Deduplicação

Proteja o efeito de negócio contra reentregas

Pré-requisitos

Conheça SQS, transações e restrições únicas. Use Python 3 com SQLite 3.24 ou posterior. O laboratório conta confirmações por dia e registra quais eventos o consumidor de relatórios já processou.

Separe identidade da entrega e do efeito [1]

A fila pode reenviar uma mensagem. Use um identificador estável do evento lógico, preservado nas novas tentativas. Combine origem e ID se a unicidade for por produtor. Se vários consumidores independentes usam a mesma tabela, acrescente o nome do consumidor para preservar o trabalho de cada um.

Faça a disputa de unicidade no banco [2]

Quem insere pela primeira vez a chave de processamento pode aplicar o efeito. Quem recebe conflito já tem o evento marcado. Este padrão depende de que o recibo e a atualização do relatório sejam gravados na mesma transação. Uma consulta prévia isolada deixa duas execuções enxergarem a chave como ausente.

Planeje falhas e retenção

Se o efeito falha após a inserção da marca, desfaça ambos. Só exclua da fila após a confirmação do banco. Se a exclusão falhar, a nova entrega encontra o recibo. Preserve as chaves durante a janela de reenvio e reprocessamento. Para reconstituir uma projeção do início, use um destino isolado ou um plano que restaure também a trilha de deduplicação.

Escolha outra proteção para efeitos externos

Uma transação local não controla o envio de e-mail. Registre a intenção de notificar em uma outbox e use uma chave estável no serviço que efetivamente realiza o envio quando houver suporte. Alguns eventos distintos podem descrever o mesmo efeito. Nesse caso, proteja também a identidade do efeito.

Deduplicação

Exercício resolvido • Conte uma confirmação

Enunciado

E45 chega duas vezes para 24/11. E46 chega quando o total de 25/11 ainda não existe. Preserve o total de 24/11 em 1 e permita processar E46 após preparar o dia faltante.

1. Comece deduplicar.py

```
import sqlite3

db = sqlite3.connect(':memory:')
db.executescript("""
CREATE TABLE processados (
  consumidor TEXT NOT NULL,
  origem TEXT NOT NULL,
  evento TEXT NOT NULL,
  PRIMARY KEY (consumidor, origem, evento)
);
CREATE TABLE totais (dia TEXT PRIMARY KEY, total INTEGER NOT NULL);
INSERT INTO totais VALUES ('2026-11-24', 0);
""")

def contar(origem, evento, dia):
    with db:
        marca = db.execute("""
            INSERT INTO processados VALUES ('relatorio', ?, ?)
            ON CONFLICT(consumidor, origem, evento) DO NOTHING
        """, (origem, evento))
        if marca.rowcount == 0:
            return False
        efeito = db.execute(
            'UPDATE totais SET total = total + 1 WHERE dia = ?', (dia,)
        )
        if efeito.rowcount != 1:
            raise ValueError('Dia ainda indisponivel')
    return True
```

Deduplicação

Confira a recuperação após uma falha

2. Acrescente ao mesmo arquivo

```
dia = '2026-11-24'
assert contar('reservas', 'E45', dia)
assert not contar('reservas', 'E45', dia)

try:
    contar('reservas', 'E46', '2026-11-25')
except ValueError:
    pass
else:
    raise AssertionError('O dia ainda nao existe')

with db:
    db.execute("INSERT INTO totais VALUES ('2026-11-25', 0)")
assert contar('reservas', 'E46', '2026-11-25')
assert db.execute('SELECT total FROM totais ORDER BY dia').fetchall() == [
    (1,), (1,),
]
print('Uma contagem por confirmacao')
```

3. Execute python deduplicar.py

O primeiro INSERT de E45 insere a marca e o total vira 1. A reentrega não altera o total. Para E46, a falha de atualização faz o contexto reverter a marca. A tentativa seguinte insere a marca novamente e soma ao total do novo dia.

Caso de erro • Confirme a marca cedo

Se executar db.commit() logo depois de inserir a marca, a falha de E46 deixa a marca gravada. Depois de preparar o dia, a repetição retorna False e o total fica zero. Recoloque a marca e o efeito na mesma transação.

Critérios para conferir

Obtenha totais (1, 1) e duas chaves de evento distintas. O teste é local e sequencial. Para validar uma fila real, confira a competição entre consumidores, a recuperação após reinício e a exclusão da mensagem somente após o commit.

Referências primárias

[1] AWS • Entrega pelo menos uma vez no SQS

[2] SQLite • UPSERT e alvo de conflito

Deduplicação

Identifique entregas repetidas pelo ID do evento

Proteja o efeito junto com o registro

O ID do evento deve permanecer estável mesmo se o broker gerar outro ID para a mensagem republicada. Use uma chave única por consumidor e evento. Quando o efeito e o registro ficam no mesmo banco, confirme ambos na mesma transação.

Exercício adicional

Uma falha ocorre depois de atualizar o histórico, mas antes de apagar a mensagem. O que verificar?

Resolução

Reprocesse o mesmo evento e confirme que a atualização já aplicada não se repete. A quantidade de registros no histórico permanece a mesma e a mensagem pode ser reconhecida após a nova tentativa.

Ao assistir ao vídeo

Observe em quais pontos uma mensagem já processada pode voltar.

Vídeo complementar

CodeOpinion - Handling Duplicate Messages (em inglês)