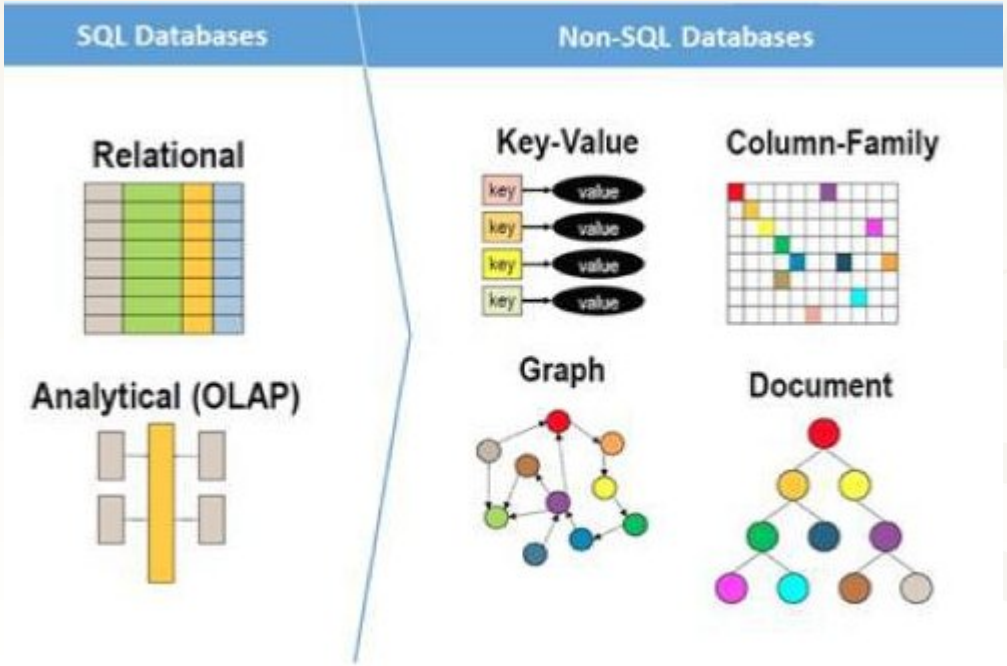
NoSQL Dünyasına Genel Bir Bakış

harezmi bilişim çözümleri

NoSQL Nedir?

- NoSQL şemsiye bir terimdir
- İlişkisel olmayan veritabanı teknolojilerini anlatmak için kullanılır



NoSQL Teknolojilerin Genel Özellikleri

JAVA Eğitimleri

- İlişkisel model yerine aggregate model'i tercih ederler
- Explicit bir şemaya sahip değillerdir, ancak uygulama düzeyinde implicit bir şema söz konusudur
- Graph DB'ler hariç cluster ortamlarla uyumludurlar
- Genelde açık kaynak kodlu sistemlerdir
- Web uygulamalarının ihtiyaçlarına cevap verecek biçimde evrilmişlerdir

Neden NoSQL?

- In-memory veri yapıları ile ilişkisel veri yapıları arasında mapping önemli bir zaman almaktadır
- NoSQL veritabanları uygulamanın ihtiyaçlarına daha doğrudan kabiliyetler sunarak bu süreci kolaylaştırıp hızlandırabilmektedir

Neden NoSQL?

- Uygulamalarda büyük miktarda veri tutulması ve bunun cluster sistemlerde yönetilme ihtiyacı söz konusudur
- İlişkisel veritabanları daha çok tek makine üzerinde çalışmaya yönelik tasarlanmışlardır
- NoSQL veritabanı sistemleri ise big data senaryoları için ve doğrudan cluster sistemler üzerinde çalışmaya yönelik tasarlanmışlardır

RDBMS Kullanımının Kapsamı

- Integration Database
  - Birden fazla uygulamanın verilerini birbirleri ile paylaşmasını sağlar
- Application Database
  - Her bir uygulamanın verisi diğerlerinden bağımsız biçimde yönetilir

Data Model vs Storage Model

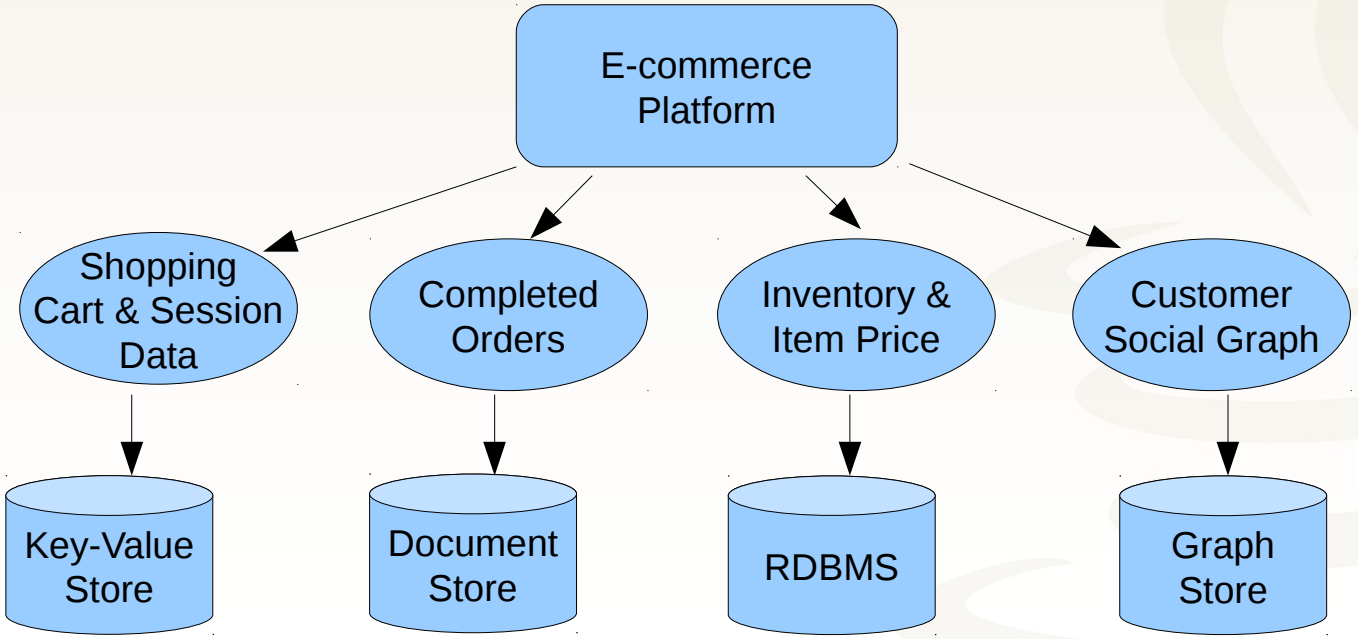
- Data Model
  - Verinin nasıl görüntüleneceğini, kullanıcı tarafından nasıl yönetileceğini tanımlar
  - Uygulamanın veri ile nasıl çalışacağını belirler
- Storage Model
  - Verinin veritabanında nasıl tutulacağını ve yönetileceğini belirler
  - Uygulama davranışı açısından dikkate alınmamalıdır
  - Ancak performans açısından dikkate alınmak zorundadır

Data Model

- Değişik veri model yaklaşımları söz konusudur
  - Relational model
  - Analytical (OLAP) model
  - Aggregate model

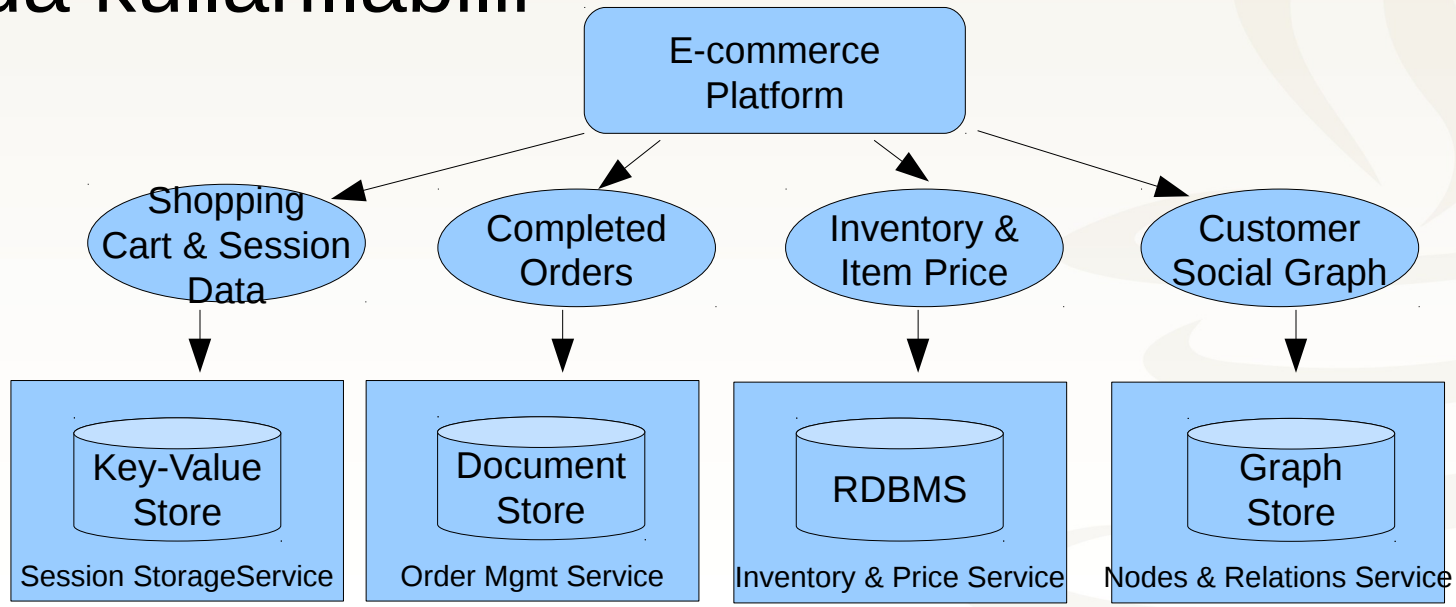
Polyglot Persistence

- Bir uygulama içerisinde farklı veri model'lerinin ve buna bağlı olarak farklı persistence stratejilerinin kullanılması demektir



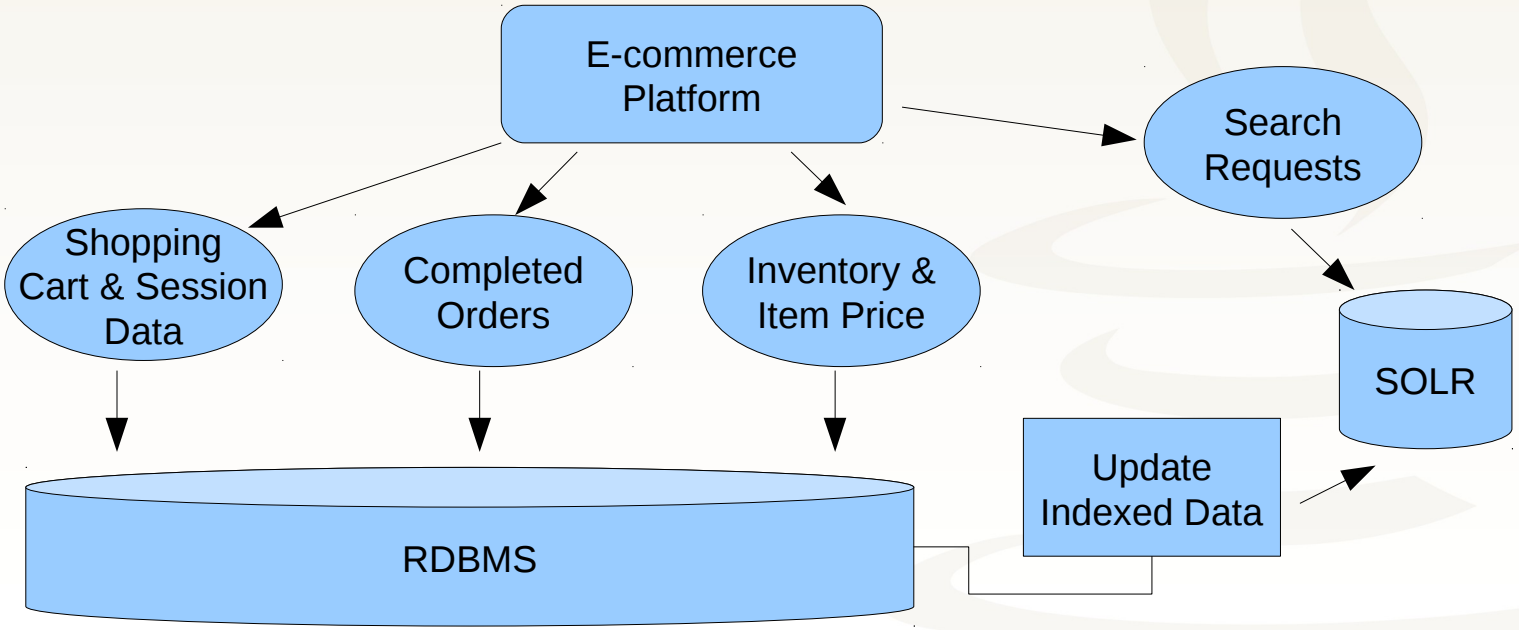
Polyglot Persistence

- Persistence katmanının ayrı bir servis katmanı arkasında izole edilmesi önem arz eder
- Böylece veri diğer uygulamalar tarafından da kullanılabilir



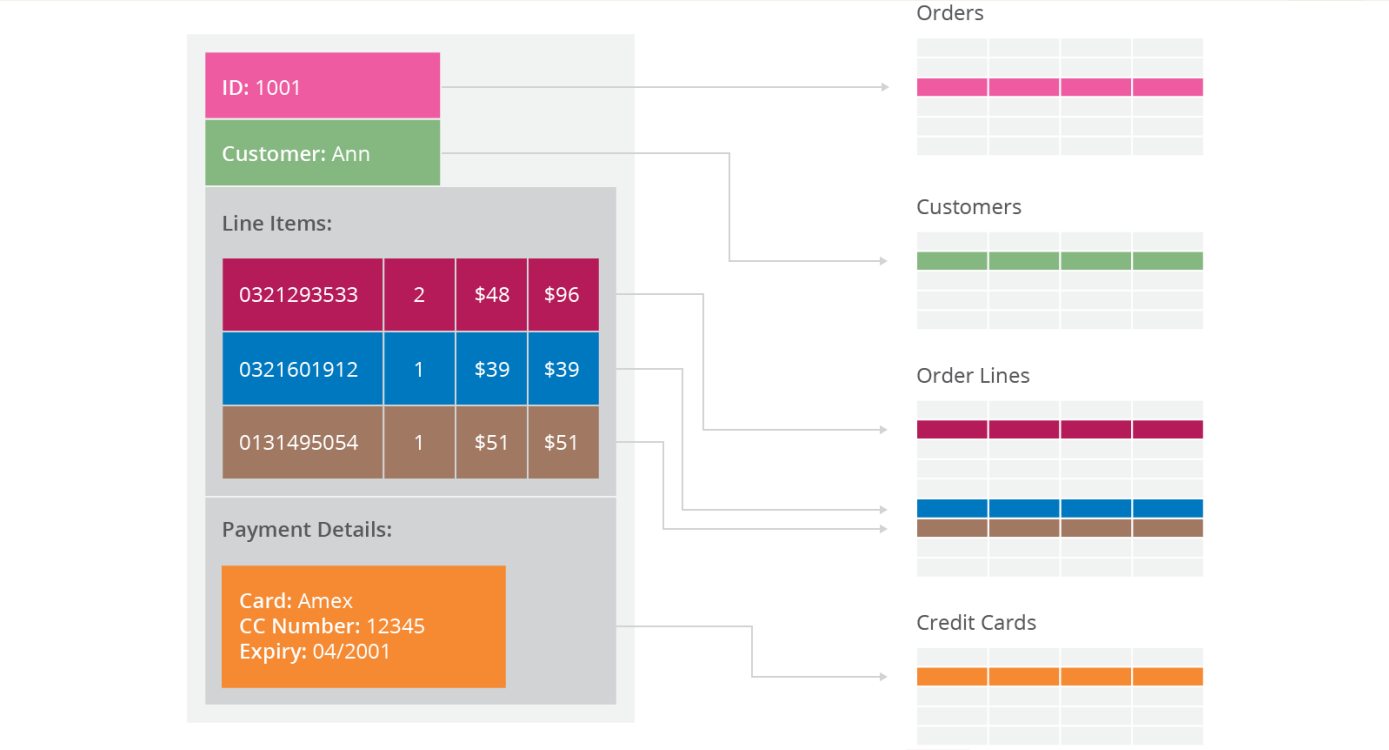
Polyglot Persistence

- Legacy sistemlerin RDBMS'e bağımlı olmalarından dolayı yeni data store kullanımı RDBMS ile data store arasındaki senkronizasyon entegrasyonu ile sağlanabilir



Relational Model

- İlişkisel model veriyi tuple (row)'lara ayırarak yönetir



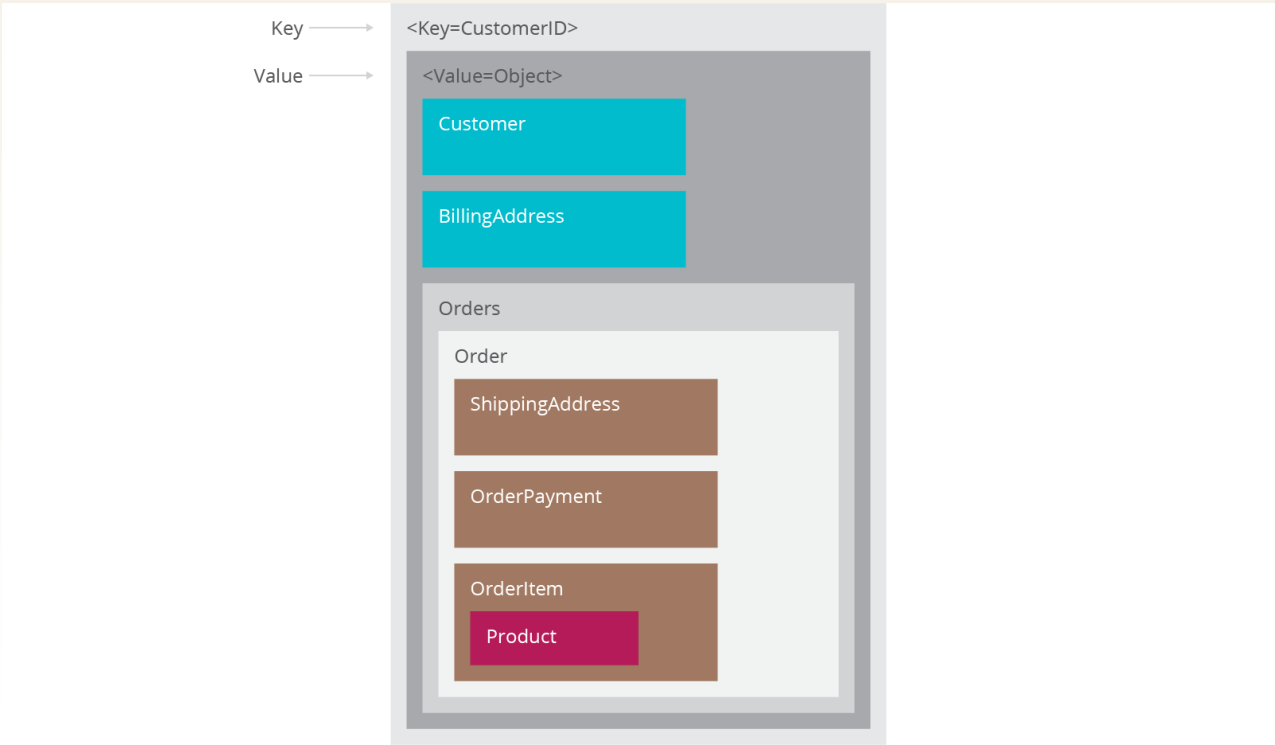
Analytical (OLAP) Model

Key-Value Stores

- Veri key-value ikilileri şeklinde tutulur
- Veri üzerinde işlemler key vasıtası ile gerçekleşir
  - Get the value for the key
  - Put a value for a key
  - Delete a key
- Hash tables & hashing key-value store'ları anlamak için iyi bir başlangıç noktasıdır

Key-Value Stores

- Value BLOB bir değerdir, DB açısından içeriğinin bir önemi yoktur



Key-Value/Tuple Store Databases

- Redis
- Riak
- Memcached
- BerkeleyDB
- LevelDB
- Project Vodemort

Ne Zaman Key-Value DB Tercih Edilmeli?

- Session, user profile/preferences, shopping cart data gibi bilgileri tutmak için uygundur
- Data üzerinden sorgu yapmak gerektiğinde sorun yaratabilir
- Bu durumda veri arasında ilişki tanımları veya multiple-key kullanımı faydalı olabilir

Document Stores

- Doküman kavramı burada ana unsurdur
  - Self describing
  - Hierarchical tree data structures
  - Consist of maps, collections, scalar values
- Bu dokümanların formatı XML, JSON veya BSON olabilir

Key/Value vs Document Stores

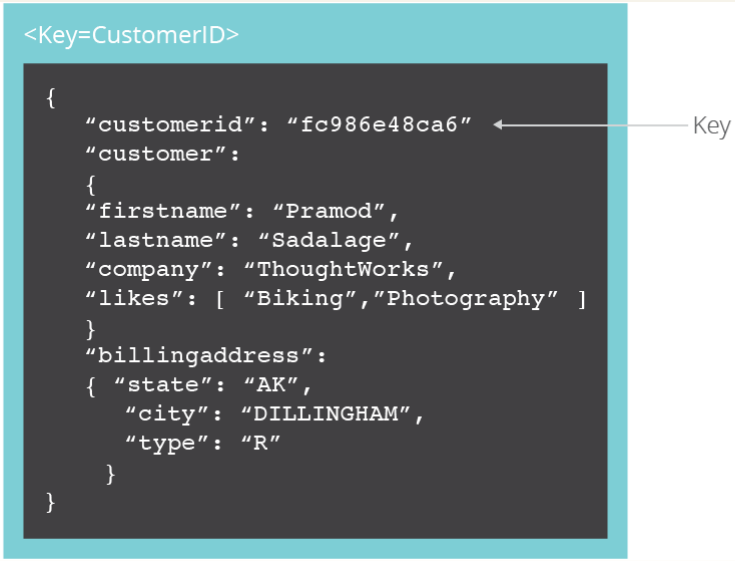
- Her ikisi de aggregate model'e uygundur
- Ancak key/value store için aggragete'in yapısının/içeriğinin hiçbir önemi yoktur (opaque)
- BLOB bir alandır
- Bu sayede, kapasitesi elverdiği ölçüde istediğimiz aggregate içeriği DB'de saklayabiliriz

Key/Value vs Document Stores

- Document store'larda ise aggregate'in sahip olacağı yapı/format önem arz eder
- Dolayısı ile aggregate içeriği olarak tutulabilecek veride kısıtlar söz konusu olabilir
- Ancak aggregate içeriğini sorgulama ve içeriğe erişim açısından key/value store'a göre daha avantajlıdır

Document Stores

- Document store'ları anlamak için de JSON iyi bir başlangıç noktasıdır



Document Databases

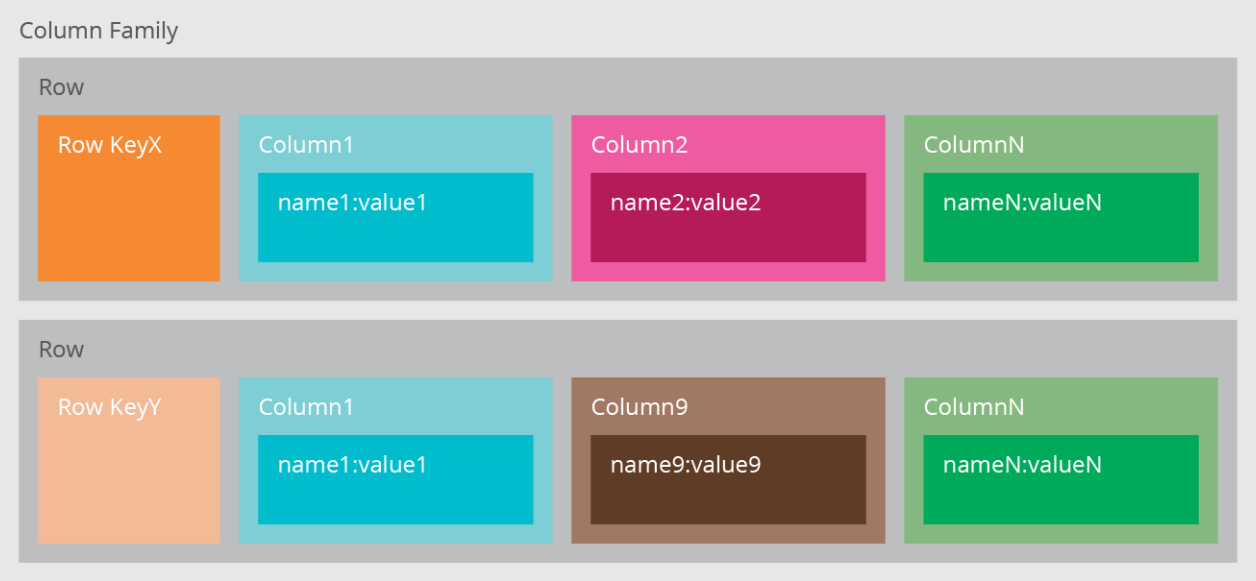
- MongoDB
- CouchDB
- RavenDB
- Terrastore
- OrientDB

Ne Zaman Document DB Tercih Edilmeli?

- Content management sistemler, blog siteleri, web/real-time analitik işlemleri için oldukça uygundur
- Birden fazla operasyonu span eden transactional işlemler için uygun değildir
- Farklı ölçekte aggregate veri yapıları üzerinde sorgu ihtiyaçları söz konusu olduğunda da sorun yaratabilir

Column Family Stores

- Her bir row içerisinde birden fazla column birbiri ile ilişkili biçimde tutulur
- Birbiri ile ilişkili bu veriye genellikle birlikte erişilir



Column Family Stores

- Key-value store'lara kıyasla **iki seviyeli aggregate** yapıları (two level aggregate) da denebilir
- **İlk seviye**deki key **satıra erişim** sağlar (row key/identifier)
- **İkinci seviye**de ise **sütunlara erişilir**
- Her bir sütun da name-value ikililerinden oluşmaktadır

Column Family Databases

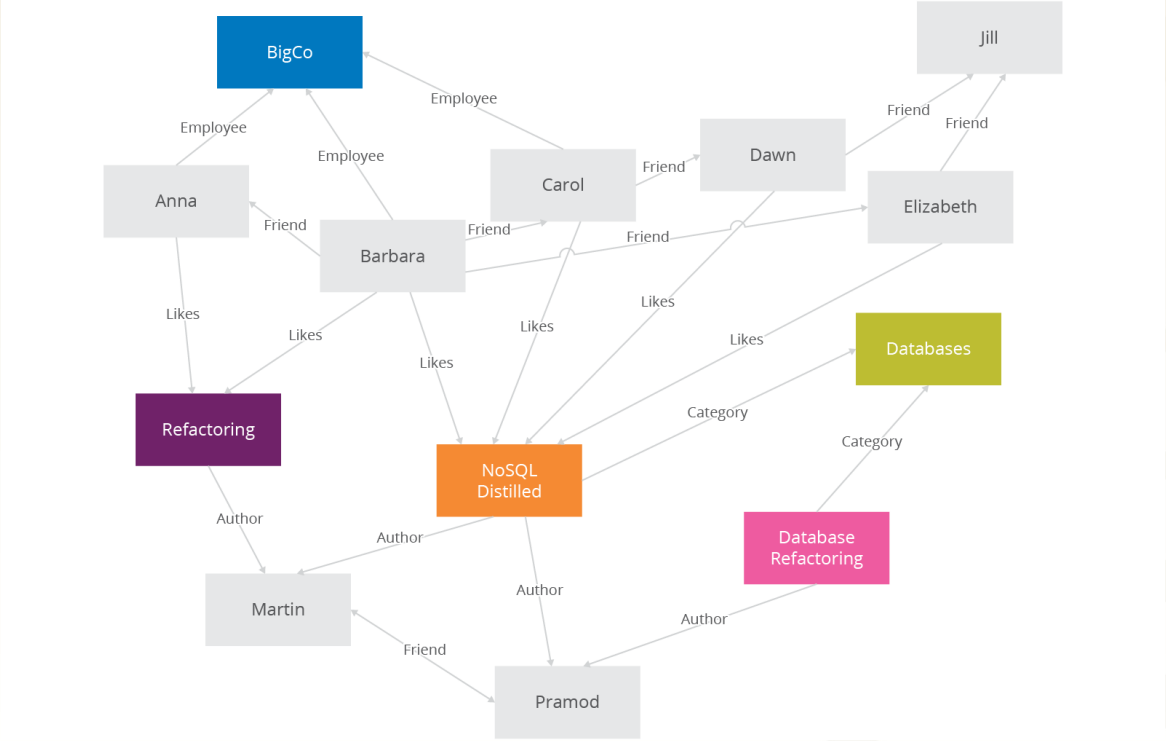
- Hadoop/Hbase
- Hypertable
- Cassandra
- Apache Flink
- Amazon SimpleDB

Ne Zaman Columnar DB Tercih Edilmeli?

- Content management sistemler, blog siteleri, sayaç (counter) yönetimi, zaman içerisinde süresi dolan kullanım durumlarının yönetimi (expiring usage), log aggregation gibi yazma yükü fazla işlemler için uygundur
- Geliştirme sürecinin başındaki sistemlerde sorgu örüntülerinin tam olarak ortaya çıkmadığı durumlarda kullanımından kaçınılmalıdır

Graph Database

- Veri node ve edge'ler (relationship) şeklinde ifade edilir



Graph Database

- Entity ve entity'ler arası ilişkileri (relationships) tutar
- Entity'lere node adı verilir ve property'leri de vardır
- İlişkilerde'de edge adı verilir, bunların da property'leri vardır
- Sorgularda bu property'ler de kullanılabilir

Graph Model vs Relational Model

- İlişkisel model'de sadece tek bir tip ilişki tutulur
- Farklı türde ilişkiler eklemek için şema ve veri üzerinde değişiklikler yapmak gerekebilir
- İlişkisel model belirli bir data traversal path'i karşılamaya yönelik düzenlenir
- Data traversal değişirse, data'nın da değişmesi gerekir

Graph Model vs Relational Model

JAVA Eğitimleri

- Graph model'de ilişkiler ve join'ler üzerinde traversal oldukça hızlıdır
- İlişkiler öncesinde persist edilmiştir, runtime'da her sorgu için tekrardan hesaplanmaz
- Node'lar arasında farklı türlerde ilişkiler olabilir

Graph Store vs Aggregate Model

- Graph DB'ler aggregate ignorant'tır
- Node'lar arasındaki edge (relationship)'ler ön plandadır
- Node ve edge'lerin birlikte atomik olarak ele alınması önem arz edebilir
- Dolayısı ile aggregate odaklı NoSQL DB'ler gibi cluster sistemlere çok uygun olmayabilir

Graph Databases

- Neo4J
- Sparksee
- InfoGrid
- HyperGraphDB
- GraphBase
- Infinite Graph

Ne Zaman Graph DB Tercih Edilmeli?

- Sosyal ağlar, mekansal (spatial) veri ve rota oluşturma (routing), işlemlerinde kullanımı uygundur

Diğer NoSQL Veritabanı Teknolojileri

- Object Databases
- Grid & Cloud Databases
  - Oracle Coherence
  - Gigaspaces
  - Gemfire
  - Infinispan
  - Hazelcast
- XML Databases

Diğer NoSQL Veritabanı Teknolojileri

- Multidimensional Databases
- Multivalue Databases
- Event Sourcing
- Time Series/Streaming Databases

Kaynaklar

- NoSQL (Your Ultimate Guide to the Non-Relational Universe!)
- 7 Steps to Understanding NoSQL Databases
- NoSQL Databases: An Overview
- NoSQL Distilled Book

İletişim

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