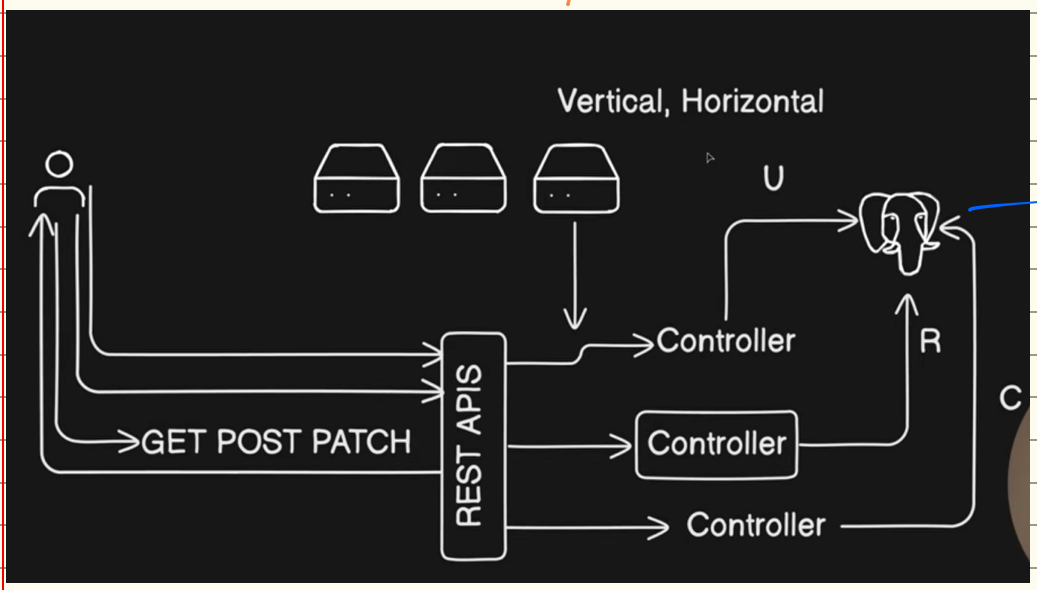
ADVANCE SYSTEM DESIGN.

↳CQRS.

CQRS - Command Query Responsibility Segregation.

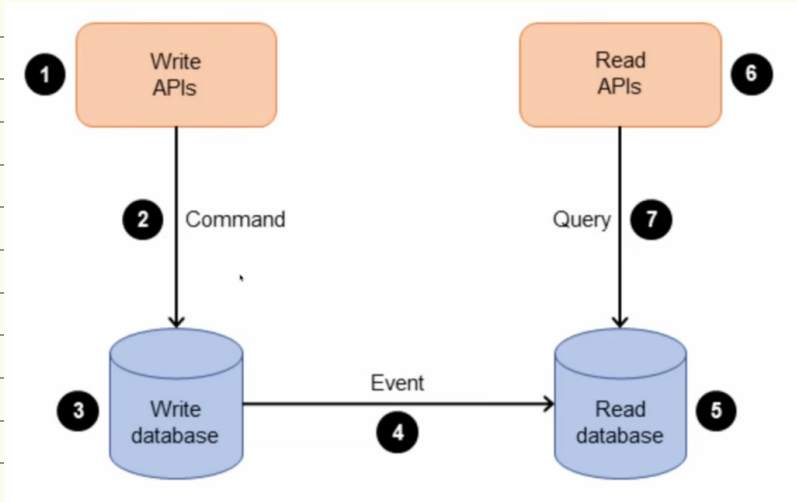
→ Traditional REST CRUD Endpoint



→ CRUD.
When we perform all the operations on a database, the database becomes overwhelmed.

→ CQRS pattern :-

The command query responsibility segregation (CQRS) pattern separates the data mutation, or the command part of a system, from the query part. You can use the CQRS pattern to separate updates and queries if they have different requirements for throughput, latency, or consistency. The CQRS pattern splits the application into two parts—the command side and the query side—as shown in the following diagram. The command side handles `create`, `update`, and `delete` requests. The query side runs the `query` part by using the read replicas.

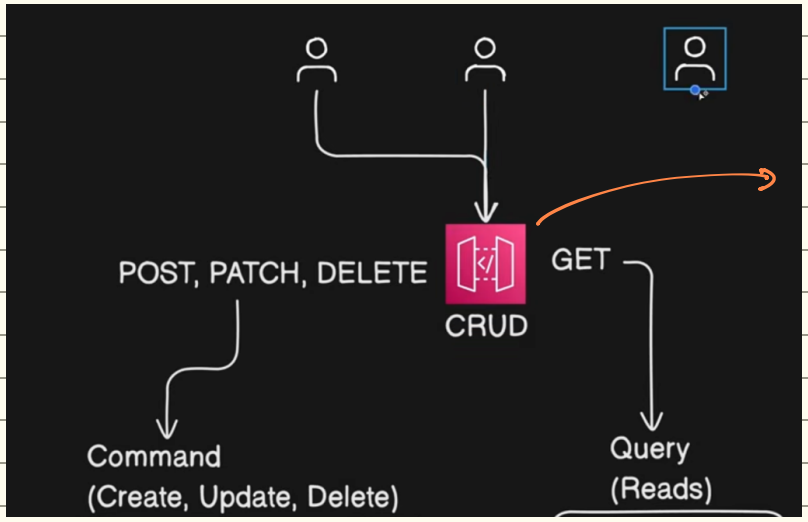


CRUD

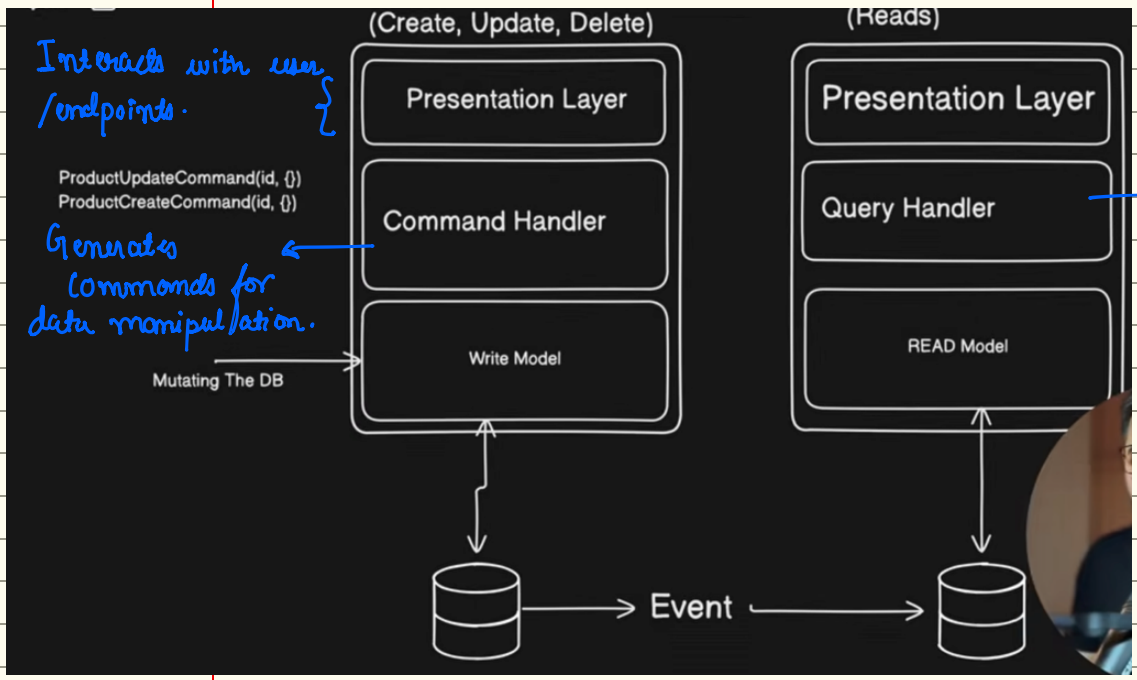
API Gateway.

Command
(Create, Update, Delete)

Query.
(Reads)



The API gateway segregate different type of requests according to Reads and Writes.



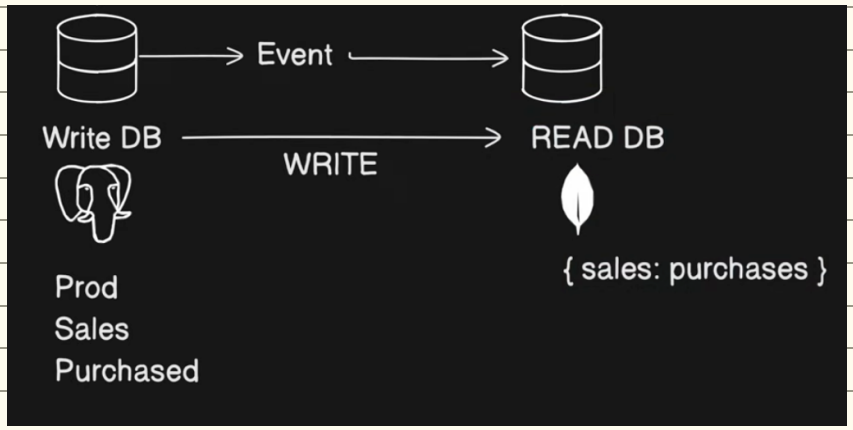
Handles READ query.

The diagram shows the following process:

1. The business interacts with the application by sending commands through an API. Commands are actions such as creating, updating or deleting data.
2. The application processes the incoming command on the command side. This involves validating, authorizing, and running the operation.
3. The application persists the command's data in the write (command) database.
4. After the command is stored in the write database, events are triggered to update the data in the read (query) database.
5. The read (query) database processes and persists the data. Read databases are designed to be optimized for specific query requirements.
6. The business interacts with read APIs to send queries to the query side of the application.
7. The application processes the incoming query on the query side and retrieves the data from the read database.

You can implement the CQRS pattern by using various combinations of databases, including:

- Using relational database management system (RDBMS) databases for both the command and the query side. Write operations go to the primary database and read operations can be routed to read replicas. Example: Amazon RDS read replicas
- Using an RDBMS database for the command side and a NoSQL database for the query side. Example: Modernize legacy databases using event sourcing and CQRS with AWS DMS
- Using NoSQL databases for both the command and the query side. Example: Build a CQRS event store with Amazon DynamoDB
- Using a NoSQL database for the command side and an RDBMS database for the query side, as discussed in the following example.



CQRS System Architecture.