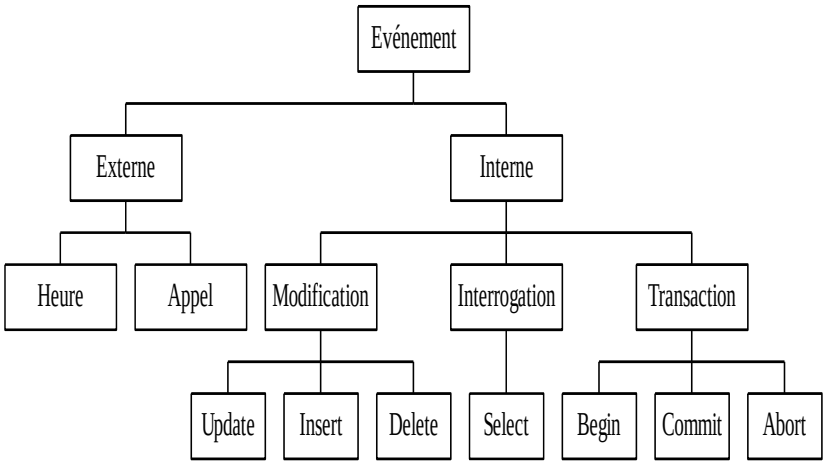
19

Événement

- Événement
  - signal instantané daté, externe ou interne détecté par le SGBD
  - type d'événement : interne, externe, opération, temps, etc.
  - instance d'événement : à un temps donné
  - paramètres : valeur, objet, colonne, etc.
  - contexte : structure nommée contenant les données nécessaire à l'évaluation de la règle, dont les paramètres

20

Typologie événements simples



21

Événements simples et composés

- Événement simple (primitif)
  - Modification de données (Update, Insert, Delete)
  - Recherche de données (Select)
  - Début, validation, annulation de transactions (Begin, Commit, Abort)
  - Temporel (Heure)
  - Utilisateur (Appel de méthode)
- Événement composé
  - Composition d'événements simples par des opérateurs logiques ou/et temporels :
    - ET, OU
    - Séquence , THEN, N TIMES IN <Intervalle>, NOT IN <Intervalle>
    - Temps relatif ou absolu
  - Exemple
    - (1H AFTER UPDATE) OR (AFTER INSERT THEN DELETE)

22

Condition

- Qualification portant sur la base et donnant en résultat
  - soit une valeur booléenne
  - soit un ensemble de tuples
- Des variables de contexte de l'événement peuvent être utilisées
- Des résultats peuvent être passés à l'action dans le contexte
- Optionnelle en générale (règles EA)

Action

- Code exécuté lorsque la règle est déclenchée
- Peut être :
  - une opération BD (requête SQL)
  - une procédure agissant sur la base (L4G)
  - une opération sur transaction (Abort, Commit)
- Peut utiliser les paramètres du contexte
- Peut être exécutée une fois ou pour chaque tuple satisfaisant la condition

EXPRESSION EN SQL3

- Tous les objets (règles, contraintes, ...) sont nommés
- Événement simple : INSERT, UPDATE, DELETE
- Déclenchement avant (BEFORE) ou après (AFTER)
- Granularité ligne (exécuté pour chaque ligne modifiée) ou requête (exécuté une fois)
- Peut référencer les valeurs avant et après mise à jour
- Peut remplacer la mise à jour (INSTEAD OF)

25

Syntaxe

CREATE TRIGGER <nom>
// événement (avec paramètres)
{BEFORE | AFTER | INSTEAD OF}
{**INSERT | DELETE | UPDATE [OF** <liste de colonnes>]}
ON <table> [**ORDER** <valeur>]
[**REFERENCING {NEW|OLD| NEW_TABLE|OLD_TABLE} AS** <nom>]...
// condition
(WHEN (<condition de recherche SQL>)
// action
<Procédure SQL3>
// granularité
[**FOR EACH {ROW | STATEMENT**}])

26

Quelques exemples (1)

- Contrôle d'intégrité
- Ajout d'un abus

```
CREATE TRIGGER InsertAbus BEFORE INSERT ON ABUS
REFERENCING NEW AS N
(WHEN NOT EXIST (SELECT * FROM Vins WHERE NV=N.NV )
THEN ABORT
FOR EACH ROW) ;
```

- Suppression d'un vins

```
CREATE TRIGGER DeleteVins BEFORE DELETE ON VINS
REFERENCING OLD AS O
(DELETE FROM ABUS WHERE NV = O.NV
FOR EACH ROW );
```


Exemple (2)

- Contrôle d'intégrité temporelle
  - EMPLOYE (ID Int, Nom Varchar, Salaire Float)

```
CREATE TRIGGER SalaireCroissant
BEFORE UPDATE OF Salaire ON EMPLOYE
REFERENCING OLD AS O, NEW AS N
(WHEN O.Salaire > N.Salaire
SIGNAL.SQLState '7005' (''Les salaires ne peuvent
  décroitre')) ;
```

Exemple (3)

- Mise à jour automatique de colonnes
  - PRODUITS (NP Int, NF Int, Coût Real, Auteur String, DateMaj Date)

```
CREATE TRIGGER SetAuteurDate
BEFORE UPDATE ON PRODUITS
REFERENCING NEW_Table AS N
(UPDATE N
 SET N.Auteur = USER, N.DateMaj = CURRENT DATE );
```

29

Exemple (4)

- Création automatique de clé

```
CREATE TRIGGER SetCléVins
BEFORE UPDATE ON VINS
REFERENCING NEW_TABLE AS N
(UPDATE N
SET N.NV = SELECT COUNT(*) +1 FROM VINS );
```

30

Exemple (5)

- Gestion de données redondantes
  - EMPLOYE (ID int, salaire float)
  - CUMUL (ID int, Augmentation float)
  - Déclencheur de mise à jour du salaire

```
CREATE TRIGGER
AFTER UPDATE OF salaire ON EMPLOYE
REFERENCING OLD AS a, NEW AS  n
(UPDATE CUMUL SET Augmentation = Augmentation +
   n.salaire - a.salaire
WHERE ID = a.ID
FOR EACH ROW );
```

31

EXECUTION DES REGLES

- Elles sont exécutées suite à une modification produisant un événement déclenchant
- Elles interfèrent avec les contraintes d'intégrité
- Différents types de déclencheurs à considérer :
  - BEFORE puis AFTER
  - ROW ou STATEMENT

32

Algorithme de modification

```
// Exécution d'une modification d'une relation R
Modification(R) {
      // Préparer les mises à jour de R dans R+ et R- ;
      // Exécuter les déclencheur avant mise à jour (BEFORE)
        For each "déclencheur BEFORE d" do { Exécuter(d.Règle) } ;
      // Effectuer les contrôles d'intégrité, puis la mise à jour
        If (Not Integre) then ABORT TRANSACTION ;
      // effectuer la mise à jour
        R = (R - R-) ∪ R+ ;
      // Exécuter les déclencheurs après mise à jour (AFTER)
        For each "déclencheur AFTER" do { Exécuter(d.Règle) } ;
} ;
```

33

Exécution d'une règle

```
// Exécution d'une règle
// WHEN Condition Action FOR EACH <Option>
Exécuter(Condition, Action, Option) {
     // Appliquer à chaque tuple  si option ligne
        if Option = "FOR EACH ROW" then
         { For each t de R = R+  R- do {
           if (Condition(t) = True) then Exécuter (Action(t)) ; }
        if Option = "FOR EACH STATEMENT" then {
           if (Condition(R) = True) then Exécuter (Action(R) ; }
} ;
```

34

Priorités et Imbrications

- Déclencheurs multiple suite à un même événement
  - La priorité est prise en compte (clause ORDER)
  - Sinon l'ordre de déclaration
- Les déclencheurs peuvent être imbriqués
  - Cas d'une mise à jour déclenchant un déclencheur dont la mise à jour déclenche un autre déclencheur
  - Les contextes des déclencheurs sont empilés
- Attention aux boucles
  - Peuvent être valides (calcul de point fixe)
  - En pratique, limitation du niveau d'imbrication

Déclencheurs et transactions

- Différents types de couplage sont possible
  - pour les déclencheurs après, pour les conditions et/ou les actions
- Mode d'exécution :
  - immédiat : dès que l'événement se produit (IMMEDIAT)
  - différé : en fin de transaction (DEFERRED)
- Mode de couplage :
  - dans la même transaction
  - dans une transaction indépendante (DECOUPLED)
- Mode de synchronisation
  - synchrone (après) ou asynchrone (en parallèle) (SYNCHRONOUS)

CONCLUSION

- Un système de règles supporte des événements simples de mise à jour; possibilité d'événement de recherche et d'événements complexes avec intervalles
- Supporte des conditions et actions exécutées avant le traitement naturel de l'événement, ou après, ou en place
- Possibilité de reporter l'exécution en fin de transaction ou dans une transaction différente
- La sémantique est souvent obscure : nécessité de point fixe, possibilité de boucles !

37

38