NoSQL ve Aggregate Model

harezmi bilişim çözümleri

Aggregate Data Model

- Verinin RDBMS'deki tuple düzeyinden daha **kompleks bir yapı**da ifade edildiği model'dir
- NoSQL store'larda (graph store hariç) bu ortak kompleks yapıya **aggregate** adı verilir
- Aggregate, uygulama içerisinde verinin anlamlı ve consistent biçimde yönetilebilmesi için **temel birim**dir

Aggregate Data Model

```
//customer aggregate
{
    "id":1,"name":"Kenan",
    "billingAddress":[{"city":"Ankara"}]
}

//order aggregate
{
    "id":101,"customerId":1,
    "orderItems":[
        {"productId":1001,"price":10.90},
        {"productId":1002,"price":20.50}
    ],
    "shippingAddress":[{"city":"İstanbul"}],
    "orderPayment":["ccInfo":"111-111-111"]
}
```

Aggregate Data Model'in Avantajı

JAVA Eğitimleri

- NoSQL DB için aggregate temelli storage unit replication ve sharding'de avantajlıdır
- İlişkisel model'de ise aggregate model tuple'lara ayrışmakta ve veri arasındaki ilişki foreign-key'ler ile takip edilmektedir
- DB düzeyinde aggregate ile ilgili bir bilgi mevcut değildir
- Bu nedenle DB'nin verinin nasıl dağıtılacağı ile ilgili çıkarım yapması mümkün değildir

Aggregate Data Model'in Avantajı

JAVA Eğitimleri

- İlişkisel DB'lere bu nedenle aggregate-ignorant denir
- NoSQL dünyasında Graph DB'ler de aggregate ignorant'dır
- Eğer veri üzerinde belirli bir aggregate yapı yoksa aggregate ignorant bir veri modeli veriye erişim açısından daha iyi olabilir

Aggregate Model ve Transaction

- NoSQL DB'ler birden fazla aggregate'i kapsayan ACID transaction'larını desteklemezler
- Ancak belirli bir anda tek bir aggregate üzerinde atomik işlem yapılabilir
- Birden fazla aggregate üzerinde atomik işlem uygulama düzeyinde transaction yönetimi gerektirir

Materialized View

- Veriye, önceden belirlenmiş veri yapısının dışında hızlıca erişim ihtiyacı söz konusu olduğunda kullanılır
- İhtiyaç duyulan veriye hızlı erişim sağlamak için veri önceden hazırlanarak **cache**'lenir
- Buna materialized view adı verilir
- Aggregate-oriented DB'lerde de bu tür view'lara sıklıkla ihtiyaç duyulur

Materialized View

- View'ın oluşturulması için iki farklı strateji tercih edilebilir
  - Eager approach
  - Incremental approach (batch)
- Aynı aggregate içerisinde de materialized view kullanılabilir
- Materialized view için farklı column family'leri kullanmak da yaygın bir yaklaşımdır

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com