Darbas su duomenimis

12. Transakcijos

13. Indeksai ir constrains

14. Views

Transakcijos

Visos arba nei viena

Tarkim jūs darote ataskaitą, kurioje turite pateikti šiandienos top 10 filmų. Kol jūs kuriate ataskaitą

- Klientas išsinuomoja filmą
- Klientas pavėlavęs gražina filma
- Nauji 5 filmai yra įtraukiami į inventorių

Kaip tai įtakos ataskaitą?

Locking - užrakinimas

Dvi strategijos

1. Norint įrašyti duomenis reikia gauti rašymo spyną (write lock), norint skaityti duomenis reikia gauti read lock. Write lock yra tik viena ir jeigu jinai yra išduota, read lock negali būti išduodamos. Skaitymo operaciją galima daryti lygiagrečiai, todėl kelios read lock gali būti išduotos.
2. Norint įrašyti duomenis reikia gauti write lock. Norint skaityti nereikia jokios spynos. Duomenų bazės serveris užtikrina, kad nuo operacijos pradžios iki jos pabaigos duomenys bus užfiksuoti ir vartotojui nesikeis. Tokia strategija vadinama – versijavimo.

Ką galima užrakinti?

- Table lock
- Page lock – page (puslapis) yra atminties segmentas nuo 2 iki 16KB. Page priklauso lentelei. T.y. Užrakinam dalį lentelės.
- Row lock – užrakinama eilutė.

Kas yra transakcija?

Transakcija, tai sql komandų grupė, kurios visos komandos turi įvykti, kad būtų galima ją užskaityti. Jeigu bent viena komanda neįvyksta, transkacijos neužskaitome. Ši savybė vadinama atomicity (atomiškumas).

```
START TRANSACTION;

 /* withdraw money from first account, making sure balance is suffici
UPDATE account SET avail_balance = avail_balance - 500
WHERE account_id = 9988
  AND avail_balance > 500;

IF <exactly one row was updated by the previous statement> THEN
  /* deposit money into second account */
  UPDATE account SET avail_balance = avail_balance + 500
    WHERE account_id = 9989;

  IF <exactly one row was updated by the previous statement> THEN
    /* everything worked, make the changes permanent */
    COMMIT;
  ELSE
    /* something went wrong, undo all changes in this transaction */
    ROLLBACK;
  END IF;
ELSE
  /* insufficient funds, or error encountered during update */
  ROLLBACK;
END IF;
```

Transakcija

https://dev.mysql.com/doc/refman/8.0/en/commit.html

START TRANSACTION;

Query 1;

Query 2;

...

Query N;

COMMIT;

Užduotis

Generate a unit of work to transfer $50 from account 123 to account 789. You will need to insert two rows into the `transaction` table and update two rows in the `account` table. Use the following table definitions/data:

```
                    Account:
account_id      avail_balance   last_activity_date
----------      -------------   ------------------
123             500             2019-07-10 20:53:27
789             75              2019-06-22 15:18:35

                    Transaction:
txn_id          txn_date        account_id      txn_type_cd     amount
---------       ------------    -----------     -----------     ------
1001            2019-05-15      123             C               500
1002            2019-06-01      789             C               75
```

Use `txn_type_cd = 'C'` to indicate a credit (addition), and use `txn_type_cd = 'D'` to indicate a debit (subtraction).

Indeksai

Table scan

Help comes from indexes

```
ALTER TABLE customer ADD INDEX idx_email (email);

SHOW INDEX FROM customer \G;

ALTER TABLE customer DROP INDEX idx_email;

ALTER TABLE customer ADD UNIQUE idx_email (email);

ALTER TABLE customer ADD INDEX idx_full_name (last_name, first_name);
```

Indeksų tipai

Balanced-tree indexes

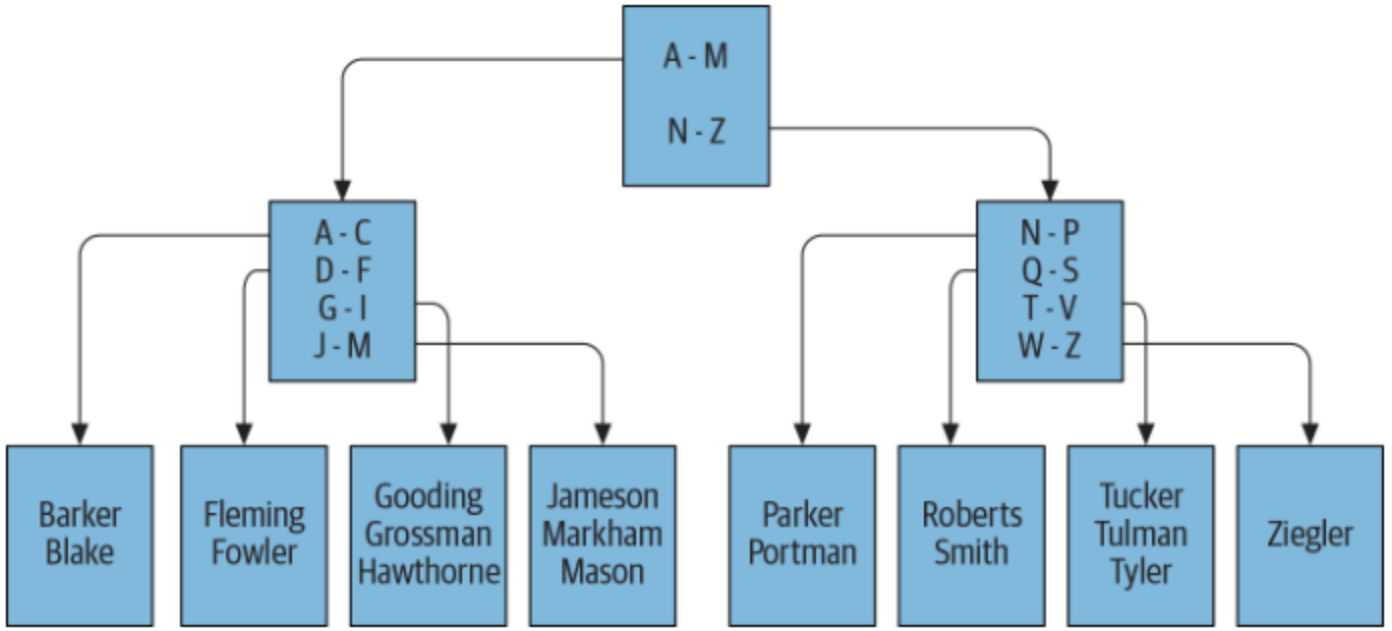

Figure 13-1. B-tree example

Indeksų tipai

- Bitmap, kai turime mažai reikšmių ir daug eilučių (active, inactive).
- Text indexes (teksto indeksai) –

```
SELECT customer_id, first_name, last_name
FROM customer
WHERE first_name LIKE 'S%' AND last_name LIKE 'P%‘;
```

1. Skenuoti lentelę
2. Naudoti last_name stulpelio indeksą
3. Naudoti last_name ir first_name stulpelių indeksą

Kaip patikrinti, ką daro duomenų bazės serveris?

```
EXPLAIN
SELECT customer_id, first_name, last_name
FROM customer
WHERE first_name LIKE 'S%' AND last_name LIKE 'P%' \G;
```

Kiek rekia turėti indeksų?

1. Visi primary key stulpeliai
    1. Jegu primary key sudaro keli stulpeliai, galima sukurti atskirus indeksus kiekvienam stulpeliui
2. Visi foreign key constrains stulpeliai
3. Indeksuoti stulpelius, kurie dažnai bus naudojami dumenų paieškai, pvz. Datų stulpeliai

Constrains (apribojimai)

Constrain – apribojimas taikomas vienam ar keliems lentelės stulpeliams.

Primary key constrains – garantuoja lentelės eilučių unikalumą

Foreign key constrains – galimos tiks tos reikšmės, kurios yra kitos lentelės primary key

Unique constrains – visos reikšmės turi būti unikalios

Check constrains – apriboti stulpelio leidžiamas reikšmes

Constrains kūrimas

```
ALTER TABLE customer
ADD CONSTRAINT fk_customer_address FOREIGN KEY (address_id)
REFERENCES address (address_id) ON DELETE RESTRICT ON UPDATE CASCADE;

ALTER TABLE customer
ADD CONSTRAINT fk_customer_store FOREIGN KEY (store_id)
REFERENCES store (store_id) ON DELETE RESTRICT ON UPDATE CASCADE;

ALTER TABLE table_name
ADD CONSTRAINT constraint_name
UNIQUE (column_list);
```

Kokios gali būti strategijos?

- on delete restrict
- on delete cascade
- on delete set null
- on update restrict
- on update cascade
- on update set null

Užduotys

Exercise 13-1

Generate an `alter table` statement for the `rental` table so that an error will be raised if a row having a value found in the `rental.customer_id` column is deleted from the `customer` table.

Exercise 13-2

Generate a multicolumn index on the `payment` table that could be used by both of the following queries:

```
SELECT customer_id, payment_date, amount
FROM payment
WHERE payment_date > cast('2019-12-31 23:59:59' as datetime);

SELECT customer_id, payment_date, amount
FROM payment
WHERE payment_date > cast('2019-12-31 23:59:59' as datetime)
  AND amount < 5;
```

Views

```
CREATE VIEW customer_vw
 (customer_id,
  first_name,
  last_name,
  email
 )
AS
SELECT
  customer_id,
  first_name,
  last_name,
  concat(substr(email,1,2), '*****', substr(email, -4)) email
FROM customer;

 SELECT first_name, last_name, email FROM customer_vw;
```

Ką galima daryti su views?

Galima rašyti select užklausas

Galima jungti prie lentelių

Kodėl naudoti views?
Duomenų saugumas

```
CREATE VIEW active_customer_vw
 (customer_id,
  first_name,
  last_name,
  email
 )
AS
SELECT
  customer_id,
  first_name,
  last_name,
  concat(substr(email,1,2), '*****', substr(email, -4)) email
FROM customer
WHERE active = 1;
```

Kodėl naudoti views?
Duomenų agregavimas

```
CREATE VIEW sales_by_film_category
AS
SELECT
  c.name AS category,
  SUM(p.amount) AS total_sales
FROM payment AS p
  INNER JOIN rental AS r ON p.rental_id = r.rental_id
  INNER JOIN inventory AS i ON r.inventory_id = i.inventory_id
  INNER JOIN film AS f ON i.film_id = f.film_id
  INNER JOIN film_category AS fc ON f.film_id = fc.film_id
  INNER JOIN category AS c ON fc.category_id = c.category_id
GROUP BY c.name
ORDER BY total_sales DESC;
```

Kodėl naudoti views?
Sudėtingumo paslėpimas

```
CREATE VIEW film_stats
AS
SELECT f.film_id, f.title, f.description, f.rating,
 (SELECT c.name
  FROM category c
    INNER JOIN film_category fc
    ON c.category_id = fc.category_id
  WHERE fc.film_id = f.film_id)
category_name,
 (SELECT count(*)
  FROM film_actor fa
  WHERE fa.film_id = f.film_id
 ) num_actors,
(SELECT count(*)
 FROM inventory i
 WHERE i.film_id = f.film_id
) inventory_cnt,
(SELECT count(*)
 FROM inventory i
   INNER JOIN rental r
   ON i.inventory_id = r.inventory_id
 WHERE i.film_id = f.film_id
) num_rentals
FROM film f;
```

Kodėl naudoti views?
Duomenų surinkimas į vieną lentelę

```
CREATE VIEW payment_all
 (payment_id,
  customer_id,
  staff_id,
  rental_id,
  amount,
  payment_date,
  last_update
 )
AS
SELECT payment_id, customer_id, staff_id, rental_id,
  amount, payment_date, last_update
FROM payment_historic
UNION ALL
SELECT payment_id, customer_id, staff_id, rental_id,
  amount, payment_date, last_update
FROM payment_current;
```

Ar galime modifikuoti views duomenis?

Taip, bet tik tuos stulpelius, kurie nėra agreguoti.

```
UPDATE customer_vw
SET last_name = 'SMITH-ALLEN'
WHERE customer_id = 1;
```

Užduotys

Exercise 14-1

Create a view definition that can be used by the following query to generate the given results:

```
SELECT title, category_name, first_name, last_name
FROM film_ctgry_actor
WHERE last_name = 'FAWCETT';

+---------------------+---------------+------------+-----------+
| title               | category_name | first_name | last_name |
+---------------------+---------------+------------+-----------+
| ACE GOLDFINGER      | Horror        | BOB        | FAWCETT   |
| ADAPTATION HOLES    | Documentary   | BOB        | FAWCETT   |
| CHINATOWN GLADIATOR | New           | BOB        | FAWCETT   |
| CIRCUS YOUTH        | Children      | BOB        | FAWCETT   |
```

Exercise 14-2

The film rental company manager would like to have a report that includes the name of every country, along with the total payments for all customers who live in each country. Generate a view definition that queries the `country` table and uses a scalar subquery to calculate a value for a column named `tot_payments`.