Veri Saklama Yöntemleri ve Büyük Veri

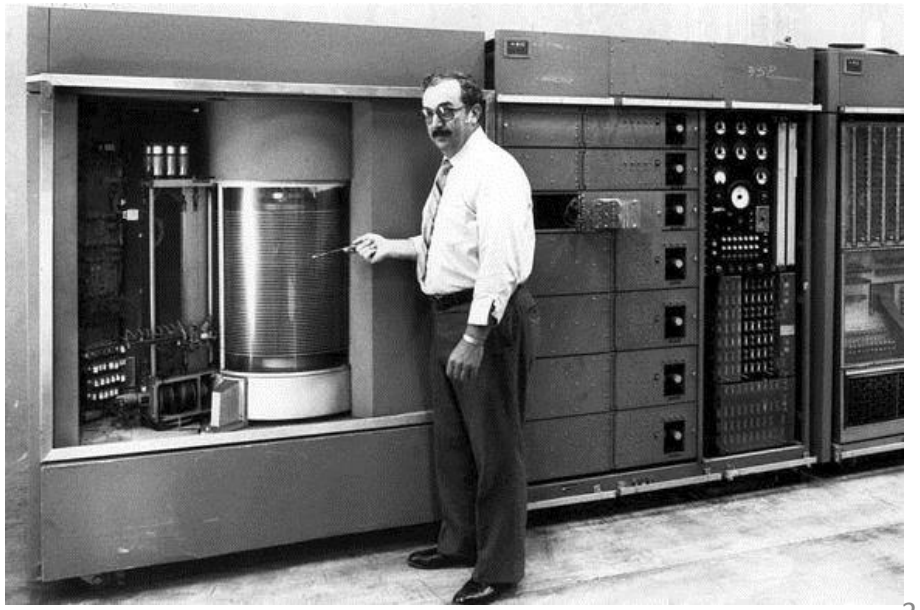
SULTAN TURHAN

Veri nereden toplanır ?

- Bilgi sistemleri
  - Kamu (Nüfus İdaresi, Sağlık Sistemleri, Vergi Sistemleri v.b.)
  - Özel (Uçuş takibi, Lojistik Firmaları v.b.)
- Akıllı telefonlar
  - Telekommünikasyon altyapısı
- Sensörler ~ Akıllı cihazlar
- Giyilebilir cihazlar
- Internet
- E- Ticaret Sistemleri
- Sosyal Medya

Veri Saklama (Donanım)

- 1956 - İlk Sabit Disk icat edildi.
- 1963 - Philips, ilk kompakt ses kasetini icat etti (Sony'nin Walkman'in icadıyla müzik dağıtımı için popüler oldu.)
- 1971 - IBM, plastikle kaplanmış manyetik filmden yapılmış ilk 8” lik disketi piyasaya sürdü
- 1975 - Alan Shugart, 5.25”lik disketi 110 KB depolama kapasitesine sahip bir hale getirdi.
- 1980 - James T. Russell, ilk kompakt diski (CD) geliştirdi
- 1981 - 3,5 ”lik disket geliştirildi. Bu yeni disketin, sert bir metal kapağı vardır ve daha fazla saklama alanına sahipti.
- 1984 - Daha fazla veri depolamak için Salt okunabilir CD-Romlar üretildi.
- 1994 - Compact Flash üretildi.
- 1995 - DVD, yeni nesil disk depolama alanı oldu.
- 1999 - USB bellekler taşınabilir depolamalar için çözüm oldu.
- 2003 - Yeni Blu-Ray ile daha fazla veri depolama imkânı doğdu
- 2017 – Bulut sistemleri



Veri Saklama (Yazılım)

- Metin tabanlı dosyalar
- Hesap tabloları
- Veri tabanı (Database)
- Veri ambarı (Datawarehouse)
  - Data Mart
- Veri gölü (Data Lake)
- Bulut Depolama çözümleri : Data as a Service (DaaS)

Veri Modeli

- Dijital depolama ve internetin icadıyla veriyi fiziksel olarak saklama imkânlarımız gittikçe artan bir hızla gelişmiştir.
- Verilerin derlenmesi, gerektiğinde ilişkilendirilip bir enformasyon haline dönüştürülebilmesi için belirli bir düzen ve sistematik doğrultusunda kayıt altına alınmaları gerekir.
- **Veri modeli**; verileri, veriler arasındaki ilişkileri, uygulamadaki verilere koyulan sınırlamaları gösteren soyut ifade biçimidir
- Veri modeli, dosya içinde bulunan ve dosyalar arasında iletilen, bağlantı oluşturulan **veri yapısını** belirler.
- Problemin çözümü için **kavramsal** bir yaklaşım yöntemidir.

Veri Modeli

Bugüne kadar geliştirilmiş olan çok sayıda veri modeli vardır. Geçmişte ve günümüzde yaygın kullanılan veri modellerini 4 grupta toplamak mümkündür:

1. Sıradüzensel Veri Modeli (Hierarchical Data Model)
2. Ağ Veri Modeli (Network Data Model)
3. İlişkisel Veri Modeli (Relational Data Model)
4. Nesneye-Yönelik Veri Modeli (Object - Oriented Data Model)

Yukarıdaki sıralama aynı zamanda kronolojik bir sıralamadır.

Veri tabanı nedir ?

Veri Tabanı,

- **bütünleşik,**
- **yapılandırılmış,**
- **birbirleriyle ilişkilendirilmiş ,**
- **bir uygulama amacıyla oluşturulmuş,**
- **kendine ait özel fiziksel ortama yerleştirilmiş**

veri kümesidir

Veri tabanı üzerinde veriler, tekrara yer vermeden çok amaçlı kullanıma olanak sağlayacak şekilde modellenerek depolanırlar.

Veri tabanının yapısı kullanılan soyut veri modeline göre belirlenir.

İlişkisel Model (Relational Model)

- 1970 yılında Edgar Frank «Ted» Codd tarafından bulunmuştur.
- İlişkisel model (relational model), günümüzde en yaygın biçimde kullanılan bir veri tabanı modelidir.
- Varlıklar arasındaki bağlantının, içerdiği değerlere göre sağlanması esasına dayanır. İlişkisel model, varlıklar arasında oluşan karmaşık ilişkileri basite indirgemek amacıyla geliştirilmiştir.

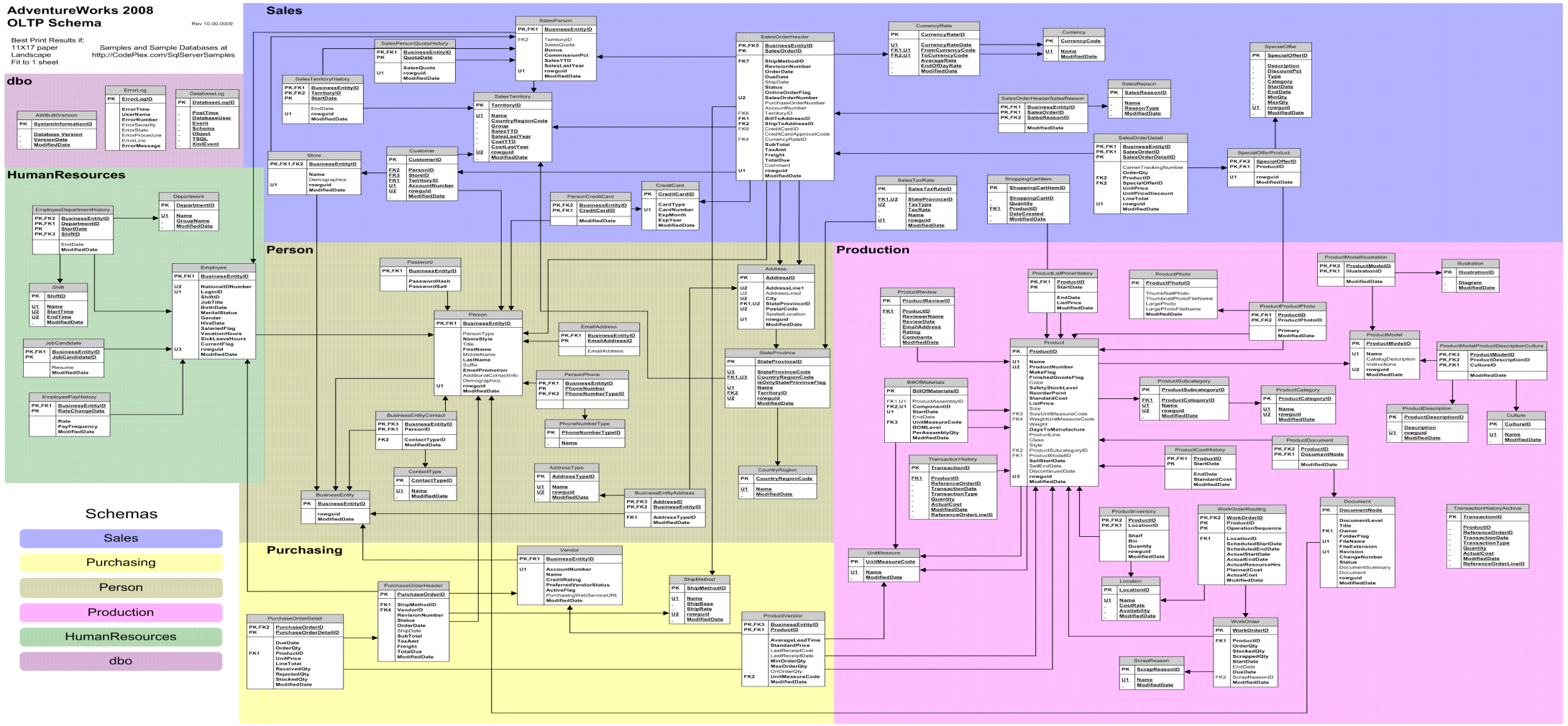
Kavramlar

- İlişki (Relation): Satır ve sütunlardan oluşan iki boyutlu tablo
- Özellik (Attribute): Varlıkların niteliklerinin tanımlandığı, ilişkinin adlandırılmış sütunları
- Alan (Domain): Herhangi bir özelliğin alabileceği değerler kümesi
- Satır (Tuple):İlişki(tablo)deki bir satır
- Derece (Degree): İlişkinin özelliklerinin sayısı
- Nicelik (Cardinality): İlişkinin gerçeklenme sayısı
- İlişkisel veri tabanı (Relational Database): Normalleştirilmiş ilişkiler kümesi

Anahtarlar

- Bir varlık kümesi içindeki varlıkları ya da bir ilişki kümesi içindeki ilişkileri birbirinden ayırt etmek için kullanılan nitelik ya da nitelik grubuna bu varlık ya da ilişki kümesinin anahtarı denir.
- Anahtar Türleri :
  - **Birincil anahtar (Primary key) :** Varlığın özellikleri arasından seçilmiş ya da türetilmiş, ilişki (tablo)deki her bir satırı tekil (unique) halde tanımlayan niteliktir.
  - **Yabancı anahtar (Foreign Key):** İlişkisel veri tabanlarının temelini oluşturan Referans Bütünlük Kısıtlamalarının tanımlayan yapıdır. Bir ilişkinin başka bir ilişkinin aday anahtarı ile eşleşen özellik ya da özellikler kümesidir.

İlişkisel Veri Tabanı Örneği



Yapısal Sorgulama Dili (SQL-Structured Query Language)

- Veri Tanımlama Dili (DDL-Data Definition Language)
- Veri Yönetim Dili (DAL-Data Administration Language)
- Veri İşleme Dili (DML-Data Manipulation Language): Bir tablodaki veriler üzerinde sorgulama, ekleme, güncelleme ve silme işlemleri gerçekleştirir.
  - Select : Kayıt sorgulama komutu
  - Insert: Kayıt ekleme komutu
  - Update: Kayıt güncelleme komutu
  - Delete: Kayıt silme komutu

Veri Tabanı Yönetim Sistemi (VTYS)

- Veri tabanları ile kullanıcılar arasındaki etkileşim **Veri Tabanı Yönetim Sistemi** adı verilen yazılımlar ile gerçekleştirilir.
- Veri tabanı Yönetim Sistemi (VTYS), kullanıcılara veri tabanı oluşturma ve bu yapıyı yürütme olanağı sağlayan programlar bütünüdür. VTYS, çeşitli kullanıcı ve uygulamaların veri tabanını tanımlama, biçimleme, değiştirme, paylaşım ve koruma işlemlerini yürüten genel amaçlı yazılım sistemidir.
- Her VTYS **belirli bir veri modelini** kullanır.
- Bir VTYS'yi kullanarak oluşturulacak her veri tabanında yer alacak veriler ve veriler arası ilişkiler, mantıksal düzeyde ilgili veri modeline göre düzenlenir; bu veri modeli kullanılarak veri tabanının kavramsal ve dış şemaları oluşturulur.

VTYS – Amaç ve İşlevler

AMAÇ

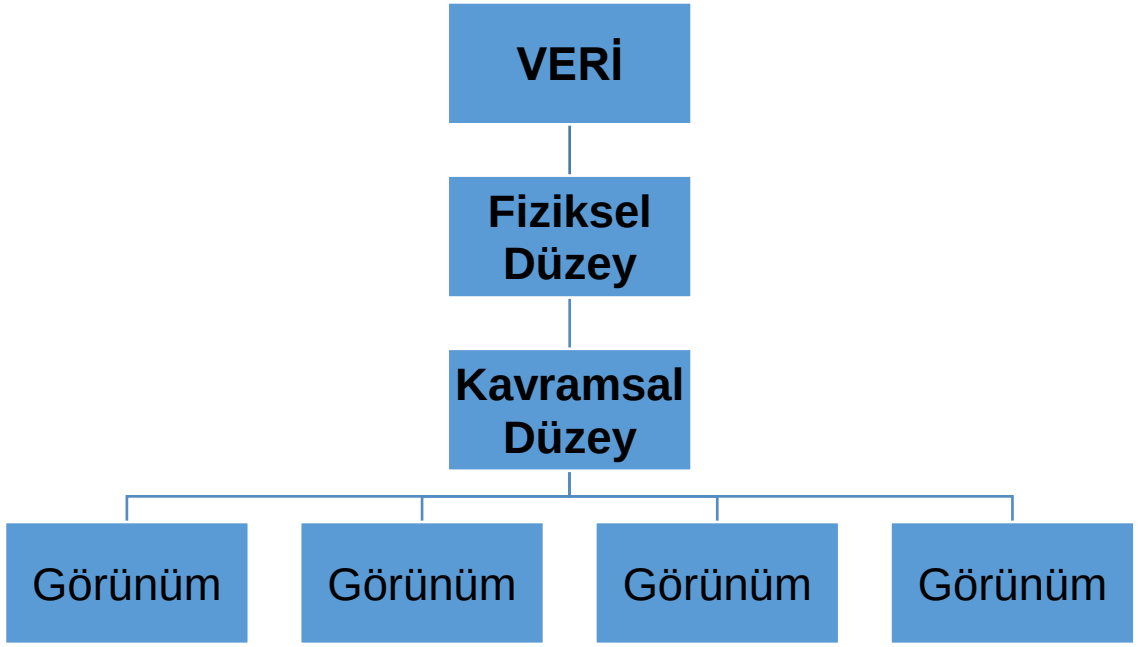
Kullanıcıya verinin soyutlanmış görüntülerini sağlamaktır. Sistem verinin nasıl ve nerede saklandığı gibi ayrıntıları saklar. Kullanıcıların çoğu bilgisayar konusunda eğitimli olmadığından verinin çeşitli seviyelerde soyutlanması ile karmaşık yapıyı gizler ve kullanıcıların sistemle etkileşimini basitleştirir.

İŞLEVLER

- **Tanımlama:** Veri tipleri, yapıları ve uygulamaları
- **Biçimleme:** VTYS tarafından kontrol edilen kayıt biriminde saklama
- **Değiştirme:** Saklanan bilginin sorgulanması, gerçek hayatta temsil ettiği örneğin değişimlerine göre güncelleme, rapor üretilmesi
- **Paylaşım:** Aynı anda birden çok kullanıcı ve uygulamanın erişimine izin verme
- **Koruma:** Yazılım ve donanım hatalarına, yetkisiz ve zararlı erişime karşı

VTYS DÜZEYLERİ

- Fiziksel Düzey
  - Verinin nasıl saklandığı, veriye nasıl ulaşılacağı vb. tanımlayan en düşük düzeydir.
- Kavramsal Düzey
  - Saklanan verinin ve bu veriler arasındaki ilişkilerin ne olduğunu, kullanıcıları, yetki ve güvenlik kısıtlamalarını tanımlayan ikinci düzeydir.
- Görünüm (Dış) Düzeyi
  - Veri tabanının ilgili kullanıcı grubu için ilgili parçasını veren, geri kalanını saklayan en üst düzey



On Line Transactional Processing (OLTP)

- OLTP sistemler genellikle ilişkisel veri tabanları üzerine kurulmuş, üzerinde sürekli işlem yapılan veri tabanları sistemleridir. Adından da anlaşılacağı gibi çok fazla transactional işlemler içeren yani INSERT, UPDATE, DELETE şeklinde DML (Data Manipulation Language) işlemleri içeren bilgi sistemleridir.
- OLTP sistemlerde sürekli yoğun işlemler yapılır.
- Sağlam bir ilişkisel yapı üzerine kurulmuştur
- Günlük hayatta kullanılan sistemlerin çoğu OLTP ürüne kurulu sistemlerdir.

Çok Boyutlu Veri Modelleme

- Verilerin çok olması bu verilerin faydalı veriler olduğu anlamına gelmez. Bu verileri işleyebilecek, üzerinde analizler yapabileceğimiz forma dönüştürdüğümüz zaman bu veriler pahalı veri haline dönüşür.
- İlişkisel modelle iki boyutlu olarak modellenmiş veriler standart sorgulara cevap vermektedir.
- Üstelik bu veriler normalize edilmiştir. Sorgular belirli kalıplar çerçevesinde yapılabilmektedir.
- Verinin sorgulamasında katı bağıntılar sorgu süresinin uzamasına neden olmaktadır.

Veri Ambarı

- Veri ambarı, çeşitli iç ve dış veri kaynaklarından elde edilen verilerin uygun dönüşümler yapılarak birleştirilmesiyle oluşturulan veri kaynağıdır.
- Aktif veri kaynağı en son haliyle veri ambarına yansıtılır. Veri ambarına girmiş veriler üzerinde değişiklik yapılamaz.
- Veri ambarı raporlama, analiz ve veri madenciliği amaçlı olarak kullanılır.
- Veri ambarı, kritik kararlar için üst yöneticilere bilgi sağlayan karar destek sisteminin veri kaynağıdır.

Veri Ambarı Özellikleri

- Veri ambarı **konu yönelimlidir**.
  - Veri ambarı belli bir konuyu analiz etmek üzere kullanılır. Örneğin *satış verileri* gibi.
- Veri ambarı **bütünleşiktir**.
  - Veri ambarı farklı kaynakların birleştirilmesiyle elde edilir.
  - A ve B veri kaynaklarında belli bir ürünün öznitelikleri farklı olabilir, ancak veri ambarında bunlar tek olmalıdır.
- Veri ambarının **zaman boyutu** vardır.
  - Tarihsel veriler veri ambarında tutulur.
  - Online sistemde müşterinin son adresi tutulurken, veri ambarında önceki adresleri de tutulur.
- Veri ambarındaki veriler **kalıcıdır**. Veri ambarına giren bir veri değiştirilemez.

Veri Ambarı Modelleme – 1

Star Schema – Yıldız Şema

Yıldız Şema yandaki şekilde gösterildiği gibi merkezde bir olgu tablosu ve etrafını saran çok boyutlu tablolardan oluşur.

Veri tabanında bir tane olgu tablosu ve her bir boyut için bir tane boyut tablo yer alır.

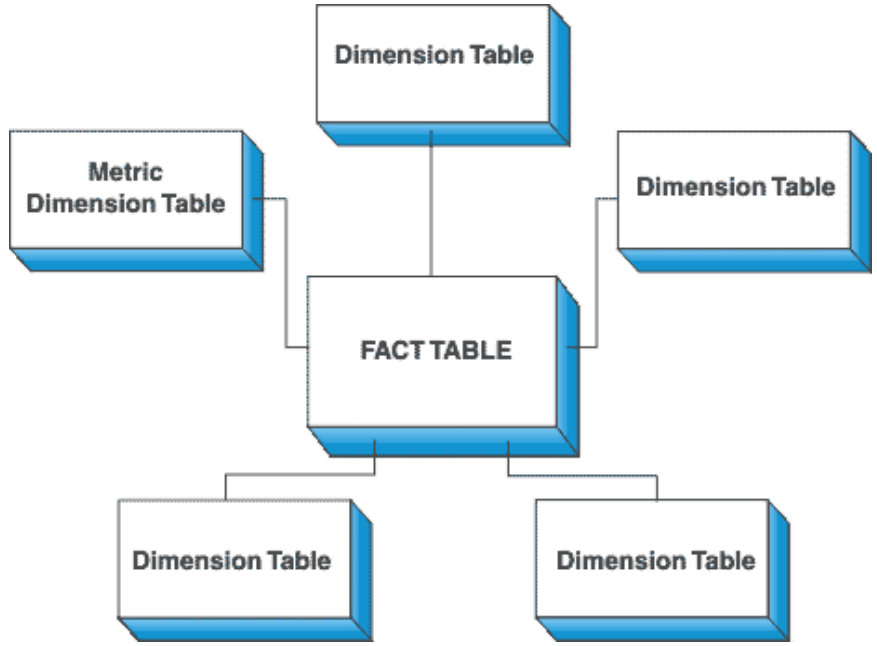
Her boyut tablosunda, boyuta ait nitelikleri tutan kolonlar yer alır.

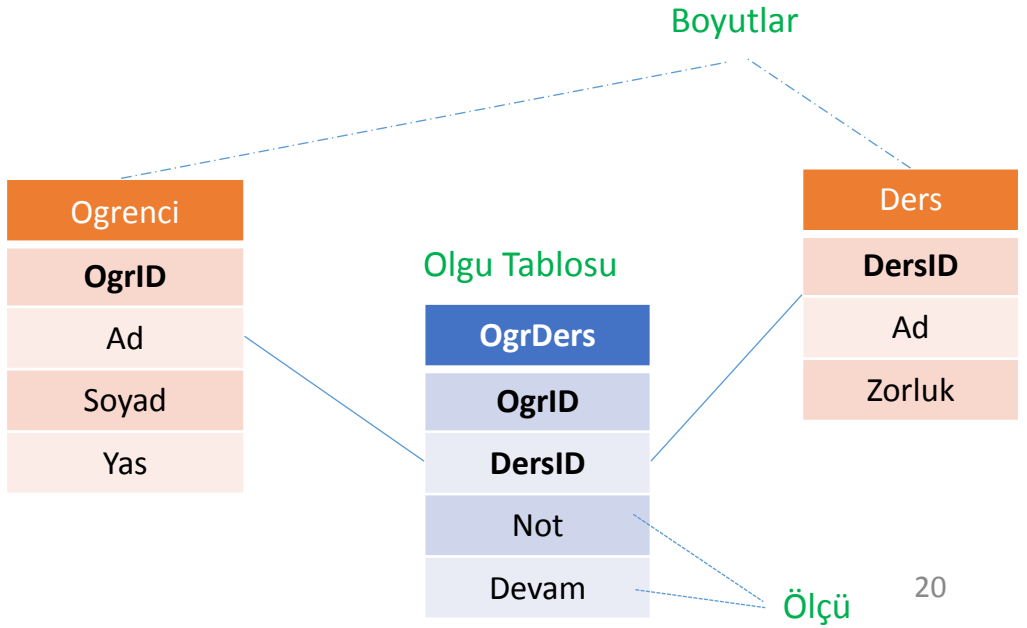
Olgu tablosundaki her çoklu her bir boyut tablosunu referans veren göstergeler içerir.

Yıldız şemanın etrafında bulunan bu tablolar var olan spesifik iş süreçlerini göstermekte ve sayısal veriler hakkında bilgi sunmak amacıyla kullanılırlar.

Verinin çekilmesinde, gösteriminde ve analiz aşamasında kolaylık sağlayan bir yapıdır.

Verinin modellenmesi tamamen denormalizedir.





Veri Ambarı Modelleme – 2

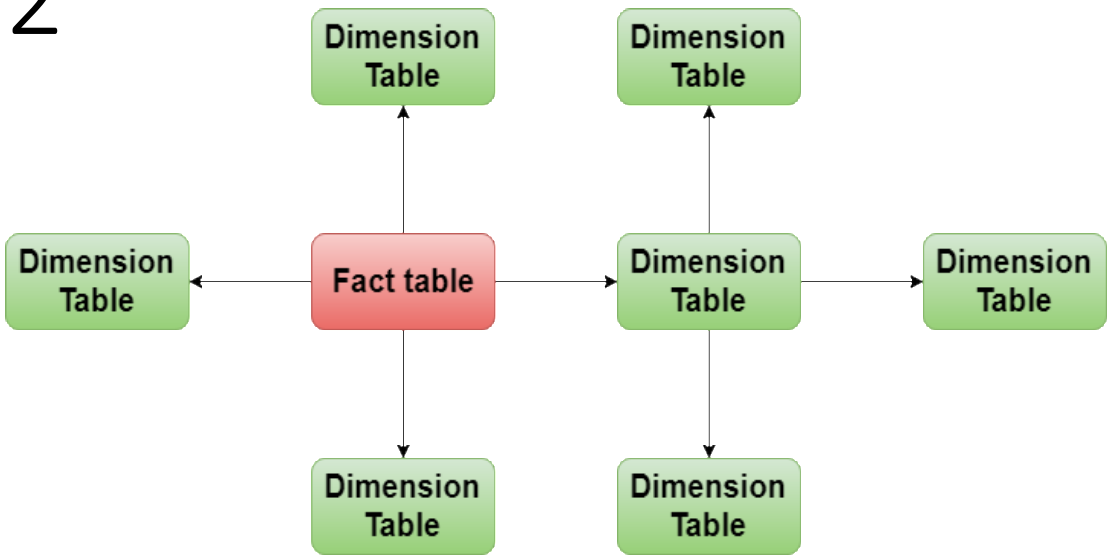
Snowflake Schema – Kar Tanesi Şema

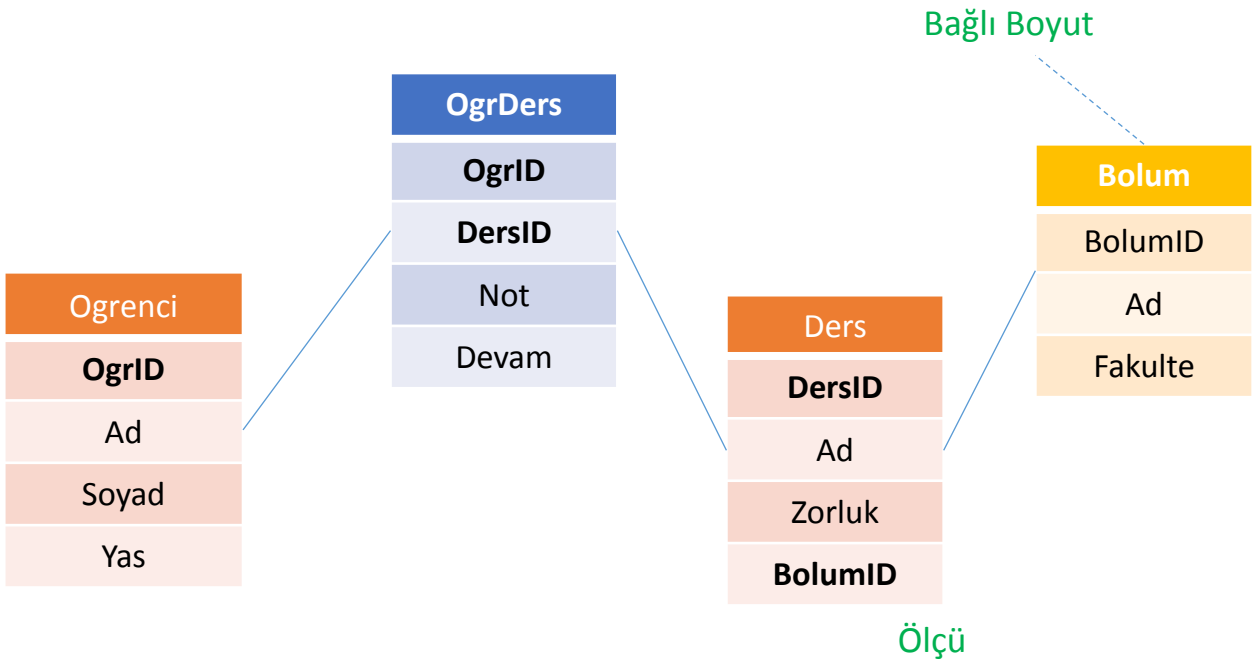
Kar tanesi şeması yıldız şemasının boyut hiyerarşilerinin bir çok boyut tablosunda saklanmış şeklidir.

Kar tanesi şeması verileri farklı tablolara böler ve böylece yıldız şemanın aksine, normalizasyon imkânından daha fazlasını sunar.

Sadece tek bir tablo olgu tablosu ile ilişkilendirilir, diğer boyut tabloları da ikincil anahtarları ile bu ilişkiye dâhil olurlar.

Bir boyut tanımlamak için birden çok boyut tablosu kullanılabilir





Data Mart

- Data Mart, Veri ambarının alt kümesi olarak tanımlanabilir. Veri ambarı bir iş probleminin tamamına yönelik bir bakış sağlarken, Data Mart sadece belli bir kısma bakış sağlar.
- Kurum genelinde karar alıcıların, işe ait tüm veriler üzerinde analiz yapmasına gerek olmayabilir. Bu kişiler sadece kendi birimleriyle ilgili verilere ulaşarak bunlara bağlı analizler yapmayı isteyebilirler, bu durumda veri ambarındaki tüm karmaşıklık içinde boğulmalarına gerek yoktur.
- Veri ambarının sadece bir konu kapsamında alt kümesini temsil eden data mart, veri ambarı kadar ayrıntılı veri barındırmazlar. Bu yüzden kolay anlaşılabilir ve yönlendirilebilirler.

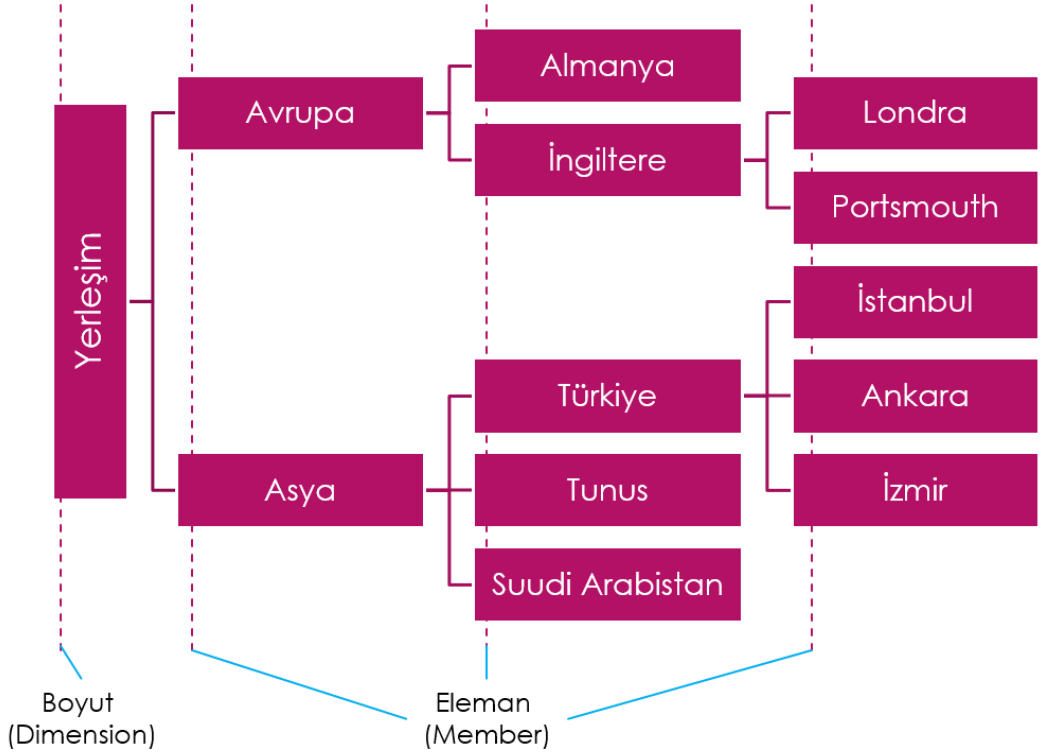
Online Analytic Processing (OLAP)

- Veri ambarı ile sağlanan veri kaynağı temelinde karar vermeye yardımcı olacak şekilde yapılan veri analizi ve sorgulama işlemlerine OLAP denir
- OLAP, ilişkisel veri tabanın aksine veriyi tekrarlayarak oluşturulan, veriyi bu şekilde depolayan, raporlama ve analiz için kullanılan, veriye hızlı erişim sağlayan yapılardır.
- İş zekası çözümleri için OLAP kullanılır.
- OLAP yapısı ise daha yüksek performansla, var olan verileri anlamlandırmayı ve raporlamayı sağlar.
- OLAP sistemlerinin en önemli özelliği verilerin mutlaka zaman boyutu olmasıdır. Yani daima tarihsel (historical) veri üzerinde çalışır.
- OLAP temelde OLTP sistemlerinden beslenerek organizasyonun tamamı hakkında çok hızlı bir şekilde bilgi sağlanması amacıyla oluşturulmuş yapılardır.

OLAP / OLTP Karşılaştırması

Kriter	OLTP	OLAP
Amaç	Günlük iş fonksiyonlarını yerine getirmek	Karar vermeyi desteklemek ve iş ve yönetim sorgularını cevaplamak
Veri kaynağı	İşlem veri tabanı (etkinlik ve tutarlılığa dayanan normalize edilmiş veri deposu)	Veri ambarı veya özel veri tabanı (doğruluk ve tamlığa dayanan normalize edilmemiş veri deposu)
Raporlama	Rutin, periyodik, odaklanılmış raporlar	Özel amaçlı, çok boyutlu, geniş odaklı sorgular ve raporlar
Kaynak ihtiyaçları	Olağan ilişkisel veri tabanları	Çok işlemcili, yüksek-kapasiteli, özel veri tabanları
Sistem yönelimi	Müşteri odaklıdır, işlemler ve sorgular IT personeli veya müşteriler tarafından yapılır.	Konu odaklıdır ve karar vericiler, yöneticiler, analistler tarafından kullanılır.
Tasarım	Varlık-ilişki modeli	Yıldız, Kar Tanesi veya Galaksi modeli

OLAP – Temel Kavramlar



- **Küp:** Herhangi bir OLAP veritabanı içinde kaydetme ve geri alma işlemleri için kullanılan temel veri yapısıdır.
- **Boyut:** Bağımsız mantıksal bölümlerle veriyi organize etme bir yolunu sağlar.
  - Boyutu bir bilgi kategorisi gibi düşünmek uygundur.
  - Zaman, Yerleşim, Ürün
- **Eleman:** Boyutun alt kategorisi olarak düşünülebilir.
  - Asya, Türkiye, İstanbul
- **Ölçü (Measure):** Analiz edilmek istenen verilerdir..
  - Satış sayısı, kâr, çalışan sayısı
  - Ölçü kendi başına anlam ifade etmeyen bir değerdir. Ancak boyutlarla kullanıldığında anlamlı hale gelir.
  - Kâr = 300 | Asya 2015 Kârı 300
- **Öznitelik:** Boyutun iç hiyerarşiye sahip olmayan bir özelliğidir. (Müşteri – Şehir)
  - Bu ilişki başka durumlarda da kullanılabilir.

  (Kıta – Ülke – Şehir, Şehir – Cinsiyet – Eğitim)

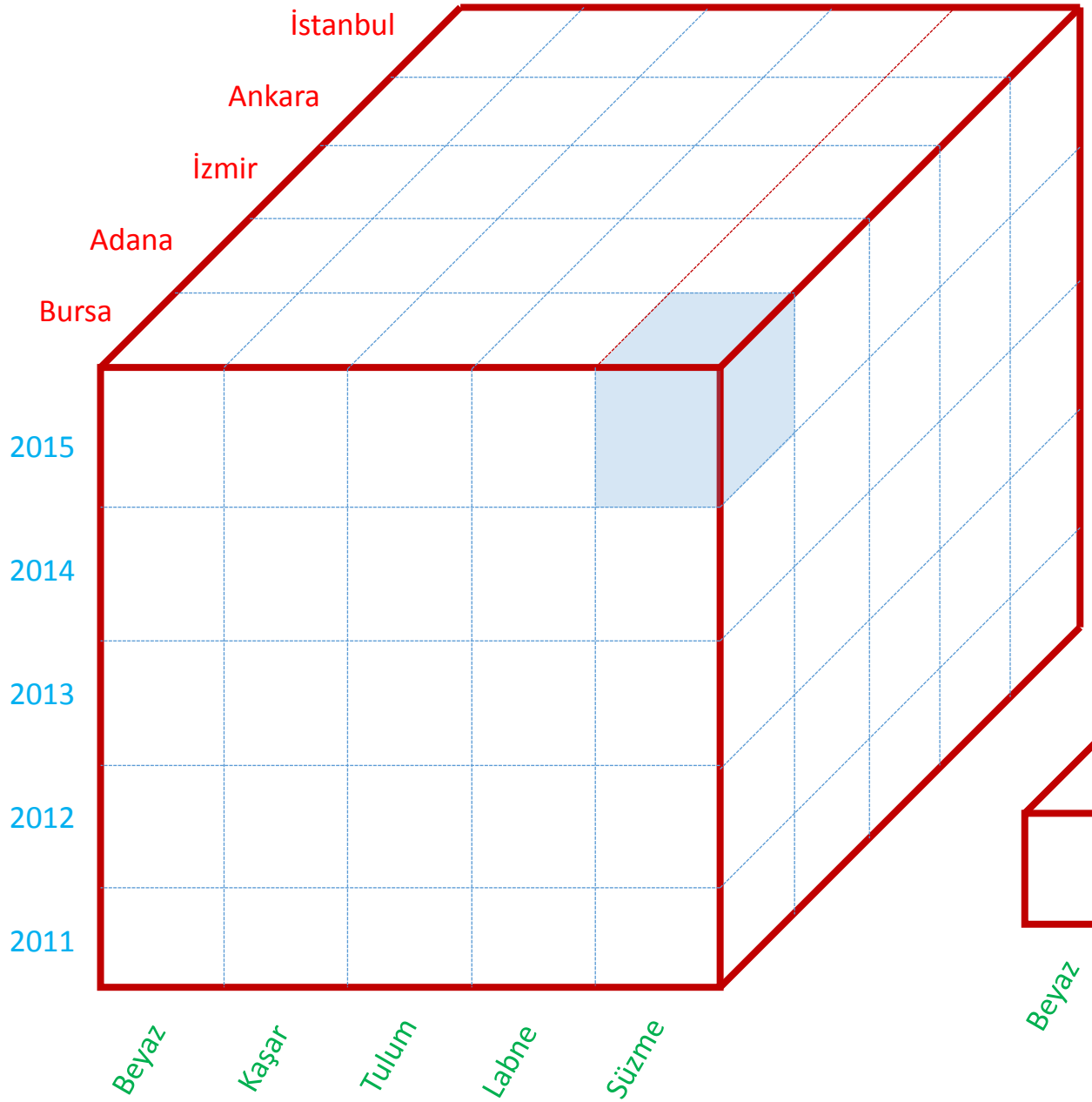
OLAP – Temel İşlemler

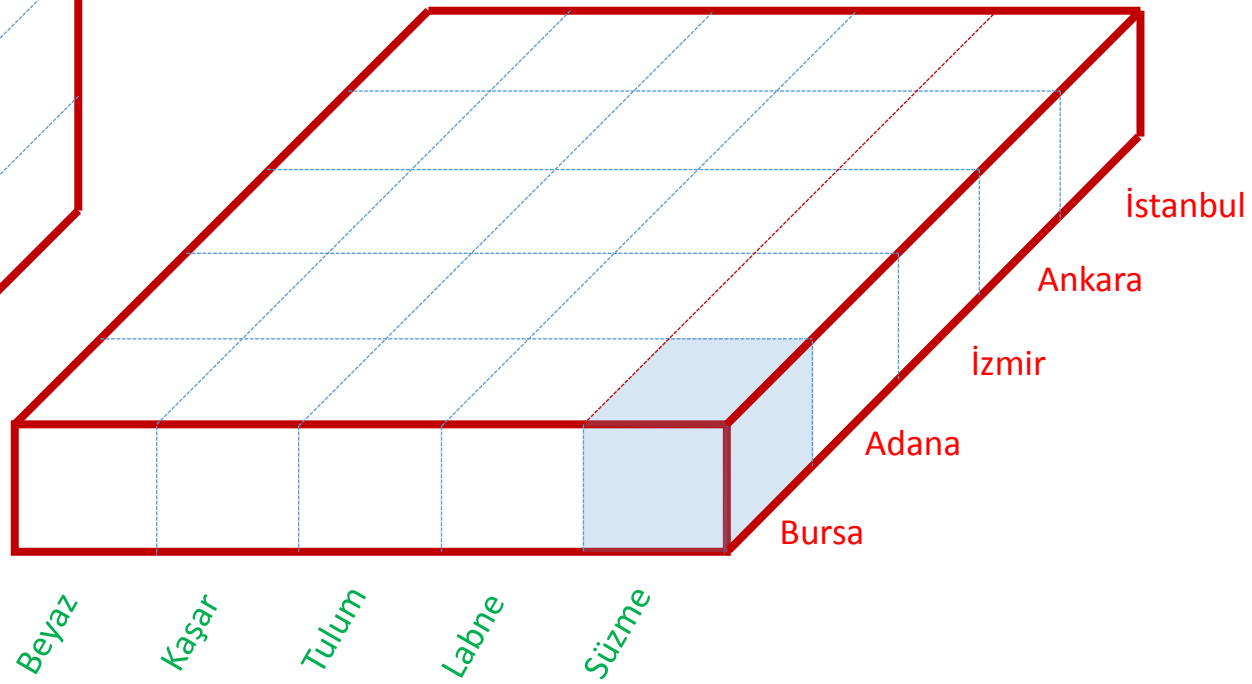
- OLAP, paylaşılan çok boyutlu bilginin hızlı analizi (**FASMI**) olarak da tanımlanır.
  - **F**ast (Hızlı)
  - **A**nalysis (Analiz)
  - **S**hared (Paylaşımlı)
  - **M**ultidimensional (Çok Boyutlu)
  - **I**nformation (Bilgi)

Bir OLAP küpü üzerinde aşağıdaki işlemler yapılabilir:

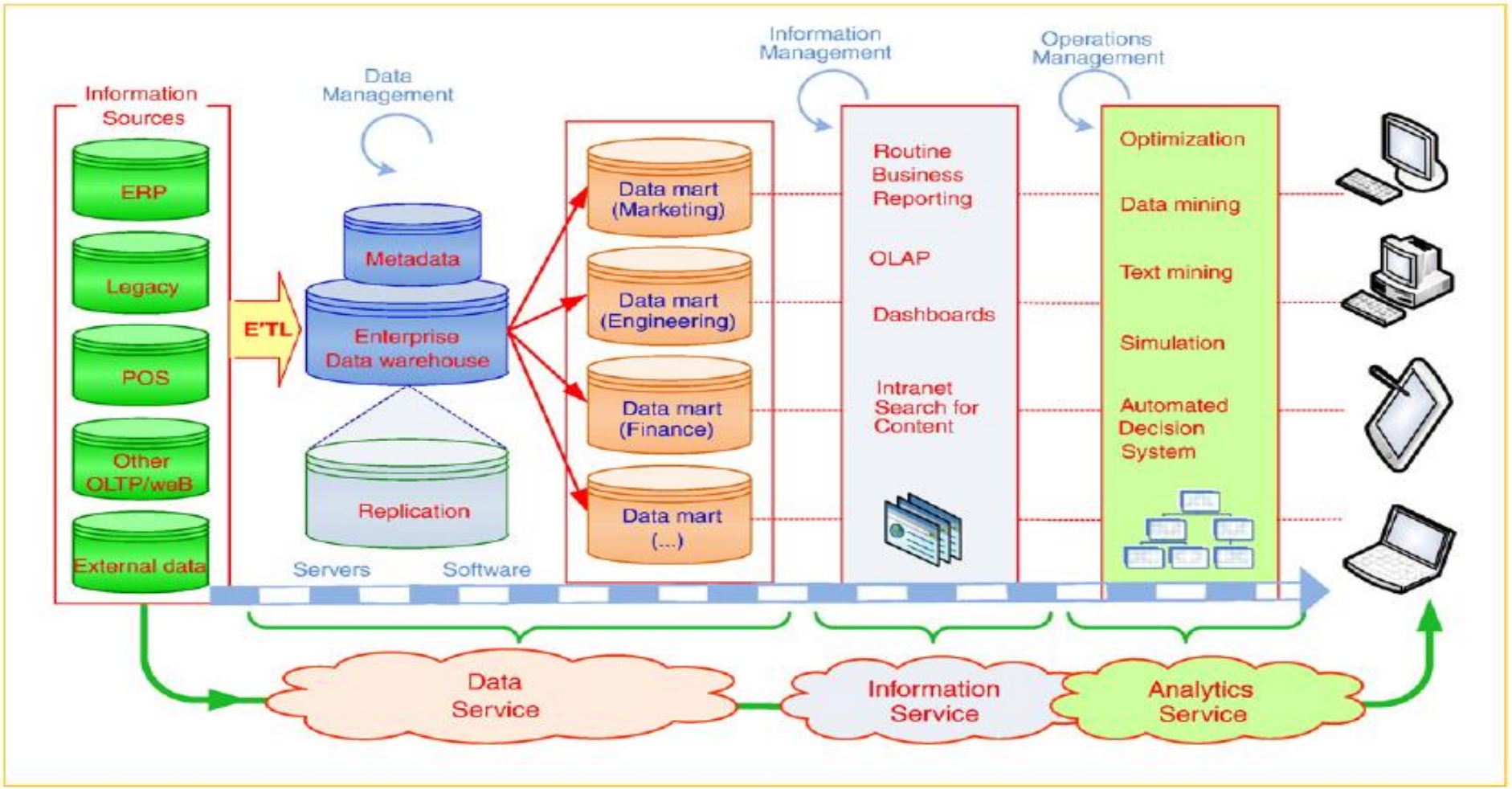
- **Dice (Çevir)**
  - Yer – Zaman → Ürün – Zaman
- **Slice (Dilimle)**
  - Son 1 yılın verileri
- **Drill Up / Down (Birleştir / Detaylandır)**
  - Yıl → Ay, Kıta → Ülke → Şehir

Çok Boyutlu Veri Modeli – Veri Küpü





Veri Ambarı Ortamı

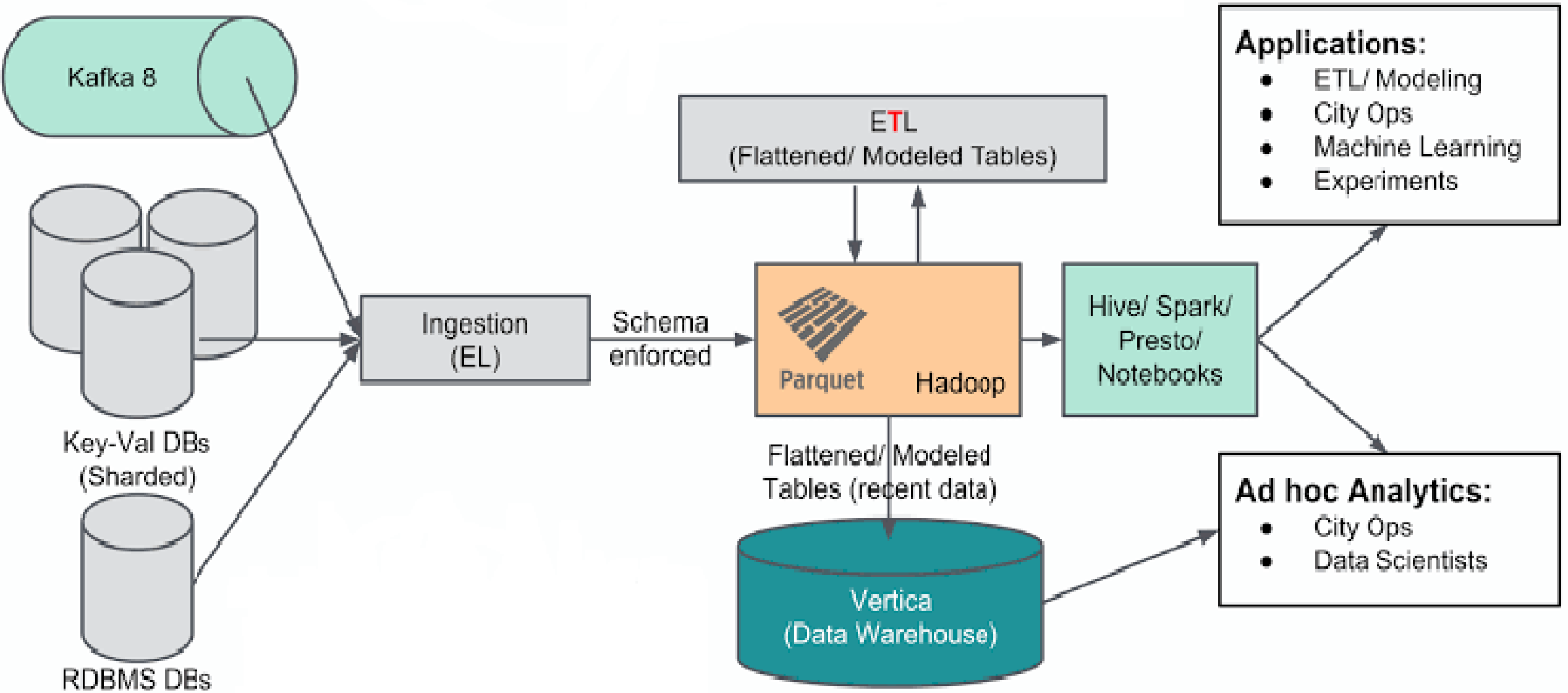


ETL İşlemleri (Extraction – Transformation – Loading)

- **Çıkarım (Extraction):** Bir veya daha fazla veri kaynağından verinin çıkarılması, alınması işlemidir.
  - Veri ambarının ilk oluşturulması sürecinde eski sistemlerdeki tüm veriler tümüyle veri ambarına aktarılır.
  - Aktarım işlemleri daha sonraki zamanlarda aktif sistemlerdeki verilerin güncellenmesine bağlı olarak yapılır.
- **Dönüşüm (Transformation):** Çekilen verinin dönüştürülmesidir.
  - Dönüştürmedeki amaç, verilerin kalitesinin arttırılmasıdır. Tekrarlar, eksiklikler, tutarsızlıklar giderilir, normalleştirme ve birleştirme yapılır.
- **Yükleme (Loading):** Verilerin fiziksel olarak veri ambarına yüklenmesi işlemidir.

Veri Tipleri

- Yapılandırılmış veri – Structured Data: Yapılandırılmış veri, kavramsal olarak modellenmiş bir saklama alanında, yani bir veri tabanında düzenlenen verilerdir. Bu şekilde daha etkili işleme ve analiz için adreslenebilir hale getirilmişlerdir.
  - OLTP Sistemler (Veri tabanı)
  - OLAP Sistemler (Veri Ambarı – Data Mart)
- Yarı yapılandırılmış veri – Semi-Structured Data: Yarı yapılandırılmış veri, ilişkisel veri tabanlarıyla veya diğer veri tablolarıyla ilişkili veri modellerinin resmi yapısına uymayan, ancak yine de anlamsal öğeleri ayırmak ve içindeki kayıt ve alan hiyerarşilerini zorlamak için etiketler veya başka işaretler içeren yapılandırılmış bir veri biçimidir.
  - *.csv
  - *.xlxs
  - XML
  - JSO
- Yapılandırılmamış veri – Unstructured Data: Yapılandırılmamış veri, önceden tanımlanmış bir veri modeline sahip olmayan veya önceden tanımlanmış bir şekilde organize edilmemiş verilerdir. Yapılandırılmamış veri, tipik olarak metin ağırlıklı olmakla birlikte ses, görüntü, v.b. biçemler de içerebilir.
  - Sosyal Medya
  - RSS Feeds


Kafka 8
Key-Val DBs
(Sharded)
RDBMS DBs
Ingestion
(EL)
Schema
enforced
ETL
(Flattened/ Modeled Tables)
Parquet
Hadoop
Flattened/ Modeled
Tables (recent data)
Vertica
(Data Warehouse)
Hive/ Spark/
Presto/
Notebooks
Applications:
ETL/ Modeling
City Ops
Machine Learning
Experiments
Ad hoc Analytics:
City Ops
Data Scientists

Big Data – Büyük Veri

- Büyük veri, mevcut bilgi sistemlerinin işleyemeyeceği kadar geniş ve karmaşık veri kümelerine verilen addır. Diğer bir deyişle, bilinen veri tabanı yönetim sistemleri ve yazılım araçlarının, verileri toplama, saklama, yönetme ve çözümleme yeteneklerini aşan büyüklükteki verilere **Big Data – Büyük Veri** denir.
- Büyük veri, yüksek hacimlerinin yanında, yüksek veri üretim hızı ve yüksek veri değişkenliğe sahip enformasyonlardan oluşur ve ileri düzeyde karara destek, verilerden anlam çıkarma ve süreç optimizasyonu yapabilmemizi sağlar. Ama bunun için de yepyeni bilgi işleme ve analiz yöntemleri gerektirir.

Büyük Veri Bileşenleri

Büyük veri oluşumunda 5 bileşen vardır. İngilizce karşılıklarından ötürü 5V olarak adlandırılan bileşenler şunlardır:

- **Volume:** IDC istatistiklerine göre, 2020' de ulaşılacak veri miktarı 2009'un 44 katı olacak. Kurumun veri arşivleme, işleme, bütünleştirme, saklama vb. teknolojilerinin bu büyüklükteki veri hacmiyle nasıl başa çıkacağının kurgulanması gerekmekte
- **Velocity:** Hızlı üreyen veri, o veriye muhtaç olan işlem sayısının ve çeşitliliğinin de aynı hızda artması sonucunu doğurmaktadır.
- **Variety:** Üretilen verinin %80' i yapısal veri değildir. Her veri kaynağının ürettiği veri farklı teknolojiler içerecektir. Dolayısıyla, «veri tipi» problemleri ile uğraşılması gerekmektedir.
- **Verification:** Verinin akışı sırasında, doğru katmandan, olması gerektiği güvenlik seviyesinde izlenmesi, doğru kişiler tarafından görünebilir veya gizli kalması gerekiyor.
- **Value:** En önemli bileşen ise değer yaratmasıdır. Büyük verinin, veri üretim ve işleme katmanlarınızdan sonra kurum içi bir artı değer yaratıyor olması gerekmektedir. Karar verme süreçlerine anlık olarak etki etmesi, doğru kararın verilmesinde hemen el altında olması beklenir.

Büyük Veri Bileşenleri

Farklı açılardan dikkate alındığında bileşenleri aşağıdaki gibi gruplamamız mümkün.

- Veri alanı : Örüntü aramak / yorumlamak
- İş zekası alanı: Tahmin/öngörüde bulunmak
- İstatistik alanı: Varsayımlarda bulunmak

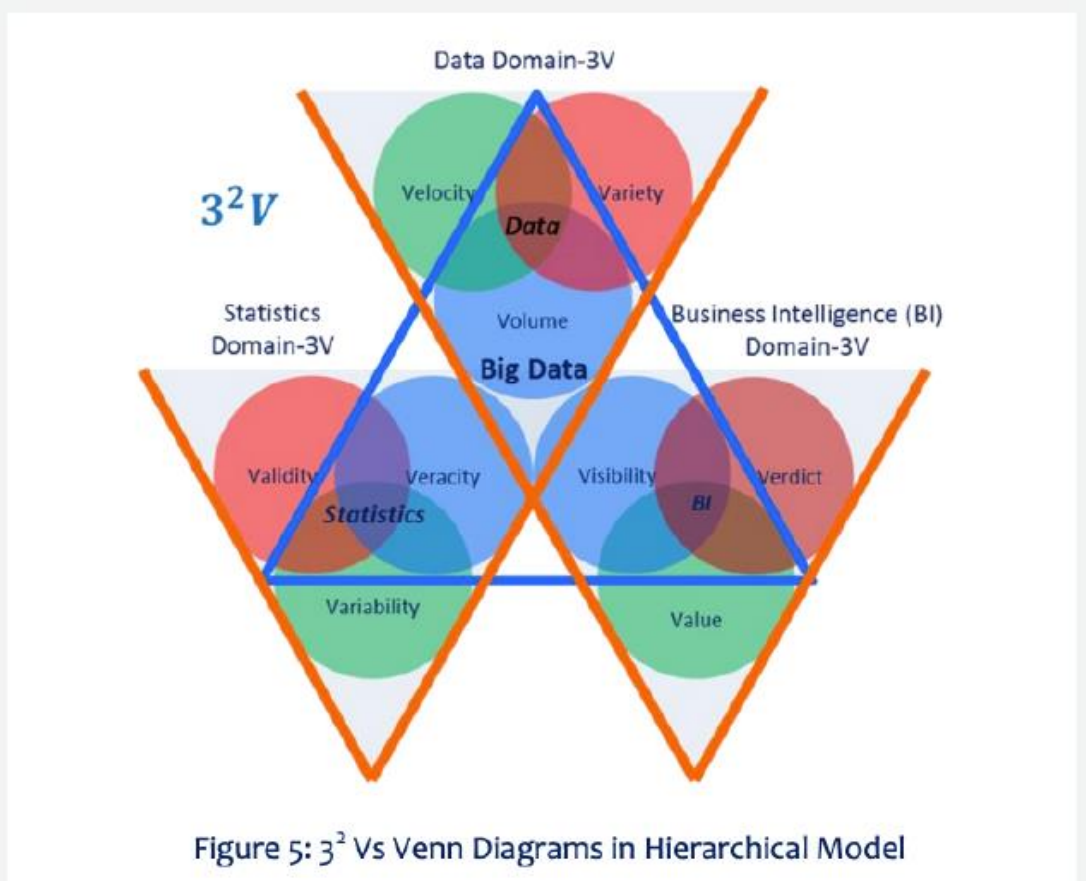
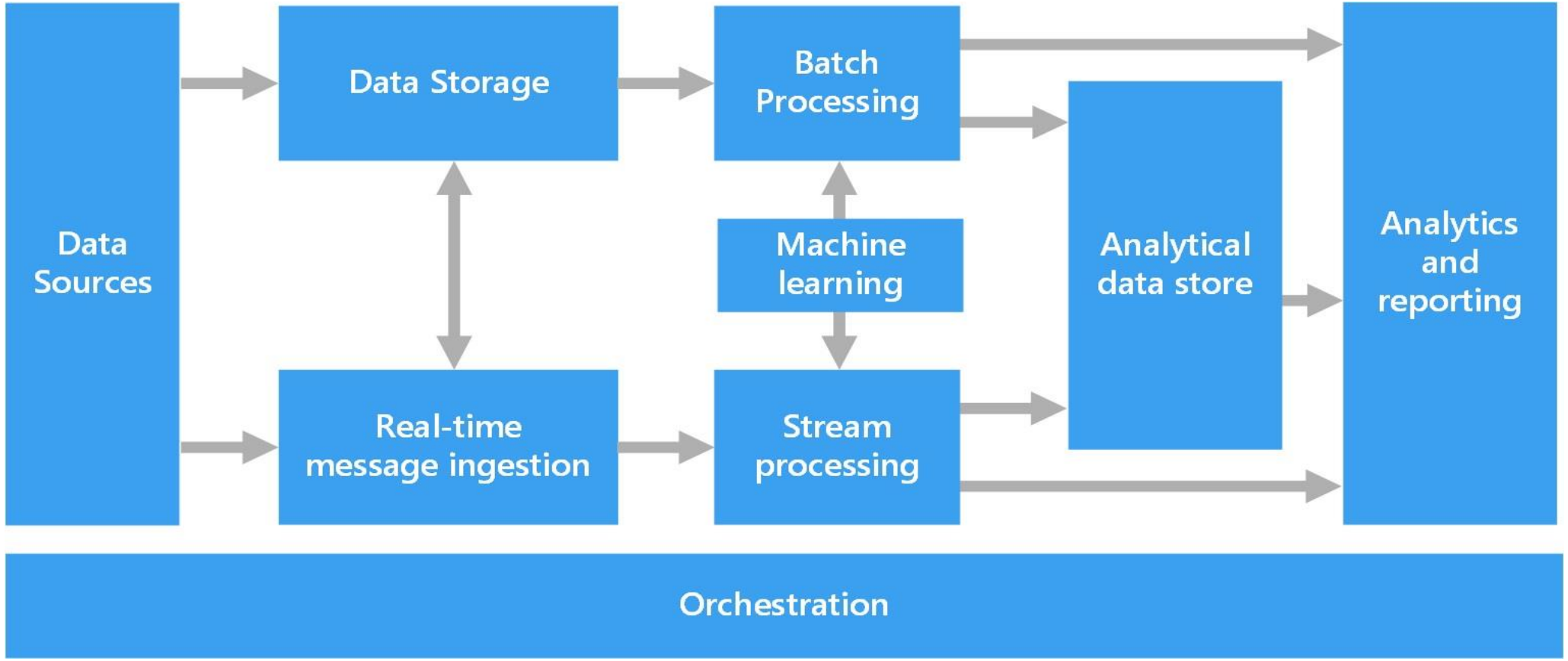


Figure 5: 3^2 Vs Venn Diagrams in Hierarchical Model

Büyük Veri Mimarisi (1/4)

Aşağıdaki diyagramda bir büyük veri mimarisine uygun olan mantıksal bileşenler gösterilmiştir. Her çözüm, bu diyagramdaki her bir öğeyi içermeyebilir.



Büyük Veri Mimarisi (2/4)

Büyük veri mimarilerinin çoğu, aşağıdaki bileşenlerin bazılarını veya tümünü içerir:

- **Veri kaynakları**. Tüm büyük veri çözümleri bir veya daha fazla veri kaynağıyla başlar. Örneklere şunlar dahildir:
  - İlişkisel veri tabanları gibi uygulama verisi depoları.
  - Uygulamalar tarafından üretilen web sunucusu günlük dosyaları gibi statik dosyalar.
  - IoT cihazları gibi gerçek zamanlı veri kaynakları.
- **Veri depolama**. Toplu işleme yönelik veriler genellikle çeşitli biçimlerdeki büyük dosyaları yüksek miktarlarda tutabilen bir dağıtılmış dosya deposunda depolanır. Bu tür depolara genelde ***veri gölü*** adı verilir.
- **Toplu işleme**. Veri kümeleri çok büyük olduğundan büyük veri çözümleri genellikle uzun süre çalışan toplu işler aracılığıyla verileri filtreleyerek, toplayarak ve diğer yollardan analize hazırlayarak veri dosyalarını işlemelidir. Bu işler, çoğu zaman kaynak dosyaların okunması, işlenmesi ve çıktının yeni dosyalara yazılmasını içerir. Seçenekler arasında Azure Data Lake Analytics'te U-SQL işleri çalıştırma, bir HDInsight Hadoop kümesinde Hive, Pig veya özel Map/Reduce işleri kullanma veya bir HDInsight Spark kümesinde Java, Scala veya Python programları kullanma bulunur.

Büyük Veri Mimarisi (3/4)

- **Gerçek zamanlı ileti alımı.** Çözüm gerçek zamanlı kaynaklar içeriyorsa mimarinin akış işlemek için gerçek zamanlı iletileri yakalayıp depolayacak bir yol içermesi gerekir. Bu yol, gelen iletilerin işlenmek üzere bir klasöre bırakıldığı basit bir veri deposu olabilir. Ancak, birçok çözüm ileti alım deposunun iletiler için bir arabellek görevi görmesini, ayrıca ölçeği genişletme işlemini, güvenilir teslimi ve diğer iletiyi kuyruğa alma semantiğini desteklemesini gerektirir. Bir akış mimarisinin bu kısmı genellikle akışı arabelleğe alma olarak adlandırılır.
- **Akış işleme**. Çözümün, gerçek zamanlı iletileri yakaladıktan sonra filtreleme, toplama ve diğer verileri analize hazırlama işlemleriyle bu iletileri işlemesi gerekir. İşlenen akış verileri daha sonra bir çıkış havuzuna yazılır.
- **Analitik veri deposu**. Büyük veri çözümlerinin çoğu, verileri analiz için hazırlar ve sonra işlenen verileri analiz araçları kullanılarak sorgulanabilecek yapılandırılmış bir biçimde sunar. Bu sorgulara hizmet veren analitik veri deposu, birçok geleneksel iş zekası (BI) çözümünde görüldüğü gibi Kimball tarzında ilişkisel bir veri ambarı olabilir. Alternatif olarak, veriler HBase gibi düşük gecikme süreli bir **NoSQL** teknolojisi veya dağıtılmış veri deposundaki veri dosyalarının meta verilerle özetini sağlayan etkileşimli bir Hive veritabanı aracılığıyla sunulabilir.

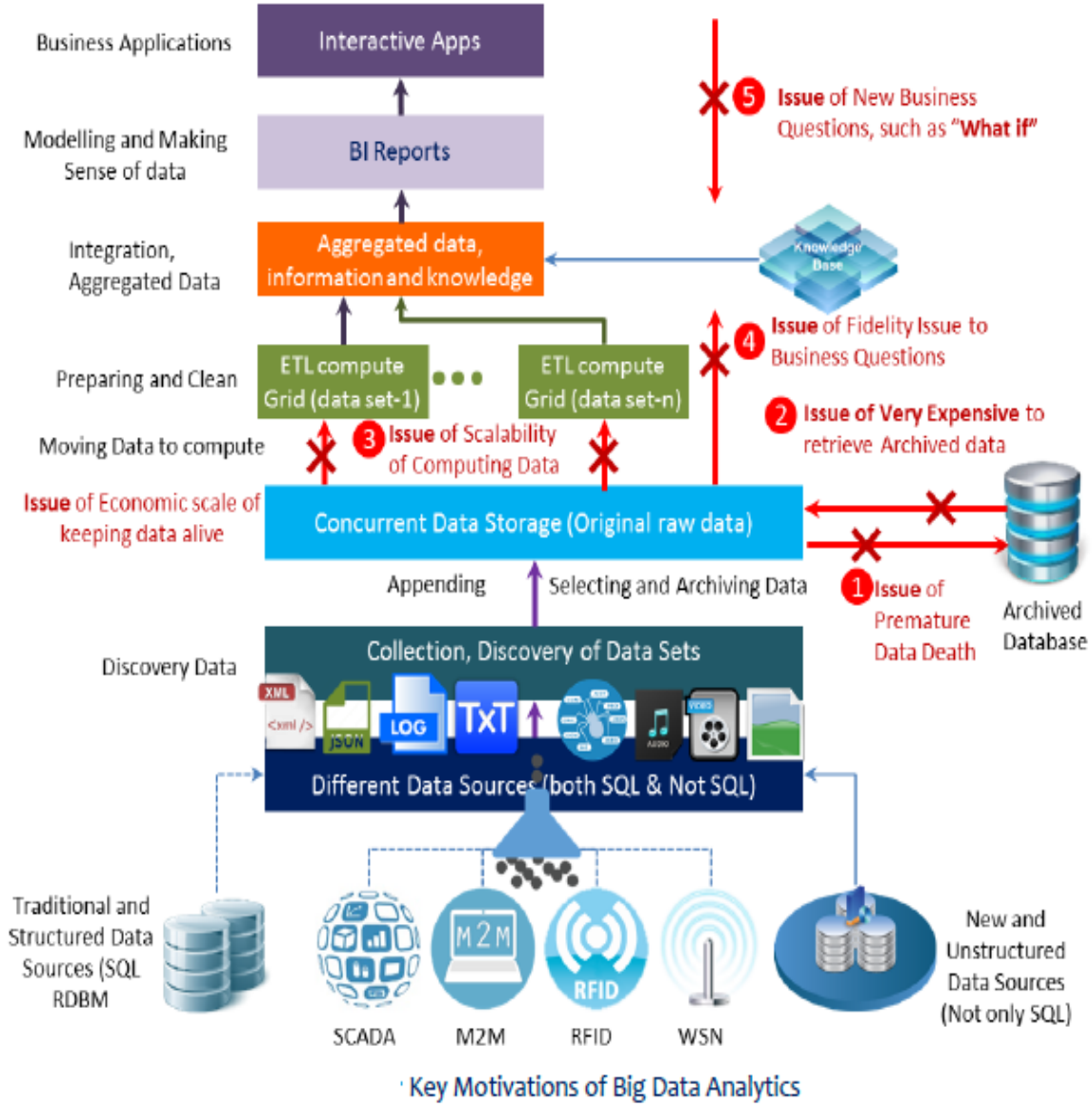
Büyük Veri Mimarisi (4/4)

- **Analiz ve raporlama**. Büyük veri çözümlerinin çoğunun amacı analiz ve raporlama aracılığıyla veriler hakkında öngörüler sağlamaktır. Mimari, kullanıcılara veri analizi desteği sağlamak amacıyla Azure Analysis Services'te çok boyutlu OLAP küpü veya tablo verisi modeli gibi bir veri modelleme katmanı içerebilir. Ayrıca, Microsoft Power BI veya Microsoft Excel'deki modelleme ve görselleştirme teknolojilerini kullanarak self servis BI işlemlerini de destekleyebilir. Analiz ve raporlama, veri uzmanları veya veri analistlerinin etkileşimli veri incelemelerini de içerir. Azure hizmetlerinin çoğu bu senaryolar için Jupyter gibi analitik not defterlerini destekleyerek söz konusu kullanıcıların mevcut Python veya R becerilerinden yararlanmasına olanak sağlar. Büyük ölçekli veri incelemeleri için Microsoft R Server'ı tek başına veya Spark ile kullanabilirsiniz.
- **Düzenleme**. Büyük veri çözümlerinin çoğu, iş akışlarıyla temsil edilen şekilde yinelenen veri işleme faaliyetlerinden oluşur. Bu iş akışları kaynak verileri dönüştürür, verileri çeşitli kaynaklar ve havuzlar arasında taşır, işlenen verileri bir analitik veri deposuna yükler ya da sonuçları doğrudan bir rapora veya panoya gönderir. Bu iş akışlarını otomatik hale getirmek için Azure Data Factory veya Apache Oozie ve Sqoop gibi bir düzenleme teknolojisi kullanabilirsiniz.

Big Data Challenges

1. Erken veri ölüm sorunu
2. Arşivlenmiş verileri almak maliyetinin çok yüksek olması sorunu
3. Hesaplanan verinin ölçeklenebilirliği sorunu
4. Ortaya konulan iş problemlerine sadakat sorunu
5. Yeni iş soruları sorunu (Varsayalım ki…)

Verilerin canlı tutulması için gerekli ekonomik ölçek sorunu



Veri Gölü – Data Lake

- 2011 yılında, Pentaho'nun CTO'su James Dixon, Veri Ambarı – Data Warehouse yerine Veri Gölü – Data Lake kavramını ortaya attı.
- Dixon, bir Veri Ambarı ile Veri Gölü arasındaki farkın Veri Ambarının giriş noktasında verileri ön kategorilere ayırarak zaman ve enerjiyi boşa harcaması, Veri Gölünün ise ilişkisel olmayan bir veri tabanı (**NoSQL)** kullanarak verileri olduğu gibi, sınıflandırmadan, kategorize etmeden kabul edip, basitçe depolaması olarak tanımlar.
- Veri gölü genel yapısı gereği veri kaynaklarından gelen hiçbir veriyi geri çevirmez, hepsini kabul eder.
- Veri, yaprak seviyesinde hiç yapılandırılmamış ya da yarı yapılandırılmış şekilde tutulur.

Veri Gölü vs Veri Ambarı

Bir veri gölünün beş temel özelliğini ve bunların veri ambarı yaklaşımıyla nasıl kontrast oluşturduklarını inceleyelim:

1. Veri Gölü bütün veriyi tutar.
2. Veri Gölü bütün veri türlerini destekler.
3. Veri Gölü bütün kullanıcı tiplerini destekler.
4. Veri Gölü değişimlere kolayca uyum sağlar.
5. Veri Gölü çok daha çabuk “içgörü” sağlar.

Veri Gölü tüm veriyi tutar

Bir veri ambarının geliştirilmesi sırasında, veri kaynaklarını analiz etmek, iş süreçlerini anlamak ve verinin profilini çıkarmak için önemli miktarda zaman harcanır. Sonuç, raporlama için tasarlanmış oldukça yapılandırılmış bir veri modelidir. Bu sürecin büyük bir kısmı, ambara hangi verilerin dahil edilip dahil edilmeyeceği hakkında karar vermeyi içerir. Genel olarak, veriler belirli soruları yanıtlamak için kullanılmazsa veya tanımlanmış bir raporda kullanılmazsa, ambardan çıkarılabilir. Bu genellikle veri modelini basitleştirmek ve aynı zamanda veri ambarının performansını arttırmak için kullanılan pahalı disk depolama alanını korumak için yapılır.
Buna karşılık, veri gölü TÜM verileri korur. Sadece günümüzde kullanılmakta olan veriler değil, bir gün kullanılabileceği için ve hatta asla kullanılamayacak veriler de her zaman tutulur, böylece analiz yapmak için herhangi bir noktaya zamanda geri gidebiliriz. Bu yaklaşım mümkün olur çünkü bir veri gölünün donanımı genellikle bir veri ambarı için kullanılandan büyük ölçüde farklıdır. Emtia, kullanıma hazır sunucular, ucuz depolama alanı ile birleştiğinde veri gölünü terabaytlara ve petabitlere ölçeklendirmeyi oldukça ekonomik kılmaktadır.

Veri Gölü tüm veri türlerini destekler

Veri ambarları genellikle işlem sistemlerinden elde edilen verilerden ve nicel metriklerden ve onları tanımlayan niteliklerden oluşur. Web sunucusu günlükleri, sensör verileri, sosyal ağ etkinliği, metin ve görüntüler gibi geleneksel olmayan veri kaynakları büyük ölçüde göz ardı edilir. Bu veri türleri için yeni kullanımlar bulunmaya devam etmektedir, ancak bunları tüketmek ve saklamak pahalı ve zor olabilir. Veri gölü yaklaşımı bu geleneksel olmayan veri türlerini de kapsar. Veri gölünde, kaynak ve yapıdan bağımsız olarak tüm verileri tutulur. Veriler “ham veri – raw data” diye adlandırılan formda tutulur ve yalnızca kullanım anında dönüştürülür. Bu yaklaşıma ”Schema on Read – Okuma anında şemalama” adı verilir ve veri ambarı modellemelerinde kullanılan “Schema on Write – Yazma anında şemalama” kavramının tersidir.

Veri Gölü bütün kullanıcı tiplerini destekler

Kurumlar bünyesinde çalışanları 3 gruba ayırmak mümkündür:

1. Operasyonel çalışanlar: Kurum çalışanlarının %80 hatta daha fazlasını oluştururlar. Temel istekleri günlük standart KPI ölçümlerini yapmak, bir hesap tablosu üzerinde işlemlerini yapmak kısaca düzgün yapılandırılmış bir veri üzerinden rapor almaktır. Aslında bu şekilde bakıldığı zaman iyi modellenmiş bir ambar her türlü ihtiyaçlarına kolaylıkla cevap verir.
2. Analiz yapanlar: Kurum çalışanlarının %10 belki biraz daha fazlasını oluştururlar. Üzerinde analiz yapacakları veriyi toplarken ilk kaynak olarak veri ambarına başvururlar ama daha da ötesine giderek kimi zaman kaynak sistemlerden, kimi zaman da kurum dışından veri toplar, ve bu topluluk üzerinde analizlerini yaparlar. Dolayısıyla ambar içinde toplanmış olan veri onlar için yeterli olmayabilir.
3. Derin Analiz yapmak isteyenler: Sayıları çok azdır. Kurum çalışanların ancak %1-2sini oluştururlar. Kendi araştırmaları doğrultusunda kendilerine özgü veri kümesini oluştururlar. Birçok farklı türde veriyi karıştırırlar ve cevaplanacak tamamen yeni sorular bulurlar. Bu kullanıcılar veri ambarını kullanabilir, ancak genellikle ambarın yeteneklerinin çok ötesine geçtikleri için çoğu zaman ambarı görmezden gelirler. Data Scientist diye adlandırdığımız çalışanlar bu grubun üyesidir.

Ambar modellemesi üç gruba birden hitap etmezken, Veri Gölü kavramı yukarıda sayılan tüm kullanıcı türlerine rahatlıkla hizmet verebilecek bir yapıya sahiptir

Veri Gölü değişimlere kolayca uyum sağlar

Veri ambarlarının en hantal olduğu konu “değişim” dir. Ambarın doğru modellenebilmesi için geliştirilmeye başlandığı ilk andan itibaren önemli miktarda zaman harcanmaktadır. Bunun akabinde, ambar, hizmet vereceği kurumun ihtiyaçları doğrultusunda yapılacak analizlere en hızlı şekilde cevap verecek şekilde sürekli uyarlanır.
Öte yandan, veri gölünde, tüm veriler ham formunda depolandığından ve onu kullanmak isteyen biri tarafından her zaman erişilebilir olduğundan, kullanıcılar klasik veri ambarı kullanımının çok ötesinde yöntem ve yollarla veri analizi yapabilirler.

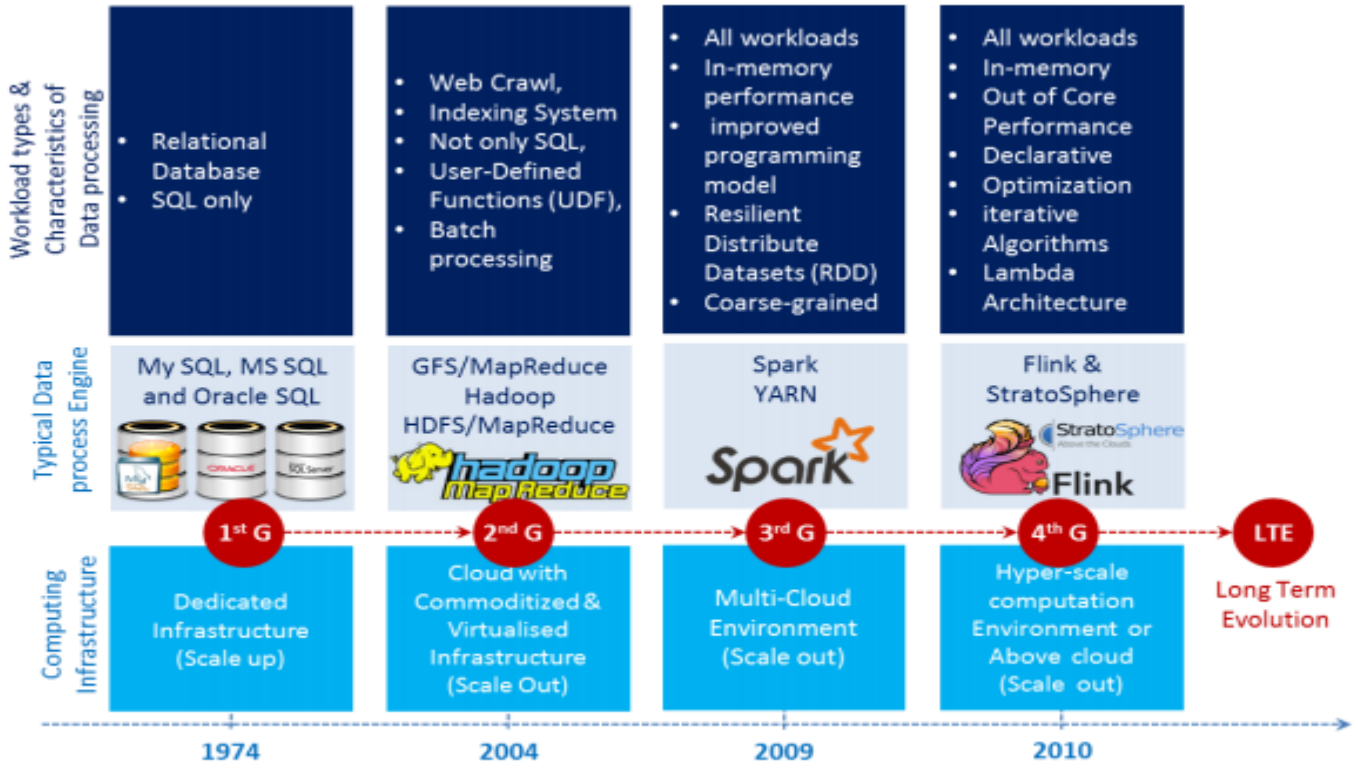
Veri Gölü çok daha çabuk “içgörü” sağlar.

Bu son fark, diğer dört farkın sonucudur. Veri gölleri tüm veri ve veri türlerini içerdiğinden ve kullanıcıların verilere temizlenmeden, dönüştürülmeden, yapılandırılmadan ulaşmasını sağladığından geleneksel veri ambarı yaklaşımından çok daha hızlı bir şekilde analiz sonuçlarına ulaşmalarını sağlar.

Ancak, verilere bu erken erişimin bir bedeli vardır. Veri ambarı geliştirme ekibi tarafından yapılan tüm işlemler, veri gölü kullanıcıları tarafından yapılması gerekmektedir. Bu da kullanıcılara çok daha fazla iş yükü yüklemesine neden olmaktadır.

Öte yandan özellikle operasyonel raporlar ile çalışan kullacılar, veri gölündeki veriyi, tıpkı daha önce ambar uygulamalarında kullandıkları veri gibi, daha çok yapılandırılmış hali üzerinde çalışacaklardır. Buradaki temel fark, veri gölünde kullanıcıların üzerinde çalıştıkları bu veriler, ambar uygulamalarındaki gibi değişime kapalı, katı fiziksel tablolardan ziyade, ham veri üzerine inşa edilmiş mantıksal meta data görünümleri olacaktır.

Veri İşleme ve Büyük Veri işleme araçlarının Gelişimi



Evolution of Data and Big Data process engines

NoSQL Veri Tabanı

- NoSQL → Not Only SQL demektir
- Internetin gün geçtikçe artan verisini depolayabilmek ve yüksek trafiğe sahip sistemlerin ihtiyaçlarına cevap verebilmek amacıyla ortaya çıkmış yatay olarak ölçeklendirilebilen sistemlere verilen genel addır.
- İlişkisel veri tabanı sistemlerinde veri modellenmesi, daha sonra sistemin her açıdan maliyetinin yükselmemesi adına, sistem kurulumundan önce çok iyi planlanarak yapılması gerekmektedir. Bu durum ilişkisel veri tabanlarını son derece rijit ve statik bir hale getirmektedir.
- Buna karşın NoSQL veri tabanı sistemlerinde, verinin kayıt altına alınması işleminde ilk adımda yapılan veri modellemesine sonradan çok rahat ekleme yapılabilmekte ve bu durum sisteme ek bir maliyet getirmemekte ya da az bir maliyet getirmektedir.

NoSQL Veri tabanı – Dinamik yapı

- NoSQL veri tabanı dinamik bir yapıya sahiptir. İlişkisel veri tabanında görmeye alıştığımız tablo, sütun ya da tablolar arası katı bağlantılar bu sistem üzerinde yoktur.
- NoSQL sistemlerde yeni bir alana ihtiyaç duyulduğu durumlarda kaydı direk eklemek yeterli olmaktadır. NoSQL kendi kendine alanı oluşturur ve değeri kaydeder.
- Kayıtların maliyetsizce gerçekleşmesinin nedeni ise verilerin tablo ve sütunlarda saklanması yerine JSON ve XML formatına benzer yapıda saklanmasıdır.

NoSQL Veri Tabanı Sisteminin Avantajları

- İlişkisel veri tabanlarına göre yüksek erişilebilirlik imkanı sunarlar.
- Okuma ve yazma performansları olarak göreceli olarak ilişkisel veri tabanı sistemlerine göre daha performanslıdırlar
- Yatay olarak genişletilebilirler. Binlerce sunucu bir arada küme olarak çalışabilir ve çok büyük veri üzerinde işlem yapabilirler.
- Esnek yapılarından dolayı programlama ve özellikle bakım anlamında kolaylık sağlarlar.
- Farklı özelliklere sahip birçok kurulum arasından seçim yapma şansı vardır.
- Birçok açık kaynak kodlu projelere ve bulut bilişim teknolojilerine uygun olduğu için maliyet olarak ilişkisel veri tabanı yönetim sistemlerine göre daha avantajlıdır.

NoSQL Veri Tabanı Sisteminin Dezavantajları

- İlişkisel veri tabanı yönetim sistemlerini kullanan uygulamaların NoSQL sistemlerle ortak çalışması başlangıçta zor olacaktır. Veri başarılı bir şekilde taşınsa bile, özellikle **join** kullanan kodlarda düzenlemelerin yapılması gerekecektir.
- İlişkisel veritabanı yönetim sistemlerindeki işlem hareketleri (transaction) kavramı, NoSQL veritabanı sistemlerinde bulunmadığı için veri kaybı söz konusu olabilmektedir.
- NoSQL veritabanı sistemleri veri güvenliği konusunda ilişkisel veri tabanı yönetim sistemleri kadar gelişmiş özelliklere henüz sahip değillerdir.

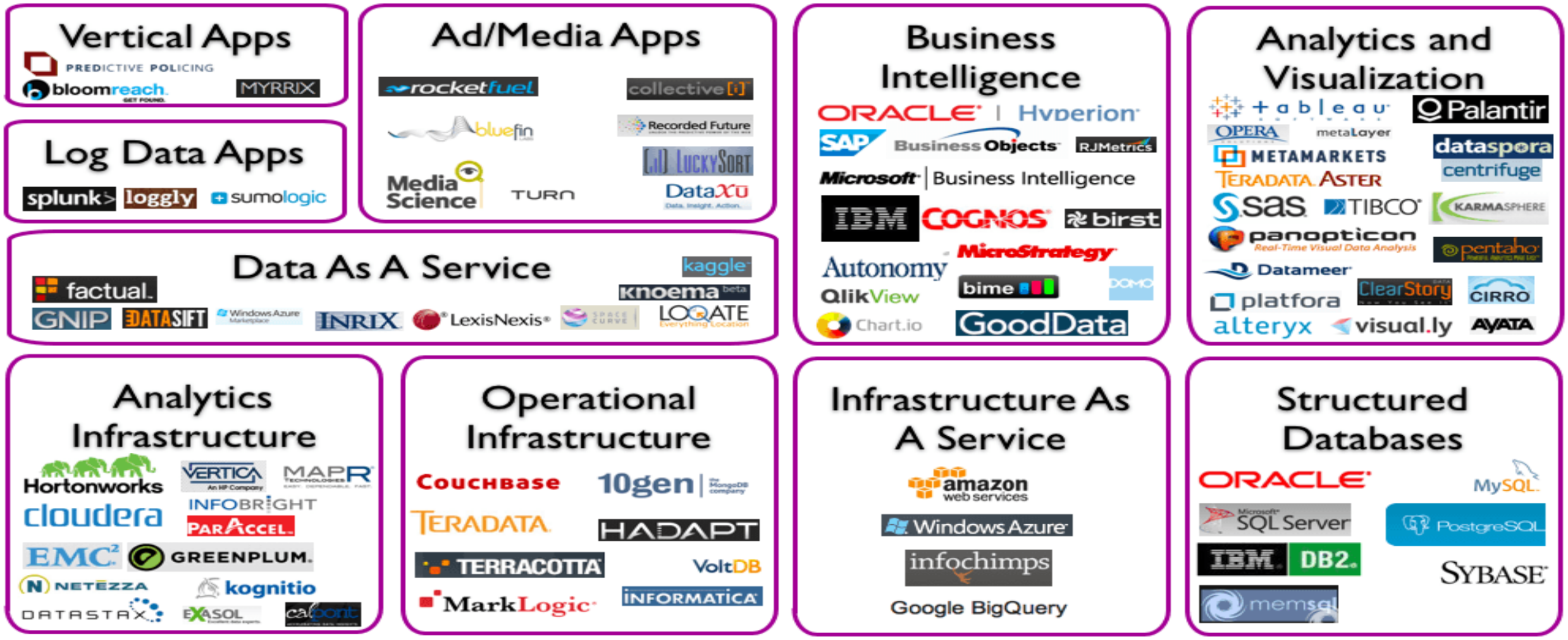
NoSQL Veri Tabanı Sisteminin Dezavantajları

- İlişkisel veri tabanı yönetim sistemlerindeki sorgu tabanlı veri erişimi yerine NoSQL sistemlerdeki anahtar tabanlı veri erişimi sağlamak gerekmektedir.
- NoSQL sistemler temel olarak verilere tekil anahtarlar üzerinden erişir. Her NoSQL sisteminde ikincil indeks (secondary index) yeteneği de bulunmayabilir. Bu yüzden verilere erişim SQL sorguları ile yapıldığı gibi kolay yapılamaz. Uygulamanın özelliğine göre birincil anahtarlar belirli bir mantığa göre verilir ve bu sayede veriye erişmeden önce zaten bu anahtar biliniyor ya da oluşturulabiliyor olur.
- Örneğin zamana göre artan ön ek (prefix), kaydın anahtarı ile birleştirilerek tek seferde ilgili kaydın belirli bir zaman aralığındaki kayıtlarına erişmek mümkündür. Bu sayede herhangi bir sorguya gerek olmaksızın tamamen anahtarlar üzerinden verilere hızlıca erişim sağlanmış olur

NoSQL Veri Tabanı Sistemleri

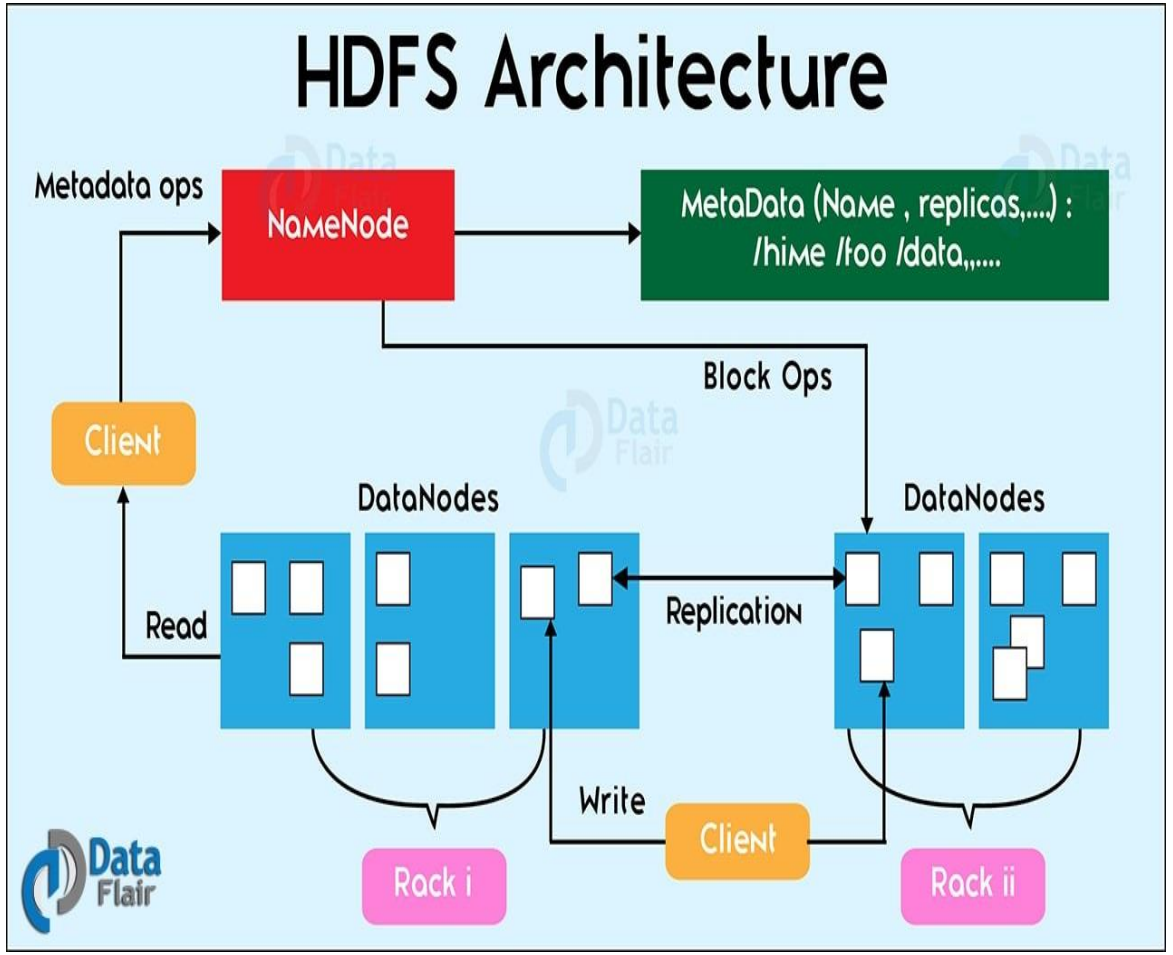
- Piyasada kullanımda olan NoSQL sistemlerinin SQL-92 gibi bir standardı olmadığı için sahip oldukları genel özelliklerin dışında kendi aralarında farklılıklar göstermektedir.
- Her sistemin veri tutarlılığı ve erişimi ile ilgili farklı özellikleri ve yetenekleri bulunmaktadır. Fakat NoSQL sistemlerini genel olarak 3 grupta toplayabiliriz:
  - **Döküman (Document) tabanlı**: Bu sistemlerde bir kayıt döküman olarak isimlendirilir. Dökümanlar genelde JSON formatında tutulur. Bu dökümanların içerisinde sınırsız alan oluşturulabilir. MongoDB, CouchDB, HBase, Cassandra ve Amazon SimpleDB bunlara örnektir.
  - **Anahtar / Değer (Key / Value) tabanlı**: Bu sistemlerde anahtara karşılık gelen tek bir bilgi bulunur. Yani kolon kavramı yoktur. Azure Table Storage, MemcacheDB ve Berkeley DB bunlara örnektir.
  - **Grafik (Graph) tabanlı**: Diğerlerinden farklı olarak verilerin arasındaki ilişkiyi de tutan,Graph Theory modelindeki sistemlerdir. Neo4J, FlockDB bunlara örnektir.

Big Data Landscape



Hadoop – HDFS

- 2000li yılların başında geliştirilmiş dağıtık dosya sistemidir.
- RDBMS(Relational Database Management System) yani ilişkisel veri tabanı yönetim sistemlerinden farklı olarak verileri tek bir bilgisayarda tutmayıp gelen verileri -her birinin kendine ait işlemcisi ve RAMi- olan Node'larda (küme) HDFS dosya sistemi ile **denormalize** bir şekilde veriyi saklayan ve işlenmesine olanak sağlayan açık kaynak kodlu kütüphanedir.
- Esnekliği, özellikle ürettiğimiz ve topladığımız **yapılandırılmamış verileri** (ses, video, işlenmemiş metin vb.) yönetmek için kullanışlıdır.



Hadoop Distributed File System

- Büyük veri altyapısını anlayabilmek için dağıtık mimarinin anlaşılması elzemdir. Bu kapsamda karşımıza en sık çıkan mimari **Hadoop** mimarisidir.
- Hadoop, sıradan sunuculardan oluşan bir küme üzerinde büyük verileri işlemek amacıyla çalıştırılan ve **HDFS (Hadoop Distributed File System)** olarak adlandırılan bir dağıtık dosya sistemi ile **MapReduce** özelliklerini bir araya getiren açık kaynak kodlu bir kütüphanedir. Dolayısıyla bilgi erişim merkezlerine ait n-gram özetleri HDFS sistemindeki sıradan sunucular üzerine dağıtılacaktır.
- HDFS sayesinde sıradan sunucuların diskleri bir araya gelerek büyük, tek bir sanal disk oluştururlar. Bu sayede çok büyük boyutta birçok dosya bu sistemde saklanabilir.
- Dosyalar bloklar halinde birden fazla ve farklı sunucu üzerine dağıtılarak RAID benzeri bir yapıyla yedeklenir. Bu sayede veri kaybı önlenmiş olur. Ayrıca HDFS çok büyük boyutlu dosyalar üzerinde okuma işlemi imkânı sağlamaktadır. Ancak rasgele erişim özelliği bulunmaz. HDFS, NameNode ve DataNode işlem süreçlerinden oluşmaktadır
- Apache Hadoop Dağıtılmış Dosya Sistemi Java ile yazılmıştır ve farklı işletim sistemleri üzerinde koşabilir.

Temel Hadoop Bileşenleri

- **NameNode :** HDFS'in kilit noktalarıdır. HDFS üzerindeki tüm dosyaların yer bilgilerini içeren bir ağaç (directory) yapısıdır. Ana (master) süreç olarak blokların sunucular üzerindeki dağılımınından, yaratılmasından, silinmesinden, bir blokta sorun meydana geldiğinde yeniden oluşturulmasından ve her türlü dosya erişiminden sorumludur. Kısacası HDFS üzerindeki tüm dosyalar hakkındaki bilgiler (metadata) NameNode tarafından saklanır ve yönetilir. Her kümede yalnızca bir adet NameNode olabilir.
  - Secondory NameNode
- **DataNode:** işlevi blokları saklamak olan işçi (slave) süreçtir. Her DataNode kendi yerel diskindeki veriden sorumludur. Ayrıca diğer DataNode'lardaki verilerin yedeklerini de barındırır. DataNode'lar küme içerisinde birden fazla olabilir.
- **Job Tracker :** MapReduce işlemlerini takip eden yazılımlardır.
- **Test Tracker:** TaskTracker Hadoop kümesindeki data nodelarda çalışır. TaskTracker, JobTracker'dan işlem istekleri alır. Başlıca sorumluluğu, üzerinde koştuğu DataNodeda yerel olarak meydana gelen MapReduce iş yüklerinin yürütülmesini izlemek ve JobTracker'a durum güncellemeleri göndermektir.

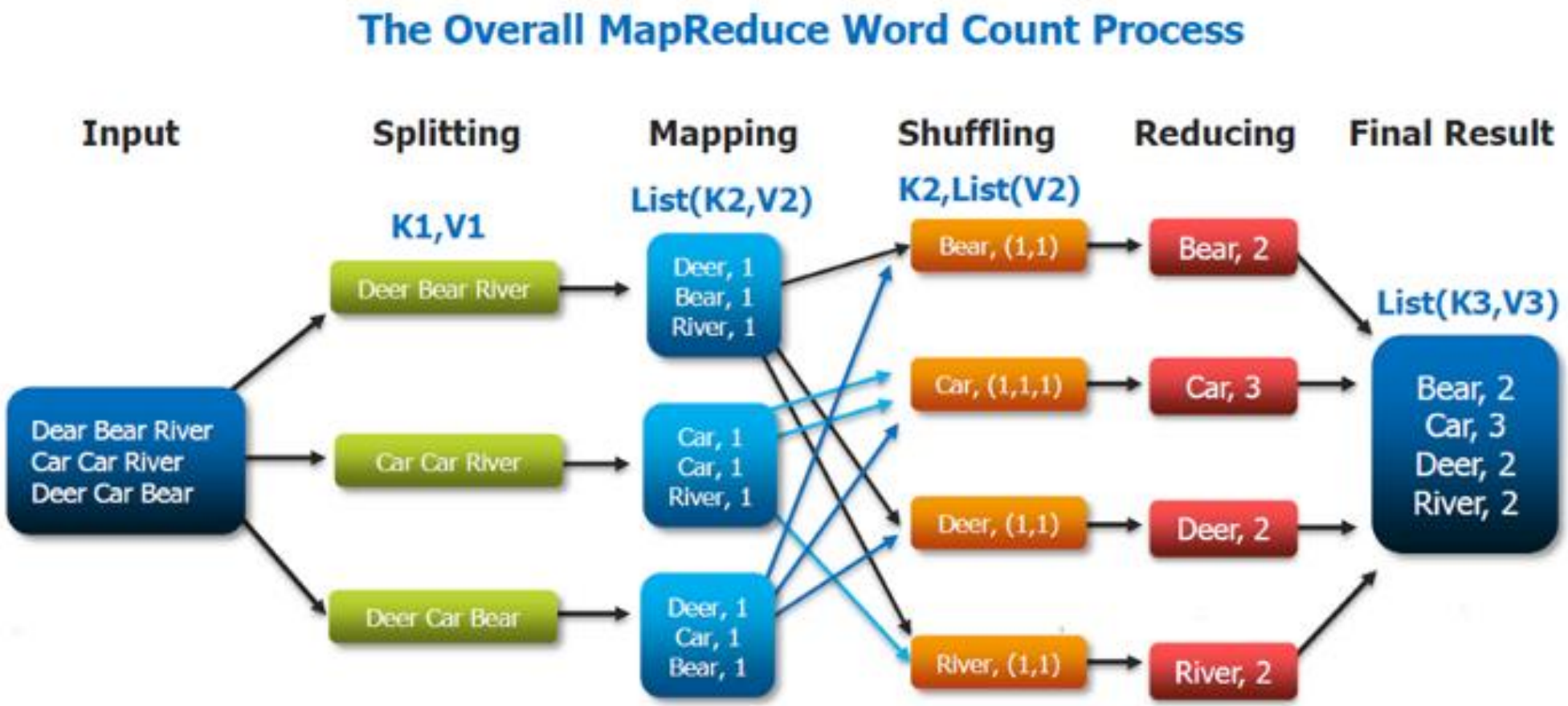
Hadoop verileri nasıl işler

- Node’lerde yer alan verileri merkeze toplayıp işlemek yerine SQL diline yakın sorguları Node’lere dağıtarak Node’lerde gerekli işlemi gerçekleştirir. Çünkü her Node’nin kendisine ait işlemcisi ve RAMi vardır. Bunu yapmasının sebebi ise veriyi merkeze çekip trafik oluşturmamaktır

Map - Reduce

- MapReduce, büyük veriyi büyük kümelerde (clusters) işlemek üzere geliştirilmiş bir dağıtık programlama modelidir.
- Genel yapısı itibariyle MapReduce adından da anlaşılacağı üzere iki ana fonksiyondan oluşur.
  - **Map:** Bir yığının tüm üyelerini sahip olduğu fonksiyon ile işleyip bir sonuç listesi döndürür.
  - **Reduce**: Paralel bir şekilde çalışan iki veya daha fazla map fonksiyonundan dönen sonuçları harmanlar ve çözer.
  - Map() ve Reduce() işlemleri paralel bir şekilde çalışırlar, aynı anda aynı sistemde olmaları gerekmez.

Hadoop dağıtık veri işleme modeli: Map-Reduce



Map Reduce – 6 Adım

- Adım 1- Input : Veri girişlerinin yapıldığı adımdır.
- Adım 2 - Splitting : Gelen veriler bu aşamada işlemesi daha kolay olabilmesi için parçalara bölünür.
- Adım 3 - Mapping: Veriler bu aşamada ilgili düğümlere dağıtılır ve kaç tane yedeği olacağı bu adımda belirtilir, ardından ilgili düğümde işlenir.
- Adım 4 - Shuffling : Her düğümde verilerin sayma işlemi yapılır. Örneğin bir text dökumanını input olarak verdiysek ve sonuç olarak hangi kelimenin kaç defa geçtiğini arıyorsak bu aşamada kelime sayıları Node'lerde belirlenir.
- Adım 5 - Reducing: Her Node'den gelen sonuç bu aşamada toplanır.
- Adım 6 - Final Result: Sonuçlar artık elimizdedir. Bunun raporlamasını yapabiliriz.

YARN – Yet Another Resource Negotiator

- YARN, Hadoop 2.0 ile gelen bir geliştirme çabasının sonucu.
- YARN MapReduce üzerinden kaynak planlamasını alarak veriyi işlemeden ayrıştırabilir. Böylelikle MapReduce batch job ile streaming ve interaktif sorgular eş zamanlı çalışma şansını elde etmektedir.
- YARN Hadoop'un yeteneklerini genişleten ve Hadoop ekosisteminin merkezinde yer alan temel bir unsurdur.
- YARN'dan önce Hadoop'un ölçeklenebilirliğinde de sıkıntılar oluşmaya başlamıştı. Yahoo'nun denemelerine göre 5.000 sunucu ve 40.000 görevde Hadoop tıkanıyordu. Ayrıca dinamik kaynak tahsisi de olmadığından cluster kaynakları görevler tarafından verimli bir şekilde kullanılamıyordu.
- YARN sayesinde, başlangıçta sadece MapReduce için tasarlanmış olan Hadoop, Graph Programming gibi yeni programlama yaklaşımlarına da uyum sağlamıştır.

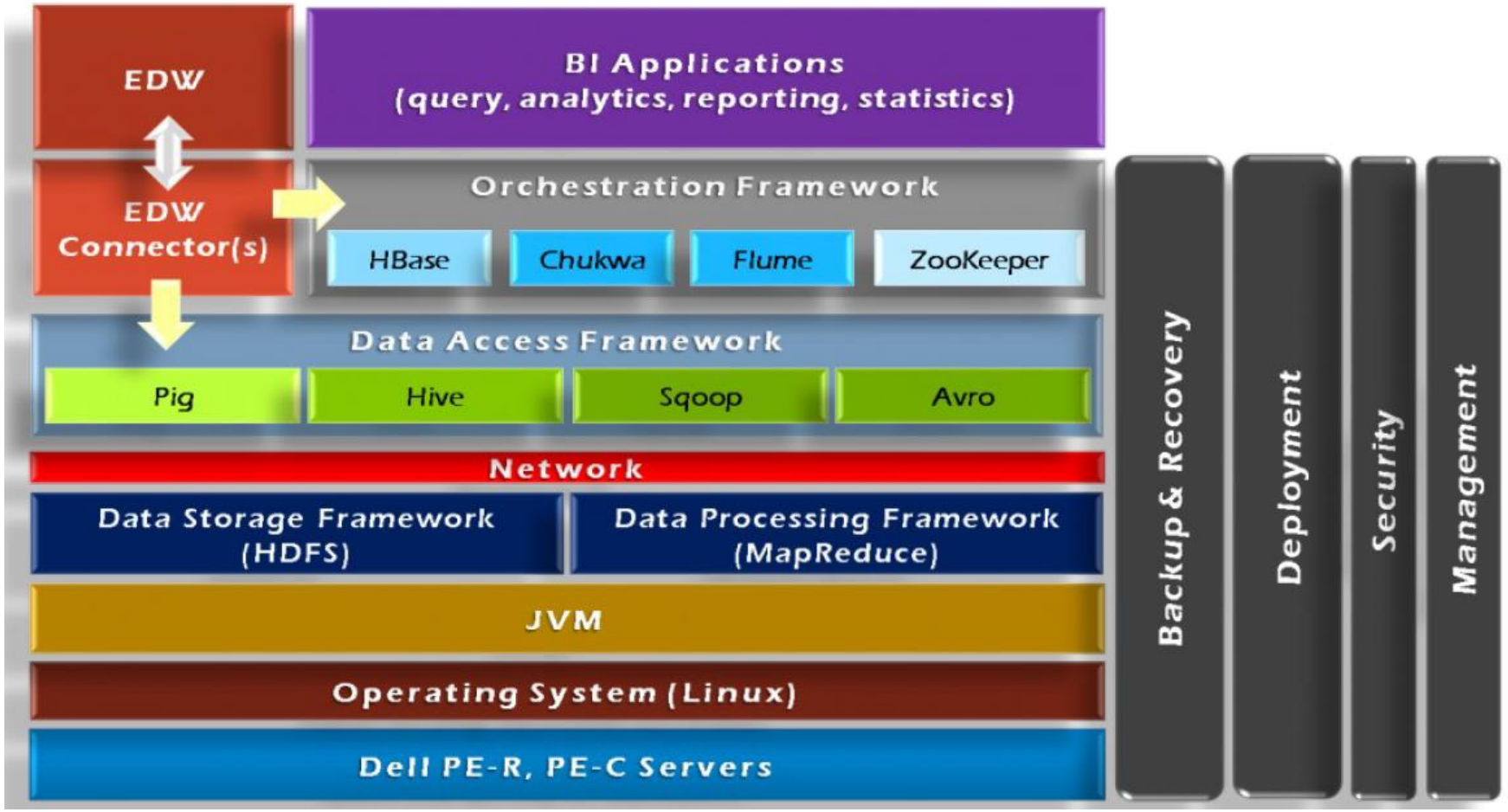
Spark

- Spark büyük veri kümeleri üzerinde paralel olarak işlem yapmamızı sağlayan Scala ile geliştirilmiş açık kaynak kodlu kütüphanedir
- In-Memory çalışmaktadır.
- Map-Reduce veriyi disk ile ram arasında götürüp getirdiği için Spark'a göre daha yavaştır çalışır.
- Spark HDFS'ye erişir ama dağıtık işleme modeli değil de kendi In-Memory modelini uygular.
- Python, R, Scala gibi arayüzleri destekleyerek SQL dilini kullanır.
- Güçlü yanları
  - Streaming veri işleme
  - Grafik veri işleme
  - Ölçeklenebilir/Dağıtık Machine Learning Algoritmaları

Hadoop Sistem Gereksinimleri

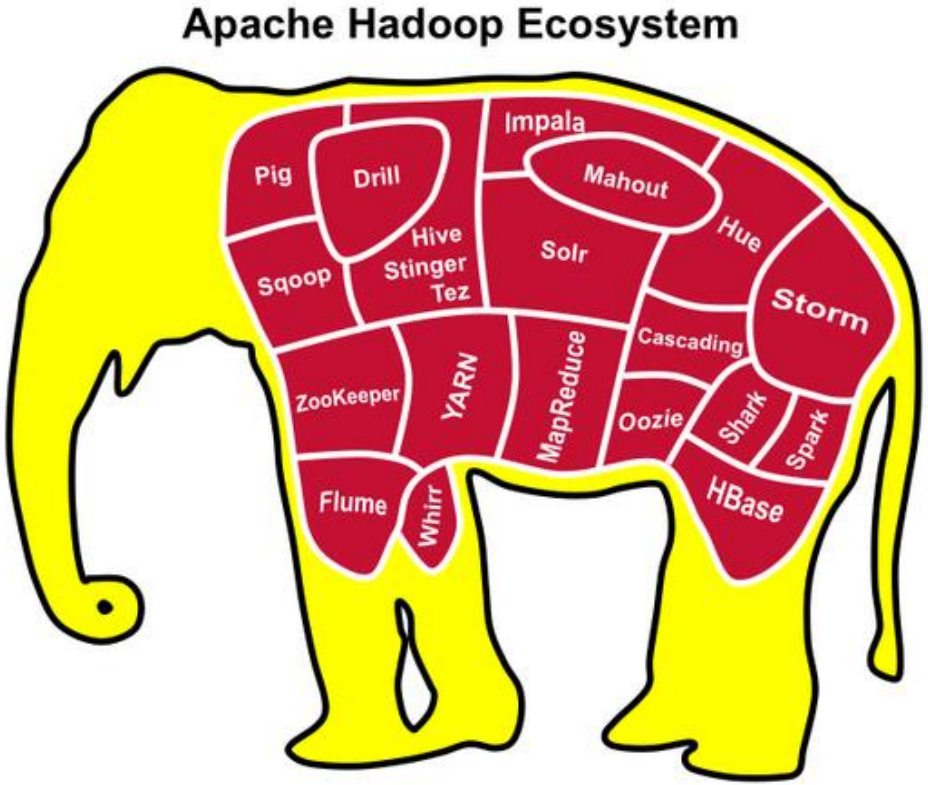
- İşlemci: Intel i7 (i5'de olabilir ancak i7, hatta i9 veya AMD Ryzen olsa daha iyi)
- Ana Bellek (RAM): 32 GB (16 GB olabilir)
- Harddisk: 512 GB SSD
- İşletim Sistemi: Windows 10 / Ubuntu / **CentOS**/Redhat
- Gerekli Yazılımlar:
  - Sanallaştırma Yazılımı (Örnekler vmware workstation üzerinden verilmiştir. Virtual Box veya Hyper-V de kullanılabilir)
  - Ana makineden sanal clustera bağlanmak ve linux komutları kullanmak için ana makine üzerine Cygwin64 veya Putty terminal.

Hadoop Framework



Apache Hadoop Ecosystem

- **Data Access:** Pig, Hive
- **Data Storage**: HBase, Cassandra
- **Interecation, Visualization, Execution, Development :** HCatalog, Lucene, Hama, Crunch
- **Data Serialization:** Avro, Thrift
- **Data Intelligence:** Drill, Mohout
- **Data Integration:** Sqoop, Flume, Chuwka
- **Management:** Ambari(Portal)
- **Monitoring:** Zookeeper
- **Orchestration:** Oozie



Apache Hadoop Ecosystem

- **Hive :** Hive HQL olarak bilinen bir SQL'e çok benzer bir dil ile Hadoop sistemlerinde verilere erişim ve sorgulama gibi işlemleri gerçekleştirir. Gerçek zamanlı sorgulama yapamaz.
- **Pig:** HDFS üzerindeki verinin işlenmesinden sorumludur. Karmaşık veri dönüşüm işlemlerini Java'ya ihtiyaç duymadan Latin gibi betik/script dili ile gerçekleştirmemizi sağlayan Hadoop bileşenidir. Yapısal olan ve yapısal olmayan veriler üzerinde çalışarak veriyi HDFS'de saklayabilir. Hadoop üzerindeki veriyi paralel bir şekilde işler. Peki bunu nasıl yapıyor?
  Pig, Latin dilini kullanıyor. Latin dilinde yazılmış görevleri otomatik olarak Java/Map-Reduce görevine çevirir. Kısa tabiriyle Pig, Latin Scriptleri YARN üzerinde çalışan Map-Reduce fonksiyonlarına çevirir. Böylece HDFS Node'leri üzerindeki veri işlenmiş olur.
- **HBase :** HDFS üzerinde çalışan bir NoSQL veritabanı yönetim sistemidir. SQL desteği sunmaz. Bir HBase sistemi bir grup tablolardan oluşur ve tablolara erişmek için bu tablolarda birincil anahtar kullanılır.
- **Sqoop :** Yapısal verilerin ETL(Extract/Transform/Load) ile Hadoop'a aktarılması için kullanılır. Bir komut satırı arayüzüne sahiptir.
- **Ambari :** Hadoop Node'lerini yönetmek için kullanılan web ara yüzüdür.
- **Flume:** Streaming verilerin toplanması ve birleştirilmesi için kullanılır.
- **Kafka:** Bir mesajlaşma servisidir. Veriyi Hadoop'a stream olarak aktarır.
- **HCatalog :** HDFS sisteminde kayıtlı olan her verinin konumunu ve şema bilgisini tutar. Pig Latin dili bu araç üzerinden **HCatLoader**(okuma) **HCatStorer**(Yazma) API'leri ile HCatalog tarafından yönetilen tablolara okuma yazma yapar.

Data as a Service (DaaS)

- Temel olarak «Data as a Service» kavramını «Software as a Service» kavramının bir türevidir. "as a service" ailesinin bütün üyeleri gibi, DaaS tedarikçinin ve tüketicinin coğrafi veya örgütsel ayrımına bakılmaksızın, ürünün kullanıcıya talep üzerine sağlanmasını kavramı üzerine kuruludur.
- Data as a Service (DaaS), veri dosyalarının (metin, resimler, sesler ve videolar dahil) ağ üzerinden müşterilere, genellikle Internet üzerinden erişilebildiği bir bilgi sağlama ve dağıtım modelidir.