23/10/2020

CER 6.8 JONTURES, VUES, TRIGGERS

Heidy Kengne
UCAC-ICAM

Table des matières

I. ANALYSE DU BESOIN

a) ANALYSE DU CONTEXTE

L'entreprise AOC décide, pour palier à la surabondance des données qui cause un ralentissement de sa productivité, d'épurer sa base de données. Deux options s'offrent à elle : la suppression ou l'archivage. Malheureusement, les données à supprimer sont toujours liées aux données en cours d'utilisation.

b) BESONS

- Besoin de savoir bien manipuler une BDD
- Besoin d'aquerir des connaissances sur les jointures

c) CONTRAINTES

- Impossibilité de supprimer les données directement
- MCD fourni
- L'idée de deplacer les données a ete abandonnée

II. PROBLEMATIQUE

Comment reduire le nombre d'informations d'une base de donnée tout en la gardant à jour?

III. GENERALISATION

Epuration des données

IV. PISTES DE SOLUTION

- Le Langage de Contrôle de Données nous permettra d'archiver les données
- L'étude des syntaxes d'automatisation des requêtes nous aidera à restreindre la suppression
- L'utilisation des procédures stockées nous permettra de résoudre le problème
- L'utilisation de la différence nous permettra de résoudre le problème

V. PLAN D'ACTION

1. Définir les mots clés
2. Etude des méthodes d'archivage des données et de gestion utilisateur dans une BDD
3. Etude des Triggers et les vues
4. Epurer la base de donnée

VI. REALISATION DU PLAN D'ACTION

1. Définitions des mots clés

Archiver : est un moyen de sauvegarde des données à des fins personnels

Données en cours : données en cours d'utilisation

2. Etude des méthodes d'archivage des données et de gestion utilisateur dans une BDD

Une donnée est un triplet composé d'un intitulé renvoyant à un concept (un salaire brut, par exemple), d'un domaine de définition (spécifiant l'ensemble des valeurs admises pour ce concept) et d'une valeur à un instant t.

Une base de données est « a collection of related data with a logically coherent structure, some inherent meaning, a specific purpose, a largely varying size, a scope or content of varying breadth, a physical organization of varying complexity and a persistence over a long period of time ».3

Il existe de nombreux types de bases de données : relationnel, hiérarchique, objet, XML, NoSQL… La démarche et les solutions conceptuelles proposées dans l'étude sont indépendantes du type de base de données. Toutefois, les solutions sur le marché sont exclusivement destinées au modèle relationnel et XML (dans une moindre mesure).

L'enjeu est donc d'archiver les données issues d'une base de données en prenant soin d'en préserver la signification, d'en capturer tous les éléments pour être certain que les données seront, après archivage, toujours utilisables. Ceci est d'autant plus difficile que les données à archiver ont une structure complexe (certaines données interdépendantes pouvant être réparties sur plusieurs tables, voire plusieurs bases de données) et que le volume des données peut être très important. Enfin, l'extraction des données ne doit pas mettre en péril la consistance de la base de données de production.

Dans les bases de données administratives, les données sont parfois soumises à une valeur probante, puisqu'elles sont la preuve d'un fait correspondant du réel à un moment t. Par conséquent, les valeurs correspondantes doivent être conservées, y compris en cas de modifications pour des raisons administratives ou de contrôle. Afin de capturer ces modifications, il est nécessaire de mettre en place un historique des versions via un design adéquat du schéma de la base. À cet égard, nous renvoyons aux travaux d'Isabelle Boydens4, responsable du centre de compétence Data Quality au sein de la section Recherche de Smals, qui représentent à l'heure actuelle les réflexions les plus avancées sur le sujet. Dans tous les cas, ce schéma doit être prévu en amont de ARCHIVAGE DES BASES DE DONNEES – AVRIL 2013

l'application, dès la conception de la base de données. On ne le répétera donc jamais assez : l'archivage doit se penser dès le début d'un projet.

L'archivage rigoureuse des données nécessite un suivie méthodologique :

1) Définir le business case : pour quelles raisons souhaite-t-on archiver les données ?

Pour des raisons légales ? Pourinformation ? Pour augmenter les performances des applications en production ? Les réponses à ces questions auront naturellement une influence sur la solution à mettre en œuvre.

2) Sélectionner les données à archiver sur la base des obligations légales, des besoins en consultation et en utilisation des données. Cette étape consiste à définir des objets métier autonomes, par la définition d'une table racine et le suivi des relations. C'est un travail important et délicat. On prendra également soin de déterminer la valeur des données (probantes vs informatives) ainsi que les risques liés à leur non-conservation.
3) Sélectionner les métadonnées, c'est-à-dire les informations complémentaires de description, de structure et de préservation qu'il faut archiver avec les données en vue d'assurer leur utilisation logique, syntaxique et sémantique dans le futur. Conserver des données inutilisables à terme est coûteux et inutile.
4) Définir les accès, à savoir qui peut accéder aux données, selon quelle fréquence, à quelles données.
5) Définir un format d'archivage : la durée de vie des données est largement supérieure à la durée de vie des applications, en ce compris les systèmes d'archivage. Il faut donc sélectionner un format dit pérenne, afin de réduire autant que possible les migrations de format (qui seront de toute façon un jour inévitable). Dans la grande majorité des cas, il s'agira de XML, CSV ou flat file, plus rarement de dump SQL.
6) Élaborer la capture des données, ce qui inclut aussi bien la définition d'une périodicité d'extraction que le processus et les critères de sélection.
7) Définir les durées de conservation et le sort final au terme de celles-ci : on n'archive que rarement ad vitam aeternam. Une durée temporelle ainsi qu'un évènement déclencheur de cette durée doivent être définis.
8) Enfin, mettre en place une organisation permettant de réaliser

3. Etude des Triggers et les vues
   a. Notions de vues

Pour Une vue est un objet de la base de données constitués d'un nom et d'une requête de sélection.

Principe

Le principe d'une vue est simple, on stocke une requête « SELECT » en lui donnant un nom, et on peut l'appeler directement par son nom. Une vue permet de visualiser le résultat d'une requête ayant été effectuée au préalable sans avoir à la retaper.

Elle est stockée de manière durable comme les tables ou procédures stockées

Création

Pour créer une vue, on fait usage de la commande « CREATE VIEW », la syntaxe est la suivante :

CREATE [OR REPLACE] VIEW nom_vue

AS requete_select;

La clause « **OR REPLACE** » est facultative ; lorsqu'elle est utilisée durant la création d'une vue si cette vue n'existe pas elle sera créée sinon elle sera remplacée. Si la clause « OR REPLACE » n'est pas utilisée si la vue créée existe déjà, une erreur sera signalée.

Colonnes d'une vue

Les colonnes d'une vue peuvent être lister directement après la création de la vue sans toutefois utiliser les alias dans la clause « SELECT », voir l'exemple ci-dessous :

CREATE VIEW V_Animal_details (id, sexe, date_naissance, nom, commentaires, espece_id,

race_id, mere_id, pere_id, disponible, espece_nom, race_nom)

AS SELECT Animal.id, Animal.sexe, Animal.date_naissance, Animal.nom, Animal.commentaires,

Animal.espece_id, Animal.race_id, Animal.mere_id, Animal.pere_id, Animal.disponible,

Espece.nom_courant, Race.nom

FROM Animal

INNER JOIN Espece ON Animal.espece_id = Espece.id

LEFT JOIN Race ON Animal.race_id = Race.id;

Pour afficher les colonnes d'une vue, on utilise la commande « **DESCRIBE nom_vue** ».

Requêtes stockées dans une vue

Dans une vue, on peut tout simplement avoir une requête « **SELECT** » comme suit :

CREATE VIEW V_lapins

AS SELECT id, sexe, date_naissance, nom, commentaires, espece_id, race_id, mere_id, pere_id,

disponible

FROM Animal

WHERE espece_id = 1;

On peut utiliser des fonctions d'agrégations dans une vue, à l'exemple de :

CREATE OR REPLACE VIEW V_Nombre_espece

AS SELECT Espece.id, COUNT(Animal.id) AS nb

FROM Espece

LEFT JOIN Animal ON Animal.espece_id = Espece.id

GROUP BY Espece.id;

On a aussi une vue dans une vue, c'est-à-dire qu'on utilise une vue déjà créé comme une table dans la vue, à l'exemple de :

CREATE OR REPLACE VIEW V_lapins_race

AS SELECT id, sexe, date_naissance, nom, commentaires, espece_id, race_id, mere_id, pere_id, disponible

FROM V_lapins

WHERE race_id IS NOT NULL;

La clause « **SELECT** » est figée, dans le sens où après la création d'une vue, sa requête ne peut être modifiée. S'il y'a eu une modification sur la table, nous devons créer une autre vue qui prendra en compte les modifications.

Tri de données dans une vue

Le **tri** peut s'effectuer avec la commande « **ORDER BY** ». Dans la requête définissant une vue, si l'on met un « **ORDER BY** » et que dans la requête ne figure un autre, celui-ci prendra effet. Mais si nous mettons un autre « **ORDER BY** » dans la requête, celui qui prendra effet c'est celui dans la requête.

Modification d'une vue et Suppression d'une vue

Il existe une commande « **ALTER VIEW** » pour modifier la vue si elle existe déjà si non une erreur sera signalée.

ALTER VIEW nom_vue [(liste_colonnes)]

AS requete_selec

Pour la suppression d'une vue c'est simple. La commande « DROP » suivie du nom de la vue :

DROP VIEW V_Race;

b. Notions du déclencheur

Les **triggers** sont des objets de la base de données. Ils sont attachés à une table et se déclenchent à l'exécution d'une instruction, ou d'un bloc d'instructions, lorsqu'une ou plusieurs lignes sont insérées, supprimées ou modifiées dans la table à laquelle ils sont attachés. Les **déclencheurs** s'activent par un évènement. Cet évènement peut être :

Une insertion dans la table avec la requête **INSERT**

La suppression d'une partie des données de la table avec la requête **DELETE**

La modification d'une partie des données de la table avec la requête **UPDATE**

FONCTIONNEMENT

L'exécution d'un déclencheur se fait de façon automatique à la suite d'un évènement. Le déclencheur exécute un traitement pour chaque ligne insérée, modifiée ou supprimée par l'évènement déclencheur. Le déclencheur a la possibilité de modifier, insérer des données dans n'importe quelle table sauf les tables utilisées dans la requête qui l'a déclenché.

Applications des déclencheurs

Les **déclencheurs** sont utiles pour de nombreuses utilisations sur la BDD à savoir :

- La vérification des données
- L'intégrité des données
- Pour l'archivage des données
- Mise à jour d'informations qui dépendent d'autres données

SYNTAXE

Pour créer un trigger :

CREATE TRIGGER nom_trigger moment_trigger evenement_trigger

ON nom_table FOR EACH ROW

Corps_trigger ;

Lorsqu'un trigger est déclenché, es instructions peuvent être exécutées à deux moments différents : soit juste avant que l'évènement déclencheur n'ait lieu (**BEFORE**), soit juste après (**AFTER**).

Le corps du trigger représente le contenu du trigger constitué d'une ou plusieurs instructions. Dans ce corps, MySQL met à disposition les mots clés :

- **OLD** pour représenter les valeurs des colonnes de la ligne traitée avant qu'elle ne soit modifié par l'évènement déclencheur. Ces valeurs peuvent être lues, mais pas modifiés
- **NEW** pour représenter les valeurs des colonnes de la ligne traitée après qu'elle a été modifiée par l'évènement déclencheur.

Suppression des triggers :

DROP TRIGGER nom_trigger ;

Restrictions des triggers :

- Il est impossible de travailler avec des transactions à l'intérieur d'un trigger.
- Les requêtes préparées ne peuvent pas non plus être utilisées
- Il est impossible de modifier les données d'une table utilisée par la requête ayant déclenché le trigger
- Une suppression ou modification de données déclenchée par une clé étrangère ne provoquera pas l'exécution du trigger correspondant

4. Epurer la base de donnée

CREATE VIEW INFO_RECENT AS

SELECT * FROM fromage

WHERE LEFT (fromage.DateObtentionAOC ,4) >2013 ;

GRANT SELECT ON INFO_RECENT TO CURRENT_USER WITH GRANT OPTION ;

VII. VALDATON DES HYPOTHESES

- Le Langage de Contrôle de Données nous permettra d'archiver les données
- L'étude des syntaxes d'automatisation des requêtes nous aidera à restreindre la suppression
- L'utilisation des procédures stockées nous permettra de résoudre le problème
- L'utilisation de la différence nous permettra de résoudre le problème

VIII. REFERENCES DES METHODES UTLSEE

- WampServer
- Jmerise

IX. REFERENCES FOURNT PAR LE PROST

- Fromage.mcdFichier

  Le MCD de la BDD
- P8cheeseshop.sqlFichier
- Ressource 1URL

  Aide en ligne en anglais sur la syntaxe SQL
- Ressource 2URL
- Ressource 3