# Bases de Dados

**Módulo 21: Criação de índices para otimizar consultas e análise exploratória com uso de SQL no Data Warehouse da COVID.**

**Prof. André Bruno de Oliveira**

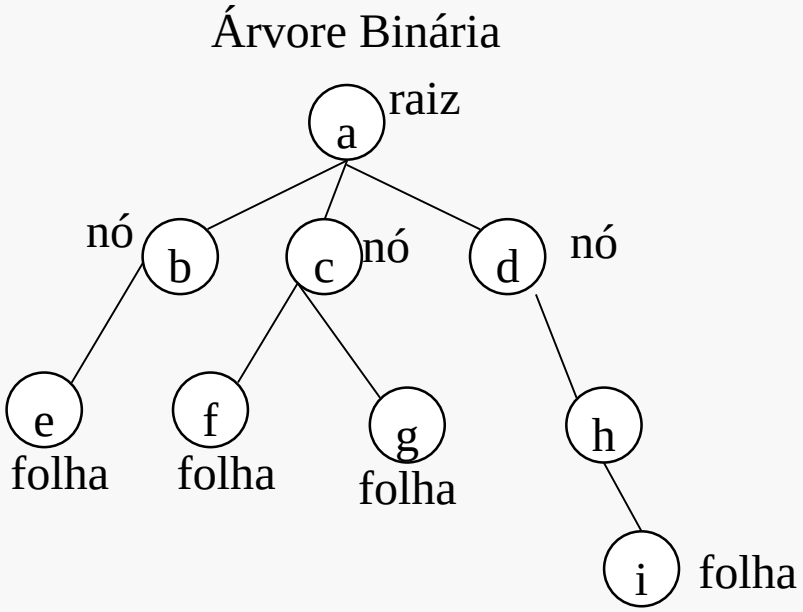
30/05/24 19:35

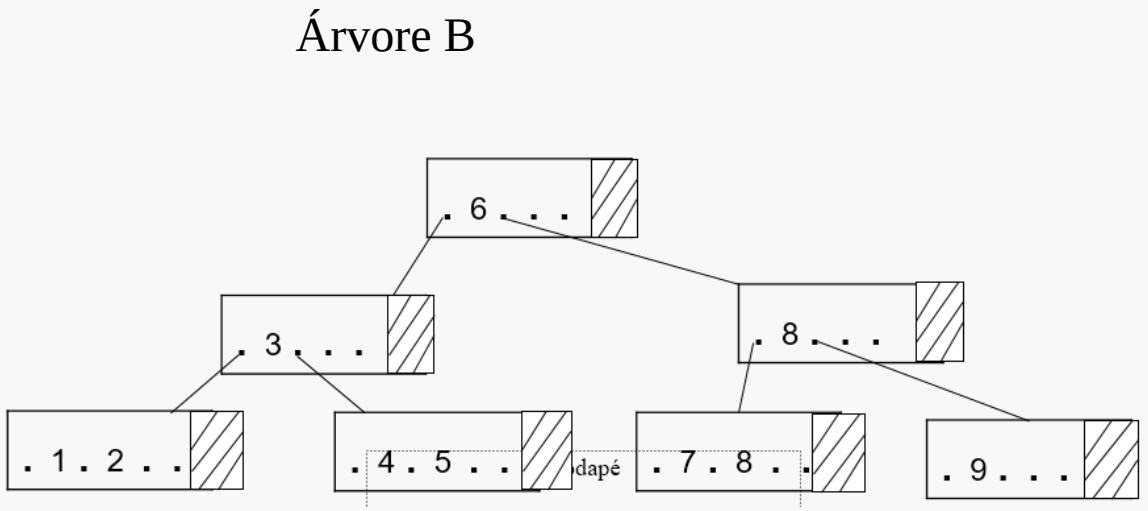
# Construção de Índices

- O que é um índice?
  - Em bancos de dados relacionais, uma tabela contem um conjunto linhas. Cada linha tem a mesma estrutura de coluna que consiste em células. Cada linha também tem um número de sequência consecutivo usado para identificar a linha. Portanto, você pode considerar uma tabela como uma lista de pares: (lança, linha).
  - Um índice é uma estrutura de dados adicional que ajuda a melhorar o desempenho de uma consulta.
  - SQLite e outros BD comerciais usam árvore (B-tree) para organizar índices. Note que B significa equilibrado, árvore B é uma árvore equilibrada, não uma árvore binária.

# Construção de Índices

- A árvore B mantém a quantidade de dados em ambos os lados da árvore equilibrada para que o número de níveis que deve ser atravessado para localizar uma linha esteja sempre no mesmo número aproximado. A consulta usando este tipo de solução é muito eficiente.



# Construção de Índices

- Declaração SQLite para criar índice.
  - Para criar um índice, você usa o script SQL CREATE INDEX com a seguinte sintaxe:

    CREATE INDEX nome do índice

    ON table_name (lista de colunas separadas por vírgula);

    Para criar um índice, especifique as seguintes informações:

    O nome do índice após o CREATE INDEX.

    O nome da tabela ao qual o índice pertence.

    Uma lista de colunas do índice.

# Construção de Índices

- **Índice único**.
- Quando os valores de uma ou mais colunas são únicos, como e-mail e cpf, usa-se a opção UNIQUE no script de criação do índice. Note que a cláusula UNIQUE é opcional.

```
CREATE [UNIQUE] INDEX nome do índice
ON table_name (lista de colunas separadas por vírgula);
```

# Construção de Índices

- **Exemplos:**
  - Vamos criar uma tabela chamada contatos para exemplificar.

```
CREATE TABLE contatos (
    primeiro_nome text NOT NULL,
    ultimo_nome text NOT NULL,
    email text NOT NULL
);
```

Suponha que você queira impor que o e-mail seja único, você cria um índice exclusivo da seguinte forma:

```
CREATE UNIQUE INDEX idx_contatos_email
ON contatos (email);
```

# Construção de Índices

- Vamos fazer um teste:

  Faça a primeira a inserção

  INSERT INTO contatos (primeiro_nome, ultimo_nome, email)

  VALUES('John','Doe','john.doe@sqlitetutorial.net');

  Agora vamos incluir mais duas linhas:

  INSERT INTO contatos (primeiro_nome, ultimo_nome, email)

  VALUES('Johny','Doe','john.doe@sqlitetutorial.net');

  SQLite emitiu uma mensagem de erro indicando que o índice exclusivo foi violado. Porque quando você inseriu a segunda linha, o SQLite verificou e se certificou de que o e-mail é exclusivo em linhas da tabela contatos.

# Construção de Índices

- Vamos inserir mais duas linhas:

```
INSERT INTO contatos (primeiro_nome, ultimo_nome, email)
VALUES('David','Brown','david.brown@sqlitetutorial.net'),
('Lisa','Smith','lisa.smith@sqlitetutorial.net');
```

- Faça uma consulta na tabela *Contatos* com base em um e-mail específico, como teste. Assim, o SQLite vai usar o índice para localizar os dados. Veja a seguinte declaração:

```
SELECT primeiro_nome, ultimo_nome, email FROM Contatos
WHERE    email = 'lisa.smith@sqlitetutorial.net';
```

# Construção de Índices

- Para verificar se o SQLite usa o índice ou não, você usa o script EXPLAIN QUERY PLAN, conforme o script abaixo:

```
EXPLAIN QUERY PLAN
SELECT primeiro_nome, ultimo_nome, email FROM Contatos
FROM Contatos
WHERE email = 'lisa.smith@sqlitetutorial.net';
```

| | id | parent | notused | detail |
|---|---|---|---|---|
| 1 | 3 | 0 | 0 | SEARCH Contatos USING INDEX idx_contatos_email (email=?) |

# Construção de Índices

- Criação de índice multicoluna no SQLite
- Quando se cria um índice que consiste em uma coluna, SQLite usa essa coluna como a chave de classificação.
- Quando  é criado um índice que tenha várias colunas, o SQLite usa as colunas adicionais como a segunda, terceira, ... como as chaves de classificação.
  - SQLite classifica os dados no índice de multicoluna pela primeira coluna especificada no CREATE INDEX, depois pela segunda coluna, depois pela terceira e assim por diante. Portanto, a ordem da coluna é muito importante quando você cria um índice.
  - Para que o SQLite use um índice multicoluna, a consulta deve conter as colunas definida no índice, preferencialmente na ordem de criação dos índices.
  - Dependendo da versão do SQLite e de outros bancos de dados populares, a ordem das colunas pode ser decisiva para que o mecanismos de BD faça uso do índice.

# Construção de Índices

- Exemplo de criação de índice multicoluna.

```
CREATE INDEX idx_conttaos_nome
ON contatos (primeiro_nome, ultimo_nome);
```

•Se você consultar a tabela *Contatos* com uma das seguintes condições na cláusula WHERE:

(1)
```
WHERE
        ultimo_nome = 'Doe';
```

(2)
```
Where
    priemiro_nome= 'John  or ultimo_nome = 'Doe';
```

**O SQLite não vai usar o índice.**

# Construção de Índices

- Se a tabela *Contatos* for consultada com uma das seguintes condições na cláusula WHERE:

(1) WHERE primeito_nome = 'John' AND ultimo_nome = 'Doe‘ and email='lisa.smith@sqlitetutorial.net';

(2) WHERE primeito_nome = 'John' AND ultimo_nome = 'Doe‘ and email='lisa.smith@sqlitetutorial.net';

**O SQLite vai usar o índice.**

Vejamos a execução das consultas a seguir.

(a) EXPLAIN QUERY PLAN
SELECT primeiro_nome, ultimo_nome, email FROM Contatos
WHERE primeiro_nome = 'John' and ultimo_nome = 'Doe' and email='lisa.smith@sqlitetutorial.net';

(b) EXPLAIN QUERY PLAN
SELECT primeiro_nome, ultimo_nome, email FROM Contatos
WHERE primeiro_nome = 'John' and email='lisa.smith@sqlitetutorial.net' and ultimo_nome = 'Doe';

WHERE first_name = 'John' AND last_name = 'Doe';

# Construção de Índices

- Para remover um índice de um banco de dados, você usa o seguinte script SQL:
  - DROP INDEX nome do índice;

# DW COVID com base de dados SQLite

## Descrição do projeto:

- É construído um Data Warehouse (na verdade um data Mart) seguido de uma modelagem dimensional estrela com os microdados da COVID em Minas Gerais.
  - (**Etapa de extração**)
  - Um conjunto de dados foi obtido no portal de Dados Abertos da Secretaria de Estado de Saúde de Minas Gerais no formato CSV desde 02/06/2021.
  - O dicionário de dados que descreve o CSV foi baixado para ajudar na etapa de transformação e carga dos dados.
  - Foram usadas tabelas dos 'Sistemas' e 'Laboratórios‘: Ambas descrevem, para cada linha da tabela, uma descrição de um paciente de SARS-Cov 2 do estado de Minas Gerais.
  - Foram coletados também dados referentes a estimativa de população no site do IBGE e do PIB.

# DW COVID

## (**Etapa de transformação**)

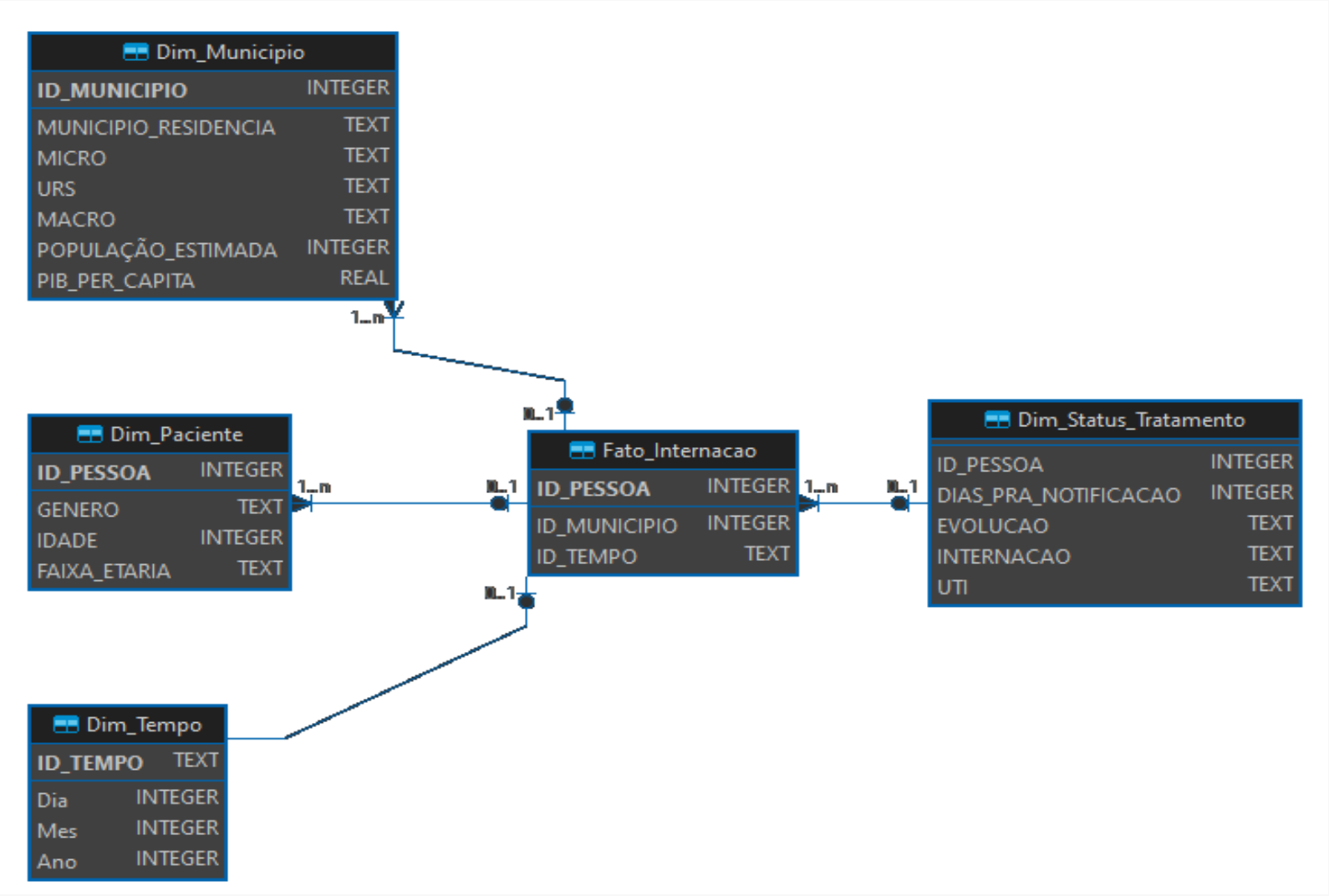
- Foi realizado um trabalho de Data Wrangling (é o processo de transformar e estruturar dados de um formato bruto em um formato desejado com a intenção de melhorar a qualidade deles e torná-los mais consumíveis e úteis para análises).
  - Foram feitas remoções de colunas
  - Remoção de nulos, assim linhas com nulos foram eliminadas
  - Renomeou o nome de algumas colunas para facilitar a compreensão.
  - Correção de alguns valores de idade e ID.

# DW COVID

- **Etapa de Carga**
  - Após as transformações feitas nos dados e ajustes nas tabelas fato e dimensão, foi realizada a carga usando Python.
- **Etapa de análise**
  - Foram feitas consultas ao DW para se extrair informações e realizar uma análise exploratória. Para isso foi usado o Jupyter Notebook combinado com Python.
  - O Jupyter Notebook é um ambiente de desenvolvimento integrado (IDE) que combina a edição e execução de código com a visualização de resultados, sendo útil tanto para desenvolvedores assim como para profissionais de Ciência de Dados.
  - As análises exploratórias trouxeram alguns insumos para construção de um algoritmo de aprendizado de máquina.

# Modelo de Data Warehouse da COVID

- O DW original não tem FK e PK. O Esquema abaixo foi feito para facilitar a compreensão.






Fonte: https://github.com/leticiatavaresds/

# Exercícios de criação de PK, FK e índices.

- O esquema deste BD da COVID não possui chaves primaras e nem chaves estrangeiras possivelmente para dar maior liberdade para as operações de atualização das tabelas.

# Exercícios de criação de PK e índices.

- (1) Primeiro devemos providenciar as criação de chaves primárias. Uma estratégia para avaliar a otimização de busca nas tabelas é priemiro partir de alguns tipos de buscas que serão muito utilizadas pelas queries.
- A consulta a seguir, foi criada para reunir as informações num único local que facilite a criação de outras consultas. Use-a em conjunto com o comando de avaliação de custo de queries para ajudar a descobrir os índices necessários. Quando aparece o texto **SCAN** (significa percorrer toda a tabela) sem a indicação de uso do índice devemos procurar otimizar a consulta.
- Execute a consulta para como forma de avaliar a situação inicial que precisa ser otimizada. Você vai perceber que a tabela F (Fato) precisa ser otimizada. Vá criando as PK de cada tabela e avalie o resultado da otimização.

# Exercícios de criação de PK e índices.

```
EXPLAIN QUERY PLAN
SELECT
F.*, T.Ano,T.Mes,T.Dia,P.IDADE, P.FAIXA_ETARIA, P.GENERO,
M.MUNICIPIO_RESIDENCIA,M.POPULAÇÃO_ESTIMADA POPULACAO,
S.INTERNACAO,S.INTERNACAO, MACRO, MICRO
FROM Fato_Internacao F
INNER JOIN DIM_TEMPO T ON T.ID_TEMPO=F.ID_TEMPO
INNER JOIN DIM_PACIENTE P ON P.ID_PESSOA=F.ID_PESSOA
INNER JOIN DIM_MUNICIPIO M ON M.ID_MUNICIPIO=F.ID_MUNICIPIO
INNER JOIN DIM_STATUS_TRATAMENTO S ON S.ID_PESSOA=F.ID_PESSOA;
```

| | id | parent | notused | detail |
|---|---|---|---|---|
| 1 | 6 | 0 | 0 | SCAN F |
| 2 | 21 | 0 | 0 | SEARCH T USING AUTOMATIC COVERING INDEX (ID_TEMPO=?) |
| 3 | 40 | 0 | 0 | SEARCH P USING AUTOMATIC COVERING INDEX (ID_PESSOA=?) |
| 4 | 61 | 0 | 0 | SEARCH M USING AUTOMATIC COVERING INDEX (ID_MUNICIPIO=?) |
| 5 | 79 | 0 | 0 | SEARCH S USING AUTOMATIC COVERING INDEX (ID_PESSOA=?) |

# Exercícios de criação de PK, FK e índices.

(2) Após a criação das PK, faça nova avaliação e procure criar de índices para as tabelas com SCAN.

```
EXPLAIN QUERY PLAN
SELECT
F.*, T.Ano,T.Mes,T.Dia,P.IDADE, P.FAIXA_ETARIA, P.GENERO,
M.MUNICIPIO_RESIDENCIA,M.POPULAÇÃO_ESTIMADA POPULACAO,
S.INTERNACAO,S.INTERNACAO, MACRO, MICRO
FROM Fato_Internacao F
INNER JOIN DIM_TEMPO T ON T.ID_TEMPO=F.ID_TEMPO
INNER JOIN DIM_PACIENTE P ON P.ID_PESSOA=F.ID_PESSOA
INNER JOIN DIM_MUNICIPIO M ON M.ID_MUNICIPIO=F.ID_MUNICIPIO
INNER JOIN DIM_STATUS_TRATAMENTO S ON S.ID_PESSOA=F.ID_PESSOA;
```

# Exercícios SQL com DW da COVID

(1) Encontre a frequência de casos de COVID por gênero e grupo etário. Faça uma avaliação visual e identifique o grupo etário modal de ambas as distribuições, Feminino e Masculino.

(2) Encontre a idade média de casos de COVID por gênero.

(3) Encontre a taxa de pessoas que coontrairam COVID no período de estudo por macro região, classifique nas ordem descendente por taxa. (Taxa = 100.000 x número casos / População). Identifique as cinco macro regiões com pior situação, de acordo com o valor da taxa.

(4) Encontre o valor médio do PIB per capita médio de cada macro de regiões para o resultado da consulta da pergunta (3) . (valor médio do PIB per capta da macro região = PIB_PER_CAPTA * POPULACAO do Município 1 + PIB_PER_CAPTA * POPULACAO do Município 2 + ... + PIB_PER_CAPTA * POPULACAO do Município N) / (POPULACAO do município 1 + ...+ POPULACAO do município N).

(5) Concatene o resultado da consulta (4) com o resultado da consulta(3). Observe se pode haver algum tipo de correlação entre condição econômica e a taxa de COVID.

Obrigado