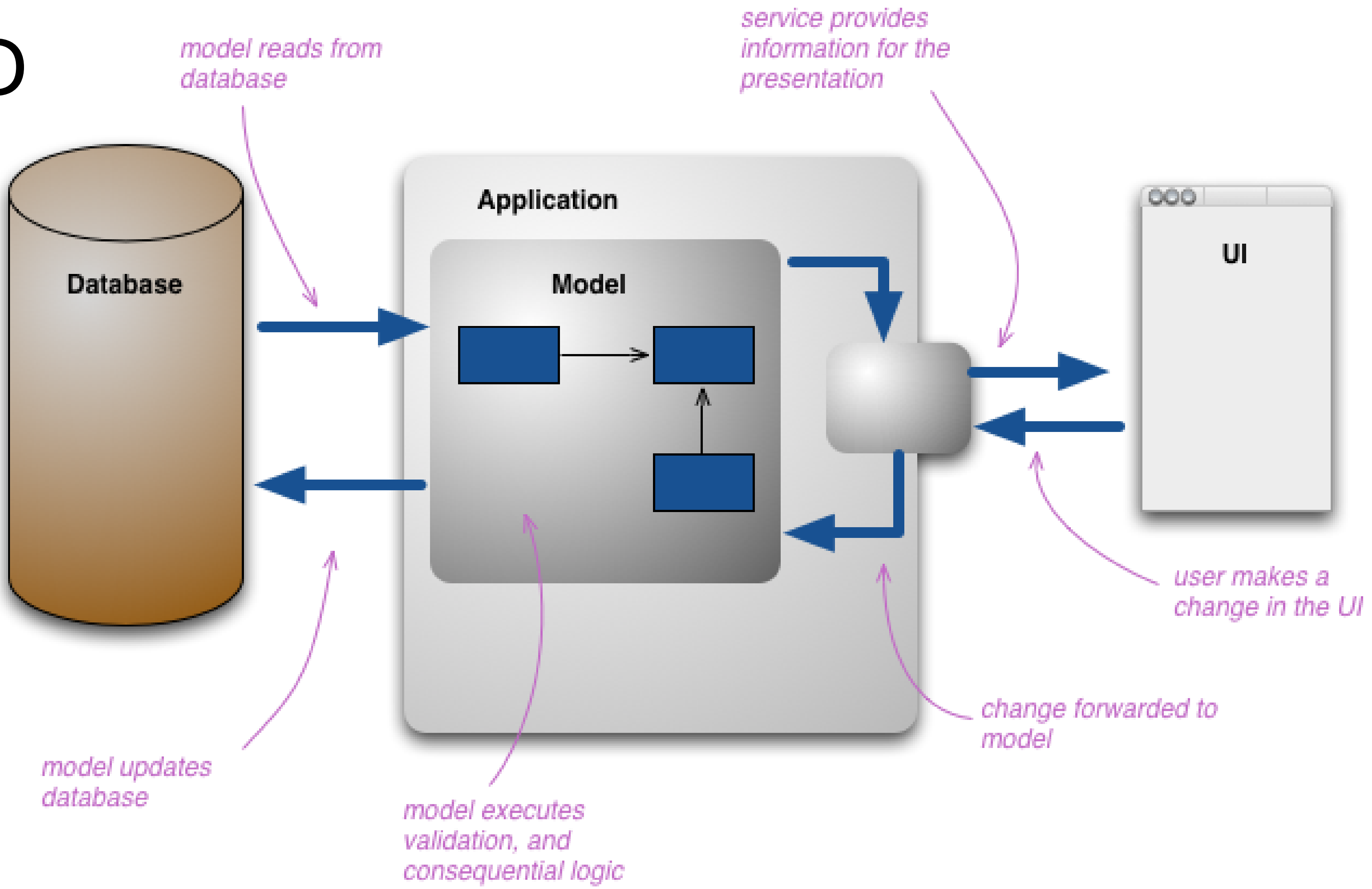
Command Query Responsibility Segregation (CQRS): Separating Reads from Writes

The Problem with CRUD

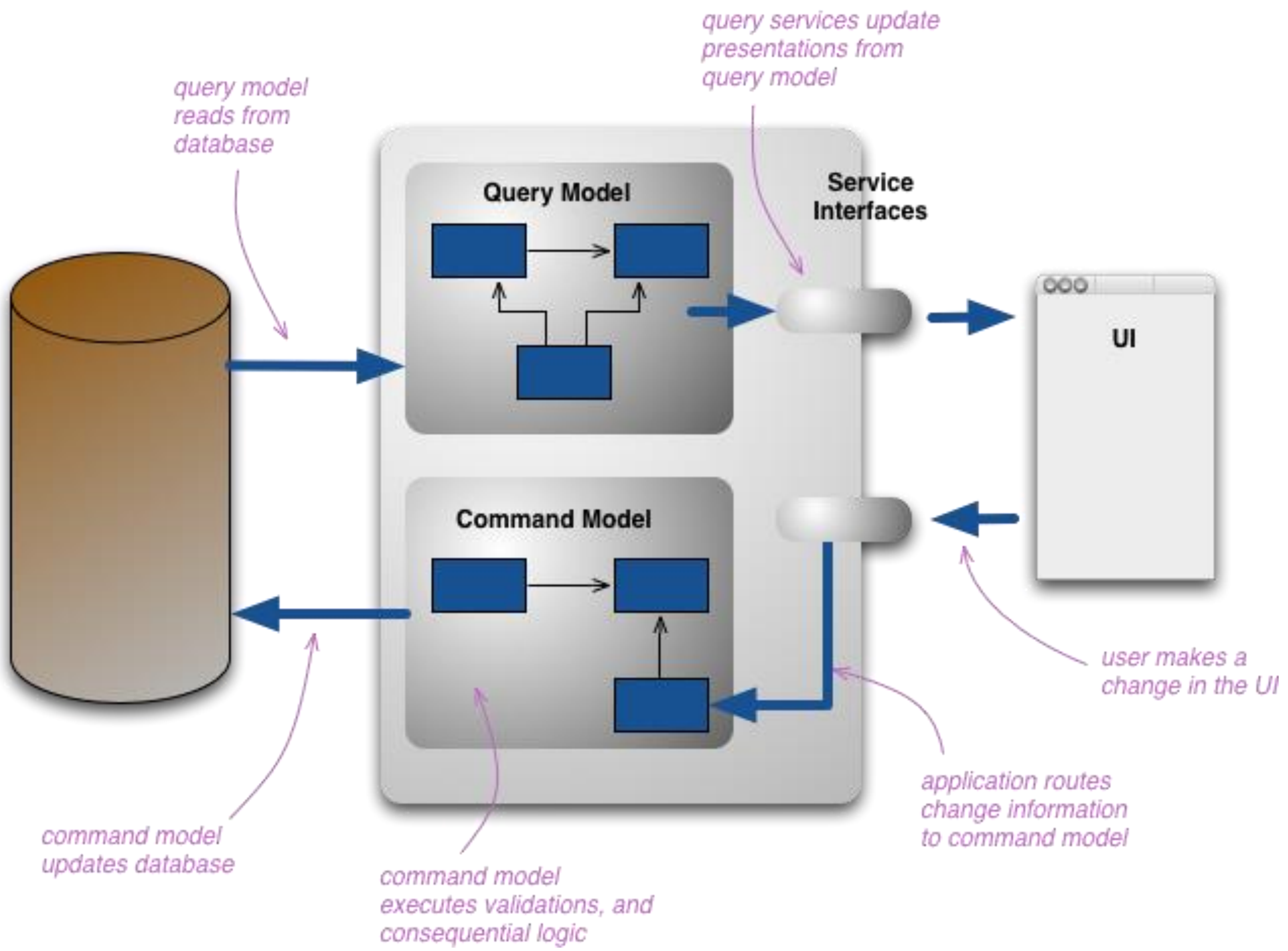
- CRUD models tightly couple read and write logic.
- Different performance/scaling needs for reads vs. writes.
- Complex domains (e.g., finance, e-commerce) suffer under one-size-fits-all models.

CRUD


model reads from database
service provides information for the presentation
Application
Database
Model
UI
user makes a change in the UI
change forwarded to model
model updates database
model executes validation, and consequential logic

https://martinfowler.com/bliki/CQRS.html

CQRS


query model reads from database
query services update presentations from query model
Query Model
Service Interfaces
UI
Command Model
user makes a change in the UI
application routes change information to command model
command model updates database
command model executes validations, and consequential logic

https://martinfowler.com/bliki/CQRS.html

The CQRS Principle & Motivation

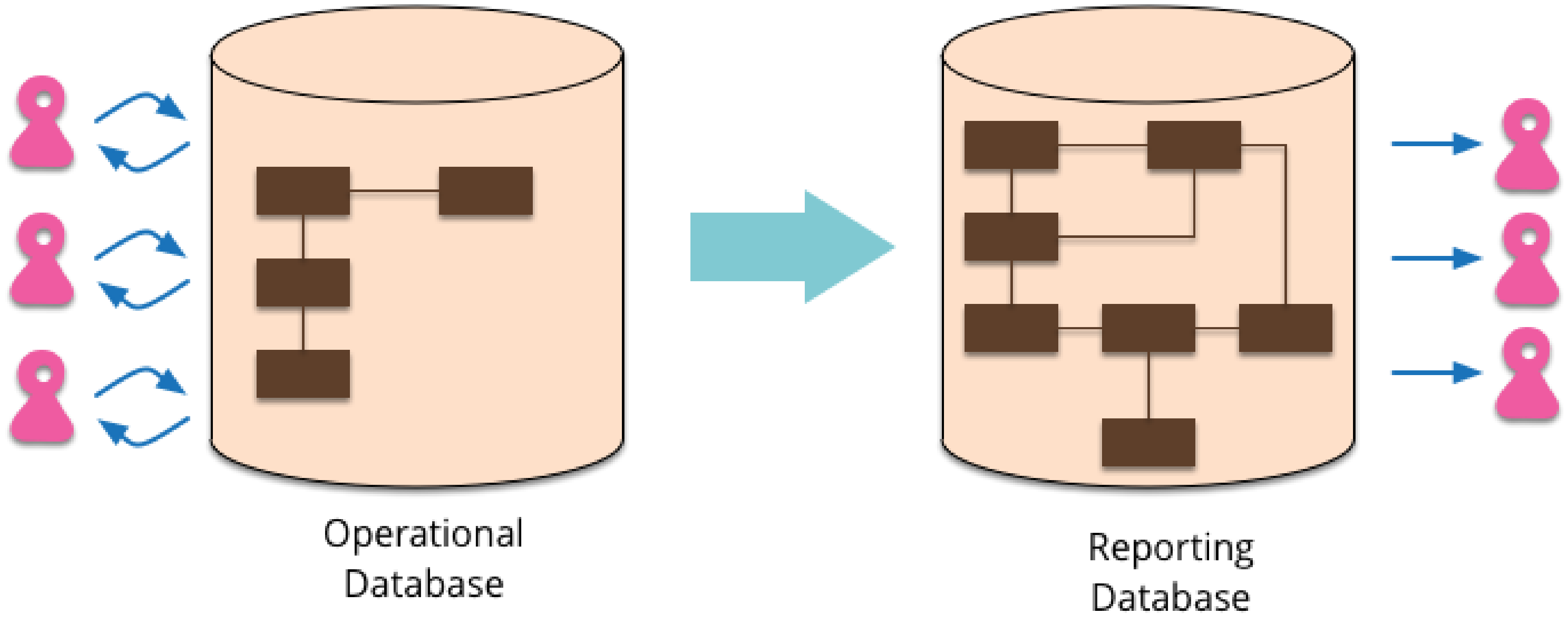
- **Command** = intent to change system state
- **Query** = request for information
- Each has distinct models, interfaces, and sometimes persistence layers
- Reads often dominate traffic (e.g., dashboards, user feeds)
- Writes need correctness; reads need performance
- CQRS gives architectural flexibility

CQRS & other Architectural Patterns

1. Event-Driven Architectures
   - CQRS commands may trigger messages/events (Kafka, Rabbit, …)
   - CQRS promotes decoupling between contexts and async processes
2. Event Sourcing
   - Storing events instead of state
   - Debugging points-in-time and replaying queries
3. Domain-Driven Design (DDD)
   - CQRS isolates domain logic from read concerns.

CQRS & other Architectural Patterns (contd.)

4. Reporting Database
5. Eventual Consistency



Advantages & Challenges

Advantage	Challenge
Scalability	Too complex for small apps
Clear domain separation	Complex debugging and testing
Fits microservices	Operational overhead (messaging, data storage)
Enables eventual consistency	Enables eventual consistency

Code example

- Let’s take a break from Java ☺
- https://github.com/alexovidiupopa/cqrs