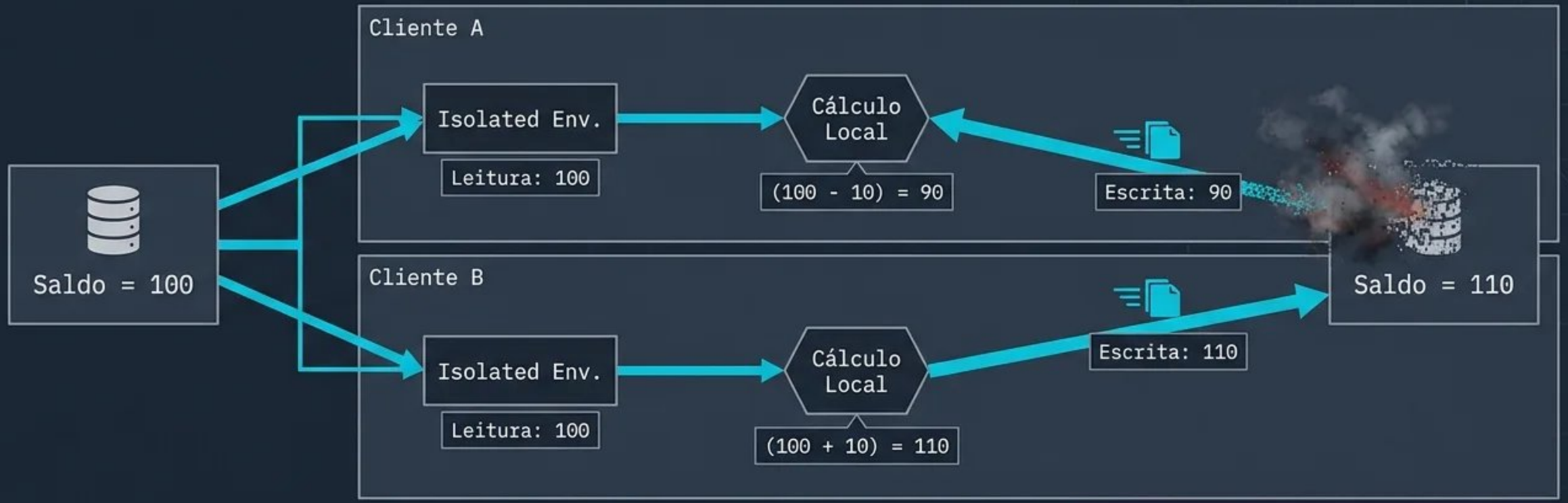
Bloqueio Otimista vs. Pessimista

A física da concorrência: como impedir que gravações simultâneas destruam seus dados (e como escolher a arma certa).

A Ameaça Invisível: The Lost Update



Status: Log Limpo. Zero Erros.

Duas leituras, dois cálculos, duas gravações. O último a chegar apaga silenciosamente o trabalho do primeiro. O perigo não é o erro; é a ausência dele.

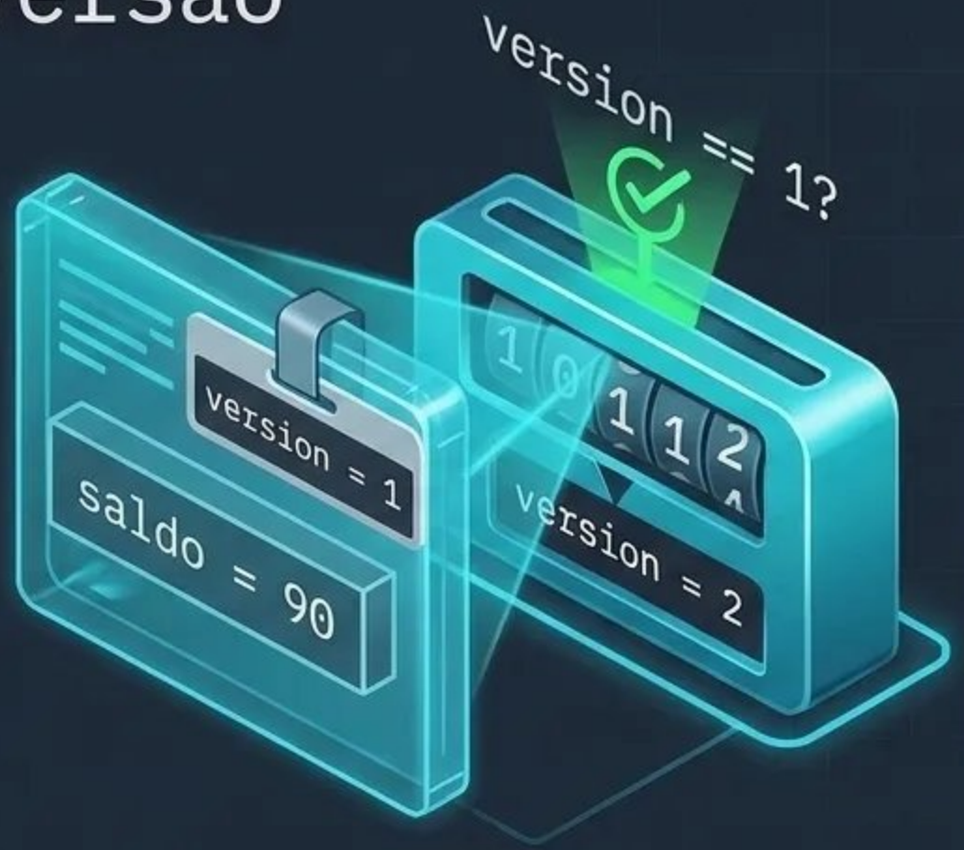
Detecção de Conflito

Trabalhe à vontade, negocie depois.
Aposta que o conflito é raro.
O custo normal é zero, mas o trabalho pode ser perdido.

Prevenção de Conflito

Ninguém mexe até eu terminar.
Aposta que o conflito é provável.
Ninguém perde trabalho, mas todos pagam o preço da espera.

Bloqueio Otimista: A Mecânica da Versão



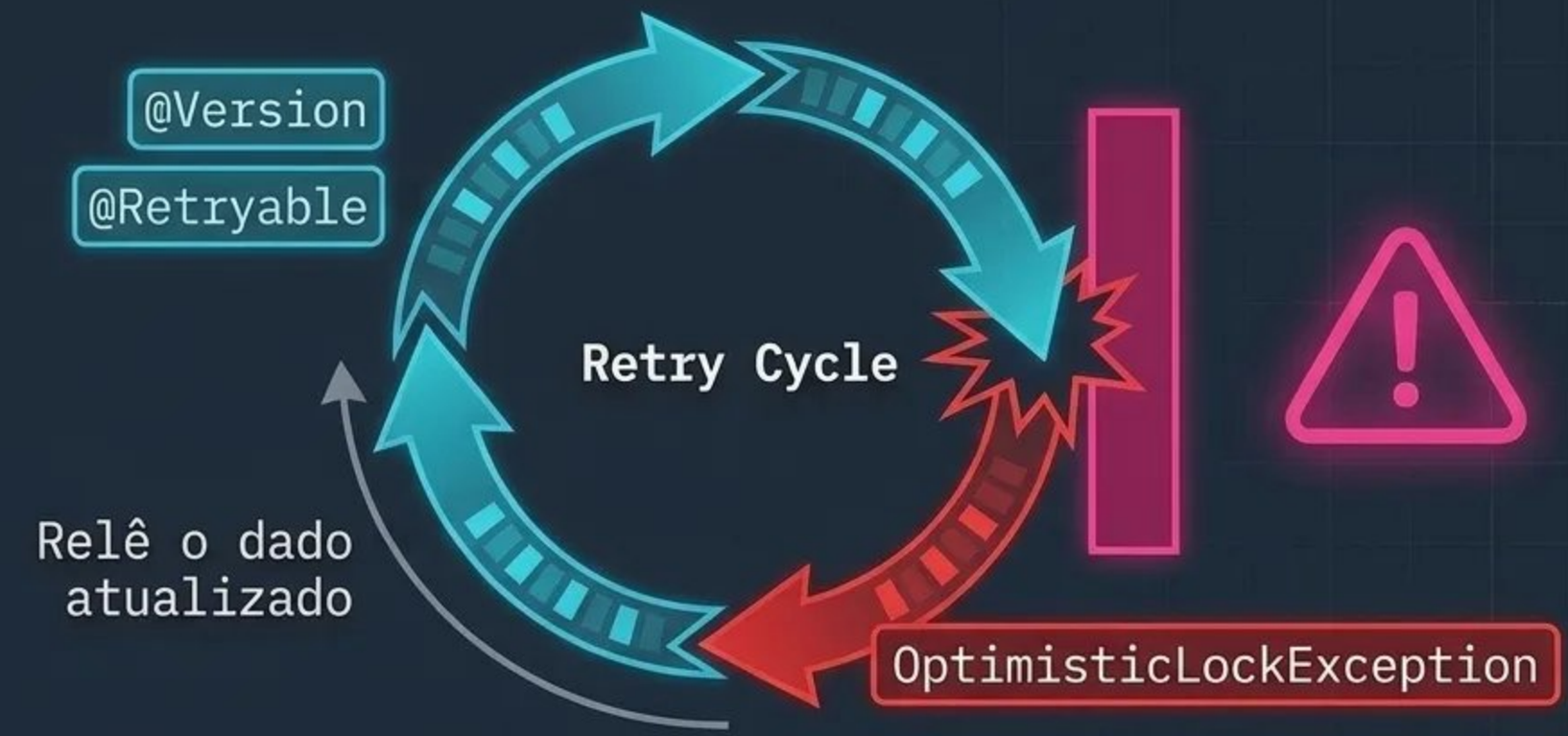
Se outra transação gravou antes, o filtro não encontra a linha (version mudou). O banco devolve 0 linhas afetadas.

Esta é a detecção. Zero travas, zero espera.

```
-- A mágica: verificação e escrita no mesmo comando
UPDATE conta
  SET saldo = 90, version = version + 1
WHERE id = 1 AND version = 1;
```

Otimismo na Prática (E a Queda do Dominó)

Como o conflito vira um erro, a aplicação precisa assumir o controle.



A **Avalanche de Retentativas:** O custo do otimismo é o processamento. Sob alta disputa, dezenas de threads releem e retentam simultaneamente, multiplicando a carga no banco. Nunca faça *retry* sem teto e sem reler o dado atualizado.

Bloqueio Pessimista:
A Abóbada de Aço


```
BEGIN;
SELECT saldo FROM conta WHERE id = 1 FOR UPDATE;
-- A linha está fisicamente travada no disco
UPDATE conta SET saldo = saldo - 10 WHERE id = 1;
COMMIT; -- A trava só sai aqui
```

Prevenção no nível do banco. A segunda transação fica congelada. Quando a primeira faz COMMIT, a segunda acorda e lê a versão já atualizada. Não há lost update, mas há fila.

O Perigo da Trava: Filas e Timeouts

Nunca chame um serviço externo com uma trava na mão.

Toolbox

1	`NOWAIT`

Falha imediatamente se estiver travado. (Bom quando esperar é pior que desistir).

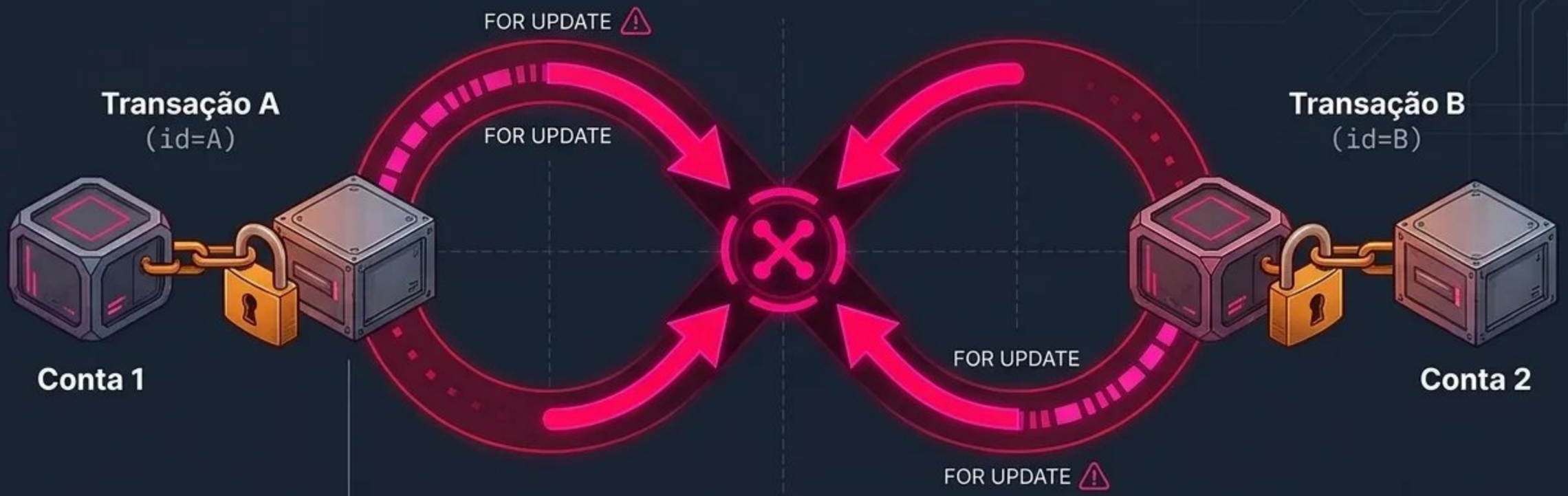
2	`SKIP LOCKED`

Pula linhas travadas. (Perfeito para filas de trabalho/workers paralelos).

3	`lock_timeout`

O limite de tempo (ex: **SET LOCAL lock_timeout = '3s'**). Sem isso, a espera no banco é indefinida.

O Colapso: O Nó Cego do Deadlock

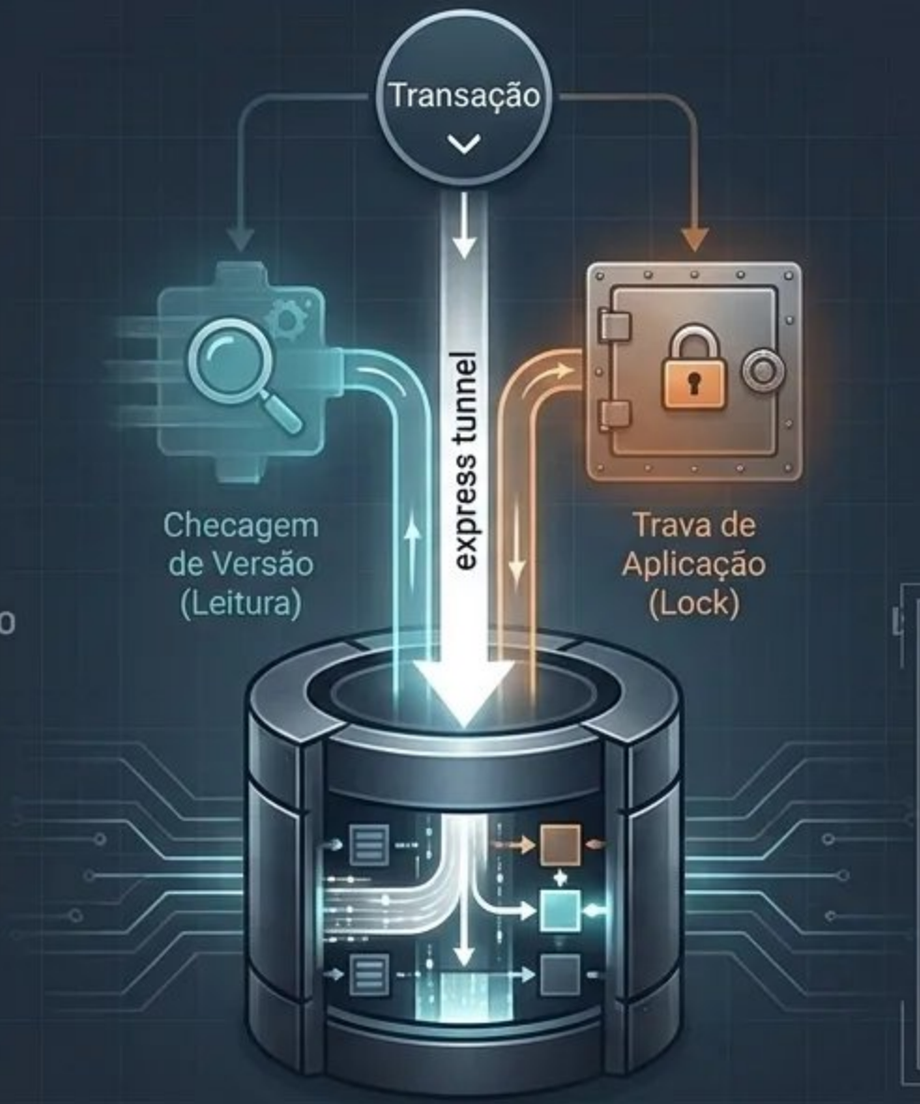


Duas transações travam as mesmas **linhas** em **ordem invertida**. O PostgreSQL detecta o **ciclo** após 1 segundo (deadlock_timeout) e aborta uma delas.

A prevenção definitiva não requer configurações complexas. Basta travar entidades sempre na **mesma ordem** (ex: **ordem alfabética** ou **numérica** de IDs).

O Truque de Mestre: O UPDATE Condicional

```
UPDATE conta
  SET saldo = saldo - 10
  WHERE id = 1 AND saldo >= 10;
-- O banco faz a conta e o filtro
de uma vez só
```



Muita coisa que parece precisar de trava não precisa.

Não há leitura prévia na aplicação. Logo, não há janela de tempo.

Logo, não há Lost Update.

Se dois débitos chegam juntos, o segundo espera o primeiro e reavalia o WHERE sobre o saldo já atualizado.

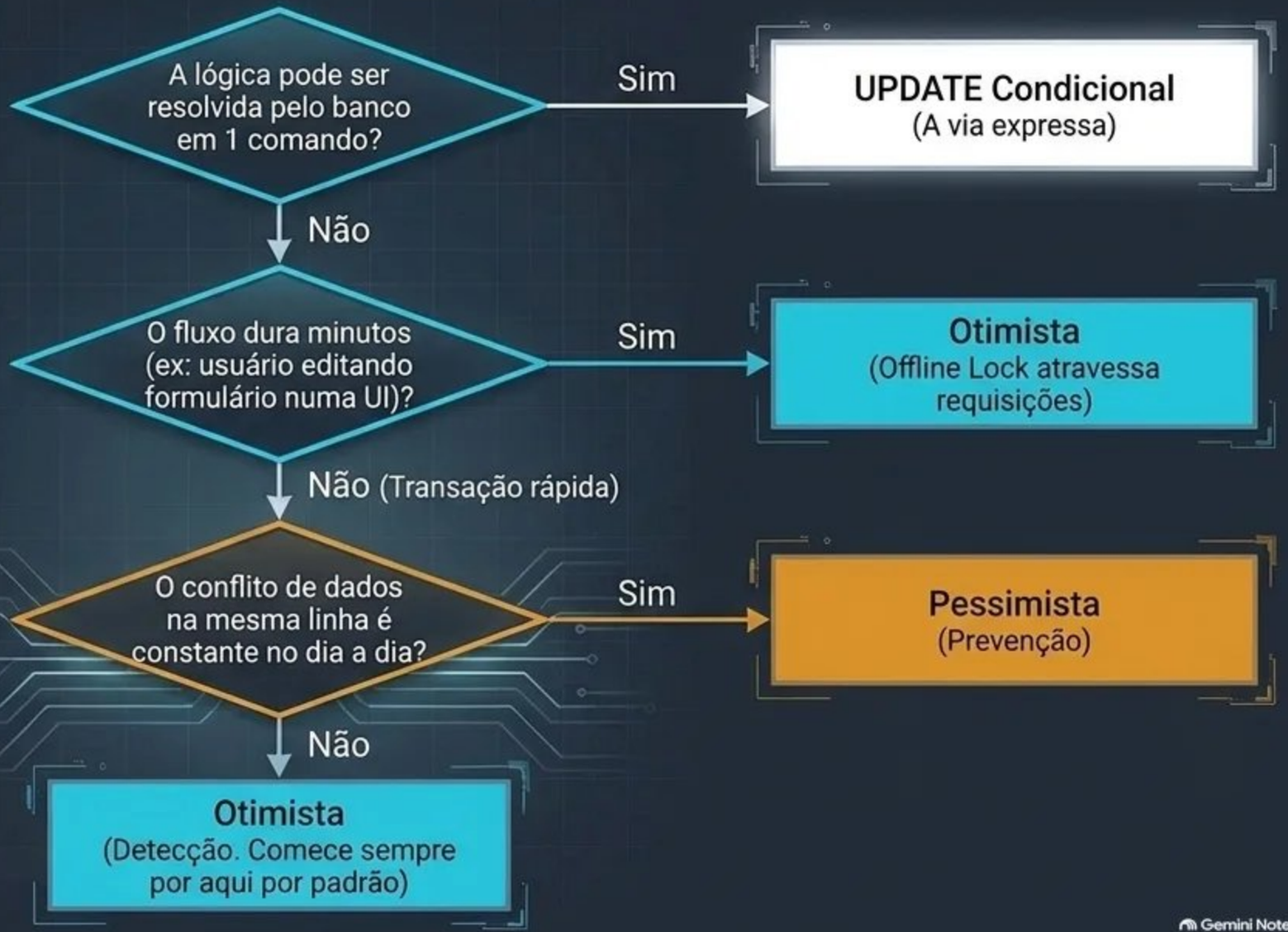
A primeira pergunta de arquitetura: Dá para escrever isso como um único comando SQL?

Matriz de Diagnóstico e Escolha

	Otimista	Pessimista
Custo no Caso Normal	Nenhum. Ninguém espera.	Espera de quem chega depois.
Custo Sob Disputa (Alta Carga)	Retentativas (CPU alta, risco de avalanche).	Filas, timeouts e risco de deadlock.
Quem Trata a Falha?	A aplicação (relendo e tentando de novo).	Ninguém no fluxo normal (mas timeouts viram exceções).
Caso de Uso Ideal	Conflito raro, leitura pesada, edição humana via UI.	Conflito frequente, operação crítica automatizada, transações de milissegundos.

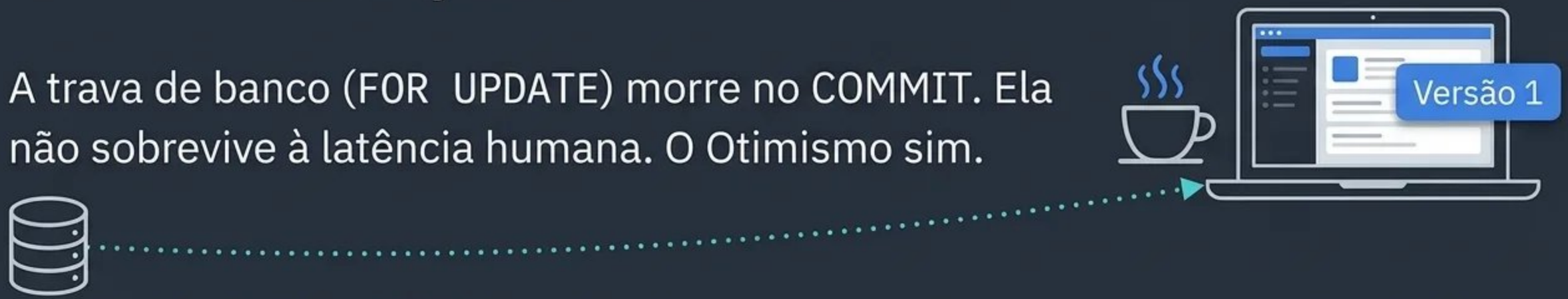
A Árvore de Decisão do Arquiteto

Comece otimista por padrão. Apele ao pessimista sob pressão.



Além da Transação: Os Padrões Offline

A trava de banco (`FOR UPDATE`) morre no `COMMIT`. Ela não sobrevive à latência humana. O Otimismo sim.



- **Optimistic Offline Lock:** A proteção viaja na payload da requisição, muito além da transação do banco de dados.
- **Web Nativa:** É por isso que HTTP possui ETags (If-Match para prevenir lost updates, RFC 9110).
- **NoSQL:** O Apache CouchDB utiliza `_rev` nativamente, devolvendo erro `409 Conflict` em revisões defasadas.

O Checklist Sênior (Perguntas de Code Review)

```
> Quem trata a exceção gerada pela anotação @Version e o que o cliente vê?

> Como sua monitoria detecta uma avalanche de retentativas otimistas antes
do banco cair?

> Onde está configurado o limite de tempo (lock_timeout) do seu FOR UPDATE?

> Suas transações pessimistas travam as entidades sempre em uma ordem
previsível?
```

Concorrência não é adivinhação. É a medição implacável do comportamento do sistema sob carga. Escolha o seu atrito.