Criação das tabelas

Iniciamos a criação das tabelas a partir do modelo relacional proposto pelo professor, e optamos por algumas adaptações, para simplificar a utilização de chaves estrangeiras. O modelo foi implementado utilizando UUIDs como chaves primárias para as tabelas principais (canal, video, comentario, doacao), e chaves compostas quando necessário, para garantir escalabilidade e integridade referencial.

São elas:

- Criação da PK com uuid artificial para a tabela canal. Fizemos isso, pois a chave candidata para PK teria 2 atributos e esta tabela está referenciada em 3 outras tabelas (video, patrocinio e nivel_canal).
- Criação de um uuid artificial para a tabela vídeo, e criação da PK composta com id_canal. Fizemos isso, pois a chave candidata para PK teria 4 atributos e esta tabela está referenciada em 2 outras tabelas (participa e comentario) e, dada a grande quantidade de vídeos esperados, optamos por termos a PK como a combinação do canal e do video.
- Criação de um uuid artificial para a tabela comentario, e criação da PK composta com id_canal e id_video. Fizemos isso, pois a chave candidata para PK teria 3 atributos e esta tabela está referenciada em 2 outras tabelas (doacao e comentario(referência a própria tabela)) e, dada a grande quantidade de comentários para cada vídeo esperados, optamos por termos a PK como a combinação do canal e do video.
- Criação de um uuid artificial para a tabela doacao. Fizemos isso, pois a chave candidata para PK teria 2 atributos e esta tabela está referenciada em 4 outras tabelas (bitcoin, paypal, cartao_credito, mecanismo_plat), além da mesma expectativa acima da grande quantidade de dados inseridos estourarem a quantidade de uuids disponíveis.
- **Além disso, todas as FKs e PKs seguem o padrão do modelo relacional, com restrições de unicidade e not null apropriadas.**

Observação: Não há uso de ids artificiais sequenciais para as tabelas principais. Utilizamos UUIDs e chaves compostas para garantir unicidade e escalabilidade. Foram criadas também constraints Not Null, observando cada cenário.

Criação das Views

Foram definidas **5 Visões** (3 Virtuais e 2 Materializadas) no arquivo `2.criar_views.sql` para pré-calcular dados complexos e otimizar as 8 consultas obrigatórias, com foco na eficiência e no uso adequado de cada tipo de visão:

Visões Virtuais (Atualização em Tempo Real)

As **Visões Virtuais** (`VW_`) foram escolhidas para dados que podem ser filtrados ou que dependem de transações recentes (patrocínio vigente, doações recentes):

- **`VW_RECEITA_MEMBROS_BRUTA`** : Calcula a receita mensal bruta por canal, somando os valores de cada nível de inscrição vigente. É essencial para as consultas que envolvem faturamento de membros (Consultas 2, 6 e 8), pois os dados de `inscricao` (membros vigentes) precisam ser lidos em tempo real.
- **`VW_CANAL_RECEITA_PATROCINIO`** : Simplifica a consulta de patrocínio vigente, relacionando o canal, a empresa patrocinadora e o valor do patrocínio. Utilizada nas Consultas 1, 5 e 8.

Visões Materializadas (Otimização para Rankeamento Pesado)

As **Visões Materializadas** (`MV_`) foram escolhidas para agregações pesadas que mudam com pouca frequência (como o total de doações ou o faturamento total), otimizando as consultas de ranking (top *k* canais):

- **`MV_DOACAO_TOTAL_CANAL`** : Calcula a soma total de doações recebidas por cada canal. A agregação de doações ao longo de todos os vídeos é uma operação custosa, e materializá-la otimiza a Consulta 7 (ranking de doações) e a composição da `MV_FATURAMENTO_TOP_CANAIS` .
- **`MV_FATURAMENTO_TOP_CANAIS`** : Combina as três fontes de receita (Patrocínio, Membros, Doações) em um único registro de faturamento total para cada canal. Esta agregação é a mais custosa do sistema e materializá-la garante a máxima performance para a Consulta 8 (ranking de faturamento total). O cálculo utiliza as views virtuais anteriores.
- **`MV_CANAL_VISUALIZACOES`** : Soma do total de visualizações dos vídeos de cada canal.

Criação de Indices

Para otimizar o desempenho das buscas e das operações de junção nas consultas obrigatórias (principalmente as que envolvem faturamento e listagem), foram definidos **5 Índices de Apoio** no arquivo `3.criar_indexes.sql` . A escolha destes índices visou minimizar o *overhead* de inserção, focando em colunas que são chaves estrangeiras ou que são frequentemente usadas em filtros e ordenações:

- **`idx_mdtc_total_doacao`** : Criado na coluna `total_doacao_bruta` da view materializada `MV_DOACAO_TOTAL_CANAL` . Otimiza o sort e filtra os resultados que não serão retornados nas consultas.
- **`idx_mftc_total`** : É um índice **composto** criado nas colunas `faturamento_total` e `nome_plataforma` da view materializada `MV_FATURAMENTO_TOP_CANAIS` . Otimiza o sort e o filtra dos resultados.
- **`idx_patrocinio_valor`** : Criado nas colunas `valor` da tabela `patrocinio` . Otimiza o sort e o filtra dos resultados.
- **`idx_doacao_status_idcomentario`** : Indice composto, criado nas colunas `UPPER(status)` e `id_comentario` da tabela `doacao` . Auxilia nos filtros de busca da consulta 4, já aplicando o modificador UPPER na coluna status.
- **`idx_inscricao_id_usuario`** : Criado na coluna `id_usuario` da tabela `inscricao` . Auxilia no filtro da consulta 2, melhorando a performance ao buscar as inscrições de um usuário específico.

Criação de Triggers

Foram implementadas **4 Triggers** no arquivo `4.criar_triggers.sql` para garantir a consistência e a integridade do banco de dados, conforme as regras de negócio e os requisitos de atributos derivados:

- **`tg_user_count` (Função: `fn_update_user_count`)**: Responsável por manter a consistência do atributo derivado `qtd_users` na tabela `plataforma`. Esta trigger é acionada **APÓS** uma inserção ou remoção na tabela `plataforma_usuario`, atualizando automaticamente o contador de usuários na plataforma correspondente.
- **`tg_check_comentario_cronologia` (Função: `fn_check_comentario_cronologia`)**: Responsável por manter a consistência dos comentários na tabela `comentarios`. É acionada **ANTES** uma inserção ou update na tabela `comentario`, checando se o comentário foi feito antes do horário do video.
- **`tg_check_status_doacao` (Função: `fn_check_status_doacao`)**: Atua **ANTES** de qualquer inserção ou atualização na tabela `doacao`. Esta trigger garante que o campo `status` só receba valores válidos (como 'RECUSADA', 'RECEBIDA', 'LIDA' ou 'CONFIRMADA'), atendendo a um requisito de consistência de dados do projeto.
- **`tg_check_streamer` (Função: `fn_check_streamer_exists`)**: Atua **ANTES** da inserção na tabela `streamer_pais`. Sua função é garantir a integridade referencial e a lógica de negócio, assegurando que o `nick_streamer` a ser inserido já exista na tabela `usuario` (validando o subtipo).
- **`tg_update_channel_views` (Função: `fn_update_channel_views`)**: Atua **DEPOIS** de update na tabela `video`, porém a atualização é realizada apenas se random_value for = 1, o que limita a atualização a uma chance de 1 em 1000 atualizações de um canal específico. Sua função é atualizar a quantidade de visualizações de cada canal.
- Foi criada a function fn_update_all_channel_views(), que irá atualizar a quantidade de visualisações de todos canais do Banco. Seria utilizado um Cron de hora em hora que faria este update completo, garantindo que o dado esteja atualizado, além da atualização randomica acima.

Criação dos dados artificiais

Para criação dos dados artificiais, optamos por criar functions responsáveis pela inserção dos dados com a criação das chaves estrangeiras e garantindo a integridade. Após as functions criadas, criamos outras functions que populam o banco utilizando as functions anteriores, simulando a criação natural de dados.

Consultas

Todas as consultas foram implementadas como **Functions** no arquivo `7.executar_consultas.sql`, utilizando a linguagem **PL/pgSQL** ou **SQL** (conforme aplicável) e fazendo uso das Views e Índices criados para garantir alta performance.

As funções com parâmetro `DEFAULT NULL` ou `k` atendem ao requisito de ter parâmetros opcionais ou de listar os *top k* elementos.

1. Detalhamento das Funções Parametrizadas (Com Filtro Opcional)

- **Consulta 1: Canais Patrocinados** (`fn_canais_patrocinados(_empresa_nome varchar default null, _empresa_tax_id varchar default null, _nome_plataforma varchar default null)`): Lista canais e o valor do patrocínio vigente. A função otimiza o filtro por empresa patrocinadora e/ou por plataforma.
- **Consulta 2: Valor Desembolsado por Membro** (`fn_membros_valor_desembolsado(_nick_usuario VARCHAR DEFAULT NULL)`): Calcula a quantidade de canais que cada usuário é membro e a soma do valor mensal desembolsado por ele. Permite filtro opcional para focar em um `_nick_usuario` específico.
- **Consulta 3: Doações Recebidas por Canal** (`fn_canais_doacao_recebida(_nome_plataforma varchar DEFAULT NULL, _nome_canal varchar DEFAULT NULL)`): Lista e ordena os canais que receberam doações. **Otimização:** Utiliza a **`MV_DOACAO_TOTAL_CANAL`** para acesso rápido aos valores totais agregados e permite filtro opcional por plataforma e por canal.
- **Consulta 4: Soma de Doações Lidas por Vídeo** (`fn_doacoes_lidas_por_video(_nome_plataforma varchar DEFAULT NULL, _nome_canal varchar DEFAULT NULL, _titulo_video varchar DEFAULT NULL)`): Lista a soma das doações geradas por comentários cujo `status` é **'LIDA'**, agregadas por vídeo. **Otimização:** A filtragem por `status = 'LIDA'` é acelerada pelo índice **`idx_doacao_status_idcomentario`**, permitindo também filtro opcional por plataforma, canal e titulo.

2. Detalhamento das Funções de Ranking (Top K)

- **Consulta 5: Top K Patrocínio** (`fn_top_k_patrocinio(k INTEGER, _nome_plataforma varchar DEFAULT NULL)`): Lista e ordena os **k** canais com maior valor de patrocínio. **Otimização:** Usa a **`VW_CANAL_RECEITA_PATROCINIO`** para dados em tempo real e aplica `ORDER BY` e `LIMIT k`. Pode ser filtrada por plataforma.
- **Consulta 6: Top K Aportes de Membros** (`fn_top_k_membros(k INTEGER, _nome_plataforma varchar DEFAULT NULL)`): Lista e ordena os **k** canais com maior receita de aportes mensais de membros. **Otimização:** Utiliza a **`VW_RECEITA_MEMBROS_BRUTA`** e aplica `ORDER BY` e `LIMIT k`. Pode ser filtrada por plataforma.
- **Consulta 7: Top K Doações Recebidas** (`fn_top_k_doacoes(k INTEGER, _nome_plataforma varchar DEFAULT NULL)`): Lista e ordena os **k** canais que mais receberam doações (total acumulado). **Otimização:** Executa o `REFRESH MATERIALIZED VIEW MV_DOACAO_TOTAL_CANAL` para garantir a atualização dos dados antes de rankear com `ORDER BY` e `LIMIT k`. Pode ser filtrada por plataforma.
- **Consulta 8: Top K Faturamento Total** (`fn_top_k_faturamento_total(k INTEGER, _nome_plataforma varchar DEFAULT NULL)`): Lista e ordena os **k** canais que mais faturam, considerando as três fontes de receita (Patrocínio, Membros e Doações). **Otimização:** Executa o `REFRESH MATERIALIZED VIEW MV_FATURAMENTO_TOP_CANAIS` e rankeia o faturamento total agregado. Pode ser filtrada por plataforma.