Pattern to decompose microservices. There are three patterns to decompose microservices. They are strangler, saga and cqrs.

Strangler is used for refactoring. Means when we want to convert any monolithic system to microservice. In strangler there is a controller. It controls API traffic. It give access to API of some of the functionality to use microservice and some to use monolithic. Gradually the ratio for microservices will be increased and monolithic will be decreased. Ultimately all the functionality will be converted into microservices and then the monolithic codebase will be deleted.

Before understanding saga first we have to understand data management in database. We can use database per service or shared database. But generally shared database is not use.

Because we have to scale up whole database in the case where only one service require more store.

We can not modify the database based on only one service requirements. Because if we fulfill the requirement ( lets say deleting a table) it may creates problem for other dependable services.

But shared database has advantages in joining tables because of same db.

Also in transaction property. If it fails to update one table the entire process will be roll back. It follows ACID property. Either full process will be complete or there will be failure.

Easiness of the shared database are the challenges for database per service.

If one service need information from the db of another service it can not directly access rather it should ask the db.

Saga solved one of the issue of microservices which is transaction management. If any task require to update multiple db of different services it become really difficult to roll back. Because of saga it is possible to easily roll back.

Saga means sequence of local transaction. In database per service system, when a service done anything to its db it publishes an event and the next service listen to this event and on success publish an event and the next services also do the same thing so on and so on. But if db update of any of the services fails that service will publish failure event. The previous service will listen to this event and publish failure event. All the previous services will also do the same thing. In such manner the whole process will be rolled back.

There are two types of Saga. One is choreography and the other one is orchestrator.

In choreography there are two types of event block or queue. One is message event and another is failure event. Each service will set an event on the message event queue on success of the transaction and the next service will listen to this. Each service will set an event on the failure event queue on the failure of the transaction and the previous service will listen to this. Drawback of this is cyclic dependency.

In orchestrator, there is a central single controller which is called orchestrator. It will call the services one by one on the successful reply of the previous service and to do roll back on failure.

For example there is a scenario per A want to 10 taka to person B. There are two services in the system one is balance which store the current amount and another one is payment which stores the history of different transaction. Lets say the db of balance has been successfully updated. Now the current balance is set from 100 to 90. But the process of the db of payment has failed. Which pattern you will choose. The answer is Saga orchestrator.

CQRS is Command Query Request Segregation. Command means create, update and delete and query means select. It is used to overcome the situation of table joining issue in microservice. In this pattern, another db will be created. Let's say this db is called view db. Every create, update and delete operation in microservices will be happened in their own db only the read operation will be happened from the view db. This view db will contain all the data of db of all the services. So for changes in db of any of the services, the data of view db should be changed. How will this happen? When data changes in db of any of the services, it will publish an event and the view db will listen to this and will update accordingly. Another way is triggering db. After changes the process of view db will be triggered.

===========================

Questions

===========================

1. Deeply understand strangler, saga, cqrs

2. Deeply understand choreography and orchestrator.