NICOLAS VINICIUS NOGUEIRA ALVES

COVID 19 dados anonimizados de casos confirmados

Três Lagoas-MS

2023

RELATÓRIO TÉCNICO: COVID 19 dados anonimizados de casos confirmados

1 Introdução: descrição do conjunto de dados escolhido;

Este conjunto de dados apresentar a relação de casos confirmados de COVID-19 no âmbito do Estado de Santa Catarina, conforme as recomendações da Open Knowlegdde Foundation – Brasil (OKBR).

1.1 DICIONÁRIO DE DADOS

- data_publicacao: data de publicação do conjunto de dados no portal de dados abertos
- recuperados: indicação de que o paciente foi recuperado
- data_inicio_sintomas: data do início dos sintomas
- data_coleta: data da coleta da amostra
- sintomas: sintomas do paciente
- comorbidades: comorbidades do paciente
- gestante: indica os casos de gestantes ou puérpera
- internacao: indicação de que o paciente está internado
- internacao_uti: indicação de quee o paciente está internados em UTI
- sexo: indicação de sexo biológico do paciente
- municipio: município de residência do paciente
- obito: indicação de que o paciente veio a óbito
- data_obito: data do óbito do paciente
- idade: idade do paciente
- regional: mesorregião de residência do paciente
- raca: raça do paciente
- data_resultado: data e hora da confirmação
- codigo_ibge_municipio: codigo do IBGE do município de residência do paciente
- latitude: latitude do município de residência do paciente
- longitude: longitude do município de residência do paciente
- estado: nome do estado de residência do paciente

- criterio_ confirmacao: critério utilizado para confirmação do caso
- tipo_teste: tipo de teste utilizado para confirmação do caso
- municipio_notificacao: município onde a notificação foi registrada
- codigo_ibge_municipio_notificacao: codigo do IBGE do município onde a notificação foi registrada
- latitude_notificacao: latitude do município onde foi registrada a coordenada geográfica em fracções decimais
- longitude_notificacao: longitude do município onde foi registrada a coordenada geográfica em frações decimais
- classificacao: classificação de confirmação de caso positivo
- origem_esus: indica que a origem da informação se encontra no e-SUS VE
- origem_sivep: indica que a origem da informação se encontra no SIVEP Gripe
- origem_lacen: indica que a origem da informação se encontra no LACEN/SC
- origem_laboratorio_privado: indica que a origem da informação se encontra em um laboratório privado
- nom_laboratorio: quando campo origem_laboratorio_privado for preenchido, neste campo constará o nome do laboratório
- fez_teste_rapido: indicativo se o paciente (sem informação do paciente) fez teste rápido
- fez_pcr: indicativo se o paciente (sem informação do paciente_) fez PCR
- data_internacao: data da internação informada no SIVEP Gripe
- data_entrada_uti: data da internação UTI informada no SIVEP Gripe
- regional_saude: região onde teve evolução do caso informada no SIVEP Gripe
- data_evolucao_caso: data da internação informada no SIVEP Gripe
- data_saida_uti: Regional de saúde do município de notificação
- bairro: bairro do paciente

1.2 O objetivo deste relatório técnico é criar uma função (stored procedure) que:

- Faça a importação dos dados do arquivo csv para uma tabela física do BD, criada dentro da procedure;
- Normalize os dados: criação das tabelas e inserção/atualização dos dados;
- Criptografar os dados sensíveis;
- Criar uma visão de banco de dados, que denormalize e descriptografe os dados;
- Retorne corretamente uma tabela com os dados da visão

2 Desenvolvimento da Stored Procedure

2.1 Importação dos Dados do CSV para uma Tabela Física

Para a etapa de importação dos dados do arquivo CSV para uma tabela física no banco de dados PostgreSQL, foi criada uma tabela denominada dados_pacientes. Esta tabela foi criada para ser fiel estrutura do arquivo CSV, onde cada coluna representa um atributo específico dos dados sobre os pacientes de COVID-19 no Estado de Santa Catarina.

A estrutura da tabela dados_pacientes é igual as colunas do arquivo CSV, com exceção de uma coluna adicional, ID, que foi adicionada para servir como chave primária. A ausência de uma coluna essencial no arquivo CSV para usar como chave primária levou à necessidade de criar essa coluna.

A importação foi realizada utilizando o comando COPY, uma funcionalidade eficiente do PostgreSQL para a carga em massa de dados. O delimitador utilizado foi ';', e a opção CSV HEADER indicou que a primeira linha do arquivo CSV contém os cabeçalhos das colunas, facilitando a correspondência entre os campos no arquivo e na tabela.

2.2 Exemplo de código SQL utilizado para a importação:

Criação da tabela para fazer a importação

```
CREATE TABLE dados_pacientes(
ID serial PRIMARY KEY,
Data_publicacao TIMESTAMP,
.....
);
```

Importação dos dados do CSV

```
COPY dados_pacientes (data_publicacao, recuperados, data_inicio_sintomas, .....) FROM '/dados/script_covid.sql' DELIMITER ';' CSV HEADER;
```

Está prática assegura que a estrutura da tabela permaneça em conformidade com os dados do arquivo CSV, fornecendo uma base solida para as fases seguinte do processo de normalização.

3 Normalização dos Dados

3.1 Organização das Tabelas Normalizadas e Relações

Para otimizar a estrutura do banco de dados e garantir a integridade referencial, os dados foram normalizados em diversas tabelas, cada uma com um propósito específico. As tabelas normalizadas e suas relações são as seguintes:

1. **Tabela pacientes:**

Contém informações pessoais sobre os pacientes, como data_inicio_sintomas, sintomas, comorbidades, gestante, sexo, idade, raca, e bairro.
Possui uma chave primária (ID) para identificação única de cada paciente.

2. **Tabela localizacao:**

Armazena dados relacionados à localização geográfica dos pacientes, como codigo_ibge, nome_municipio, latitude, longitude, e estado.

3. **Tabela resultados_teste:**

Registra resultados de testes COVID-19, como data_coleta, data_resultado, criterio_confirmacao, tipo_teste, fez_teste_rapido, e fez_pcr.

4. **Tabela notificacoes:**

Armazena dados relacionados a notificações sobre casos de COVID-19, como municipio_notificacao, latitude_notificacao, longitude_notificacao, classificacao, origem_esus, origem_sivep, origem_lacen, origem_laboratorio_privado, e nom_laboratorio.

5. **Tabela evolucao_caso:**

Registra informações sobre a evolução do caso, como data_internacao, data_entrada_uti, regional_evolucao_caso, e data_saida_uti.

6. **Tabela recuperados:**

Contém dados sobre pacientes que se recuperaram

Inclui apenas uma tabela chamada recuperado.

7. **Tabela obito:**

Registra informações sobre óbitos relacionados à COVID-19, como obito e data_obito.

As tabelas localizacao, resultados_teste, notificacoes, evolucao_caso, recuperados, e obitos estão normalizadas e possuem uma relação fundamental com a tabela pacientes. Essa relação é estabelecida por meio da utilização da coluna paciente_id, que atua como uma chave estrangeira nessas tabelas, referenciando diretamente a coluna **ID** da tabela **pacientes**.

Dessa forma, a integridade referencial é assegurada, possibilitando uma ligação eficiente entre os detalhes específicos presentes em cada tabela e os registros associados na tabela principal de pacientes. Essa abordagem contribui para a criação de uma estrutura de banco de dados otimizada, simplificando a busca e análise de dados relacionados ao casos de COVID-19.

3.2 Exemplo de código SQL utilizado na normalização:

```
CREATE TABLE recuperados
(
   id SERIAL PRIMARY KEY,
   paciente_id INT REFERENCES pacientes(id) NOT NULL,
   recuperado VARCHAR(3)
);
INSERT INTO recuperados (paciente_id, recuperado)
SELECT p.id, dp.recuperados FROM dados_paciente dp
JOIN pacientes p ON p.id = dp.id;
```

4 Criptografia dos Dados Sensíveis

4.1 identificação dos Dados Sensíveis:

O procedimento foi realizado utilizando a extensão pgcrypto no PostgreSQL e focou na tabela "paciente", selecionando cuidadosamente os campos sensíveis: sintomas, sexo, idade, raça e bairro, contêm dados pessoais que requerem proteção devido à sua natureza confidencial e sensível.

4.2 Descrição do Método de Criptografia Utilizado:

Criação de novas colunas para armazenar os dados criptografados, nomeadamente: sintomas_criptografado, sexo_criptografado, idade_criptografado, raca_criptografado e bairro_criptografado. Essas colunas foram definidas como do tipo BYTEA para permitir o armazenamento eficiente de dados binários, uma escolha adequada para informações criptografadas.

O processo de criptografia foi efetuado mediante a utilização da função pgp_sym_encrypt, garantindo assim uma criptografia dos dados. Chave de acesso '1567' para descriptografar os dados.

4.3 Exemplo de Código SQL para Criptografia:

```
-- ADICIONANDO extensao pgcrypto

CREATE EXTENSION IF NOT EXISTS pgcrypto;

-- ALTERANDO O NOME DAS COLUNAS QUE VAI SER CRIPTOGRAFADA
ALTER TABLE pacientes ADD COLUMN sintomas_criptografado BYTEA;
ALTER TABLE pacientes ADD COLUMN sexo_criptografado BYTEA;
-- CRIPTOGRAFANDO AS COLUNAS
UPDATE pacientes SET sintomas_criptografado pgp_sym_encrypt(p.sintomas, '1567');
UPDATE pacientes SET sexo_criptografado = pgp_sym_encrypt(p.sexo, '1567');
-- Removendo as colunas que foram substituidas
ALTER TABLE pacientes DROP COLUMN sintomas;
ALTER TABLE pacientes DROP COLUMN sexo;
```

5 Criação da View para Denormalização e Descriptografia:

Apos concluir a etapa de criptografação dos Dados sensiveis, foi criada uma view que denormaliza e descriptografa os dados sensíveis da tabela ‘’paciente’’. As colunas que foram criptografadas agora são descriptogradas durante a seleção de dados, facilitando a consulta e proporcionando uma visão mais comprrensível para consultas.

5.1 Exemplo de Código da criação da VIEW:

```
-- Criando a View para Denormalização e Descriptografia
CREATE OR REPLACE VIEW dados_view AS
SELECT
    p.id AS paciente_id,
    p.data_inicio_sintomas,
    pgp_sym_decrypt(p.sintomas_criptografado,'1567')::VARCHARAS
sintomas_descriptografado,
    p.comorbidades,
    p.gestante,
    pgp_sym_decrypt(p.sexo_criptografado,'1567')::VARCHAR AS
sexo_descriptografado,
    pgp_sym_decrypt(p.idade_criptografado, '1567')::VARCHAR AS
idade_descriptografado,
    pgp_sym_decrypt(p.raca_criptografado, '1567')::VARCHAR AS
raca_descriptografado,
    pgp_sym_decrypt(p.bairro_criptografado, '1567')::VARCHAR AS
bairro_descriptografado,
    .....
```

Nesse exemplo a VIEW foi criada e chamada “dados_view”, as colunas sintomas_criptografado, sexo_criptografado, idade_criptografado, raca_criptofado e bairro criptografado da tabela “pacientes” são descriptografada utilizando a função “pgp_sym_decrypt”.

5.2 Retornando a tabela por meio da VIEW:

Executando o comando RETURN QUERY SELECT * FROM dados_view; obtemos todas as tabelas denormalizadas e descriptografada dos dados sensíveis, assim fornecendo uma visão compreensível das informações anteriormente criptografadas e normalizadas.

A criação desta view representa uma etapa crucial para a apresentação simplificada dos dados sensíveis, garantindo que a descriptografia e a denormalização sejam realizadas de forma eficiente e segura.

6 Criação da função STORED PROCEDURE

Apos conclusão da criaçaõ da VIEW, foi criado a função STORE PROCEDURE que ela executa todo o script, que inclui a manipulação de tabelas, inserção de dados, criptografia dos dados e criação e retorno as tabelas da view denormalizadas e descriptografadas.

6.1 Exemplo do codigo

```
CREATE OR REPLACE FUNCTION funcao_dados()
RETURNS TABLE
(
    id INT,
    data_inicio_sintomas DATE,
    sintomas VARCHAR,
    comorbidades VARCHAR(100),
    gestante VARCHAR(100),
    sexo VARCHAR,
....);
AS $$
 BEGIN
RETURN QUERY SELECT * FROM dados_view;

END $$ LANGUAGE plpgs;
```