Plan : Contraintes d'intégrité et bases de données actives

1. Les Contraintes
    1. Les différents types de contraintes
    2. Contraintes et vérifications pendant la transaction
    3. Analyse des contraintes.
    4. Différentes méthodes de vérification pendant la transaction.
    5. Effets de bord indésirables
2. Les contraintes et assertions en SQL2.
    1. Contraintes et assertions
    2. Les inconvénients
3. Les déclencheurs.
    1. Syntaxe SQL
    2. Problèmes à gérer
    3. Déclencheurs vs Assertions

Tables utilisées

CLIENTS	(nclient, nom, prenom, statut)
FACTURES	(Nfact, remise, client)
LIGNEFACT	(Nfact, NArt, nb)
ARTICLES	(NArt, prixHT, catTaux)
TAUX	(Ntaux, TauxTVA)

1. Contraintes 1. Typologie

a) Structurelles :

Partie intégrante du modèle, spécifiable dans le schéma relationnel.
Concerne un seul enregistrement (hors cas particulier de la clé primaire).

- Unicité de clé : Primary key,
- Contrainte référentielle Foreign Key references
- Contrainte de domaine Check AGE < 130
- De non nullité Not NULL

En SQL, il est possible en général de préciser ces contraintes lors de la déclaration de la table lors du CREATE TABLE.

1. Contraintes 1. Typologie

b) Non structurelles :

Non inhérentes au modèle, ne peuvent être définies dans le schéma relationnel. Concernent souvent plusieurs tables.

- Sur une table :

  CMINARTNB Il y a au moins 10 articles de chaque **catTAux**.

- Sur plusieurs tables.

  CCOUTMIN Toutes les factures sont d'un montant HT > 100

  CREMISECOUT Si la remise est >0,2, alors le montant TTC est > 2000

Peut s'exprimer comme une expression SQL.

Exemple : NOT EXIST (SELECT CATEGORIES, COUNT(*) AS NB
FROM ARTICLES
GROUP BY CATEGORIES HAVING NB < 10)

1. Contraintes 1. Typologie

c) Contraintes temporelles :

concernent l'évolution dans le temps des données de la base.

- **CBAISSE** On ne peut diminuer en une seule fois le prix d'un article de + de 50 %.
- **CSTATUT** Entre **veuf**, **célibataire**, **divorcé, marie**, toutes les évolutions ne sont pas possibles.

1. Contraintes 2. Contraintes et transactions

Contraintes respectées avant et en fin de *transaction*, mais pendant ?
Parfois, il est nécessaire de désactiver les contraintes le temps d'une transaction.

- Cas de « circuit » de clés étrangères dans le schéma relationnel. (l'œuf ou la poule?)
- **CCOUTMIN** : impossibilité de créer la moindre facture ?
- **CMINARTNB** : Impossibilité de remplir la base ?

1. Contraintes 3 Analyse des contraintes.

Lorsqu'on étudie le problème, on fait la liste de toutes les contraintes que doit respecter la base de données.

Il faut n'en oublier aucune.

1. Contraintes 3 Analyse des contraintes.

a) Cohérence : l'ensemble des contraintes est-il cohérent ?

Ex : si l'on rajoute **CCOUTMAX :** toutes les factures sont d'un montant HT < 90,

Aucune instance de la base ne peut respecter à la fois **CCOUTMAX** et **CCOUTMIN !**

Pas de système vérifiant automatiquement la cohérence des contraintes.

1. Contraintes 3 Analyse des contraintes.

b) Non redondance : l'ensemble des contraintes est-il minimal ?

Ex : si l'on rajoute **CCOUTMIN2 :** toutes les factures sont d'un montant HT > 90,

La contrainte **CCOUTMIN** devient maintenant redondante.

Pas de système vérifiant automatiquement la non-redondance des contraintes.

1. Contraintes 3 Analyse des contraintes.

c) Simplification opérationnelle : Limiter les moments où l'on fait les vérifications.

Suivant la contrainte, et l'opération faite sur la base, la vérification n'est pas toujours utile. Donner pour chaque contrainte, la liste des opérations (T,OP) (T : table, OP : {INSERT, DELETE, UPDATE}) risquant de poser problème pour la contrainte.

Exemple :

CMINARTNB : (ARTICLES,DELETE),(ARTICLES,UPDATE), (TAUX,INSERT)

CSTATUT : (CLIENT,UPDATE)

CCOUTMIN : (ARTICLES, UPDATE), (LIGNES,UPDATE), (LIGNES,DELETE), (FACTURES,INSERT)

Parfois, le système peut le faire automatiquement.

1. Contraintes 3 Analyse des contraintes.

d) Simplification différentielle : Limiter les accès à la base.

Suivant la contrainte, et l'opération faite sur la base : il est parfois inutile de consulter le contenu complet de la base. La consultation des tuples ajoutés R+ et supprimés R- ne suffit-elle pas ?

Exemples :

- **CPRIXMIN :** Contrainte de domaine Check PRIXHT > 20 ?
  - (ARTICLES,INSERT), ARTICLES+
  - (ARTICLES,UPDATE), ARTICLES+
- **CCOUTMIN** : Nécessaire à chaque fois de recalculer la facture.
- **CMINARTNB** :
  - (ARTICLES,DELETE), ARTICLES
  - (ARTICLES,UPDATE), ARTICLES

1. Contraintes 4 Différentes méthodes de vérification

a) Méthode Brute :

Le système fait la transaction, évalue complètement les contraintes, si échec, on défait.

b) Méthode différentielle :

Utilise les simplifications, on calcule R+,R-, on fait les tests, on reporte R+ et R- si les tests sont validés.

c) Méthode du prétest :

On complète la base en rajoutant des informations.

CMINNBART ; garder à part le nb d'articles par catégories.

Lors de la suppression (ou modification) d'un article, on saura immédiatement si l'on peut la faire sans consulter la table complète.

1. Contraintes 5 Effets de bord indésirables.

a) Sur la requête.

CMINARTNB : Il y a 12 articles de luxe, dont 4 de nart < 20

Que fait « DELETE FROM ARTICLES WHERE Nart <20 »

On annule tout ? On garde 2 articles, lesquels ?

CPRIXMIN : tout prixHT >10. (3 prix sont inférieurs à 20).

UPDATE ARTICLES SET PRIX = PRIX *0.5,

On annule tout ? Ou seulement les 3 mises à jour posant problème ?

1. Contraintes 5 Effets de bord indésirables.

b) Deux contraintes aux effets différents.

```
CREATE TABLE T1 (
        C1 INT PRIMARY KEY)
CREATE TABLE T2 (
        C3 INT PRIMARY KEY,
        C2  INT FOREIGN KEY REFERENCES T1(C1) ON DELETE
    CASCADE),
```

Contrainte **CMINT2**, T2 contient au moins 10 enregistrements.

Supprimer un enregistrement de T1 ?

DELETE ou non dans le T2 ?

2. Les contraintes et assertions en SQL

- FOREIGN et PRIMARY KEY, UNIQUE, NOT NULL assez standard
- Tout le reste assez VARIABLE suivant le SGBD

2. Les contraintes en SQL. 1 Constraints et assertions.

Partie déclarative, on déclare simplement ce qui doit être respecté, on ne décrit pas comment il faut le faire, ni traiter l'échec.

Si une mise à jour ne respecte pas la contrainte, la mise à jour est annulée c'est tout. **Et rien de plus.**

2. Les contraintes en SQL. 1 Constraints et assertions.

1) Contraintes structurelles : *CONSTRAINTS*

   PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK …

2) Contraintes non structurelles (hors temporelles) : *ASSERTIONS*

   CREATE ASSERTION nomassert {DEFERRABLE|NOT DEFERRABLE}

   [BEFORE COMMIT | {AFTER|BEFORE} {INSERT|DELETE| UPDATE[OF {nomsattributs}]

   ON nomtable

   CHECK conditionsql

   [FOR EACH ROW OF nomtable]

2. Les contraintes en SQL. 1 Constraints et assertions.

Exemple :

```
CREATE ASSERTION PRIXMIN       AFTER INSERT ON ARTICLE
            CHECK NOT EXIST (SELECT * FROM ARTICLES WHERE PRIXHT < 10 )
            FOR EACH ROW OF ARTICLES

CREATE ASSERTION COUTMIN DEFERRABLE
            BEFORE INSERT INTO FACTURE
            CHECK NOT EXIST (SELECT * FROM COUTTOTAL WHERE MONTANT < 100 );
```

Utilisation

debut de transaction.

SET ASSERTION COUTMIN INITIALLY DEFERRED *{IMMEDIATE}* ;

……

COMMIT ;

2. Les contraintes en SQL. 2 Les inconvénients.

- Impossibilité d'exprimer les contraintes temporelles

- Pas de traitements autre que ne rien faire,
  - Message d'erreur adapté ?
  - Correction automatique et traitement suivant le problème ?

3. Les déclencheurs

Bases de données actives : la base de données réagit aux événements.

Un *déclencheur* c'est la donnée de :

- Un événement, (+- comme pour les assertions)
- Une condition (comme pour les assertions) optionnelle
- Une action (requête SQL, procédure PL/SQL, etc….

Chaque fois que l'événement se produit :

1) On évalue la condition.
2) Si la condition est vérifiée, on effectue l'action.

(Si pas de condition, on effectue l'action systématiquement)

3. Les déclencheurs . 1 Syntaxe SQL

CREATE TRIGGER *nomdeclencheur*

evenement ON *nomtable*

[REFERENCING {OLD vieille_ligne, NEW nouvelle_ligne}

|{OLD_TABLE vieille_table, NEW_TABLE nouvelle_table}]

{FOR EACH ROW | FOR EACH STATEMENT}

[WHEN CONDITION]

BEGIN

….. {*Suite de requêtes SQL, morceau de code PL/SQL, ABORT, COMMIT,etc…}*

END ;

3. Les déclencheurs . 1 Syntaxe SQL

Evenement

{BEFORE|AFTER|INSTEAD OF }

{INSERT|DELETE|UPDATE [OF *liste_de_colonnes*]}]

ON *nomtable*

3. Les déclencheurs . 1 Syntaxe SQL

Exemple :

```
CREATE TRIGGER BAISSE
    INSTEAD OF UPDATE OF PRIX ON ARTICLES
    REFERENCING OLD V, NEW N
    FOR EACH ROW
    WHEN V. PRIXHT > 2 * N.PRIXHT
    BEGIN
            afficher (« BAISSE INTERDITE » )
    END ;
```

Remarque : OLD interdit si événement INSERT, NEW interdit si événement DELETE.

3. Les déclencheurs . 2 Difficultés à prendre en compte.

- Quand prendre en compte un déclencheur ?

  Dès que l'événement se produit ? Après la fin de la requête ayant produit l'évènement ? Priorité des déclencheurs par les événements ?
- Si plusieurs déclencheurs sur le même événement ?

  Lequel choisir en priorité ? Si l'un annule (INSTEAD OF) l'événement, quid des autres déclencheurs ?
- Action exécutée dans la foulée de l'évaluation ou bien attente possible ?
- Déclencheurs en cascade :

  L'action d'un déclencheur produit des événements. Quand prendre en compte ces évènements, immédiatement ou bien après traitement de tous les déclencheurs déjà en attente ?

3. Les déclencheurs . 3 Déclencheurs vs Assertions

DECLENCHEURS

- Vision Procédurale
- Plus de possibilités.
- Risque d'usine à gaz, Attention à ne pas perdre la vue d'ensemble.

ASSERTIONS

- Vision déclarative
- Moins de possibilité
- Simplicité/Clarté.