Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Curso de Bacharelado em Sistemas de Informação

Projeto Final Base de Dados Covid

Pedro Fernando Christofoletti dos Santos
Vinícius Gonçalves de Carvalho

Relatório apresentado na disciplina SCC541- Laboratório de Bases de Dados, como requisito parcial para obtenção da aprovação na mesma.

São Carlos/SP, 21 de Julho de 2021

SUMÁRIO

Abstract: *This report will address and discuss the solutions found for each of the statements of the Final Project of the Database Laboratory discipline, where the object of analysis was a database created from data available from FAPESP COVID-19 DataSharing/BR . The project aims to analyze the student's abilities as a database analyst. Furthermore, the DBMS used was PostgreSQL.*
Keywords: Database, FAPESP COVID-19, Final Project, PostgreSQL.

Resumo: *Esse relatório irá abordar e discutir as soluções encontradas para cada um dos enunciados do Projeto Final da disciplina de Laboratório de Bases de Dados, onde o objeto de análise foi uma base de dados criada a partir de dados disponibilizados da FAPESP COVID-19 DataSharing/BR. O projeto visa analisar as capacidades do aluno como analista de bases de dados. Ademais, o SGBD utilizado foi o PostgreSQL.*
Palavras-chave: Bases de Dados, FAPESP COVID-19, Projeto Final, PostgreSQL.

1. Atividade 1

1.1. Enunciado

Para otimizar a performance de consultas frequentes executadas por especialistas, foi pedido ao analista do servidor do banco de dados que realize uma análise sobre a necessidade da aplicação de índices na base de dados Covid-Fapesp. Com isso, o analista identificou a seguinte consulta:

```
SELECT * FROM exames ex
JOIN pacientes p ON p . id_paciente = ex . id_paciente
JOIN desfechos d ON p . id_paciente = d . id_paciente
WHERE ex.de_origem = 'Unidades de Internação'
```

Além desta consulta, o grupo de especialistas requisitou ao analista a identificação de uma consulta mais recorrente que necessita de índices. Com isso, **deve ser proposta uma consulta frequente, que recupere tuplas suficientes para analisar e explorar a aplicação de índices, justificando a semântica e relevância da consulta**.

Requisitos. Para ambas as consultas, devem ser respondidos os seguintes pontos:

- Mostrar a informação que a consulta recupera (20 primeiras linhas);
- Colher o tempo de execução sem a utilização de índices;
- Implementar índices efetivos para cada consulta, ou seja, que alterem o plano de execução em relação à consulta sem índice;
- Colher o tempo de execução com os índices criados;
- Justificar a implementação do índice de acordo com a consulta, explicitando o porquê do índice otimizar tal consulta, conforme visto durante as aulas;
- É obrigatório o uso de ferramentas disponibilizadas pelo SGBDR para a análise da consulta.

1.2. Solução e Justificativa

Do enunciado podemos extrair algumas tarefas a serem realizadas e é possível dividi-las em dois grupos:

A. Recuperar informações da consulta identificada pelo analista (20 primeiras linhas) e colher o seu tempo de execução sem e posteriormente com o uso de índices
B. Identificar uma nova consulta frequente, recuperar suas informações (20 primeiras linhas) e colher o seu tempo de execução sem e posteriormente com o uso de índices

Para resolver o proposto na **letra A** foi adicionado um limite na consulta já identificada pelo analista, para que retorne apenas as 20 primeiras linhas:

```
SELECT * FROM exames ex
JOIN pacientes p ON p . id_paciente = ex . id_paciente
JOIN desfechos d ON p . id_paciente = d . id_paciente
WHERE ex.de_origem = 'Unidades de Internação'
LIMIT 20;
```

Após realizar esta consulta, mostrando as 20 primeiras tuplas do resultado obtido, foi necessário utilizar a ferramenta EXPLAIN ANALYZE fornecida pelo SGBD para que pudéssemos coletar o tempo de execução desta consulta. A consulta com essa ferramenta ficou assim:

```
EXPLAIN ANALYZE
      SELECT * FROM exames ex
      JOIN pacientes p ON p.id_paciente = ex.id_paciente
      JOIN desfechos d ON p.id_paciente = d.id_paciente
      WHERE ex.de_origem = 'Unidades de Internação';
```

O tempo de execução coletado para esta consulta foi o de **2066,401 ms**.

Com os resultados obtidos e com o tempo de execução coletado então partiu-se para a criação de um índice pertinente e posterior execução e coleta de tempo de execução. O índice criado para esta consulta foi:

```
CREATE INDEX origExUnidIntern ON exames(id_exame)
      WHERE de_origem = 'Unidades de Internação';
```

Como a tabela de exames possui um número elevado de tuplas, cerca de 4 milhões, foi identificado que o melhor lugar para se criar um índice e evitar o custo de uma busca sequencial ineficaz seria nesta tabela. Ademais, foi identificado que seria interessante utilizar o mesmo filtro utilizado na consulta frequente trazida pelo analista, justamente por ser parte de uma consulta frequente ao banco de dados, ou seja, filtrar os exames que tivessem origem em ‘Unidades de Internação’.

Após a criação do índice, o tempo de execução foi novamente coletado e apresentou uma pequena melhora de cerca de 26% com relação ao tempo coletado sem a utilização do índice, caindo para o valor de **1525,006 ms**.

Com isto, a letra A está finalizada e partiu-se, então, para a **letra B**. Inicialmente foi necessário se identificar uma outra consulta frequente e então realizar as mesmas coletas feitas na letra A.

Pensando no ponto de vista mais anaítico e de consultas que serão frequentes e que poderão fer a fonte de informação valiosa no combate à Covid-19, a seguinte consulta foi identificada:

```
SELECT (EXTRACT(YEAR FROM e.dt_coleta) - p.aa_nascimento) AS idade,
COUNT(DISTINCT e.id_paciente) AS casos_positivos
      FROM pacientes p, exames e
WHERE p.id_paciente = e.id_paciente
      AND upper(e.de_exame) LIKE '%COVID%'
      AND (upper(e.de_resultado) LIKE '%POSITIVO%'
                  OR upper(e.de_resultado) LIKE 'DETECTADO%'
                  OR upper(e.de_resultado) LIKE 'DETECTADOS ANTICORPOS%'
                  OR upper(e.de_resultado) LIKE 'REAGENTE%'
                  OR upper(e.de_resultado) LIKE 'AMOSTRA REAGENTE%')
      AND upper(e.de_resultado) NOT LIKE '%A DINÂMICA DE PRODUÇÃO DE
ANTICORPOS NA COVID-19 AINDA NÃO É BEM ESTABELECIDA%'
      AND to_char(e.dt_coleta, 'YYYY-MM') = '2020-12'
GROUP BY idade
ORDER BY casos_positivos DESC
LIMIT 20;
```

A consulta acima busca contar todos os pacientes que testaram positivo para COVID em um determinado mês e mostra estes números agrupados por idade. Desta forma seria possível identificar quais são as faixas etárias mais afetadas pela pandemia num dado mês e políticas públicas poderiam ser tomadas com base nesta informação. Na consulta mostrada acima apenas as 20 idades que apresentam mais casos positivos de COVID em Dezembro de 2020 são listadas.

Então para se coletar o tempo de execução desta consulta foi utilizada a mesma ferramenta do SGBD utilizada na letra A, EXPLAIN ANALYZE:

```
EXPLAIN ANALYZE
      SELECT (EXTRACT(YEAR FROM e.dt_coleta) - p.aa_nascimento) AS idade,
COUNT(DISTINCT e.id_paciente) AS casos_positivos
            FROM pacientes p, exames e
      WHERE p.id_paciente = e.id_paciente
            AND upper(e.de_exame) LIKE '%COVID%'
            AND (upper(e.de_resultado) LIKE '%POSITIVO%'
```

```
                        OR upper(e.de_resultado) LIKE 'DETECTADO%'
                        OR upper(e.de_resultado) LIKE 'DETECTADOS
ANTICORPOS%'
                        OR upper(e.de_resultado) LIKE 'REAGENTE%'
                        OR upper(e.de_resultado) LIKE 'AMOSTRA
REAGENTE%')
            AND upper(e.de_resultado) NOT LIKE '%A DINÂMICA DE PRODUÇÃO
DE ANTICORPOS NA COVID-19 AINDA NÃO É BEM ESTABELECIDA%'
            AND to_char(e.dt_coleta, 'YYYY-MM') = '2020-12'
      GROUP BY idade
      ORDER BY casos_positivos DESC;
```

A consulta acima retorna um tempo de execução igual a **3075,859 ms**. Novamente a tabela de exames foi utilizada e por isso o índice criado para esta nova consulta também está relacionado a esta tabela. Foi encontrado que dentre as quase 4 milhões de tuplas da tabela exames, apenas uma pequena parcela dela correspondia a testes de covid (cerca de 20 mil). Dado que haveria um grande desperdício de performance caso toda vez que essa consulta fosse realizada uma busca sequencial fosse feita em exames, o índice abaixo foi criado, para todos os exames de COVID encontrados na tabela de exames:

```
CREATE INDEX exameCovid ON exames(id_exame)
      WHERE upper(de_exame) LIKE '%COVID%';
```

Após a criação deste índice, o tempo de execução da nova consulta foi coletado novamente, caindo consideravelmente, com o valor de **1102,034 ms**. Este novo valor representou uma melhora de cerca de **65%** na performance.

Resultados em formato de tabela:

Consulta	Tempo
1 sem índice	2066,401 ms
1 com índice	1525,006 ms
2 sem índice	3075,859 ms
2 com índice	1102,034 ms

2. Atividade 2

2.1. Enunciado

Para gerenciar o acesso às informações do sistema da base de dados, os técnicos necessitam criar rotinas para garantir a segurança e manutenção. Você, enquanto técnico deste sistema, deve criar rotinas de segurança do sistema, conforme as seguintes especificações:

A. O sistema necessita armazenar os dados de acesso de cada paciente, incluindo o identificador de cada paciente já registrado, juntamente a uma senha (tratada adequadamente), a partir de um critério de criação de senhas (por exemplo, uma combinação de atributos). O sistema deverá automatizar o processo de cadastro de usuário e senha para novos pacientes que serão inseridos na base.
B. Para gerenciar os acessos e transações realizadas na base de dados, é necessário criar um log do sistema para registrar todas as operações possíveis. Com isso, é necessário criar rotinas que coletem e armazenem as informações em uma tabela "LogAcesso", de modo automático. A partir desta especificação, você (como técnico do sistema) deve analisar três situações pertinentes para realizar a coleta da informação automaticamente. Devem ser armazenadas pelo menos informações como data/hora, tipo de operação (por exemplo seleção, atualização, inserção entre outras), e tabela requisitada/alterada. Outras informações adicionais ficam a critério do técnico do sistema.

   Avaliação: as situações devem ser devidamente registradas e validadas por meios de testes, além de serem sucintamente explicadas, em relação a semântica da operação e relevância para o sistema em si.

2.2. Solução e Justificativa

Para solucionar o problema de segurança da **letra A** foi criada uma função chamada "*password_hash*" que recebe um parâmetro do tipo TEXT e retorna um TEXT com o valor do parâmetro criptografado utilizando duas funções do postgres a **crypt** e a **gen_salt**. Caso a função receba um valor inválido ela irá disparar uma exceção.

```
CREATE EXTENSION pgcrypto;
CREATE OR REPLACE FUNCTION password_hash(passwordString TEXT)
RETURNS TEXT
LANGUAGE plpgsql
AS $$
      DECLARE MessageText TEXT;
            HintText TEXT;
      BEGIN
           IF passwordString IS NULL OR passwordString = '' THEN
               RAISE EXCEPTION null_value_not_allowed USING MESSAGE =
'Senha nula', HINT = 'Insira uma senha para obter seu retorno
criptografado';
               RETURN NULL;
```

```
            ELSE -- Retornando senha criptografada
                RETURN crypt(passwordString, gen_salt('md5'));
            END IF;
            EXCEPTION -- Imprimindo exceções encontradas
                WHEN OTHERS THEN
                GET STACKED DIAGNOSTICS MessageText = MESSAGE_TEXT,
HintText = PG_EXCEPTION_HINT;
                RAISE NOTICE E'Erro: %\nMensagem: %\nDica: %',
                       SQLSTATE, MessageText, HintText;
      END;
$$
```

Agora iremos alterar a tabela pacientes para adicionar a coluna “senha”:

```
ALTER TABLE pacientes
    ADD COLUMN senha text;
```

Agora usaremos a função criada para atualizar todas as senhas da tabela pacientes:

```
UPDATE pacientes
      SET senha = password_hash(CONCAT(id_paciente, ic_sexo,
aa_nascimento));
```

Agora criamos a função que retorna a trigger para criar senhas automaticamente após o paciente ser inserido, usamos a concatenação dos campos id_paciente, ic_sexo e aa_nascimento, esses campos foram escolhidos pois podem ser rapidamente validados no momento de autenticação e possui um campo único que é o id_paciente:

```
CREATE OR REPLACE FUNCTION setPwdTriggerFunction()
      RETURNS TRIGGER
      LANGUAGE PLPGSQL
      AS $$
      BEGIN
            NEW.senha = password_hash(CONCAT(NEW.id_paciente,
NEW.ic_sexo, NEW.aa_nascimento));
            RETURN NEW;
      END;
      $$
```

Por fim, criamos a trigger para a tabela pacientes, para que assim que um paciente novo for inserido, automaticamente será gerada uma senha para ele usando da função criada anteriormente:

```
CREATE TRIGGER setPwd
```

```
    BEFORE INSERT ON pacientes
    FOR EACH ROW
      EXECUTE PROCEDURE setPwdTriggerFunction();
```

Para atender a necessidade de gerenciar os acessos e transações na base de dados solicitado na **letra B** foi criada a tabela LogAcesso que possui um id com auto incremento para facilitar a inserção, o campo autor que armazenará qual usuário do banco de dados manipulou a base, o campo carimbo_de_tempo que guarda a data e hora do momento que a alteração foi realizada, o campo operação que guarda a operação realizada (INSERT, UPDATE e DELETE) e por fim o campo tabela que guarda a informação de qual tabela foi feita a alteração.

Essas informações já são o suficiente para gerenciar e monitorar a manipulação do banco pois sabemos quem fez a alteração, a data e hora que a alteração foi feita, qual tabela foi almejada e qual operação foi realizada nela. Além disso, o próprio SGBD fornece as informações necessárias para preencher a tabela no momento de log. Todos os campos são relevantes, portanto, nenhum deles podem ser nulos. Aqui temos o script para criar a tabela LogAcesso:

```
CREATE TABLE LogAcesso
(
    id bigserial NOT NULL,
    autor char(50) NOT NULL,
    carimbo_de_tempo timestamp NOT NULL,
    operacao char(8) NOT NULL,
    tabela char(30) NOT NULL,
    PRIMARY KEY (id)
);
```

Agora criaremos a função que retorna trigger para inserir os logs na tabela Log Acesso. Essa função utiliza os objetos e funções do próprio SGBD para criar uma nova tupla na tabela Log Acesso. O objeto USER nesse contexto é o usuário do banco de dados dessa conexão atual, a função NOW() retorna a data e hora no formato *"timestamp"*, o objeto TG_OP é a operação que disparou a trigger, e por fim, o objeto TG_TABLE_NAME é o nome da tabela para qual a trigger foi atribuída.

```
CREATE OR REPLACE FUNCTION newLogTriggerFunction()
      RETURNS TRIGGER
      LANGUAGE PLPGSQL
      AS $$
      BEGIN
            INSERT INTO LogAcesso(
                  autor, carimbo_de_tempo, operacao, tabela)
                  VALUES (USER, NOW(), TG_OP, TG_TABLE_NAME);
            RETURN NEW;
      END;
```

```
    $$
```

Criada a função, agora só precisamos criar três triggers, uma para cada tabela, e associar a execução da função. Veja o script a seguir:

```
CREATE TRIGGER LogPacientes
    AFTER INSERT OR UPDATE OR DELETE ON pacientes
    FOR EACH ROW
    EXECUTE PROCEDURE newLogTriggerFunction();
CREATE TRIGGER LogExames
    AFTER INSERT OR UPDATE OR DELETE ON exames
    FOR EACH ROW
    EXECUTE PROCEDURE newLogTriggerFunction();
CREATE TRIGGER LogDesfechos
    AFTER INSERT OR UPDATE OR DELETE ON desfechos
    FOR EACH ROW
    EXECUTE PROCEDURE newLogTriggerFunction();
```

Consideramos relevante para a tabela log as operações de Inserção, Atualização e Deleção nas tabelas pacientes, exames e desfechos pois são operações que alteram o conteúdo da base e podem um dia necessitar de monitoramento e/ou checagem das últimas atividades.

3. Atividade 3

3.1. Enunciado

Para demonstrar a viabilidade de expandir a coleta de registro em hospitais, um analista decidiu procurar por análises relevantes existentes na base de dados atual. Desse modo, buscou-se responder algumas perguntas antes de realizar uma análise mais detalhada sobre os dados:

A. Que tipo de informações relacionadas a COVID é possível recuperar analisando os tipos de exames e analitos registrados?
B. Analisando a quantidade de registros de um determinado exame de COVID em relação a data de coleta ou período (por exemplo semanal), é possível indicar tendências de alta e/ou baixa que auxiliam os especialistas médicos em análises futuras? (Dados de apoio: https://covid.saude.gov.br/)
C. Que tipo de informações adicionais, em versões futuras da base de dados, poderiam ser coletadas em novos hospitais para melhorar a qualidade dos dados analisados? Esta questão deve ser baseada nas justificativas das questões anteriores em relação às informações existentes na base.

Avaliação: Para as questões acima, é necessário realizar uma análise exploratória nos dados, selecionando informações relevantes com base em algum critério pertinente, justificando sua relevância no auxílio a especialistas. Por exemplo:

- *"Foi encontrado, em um determinado período de 15 dias, grandes quantidades de exames de PCR no qual a maioria dos pacientes veio a óbito em seguida";*
- *"Grandes quantidades de determinado exame e analito são muito recorrentes, que segundo estudos [1][2] apontam uma tendência para casos graves de COVID".*

As discussões levantadas podem ser verificadas e justificadas a partir de estudos oficiais relacionados a COVID, como apoio.

3.2. Solução e Justificativa

Para responder a pergunta da **letra A** sobre quais informações sobre COVID podemos tirar utilizando os exames e analitos decidimos primeiro avaliar a quantidade de cada um dos tipos de exames de covid realizados:

```
SELECT de_exame, COUNT(de_exame) AS quantidade FROM exames
 WHERE upper(de_exame) LIKE '%COVID%' GROUP BY de_exame;
```

Obtivemos o seguinte resultado:

Exame	Quantidade
covid-19-pcr para sars-cov-2, vários materiais (fleury)	12160
covid-19-sorologia igm e igg por quimiluminescência, soro	5467
covid-19-teste rápido (igm e igg), soro	398
covid-19 - pesquisa de anticorpos igg	1518
covid-19, anticorpos iga e igg, soro	30
covid teste líquor	11
teste rápido para covid-19 de ponta de dedo	1
teste rápido para sars-cov-2- pesquisa de anticorpos igg e igm (sorologia para covid-19)	291

Após analisar os resultados nos perguntamos se seria possível identificar os tipos de exames que mais apresentam **falsos negativos**, pois dessa forma, as autoridades governamentais poderiam reavaliar o uso desse exame e/ou aprimorá-lo. Então optamos por avaliar os dois tipos de exame com maior número de registros, o ***covid-19-pcr*** e o ***covid-19-sorologia***, extraindo a porcentagem de casos positivos. Primeiro coletamos quantos casos positivos foram encontrados para cada exame:

```
SELECT de_exame, COUNT(DISTINCT(id_paciente)) AS casos_positivos FROM
exames
            WHERE de_exame = 'covid-19-pcr para sars-cov-2, vários
```

```
materiais (fleury)' OR de_exame = 'covid-19-sorologia igm e igg por
quimiluminescência, soro'
            AND (upper(de_resultado) LIKE '%POSITIVO%'
                        OR upper(de_resultado) LIKE 'DETECTADO%'
                        OR upper(de_resultado) LIKE 'DETECTADOS
ANTICORPOS%'
                        OR upper(de_resultado) LIKE 'REAGENTE%'
                        OR upper(de_resultado) LIKE 'AMOSTRA REAGENTE%')
            AND upper(de_resultado) NOT LIKE '%A DINÂMICA DE PRODUÇÃO DE
ANTICORPOS NA COVID-19 AINDA NÃO É BEM ESTABELECIDA%'
      GROUP BY de_exame;
```

Obtivemos o seguinte resultado:

Exame	Casos positivos
covid-19-pcr para sars-cov-2, vários materiais (fleury)	4583
covid-19-sorologia igm e igg por quimiluminescência, soro	373

Agora só precisamos calcular a porcentagem EXAMES REALIZADOS **X** CASOS POSITIVOS.

PRC: $4583 \div 12160 \simeq 0,376$

Sorologia: $373 \div 5467 \simeq 0,068$

Com esses dados, constatamos que o exame de PCR acusou positivo para covid em **37,6%** dos casos e o exame de sorologia apenas em **6,8%**.Visto isso, podemos nos atentar a duas possibilidades, o exame de sorologia está sujeito a apresentar muitos falsos negativos ou os pacientes realizaram o exame sem respeitar o tempo mínimo de 7 dias após os sintomas para realizá-lo, pois esse exame não detecta o vírus e sim a presença de anticorpos. Portanto, as autoridades governamentais podem reavaliar o uso do exame de sorologia e melhorar sua logística de aplicação para trazer mais consistência para o panorama geral do vírus e também economizar tempo e dinheiro ao aplicar o teste somente quando atenderem aos requisitos.

Para responder a pergunta da **letra B** sobre a possibilidade de indicar indicar tendências de alta e/ou baixa que auxiliam os especialistas médicos em análises futuras ao analisar a quantidade de registros de um determinado exame de COVID em relação a data de coleta ou período, decidimos avaliar a quantidade de casos positivos coletados em cada um dos meses:

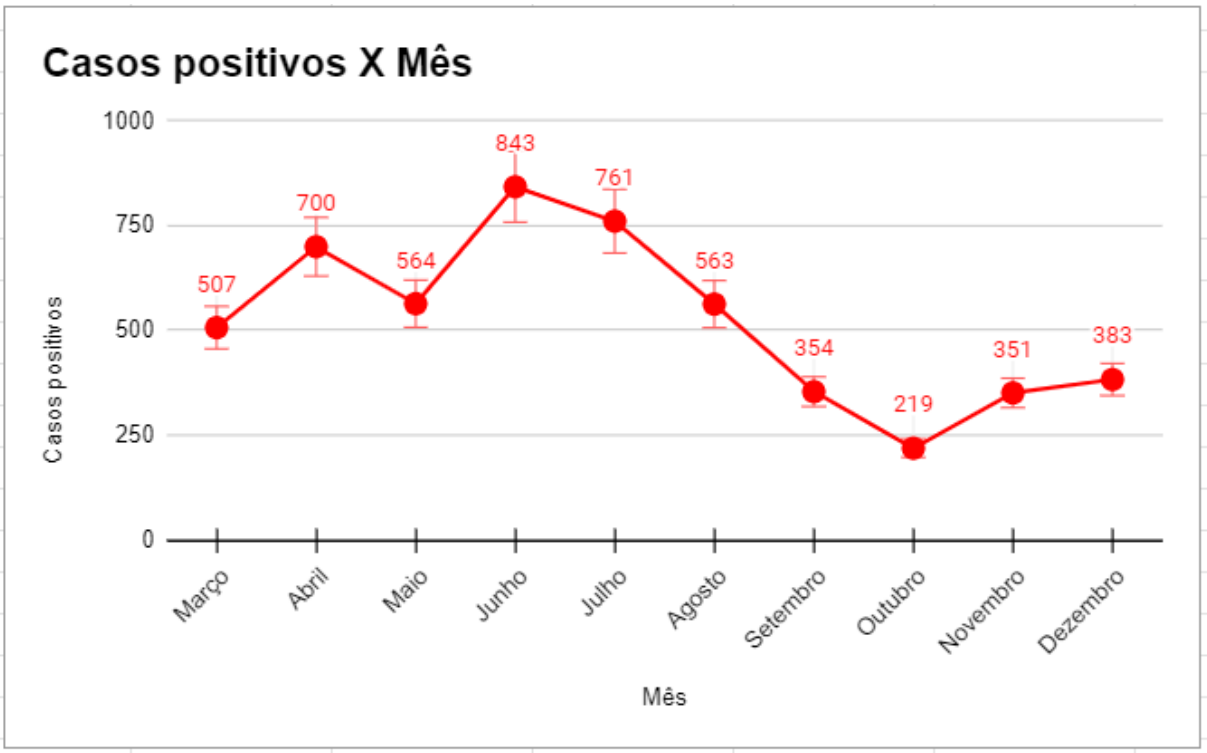
```
SELECT EXTRACT(MONTH FROM dt_coleta) as mes, COUNT(DISTINCT(id_paciente))
AS casos_positivos FROM exames
            WHERE upper(de_exame) LIKE '%COVID%'
            AND (upper(de_resultado) LIKE '%POSITIVO%'
```

```
                        OR upper(de_resultado) LIKE 'DETECTADO%'
                        OR upper(de_resultado) LIKE 'DETECTADOS
ANTICORPOS%'
                        OR upper(de_resultado) LIKE 'REAGENTE%'
                        OR upper(de_resultado) LIKE 'AMOSTRA REAGENTE%')
            AND upper(de_resultado) NOT LIKE '%A DINÂMICA DE PRODUÇÃO DE
ANTICORPOS NA COVID-19 AINDA NÃO É BEM ESTABELECIDA%'
      GROUP BY mes
      ORDER BY mes;
```

Obtivemos o seguinte resultado:

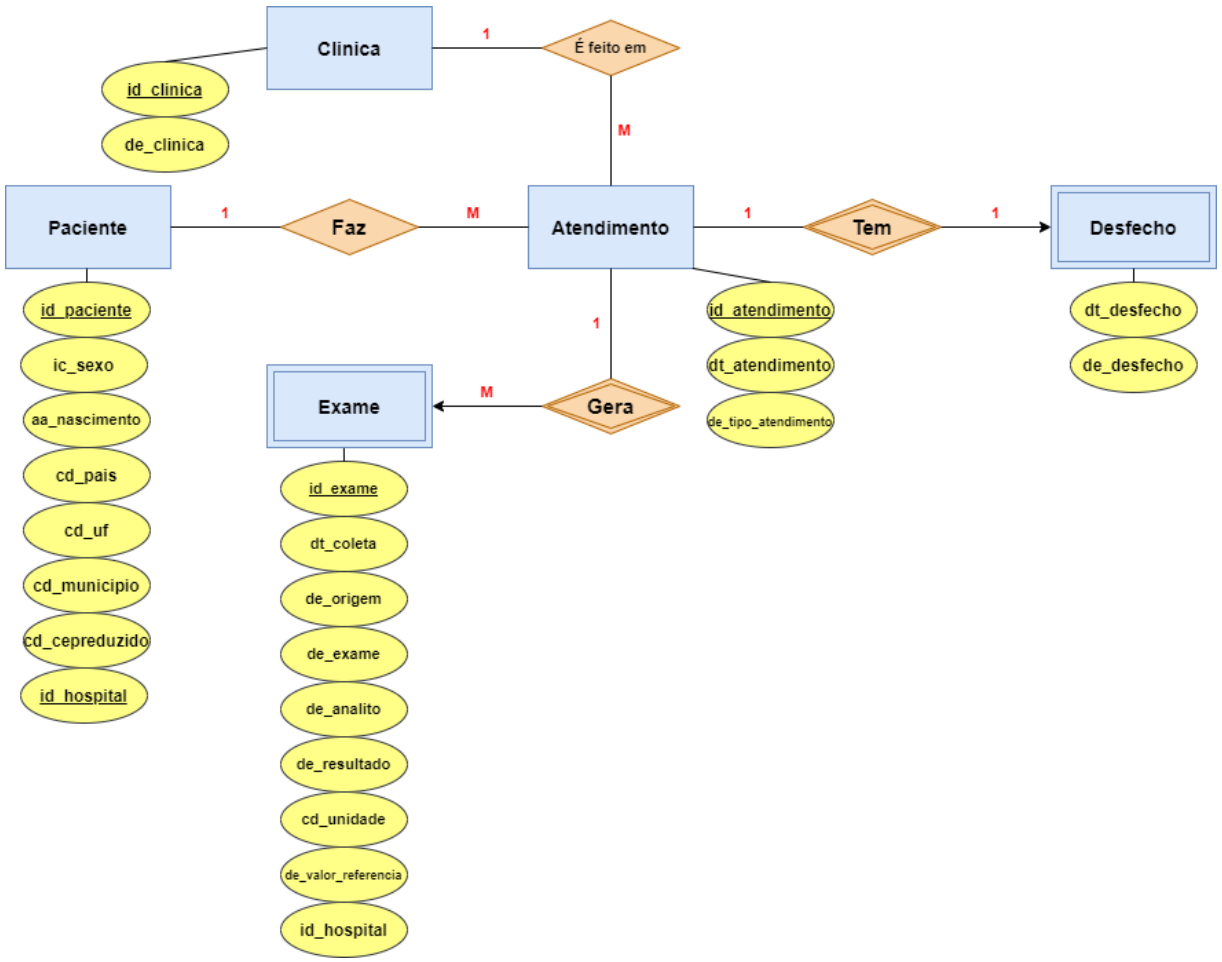
Mês	Casos positivos
Março	507
Abril	700
Maio	564
Junho	843
Julho	761
Agosto	563
Setembro	354
Outubro	219
Novembro	351
Dezembro	383

No formato de um gráfico temos:



Analisando os dados podemos claramente observar que o pico de casos aconteceu na metade do ano de 2020 e a partir de julho houve uma tendência de baixa até outubro onde voltou a aumentar o número de casos que a partir de novembro já foi possível identificar uma tendência de alta devido ao afrouxamento das medidas restritivas e que infelizmente culminou em um aumento desenfreado no ano de 2021.

Para responder a pergunta da **letra C** sobre quais tipos de informações adicionais, em versões futuras da base de dados, poderiam ser coletadas em novos hospitais para melhorar a qualidade dos dados analisados primeiro devemos levar em conta a necessidade de normalização do banco, pois em uma versão otimizada ele deveria estar próximo do modelo a seguir:



Agora em relação a informações adicionais uma interessante seria a de se um paciente já foi vacinado e/ou completamente imunizado e qual foi a vacina tomada por ele. Outra informação interessante seria a cepa do vírus detectado em todos os casos positivos, para que fosse possível auxiliar no controle de variantes de preocupação em determinados territórios, mas isso exigiria uma testagem mais detalhada de cada paciente para que fosse possível ter esse dado armazenado.

Para saber a respeito da vacinação poderíamos criar a seguinte tabela:

```
CREATE TABLE VacinacaoCovid
(
    id_paciente TEXT NOT NULL,
    vacina TEXT NOT NULL,
    dose CHAR NOT NULL,
    CONSTRAINT vacina_id_paciente_fkey FOREIGN KEY (id_paciente)
        REFERENCES pacientes (id_paciente) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

Com essa tabela conseguimos manter o controle de quais pacientes tomaram quais vacinas e qual dose, se tomou a primeira a dose é igual 1, se já tomou a segunda, altera o registro desse paciente e coloca dose igual a 2.

Já para sabermos a respeito da cepa, poderíamos criar a tabela a seguir:

```
CREATE TABLE CepaCovid
(
    id_exame INTEGER NOT NULL,
    cepa TEXT NOT NULL,
    CONSTRAINT cepa_id_exame_fkey FOREIGN KEY (id_exame)
        REFERENCES exames (id_exame) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

Nessa tabela seriam inseridos apenas ids de exame que confirmaram positivo para covid, com o id do exame podemos ter acesso a todas as informações relacionadas ao exame como data da coleta, tipo do exame, hospital realizado etc. Além disso, através da tabela exame temos acesso ao paciente, então também saberemos de qual região ele é, sua idade, sexo etc. Ademais, essa tabela atende aos requisitos de normalização.

Referências bibliográficas

https://covid.saude.gov.br/
acesso: 20 de Julho de 2021;
https://saude.abril.com.br/medicina/testes-do-novo-coronavirus/
acesso: 21 de Julho de 2021;