Intégrité des données

Définition des contraintes
Vérification des contraintes
Triggers

1. Définition des contraintes

- **Objectif :**
  - Détecter les mises à jour erronées et réagir soit en rejetant la transaction, soit en compensant les erreurs.
- **Ceci suppose :**
  - un langage de définition de contraintes d'intégrité
  - la vérification automatique de ces contraintes
- **Avantages :**
  - simplification du code des applications
  - sécurité renforcée par l'automatisation
  - mise en commun et cohérence globale des contraintes

Typologie des contraintes

- **CONTRAINTES STRUCTURELLES**
  - Contraintes de DOMAINE
    - ex: le cru d'un vin est de type chaîne de caractères
  - Contraintes d'ENTITE (unicité et non nullité)
    - toute relation doit posséder au moins une clé et cette clé ne peut pas prendre de valeurs nulles
  - Contraintes REFERENTIELLE
    - ex: l'ensemble des valeurs de l'attribut ABUS.NV doit être inclus dans l'ensemble des valeurs de l'attribut VINS.NV
- **CONTRAINTES COMPORTEMENTALES**
  - Contraintes générales liées à une application spécifique
    - ex: la somme des quantités bues d'un vin doit être inférieure a la quantité produite de ce même vin

Typologie des contraintes comportementales (1)

- **Domaine de variation**
  - Ex: le degré d'un vin ne peut être inférieur à 8
- **Contraintes multi-attributs (horizontales)**
  - Ex: le prix de vente d'un produit doit être supérieur à son coût de production
- **Dépendance fonctionnelle**
  - Ex : CRU, ANNEE -------> DEGRE dans la relation VINS
- **Contraintes temporelles**
  - Ex : le degré d'un vin ne peut pas décroître
- **Contraintes agrégatives (verticales)**
  - Ex : la somme des quantités bues d'un vin doit être inférieure a la quantité produite de ce même vin

Typologie des contraintes comportementales (2)

- **DEPENDANCE D'INCLUSION**
  - Concept de généralisation
  - {valeurs d'un {groupe d'attributs x}} inclus dans {valeurs d'un {groupe d'attributs y}}
- **EXEMPLE :**
  - ENSEIGNANT.NOM inclus dans PERSONNE.NOM
  - ENSEIGNANT ----g----> PERSONNE
- **La dépendance référentielle est un cas particulier de dépendance d'inclusion**
  - {VALEURS DE X} inclus dans {VALEURS DE Y} et Y EST CLE
    - EXEMPLE : ABUS.NV inclus dans VINS.NV

Association des contraintes

- **Une contrainte d'intégrité peut être :**
  - Associée à un domaine
    - Spécifiée au travers de la clause CREATE DOMAIN
  - Associée à une relation
    - Spécifiée au travers de la clause CREATE TABLE
  - Dissociées
    - Spécifiée au travers de la clause CREATE ASSERTION

Contraintes associées aux domaines

CREATE DOMAIN <nom> <type> [valeur]
[**CONSTRAINT** nom_contrainte **CHECK (**condition**)**]

Exemple:

CREATE DOMAIN couleur_vins **CHAR(5) DEFAULT** 'rouge'

CONSTRAINT couleurs_possibles **CHECK**
(VALUE IN ('rouge', 'blanc', 'rosé'))

Contraintes associées aux relations

CREATE TABLE <nom_table>
(<def_colonne> *
[<def_contrainte_table>*]) ;

< def_colonne > ::= <nom_colonne> < type | nom_domaine >
[CONSTRAINT nom_contrainte
< NOT NULL | UNIQUE | PRIMARY KEY |
CHECK (condition) | REFERENCES nom_table (liste_colonnes) >]
[NOT] DEFERRABLE

< def_contrainte_table > ::= CONSTRAINT nom_contrainte
< UNIQUE (liste_colonnes) | PRIMARY KEY (liste_colonnes) |
CHECK (condition) |
FOREIGN KEY (liste_colonnes) REFERENCES nom_table (liste_colonnes) >

Contraintes associées aux relations

```
CREATE TABLE VINS
  (NV INTEGER PRIMARY KEY,
   couleur COULEURS_VINS,
   cru VARCHAR(20),
   millesime DATE,
   degre CHECK (degre BETWEEN 8 AND 15) NOT DEFERRABLE,
   quantite INTEGER,

   CONSTRAINT dependance_fonctionnelle
   CHECK (NOT EXISTS (SELECT *
      FROM VINS
      GROUP BY cru,millesime
      HAVING COUNT(degre) > 1)
   NOT DEFERRABLE) ;
```

Contraintes référentielles

FOREIGN KEY (liste_colonnes**)**
REFERENCES nom_table **(**liste_colonnes**)**
[ON DELETE {CASCADE | SET DEFAULT | SET NULL}]
[ON UPDATE {CASCADE | SET DEFAULT | SET NULL}]
[NOT] DEFERRABLE

- **Les contraintes référentielles caractérisent toutes les associations**
- **Problème des contraintes référentielles croisées ==> mode DEFERRABLE**
- **En cas de violation de la contrainte, la mise à jour peut être rejetée ou bien une action de correction est déclenchée ==>**

  - **ON DELETE spécifie l'action à effectuer en cas de suppression d'un tuple référencé**
  - **ON UPDATE spécifie l'action à effectuer en cas de mise à jour de la clé d'un tuple référencé**

Contraintes référentielles: exemple

```
CREATE TABLE ABUS
  ( NB INTEGER NOT NULL,
    NV INTEGER NOT NULL,
    date DATE,
    qte QUANTITE,
    UNIQUE (NB, NV, date)

    CONSTRAINT référence_buveurs
          FOREIGN KEY NB
          REFERENCES  BUVEURS (NB)
          ON DELETE CASCADE
          DEFERRABLE
    ) ;
```

Contraintes dissociées

CREATE ASSERTION nom_contrainte **CHECK (**condition**)**

Remarque: les contraintes dissociées peuvent être multi-tables

Exemple:

CREATE ASSERTION quantite_produite

CHECK ((SELECT SUM(quantite) **FROM** VINS) **>**

(**SELECT** SUM(quantite) **FROM** ABUS) **)**

2. Vérification des contraintes (1)

- **Méthode par détection d'incohérence**
  - toute mise à jour m est exécutée sur la base D;
  - l'état de la base D est changée en Dm ;
  - si Dm est détecté incohérent, on doit restituer l'état D .
- **Notion de post-test:**
  - A et A' sont des assertions
  - A' est un post-test pour A et m ssi
  - { D / A => Dm / A} <=> Dm / A'
- **Difficultés :**
  - (i) trouver un A' plus simple à vérifier que A
  - (ii) défaire la transaction en cas d'incohérence.

Vérification des contraintes (2)

- **Méthode par prévention des incohérences**
  - une mise à jour m n'est exécutée que si l'état résultant de la base Dm est garanti être cohérent
- **notion de pre-test**
  - A et A' sont des assertions
  - A' est un pré-test pour A et m ssi
  - {D / A => Dm / A} <=> D / A'
- **problèmes:**
  - (i) Comment laisser passer les seules mises à jour permises ?
  - (ii) Modifier la mise à jour en ajoutant condition : généralité ?

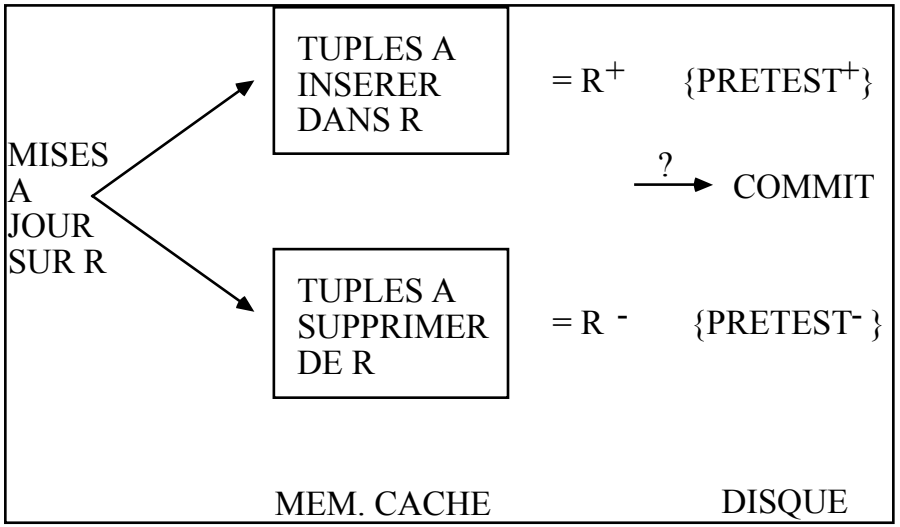
Vérification des contraintes (3)

- **Exemple de vérification préventive**
  - PRE-TEST A' = Update(A)
- **L'algorithme ajoute conjonctivement l'assertion A à la condition de la mise à jour.**

- **Exemple: SAL> SMIC**
  - UPDATE EMPLOYE
  - SET SAL=SAL*0.9
  - WHERE NOM = 'RALEUR'
- **Devient:**
  - UPDATE EMPLOYE
  - SET SAL=SAL*0.9
  - WHERE NOM = 'RALEUR'
  - AND SAL*0.9 > SMIC

Vérification des contraintes (4)

* NOTION DE PRE-TESTS DIFFERENTIELS



EXEMPLE : *ABUS REFERENCE VINS*

PRE-TEST+ (ABUS) : ABUS+.NV = VINS.NV
PRE-TEST- (ABUS) : RIEN
PRE-TEST+ (VINS) : RIEN
PRE-TEST- (VINS) : COUNT (ABUS.NV WHERE (ABUS.NV=VINS-.NV) = 0

⇒ TRES EFFICACE MAIS COMPLEXE A IMPLANTER

Exemples de 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^-

3. Déclencheurs (Triggers)

- **Déclencheur :**
  - action ou ensemble d'actions déclenchée(s) automatiquement lorsqu'une condition se trouve satisfaite après l'apparition d'un événement
- **Un déclencheur est une règle ECA**
  - Evénement = mise à jour d'une relation
  - Condition = optionnelle, équivaut à une clause <WHERE>
  - Action = exécution de code spécifique (requête SQL de mise à jour,exécution d'une procédure stockée, abandon d'une transaction, ...)
- **De multiples usages sont possibles :**
  - contrôle de l'intégrité
  - maintien de statistiques
  - mise à jour de copies multiples, ...

Définition des triggers

CREATE TRIGGER <nom-trigger>
<événement>
[<condition>]
<action *>

<événement> ::=
BEFORE | **AFTER**
{INSERT | **DELETE** | **UPDATE [OF** <liste_colonnes>**]}**
ON <nom_de_table>
<condition> ::=
[REFERENCING OLD AS <nom_tuple> **NEW AS** <nom_tuple>**]**
WHEN <condition_SQL**>**
<action> ::=
{requête_SQL **[FOR EACH ROW]**
| exec_procédure | **COMMIT** | **ROLLBACK**}

Exemples de trigger

```
CREATE TRIGGER degré_croissant
BEFORE UPDATE OF degre ON VINS
REFERENCING OLD AS old_vin NEW AS new_vin
WHEN (new_vin.degre < old_vin.degre)
ROLLBACK
FOR EACH ROW

CREATE TRIGGER référence_vins
BEFORE DELETE ON VINS
DELETE FROM ABUS
WHERE ABUS.NV = VINS.NV
FOR EACH ROW
```

4. Conclusion

- **Le modèle relationnel offre**
  - des contraintes d'intégrités riches
  - des mécanismes de vérification efficaces
  - Des mécanismes événementiels puissants
- **Problèmes difficiles :**
  - Contraintes avec agrégats
  - Triggers récursifs