

# Relatório Projeto de Banco de Dados

## Criação de IDs Fake

Para melhorar a estrutura das tabelas do banco de dados, optamos por criar IDs artificiais (fake IDs) para substituir chaves compostas fracas e strings que não são ideais como chaves primárias. Este foi o caso das tabelas:

- Conversao
- Pais
- Usuario"
- Canal
- Video
- Comentario
- Doacao
- Bitcoin
- PayPal
- CartaoCredito
- MecanismoPlat

## Criação das Views

Após uma análise das consultas que o projeto deve contemplar, observou-se que frequentemente utilizaremos como base as tabelas: Doação, Inscrição e Patrocínio. Além disso, essas tabelas precisarão ser unidas com outras para obter os resultados desejados. As principais consultas a serem realizadas incluem:

1. Identificar quais são os **canais patrocinados** e os valores de patrocínio pagos por empresa.
2. Descobrir de quantos canais cada **usuário é membro** e a soma do valor desembolsado por usuário por mês.
3. Listar e ordenar os canais que já receberam **doações** e a soma dos valores recebidos em
4. doação.
5. Listar a soma das **doações** geradas pelos comentários que foram lidos por vídeo.

6. Listar e ordenar os k canais que mais recebem **patrocínio** e os valores recebidos.
7. Listar e ordenar os k canais que mais recebem aportes de **membros** e os valores recebidos.
8. Listar e ordenar os k canais que mais receberam **doações** considerando todos os vídeos.
9. Listar os k canais que mais faturam considerando as três fontes de receita: **patrocínio, membros inscritos e doações**.

Para otimizar essas consultas e aumentar o desempenho, foram criadas as seguintes **views materializadas**:

- **patrocinio\_empresa**

Essa view materializada calcula e armazena informações sobre patrocínios de empresas a canais, permitindo acesso rápido e eficiente a dados frequentemente consultados sem a necessidade de executar joins complexos entre Empresa, Patrocínio e Canal.

- **inscricao\_usuario**

Essa view materializada agrega informações sobre as inscrições dos usuários, como a quantidade de canais inscritos e o valor total das inscrições. Isso otimiza consultas frequentes sobre dados de inscrição, eliminando a necessidade de recalcular agregações sempre que forem necessárias.

- **doacao\_canal**

Essa view materializada agrega informações sobre as doações feitas a canais, incluindo o total de doações, data e status das doações. Isso otimiza as consultas relacionadas a doações, reduzindo a complexidade e o tempo de execução de joins e agregações entre as tabelas Canal, Vídeo, Comentário e Doação.

## Criação dos Índices

- CREATE INDEX ON Comentario (id\_video, id\_usuario);
- CREATE INDEX ON Patrocinio (nro\_empresa, id\_canal);
- CREATE INDEX ON Inscricao (id\_canal, id\_usuario);

## Justificativa

**Índice 1: CREATE INDEX ON Comentario (data\_registro\_comentario);**

1. Consultas por Data:

- Consultas que buscam comentários feitos em um determinado período ou ordenados por data de registro se beneficiam desse índice.
  - Por exemplo, ao listar todos os comentários feitos em um vídeo em uma determinada data ou intervalo de datas, o índice permite acessar rapidamente os comentários sem fazer uma varredura completa na tabela.
2. Otimização de Agregações e Filtros:
- Melhor desempenho em operações de agregação e filtragem baseadas em data, como contagens de comentários por data ou intervalos de tempo.
  - Facilita a geração de relatórios e análises temporais, como o número de comentários por dia ou mês.

### **Índice 2: CREATE INDEX ON Patrocinio (nro\_empresa, id\_canal);**

1. Consultas por Empresa e Canal:
- Consultas que verificam patrocínios de uma empresa específica a um canal específico se beneficiam desse índice.
  - Por exemplo, ao buscar todos os patrocínios de uma empresa específica para todos os canais que ela patrocina, esse índice melhora a eficiência.
2. Eficiência em Joins e Agrupamentos:
- Melhora o desempenho em joins entre Patrocinio e outras tabelas, como Empresa e Canal.
  - Facilita operações de agrupamento e agregação, como calcular o total de patrocínios de uma empresa ou listar canais patrocinados por uma empresa.

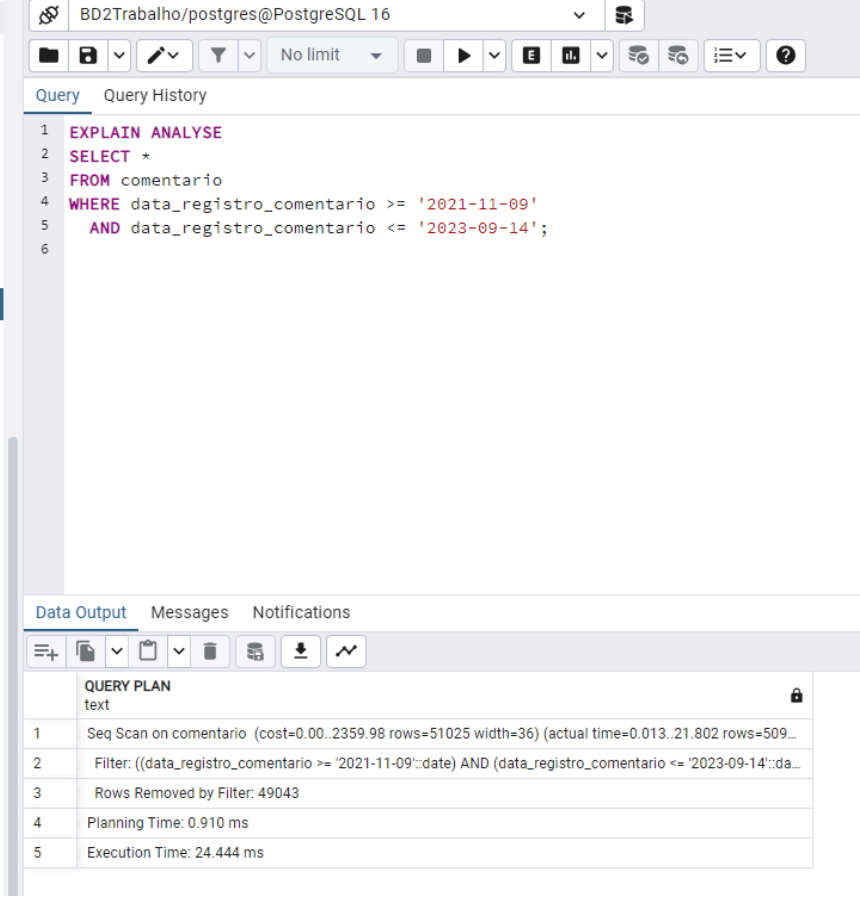
### **Índice 3: CREATE INDEX ON Inscricao (id\_canal, id\_usuario);**

1. Consulta de Inscrições por Canal e Usuário:
- Consultas que buscam inscrições de um usuário específico em um canal específico se beneficiam diretamente desse índice.
  - Por exemplo, ao listar todos os usuários inscritos em um canal ou verificar se um usuário específico está inscrito em um canal, esse índice melhora a eficiência.
2. Eficiência em Joins e Agrupamentos:
- Melhora o desempenho em joins entre Inscricao e outras tabelas, como Usuario e Canal.
  - Facilita operações de agrupamento e agregação, como calcular o número total de inscrições por canal ou listar canais inscritos por um usuário específico.

## Benefícios dos Índices Exemplo Prático:

Observação: Para a realização dos seguintes testes utilizamos um script para gerar 100000 tuplas que deve ser inserido na tabela comentario no lugar das inserções que estão presentes no sql de inserção, portanto, para realizar esse teste disponibilizamos um sql de inserção alternativo para a tabela comentário(ComentariosInsercaoAlternativa.sql), o uso dessa quantidade maior de inserção foi para evidenciar a diminuição do tempo de busca após a criação do index.

### Sem Index:



The screenshot shows a PostgreSQL query editor interface. The query being executed is:

```
1 EXPLAIN ANALYSE
2 SELECT *
3 FROM comentario
4 WHERE data_registro_comentario >= '2021-11-09'
5 AND data_registro_comentario <= '2023-09-14';
6
```

The results pane shows the query plan:

	QUERY PLAN
1	Seq Scan on comentario (cost=0.00..2359.98 rows=51025 width=36) (actual time=0.013..21.802 rows=509...
2	Filter: ((data_registro_comentario >= '2021-11-09'::date) AND (data_registro_comentario <= '2023-09-14'::da...
3	Rows Removed by Filter: 49043
4	Planning Time: 0.910 ms
5	Execution Time: 24.444 ms

## Com Index:

The screenshot shows a PostgreSQL query editor interface. The top bar indicates the database is 'BD2Trabalho/postgres@PostgreSQL 16'. The query editor contains the following SQL code:

```
1 EXPLAIN ANALYSE
2 SELECT *
3 FROM comentario
4 WHERE data_registro_comentario >= '2021-11-09'
5 AND data_registro_comentario <= '2023-09-14';
6 CREATE INDEX ON Comentario (data_registro_comentario);
```

Below the query editor, the 'Data Output' tab is active, displaying the 'QUERY PLAN' for the executed query. The plan details are as follows:

Step	Operation	Cost	Rows	Width	Actual Time	Actual Rows	Loops
1	Bitmap Heap Scan on comentario	(cost=707.30..2332.67 rows=51025 width=36)	51025	36	4.733..13.451	50956	1
2	Recheck Cond: ((data_registro_comentario >= '2021-11-09'::date) AND (data_registro_comentario <= '2023-09-14'::date))						
3	Heap Blocks: exact=860						
4	-> Bitmap Index Scan on comentario_data_registro_comentario_idx	(cost=0.00..694.54 rows=51025 width=0)	51025	0	4.596..4.596	50956	1
5	Index Cond: ((data_registro_comentario >= '2021-11-09'::date) AND (data_registro_comentario <= '2023-09-14'::date))						
6	Planning Time: 1.751 ms						
7	Execution Time: 16.159 ms						