DOCUMENTO DE JUSTIFICATIVA

Criação de Triggers para o Banco de Dados Steam Games Dataset.

Sumário:

Atualização de score do usuário

1. Por que essa trigger existe

A trigger existe para manter o campo user_score sempre atualizado automaticamente, baseado nos valores de positive e negative.
Ela garante que toda vez que um registro da tabela detalhes for inserido ou atualizado, o user_score seja recalculado sem depender da aplicação.
Ou seja, ela centraliza a lógica do cálculo dentro do banco de dados.

2. Qual problema ela resolve

Sem essa trigger, o sistema poderia ter:

- Inconsistência de dados
- A aplicação poderia esquecer de recalcular o score.
- Múltiplos serviços poderiam calcular o score de forma diferente.
- Um update poderia alterar positive e negative, mas não atualizar user_score.
- Lógica duplicada (cada parte da aplicação (API, scripts, jobs) teria que repetir o cálculo do score).

A trigger evita isso.

3. Como ela melhora o sistema

A trigger melhora o sistema de várias formas:

- Garante integridade dos dados user_score sempre reflete o estado real de positive e negative.
- Evita cálculos desnecessários em UPDATE
- Simplifica o código da aplicação (a aplicação não precisa calcular nada — só salva positive e negative e o banco faz o resto).
- O cálculo passa a ser definido uma vez, no banco:
    - Fácil de manter
    - Não tem variação entre serviços diferentes
    - Não depende de linguagem de programação

Verificação de limite de tag

1. Por que essa trigger existe

A trigger existe para garantir que cada jogo tenha no máximo um número permitido de tags, evitando excesso de classificações e mantendo o modelo de dados organizado e coerente.
Ela centraliza essa regra no banco de dados, de forma que qualquer tentativa de inserir uma nova tag para um jogo passe automaticamente por essa validação — independentemente de qual parte da aplicação esteja realizando o INSERT.

2. Qual problema ela resolve

3.

- Evita excesso de tags por jogo, impedindo que um game ultrapasse o limite configurado (ex.: 20).
- Previne inconsistências causadas por múltiplos processos ou serviços tentando adicionar tags simultaneamente.
- Elimina a necessidade de validação manual na aplicação, evitando situações em que uma API verifica o limite, mas outra esquece.
- Protege o banco contra inserts incorretos em massa, migrações ou scripts que poderiam quebrar a regra de limite.
- Evita duplicação de lógica entre serviços diferentes, garantindo que a regra esteja sempre sendo aplicada.

4. Como ela melhora o sistema

- Integridade garantida no nível do banco: o limite é aplicado diretamente no PostgreSQL, tornando impossível ultrapassá-lo.
- Comportamento previsível e centralizado: a regra de limite não depende de linguagens, serviços ou validações externas.
- Evita condições de corrida (race conditions): a contagem é feita dentro da transação que está inserindo a tag, após adquirir o lock necessário.
- Reduz necessidade de código na aplicação: elimina validações repetidas e torna o fluxo mais simples e seguro.
- Erros claros e imediatos: o sistema retorna uma mensagem explícita (“Limite atingido…”) no momento da tentativa, facilitando diagnóstico.

Automação de tags para jogos 18+

1. Por que essa trigger existe

   Essa trigger existe para atribuir automaticamente a tag “mature” sempre que um jogo com required_age >= 18 é inserido no sistema.
   A versão aprimorada melhora a segurança e resiliência da regra: ela valida a existência da tag no banco, trata ausências, evita duplicações e evita falhas silenciosas.
   O objetivo é manter a classificação adulta dos jogos correta, automática e centralizada no banco, sem depender da aplicação ou de processos externos.

   Ela também trabalha junto com a procedure para inserir a tag “mature”.

2. Qual problema ela resolve
   - Garante que jogos adultos recebam a tag correta, mesmo que o backend, um serviço externo ou uma importação de dados esqueça de adicioná-la.
   - Previne inconsistência entre o campo required_age e as tags atribuídas ao jogo.
   - Elimina problemas caso a tag “mature” não exista, evitando exceções inesperadas e registrando avisos claros.
   - Melhora confiabilidade em cenários de migração, importações em massa ou integrações automáticas.
   - Aumento da complexidade
3. Como ela melhora o sistema
   - Consistência automática e imediata: a tag é vinculada dentro da mesma transação em que o jogo é criado.
   - Busca a tag pelo nome exato, depois por aproximação (LIKE '%mature%'), reduzindo risco de falhas por pequenas diferenças de texto.
   - Tratamento de erros:
     - Se appid vier nulo, a trigger apenas notifica e não quebra o fluxo.
     - Se a tag não existir, ela não tenta inserir nada e registra aviso legível.
   - Facilita integrações e relatórios
   - Melhora a performance
   - Oferece maior flexibilidade nas filtragens