Όψεις (Views) και Ασφάλεια

Αντώνιος Δεληγιαννάκης

Όψεις (Views) – Τι Είναι

Μία όψη επιτρέπει να εκφράσεις ως πίνακα το αποτέλεσμα ενός SQL ερωτήματος

- Μια όψη μπορεί να χρησιμοποιηθεί ως πίνακας σε άλλα ερωτήματα

```
CREATE VIEW viewName AS
SQL Ερώτημα
```

```
CREATE VIEW viewName(A1,…,An) AS
SQL Ερώτημα
```

Παραδείγματα:

```
CREATE VIEW topStudents AS
SELECT * FROM Student
WHERE gpa > 3.8
```

```
CREATE VIEW csApplicants(id, name) AS
SELECT DISTINCT S.sID, S.sName
FROM Student S, Apply A
WHERE S.sID = A.sID AND major='CS'
```

```
CREATE VIEW topCSApplicants AS
SELECT * FROM csApplicants
WHERE id in
        (SELECT sID from topStudents)
```

```
SELECT *
FROM csApplicants
```

```
DROP VIEW csApplicants
```

Recursive Views – Αναδρομή

Μπορούμε να ορίσουμε μία όψη που θα παρέχει τον ορισμό ενός αναδρομικού ερωτήματος

Μετά μπορούμε στο FROM ερωτημάτων να χρησιμοποιούμε την όψη σαν τους υπόλοιπους πίνακες

```
CREATE RECURSIVE VIEW Anc(a,d)  as (
                    select parent as a, child as d from ParentOf
                    where child='Antonis'
                    union
                    select ParentOf.parent as a, Anc.d as d
                    from Anc, ParentOf
                    where Anc.a = ParentOf.child );
select a from Anc where d = 'Antonis';
```

Εδώ τελειώνει η δήλωση του view

Όψεις (Views) – Γιατί Είναι Χρήσιμες;

- Διευκολύνουν την έκφραση άλλων ερωτημάτων

- Σκεφτείτε ότι το σχήμα μιας βάσης μπορεί να περιέχει >100 πίνακες
  - Χρήστες μπορεί να ενδιαφέρονται (ή θέλουμε) να δούνε μόνο τμήμα της
  - Πχ, το τμήμα εξυπηρέτησης πελατών να βλέπει μόνο στοιχεία πελατών, παράπονα και αναφορές βλαβών

- Οι όψεις επιτρέπουν τον περιορισμό των δεδομένων που βλέπουν κάποιοι χρήστες
  - Για ευκολία ή για λόγους ασφάλειας
  - Μπορεί ένας χρήστης να έχει δικαιώματα σε ένα view, αλλά όχι απευθείας στους πίνακες από τους οποίους υπολογίζεται
  - Δεν είναι το ίδιο με το να δίνεις δικαιώματα σε πίνακες
    - Μπορείς να επιτρέψεις σε ένα χρήστη να βλέπει πληροφορία που **παράγεται από ορισμένες πλειάδες ενός πίνακα**

Virtual Views

Υπάρχουν 2 είδη όψεων με διαφορετικά χαρακτηριστικά

- Virtual views (εικονικές όψεις)
  - Εκφράζονται ως SQL ερώτημα αλλά το αποτέλεσμά τους **ΔΕΝ αποθηκεύεται** ως πίνακας στη βάση
  - Μπορείς όμως να τις χρησιμοποιείς σε ερωτήματα ή για τον ορισμό άλλων views
  - Κατά την εκτέλεση του ερωτήματος, ο optimizer μετασχηματίζει το ερώτημα σε ένα που αποτελείται από τους **πίνακες βάσης (base tables)** της όψης
    - Οι πίνακες βάσης είναι οι πίνακες από τους οποίους τελικά υπολογίζεται η όψη
  - Αν η όψη εκφράζει ένα πολύ σύνθετο ερώτημα με αρκετά JOIN, θα κοστίσει πολύ η χρήση της αντίστοιχης όψης
  - Τα αποτελέσματα της όψης θα είναι όμως πάντα έγκυρα, άσχετα με το αν κάνουμε ενημερώσεις, εισαγωγές ή διαγραφές

Materialized Views

- Materialized Views (υλοποιημένες όψεις):
  - Υπολογίζουν και αποθηκεύουν το αποτέλεσμα τους ως πίνακα στη βάση
  - Αποδοτικά σε ερωτήματα (σε αντίθεση με τα virtual views), αλλά θέματα με ενημερώσεις…
    - Θα είναι σωστά τα δεδομένα τους αν μετά τον υπολογισμό, αλλάξουν τα δεδομένα των πινάκων βάσης;
    - Η απάντηση είναι ΌΧΙ - συνήθως τις ανανεώνουμε (REFRESH MATERIALIZED VIEW view_name) περιοδικά.
  - Πότε να τις προτιμάτε;
    - Σε πολλές εφαρμογές οι ενημερώσεις της βάσης γίνονται μόνο το βράδυ, όταν δεν υπάρχουν πολλοί χρήστες/ερωτήματα
    - Μπορείτε μετά τις ενημερώσεις της βάσης να υπολογίσετε materialized views και να τα χρησιμοποιείται αποδοτικά κατά τη διάρκεια της ημέρας
  - Πότε να μην τις προτιμάτε;
    - Αν μπορεί να έχετε συχνά ενημερώσεις/εισαγωγές/διαγραφές στους πίνακες βάσης παράλληλα με τα ερωτήματα

Παραδείγματα

```
CREATE VIEW csApplicants(id, name) AS
SELECT DISTINCT S.sID, S.sName
FROM Student S, Apply A
WHERE S.sID = A.sID AND major='CS'
```

Οι πίνακες βάσης (base tables) είναι οι Student και Apply

```
CREATE VIEW strangeView AS
SELECT * FROM csApplicants, College
WHERE College.cName = csApplicants.name
```

Οι πίνακες βάσης (base tables) είναι οι Student, Apply και College

Παραδείγματα Materialized Views

```
CREATE MATERIALIZED VIEW csApplicants(id, name) AS
SELECT DISTINCT S.sID, S.sName
FROM Student S, Apply A
WHERE S.sID = A.sID AND major='CS'
```

Η σύνταξη είναι η ίδια με τα virtual views, με τη διαφορά της λέξης κλειδί MATERIALIZED

```
CREATE MATERIALIZED VIEW strangeView AS
SELECT * FROM csApplicants, College
WHERE College.cName = csApplicants.name
```

Views – Μπορούν να Ενημερωθούν;

- Κάποιοι χρήστες μπορούν να δούνε ίσως μόνο Views
- Δε θα είχε νόημα να μπορούν να κάνουν insert, delete, updates σε κάποια από αυτά τα views;
  - Μα τα virtual views ΔΕΝ ΑΠΟΘΗΚΕΥΟΝΤΑΙ
  - Μπορούν ίσως ενημερώσεις σε views να μεταφραστούν σε ενημερώσεις στα base tables;

Γιατί η διαφορά αυτή;

Π.χ.:

```
CREATE VIEW topStudents AS
SELECT * FROM Student
WHERE gpa > 3.8
```

Μπορεί κάποιος να κάνει insert στην topStudents και αυτό να μετατραπεί αυτόματα σε insert στην Student;

Ναι

```
CREATE VIEW csApplicants AS
SELECT DISTINCT S.sID, S.sName
FROM Student S, Apply A
WHERE S.sID = A.sID AND major='CS'
```

Μπορεί κάποιος να κάνει insert στην csApplicants και αυτό να μετατραπεί αυτόματα σε insert σε Student και Apply;

Όχι

Views – Μπορούν να Ενημερωθούν;

Τι θα δούμε…

- Γιατί ΔΕΝ είναι απλό το να επιτρέπουμε μεταβολές δεδομένων σε VIEWS
  - Δε θα υπάρχει "σωστός" ή μοναδικός τρόπος να μεταφράσουμε τη μεταβολή του View σε μεταβολές στους πίνακες βάσης
- Updatable VIEWS: views που μπορούν να ενημερωθούν αυτόματα (σε insert, delete, update)
- Ενημέρωση (insert, delete, update) σε views μέσω triggers

Views – Δυσκολίες σε Ενημερώσεις

```
CREATE VIEW topStudents AS
SELECT sID, sName, gpa FROM Student WHERE gpa > 3.8
```

Αν θέλουμε να κάνουμε εισαγωγή το (5, 'Jim', 3.9) στο topStudents, τι hs θα εισάγουμε στη Student; Άπειρες εναλλακτικές

Αν κάποιος πάει να κάνει εισαγωγή το (6, 'George', 3.7) παραβιάζει τη λογική του view

```
CREATE VIEW topStudents2 AS
SELECT sID, sName, gpa FROM Student WHERE gpa > 3.8
WITH CHECK OPTION
```

Ελέγχει ότι η μετάφραση στα base tables ενημέρωσε σωστά την όψη

```
insert into topStudents2 values(6,  'George', 3.7);          --- fails
```

Views – Δυσκολίες σε Ενημερώσεις

```
CREATE VIEW avgGPA AS
SELECT avg(gpa) FROM Student WHERE gpa > 3.8
```

Τι νόημα έχει να αλλάξουμε το μέσο όρο στο avgGPA πχ σε 3.9;

Σε τι αλλαγές είναι σωστό να μεταφραστεί στον πίνακα Student;

Καμία απάντηση…

Τα πράγματα είναι ακόμα πιο θολά όταν έχουμε joins, groups, subqueries κτλ…

Updatable Views
Αυτόματα Ενημερώσιμες Όψεις

Στην SQL2, ένα VIEW μπορεί να ενημερωθεί αυτόματα αν:

- Υπάρχει μόνο 1 πίνακας στο FROM, χωρίς JOIN
- Ο πίνακας του FROM δεν υπάρχει σε subquery
- Δεν υπάρχει SELECT DISTINCT
- Δεν υπάρχει GROUP BY ή aggregates στο SELECT
- Τα attributes του πίνακα που ΔΕΝ επιλέγονται στο SELECT πρέπει να επιτρέπουν NULL ή να έχουν ορισμένες default τιμές
  - Γιατί; Σε insert, ό,τι δεν περιλαμβάνεται στο SELECT θα λάβει default τιμή (αν έχει default τιμή), αλλιώς NULL

Και σε non Updatable Views;;;

Μπορούμε σε όψεις που δεν είναι αυτόματα ενημερώσιμες να επιτρέψουμε insert, delete, update με INSTEAD OF triggers

```
CREATE TRIGGER propagate_modifications
INSTEAD OF INSERT OR DELETE ON csApplicants
FOR EACH ROW
EXECUTE PROCEDURE push_modifications();
```

Σημείωση: Το ακόλουθο παράδειγμα υποθέτει ότι στην Apply τα cName και dec μπορούν να είναι NULL

Και σε non Updatable Views;;;

```
CREATE OR REPLACE FUNCTION push_modifications()
  RETURNS trigger AS $BODY$
BEGIN
IF (TG_OP = 'DELETE') THEN
  DELETE FROM Apply     WHERE sID = OLD.id;
  DELETE FROM Student WHERE sID = OLD.id;
  RETURN OLD;
END IF;
IF (TG_OP = 'INSERT') THEN
  INSERT INTO Student values(NEW.id, NEW.name, null, null);
  INSERT INTO Apply     values(NEW.id, null, 'CS', null);
  RETURN NEW;
END IF;
END;
$BODY$
LANGUAGE plpgsql
```

Παραδείγματα

update csApplicants set id = 10 where id < 5;

Παίρνουμε σφάλμα γιατί δεν είναι updatable και δεν έχουμε ορίσει INSTEAD OF trigger για το update

insert into csApplicants values(24, 'Dad');

Προσθέτει το (24, ' Dad', null, null) στη Student

Προσθέτει το (24, null, 'CS', null) στην Apply

delete from csApplicants where id=24;

Διαγράφει τα 2 tuples από Student, Apply

Δικαιώματα Χρηστών/Ασφάλεια

Δικαιώματα σε Χρήστες

- Οι πίνακες/όψεις ανήκουν σε αυτόν που τους δημιούργησε
- Ο ιδιοκτήτης μπορεί να αναθέσεις δικαιώματα (όποια θέλει) σε άλλους χρήστες πάνω σε πίνακες και όψεις

- SELECT ή SELECT(A1, …, An)

  Σκεφτείτε το σαν read όλων των tuples ή μόνο ορισμένων attributes σε όλα τα tuples
- INSERT ή INSERT(A1, …, An)
- UPDATE ή UPDATE(A1, …, An)
- DELETE

Πιο συνηθισμένο είναι να μην περιορίζουμε τα δικαιώματα σε συγκεκριμένα attributes

Παράδειγμα

```
DELETE FROM STUDENT
WHERE sID in (SELECT APPLY.sID FROM APPLY
                    WHERE APPLY.sID= STUDENT.sID AND
                          cName = 'TUC')
```

Τι δικαιώματα χρειάζεται ένας χρήστης;

STUDENT: DELETE και SELECT (για να διαβάσει το sID)

APPLY: SELECT

Παράδειγμα

Πώς επιτρέπουμε σε κάποιον να βλέπει δεδομένα μόνο φοιτητών που έχουν κάνει αίτηση στο TUC;

Απάντηση: Όταν θέλουμε υποσύνολο tuples μιας σχέσης, οι όψεις είναι λύση

```
CREATE VIEW tucStudents AS
SELECT * FROM STUDENT
WHERE sID in (SELECT APPLY.sID FROM APPLY
                    WHERE APPLY.sID= STUDENT.sID AND
                          cName = 'TUC')
```

Μετά δίνουμε SELECT δικαιώματα στην tucStudents

Παροχή Δικαιωμάτων: GRANT

GRANT privileges on R TO users

GRANT ALL PRIVILEGES on R to adeli; -- όχι διαθέσιμο παντού

GRANT SELECT, INSERT, UPDATE, DELETE
ON Student TO adeli;

GRANT SELECT ON College TO public; --- δικαίωμα σε όλους

Αν δώσουμε ένα δικαίωμα σε ένα χρήστη, μπορεί αυτός με τη σειρά του να το δώσει σε άλλους; Επιλογή WITH GRANT OPTION

GRANT SELECT, INSERT, UPDATE, DELETE
ON Student TO adeli
WITH GRANT OPTION;

Ανάκληση Δικαιωμάτων: REVOKE

REVOKE privileges on R FROM users
[CASCADE | RESRICT]

REVOKE DELETE ON Student FROM adeli;

Η παραπάνω εντολή θα αποτύχει αν ο adeli είχε μεταδώσει το δικαίωμα σε άλλον

REVOKE DELETE ON Student FROM adeli CASCADE;

Η παραπάνω εντολή θα αφαιρέσει το δικαίωμα DELETE από τον adeli ΚΑΙ από όσους το έχει μεταδώσει αν ΔΕΝ το έχουν λάβει το ίδιο δικαίωμα και από κάπου αλλού

Ίσως Χρειαστείτε στην PostgreSQL

grant all privileges on database dbname to adeli;

Δίνει δικαιώματα CREATE, CONNECT

GRANT ALL PRIVILEGES ON ALL TABLES
IN SCHEMA public
TO adeli;

Δίνει SELECT, INSERT, DELETE, UPDATE και πολλά άλλα, όπως EXECUTE σε FUNCTIONS…

Τα παραπάνω επηρεάζουν πίνακες που ΗΔΗ ΥΠΑΡΧΟΥΝ

Για μελλοντικούς πίνακες…

ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT SELECT ON TABLES TO adeli;