Báo cáo tuần 4 DE: SQL

1. **Các thành phần của cơ sở dữ liệu truyền thống**

Parser: Kiểm tra syntax của câu query

Optimizer: Xây dựng chiến lực thực thi cho câu query

Storage Engine: Quyết định cách dữ liệu được lưu trữ và truy cập trên hệ thống.

Quá trình thực hiện của MySQL:

A black background with white squares

Description automatically generated

Một số lưu ý về các thành phần trong MySQL:

1. Optimizer: Nếu như bạn thực hiện 1 số câu query có cấu trúc giống nhau, chỉ cần thay số. VD “SELECT \* FROM class WHERE name = “Hoc sinh i”” với i chạy từ 1->10000 thì rõ rang chiến lược thực thi của các câu này là giống nhau. Do đó nếu như phải xây dựng 1000 chiến lược thực thi sẽ tốn kém.

MySQL hỗ trợ prepare statements cho những trường hợp này.

|  |
| --- |
| “PREPARE stmt FROM 'SELECT \* FROM class WHERE name = ?';  SET @name = '';  $stmt = $mysqli->prepare("SELECT \* FROM class WHERE name = ?");  for ($i = 1; $i <= 10000; $i++) {  $name = "Hoc sinh " . $i;  $stmt->bind\_param("s", $name); // 's' chỉ ra rằng biến $name là một chuỗi  $stmt->execute();  $result = $stmt->get\_result();  }  $stmt->close();  DEALLOCATE PREPARE stmt;” |

1. Storage Engine:

Các loại Storage Engine

1. **InnoDB:**

* Là storage engine mặc định từ MySQL 5.5 trở đi.
* Hỗ trợ giao dịch ACID (Atomicity, Consistency, Isolation, Durability), điều này làm cho nó phù hợp với các ứng dụng yêu cầu độ tin cậy cao và hỗ trợ giao dịch.
* Cung cấp tính năng khóa hàng, giảm thiểu xung đột và tăng hiệu suất trong môi trường đa người dùng.
* Hỗ trợ tính năng Foreign Key constraints để bảo đảm tính toàn vẹn dữ liệu giữa các bảng.

1. **CSV:**

* Lưu trữ dữ liệu dưới dạng tệp CSV
* Cho phép dữ liệu có thể được nhập và xuất dễ dàng với các công cụ bảng tính hoặc ứng dụng xử lý văn bản

1. **Archive**:

* Được thiết kế để lưu trữ một lượng lớn dữ liệu không thay đổi, như log files.
* Hỗ trợ nén dữ liệu để tiết kiệm không gian lưu trữ.
* Không hỗ trợ khóa, do đó không thích hợp cho các trường hợp sử dụng với nhiều giao dịch cập nhật.

Thêm storage engine vào bảng khi khởi tạo:

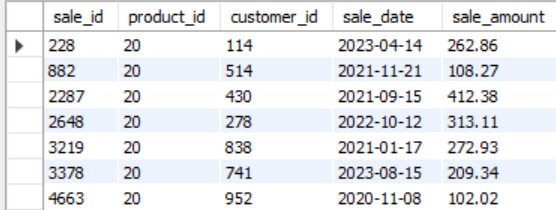
|  |
| --- |
| CREATE TABLE my\_table (  id INT AUTO\_INCREMENT PRIMARY KEY,  data VARCHAR(100)  ) ENGINE=InnoDB; |

Thay đổi Storage Engine cho bảng đã có sẵn:

|  |
| --- |
| ALTER TABLE my\_table ENGINE = MyISAM; |

1. **Các câu truy vấn cần tránh khi sử dụng SQL**
2. **Sử dụng tên cột thay cho \* nếu không cần thiết**

Chúng ta hãy bắt đầu bằng bảng *sales:*

**

Bây giờ ta hãy thực hiện 2 câu truy vấn, một có *SELECT \** và một chỉ lựa chọn các cột phù hợp:

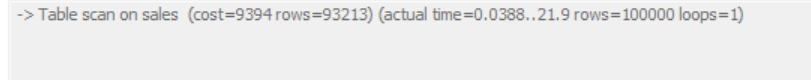
A close up of a sign

Description automatically generated



A close-up of a white background

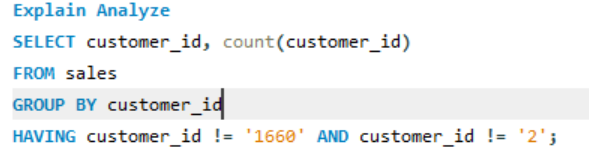
Description automatically generated

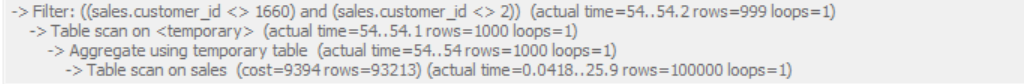


Ta thấy thời gian khi sử dụng *SELECT \** là 0.06 và thời gian khi chỉ lựa chọn cột phù hợp là 0.04.

1. **Tránh đưa mệnh đề HAVING theo sau SELECT**

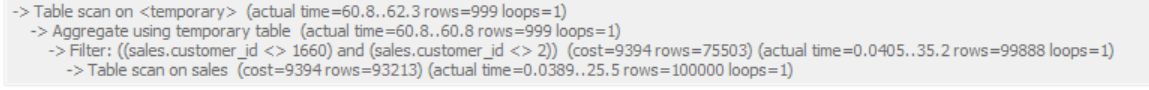
HAVING sẽ quét qua bảng cuối và lấy ra các kết quả phù hợp. Điều này khiến cho HAVING không hoạt động tốt với SELECT khi nó phải duyệt qua bảng 2 lần.





A close-up of a computer code

Description automatically generated



Ta thấy khi sử dụng HAVING như cách 1 chi phí sẽ là 54 + 54 + 54 + 0.04 còn cách 2 sử dụng WHERE thay thế sẽ là 60 + 60 + 0.04 + 0.04. Cách sau rõ rang có chi phí thấp hơn.

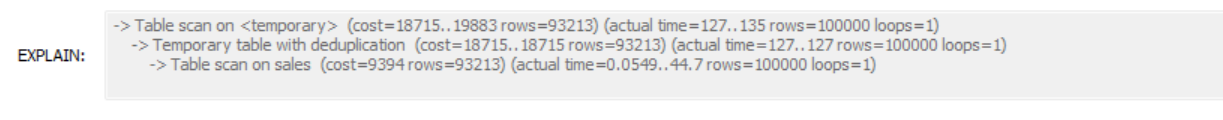
Điều khác biệt ở đây chính là nằm ở bước filter của HAVING tốn 54 còn của WHERE chỉ tốn 0.04.

1. **Loại bỏ các mệnh đề DISTINCT không cần thiết**

Khi thực hiện DISTINCT, hệ thống sẽ phải sắp xếp lại dữ liệu để loại bỏ những dữ liệu trùng lặp. Điều này tốn kém nên cần hạn chế sử dụng nếu thực sự không cần.

A close-up of a text

Description automatically generated



A black text on a white background

Description automatically generated

A close-up of a computer screen

Description automatically generated

Rõ rang thời gian chạy của DISTINCT lớn hơn rất nhiều khi so với không có DISTINCT.

1. **Un-nest các truy vấn nội bộ**

Viết lại các sub-query sẽ giúp truy vấn chạy hiệu quả và tối ưu hơn. Nhìn chung, việc unnest các sub-query luôn được tiến hành với sub-query tương tác với tối đa một bảng từ mệnh đề FROM, được sử dụng trong các mệnh đề ANY, ALL, và EXISTS.

A computer screen shot of a number

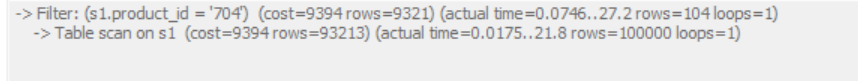
Description automatically generated

A white text with black text

Description automatically generated

A screenshot of a computer code

Description automatically generated

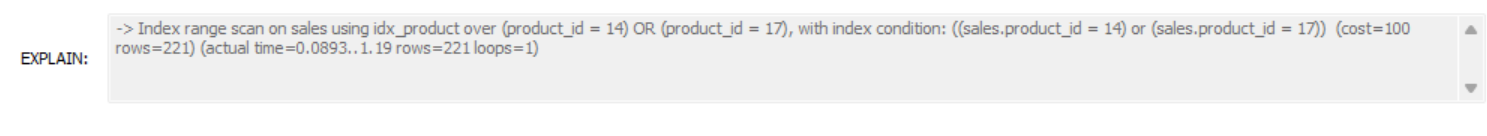


1. **Cân nhắc sử dụng mệnh đề IN khi truy vấn một cột đã được đánh index**

Mệnh đề IN có thể được khai thác cho các lệnh truy vấn sử dụng bảng đã được đánh index. Cần chú ý rằng danh sách IN chỉ được chứa các hàm, hoặc các giá trị là hằng số trong việc thực hiện khối truy vấn, ví dụ như các tham chiếu ngoài.

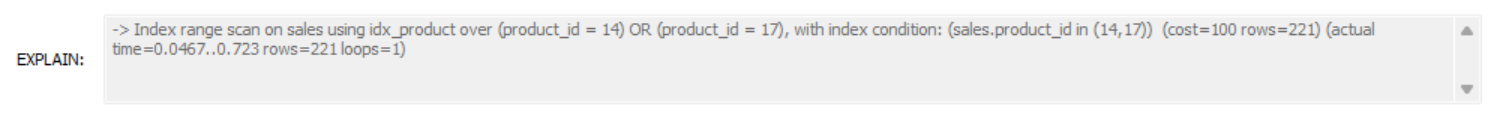
A white background with black and orange numbers

Description automatically generated



A computer code with text

Description automatically generated with medium confidence

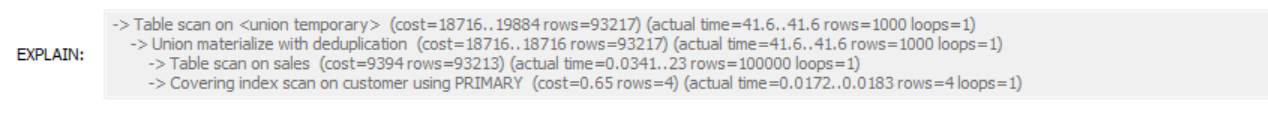


1. **Sử dụng UNION ALL thay cho UNION**

Mệnh đề UNION ALL nhanh hơn UNION bỏi vì mệnh đề UNION ALL không tính tới các điểm trùng lập, ưu tiên sử dụng UNION ALL khi chúng ta biết chắc chắn mỗi dòng trong kết quả sẽ là duy nhất hoặc có thể chấp nhận việc trùng lặp.

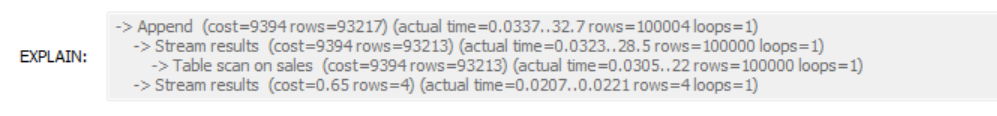
A computer code with black text

Description automatically generated



A screenshot of a computer code

Description automatically generated

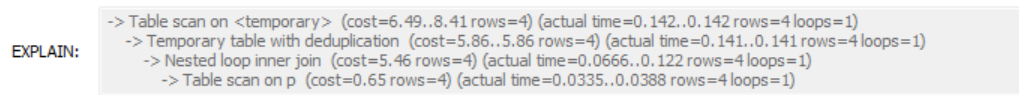


1. **Sử dụng EXIST thay cho DISTINCT**

Câu truy vấn gốc

**A screenshot of a computer

Description automatically generated**

****

Cấu truy vấn được tối ưu:

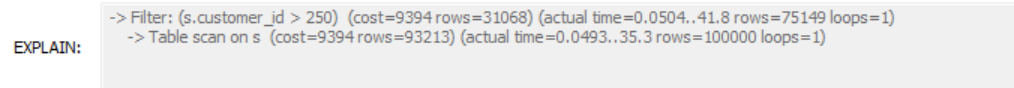
**A screenshot of a computer

Description automatically generated**

1. **Loại bỏ các phép toán thừa**

A white background with yellow and orange text

Description automatically generated



A screenshot of a computer

Description automatically generated

1. **Các loại NoSQL**
2. **Columnar Database**

Khác với các loại Database truyền thống (RDBMS) khi dữ liệu được lưu theo các hàng, Columnar Database lưu trữ dữ liệu theo các cột.

Một số đặc điểm của Columnar Database:

* Thực hiện các phép toán nhanh hơn so với RDBMS do chỉ cần thực hiện ngay trên cột.
* Lưu trữ tốn kém ít hơn do có thể áp dụng các phương pháp nén và phù hợp với cơ chế windows trên các hệ điều hành.
* Thêm, sửa, xóa dữ liệu tốn nhiều thời gian hơn so với RDBMS.

Một số columnar Database:

* Monet
* Google Big Query
* Apache Cassandra

A screenshot of a data visualization

Description automatically generated

1. **Document Database**

Document Database cho phép ta lưu trữ dữ liệu dạng bán cấu trúc như JSON, XML. MongoDB là đại diện tiêu biểu cho loại NoSQL này.

A screenshot of a computer code

Description automatically generated

Một số đặc điểm của Document Database:

* Không cần định nghĩa Schema. Mỗi collection có thể chứa những dữ liệu với cấu trúc khác nhau hoàn toàn
* Query dựa trên các cặp key-value
* Cho phép sửa đổi dữ liệu đã có sẵn.
* Có khả năng scale theo cả chiều dọc và chiều ngang

1. Graph Database

Graph Database giúp biểu diễn mối quan hệ giữa các đối tượng. Ở những RDBMS, muốn tìm mối liên hệ giữa các bảng, chúng ta cần sử dụng các lệnh JOIN, tuy nhiên các lệnh này sẽ có chi phi lớn khi lượng dữ liệu lớn. Graph Database ra đời nhằm giải quyết vấn đề này.

Đại diện tiêu biểu: Neo4j.

Một số đặc điểm của Graph Database:

* Đối tượng được biểu diễn thành các Node, bao gồm label, properties
* Giữa các đối tượng khác nhau có thể định nghĩa các Relation giữa chúng

A diagram of a work flow

Description automatically generated