**CÂU HỎI ÔN TẬP CUỐI KỲ**

**MÔN: BẢO MẬT CƠ SỞ DỮ LIỆU**

**------o0o------**

1. **Sinh viên hãy giải thích vì sao điều khiển truy cập tùy quyền (Discretionary Access Control) có tính linh hoạt trong quản lý quyền truy cập. Cho ví dụ minh hoạ**

Discretionary Access Control (DAC) là một phương thức quản lý quyền truy cập trong hệ thống máy tính. Tính linh hoạt của DAC nằm ở khả năng cho phép chủ sở hữu tài nguyên (resource owner) quyết định việc cấp hoặc thu hồi quyền truy cập của người dùng hay nhóm người dùng khác để truy cập vào tài nguyên đó.

Điều này cho phép các chủ sở hữu tài nguyên tự do và linh hoạt quyết định về số lượng và quyền truy cập mà người dùng hay nhóm người dùng có được. Cho phép họ tùy ý cấp hoặc thu hồi quyền truy cập của một người dùng hoặc nhóm người dùng đến tài nguyên.Nó cung cấp cho chủ sở hữu tài nguyên khả năng quyết định và kiểm soát quyền truy cập của người dùng đến tài nguyên của họ.

**Ví dụ minh hoạ:** Trong một hệ thống máy tính, người dùng Alice và Bob là hai nhân viên cùng làm việc cho một công ty. Alice có quyền truy cập vào tất cả các tài liệu của phòng nhân sự, trong khi Bob chỉ có quyền truy cập vào một số tài liệu cụ thể. Với DAC, Alice có thể kiểm soát quyền truy cập của mình trên các tài nguyên trong phòng nhân sự, bao gồm việc chỉnh sửa, xóa hoặc chia sẻ các tài liệu của mình với các đồng nghiệp khác trong phòng nhân sự. Tương tự, Bob cũng có thể kiểm soát quyền truy cập của mình trên các tài nguyên mà anh ta được phân quyền truy cập, bao gồm việc đọc, chỉnh sửa hoặc chia sẻ các tài liệu của mình với những người khác được phép truy cập. Nếu Alice muốn chia sẻ một tài liệu trong phòng nhân sự với Bob, cô ấy có thể cấp quyền truy cập cho Bob đối với tài liệu đó hoặc cô ấy có thể đổi lại quyền truy cập của mình với tài liệu đó.

Tính linh hoạt của DAC cho phép người dùng quản lý quyền truy cập của mình một cách tự do và linh hoạt, tùy thuộc vào nhu cầu của họ và các quy định và quy trình của công ty. Tuy nhiên, điều này cũng có thể dẫn đến các vấn đề an ninh, nếu người dùng không quản lý và bảo vệ quyền truy cập của mình một cách cẩn thận.

1. **Sinh viên hãy trình bày về bài toán xác thực quyền hạn (Authorization) trong lĩnh vực bảo mật cơ sở dữ liệu. Cho ví dụ minh hoạ**

Bài toán xác thực quyền hạn (Authorization) trong lĩnh vực bảo mật cơ sở dữ liệu (Database Security) là quá trình kiểm tra và xác nhận quyền truy cập của người dùng vào cơ sở dữ liệu và các tài nguyên được bảo mật bên trong. Bao gồm việc đưa ra quyết định xem người dùng có được phép truy cập dữ liệu hay không, và nếu có, thì mức độ truy cập của họ là gì.

Các quyền hạn xác thực bao gồm quyền truy cập vào cơ sở dữ liệu, được phép thực thi câu lệnh truy vấn và thực hiện các thao tác dữ liệu như thêm, sửa, xóa. Các quyền hạn này được cấp phép bởi quản trị viên cơ sở dữ liệu hoặc người được ủy quyền.

**Ví dụ về bài toán xác thực quyền hạn trong lĩnh vực bảo mật cơ sở dữ liệu có thể là như sau:**

Giả sử một hệ thống quản lý khách sạn sử dụng cơ sở dữ liệu để lưu thông tin về khách hàng và các phòng khách sạn. Trong hệ thống này, quản trị viên của khách sạn có quyền truy cập vào toàn bộ cơ sở dữ liệu để quản lý thông tin các phòng và khách sạn, còn nhân viên lễ tân chỉ có quyền truy cập vào một phần thông tin cần thiết về khách hàng và phòng.

Để xác thực quyền hạn, hệ thống cần cung cấp cho mỗi người dùng riêng một tài khoản đăng nhập vào hệ thống và một bộ quyền hạn. Các đối tượng trong cơ sở dữ liệu được phân chia thành các nhóm khác nhau dựa trên tính năng và mức độ quản lý của nó. Mỗi nhóm đối tượng được gán một bộ quyền hạn riêng để kiểm soát truy cập và thao tác trên các đối tượng đó.

Với ví dụ này, hệ thống sẽ cung cấp cho quản trị viên toàn quyền truy cập vào các bảng trong cơ sở dữ liệu, cho phép họ cập nhật, thêm mới và xóa bỏ các thông tin khách hàng và phòng khách sạn. Trong khi đó, nhân viên lễ tân chỉ được phép truy cập vào một số bảng nhất định, chẳng hạn như bảng thông tin khách hàng và phòng, và các chức năng truy cập sửa đổi bị hạn chế để đảm bảo tính an toàn của dữ liệu.

Trong hệ thống này, việc áp dụng quyền hạn và xác thực quyền hạn rất quan trọng để đảm bảo tính bảo mật của thông tin khách hàng và phòng khách sạn. Một lỗ hổng bảo mật trong việc xác thực quyền hạn có thể dẫn đến các vấn đề bảo mật, truy cập trái phép, lộ thông tin hoặc vi phạm quy định pháp lý.

1. **Sinh viên hãy giải thích vì sao mô hình điều khiển truy cập bắt buộc (Mandatory Access Control) ít được vận dụng trong các ứng dụng thực tế. Cho ví dụ minh hoạ**

Mô hình điều khiển truy cập bắt buộc (Mandatory Access Control - MAC) là một hình thức bảo mật dữ liệu, trong đó các quyết định truy cập của người dùng đến tài nguyên được quản lý bởi hệ thống dựa trên các quy tắc được thiết lập từ trước. Trong mô hình MAC, mỗi tài nguyên được xác định bằng một mức độ bảo mật, và các người dùng và các quy trình được gắn nhãn với mức độ bảo mật tương ứng. Mô hình MAC khá khắt khe và ít được sử dụng trong các ứng dụng thực tế vì:

1. Tính phức tạp: Mô hình MAC yêu cầu thiết lập một hệ thống cấp phép chặt chẽ, bao gồm các quy tắc bảo mật được thiết lập từ trước, và việc thêm mới hoặc sửa đổi quy tắc rất phức tạp và tốn thời gian.

2. Không linh hoạt: Mô hình MAC không linh hoạt trong việc xác định quyền truy cập tài nguyên và phù hợp cho nhiều nhu cầu sử dụng khác nhau. Trong các ứng dụng thực tế, nhiều người dùng và quy trình có nhu cầu truy cập đến các tài nguyên khác nhau, và sử dụng mô hình MAC có thể gây ra nhiều rào cản.

**Ví dụ về mô hình điều khiển truy cập bắt buộc ít được áp dụng trong thực tế là:**

Trong môi trường hệ thống thông tin y tế của các bệnh viện, các loại thông tin nhạy cảm của bệnh nhân, chẳng hạn như thông tin y tế và tiền sử bệnh tật, phải được bảo vệ cẩn thận để đảm bảo quyền riêng tư của bệnh nhân. Một giải pháp bảo mật cho việc quản lý thông tin y tế được đề xuất là sử dụng mô hình điều khiển truy cập bắt buộc, nơi các thông tin được phân loại thành các cấp độ riêng biệt như thông tin nhạy cảm nhất, cao cấp, trung bình và thông tin cơ bản. Mỗi cấp độ bảo mật có các quy tắc quản lý truy cập được xác định bởi quản trị viên và không thể thay đổi bởi người dùng.

Tuy nhiên, mô hình điều khiển truy cập bắt buộc ít được sử dụng trong thực tế trong lĩnh vực thông tin y tế vì sự phức tạp của cấu hình, việc đánh giá được các cấp độ bảo mật và giám sát hoạt động truy cập thông tin. Thay vào đó, phương pháp kiểm soát truy cập sử dụng kiểm soát truy cập dựa trên vai trò (Role-based Access Control - RBAC) thường được sử dụng phổ biến cùng với công nghệ xác thực hai yếu tố để đảm bảo bảo mật thông tin y tế.

1. **Sinh viên hãy giải thích vì sao điều khiển truy cập tùy quyền không kiểm soát được luồng thông tin. Cho ví dụ minh hoạ //Hướng dẫn: xem ví dụ về cách tấn công Trojan**

**Điều khiển truy cập tùy quyền** là một công nghệ được sử dụng để giới hạn quyền truy cập vào các tài nguyên hệ thống, nhằm ngăn chặn việc truy cập trái phép hoặc lộ thông tin quan trọng.

Điều khiển truy cập tùy quyền chỉ giới hạn quyền truy cập vào các tài nguyên cụ thể và cho phép người dùng chỉ truy cập vào những thông tin mà họ được phép xem. Tuy nhiên, điều này không kiểm soát được luồng thông tin do các yếu tố như:

1. Các vấn đề về mã độc: Mã độc có thể truy cập vào tài nguyên mà người dùng không được phép truy cập và gửi thông tin đó đến các nơi khác.

2. Tấn công từ bên ngoài: Kẻ tấn công có thể xâm nhập vào hệ thống và truy cập vào các tài nguyên mà họ không được phép truy cập.

3. Lỗi hệ thống: Các lỗi trong hệ thống có thể cho phép người dùng truy cập vào các tài nguyên mà họ không được phép truy cập.

4. Sự nhầm lẫn: Người dùng có thể nhầm lẫn và truy cập vào các tài nguyên mà họ không được phép truy cập.

Vì vậy, điều khiển truy cập tùy quyền không đảm bảo rằng luồng thông tin sẽ được kiểm soát hoàn toàn và người dùng cần phải cẩn thận khi sử dụng các tài nguyên.

**Ví dụ,** một loại tấn công Trojan có tên là "keylogger" có thể được cài đặt trên máy tính của một người dùng mà không cần quyền truy cập cao. Keylogger sẽ ghi lại tất cả các phím được nhấn trên bàn phím và gửi thông tin này đến máy chủ từ xa. Do đó, dù cho điều khiển truy cập tùy quyền đã được cài đặt để hạn chế quyền truy cập vào tài khoản người dùng, nhưng keylogger vẫn có thể lấy được thông tin quan trọng thông qua luồng thông tin từ máy tính của người dùng đó.

**Ví dụ về điều khiển truy cập tùy quyền không kiểm soát được luồng thông tin đó là trong trường hợp của Trojan.**

Nhân viên mới vào làm của công ty vô tình nhận được email có chứa Trojan, do không nghi ngờ và không được đào tạo nghiệp vụ chuyên sâu nên đã mở và tải email xuống máy tính của công ty việc này đã vô tình làm cho Trojan xâm nhập vào máy tính và hệ thống của công ty.Thường thì Trojan được phát tán qua email lừa đảo hoặc được tải xuống từ các trang web không đáng tin cậy. Sau khi nó xâm nhập vào máy tính, Trojan có thể thực hiện các hoạt động không mong muốn như lấy trộm thông tin đăng nhập, quét mạng hoặc phát tán các chương trình độc hại khác.

Trojan có thể xâm nhập vào máy tính của nhân viên đó và lấy cắp các thông tin đăng nhập bằng cách chiếm đoạt các quyền truy cập của người dùng. Ví dụ, một Trojan có thể thực hiện một hành động quyền truy cập như một quản trị viên. Nếu nhân viên đó đang đăng nhập vào máy tính với tư cách quản trị viên, Trojan sẽ có quyền truy cập vào tất cả các tài nguyên của hệ thống, bao gồm các tài khoản người dùng và các tệp tin quan trọng khác.

Trong trường hợp này, Trojan được coi là một ví dụ điển hình của điều khiển truy cập tùy quyền không kiểm soát được luồng thông tin. Nó có thể truy cập và thực hiện các hoạt động không mong muốn mà không cần sự đồng ý hoặc kiểm soát của người dùng chính thức. Điều này giúp kẻ tấn công có thể mở khóa các thông tin nhạy cảm mà không phải vượt qua các rào cản bảo vệ thông tin khắt khe trong một hệ thống.

Để bảo vệ khỏi các tấn công Trojan và các hình thức tấn công khác, các tổ chức cần áp dụng các giải pháp bảo mật, bao gồm việc giám sát và kiểm soát truy cập, cài đặt phần mềm chống virus và bảo mật chặt chẽ hệ thống.

1. **Sinh viên hãy trình bày về bài toán xác thực người dùng (Authentication) trong lĩnh vực bảo mật cơ sở dữ liệu. Cho ví dụ minh hoạ**

**Bài toán xác thực người dùng (Authentication)** trong lĩnh vực bảo mật cơ sở dữ liệu (Database Security) là quá trình xác định danh tính người dùng và xác nhận liệu họ có quyền truy cập vào cơ sở dữ liệu hay không. Điều này đảm bảo rằng chỉ những người dùng được ủy quyền mới có thể truy cập và sửa đổi các thông tin nhạy cảm trong cơ sở dữ liệu. Quá trình xác thực sẽ đảm bảo tính bảo mật và các thông tin trong cơ sở dữ liệu sẽ không bị lộ ra hay bị sửa đổi bởi người dùng không được ủy quyền.

Các phương pháp xác thực người dùng có thể bao gồm tên đăng nhập và mật khẩu, xác minh vân tay, smart card, chứng chỉ số hoặc xác thực địa chỉ IP.

**Ví dụ:** khi một ngân hàng muốn đảm bảo rằng chỉ những người dùng được ủy quyền mới có thể truy cập vào hệ thống của họ để sử dụng các tính năng liên quan đến cơ sở dữ liệu của khách hàng.

Khi một khách hàng muốn truy cập vào tài khoản của mình, họ phải đăng nhập vào hệ thống. Quá trình đăng nhập này sẽ bao gồm việc yêu cầu người dùng nhập thông tin đăng nhập, chẳng hạn như tên đăng nhập và mật khẩu. Sau đó, hệ thống sẽ kiểm tra thông tin đăng nhập này với cơ sở dữ liệu của ngân hàng và xác thực rằng thông tin đăng nhập là hợp lệ.

Nếu thông tin là hợp lệ, khách hàng sẽ được đăng nhập vào hệ thống và truy cập vào các thông tin của tài khoản của mình. Nếu thông tin không hợp lệ, hệ thống sẽ từ chối truy cập và cho biết lý do tại sao.

Việc xác thực người dùng giúp ngăn chặn các hành vi trái phép hoặc đánh cắp thông tin bởi những người dùng không được phép truy cập vào hệ thống. Nó cũng giúp đảm bảo rằng chỉ có các người dùng được ủy quyền mới có thể truy cập vào các tính năng liên quan đến cơ sở dữ liệu của khách hàng, giúp bảo vệ thông tin của khách hàng được giữ an toàn và bảo mật.

1. **Sinh viên hãy trình bày về bài toán xác thực quyền hạn (Authorization) trong lĩnh vực bảo mật cơ sở dữ liệu. Cho ví dụ minh hoạ**

**Bài toán xác thực quyền hạn (Authorization)** trong lĩnh vực bảo mật cơ sở dữ liệu đó là quyết định liệu một người dùng có được phép truy cập vào một tài liệu hoặc tài nguyên nhất định trong cơ sở dữ liệu hay không, dựa trên nhóm quyền và chính sách bảo mật nào đó. Nó đảm bảo rằng chỉ những người dùng được ủy quyền mới có thể truy cập và sửa đổi các thông tin nhạy cảm trong cơ sở dữ liệu, trong khi các người dùng không được ủy quyền sẽ không thể truy cập hoặc thay đổi các thông tin trong cơ sở dữ liệu.

Phương pháp xác thực quyền hạn có thể bao gồm các phương pháp như roles-based access control, attribute-based access control và mandatory access control.

**Ví dụ.** Trong hệ thống quản lý thông tin bệnh nhân, nhân viên y tế được cấp một nhóm quyền cụ thể để truy cập và sửa đổi các thông tin bệnh nhân.

Thông tin của bệnh nhân trong hệ thống được bảo vệ bởi một chính sách an ninh mạnh, yêu cầu authentication hoặc xác thực danh tính trước khi xem thông tin, và authorization để xác định quyền truy cập cho nhân viên y tế. Chính sách này đảm bảo rằng chỉ những người được phép mới có thể truy cập và sửa đổi các thông tin bệnh nhân trong hệ thống.

Khi nhân viên y tế muốn truy cập thông tin của một bệnh nhân cụ thể, họ sẽ được yêu cầu đăng nhập vào hệ thống và xác thực danh tính của họ. Sau đó, hệ thống sẽ kiểm tra nhóm quyền được cấp cho họ và cho phép họ truy cập vào thông tin của bệnh nhân. Nếu nhân viên y tế không được cấp quyền truy cập vào một bệnh nhân cụ thể, họ sẽ không thể xem thông tin hoặc sửa đổi các thông tin bệnh nhân đó.

Việc sử dụng phương pháp xác thực quyền hạn trong bảo mật cơ sở dữ liệu bệnh nhân này rất quan trọng, nó đảm bảo rằng thông tin của bệnh nhân sẽ ít bị lộ ra, bị sửa đổi hoặc bị truy cập bởi những người không được ủy quyền và đảm bảo phiên làm việc truy cập thông tin của nhân viên y tế sẽ được ghi lại.

1. **Sinh viên hãy trình bày về bài toán kiểm toán truy cập (Auditing) trong lĩnh vực bảo mật cơ sở dữ liệu. Cho ví dụ minh hoạ**

**Bài toán kiểm toán truy cập (Auditing)** trong lĩnh vực bảo mật cơ sở dữ liệu đó là quá trình theo dõi và ghi lại các hoạt động truy cập vào cơ sở dữ liệu từ các người dùng, nhất là những truy cập không được ủy quyền hoặc đầy nguy cơ. Nó giúp đảm bảo an toàn và bảo mật thông tin trong cơ sở dữ liệu.

Các yếu tố được lưu trữ trong quá trình kiểm toán truy cập bao gồm người dùng, hoạt động thực hiện, thời điểm, nơi thực hiện và kết quả của hoạt động. Các thông tin này sẽ giúp quản trị viên tổng hợp các hoạt động truy cập, xác định các hành vi khả nghi hay gian lận và giúp phát hiện các cuộc tấn công cũng như giải quyết các vấn đề kỹ thuật trong hệ thống.

**Ví dụ:** Khi một ngân hàng muốn theo dõi các hoạt động truy cập của nhân viên vào cơ sở dữ liệu của khách hàng. Đây là để đảm bảo rằng thông tin của khách hàng được giữ an toàn và các hoạt động truy cập vào cơ sở dữ liệu của họ là hợp lệ và được ủy quyền.

Khi một nhân viên ngân hàng đăng nhập vào hệ thống cơ sở dữ liệu của khách hàng, một bản ghi sẽ được tạo để theo dõi hoạt động của nhân viên đó. Bản ghi này sẽ bao gồm các thông tin liên quan đến nhân viên, thời gian truy cập, các hoạt động được thực hiện, như xem thông tin khách hàng, sửa đổi thông tin hoặc tạo tài khoản mới.

Việc ghi lại các hoạt động này cho phép ngân hàng theo dõi và phát hiện các hành vi đáng ngờ. Nếu một hoạt động không hợp lệ được thực hiện, ví dụ như nhân viên truy cập thông tin của khách hàng không được ủy quyền hoặc thực hiện một hành vi không đúng quy trình, ngân hàng có thể tìm ra nguyên nhân của sự cố và đưa ra giải pháp để khắc phục tình trạng đó và ngăn chặn lại xảy ra trong tương lai.

Ví dụ này cho thấy rằng phương pháp kiểm toán truy cập rất quan trọng để đảm bảo an toàn và bảo mật thông tin của khách hàng. Nó giúp phát hiện và ngăn chặn các hoạt động không hợp lệ hoặc đáng ngờ cũng như giúp giải quyết các vấn đề kỹ thuật trong hệ thống cơ sở dữ liệu.

1. **Giải thích tính “tùy ý” trong mô hình điều khiển truy cập tùy ý? Trong mô hình điều khiển truy cập tùy ý, những loại người dùng nào có quyền phân quyền/thu hồi quyền hạn? Trong mô hình điều khiển truy cập tùy ý, tại sao ưu điểm là tính linh hoạt, tính dễ dàng.**

**Tính "tùy ý"** trong mô hình điều khiển truy cập tùy ý (ABAC - Attribute-Based Access Control) đề cập đến khả năng kiểm soát truy cập dựa trên các thuộc tính và quy tắc định nghĩa trước. Nó cho phép các quyền truy cập của một đối tượng đến một tài nguyên trong hệ thống được xác định thông qua các thuộc tính liên quan đến đối tượng, tài nguyên và môi trường.

**Với tính "tùy ý",** các quản trị viên hoặc người sử dụng có thể xác định các quyền truy cập dựa trên các ràng buộc logic phức tạp hơn so với các hệ thống điều khiển truy cập truyền thống như RBAC (Mô hình kiểm soát truy cập dựa trên vai trò). Các quy tắc và thuộc tính cho phép định nghĩa các trường hợp đối tượng nào có quyền truy cập đến tài nguyên nào và thuộc tính nào sẽ được xác định trong quá trình đó.

Trong mô hình điều khiển truy cập tùy ý (ABAC- Attribute-Based Access Control), có thể xác định nhiều **người dùng có quyền phân quyền hoặc thu hồi quyền hạn** trong hệ thống để thực hiện kiểm soát truy cập trên cơ sở các thuộc tính.

Các quản trị viên hoặc người sử dụng có đủ đặc quyền đặc biệt có quyền phân quyền hoặc thu hồi quyền hạn trên các đối tượng ở cấp độ thấp hơn của hệ thống. Họ sẽ định nghĩa các nhóm người dùng tùy ý và kết hợp các thuộc tính phù hợp với từng nhóm người dùng.

Người dùng cũng có thể có quyền hạn để quản lý quyền truy cập dựa trên chức năng hoặc cấp độ truy cập của họ. Ví dụ, người dùng có thể cấp quyền truy cập vào một tài liệu cụ thể cho một đối tượng hoặc người dùng để sửa đổi hoặc xóa tài liệu đó.

Do đó, trong mô hình điều khiển truy cập tùy ý ABAC, các quản trị viên hoặc người sử dụng có đủ đặc quyền đặc biệt và người sử dụng cuối có thể có quyền đưa ra quyết định phân quyền/truy cập trên cơ sở các thuộc tính của hệ thống để đáp ứng yêu cầu kinh doanh khác nhau.

**Tính linh hoạt và dễ dàng** trong mô hình điều khiển truy cập tùy ý ABAC (Attribute-Based Access Control) nằm ở cách thức quản lý quyền truy cập được định nghĩa dựa trên các thuộc tính và quy tắc định nghĩa trước.

Các thuộc tính, quy tắc và môi trường giúp cho quyền truy cập có thể được cấu hình linh hoạt hơn và dễ dàng hơn, do đó giúp cho công việc cấu hình và quản lý quyền truy cập đơn giản và dễ dàng hơn. Các loại thuộc tính khác nhau, từ các thuộc tính kiểu số đến các thuộc tính văn bản và cả các thuộc tính động, có thể được sử dụng để xác định quyền truy cập.

Các đối tượng và tài nguyên có thể được nhóm lại dựa trên các thuộc tính quy định trước để đáp ứng các mục đích khác nhau. Nhóm người dùng và quy tắc truy cập có thể được thiết lập tự động cho từng đối tượng trong cùng một nhóm, giúp giảm thời gian cấu hình và giảm thiểu sự nhầm lẫn.

Do đó, tính linh hoạt và dễ dàng trong mô hình điều khiển truy cập tùy ý ABAC giúp cho việc quản lý quyền truy cập trở nên dễ dàng và nhanh chóng hơn, đồng thời giúp cho hệ thống có thể dễ dàng thích nghi và phù hợp với các yêu cầu kinh doanh khác nhau.

1. **Tại sao khi cài đặt, người ta sử dụng danh sách khả năng thay thế cho cấu trúc ma trận phân quyền?**

Khi cài đặt hệ thống phân quyền, người ta thường sử dụng danh sách khả năng thay thế cho cấu trúc ma trận phân quyền để đơn giản hóa quá trình quản lý và cập nhật quyền truy cập của người dùng.

Việc sử dụng danh sách khả năng thay thế cho cấu trúc ma trận phân quyền khi cài đặt nhằm mục đích giúp người quản trị hệ thống có thể linh hoạt thay đổi quyền truy cập của người dùng một cách dễ dàng. Khi sử dụng danh sách khả năng thay thế, người quản trị có thể thêm hoặc xóa các quyền truy cập một cách nhanh chóng và dễ dàng hơn so với việc phải sửa đổi cấu trúc ma trận phân quyền trực tiếp. Ngoài ra, danh sách khả năng thay thế cũng giúp giảm thiểu rủi ro khi thay đổi cấu trúc ma trận phân quyền bằng cách đảm bảo rằng các quyền truy cập được thêm hoặc xóa chỉ được thực hiện bởi người quản trị hệ thống có đủ quyền hạn.

Danh sách khả năng thay thế cho phép quyền truy cập được kiểm soát theo cách tổng quát hơn và dễ dàng chỉnh sửa quyền truy cập của đối tượng mà không phải tạo lại ma trận to lớn. Do đó, việc sử dụng danh sách khả năng thay thế cho cấu trúc ma trận phân quyền giúp cho việc quản lý quyền truy cập trong mô hình điều khiển truy cập tùy ý ABAC trở nên linh hoạt và dễ dàng hơn.

1. **Giải thích tính “bắt buộc” trong mô hình điều khiển truy cập tùy ý? Trong mô hình điều khiển truy cập bắt buộc, tại sao nó ít được vận dụng trong thực tế? Tại sao mô hình điều khiển truy cập bắt buộc được áp dụng trong môi trường quân đội. Trong mô hình điều khiển truy cập bắt buộc, tại sao nó ít mang tính nghiêm ngặt/khắc khe?**

**Tính "bắt buộc"** trong mô hình điều khiển truy cập tùy ý (ABAC - Attribute-Based Access Control) ám chỉ việc quyền truy cập đối với tài nguyên được quy định bởi các thuộc tính và các quy tắc được định nghĩa trước đó. Tính bắt buộc ở đây nghĩa là một đối tượng chỉ có thể truy cập tài nguyên nếu đáp ứng đủ các điều kiện được quản lý đầy đủ và nghiêm ngặt.

Các đối tượng có thể được đánh dấu bởi các thuộc tính về danh tính, quyền hạn và vai trò của họ. Tài nguyên cũng có thể được xác định và phân loại dựa trên các thuộc tính liên quan đến tính bảo mật, như kiểu, mức độ nhạy cảm và cấp độ truy cập.

Một cách hoạt động của tính bắt buộc trong mô hình ABAC là đối tượng được phép truy cập tài nguyên nếu có đủ các thuộc tính yêu cầu do quản trị viên đặt ra. Nếu bất kỳ một thuộc tính nào không được đáp ứng, quyền truy cập sẽ bị từ chối.

Tính "bắt buộc" trong mô hình ABAC cung cấp một cách tiếp cận bảo mật linh hoạt và tùy chỉnh cao hơn với khả năng kiểm soát được quyền truy cập bằng các yêu cầu về thuộc tính. Do đó, hệ thống sử dụng mô hình này có thể quản lý được quyền truy cập đến các tài nguyên, giúp bảo vệ thông tin và giảm thiểu rủi ro an ninh một cách rõ rệt.

Mô hình điều khiển truy cập bắt buộc là một mô hình bảo mật rất chặt chẽ và nghiêm ngặt, nó đòi hỏi người dùng phải được xác thực và được cấp quyền truy cập rõ ràng để có thể truy cập vào các tài nguyên. Tuy nhiên, mô hình này **ít được vận dụng trong thực tế** vì một số lý do sau đây:

1. Tính phức tạp: Mô hình điều khiển truy cập bắt buộc rất phức tạp và đòi hỏi một số kỹ thuật và kiến thức chuyên môn để thiết lập và quản lý. Nó đòi hỏi người quản trị phải có kiến thức về các cơ chế an ninh, các chính sách và quy trình quản lý, và các công nghệ bảo mật.

2. Tính linh hoạt: Mô hình này ít linh hoạt để thích ứng với các yêu cầu thay đổi của môi trường do tính chất nghiêm ngặt và chặt chẽ của nó. Nếu các quy tắc và chính sách không được cập nhật đầy đủ, nó có thể dẫn đến các vấn đề an ninh và ảnh hưởng đến hiệu suất của hệ thống.

3. Tính tiện lợi: Mô hình điều khiển truy cập bắt buộc có thể làm giảm tính tiện lợi và tốc độ truy cập của người dùng. Nó đòi hỏi người dùng phải xác thực nhiều lần trước khi truy cập vào các tài nguyên và có thể tạo ra sự khó chịu và giảm hiệu suất làm việc.

4. Chi phí: Mô hình điều khiển truy cập bắt buộc đòi hỏi phải có các công nghệ bảo mật cao cấp và các kỹ thuật quản lý chuyên nghiệp, điều này có thể tăng chi phí đầu tư ban đầu và chi phí vận hành hệ thống.

Vì những lý do này, mô hình điều khiển truy cập bắt buộc ít được sử dụng trong thực tế và thường chỉ được áp dụng trong các môi trường an ninh cao hoặc các tổ chức quân sự, chính phủ, hoặc các doanh nghiệp có nhu cầu bảo mật cao.

Mô hình điều khiển truy cập bắt buộc được **áp dụng trong môi trường quân đội vì** tính bảo mật và quản lý cao của nó. Trong quân đội, các thông tin và tài nguyên quan trọng cần được bảo vệ khỏi những người dùng không có quyền truy cập. Mô hình này đảm bảo rằng chỉ những người dùng được phép truy cập vào các tài nguyên và thông tin quan trọng, giúp đảm bảo tính bảo mật của hệ thống.

Mô hình điều khiển truy cập bắt buộc cũng cho phép quản lý quyền truy cập được thực hiện một cách chặt chẽ và có tính linh hoạt cao, giúp quản trị viên có thể quản lý và kiểm soát quyền truy cập của người dùng một cách hiệu quả. Ngoài ra, mô hình này cũng giúp các tổ chức quân đội đáp ứng được các yêu cầu bảo mật của chính phủ và các cơ quan quản lý khác.

Mô hình MAC giúp đảm bảo tính bảo mật trên các hệ thống thông tin quan trọng của quân đội. Tại đây, quyền truy cập của người dùng được kiểm soát chặt chẽ thông qua việc phân loại các tài nguyên, chia ra các cấp độ truy cập và áp dụng các quy tắc chặt chẽ về bảo mật.

Mô hình MAC cũng giúp đảm bảo tính thống nhất và nhất quán của quy tắc bảo mật trong toàn bộ môi trường quân đội. Các quy tắc và giới hạn an ninh được áp dụng đồng nhất trên toàn hệ thống thông tin, giúp duy trì tính bảo mật và giảm thiểu rủi ro an ninh một cách rõ rệt.

Mô hình điều khiển truy cập bắt buộc là một mô hình bảo mật rất chặt chẽ và nghiêm ngặt, nó đòi hỏi người dùng phải được xác thực và được cấp quyền truy cập rõ ràng để có thể truy cập vào các tài nguyên. Tuy nhiên, mô hình này **ít mang tính nghiêm ngặt/khắc khe** vì một số lý do sau đây:

1. Tùy chỉnh được: Mô hình này có thể được tùy chỉnh để phù hợp với nhu cầu của mỗi tổ chức. Nó cho phép người quản trị có thể thiết lập các chính sách và quy tắc truy cập cụ thể cho từng tài nguyên, nhóm người dùng hoặc người dùng riêng lẻ. Điều này giúp tăng tính linh hoạt và giảm bớt tính nghiêm ngặt/khắc khe của mô hình.

2. Tính dễ dàng: Mô hình này có tính dễ dàng trong việc triển khai và quản lý. Các công nghệ bảo mật như xác thực hai yếu tố (2FA), mã hóa và chữ ký số được tích hợp để giúp đơn giản hóa quá trình xác thực và cấp quyền truy cập.

3. Tính linh hoạt: Mô hình này cho phép người quản trị có thể tùy chỉnh các quy tắc và chính sách truy cập để phù hợp với nhu cầu của từng người dùng. Điều này giúp tăng tính linh hoạt và giảm bớt tính nghiêm ngặt/khắc khe của mô hình.

4. Hiệu quả: Mô hình này có hiệu quả trong việc đảm bảo an ninh thông tin và giảm thiểu các lỗ hổng an ninh. Nó giúp hạn chế sự truy cập trái phép vào các tài nguyên quan trọng và giảm thiểu các mối đe dọa bảo mật.

Vì những lý do này, mô hình điều khiển truy cập bắt buộc ít mang tính nghiêm ngặt/khắc khe và có thể được tùy chỉnh để phù hợp với nhu cầu của từng tổ chức.

1. **Ý tưởng trọng tâm của mô hình RBAC là gì? Vì sao RBAC hỗ trợ nguyên tắc bảo mật: đặc quyền ít nhất (Least Privilege)? Vì sao RBAC hỗ trợ nguyên tắc bảo mật: sự tách biệt các nhiệm vụ (Separation of duties). Vì sao RBAC hỗ trợ nguyên tắc bảo mật: trừu tượng hóa dữ liệu (Data Abstraction).**

**Ý tưởng trọng tâm** của mô hình RBAC (Role-Based Access Control) là quản lý quyền truy cập vào tài nguyên dựa trên vai trò của người dùng. RBAC xác định các vai trò và quyền truy cập cho từng vai trò, sau đó phân bổ các vai trò cho người dùng. Người dùng chỉ có thể truy cập vào các tài nguyên được phép dựa trên vai trò của họ.

RBAC hỗ trợ nguyên tắc bảo mật **đặc quyền ít nhất** bởi vì nó cho phép quản trị viên chỉ định quyền truy cập tối thiểu cần thiết để người dùng hoặc vai trò thực hiện các nhiệm vụ cụ thể. Việc áp dụng RBAC giúp giảm thiểu rủi ro bảo mật bằng cách đảm bảo rằng các người dùng chỉ có quyền truy cập đến các tài nguyên và chức năng cần thiết cho công việc của họ. Điều này giúp ngăn chặn các cuộc tấn công bảo mật và giảm thiểu thiệt hại trong trường hợp xảy ra sự cố bảo mật. Ngoài ra, RBAC cũng giúp quản lý và bảo trì hệ thống bảo mật một cách hiệu quả hơn bằng cách giảm thiểu sự phức tạp trong quản lý quyền truy cập.

RBAC hỗ trợ nguyên tắc bảo mật **sự tách biệt các nhiệm vụ** bởi vì nó cho phép quản trị viên phân chia các quyền và vai trò sao cho các nhiệm vụ liên quan đến việc quản lý tài nguyên và thực hiện các hoạt động khác nhau được phân tách rõ ràng. Điều này giúp ngăn chặn các cuộc tấn công bảo mật bằng cách đảm bảo rằng không có người dùng nào có quyền truy cập đầy đủ đến các tài nguyên và chức năng của hệ thống. Nó cũng giúp giảm thiểu các rủi ro bảo mật do lỗi trong việc quản lý quyền truy cập bằng cách đảm bảo rằng các quyền truy cập được phân chia và kiểm soát một cách hiệu quả. Ngoài ra, RBAC cũng giúp tạo ra một mô hình quản lý quyền truy cập đơn giản và dễ dàng bảo trì, giúp tăng cường độ tin cậy và tính khả dụng của hệ thống.

RBAC hỗ trợ nguyên tắc bảo mật **trừu tượng hóa dữ liệu** bởi vì nó cho phép quản trị viên chỉ định quyền truy cập dựa trên các vai trò thay vì trực tiếp đến các đối tượng hoặc người dùng cụ thể. Việc áp dụng RBAC giúp giảm thiểu rủi ro bảo mật do lỗi trong việc quản lý quyền truy cập cho các đối tượng cụ thể. Ngoài ra, RBAC cũng giúp tạo ra một mô hình quản lý quyền truy cập trừu tượng hóa, giúp dễ dàng quản lý và bảo trì hệ thống bảo mật.

1. **Điểm đặc trưng của mô hình Hierarchical RBAC là gì? Điểm đặc trưng của mô hình Static Separation of Duty Relations là gì? Điểm đặc trưng của mô hình Dynamic Separation of Duty Relations là gì?**

Mô hình **Hierarchical RBAC** là một phiên bản của RBAC, trong đó các vai trò được tổ chức theo cấp độ hiện tại của chúng. Các vai trò ở cấp độ cao hơn có quyền truy cập vào các tài nguyên và chức năng của các vai trò ở cấp độ thấp hơn. Điểm đặc trưng của mô hình này bao gồm:

* + Tổ chức vai trò theo cấp độ: Các vai trò được tổ chức theo cấp độ khác nhau, với các vai trò ở cấp độ cao hơn có quyền truy cập vào các tài nguyên và chức năng của các vai trò ở cấp độ thấp hơn.
  + Quản lý quyền truy cập linh hoạt: Mô hình Hierarchical RBAC cho phép quản trị viên dễ dàng thêm hoặc xóa các vai trò và quyền truy cập tương ứng, giúp tăng tính linh hoạt và hiệu quả trong việc quản lý quyền truy cập.
  + Kiểm soát quyền truy cập chính xác: Mô hình này giúp đảm bảo rằng các người dùng chỉ có quyền truy cập đến các tài nguyên và chức năng cần thiết cho công việc của họ, giúp giảm thiểu các rủi ro bảo mật.
  + Tính mở rộng cao: Mô hình Hierarchical RBAC có tính mở rộng cao, cho phép quản trị viên dễ dàng thêm hoặc xóa các cấp độ và các vai trò tương ứng, giúp tăng tính linh hoạt và hiệu quả trong việc quản lý quyền truy cập.
  + Tính phân cấp rõ ràng: Mô hình này giúp định rõ các mối quan hệ giữa các vai trò và cấp độ tương ứng, giúp người dùng và quản trị viên dễ dàng hình dung và hiểu được các quyền và trách nhiệm của từng vai trò.

Mô hình **Static Separation of Duty Relations** là một phiên bản của RBAC, trong đó các quy tắc tách biệt các nhiệm vụ được thiết lập trước và không thay đổi trong quá trình hoạt động của hệ thống. Điểm đặc trưng của mô hình này bao gồm:

* Thiết lập quy tắc tách biệt các nhiệm vụ trước: Trong mô hình này, các quy tắc tách biệt các nhiệm vụ được thiết lập trước và không thay đổi trong quá trình hoạt động của hệ thống. Điều này giúp đảm bảo tính chính xác và đáng tin cậy của quyền truy cập.
* Kiểm soát quyền truy cập chính xác: Mô hình Static Separation of Duty Relations giúp đảm bảo rằng các người dùng chỉ có quyền truy cập đến các tài nguyên và chức năng cần thiết cho công việc của họ, giúp giảm thiểu các rủi ro bảo mật.
* Giảm thiểu sự phức tạp trong quản lý quyền truy cập: Mô hình này giúp đơn giản hóa việc quản lý quyền truy cập bằng cách thiết lập các quy tắc tách biệt các nhiệm vụ trước, giảm thiểu sự phức tạp trong việc quản lý và bảo trì hệ thống bảo mật.
* Tính linh hoạt thấp: Mô hình Static Separation of Duty Relations không cho phép thay đổi các quy tắc tách biệt các nhiệm vụ trong quá trình hoạt động của hệ thống, giới hạn tính linh hoạt trong việc quản lý quyền truy cập.
* Không thích hợp cho các hệ thống phức tạp: Mô hình này thích hợp cho các hệ thống đơn giản hoặc có số lượng người dùng ít. Đối với các hệ thống phức tạp hoặc có số lượng người dùng lớn, mô hình này có thể không đáp ứng được các yêu cầu về quản lý quyền truy cập.

Mô hình **Dynamic Separation of Duty Relations** là một phiên bản của RBAC, trong đó các quy tắc tách biệt các nhiệm vụ được thiết lập và thay đổi trong quá trình hoạt động của hệ thống. Điểm đặc trưng của mô hình này bao gồm:

* Thiết lập và thay đổi quy tắc tách biệt các nhiệm vụ trong quá trình hoạt động: Mô hình Dynamic Separation of Duty Relations cho phép quản trị viên thiết lập và thay đổi các quy tắc tách biệt các nhiệm vụ trong quá trình hoạt động của hệ thống. Điều này giúp tăng tính linh hoạt và hiệu quả trong việc quản lý quyền truy cập.
* Kiểm soát quyền truy cập chính xác: Mô hình này giúp đảm bảo rằng các người dùng chỉ có quyền truy cập đến các tài nguyên và chức năng cần thiết cho công việc của họ, giúp giảm thiểu các rủi ro bảo mật.
* Tính linh hoạt cao: Mô hình Dynamic Separation of Duty Relations có tính linh hoạt cao, cho phép quản trị viên tùy chỉnh các quy tắc tách biệt các nhiệm vụ dựa trên các yêu cầu cụ thể của hệ thống và các người dùng.
* Tính phân cấp rõ ràng: Mô hình này giúp định rõ các mối quan hệ giữa các quy tắc tách biệt các nhiệm vụ và các người dùng tương ứng, giúp người dùng và quản trị viên dễ dàng hình dung và hiểu được các quyền và trách nhiệm của từng người dùng.
* Giảm thiểu sự phức tạp trong quản lý quyền truy cập: Mô hình này giúp đơn giản hóa việc quản lý quyền truy cập bằng cách cho phép quản trị viên tùy chỉnh các quy tắc tách biệt các nhiệm vụ trong quá trình hoạt động của hệ thống.
* Tính phù hợp cho các hệ thống phức tạp: Mô hình Dynamic Separation of Duty Relations thích hợp cho các hệ thống phức tạp hoặc có số lượng người dùng lớn, giúp đáp ứng các yêu cầu về quản lý quyền truy cập.

1. **So sánh điểm giống/khác nhau giữa hai khái niệm kế thừa vai trò tổng quát (General Role Hierarchies) và kế thừa vai trò bị giới hạn (Limited Role Hierarchies).**

**Giống nhau:**

- Cả kế thừa vai trò tổng quát và kế thừa vai trò bị giới hạn đều là các kỹ thuật được sử dụng trong RBAC để quản lý quyền truy cập.

- Cả hai kỹ thuật đều cho phép các vai trò con kế thừa các quyền và truy cập của các vai trò cha.

**Khác nhau:**

- Kế thừa vai trò tổng quát cho phép các vai trò con kế thừa tất cả các quyền và truy cập của các vai trò cha, trong khi kế thừa vai trò bị giới hạn chỉ cho phép các vai trò con kế thừa một phần các quyền và truy cập của các vai trò cha.

- Kế thừa vai trò tổng quát cho phép các vai trò con thừa hưởng các quyền và truy cập của tất cả các vai trò cha trong cùng hệ thống, trong khi kế thừa vai trò bị giới hạn chỉ cho phép các vai trò con thừa hưởng các quyền và truy cập của các vai trò cha trong cùng nhóm hoặc cùng loại.

- Kế thừa vai trò tổng quát có tính linh hoạt cao hơn và cho phép quản trị viên tùy chỉnh các quyền và truy cập theo cấu trúc phân cấp, trong khi kế thừa vai trò bị giới hạn có tính linh hoạt thấp hơn và không cho phép quản trị viên tùy chỉnh quyền và truy cập của các vai trò con.

- Kế thừa vai trò tổng quát có tính phức tạp cao hơn trong việc quản lý và bảo trì hệ thống, trong khi kế thừa vai trò bị giới hạn đơn giản hơn trong việc quản lý và bảo trì hệ thống.

1. **Trong mô hình RBAC, xung đột (Conflit) quyền hạn là gì? Bạn hãy nêu một chiến lược (strategy) hoặc giải pháp (solution) giải quyết xung đột.**

Trong mô hình RBAC (Role-Based Access Control), **xung đột quyền hạn** xảy ra khi một người dùng được gán nhiều hơn một vai trò hoặc nhóm và mỗi vai trò hoặc nhóm có các quyền hạn khác nhau, dẫn đến sự nghi ngờ hoặc tranh chấp quyền truy cập vào các tài nguyên..

Một **chiến lược giải quyết xung đột** là sử dụng quy tắc ưu tiên (priority rules) để giải quyết xung đột. Quy tắc ưu tiên có thể được thiết lập để xác định vai trò nào sẽ có quyền hạn cao hơn trong trường hợp xảy ra xung đột. Ví dụ, nếu một người dùng có vai trò quản trị viên và vai trò nhân viên, quy tắc ưu tiên có thể được thiết lập để cho vai trò quản trị viên có quyền hạn cao hơn trong trường hợp xảy ra xung đột.

Một **giải pháp khác** là sử dụng phân quyền chi tiết (fine-grained authorization) để giải quyết xung đột. Phân quyền chi tiết cho phép các quyền hạn được chia nhỏ và xác định rõ ràng cho từng tài nguyên, giúp giảm thiểu xung đột quyền hạn. Ví dụ, nếu một người dùng có vai trò quản trị viên và vai trò nhân viên, quyền hạn truy cập vào từng tài nguyên có thể được xác định rõ ràng cho từng vai trò để giảm thiểu xung đột.

1. Giải quyết xung đột bằng cách ưu tiên quyền hạn cao: Trong trường hợp xung đột, quyền hạn cao hơn sẽ được ưu tiên. Người dùng sẽ truy cập các tài nguyên được gán cho vai trò hoặc nhóm có quyền hạn cao hơn.
2. Kết hợp các quyền hạn từ các vai trò hoặc nhóm khác nhau: Nếu các vai trò hoặc nhóm của một người dùng có xung đột, người quản trị có thể kết hợp các quyền hạn từ các vai trò hoặc nhóm khác nhau để giải quyết xung đột và đảm bảo rằng người dùng có quyền truy cập đầy đủ.
3. Sử dụng cấp độ ưu tiên: Trong RBAC, quản trị viên có thể cấp độ ưu tiên cho các vai trò hoặc nhóm. Nếu xung đột xảy ra, người dùng sẽ được cấp độ ưu tiên cao hơn để tránh mâu thuẫn các quyền hạn của các vai trò hoặc nhóm khác nhau.
4. Sử dụng quy tắc ưu tiên xác định truy cập: Nếu xung đột vẫn còn tồn tại, quản trị viên có thể sử dụng các quy tắc ưu tiên cụ thể để xác định quyền truy cập của người dùng tới các tài nguyên.
5. **Trình bày các phương pháp bảo mật CSDL mà bạn biết**

Dưới đây là các phương pháp bảo mật Cơ sở dữ liệu (CSDL) mà tôi biết:

1. Mã hóa dữ liệu (Encryption): Phương pháp này dùng để bảo vệ dữ liệu khỏi việc truy cập trái phép bằng cách mã hóa dữ liệu sử dụng thuật toán mã hóa. Việc giải mã yêu cầu mật khẩu hoặc khóa bí mật.

2. Giám sát (Auditing): Nó bao gồm việc chứng nhận các tương tác của người dùng với cơ sở dữ liệu tới mục đích giám sát các hành động người dùng, từ đó phát hiện và ngăn chặn các hành vi trái phép.

3. Xác thực người dùng (Authentication): Phương pháp này đảm bảo rằng chỉ những người được ủy quyền mới có thể truy cập vào cơ sở dữ liệu. Các phương pháp xác thực như truyền thông tin tài khoản và mật khẩu, sử dụng mã thông báo, chứng chỉ số...

4. Phân quyền (Authorization): Phương pháp này cho phép các người dùng được ủy quyền được phép truy cập chỉ vào các phần của cơ sở dữ liệu mà họ có quyền truy cập, hạn chế truy cập các phần mà họ không được phép.

5. Tạo sao lưu (Backup and Recovery): Phương pháp này đảm bảo tính toàn vẹn của cơ sở dữ liệu bằng cách thực hiện sao lưu định kỳ và khôi phục cơ sở dữ liệu nếu gặp sự cố.

6. Tường lửa (Firewall): Phương pháp này sử dụng firewall để ngăn chặn các địa chỉ IP truy cập vào cơ sở dữ liệu, và ngăn chặn các tấn công từ xa vào cơ sở dữ liệu.

7. Kiểm tra quy trình (Process Review): Phương pháp này đảm bảo rằng các thủ tục được áp dụng đầy đủ, tất cả những người được đảm bảo quyền truy cập là chính họ.

8. Đảm bảo tính toàn vẹn dữ liệu: Các cơ chế kiểm tra tính toàn vẹn dữ liệu được sử dụng để đảm bảo rằng dữ liệu không bị thay đổi hoặc xâm nhập trong quá trình truyền và lưu trữ.

1. **Encryption Hierarchy là gì? Trình bày Applications for SQL Server and Database Keys và Managing SQL Server and Database Keys**

Encryption Hierarchy (Cấu trúc mã hóa) là một cấu trúc đặc biệt của SQL Server được sử dụng để mã hóa và giải mã cơ sở dữ liệu. Nó là một tập hợp các phương pháp mã hóa khác nhau với mức độ khó giải mã là khác nhau, để đảm bảo rằng dữ liệu được lưu trữ trên SQL Server là bảo mật và an toàn.

Nó bao gồm hai phần chính:

* Database Encryption Key (DEK): Là khóa chính được sử dụng để mã hóa dữ liệu của một database trên SQL Server.
* Server Certificate: Là chứng chỉ bảo mật được sử dụng để bảo mật DEK.

Cụ thể, DEK được mã hóa bằng sử dụng một mật khẩu, và mật khẩu này lại được mã hóa trở lại bằng cách sử dụng Server Certificate. Quá trình này tạo thành một hệ thống mã hóa thành các lớp, tạo ra một chuỗi Encryption Hierarchy có tính bảo mật cao.

**Ứng dụng cho SQL Server và các khóa cơ sở dữ liệu:**

- SQL Server Encryption Hierarchy được sử dụng để tạo và quản lý các khóa cơ sở dữ liệu. SQL Server hỗ trợ xác thực khóa riêng tư và khóa công khai khi mã hóa và giải mã cơ sở dữ liệu.

- Mã hóa dữ liệu giúp bảo đảm tính toàn vẹn của dữ liệu bằng cách chống lại đánh cắp dữ liệu, sai sót và thậm chí truy cập trái phép. Ngoài ra, mã hóa dữ liệu mang lại sự an toàn trong việc truyền thông qua mạng hoặc chia sẻ dữ liệu giữa các hệ thống.

- SQL Server có thể sử dụng các khóa dành cho một cá nhân (Asymmetric keys), để bảo vệ thông tin người dùng, giữ liệu của người dùng riêng tư và yêu cầu người sử dụng xác thực trước khi truy cập cơ sở dữ liệu.

**Quản lý SQL Server và các khóa cơ sở dữ liệu:**

- SQL Server cung cấp cho chúng ta các công cụ để tạo và quản lý các khóa riêng tư / công khai và khóa đối xứng. Chúng ta cũng có thể sử dụng các nguyên lý của quản lý khóa hiệu quả trong SQL Server quản lý các khóa riêng tư / công khai và khóa đối xứng.

- Quản lý khóa mật khẩu là một phương pháp tuyệt vời để đảm bảo rằng các khóa của chúng ta an toàn và được bảo vệ một cách hiệu quả. SQL Server cung cấp cho người dùng các chức năng và công cụ để quản lý các khóa này, giúp ngăn chặn việc rò rỉ khóa, mã hóa và truy cập trái phép.

- Cũng như các phương pháp mã hóa khác, cấu trúc mã hóa của SQL Server phải được đảm bảo bảo mật và được quản lý cẩn thận. Việc xây dựng, tạo và quản lý các khóa phù hợp với các tiêu chuẩn bảo mật là cực kỳ quan trọng khi sử dụng các phương pháp mã hóa để bảo vệ dữ liệu một cách an toàn và hiệu quả.

**\*\*Cách 2:**

**Applications for SQL Server and Database Keys:**

SQL Server và Database Keys có nhiều ứng dụng khác nhau để bảo vệ dữ liệu trên SQL Server, bao gồm:

* + Bảo vệ dữ liệu nhạy cảm: Sử dụng mã hóa DEK giúp bảo vệ dữ liệu của một database trên SQL Server khỏi các mối đe dọa như vi phạm dữ liệu, lộ thông tin hay tấn công từ bên ngoài.
  + Thực hiện các yêu cầu bảo mật của chính sách quản lý về bảo mật: Tùy thuộc vào chính sách bảo mật của các tổ chức khác nhau, các quản trị viên có thể thiết lập các phương thức mã hóa phù hợp để đáp ứng các yêu cầu của chính sách.
  + Đảm bảo tính toàn vẹn và an ninh cho dữ liệu trên SQL Server: Mã hóa DEK cung cấp một giải pháp để giảm thiểu rủi ro an ninh và đảm chỉnh tính toàn vẹn của database trên SQL Server.

**Managing SQL Server and Database Keys:**

Quản lý SQL Server và Database Keys rất quan trọng để đảm bảo tính bảo mật và sự hoạt động liên tục của hệ thống. Một số chính sách và thủ tục quản lý bảo mật khóa phổ biến bao gồm:

* Đảm bảo tính khả dụng của các khóa: Quản trị viên có trách nhiệm đảm bảo rằng các khóa đã được mã hóa được đảm bảo tính khả dụng nhằm giảm thiểu rủi ro mất mát dữ liệu hoặc gián đoạn hoạt động của hệ thống;
* Xác định và cập nhật các phương thức mã hóa phù hợp cho từng mức độ dữ liệu: Tương tự như việc quản lý bảo mật của một database, quản trị viên cần phải xác định và cập nhật các loại mã hóa phù hợp với từng mức độ dữ liệu trên SQL Server;
* Thực hiện kiểm tra thường xuyên: Quản trị viên có trách nhiệm thực hiện kiểm tra thường xuyên hoặc các biện pháp tham chiếu đến các tiêu chuẩn bảo mật khác để đảm bảo tính năng suất của khóa và các phương tiện mã hóa khác.

1. **Bạn hãy trình bày và mô tả các cơ chế mã hoá trong SQL Server sau:**

* **Transact-SQL functions**

SQL Server cung cấp một số Transact-SQL (T-SQL) functions cho phép mã hóa dữ liệu. Dưới đây là một số T-SQL functions và cách chúng hoạt động:

**1. HASHBYTES:**

HASHBYTES mã hóa một giá trị bằng phương thức băm một chiều, ví dụ như SHA-256 hoặc MD-5. Hàm HASHBYTES trả về một giá trị băm bằng các loại có thể thể hiện được (expressible types) như int, char, nvarchar, và binary. Ví dụ:

```

SELECT HASHBYTES('sha2\_256', 'Hello World')

```

Lưu ý: Giá trị băm bởi HASHBYTES không phải là một giá trị mã hoá mật khẩu mà còn được sử dụng làm chữ ký số hay biện chứng dữ liệu.

**2. ENCRYPTBYKEY:**

ENCRYPTBYKEY được sử dụng để mã hóa dữ liệu bằng khóa đối xứng (symmetric key). Để sử dụng hàm này, bạn cần tạo một symmetric key trước đó và tạo một khóa mã hoá dưới dạng mật khẩu, mã hóa bằng asymmetric key, hoặc được trích xuất từ khóa mã hóa của một certificate. Ví dụ:

```

OPEN SYMMETRIC KEY MySymmetricKey DECRYPTION BY CERTIFICATE MyCertificate

SELECT ENCRYPTBYKEY(KEY\_GUID('MySymmetricKey'), 'MyPassword')

```

**3. ENCRYPTBYCERT:**

ENCRYPTBYCERT được sử dụng để mã hóa bằng certificate. Ví dụ:

```

SELECT ENCRYPTBYCERT(CERT\_ID('MyCertificate'), 'MyPassword')

```

**4. DECRYPTBYKEY:**

DECRYPTBYKEY được sử dụng để giải mã một giá trị được mã hóa bằng symmetric key. Ví dụ:

```

OPEN SYMMETRIC KEY MySymmetricKey DECRYPTION BY CERTIFICATE MyCertificate

SELECT DECRYPTBYKEY(MyEncryptedValue)

```

Trong các hàm T-SQL functions trên, việc mã hóa và giải mã dữ liệu được thực hiện một cách chính xác thông qua các khóa được sử dụng. Tuy nhiên, ta cần đảm bảo rằng các khóa này được lưu trữ và sử dụng một cách an toàn để đảm bảo tính bảo mật của dữ liệu. Ngoài ra, ta cần lưu ý rằng một số T-SQL functions có thể ảnh hưởng đến hiệu suất của hệ thống, do đó cần phải xem xét kỹ càng khi sử dụng trong các truy vấn phức tạp.

* **Asymmetric keys**

Trong SQL Server, Asymmetric key là một loại khóa kỹ thuật số (digital key) sử dụng để bảo mật dữ liệu. Asymmetric key bao gồm hai khóa liên quan đến nhau, là Khóa công khai (public key) và Khóa riêng tư (private key). Khóa công khai được sử dụng để mã hóa dữ liệu trước khi nó được gửi đi, trong khi khóa riêng tư được sử dụng để giải mã dữ liệu khi nó được nhận. Ta hãy cùng tìm hiểu cách sử dụng Asymmetric keys trong SQL Server:

**1. Tạo khóa Asymmetric key:**

Sử dụng lệnh CREATE ASYMMETRIC KEY để tạo khóa Asymmetric key. Ví dụ:

```

CREATE ASYMMETRIC KEY MyAsymmetricKey

WITH ALGORITHM = RSA\_2048

ENCRYPTION BY PASSWORD = 'MyPassword'

```

**2. Mã hóa dữ liệu bằng khóa công khai:**

Sử dụng hàm ENCRYPTBYKEY để mã hóa dữ liệu bằng khóa công khai. Ví dụ:

```

OPEN ASYMMETRIC KEY MyAsymmetricKey

SELECT ENCRYPTBYKEY(ASYMKEY\_ID('MyAsymmetricKey'), 'MySensitiveData')

```

**3. Giải mã dữ liệu bằng khóa riêng tư:**

Sử dụng hàm DECRYPTBYKEY để giải mã dữ liệu bằng khóa riêng tư. Ví dụ:

```

OPEN ASYMMETRIC KEY MyAsymmetricKey

SELECT DECRYPTBYKEY(MyEncryptedData)

```

**4. Xuất khóa riêng tư:**

Sử dụng lệnh EXPORT ASYMMETRIC KEY để xuất khóa riêng tư. Lưu ý rằng quá trình xuất khóa riêng tư cần phải được thực hiện một cách an toàn để tránh mất mát dữ liệu và đảm bảo bảo mật. Ví dụ:

```

BACKUP CERTIFICATE MyCertificate TO FILE = 'D:\Backup\MyCertificate.cer'

WITH PRIVATE KEY (FILE = 'D:\Backup\MyCertificate.pvk', ENCRYPTION BY PASSWORD = 'MyPassword')

```

Trong các cơ chế mã hoá Asymmetric key, việc mã hóa và giải mã dữ liệu được thực hiện một cách chính xác thông qua các khóa được sử dụng. Tuy nhiên, ta cần đảm bảo rằng các khóa này được lưu trữ và sử dụng một cách an toàn để đảm bảo tính bảo mật của dữ liệu.

* **Symmetric keys**

Trong SQL Server, Symmetric key là một loại khóa đối xứng (symmetric key) được sử dụng để bảo mật dữ liệu. Khi sử dụng symmetric key, cùng một khóa được sử dụng để mã hóa và giải mã dữ liệu. Ta hãy cùng tìm hiểu cách sử dụng Symmetric keys trong SQL Server:

**1. Tạo khóa Symmetric key:**

Sử dụng lệnh CREATE SYMMETRIC KEY để tạo khóa Symmetric key. Ví dụ:

```

CREATE SYMMETRIC KEY MySymmetricKey

WITH ALGORITHM = AES\_256

ENCRYPTION BY PASSWORD = 'MyPassword'

```

**2. Mã hóa dữ liệu bằng khóa Symmetric key:**

Sử dụng hàm ENCRYPTBYKEY để mã hóa dữ liệu bằng khóa Symmetric key. Ví dụ:

```

OPEN SYMMETRIC KEY MySymmetricKey

SELECT ENCRYPTBYKEY(KEY\_GUID('MySymmetricKey'), 'MySensitiveData')

```

**3. Giải mã dữ liệu bằng khóa Symmetric key:**

Sử dụng hàm DECRYPTBYKEY để giải mã dữ liệu bằng khóa Symmetric key. Ví dụ:

```

OPEN SYMMETRIC KEY MySymmetricKey

SELECT DECRYPTBYKEY(MyEncryptedData)

```

**4. Xuất khóa Symmetric key:**

Sử dụng lệnh BACKUP SYMMETRIC KEY để xuất khóa Symmetric key. Lưu ý rằng quá trình xuất khóa cần phải được thực hiện một cách an toàn để tránh mất mát dữ liệu và đảm bảo bảo mật. Ví dụ:

```

BACKUP CERTIFICATE MyCertificate TO FILE = 'D:\Backup\MyCertificate.cer'

WITH PRIVATE KEY (FILE = 'D:\Backup\MyCertificate.pvk', ENCRYPTION BY PASSWORD = 'MyPassword')

```

Trong các cơ chế mã hoá Symmetric key, việc mã hóa và giải mã dữ liệu được thực hiện một cách chính xác thông qua khóa sử dụng. Tuy nhiên, ta cần đảm bảo rằng các khóa này được lưu trữ và sử dụng một cách an toàn để đảm bảo tính bảo mật của dữ liệu. Ngoài ra, ta cần lưu ý rằng một số lệnh có thể ảnh hưởng đến hiệu suất của hệ thống, do đó cần phải xem xét kỹ càng khi sử dụng trong các truy vấn phức tạp.

* **Certificates**

Trong SQL Server, Certificate là một loại đối tượng bảo mật được sử dụng để xác thực và mã hóa dữ liệu. Certificate có thể được sử dụng để mã hóa dữ liệu bằng khóa Asymmetric key hoặc Symmetric key, cũng như ký và xác thực bởi chữ ký số. Ta hãy cùng tìm hiểu cách sử dụng Certificates trong SQL Server:

**1. Tạo Certificate:**

Sử dụng lệnh CREATE CERTIFICATE để tạo Certificate. Ví dụ:

```

CREATE CERTIFICATE MyCertificate

WITH SUBJECT = 'My Certificate'

```

**2. Ký dữ liệu bằng Certificate:**

Sử dụng hàm SIGNBYCERT để ký dữ liệu bằng Certificate. Ví dụ:

```

SIGNBYCERT(CERT\_ID('MyCertificate'), 'MyData')

```

**3. Xác thực dữ liệu bằng chữ ký số:**

Sử dụng hàm VERIFYBYCERT để xác thực dữ liệu bằng chữ ký số. Ví dụ:

```

SELECT VERIFYBYCERT(CERT\_ID('MyCertificate'), 'MySignedData', 'MySignature')

```

**4. Mã hóa dữ liệu bằng khóa Asymmetric key và Certificate:**

Sử dụng hàm ENCRYPTBYCERT để mã hóa dữ liệu bằng khóa Asymmetric key và Certificate. Ví dụ:

```

OPEN ASYMMETRIC KEY MyAsymmetricKey

ENCRYPTBYCERT(CERT\_ID('MyCertificate'), MySensitiveData, 1,

ADD\_SIGNATURE = 1,

ALGORITHM = 'RSA\_OAEP',

SIGNER\_CERT = CERT\_ID('MyCertificate'),

SIGNER\_KEY = KEY\_ID('MyAsymmetricKey'))

```

Trong các cơ chế mã hoá Certificate, việc mã hóa và xác thực dữ liệu được thực hiện một cách chính xác thông qua Certificate sử dụng. Tuy nhiên, ta cần đảm bảo rằng Certificate được lưu trữ và sử dụng một cách an toàn để đảm bảo tính bảo mật của dữ liệu. Ngoài ra, ta cần lưu ý rằng một số lệnh có thể ảnh hưởng đến hiệu suất của hệ thống, do đó cần phải xem xét kỹ càng khi sử dụng trong các truy vấn phức tạp.

* **Transparent Data Encryption**

Trong SQL Server, Transparent Data Encryption (TDE) là một tính năng cho phép mã hoá dữ liệu ở mức cơ sở dữ liệu một cách tự động, giúp bảo đảm tính toàn vẹn và bảo mật của dữ liệu. TDE sử dụng khóa đối xứng để mã hóa dữ liệu và khóa này được lưu trữ trong database master key (DMK) của SQL Server. Ta hãy cùng tìm hiểu cách sử dụng TDE trong SQL Server:

**1. Chọn cơ sở dữ liệu:**

Sau khi cài đặt TDE, ta chọn cơ sở dữ liệu mà ta muốn áp dụng tính năng TDE. Ví dụ:

```

USE MyDatabase

```

**2. Tạo database encryption key:**

Sử dụng lệnh CREATE DATABASE ENCRYPTION KEY để tạo database encryption key (DEK) cho cơ sở dữ liệu. Ví dụ:

```

USE MyDatabase

CREATE DATABASE ENCRYPTION KEY

WITH ALGORITHM = AES\_256

ENCRYPTION BY SERVER CERTIFICATE MyCertificate

```

**3. Kích hoạt TDE:**

Sử dụng lệnh ALTER DATABASE để kích hoạt TDE cho cơ sở dữ liệu. Ví dụ:

```

USE MyDatabase

ALTER DATABASE SET ENCRYPTION ON

```

**4. Backup và Restore:**

Khi sử dụng TDE, ta cần backup và restore cả DEK và certificate để đảm bảo tính bảo mật của dữ liệu. Ví dụ:

- Backup DEK và certificate:

```

USE MyDatabase

BACKUP CERTIFICATE MyCertificate

TO FILE = 'D:\Backup\MyCertificate.cer'

WITH PRIVATE KEY (

FILE = 'D:\Backup\MyCertificate.pvk',

ENCRYPTION BY PASSWORD = 'MyPassword'

)

BACKUP SERVICE MASTER KEY

TO FILE = 'D:\Backup\ServiceMasterKey\MyServiceMasterKey.smk'

ENCRYPTION BY PASSWORD = 'MyPassword'

```

- Restore DEK và certificate:

```

RESTORE SERVICE MASTER KEY

FROM FILE = 'D:\Backup\ServiceMasterKey\MyServiceMasterKey.smk'

DECRYPTION BY PASSWORD = 'MyPassword'

ALTER MASTER KEY ADD ENCRYPTION BY SERVICE MASTER KEY

RESTORE CERTIFICATE MyCertificate

FROM FILE = 'D:\Backup\MyCertificate.cer'

WITH PRIVATE KEY (

FILE = 'D:\Backup\MyCertificate.pvk',

DECRYPTION BY PASSWORD = 'MyPassword'

)

```

**\*\*\*Cách trả lời 2**

Trong các cơ chế mã hoá TDE, việc mã hóa và giải mã dữ liệu được thực hiện một cách tự động và có tính bảo mật cao. Tuy nhiên, ta cần đảm bảo rằng DEK và certificate được lưu trữ và sử dụng một cách an toàn để đảm bảo tính bảo mật của dữ liệu. Ngoài ra, ta cần lưu ý rằng việc sử dụng TDE có thể giảm hiệu suất của hệ thống một chút và cần xem xét kỹ càng khi sử dụng trong các truy vấn phức tạp.

1. **Transact-SQL Functions:**

Trong SQL Server, có một số hàm T-SQL mã hoá sẵn được sử dụng để mã hoá dữ liệu. Các hàm tiêu biểu trong nhóm này bao gồm: HASHBYTES, ENCRYPTBYKEY, và ENCRYPTBYCERT. Các hàm này hoạt động tương tự với các hàm mã hoá đối với chuỗi, và có thể sử dụng để mã hoá các giá trị dạng số, ngày tháng, … trong cơ sở dữ liệu.

**2. Asymmetric keys:**

Asymmetric key cung cấp một cặp khóa gồm khóa riêng (private key) và khóa công khai (public key) được sử dụng để mã hoá và giải mã dữ liệu. Khóa công khai có thể cho ai đó võ tùy ý sử dụng để mã hoá dữ liệu. Tuy nhiên, khóa riêng chỉ có thể giữ bởi người tạo ra nó để giải mã dữ liệu được mã hóa bằng khóa công khai. SQL Server cung cấp hỗ trợ cho Asymmetric key, giúp cho ta dễ dàng sử dụng việc mã hóa đối với dữ liệu.

**3. Symmetric keys:**

Symmetric key mã hóa dữ liệu bằng cách sử dụng một khóa duy nhất. Một dạng phổ biến nhất của loại mã hóa này là AES (Advanced Encryption Standard). Symmetric key cần phải được bảo vệ và giữ mật khẩu an toàn khi không dùng nó. Nếu không, dữ liệu cơ sở dữ liệu có thể bị chiếm đoạt.

**4. Certificates:**

Certificate là một chứng nhận kỹ thuật số, được sử dụng làm bảo mật dữ liệu trong SQL Server. Chứng nhận sẽ bao gồm một khóa công khai có thể được sử dụng để điện toán và một khóa bí mật được sử dụng để giải mã dữ liệu. Certificate cũng được sử dụng để bảo mật kết nối an toàn trong hệ thống SQL Server.

**5. Transparent Data Encryption:**

Transparent Data Encryption (TDE) được sử dụng để mã hoá toàn bộ cơ sở dữ liệu trong SQL Server. TDE mã hoá cơ sở dữ liệu bằng cách sử dụng một khoá bí mật được lưu trong cơ sở dữ liệu. Khi thông tin được thêm vào cơ sở dữ liệu, nó sẽ được mã hoá ngay lập tức bằng khoá bí mật. Khi thông tin được lấy ra, nó sẽ được giải mã bằng khoá bí mật.

Kết luận, SQL Server cung cấp nhiều cơ chế mã hoá để bảo vệ dữ liệu, từ những hàm T-SQL, Asymmetric key, Symmetric key, Certificates, đến TDE. Ta cần phải lựa chọn phương pháp phù hợp trong từng trường hợp cụ thể, đảm bảo an toàn và bảo mật cơ sở dữ liệu.

1. **Trình bày kiến trúc của Transparent Data Encryption (TDE)? Mô tả các bước sử dụng Transparent Data Encryption (TDE)**

Transparent Data Encryption (TDE) là một tính năng bảo mật trong SQL Server cho phép mã hóa toàn bộ cơ sở dữ liệu (database encryption) một cách tràn đầy đủ và bảo vệ dữ liệu trên ổ đĩa mà không làm ảnh hưởng đến ứng dụng. Bản thân dữ liệu lưu trên đĩa sẽ được mã hoá, và chỉ có những ai có chứng minh thư (certificate) hoặc khóa bí mật (private key) mới có thể giải mã.

Kiến trúc của Transparent Data Encryption được thể hiện qua các thành phần sau:

1. DMK (Database Master Key): là một chứng minh thư dùng để bảo vệ khóa chính (Database Encryption Key). DMK được tạo trong mỗi database và có thể được tạo bằng lệnh CREATE MASTER KEY.

2. Database Encryption Key (DEK): là một khóa chính dùng để mã hóa toàn bộ dữ liệu trong database. DEK sẽ được mã hóa bởi DMK.

3. Certificate: là một đối tượng được lưu trữ trong SQL Server, cho phép lưu trữ khóa bí mật (private key) được sử dụng để giải mã DEK.

4. Database Encryption Key (DEK) được mã hóa bởi chứng minh thư (certificate). Certificate này được bảo mật bởi một mật khẩu.

Sử dụng Transparent Data Encryption trong SQL Server có các bước như sau:

1. Tạo chứng minh thư (certificate) và khóa bí mật (private key): sử dụng lệnh CREATE CERTIFICATE và OPEN MASTER KEY để tạo DMK và khóa chứng minh thư.

2. Tạo database encryption key (DEK): sử dụng lệnh CREATE DATABASE ENCRYPTION KEY.

3. Mã hóa database encryption key: sử dụng lệnh ALTER DATABASE và chỉ định Certificate để mã hóa DEK.

4. Bật tính năng TDE: lệnh ALTER DATABASE và chỉ định WITH ENCRYPTION BY SERVER CERTIFICATE.

5. Sau khi cài đặt TDE, có thể tạo các database backup. Nhưng khi lưu trữ backup ở nơi khác, ai muốn phục hồi cơ sở dữ liệu phải có chứng minh thư và mật khẩu.

Kết luận, sử dụng Transparent Data Encryption giúp bảo vệ dữ liệu của bạn và đảm bảo an ninh cơ sở dữ liệu. Khi tạo chứng minh thư và mã hóa DEK, giúp bảo mật cơ sở trong trường hợp dữ liệu được truy cập bởi những người không được ủy quyền.

1. **Viết đoạn mã lệnh để mã hoá mức cột sử dụng khoá đối xứng**

-- Tạo khoá đối xứng

CREATE SYMMETRIC KEY MySymKey

WITH ALGORITHM = AES\_256

ENCRYPTION BY PASSWORD = 'password';

-- Mã hoá mức cột sử dụng khoá đối xứng

OPEN SYMMETRIC KEY MySymKey DECRYPTION BY PASSWORD = 'password';

UPDATE mytable

SET sensitive\_data = ENCRYPTBYKEY(KEY\_GUID('MySymKey'), sensitive\_data)

WHERE id = 1;

CLOSE SYMMETRIC KEY MySymKey;

1. **Viết đoạn mã lệnh để mã hoá mức cột sử dụng khoá bất đối xứng (Enable Transparent Data Encryption (TDE) Using EKM)**

-- Kích hoạt SafeNet Luna SA EKM Provider

USE master;

CREATE CRYPTOGRAPHIC PROVIDER SafeNet\_Luna\_SA\_EKM\_Provider

FROM FILE = 'C:\ProgramFiles\SafeNet\LunaSA\client\cpp\bin\snlekm\_client.dll'

-- Thay đổi đường dẫn đến thư viện để phù hợp với cài đặt máy trạm

WITH ROLLOVER = SUPPORTED

-- Tạo cặp khóa RSA

CREATE ASYMMETRIC KEY MyRSAKey

WITH ALGORITHM = RSA\_2048

ENCRYPTION BY PASSWORD = 'mypassword'

\_PROVIDER = 'SafeNet\_Luna\_SA\_EKM\_Provider';

-- Kích hoạt tính năng TDE

ALTER DATABASE mydatabase

SET ENCRYPTION ON;

-- Tạo Database Encryption Key (DEK)

USE mydatabase;

CREATE DATABASE ENCRYPTION KEY

WITH ALGORITHM = RSA\_2048

ENCRYPTION BY SERVER ASYMMETRIC KEY MyRSAKey

\_PROVIDER = 'SafeNet\_Luna\_SA\_EKM\_Provider';

-- Mã hoá mức cột sử dụng khoá bất đối xứng

OPEN SYMMETRIC KEY MyDEK DECRYPTION BY CERTIFICATE MyCertificate

UPDATE mytable

SET sensitive\_data = ENCRYPTBYASYMKEY(ASYMKEY\_ID('MyRSAKey'), sensitive\_data)

WHERE id = 1;

CLOSE SYMMETRIC KEY MyDEK;

1. **Nêu các thành phần của SQL Server Audit?**

SQL Server Audit là tính năng audit được tích hợp sẵn trong SQL Server, cho phép người quản trị theo dõi và ghi lại các sự kiện liên quan đến việc truy cập cơ sở dữ liệu, tương tác với các đối tượng trong cơ sở dữ liệu và các hoạt động quản trị hệ thống. Các **thành phần chính của SQL Server Audit** bao gồm:

1. Audit: Audit là một tập hợp các quy tắc cấu hình cho phép người quản trị xác định các sự kiện nào sẽ được lưu trữ. Các sự kiện này bao gồm truy cập vào cơ sở dữ liệu, tương tác với các đối tượng và các hoạt động như đăng nhập, đăng xuất và thay đổi mật khẩu.

2. Server Audit Specification: Server Audit Specification là một tập hợp các sự kiện được chọn để ghi lại trong cơ sở dữ liệu. Server Audit Specification cho phép người quản trị xác định các sự kiện cụ thể sẽ được ghi lại cho mỗi cơ sở dữ liệu.

3. Database Audit Specification: Database Audit Specification tương tự như Server Audit Specification, nhưng nó áp dụng cho từng cơ sở dữ liệu. Điều này cho phép người quản trị cấu hình các sự kiện được ghi lại cho từng cơ sở dữ liệu.

4. Audit Action: Audit Action là một hoạt động được lưu trữ trong bảng sự kiện của cơ sở dữ liệu. Các hành động này bao gồm truy cập cơ sở dữ liệu, tương tác với các đối tượng và các hành động quản trị hệ thống.

5. Audit Action Group: Audit Action Group là một tập hợp các Audit Action liên quan đến nhau. Nó cho phép người quản trị lưu trữ một tập hợp các sự kiện liên quan như thao tác đầu vào, thao tác thời gian thực, thao tác lỗi và thao tác phát hiện.

6. Audit Logs: Audit Logs là tệp tin hoặc bảng cơ sở dữ liệu lưu trữ các sự kiện được ghi lại. Chúng cho phép người quản trị xem lại các hoạt động của người dùng, đối tượng và hệ thống trong cơ sở dữ liệu.

Tổng quan, các thành phần của SQL Server Audit cung cấp cho người quản trị khả năng kiểm soát và giám sát các hoạt động trong cơ sở dữ liệu để đảm bảo an toàn và bảo mật dữ liệu.

1. **Trình bày tổng quan về việc sử dụng SQL Server Audit (Overview of Using SQL Server Audit)**

SQL Server Audit là tính năng tích hợp trong SQL Server cho phép các quản trị viên hệ thống cho phép việc kiểm tra, giám sát và ghi lại các sự kiện và các hoạt động xảy ra trong SQL Server. Tính năng này giúp các quản trị viên xác định các nguy cơ bảo mật trong hệ thống và thực hiện các biện pháp bảo mật nhằm đảm bảo tính bảo mật và tuân thủ các quy định bảo mật và pháp lý.

SQL Server Audit cho phép người dùng kiểm tra các hoạt động như tạo, xoá, cập nhật và truy cập dữ liệu trên SQL Server, giúp họ có thể phát hiện và ngăn chặn các vấn đề bảo mật như truy cập trái phép, lộ thông tin hoặc thao túng dữ liệu. Việc ghi lại các hoạt động này trong SQL Server cũng giúp quản trị viên tạo một cơ sở dữ liệu quan trọng để phân tích và đánh giá rủi ro bảo mật của hệ thống.

SQL Server Audit được sử dụng để bảo đảm tính toàn vẹn dữ liệu, phục vụ mục đích an ninh và tuân thủ các quy định pháp luật liên quan đến bảo mật dữ liệu. Việc sử dụng SQL Server Audit giúp người quản trị cơ sở dữ liệu dễ dàng giám sát và theo dõi các hoạt động của người dùng và chứng minh cho quản lý về việc tuân thủ các quy định pháp luật và các chính sách có liên quan.

**Để sử dụng SQL Server Audit**, người quản trị cần tạo các chuẩn hóa tương ứng: Server Audit, Server Audit Specification và Database Audit Specification. Sau khi cấu hình các chuẩn hóa này, SQL Server Audit sẽ bắt đầu ghi lại các sự kiện và các hoạt động được chỉ định trong chuẩn hóa. Người quản trị có thể dễ dàng xem các sự kiện được ghi lại trong các tệp log hoặc theo dõi chúng trực tiếp trong SQL Server Management Studio.

1. **Bạn hiểu thế nào về Database Mirroring and SQL Server Audit. Cho ví dụ minh hoạ**

**Database Mirroring** là một tính năng trong SQL Server cho phép sao lưu cơ sở dữ liệu và khôi phục tự động trong trường hợp xảy ra lỗi hoặc sự cố. Database Mirroring tạo ra một bản sao lưu của cơ sở dữ liệu trên một máy chủ đích, được gọi là Mirror Server, và đồng bộ hóa các phiên bản sao lưu này với cơ sở dữ liệu chính. Nếu máy chủ chính gặp trục trặc hoặc không thể truy cập được, Mirror Server sẽ được sử dụng để phục hồi dữ liệu.

**SQL Server Audit** là một tính năng trong SQL Server cho phép ghi lại các hoạt động trong cơ sở dữ liệu như truy cập vào cơ sở dữ liệu, đăng nhập và đăng xuất người dùng, tạo hoặc sửa đổi đối tượng cơ sở dữ liệu, và thực hiện các thao tác truy vấn. SQL Server Audit cho phép người quản trị có thể giám sát và kiểm soát các hoạt động trong cơ sở dữ liệu một cách chặt chẽ.

**Cho ví dụ,** giả sử bạn có một cơ sở dữ liệu quan trọng lưu trữ thông tin khách hàng và giao dịch của một công ty. Bạn muốn đảm bảo tính toàn vẹn và sẵn sàng của cơ sở dữ liệu trong trường hợp xảy ra sự cố hay thời gian ngưng truy cập. Bạn cũng muốn quản lý các hoạt động cơ sở dữ liệu hàng ngày để đảm bảo rằng dữ liệu được bảo mật và đáp ứng các quy định pháp luật liên quan.

Để giải quyết vấn đề này, bạn có thể sử dụng tính năng Database Mirroring để tạo bản sao lưu của cơ sở dữ liệu và đồng bộ hóa các phiên bản sao lưu này với Mirror Server. Khi máy chủ chính gặp sự cố hoặc không thể truy cập được, Mirror Server sẽ được sử dụng để phục hồi dữ liệu. Điều này cho phép bạn kiểm soát được thời gian ngưng truy cập và đảm bảo rằng cơ sở dữ liệu của bạn sẽ luôn sẵn sàng.

Để giám sát các hoạt động cơ sở dữ liệu hàng ngày, bạn có thể sử dụng tính năng SQL Server Audit để ghi lại các hoạt động như truy cập cơ sở dữ liệu, tạo đối tượng mới, thực hiện các thao tác truy vấn và thay đổi quyền truy cập. Bằng cách theo dõi các hoạt động này, bạn có thể phát hiện sớm các hoạt động không chính thức và lỗ hổng bảo mật trong cơ sở dữ liệu của bạn.

Tóm lại, tính năng Database Mirroring và SQL Server Audit là những tính năng quan trọng trong SQL Server để đảm bảo tính toàn vẹn và bảo mật của cơ sở dữ liệu. Chúng giúp bạn tự động sao lưu và khôi phục dữ liệu, giám sát và kiểm soát các hoạt động trong cơ sở dữ liệu một cách chặt chẽ.

1. **Mô tả các bước Creating and Managing Audits with Transact-SQL. Cho ví dụ minh hoạ**

Các bước để tạo và quản lý SQL Server Audit bằng Transact-SQL bao gồm:

1. Tạo Audit - Bước đầu tiên là tạo Audit bằng lệnh `CREATE SERVER AUDIT`. Lệnh này sẽ tạo một Audit với tên và đối tượng audit được chỉ định. Ví dụ:

```

CREATE SERVER AUDIT Audit1

TO FILE

( FILEPATH = 'C:\Audit\' )

WITH

(

QUEUE\_DELAY = 1000,

ON\_FAILURE = CONTINUE

);

```

2. Tạo Audit Specification - Sau khi tạo Audit, bạn cần tạo Audit Specification để chứa một hoặc nhiều đối tượng và sự kiện bạn muốn ghi log. Bạn có thể tạo Audit Specification bằng lệnh `CREATE SERVER AUDIT SPECIFICATION`. Ví dụ:

```

CREATE SERVER AUDIT SPECIFICATION AuditSpec1

FOR SERVER AUDIT Audit1

ADD (DATABASE\_OBJECT\_CHANGE\_GROUP)

ADD (DATABASE\_PRINCIPAL\_CHANGE\_GROUP)

WITH (STATE = ON);

```

3. Kích hoạt Audit - Sau khi tạo Audit và Audit Specification, bạn cần kích hoạt Audit bằng lệnh `ALTER SERVER AUDIT`. Ví dụ:

```

ALTER SERVER AUDIT Audit1

WITH (STATE = ON);

```

4. Kiểm tra Audit Status - Bạn có thể kiểm tra trạng thái của Audit bằng cách sử dụng lệnh `sys.server\_audits`. Ví dụ:

```

SELECT \* FROM sys.server\_audits

WHERE name = 'Audit1'

```

5. Kiểm tra Audit Specification Status - Bạn cũng có thể kiểm tra trạng thái của Audit Specification bằng cách sử dụng lệnh `sys.server\_audit\_specifications`. Ví dụ:

```

SELECT \* FROM sys.server\_audit\_specifications

WHERE name = 'AuditSpec1'

```

6. Cập nhật Audit Specification - Nếu bạn muốn thêm hoặc xóa sự kiện hoặc đối tượng audit khác, bạn có thể cập nhật Audit Specification bằng lệnh `ALTER SERVER AUDIT SPECIFICATION`. Ví dụ:

```

ALTER SERVER AUDIT SPECIFICATION AuditSpec1

WITH (STATE = OFF);

ALTER SERVER AUDIT SPECIFICATION AuditSpec1

REMOVE (DATABASE\_PRINCIPAL\_CHANGE\_GROUP);

ALTER SERVER AUDIT SPECIFICATION AuditSpec1

ADD (DATABASE\_ROLE\_MEMBER\_CHANGE\_GROUP),

ADD (SCHEMA\_OBJECT\_CHANGE\_GROUP);

ALTER SERVER AUDIT SPECIFICATION AuditSpec1

WITH (STATE = ON);

```

Tóm lại, đây là các bước để tạo và quản lý SQL Server Audit bằng Transact-SQL. Việc sử dụng SQL Server Audit giúp giáo dụcIT tăng khả năng quản trị và bảo mật cho cơ sở dữ liệu của họ và giúp phát hiện sớm các sự cố bảo mật trong cơ sở dữ liệu.

1. **Trình bày các bước tạo Create a Server Audit and Server Audit Specification. Cho ví dụ minh hoạ**

Để tạo một Server Audit và Server Audit Specification trong SQL Server, các bước cơ bản như sau:

1. Tạo Server Audit – Bước đầu tiên là tạo một Server Audit bằng cách sử dụng lệnh `CREATE SERVER AUDIT` và chỉ định các thông tin cần thiết như tên, thành phần và vị trí của các bản ghi Audit. Ví dụ:

```

CREATE SERVER AUDIT [Audit1]

TO FILE (FILEPATH = 'C:\Audit\')

WITH (QUEUE\_DELAY = 1000, ON\_FAILURE = CONTINUE);

```

2. Tạo Server Audit Specification – Sau khi tạo Server Audit, bạn cần tạo một Server Audit Specification để chỉ định những hành động mà bạn muốn ghi Audit. Bạn có thể làm điều này bằng cách sử dụng lệnh `CREATE SERVER AUDIT SPECIFICATION` và chỉ định tên Audit Specification, tên Audit mà nó liên kết và những hành động cụ thể mà bạn muốn theo dõi. Ví dụ:

```

CREATE SERVER AUDIT SPECIFICATION [AuditSpec1]

FOR SERVER AUDIT [Audit1]

ADD (DATABASE\_OBJECT\_CHANGE\_GROUP)

ADD (DATABASE\_PRINCIPAL\_CHANGE\_GROUP)

WITH (STATE = ON);

```

Lệnh trên sẽ tạo một Audit Specification mang tên AuditSpec1, liên kết với Audit có tên Audit1 và sẽ ghi lại các hành động thay đổi đối tượng cơ sở dữ liệu và nhóm thay đổi người dùng.

3. Kích hoạt Audit Specification – Sau khi tạo Audit Specification, bạn cần kích hoạt nó. Bạn có thể làm điều này bằng cách sử dụng lệnh `ALTER SERVER AUDIT SPECIFICATION` và chỉ định tên Audit Specification và trạng thái ON. Ví dụ:

```

ALTER SERVER AUDIT SPECIFICATION [AuditSpec1]

WITH (STATE = ON);

```

4. Kiểm tra Audit Status – Bạn có thể kiểm tra trạng thái của Audit bằng cách sử dụng lệnh `sys.server\_audits`. Ví dụ:

```

SELECT \* FROM sys.server\_audits

WHERE name = 'Audit1'

```

5. Kiểm tra Audit Specification Status – Bạn có thể kiểm tra trạng thái của Audit Specification bằng cách sử dụng lệnh `sys.server\_audit\_specifications`. Ví dụ:

```

SELECT \* FROM sys.server\_audit\_specifications

WHERE name = 'AuditSpec1'

```

Tóm lại, đây là các bước để tạo và quản lý một Server Audit và Server Audit Specification trong SQL Server bằng Transact-SQL. Việc sử dụng Server Audit và Audit Specification giúp quản trị viên có khả năng giám sát và kiểm soát các hoạt động trong cơ sở dữ liệu một cách chặt chẽ và giúp phát hiện sớm các sự cố bảo mật.

1. **Trình bày những gì bạn biết về minorring, logshipping và replication. Cho ví dụ minh hoạ**

1. Mirroring:

Mirroring (đồng bộ hóa) là một kỹ thuật để sao chép quy trình xử lý và cơ sở dữ liệu trên các máy chủ để cung cấp khả năng cao cho hệ thống. Khi sử dụng mirroring, có hai máy chủ: một máy chủ chính (principal) và một máy chủ đồng bộ (mirror). Các thay đổi dữ liệu được thực hiện trên máy chủ chính sẽ được sao chép đến máy chủ đồng bộ bằng cách sử dụng mạng. Nếu máy chủ chính bị lỗi, máy chủ đồng bộ sẽ được viết để lấy chỗ của máy chủ chính để tiếp tục thành công. Ví dụ về mirroring:

```

-- Tạo Principal Database

CREATE DATABASE Database1;

GO

-- Tạo Mirror Database với tùy chọn Mirror

CREATE DATABASE Database1\_Mirror

ON

PRIMARY

(NAME = 'MinhTien',

FILENAME = 'D:\BMCSDL\MinhTien\_Mirror.mdf')

LOG ON

(NAME = 'Database1\_log',

FILENAME = 'D:\BMCSDL\MinhTien\_log.ldf')

GO

-- Enable Mirroring

ALTER DATABASE Database1

SET PARTNER = 'TCP://[Mirror\_Server\_Name]:[Port]';

GO

ALTER DATABASE Database1\_Mirror

SET PARTNER = 'TCP://[Principal\_Server\_Name]:[Port]';

GO

```

2. Log Shipping:

Log Shipping là kỹ thuật khác để sao chép dữ liệu. Kỹ thuật này là một giải pháp phổ biến cho việc tạo bản sao phụ bản của cơ sở dữ liệu trên máy chủ khác, giúp bảo vệ dữ liệu bằng cách sao lưu các giao dịch vào tệp log và chuyển chúng đến máy chủ sao lưu. Khi máy chủ chính gặp sự cố và dữ liệu bị mất, ta có thể khôi phục chúng từ bản sao phụ bản. Ví dụ về Log Shipping:

```

-- Tạo sao lưu

BACKUP DATABASE MinhTien TO DISK = 'D:\Backup\ MinhTien.bak';

BACKUP LOG MinhTien TO DISK = 'D:\Backup\ MinhTien.log';

GO

-- Tạo nơi chưa nhật ký

RESTORE DATABASE MinhTien\_Copy

WITH NORECOVERY

FROM DISK = 'D:\Backup\ MinhTien.bak';

GO

RESTORE LOG MinhTien\_Copy

WITH NORECOVERY

FROM DISK = 'D:\Backup\ MinhTien.log';

GO

-- Enable Log Shipping

USE master

GO

EXEC sp\_add\_log\_shipping\_primary\_database

@database = N' MinhTien ',

@backup\_directory = N'\\[Backup\_Server\_Name]\BackupFolder',

@backup\_share = N'\\[Backup\_Server\_Name]\BackupShare',

@backup\_retention\_period = 1440,

@monitor\_server = N'[Monitor\_Server\_Name]',

@monitor\_server\_security\_mode = 1,

@backup\_threshold = 5,

@threshold\_alert\_enabled = 0,

@history\_retention\_period = 5760

GO

```

3. Replication:

Replication là kỹ thuật sao chép và phân phối cơ sở dữ liệu trên nhiều máy chủ để đáp ứng nhu cầu di chuyển dữ liệu giữa các hoạt động của một tổ chức. Sử dụng Replication, ta có thể tạo ra nhiều bản sao cơ sở dữ liệu độc lập nhau. Các thay đổi sẽ được sao chép và phân phối đến các bản sao khác để đảm bảo dữ liệu đồng bộ. Ví dụ về Replication:

```

-- Tạo bản sao (Subscriber)

USE [master]

GO

EXEC sp\_addsubscriber @subscriber = N'Subscriber\_Server\_Name'

, @description = N'Subscriber\_Server\_Description'

, @security\_mode = 0

, @frequency\_type = 4

, @frequency\_interval = 1

, @frequency\_relative\_interval = 1

, @frequency\_recurrence\_factor = 0

, @active\_start\_time\_of\_day = 190000

, @active\_end\_time\_of\_day = 235959

, @active\_start\_date = 20161011

, @active\_end\_date = 20161231

GO

-- Thêm bản sao db (Subscriber\_db) vào publication

USE [Publisher\_db]

GO

EXEC sp\_addsubscription @publication = N'publication\_name'

, @subscriber = N'Subscriber\_Server\_Name'

, @destination\_db = N'Subscriber\_db'

, @subscription\_type = N'Push'

, @sync\_type= N'automatic'

, @article = N'all'

, @update\_mode = N'read only'

GO

```

Tóm lại, đó là những điều ta biết về Mirroring, Log Shipping và Replication trong SQL Server. Các kỹ thuật này đều có thể giúp giảm thiểu mất dữ liệu và giúp đảm bảo sự tin cậy của cơ sở dữ liệu trong các hệ thống lớn và phức tạp. Việc lựa chọn con phù hợp nhất cần phải dựa trên nhu cầu của tổ chức và sẽ có sự khác biệt về hiệu suất và chi phí.

**Giám sát hiệu suất khi giám sát bảng sản phẩm, các sản lượng sản phẩm tạo**

**Lập bảng stt, tên, thời gian giám sát ngày giờ cụ thể, đối tượng nào bị giám sát, có sự kiện gì xẩy ra không**

**Thêm giám sát bên ngoài nữa**

**MỤC LỤC**

**1. Sinh viên hãy giải thích vì sao điều khiển truy cập tùy quyền (Discretionary Access Control) có tính linh hoạt trong quản lý quyền truy cập. Cho ví dụ minh hoạ**

**2. Sinh viên hãy trình bày về bài toán xác thực quyền hạn (Authorization) trong lĩnh vực bảo mật cơ sở dữ liệu. Cho ví dụ minh hoạ**

**3. Sinh viên hãy giải thích vì sao mô hình điều khiển truy cập bắt buộc (Mandatory Access Control) ít được vận dụng trong các ứng dụng thực tế. Cho ví dụ minh hoạ**

4. **Sinh viên hãy giải thích vì sao điều khiển truy cập tùy quyền không kiểm soát được luồng thông tin. Cho ví dụ minh hoạ //Hướng dẫn: xem ví dụ về cách tấn công Trojan**

**5. Sinh viên hãy trình bày về bài toán xác thực người dùng (Authentication) trong lĩnh vực bảo mật cơ sở dữ liệu. Cho ví dụ minh hoạ**

**6. Sinh viên hãy trình bày về bài toán xác thực quyền hạn (Authorization) trong lĩnh vực bảo mật cơ sở dữ liệu. Cho ví dụ minh hoạ**

**7. Sinh viên hãy trình bày về bài toán kiểm toán truy cập (Auditing) trong lĩnh vực bảo mật cơ sở dữ liệu. Cho ví dụ minh hoạ**

**8. Giải thích tính “tùy ý” trong mô hình điều khiển truy cập tùy ý? Trong mô hình điều khiển truy cập tùy ý, những loại người dùng nào có quyền phân quyền/thu hồi quyền hạn? Trong mô hình điều khiển truy cập tùy ý, tại sao ưu điểm là tính linh hoạt, tính dễ dàng.**

9.**Tại sao khi cài đặt, người ta sử dụng danh sách khả năng thay thế cho cấu trúc ma trận phân quyền?**

**10. Giải thích tính “bắt buộc” trong mô hình điều khiển truy cập ~~tùy ý~~ bắt buộc? Trong mô hình điều khiển truy cập bắt buộc, tại sao nó ít được vận dụng trong thực tế? Tại sao mô hình điều khiển truy cập bắt buộc được áp dụng trong môi trường quân đội. Trong mô hình điều khiển truy cập bắt buộc, tại sao nó ít mang tính nghiêm ngặt/khắc khe?**

**11. Ý tưởng trọng tâm của mô hình RBAC là gì? Vì sao RBAC hỗ trợ nguyên tắc bảo mật: đặc quyền ít nhất (Least Privilege)? Vì sao RBAC hỗ trợ nguyên tắc bảo mật: sự tách biệt các nhiệm vụ (Separation of duties). Vì sao RBAC hỗ trợ nguyên tắc bảo mật: trừu tượng hóa dữ liệu (Data Abstraction).**

**12. Điểm đặc trưng của mô hình Hierarchical RBAC là gì? Điểm đặc trưng của mô hình Static Separation of Duty Relations là gì? Điểm đặc trưng của mô hình Dynamic Separation of Duty Relations là gì?**

**13. So sánh điểm giống/khác nhau giữa hai khái niệm kế thừa vai trò tổng quát (General Role Hierarchies) và kế thừa vai trò bị giới hạn (Limited Role Hierarchies).**

**14. Trong mô hình RBAC, xung đột** **(Conflit) quyền hạn là gì? Bạn hãy nêu một chiến lược (strategy) hoặc giải pháp (solution) giải quyết xung đột.**

**15. Trình bày các phương pháp bảo mật CSDL mà bạn biết**

**16. Encryption Hierarchy là gì? Trình bày Applications for SQL Server and Database Keys và Managing SQL Server and Database Keys**

**17. Bạn hãy trình bày và mô tả các cơ chế mã hoá trong SQL Server sau:**

* **Transact-SQL functions**
* **Asymmetric keys**
* **Symmetric keys**
* **Certificates**
* **Transparent Data Encryption**

**18. Trình bày kiến trúc của Transparent Data Encryption (TDE)? Mô tả các bước sử dụng Transparent Data Encryption (TDE)**

**19. Viết đoạn mã lệnh để mã hoá mức cột sử dụng khoá đối xứng**

**20. Viết đoạn mã lệnh để mã hoá mức cột sử dụng khoá bất đối xứng (Enable Transparent Data Encryption (TDE) Using EKM)**

**21. Nêu các thành phần của SQL Server Audit?**

**22. Trình bày tổng quan về việc sử dụng SQL Server Audit (Overview of Using SQL Server Audit)**

**23. Bạn hiểu thế nào về Database Mirroring and SQL Server Audit. Cho ví dụ minh hoạ**

**24. Mô tả các bước Creating and Managing Audits with Transact-SQL. Cho ví dụ minh hoạ**

**25. Trình bày các bước tạo Create a Server Audit and Server Audit Specification. Cho ví dụ minh hoạ**

**26. Trình bày những gì bạn biết về minorring, logshipping và replication. Cho ví dụ minh hoạ**

**16) Cho bảng dữ liệu SINHVIEN(masv, hodem, ten, sodienthoai, diachi, sotaikhoan) chứa nhiều dữ liệu phục vụ hệ thống phần mềm đã được vận hành nhiều năm. Do nhu cầu nâng cấp bảo mật bằng phương pháp mã hóa dữ liệu sử dụng giải pháp được Microsoft SQL Server cung cấp sẵn, thực hiện các yêu cầu sau đây:**

**a. Tạo bảng SINHVIEN\_MAHOA để chứa dữ liệu được chuyển đổi từ bảng SINHVIEN, viết câu lệnh chuyển đổi dữ liệu bản rõ từ bảng SINHVIEN sang dữ liệu bản mã và lưu trữ vào bảng SINHVIEN\_MAHOA.**

create database test

use test

create table SINHVIEN (

masv int identity,

hodem nvarchar(100),

ten nvarchar(100),

sodienthoai nvarchar(20),

diachi nvarchar(50),

sotaikhoan nvarchar(100)

)

insert into SINHVIEN (hodem, ten, sodienthoai, diachi, sotaikhoan)

values (N'Nguyễn Thành', N'Phát', '091234567', N'Gò Vấp', '01122233')

insert into SINHVIEN (hodem, ten, sodienthoai, diachi, sotaikhoan)

values (N'Trần Ngọc', N'Thái', '09321654', N'Q2', '0223344')

insert into SINHVIEN (hodem, ten, sodienthoai, diachi, sotaikhoan)

values (N'Phạm Thanh', N'Trang', '09456789', N'Q5', '0556677')

select \* from SINHVIEN

create table SINHVIEN\_MAHOA (

masv\_mahoa varbinary(max),

hodem\_mahoa varbinary(max),

ten\_mahoa varbinary(max),

sodienthoai\_mahoa varbinary(max),

diachi\_mahoa varbinary(max),

sotaikhoan\_mahoa varbinary(max)

)

select \* from SINHVIEN\_MAHOA

-- Chuyển dữ liệu từ bảng SINHVIEN sang bảng SINHVIEN\_MAHOA

insert into SINHVIEN\_MAHOA (masv\_mahoa, hodem\_mahoa, ten\_mahoa, sodienthoai\_mahoa, diachi\_mahoa, sotaikhoan\_mahoa)

SELECT ENCRYPTBYPASSPHRASE('1', cast(masv as varchar(100))), ENCRYPTBYPASSPHRASE('1', hodem), ENCRYPTBYPASSPHRASE('1', ten),

ENCRYPTBYPASSPHRASE('1', sodienthoai), ENCRYPTBYPASSPHRASE('1', diachi), ENCRYPTBYPASSPHRASE('1', sotaikhoan)

FROM SINHVIEN

**b. Với bảng dữ liệu SINHVIEN\_MAHOA chứa dữ liệu đã được mã hóa, trình bày giải pháp được thực hiện ở tầng cơ sở dữ liệu sao cho tầng ứng dụng (app) ít sửa đổi nhất? Cho ví dụ với các trường hợp xem, thêm, xóa, sửa dữ liệu sinh viên?**

**Sử dụng**

**Stored Procedures:**

* Tạo các stored procedures trong cơ sở dữ liệu để thực hiện các thao tác (xem, thêm, xóa, sửa) trên bảng SINHVIEN\_MAHOA.
* Ứng dụng chỉ gọi các stored procedures thay vì viết trực tiếp các câu lệnh SQL.

**Views:**

* Tạo views trên bảng SINHVIEN\_MAHOA để ẩn chi tiết mã hóa.
* Ứng dụng truy vấn views thay vì trực tiếp truy vấn bảng.

**Triggers:**

* Sử dụng triggers để tự động thực hiện các thay đổi khi dữ liệu được thêm, sửa hoặc xóa.

-- Xem

select convert(varchar(max),DECRYPTBYPASSPHRASE('1', masv\_mahoa)) as masv,

convert(nvarchar(max),DECRYPTBYPASSPHRASE('1', hodem\_mahoa)) as hodem,

convert(nvarchar(max),DECRYPTBYPASSPHRASE('1', ten\_mahoa)) as ten,

convert(nvarchar(max),DECRYPTBYPASSPHRASE('1', sodienthoai\_mahoa)) as sodienthoai,

convert(nvarchar(max),DECRYPTBYPASSPHRASE('1', diachi\_mahoa)) as diachi,

convert(nvarchar(max),DECRYPTBYPASSPHRASE('1', sotaikhoan\_mahoa)) as sotaikhoan

from SINHVIEN\_MAHOA

-- Thêm

insert into SINHVIEN\_MAHOA

values (ENCRYPTBYPASSPHRASE('1', '4'), ENCRYPTBYPASSPHRASE('1', N'Thanh Long'),

ENCRYPTBYPASSPHRASE('1', N'Lang'), ENCRYPTBYPASSPHRASE('1', N'09643754'),

ENCRYPTBYPASSPHRASE('1', N'Q7'), ENCRYPTBYPASSPHRASE('1', N'0322233'))

-- Xóa

delete SINHVIEN\_MAHOA

where convert(varchar(max), DECRYPTBYPASSPHRASE('1', masv\_mahoa)) = 1

-- Sửa

update SINHVIEN\_MAHOA

set ten\_mahoa = ENCRYPTBYPASSPHRASE('1', N'Thông')

where CONVERT(nvarchar(max), DECRYPTBYPASSPHRASE('1', ten\_mahoa)) = N'Thái'

**17) Giải thích mô hình Audit sau:**

**A diagram of a cloud computing system

Description automatically generated**