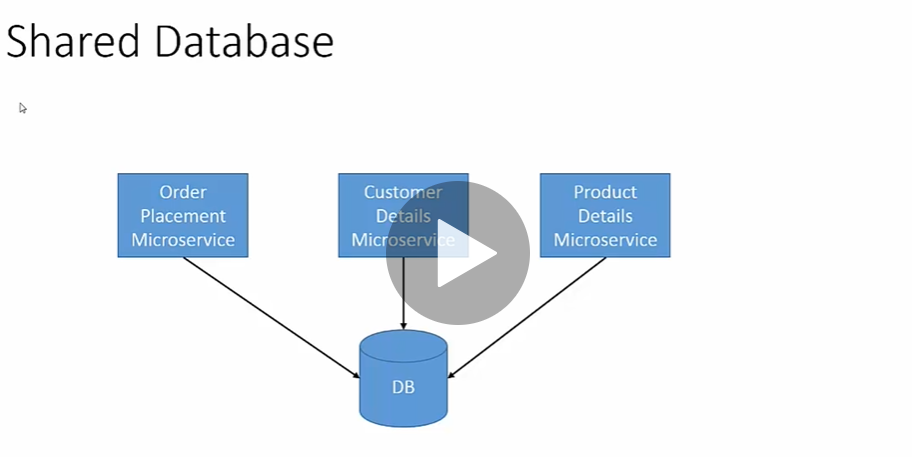
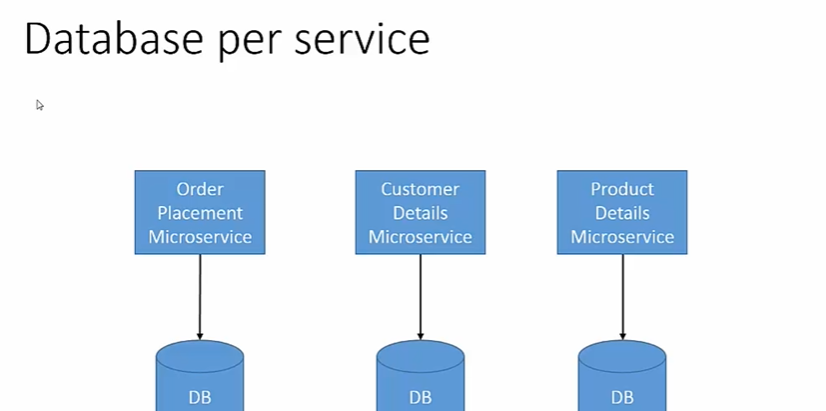
The different patterns as how data can be managed in a microservice environment.



The shared database pattern the microservices shared a common schema; the order payment microservice has access to the customer and product data. Each of the micro services are responsible to control the data consistency and integrity. This makes data management easy as all the data is in a central place. However drawbacks are changes in one table impacts multiple microservices and performance issue sometimes created by data lock on large data blocks. Hence this pattern is an anti-pattern for large complex data systems.

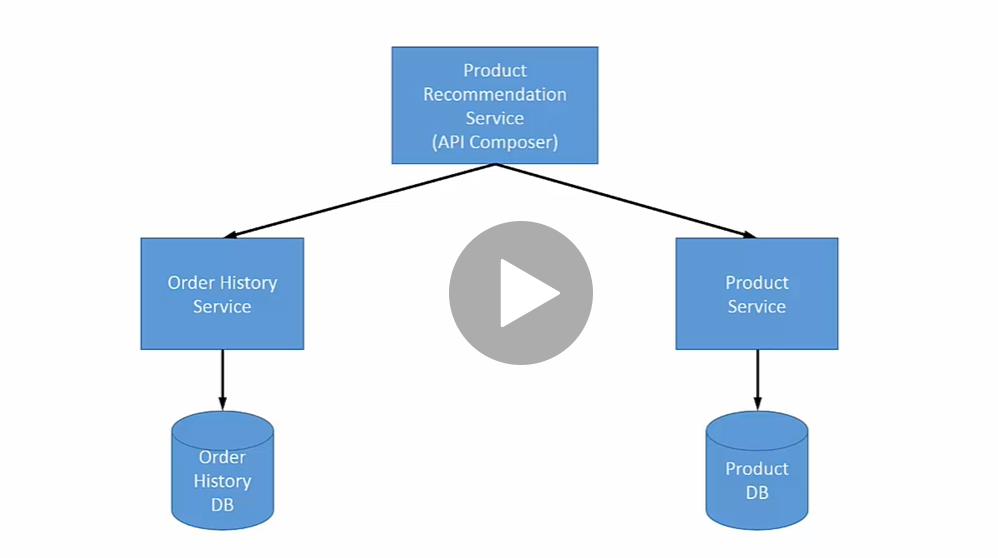


In the database per service; each microservice; has its own database/schema/table. This makes the microservices independent of each other and preventing data lock issues hampering other microservices. With this pattern the disadvantages that comes are more databases (infra costs) and cross-data access. Say a query that needs to join the order and product microservice.

To achieve data access from different microservices or data stores the 2 patterns are –

1. **API Composition**
2. **Event Sourcing**

In **API Composition** – Wrapper API’s are created which can access data from different microservices and implements the logic to merge, in-memory join and filter the data as needed. An example is – The product recommendation services needs data from the Order service that the customer has placed and the products details to make the recommendation. This is a simpler pattern, but problem happens with large data sets has to be transferred to the composer services.



In **Event Sourcing Pattern**: - API’s works on real time event streaming. Events are published whenever there is a change in the state of an object/entity. For example, in the following diagram whenever there is a activity in the Order Services (Customer places an order) then order events are published in the event store. Similarly, when the product services updates the product information then events are published to the Event store. The recommendation service can work on the event store (which Serves as a single source of truth). This eliminates the drawback of transferring large data sets between the micro services.

