Laboratorio de Controle de Acesso

Laboratório de Construção de Arquitetura de Controle de Acesso no Postgresql para o banco de exemplo dvdrental

Executar os passos abaixo, gerando os entregáveis listados no final do documento.

Passo 1: A Nova Arquitetura de Schemas

Primeiro, criaremos uma estrutura de schemas que reflete as diferentes áreas de negócio da nossa locadora. Cada schema será uma "sala" com um propósito específico.

```
-- Schema para os dados brutos, as tabelas físicas. O coração do sistema.
CREATE SCHEMA data_mart;

-- Schema para as operações do dia a dia da loja (aluguéis, devoluções, clientes).
CREATE SCHEMA operations;

-- Schema para a equipe de análise de dados (BI, relatórios de performance).
CREATE SCHEMA analytics;

-- Schema para o departamento financeiro (análise de pagamentos, faturamento).
CREATE SCHEMA finance;

-- Schema para a alta gestão (dashboards de performance geral).
CREATE SCHEMA management;
```

Passo 2: Migrando os Objetos para o data_mart

Agora, vamos mover todas as tabelas, views e sequências existentes do schema public para o nosso novo schema data_mart, que servirá como a camada de armazenamento físico. Usaremos um bloco DO em PL/pgSQL para automatizar essa tarefa.

- Pesquisar como executar scripts PL/pgSQL (seja via **pgadmin** ou **psql**).

```
-- Este script move todos os objetos do 'public' para o 'data_mart'
-- É uma forma poderosa e automatizada de refatorar o banco de dados.
DO $$
DECLARE
    row record;
BEGIN
    -- Mover todas as tabelas
    FOR row IN SELECT tablename FROM pg_tables WHERE schemaname = 'public' LOOP
        EXECUTE 'ALTER TABLE public.' || quote_ident(row.tablename) || ' SET SCHEMA
data_mart;';
    END LOOP;

    -- Mover todas as views
    FOR row IN SELECT viewname FROM pg_views WHERE schemaname = 'public' LOOP
        EXECUTE 'ALTER VIEW public.' || quote_ident(row.viewname) || ' SET SCHEMA data_mart;';
    END LOOP;

    -- Mover todas as sequências
    FOR row IN SELECT sequencename FROM pg_sequences WHERE schemaname = 'public' LOOP
        EXECUTE 'ALTER SEQUENCE public.' || quote_ident(row.sequencename) || ' SET SCHEMA
data_mart;';
    END LOOP;
END;
$$;
```

Passo 3: Criando a Hierarquia de Roles (Modelo de 3 Camadas)

Aqui implementaremos nossa estrutura de roles, separando identidades, funções de negócio e

permissões granulares.

```
-- ===== CAMADA 2: ROLES DE ACESSO (PERMISSÕES GRANULARES) =====
-- Estes roles NUNCA fazem login. Eles apenas detêm privilégios.
CREATE ROLE dm_read_access NOLOGIN; -- Acesso de leitura a todas as tabelas base
CREATE ROLE dm_sequence_usage NOLOGIN; -- Acesso para usar sequências (para INSERTs)
CREATE ROLE ops_rental_write_access NOLOGIN; -- Permissão para registrar aluguéis e pagamentos
CREATE ROLE ops_customer_write_access NOLOGIN; -- Permissão para gerenciar clientes
CREATE ROLE fin_read_access NOLOGIN; -- Acesso de leitura a views financeiras
CREATE ROLE anl_read_access NOLOGIN; -- Acesso de leitura a views de análise
CREATE ROLE mgmt_read_access NOLOGIN; -- Acesso de leitura a dashboards de gestão

-- ===== CAMADA 3: ROLES FUNCIONAIS (CARGOS DA EMPRESA) =====
-- Estes roles também NUNCA fazem login. Eles representam os cargos.
CREATE ROLE job_cashier NOLOGIN;       -- O funcionário do caixa
CREATE ROLE job_store_manager NOLOGIN; -- O gerente da loja
CREATE ROLE job_data_analyst NOLOGIN;  -- O analista de BI
CREATE ROLE job_finance_director NOLOGIN; -- O diretor financeiro
CREATE ROLE job_owner NOLOGIN;         -- O dono da empresa
CREATE ROLE service_accounts NOLOGIN;  -- Para serviços automatizados (aplicações)

-- ===== CAMADA 1: ROLES DE IDENTIDADE (USUÁRIOS E SERVIÇOS) =====
-- Estes são os únicos roles com permissão de LOGIN.
CREATE ROLE pedro_caixa WITH LOGIN PASSWORD 'senha_forte_para_pedro';
CREATE ROLE maria_gerente WITH LOGIN PASSWORD 'senha_forte_para_maria';
CREATE ROLE ana_analista WITH LOGIN PASSWORD 'senha_forte_para_ana';
CREATE ROLE carlos_diretor WITH LOGIN PASSWORD 'senha_forte_para_carlos';
CREATE ROLE helena_dona WITH LOGIN PASSWORD 'senha_forte_para_helena';
CREATE ROLE app_service WITH LOGIN PASSWORD 'senha_muito_longa_e_complexa_para_app';
```

Passo 4: Conectando a Hierarquia (Concedendo Filiação)

Agora, conectamos as camadas, dando aos cargos (Camada 3) as permissões (Camada 2), e aos funcionários (Camada 1) os seus cargos (Camada 3).

```
-- Dando permissões aos cargos
GRANT dm_read_access TO job_cashier, job_store_manager, job_data_analyst,
job_finance_director, job_owner, service_accounts;
GRANT dm_sequence_usage TO job_cashier, job_store_manager, service_accounts;
GRANT ops_rental_write_access TO job_cashier, job_store_manager;
GRANT ops_customer_write_access TO job_store_manager; -- Apenas gerentes podem criar/editar
clientes
GRANT fin_read_access TO job_finance_director, job_owner;
GRANT anl_read_access TO job_data_analyst, job_store_manager, job_owner;
GRANT mgmt_read_access TO job_store_manager, job_owner;

-- Atribuindo os cargos aos funcionários
GRANT job_cashier TO pedro_caixa;
GRANT job_store_manager TO maria_gerente;
GRANT job_data_analyst TO ana_analista;
GRANT job_finance_director TO carlos_diretor;
GRANT job_owner TO helena_dona;
GRANT service_accounts TO app_service;
```

Passo 5: Criando a Camada de Abstração Segura (Views)

Agora, o passo mais importante: criar as "janelas" seguras em cada schema funcional. Os usuários interagirão com estas views, não com as tabelas do data_mart.

```
-- VIEW para OPERAÇÕES (registrar um novo aluguel)
-- Esta view é simplificada. Na prática, poderia chamar uma função para garantir a lógica de
negócio.
CREATE VIEW operations.new_rental AS
SELECT film_id, customer_id, store_id FROM data_mart.rental; -- Exemplo simplificado

-- VIEW para ANÁLISE (usando a view original que foi movida)
CREATE VIEW analytics.film_performance_summary AS
SELECT * FROM data_mart.sales_by_film_category;

-- VIEW para FINANÇAS (escondendo dados sensíveis)
CREATE VIEW finance.daily_revenue_summary AS
SELECT payment_date::date AS day, sum(amount) AS total_revenue
FROM data_mart.payment
GROUP BY day
ORDER BY day DESC;

-- VIEW para GESTÃO (usando outra view original)
CREATE VIEW management.store_performance AS
```

```
SELECT * FROM data_mart.sales_by_store;
```

Passo 6: Concedendo as Permissões Finais nos Objetos

Com a estrutura montada, fazemos as concessões finais. Note que os GRANTs são feitos para os roles de permissão (Camada 2).

```
-- Permissão de USAGE nos schemas para os roles que precisam "olhar dentro"
GRANT USAGE ON SCHEMA data_mart, operations, analytics, finance, management
TO dm_read_access; -- O role de leitura base precisa ver todos os schemas para as views funcionarem

-- Permissão nas tabelas base do data_mart
GRANT SELECT ON ALL TABLES IN SCHEMA data_mart TO dm_read_access;
GRANT USAGE ON ALL SEQUENCES IN SCHEMA data_mart TO dm_sequence_usage;

-- Permissão de escrita para operações
GRANT INSERT, UPDATE ON operations.new_rental TO ops_rental_write_access;
-- (Permissões de escrita seriam mais complexas na realidade, possivelmente em funções)

-- Permissões de leitura nas views
GRANT SELECT ON ALL TABLES IN SCHEMA analytics TO anl_read_access;
GRANT SELECT ON ALL TABLES IN SCHEMA finance TO fin_read_access;
GRANT SELECT ON ALL TABLES IN SCHEMA management TO mgmt_read_access;
```

Conclusão e Verificação

Vamos testar alguns aspectos da arquitetura que acabamos de construir.

- Pedro, o caixa (`pedro_caixa`), pode se conectar. Ele herda as permissões do job_cashier. Ele pode fazer `SELECT` nas tabelas base (via `dm_read_access`) para procurar um filme e pode usar a view `operations.new_rental` para registrar um aluguel (via `ops_rental_write_access`). Ele não pode ver a view `finance.daily_revenue_summary`.

  ```
  SELECT title FROM data_mart.film WHERE film_id = 10; --> SUCESSO

  INSERT INTO operations.new_rental (...) VALUES (...); --> SUCESSO

  SELECT * FROM finance.daily_revenue_summary; --> ERRO: permission denied for schema finance
  ```

- Ana, a analista (`ana_analista`), pode se conectar. Ela herda as permissões de `job_data_analyst`. Ela pode ler as tabelas base e também consultar a view `analytics.film_performance_summary`. Ela não pode registrar um aluguel.

  ```
  SELECT * FROM analytics.film_performance_summary; --> SUCESSO

  INSERT INTO operations.new_rental (...) VALUES (...); --> ERRO: permission denied for view new_rental
  ```

Helena, a dona (`helena_dona`), tem acesso a tudo, pois seu cargo `job_owner` recebeu filiação a todos os grupos de leitura relevantes.

Entregáveis

1. Prints de tela das checagens acima
2. Saída da execução do script sql/ownership_query.sql