Cap Teoremi Ve Spring Framework 'de Transaction Propagation

Medine Aktaş

CAP TEOREMİ NEDİR?

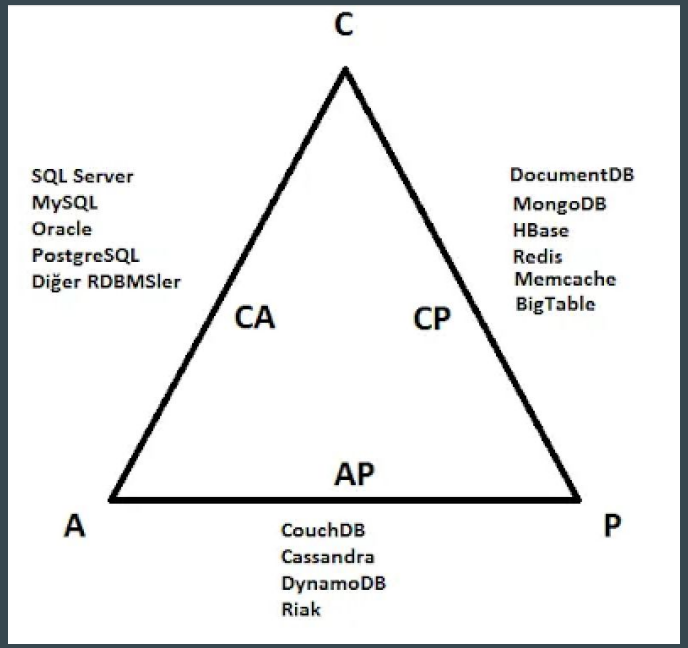
CAP Teoremi ya da diğer adıyla Brewer Teoremi 1998 yılında Eric Brewer adındaki bir profesör tarafından ortaya atılmıştır.

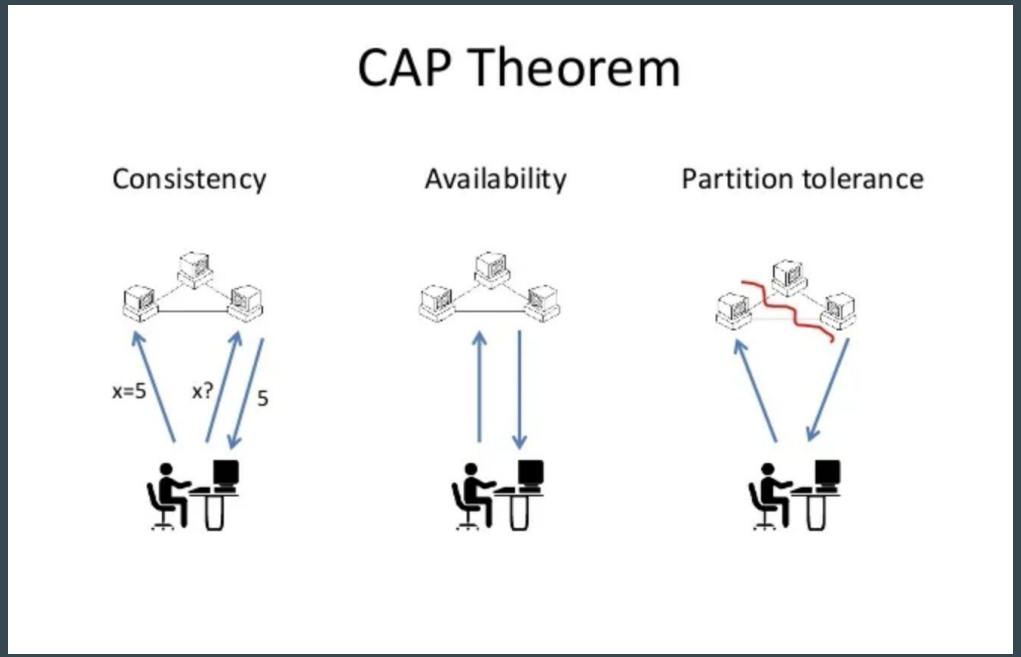
Consistency : Bütün düğümlerde aynı verinin olmasıdır.

Availability : Sunucuların biri göçtüğünde ,diğerinin yerini alabilmesidir.

Partition Tolerance : Verinin network üzerinde, dağıtılıp dağıtılmayacağıdır.

Teorem basit olarak dağıtık bir sistemde veri üzerinden sunulan hizmet için aynı anda 3 özelliğin sağlanamayacağıdır.





Consistency : ACID vs BASE

- Atomicity : İşlemler bölünmez.
- Consistency : İşlem öncesi ve sonrası veri bütünlüğü.
- İsolation : İşlerin sıralı bir şekilde çalışması
- Durability : Bir işlem gerçekleştikten sonra değişiklikler kalıcıdır ve veri tabanın da bir problem olsa bile korunur.

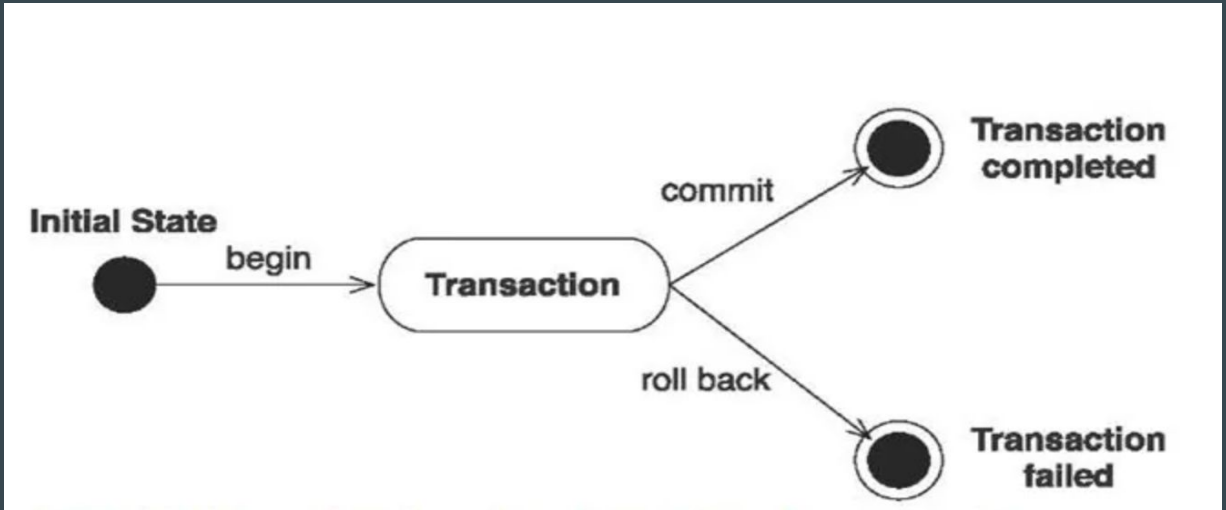
- Basic Availablitiy (Temelde Erişilebilir): Sistem her zaman temelde kullanılabilir durumda olmalıdır.
- Soft state (Yumuşak Durum): Sistem, bir süre boyunca tutarsız olabilir, ancak sonunda belirli bir tutarlılık seviyesine ulaşır.
- Eventually consistent (Sonunda Tutarlı): Veriler zaman içinde birbirine yaklaşarak veya belirli bir senkronize olma sürecini takiben tutarlı hale gelir.

ACID ve BASE arasındaki fark şu şekilde özetlenebilir:

- ACID, verilerin her zaman tutarlı olmasını garanti eder.
- BASE, verilerin her zaman tutarlı olmasını garanti etmez. Ancak, veriler zaman içinde tutarlı hale gelir.

Transaction Nedir?

Bir dizi işlemin, bütünlük içerisinde gerçekleştirilmesi. Tüm işlemler ya tam olarak gerçekleştirilir ya da hiçbiri gerçekleştirilmez.



@Transactional Anotasyonu

Java’da transaction yönetimini kolaylaştırır.

Transaction Yayılımı (Transaction Propagation)

Mevcut bir transaction içinde başka bir transaction in nasıl davranacağını belirleyen yönetimsel bir mekanizmadır.

REQUIRED

REQUIRED

SUPPORTS

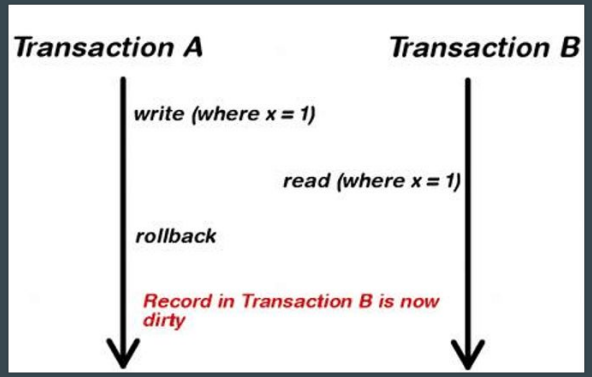
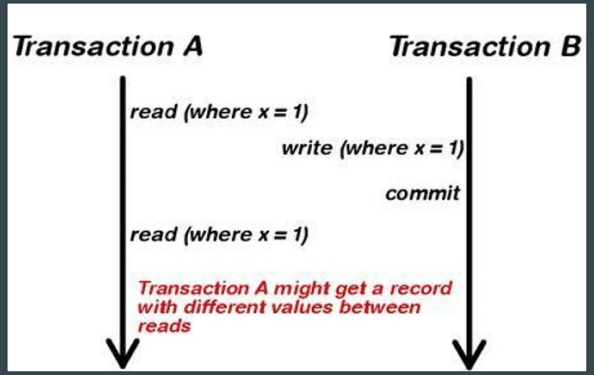
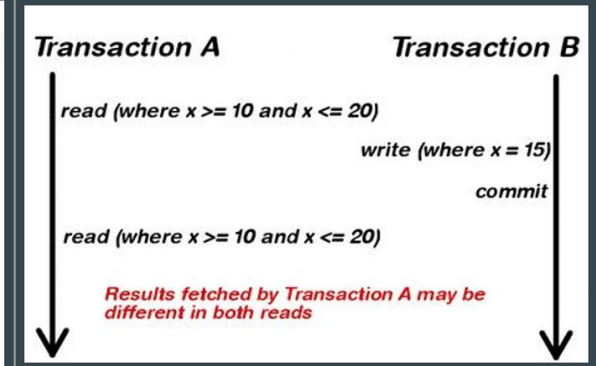
REQUIRES_NEW

NOT_SUPPORTED

NEVER

MANDATORY

Transaction İzolasyonu

• Dirty Read	• Non-Repeatable Read	• Phantom Read
		

İzolasyon Seviyeleri

- READ_UNCOMMITTED
- READ_COMMITTED
- REPEATABLE_READ
- SERIALIZABLE

READ_UNCOMMITTED

Diğer transansaction'lar tarafından commit edilmemiş verilere erişim sağlar.

Avantaj: Performans.

Dezavantaj: Dirty Read, Non-Repeatable Read, Phantom Read.

```
@Transactional(isolation = Isolation.READ_UNCOMMITTED)
public void log(String message) {
   // ...
}
```

READ_COMMITTED

Diğer transansaction’lar tarafından commit edilmiş verilere erişim sağlar.

Avantaj: Dirty Read sorununu engeller.

Dezavantaj: Non-Repeatable Read, Phantom Read.

```
@Transactional(isolation = Isolation.READ_COMMITTED)
public void log(String message){
   // ...
}
```

REPEATABLE_READ

Verinin işlem süresince değişmediğini garantiler.

Avantaj: Dirty Read ve Non-Repeatable Read sorunlarını engeller.

Dezavantaj: Phantom Read.

```
@Transactional(isolation = Isolation.REPEATABLE_READ)
public void log(String message){
   // ...
}
```

SERIALIZABLE

İşlemlerin sıralı bir şekilde gerçekleştirilmesini garantiler.

Avantaj: Dirty Read, Non-Repeatable Read ve Phantom Read sorunlarını engeller.

Dezavantaj: Performans.

```
@Transactional(isolation = Isolation.SERIALIZABLE)
public void log(String message){
   // ...
}
```

Teşekkürler