



Khoa CNTT
Đại Học Đà Lạt



HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

FULLTEXT Search

Th.S. Đoàn Minh Khuê
khuedm@dlu.edu.vn

Đặt vấn đề

Định nghĩa

Cách sử dụng

Các hàm trong Full Text Search

Query Expansion Search

Đặt vấn đề

- ❖ Cần tìm kiếm cả câu mà chỉ nhớ một số từ trong câu đó.
- ❖ Làm cách nào để search được bản ghi đó?
- ❖ **Ví dụ:** tìm các bản ghi có từ “*database*”

| | id | title | body |
|---|----|--------------------------|-------------------------------|
| | 1 | MySQL Tutorial | This database tutorial ... |
| | 2 | How To Use MySQL | After you went through a ... |
| ▶ | 3 | Optimizing Your Database | In this database tutorial ... |

```
SELECT *  
FROM arcticles  
WHERE body LIKE '%database%'
```

| | id | title | body |
|---|----|--------------------------|-------------------------------|
| ▶ | 1 | MySQL Tutorial | This database tutorial ... |
| | 3 | Optimizing Your Database | In this database tutorial ... |

❖ Hạn chế:

- Khi không đánh index thì tốc độ tìm kiếm chậm.
- Hiệu suất không cao.
- Xảy ra tình trạng overload nếu dữ liệu quá dài hoặc quá nhiều

- ❖ **Full Text Search** là một kỹ thuật tìm kiếm các tài liệu không phù hợp với tiêu chí tìm kiếm. Tài liệu ở đây có thể là một mô tả sản phẩm, một bài viết được lưu trữ trong Database của MySQL.
- ❖ MySQL chỉ hỗ trợ **FULLTEXT** cho các kiểu dữ liệu CHAR, VARCHAR hoặc TEXT, kiểu lưu trữ table phải là *MyISAM* hoặc *InnoDB* (từ phiên bản 5.6)

Inverted index

- ❖ Là kĩ thuật đánh *index* theo đơn vị *term*
- ❖ Nhằm mục đích *map* giữa các *term* với các bản ghi chứa *term* đó.
- ❖ Ví dụ: Cho 3 dòng dữ liệu

D1 = "Tôi đi đến Đà Lạt"

D2 = "Tôi đi Hà Nội"

D3 = "đến Hà Nội"

Inverted Index sẽ được lưu dưới dạng như sau:

"Tôi" => {D1, D2} "đến" => {D1, D3}

"đi" => {D1, D2} "Hà" => {D2, D3}

"Đà" => {D1} "Nội" => {D2, D3}

"Lạt" => {D1}

Khi tìm kiếm cụm từ "*Tôi đi Đà Lạt*", chỉ cần tìm kiếm ở {**D1, D2**}

- ❖ Khi tìm từ "*and*" hoặc "*I*" thì mặc định MySQL sẽ xác định đó là những từ vô nghĩa → trong **Full-text search** những từ có 3 chữ cái đều là vô nghĩa.
- ❖ Độ dài tối thiểu trong MySQL:
 - **InnoDB**: 3
 - **MyISAM**: 4
- ❖ Để thay đổi độ dài này bằng cách mở file `/etc/mysql/my.cnf` và thực hiện thay đổi giá trị.
 - Với **InnoDB**
 - `innodb_ft_min_token_size` (độ dài tối thiểu)
 - `innodb_ft_max_token_size` (độ dài tối đa)
 - Với **MyISAM**
 - `ft_min_word_len` (độ dài tối thiểu)
 - `ft_max_word_len` (độ dài tối đa)

Định nghĩa

| MyISAM | InnoDB |
|--|---|
| Hỗ trợ Table - Level Locking | Hỗ trợ Row - Level Locking |
| Được thiết kế cho nhu cầu về tốc độ | Được thiết kế để đạt hiệu suất tối đa khi xử lý lượng dữ liệu lớn |
| Không hỗ trợ Foreign keys vì thế nên chúng ta gọi MySQL với MyISAM là DBMS | Hỗ trợ Foreign keys vì thế nên chúng ta gọi MySQL với InnoDB là RDBMS |
| Lưu trữ tables, data, indexes của nó trong không gian đĩa bằng cách sử dụng 3 file riêng biệt (table_name.FRM, table_name.MYD, table_name.MYI) | Lưu trữ tables, indexes của nó trong 1 không gian bảng |
| Không hỗ trợ Transaction | Có hỗ trợ Transaction |
| Hỗ trợ Full Text Search | Từ phiên bản 5.6 mới có |



Cách tạo Full Text Search

- ❖ Lúc tạo bảng **CREATE TABLE**
- ❖ Trong lệnh **ALTER TABLE**
- ❖ Bằng lệnh **CREATE INDEX**

Lúc tạo bảng CREATE TABLE

❖ Cú pháp:

```
CREATE TABLE table_name(  
    column_list,  
    ...,  
    FULLTEXT (column1,column2,..)  
);
```

❖ Ví dụ:

```
CREATE TABLE BaiBao (  
    id INT NOT NULL AUTO_INCREMENT,  
    tieude VARCHAR(255) NOT NULL,  
    noidung TEXT,  
    PRIMARY KEY (id),  
    FULLTEXT (noidung)  
);
```

Trong lệnh ALTER TABLE

❖ Cú pháp:

```
ALTER TABLE table_name  
ADD FULLTEXT(column_name1,  
column_name2,...)
```

❖ Ví dụ:

```
ALTER TABLE products  
ADD FULLTEXT(productDescription,productLine)
```



Bảng lệnh CREATE INDEX

❖ Cú pháp:

```
CREATE FULLTEXT INDEX index_name  
ON table_name(idx_column_name,...)
```

❖ Ví dụ:

```
CREATE FULLTEXT INDEX address  
ON offices(addressLine1,addressLine2)
```



Xóa Index Full Text Search

❖ Cú pháp:

```
ALTER TABLE table_name  
DROP INDEX index_name;
```

❖ Ví dụ:

```
ALTER TABLE offices  
DROP INDEX address;
```

❖ **Hàm MATCH và AGAINST**

❖ **Sắp xếp kết quả trả về dựa vào mức độ liên quan**

```
SELECT * FROM table_name  
WHERE MATCH (column_name_list)  
AGAINST ('keyword')
```



Hàm MATCH và AGAINST

```
CREATE TABLE articles (  
  ID INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,  
  title VARCHAR(200),  
  body TEXT,  
  FULLTEXT (title, body)  
)  
ENGINE=InnoDB;
```

```
INSERT INTO articles (title,body) VALUES  
('MySQL Tutorial','This database tutorial ...'),  
('How To Use MySQL','After you went through a ...'),  
('Optimizing Your Database','In this database tutorial ...'),  
('MySQL vs. YourSQL','When comparing databases ...'),  
('MySQL Security','When configured properly, MySQL ...'),  
('Database, Database, Database','database database database'),  
('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),  
('MySQL Full-Text Indexes', 'MySQL fulltext indexes use a ..');
```

Hàm MATCH và AGAINST

❖ Để truy vấn tìm kiếm bằng kỹ thuật Full Text Search:

```
SELECT * FROM articles  
WHERE MATCH (title,body)  
AGAINST ('database Tutorial')
```

| id | title | body |
|----|------------------------------|-------------------------------|
| 6 | Database, Database, Database | database database database |
| 1 | MySQL Tutorial | This database tutorial ... |
| 3 | Optimizing Your Database | In this database tutorial ... |



Sắp xếp kết quả trả về dựa vào mức độ liên quan

- ❖ Trong việc xử lý tìm kiếm thì quan trọng là sắp xếp kết quả trả về theo thứ tự: tài liệu nào giống nhiều nhất thì nằm trên cùng, giống ít nhất thì nằm dưới cùng.
- ❖ Khi sử dụng hàm **MATCH()** ở lệnh **WHERE** thì MySQL sẽ trả về giá trị có mức độ liên quan lên đầu tiên.



Sắp xếp kết quả trả về dựa vào mức độ liên quan

Ví dụ:

```
ALTER TABLE products  
ADD FULLTEXT(productName);
```

Truy vấn tìm kiếm trên field này, bằng hai từ khóa **Ford** hoặc **1932** hoặc có cả hai.

```
SELECT  
    productName,  
    productLine  
FROM products  
WHERE  
    MATCH(productName)  
    AGAINST( '1932, Ford' );
```

Sắp xếp kết quả trả về dựa vào mức độ liên quan

SELECT

productName,
productLine

FROM products

WHERE

MATCH(productName)

AGAINST('1932,Ford');

| productName | productLine |
|-------------------------------------|------------------|
| 1932 Model A Ford 1-Coupe | Vintage Cars |
| 1932 Alfa Romeo 8C2300 Spider Sport | Vintage Cars |
| 1968 Ford Mustang | Classic Cars |
| 1969 Ford Falcon | Classic Cars |
| 1940 Ford Pickup Truck | Trucks and Buses |
| 1911 Ford Town Car | Vintage Cars |
| 1926 Ford Fire Engine | Trucks and Buses |
| 1913 Ford Model T Speedster | Vintage Cars |
| 1934 Ford V8 Coupe | Vintage Cars |
| 1903 Ford Model A | Vintage Cars |



Các hàm trong Full Text Search

- ❖ **Natural Language Full-Text Searches**
- ❖ **Boolean Full-Text Searches**

- ❖ Để thực hiện tìm kiếm theo ngôn ngữ tự nhiên, MySQL sử dụng hai hàm **MATCH()** và **AGAINST()** .
- ❖ Hàm **AGAINST()** theo mặc định sẽ nằm ở chế độ **IN NATURAL LANGUAGE MODE**.
- ❖ Xác định tường minh bằng cách thêm từ khóa **IN NATURAL LANGUAGE MODE** vào tham số thứ hai của hàm.
- ❖ Chế độ sắp xếp mặc định theo mức độ phù hợp nhất. Được tính theo công thức :

$$w = (\log(\text{dtf})+1)/\text{sumdtf} * U/(1+0.0115*U) \log((N-\text{nf})/\text{nf})$$

```
SELECT * FROM articles
WHERE MATCH (title,body)
AGAINST ('database Tutorial IN NATURAL LANGUAGE MODE)
```

Boolean Full-Text Searches

- ❖ Để thực hiện tìm kiếm toàn văn bản trong chế độ **Boolean**, sử dụng **IN BOOLEAN MODE** trong biểu thức **AGAINST**
- ❖ **Ví dụ** : Tìm kiếm bài viết bắt buộc phải có cả hai từ khóa *mysql database*

```
SELECT
    *
FROM
    articles
WHERE
    MATCH(title, body) AGAINST(
        '+mysql +database' IN
        BOOLEAN MODE)
```

| id | title | body |
|----|----------------|----------------------------|
| 1 | MySQL Tutorial | This database tutorial ... |

Các toán tử trong Boolean Full-Text Searches

| Toán tử | Mô tả |
|---------|--|
| + | Từ phải xuất hiện |
| – | Từ không được xuất hiện |
| > | Bao gồm từ này, và tăng giá trị xếp hạng |
| < | Bao gồm từ này, và giảm giá trị xếp hạng |
| () | Nhóm các từ thành các biểu thức con (cho phép chúng được bao gồm, loại trừ, xếp hạng, v.v. như một nhóm). |
| ~ | Phủ định một từ được xếp hạng |
| * | Ký tự đại diện ở cuối từ |
| “” | Xác định một cụm từ (trái ngược với danh sách các từ riêng lẻ, toàn bộ cụm từ được khớp để đưa vào hoặc loại trừ). |

- ❖ Để tìm kiếm các hàng có chứa ít nhất một trong hai từ: “*mysql*” hoặc “*tutorial*”

'mysql tutorial'

- ❖ Để tìm kiếm các hàng có chứa cả hai từ: “*mysql*” và “*tutorial*”

'+mysql +tutorial'

- ❖ Để tìm kiếm các hàng có chứa từ “*mysql*”, nhưng đặt thứ hạng cao hơn cho các hàng có chứa “*tutorial*”:

'+mysql tutorial'

- ❖ Để tìm kiếm các hàng có chứa từ “*mysql*” nhưng không chứa từ “*tutorial*”

'+mysql -tutorial'

- ❖ Để tìm kiếm các hàng có chứa từ “*mysql*” và xếp hạng hàng thấp hơn nếu nó chứa từ “*tutorial*”.

`'+mysql ~tutorial'`

- ❖ Để tìm kiếm các hàng có chứa các từ “*mysql*” và “*tutorial*” hoặc “*mysql*” và “*training*” theo bất kỳ thứ tự nào, nhưng hãy đặt các hàng có chứa “*mysql tutorial*” cao hơn “*mysql training*”.

`'+mysql +(>tutorial <training)'`

- ❖ Để tìm các hàng có chứa các từ bắt đầu bằng “*my*”, chẳng hạn như “*mysql*”, “*mydatabase*”, ...

`'my*'`



Các tính chất của Boolean Full Text Searches

- ❖ MySQL không tự động sắp xếp các hàng theo mức độ liên quan theo thứ tự giảm dần trong kỹ thuật **Boolean full text search**.
- ❖ Để thực hiện các truy vấn **Boolean**, các bảng InnoDB yêu cầu tất cả các cột của biểu thức **MATCH** phải có chỉ mục **FULLTEXT**. Các bảng **MyISAM** không yêu cầu điều này, mặc dù tìm kiếm khá chậm.
- ❖ MySQL không hỗ trợ nhiều toán tử Boolean trên truy vấn tìm kiếm trên các bảng **InnoDB**. Ví dụ từ `'++ mysql'` sẽ trả về một lỗi. Với **MyISAM** bỏ qua các toán tử khác và sử dụng toán tử gần nhất. Ví dụ từ `'+ -mysql'` sẽ trở thành `'-mysql'`.
- ❖ Full Text Search của **InnoDB** không hỗ trợ dấu cộng (+) hoặc dấu trừ (-) trong từ khóa tìm kiếm. MySQL sẽ báo lỗi nếu tìm kiếm từ là `'mysql +'`, hoặc `'mysql-'`.
- ❖ Ngưỡng 50% có nghĩa là nếu một từ xuất hiện hơn 50% số hàng, MySQL sẽ bỏ qua từ đó trong kết quả tìm kiếm

- ❖ Việc mở rộng truy vấn được sử dụng để mở rộng kết quả tìm kiếm của các Full Text search dựa trên phản hồi liên quan (hoặc mở rộng truy vấn mù).
- ❖ Về mặt kỹ thuật, MySQL full text search thực hiện các bước sau khi mở rộng truy vấn được sử dụng:
 - Đầu tiên, tìm kiếm tất cả các hàng khớp với truy vấn tìm kiếm.
 - Thứ hai, tìm các từ có liên quan trong tất cả các hàng từ kết quả tìm kiếm.
 - Thứ ba, tìm kiếm lại dựa trên các từ có liên quan thay vì các từ khóa ban đầu được chỉ định bởi người dùng.



Sử dụng Query Expansion Full Text Search

- ❖ Để sử dụng Query Expansion thì sử dụng từ khóa **WITH QUERY EXPANSION** đặt trong hàm **AGAINST ()**.
- ❖ Cú pháp:

```
SELECT column1, column2  
FROM table1  
WHERE MATCH(column1,column2)  
      AGAINST('keyword'WITH QUERY EXPANSION);
```

❖ Ví dụ :

Tìm kiếm theo ngôn ngữ tự nhiên:

```
SELECT *  
FROM articles  
WHERE  
    MATCH(title, body) AGAINST(  
        'database' IN NATURAL LANGUAGE MODE);
```

| id | title | body |
|----|------------------------------|-------------------------------|
| 6 | Database, Database, Database | database database database |
| 3 | Optimizing Your Database | In this database tutorial ... |
| 1 | MySQL Tutorial | This database tutorial ... |

Tìm kiếm mở rộng

```
SELECT *  
FROM articles  
WHERE  
    MATCH(title, body) AGAINST(  
        'database' WITH QUERY EXPANSION)
```

| id | title | body |
|----|------------------------------|-------------------------------------|
| 3 | Optimizing Your Database | In this database tutorial ... |
| 6 | Database, Database, Database | database database database |
| 1 | MySQL Tutorial | This database tutorial ... |
| 5 | MySQL Security | When configured properly, MySQL ... |
| 8 | MySQL Full-Text Indexes | MySQL fulltext indexes use a .. |

ngram Full-Text Parser

- ❖ **Định nghĩa: ngram** là một chuỗi liên tiếp của một số ký tự từ một chuỗi văn bản.
- ❖ Chức năng chính của trình phân tích cú pháp toàn văn **ngram**: mã hóa một chuỗi văn bản thành một chuỗi n ký tự liên nhau.
- ❖ MySQL hỗ trợ trình phân tích cú pháp toàn văn **ngram** cho cả kiểu lưu trữ **InnoDB** và **MyISAM**

```
n = 1: 'm', 'y', 's', 'q', 'l'
n = 2: 'my', 'ys', 'sq', 'ql'
n = 3: 'mys', 'ysq', 'sql'
n = 4: 'mysq', 'ysql'
n = 5: 'mysql'
```

Tạo ngram Full-Text Parser

- ❖ Để tạo chỉ mục **FULL TEXT** sử dụng trình phân tích cú pháp **ngram** thì thêm **WITH PARSER NGRAM** trong câu lệnh **CREATE TABLE**, **ALTER TABLE** hoặc **CREATE INDEX**.

❖ Ví dụ :

```
CREATE TABLE posts (  
    ID INT PRIMARY KEY AUTO_INCREMENT,  
    title VARCHAR(255),  
    body TEXT,  
    FULLTEXT ( title , body ) WITH PARSER NGRAM  
) ENGINE=INNODB CHARACTER SET UTF8MB4;
```

```
SET NAMES utf8mb4;
```

```
INSERT INTO posts(title,body)  
VALUES( 'MySQL全文搜索', 'MySQL提供了具有许多好的功能的  
内置全文搜索'),  
      ( 'MySQL教程', '学习MySQL快速, 简单和有趣');
```

Tạo ngram Full-Text Parser

❖ Ví dụ :

```
CREATE TABLE posts (  
  ID INT PRIMARY KEY AUTO_INCREMENT,  
  title VARCHAR(255),  
  body TEXT,  
  FULLTEXT ( title , body ) WITH PARSER NGRAM  
) ENGINE=INNODB CHARACTER SET UTF8MB4;
```

```
SET NAMES utf8mb4;
```

```
INSERT INTO posts(title,body)  
VALUES( 'MySQL全文搜索', 'MySQL提供了具有许多好的功能的  
内置全文搜索'),  
      ( 'MySQL教程', '学习MySQL快速, 简单和有趣');
```

Tìm kiếm cụm từ **搜索** trong bảng **posts**

```
SELECT ID, title, body  
FROM posts  
WHERE  
  MATCH (title , body) AGAINST ( '搜索' );
```

| id | title | body |
|----|-----------|-------------------------|
| 1 | MySQL全文搜索 | MySQL提供了具有许多好的功能的内置全文搜索 |

Tạo ngram Full-Text Parser

❖ Xem cách ngram mã hóa văn bản

```
SET GLOBAL innodb_ft_aux_table="test/posts";

SELECT
    *
FROM
    information_schema.innodb_ft_index_cache
ORDER BY
    doc_id ,
    position;
```

| WORD | FIRST_DOC_ID | LAST_DOC_ID | DOC_COUNT | DOC_ID | POSITION |
|------|--------------|-------------|-----------|--------|----------|
| my | 2 | 4 | 3 | 2 | 0 |
| ys | 2 | 4 | 3 | 2 | 1 |
| sq | 2 | 4 | 3 | 2 | 2 |
| ql | 2 | 4 | 3 | 2 | 3 |
| l全 | 2 | 2 | 1 | 2 | 4 |
| 全文 | 2 | 2 | 1 | 2 | 5 |
| 文搜 | 2 | 2 | 1 | 2 | 8 |
| 搜索 | 2 | 2 | 1 | 2 | 11 |

- ❖ Dành cho tiếng Nhật
- ❖ Mã hóa văn bản thành các từ có nghĩa
- ❖ Hỗ trợ để sử dụng với **InnoDB** và **MyISAM**.
- ❖ Ví dụ:

MeCab mã hóa “データベース管理” (“*Quản lý cơ sở dữ liệu*”) thành “データベース” (“*Cơ sở dữ liệu*”) và “管理” (“*Quản lý*”).

- ❖ **Ưu điểm:** **MeCab** nhỏ hơn chỉ mục **ngram** nên tìm kiếm toàn văn bản nhanh hơn.
- ❖ **Hạn chế:** Mất nhiều thời gian hơn để mã hóa tài liệu
- ❖ **Tham khảo:**

<https://dev.mysql.com/doc/refman/8.0/en/fulltext-search-mecab.html>

