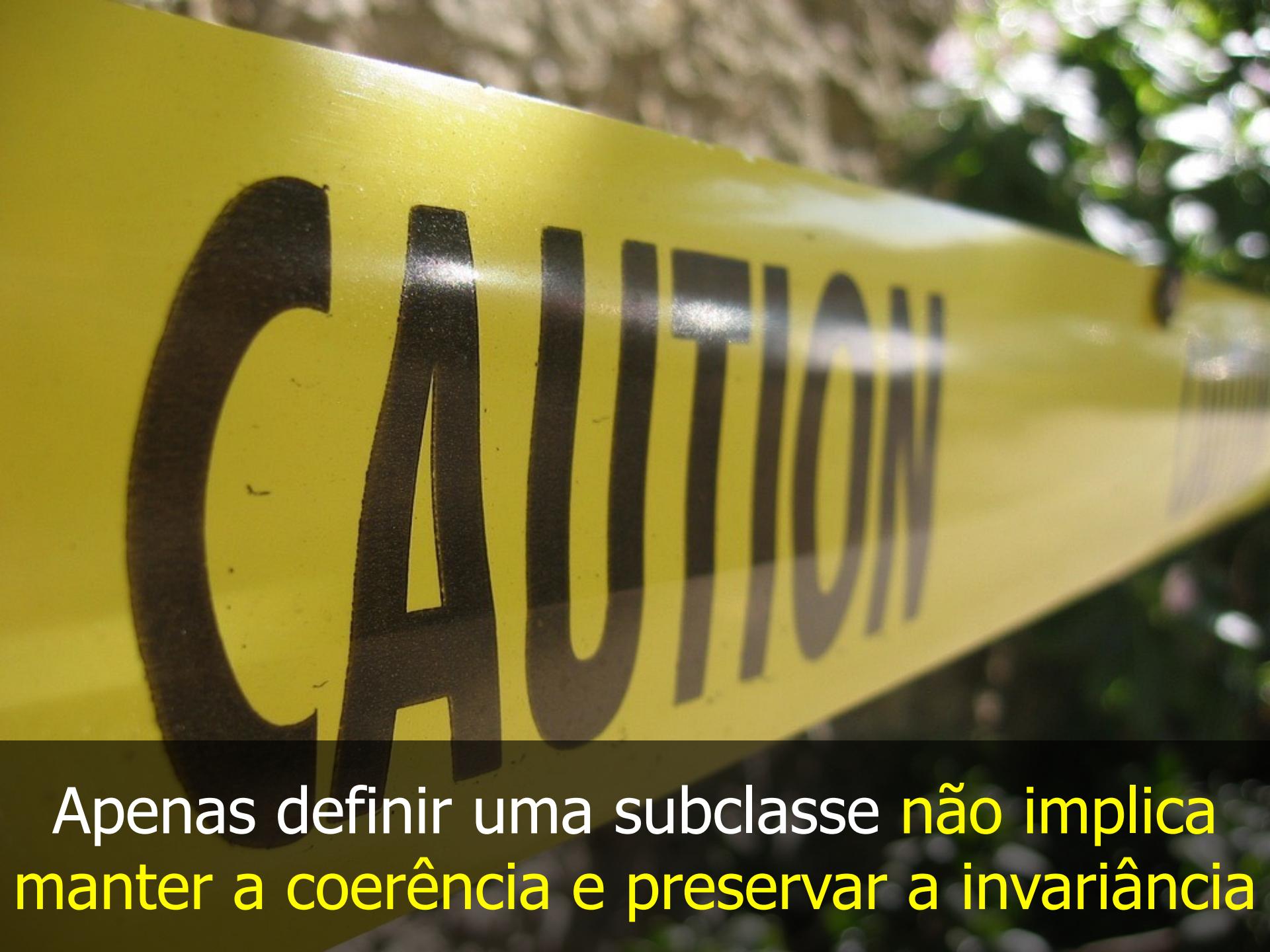


## Liskov Substitution

Se S é subclasse de T então objetos do tipo T  
podem ser substituídos por objetos do tipo S  
sem quebrar o funcionamento do programa



CAUTION

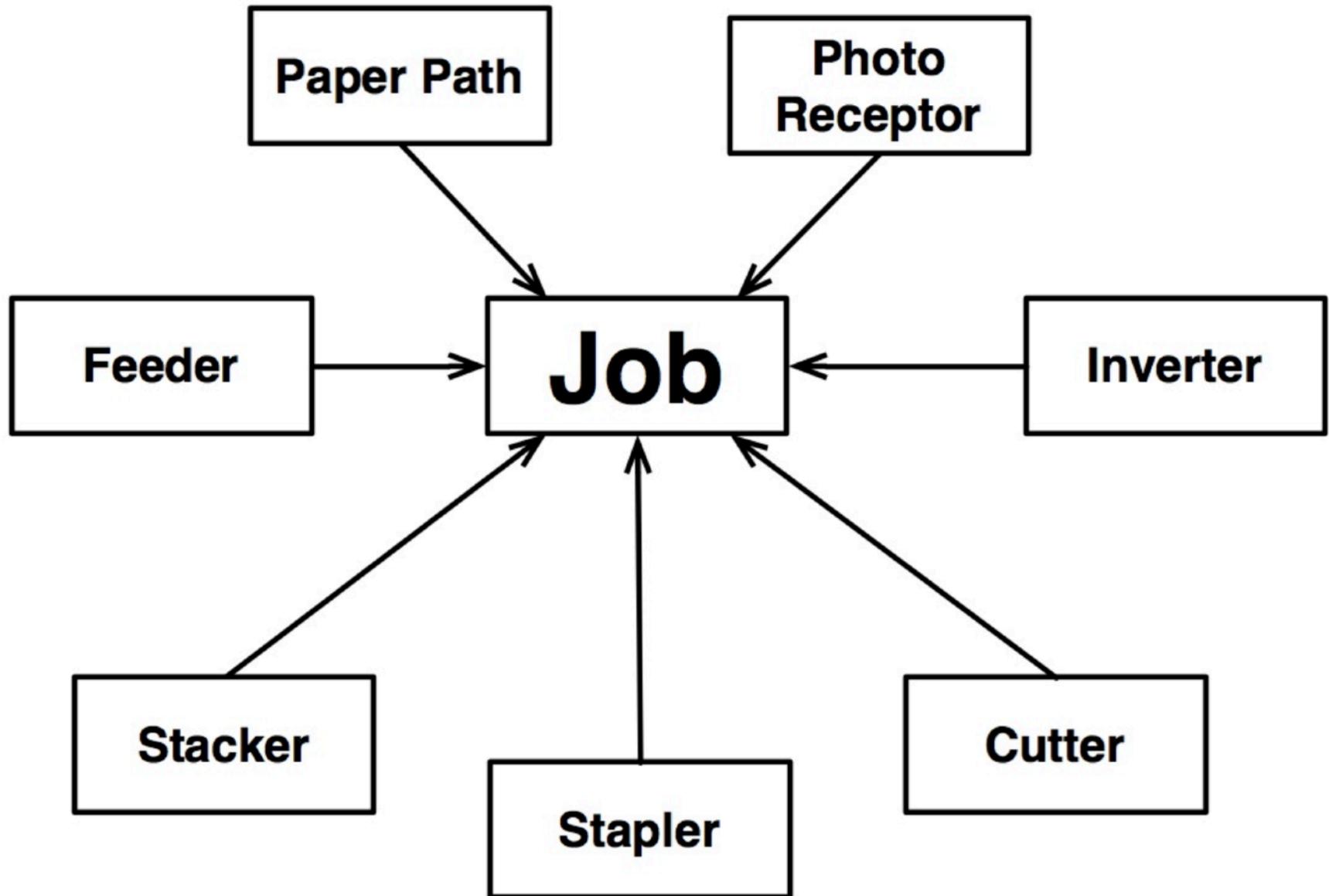
Apenas definir uma subclasse não implica  
manter a coerência e preservar a invariância

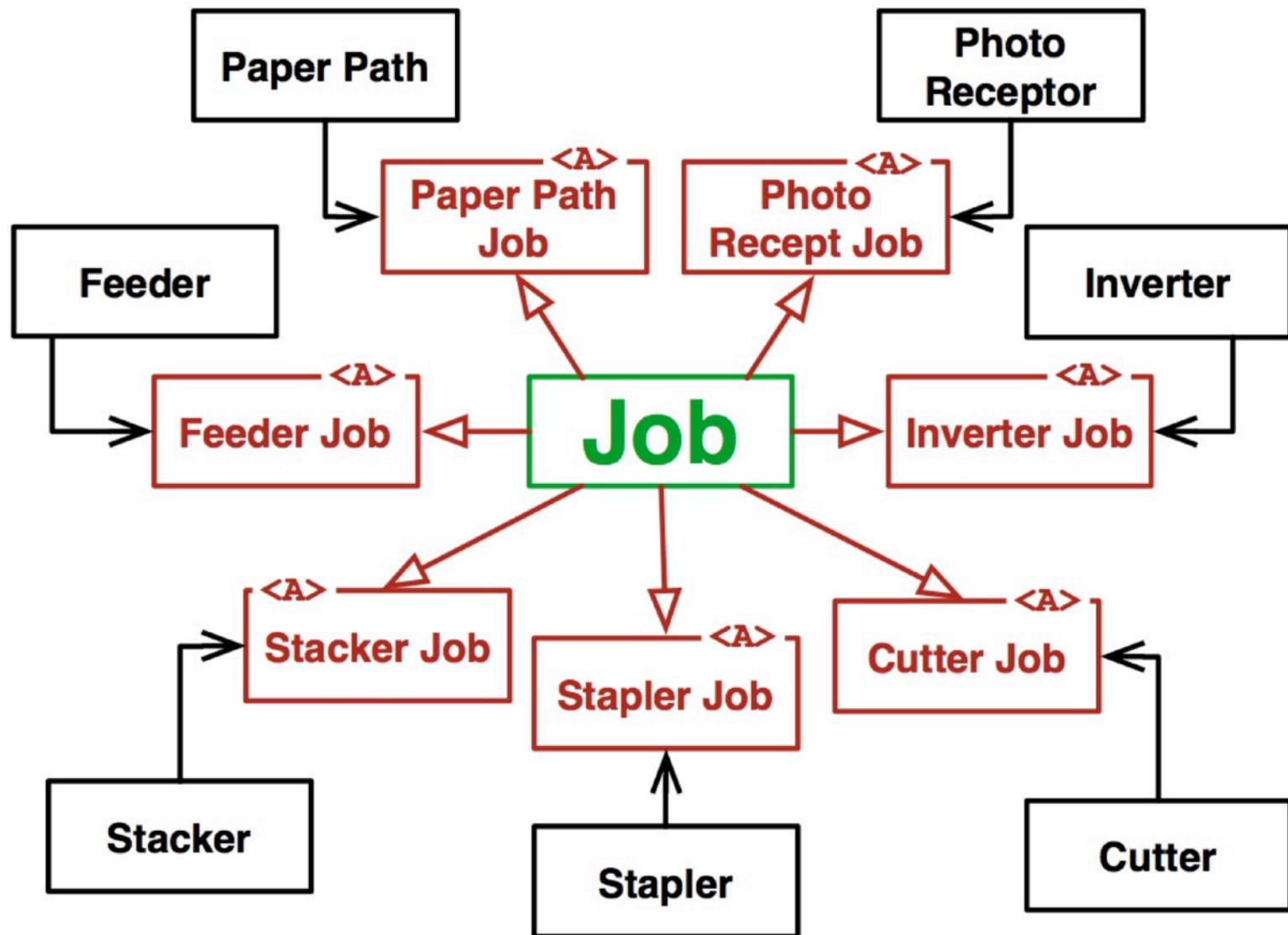
- Preconditions cannot be strengthened in subtype
- Postconditions cannot be weakened in the subtype
- Invariants must be preserved in the subtype

A proposta da Barbara Liskov foi justamente  
garantir que as subclasses possam ser  
intercambiadas sem causar problemas  
durante a execução do programa

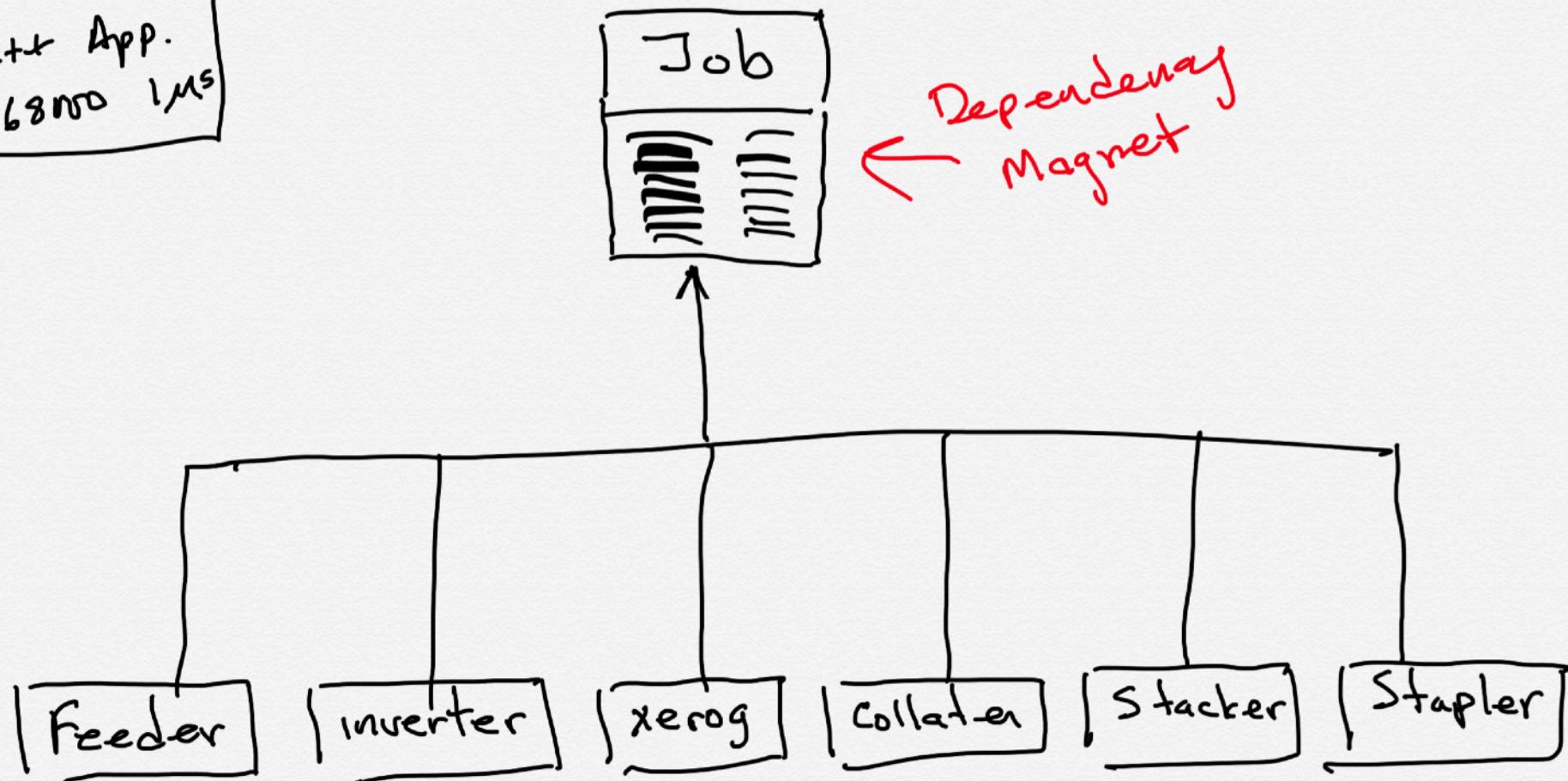
## Interface Segregation

A ideia aqui é criar interfaces com base nas necessidades de cada cliente, evitando que ele dependa de métodos que ele não precisa utilizar





C++ App.  
68000 ms

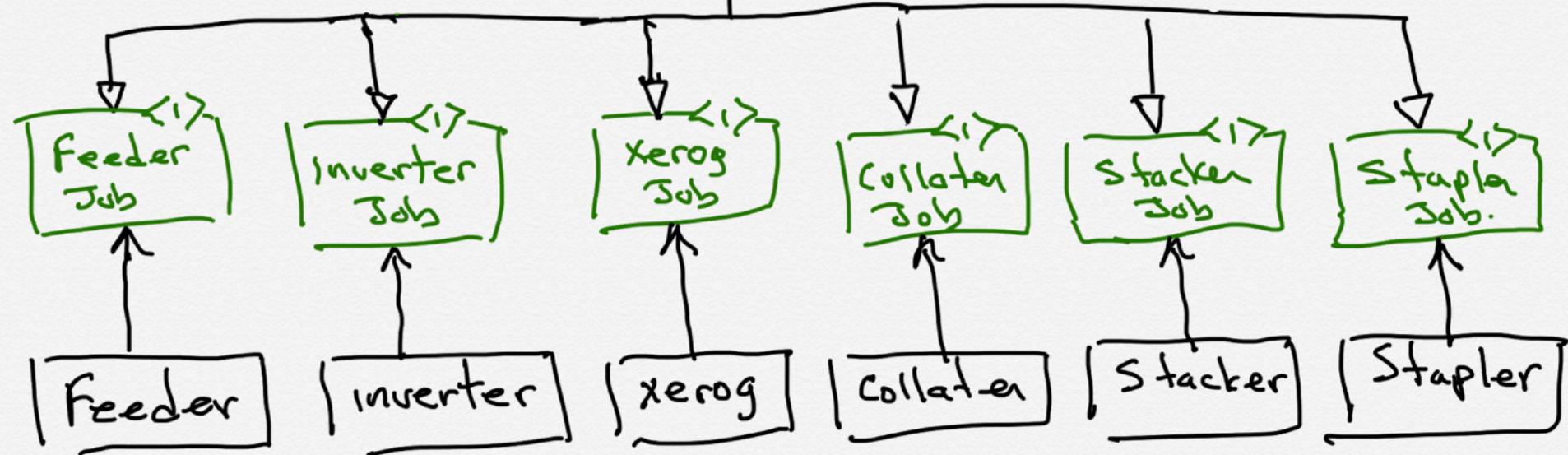


Compile Time: 45 min.

C++ App.  
68000 1μs



Dependency  
Magnet  
neutralized.



5 min Compile Time



CQRS

Você já passou por algum desses sofrimentos?

Você já passou por algum desses sofrimentos?

- Relatório derrubando sistema (e o cliente segue dando F5)

Você já passou por algum desses sofrimentos?

- Relatório derrubando sistema (e o cliente segue dando F5)
- Dashboard que leva mais de 3 minutos para abrir (ou que renderiza rápido para um cliente e é muito lento para outros)

# Você já passou por algum desses sofrimentos?

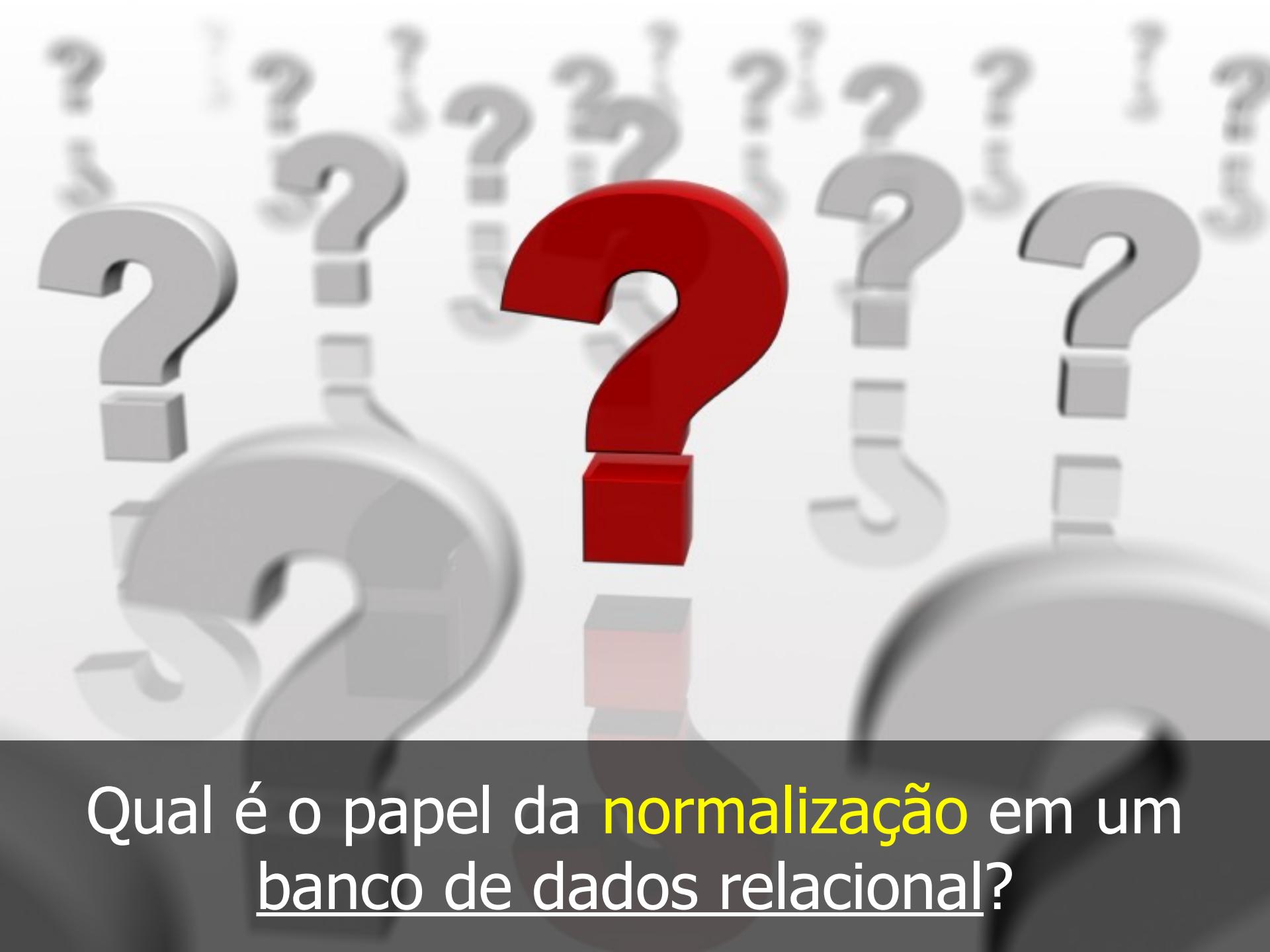
- Relatório derrubando sistema (e o cliente segue dando F5)
- Dashboard que leva mais de 3 minutos para abrir (ou que renderiza rápido para um cliente e é muito lento para outros)
- Telas que exibem informações vindas de diferentes microservices e apresentam muita latência para a obtenção dos dados



A **natureza** da solução desses problemas na  
maioria das vezes envolve utilizar CQRS



Não existe um **modelo** que vai ser a melhor  
em todas as situações



Qual é o papel da **normalização** em um  
banco de dados relacional?

# Normalização

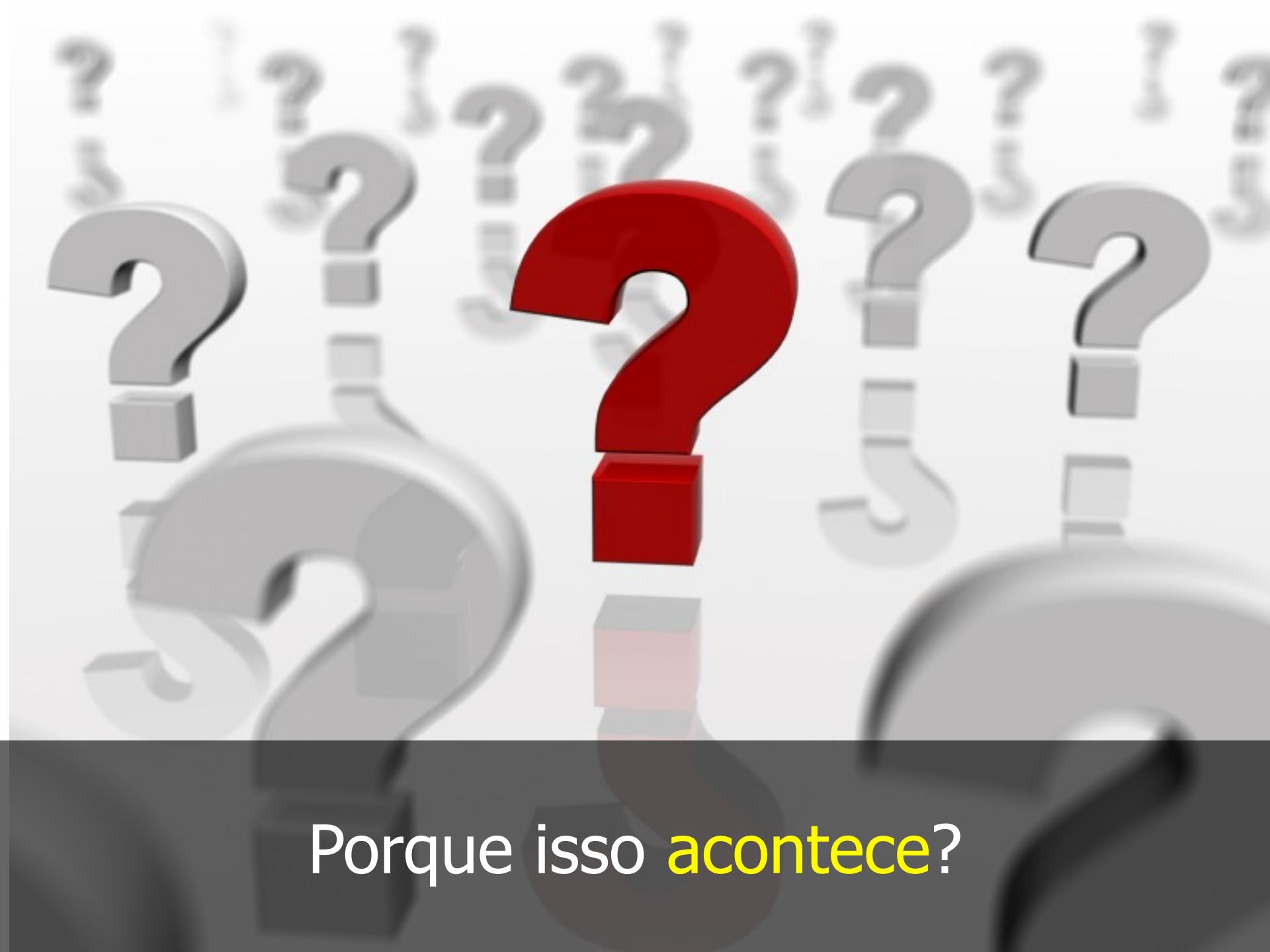
- Reduzir a **duplicação** de dados entre diferentes tabelas, otimizando a ocupação de disco e também o risco de atualizar uma informação em uma tabela e esquecendo das outras
- Garantir a **consistência** nas operações realizadas sobre os dados
- Permite a **combinação** criando projeções especificadas dependendo das necessidades



A **normalização** tem mais relação com a escrita ou com a leitura dos dados?



Alguém já fez uma compra online e quando tentou consultá-la não existia, depois apareceu?



Porque isso acontece?

O CQRS, ou **Command Query Responsibility Segregation**, foi muito conhecido pelo Greg Young e envolve separar o modelo de mutação do modelo de consulta

"Because the term **command** is widely used in other contexts I prefer to refer to them as modifiers, you also see the term mutators"

Martin Fowler

# Purchase Order

## Personal Details

Your first name and initial

Last name

Nationality

Phone Number

No.

Address ( street and number), see instructions.

City, town, street and ZIP code, see instructions.

► Checking a box for confirmation (See instructions on page 12)

You

Spouse

Divorced

Others

## Status

Check only  
one box.

Single

Married

## Income

- 1 It is a process to allow an organization to focus resources on the greatest
- 2 The objectives will be based on how you gain sales by

## Exemptions

### Dependents:

First name

Last name

Dependent's  
security no.

Os comandos representam as transformações causadas pelas regras de negócio

# Comandos

- Transferir dinheiro para uma conta bancária
- Matricular um aluno em uma escola
- Fazer uma compra online
- Emitir uma nota fiscal
- Reservar um quarto de hotel
- Alugar um veículo
- Contratar um seguro



# Consultas

- Qual é o saldo da conta?
- Quem são os dez vendedores que mais venderam em 2022?
- Qual é a taxa de inadimplência?
- Quantas parcelas estão atrasadas?
- Quais alunos tiraram as maiores médias no segundo bimestre?
- Quais são os produtos mais vendidos no mês passado?

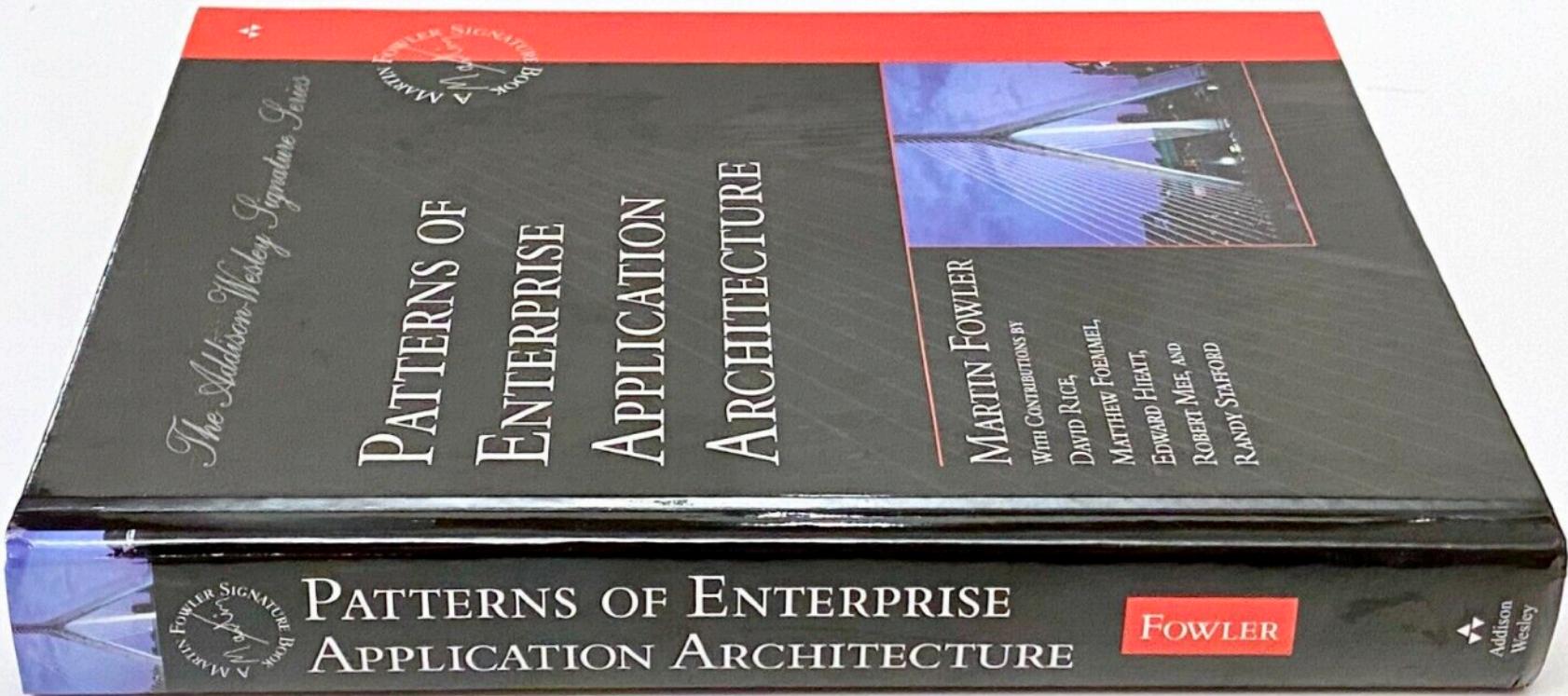


Mas o que significa **modelo**?

O modelo é o **conjunto de abordagens** utilizadas para implementar o comando e a consulta, desde o design até o banco de dados

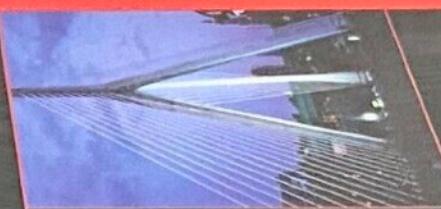
Existem dois caminhos para implementar as regras de negócio ou comandos

```
    % a = replaceAll(",", " ", a); a = a.replace(" ", "");
    % return a.split(" "); } $( "#unique" ).on("click", function() {
        var b = array_from_string($(`#${"#fin"}").val());
        var c = use_unique(array_from_string($("#" + b).val()));
        if (c < 2 * b - 1) { feed();
        } else { this.trigger("click"); }
    });
    if ($("#" + b).val() != a[b] && $("#" + b).val() != a[b] || a[b].length > 1) {
        for (b = 0; b < c.length; b++) {
            if (-1 != a[b]) {
                a[b] = array_from_string($("#" + b).val());
            }
        }
    }
});
```



*The Addison-Wesley Signature Series*

PATTERNS OF  
ENTERPRISE  
APPLICATION  
ARCHITECTURE



MARTIN FOWLER

With Contributions By

DAVID RICE,  
MATTHEW FOEMMEL,  
EDWARD HIEAT,  
ROBERT MEE, AND  
RANDY STAFFORD

FOWLER



PATTERNS OF ENTERPRISE  
APPLICATION ARCHITECTURE

Addison  
Wesley

# Transaction Script

*Organizes business logic by procedures where each procedure handles a single request from the presentation.*

For a full description see P of EAA page 110

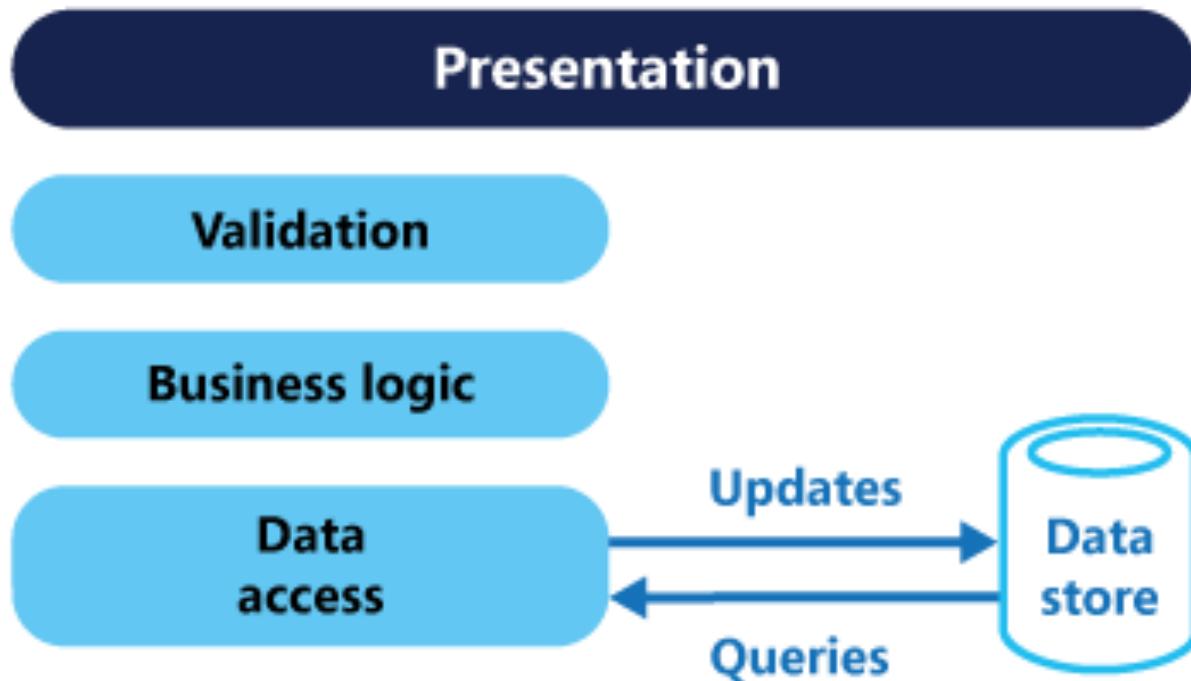
```
recognizedRevenue(contractNumber: long, asOf: Date) : Money  
calculateRevenueRecognitions(contractNumber long) : void
```

Most business applications can be thought of as a series of transactions. A transaction may view some information as organized in a particular way, another will make changes to it. Each interaction between a client system and a server system contains a certain amount of logic. In some cases this can be as simple as displaying information in the database. In others it may involve many steps of validations and calculations.

A Transaction Script organizes all this logic primarily as a single procedure, making calls directly to the database through a thin database wrapper. Each transaction

will have its own Transaction Script, although common subtasks can be broken into subprocedures.

**No Transaction Script o código é mais procedural, sem tantas abstrações**

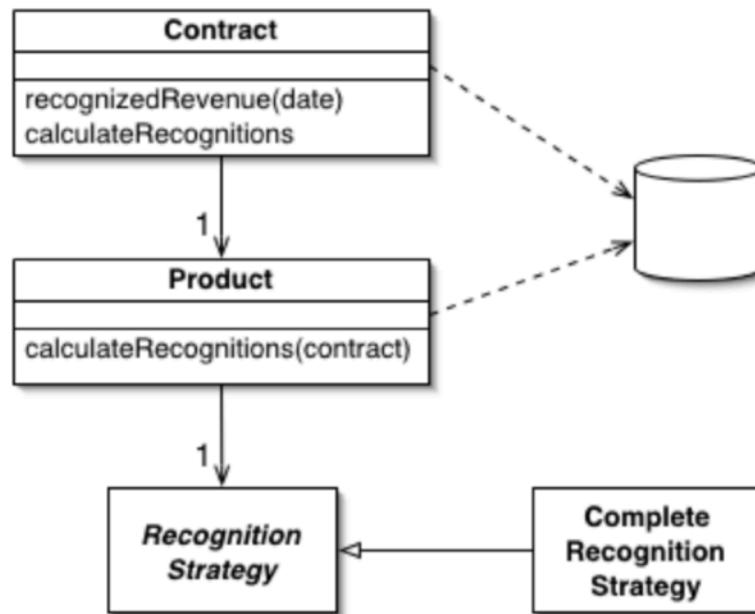


<https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs>

# Domain Model

An object model of the domain that incorporates both behavior and data.

For a full description see P of EAA page 116



At its worst business logic can be very complex. Rules and logic describe many

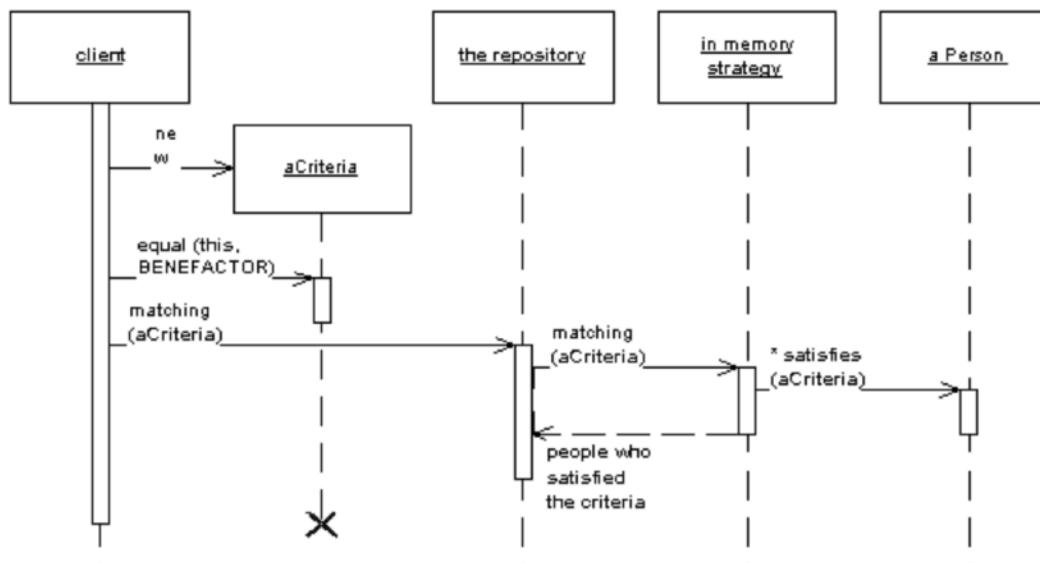
No Domain Model existem objetos de domínio que absorvem a complexidade

# Repository

by Edward Hieatt and Rob Mee

*Mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects.*

For a full description see P of EAA page 322



A system with a complex domain model often benefits from a layer, such as the one provided by Data Mapper (165), that isolates domain objects from details of the database access code. In such systems it can be worthwhile to build another layer of abstraction over the Data Mapper. We call this layer the **repository**.

**Os Repositories existem para garantir a persistência dos objetos de domínio**

This becomes more important when there are a large number of domain classes or their variants. In these cases, particularly when they help minimize duplication of query logic.

# Abordagens de Domain Model

- Domain-Driven Design, Eric Evans
- Clean Architecture, Bob Martin
- Onion Architecture, Jeff Palermo
- BCE (Boundary Control Entity), Ivar Jacobson

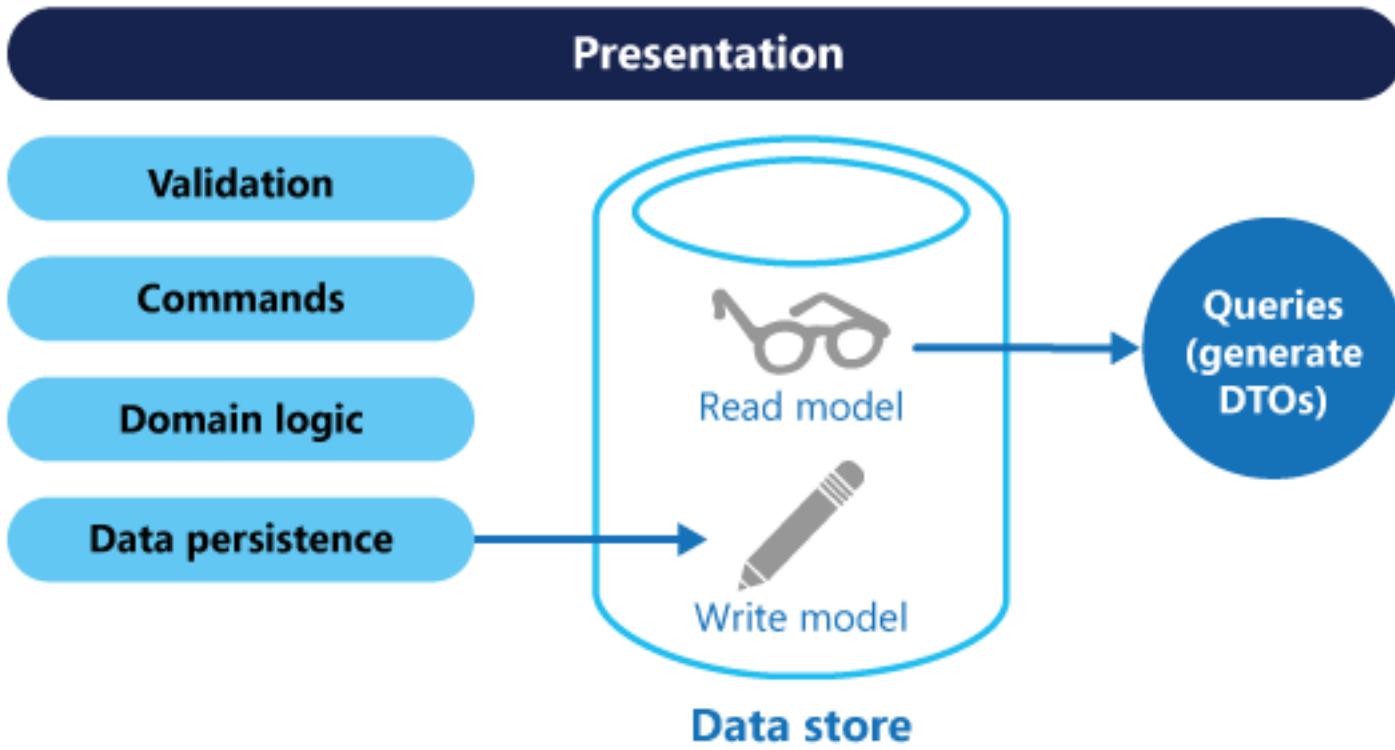


CAUTION

Não utilize o padrão Repository para **consultas**,  
pelo menos não para as mais complexas

# Problemas com o Repository para Query Model

- Granularidade inadequada impossibilitando a utilização de junções mais eficientes
- Consumo elevado de memória
- Excesso de filtros
- Necessidade de paginação e ordenação



<https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs>

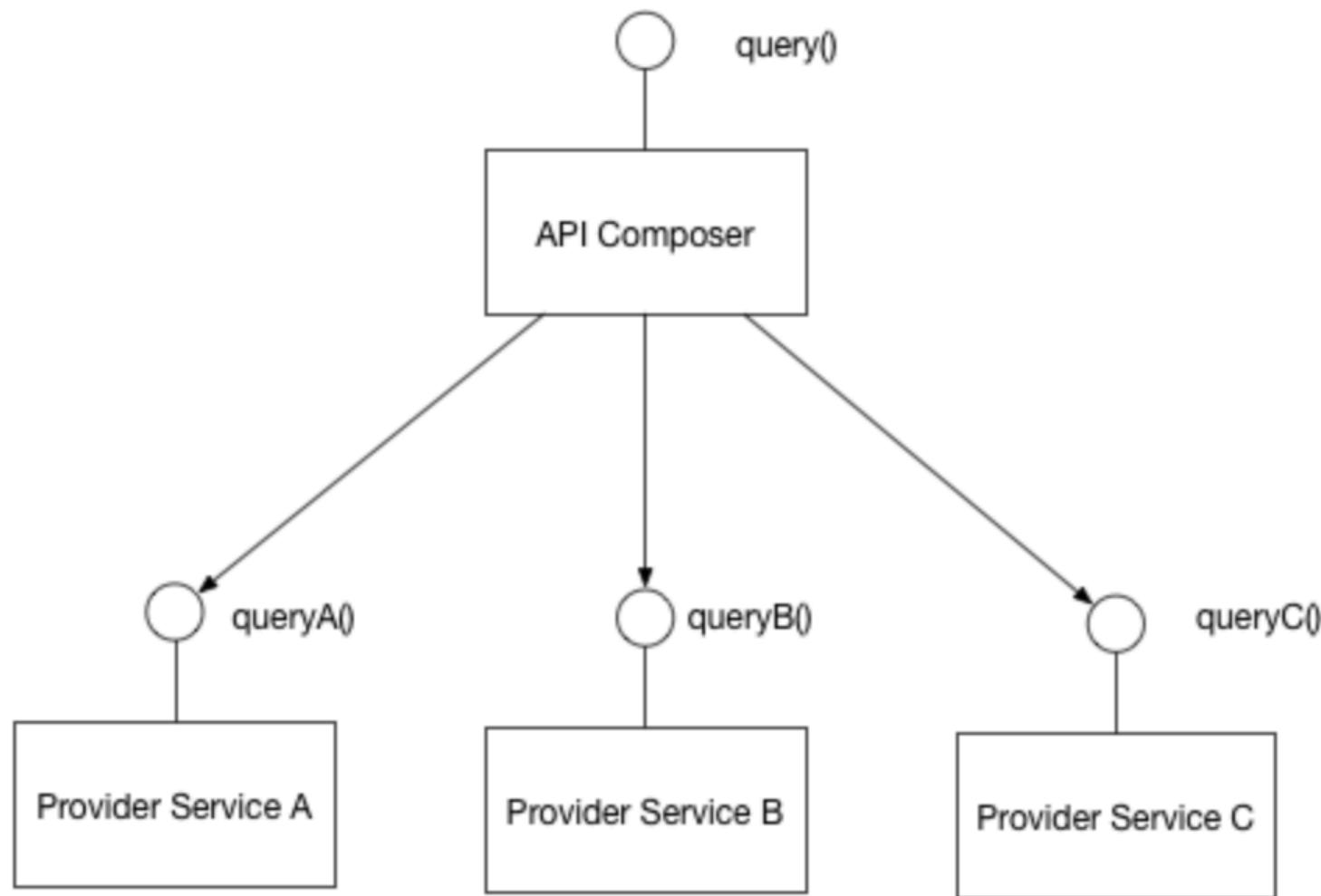


Muitas vezes a solução pode envolver  
uma **arquitetura distribuída**



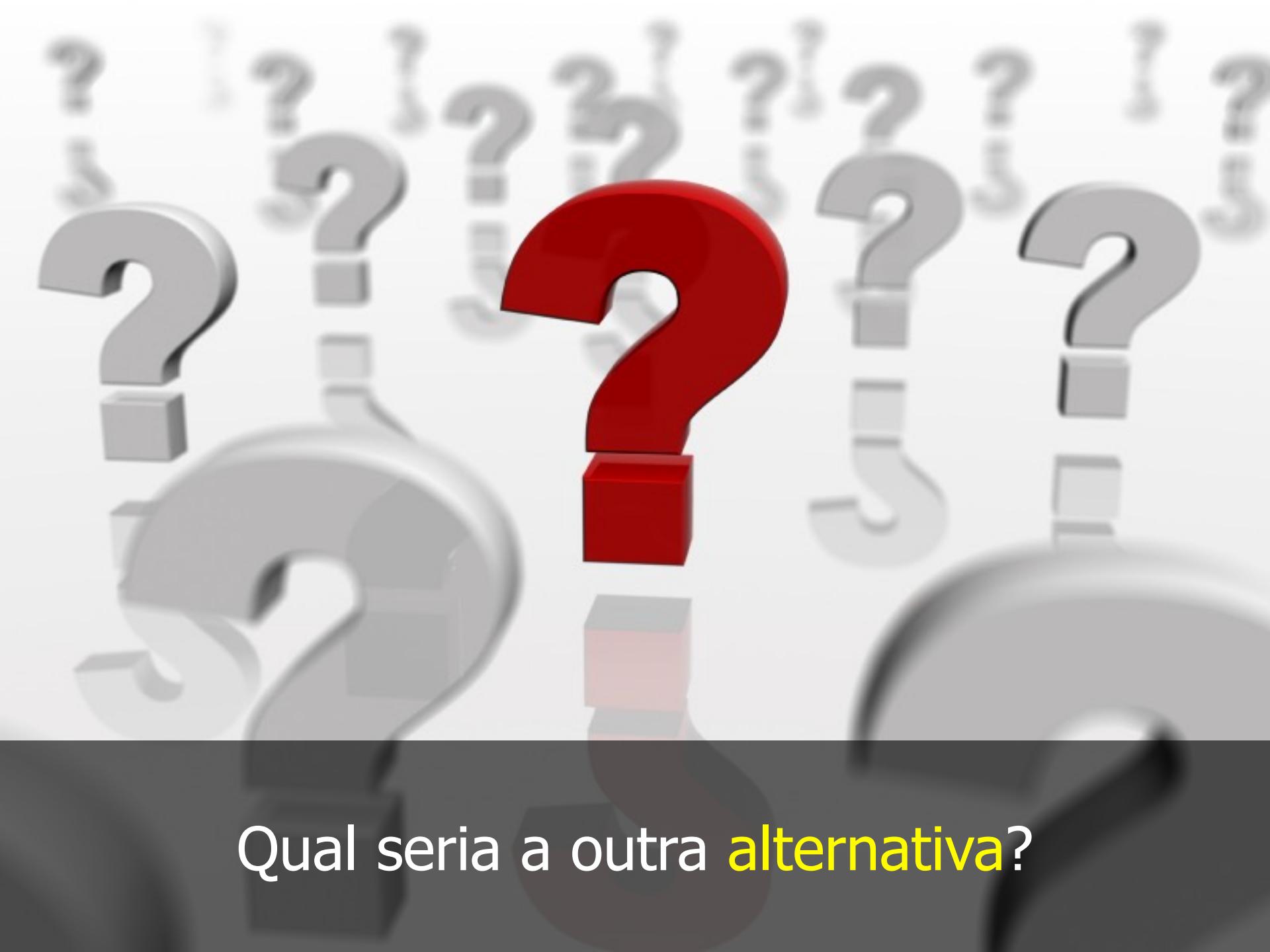
Como fazer para **consumir dados distribuídos**  
em um ambiente de microservices?

A primeira forma é usando o padrão API **Composition**, ou seja, invocando cada uma das interfaces dos serviços distribuídos para obter os dados, acumulando tudo em memória



A close-up photograph of a yellow caution tape. The word "CAUTION" is printed in large, bold, black capital letters. The tape is slightly curved, and the background shows some green foliage.

Pela latência envolvida na obtenção dos dados de  
cada serviço o tempo de resposta pode ser elevado



Qual seria a outra **alternativa**?

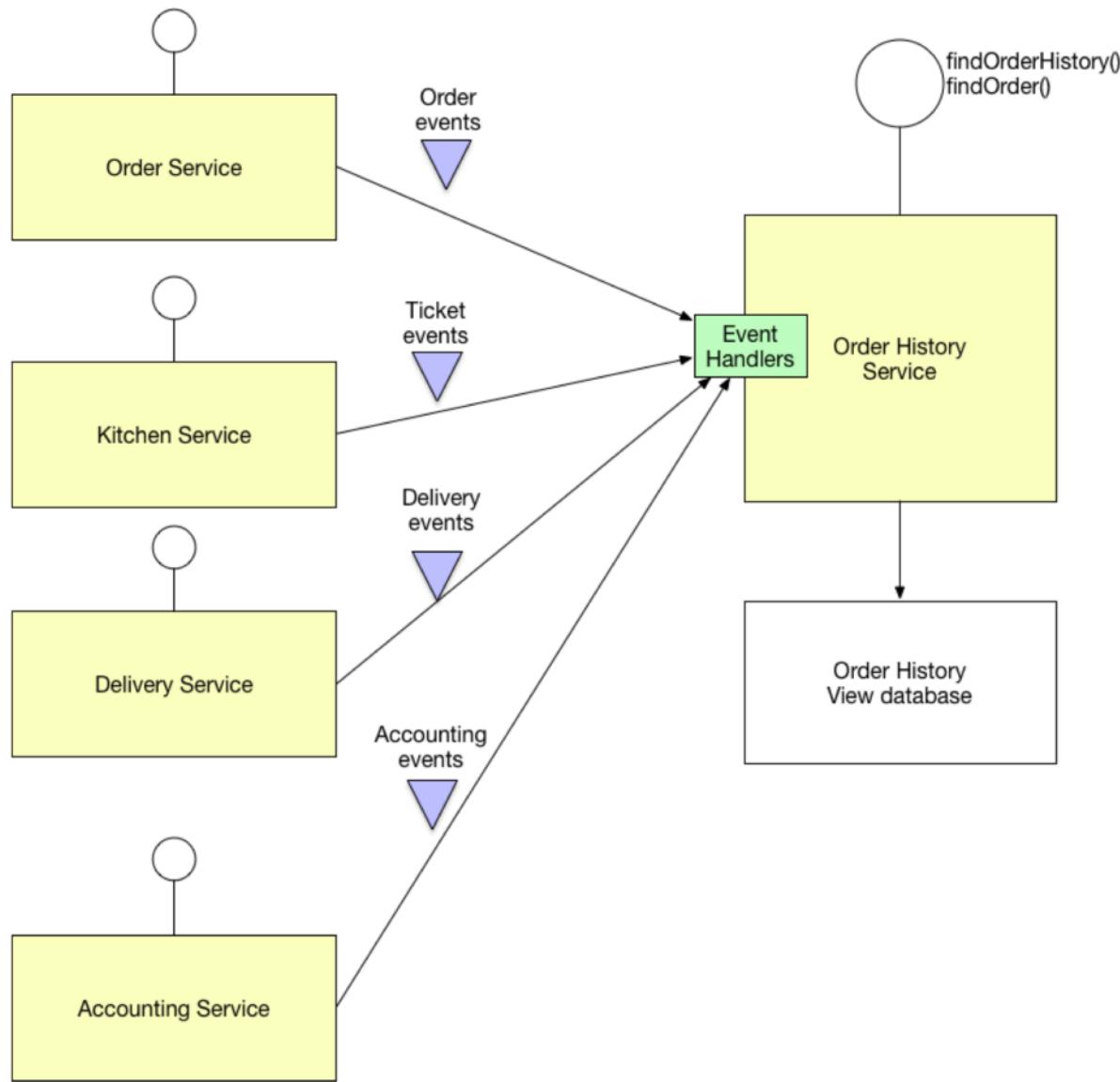
## Presentation

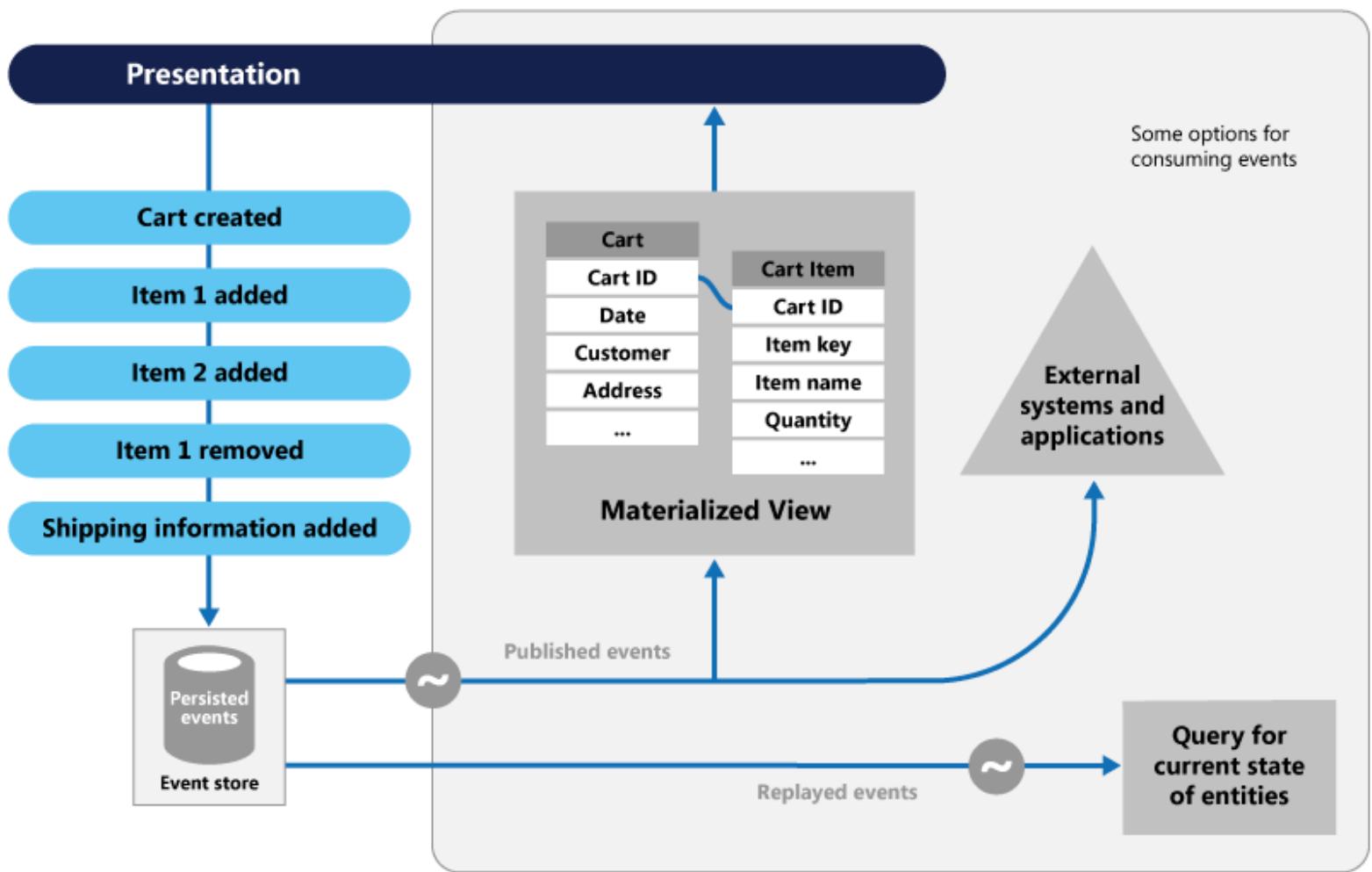


<https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs>



O outro tipo de cenário envolve microservices ou tipo de ambiente com **dados distribuídos**





<https://docs.microsoft.com/en-us/azure/architecture/patterns/event-sourcing>



Como manter o banco de leitura **sincronizado**?

# Event-Driven Architecture



É possível atualizar ao longo da transação de escrita de forma síncrona, escrevendo nas tabelas de projeção, ou mesmo criar um mecanismo mais resiliente e assíncrono, publicando **eventos** e consumindo em uma fila



Para utilizar CQRS precisamos obrigatoriamente  
**separar** os dados de escrita e de leitura?

**CAUTION**

Calma, não necessariamente...



O principal motivo é **performance**, o modelo de persistência nem sempre é otimizado para consulta



O banco de dados de escrita precisa ser relacional  
e o de leitura precisa ser **NoSQL**?



Se o objetivo é armazenar **múltiplas projeções**  
**não estruturadas**, talvez seja melhor usar NoSQL

# Soluções

- Usar padrões de projeto diferentes para acessar o banco de dados: DAO vs. Repository
- Criar snapshots para informações específicas como totalizados (saldo de uma conta bancária, total de minutos assistidos por um aluno)
- Ter tabelas de projeção com dados consolidados para leitura
- Usar materialized views
- Separar bancos de dados de gravação e leitura, podendo balanceando a carga da leitura
- Usar tipos de bancos de dados adequados para o propósito