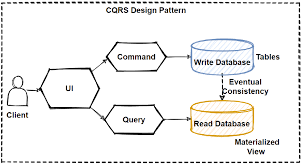
#### **Structure**



#### **Participants**

* Client

Interacts with UI to make requests.

* UI

Interface to make requests for command and query operations.

* Command

Defines create, update and delete operations.

* Query

Defines read operation.

* Write Database

Write data to the database for the client's command request.

* Read Database

Return data from the database for the client's query request.

#### **Consequences**

##### **Pros:**

* Utilizing the best database technology for the task at hand, such as a SQL database for writing and a noSQL database for reading, is made possible by separating write activity from ready action.
* Since reading occurs more frequently than writing, you can improve performance by locating read data sources in advantageous geolocations to reduce response latency.
* By separating write from read activities, storage capacity may be scaled more effectively based on actual demand.

##### **Cons:**

* Support for the CQRS pattern necessitates knowledge of several database technologies.
* Since more database technologies are needed to apply the CQRS patterns, there is a higher intrinsic cost, either in terms of hardware or, if a cloud provider is employed, utilization costs.
* Regarding service level agreements, extra emphasis must be given to ensuring data consistency.
* Utilizing numerous databases increases the risk of failure, thus businesses must implement thorough monitoring and fail-safe systems to ensure appropriate operation.

#### **Related Patterns**

* The Database per Service pattern creates the need for this pattern.
* The API Composition pattern is an alternative solution.
* The Domain event pattern generates the events.

CQRS is often used with Event sourcing.