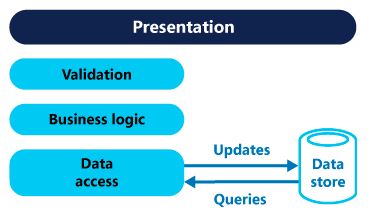
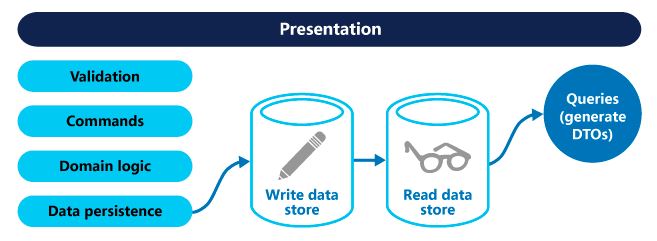
### **Command and Query Responsibility Segregation Design Pattern**

#### **Motivation**

The same data model is used to query and update a database in conventional designs. That is easy to understand and effective for CRUD processes. This method, however, may become cumbersome in applications that are increasingly complicated. For instance, the application might run a variety of queries on the read side, yielding data transfer objects (DTOs) of various shapes. Object mapping can get challenging. The model may include intricate business logic and validation on the write side. As a result, you risk creating a model that accomplishes too much and is unnecessarily complex.



Using commands to update data and queries to read data, CQRS isolates reads and writes into distinct models. The design and implementation are made simpler by having separate query and update models. However, one drawback is that scaffolding methods like O/RM tools cannot be used to automatically construct CQRS code from a database schema.



#### **Collaborations**

User interface will enable to use command and query operations and command and query will perform the subsequent operations on the database.

#### **Known Uses**

* large projects where independent scalability and great performance are required.
* In applications with complicated business logic. In this situation, you can make things simpler by separating your reads from your writes.
* If you want to develop in parallel, you can have one team work on reading models while the other team writes models.