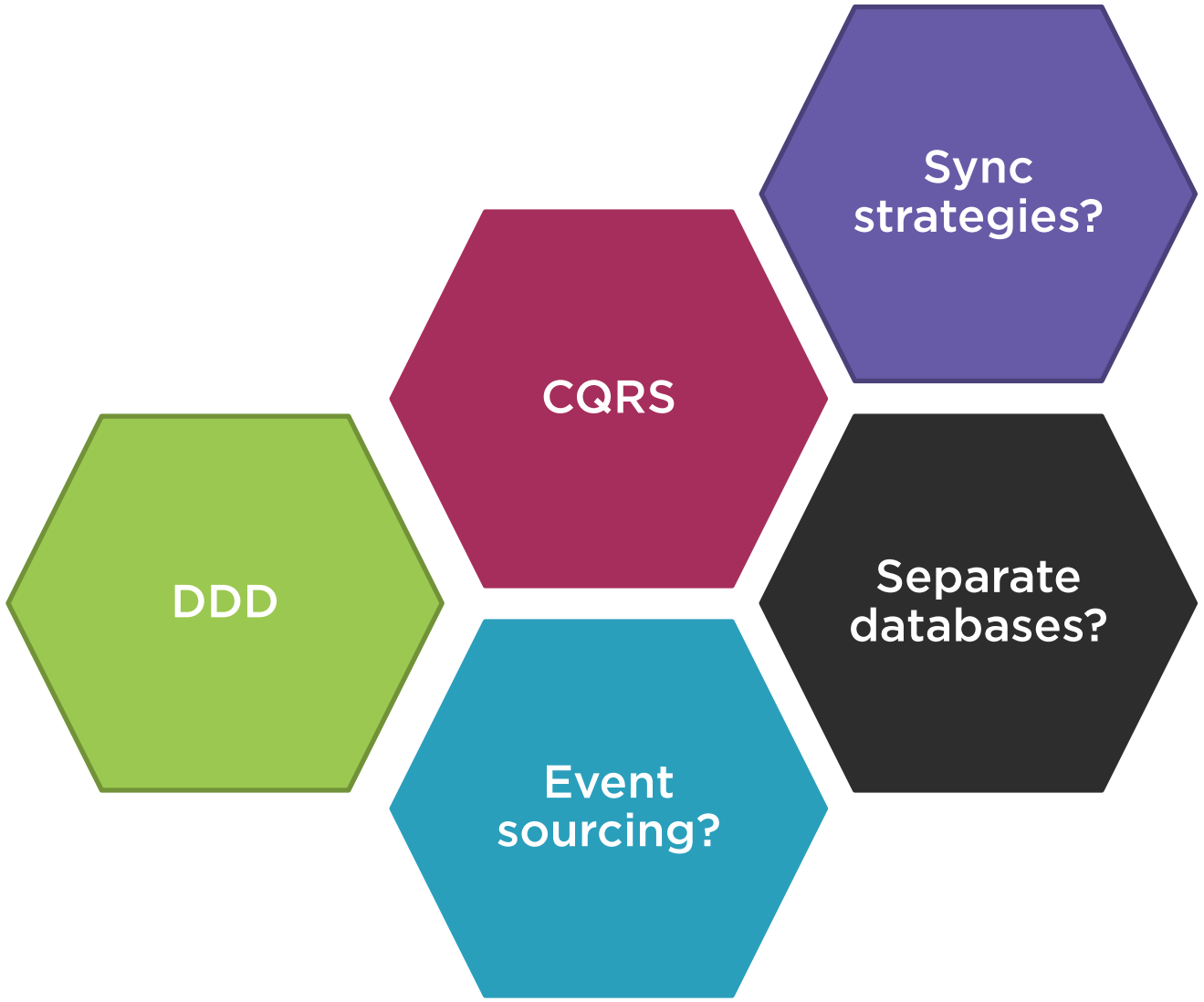
CQRS in Practice

INTRODUCTION

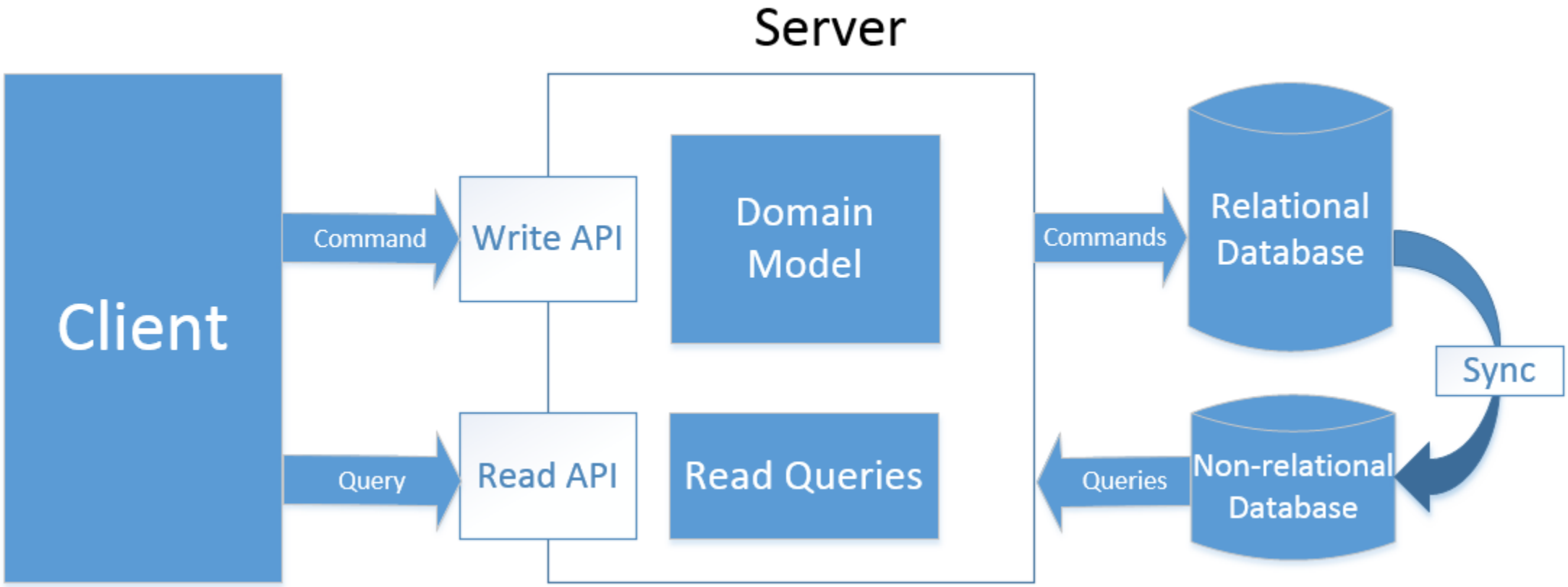
Vladimir Khorikov

@vkhorikov www.enterprisecraftsmanship.com

CQRS


Sync strategies?
CQRS
DDD
Separate databases?
Event sourcing?

CQRS


Server
Client
Command
Write API
Domain Model
Commands
Relational Database
Sync
Query
Read API
Read Queries
Queries
Non-relational Database

Overview

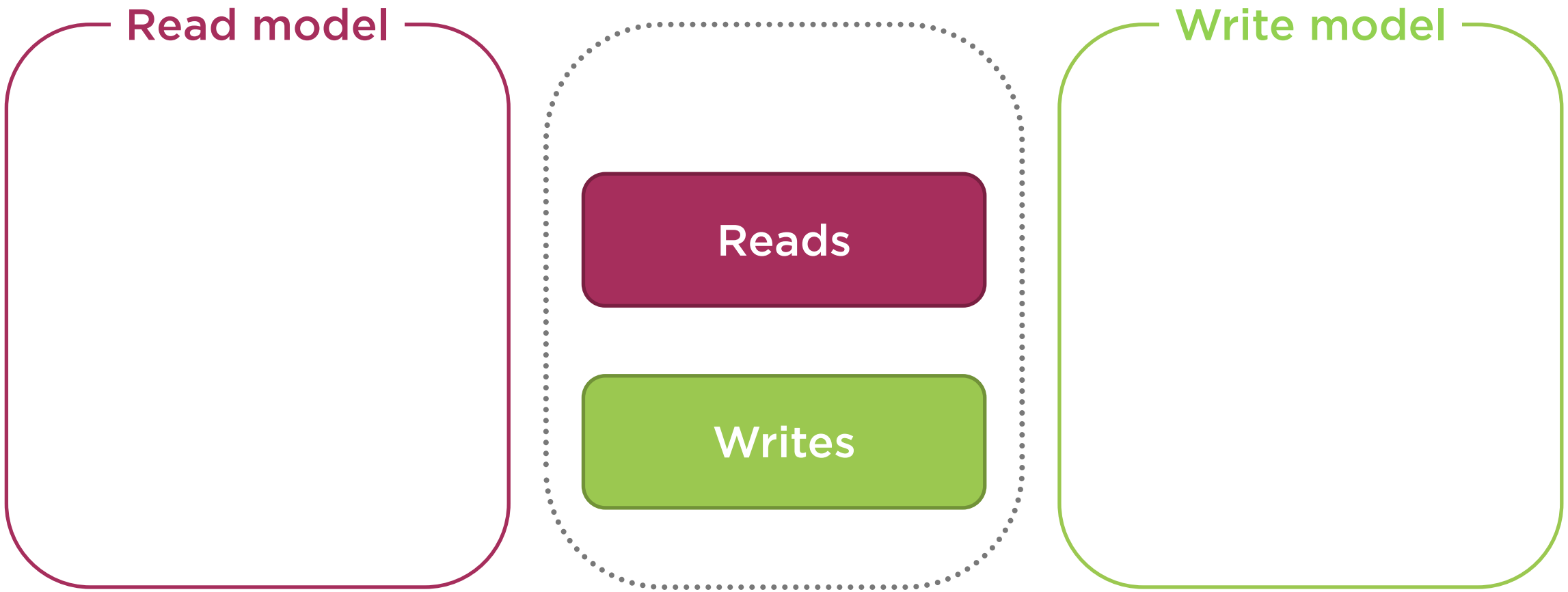
- Introduction
- Introducing a sample project
- Segregating Commands and Queries
- Refactoring Towards Task-based Interface
- Simplifying the Read Model
- Using MediatR to Implement CQRS
- Introducing a Separate Database for Queries
- Synchronizing the Commands and Queries Databases
- CQRS Best Practices and Misconceptions

Prerequisites

“Domain-Driven Design in Practice” by Vladimir Khorikov

CQRS and Its Origins

Command and Query Responsibility Segregation



CQRS and Its Origins

Command and Query Responsibility Segregation

Greg Young, 2010

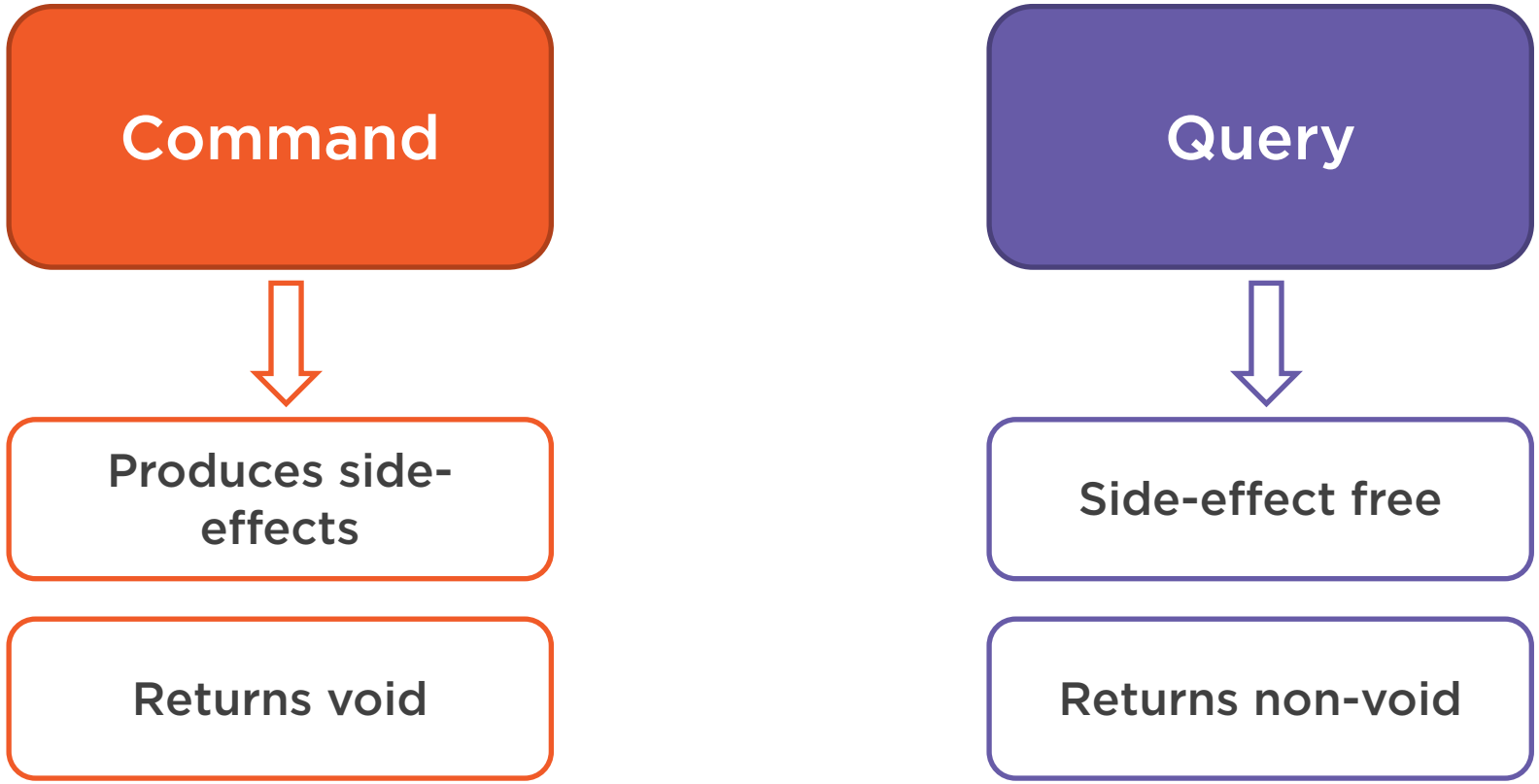
Command Query Separation

Bertrand Meyer

https://cqrs.files.wordpress.com/2010/11/cqrs_documents.pdf

CQS

Command–query Separation Principle



CQS Limitations

```
var stack = new Stack<string>();
stack.Push("value");              // Command
string value = stack.Pop();       // Both query and command
```

When result of a query can become stale quickly

CQS Limitations

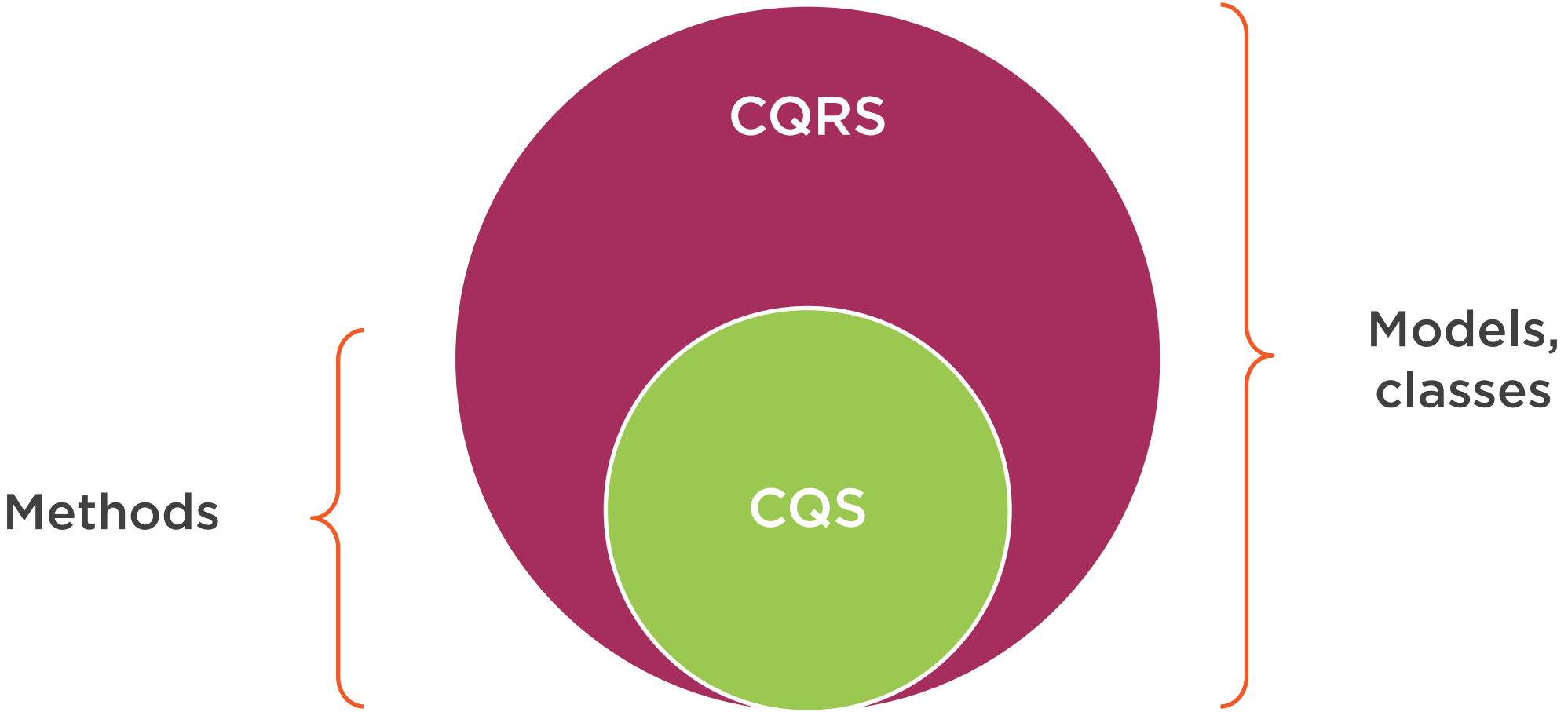
```
public bool FileExists(string path)
{
    return File.Exists(path);
}

public void WriteToFile(
    string path, string content)
{
    if (!FileExists(path))
        throw new ArgumentException();

    File.WriteAllText(path, content);
}
```

Follows CQS

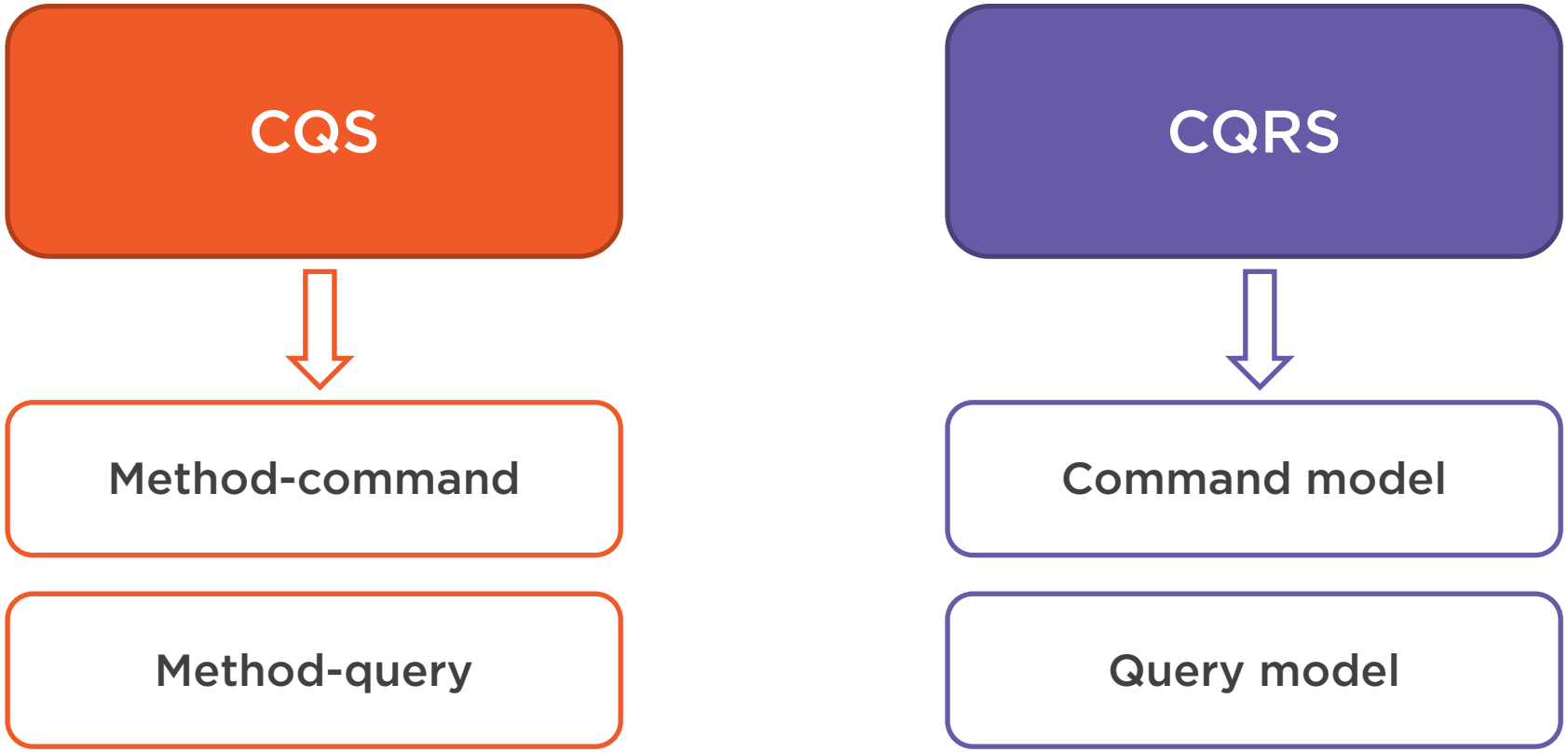
```
public Result WriteToFile(
    string path, string content)
{
    try
    {
        File.WriteAllText(path, content);
        return Result.Ok();
    }
    catch (FileNotFoundException)
    {
        return Result.Fail("File not found");
    }
}
```

Doesn't follow CQS

CQRS and CQS



CQRS and CQS


CQS
Method-command
Method-query
CQRS
Command model
Query model

Why CQRS?

Scalability

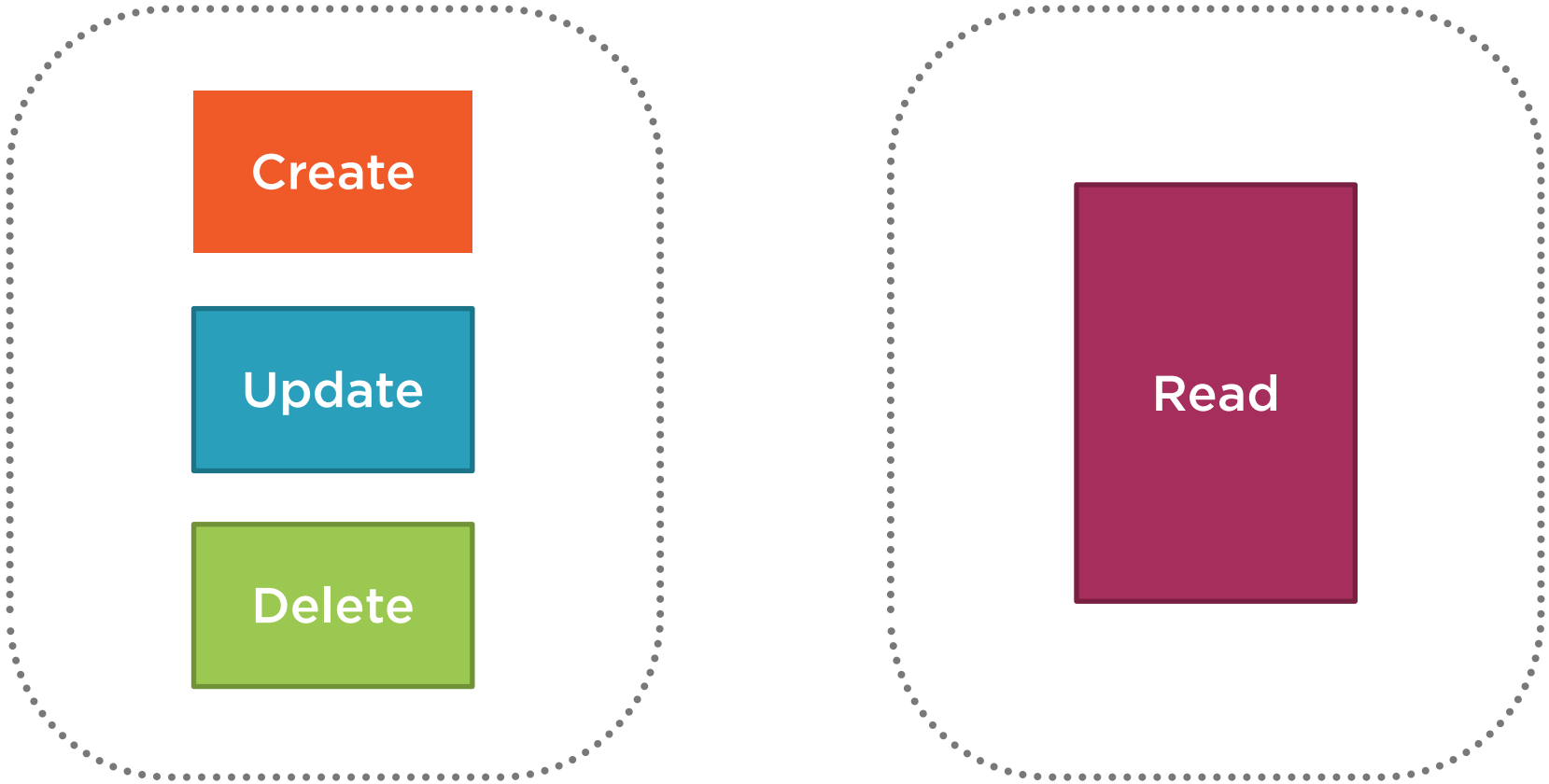
Create

Read

Update

Delete

Why CQRS?



Command side

1 server

Query side

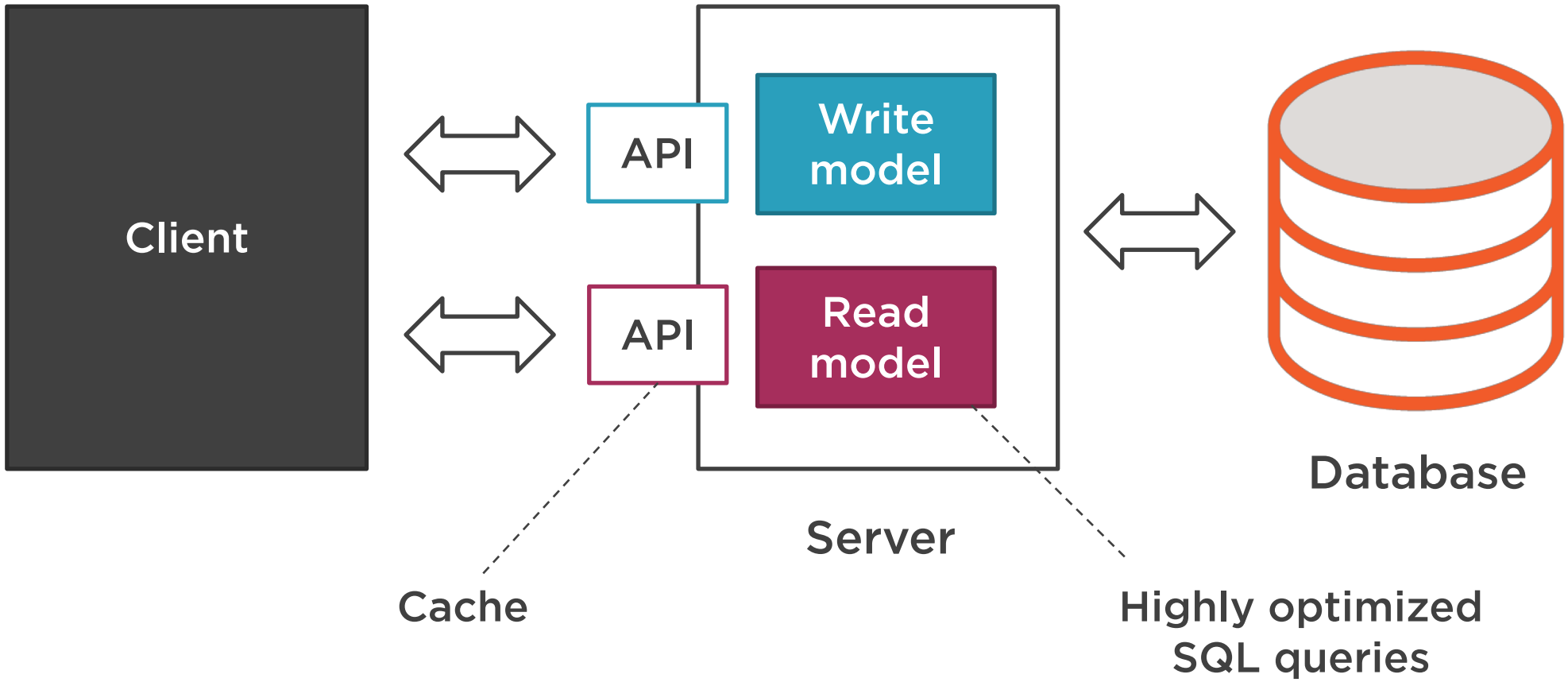
10 servers

Why CQRS?

Scalability

Performance

Why CQRS?


Client
API
Write model
API
Read model
Database
Server
Cache
Highly optimized SQL queries

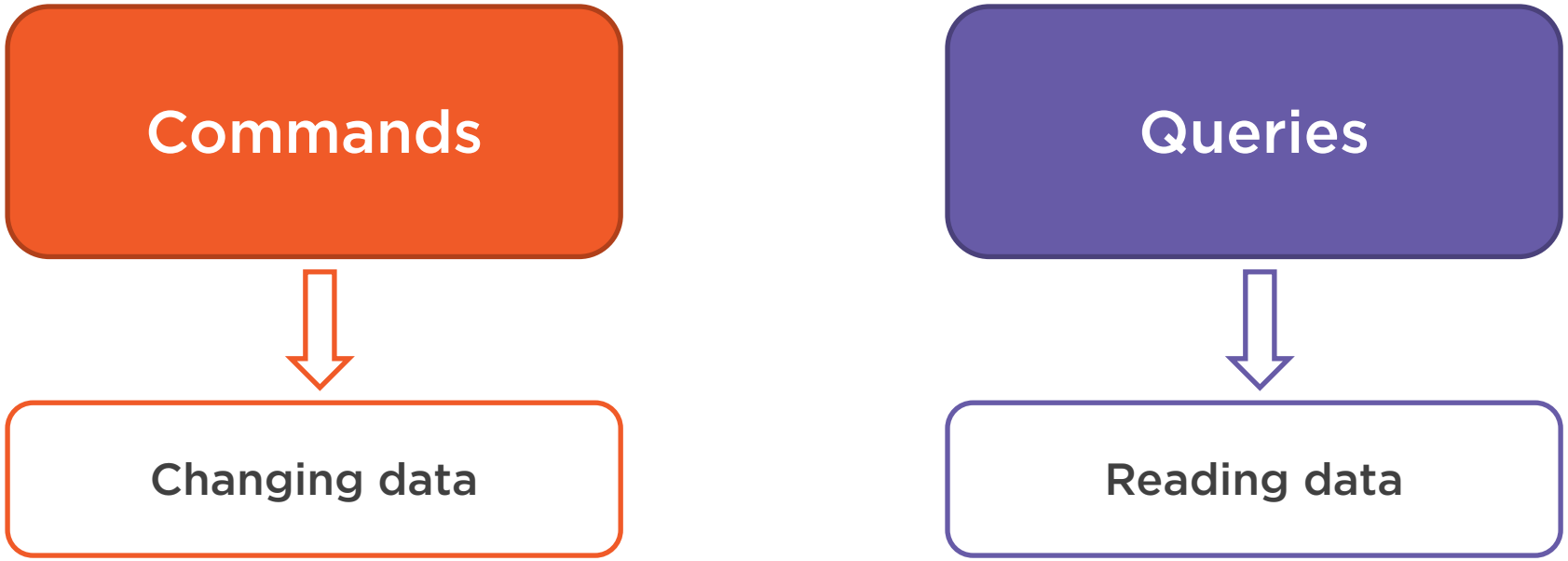
Why CQRS?

Scalability

Performance

Simplicity

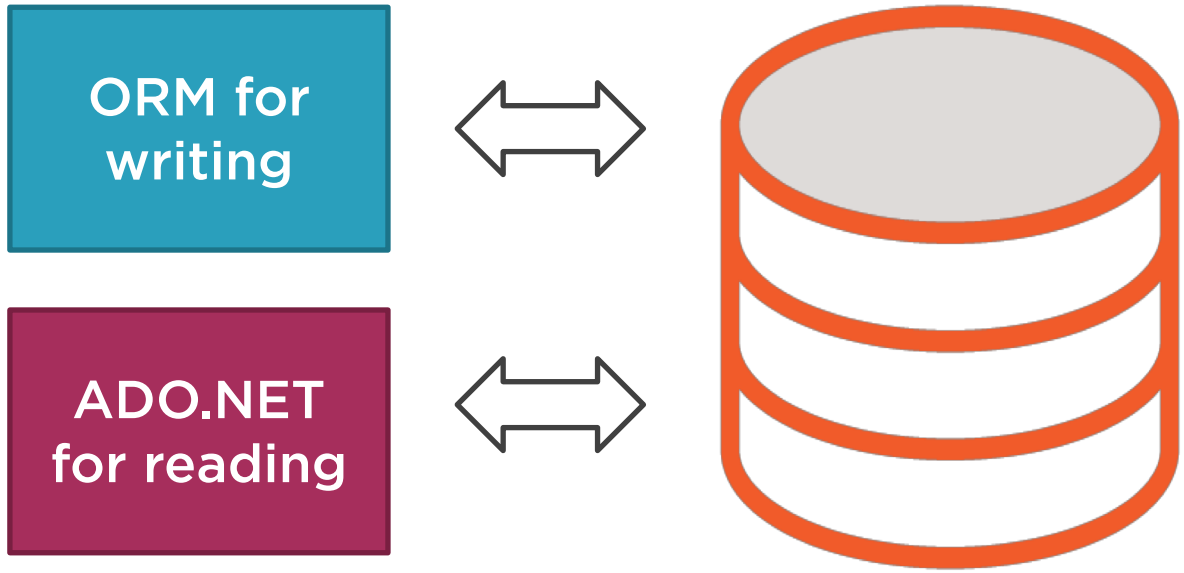
Why CQRS?



- Offloading complexity
- SRP applied at the architectural level

CQRS is about optimizing decisions for different situations.

CQRS in the Real World



CQRS

CQRS in the Real World

Summary

Command Query Responsibility Segregation (CQRS) originates from the Command Query Separation Principle (CQS)

- CQRS extends CQS to the architectural level
- Split a unified domain model into two: for commands and for queries

CQRS allows us to make different decisions for reads and writes

- Better scalability
- Better performance
- Simpler code

CQRS is SRP applied at the architectural level

Examples of CQRS in the real world

In the Next Module

Introducing a Sample Project