BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC VÀ QUẢN LÝ CÔNG NGHỆ HẢI PHÒNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên: Nguyễn Quốc Anh

Giảng viên hướng dẫn: Nguyễn Như Chiến

HẢI PHÒNG - 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC VÀ QUẢN LÝ CÔNG NGHỆ HẢI PHÒNG

TÌM HIỂU, TRIỂN KHAI MỘT SỐ CƠ CHẾ MÃ HOÁ DỮ LIỆU TRONG HQTCSDL POSTGRESQL

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên: **Nguyễn Quốc Anh**

Giảng viên hướng dẫn: **Nguyễn Như Chiến**

HẢI PHÒNG - 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC VÀ QUẢN LÝ CÔNG NGHỆ HẢI PHÒNG

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Sinh viên: Nguyễn Quốc Anh **Mã sinh viên:** 1812101003

Lớp: CT2201C

Ngành: Quản trị mạng

Tên đề tài: Tìm hiểu, triển khai một số cơ chế mã hóa dữ liệu trong HQTCSDL POSTGRESQL

LỜI CẢM ƠN

Để hoàn thành tốt được Đồ án tốt nghiệp, em xin gửi lời cảm ơn chân thành đến các thầy cô trong Khoa Công Nghệ Thông tin của Trường ĐH Quản Lý và Công Nghệ Hải Phòng đã tạo điều kiện tốt nhất cho em để em hoàn thành đề tài đúng như dự kiến. Đặc biệt em xin gửi lời cảm ơn sâu sắc đến Cô Nguyễn Thị Xuân Hương – Lãnh đạo Khoa Công Nghệ Thông Tin và Thầy Nguyễn Như Chiến – Giảng viên hướng dẫn đồ án đã trực tiếp hướng dẫn và tận tình giúp đỡ em để em có thể hoàn thành tốt đồ án tốt nghiệp của mình.

Em xin chân thành cảm ơn các lãnh đạo của Trường ĐH Quản Lý và Công Nghệ, các Thầy, Cô trong khoa Công Nghệ Thông Tin đã tạo cho em điều kiện tốt nhất từ khi còn ngồi trên ghế nhà trường cho đến khi hoàn thành đồ án tốt nghiệp quan trọng nhất trong cuộc đời sinh viên.

Trong quá trình thực tập, cũng như là trong quá trình làm đồ án tốt nghiệp em không tránh khỏi những sai sót, em rất mong các Thầy, Cô bỏ qua. Đồng thời do trình độ lý luận cũng như trong kinh nghiệm thực tiễn của em còn nhiều hạn chế nên không tránh khỏi những thiếu sót. Vậy nên, em rất mong sự đóng góp ý kiến từ Thầy, Cô để em học thêm được nhiều kinh nghiệm và kiến thúc để có thể góp ích cho những công việc sau này.

Em xin chân thành cảm ơn!

Hải Phòng, ngày 10 tháng 6 năm 2023

Sinh viên

(Ký và ghi rõ họ tên)

LỜI CAM ĐOAN

Em xin cam đoan rằng đề tài này được tiến hành một cách minh bạch, công khai. Mọi thứ được dựa trên sự cố gắng cũng như sự nỗ lực của bản thân cùng với sự giúp đỡ của thầy Nguyễn Như Chiến.

Các số liệu và kết quả nghiên cứu được đưa ra trong đồ án là trung thực và không sao chép hay sử dụng kết quả của bất kỳ đề tài nghiên cứu nào tương tự. Nếu như phát hiện rằng có sự sao chép kết quả nghiên cứu đề những đề tài khác bản thân em xin chịu hoàn toàn trách nhiệm.

Hải Phòng, ngày 10 tháng 6 năm 2023

Sinh viên

(Ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ TỪ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
MVCC	Multiversion concurrency control	Kiểm soát truy cập đồng thời nhiều phiên bản
RAM	Random Access Memory	Bộ nhớ truy cập ngẫu nhiên
SQL	Structured Query Language	Ngôn ngữ truy vấn mang tính cấu trúc
TCP	Transmission control protocol	Giao thức điều khiển truyền nhận ở tầng vận tải
DDoS	Distributed denial-of-service	Tấn công từ chối dịch vụ phân tán
TDE	Transparent data encryption	Mã hóa dữ liệu trong suốt
DEK	Data Encryption Key	Khóa mã hóa dữ liệu
AES	Advance Encryption Standard	Tiêu chuẩn mã hóa tiên tiến
TLS	Transport Layer Security	Giao thức bảo mật dữ liệu truyền trên mạng Internet

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH VẼ

LỜI NÓI ĐẦU

Cùng với sự phát triển của xã hội ngày nay. Công nghệ thông tin ngày càng phát triển bùng nổ, khi sử dụng máy tính, thông thường ta lưu trữ thông tin dưới dạng tập tin, thư mục khác nhau. Ngoài ra, dữ liệu còn lưu trữ dưới dạng các bảng có quan hệ với nhau hay chính là lưu trữ trên các hệ quản trị cơ sở dữ liệu để có thể thực thi các thao tác như: Chỉnh sửa, thêm, xoá, tìm kiếm dữ liệu được dễ dàng và nhanh gọn. Hệ quản trị cơ sở dữ liệu (CSDL) là một trong những kiến thức có ứng dụng quan trọng và được sử dụng phổ biến trong hầu hết mọi lĩnh vực được thiết kế để quản lý cơ sở dữ liệu một cách tự động và có trật tự.

Trong một số cơ quan, tổ chức của nhà nước tính bí mật của dữ liệu cần được bảo đảm tuyệt đối. Điều đó cho thấy tầm quan trọng của việc bảo vệ dữ liệu trong một hệ quản trị CSDL. Hiện nay có rất nhiều hệ quản trị CSDL mã nguồn mở với chi phí triển khai thấp. Và hệ quản trị CSDL PostgreSQL được phần lớn các doanh nghiệp thuộc các khối viễn thông, ngân hàng, chính phủ sử dụng vì PostgreSQL có các tính năng như:

- Câu truy vấn phức hợp
- Thủ tục sự kiện
- Tính toàn vẹn của các giao dịch
- Việc kiểm tra truy cập đồng thời đa phiên bản
- Truy vấn xử lý song song
- Sao chép dữ liệu dạng luồng

Trong HQTCSDL PostgreSQL còn hỗ trợ nhiều các cơ chế an toàn khác nhau trong đó có các cơ chế mã hoá đảm bảo tính bí mật dữ liệu như cơ chế mã TDE hay mã hoá kênh truyền với TLS1.3. Việc tìm hiểu rõ những cơ chế mã hoá này là điều cần thiết đối với quản trị viên để có thể đảm bảo an toàn cho HQTCSDL. Dựa trên lý do đó, em đã chọn đề tài "**Tìm hiểu, triển khai một số cơ chế mã hoá dữ liệu trong HQTCSDL PostgreSQL**" để làm đồ án tốt nghiệp.

CHƯƠNG 1. TỔNG QUAN VỀ HQTCSDL POSTGRESQL

1.1. Giới thiệu chung về HQTCSDL PostgreSQL

PostgreSQL là hệ quản trị cơ sở dữ liệu được viết theo hướng mã nguồn mở và rất mạnh mẽ. Hệ quản trị cơ sở dữ liệu này đã có hơn 15 năm phát triển, đồng thời cấu trúc đã được kiểm chứng và tạo được lòng tin với người sử dụng về độ tin cậy, tính toàn vẹn dữ liệu, và tính đúng đắn. PostgreSQL có thể chạy trên tất cả các hệ điều hành, bao gồm cả Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), và Windows. Có hỗ trợ đầy đủ các foreign keys, joins, views, triggers, và stored procedures (trên nhiều ngôn ngữ). Hệ quản trị này còn bao gồm các kiểu dữ liệu SQL: 2008 như INTEGER, NUMBER, BOOLEAN, CHAR, VARCHAR, DATE INTERVAL, và TIMESTAMPs. PostgreSQL cũng hỗ trợ lưu trữ các đối tượng có kiểu dữ liệu nhị phân lớn, bao gồm cả hình ảnh, âm thanh, hoặc video. Hệ quản trị cơ sở dữ liệu này được sử dụng thông qua giao diện của các ngôn ngữ C / C + +, Java,. Net, Perl, Python, Ruby, Tcl, ODBC…

Hình 1.1. PostgreSQL

Là một hệ quản trị cơ sở dữ liệu mạnh, PostgreSQL có các tính năng phức tạp như kiểm soát truy cập đồng thời nhiều phiên bản (MVCC), khôi phục dữ liệu tại từng thời điểm (Recovery), quản lý dung lượng bảng (tablespaces), sao chép không đồng bộ, giao dịch lồng nhau (savepoints), sao lưu trực tuyến hoặc nội bộ, truy vấn phức tạp và tối ưu hóa, viết trước các khai báo để quản lý và gỡ lỗi. PostgreSQL hỗ trợ bộ ký tự quốc tế, hỗ trợ bảng mã nhiều byte, Unicode, và cho phép định dạng, sắp xếp và phân loại ký tự văn bản (chữ hoa, thường). PostgreSQL

còn được biết đến với khả năng mở rộng để nâng cao cả về số lượng dữ liệu quản lý và số lượng người dùng truy cập đồng thời. Đã từng có những hệ thống PostgreSQL hoạt động trong môi trường thực tế thực hiện quản lý vượt quá 4 terabyte dữ liệu.

Dưới đây là thông tin về giới hạn khả năng của PostgreSQL.

Bảng 1. Giới hạn khả năng của PostgreSQL

Giới hạn	Giá trị
Dung lượng tối đa của cơ sở dữ liệu	Không giới hạn
Dung lượng bảng tối đa	32 TB
Dung lượng tối đa của cột	1,6 TB
Dung lượng tối đa của trường	1 GB
Hàng tối đa mỗi Bảng	Không giới hạn
Số cột tối đa của mỗi bảng	250 - 1600 tùy thuộc vào loại cột
Chỉ số tối đa của mỗi bảng	Không giới hạn

Vài nét về lịch sử của PostgreSQL

Hệ thống quản lý cơ sở dữ liệu đối tượng-quan hệ bây giờ được gọi là PostgreSQL có nguồn gốc từ các gói Postgres viết tại Đại học California ở Berkeley. Với hơn hai thập kỷ phát triển, PostgreSQL bây giờ là cơ sở dữ liệu mã nguồn mở tiên tiến nhất sẵn sàng ở bất cứ đâu.

Dự án Postgres, do Giáo sư Michael Stonebraker dẫn dắt, được Cơ quan Dự án Tìm hiểuCao cấp Quốc phòng - DARPA (Defense Advanced Research Projects Agency), Văn phòng Tìm hiểuQuân đội - ARO (Army Research Office), Quỹ Khoa học Quốc gia – NSF (National Science Foundation), và ESL Inc., tài trợ. Dự án Postgres bắt đầu triển khai vào năm 1986. Các khái niệm ban đầu cho hệ thống đã được trình bày trong “Các thiết kế của Postgres- Stonebraker and Rowe, 1986”, và định nghĩa của mô hình dữ liệu ban đầu đã xuất hiện trong “ Các mô hình dữ liệu Postgres - Rowe and Stonebraker, 1987”. Thiết kế của hệ thống các qui tắc khi đó đã được mô tả trong “Các thiết kế của hệ thống quy tắc Postgres - Stonebraker, Hanson, Hong, 1987”. Nhân tố căn bản và kiến trúc của người quản lý kho lưu trữ

đã được trình bày chi tiết trong "Các thiết kế của hệ thống lưu trữ Postgres - Stonebraker, 1987".

Postgres đã trải qua vài phiên bản chính kể từ đó. Hệ thống "phần mềm trình diễn" (demoware) đầu tiên đã hoạt động vào năm 1987 và đã được trình bày tại Hội nghị ACM-SIGMOD 1988. Phiên bản 1, được mô tả trong "Việc triển khai Postgres - Stonebraker, Rowe, Hirohama, 1990", được phát hành với một vài người dùng bên ngoài trong tháng 6 năm 1989. Để đáp ứng với một bài phê bình của hệ thống quy tắc đầu tiên, hệ thống quy tắc đã được thiết kế lại, và phiên bản 2 đã được phát hành vào tháng Sáu năm 1990 với hệ thống qui tắc mới. Phiên bản 3 xuất hiện vào năm 1991 và đã bổ sung hỗ trợ cho người quản lý nhiều kho lưu trữ, thi hành truy vấn được cải thiện, và hệ thống quy tắc được viết lại. Đối với hầu hết các phần, phiên bản tiếp theo cho đến Postgres 95 (xem bên dưới) tập trung vào tính di động và độ tin cậy.

POSTGRES đã dùng để thực hiện nhiều ứng dụng sản xuất và Tìm hiểukhác nhau. Chúng bao gồm: Một hệ thống phân tích cơ sở dữ liệu tài chính, một gói giám sát hiệu năng của động cơ phản lực, một cơ sở dữ liệu theo dõi các hành tinh nhỏ, một cơ sở dữ liệu thông tin y tế, và vài hệ thống thông tin địa lý, POSTGRES cũng đã được sử dụng như một công cụ giáo dục tại một số trường đại học. Cuối cùng, Illustra Information Technologies (sau này sát nhập vào Informix, mà bây giờ thuộc sở hữu của IBM) đã chọn mã và thương mại hóa nó. Vào cuối năm 1992, POSTGRES đã trở thành trình quản lý cơ sở dữ liệu chính cho dự án tính toán khoa học Sequoia 2000. Quy mô của cộng đồng người dùng bên ngoài tăng gần gấp đôi trong năm 1993. Rõ ràng là việc bảo trì các mã mẫu và sự hỗ trợ đã chiếm rất nhiều thời gian mà lẽ ra phải được dành cho Tìm hiểucơ sở dữ liệu. Trong nỗ lực để giảm bớt gánh nặng hỗ trợ này, các dự án Postgres ở Berkeley chính thức kết thúc với phiên bản 4.2.

Vào năm 1994, Andrew Yu và Jolly Chen đã bổ sung thêm một trình biên dịch ngôn ngữ SQL vào POSTGRES. Dưới cái tên mới, Postgres95 sau đó đã được phát hành lên web để tìm ra cách đi của riêng mình trên thế giới như một hậu duệ nguồn mở của mã POSTGRES ở Berkeley. Mã của Postgres95 đã hoàn toàn là ANSI C và được giảm kích thước tới 25%. Nhiều thay đổi nội bộ đã tăng hiệu suất và khả năng bảo trì. Postgres95 phát hành bản 1.0.x chạy nhanh hơn khoảng 30-50% so với chuẩn Wisconsin so với Postgres, phiên bản 4.2. Ngoài việc sửa lỗi, sau đây là những cải tiến quan trọng:

- Ngôn ngữ truy vấn PostQUEL đã được thay thế bằng SQL (được triển khai trong máy chủ).Các truy vấn phụ (Subqueries) đã được hỗ trợ cho tới PostgreSQL, nhưng chúng có thể được mô phỏng trong Postgres95 với các hàm SQL do người sử dụng định nghĩa. Các hàm tổng hợp đã được tái triển khai. Hỗ trợ cho câu truy vấn GROUP BY cũng đã được bổ sung.

- Một chương trình mới (psql) đã được đưa ra cho các truy vấn SQL tương tác, nó sử dụng GNU Readline. Điều này đã thay thế phần lớn chương trình giám sát cũ.

- Một thư viện (front-end) mới "libpgtcl", được các máy trạm dựa vào Tcl hỗ trợ. Một trình biên dịch (shell) mẫu, pgtclsh, đã cung cấp các lệnh Tcl mới cho các chương trình giao tiếp Tcl với máy chủ Postgres95.

- Giao diện đối tượng lớn đã được kiểm tra kỹ lưỡng. Các đối tượng nghịch đảo lớn đã chỉ còn là cơ chế cho việc lưu trữ các đối tượng lớn. (Hệ thống tệp nghịch đảo đã được loại bỏ).

- Một sách chỉ dẫn ngắn gọn giới thiệu các tính năng SQL thông thường cũng như các tính năng của Postgres95 đã được phát hành cùng với mã nguồn.

Tới năm 1996, rõ ràng cái tên "Postgres95" không phù hợp theo thời gian. PostgreSQL là tên mới được chọn, để phản ánh mối quan hệ giữa POSTGRES gốc ban đầu và các phiên bản gần đây với SQL. Đồng thời, thiết lập việc đánh số phiên bản bắt đầu từ 6.0, đưa con số trở lại trình tự xuất phát ban đầu của dự án POSTGRES ở Berkeley.

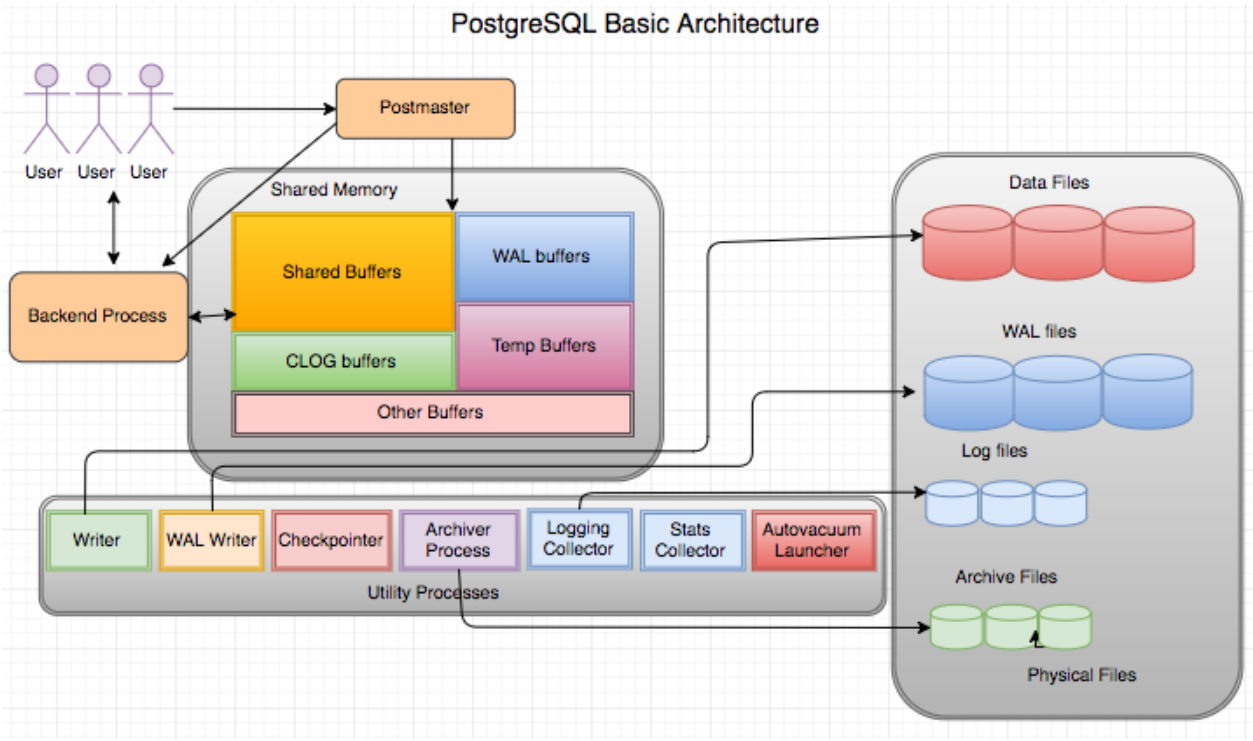
Nhiều người vẫn coi PostgreSQL như "Postgres" vì truyền thống hay bởi vì nó dễ phát âm. Việc sử dụng này được chấp nhận rộng rãi như một biệt danh hoặc bí danh.

Điểm nhấn của Postgres95 trong quá trình phát triển là việc xác định và hiểu được các vấn đề đang tồn tại trong mã của máy chủ. Với PostgreSQL, trọng tâm chuyển sang nâng cao tính năng, mặc dù vẫn tiếp tục làm việc trong tất cả các lĩnh vực.

1.2. Kiến trúc, thành phần PostgreSQL

Trong phần này sẽ trình bày về kiến trúc và thành phần của hệ quản trị cơ sở dữ liệu PostgreSQL.

Kiến trúc của PostgreSQL được thể hiện trong hình sau:



Hình 1.2. Kiến trúc PostgreSQL

Kiến trúc của PostgreSQL là một kiến trúc với rất nhiều thành phần liên kết với nhau, bao gồm các phần chính: Bộ nhớ chung (Shared Memory), các tiến trình nền (Utility Processes), các file dữ liệu (Physical Files).

1.2.1. Bộ nhớ chung

Bộ nhớ chung (Shared Memory) là bộ nhớ dành riêng cho lưu trữ cơ sở dữ liệu và nhật ký giao dịch. Các thành phần quan trọng của bộ nhớ chung có thể kể đến là Shared Buffer,WAL Buffer, CLOG Buffer, Temp Buffer, Work Memory và Vacuum Buffer.

- Shared Buffer:

Thao tác đọc và ghi trong bộ nhớ luôn nhanh hơn bất kỳ thao tác nào khác. Do đó các cơ sở dữ liệu luôn cần bộ nhớ để truy cập nhanh dữ liệu, mỗi khi có truy cập READ và WRITE xuất hiện. Trong PostgreSQL chính là Shared Buffer (được điều khiển bởi tham số shared_buffers). Dung lượng RAM được cấp phát cho Shared Buffer là cố định trong suốt thời gian chạy PostgreSQL. Shared Buffer có thể được truy cập bởi tất cả tiến trình server và người dùng kết nối đến cơ sở dữ liệu.

Dữ liệu được ghi hay chỉnh sửa trong Shared Buffer được gọi là dirty data, và các đơn vị thao tác trong cơ sở dữ liệu block (hay page) thay đổi được gọi là

dirty block hay dirty page. Dirty data sẽ được ghi vào file vật lý liên tục trên ở đĩa, các file này được gọi là file dữ liệu (data file).

Mục đích của Shared Buffer là để giảm thiểu các tác vụ I/O lên đĩa (DISK IO). Để đạt được mục đích đó, nó phải đáp ứng được những yêu cầu sau:

+ Phải truy cập bộ nhớ đệm lớn (hàng chục, trăm gigabites) nhanh chóng.

+ Tối thiểu hoá xung đột khi nhiều người dùng truy cập cùng lúc.

+ Các blocks được sử dụng thường xuyên phải ở trong bộ đệm càng lâu càng tốt.

- WAL Buffer:

WAL Buffer còn gọi là transaction log buffers, là bộ nhớ đệm để lưu trữ dữ liệu WAL. Dữ liệu WAL là thông tin về những thay đổi đối với dữ liệu thực tế và dùng để tạo lại dữ liệu trong quá trình sao lưu và phục hồi cơ sở dữ liệu. Dữ liệu WAL được ghi trong file vật lý ở các vị trí liên tục gọi là WAL segments hoặc checkpoint segments.

WAL Buffer được điều khiển bởi tham số wal_buffers, nó được cấp phát bởi RAM của hệ điều hành. Mặc dù nó cũng có thể được truy cập bởi tất cả tiến trình server và người dùng, nhưng nó không phải là một phần của Shared Bufer. WAL Buffer nằm ngoài Shared Buffer và rất nhỏ nếu so sánh với Shared Buffer. Dữ liệu WAL lần đầu tiên sửa đổi sẽ được ghi vào WAL Buffer trước khi được ghi vào WAL Segments trên ổ đĩa. Theo thiết lập mặc định, nó sẽ được phân bổ với kích thước bằng 1/16 Shared Buffer.

- CLOG Buffer:

CLOG là viết tắt của commit log, và CLOG Buffer là bộ đệm dành riêng cho lưu trữ các trang commit log được cấp phát bởi RAM của hệ điều hành. Các trang commit log chứa nhật ký về giao dịch và các thông tin khác từ dữ liệu WAL. Các commit log chứa trạng thái commit của tất cả giao dịch và cho biết một giao dịch đã hoàn thành hay chưa.

Không có tham số cụ thể để kiểm soát vùng nhớ này. Sẽ có công cụ cơ sở dữ liệu tự động quản lý với số lượng rất nhỏ. Đây là thành phần nhớ dùng chung, có thể được truy cập bởi tất cả tiến trình server và người dùng của csdl PostgreSQL.

- Memory for Lock:

Thành phần nhớ này là để lưu trữ tất cả các khóa (lock) nặng được sử dụng bởi PostgreSQL. Các khoá này được chia sẻ trên tất cả tiến trình server hay user kết nối đến csdl. Một thiết lập giữa hai tham số là max_locks_per_transaction và

max_pred_locks_per_transaction sẽ ảnh hưởng theo một cách nào đó đến kích thước của bộ nhớ này.

- Vacuum Buffer:

Đây là lượng bộ nhớ tối đa được sử dụng cho mỗi tiến trình autovacuum worker, được điều khiển bởi tham số autovacuum_work_mem. Bộ nhớ được cấp phát bởi RAM của hệ điều hành. Tất cả thiết lập tham số chỉ có hiệu quả khi tiến trình auto vacuum được bật, nếu không các thiết lập này sẽ không ảnh hưởng đến VACUUM đang chạy ở ngữ cảnh khác. Thành phần nhớ này không được chia sẻ bởi bất kỳ tiến trình máy chủ hay người dùng nào.

- Work Memory:

Đây là bộ nhớ dành riêng cho một thao tác sắp xếp hoặc bảng băm cho một truy vấn nào đó, được điều khiển bởi tham số work_mem. Thao tác sắp xếp có thể là ORDER BY, DISTINCT hay MERGE JOIN. Thao tác trên bảng băm có thể là hash-join hoặc truy vấn IN.

Các câu truy vấn phức tạp hơn như nhiều thao tác sắp xếp hoặc nhiều bảng băm có thể được cấp phát bởi tham số work_mem. Vì lý do đó không nên khai báo work_mem với giá trị quá lớn, vì nó có thể dẫn đến việc sử dụng vùng nhớ của hệ điều hành chỉ cho một câu truy vấn lớn, khiến hệ điều hành thiếu RAM cho các tiến trình cần thiết khác.

- Maintenance Work Memory:

Đây là lượng nhớ tối đa mà RAM sử dụng cho các hoạt động bảo trì (maintenance). Các hoạt động bảo trì có thể là VACUUM, CREATE INDEX hay FOREIGN KEY, và được kiểm soát bởi tham số maintenance_work_mem.

Một phiên cơ sở dữ liệu chỉ có thể thực hiện bất kỳ hoạt động bảo trì nào đã đề cập ở trên tại một thời điểm và PostgreSQL thường không thực hiện đồng thời nhiều hoạt động bảo trì như vậy. Do đó tham số này có thể thiết lập lớn hơn nhiều so với tham số work_mem.

- Temp Buffer:

Các cơ sở dữ liệu cần một hay nhiều bảng mẫu, và các block(page) của bảng mẫu này cần nơi để lưu trữ. Temp Buffer sinh ra nhằm mục đích này, bằng cách sử dụng một phần RAM, được xác định bởi tham số temp_buffer.

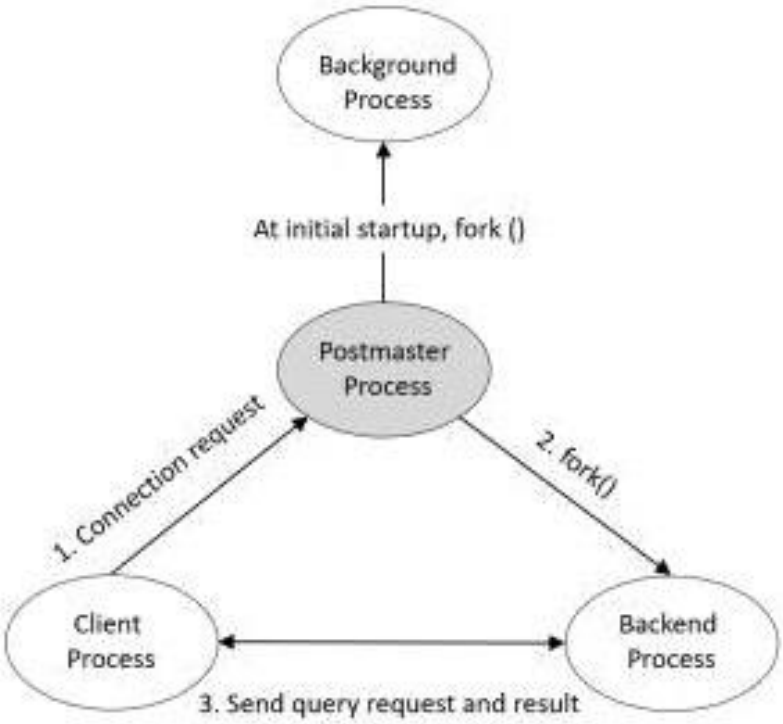
Temp Buffer chỉ được sử dụng để truy cập bảng tạm thời trong phiên người dùng. Không có liên hệ gì giữa temp buffer với các file mẫu được tạo trong thư mục pgsql_tmp để thực hiện sắp xếp lớn hay bảng băm.

1.2.2. Các tiến trình nền

PostgreSQL có 4 tiến trình: Postmaster (Daemon) Process, Background Process, Backend Process, Client Process.

- Postmaster Process:

Postmaster Process là tiến trình khởi tạo đầu tiên sau khi PostgreSQL khởi động. Tiến trình này đảm nhiệm việc khởi động hoặc dừng các tiến trình khác nếu cần hoặc có yêu cầu. Sau khi khởi động postmaster process sẽ khởi động các background process.



Hình 1.3. Các tiến trình của PostgreSQL

- Background Process

Background process là tiến trình có nhiều tiến trình nền con, mỗi tiến trình đều đảm nhận các nhiệm vụ khác nhau để giúp cho cơ sở dữ liệu hoạt động ổn định

Dưới đây là các tiến trình nền con:

+ Background writer: Tiến trình này kết hợp với checkpointer để đảm bảo việc ghi dữ liệu từ bộ đệm xuống vùng lưu trữ. Thông thường khi checkpoint không hoạt động, tiến trình này sẽ ghi từng chút một dữ liệu xuống vùng lưu trữ.

+ Checkpointer: Tiến trình này chủ yếu giữ vai trò thực hiện checkpoint (đồng bộ dữ liệu từ bộ nhớ đệm xuống vùng lưu trữ) khi cần thiết.

+ Autovacuum launcher: Tiến trình này hoạt động khi tham số autovacuum = on. Nó thực hiện chức năng tự động lấy vùng dữ liệu dư thừa sau khi DELETE hoặc UPDATE dữ liệu. Tiến trình này khởi động các VACUUM worker processes sau mỗi autovacuum_naptime. Các VACUUM worker processes sẽ thực hiện việc VACUUM dữ liệu trên các database.

+ WAL writer: Đảm nhiệm việc đồng bộ WAL từ bộ nhớ đệm xuống vùng lưu trữ. Thông thường WAL sẽ được ghi từ bộ đệm xuống vùng lưu trữ khi transaction được commit. Nếu dữ liệu đệm của WAL trên bộ nhớ đệm vượt quá tham số wal_buffers, dữ liệu WAL trên vùng nhớ đệm sẽ tự động ghi xuống vùng lưu trữ dữ liệu thông qua tiến trình này.

+ Statistics collector: Tiến trình này thực hiện vai trò lưu trữ các thông tin thống kê hoạt động của PostgreSQL và cập nhật vào các system catalog (thông tin nội bộ của PostgreSQL hiện diện bởi các bảng pg_stat_all_tables hoặc pg_stat_activity).

+ Logging collector: Tiến trình này đảm nhiệm việc ghi log của PostgreSQL.

+ Archiver: Khi ở chế độ Archive.log, tệp WAL sẽ được sao chép vào thư mục được chỉ định.

- Backend Process:

Số lượng tối đa backend process được thiết lập bởi tham số max_connections có giá trị mặc định là 100. Backend process thực hiện yêu cầu truy vấn của user process, sau đó truyền kết quả. Một số cấu trúc bộ nhớ được yêu cầu để thực thi truy vấn, được gọi là bộ nhớ cục bộ (local memory). Các tham số chính liên quan đến bộ nhớ cục bộ là:

+ work_mem được sử dụng cho điều chỉnh Work Memory. Thiết lập mặc định là 4 MB.

+ maintenance_work_mem được sử dụng cho điều chỉnh Maintenance Work Memory. Thiết lập mặc định là 64 MB.

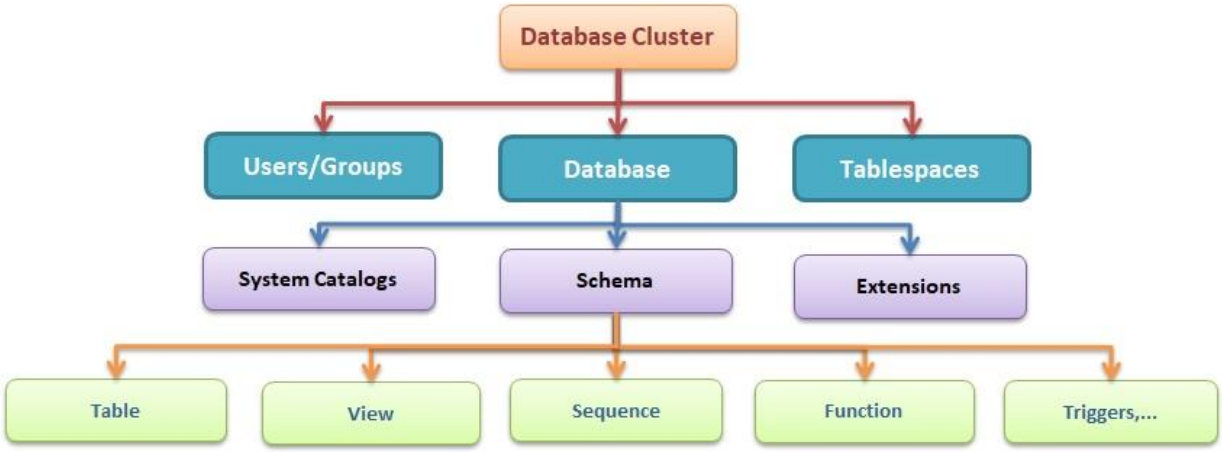
+ temp_buffers được sử dụng cho điều chỉnh Temp Buffer. Thiết lập mặc định là 8 MB.

- Client Process:

Client Process là tiến trình được thể hiện cho mội kết nối với người dùng. Thông thường, postmaster process sẽ phân ra một tiến trình con dành riêng phục vụ cho kết nối người dùng.

1.2.3. Cấu trúc Database

Cấu trúc cơ sở dữ liệu của hệ quản trị cơ sở dữ liệu PostgreSQL được thể hiện như hình dưới đây:



Hình 1.4. Cấu trúc database

Cấu trúc cơ sở dữ liệu được chia thành các phần như sau:

- Database Cluster

Database Cluster là đơn vị lưu trữ lớn nhất của một máy chủ cơ sở dữ liệu PostgreSQL. Database cluster được tạo ra bởi câu lệnh initdb(), bao gồm các files config (PostgreSQL.conf, pg_hba.conf, ...), và tất cả các đối tượng lưu trữ đều nằm trong database cluster.

- Users/Groups, Database, Tablespaces

- Database: Là đơn vị lớn sau Database cluster. Để thực hiện được câu truy vấn, bạn phải truy cập vào một database nào đó. Khi initdb() thực thi, mặc định PostgreSQL sẽ tạo ra 3 cơ sở dữ liệu là template0, template1 và postgres. Với:

+ template0 là cơ sở dữ liệu mẫu. Không thể truy nhập và chỉnh sửa các đối tượng trong đó. Người dùng có thể tạo database mới dựa trên template0 này bằng cách chỉ định TEMPLATE trong câu lệnh CREATE DATABASE.

+ template1 là cơ sở dữ liệu mẫu. Người dùng có thể truy nhập và chỉnh sửa các đối tượng trong đó. Khi thực hiện câu lệnh "CREATE DATABASE", PostgreSQL sẽ copy template1 này để tạo database mới.

+ postgres: Cơ sở dữ liệu mặc định của PostgreSQL khi tạo database cluster.

- Tablespace: Là đơn vị lưu trữ dữ liệu về phương diện vật lý bên dưới database. Thông thường dữ liệu vật lý được lưu trữ tại thư mục dữ liệu (nơi người dùng chỉ định lúc ta tạo database cluster). Nhưng có một phương pháp lưu trữ dữ liệu ngoài phân vùng này, nhờ sử dụng chức năng TABLESPACE. Khi tạo một TABLESPACE tức là ta đã tạo ra một vùng lưu trữ dữ liệu mới độc lập với dữ liệu bên dưới thư mục dữ liệu.

- Schema: Là đơn vị lưu trữ bên dưới database, quản lý dữ liệu dưới dạng logic. Mặc định trong mỗi database có một schema cho người sử dụng, đó là schema public. Ta có thể tạo schema bằng câu lệnh **CREATE SCHEMA**. Đặc điểm của 1 schema như sau:

+ Có thể sử dụng tên trùng với schema ở database khác nhưng không trùng tên trên cùng database.

+ Ngoài TABLESPACE và user ra, schema có thể chứa hầu hết các đối tượng còn lại (như table, index, sequence, constraint...) để truy cập schema ta có thể thêm tên schema vào phía trước đối tượng muốn truy cập hoặc sử dụng tham số search_path để thay đổi schema truy cập hiện tại.

+ Schema có thể sử dụng với các mục đích như tăng cường bảo mật, quản lý dữ liệu dễ dàng hơn.

- Bảng (Table): Bảng là đối tượng lưu trữ dữ liệu từ người dùng. Một bảng bao gồm 0 hoặc nhiều cột (column) tương ứng với từng kiểu dữ liệu khác nhau của PostgreSQL. Tổng quan có 3 kiểu tables mà PostgreSQL hỗ trợ, đó là:

+ unlogged table: là kiểu table mà các thao tác đối với bảng dữ liệu này không được lưu trữ vào WAL. Tức là không có khả năng phục hồi nếu bị hỏng.

+ temporary table: là kiểu table chỉ được tạo trong phiên làm việc đó. Khi kết nối bị ngắt, nó sẽ tự động mất đi.

+ table thông thường: Khác với 2 kiểu table trên, là loại table thông thường để lưu trữ dữ liệu. Có khả năng phục hồi khi bị hỏng và tồn tại vĩnh viễn nếu không có thao tác xóa bỏ nào.

1.3. Một số đặc trưng chính của PostgreSQL

1.3.1. Các tính năng

PostgreSQL tích hợp nhiều tính năng tuyệt vời giúp hỗ trợ nhà phát triển xây dựng ứng dụng đáp ứng các chức năng phức tạp, truy vấn nhanh chóng và bảo mật duy trì tính toàn vẹn và độ tin cậy. Để đáng tin cậy hơn, PostgreSQL cung cấp các

tùy chọn bảo mật, xác thực và khôi phục thảm họa khác nhau. PostgreSQL được chứng minh là có khả năng mở rộng cao cả về số lượng dữ liệu và số lượng người dùng có thể thao tác cùng lúc.

❖ Các tính năng chính

- Câu truy vấn phức hợp (complex query)
- Thủ tục sự kiện (trigger)
- Các khung nhìn (view)
- Tính toàn vẹn của các giao dịch (integrity transactions)
- Việc kiểm tra truy cập đồng thời đa phiên bản (multiversion concurrency control)
- Truy vấn xử lý song song (parallel query)
- Sao chép dữ liệu dạng luồng (Streaming replication)

❖ Các tính năng khác

- Kiểu dữ liệu:

+ Nguyên hàm: Số nguyên, số, chuỗi, Boolean
+ Cấu trúc: Date/Time, Array, Phạm vi
+ Document: JSON/JSONB, XML, Key-value (Hstore)
+ Hình học: Điểm, Đường thẳng, Vòng tròn, Đa giác
+ Tùy chỉnh: Composite, Các kiểu tùy chỉnh

- Toàn vẹn dữ liệu:

+ UNIQUE, NOT NULL
+ Primary Keys
+ Foreign Keys
+ Ràng buộc loại trừ
+ Khóa hàm số, Khóa khuyến nghị
+ Đồng quy, hiệu suất
+ Lập danh mục: B-tree, Multicolumn, Expressions, Partial
+ Lập danh mục nâng cao: GiST, SP-Gist, KNN Gist, GIN, BRIN, Bloom filters
+ Trình lập kế hoạch / trình tối ưu hóa truy vấn phức tạp, quét index-only, thống kê số liệu trên nhiều cột.
+ Giao tác, Giao tác dạng nest (thông qua lưu điểm)
+ Điều khiển đồng thời nhiều phiên bản (MVCC)
+ Truy vấn đọc song song
+ Phân vùng bảng

+ Tất cả các mức độ giao dịch độc lập được xác định trong tiêu chuẩn SQL, bao gồm cả Serializable

+ Độ tin cậy, phục hồi sau thảm hoạ

+ Ghi nhật ký ghi trước (Write-ahead Logging - WAL)

+ Replication: Không đồng bộ, Đồng bộ, Logical

+ Khôi phục điểm-theo-thời gian (Point-in-time-recovery - PITR), active standbys

+ Không gian bảng

- Bảo mật:

+ Xác thực: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, Certificate và các hình thức khác

+ Hệ thống kiểm soát truy cập mạnh mẽ

+ Bảo mật cấp độ cột và hàng

- Khả năng mở rộng:

+ Phương pháp lưu trữ

+ Ngôn ngữ thủ tục: PL / PGSQL, Perl, Python (và nhiều ngôn ngữ khác)

+ Trình liên kết dữ liệu ngoài: kết nối với các cơ sở dữ liệu hoặc luồng khác với giao diện SQL chuẩn

+ Tiện ích mở rộng cung cấp chức năng bổ sung, bao gồm cả PostGIS

+ Hỗ trợ các bộ ký tự quốc tế, ví dụ: thông qua ICU collations

+ Tìm kiếm văn bản đầy đủ

1.3.2. Các câu lệnh chính

Hệ quản trị cơ sở dữ liệu PostgreSQL cung cấp các câu lệnh chính như sau:

- **CREATE TABLE**

CREATE TABLE là từ khóa để khởi tạo một bảng rỗng mới trong CSDL. Bảng này sẽ thuộc sở hữu của user đã viết ra câu lệnh này. Ví dụ như sau:

```
postgres=# create table dummy_table (name varchar(20), address text, age int);
CREATE TABLE
```

Khi sử dụng lệnh CREATE TABLE, lệnh đã tạo ra một bảng dummy_table với các thuộc tính:

+ name có kiểu dữ liệu là varchar

+ address có kiểu dữ liệu là text

+ age có kiểu dữ liệu là int

- **INSERT**

Câu lệnh INSERT được sử dụng để chèn dữ liệu vào một bảng. Ví dụ:

```
postgres=# insert into dummy_table values('XYZ','location-A',25);
INSERT 0 1
postgres=# insert into dummy_table values('ABC','location-B',35);
INSERT 0 1
postgres=# insert into dummy_table values('DEF','location-C',40);
INSERT 0 1
postgres=# insert into dummy_table values('PQR','location-D',54);
INSERT 0 1
```

Như vậy đã dùng 4 câu lệnh insert để chèn thêm bốn hàng dữ liệu vào bảng dummy_table. Các dữ liệu được chèn tương ứng với 3 thuộc tính của bảng khi tạo bảng.

- **SELECT**

Câu lệnh SELECT (khi không có tùy chọn WHERE) sẽ tìm đến hết tất cả các dữ liệu có trong bảng:

```
postgres=# select * from dummy_table;
name |  address      | age
--------+--------------+ -----
XYZ   | location-A |  25
ABC   | location-B |  35
DEF   | location-C |  40
PQR   | location-D |  54
(4 rows)
```

Nếu câu lệnh có tùy chọn Where, kết quả truy vấn sẽ trả về các kết quả thỏa mãn điều kiện trong Where. Ví dụ:

```
postgres=# select from dummy_table where age <40;
Câu lệnh truy vấn sẽ trả về kết quả như sau:
name |  address      | age
--------+--------------+ -----
XYZ   | location-A |  25
```

```
ABC   | location-B |  35
```

- **UPDATE**

Câu lệnh UPDATE được sử dụng để cập nhật các dữ liệu trong bảng. Trong ví dụ dưới đây, sử dụng UPDATE để thay đổi tuổi của một người có tên là 'PQR':

```
postgres=# update dummy_table set age=50 where name='PQR';
UPDATE 1
postgres=# select * from dummy_table;
name |  address   | age
--------+--------------+-------
XYZ  | location-A |  25
ABC  | location-B |  35
DEF  | location-C |  40
PQR  | location-D |  50
(4 rows)
```

Tiếp theo, sử dụng lệnh UPDATE để thay đổi tên và tuổi của một người có địa chỉ là 'location-D':

```
postgres=# update dummy_table set name='GHI', age=54 where address='location-D';
UPDATE 1
postgres=# select * from dummy_table;
 name |  address   | age
------+------------+-----
 XYZ  | location-A |  25
 ABC  | location-B |  35
 DEF  | location-C |  40
 GHI  | location-D |  54
(4 rows)
```

Nếu muốn sửa đổi tất cả các giá trị trong address và cột age trong bảng dummy_table, thì chúng ta không cần sử dụng mệnh đề WHERE. Truy vấn UPDATE sẽ trông giống như sau:

```
postgres=# update dummy_table set age=54, address='location-X';
```

```
UPDATE 4
postgres=# select * from dummy_table ;
 name |  address   | age
------+------------+--------
 XYZ  | location-X |  54
 ABC  | location-X |  54
 DEF  | location-X  |  54
 GHI  | location-X   |  54
(4 rows)
```

Mệnh đề RETURNING trả về các hàng đã cập nhật. Đây là tùy chọn trong UPDATE:

```
postgres=# update dummy_table set age=30 where name='XYZ' returning age as age_no;
 age_no
---------
30
(1 row)
```

- **DELETE**

Câu lệnh DELETE được sử dụng để xóa các hàng trong bảng. Điều kiện WHERE tùy chọn, nếu thiếu điều kiện WHERE, lệnh sẽ xóa tất cả các hàng, để lại cho bạn một bảng trống.

```
postgres=# delete from dummy_table where age=65;
DELETE 1
```

- **DROP TABLE**

Lệnh DROP TABLE này được sử dụng để xóa một bảng khỏi cơ sở dữ liệu:

```
PostgreSQL=# drop table if exists dummy;
```

```
NOTICE:  table "dummy" does not exist, skipping
DROP TABLE
```

Lệnh này đã xóa toàn bộ bảng, bao gồm mọi dữ liệu liên quan, chỉ mục, quy tắc và ràng buộc cho bảng đó.

1.4. Ưu nhược điểm của PostgreSQL

- PostgreSQL là hệ quản trị cơ sở dữ liệu mã nguồn mở với những ưu điểm sau:

+ Dễ sử dụng.

+ Hỗ trợ nhiều kiểu dữ liệu.

+ Mã nguồn mở.

+ Rất nhiều sự hỗ trợ của cộng đồng.

+ Sử dụng các thủ tục đã lưu trữ.

+ Hỗ trợ ACID, tức là Tính nguyên tử, Tính nhất quán, Tính cô lập, Độ bền.

- Nhược điểm:

+ Sử dụng nhiều bộ nhớ.

+ Chưa phổ biến so với các hệ quản trị cơ sở dữ liệu khác.

+ Tốc độ chưa bằng các hệ quản trị cơ sở dữ liệu khác.

+ Cài đặt không dễ dàng cho người mới bắt đầu.

1.5. Kết luận chương 1

Chương 1 đã trình bày một cách chi tiết, rõ ràng về tổng quan hệ quản trị cơ sở dữ liệu PostgreSQL. Cụ thể đã trình bày được các nội dung sau:

- Giới thiệu về PostgreSQL.

- Kiến trúc, thành phần trong PostgreSQL.

- Một số đặc trưng chính trong PostgreSQL.

- Ưu nhược điểm của PostgreSQL.

Từ cơ sở trên đồ án sẽ tìm hiểu rõ thêm về những nguy cơ mất an toàn trong hệ quản trị cơ sở dữ liệu và cơ chế đảm bảo an toàn và cơ chế mã hóa trong hệ quản trị cơ sở dữ liệu PostgreSQL trong chương 2.

CHƯƠNG 2. CƠ CHẾ MÃ HOÁ TRONG HQTCSDL POSTGRESQL

2.1. Nguy cơ mất an toàn trong triển khai trong PostgreSQL

Trước tiên, có thể lý giải tại sao cơ sở dữ liệu luôn nằm trong tầm ngắm của nhiều tin tặc. Bởi vì, cơ sở dữ liệu lưu giữ những thông tin quan trọng như: thông tin của khách hàng, kế hoạch phát triển của một doanh nghiệp, các tập đoàn kinh tế, và nhiều mục đích quan trọng khác…Cơ sở dữ liệu còn chứa nhiều thông tin riêng tư của các cá nhân mà tin tặc có thể thu được bằng cách truy vấn những dữ liệu công khai. Trong phần này đồ án sẽ trình bày một số nguy cơ mất an toàn chung trong triển khai các dạng cơ sở dữ liệu nói chung và PostgreSQL nói riêng.

2.1.1. Tấn công bên trong và bên ngoài

Tấn công bên trong

Kẻ tấn công có thể là nhân viên hoặc quản trị viên của chính cơ quan, tổ chức triển khai máy chủ cơ sở dữ liệu.

+ Kẻ tấn công có quyền truy cập vật lý vào máy chủ cơ sở dữ liệu để chiếm giữ các bản ghi trong hệ quản trị cơ sở dữ liệu được triển khai. Thông thường các hệ quản trị cơ sở dữ liệu triển khai dưới dạng rõ, vì vậy kẻ tấn công có thể xảy ra ở hầu hết các cơ sở dữ liệu nói chung và cơ sở dữ liệu PostgreSQL nói riêng.

+ Để phòng chống tấn công bên trong cần thiết lập các biện pháp nghiệp vụ. Quy định về đặt máy chủ cơ sở dữ liệu tại vị trí địa lý riêng biệt và chỉ cho phép truy cập trực tiếp với những quản trị viên được phép truy cập hợp pháp.

+ Triển khai hệ thống tường lửa để phân chia vùng mạng logic tách biệt với các máy chủ dịch vụ khỏi các phân vùng của nhân viên trong cơ quan, công ty hoặc doanh nghiệp.

+ Triển khai hệ thống phát hiện và ngăn chặn xâm nhập mạng để phát hiện các luồng truy cập và những hành động bất thường truy cập vào hệ thống quản trị cơ sở dữ liệu trong PostgreSQL.

+ Triển khai hệ thống ghi nhật ký cho hệ thống quản trị cơ sở dữ liệu trong PostgreSQL, việc này là căn cứ để điều tra nguyên nhân hay thủ phạm gây ra các vụ tấn công từ bên trong này.

Tấn công bên ngoài

Thông thường kẻ tấn công sử dụng tấn công kỹ nghệ xã hội để có thể lấy được thông tin, tài khoản người dùng hợp pháp trong hệ quản trị cơ sở dữ liệu trong PostgreSQL. Từ đó kẻ tấn công có quyền hợp pháp đánh cắp dữ liệu người dùng. Để phòng chống tấn công này thì đối với cơ quan, tổ chức cần có quy định và biện pháp rõ ràng về quản lý tài khoản mật được an toàn.

2.1.2. Tấn công leo thang đặc quyền

Khi người dùng (hay ứng dụng) được gán các đặc quyền truy nhập cơ sở dữ liệu vượt quá các yêu cầu trong chức năng công việc của họ, thì những đặc quyền này có thể bị lạm dụng cho các mục đích xấu. Ví dụ, một người phụ trách cho cơ sở dữ liệu của trường đại học với công việc là thay đổi các thông tin liên lạc của sinh viên, người này có thể lạm dụng quyền của mình với quyền cập nhật cơ sở dữ liệu để sửa đổi điểm của sinh viên (trái phép).

Một lý do đơn giản là những quản trị viên cơ sở dữ liệu vì bận rộn với công việc quản trị của mình nên không có thời gian để định nghĩa và cập nhật cơ chế kiểm soát quyền truy nhập mặc định cho mỗi người dùng. Kết quả là một số lượng lớn người dùng được gán các đặc quyền truy nhập mặc định vượt xa so với yêu cầu công việc của họ.

Trong PostgreSQL khi một cơ sở dữ liệu được tạo ra của bất kỳ người dùng nào thì hệ thống đều cấp quyền truy cập cơ sở dữ liệu riêng cho ba người dùng mặc định là: Admin, writer và reader

- Admin: Người dùng này có quyền truy cập vào tất cả các chức năng trên cơ sở dữ liệu mà không bị giới hạn.
- Reader: Người dùng này là người dùng chỉ đọc. Người đọc có thể truy vấn bất kỳ bản ghi nào trong cơ sở dữ liệu, nhưng không thể sửa đổi hoặc xóa chúng. Nó không có quyền truy cập vào thông tin nội bộ.
- Writer: Người dùng này giống như Reader nhưng nó cũng có thể tạo, cập nhật và xóa các bản ghi.

Trong trường hợp này quản trị viên không thay đổi mật khẩu người dùng này thì mặc định kẻ tấn công có thể truy cập vào các cơ sở dữ liệu này một cách hợp pháp. Để ngăn chặn tấn công người quản trị cần phải thay đổi mật khẩu mặc định của các người dùng (Admin, Writer và Reader).

Ví dụ: Kẻ tấn công có thể thay đổi mật khẩu mặc định như sau:

+ Admin thay đổi mật khẩu mặc định với câu lệnh

postgres=# update OUser set password="newadminpass" where name="admin"
Updated 1

+ Reader thay đổi mật khẩu mặc định với câu lệnh

postgres=# update OUser set password="newreaderpass" where name="reader"
Updated 1

+ Writer thay đổi mật khẩu mặc định với câu lệnh

postgres=# *update OUser set password="newwriterpass" where name="writer"*
Updated 1

2.1.3. Tấn công dựa trên điểm yếu của nền tảng cài đặt

Hệ quản trị cơ sở dữ liệu PostgreSQL hiện nay hỗ trợ nhiều nền tảng cài đặt trên window, linux (centos, ubuntu) Unix, MAC.

Các nền tảng này thì đều sẽ có lỗ bảo mật đã tồn tại hoặc phát hiện trong tương lai. Kẻ tấn công có thể dựa vào các lỗ hỏng bảo mật để xâm nhập bất hợp pháp vào máy chủ cơ sở dữ liệu PostgreSQL trong trường hợp cơ sở dữ liệu lưu ở dạng rõ, kẻ tấn công hoàn toàn có thể đọc được nội dung của các bản ghi.

2.1.4. Tấn công xác thực yếu

Hệ quản trị cơ sở dữ liệu PostgreSQL cho phép người dùng sử dụng mật khẩu yếu để đăng ký, đăng nhập vào hệ thống.

Ví dụ: 1234, abcd… Kẻ tấn công có thể đoán và đăng nhập thử.

Theo mặc định trong PostgreSQL không hạn chế số lần đăng nhập của người dùng thì kẻ tấn công có thể sử dụng tấn công vét cạn để dò tìm ra mật khẩu của người dùng.

- Cách thức phòng chống tấn công

+ Yêu cầu ngưồi dùng hệ thống đặt mật khẩu mạnh, độ phức tạp lớn đảm bảo an toàn.

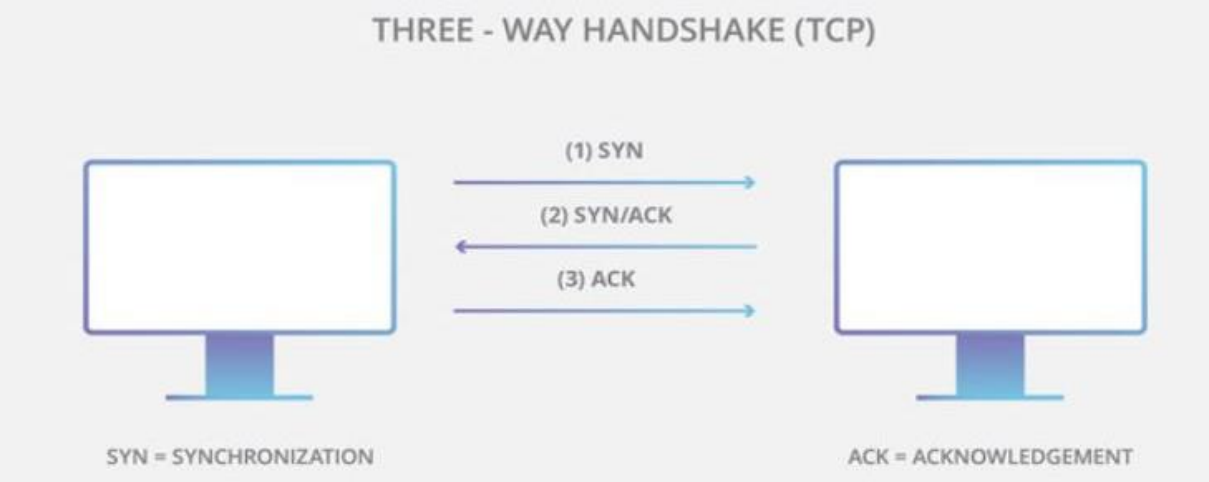
Ví dụ: Độ dài mật khẩu 8 ký tự trở lên, có kết hợp chữ hoa và chữ thường, có dùng ký tự đặc biệt…

+ Đặt luật về số lần đăng nhập được phép nhập.

Ví dụ: Đăng nhập 5 lần sai thì ngừng đăng nhập thời gian sau 3-5 giờ và thậm chí lâu hơn để tiếp tục nhập mật khẩu tiếp tục.

2.1.5. Tấn công từ chối dịch vụ (Denial of Service)

Hệ quản trị cơ sở dữ liệu PostgreSQL sẽ mở các cổng dịch vụ từ 2424 – 2440 trên giao thức TCP (Transmission control protocol) là giao thức điều khiển truyền nhận ở tầng vận tải, kẻ tấn công có thể sử dụng kỹ thuật tấn công Syn flood vào máy chủ PostgreSQL gây ra tình trạng quá tải của máy chủ PostgreSQL , tình trạng này sẽ không phục hồi các dịch vụ đã cung cấp. Để hiểu về tấn công SYN flood ta cần hiểu:



Hình 2.1. Mô hình hoạt động của TCP

Tấn công SYN Flood DDoS khai thác từ lỗ hỏng bên trong TCP/IP, bắt tay nhau trong việc tấn công phá vỡ một dịch vụ web.

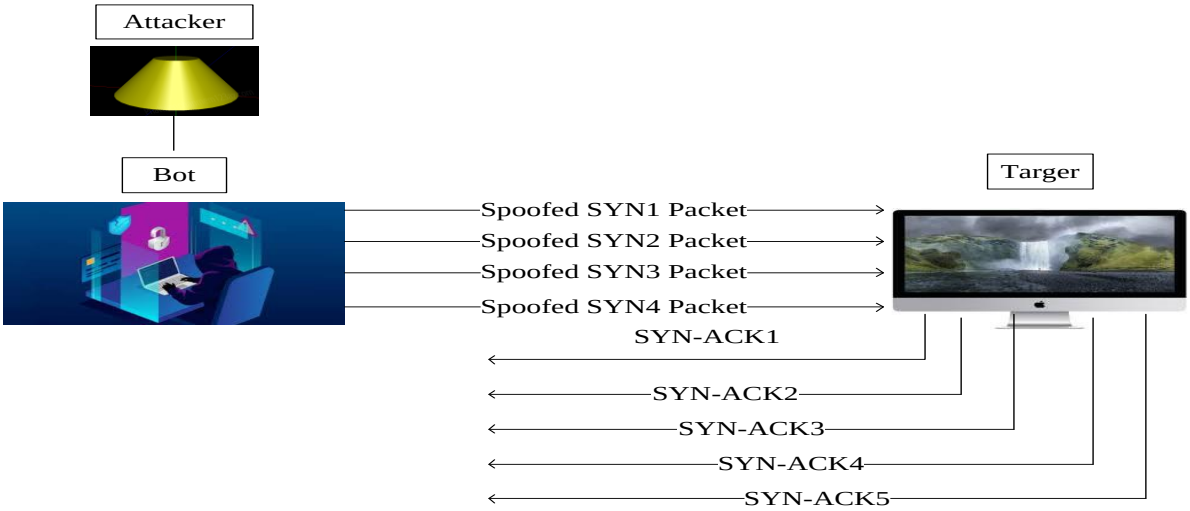
SYN flood (half – open attack) là một kiểu tấn công từ chối dịch vụ (DDoS). Tấn công này với mục đích làm cho Server không có lưu lượng để truy cập hợp pháp. Bằng cách tiêu thu tất cả tài nguyên Server đang có sẵn. Kẻ tấn công có thể áp đảo tất cả các cổng Server. Làm cho thiết bị client đáp ứng lưu lượng hợp pháp một cách chậm chạp hình 2. 1

Một cuộc tấn công DDoS SYN flood tận dụng quy trình bắt tay ba chiều TCP. Trong điều kiện bình thường, kết nối TCP được thể hiện quy trình ba bước (hình 2.2) như sau :

+ B1: Đầu tiên máy tấn công gửi 1 packet tin SYN đến SYN đến server để yêu cầu kết nối.

+ B2: Sau khi tiếp nhập packet SYN, server phản hồi lại máy khách bằng một packet SYN/ACK, để xác nhận thông tin từ client.

+ B3: Cuối cùng, client nhận được packet tin SYN/ACK thì sẽ trả lời sever bằng packet tin ACK báo với server biết rằng nó đã nhận được packet tin SYN/ACK, kết nối đã được thiết lập và sẵn sàng trao đổi dữ liệu.



Hình 2.2. Tấn công Syn Food

Để tạo từ chối dịch vụ (DOS), kẻ tấn công sẽ khai thác sau khi nhận được packet SYN ban đầu từ client. Server sẽ phản hồi lại một hoặc nhiều packet SYN/ACK và chờ đến bước cuối cùng trong quá trình bắt tay (handshake). Cách thức thực hiện như sau: (hình 2.2)

B1: Kẻ tấn công sẽ gửi một khối lượng lớn các packet tin SYN đến server. Được xem là mục tiêu và thông thường là địa chỉ IP giả mạo.

B2: Sau đó server sẽ phản hồi lại từng yêu cầu kết nối. Để lại 1 cổng mở sẵn sàng tiếp nhận và phản hồi.

B3: Trong khi server chờ packet ACK ở bước cuối cùng từ client, packet mà không bao giờ đến. Kẻ tấn công tiếp tục gửi thêm các packet SYN. Sự xuất hiện các packet SYN mới khiến máy chủ tạm thời duy trì cổng mở mới trong một thời gian nhất định. Một khi các cổng có sẵn được sử dụng thì server không thể hoạt động như lúc bình thường.

2.2. Cơ chế đảm bảo ATTT trong HQTCSDL

2.2.1. Yêu cầu để đảm bảo an toàn cho cơ sở dữ liệu

Việc tổ chức, lưu trữ thông tin nói chung có thể được triển khai dưới nhiều hình thức và công nghệ khác nhau, sử dụng các phần mềm và phương tiện lưu trữ khác nhau, trong đó việc sử dụng cơ sở dữ liệu để lưu trữ thông tin là rất phổ biến.

Do vậy, việc đảm bảo an toàn cho cơ sở dữ liệu là vấn đề cấp thiết và phải được xác định ngay từ ban đầu.

Bảo mật hay an toàn cho cơ sở dữ liệu (Database Security) là một tập hợp các thủ tục, chuẩn, chính sách và công cụ được sử dụng để bảo vệ dữ liệu nhằm tránh bị trộm cắp, lạm dụng, đột nhập, tấn công, hoặc các hành động không mong muốn khác. Nói cách khác, mục đích của việc bảo mật hay an toàn cho cơ sở dữ liệu là để đảm bảo 3 thuộc tính cơ bản của an toàn cơ sở dữ liệu, bao gồm tính bí mật (Confidentiality), tính toàn vẹn (Integrity) và tính sẵn sàng (Availability). Trong đó:

- Tính bí mật: Đảm bảo rằng chỉ có người dùng có thẩm quyền (Authorised Users) mới có thể truy nhập và thực hiện các thao tác trên cơ sở dữ liệu. Tính bí mật có thể được đảm bảo thông qua các cơ chế kiểm soát truy nhập bao gồm xác thực (Authentication) và trao quyền (Authorisation) vào các đối tượng trong cơ sở dữ liệu. Ngoài ra tính bí mật còn được đảm bảo bởi nhiều biện pháp bảo mật bổ sung, như bảo vệ vật lý, tường lửa, mã hóa,…

- Tính toàn vẹn: Đảm bảo rằng dữ liệu có thể được sửa đổi bởi những người dùng có thẩm quyền. Tính toàn vẹn liên quan đến tính hợp lệ (Validity), tính nhất quán (Consistency) và chính xác (Accuracy) của dữ liệu. Dữ liệu được xem là toàn vẹn nếu nó không bị thay đổi, hợp lệ và chính xác. Tính toàn vẹn có thể được đảm bảo bởi các ràng buộc dữ liệu (Constraints), các phép kiểm tra và các cơ chế xử lý dữ liệu.

- Tính sẵn sàng: Đảm bảo rằng cơ sở dữ liệu có thể truy nhập bởi người dùng hợp pháp bất cứ khi nào họ yêu cầu. Tính sẵn sàng có liên quan mật thiết đến hạ tầng mạng và năng lực của các máy chủ cơ sở dữ liệu và có thể được đảm bảo bằng chuỗi máy chủ cơ sở dữ liệu và hệ thống cân bằng tải. Tính sẵn sàng có thể được đo bằng các yếu tố:

+ Thời gian cung cấp dịch vụ (Uptime);

+ Thời gian ngừng cung cấp dịch vụ (Downtime);

+ Tỷ lệ phục vụ: A = (Uptime)/(Uptime + Downtime);

+ Thời gian trung bình giữa các sự cố;

+ Thời gian trung bình ngừng để sửa chữa;

+ Thời gian khôi phục sau sự cố.

Với các yêu cầu về an toàn cơ sở dữ liệu nêu trên, chúng ta dễ dàng nhận thấy các mối đe dọa cho cơ sở dữ liệu như sau:

- Sự thất thoát thông tin không hợp lý, điều này đe dọa tính bí mật của dữ liệu.

- Thay đổi dữ liệu không hợp lý, điều này đe dọa tính toàn vẹn của dữ liệu.

- Từ chối dịch vụ, điều này đe dọa tính sẵn sàng của dữ liệu.

2.2.2. Các cơ chế để bảo vệ tính an toàn của cơ sở dữ liệu

a) Phân quyền user

Tạo user

User trong Postgres được gọi là ROLE. Có 2 loại role là login role và non - login role. Khi tạo USER trong Postgres thì tức là đang tạo một login role đó.

Để tạo một user (hay login role) mới thì sử dụng lệnh CREATE ROLE. Ví dụ để tạo user myuser, ta sử dụng như sau:

```
CREATE ROLE myuser WITH LOGIN;
```

Mặc định thì một role sẽ là no - login role, do đó cần thêm LOGIN để tạo một login role. Cũng có thể dùng CREATE USER thay vì CREATE ROLE như sau:

```
CREATE USER myuser;
```

Để cài đặt password, thêm tùy chọn PASSWORD như sau:

```
CREATE USER myuser WITH PASSWORD 'mypassword';
```

Hoặc

```
CREATE ROLE myuser WITH LOGIN PASSWORD 'mypassword';
```

PostgresSQL không có cài đặt xác thực đầu cuối như MySQL/MariaDB. Tuy nhiên, người dùng vẫn có thể cài một số config tương tự như chỉ cho phép login hoặc yêu cầu password/certificate tùy theo host ở file pg_hba.conf.

Tạo DB

Để tạo DB, ta dùng query CREATE DATABASE. Các object trong PostgresSQL đều có owner, thông thường người nào tạo ra nó sẽ là owner. Với DB thì chỉ có user với quyền CREATE DATABASE mới tạo được nên để tạo DB cho một user khác chúng ta sẽ thêm option OWNER.

Ví dụ để tạo DB mydatabase với owner là myusser thì query của chúng ta sẽ như sau:

```
CREATE DATABASE mydatabase OWNER myuser;
```

Tương tự như CREATE USER, CREATE DATABASE cũng không có option IF NOT EXISTS. Tuy nhiên, khác với query trước, CREATE DATABASE còn không được phép chạy trong một transaction khác nữa, vậy nên chúng ta không thể dùng cách trên để tạo DB chỉ khi nó chưa tồn tại được.

Nếu chạy từ shell script thì có thể dùng psql để tạo DB như sau:

```
echo "SELECT 'CREATE DATABASE mydb' WHERE NOT EXISTS (SELECT FROM pg_database WHERE datname = 'mydb')\gexec" | psql
```

Nếu phải chạy trong SQL script thì sẽ hơi phức tạp một chút. Chúng ta sẽ cần đến extension dblink. Người dùng sẽ phải tự connect đến server DB hiện tại để chạy query thay vì chạy trực tiếp.

```
DO $$
BEGIN
PERFORM dblink_exec('password=yourpassword', 'CREATE DATABASE mydb OWNER myuser');
EXCEPTION WHEN DUPLICATE_DATABASE THEN
RAISE NOTICE 'Database mydb already exists, skipping';
END
$$;
```

Ngoài ra trong mỗi database còn có các schema. Mặc định thì mỗi DB sẽ được tạo với một schema . Ai có quyền truy cập DB cũng sẽ truy cập được schema này. Có thể tạo schema với query CREATE SCHEMA.

```
CREATE SCHEMA myschema;
```

Phân quyền cho user

Hệ thống phân quyền của Postgres khá phức tạp. Ngoài owner và superuser ra thì các user khác sẽ không có quyền gì với các đối tượng mới được tạo ra. Để cấp quyền cho user thì chúng ta dùng query GRANT.

Có các quyền như sau

+ CONNECT: connect tới DB.

+ USAGE: Mức độ sử dụng.

+ CREATE/SELECT/INSERT/UPDATE/DELETE/TRUNCATE:

được chạy các query tương ứng.

+ EXECUTE: gọi function.

+ REFERENCES/TRIGGER/TEMP/TEMPORARY: quyền tạo foreign key, trigger, bảng tạm thời.

+ ALL PRIVILEGES: tất cả mọi quyền có thể GRANT.

Để thêm quyền xem (read only) cho một user chúng ta cần những query như thế này.

+ GRANT CONNECT ON DATABASE mydb TO my_readonly;

+ GRANT USAGE ON SCHEMA public TO my_readonly;

+ GRANT SELECT ON ALL TABLES IN SCHEMA public TO my_readonly;

+ GRANT SELECT ON ALL SEQUENCES IN SCHEMA public TO my_readonly;

+ GRANT USAGE ON ALL SEQUENCES IN SCHEMA public TO my_readonly;

+ GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO my_readonly;

Ngoài ra mỗi role trong postgres vừa có thể là user mà cũng có thể là một group. Các user trong cùng group sẽ được "kế thừa" những quyền của group đó. Ví dụ sau khi đã có user my_readonly ở trên, muốn tạo một user my_readwrite với quyền ghi, thay vì phải lặp lại các quyền như trên, người dùng chỉ cần GRANT cho nó các quyền của my_readonly.

GRANT my_readonly TO my_readwrite;

Sau đó có thể thêm các quyền khác.

GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO my_readwrite;

Phân quyền theo group

Một role thì ngoài user ra còn có thể là một group. Role login dành cho user và no-login role được dùng để tạo group. Người dùng có thể tạo group với các quyền như read only, read write, hay thậm chí là cả owner để quản lý cho các user khác. Ví dụ tạo database với nhiều owner bằng cách cấp quyền quyền my_owner cho các user khác như sau.

```
CREATE ROLE my_owner;
```

```
CREATE DATABASE mydb OWNER my_owner;
GRANT my_owner TO john_smith;
GRANT my_owner TO whoever;
```

Các DEFAULT PRIVILEGES cấp quyền cho cả group nên các user trong group cũng sẽ được kế thừa quyền của group. Ví dụ với default privileges như sau

```
ALTER DEFAULT PRIVILEGES FOR ROLE myuser IN SCHEMA public GRANT SELECT ON TABLES TO my_readonly;
```

Các user của group my_readonly sẽ có quyền select trên tất cả table mới được tạo bởi myuser. Tuy nhiên, trong trường hợp ngược lại, nếu table được tạo bởi một user khác trong group myuser thì default privileges sẽ không có tác dụng. Q

b) Kiểm toán (Auditting)

Kiểm toán là một cách tốt để giữ cho dữ liệu an toàn nhất có thể và để biết điều gì đang xảy ra trong cơ sở dữ liệu. Nó cũng được yêu cầu đối với nhiều quy định hoặc tiêu chuẩn bảo mật. Do đó, hệ quản trị cơ sở dữ liệu PostgreSQL cũng không phải ngoại lệ.

PostgreSQL đã tạo được danh tiếng mạnh mẽ về kiến trúc đã được chứng minh, độ tin cậy, tính toàn vẹn của dữ liệu, bộ tính năng mạnh mẽ, khả năng mở rộng và sự cống hiến của cộng đồng nguồn mở đằng sau phần mềm để liên tục cung cấp các giải pháp hiệu quả và sáng tạo.

Để sử dụng tính năng kiểm toán trên PostgreSQL, người dùng cần cài đặt pgAudit. pgAudit cung cấp ghi phiên nhật ký kiểm toán và đối tượng thông qua tiêu chuẩn ghi nhật ký PostgreSQL.

Cài đặt cơ chế kiểm toán pgAudit

- Kiểm tra tính khả dụng của pgAudit với thông tin:

```
pgaudit14_12.x86_64 : PostgreSQL Audit Extension
```

- Cài đặt pgAudit bằng lệnh:

```
$ yum install pgaudit14_12
```

- cần thêm nó vào tệp cấu hình PostgreSQL.conf, được đặt theo mặc định trong /var/lib/pgsql/12/data/PostgreSQL.conf và khởi động lại dịch vụ PostgreSQL để áp dụng thay đổi.

```
shared_preload_libraries = 'pgaudit, pg_stat_statements'
```

- Sau khi khởi động lại dịch vụ, người dùng cần thực hiện tạo tiện ích mở rộng:

```
postgres=# CREATE EXTENSION pgaudit;

CREATE EXTENSION

And now, you can run the following query to check the new extension created:

postgres=# SELECT * FROM pg_available_extensions WHERE name LIKE
'%audit%';
  name   | default_version | installed_version |            comment

---------+-----------------+-------------------+-------------------------------

 pgaudit | 1.4.1           | 1.4.1             | provides auditing functionality

(1 row)
```

Thiết lập pgAudit

Người dùng có thể kiểm tra những thiết lập đang sử dụng như sau

```
postgres=# SELECT name,setting FROM pg_settings WHERE name LIKE
'pgaudit%';
           name            | setting
---------------------------+---------
 pgaudit.log               | none

 pgaudit.log_catalog       | on

 pgaudit.log_client        | off

 pgaudit.log_level         | log

 pgaudit.log_parameter     | off
```

```
 pgaudit.log_relation       | off
 pgaudit.log_statement_once | off
 pgaudit.role               |
(8 rows)
```

Các tham số thiết lập có ý nghĩa như sau:

+ pgaudit.log: chỉ định các lớp câu lệnh nào sẽ được ghi lại bằng ghi nhật ký kiểm toán. Mặc định là không có.

+ pgaudit.log_catalog: Chỉ định ghi nhật ký phiên sẽ được bật trong trường hợp tất cả các quan hệ trong một câu lệnh đều nằm trong pg_catalog. Việc tắt cài đặt này sẽ giảm nhiễu trong nhật ký từ các công cụ như psql và PgAdmin truy vấn danh mục nhiều. Mặc định được bật.

+ pgaudit.log_client: Chỉ định việc thông báo nhật ký có được hiển thị cho một quy trình khách hàng chẳng hạn như psql hay không. Cài đặt này thường bị tắt nhưng có thể hữu ích cho việc gỡ lỗi hoặc các mục đích khác. Mặc định là tắt.

+ pgaudit.log_level: Chỉ định mức nhật ký sẽ được sử dụng cho các mục nhập nhật ký. Cài đặt này được sử dụng để kiểm tra hồi quy và cũng có thể hữu ích cho người dùng cuối để kiểm tra hoặc các mục đích khác. Mặc định là nhật ký.

+ pgaudit.log_parameter: Chỉ định rằng ghi nhật ký kiểm toán phải bao gồm các tham số đã được thực thi với câu lệnh. Khi các tham số hiện diện, chúng sẽ được đưa vào định dạng CSV sau câu lệnh. Mặc định là tắt.

+ pgaudit.log_relation: Chỉ định việc ghi nhật ký kiểm tra phiên có nên tạo một mục nhật ký riêng biệt cho mỗi quan hệ (TABLE, VIEW, v.v.) được tham chiếu trong một câu lệnh SELECT hoặc DML hay không. Đây là một phím tắt hữu ích để ghi nhật ký toàn diện mà không cần sử dụng ghi nhật ký kiểm toán đối tượng. Mặc định là tắt.

+ pgaudit.log_statement_once: Chỉ định việc ghi nhật ký sẽ bao gồm câu lệnh và các tham số với mục nhập nhật ký đầu tiên cho một tổ hợp câu lệnh / câu lệnh con hoặc với mọi mục nhập. Việc tắt cài đặt này sẽ dẫn đến việc ghi ít chi tiết hơn nhưng có thể gây khó khăn hơn trong việc xác định câu lệnh đã tạo mục nhập nhật ký, mặc dù cặp câu lệnh / câu lệnh con cùng với id quy trình đủ để xác định văn bản câu lệnh được ghi bằng mục nhập trước đó. Mặc định là tắt.

+ pgaudit.role: Chỉ định vai trò chính để sử dụng cho việc ghi nhật ký kiểm toán đối tượng.

c) Thiết lập chính sách

```
CREATE POLICY name ON table_name
   [ AS { PERMISSIVE | RESTRICTIVE } ]
   [ FOR { ALL | SELECT | INSERT | UPDATE | DELETE } ]
   [ TO { role_name | PUBLIC | CURRENT_ROLE | CURRENT_USER |
SESSION_USER } [, ...] ]
   [ USING ( using_expression ) ]
   [ WITH CHECK ( check_expression ) ]
```

Để thiết lập chính sách, bảo mật mức hàng phải được thiết lập trên bảng, bằng cách sử dụng các lệnh ALTER TABLE...ENABLE ROW LEVEL SECURITY để các chính sách được áp dụng.

Chính sách cấp quyền select, insert, update, or delete các hàng khớp với biểu thức chính sách có liên quan. Các hàng hiện có trong bảng được kiểm tra so với biểu thức được chỉ định trong USING, trong khi các hàng mới sẽ được tạo thông qua INSERT hoặc UPDATE được kiểm tra dựa trên biểu thức được chỉ định trong WITH CHECK. Khi một biểu thức USING trả về true cho một hàng nhất định thì hàng đó sẽ hiển thị cho người dùng, trong khi nếu trả về false hoặc null thì hàng đó sẽ không hiển thị. Khi biểu thức WITH CHECK trả về true cho một hàng thì hàng đó sẽ được chèn hoặc cập nhật, trong khi nếu trả về false hoặc null thì sẽ xảy ra lỗi.

Đối với các câu lệnh INSERT và UPDATE, biểu thức WITH CHECK được thực thi sau khi lệnh kích hoạt BEFORE được kích hoạt và trước khi thực hiện bất kỳ sửa đổi dữ liệu thực tế nào. Do đó, trình kích hoạt BEFORE ROW có thể sửa đổi dữ liệu được chèn vào, ảnh hưởng đến kết quả kiểm tra chính sách bảo mật. Biểu thức WITH CHECK được thực thi trước bất kỳ ràng buộc nào khác.

Tên chính sách tương ứng với mỗi bảng. Do đó, một tên chính sách có thể được sử dụng cho nhiều bảng khác nhau và có định nghĩa cho từng bảng phù hợp với bảng đó.

Các chính sách có thể được áp dụng cho các lệnh cụ thể hoặc cho các roles cụ thể. Mặc định cho các chính sách mới được tạo là chúng áp dụng cho tất cả các lệnh và các roles , trừ khi được chỉ định khác. Nhiều chính sách có thể áp dụng cho một lệnh.

Đối với các chính sách có thể có cả biểu thức USING và WITH CHECK (ALL và UPDATE), nếu không có biểu thức WITH CHECK nào được xác định,

thì biểu thức USING sẽ được sử dụng cả hai để xác định hàng nào được hiển thị (trường hợp USING thông thường) và hàng mới nào sẽ được được phép thêm vào (trường hợp WITH CHECK).

Nếu bảo mật cấp hàng được bật cho một bảng nhưng không tồn tại chính sách hiện hành, thì chính sách "default deny - từ chối mặc định" sẽ được giả định để không có hàng nào được hiển thị hoặc có thể cập nhật.

Các tham số chính

+ Name: Tên chính sách cần tạo, cần phải phân biệt với tên của các chính sách khác trong bảng.

+ table_name: Tên của bảng mà các chính sách áp dụng.

+ PERMISSIVE: Chỉ định rằng chính sách sẽ được tạo như một chính sách cho phép. Tất cả các chính sách được phép áp dụng cho một truy vấn nhất định sẽ được kết hợp với nhau bằng cách sử dụng toán tử Boolean “OR”. Bằng cách tạo các chính sách cho phép, quản trị viên có thể thêm vào tập hợp các bản ghi có thể được truy cập. Các chính sách được cho phép theo mặc định.

+ RESTRICTIVE: Chỉ định rằng chính sách sẽ được tạo như một chính sách hạn chế. Tất cả các chính sách hạn chế áp dụng cho một truy vấn nhất định sẽ được kết hợp với nhau bằng cách sử dụng toán tử Boolean “AND”. Bằng cách tạo các chính sách hạn chế, quản trị viên có thể giảm tập hợp các bản ghi có thể được truy cập vì tất cả các chính sách hạn chế phải được thông qua cho mỗi bản ghi.

Lưu ý rằng cần có ít nhất một chính sách cho phép để cấp quyền truy cập vào các bản ghi trước khi các chính sách hạn chế có thể được sử dụng hữu ích để giảm quyền truy cập đó. Nếu chỉ tồn tại các chính sách hạn chế, thì sẽ không có bản ghi nào có thể truy cập được. Khi có sự kết hợp giữa các chính sách cho phép và hạn chế, chỉ có thể truy cập hồ sơ nếu ít nhất một trong các chính sách cho phép được thông qua, ngoài tất cả các chính sách hạn chế.

+ command: Lệnh áp dụng chính sách. Các tùy chọn hợp lệ là ALL, SELECT, INSERT, UPDATE, and DELETE. ALL là mặc định.

+ role_name: Các roles mà chính sách sẽ được áp dụng, mặc đinh là PUBLIC, sẽ áp dụng chính sách cho tất cả các roles.

+ using_expression: Bất kỳ biểu thức điều kiện SQL nào (trả về boolean). Biểu thức điều kiện không được chứa bất kỳ hàm tổng hợp hoặc hàm cửa sổ nào. Biểu thức này sẽ được thêm vào các truy vấn tham chiếu đến bảng nếu bảo mật mức hàng được bật. Các hàng mà biểu thức trả về true sẽ được hiển thị. Bất kỳ

hàng nào mà biểu thức trả về false hoặc null sẽ không hiển thị với người dùng (trong SELECT) và sẽ không có sẵn để sửa đổi (trong UPDATE hoặc DELETE).

+ check_expression: Bất kỳ biểu thức điều kiện SQL nào (trả về boolean). Biểu thức điều kiện không được chứa bất kỳ hàm tổng hợp hoặc hàm cửa sổ nào. Biểu thức này sẽ được sử dụng trong các truy vấn INSERT và UPDATE đối với bảng nếu bảo mật cấp hàng được bật. Chỉ những hàng mà biểu thức đánh giá là true mới được phép. Một lỗi sẽ được đưa ra nếu biểu thức đánh giá là false hoặc null cho bất kỳ bản ghi nào được chèn vào hoặc bất kỳ bản ghi nào là kết quả của quá trình cập nhật. Lưu ý rằng check_expression được đánh giá dựa trên nội dung mới được đề xuất của hàng, không phải nội dung ban đầu.

Chính sách cho mỗi lệnh

+ ALL: Sử dụng ALL cho một chính sách có nghĩa là nó sẽ áp dụng cho tất cả các lệnh, bất kể loại lệnh nào. Nếu tồn tại chính sách ALL và các chính sách cụ thể hơn tồn tại, thì cả chính sách ALL và chính sách (hoặc các chính sách) cụ thể hơn sẽ được áp dụng. Ngoài ra, ALL các chính sách sẽ được áp dụng cho cả phía lựa chọn của truy vấn và phía sửa đổi, sử dụng biểu thức USING cho cả hai trường hợp nếu chỉ một biểu thức USING đã được xác định.

+ SELLECT: Sử dụng SELECT cho một chính sách có nghĩa là nó sẽ áp dụng cho các truy vấn SELECT và bất cứ khi nào các quyền SELECT được yêu cầu đối với quan hệ mà chính sách được xác định. Kết quả là chỉ những bản ghi từ quan hệ vượt qua chính sách SELECT sẽ được trả về trong một truy vấn SELECT và các truy vấn yêu cầu quyền SELECT, chẳng hạn như UPDATE, cũng sẽ chỉ thấy những bản ghi được chính sách SELECT cho phép. Chính sách SELECT không được có biểu thức WITH CHECK, vì nó chỉ áp dụng trong trường hợp các bản ghi đang được truy xuất từ quan hệ.

+ INSERT: Sử dụng INSERT cho một chính sách có nghĩa là nó sẽ áp dụng cho các lệnh INSERT. Các hàng được chèn vào không vượt qua chính sách này sẽ dẫn đến lỗi vi phạm chính sách và toàn bộ lệnh INSERT sẽ bị hủy bỏ. Chính sách INSERT không được có biểu thức USING, vì nó chỉ áp dụng trong trường hợp các bản ghi đang được thêm vào mối quan hệ.

+ UPDATE: Sử dụng UPDATE cho một chính sách có nghĩa là nó sẽ áp dụng cho các lệnh UPDATE, SELECT FOR UPDATE và SELECT FOR SHARE, cũng như các mệnh đề ONDATE bổ trợ của lệnh INSERT. Vì UPDATE liên quan đến việc kéo một bản ghi hiện có và thay thế nó bằng một bản ghi mới được sửa

đổi, các chính sách UPDATE chấp nhận cả biểu thức USING và biểu thức WITH CHECK. Biểu thức USING xác định bản ghi nào mà lệnh UPDATE sẽ thấy để hoạt động chống lại, trong khi biểu thức WITH CHECK xác định các hàng đã sửa đổi nào được phép lưu trữ trở lại quan hệ.

+ DELETE: Sử dụng DELETE cho một chính sách có nghĩa là nó sẽ áp dụng cho các lệnh DELETE. Chỉ những hàng vượt qua chính sách này mới được nhìn thấy bằng lệnh DELETE. Có thể có các hàng hiển thị thông qua một SELECT không có sẵn để xóa, nếu chúng không chuyển biểu thức USING cho chính sách DELETE.

Chính sách DELETE không được có biểu thức WITH CHECK, vì nó chỉ áp dụng trong trường hợp các bản ghi đang bị xóa khỏi quan hệ, do đó không có hàng mới để kiểm tra.

Bảng 2. Chính sách áp dụng cho các lệnh

Lệnh	SELECT/ALL	INSERT/ALL	UPDATE/ALL		DELETE/ALL
	USING	WITH CHECK	USING	WITH CHECK	USING
SELECT	Hàng có sẵn				
SELECT FOR UPDATE/SHARE	Hàng có sẵn		Hàng có sẵn		
INSERT		Hàng mới			
INSERT ... RETURNING	Hàng mới	Hàng mới			
UPDATE	Hàng có sẵn và mới		Hàng có sẵn	Hàng mới	
DELETE	Hàng có sẵn				Hàng có sẵn
ON CONFLICT DO UPDATE	Hàng có sẵn và mới		Hàng có sẵn	Hàng mới	

2.3. Cơ chế mã hoá dữ liệu lưu trữ

Để bảo vệ các tệp dữ liệu này, các hệ quản trị cơ sở dữ liệu đã cung cấp cơ chế mã hóa dữ liệu trong suốt. Đây là cơ chế an toàn bảo vệ các dữ liệu nhạy cảm kể cả trong trường hợp phương tiện lưu trữ hoặc tệp dữ liệu bị đánh cắp.

Mã hóa dữ liệu trong suốt (Transparent data encrytion – TDE) là một trong những cơ chế an toàn, cho phép mã hóa dữ liệu nhạy cảm được lưu trữ trong bảng và không gian bảng. Dữ liệu được mã hóa và giải mã trong suốt đối với người

dùng và các ứng dụng có quyền truy cập vào dữ liệu. Để ngăn chặn việc giải mã trái phép, TDE lưu trữ các khóa mã hóa trong mô-đun an toàn bên ngoài cơ sở dữ liệu (CSDL).

TDE thực hiện mã hóa và giải mã thời gian thực các dữ liệu. Việc mã hóa sử dụng khóa mã hóa dữ liệu (Data Encryption Key - DEK) được lưu trữ trong bản ghi khởi động CSDL để sẵn sàng trong quá trình phục hồi. Bên cạnh đó, TDE có khả năng bảo vệ dữ liệu lưu trữ, nghĩa là các tệp dữ liệu và nhật ký. Nó cung cấp khả năng tuân thủ nhiều luật, quy định và hướng dẫn được thiết lập trong các ngành công nghiệp khác nhau. Điều này cho phép các nhà phát triển phần mềm mã hóa dữ liệu bằng cách sử dụng thuật toán mã hóa AES và 3DES mà không cần thay đổi các ứng dụng hiện có.

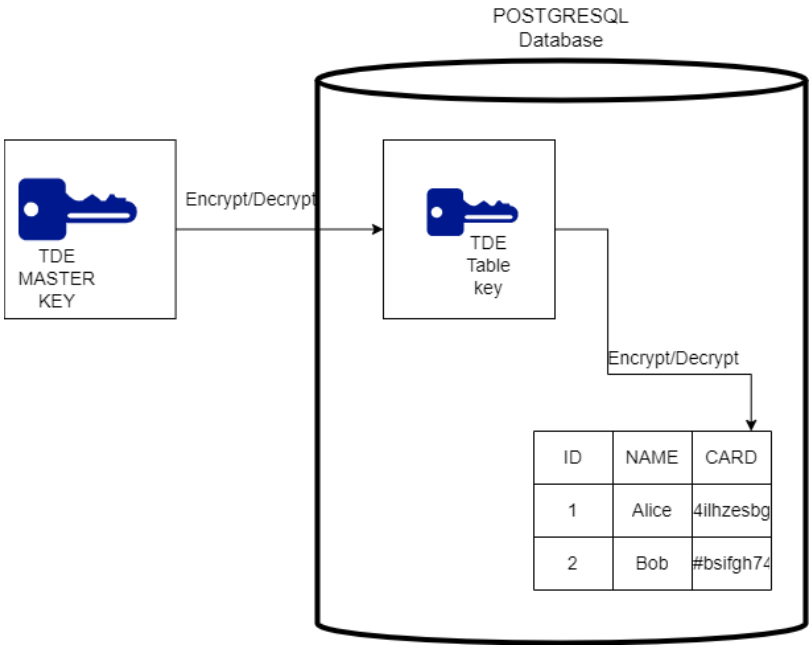
Ưu điểm của mã hóa TDE:

- Đảm bảo an toàn các dữ liệu nhạy cảm trong trường hợp phương tiện lưu trữ hoặc tệp dữ liệu bị đánh cắp.
- Tuân thủ các quy định liên quan đến an toàn.
- Không cần tạo trình kích hoạt hoặc chế độ xem để giải mã dữ liệu cho người dùng hoặc ứng dụng được ủy quyền. Dữ liệu từ các bảng được giải mã trong suốt cho người dùng và ứng dụng CSDL.
- Các ứng dụng không cần phải sửa đổi để xử lý dữ liệu được mã hóa. Mã hóa và giải mã dữ liệu được quản lý bởi cơ sở dữ liệu.

Mã hóa TDE được chia làm 2 loại là mã hóa TDE mức cột và mã hóa TDE không gian bảng. Trong khi mã hóa cột TDE cho phép mã hóa dữ liệu nhạy cảm được lưu trữ trong các cột của bảng được chọn, thì mã hóa không gian bảng TDE cho phép mã hóa tất cả dữ liệu được lưu trữ trong một vùng bảng.

Cả mã hóa cột và mã hóa không gian bảng TDE đều sử dụng kiến trúc dựa trên khóa hai tầng. Ngay cả khi dữ liệu mã hóa được truy xuất, dữ liệu cũng không được hiển thị cho đến khi quá trình giải mã được ủy quyền xảy ra, quá trình này là tự động cho người dùng được ủy quyền truy cập vào bảng.

Mã hóa cột TDE



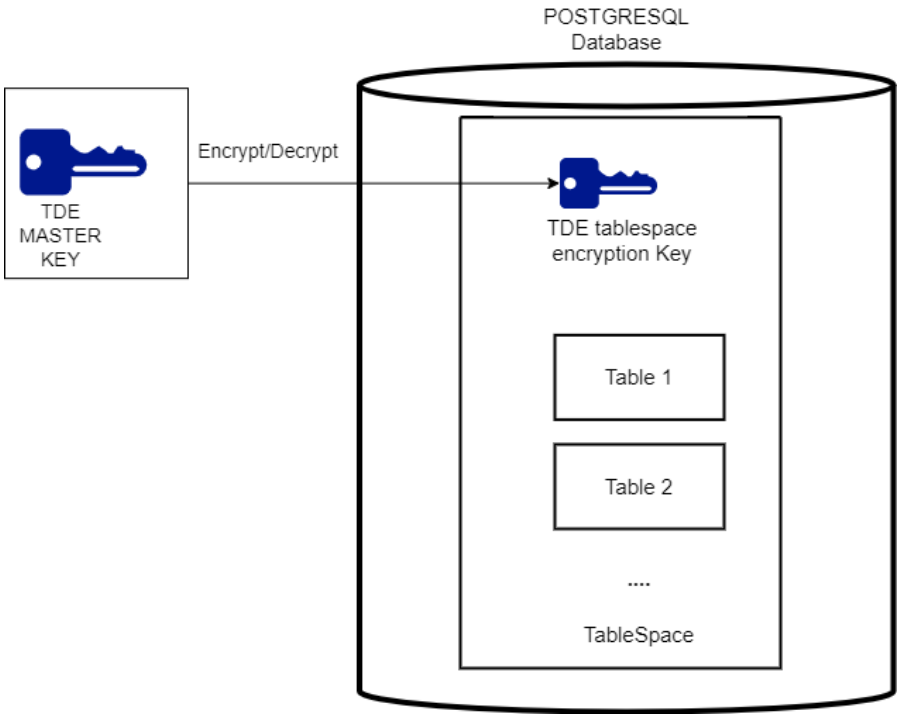
Hình 2.3. Mã hóa cột TDE

Mã hóa cột TDE được sử dụng để bảo vệ dữ liệu nhạy cảm, chẳng hạn như thẻ tín dụng và số an sinh xã hội, được lưu trữ trong các cột của bảng. Loại mã hóa này sử dụng kiến trúc hai lớp, dựa trên khóa để mã hóa và giải mã các cột bảng nhạy cảm. Trong đó, khóa mã hóa chính (Master encryption key) được sử dụng để mã hóa khóa bảng (Table key). Khóa này được lưu trữ an toàn phụ thuộc vào cơ chế của hệ quản trị cơ sở dữ liệu đang sử dụng.

Việc lưu trữ theo cách này sẽ ngăn việc sử dụng trái phép các khóa mật và tách biệt các chức năng chương trình thông thường khỏi các hoạt động mã hóa, giúp phân chia nhiệm vụ giữa người quản trị CSDL và quản trị an toàn, dẫn đến tăng cường khả năng bảo mật.

Mã hóa không gian bảng TDE

Mã hóa không gian bảng TDE cho phép mã hóa toàn bộ vùng bảng. Tất cả các đối tượng được tạo trong không gian bảng sẽ được mã hóa tự động. Mã hóa không gian bảng TDE rất hữu ích nếu trong trường hợp muốn bảo vệ dữ liệu nhạy cảm trong các bảng. Trường hợp này, không cần thực hiện phân tích chi tiết từng cột trong bảng để xác định các cột cần mã hóa.



Hình 2.4. Mã hóa không gian bảng TDE

Tất cả dữ liệu trong một vùng bảng được mã hóa sẽ được lưu trữ ở định dạng mã hóa trên đĩa. Dữ liệu được giải mã trong suốt cho người dùng được ủy quyền có các đặc quyền cần thiết để được xem hoặc sửa đổi dữ liệu. Người dùng hoặc ứng dụng CSDL không cần biết liệu dữ liệu trong một bảng cụ thể có được mã hóa trên đĩa hay không. Trong trường hợp các tệp dữ liệu trên đĩa hoặc phương tiện sao lưu bị đánh cắp, dữ liệu sẽ không bị xâm phạm.

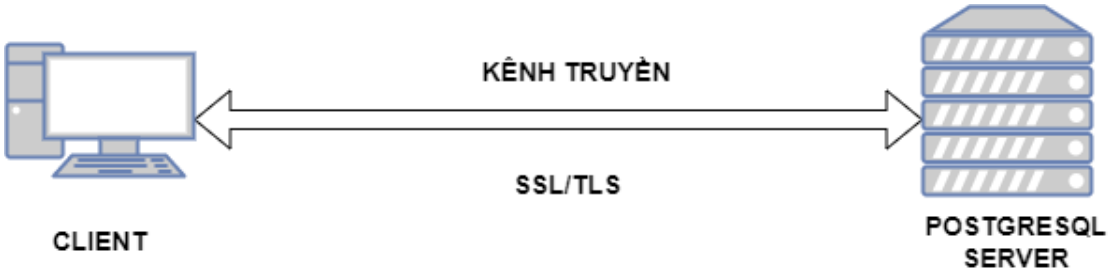
Mã hóa không gian bảng TDE cũng sử dụng kiến trúc hai tầng, dựa trên khóa để mã hóa trong suốt và giải mã không gian bảng. Khóa chính TDE cũng có chức năng và cách thức lưu trữ như mã hóa cột TDE.

Các thuật toán mã hóa được hỗ trợ

Các thuật toán được hỗ trợ hiện nay là các thuật toán mã hóa khóa đối xứng: AES, 3DES, BLOWFISH.

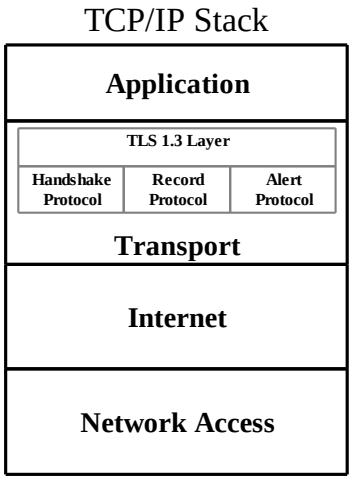
2.4. Cơ chế mã hoá dữ liệu kênh truyền

Cơ sở dữ liệu PostgreSQL hỗ trợ người dùng truy cập thông qua giao thức HTTP. Giao thức này không có cơ chế bảo mật đường truyền, vì vậy cần bảo mật cho giao thức này bằng cách kết hợp giao thức SSL/TLS để mã hóa đường truyền.



Hình 2.5. Bảo mật kênh truyền theo mô hình mạng Client – PostgreSQL Server

Giao thức TLS (Transport Layer Security) là một giao thức bảo mật dữ liệu truyền trên mạng Internet và hoạt động ở tầng giao vận trong kiến trúc mạng TCP/IP.



Hình 2.6. Kiến trúc, vị trí TLS 1.3 trong mô hình TCP/IP

Giao thức SSL/TLS (Secure Sockets Layer/Transport Layer Security) là giao thức được sử dụng để bảo mật kênh truyền cho rất nhiều dịch vụ mạng hiện nay như: Dịch vụ Web, Email, Database, VoIP... TLS 1.3 là phiên bản mới nhất của giao thức này với nhiều ưu điểm như tốc độ nhanh và độ an toàn cao hơn so với các phiên bản trước.

TLS 1.3 được công bố vào tháng 8/2018 sau thời gian rất dài kể từ khi bắt đầu được phát triển bởi IETF năm 2014 và sau 28 bản nháp khác nhau.

TLS 1.3 thực hiện xử lý dữ liệu tầng ứng dụng chuyển xuống trước khi giao thức tầng vận tải xử lý và có kiến trúc bao gồm 3 giao thức chính (hình 2.7): giao thức bắt tay (Handshake protocol), giao thức bản ghi (Record protocol), giao thức cảnh báo (Alert protocol).

TLS 1.3 bảo mật kênh truyên theo mô hình mạng Client – Server và có cách thức được mô tả như trong hình 2.6.

TLS 1.3 được thiết lập ban đầu với giao thức bắt tay gồm 3 bước chính như sau:

+ Bước 1: Phía máy trạm gửi gói Client Hello bao gồm các trường bắt buộc như một số nguyên ngẫu nhiên, phiên bản TLS… Trong các trường này cần chú ý hai trường chính:

Supported Cipher Suites: Là định danh các bộ thuật toán mã hóa có xác thực với dữ liệu liên kết mà bên máy trạm hỗ trợ.

Key_share: Đây chính là trường lưu trữ giá trị khóa công khai của máy trạm dùng để cho thỏa thuận một khóa chủ cho phiên liên lạc này giữa máy trạm và máy chủ.

+ Bước 2: Phía máy chủ gửi lại gói Server Hello bao gồm các trường bắt buộc. Trong các trường này cần chú ý ba trường chính:

Selected Cipher Suite: Là định danh bộ thuật toán mã hóa có xác thực bên máy chủ chọn sẽ dùng để mã hóa dữ liệu phiên liên lạc giữa máy trạm và máy chủ trong phiên liên lạc này.

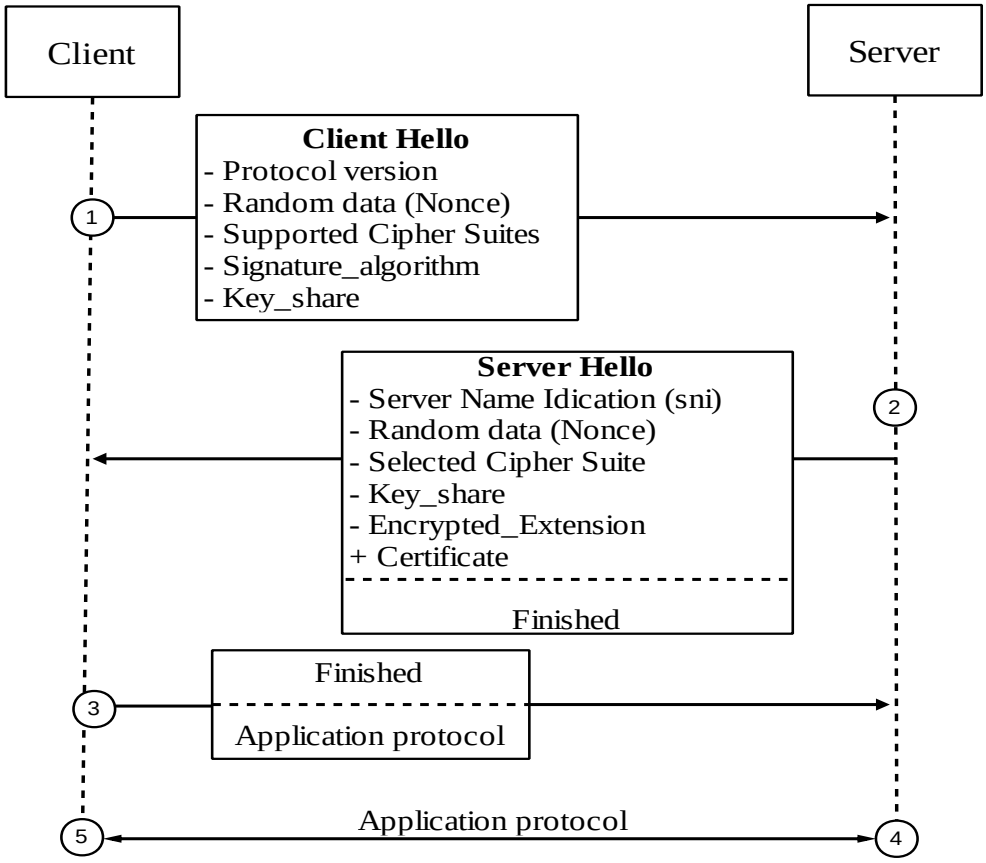
Key share: Đây chính là trường lưu trữ giá trị khóa công khai của máy chủ dùng để cho thỏa thuận một khóa chủ cho phiên liên lạc này giữa máy trạm và máy chủ.

Certificate: Đây là trường mở rộng được bảo mật có nội dung là chứng thư số của máy chủ cung cấp cho máy trạm để kiểm tra xác thực máy chủ đang liên lạc.

+ Bước 3: Kết thúc quá trình bắt tay với thông điệp Finished và bắt đầu thực hiện kết nối của giao thức trên tầng ứng dụng.

- Sau khi thực hiện xong giao thức bắt tay, TLS 1.3 sẽ thực hiện giao thức bản ghi để xử lý phân mảnh dữ liệu gửi từ tầng ứng dụng chuyển xuống trong các bước 4, 5 hình 2.8.

+ Các phân mảnh dữ liệu được mã hóa bởi bộ thuật toán mật mã đã thỏa thuận được giữa Client và Server trong quá trình bắt tay trước đó.



Hình 2.7. Hoạt động giao thức TLS 1.3

+ Khóa phiên mã hóa sử dụng được dẫn xuất từ khóa bí mật thỏa thuận được giữa hai bên liên lạc trong quá trình bắt tay.

- Trong phiên liên lạc TLS 1.3 được thiết lập và thực thi mọi vấn đề phát sinh về trạng thái, lỗi kết nối, sai thông số được thông báo dưới dạng các thông điệp bởi giao thức cảnh báo cho các bên liên lạc.

Chứng thư số

- Để tạo Chứng thư tự ký đơn giản cho máy chủ PostgreSQL , có giá trị trong 365 ngày, người dùng sử dụng lệnh OpenSSL sau, thay thế dbhost.yourdomain.com bằng tên miền của máy chủ:

```
openssl req -new -x509 -days 365 -nodes -text -out server.crt
\-keyout server.key -subj "/CN=dbhost.yourdomain.com"
```

Để tạo Chứng thư máy chủ mà khách hàng có thể xác thực danh tính, bước đầu cần tạo yêu cầu ký Chứng thư (CSR) và tệp khóa công khai / bí mật:

```
openssl req -new -nodes -text -out root.csr \ -keyout root.key -
subj "/CN=root.yourdomain.com"
```

```
chmod og-rwx root.key
```

Sau đó, ký yêu cầu bằng khóa để tạo root CA (sử dụng vị trí tệp cấu hình OpenSSL mặc định trên Linux):

```
openssl x509 -req -in root.csr -text -days 3650 \ -extfile
/etc/ssl/openssl.cnf -extensions v3_ca \ -signkey root.key -out
root.crt
```

Cuối cùng, tạo Chứng thư máy chủ được ký bởi root CA mới:

```
openssl req -new -nodes -text -out server.csr \ -keyout
server.key -subj "/CN=dbhost.yourdomain.com"
chmod og-rwx server.key
openssl x509 -req -in server.csr -text -days 365 \ -CA root.crt
-CAkey root.key -CAcreateserial \ -out server.crt
```

2.5. Kết luận chương 2

Trong chương 2 đã trình bày các nội dung về cơ chế mã hóa trong HQTCSDL PostgreSQL. Cụ thể :

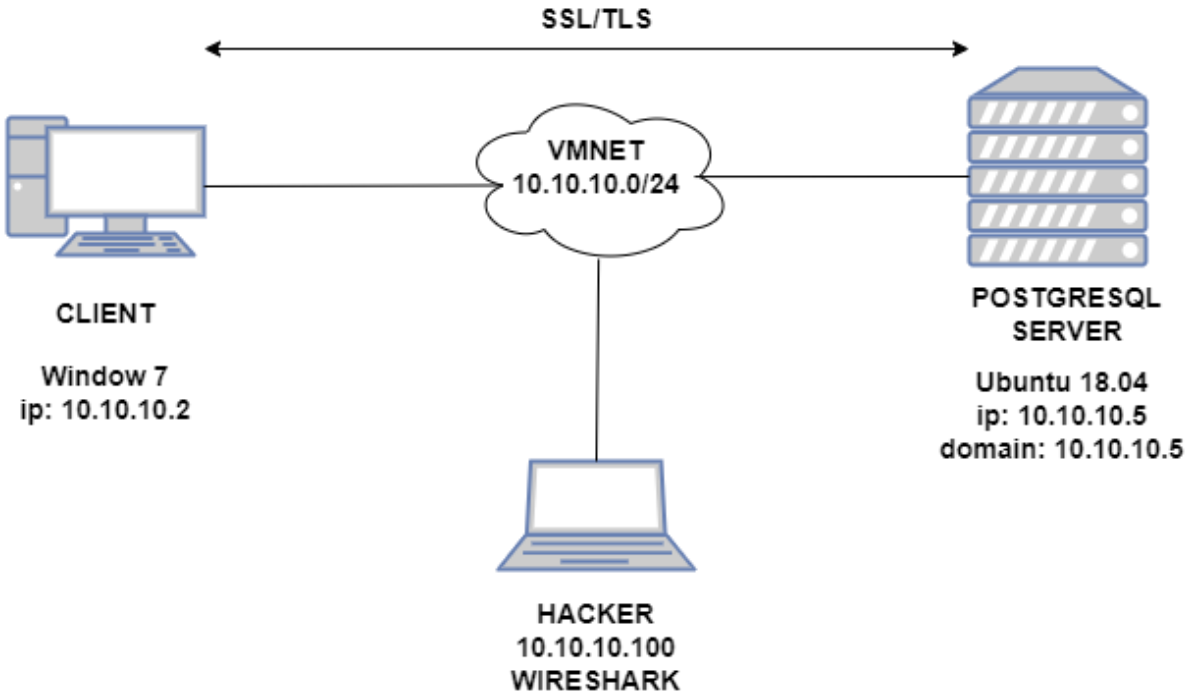
- Nguy cơ mất an toàn trong triển khai HQTCSDL
- Cơ chế đảm bảo ATTT trong HQTCSDL
- Cơ chế mã hóa dữ liệu lưu trữ
- Cơ chế mã hóa dữ liệu kênh truyền

Nội dung triển khai mô hình thực nghiệm sẽ được trình bày trong chương 3 với cơ sở là những lý thuyết đã trình bày trong chương 2.

CHƯƠNG 3. TRIỂN KHAI CÀI ĐẶT MỘT SỐ CƠ CHẾ MÃ HÓA DỮ LIỆU TRONG POSTGRESQL

3.1. Mô hình triển khai

Trong chương này, đồ án tiến hành triển khai cơ sở dữ liệu PostgreSQL trên máy chủ Ubuntu 18.04. Cung cấp dịch vụ cơ sở dữ liệu cho người dùng thông qua mạng Vmnet 10.10.10.0/24 như mô hình 3.1.



Hình 3.1. Mô hình triển khai dịch vụ cơ sở dữ liệu PostgreSQL

Các bước triển khai mô hình sẽ bao gồm:

- Bước 1: Cấu hình mạng cơ bản

+ Cài đặt HĐH trên các máy chủ, máy trạm.

+ Cài đặt địa chỉ IP

+ Kiểm tra kết nối giữa các máy.

- Bước 2: Triển khai cơ sở dữ liệu PostgreSQL

+ Cài đặt PostgreSQL trên máy chủ

+ Cài đặt cơ sở dữ liệu người dùng

+ Tiến hành truy cập vào CSDL từ người dùng ở xa

- Bước 3: Triển khai cơ chế mã hoá trên máy chủ PostgreSQL

+ Mã hoá dữ liệu kênh truyền

+ Mã hoá cơ sở dữ liệu người dùng

- Bước 4: Đánh giá và kiểm tra

+ Tiến hành phân tích gói tin với Wireshark

3.2. Các bước cài đặt cơ bản

Trong phần này trình bày các bước cài đặt PostgreSQL trên máy chủ và máy client.

Bước 1: Cài đặt Vmware trên máy tính

Bước 2: Cài đặt Ubuntu 18.04 trên máy chủ.

Bước 3: Cài đặt PostgreSQL trên máy chủ và máy client. Khởi tạo cơ sở dữ liệu và truy cập từ máy client.

Bước 4: Cấu hình địa chỉ IP và kiểm tra kết nối thông thường giữa máy chủ và máy client.

Với các bước 1 và 2 là các bước cơ bản khi cài đặt môi trường. Trong đồ án này trình bày từ bước 3.

Sau khi có được môi trường cài đặt, tiến hành thực hiện cài đặt PostgreSQL

- **Thêm PostgreSQL 12 repository**

+ Nhập khóa GPG và thêm kho lưu trữ PostgreSQL 12 vào máy Ubuntu.

```
curl -fsSL https://www.PostgreSQL.org/media/keys/ACCC4CF8.asc|sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/PostgreSQL.gpg
```

+ Thêm repository bao gồm: PostgreSQL-client, PostgreSQL, libpq-dev, PostgreSQL-server-dev, pgadmin packages như hình sau

```
echo "deb http://apt.PostgreSQL.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" |sudo tee  /etc/apt/sources.list.d/pgdg.list
```

+ Sau khi cài đặt thành công các gói PostgreSQL 12 repository, tiến hành cài đặt PostgreSQL phiên bản 12 với lệnh sau, và kết quả được như hình:

```
sudo apt -y install PostgreSQL-12 PostgreSQL-client-12
```

+ Kiểm tra xem PostgreSQL 12 đã hoạt động chưa bằng lệnh

```
Systemctl status PostgreSQL
```

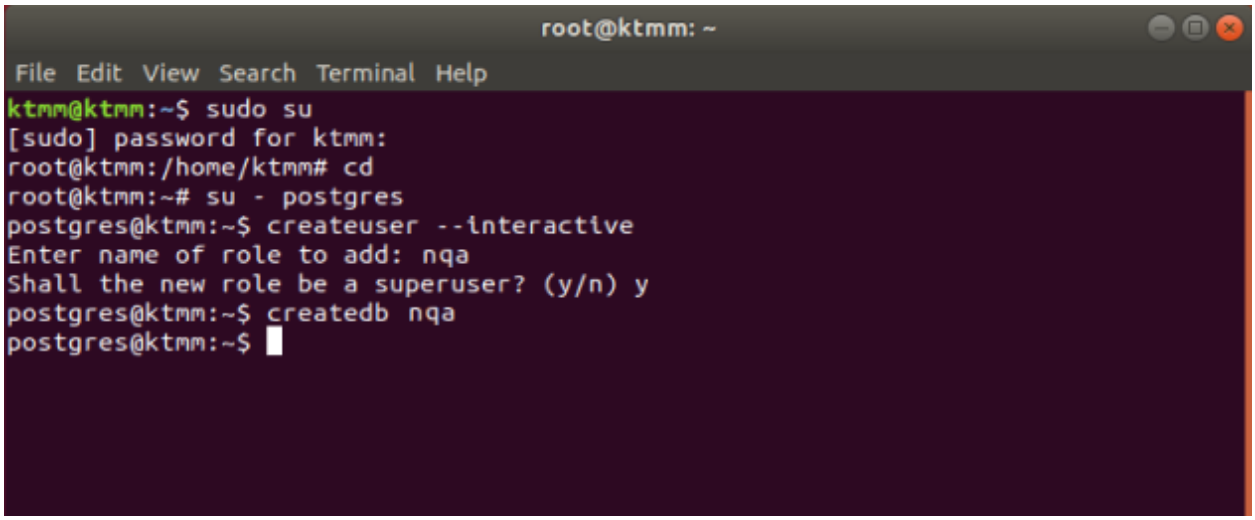
+ Kết quả được như hình. PostgreSQL đã hoạt động trên máy chủ thành công.

3.3. Triển khai thực nghiệm và đánh giá kết quả

3.3.1. Cơ chế mã hóa dữ liệu kênh truyền

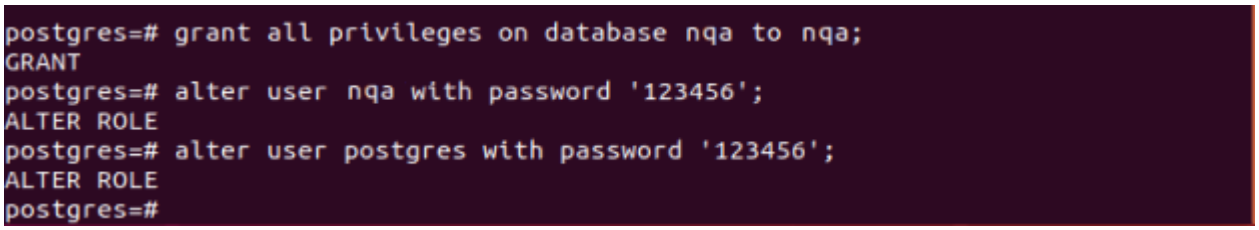
Tạo cơ sở dữ liệu

Ví dụ tạo database có tên nqa với user nqa như sau:



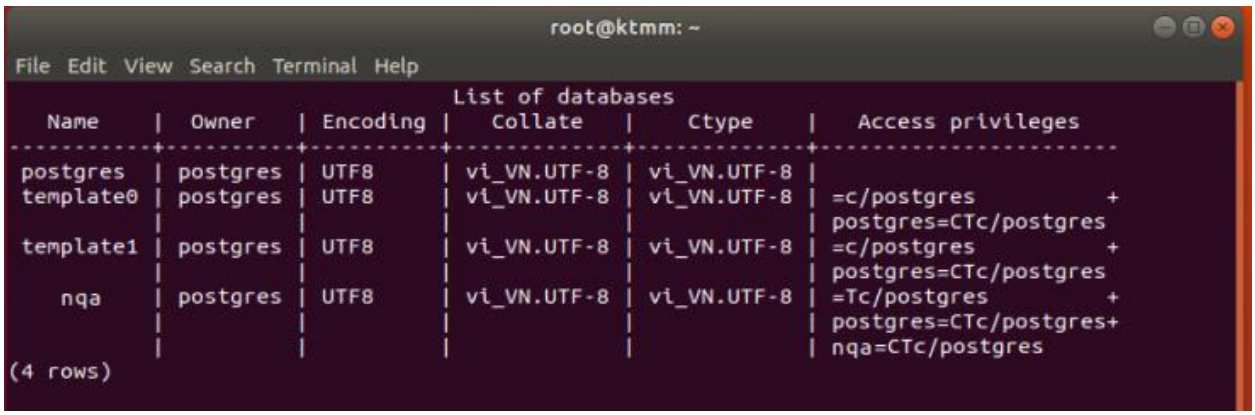
Hình 3.2. Tạo database và user

Tiếp theo, đặt mật khẩu đăng nhập database cho các user như sau:



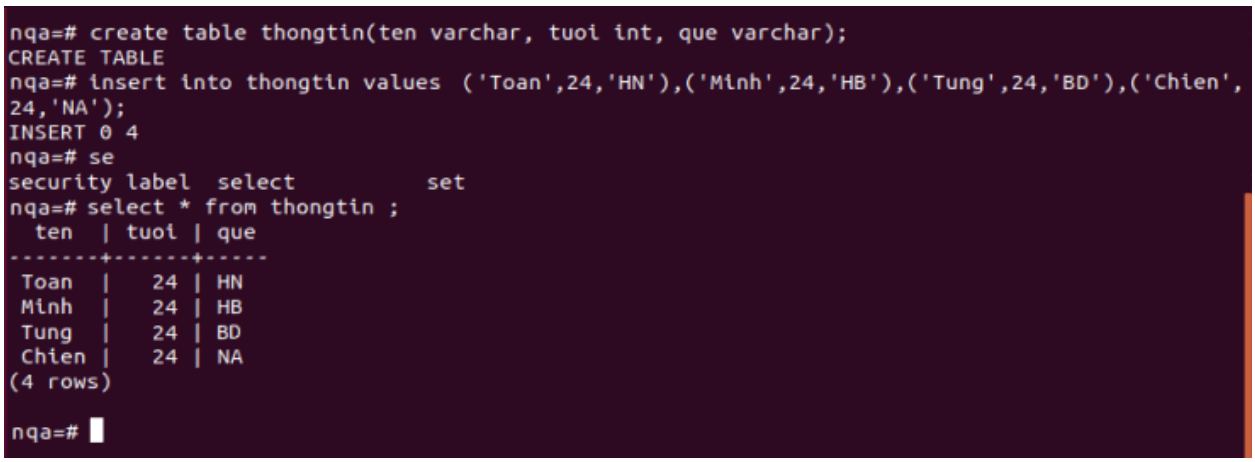
Hình 3.3. Đặt mật khẩu cho user

Kiểm tra danh sách database và user đã tạo, thấy database nqa và user nqa đã tạo thành công.



Hình 3.4. Danh sách database và user đã có

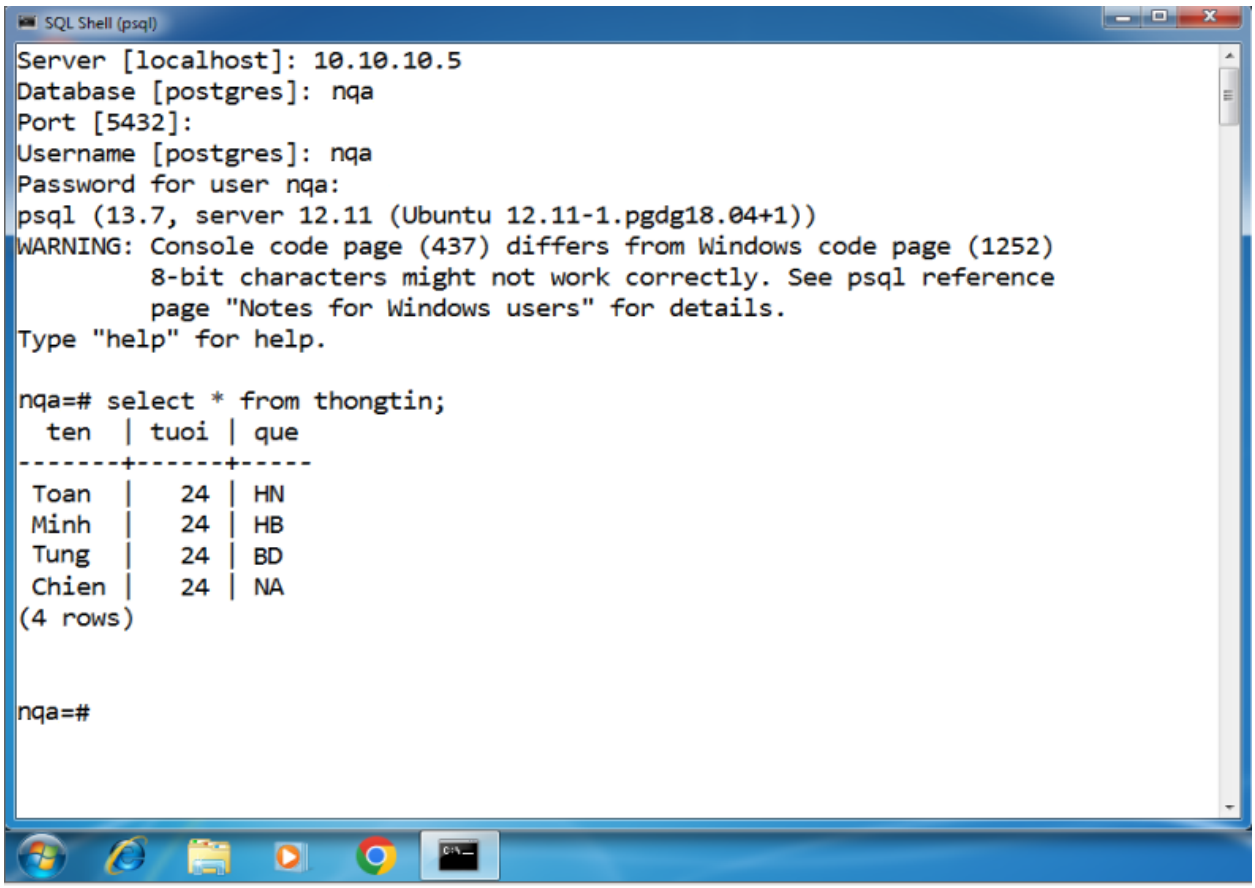
Tiến hành tạo bảng *thongtin* và thêm dữ liệu vào bảng như sau:



Hình 3.5. Thêm dữ liệu vào database

Sau khi đã khởi tạo, thêm dữ liệu vào database thành công, tiến hành kết nối tới database từ máy Client để kiểm tra xem Client có thể thao tác trên cơ sở dữ liệu được hay chưa.

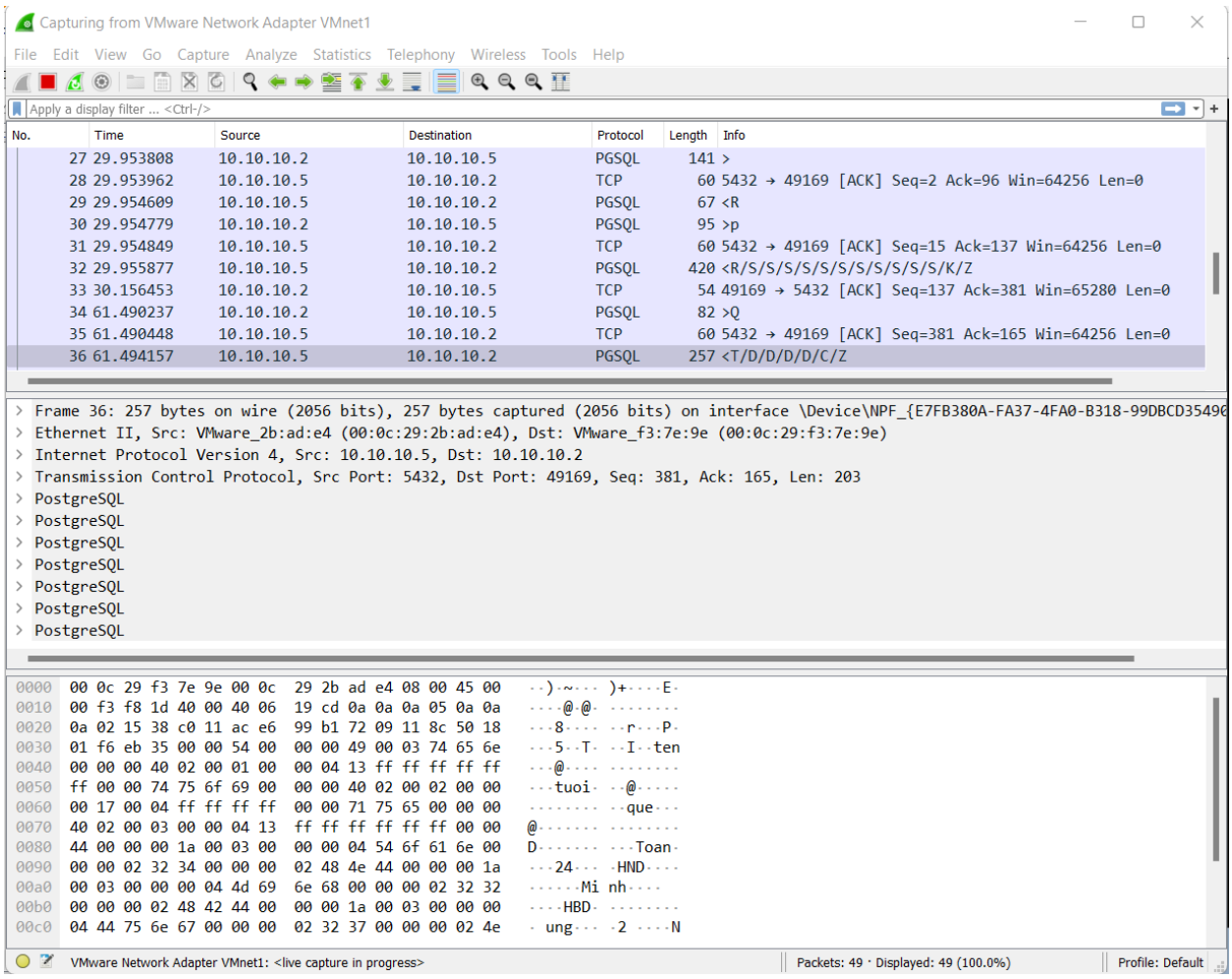
Trên máy Client kết nối tới database như sau:



Hình 3.6. Kết nối và truy vấn thông tin từ máy Client

Có thể thấy rằng khi kết nối database từ client đến server, đường truyền chưa được bảo vệ bởi SSL/TLS. Các bước thực hiện bảo mật dữ liệu kênh truyền với SSL/TLS sẽ được trình bày dưới đây.

Khi chặn bắt gói tin với phần mềm Wireshark, ta được kết quả như sau:



Hình 3.7. Dùng WireShark để bắt gói tin khi chưa thiết lập TLS 1.3

Có thể thấy rằng, nếu không được bảo mật đường truyền với TLS, khi người dùng truy vấn dữ liệu từ xa, các gói tin đều ở dạng rõ và kẻ tấn công hoàn toàn có thể đọc được nội dung bên trong. Để bảo mật dữ liệu kênh truyền, ta cần thiết lập như phần trình bày dưới đây.

Bảo mật dữ liệu kênh truyền với SSL/TLS

+ Sử dụng OpenSSL để tạo Chứng thư máy chủ tự ký và khóa riêng tư cho máy chủ PostgreSQL:

```
# openssl req -new -x509 -days 365 -nodes -text -out server.crt -keyout server.key \
  -subj "/CN=10.10.10.5"
# chmod og-rwx server.key server.crt
```

+ Tạo thư mục ~/.PostgreSQL/ chứa Chứng thư và di chuyển Chứng thư vào thư mục

```
# mkdir ~/.PostgreSQL
# mv server.* ~/.PostgreSQL/
```

Vì Chứng thư là Chứng thư tự ký, nên có thể sử dụng Chứng thư máy chủ (server.crt) làm Chứng thư gốc đáng tin cậy (root.crt) cho máy client. Vì root.crt cũng giống như server.crt. Lưu ý "~ / .PostgreSQL / root.crt" được psql sử dụng là vị trí mặc định và có thể định cấu hình.

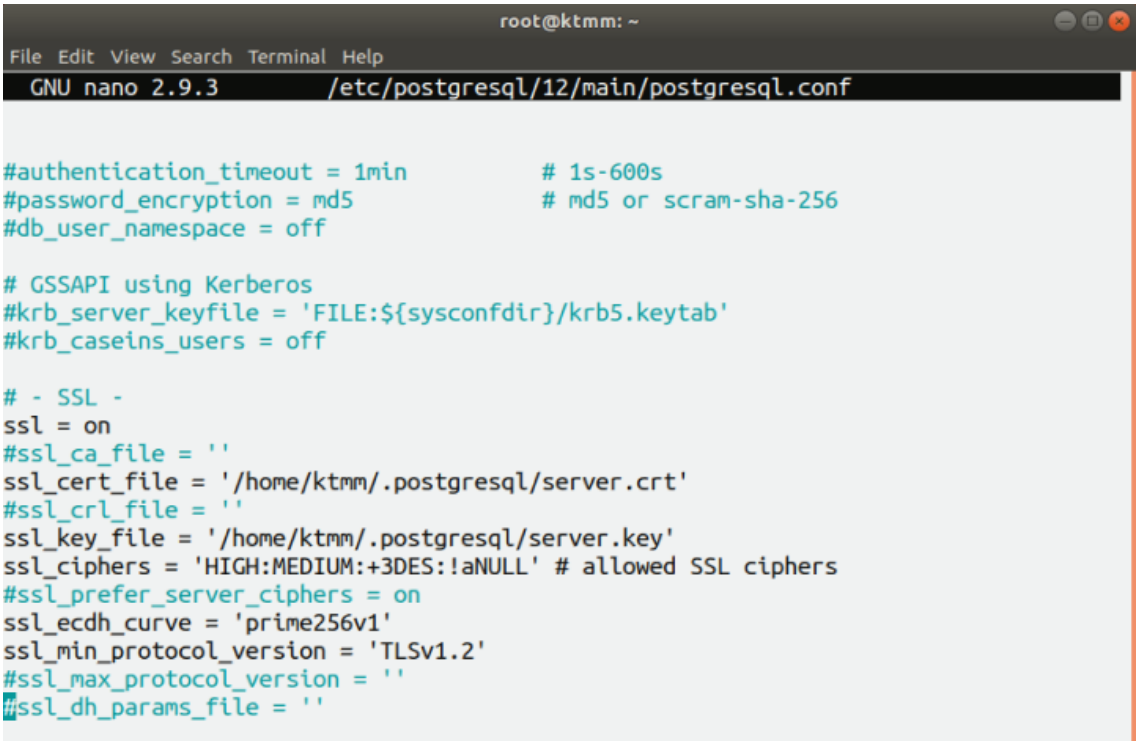
```
# cp ~/.PostgreSQL/server.crt ~/.PostgreSQL/root.crt
```

+ Sau đó cấp quyền chủ sở hữu cho user postgres với lệnh sau:

```
# sudo chown postgres:postgres ~/.PostgreSQL/server*
```

Thay đổi *ssl_cert_file* and *ssl_key_file* trong file *PostgreSQL.conf* và trỏ đến thư mục *~/.PostgreSQL* :

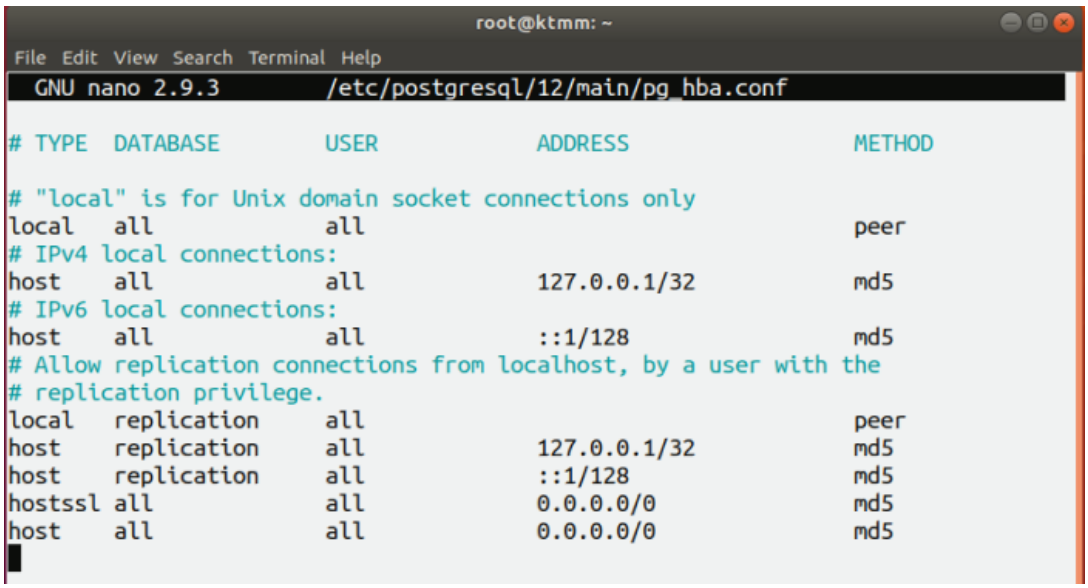
```
# nano /etc/PostgreSQL/12/main/PostgreSQL.conf
```



Hình 3.8. Chỉnh sửa file PostgreSQL.conf

Chỉnh sửa file pg_hba.conf , thêm hostssl để cho phép client kết nối thông qua SSL như sau:

```
# nano /etc/PostgreSQL/12/main/pg_hba.conf
```



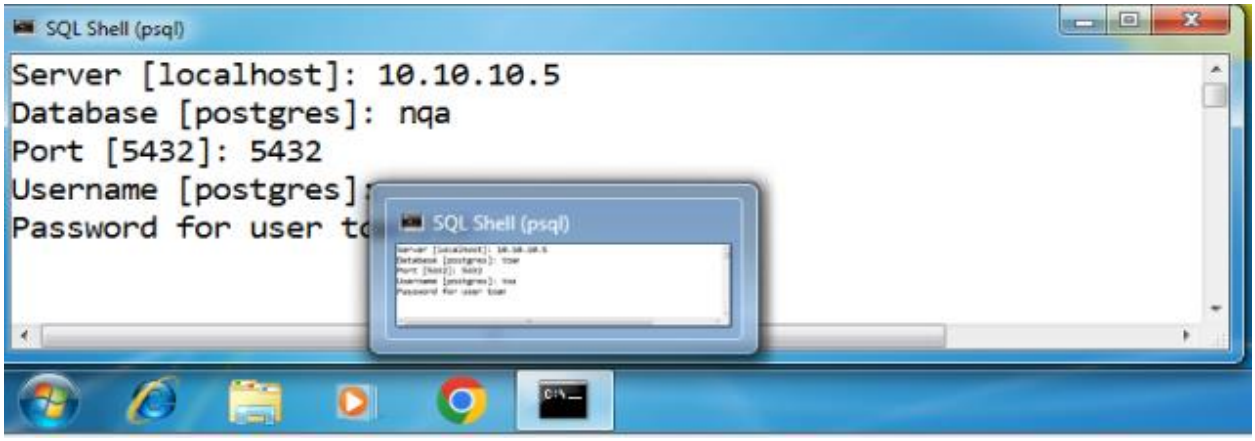
Hình 3.9. Chỉnh sửa file pg_hba.conf

Sau khi chỉnh sửa 2 file *PostgreSQL.conf* và *pg_hba.conf*, ta tiến hành lưu lại và khởi động lại hệ thống

```
# systemctl restart postgresql
```

Kiểm tra kết nối

Trên máy Client chạy hệ điều hành Window 7 đã cài PostgreSQL 12, ta tiến hành mở SQL shell để kết nối đến database đã tạo trước đó là nqa như sau:

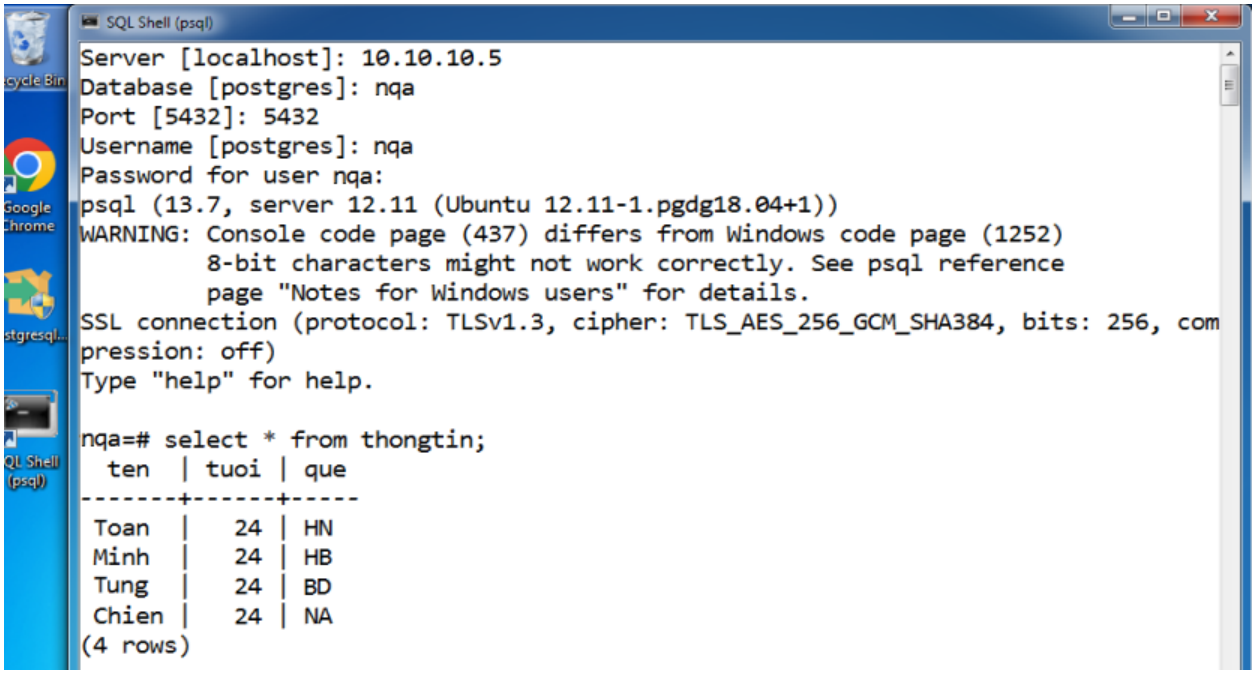


Hình 3.10. Kết nối đến server

Với các tham số cần nhập:

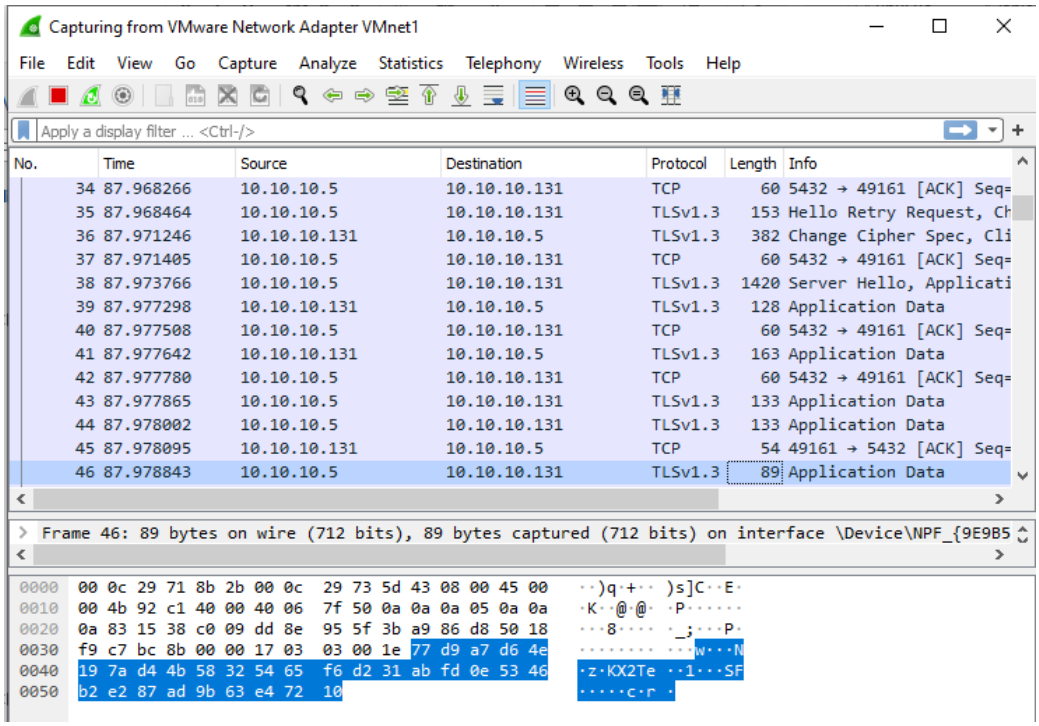
+ Server : 10.10.10.5 là địa chỉ ip của máy server
+ Database : nqa là tên database đã tạo trước đó
+ Port: 5432 là cổng truy cập
+ Username: nqa
+ Password: mật khẩu của user nqa

Kết quả ta có thể kết nối thành công đến database nqa từ máy client với kết nối SSL/TLS như hình. Tại đây có thể truy vấn dữ liệu từ database.



Hình 3.11. Kết nối thành công với giao thức TLS 1.3 và truy vấn dữ liệu từ database

Khi sử dụng phần mềm chặn bắt gói tin WireShark để bắt gói tin khi máy client kết nối đến server ta được kết quả như sau:



Hình 3.12. Dùng WireShark để bắt gói tin khi đã thiết lập TLS 1.3

Như vậy có thể thấy rằng đường truyền đã được mã hóa. Bộ thuật toán được sử dụng để mã hóa là TLS_AES_256_GCM_SHA384.

Do đó việc triển khai cơ chế mã hóa dữ liệu kênh truyền của đồ án đã thực hiện thành công.

3.3.2. Cơ chế mã hóa dữ liệu lưu trữ

Với những lý thuyết đã được trình bày ở mục 2.3, trong phần 3.3.2 này đồ án sẽ trình bày triển khai thực nghiệm cơ chế mã hóa dữ liệu lưu trữ với hệ quản trị cơ sở dữ liệu PostgreSQL. Cơ chế được sử dụng trong triển khai này là cơ chế mã hóa dữ liệu TDE mức cột.

```
nqa=# create table thongtin(ten varchar, tuoi int, que varchar);
CREATE TABLE
nqa=# insert into thongtin values  ('Toan',24,'HN'),('Minh',24,'HB'),('Tung',24,'BD'),('Chien',
24,'NA');
INSERT 0 4
nqa=# se
security label  select          set
nqa=# select * from thongtin ;
  ten   | tuoi | que
--------+------+-----
 Toan   |   24 | HN
 Minh   |   24 | HB
 Tung   |   24 | BD
 Chien  |   24 | NA
(4 rows)

nqa=#
```

Hình 3.13. Dữ liệu đã khởi tạo trước

Với các dữ liệu đã được khởi tạo trước đó, giả sử thông tin cột *que* là những thông tin về quê quán cần được bảo mật.

Bước 1. Khởi tạo tiện ích bảo mật pgcrypto. Đây là tiện ích bảo mật cung cấp các hàm mã hóa dữ liệu cho cơ sở dữ liệu.

```
nqa=# CREATE EXTENSION pgcrypto;
CREATE EXTENSION
```

Các hàm mã hóa/ giải mã được sử dụng là:

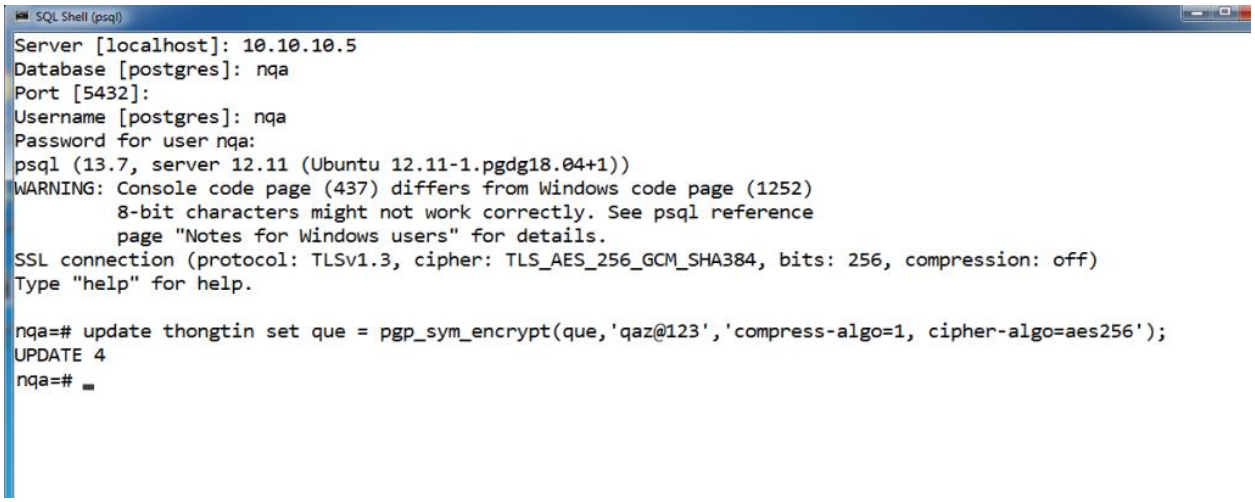
```
pgp_sym_encrypt(data, password [, options text] ) return bytea
pgp_sym_decrypt(data :: bytea, password [, options text ])  returns text
```

Với [, options text] cho phép tùy chọn thuật toán mã hóa như AES, Blowfish. Trong đồ án này sẽ triển khai mã hóa TDE mức cột với thuật toán AES-256.

Bước 2. Thao tác mã hóa với cột *que* như sau:

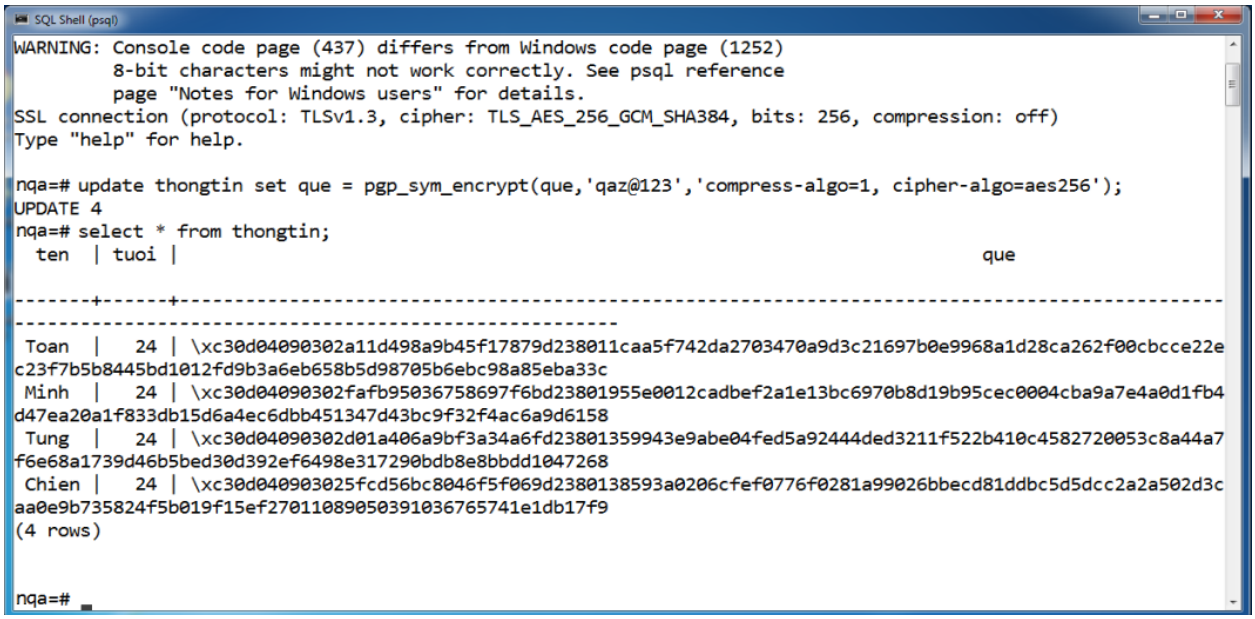
Có thể thực hiện trên bất kì máy nào khi đã kết nối thành công đến database *nqa*. Trên máy Client chạy Win 7, đã kết nối thành công đến database, thực hiện lệnh:

```
nqa=# update thongtin set que = pgp_sym_encrypt(que,'qaz@123','compress-algo=1, cipher-algo=aes256');
```



Hình 3.14. Thiết lập mã hóa TDE mức cột

Như hình trên, ta đã thực hiện mã hóa cột *que* với mật khẩu là *qaz@123*, thuật toán mã hóa là AES-256.

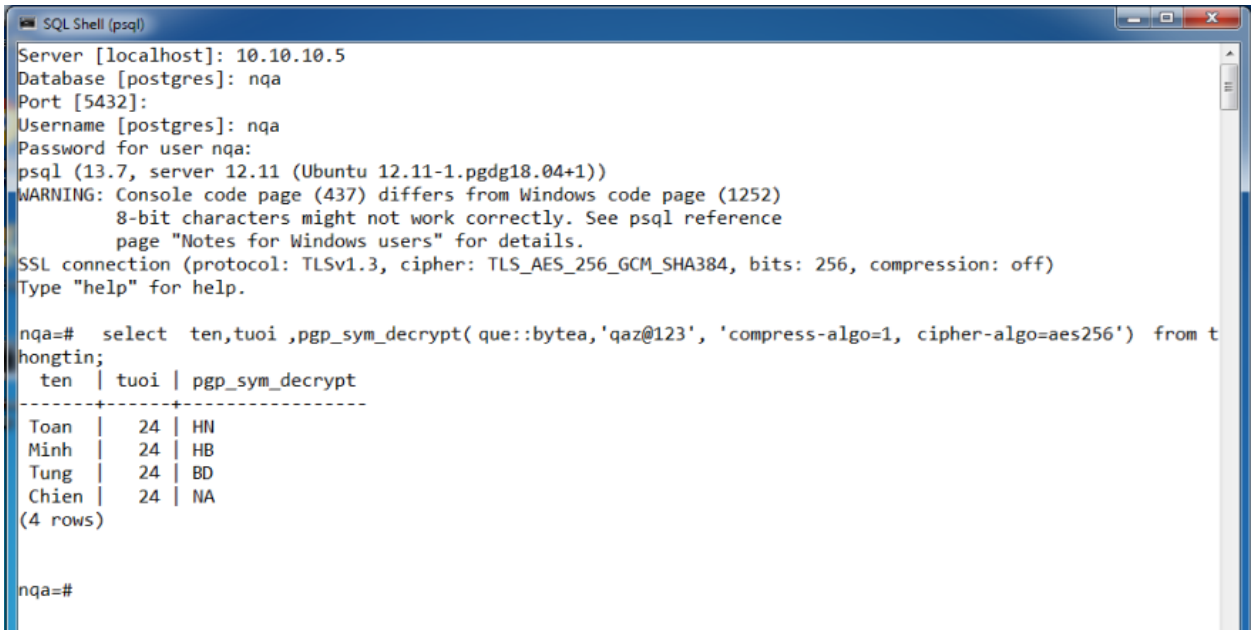


Hình 3.15. Dữ liệu cột đã được mã hóa

Có thể thấy rằng khi truy vấn dữ liệu thông thường, dữ liệu cột đã được mã hóa và không thể đọc được nếu không có mật khẩu. Để truy vấn dữ liệu rõ cần truy vấn kết hợp hàm giải mã pgp_sym_decrypt() như sau:

```
nqa=# select ten,tuoi,pgp_sym_decrypt ( que::bytea, 'qaz@123', 'compress-algo=1, cipher-algo=aes256') from thongtin;
```

Kết quả có thể truy vấn được như sau:



Hình 3.16. Truy vấn dữ liệu với hàm giải mã pgp_sym_decrypt()

3.3.3. Đánh giá kết quả

Cơ chế mã hóa dữ liệu kênh truyền

- Với cơ chế mã hóa dữ liệu kênh truyền với TLS1.3, toàn bộ đường truyền từ client đến server và ngược lại đã được mã hóa, dữ liệu được truyền một cách an toàn trên kênh như hình 3.14.

- Với việc kênh truyền được bảo mật, giúp tránh tấn công phổ biến như tấn công xen giữa, giúp đảm bảo được các tính chất bảo mật của dữ liệu: tính bí mật, tính toàn vẹn , tính xác thực.

Cơ chế mã hóa dữ liệu lưu trữ

Với cơ chế mã hóa dữ liệu lưu trữ, giả sử có các kịch bản tấn công như sau:

- Kịch bản 1: Kẻ tấn công sử dụng lỗ hỏng bí mật nào đó đã biết trên CentOS7 để truy cập vào máy chủ PostgreSQL (cài đặt Ubuntu 18.04) hoặc kẻ tấn

công có khả năng truy cập vào máy chủ PostgreSQL (kẻ tấn công là nhân viên trong công ty, doanh nghiệp:

Khi đó, kẻ tấn công sẽ có thể truy xuất file cơ sở dữ liệu bên trong hệ thống. Tuy nhiên, với cơ chế mặc định sử dụng khóa mã hóa masterkey là mật khẩu đăng nhập của user trong hệ quản trị cơ sở dữ liệu để mã hóa dữ liệu của user đó, kẻ tấn công sẽ không thể xem được nội dung bên trong file mà chúng lấy được nếu không có mật khẩu của user trong hệ quản trị cơ sở dữ liệu.

- Kịch bản 2: Kẻ tấn công đã có thể truy cập vào máy chủ PostgreSQL (cài đặt Ubuntu 18.04) như kịch bản 1, thêm vào đó chúng có thêm tài khoản và mật khẩu của user bên trong hệ quản trị. Khi đó chúng tiến hành truy vấn cơ sở dữ liệu của user. Tuy nhiên do các dữ liệu quan trọng nhạy cảm đã được mã hóa với khóa riêng của người dùng. Tức là kẻ tấn công cần biết thêm 1 khóa nữa để có thể xem được toàn bộ dữ liệu bên trong. Nếu không, kẻ tấn công chỉ có thể xem được dữ liệu như hình 3.18.

Như vậy. Việc triển khai cơ chế mã hóa TDE mức cột là cơ chế giúp đảm bảo an toàn cho dữ liệu của người dùng trong hệ quản trị cơ sở dữ liệu PostgreSQL.

3.4. Kết luận chương 3

Chương 3 đã trình bày chi tiết, rõ ràng quá trình triển khai cài đặt hệ quản trị cơ sở dữ liệu PostgreSQL, triển khai 2 cơ chế mã hóa đã trình bày tại chương 2.

Nội dung bao gồm:

+ Mô hình triển khai cài đặt PostgreSQL
+ Triển khai thực nghiệm và đánh giá kết quả.

Kết quả đạt được trong chương này là đã thực hiện thành công hai cơ chế mã hóa dữ liệu bao gồm cơ chế mã hóa dữ liệu lưu trữ và cơ chế mã hóa dữ liệu kênh truyền, với 2 cơ chế này, dữ liệu của người dùng sẽ được bảo vệ.

KẾT LUẬN

Sau thời gian nỗ lực nghiên cứu, tìm hiểu dưới sự giúp đỡ tận tình của thầy giáo Nguyễn Như Chiến, em đã hoàn thành đồ án tốt nghiệp "TÌM HIỂU, TRIỂN KHAI MỘT SỐ CƠ CHẾ MÃ HOÁ DỮ LIỆU TRONG HQTCSDL POSTGRESQL".

Các kết quả đã đạt được như sau:

- Về lý thuyết

+ Trình bày được về hệ quản trị CSDL PostgreSQL.

+ Hiểu được những nguy cơ tấn công gây mất an toàn dữ liệu trên hệ quản trị cơ sở dữ liệu nói chung và hệ quản trị cơ sở dữ liệu PostgreSQL nói riêng.

+ Đưa ra một số cơ chế mã hóa dữ liệu bên trong hệ quản trị cơ sở dữ liệu PostgreSQL.

- Kết quả đạt được

+ Nắm được một số cơ chế đảm bảo an trong PostgreSQL.

+ Đưa ra biện pháp bảo mật mã hóa dữ liệu lưu trữ mức cột và mã hóa dữ liệu kênh truyền để phòng chống lại các cuộc tấn công của hacker.

- Định hướng phát triển

+ Tiếp tục nghiên cứu, tìm hiểu sâu hơn về PostgreSQL.

+ Tìm hiểu tình hình của một số doanh nghiệp, đề xuất sử dụng PostgreSQL trong thực tế.

TÀI LIỆU THAM KHẢO

[1]. Jurgen Schoning: *Mastering PostgreSQL1*, 4th, Packt, 2020.

[2]. Documentation of PostgreSQL: Encryption options, 2022.

https://www.PostgreSQL.org/docs/current/encryption-options.html

[3]. Data page Marc linster: *Securi Best Practices for PostgreSQL*, 2021.

[4]. PostgreSQL uses pgcrypto function for data encryption

https://blog.actorsfit.com/a?ID=00001-9b767233-88f5-4d39-be55-bf1d0b3c88f2

[5]. Rajkumar Raghuwansh. *How to implement Column and Row level security in PostgreSQL*, 2020

https://www.enterprisedb.com/postgres-tutorials/how-implement-column-and-row-level-security-postgresql