Προηγμένα Θέματα Βάσεων Δεδομένων

Διδάσκων: Άγγελος Μιχάλας

Περιεχόμενα

- Views
- Indexes

Εισαγωγή

- Μία όψη (view) είναι **ένας εικονικός πίνακας βασισμένος στο αποτέλεσμα ενός SQL statement**.
  - Ορίζεται ως μία SQL SELECT επερώτηση με συνδέσεις (joins)
- Μία όψη περιέχει γραμμές και στήλες όπως ένας πραγματικός πίνακας. Τα πεδία μιας όψης είναι ουσιαστικά τα πεδία ενός ή περισσοτέρων πινάκων της ΒΔ.
- Μία όψη μπορεί να έχει SQL functions, WHERE, και JOIN statements και να παρουσιάζει τα δεδομένα σαν να προέρχονται από έναν μοναδικό πίνακα.

Εισαγωγή

- Τα περισσότερα ΣΔΒΔ επιτρέπουν την **ενημέρωση των δεδομένων** που βρίσκονται στους πραγματικούς πίνακες **μέσω μιας όψης**.
  - Πρέπει να ισχύουν κάποιες προϋποθέσεις
- Μία όψη είναι δυναμική γιατί δεν σχετίζεται με το φυσικό σχήμα.
- Η ΒΔ αποθηκεύει τις όψεις ως SQL SELECT statement με συνδέσεις (joins). Όταν τα δεδομένα των πινάκων αλλάζουν, τότε η όψη αντανακλά αυτή την αλλαγή.

Πλεονεκτήματα

- Μία όψη επιτρέπει την απλοποίηση πολύπλοκων ερωτημάτων.
  - Χρήση των όψεων για απόκρυψη της πολυπλοκότητας των υποκείμενων πινάκων από τους τελικούς χρήστες και τις εφαρμογές.
- Η όψη βοηθά τον περιορισμό στην πρόσβαση των δεδομένων σε συγκεκριμένους χρήστες
  - Δεν επιθυμούμε ευαίσθητα δεδομένα να μπορούν να εμφανιστούν σε όλους τους χρήστες

Πλεονεκτήματα

- Η όψη παρέχει ένα επιπλέον επίπεδο ασφάλειας
  - Η ΒΔ επιτρέπει τη δημιουργία read-only view η οποία εκθέτει μόνο read-only δεδομένα σε συγκεκριμένους χρήστες
- Η όψη επιτρέπει την ύπαρξη υπολογισμένων στηλών
  - Ένας πίνακας της ΒΔ δεν μπορεί να έχει υπολογισμένες στήλες σε αντίθεση με μια όψη (π.χ. Άθροισμα παραγγελιών ενός πελάτη).

Μειονεκτήματα

- Απόδοση
  - Η επερώτηση δεδομένων από μια όψη μπορεί να είναι πολύ αργή ιδιαίτερα εάν η όψη δημιουργείται με βάση άλλες όψεις
- Εξαρτήσεις πινάκων
  - Η όψη δημιουργείται με βάση τους υποκείμενους πίνακες της ΒΔ. Όταν μεταβάλλεται η δομή αυτών των πινάκων τότε θα πρέπει να μεταβληθεί και η όψη.

Επεξεργασία SQL query και View

Δύο Τρόποι

- Δημιουργία ενός **προσωρινού πίνακα** με βάση τον ορισμό της όψης και εκτέλεση **του query που αναφέρεται στην όψη** σε αυτόν τον προσωρινό πίνακα.
- **Συνδυασμός** του **query που αναφέρεται στην όψη** με το **query που ορίζεται στην όψη** και στη συνέχεια εκτέλεση του συνδυασμένου query.

Δημιουργία View

- Σύνταξη

CREATE VIEW view_name **AS**

SELECT column1, column2, ...

FROM table_name

WHERE condition;

Δημιουργία View

- Στη MySQL μπορεί να καθοριστεί και ο τρόπος επεξεργασίας του εισερχόμενου query και του view.

CREATE **[ALGORITHM = {MERGE | TEMPTABLE | UNDEFINED}]**

- Αλγόριθμος MERGE
  - Συνδυασμός του εισερχόμενου query με το SELECT statement, που ορίζεται στην όψη, σε ένα μοναδικό query
  - Δεν επιτρέπεται εάν το SELECT statement περιέχει aggregate functions όπως οι MIN, MAX, SUM, COUNT, AVG ή DISTINCT, GROUP BY, HAVING, LIMIT, UNION, UNION ALL, subquery.
    - Εάν δεν μπορεί να εφαρμοστεί, τότε χρησιμοποιείτε ο αλγόριθμος Undefined

Δημιουργία View

- Αλγόριθμος TEMPTABLE
  - Δημιουργία ενός **προσωρινού πίνακα** με βάση το SELECT statement που ορίζεται στην όψη, και μετά εκτέλεση του εισερχόμενου query πάνω στον προσωρινό πίνακα.
- UNDEFINED είναι ο προκαθορισμένος αλγόριθμος
  - Η MySQL μπορεί να επιλέξει μεταξύ του **MERGE ή TEMPTABLE** αλγορίθμου. Λόγω απόδοσης προτιμάται ο MERGE αλγόριθμος

❖ *Μία απλή όψη μπορεί να είναι updatable*

❖ *Μία όψη μπορεί να βασίζεται σε μία άλλη όψη*

Περιορισμοί

1. Δεν μπορεί να δημιουργηθεί ένα ευρετήριο σε μια όψη
   - Χρήση των ευρετηρίων των υποκείμενων πινάκων όταν τίθεται ένα query προς μία όψη η οποία χρησιμοποιεί τον αλγόριθμο merge
2. SELECT statement
   - **Μπορεί να περιέχει ένα subquery στον όρο WHERE** αλλά όχι στον όρο FROM (για την MySQL αυτό ίσχυε μέχρι την έκδοση 5.7.7)
   - Δεν μπορεί να αναφέρεται σε μεταβλητές όπως τοπικές και session
3. Μία όψη ακυρώνεται όταν καταργείται ή μετονομάζεται ένας πίνακας
4. Η MySQL δεν υποστηρίζει τις materialized όψεις
   - Όψεις που αποθηκεύουν δεδομένα στο φυσικό επίπεδο

Παράδειγμα 1

- **View**

```
CREATE VIEW amountmoneyperordernum AS
SELECT orderNumber, SUM(quantityOrdered *
    priceEach) total
FROM orderDetails
GROUP BY orderNumber
ORDER BY total DESC;
```

- **Προβολή πινάκων και views**

```
show full tables;
```

Παράδειγμα 2

- **View με χρήση Join**

```
CREATE VIEW amountmoneypercustomerorder AS
SELECT d.orderNumber, customerName, SUM
   (quantityOrdered * priceEach) AS total
FROM orderDetails d INNER JOIN orders o ON
   o.orderNumber = d.orderNumber INNER JOIN
   customers c ON o.customerNumber = c.customerNumber
GROUP BY d.orderNumber
ORDER BY total DESC;
```

Παράδειγμα 3

- **View βασισμένο σε άλλο View**

```
CREATE VIEW bigamountmoneyperordernum AS
SELECT orderNumber, total
FROM amountmoneyperordernum
WHERE total > 40000;
```

Παράδειγμα 4

- **View με χρήση sub-query**

```
CREATE VIEW amountmoneyperorderaboveavg AS
SELECT orderNumber, total
FROM amountmoneyperordernum
WHERE  (total > (
   SELECT AVG(amountmoneyperordernum.total)
   FROM amountmoneyperordernum))
```

Διαχείριση των Views

- Χρήση του **CREATE OR REPLACE VIEW** statement για τη δημιουργία μιας όψης ή τη μεταβολή μίας υπαρκτής όψης.
- Αλλαγή ενός view με την εντολή ALTER

  ALTER VIEW [database_name]. [view_name] **AS**

  SELECT column1, column2, ...

  FROM table_name

  WHERE condition;

- Κατάργηση

  DROP VIEW [IF EXISTS] [database_name].[view_name];

- Εμφάνιση

  SHOW CREATE VIEW [database_name].[view_ name];

Updatable View

- Χρήση των INSERT, UPDATE και DELETE statements για την εισαγωγή, ενημέρωση και διαγραφή εγγραφών των πινάκων μέσω ενός updatable view.
- Μία όψη βασισμένη στον αλγόριθμο TEMPTABLE δεν μπορεί να είναι updatable.
- Τα views που βασίζονται σε πολύπλοκα queries δεν είναι updatable.
- Προβολή των views που είναι updatable

```
SELECT table_name, is_updatable
FROM information_schema.views
WHERE table_schema = 'όνομα της Β.Δ.';
```

Updatable View

- Το SELECT statement **δεν πρέπει** να περιέχει:
  - Aggregate functions όπως οι **MIN**, **MAX**, **SUM**, **AVG**, και η **COUNT**.
  - **DISTINCT**
  - Τους όρους **GROUP BY**, **HAVING**.
  - Τους όρους **UNION** ή **UNION ALL**.
  - **Left join** ή **outer join**.
  - **Subquery** στον όρο **SELECT** ή στον όρο **WHERE** που **αναφέρεται** σε έναν **πίνακα που εμφανίζεται στον όρο FROM**.
  - **Αναφορά σε μία non-updatable όψη** στον όρο FROM.

Ορισμένες φορές είναι πιθανή η δημιουργία μιας updatable όψης με βάση πολλούς πίνακες με τη χρήση inner join.

Παράδειγμα

- **Updatable View**

```
CREATE VIEW customerinfo AS
SELECT customerNumber, customerName,
contactLastName, contactFirstName
FROM customers
WHERE country = "USA"
```

Παράδειγμα

- **Updatable View**

```
SELECT *
FROM customerinfo;
```

customerNumber	customerName	contactLastName	contactFirstName
112	Signal Gift Stores	King	John
124	Mini Gifts Distributors Ltd.	Nelson	Susan
129	Mini Wheels Co.	Murphy	Julie
131	Land of Toys Inc.	Lee	Kwai
151	Muscle Machine Inc	Young	Jeff

Παράδειγμα

- **Updatable View**

```
UPDATE customerinfo
SET contactLastName = "Kurt"
WHERE customerNumber = 112;
```

customerNumber	customerName	contactLastName	contactFirstName
112	Signal Gift Stores	Kurt	John
124	Mini Gifts Distributors Ltd.	Nelson	Susan
129	Mini Wheels Co.	Murphy	Julie
131	Land of Toys Inc.	Lee	Kwai
151	Muscle Machine Inc	Young	Jeff

Θέματα Συνέπειας

- Μία απλή όψη είναι updatable και επομένως είναι δυνατόν να ενημερωθούν δεδομένα τα οποία **δεν είναι ορατά μέσω της όψης → Ασυνέπεια**
- Χρήση του όρου **WITH CHECK OPTION** (προαιρετικός όρος) κατά τη δημιουργία ή μεταβολή ενός view
  - Ο όρος WITH CHECK OPTION εμποδίζει την ενημέρωση ή εισαγωγή εγγραφών που δεν είναι ορατές μέσω της όψης.
- Σύνταξη

```
CREATE VIEW view_name  AS
select_statement
WITH CHECK OPTION;
```

Παράδειγμα

- **Συνέπεια**

```
CREATE VIEW customerinfo2 AS
SELECT customerNumber, customerName, country, city
FROM customers
WHERE country = 'USA' AND city LIKE '%San%'
WITH CASCADED CHECK OPTION;
```

customerNumber	customerName	city
124	Mini Gifts Distributors Ltd.	San Rafael
129	Mini Wheels Co.	San Francisco
239	Collectable Mini Designs Co.	San Diego
321	Corporate Gift Ideas Co.	San Francisco
450	The Sharp Gifts Warehouse	San Jose

Παράδειγμα

- **Συνέπεια**

```
insert into customerinfo2(customerNumber, customerName,
    country, city) values(1001, 'Test','USA', 'NY');
```

```
Error Code: 1369. CHECK OPTION failed
    'customerproducts.customerinfo2'
```

Θέματα Συνέπειας

- Όταν δημιουργείτε μία όψη με τον όρο WITH CHECK OPTION, τότε η ΒΔ (π.χ. MySQL) ελέγχει κάθε εγγραφή που αλλάζει μέσω της όψης π.χ., εισαγωγή, ενημέρωση, διαγραφή, έτσι ώστε να είναι σύμφωνη με τον ορισμό της όψης.
- Η MySQL επιτρέπει τη δημιουργία μίας όψης με βάση μία άλλη όψη. Στην περίπτωση αυτή, ελέγχει τους κανόνες στις εξαρτόμενες όψεις για συνέπεια.
- Η MySQL παρέχει δύο επιλογές για τον καθορισμό του CHECK OPTION: LOCAL και CASCADED.
  - CASCADED είναι η προκαθορισμένη επιλογή.

Παράδειγμα

- Δημιουργία πίνακα test1 και test1_view1. Μετά εισαγωγή μίας εγγραφής με τιμή 3 αν και το view έχει συνθηκή > 5.

```
CREATE TABLE test1 (
   test1_id INT
);

CREATE VIEW test1_view1 AS
SELECT test1_id
FROM test1
WHERE test1_id > 5;

insert into test1_view1(test1_id) values(3); -- Success
```

Παράδειγμα

- Δημιουργία test1_view2. Μετά εισαγωγή μίας εγγραφής με τιμή 3 αν και το view έχει συνθηκή > 5.

```
CREATE VIEW test1_view2 AS
SELECT test1_id
FROM test1_view1
WITH CASCADED CHECK OPTION

insert into test1_view2(test1_id) values(3); -- Failed
Error Code: 1369. CHECK OPTION failed
   'customerproducts.test1_view2'
```

Θέματα Συνέπειας

- Αντικατάσταση στο view 2 του WITH CASCADED CHECK OPTION με WITH LOCAL CHECK OPTION
- Σε αυτή την περίπτωση το insert επιτυχάνει επειδή το view 2 δεν έχει κάποιο κανόνα ενώ το view 1 δεν καθορίζει κάποιο check option

Περιεχόμενα

- Views
- Indexes

Εισαγωγή

- Χρήση ευρετηρίων για τη γρήγορη εύρεση εγγραφών με συγκεκριμένες τιμές πεδίων. Χωρίς ευρετήρια, η ΒΔ πρέπει να σαρώσει ολόκληρο τον πίνακα για την εύρεση των σχετικών εγγραφών.
- Ένα ευρετήριο είναι μια δομή δεδομένων όπως τα B-Δένδρα η οποία βελτιώνει την ταχύτητα ανάκτησης των δεδομένων ενός πίνακα.
- Ο query optimizer μπορεί να χρησιμοποιήσει ευρετήρια για την γρήγορη εύρεση των δεδομένων.

Εισαγωγή

- Όταν δημιουργείτε ένα πίνακα με ένα πρωτεύον κλειδί ή ένα μοναδικό πεδίο (unique key), η ΒΔ αυτόματα δημιουργεί ένα ειδικό ευρετήριο που ονομάζεται πρωτεύον.
  - Το πρωτεύον ευρετήριο επιβάλλει τη σειρά των γραμμών στον πίνακα.
- Όλα τα ευρετήρια εκτός από το πρωτεύον ονομάζονται δευτερεύοντα

Δημιουργία Index

- Σύνταξη

  CREATE INDEX index_name

  ON table_name (column_list)

- Εμφάνιση

  SHOW INDEXES FROM [database_name].[table_name];

Παράδειγμα

- Χωρίς ευρετήριο για το πεδίο city του πίνακα customers.

```
EXPLAIN
SELECT *
FROM customers
where city='San Rafael';
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	customers	NULL	ALL	NULL	NULL	NULL	NULL	123	10.00	Using where

Παράδειγμα

- Δημιουργία index για το πεδίο city του πίνακα customers.

```
CREATE INDEX city_index on customers(city);
```

- Εφαρμογή του ίδιου query με πριν

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	customers	NULL	ref	city_index	city_index	52	const	2	100.00	NULL

Κατάργηση Index

- Σύνταξη

  DROP INDEX index_name **ON** table_name

- Παράδειγμα

  DROP INDEX city_index **ON** customers**;**

Αόρατο Index

- Τα αόρατα ευρετήρια επιτρέπουν την επισήμανση ευρετηρίων ως μη διαθέσιμα για τον query optimizer.
- Η MySQL διατηρεί τα αόρατα ευρετήρια και τα κρατά ενήμερα.
  - Τα ευρετήρια είναι ορατά (από προεπιλογή)
  - Το ευρετήριο του πρωτεύοντος κλειδιού δεν μπορεί να είναι αόρατο.

Αόρατο Index

- Σύνταξη

  CREATE INDEX index_name

  ON table_name(c1, c2, ...) **INVISIBLE;**

Παράδειγμα

- Δημιουργία πίνακα.

```
CREATE TABLE test_index (
    a INT,
    b INT
);
```

- Δημιουργία index

```
CREATE INDEX b_index on test_index(b);
```

Παράδειγμα

- Εισαγωγή τυχαίων τιμών στον πίνακα test_index

```
CREATE PROCEDURE  insertRandomData
(IN rowCount INT, IN low INT, IN high INT)
BEGIN
  DECLARE counter INT DEFAULT 0;
  REPEAT
    SET counter := counter + 1;
    INSERT INTO test_index(a,b) VALUES(
        ROUND((RAND() * (high-low))+high),
   ROUND((RAND() * (high-low))+high));
   UNTIL counter >= rowCount
   END REPEAT;
END
```

Παράδειγμα

- Εισαγωγή εγγραφών

```
CALL insertRandomData(100000,1,100000);
```

- Χρόνος εισαγωγής εγγραφών: 250.907 sec

- Χρόνος εκτέλεσης ερωτημάτων ???

Use Index

- Η MySQL παρέχει έναν εναλλακτικό τρόπο που επιτρέπει να προτείνετε στον query optimizer τα ευρετήρια που θα χρησιμοποιήσει χρησιμοποιώντας τον όρο USE INDEX.
- Σύνταξη

```
SELECT select_list
FROM table_name USE INDEX(index_list)
WHERE condition;
```