Redis Base

\*\*3. Redis 의 특수 명령어

\*데이터 만료

-데이터가 만료된다는 의미는 데이터가 특정시간에만 유효하고 그 이후에는 무효화 된다는 말

-TTL(Time To Live) 를 통해 데이터가 유효한 시간을 초단위로 기록함

-redis에서 데이터를 저장한 이후 만료시키지 않으면 계속저장공간을 차지하기 때문에 삭제하거나 적당한 시간에 만료하는게 좋음

-만료된 데이터는 요청시 조회되지 않음.

-데이터가 만료되자마자 삭제되는 것은 아니고, 만료되었다는 사실만 표시후 백그라운드에서 주기적으로 삭제 됨.

-명령어

1. SET greeting hello

EXPIRE greeting 10 (EXPIRE : 데이터 만료 명령어)

2. TTL greeting (TTL 명렁어는 만료까지 남은 시간 확인), 만약 TTL 명령어 이용했는데 -1이 리턴되면 데이터만료가 설정되지 않았다는 뜻, -2는 만료 되었다는 뜻.

3. SETEX greeting 10 hello (데이터저장과 동시에 만료까지 걸리는 시간도 같이 저장함)

\*SET NX/XX

- NX : 해당 Key가 존재하지 않는 경우에만 SET

- XX : 해당 Key가 이미 존재하는 경우에만 데이터를 덮어 쓴다.

Null Reply : NX나 XX를 이용했을 때 조건이 맞지 않으면 레디스 (nil)을 응답한다.

-이런옵션이 없다면 직접 키를 대입해서 데이터 존재 여부를 확인하고 이후에 SET을 보내는 프로그래밍이 필요하지만 NX/XX옵션을 이용하면 편리해진다.

-명령어

SET greeting hello NX

SET greeting hello XX

\*redis Pub/Sub (중요함)

- Pub/Sub이란 시스템 사이에 메시지를 통신하는 패턴중 하나로 Publisher와 Subscriber가 서로 알지 못해도 통신이 가능하도록 메시지 패턴이나.

- Pub/Sub의 최대 장점은 두 시스템 간의 강한 결합을 줄일 수 있다는 것이다.

- 발행자는 구독자에게 직접 메시지를 보내지 않고 채널에 메시지를 발행하게 되고 구독자는 관심있는 채널만 구독하여 메시지를 수신할 수 있다. 따라서 구독자의 기능이 변경되더라도 발행자는 신경 안써도 되고 , 구독자는 필요에 따라 구독하는 채널을 조절하면 된다.

- redis Stream과 다른점은 발행된 메시지가 보관되는 것과 달리 Pub/Sub은 fire and forget 매커니즘이 적용되어 있어서 구독전에 발행된 메시지는 다시 수신할 수 없다.

1. Fire and Forget? : 메시지를 발행한 쪽(publisher)이 메시지를 보낸 뒤 **응답이나 결과를 기다리지 않고 그냥 잊어버리는 방식**을 말합니다.

2. 창고를 관리하는 스톡(재고 관리) 서비스는 주문 상태에 따라서만 영향을 받기에 주문 채널만 구독하면 되고, 알림을 전송하는 Notification 서비스에서는 주문, 결제, 배송의 모든 이벤트 추적을 위해 채널 모두를 구독할 수 있다.

-명령어

SUBSCRIBE ch:order ch:payment (컨슈머는 다음과 같이 원하는 채널 구독 가능)

PUBLISH ch:order new-order (퍼블리셔는 원하는 채널에 메시지를 발행할 수 있다.)

PUBLISH ch:payment new-payment

\*Pipeline

- Pipelining이란 다수의 레디스 명령어(commands)를 한번에 요청하여 네트워크 성능을 향상 시키는 기술이다.

-일반적인 Request-Response모델에서 요청과 응답에 따라 네트워크를 통해 트래픽이 오고가며 Round Trip Time이 발생하게 된다.

1. Request-Response 모델 :

클라이언트(Client): 어떤 동작을 요청(Request)을 보냄

서버(Server): 요청을 처리한 뒤 결과(Response)를 다시 돌려줌

2. Round Trip Time (RTT) : RTT(Round Trip Time, 왕복 시간)\*\*은 네트워크에서 패킷이 출발해서 목적지까지 갔다가 응답이 돌아오기까지 걸리는 시간을 뜻한다.

-PipeLine은 이 라운드 트립의 횟수를 줄여 네트워크 시간을 최소화하는 기술이다.

-대부분 Redis Client 라이브러리에 구현이 되어 있다.

-예시)

1. SET 명령어와 EXPIRE 명령어로 각각 요청하고 응답을 받으면 총 2ms의 요청시간이 걸린다고 할 때 한번에 Pipelining하여 요청한다면 1ms만 소요된다.

2. 실제 redis 명령어가 처리되는 속도는 1ms보다도 빠른경우가 많기 때문에 구체적인 수치를 기억하기 보다는 Pipelining을 통해 라운드 트립을 줄여 전체적인 지연시간을 줄일 수 있다는 개념을 이해하자.

\*Transaction

- TRANSACTION : 다수의 명령어를 하나의 묶음으로 처리하여 데이터의 원자성을 보장하는 기술이다.

- TRANSACTION은 관계형 데이터베이스에 주료 사용되지만 Redis에서도 트렌젝션을 지원한다.

- 원자성을 유지하기 때문에 다수의 명령어 처리중 에러가 발생하면 모든 작업을 rollback 한다.

- 하나의 트렌젝션을 처리하는 도중에는 다른 클라이언트의 요청이 중간에 끼어들 수 없다.

- 원자성이란 : 모든작업이 모두 적용되거나 하나도 적용되지 않는다 라는걸 의미한다.

- Pipeline은 네트워크 퍼포먼스를 위해 요청을 한번에 전달하는 기술일뿐 다수의 명령어가 개별적으로 수행 된다. 하지만 트렌젝션은 원자성 보장을 위해 다수의 명령어가 일괄 적용되거나 모두 적용되지 않는다.

-따라서 파이파인과 트렌젝션은 다른기술이지만, 동시에 사용도 가능하다.

-명령어

1. MULTI (트렌젝션 시작 명령어, 이후에 수행할 명령어들을 입력한다.)

INCR foo (트렌젝션에 들어갈 명령어 예시)

DISCARD (트렌젝션 롤백 명령어)

EXEC (트렌젝션 적용 명령어)