Лабораторная работа № 11

Тема: Реализация мандатного и дискреционного разграничение доступа в PostgreSQL

Цель: изучить и применить модели мандатного и дискреционного управления доступом в PostgreSQL.

Время выполнения лабораторной работы (аудиторные часы): 4 часа

Оборудование и программное обеспечение: ПК с установленной PostgreSQL, текстовый редактор для написания SQL-запросов.

1. Теоретические сведения

1.1 Мандатное управление доступом (Mandatory Access Control, MAC)

Мандатное управление доступом (MAC) — это строгая модель контроля, где доступ к объектам базы данных определяется глобальными политиками безопасности. Эта модель фокусируется на метках безопасности, которые назначаются как субъектам (пользователям или ролям), так и объектам (таблицам, строкам, столбцам). Доступ возможен только в том случае, если метки субъекта и объекта совместимы с установленными правилами.

Основные характеристики MAC:

- Пользователи не могут изменять метки безопасности или права доступа.
- Объекты и субъекты получают метки, такие как секретно, конфиденциально, или общедоступно.
- Доступ предоставляется только при совпадении меток безопасности и уровней допуска.

Преимущества MAC:

- Высокий уровень безопасности.
- Предотвращение несанкционированного доступа.
- Строгий контроль за всеми операциями.

Недостатки MAC:

- Меньшая гибкость в управлении доступом.
- Сложность настройки для систем с большим числом пользователей и объектов.

Для реализации мандатного управления доступом PostgreSQL поддерживает расширение sepgsql, которое интегрируется с системой SELinux. Это расширение добавляет метки безопасности к объектам базы данных

(таблицам, строкам, столбцам) и использует политики SELinux для контроля доступа.

Этапы настройки sepgsql:

1. Убедитесь, что PostgreSQL установлен с поддержкой SELinux.
2. Активируйте расширение в конфигурации PostgreSQL: `shared_preload_libraries = 'sepgsql'`
3. Перезапустите PostgreSQL, чтобы изменения вступили в силу.

После активации sepgsql можно назначать метки безопасности объектам базы данных используя SQL запрос:

```
ALTER TABLE BD ADD COLUMN security_label TEXT;
UPDATE BD SET security_label = 'confidential' WHERE
department_id = 1;
```

В этом примере строкам, относящимся к отделу с идентификатором 1, присваивается метка confidential. Только пользователи с соответствующим уровнем доступа смогут получить доступ к этим данным.

1.2 Дискреционное управление доступом (Discretionary Access Control, DAC)

Дискреционное управление доступом (DAC) — это гибкая модель, где права доступа к объектам определяются владельцами этих объектов. Владельцы могут назначать права другим пользователям или ролям, а также удалять их. DAC широко используется в PostgreSQL благодаря своей простоте и гибкости.

Основные характеристики DAC:

- Владельцы объектов имеют полный контроль над правами доступа.
- Доступ управляется с помощью команд GRANT и REVOKE.
- PostgreSQL позволяет объединять пользователей в роли для упрощения управления.

Достоинства:

- Простота в использовании.
- Гибкость в настройке доступа.
- Возможность делегирования прав.

Недостатки DAC:

- Риск несанкционированной передачи прав.
- Сложность контроля в системах с большим числом объектов и пользователей.

DAC в PostgreSQL подходит для большинства стандартных приложений, где необходимы точные и индивидуальные настройки доступа.

В PostgreSQL управление доступом начинается с создания пользователей и ролей. Роли позволяют централизовать управление правами, а пользователи могут быть связаны с одной или несколькими ролями.

Пример создания ролей:

```
CREATE ROLE manager_role;
CREATE ROLE analyst_role;
```

Пример создания пользователей:

```
CREATE USER manager WITH PASSWORD 'securepassword';
CREATE USER analyst WITH PASSWORD 'securepassword';
```

Привязка пользователей к ролям:

```
GRANT manager_role TO manager;
GRANT analyst_role TO analyst;
```

В DAC права доступа задаются командами GRANT и REVOKE. Эти команды позволяют владельцам объектов (например, таблиц) управлять доступом других пользователей или ролей.

Пример предоставления прав:

```
GRANT SELECT, INSERT ON employees TO manager_role;
```

Пример отзыва прав:

```
REVOKE INSERT ON employees FROM analyst_role;
```

2. Задание для самостоятельного выполнения работы

1. Подготовка к работе
   - Убедитесь, что PostgreSQL установлен и настроен.
   - Создайте тестовую базу данных для выполнения задания.
2. Выбор варианта задания
   - Получите свой вариант задания из таблицы вариантов (см. таблицу ниже).
   - Ознакомьтесь с требованиями своего варианта и выполните соответствующие шаги.
3. Создание пользователей и ролей
   - Создайте пользователей и роли, указанные в вашем варианте задания.
   - Назначьте роли пользователям.
4. Реализация мандатного управления доступом
   - Настройте метки безопасности для объектов базы данных в соответствии с вашим вариантом.

- Используйте расширение **sepgsql.**

5. Реализация дискреционного управления доступом
   - Настройте права доступа для пользователей и ролей с использованием команд GRANT и REVOKE в соответствии с вашим вариантом.
   - Проверьте, что права доступа соответствуют требованиям задания.
6. Тестирование настроек доступа
   - Переключайтесь между ролями и пользователями, чтобы проверить правильность настроек.
   - Убедитесь, что доступ к данным ограничен согласно заданным правилам.
7. Задокументируйте процесс выполнения задания.

Таблица 1 – Варианты заданий для самостоятельного выполнения

Вариант	**Задание**
1.	Создайте таблицу **clients** с полями `client_id`, `name`, `email`. Реализуйте мандатное управление доступом: пользователи с уровнем допуска confidential могут просматривать данные, но не изменять их. Настройте дискреционное управление для предоставления права на вставку данных роли manager_role.
2.	Создайте таблицу **orders** с полями `order_id`, `client_id`, `order_date`. Реализуйте мандатное управление доступом: данные с меткой secret доступны только пользователям с уровнем допуска secret. Настройте дискреционный доступ для роли analyst_role с правами на чтение данных.
3.	Создайте таблицу **products** с полями `product_id`, `product_name`, `price`. Задайте мандатные метки безопасности public и restricted. Пользователи с уровнем доступа restricted могут изменять данные, а пользователи с уровнем public — только просматривать. Настройте права для ролей manager_role и analyst_role с разными уровнями доступа.
4.	Создайте таблицу **employees** с полями `employee_id`, `name`, `salary`. Реализуйте мандатное управление доступом: зарплаты с меткой confidential доступны только пользователям с уровнем допуска confidential. Настройте дискреционный доступ для роли hr_role с правами на чтение и обновление данных.
5.	Создайте таблицу **transactions** с полями `transaction_id`, `amount`, `transaction_date`. Используйте мандатное управление для защиты данных с меткой financial. Настройте дискреционный доступ для роли finance_role, которая может читать и добавлять данные, но не удалять их.

6.	Создайте таблицу **inventory** с полями `item_id`, `product_id`, `quantity`. Реализуйте мандатное управление доступом: метки restricted запрещают изменение данных пользователями с уровнем доступа ниже restricted. Настройте дискреционный доступ для роли warehouse_role, которая может только просматривать данные.
7.	Создайте таблицу **sales** с полями `sale_id`, `product_id`, `quantity`, `sale_date`. Реализуйте мандатное управление для меток public и private. Настройте дискреционный доступ для роли sales_role с правами на добавление и просмотр данных.
8.	Создайте таблицу **shipments** с полями `shipment_id`, `order_id`, `shipment_date`. Данные с меткой restricted должны быть доступны только пользователям с соответствующим уровнем доступа. Настройте дискреционный доступ для роли logistics_role с правами на чтение и обновление данных.
9.	Создайте таблицу **projects** с полями `project_id`, `project_name`, `budget`. Реализуйте мандатное управление для меток classified и unclassified. Настройте дискреционный доступ для роли project_manager_role с правами на добавление и изменение данных.
10.	Создайте таблицу **invoices** с полями `invoice_id`, `client_id`, `total_amount`. Реализуйте мандатное управление для защиты данных с меткой financial_confidential. Настройте дискреционный доступ для роли billing_role, которая может читать и добавлять данные, но не удалять их.

Контрольные вопросы

1. Чем отличается мандатное управление доступом от дискреционного?
2. Как установить метки безопасности для объектов базы данных в PostgreSQL?
3. Какие команды используются для настройки дискреционного управления доступом?
4. Как протестировать, что права доступа настроены правильно?
5. Какие преимущества и ограничения есть у мандатного управления доступом?
6. Какие ограничения существуют при использовании мандатного управления доступом (MAC) в PostgreSQL?
7. Как реализовать настройку меток безопасности для строк таблицы с помощью расширения sepgsql?
8. Какие проблемы могут возникнуть при использовании дискреционного управления доступом (DAC) в системах с большим количеством пользователей?

9. Чем отличается команда GRANT от команды REVOKE, и в каких случаях они используются?
10. Как проверять и тестировать права доступа пользователей или ролей в PostgreSQL?

Литература

1. Blog PostgreSQL. Using sepgsql for Mandatory Access Control. URL: https://www.postgresql.org/
2. Агафонов В.В. Безопасность баз данных: учебное пособие. — М.: Юрайт, 2021. — 340 с.
3. Липаев В.В. Основы информационной безопасности. — М.: Академия, 2019. — 432 с.
4. PostgreSQL Security: Role and Privilege Management. URL: https://severalnines.com/blog/postgresql-security
5. Using GRANT and REVOKE in PostgreSQL. Digital Ocean Community. URL: https://www.digitalocean.com/community/tutorials/