

BLM442 Büyük Veri Analizine Giriş

İlişkisel Veritabanları ve SQL



Dr. Süleyman Eken

Bilgisayar Mühendisliği
Kocaeli Üniversitesi

Sunum Planı

- Temel Kavramlar
- Varlık-İlişki (Entity-Relationship) Modeli
- Verileri düz dosyalar şeklinde saklama
- Oracle, Microsoft SQL Server, IBM DB2, vd (ticari)
- MySQL, SQLite, PostgreSQL, vd (açık kaynak)
- Veri oluşturma ve yükleme
- Sorgular, joins, veri modifiyesi
- ipython-sql: Jupyter Notebook vasıtasıyla ilişkisel veritabanına erişim

Verileri dosyalarda tutmuş olsak

■ Geleneksel Dosya Sistemleri

- Veritabanı yönetim sistemleri öncesinde veri depolamak için kullanılan sistemlerdir.

■ Sakıncaları

- Veri tekrarı
- Verinin birkaç dosyada güncellenmesi
- Belleğin tekrarlı bilgi nedeniyle israfı
- Sadece belirli bir dilin kullanılması

VT ve VTYS?

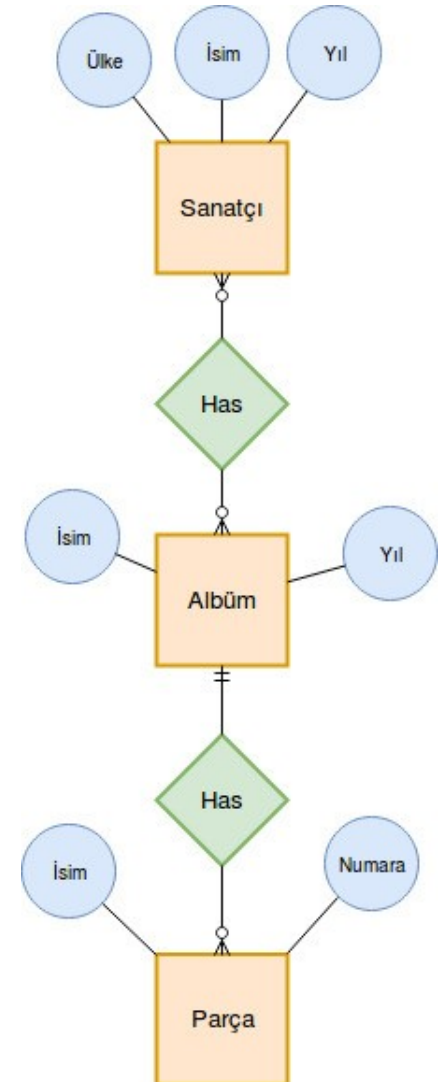
- Veritabanı (Database)
 - Gerçek dünyanın bir yönünü modelleyen birbiriyle ilişkili (inter-related) ve düzenli bilgiler topluluğudur.
 - Günümüzde hemen hemen tüm bilgisayar uygulamalarında core bileşen olarak kullanılmaktadır.
- Veritabanı Yönetim Sistemi (VTYS)
 - Bir veritabanını oluşturmak, saklamak, çoğaltmak, güncellemek
 - ve yönetmek için kullanılan programlara denir.

Varlık-İlişki (Entity-Relationship) Modeli

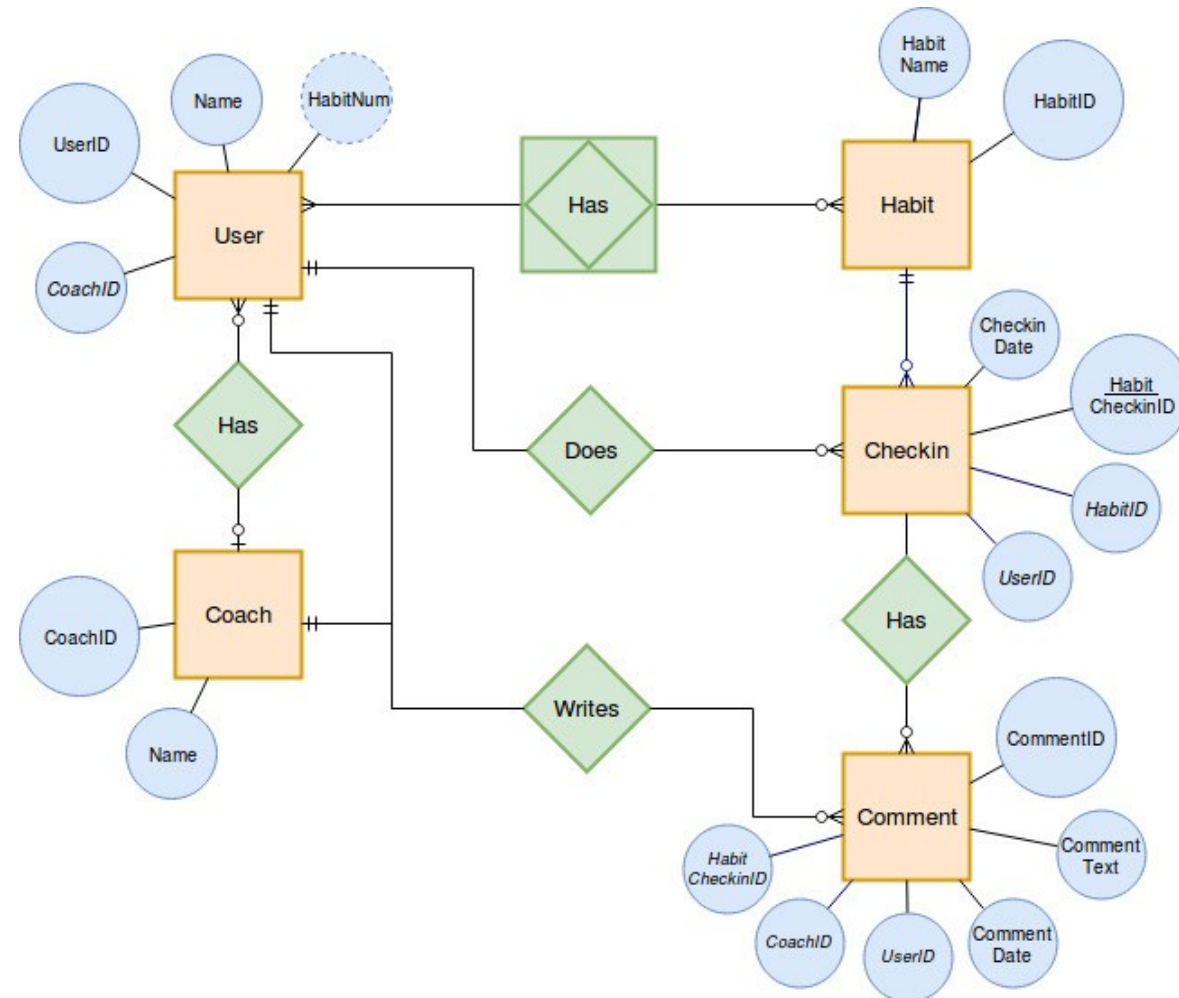
- Dijital bir müzik mağazasını modelleyen bir veritabanı oluşturalım.
- İhtiyacımız olan şeyler:
 - Sanatçılar hakkında bilgi
 - Bu sanatçıların yayınladığı albümler
 - Bu albümlerdeki parçalar

Varlık-İlişki (Entity-Relationship) Modeli

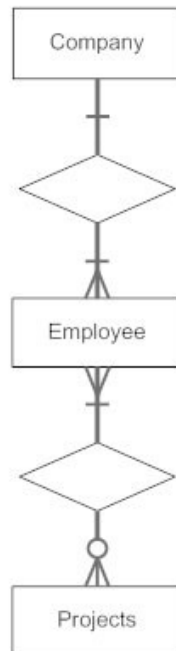
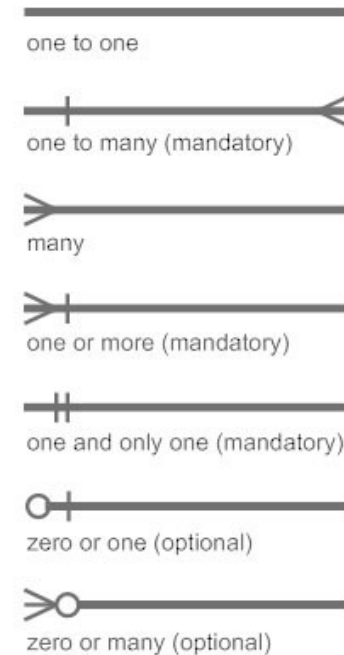
- Sanatçıların isimleri, işe başladıkları yıl ve ülkeleri vardır.
- Albümlerin isimleri var, yayın yılı.
- Parçaların bir adı ve numarası var.
- Bir Albümde bir veya daha fazla Sanatçı var.
- Bir Albümde birden fazla Parça var.
- Bir Parça sadece bir Albümde görünebilir.



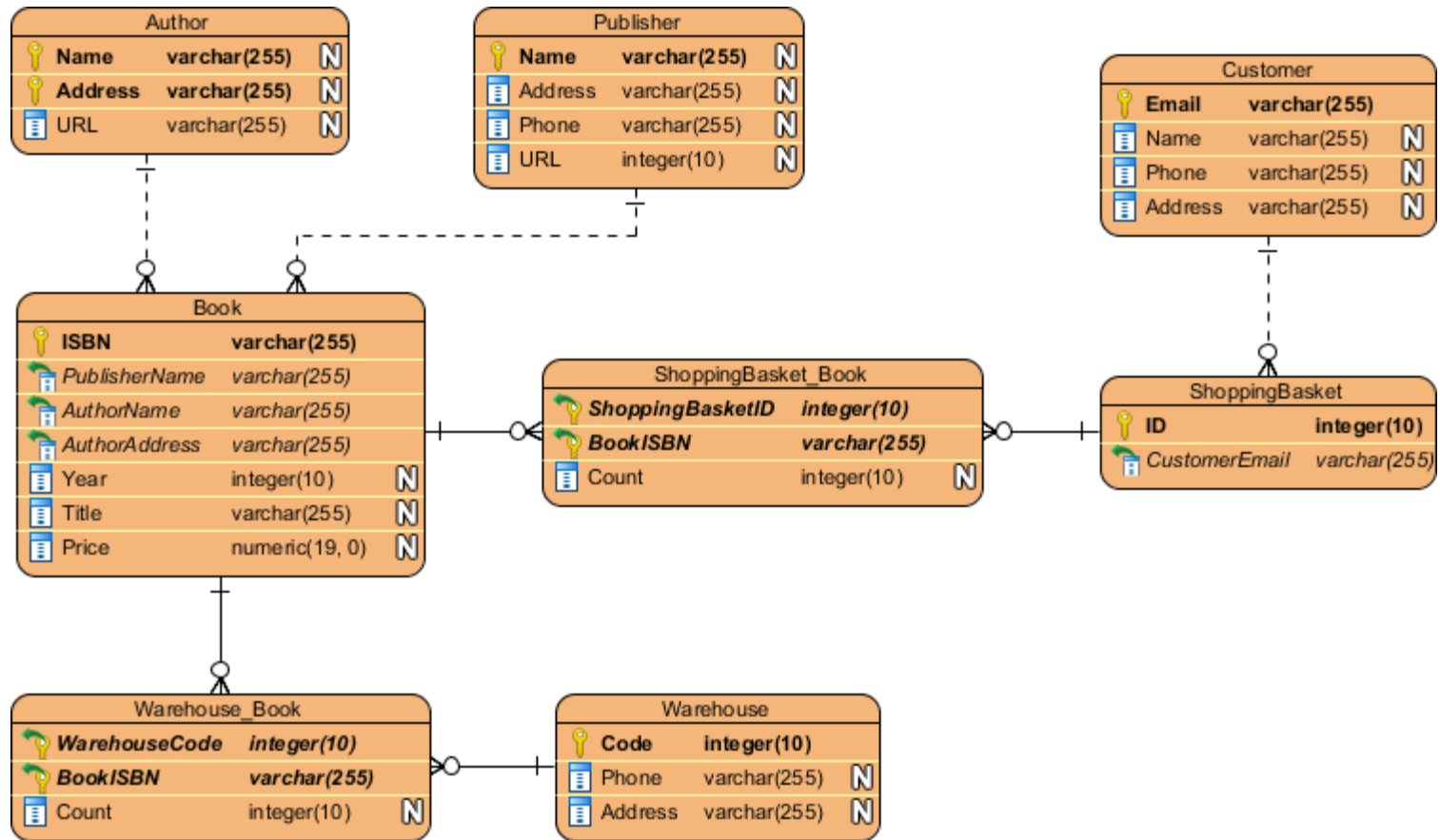
Varlık-İlişki (Entity-Relationship) Modeli



Information Engineering Style



Varlık-İlişki (Entity-Relationship) Modeli



Düz dosya (flat file)

- Değerleri virgülle ayrılmış olarak (CSV) verileri saklayalım.
 - Varlık başına ayrı bir dosya
 - Uygulamada kayıtları okumak/güncellemek istediğinde dosyaları ayrıştırmak zorundayız.
 - Ice Cube'ün parça çıkardığı yılları bulalım

Sanatçı (isim, yıl, ülke)

```
"Wu Tang Clan",1992,"USA"  
"Notorious BIG",1992,"USA"  
"Ice Cube",1989,"USA"
```

Albüm (isim, sanatçı, yıl)

```
"Enter the Wu Tang","Wu Tang Clan",1993  
"St.Ides Mix Tape","Wu Tang Clan",1994
```

```
for line in file:  
    record = parse(line)  
    if "Ice Cube" == record[0]:  
        print int(record[1])
```

Düz dosya üzerine deli sorular

■ Data integrity

- Sanatçı bilgilerinin her albüm girişi için aynı olmasını nasıl sağlayabiliriz?
- Biri albüm yılını geçersiz bir string ile değiştirirse?
- Bir albümde birden fazla sanatçı olacak şekilde nasıl kaydederiz?

■ Implementation

- Belirli bir kaydı nasıl bulursunuz?
- Aynı veritabanını kullanan yeni bir uygulama oluşturmak istiyorsak?
- İki thread aynı anda aynı dosyaya yazmaya çalışırsa ne olur?

■ Durability

- Bir kaydı güncellerken makine çökerse?
- Veritabanını yüksek kullanılabilirlik için birden fazla makineye kopyalamak istiyorsak ne olur?

Veritabanı yönetim sistemleri (VTYS)

- VTYS, uygulamaların bir veritabanında bilgileri depolamasını ve analiz etmesini sağlayan yazılımdır.
- Genel amaçlı bir VTYS, veritabanlarının tanımlanmasına, oluşturulmasına, sorgulanmasına, güncelleştirilmesine ve yönetimine izin vermek için tasarlanmıştır.
- VTYS'ler hemen hemen her uygulamada, web sitesinde, aklınıza gelebilecek yazılım sisteminde kullanılır.

Veritabanı yönetim sistemleri (VTYS)

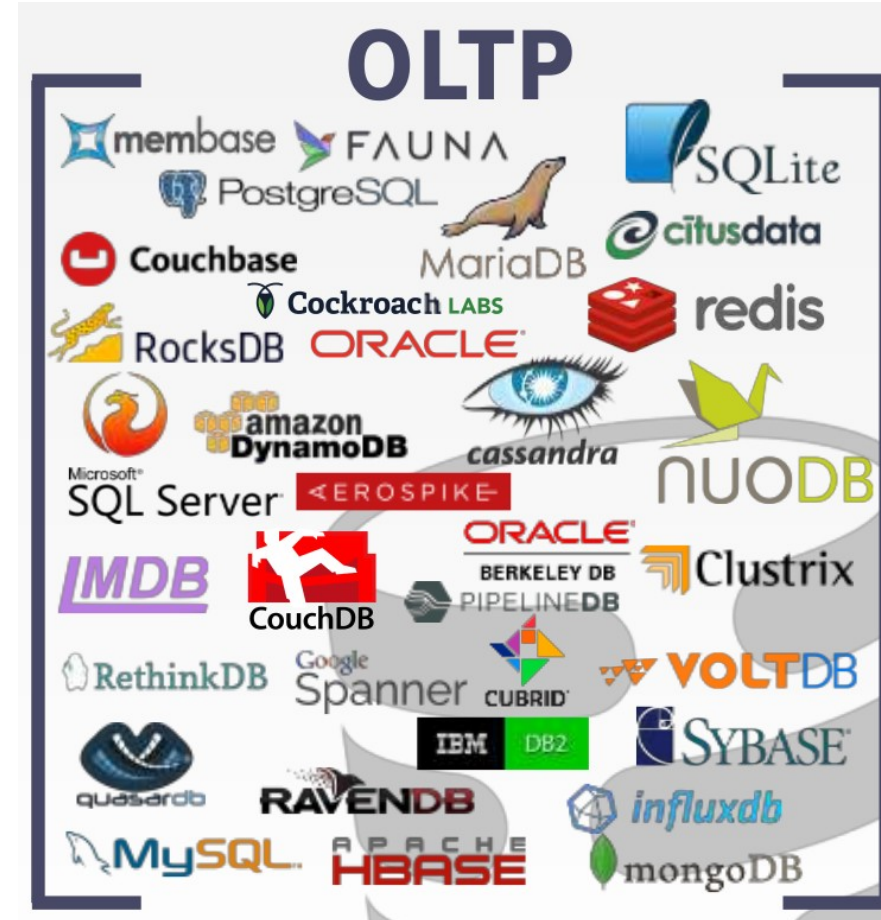
- Veritabanı Yönetim Sistemlerinin Avantajları
 - Gereksiz veri tekrarının olmaması
 - Veri güvenliği
 - Veri bütünlüğü
 - Veri bağımsızlığı
- Veritabanı Yönetim Sistemlerinin Dezavantajları
 - Veri tabanı sisteminin kurulumu ve bakımı klasik dosya sistemine göre daha maliyetli olabilir.

VTYS Aktörleri

- Veritabanı Yönetim Sistemleri Aktörleri
 - Sistem mühendisleri
 - VTYS yöneticisi (Admin)
 - VYTS tasarımcısı
 - Uygulama geliştirenler
 - Son kullanıcılar
 - Casual End users: sıradan
 - Sophisticated End Users: uzman kullanıcılar

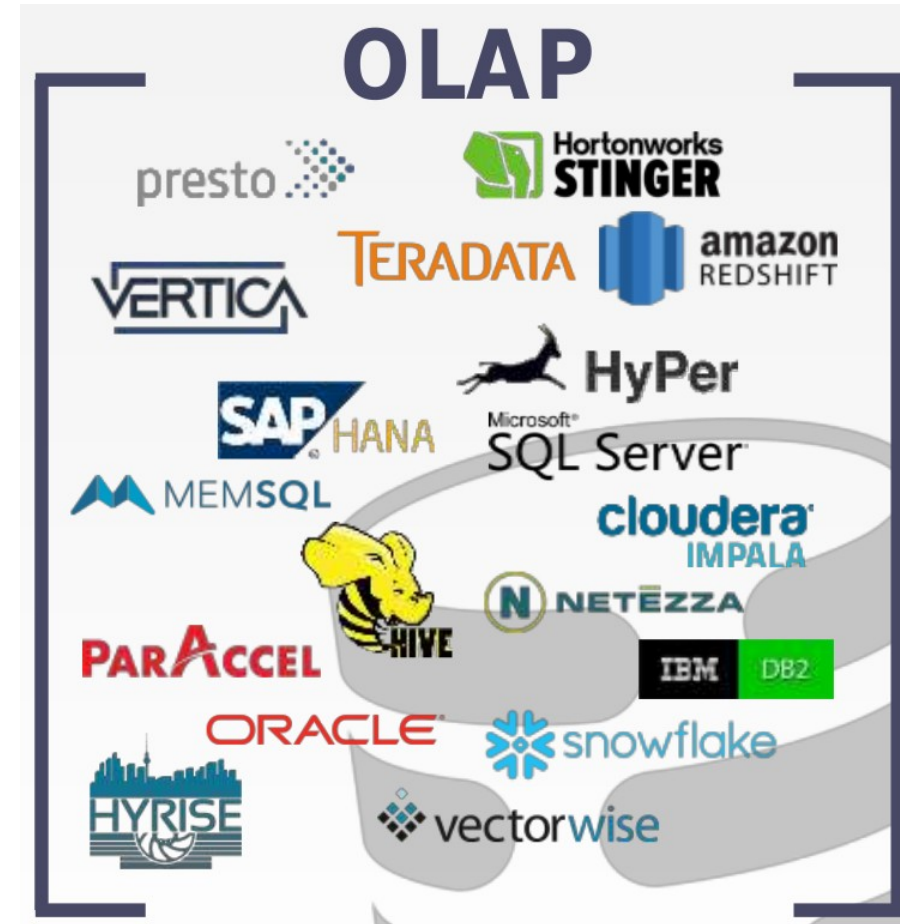
VTYS Tipleri: Hedef çalışma yükü açısından

- OLTP (On Line Transactional Processing)
- Organizasyonlarda kullanılan veri girişi, veri güncelleme, veri silme gibi işlemlere olanak tanıyan sistemlerdir.
- Günlük hayatta kullandığımız çoğu veri tabanı ve veri tabanına bağlı programlar OLTP tarzı işlem gören veri tabanlarıdır.
- Küçük miktarda veri üzerinde işlem yapılır, karmaşık ilişkiler bulunan büyük veriler için uygun değildir.
- OLTP sistemlerde tablolar arasında ilişkiler oluşturulur, burada normalizasyon seviyesine dikkat edilmeli ve ayrıca ACID prensiplerine göre işlem görülmelidir.



VTYS Tipleri: Hedef çalışma yükü açısından

- OLAP (On-line Analytical Processing)
- Arka planda uzun soluklu analizler için uygun bir yöntemdir.
- Veri boyutu yüksek ve karmaşık ilişkiler çok olmalı
- Analitik işlemlerden kasıt, OLAP'ın OLTP tipi sistemlerden verileri alıp, belirli kriterlere göre gruplamasıdır.

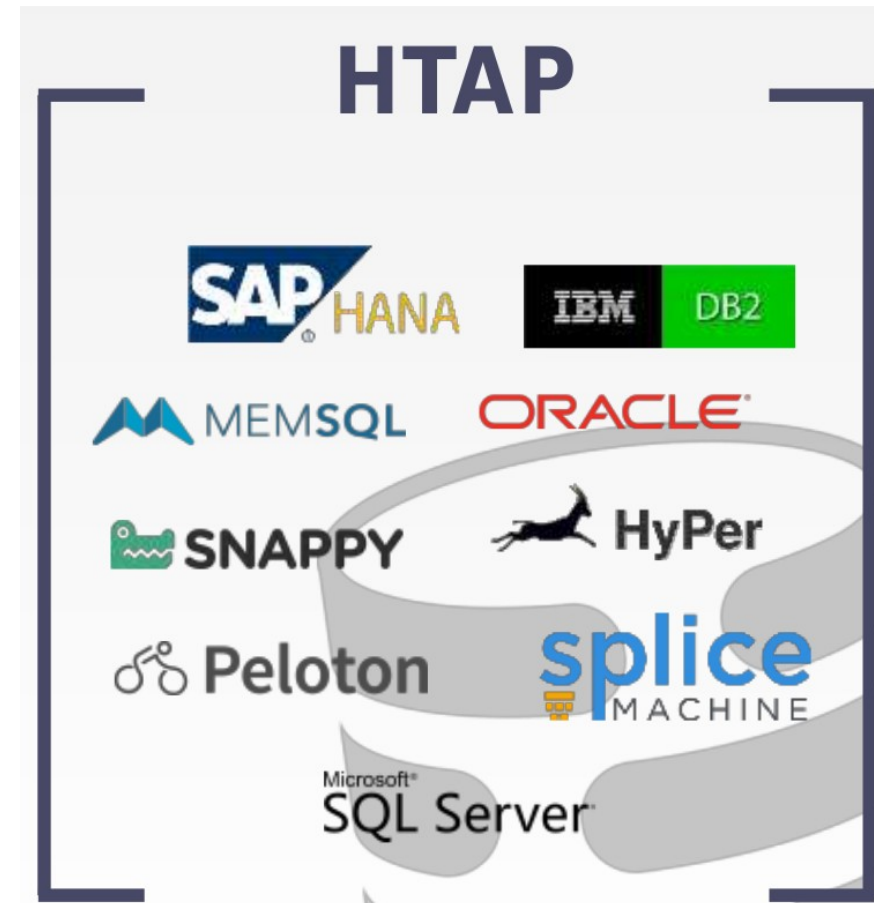


OLAP Vs. OLTP

	OLTP System Online Transaction Processing (Operational System)	OLAP System Online Analytical Processing (Data Warehouse)
Source of data	Operational data; OLTPs are the original source of the data.	Consolidation data; OLAP data comes from the various OLTP Databases
Purpose of data	To control and run fundamental business tasks	To help with planning, problem solving, and decision support
What the data	Reveals a snapshot of ongoing business processes	Multi-dimensional views of various kinds of business activities
Inserts and Updates	Short and fast inserts and updates initiated by end users	Periodic long-running batch jobs refresh the data
Queries	Relatively standardized and simple queries Returning relatively few records	Often complex queries involving aggregations
Processing Speed	Typically very fast	Depends on the amount of data involved; batch data refreshes and complex queries may take many hours; query speed can be improved by creating indexes
Space Requirements	Can be relatively small if historical data is archived	Larger due to the existence of aggregation structures and history data; requires more indexes than OLTP
Database Design	Highly normalized with many tables	Typically de-normalized with fewer tables; use of star and/or snowflake schemas
Backup and Recovery	Backup religiously; operational data is critical to run the business, data loss is likely to entail significant monetary loss and legal liability	Instead of regular backups, some environments may consider simply reloading the OLTP data as a recovery method

VTYS Tipleri: Hedef çalışma yükü açısından

- HTAP (Hybrid Transaction + Analytical Processing)
- Aynı veritabanı instance içinde bulunur



Veri modeli Vs. şema

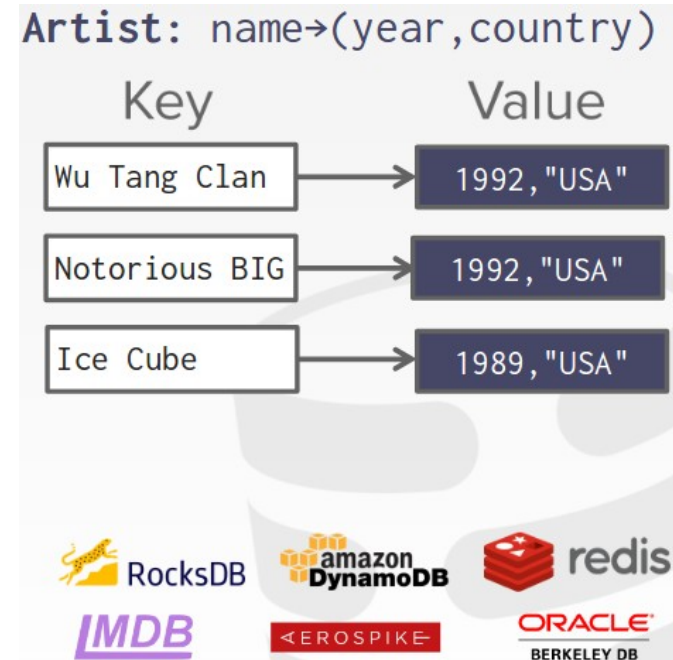
- Veri modeli: şemada neler olabileceğine karar veren yüksek düzeyde bir tasarım uygulamasıdır.
- Şema (schema): bir veri modeli kullanarak hangi alanların bulunacağını ve türlerinin ne olacağını belirleyen veri tabanının bir planıdır.
 - employee_ID column: 10 uzunluklu bir string gibi
- Örneğin bir veri modeli, verilerin tablolarda düzenleneceği ilişkisel bir model olabilir. Bu model için şema, nitelikler (attributes) kümesi ve bunlara karşılık gelen alanlar (domains) olacaktır.

VTYS Tipleri: Veri modeli açısından

- Relational ← bir çok VTYS
- Key/Value
- Graph
- Document
- Column-family ← NoSQL
- Array / Matrix ← Makine öğrenmesi
- Hierarchical
- Network ← çok nadir kullanılan

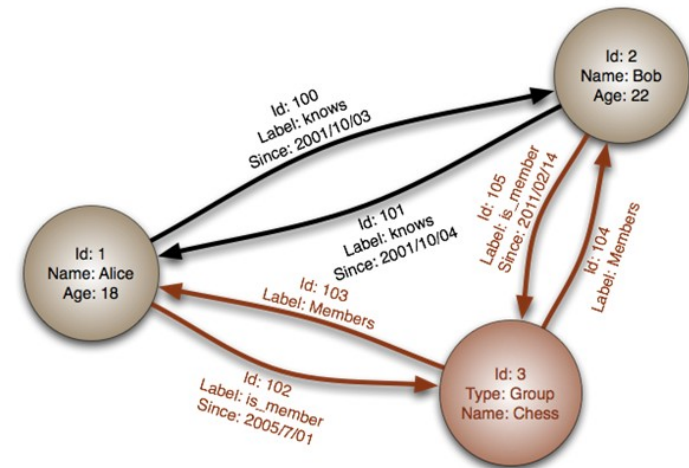
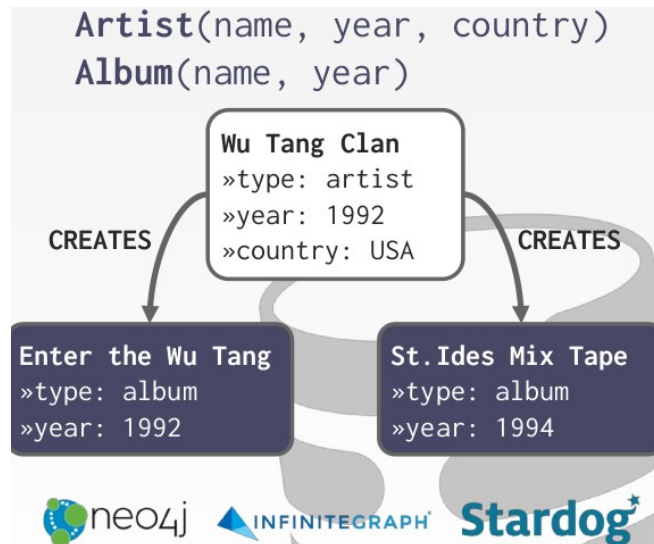
NoSQL VYTS: key-value tabanlı

- Veriler, dictionary (sözlük) veya hash tablosu olarak da adlandırılan bir “key” değerini “value” ya eşleyen bir yapıda tutulur.
- Küçük ve çok sayıda okuma yazma işleminin yapıldığı uygulamalar için uygundur.
- Bir anahtara karşılık gelen veri genellikle boolean, integer gibi basit verilerdir.
- Bu tür veritabanları, önbellek (caching) yazılımları, alışveriş sepeti uygulamaları ve görüntü dosyalarının saklanması gibi uygulamalar için uygundur.



NoSQL VYTS: çizge tabanlı

- Çizge tabanlı veritabanlarında veriler düğümler (node), ilişkiler (edge) ve özellikler (properties) şeklinde tutulurlar.
- Diğer veritabanı türlerinden farklı olarak veriler arasındaki ilişkiler de saklanabilir.
- Sorgulama dilleri: SPARQL, Gremlin, Cypher



NoSQL VYTS: doküman tabanlı

- Bir key'e karşılık gelen veriler “doküman” adı verilen nesnelerde saklanırlar.
- Nesneler genellikle JSON formatındadır. Dokümanlar çok sayıda alan içerebilir ve her dokümanın yapısı birbirinden farklı olabilir.
- Doküman tabanlı veritabanları, içerik yönetim sistemleri, elektronik ticaret uygulamaları ve günlük (blog) siteleri gibi esnek veri yapısına ihtiyaç duyan uygulamalar için uygundur.

```
{  
  "name":      "Wu Tang Clan",  
  "year":      1992,  
  "country":   "USA"  
}
```

```
{  
  "name":      "Ice Cube",  
  "year":      1989,  
  "country":   "USA",  
  "city":      "Los Angeles"  
}
```

NoSQL VYTS: kolon tabanlı

- Kolon tabanlı veritabanları, yüksek okuma yazma performansı ve yüksek erişilebilirlik (high availability) için tasarlanmıştır.
- Birden çok sunucu üzerinde dağıtık olarak çalışırlar.
- İçerik yönetim sistemleri, günlük (blog) uygulamaları, uygulama kayıtlarının (log) saklanması

Row Key

	year	country
Wu Tang Clan	1992	USA

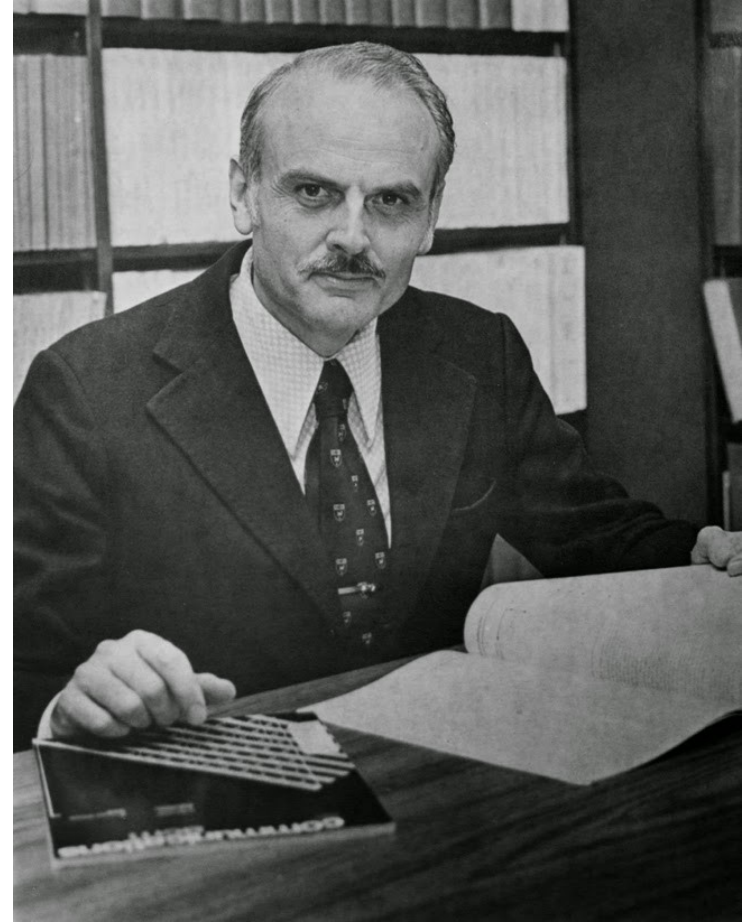
	year	country	city
Ice Cube	1992	USA	Los Angeles

NoSQL VYTS üstünlükleri

- Büyük verilerin yüksek performansla sorgulanabilmesi
- Yatay bölümlendirme (sharding) kolaylığı
- Dağıtık sistemlerdir ve yatay genişlemeye uygundur.
- Donanım olarak, pahalı sunucular yerine sıradan sunucularla (commodity hardware) çalışabilirler.
- Birçok NoSQL çözümü ücretsizdir (açık kaynaklıdır) veya ilişkisel veritabanlarına göre çok düşük lisans ücretleri vardır.

İlişkisel model pioneri

- Model 1970'de Edgar F. Codd tarafından önerildi.
- Basit veri yapısında veritabanını tutma
- Yüksek seviye dil ile veriye erişim
- Fiziksel depolama uygulamaya bırakıldı



Populer VTYS'leri

- Ticari sistemler
 - Oracle
 - Microsoft SQL Server
 - IBM DB2
 - Diğerleri ...
- Açık kaynak sistemler
 - MySQL
 - SQLite
 - PostgreSQL
 - Others ...

Spreadsheets Vs. Databases

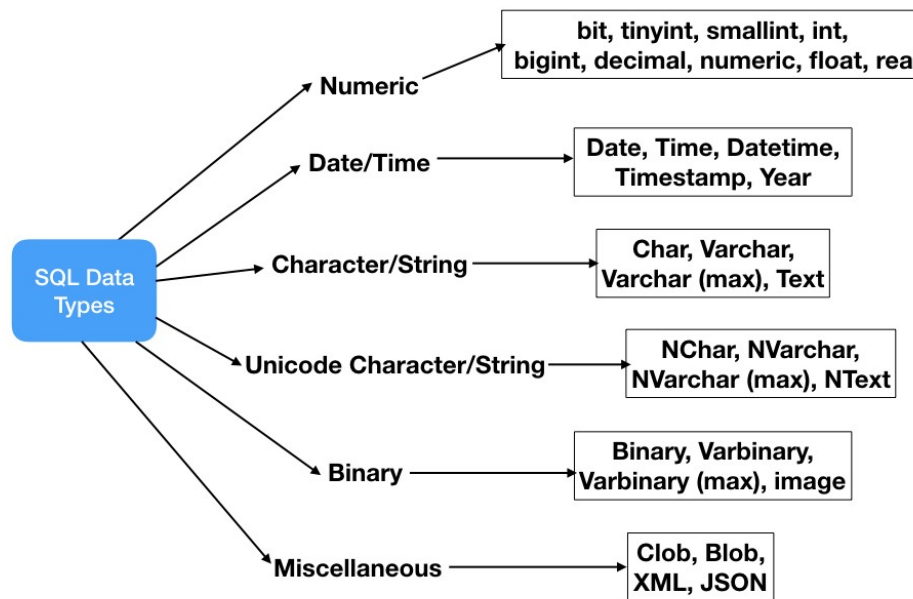
- Elektronik tablolar genellikle aşağıdakilere uygundur:
 - Veri analizi
 - Hesaplamalar
 - İstatistiksel karşılaştırmalar
 - Grafik yönetimi
- Veritabanları aşağıdakiler için iyi çalışır:
 - Veri yönetimi
 - Karmaşık arama sorguları
 - Birden çok kullanıcı
 - Veri alt kümelerini görüntüleme
 - Veri özeti raporları

Temel Kavramlar – tablo, satır

- Veritabanı içerisinde tutulacak verileri taşımak için kullanılır.
- Tablo (relation), satır (tuple) ve sütunlardan (column) oluşan verilerin depolandığı veritabanı elemanıdır.
- Bir veritabanı içerisinde birden fazla tablo kullanılabilir.
- Öğrenci bilgilerinin tutulduğu bir veritabanındaki öğrenci tablosu öğrencilerin no, ad, soyad gibi bilgilerini tutacaktır. Buradaki no, ad ve soyad bilgilerinin her biri bir sütunla gösterilecektir.
- Satır ise her bir öğrencinin tüm bilgileridir. Satır ifade yerine kayıt veya tuple ifadesi de kullanılmaktadır.

Temel Kavramlar – veri tipleri

- Oluşturulan veritabanında tutulan verilerin hepsi aynı türde (types) değildir. Tutulacak veriye göre değişiklik gösterir. Örneğin, isim karakter, no bilgisi sadece sayı, doğum tarihi bilgisi tarih bilgisini içerir.



Temel Kavramlar – primary ve foreign key

- Tabloda tutulan verilerden benzersiz yani aynı değeri iki kez içermeyecek olan sütun birincil anahtar (primary key) olarak belirlenir.
- Öğrenci tablosu için “primary key” öğrenci numarası olabilir.
- Tablodaki bir anahtar alan başka bir tablodaki anahtar alanı gösteriyorsa: “foreign key”
- Yabancı anahtar genelde diğer tablolarla ilişki kurmak için kullanılır.

ogrenci			bolum	
No	Adı	Böl. Kodu	Kodu	Adı
53	Murat	4	2	Bilgisayar Programcılığı
34	Mustafa	4	4	Elektrik
28	Ahmet	2	5	İnşaat
21	Ayhan	2		
49	Kadir	5		

Primary Key

Foreign Key

Primary Key

SQL (Structured Query Language)

- İlişkisel veritabanlarında çok geniş bir kullanım alanına sahiptir.
- SQL ile kullanıcılar veritabanı sistemleri ile iletişim kurmaktadır.
- SQL bir programlama dili değildir. SQL komutları kullanarak veritabanına kayıt ekleme, kayıt silme, kayıt güncelleme, tablo oluşturma ve kayıt listeleme gibi bir çok işlem gerçekleştirilir.
- Standart SQL ifadelerinde fonksiyon, döngü, karşılaştırma ifadeleri gibi programlamaya yönelik ifadeler kullanılamamaktadır.
- Bu sorunu çözmek için veritabanı sistemlerinde PL/SQL ve T- SQL sorgulama dilleri geliştirilmiştir.

SQL (Structured Query Language)

- **CREATE DATABASE** isim
- Bu komut belirtilen isimde veritabanı (database) oluşturur.
- **CREATE TABLE** tabloadı (
sütunAdı1 veriTipi diğerParametreler ,
sütunAdı2 veriTipi diğerParametreler ,

\

```
CREATE TABLE ogrenci (  
    ogrno INT NOT NULL , -- ogrno tamsayı türünde ve boş geçilemez.  
    ad varchar(40) NOT NULL , -- ad değişken uzunlukta karakter türünde ve boş geçilemez.  
    soyad varchar(40) NOT NULL , -- soyad değişk. uzunlukta karakter türünde ve boş geçilemez.  
    dogumtarihi DATE , -- dogumtarihi tarih türünde (boş geçilebilir)  
    ortalama REAL , -- ortalama ondalık sayı türünde (boş geçilebilir)  
    PRIMARY KEY (ogrno) -- ogrno alanı birincil anahtar olarak seçilmiş.
```


SQL (Structured Query Language)

- Gerisine burdan bknz:

<https://github.com/enochtangg/quick-SQL-cheatsheet>

- Piazza ders kaynaklar içindekilere bknz:
 - Week3_CheatSheet_SQL.pdf
 - Week3_CheatSheet_sql.pdf

SQLite Nedir?

- SQLite, dünyada en çok dağıtılan ve geliştiricilere kullanması için tavsiye edilen
- Tamamen açık kaynak kodlarına sahip
- Sunucu yazılımı ve yapılandırma gereksinimi olmayan
- İşlemsel ve ilişkisel bir SQL veritabanı motorudur.
- SQLite'in çalışması için herhangi bir sunucuya ihtiyacı olmadığı için, kurulum ve ya konfigürasyon adımları yoktur.
- Her veritabanı için sadece bir dosya vardır. Bu da veritabanının yedeklenmesini ve kopyalanmasını kolaylaştırır.
- Platform bağımsızdır.

Anaconda nedir?

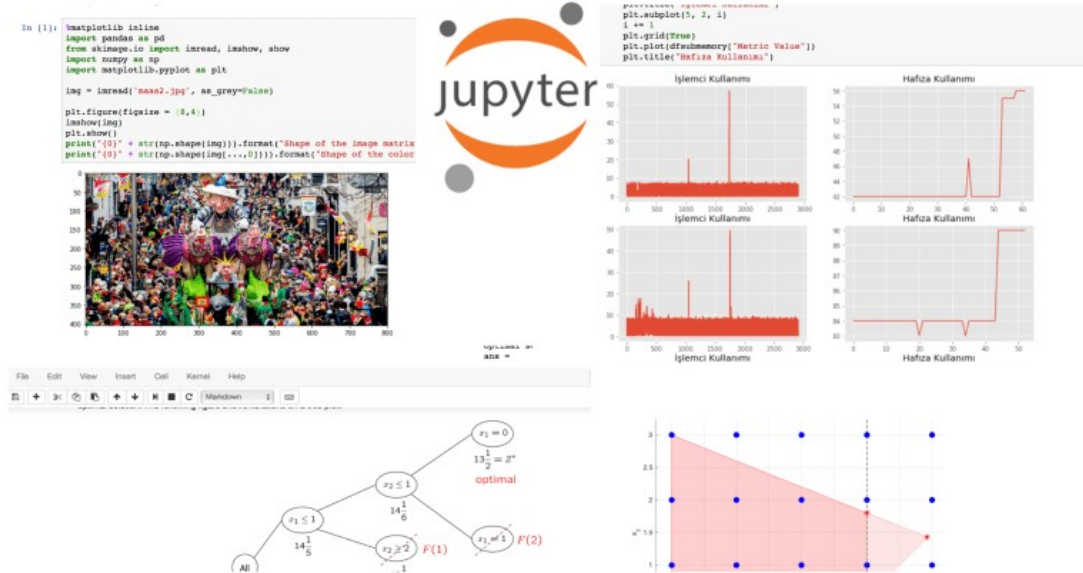
- Anaconda, veri bilimi ve benzeri bilimsel uygulamalar için python kullanmak isteyenlere hazırlanmış, bir çok paket programı içeren tümleşik bir python dağıtımıdır.
- Kütüphanelerin yanı sıra jupyter notebook ve spyder gibi araçları da barındırır.
- Kurulum için Piazza kaynaklardan [Anaconda_ve_bazi_paketlerin_kurulumu.pdf](#) bakınız.
- Başlangıç için [Anaconda_Starter_Guide_CheatSheet.pdf](#) bakınız.
- Conda (paket ve sanal ortam -environment- yöneticisi) ile ilgili kullanım bilgilerine [conda-cheatsheet.pdf](#) dosyasından bakabilirsiniz.

Jupyter Notebooks

- Aldığınız notları ve hesaplamalarınızı bir arada tutmak için kullanabileceğiniz en başarılı araçlardan biridir.
- Tekrar edilebilir araştırmada (reproducible research) ve veri biliminde sıkça kullanılır.
- Jupyter Notebook'u Python dışında başka pek çok programlama dili¹ ile de kullanabilirsiniz.
- Kabaca her defterin alt alta sıralanmış kutucuklardan (boxes/cells) oluştuğunu düşünebilirsiniz. Bu kutucuklarda kodlama ve hesaplama işlerinizi yapabilirsiniz.

¹<https://github.com/jupyter/jupyter/wiki/Jupyter-kernels>

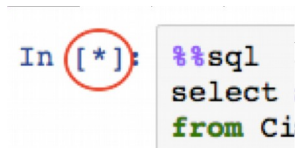
Jupyter Notebooks



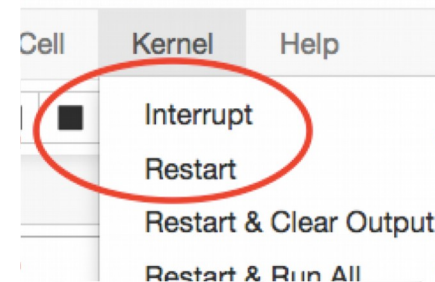
- Bir hücrenin içine yazılanları çalıştırmak için, hücrenin içine tıklayıp, Ctrl ve Enter tuşlarına birlikte basmanız (Ctrl + Enter) yeterlidir.
- Ayrıca bu hücrelere metin ve matematiksel denklemler de yazabilirsiniz. O yazılar için Markdown adı verilen ve basit kurallar ile metin biçimlendirmeye yarayan bir dil kullanılıyor.

Jupyter Notebooks

- Öğrenciler bulutta, Instabase, Google Cloud ve Amazon Web Services'de notebook kullanabilirler.
- Her durumda da, notebook'lar bir web tarayıcısında çalışır.
- Hiçbir şey olmazsa, bazı hücreler muhtemelen hala yürütmededir.



```
In [*]: %%sql
select
from Ci
```



- Kernel > Interrupt veya Kernel > Restart deneyin
- Jupyter Notebook kullanım için [cheat-sheet_Jupyter Notebook.pdf](#) dosyasını inceleyiniz.

Uygulamalar

- SQLite_Tutorial.ipynb
 - grades.csv
 - demographics.csv
- ex01-Quick Start.ipynb