**REDİS (CACHING)**

**Caching Nedir, Avantajları nelerdir?**

Verilerin hızlı erişilebilmesi içini verilerin bellekte saklanma durumuna denir. Zamanlama anlamında sabit distden veri çekmeye göre çok hızlı olması durumudur, hızlı bir erişim olduğumdan yazılım anlamında performasn da arttırıcaktır, sunucu yükünü azaltıcaktır , çevrim içi uygulamalarda caching mekanizması çok önemlidir anlık veriyi hızlı bir şekilde erişebilmek adına.

**Ne tarz veriler cachelenebilir ?**

Bu veriler sıklıkla hızlı erişilebilecek veriler olmalıdır .

Configurasyon verileri(Arka plan rengi gibi) , Menü bilgileri , yetkiler ve buna bağlı token verileri gibi..

Statik verileride(resim video gibi…) cachelenebilir.

Cachelenmeyecek veriler ise geçici ve güvenlik açısından sıkıntı oluşturucak veriler cachelenmemeli dir.

**Dezavantajları nelerdir ?**

Kapasitesi sınırlı olan belleye daha fazla yük eklemek demektir , bize bir yarar sağladığı gibi böyle bir maliyeti olduğundan uygun olan veriler cachelenmelidir, database tarafında değiştirilen bir konuda , cache yapısı hala esik dataya bakma gibi bir durum oluşabilir yani güncellik sorunu , güvenlik ile ilgili sıkıntılar.

**Caching Yaklaşımları nelerdir ?**

In-Memory Caching , Distiributed Caching yaklaşımları bulunmaktadır.

|  |  |
| --- | --- |
| In Memory Caching | Distiributed Caching |
| Veri uygulmalarında çalıştığı bilgisayarın RAM yapısına cacheleyen yaklaşımdır | Verileri birden fazla fiziksel makinede cacheleyen böylece verileri farklı noktalarda tutarak güvenli bir davranış oluşturan yaklaşımlardır |
| Kullanımı oldukça kolaydır Redis In memory ede kullanılabilir | Birden fazla yapı mevcuttur , bunlardan bazıları , redis, Memcached, hazalcast gibi… |

**Redis(Remote Dictionary Server) in özellikleri nelerdir ?**

Key Value cinsinden verilerin tutulmasını sağlamaktadır .

NoSql veri tabanı kullanımı vardır.

Opensource olup yüksek performanslı bir şekilde cacheleme yapmaktadır

App şeklinde kurulumu ya da dockerize gibi bir kurulum ile yapılandırması sağlanabilir .

Redis yüklemesi yapıldıkdan sora default olarak 6379 portundan ayağa kalkmaktadır.

Redis bir sunucudur haliyle görselleştirmek için redis insight kısmını kullanmak gerekmektedir

**Redis Değişken Türleri nelerdir ?**

**String** : Redisin en temel veri türüdür , metinsel ifadeler ile birlikte , binary düzeyde resim dosya vs. verilerinin saklanmasında kullanılmaktadır.

|  |  |  |
| --- | --- | --- |
| KOMUT | İŞLEVİ | ÖRNEK |
| SET | Ekleme | SET NAME Soykan |
| GET | Okuma | GET NAME |
| GETRANGE | Karakter aralığı okuma | GETRANGE NAME 1 2(index,kaç tane karakter okunucak ) |
| INCR & INCRBY | Arttırma | INCR SAYI |
| DECR & DECRBY | Azaltma | DECR SAYI |
| APPEND | Üzerine ekleme | APPEND NAME Saydam |

**List** : Koleksiyonel olarak olarak veri tutan bir türdür.

|  |  |  |
| --- | --- | --- |
| KOMUT | İŞLEVİ | ÖRNEK |
| LPUSH | Başa Veri Ekleme | LPUSH NAMES Hilmi Soykan |
| LRANGE | Verileri listeleme | LRANGE NAMES 0-1 |
| RPUSH | Sona veri ekleme | RPUSH NAMES Ahmet |
| LPOP | İlk datayı çıkarma | LPOP NAMES |
| RPOP | Son Datayı çıkara | RPOP NAMES |
| LINDEX | Indexe göre datayı getirme | LINDEX NAMES 1 |

**Set** : Verileri rastgele bir düzende unique olarka tutan bir türdür

|  |  |  |
| --- | --- | --- |
| KOMUT | İŞLEVİ | ÖRNEK |
| SADD | Ekleme | SADD COLOR red blur green orange |
| SREM | Silme | SREM COLOR blue |
| SISMEMBER | Arama | SISMEMBER COLOR red |
| SINTER | İki setteki kesişimi getir | SINTER user1:BOOKS user2:BOOKS |
| SCARD | Eleeman sayısını getirir | SCARD COLOR |

**Sorted Set** : Set in düzenli bir şekilde tutan versiyonudur Unique olarak tutulup bunun dışında her veriye score adı verilen değer atanıp bu değer kullanarak veriler sıralanır.

**Not:** Ekleme sırasındada verilen değerler unique sıralama aynı sıralama olarka kullanılabilir.

|  |  |  |
| --- | --- | --- |
| KOMUT | İŞLEVİ | ÖRNEK |
| ZADD | Ekleme | ZADD TEAMS 1 A |
| ZRANGE | Getirme | ZRANGE TEAMS 0-1 |
| ZREM | Silme | ZREM TEAMS A |
| ZREVRANK | Sıralama Öğrenme | ZREVRANK TEAMS B |

**Hash** : Key value formatında veri tutan türdür.

|  |  |  |
| --- | --- | --- |
| KOMUT | İŞLEVİ | ÖRNEK |
| HMSET & HSET | Ekleme | HSET EMPLOYESS username Soykan |
| HMGET & HGET | Getirme | HGET EMPLOYESS username |
| HDEL | Silme | HDEL EMPLOYESS username |
| HGETALL | Tümünü Getirme | HGETALL EMPLOYESS |

**Streams**: Log gibi hareketen eden bir türdür , sırasıyla kayıt edilip sonrasında işlenmeleri sağlanmaktadır

**Geospatial Indexes** :Coğrafi kordinatları saklamak için kullanılmkatadır .

**In-Memory Cache İşlem Sırası :**

**AddMemoryCache** servisini uygulamaya ekleyiniz

**IMemoryCache** referansını inject ediniz

**Set** metodu ile veri caclenip **Get** metodu ile cachelenmiş veri elde edebiliriz .

**Remove** fonsiyonu ile cahclenmiş veriyi silebiliriz

**TryGetValue** metodu ile kontrollü bir şekilde cacheden veriyi okuyabiliriz

**Absolute & Sliding** kavramları cache mekanizmasının ömrünü belirlemede kullanılan iki paretmetre olarak kullanılmaktadır , aynı anda hem absolute hem de sliding olarak kavramları cache tarafına tanımlana bilir

**Absolute Time**  Cache deki datanın ne kadar tutulacağına dair net ömrünün belirtilmesidir, belirtilen ömür sona erdiğinde cache direk olarak temizlenir.

**Sliding Time** Cache deki datanın memory de belirtilen süre periyodu zarfında tutulmasını belirtir , Cache yapılan erişim neticesinde datanın ömrü bir o kadar uzatılmalıdır, aksi taktirde erişim söz konusu olmassa cache verisi temizlenecektir.

**Distributed Cache İşlem Sırası :**

**StackExchangeRedis** kütüphanesini uygulamaya yükleyiniz

**AddStackExchangeRedisCache** servisini uygulamaya yükleyiniz

**IDistributedCache** referansını inject ediniz.

**SetString** metodu ile metinsel , **Set** metodu ile ise binary olarak verileri redise cacheleyebiliriz .

**GetString** ve **Get** metodu ile de cachelenmiş verileri elde edebiliriz

**Remove** fonksiyonu ile cachelenmiş verileri silebilirsiniz

**Absolute & Sliding ,** aynı şekilde bu kısımdada bu parametreler kullanılmakta olp configure edilmektedir

**Absolute Time**  Cache deki datanın ne kadar tutulacağına dair net ömrünün belirtilmesidir, belirtilen ömür sona erdiğinde cache direk olarak temizlenir.

**Sliding Time** Cache deki datanın memory de belirtilen süre periyodu zarfında tutulmasını belirtir , Cache yapılan erişim neticesinde datanın ömrü bir o kadar uzatılmalıdır, aksi taktirde erişim söz konusu olmassa cache verisi temizlenecektir.

**Redis Pub/Sub Özelliği :**

Redis her ne kadar caching mekanizması olarak kullanılsada pub/sub işlemi yapabilen message broker özelliğide barındırmaktadır.Bunu yapabilmekiçin bir kaç yol mevcuttur, Redis Cli , Redis Api , Redis Insight gibi yollar ile işlem bu konuda yapılabilir

Bu kısımda kullanım genel olarak bir den fazla dağınık olarak kullanılan redis tarafında , yapılan bir değişiklik sonucu bundan uygulamanın haberi olması için bu broker yapısı kullalırak haber verilip uygulamadaki cache mekanizmasının revizesi sağlanabilir

<https://www.youtube.com/watch?v=XP8HMl9zd9A&list=PLQVXoXFVVtp3_UlZu9qibcUzfm9ve3yVO&index=5&ab_channel=Gen%C3%A7ayY%C4%B1ld%C4%B1z> (Bu kaynaktan yararlanabilrsin doğrudan)

**Replication Nedir ve Rediste neden kullanılmaktadır ?**

Bazen redisde yaptığımız çalışmaları sunucuda herhangi bir fiziksel kesilme sıkıntı durumunda verilerin geri dönüşü olmıyacak şekilde ciddi önemli veri kaybı oluşabilir . Bundan dolayı redis deki veriler Replication davranışı uygulanarak veri kaybı gibi durumlar için daha dirençli bir alt yapı kurgulanabilir.

**Master** : Replicası alınıcak sunucuya master adını veriyoruz

**Slave** : Master ın replikasını alan sunucuya denmektedir

Master ile Slave arasında bir connection oluşturulup burada bağlantı sağlanarak replikası alınana slave tarafının devamlı güncel kalması sağlanmaktadır aam bu bağlantı bazen kopa bildiği için sorumluluk bu gibi durumlarda tekrardan master sunucularına alırak verinin kesintisiz olarak erişebilmesi sağlanmaktadır

Master tarafında yapılan değişiklik slave tarafına aktarılması sağlanacaktır , slave tarafında bir değişiklik yapılamayacaktır , sadece kayıt edilme olarak kullanılcaktır slave tarafında sadece okuma yapılabilir , ölçeklendirme amaçlı yapılmaktadır

Canlı veriler üzerinde test yapılması için slave sunucu üzerinde test yapıla bilecektir ve master tarafta gerçekte de hiç bir sıkınıt yaşanmıcaktır

Replication davranışı ciddi anlamda kaynak tüketimi yapmaktadır ama önemli noktalardada işi nerdeyse garantiye almaktadır

Bir master ın bir den fazla slave I olabilmektedir (Bu kısımdaki haberleşmede **Pub/Sub** özelliği kullanılarak veri kısmında haberleşme sağlanabilir)

**Replication Uygulayabilmek için :**

- Replication davranışlarını sergileye bilmek için Redis sunucularını ayağa kaldırmak gerekmektedir

(docker run -p 1461:6379 --name redis-slave -d redis)

- Ardından master ile slave sunucusu arasındaki replication ilişkisi oluşturabilmek için master ın IP sini elde etmek gerekicektir

(inspect -f "{{ .NetworkSettings.IPAddress }}" redis**)(Koyu puntolu kısım isim sunucunun ismidir)**

- Son olarak da master ile slave arasında ilişki oluşturulmalıdr

(docker exec -it redis-slave redis-cli slaveof 172.17.0.2 6379)**(Master ip ve port ları)**

**Redis Sentinal Nedir ?**

Redis cache amacıyla kullanılsada , çalışılan mimari geniş bir yapıya sahip birden fazla sunucuyla ölçeklendirmeye çalışması böyle bir yaklaşım ortaya çıkarmıştır bunun sebebi sistemi devamlı olarak sürdürebilir kılmak için oluşturulmuştur. Bir systemin kesintisiz çalışması yüksek kullanılabilirlik ile nitelendirilir ve redis sentinel yapısı ise bu kısımdaki yüksek kullanbilirliği sağlamak amacıyla geliştirilmiş yinetim servisidir .

**Neden Kullanılır ?**

- Redis sunucusunun arızalanması durumunda kullanılır, sentinel davranışı sayesinde , servisi farklı bir sunucu üzerinden Redis yapsıının çalışmasına devam edebilir.

- Bakım ve güncellme işlemlerinde yine aynı durum bakımından kullanılabilecektir

- Yüksek trafik de kullanılır , Yüksek trafik te redis sunucu vermesi gereken cevaplarda gecikmeye neden olabilir , böyle durunmlarda sentinel ile daha performanslı çalışmalar gerçekleşebilir

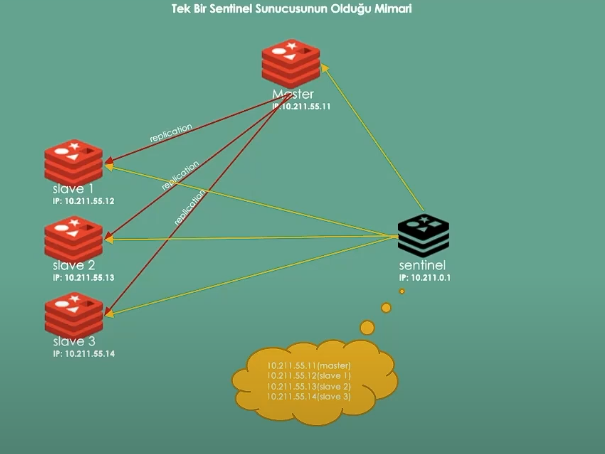
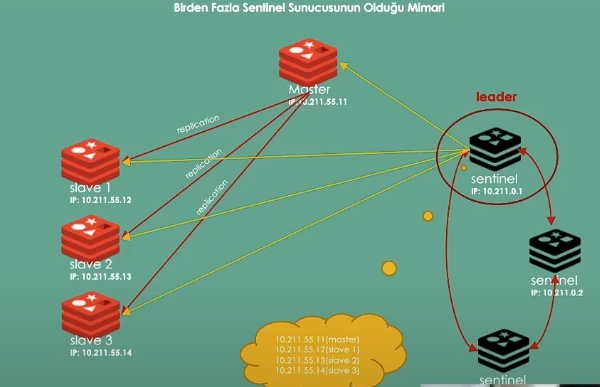
- Yedekleme ve geri yükleme durumlarında da sentinel yaklaşımı kullanılmaktadır .

**Temel Kavramları nelerdir ?**

Redis Sentinal sunucu , master slave yapısının üzerindeki bir yönetim servisidir , Replikasyon yapısında otamatik olarak yönetim anlamında kullanılmaktadır , detaylandırıcak olursak kesinti , problem sonucunda **fail over**(yük devretme) işlemini gerçekleştirerek farklı bir sunucuda işlemlerine devam edecektir

Master Slave Sentinal ve Fail over kavramları kullanılmaktadır

**Redis Sentinal Nasıl Çalışır ?**

**Sentinal Sunucusu oluşturma**

Sentinal sunucusunda yapılandırma yapılabilmesi için , sentinel.conf dosyası oluşturulup içeriğine ilgili komutlar girilşmesi sağlanır

Redis Sentinal sunucusu ile detaylı olarak verilen conf bilgileri ve Asp.Net ile yapılan sentinel kurulumu için bu kaynak verisine daha detaylı olarak bakabilirsin

https://www.youtube.com/watch?v=B9VtHMZnQx4&list=PLQVXoXFVVtp3\_UlZu9qibcUzfm9ve3yVO&index=7&ab\_channel=Gen%C3%A7ayY%C4%B1ld%C4%B1z