

Wydajność MySQL

Łukasz Rajchel

18 listopada 2008

- Benchmarki i Profilowanie MySQL

- Benchmarki i Profilowanie MySQL
- Źródła problemów

- Benchmarki i Profilowanie MySQL
- Źródła problemów
- Indeksowanie

- Benchmarki i Profilowanie MySQL
- Źródła problemów
- Indeksowanie
- Projektowanie schematów

- Benchmarki i Profilowanie MySQL
- Źródła problemów
- Indeksowanie
- Projektowanie schematów
- Kodowanie

- Benchmarki i Profilowanie MySQL
- Źródła problemów
- Indeksowanie
- Projektowanie schematów
- Kodowanie
- Parametry serwera

Indeks:

Jest to struktura danych, która przyspiesza szybkość operacji na tabeli bazy danych. Indeks jest kopią części tabeli. Indeksy zwiększają wielkość bazy danych, opóźniają niektóre operacje na bazie danych (np. INSERT, DELETE). Trzeba je stosować z umiarem.

Indeks Nieklastrowany (*non-clustered*):

Zazwyczaj jest to referencja do bloku zawierającego dane z wiersza które odpowiadają konkretnej wartości indeksowanego pola.

Indeks Klastrowany (*clustered*):

Dane są fizycznie posortowane w taki sam sposób jak indeks. Wolniejszy zapis, szybszy odczyt. Każda tabela może posiadać tylko jeden taki indeks.

Pokrywanie Indeksami (*index covering*)

Pokrywanie Indeksami to tworzenie indeksu nieklastrowanego zawierającego wszystkie kolumny używane w zapytaniu SQL. Jest to proste i szybkie rozwiązanie wielu problemów wydajności zapytań SQL.

- MyISAM to domyślny mechanizm składowania danych w MySQL (od wersji 5.1 domyślnym silnikiem jest InnoDB)

- MyISAM to domyślny mechanizm składowania danych w MySQL (od wersji 5.1 domyślnym silnikiem jest InnoDB)
- MyISAM posiada pełnotekstowe wyszukiwanie (full-text search)

- MyISAM to domyślny mechanizm składowania danych w MySQL (od wersji 5.1 domyślnym silnikiem jest InnoDB)
- MyISAM posiada pełnotekstowe wyszukiwanie (full-text search)
- W MyISAM nie można tworzyć kluczy obcych

- MyISAM to domyślny mechanizm składowania danych w MySQL (od wersji 5.1 domyślnym silnikiem jest InnoDB)
- MyISAM posiada pełnotekstowe wyszukiwanie (full-text search)
- W MyISAM nie można tworzyć kluczy obcych
- MyISAM nie wspiera transakcji

- Benchmarki pozwalają mierzyć wydajność

- Benchmarki pozwalają mierzyć wydajność
- Punktem początkowym jest punkt, w którym rozpoczyna się testowanie

- Benchmarki pozwalają mierzyć wydajność
- Punktem początkowym jest punkt, w którym rozpoczyna się testowanie
- Zawsze konieczne jest ustalenie celu

- Benchmarki pozwalają mierzyć wydajność
- Punktem początkowym jest punkt, w którym rozpoczyna się testowanie
- Zawsze konieczne jest ustalenie celu
- Zawsze zmieniaj tylko jedną rzecz w danym momencie

- Benchmarki pozwalają mierzyć wydajność
- Punktem początkowym jest punkt, w którym rozpoczyna się testowanie
- Zawsze konieczne jest ustalenie celu
- Zawsze zmieniaj tylko jedną rzecz w danym momencie
- Wszystko zapisuj

- Benchmarki pozwalają mierzyć wydajność
- Punktem początkowym jest punkt, w którym rozpoczyna się testowanie
- Zawsze konieczne jest ustalenie celu
- Zawsze zmieniaj tylko jedną rzecz w danym momencie
- Wszystko zapisuj
- Wyłącz query cache

- Pozwala na diagnozę działającego systemu

- Pozwala na diagnozę działającego systemu
- Używaj `EXPLAIN`

- Pozwala na diagnozę działającego systemu
- Używaj `EXPLAIN`
- Używaj *Slow Query Log* i `mysqldumpslow`

- Pozwala na diagnozę działającego systemu
- Używaj EXPLAIN
- Używaj *Slow Query Log* i `mysqldumpslow`
- Skup się na rzeczach, które dadzą największy zysk

- Pozwala na diagnozę działającego systemu
- Używaj EXPLAIN
- Używaj *Slow Query Log* i `mysqldumpslow`
- Skup się na rzeczach, które dadzą największy zysk
- Używaj `mytop`

Najważniejsze:

- Źle dobrane indeksy

Najważniejsze:

- Źle dobrane indeksy
- Nieefektywne projekty schematów baz

Najważniejsze:

- Źle dobrane indeksy
- Nieefektywne projekty schematów baz
- Złe praktyki kodowania

Najważniejsze:

- Źle dobrane indeksy
- Nieefektywne projekty schematów baz
- Złe praktyki kodowania

Mniej ważne:

- Źle ustawione zmienne serwera

Najważniejsze:

- Źle dobrane indeksy
- Nieefektywne projekty schematów baz
- Złe praktyki kodowania

Mniej ważne:

- Źle ustawione zmienne serwera
- Wąskie gardła sieciowe i sprzętowe

- Nieodpowiednio dobrane i brakujące indeksy są najszybszą drogą do zabicia systemu

- Nieodpowiednio dobrane i brakujące indeksy są najszybszą drogą do zabicia systemu
- Szukaj kolumn, które powinny być indeksowane

- Nieodpowiednio dobrane i brakujące indeksy są najszybszą drogą do zabicia systemu
- Szukaj kolumn, które powinny być indeksowane
- Zapewnij dobrą selektywność indeksowanych pól

- Nieodpowiednio dobrane i brakujące indeksy są najszybszą drogą do zabicia systemu
- Szukaj kolumn, które powinny być indeksowane
- Zapewnij dobrą selektywność indeksowanych pól
- Zwracaj uwagi na kolejność indeksów w przypadku indeksów wielokolumnowych

- Nieodpowiednio dobrane i brakujące indeksy są najszybszą drogą do zabicia systemu
- Szukaj kolumn, które powinny być indeksowane
- Zapewnij dobrą selektywność indeksowanych pól
- Zwracaj uwagi na kolejność indeksów w przypadku indeksów wielokolumnowych
- Gdy baza rośnie, sprawdzaj czy indeksowanie jest nadal optymalne

- Nieodpowiednio dobrane i brakujące indeksy są najszybszą drogą do zabicia systemu
- Szukaj kolumn, które powinny być indeksowane
- Zapewnij dobrą selektywność indeksowanych pól
- Zwracaj uwagi na kolejność indeksów w przypadku indeksów wielokolumnowych
- Gdy baza rośnie, sprawdzaj czy indeksowanie jest nadal optymalne
- Usuwanie redundantne indeksy aby zwiększyć predkość zapisu

- Nieodpowiednio dobrane i brakujące indeksy są najszybszą drogą do zabicia systemu
- Szukaj kolumn, które powinny być indeksowane
- Zapewnij dobrą selektywność indeksowanych pól
- Zwracaj uwagi na kolejność indeksów w przypadku indeksów wielokolumnowych
- Gdy baza rośnie, sprawdzaj czy indeksowanie jest nadal optymalne
- Usuwanie redundantne indeksy aby zwiększyć predkość zapisu
- Używaj ANALYZE TABLE (robi się to automatycznie po każdym ALTER TABLE)

Wartości liczników ANALYZE TABLE

MyISAM traktuje każdą wartość NULL jako różną wartość. InnoDB traktuje wszystkie wartości NULL jako identyczne. Dlatego wartości liczników dla tabeli MyISAM i InnoDB mogą być różne.

Zachowanie MyISAM można zmienić poprzez wydanie polecenia:
`SET myisam_stats_method = 'nulls_equal';`

Przykład - Struktura bazy

```
CREATE TABLE Tags (  
    tag_id INT UNSIGNED NOT NULL AUTO_INCREMENT,  
    tag_text VARCHAR(50) NOT NULL,  
    PRIMARY KEY (tag_id)  
) ENGINE=MyISAM;
```

```
CREATE TABLE Products (  
    product_id INT UNSIGNED NOT NULL AUTO_INCREMENT,  
    name VARCHAR(100) NOT NULL,  
    PRIMARY KEY (product_id)  
    // inne pola...  
) ENGINE=MyISAM;
```

```
CREATE TABLE Product2Tags (  
    product_id INT UNSIGNED NOT NULL,  
    tag_id INT UNSIGNED NOT NULL,  
    PRIMARY KEY (product_id, tag_id)  
) ENGINE=MyISAM;
```

To zapytanie wykorzystuje indeks:

```
SELECT p.name, COUNT(*) as tags
FROM Products2Tags p2t INNER JOIN Products p
    ON p2t.product_id = p.product_id
GROUP BY p.name;
```

Natomiast to jego nie wykorzystuje:

```
SELECT t.tag_text, COUNT(*) as products
FROM Products2Tags p2t INNER JOIN Tags t
    ON p2t.tag_id = t.tag_id
GROUP BY t.tag_text;
```

Problem można rozwiązać dodając indeks:

```
// InnoDB
CREATE INDEX ix_tag
ON Products2Tags (tag_id);

// MyISAM (pokrywanie indeksami)
CREATE INDEX ix_tag_prod
ON Products2Tags (tag_id, product_id);
```

Ważne:

Sprawdź silnik, który jest wykorzystywany dla danej tabeli gdyż MyISAM i InnoDB inaczej przechowują indeksy.

- Nieefektywne schematy są kolejną rzeczą, która łatwo zabije system

- Nieefektywne schematy są kolejną rzeczą, która łatwo zabije system
- Używaj najmniejszych typów danych jakie są potrzebne

- Nieefektywne schematy są kolejną rzeczą, która łatwo zabije system
- Używaj najmniejszych typów danych jakie są potrzebne
- Rozważ rozdzielenie wielokolumnowych tabel na mniejsze

- Nieefektywne schematy są kolejną rzeczą, która łatwo zabije system
- Używaj najmniejszych typów danych jakie są potrzebne
- Rozważ rozdzielenie wielokolumnowych tabel na mniejsze
- Rozważ rozdzielenie wielowierszowych kolumn używając partycjonowania lub tabeli złączeń

- Nieefektywne schematy są kolejną rzeczą, która łatwo zabije system
- Używaj najmniejszych typów danych jakie są potrzebne
- Rozważ rozdzielenie wielokolumnowych tabel na mniejsze
- Rozważ rozdzielenie wielowierszowych kolumn używając partycjonowania lub tabeli złączeń
- Pamiętaj: mniejsza ilość odczytów = szybsze działanie

Przykład - Złe dobieranie kluczy

```
CREATE TABLE Products2Tags (  
  record_id INT UNSIGNED NOT NULL AUTO_INCREMENT,  
  product_id INT UNSIGNED NOT NULL,  
  tag_id INT UNSIGNED NOT NULL,  
  PRIMARY KEY (record_id),  
  UNIQUE INDEX (product_id, tag_id)  
) ENGINE=MyISAM;
```

InnoDB:

Używaj małych indeksów klastrowanych gdyż są one dopisywane do wszystkich indeksowanych rekordów!

- Myśl o rekordach SQLa jako o zbiorach a nie pętlach for

- Myśl o rekordach SQLa jako o zbiorach a nie pętlach for
- Wykorzystuj procedury składowalne (5.0+)

- Myśl o rekordach SQLa jako o zbiorach a nie pętlach for
- Wykorzystuj procedury składowalne (5.0+)
- InnoDB: Wykorzystuj tabele zliczające

- Myśl o rekordach SQLa jako o zbiorach a nie pętlach for
- Wykorzystuj procedury składowalne (5.0+)
- InnoDB: Wykorzystuj tabele zliczające
- Przenieś indeksowane pola na jedna ze stron działania

- Myśl o rekordach SQLa jako o zbiorach a nie pętlach for
- Wykorzystuj procedury składowalne (5.0+)
- InnoDB: Wykorzystuj tabele zliczające
- Przenieś indeksowane pola na jedna ze stron działania
- Wykorzystuj pola wyliczane gdy są potrzebne

- Naucz się używać JOIN

- Naucz się używać JOIN
- Unikaj skorelowanych podzapytań

- Naucz się używać JOIN
- Unikaj skorelowanych podzapytań
- Nie próbuj przechytrzyć optimizera

Przykład - Przenoszenie indeksowanych pól

Zły pomysł:

```
WHERE TO_DAYS(order_created) - TO_DAYS(CURRENT_DATE()) >= 7
```

Lepszy pomysł:

```
WHERE order_created >= CURRENT_DATE() - INTERVAL 7 DAY
```

Najlepszy pomysł:

```
WHERE order_created >= '2008-11-10'
```

Przykład - Przenoszenie indeksowanych pól

Zły pomysł:

```
WHERE FROM_UNIXTIME(update_date) < '2008-11-10'
```

Lepszy pomysł:

```
WHERE update_date < UNIX_TIMESTAMP('2008-11-10')
```

Przykład - Wykorzystanie pól wyliczanych

Zły pomysł:

```
WHERE email_address LIKE '%.com';
```

Lepszy pomysł:

```
ALTER TABLE Customers
ADD COLUMN rv_email_address VARCHAR(80) NOT NULL;
UPDATE Customers SET rv_email_address = REVERSE(email_address);
CREATE INDEX ix_rv_email ON Customers (rv_email_address(20));

DELIMITER ;;
CREATE TRIGGER trg_bi_cust BEFORE INSERT ON Customers
FOR EACH ROW BEGIN
    SET NEW.rv_email_address = REVERSE(NEW.email_address);
END ;;
// ten sam trigger dla BEFORE UPDATE...

// Wybierz nowe pole:
WHERE rv_email_address LIKE CONTACT(REVERSE('%.com'), '%');
```

Zły pomysł:

```
SELECT p.name,  
       (SELECT MAX(price) FROM OrderItems  
        WHERE product_id = p.product_id)  
AS max_sold_price  
FROM Products p;
```

Lepszy pomysł:

```
SELECT p.name, MAX(oi.price)  
AS max_sold_price  
FROM Products p INNER JOIN OrderItems oi  
ON p.product_id = oi.product_id  
GROUP BY p.name;
```

Przykład - Wydzielone tablice

Zły pomysł:

```
SELECT c.company, o.*
FROM Customers c INNER JOIN Order o
  ON c.customer_id = o.customer_id
WHERE order_date = (
  SELECT MAX(order_date) FROM Orders
  WHERE customer = o.customer
) GROUP BY c.company;
```

Lepszy pomysł:

```
SELECT c.company, o.*
FROM Customers c INNER JOIN (
  SELECT customer_id, MAX(order_date) AS max_order
  FROM Orders
  GROUP BY customer_id
) AS m ON c.customer_id = m.customer_id
INNER JOIN Orders o
  ON c.customer_id = o.customer_id AND o.order_date = m.max_order
GROUP BY c.company;
```

- Pamiętaj, które parametry są globalne, a które dotyczą wątków

- Pamiętaj, które parametry są globalne, a które dotyczą wątków
- Rób niewielkie zmiany i testuj

- Pamiętaj, które parametry są globalne, a które dotyczą wątków
- Rób niewielkie zmiany i testuj
- Modyfikacja ustawień często jest szybkim rozwiązaniem ale jednocześnie tymczasowym

- Pamiętaj, które parametry są globalne, a które dotyczą wątków
- Rób niewielkie zmiany i testuj
- Modyfikacja ustawień często jest szybkim rozwiązaniem ale jednocześnie tymczasowym
- Używaj query cache gdy jest to możliwe

- Pamiętaj, które parametry są globalne, a które dotyczą wątków
- Rób niewielkie zmiany i testuj
- Modyfikacja ustawień często jest szybkim rozwiązaniem ale jednocześnie tymczasowym
- Używaj query cache gdy jest to możliwe
- Używaj różnych silników (MyISAM, InnoDB) dla różnych zastosowań

- Pamiętaj, które parametry są globalne, a które dotyczą wątków
- Rób niewielkie zmiany i testuj
- Modyfikacja ustawień często jest szybkim rozwiązaniem ale jednocześnie tymczasowym
- Używaj query cache gdy jest to możliwe
- Używaj różnych silników (MyISAM, InnoDB) dla różnych zastosowań
- Pamięć jest najtańszą, najszybszą i najłatwiejszą metodą zwiększenia wydajności serwera MySQL

- Performance Tuning Best Practices for MySQL - Google Talk
- Dokumentacja MySQL

Pytania?