**KIỂM TRA L1**

**Ngày: 01/10/2024**

**Tên: Nguyễn Văn Đạt**

**MSSV: 2286300010**

**Lớp: 22DRTA1**

**Chủ đề: Ứng Dụng MySQL trong Ngành Robot và AI**

**Thời gian làm bài:** ….. (phút)

**Mục tiêu học tập (CLO - Course Learning Outcomes):**

1. CLO1: Hiểu rõ về vai trò và ứng dụng của MySQL trong các hệ thống AI và Robot.
2. CLO2: Thiết kế và xây dựng cơ sở dữ liệu MySQL phục vụ các ứng dụng trong ngành Robot và AI.
3. CLO3: Áp dụng các truy vấn SQL để xử lý và phân tích dữ liệu trong hệ thống AI và Robot.
4. CLO4: Tối ưu hóa hệ thống cơ sở dữ liệu MySQL trong các ứng dụng AI và Robot.
5. CLO5: Tích hợp MySQL với các ứng dụng AI và Robot thời gian thực.

**Phần I: Câu hỏi lý thuyết (40 điểm)**

*(CLO1 & CLO5)*

1. **(1.0 điểm)** **[CLO1]**

Trình bày vai trò của MySQL trong quản lý và xử lý dữ liệu trong các hệ thống Robot và AI. So sánh MySQL với các hệ quản trị cơ sở dữ liệu khác (ví dụ: NoSQL) và nêu rõ lý do tại sao MySQL vẫn được sử dụng rộng rãi trong ngành Robot và AI.

Trả lời:

**Vai trò:**

MySQL được sử dụng phổ biến trong AI và robot để quản lý và xử lý dữ liệu lớn, như từ cảm biến, hình ảnh, và thuật toán học máy. Nó giúp lưu trữ và truy xuất dữ liệu nhanh chóng, hỗ trợ robot ra quyết định trong thời gian thực. MySQL còn tích hợp hiệu quả vào các hệ thống robot, cho phép quản lý dữ liệu thời gian thực, lịch sử hoạt động và kết quả từ các thuật toán, giúp tối ưu hóa quyết định và hành động.

**So Sánh MySQL và PostgreSQL:**

Hiệu suất: MySQL thường được biết đến với hiệu suất nhanh trong các ứng dụng đọc dữ liệu nặng, trong khi PostgreSQL nổi bật với khả năng xử lý truy vấn phức tạp và các ứng dụng cần tính mở rộng cao.

Tính năng: PostgreSQL hỗ trợ nhiều tính năng tiên tiến hơn như loại dữ liệu tùy chỉnh, hàm lưu trữ, và các chỉ mục đặc biệt. MySQL cung cấp một giao diện đơn giản hơn và phù hợp với các ứng dụng web phổ biến.

Mở rộng và tùy chỉnh: PostgreSQL cho phép tùy chỉnh cao hơn với các loại dữ liệu và hàm lưu trữ, trong khi MySQL cung cấp cấu trúc và tính năng đơn giản hơn.

**Tại sao MySQL vẫn được sử dụng rộng rãi trong AI và robot:**

MySQL dễ triển khai, sử dụng và tương thích với nhiều ứng dụng AI, robot. Nó xử lý tốt dữ liệu có cấu trúc, đáp ứng yêu cầu của các hệ thống robot và AI. Với cộng đồng lớn và tài liệu phong phú, MySQL hỗ trợ người dùng dễ dàng tìm giải pháp. Nhờ hiệu suất cao, tính đơn giản và sự hỗ trợ mạnh mẽ, MySQL vẫn là lựa chọn phổ biến cho các ứng dụng cần quản lý và truy xuất dữ liệu nhanh.

1. **(1.0 điểm)** **[CLO2]**

Trong một hệ thống Robot thông minh, dữ liệu từ cảm biến cần được lưu trữ và truy cập nhanh chóng để đảm bảo khả năng phản ứng tức thì. Hãy giải thích các yếu tố cần xem xét khi thiết kế cơ sở dữ liệu MySQL cho một hệ thống như vậy. Đưa ra một mô hình cơ sở dữ liệu hợp lý cho việc lưu trữ dữ liệu cảm biến thời gian thực.

Trả lời:

1. Cấu trúc dữ liệu hợp lý

Tối ưu hóa cho dữ liệu thời gian thực: Dữ liệu cảm biến thường có đặc điểm là liên tục và thay đổi theo thời gian, do đó cần lưu trữ dữ liệu với dấu thời gian (timestamp) để theo dõi thời điểm thu thập dữ liệu.

Phân đoạn dữ liệu (partitioning): Giúp tối ưu hóa việc truy vấn và lưu trữ, phân đoạn theo thời gian (ngày, giờ) sẽ giúp quản lý dữ liệu cảm biến liên tục hiệu quả hơn.

2. Hiệu suất truy xuất dữ liệu

Chỉ mục (Indexing): Tạo chỉ mục trên các cột quan trọng như thời gian, loại cảm biến để tăng tốc độ truy vấn. Tuy nhiên, cần cẩn trọng khi tạo nhiều chỉ mục vì sẽ làm tăng tải khi ghi dữ liệu.

Tối ưu hóa ghi và đọc (read-write): Cảm biến thường ghi dữ liệu liên tục, vì vậy cần tối ưu hóa các tác vụ ghi để không làm ảnh hưởng đến hiệu suất truy vấn dữ liệu.

3. Khả năng mở rộng

Quản lý khối lượng dữ liệu lớn: Khi cảm biến thu thập dữ liệu liên tục, dung lượng cơ sở dữ liệu sẽ tăng nhanh. Do đó, cần thiết kế cơ sở dữ liệu dễ mở rộng, có thể thông qua việc lưu trữ phân tán hoặc nhân bản (replication).

Xoay vòng dữ liệu cũ: Sử dụng cơ chế tự động xóa hoặc lưu trữ dữ liệu cũ sau một khoảng thời gian để tránh quá tải.

4. Độ tin cậy và tính nhất quán

Tính toàn vẹn dữ liệu: Đảm bảo các giá trị cảm biến được lưu đúng, không bị thiếu dữ liệu hoặc trùng lặp bằng cách sử dụng các ràng buộc (constraints) và khóa chính (primary key).

Đồng bộ dữ liệu: Trong hệ thống robot, nhiều cảm biến hoạt động đồng thời, do đó việc đồng bộ dữ liệu giữa các cảm biến và hệ thống cơ sở dữ liệu là quan trọng để không bị mất mát dữ liệu.

5. Phân tích dữ liệu theo thời gian thực

Bộ nhớ đệm (Caching): Sử dụng bộ nhớ đệm cho các truy vấn thường xuyên để giảm tải cho hệ thống.

Tự động ghi nhận và phân tích dữ liệu: Các quy trình xử lý và phân tích dữ liệu cảm biến theo thời gian thực cần được tích hợp để hệ thống robot có thể phản ứng nhanh chóng.

|  |
| --- |
| Mô hình cơ sở dữ liệu:  CREATE TABLE sensor\_data (  sensor\_id INT NOT NULL,  sensor\_type VARCHAR(50),  timestamp DATETIME NOT NULL,  value FLOAT,  unit VARCHAR(20),  PRIMARY KEY (sensor\_id, timestamp)  );  CREATE INDEX idx\_timestamp ON sensor\_data (timestamp); |

**sensor\_id**: Xác định cảm biến cụ thể trong hệ thống.

**sensor\_type**: Lưu trữ loại cảm biến, giúp phân loại dữ liệu.

t**imestamp**: Lưu lại thời điểm dữ liệu cảm biến được thu thập, rất quan trọng cho dữ liệu thời gian thực.

**value**: Giá trị dữ liệu cảm biến thu thập được, có thể là nhiệt độ, độ ẩm, ánh sáng,...

**unit**: Đơn vị đo lường của giá trị cảm biến, giúp tránh nhầm lẫn khi xử lý dữ liệu.

1. **(1.0 điểm)** **[CLO3]**

Với một ứng dụng AI trong Robot học máy (Machine Learning), dữ liệu đầu vào cần phải được chuẩn bị từ nhiều nguồn khác nhau. Hãy trình bày cách MySQL có thể hỗ trợ quá trình trích xuất, chuyển đổi, và nạp (ETL - Extract, Transform, Load) dữ liệu cho các mô hình học máy.

Trả lời:

Trong ứng dụng AI và robot học máy, MySQL hỗ trợ quy trình ETL qua các bước:

1. Trích xuất (Extract):

MySQL có thể kết nối và trích xuất dữ liệu từ nhiều nguồn như các cảm biến hoặc cơ sở dữ liệu khác thông qua truy vấn SQL (SELECT, JOIN) hoặc các driver kết nối (ODBC, JDBC).

2. Chuyển đổi (Transform):

Dữ liệu trích xuất có thể được làm sạch, chuẩn hóa, và chuyển đổi trực tiếp trong MySQL bằng cách sử dụng các hàm SQL như CONVERT, CAST, hoặc thực hiện tính toán, xử lý lỗi, và lọc dữ liệu.

3. Nạp (Load):

Dữ liệu sau khi chuyển đổi được nạp vào các bảng MySQL đã định nghĩa, hoặc xuất ra file để sử dụng trong các mô hình học máy. Các công cụ như LOAD DATA INFILE hoặc API giúp thực hiện quá trình nạp dữ liệu nhanh chóng và hiệu quả.

1. **(1.0 điểm)** **[CLO5]**

Tích hợp MySQL với các hệ thống AI thời gian thực đòi hỏi sự ổn định và hiệu suất cao. Hãy mô tả các kỹ thuật và công nghệ giúp MySQL có thể hoạt động hiệu quả trong các ứng dụng AI có yêu cầu xử lý dữ liệu thời gian thực từ robot.

Để MySQL hoạt động hiệu quả trong các ứng dụng AI thời gian thực từ robot, có thể áp dụng các kỹ thuật sau:

Chỉ mục hóa: Tăng tốc truy vấn trên các cột quan trọng.

Phân đoạn dữ liệu: Chia bảng lớn theo thời gian hoặc cảm biến.

Replication: Nhân bản dữ liệu để phân tán tải truy vấn.

Bộ nhớ đệm: Sử dụng Memcached hoặc Redis để lưu trữ kết quả truy vấn tạm thời.

Phân mảnh dữ liệu: Chia nhỏ cơ sở dữ liệu để xử lý song song.

InnoDB: Sử dụng InnoDB đảm bảo tính toàn vẹn và an toàn dữ liệu.

Tối ưu hóa truy vấn: Tối ưu truy vấn với EXPLAIN.

Kết nối đồng thời: Tái sử dụng kết nối để tăng hiệu suất.

Những kỹ thuật này giúp MySQL duy trì hiệu suất cao và ổn định khi xử lý dữ liệu thời gian thực từ robot.

**Phần II: Bài tập (6.0 điểm)**

*(CLO2, CLO3, CLO4, CLO5)*

**Bài 1: Thiết kế cơ sở dữ liệu cho hệ thống Robot tự động (3.0 điểm)** **[CLO2]**

Một công ty đang phát triển hệ thống robot tự động hóa trong dây chuyền sản xuất, với các loại dữ liệu bao gồm: thông tin robot, trạng thái hoạt động của robot, dữ liệu từ các cảm biến gắn trên robot. Bạn được yêu cầu thiết kế một cơ sở dữ liệu MySQL bao gồm các bảng sau:

* **Bảng Robots:** Lưu trữ thông tin về các robot (ID robot, tên robot, loại robot, trạng thái hiện tại).
* **Bảng Sensors:** Lưu trữ thông tin về các cảm biến gắn trên robot (ID cảm biến, loại cảm biến, ID robot, trạng thái cảm biến).
* **Bảng Logs:** Lưu trữ dữ liệu từ các cảm biến (ID log, ID cảm biến, giá trị dữ liệu, thời gian ghi nhận).

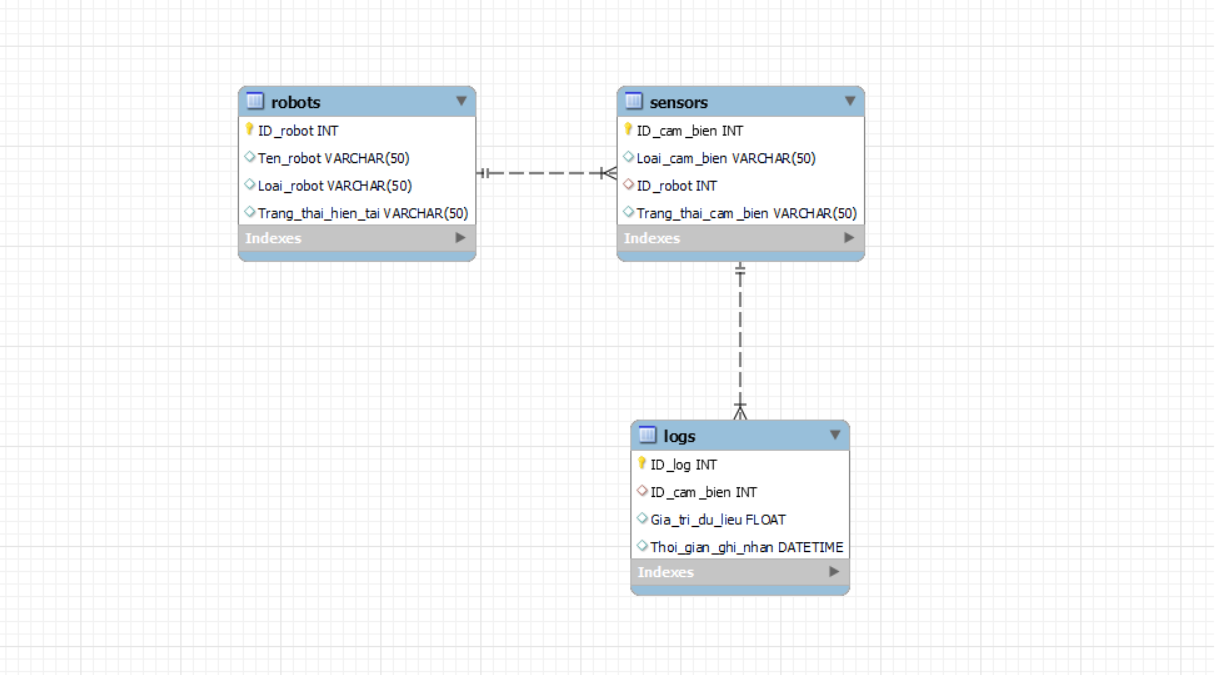
**Yêu cầu:**

1. Xây dựng lược đồ cơ sở dữ liệu bao gồm bảng Robots, Sensors và Logs với các ràng buộc khóa chính, khóa ngoại phù hợp.

1. Lược đồ Cơ sở Dữ liệu

Giải thích:

* Bảng Robots:
  + ID robot: Khóa chính, duy nhất định danh mỗi robot.
  + Tên robot: Tên gọi của robot.
  + Loại robot: Loại robot (ví dụ: robot hàn, robot sơn,...).
  + Trạng thái hiện tại: Trạng thái hoạt động hiện tại của robot (ví dụ: đang hoạt động, dừng, lỗi).
* Bảng Sensors:
  + ID cảm biến: Khóa chính, duy nhất định danh mỗi cảm biến.
  + Loại cảm biến: Loại cảm biến (ví dụ: cảm biến nhiệt độ, cảm biến áp suất,...).
  + ID robot: Khóa ngoại, liên kết với bảng Robots để xác định robot mà cảm biến được gắn vào.
  + Trạng thái cảm biến: Trạng thái của cảm biến (ví dụ: hoạt động, lỗi).
* Bảng Logs:
  + ID log: Khóa chính, tự tăng, duy nhất định danh mỗi bản ghi log.
  + ID cảm biến: Khóa ngoại, liên kết với bảng Sensors để xác định cảm biến tạo ra dữ liệu.
  + Giá trị dữ liệu: Giá trị đo được từ cảm biến.
  + Thời gian ghi nhận: Thời điểm ghi nhận dữ liệu.



1. Viết câu lệnh SQL để tạo các bảng trong MySQL.

|  |
| --- |
| CREATE TABLE Robots (  ID\_robot INT PRIMARY KEY,  Ten\_robot VARCHAR(50),  Loai\_robot VARCHAR(50),  Trang\_thai\_hien\_tai VARCHAR(50)  ); |

|  |
| --- |
| CREATE TABLE Sensors (  ID\_cam\_bien INT PRIMARY KEY,  Loai\_cam\_bien VARCHAR(50),  ID\_robot INT,  Trang\_thai\_cam\_bien VARCHAR(50),  FOREIGN KEY (ID\_robot) REFERENCES Robots(ID\_robot)  ); |

|  |
| --- |
| CREATE TABLE Logs (  ID\_log INT PRIMARY KEY AUTO\_INCREMENT,  ID\_cam\_bien INT,  Gia\_tri\_du\_lieu FLOAT,  Thoi\_gian\_ghi\_nhan DATETIME,  FOREIGN KEY (ID\_cam\_bien) REFERENCES Sensors(ID\_cam\_bien)  ); |

1. Viết câu truy vấn SQL để lấy ra danh sách các robot cùng với trạng thái cảm biến mới nhất (log gần nhất) của chúng.

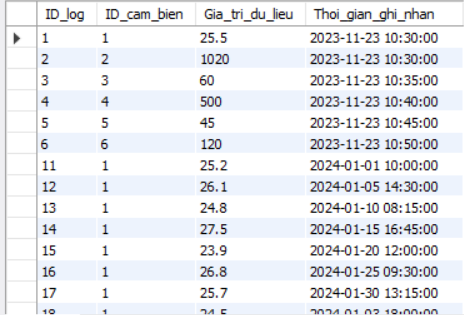
|  |
| --- |
| SELECT  r.ID\_robot,  r.Ten\_robot,  r.Loai\_robot,  r.Trang\_thai\_hien\_tai,  s.ID\_cam\_bien,  s.Loai\_cam\_bien,  s.Trang\_thai\_cam\_bien,  l.ID\_log,  l.Gia\_tri\_du\_lieu,  l.Thoi\_gian\_ghi\_nhan  FROM  Robots r  JOIN  Sensors s ON r.ID\_robot = s.ID\_robot  JOIN  Logs l ON s.ID\_cam\_bien = l.ID\_cam\_bien; |

**Bài 2: Truy vấn và phân tích dữ liệu cảm biến trong hệ thống robot (3.0 điểm)** **[CLO3, CLO4, CLO5]**

Bạn được giao nhiệm vụ truy vấn và phân tích dữ liệu từ hệ thống cảm biến của các robot để tối ưu hóa hoạt động của chúng. Dưới đây là các yêu cầu cụ thể:

1. **(1.0 điểm)** **[CLO3]**

Viết câu lệnh SQL để lấy ra giá trị trung bình của dữ liệu cảm biến nhiệt độ từ các robot trong khoảng thời gian từ 1/1/2024 đến 31/1/2024. ( dữ liệu lấy từ các cảm biến nhiệt độ của bảng logs)

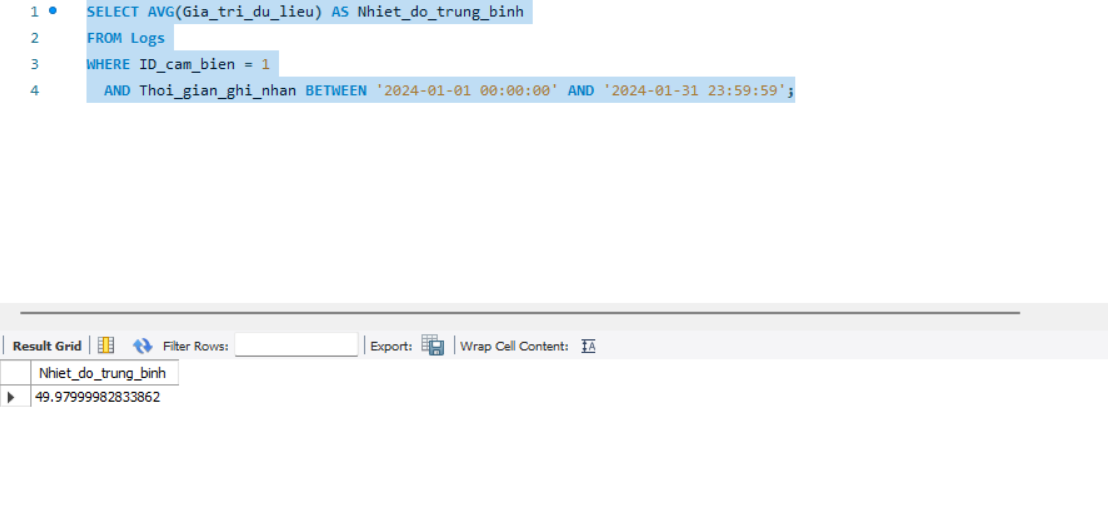


SELECT AVG(Gia\_tri\_du\_lieu) AS Nhiet\_do\_trung\_binh

FROM Logs

WHERE ID\_cam\_bien = 1

AND Thoi\_gian\_ghi\_nhan BETWEEN '2024-01-01 00:00:00' AND '2024-01-31 23:59:59';



1. **(1.0 điểm)** **[CLO4]**

Viết một stored procedure trong MySQL để tự động cập nhật trạng thái của robot dựa trên ngưỡng dữ liệu từ cảm biến (ví dụ: nếu nhiệt độ cảm biến vượt quá 75°C thì trạng thái robot được cập nhật thành "Overheated").

DELIMITER //

CREATE PROCEDURE UpdateRobotStatus()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE sensor\_id INT;

DECLARE temperature FLOAT;

DECLARE cur CURSOR FOR

SELECT ID\_cam\_bien, Gia\_tri\_du\_lieu

FROM Logs

WHERE Loai\_cam\_bien = 'Cảm biến nhiệt độ'

ORDER BY Thoi\_gian\_ghi\_nhan DESC;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO sensor\_id, temperature;

IF done THEN

LEAVE read\_loop;

END IF;

IF temperature > 75 THEN

UPDATE Robots r

INNER JOIN Sensors s ON r.ID\_robot = s.ID\_robot

SET r.Trang\_thai\_hien\_tai = 'Overheated'

WHERE s.ID\_cam\_bien = sensor\_id;

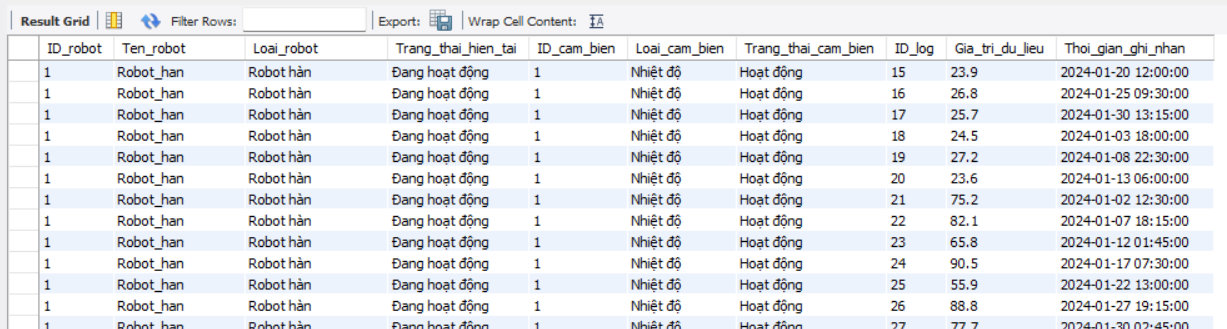
END IF;

END LOOP;

CLOSE cur;

END //

DELIMITER ;



**KẾT LUẬN**: Dù đã truy xuất được thông tin cảm biến nhiệt độ của robot nhưng khi nhiệt độ trên 75 độ thì vẫn chưa thay đổi được trạng thái từ “Hoạt động” sang “Overheated”.

1. **(1.0 điểm)** **[CLO5]**

Một robot tự động cần phản ứng ngay khi có thay đổi bất thường trong dữ liệu cảm biến. Hãy viết một trigger trong MySQL để gửi cảnh báo (cập nhật một bảng Alerts) khi giá trị cảm biến vượt ngưỡng giới hạn cho phép.

DELIMITER //

CREATE TRIGGER SendAlert

AFTER INSERT ON Logs

FOR EACH ROW

BEGIN

IF NEW.Gia\_tri\_du\_lieu > 75 THEN

INSERT INTO Alerts (

ID\_cam\_bien,

Gia\_tri\_du\_lieu,

Thoi\_gian\_ghi\_nhan,

Thong\_bao

) VALUES (

NEW.ID\_cam\_bien,

NEW.Gia\_tri\_du\_lieu,

NEW.Thoi\_gian\_ghi\_nhan,

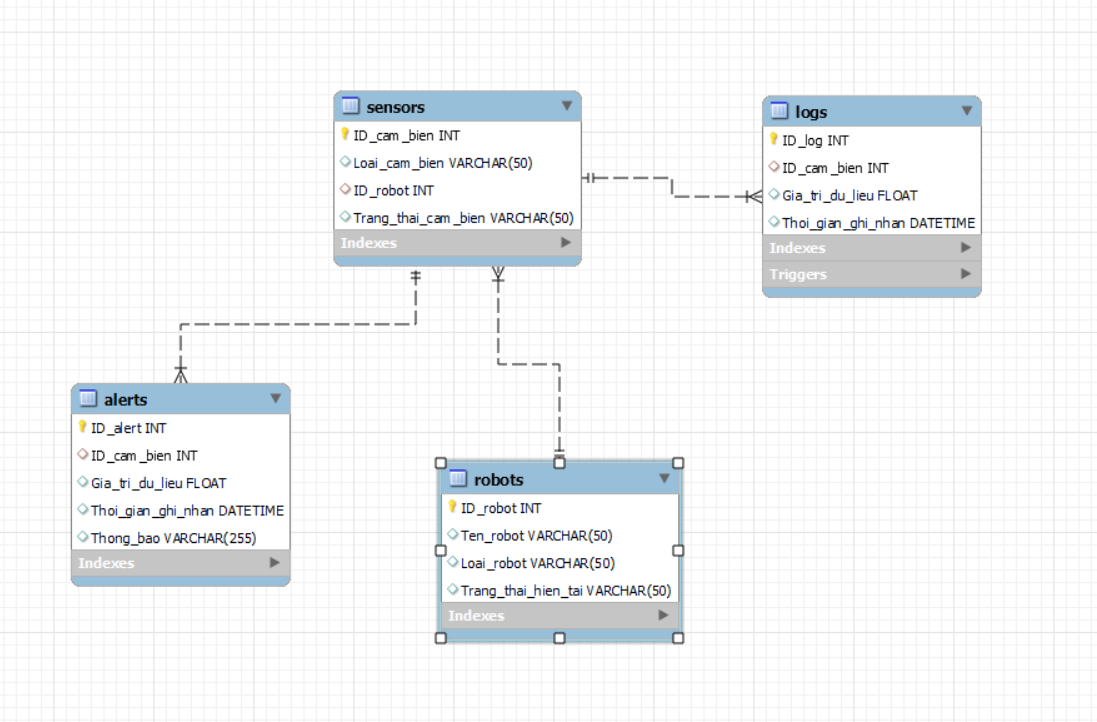
'Cảnh báo: Giá trị cảm biến vượt ngưỡng cho phép'

);

END IF;

END //

DELIMITER ;



**Phần III: Tối ưu hóa cơ sở dữ liệu MySQL (2.0 điểm)** *(CLO4)*

1. **(1.0 điểm)** Hãy nêu và giải thích ba chiến lược tối ưu hóa cơ sở dữ liệu MySQL để cải thiện hiệu suất trong việc lưu trữ và truy vấn dữ liệu lớn từ các hệ thống AI và Robot.

**Thiết kế hợp lý:**

Chuẩn hóa dữ liệu, chọn kiểu dữ liệu phù hợp, tạo chỉ mục hợp lý.

**Cấu hình MySQL:**

Điều chỉnh kích thước bộ nhớ đệm, cấu hình InnoDB, tùy chỉnh biến hệ thống.

**Tối ưu hóa truy vấn:**

Viết truy vấn hiệu quả, sử dụng EXPLAIN, tận dụng query cache.

1. **(1.0 điểm)** Trong hệ thống AI thời gian thực, việc truy vấn dữ liệu nhanh là rất quan trọng. Hãy mô tả cách tối ưu hóa các câu truy vấn SQL trong MySQL để tăng tốc độ phản hồi khi làm việc với dữ liệu lớn.

Sử dụng chỉ mục: Tạo chỉ mục cho các cột thường xuyên được truy vấn.

Tránh truy vấn toàn bảng: Sử dụng điều kiện lọc cụ thể.

Phân chia dữ liệu: Phân chia dữ liệu lớn thành các bảng con hoặc sử dụng phân vùng.

Sử dụng hàm và toán tử hiệu quả: Tránh các hàm phức tạp.

Giảm thiểu số lượng JOIN: Hạn chế các JOIN phức tạp.

Sử dụng chuẩn bị truy vấn: Giảm thiểu việc phân tích cú pháp truy vấn.

Hết