ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

ΜΕΡΟΣ ΠΕΜΠΤΟ

Views, Triggers

Γιώργος Μαρκομανώλης

Περιεχόμενα

Όψεις (Views)

Η όψη είναι ένας τύπος πίνακα που δεν αποθηκεύει δεδομένα. Λειτουργεί σαν ένα φίλτρο στις στήλες και στις γραμμές των πινάκων που αναφέρεται η όψη. Το ερώτημα που ορίζει την όψη μπορεί να αναφέρεται σε έναν ή σε παραπάνω πίνακες, καθώς και σε άλλη όψη αυτής της βάσης. Μπορείτε να κάνετε ερωτήματα στην όψη σαν να ήταν πραγματικός πίνακας. Οι όψεις δεν μπορούν να έχουν κλειδιά ή indexes. Σαν index λειτουργούν, τα indexes των πινάκων που αναφέρεται η όψη. Μια όψη δεν μπορεί να περιέχει την εντολή **ORDER BY** γιατί δημιουργείται πρόβλημα στην βάση. Μπορούμε να δημιουργήσουμε όψεις με κάθε είδους ερώτημα **SELECT**.

Παραδείγματα

Για να δημιουργήσουμε μια όψη με την οποία θα βλέπουμε τους φοιτητές που πέρασαν στο πανεπιστήμιο από το 1997 και πιο μετά, δίνουμε την εντολή (οι πίνακες είναι από το τεστ 18/1//2004):

```
CREATE VIEW AFTER_1997 AS
SELECT * FROM STUDENTS
WHERE START_DATE>'1.1.1997';
```

Για να δημιουργήσουμε μια όψη για την εύρεση των αριθμών μητρώων των φοιτητών και των ονομάτων των μαθημάτων που έχουν περάσει, δίνουμε την εντολή:

```
CREATE VIEW ID_PASS AS
SELECT STUDENTS.ID, LESSONS.NAME FROM STUDENTS, LESSONS,
GRADES WHERE STUDENTS.ID=GRADES.ID AND
GRADES.CODE=LESSONS.CODE AND GRADE>=5;
```

Χρησιμότητα όψεων:

Με τις όψεις μπορούμε:

- Να έχουμε ανεξαρτησία δεδομένων, δηλαδή αν αποφασίσουμε να χωρίσουμε έναν πίνακα, σε δύο πίνακες, τότε μπορούμε να δημιουργήσουμε μια νέα όψη η οποία θα είναι η ένωση των πινάκων και θα μπορείτε να κάνετε ερωτήματα πάνω στην όψη σαν να είναι ένας πίνακας.

- Να έχουμε ασφάλεια δεδομένων. Με τις όψεις μπορούμε να δίνουμε την δυνατότητα στους χρήστες μόνο τα δεδομένα που θέλουμε. Για παράδειγμα σε μια βάση με υπαλλήλους, μπορούμε να ορίσουμε να μην εμφανίζεται ο μισθός των υπαλλήλων.

Παρατήρηση: Για να δημιουργήσετε όψεις πρέπει να έχετε δικαιώματα στους αντίστοιχους πίνακες που χρησιμοποιείται.

Δημιουργία όψης

Σε μια όψη υποστηρίζονται όλες οι εντολές **JOIN, UNION**. Δεν είναι σωστό να χρησιμοποιείται το **ORDER BY.**

Η σύνταξη της όψης είναι η εξής:

```
CREATE VIEW όνομα_όψης
   [(όνομα_στήλης1_όψης, όνομα_στήλης2_όψης,..).]
AS
  Εντολή SELECT [Με επιλογή CHECK];
```

Όνομα όψης

Το όνομα της όψης πρέπει να είναι μοναδικό σε κάθε βάση και να μην υπάρχει με το ίδιο όνομα όψη, πίνακας ή stored procedure.

Ονόματα στηλών όψης

Ο ορισμός ονομάτων στηλών σε μια όψη είναι προαιρετικός και απαιτείται μόνο αν η όψη περιέχει ενώσεις πινάκων που έχουν ίδιο όνομα σε κάποιες στήλες.

Παράδειγμα

Έστω ότι έχουμε την βάση για την καταχώρηση των φοιτητών, των μαθημάτων και των βαθμών τους με τους ακόλουθους πίνακες:

Όνομα πίνακα: **Students**			
FNAME	LNAME	ID	START_DATE

FNAME = Όνομα φοιτητή
LNAME = Επώνυμο φοιτητή
ID = Αριθμός μητρώου
START_DATE = Ημερομηνία εισαγωγής στην σχολή

Όνομα πίνακα: **Lessons**		
NAME	CODE	CREDITS

NAME = Όνομα μαθήματος
CODE = Κωδικός μαθήματος
CREDITS = Διδακτικές μονάδες

Όνομα πίνακα: **Grades**		
ID	CODE	GRADE

Για την εύρεση των ονομάτων των μαθημάτων από τον πίνακα Lessons που έχουν 3 διδακτικές μονάδες, δημιουργούμε την ακόλουθη όψη:

```
CREATE VIEW lessons_credits_3
AS
SELECT * FROM LESSONS WHERE CREDITS=3;
```

Για την εύρεση των αριθμών μητρώων των φοιτητών και των ονομάτων των μαθημάτων που έχουν περάσει αν θέλουμε να δηλώσουμε τις δύο στήλες που θέλουμε να βρούμε νε τις λέξεις mitroo, onoma, τότε θα δώσουμε την εντολή:

```
CREATE VIEW lessons_credits_3
(mitroo, onoma)
AS
 SELECT STUDENTS.ID, LESSONS.NAME
  FROM STUDENTS, LESSONS, GRADES
  WHERE STUDENTS.ID=GRADES.ID
  AND GRADES.CODE=LESSONS.CODE
AND GRADE>=5;
```

Επιλογή CHECK

Αν ορίσουμε μια όψη έστω για τον πίνακα Students να καταχωρούνται μόνο οι φοιτητές που μπήκαν στο πανεπιστήμιο μετά το 2000, δηλαδή:

```
 CREATE VIEW after_2000
AS
 SELECT * FROM STUDENTS
WHERE START_DATE>='1.1.2000'
WITH CHECK OPTION;
```

Τότε αν προσπαθήσουμε να βάλουμε νέα δεδομένα στην όψη (τα δεδομένα θα καταχωρηθούν στους πίνακες που χρησιμοποιεί η όψη, συγκεκριμένα στο παράδειγμά μας στον πίνακα Students), επειδή έχουμε βάλει το **CHECK**, θα γίνει έλεγχος αν η νέα ημερομηνία ικανοποιεί την συνθήκη **WHERE**, αλλιώς δεν θα καταχωρηθεί.

Όψεις μόνο για εμφάνιση δεδομένων

Μπορούμε να δημιουργήσουμε όψεις, στις οποίες δεν μπορείτε να καταχωρήσετε δεδομένα ή να ενημερώσετε τα δεδομένα των πινάκων που δημιουργούν την όψη. Μια όψη, που δεν ενημερώνει τα στοιχεία των πινάκων από τους οποίους δημιουργείται πρέπει να έχει ένα από τα παρακάτω χαρακτηριστικά:

- Αν υπάρχει περιορισμός στην επιλογή δεδομένων, όπως DISTINCT, FIRST, SKIP.
- Περιέχει γνωρίσματα που δηλώνονται μέσα σε υποερωτήματα στο SELECT.
- Περιέχει γνωρίσματα που δηλώνονται μέσα σε συναρτήσεις ή δηλώνονται με GROUP BY.
- Περιέχει SELECT με UNION.
- Είναι η ένωση πολλών πινάκων
- Δεν περιέχει όλα τα γνωρίσματα του πίνακα που είναι NOT NULL (δηλαδή όταν προσπαθήσουμε να κάνουμε εισαγωγή νέων δεδομένων, τότε θα υπάρχει γνώρισμα NOT NULL και η τιμή που θα θέλουμε να βάλουμε θα είναι NULL και επομένως θα δημιουργηθεί λάθος).
- Περιέχει γνωρίσματα από άλλη όψη που δεν δέχεται ενημέρωση στοιχείων.

Αλλαγή ορισμού όψης

Για να αλλάξετε τα στοιχεία μιας όψης πρέπει να την διαγράψετε και να την δημιουργήσετε ξανά.

Διαγραφή όψης

Η σύνταξη για την διαγραφή της όψης με το όνομα AFTER_2000 είναι η εξής:

DROP VIEW AFTER_2000;

Triggers

Μια trigger είναι μια ρουτίνα που αναφέρεται σε έναν πίνακα ή μια όψη και πραγματοποιεί αυτόματα μια διαδικασία, όταν μια γραμμή προστίθεται, ενημερώνεται ή διαγράφεται. Μια trigger ποτέ δεν μπορούμε να την καλέσουμε κατευθείαν, αλλά όταν μια εφαρμογή ή ένας χρήστης προσπαθεί να εισάγει, ενημερώσει ή διαγράψει μια γραμμή, τότε ενεργοποιείται η trigger εφόσον υπάρχει.

Οι triggers μπορούν να χρησιμοποιηθούν για:

- Να πραγματοποιεί διαδικασίες κατά την επεξεργασία στοιχείων. Για παράδειγμα μπορούμε να βάλουμε μια trigger ώστε όταν γίνεται διαγραφή από την βάση δεδομένα, να αποθηκεύονται σε άλλον πίνακα για ιστορικούς λόγους.

- Να γίνεται έλεγχος στα δεδομένα που θα εισαχθούν.

- Μετατροπή δεδομένων, για παράδειγμα να μετατρέψουμε ένα κείμενο σε κείμενο με κεφαλαία γράμματα.

- Ειδοποίηση εφαρμογών για αλλαγές στην βάση χρησιμοποιώντας γεγονότα.

- Να πραγματοποιηθούν ενημερώσεις σε αναφορικούς περιορισμούς (foreign key).

Οι triggers αποθηκεύονται σαν αντικείμενα στην βάση και είναι ενεργές μέχρι να τις απενεργοποιήσουμε ή σβήσουμε.

Πλεονεκτήματα από την χρησιμοποίηση Triggers:

- Αυτόματη ενεργοποίηση περιορισμών για την εισαγωγή σωστών δεδομένων.

- Μείωση κώδικα στις εφαρμογές για τον έλεγχο των δεδομένων που θα εισαχθούν στην βάση.

- Αυτόματη καταγραφή των αλλαγών που γίνονται σε έναν πίνακα, μέσω ενός log αρχείου.

- Αυτόματη προειδοποίηση για τις αλλαγές στην βάση μέσο γεγονότων στις triggers.

Παρακάτω βλέπουμε έναν πίνακα που συνοψίζει τα είδη των triggers.

Είδος trigger	**Περιγραφή**
BEFORE INSERT	Ξεκινάει πριν την δημιουργία μιας γραμμής και οι τιμές μπορούν να αλλαχτούν.
AFTER INSERT	Ξεκινάει αφού δημιουργηθεί μια νέα γραμμή και δεν μπορούμε να αλλάξουμε τις τιμές. Συνήθως χρησιμοποιείται για να αλλάξουμε άλλους πίνακες.
BEFORE UPDATE	Ξεκινάει πριν δημιουργηθεί μια νέα έκδοση της γραμμής και μπορούμε να αλλάξουμε τις τιμές.
AFTER UPDATE	Ξεκινάει αφού δημιουργηθεί μια νέα έκδοση της γραμμής και δεν μπορούμε να αλλάξουμε τις τιμές της. Συνήθως χρησιμοποιείται για την αλλαγή άλλων πινάκων.
BEFORE DELETE	Ξεκινάει πριν διαγραφή η γραμμή και δεν επιτρέπει αλλαγές σε καμιά στήλη στην γραμμή.
AFTER DELETE	Ξεκινάει αφού διαγραφτεί μια γραμμή και δεν επιτρέπει καμιά αλλαγή στις τιμές. Συνήθως χρησιμοποιείται για την αλλαγή άλλων πινάκων.
BEFORE <γεγονός> OR <γεγονός>	Ξεκινάει πριν πραγματοποιηθεί κάποια αλλαγή στα δεδομένα. Το γεγονός πρέπει να είναι σε μορφή συνθήκης. Δεν μπορείτε να διαγράψετε καμία τιμή από τις στήλες στην γραμμή.
AFTER <γεγονός> OR <γεγονός>	Ξεκινάει αφού πραγματοποιηθεί κάποια αλλαγή στα δεδομένα. Το γεγονός πρέπει να είναι σε μορφή συνθήκης. Δεν μπορείτε να διαγράψετε καμία τιμή από τις στήλες στην γραμμή. Χρησιμοποιείται για αλλαγή άλλων πινάκων.

Σειρά εκτελέσεων των triggers

Έχει επικρατήσει για την ονομασία των triggers, να βάζουμε το πρώτο γράμμα της λέξης **BEFORE** ή **AFTER**, μετά το πρώτο γράμμα της λέξης **INSERT** ή **UPDATE** ή **DELETE** ή συνδυασμό αυτών και μετά μια κάτω παύλα και το όνομα που θέλουμε. Δηλαδή αν θέλουμε να δηλώσουμε μια trigger για πριν την καταχώρηση δεδομένων με σε πίνακα με όνομα TEST, το όνομα θα είναι TEST_BI (B=BEFORE, I=INSERT), αν θέλουμε πριν την καταχώρηση και την ενημέρωση το όνομα αλλάζει σε TEST_BIU. Αν θέλουμε να δημιουργήσουμε trigger για πριν απ'όλα ή μετά απ'όλα, θα γράψουμε το όνομα BA ή AA αντίστοιχα. Πρέπει όμως να δίνουμε και μια σειρά προτεραιότητας στις triggers, αν δηλαδή έχουμε δύο triggers για πριν την καταχώρηση για έναν πίνακα πρέπει να δηλώσουμε στην Firebird, ποια θα εκτελεσθεί πρώτη. Αυτό το πετυχαίνουμε με την δήλωση position ξεκινώντας από το μηδέν.

Δηλαδή αν έχουμε δύο triggers για BEFORE INSERT και η μία έχει δηλωθεί με position 0 και ή άλλη position 1, τότε όταν προσπαθούμε να καταχωρήσουμε μια νέα γραμμή, θα εκτελέσει πρώτα αυτή με position 1 και μετά αυτή με position 2.

Δημιουργία Triggers

Η δημιουργία trigger πραγματοποιείται με το **CREATE TRIGGER** που αποτελείται από αρχικές δηλώσεις για την trigger και το κύριο μέρος της διαδικασίας.

Το μέρος με τις αρχικές δηλώσεις περιέχει:

- Το όνομα της trigger που είναι μοναδικό στην βάση.
- Ένα όνομα πίνακα που δηλώνει τον πίνακα για τον οποίο είναι η trigger.
- Ιδιότητες που δηλώνουν την φάση (BEFORE, AFTER), το γεγονός (INSERT, UPDATE, DELETE) και την σειρά εκτέλεσης αν χρειάζεται.

Το κύριο μέρος της trigger περιέχει:

- Αν χρειάζεται λίστα με τις τοπικές μεταβλητές και τον τύπο δεδομένων τους.
- Τις εντολές BEGIN και END όπου δηλώνουν την έναρξη του κώδικα το τέλος.

Η γενική σύνταξη της trigger είναι η ακόλουθη:

```
CREATE TRIGGER όνομα_trigger FOR {όνομα πίνακα ή όψης}
{ACTIVE | INACTIVE}
{BEFORE | AFTER}
 {DELETE OR {[INSERT [OR UPDATE]} | {INSERT OR [..]} | {UPDATE OR [..]}}
[POSITION αριθμός]
AS <κύριο μέρος διαδικασίας> ^
```

Ειδική σύνταξη της PSQL για triggers

Για το κύριο μέρος της διαδικασίας στις triggers μπορείτε να χρησιμοποιήσετε τις μεταβλητές Boolean INSERTING, UPDATING και DELETING καθώς και τις μεταβλητές NEW και OLD.

Generators

Για να συνεχίσουμε θα αναφέρουμε την δήλωση **Generator**. Οι **Generators** είναι χρήσιμοι για τον υπολογισμό ενός αυτόματου αυξανόμενου μοναδικού κλειδιού.

Δημιουργία Generator

```
CREATE GENERATOR όνομα_generator;
```

Μπορούμε να ορίσουμε εμείς ο generator να ξεκινάει από μια τιμή έστω το 1.

Άρα δηλώνουμε τον generator και έχουμε:

CREATE GENERATOR count2;

SET GENERATOR count2 TO 1;

Συνάρτηση GEN_ID

Με την συνάρτηση **GEN_ID** μπορούμε να βρούμε την τιμή του generator.

Δηλαδή αν δώσουμε την εντολή GEN_ID(όνομα_generator, 1) στην κατάλληλη θέση στο κύριο μέρος της διαδικασία της trigger, θα πάρουμε την τιμή του generator αυξημένη κατά 1.

Δημιουργία trigger για την αυτόματη αύξηση της τιμής μιας στήλης (auto-increment)

Δημιουργούμε τον generator:

CREATE GENERATOR COUNTER;

Δημιουργούμε την trigger:

```
CREATE TRIGGER BI_STUDENTS FOR STUDENTS
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  IF(NEW.NUM IS NULL) THEN
    NEW.NUM=GEN_ID(COUNTER,1);
END^
COMMIT^
```

Μετασχηματισμοί

Αν θέλουμε σε μια στήλη να καταχωρείται συμβολοσειρά με κεφαλαία γράμματα μόνο, τότε δημιουργούμε την ακόλουθη trigger:

```
CREATE TRIGGER STUDENTS1 FOR STUDENTS
ACTIVE BEFORE INSERT OR UPDATE
POSITION 0
AS
BEGIN
NEW.LNAME=UPPER(NEW.LNAME);
END^
```

Default τιμές με triggers

Αν θέλουμε να έχουμε σε μια στήλη **default** τιμή, τότε δημιουργούμε την trigger:

```
CREATE TRIGGER GRADES_BIU FOR GRADES
ACTIVE BEFORE INSERT OR UPDATE
AS
BEGIN
  IF(NEW.GRADE IS NULL) THEN
   NEW.GRADE=0;
END^
```