TD N° 1
L'intégrité des données

Bases de données avancées

M. Laïdi FOUGHALI

Plan

- Introduction
- Exemples de contraintes
- Typologie des contraintes
- Contraintes structurelles
- Contraintes non structurelles
- Expression des contraintes en SQL
- Analyse des contraintes d'intégrité
- Contrôle des contraintes d'intégrité
- Contrôles au vol des contraintes simples en SQL2
- Déclencheurs (Triggers)
- Vos questions ?

Introduction

- Un SGBD doit assurer la ***cohérence des données*** lors des mises à jour.
- Les données d'une base ne sont pas indépendantes, mais obéissent à des règles sémantiques appelées ***contraintes d'intégrité***.
- Les contraintes d'intégrité peuvent être déclarées explicitement ou peuvent être discrètement implicites.
- La validation des contraintes est effectuée à la fin de chaque transaction par un contrôle de non violation.
- Les techniques de validation des contraintes sont :
  1. Prévention : empêcher les mises à jour non valides de se produire,
  2. Détection : défaire les transactions incorrectes.
  3. Déclencheurs **(*triggers*)**: déclenchement d'opérations par des évènements associés à la base ***(base de données actives)***.

Exemples de contraintes

1. Cas simple: égalité de deux données.

2. Cas complexe: assertion comportant de multiples données.

Exemple : *Le montant des commandes non réglées ne doit pas dépasser 20% du montant du chiffre d'affaire déjà réalisé par le client, (règle de gestion).*

Typologie des contraintes

- **Contraintes structurelles :**
  - *« Contrainte d'intégrité spécifique à un modèle exprimant une propriété fondamentale d'une structure de données du modèle."*
  - Les structures concernent les table, les colonnes et les lignes.
- **Contraintes non structurelles** :
  - *« Contrainte d'intégrité exprimant une règle d'évolution que doivent vérifier les données lors des mises à jour."*

Remarques :

1. Les contraintes structurelles sont définies par **CREATE TABLE**.
2. Les contraintes non structurelles sont définies par **CREATE ASSERTION** ou **CREATE TRIGGER**.

Contraintes structurelles

1. **Contrainte de non nullité** (NOT NULL): La valeur d'un attribut doit être renseignée.
2. **Unicité de clé** (PRIMARY KEY) **:** Définit des attributs non nuls comme clé primaire.
3. **Contrainte d'unicité** (UNIQUE) : Définit des attributs comme clé candidate : L'unicité par rapport aux valeurs clé renseignées.
4. **Contrainte référentielle** (FOREIGN KEY): Les valeurs des attributs d'une clé étrangère doivent être égales aux valeurs des attributs d'une clé primaire ou aux valeurs des attributs non nuls d'une clé candidate.
5. **Contrainte de domaine** (DOMAIN). Définit un type utilisateur.

Remarque : Certaines contraintes structurelles peuvent être qualifiées de contraintes de comportement, par exemple l'unicité de clé.

Contraintes non structurelles (1/2)

Pour le modèle relationnel, ces contraintes peuvent être exprimées sous forme d'assertions de la logique du premier ordre.

On distingue :

1. **Dépendances fonctionnelles**: X → Y (X détermine Y) si pour toute valeur de X il existe une valeur unique de Y associée.
   Exemple: Le numéro d'étudiant détermine un seul nom de famille.
2. **Dépendances multivaluées**: (*Généralisation du cas précédent*)
   X->> Y (X multi détermine 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 de la relation R.
3. **Dépendances d'inclusion**: Spécifie que les valeurs d'un groupe de colonnes d'une table doivent rester incluses dans celles d'un groupe de colonnes d'une autre table. Elles généralisent les contraintes référentielles aux cas de colonnes quelconques.

Contraintes non structurelles (2/2)

4. **Contraintes temporelles:**

   Plus sophistiquées! Font intervenir le temps pour suivre les mises à jour d'un attribut.

   Exemple : Un salaire ne peut que croître.

5. **Contraintes équationnelles**:

   - Comparer deux expressions arithmétiques calculées à partir de données en forçant l'égalité ou l'inégalité.
   - La dimension temporelle peut être prise en compte en faisant intervenir des données avant et après mise à jour.

Remarque: **les dépendances généralisées** regroupe les différents types de dépendances entre attributs (dépendances fonctionnelles, multivaluées et d'inclusion), pour mieux contrôler les redondances.

Expression des contraintes en SQL (1/5)

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>

Création de table et contrainte d'intégrité en SQL 1

Expression des contraintes en SQL (2/5)

```
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))
```

Exemple de création de table avec contrainte en SQL 1

Expression des contraintes en SQL(3/5)

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 référentielle en SQL 2

Mode	Explication
NO ACTION	Ne provoque aucune action, et a donc un effet identique à l'absence de la clause comme en SQL1.
CASCADE	Signifie qu'il faut répercuter l'action (DELTE/UPDATE) aux tuples correspondant de la table dépendante.
SET DEFAULT	Indique qu'il faut remplacer la clé étrangère des tuples correspondant de la table dépendante par la valeur par défaut qui doit être déclarée au niveau de l'attribut.
SET NULL	A un effet identique que (SET DEFAULT), avec la valeur nulle.

Expression des contraintes en SQL (4/5)

CREATE ASSERTION <nom de contrainte>

[{**BEFORE COMMIT** |

AFTER {INSERT|**DELETE**|**UPDATE [OF**(Attribut+)]} **ON** <Relation>}...]

CHECK <Condition>

[FOR [EACH ROW OF] <Relation>]

Création de contrainte indépendante en SQL 2

Une assertion peut être vérifiée immédiatement après chaque mise à jour précisée (clause AFTER), ou en fin de transaction (clause BEFORE COMMIT). La condition peut porter sur une table entière (option FOR), ou sur chaque ligne de la table (option FOR EACH ROW OF).

Expression des contraintes en SQL (5/5)

Exemple 1 **:** S'assure que chaque vin a un degré supérieur à 10.

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

Exemple 2 **:**

```
CREATE ASSERTION QUALITÉSUP AFTER INSERT ON VINS
    CHECK ((SELECT COUNT(*)
    FROM ABUS A, VINS V
    WHERE A.NV = V.NV
    AND V.QUALITÉ = "SUPÉRIEURE") > 10).
```

Analyse des contraintes d'intégrité

Effectuer un TEST DE COHÉRENCE ET DE NON-REDONDANCE sur les contraintes pour répondre aux questions suivantes:

1. Les contraintes sont-elles cohérentes entre elles ?
2. Ne sont-elles pas redondantes ?
3. Quelles contraintes doit-on vérifier suite à une insertion, une suppression, une mise à jour d'une table ?
4. Est-il possible de simplifier les contraintes pour faciliter le contrôle ? Ces différents points sont examinés ci-dessous.

Remarque: *Dans la pratique*, aucun SGBD (sauf peut-être les trop rares SGBD déductifs) n'est capable de vérifier le test, et encore moins de déterminer un ensemble minimal de contraintes.

Contrôle des contraintes d'intégrité

Afin de garder la base cohérente, le SGBD doit vérifier que les contraintes d'intégrité sont satisfaites à chaque fin de transaction. En pratique, cette opération s'accomplit à chaque mise à jour

- **Méthode de détection :** Méthode consistant à retrouver les tuples ne satisfaisant pas une contrainte dans la base après une mise à jour, et à rejeter la mise à jour s'il en existe. Un **Post-test** est appliqué à la base après mise à jour permettant de déterminer si une contrainte d'intégrité I est violée.
- **Méthode de prévention :** Méthode consistant à empêcher les modifications de la base qui violeraient une quelconque contrainte d'intégrité**. Un pré-test** est appliqué à la base avant une mise à jour permettant de garantir la non-violation d'une contrainte d'intégrité par la mise à jour.

Contrôles au vol des contraintes simples en SQL2

- Effectuer une prévention au vol, juste avant la mise à jour du tuple, en employant un pré-test simple consistant à vérifier que la nouvelle valeur ne figure pas dans un index, pour les cas de :
  1. **Unicité de clé.**
  2. **Intégrité référentielle :**
     i. Insertion dans la table dépendante.
     ii. Mise à jour de la colonne référençante dans la table dépendante.
     iii. Suppression dans la table maître.
     iv. Modification de clé dans la table maître.

Remarque: Les contrôles deviennent complexes en cas d'opérations en cascade.

- **Contrainte de domaine :** Pour un CHECK <condition> avec une condition simple, il suffit de vérifier avant d'appliquer la mise à jour que la valeur qui serait donnée à l'attribut après mise à jour vérifie la condition.

Déclencheurs (Triggers)

- Un SGBD **actif** pourra déclencher une opération subséquente à un événement donné, interdire une opération, ou encore envoyer un message à l'application, voire sur Internet.
- Une **base de données active** va permettre de déplacer une partie de la sémantique des applications au sein du SGBD.
- La conception de bases de données actives efficaces est un problème difficile.
- **Un déclencheur (Trigger)** est règle dont l'évaluation de type procédural est déclenchée par un événement, généralement exprimée sous la forme d'un triplet Événement – Condition – Action :

WHEN <Événement> IF <Condition sur BD> THEN <Action sur BD>

Remarque: *Les SGBD actifs ne sont donc pas standard.*

Expression des déclencheurs en SQL3

CREATE TRIGGER <nom>

// événement (avec paramètres)

{BEFORE | AFTER |

INSTEAD OF} // Remplace un ordre (sécurité)

{INSERT | DELETE | UPDATE [OF <liste de colonnes>]**}**

ON <table> [**ORDER** <valeur>] // Priorité

[REFERENCING {NEW | OLD |

NEW_TABLE | OLD_TABLE} // Tables différentielles

AS <nom>]...

// condition

(WHEN (<condition de recherche SQL>)

// action

<Procédure SQL3>

// granularité

[FOR EACH {ROW | STATEMENT}])

Déclencheurs, 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))

ABORT TRANSACTION

FOR EACH ROW) ;

// Suppression d'un vin

CREATE TRIGGER DeleteVins

BEFORE DELETE ON VINS **REFERENCING OLD AS** O

(DELETE FROM ABUS **WHERE** NV = O.NV

FOR EACH ROW) ;

Déclencheurs, Contrôle d'intégrité temporelle

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 décroître')

FOR EACH ROW);

Déclencheurs, Génération automatique de clé

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

Déclencheurs, Gestion de données agrégative

```
CREATE TRIGGER CumulSal
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) ;
```

Vos questions ?