Database (Veritabanı)

Database (Veritabanı)

- **Veritabanları sistematik erişim imkanı olan, yönetilebilir, güncellenebilir, taşınabilir, birbirleri arasında tanımlı ilişkiler bulunabilen bilgiler kümesidir.**
- **Bir bilgisayarda sistematik şekilde saklanmış, programlarca işlenebilecek veri yığınlarıdır.**

Veritabanları temelde iki kategoriye ayrılırlar:

1 – Relational Databases (SQL)

(İlişkisel Veritabanları)

2 – Non-relational Databases (NoSQL)

(İlişkisel Olmayan Veritabanları)

Relational Databases (SQL)

- **İlişkisel veri tabanını çeşitli tablolar arasında organize edilmiş verilerden oluşan veri tabanı olarak açıklanabilir. Bu farklı tablolar arasındaki veriler, çeşitli anahtarlar vasıtası ile birbirlerine bağlanırlar.**

Relational Databases (SQL)

- **İlgili tablolarda, sütunlar arasında bir anahtar sütun yer alır. Bu anahtar sütun aracılığı ile birden çok tablo verileri birbiriyle bağlantı sağlayabilir ve herhangi bir sorgulamada birlikte görüntülenebilir.**
- **Bu tür veri tabanları arasında dBase, Informix, Ingres, MySQL, Oracle ve PostgreSQL başta gelmektedir.**

SQL Örnek Tablo

Personnel Data

IdNum	Gender	Jobcode	Salary	Birth	Hired
1065	M	ME3	38090	26JAN54	07JAN92
1350	F	FA3	36886	31AUG55	29JUL91
1400	M	ME1	29769	05NOV67	16OCT90
1401	M	TA3	38822	13DEC55	17NOV93
1499	M	ME1	23025	26APR74	07JUN92
1639	F	TA1	42260	26JUN70	28JAN91
1653	F	ME2	31896	15OCT64	09AUG92
1919	M	TA2	34376	12SEP66	04JUN87

Non-Relational Databases (NoSQL)

- **İlişkisel olmayan veritabanları, modellenen verilerin depolanması ve okunması için ilişkisel veritabanlarında kullanılan tablo ilişkilerinden farklı yollarla çalışan bir mekanizma sağlar.**

Non-Relational Databases (NoSQL)

- **Yıllardır SQL veritabanları büyük, ölçeklenebilir (daha sonradan koda ilaveler yapılarak büyütülebilen programlar) sistemler inşa etmek isteyen yazılımcılar için tek seçeneklerden biriydi.**
- **Bununla birlikte, karmaşık veri yapılarının depolanabilmesine duyulan ihtiyacın sürekli artması, yazılımcıların heterojen ve yapısal olmayan verileri depolamasına izin veren NoSQL veritabanlarının doğmasına yol açtı.**

NoSQL Örnek Tablo


```
{
  "address": {
     "building": "1007",
     "coord": [ -73.856077, 40.848447 ],
     "street": "Morris Park Ave",
     "zipcode": "10462"
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "grades": [
     { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 },
     { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 },
     { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 },
     { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 },
     { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }
  ],
  "name": "Morris Park Bake Shop",
  "restaurant_id": "30075445"
}
```

SQL <--> NoSQL

- Modeller ilişkisel niteliktedir
- Veriler tablolarda saklanır
- Her kaydın aynı tür ve aynı özelliklere sahip olduğu durumlar için uygundur
- Yeni bir özellik eklemek tüm şemayı değiştirmeniz gerektiği anlamına gelir
- Şemalar (veritabanı modellemeleri) çok katıdır

- Modeller ilişkisel değildir
- Veriler JSON, anahtar/değer vb. tarzlarda depolanabilir (NoSQL veritabanı türüne bağlı olarak)
- Her kaydın aynı nitelikte olması gerekmez, bu da veritabanını çok esnek kılar
- Kolaylıkla verilere yeni özellikler eklenebilir
- Uyulması gereken bir şema koşulu yoktur

En Yaygın Veritabanları

352 systems in ranking, September 2019

Rank			DBMS	Database Model	Score		
Sep 2019	Aug 2019	Sep 2018			Sep 2019	Aug 2019	Sep 2018
1.	1.	1.	Oracle	Relational, Multi-model	1346.66	+7.18	+37.54
2.	2.	2.	MySQL	Relational, Multi-model	1279.07	+25.39	+98.60
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1085.06	-8.12	+33.78
4.	4.	4.	PostgreSQL	Relational, Multi-model	482.25	+0.91	+75.82
5.	5.	5.	MongoDB	Document	410.06	+5.50	+51.27
6.	6.	6.	IBM Db2	Relational, Multi-model	171.56	-1.39	-9.50
7.	7.	7.	Elasticsearch	Search engine, Multi-model	149.27	+0.19	+6.67
8.	8.	8.	Redis	Key-value, Multi-model	141.90	-2.18	+0.96
9.	9.	9.	Microsoft Access	Relational	132.71	-2.63	-0.69
10.	10.	10.	Cassandra	Wide column	123.40	-1.81	+3.85
11.	11.	11.	SQLite	Relational	123.36	+0.65	+7.91

PostgreSQL ve MySQL, Python web uygulamalarının verilerini depolamak için en yaygın veritabanlarıdır.

- **SQLite, diskteki tek bir dosyada depolanan bir veritabanıdır. SQLite Python'da yerleşik olarak bulunur, ancak bir seferde yalnızca tek bir bağlantıyla erişim için inşa edilmiştir. Bu nedenle bir web uygulamasını SQLite ile piyasaya sürmemeniz tavsiye edilir.**

Database Terminolojisi

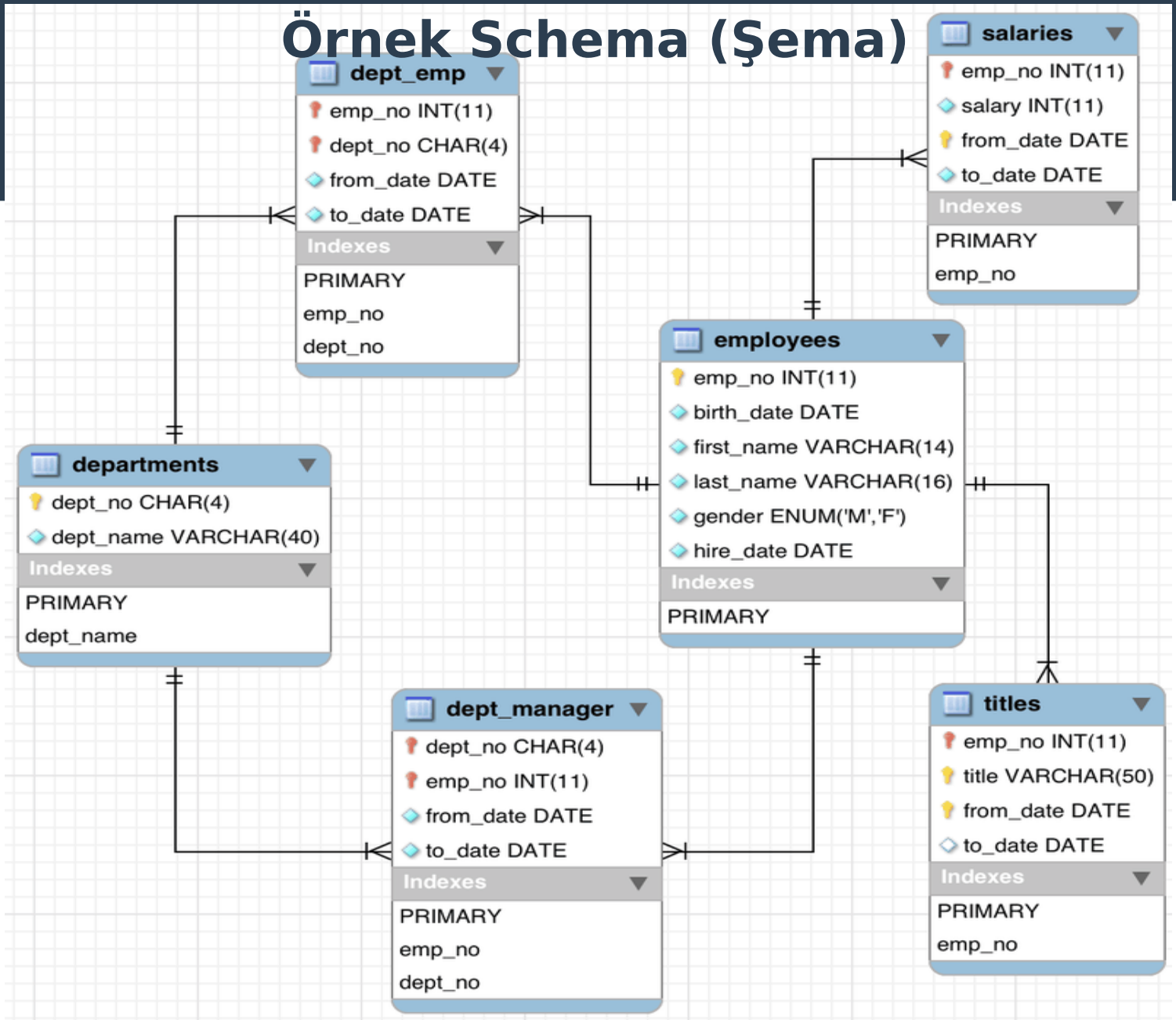
- **Schema**
  - Bir veritabanın tabloları arasındaki ilişkiyi gösteren taslak
- **Table**
  - Veritabanı içindeki her bir veri tablosu
- **Record**
  - Bir tablo içindeki her bir satırda bulunan birbiriyle ilgili veri topluluğu
- **Field**
  - Bir tablo içindeki satır/sütun kesişiminden oluşan veri
- **Entity**
  - Veritabanlarını üzerine inşa ettiğimiz soyut kavramlar
- **Query**
  - Veritabanında ilgili veriye ulaşmamızı sağlayan sorgulama komutları
- **Primary Key**
  - Veriyi benzersiz kılan, o veriye özel tanımlama numarası
- **Foreign Key**
  - İki farklı tablodaki veriler arasında bağlantı kurulmasını sağlayan referans numarası

Database Terms

Name	Age	Gender	Hometown	Finisher
Seth Rollins	29	Male	Buffalo	Pedigree
John Cena	38	Male	West Newbury	Attitude Adjustment
Dean Ambrose	29	Male	Cincinnati	Dirty Deeds
The Rock	43	Male	Miami	Rock Bottom
Brie Bella	31	Female	Sedona	Yes! Lock
Kevin Owens	31	Male	Saint-Jean-sur-Richelieu	Popup Powerbomb

Fields – individual pieces of information
Records – groups of fields which belong together
Table – a collection of records

Örnek Schema (Şema)



Database Relationships

- **One-to-One**
- **One-to-Many (Many-to-One)**
- **Many-to-Many**

One-to-One

- **A tablosundaki bir satırın B tablosunda yalnızca tek bir eşleşmesinin olabileceği ilişki türüdür.**
- **Yaygın bir kullanım değildir çünkü B tablosundaki veri A tablosuna taşınarak birden fazla tablo oluşturma ihtiyacı ortadan kaldırılabilir.**
- **Çok büyük tabloları bölme ya da güvenlik önlemleri gibi sebepler ile uygulanabilir.**

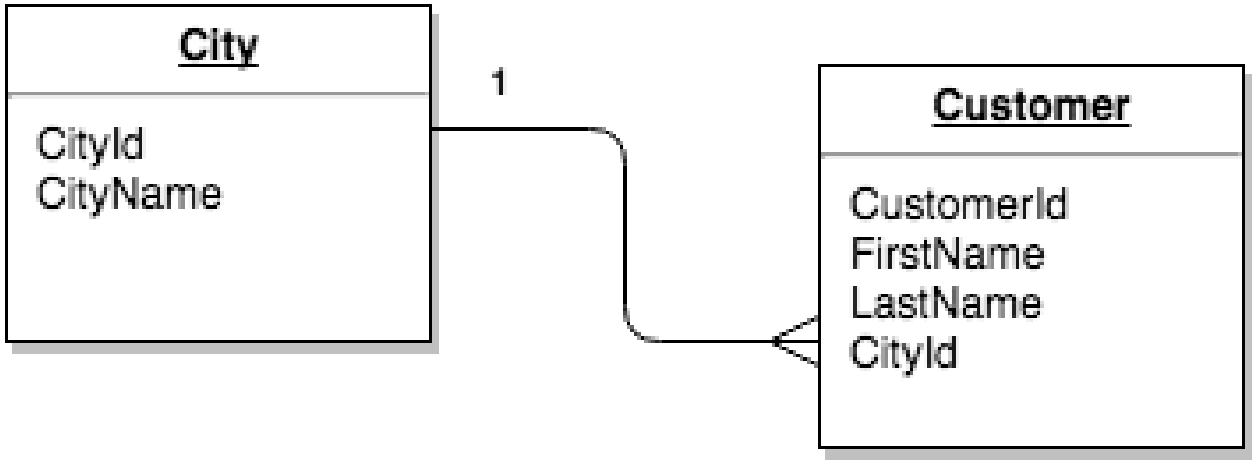
One-to-One

- **Bu örnekte HourlyRate değerini Employee tablosuna taşıyabilir ve böylelikle Pay tablosundan kurtulabilirdik. Ancak HourlyRate hassas bir bilgi içeriyor olabilir ve ayrı bir tabloda saklanarak Employee tablosuna erişimi olanların HourlyRate bilgisini görmeleri engellenebilir.**

One-to-Many

- **En yaygın veritabanı ilişkisidir. Bu ilişki durumunda A tablosundaki bir satırın B tablosunda birden fazla eşleşmesi olabilir. Ancak B tablosundaki bir satırın A tablosunda yalnızca bir eşleşmesi olabilir.**

One-to-Many

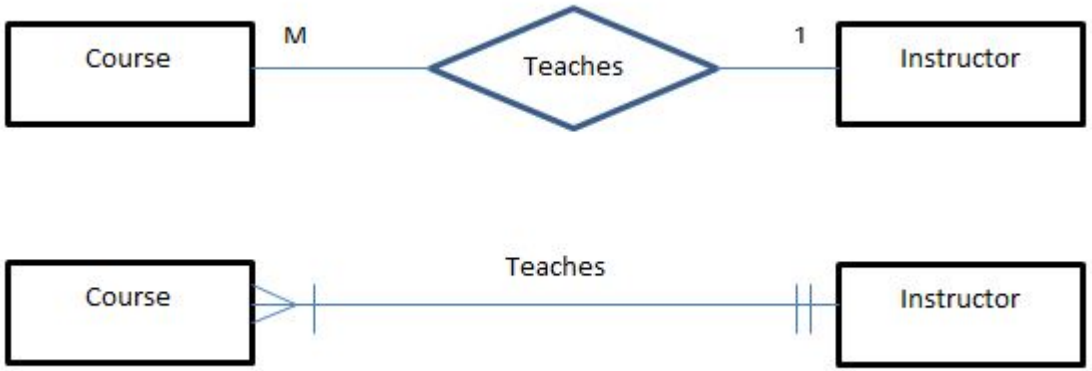


- **Bu örnekte Customer tablosu ‘many’,**

  City tablosu ‘one’ olan kısımdır.
- **Yani her müşterinin bir şehri olabilir fakat bir şehirde birden fazla müşteri olabilir.**

One-to-Many

- **Bu örnekte de bir öğretmen birden fazla ders anlatabilmektedir fakat bir dersi yalnızca bir öğretmen anlatabilmektedir. Bu yüzden veritabanı ilişkisi one-to-many'dir.**

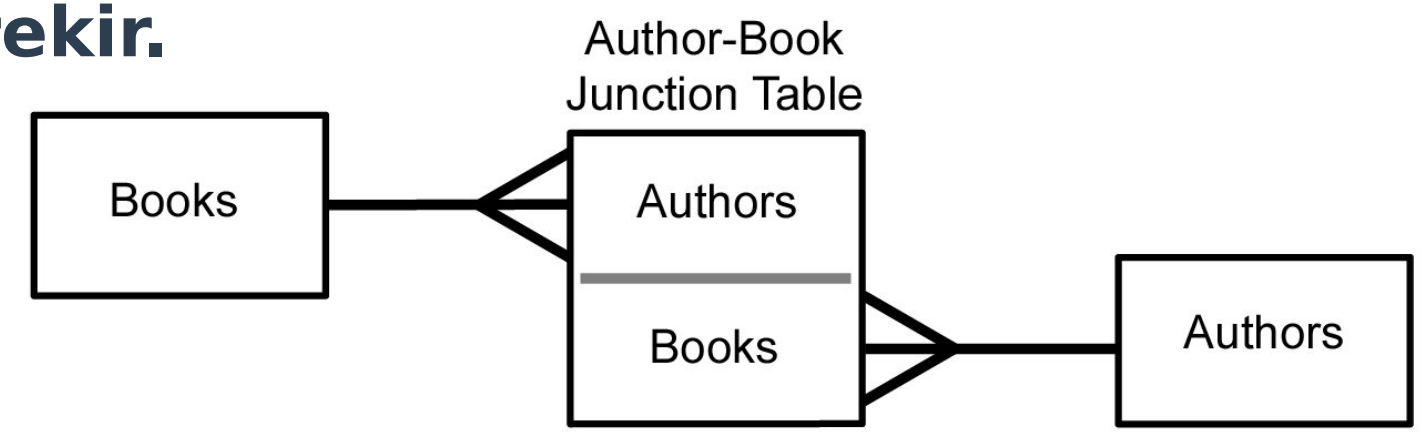


Many-to-Many

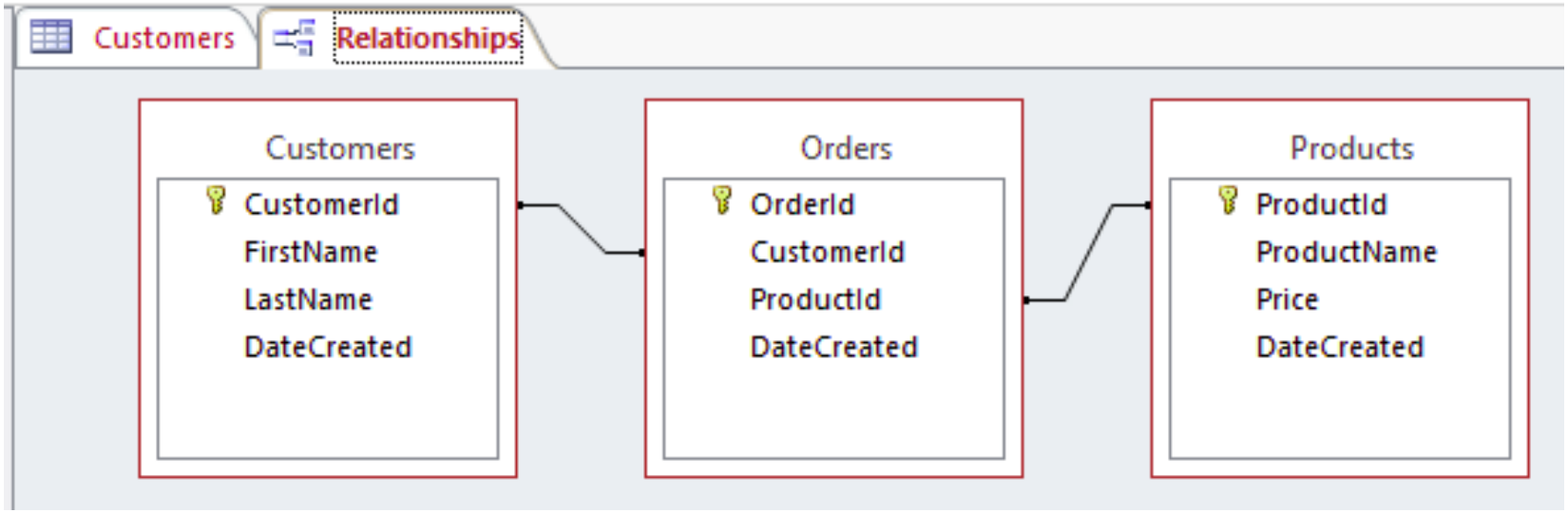
- **Bu tür veritabanı ilişkilerinde A tablosunda yer alan bir satırın B tablosunda birden fazla eşleşmesi olabilir.**
- **Aynı şekilde B tablosunda yer alan bir satırın da A tablosunda birden fazla eşleşmesi olabilir.**
- **Bu tür bir ilişki, veritabanı taslağında yeni bir tablo ile gösterilir. Bu tablolara “junction table” ya da “cross-reference table” adı verilir. Genellikle bu tabloların ismi birbiriyle many-to-many şeklinde bağlı olan iki tablonun isminin birlikte yazılmasıyla isimlendirilir.**

Many-to-Many

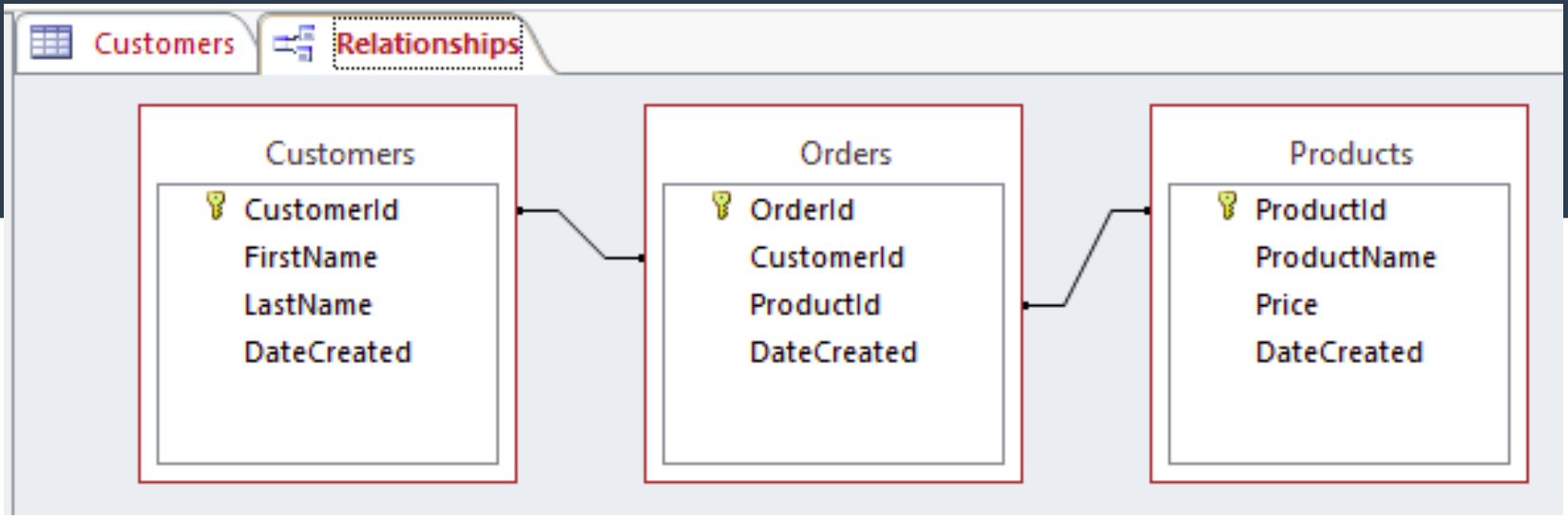
- **Örnekte kitaplar ve yazarlar isminde iki tablo yer almaktadır. Bir kitabın birden fazla yazarı olabilir, aynı şekilde bir yazarın birden fazla kitabı da olabilir. Bu yüzden bu iki tablo arasında many-to-many ilişkisi vardır.**
- **Bu ilişkiyi gösterebilmek için "author-book" isminde yeni bir bağlantı tablosu oluşturmak gerekir.**



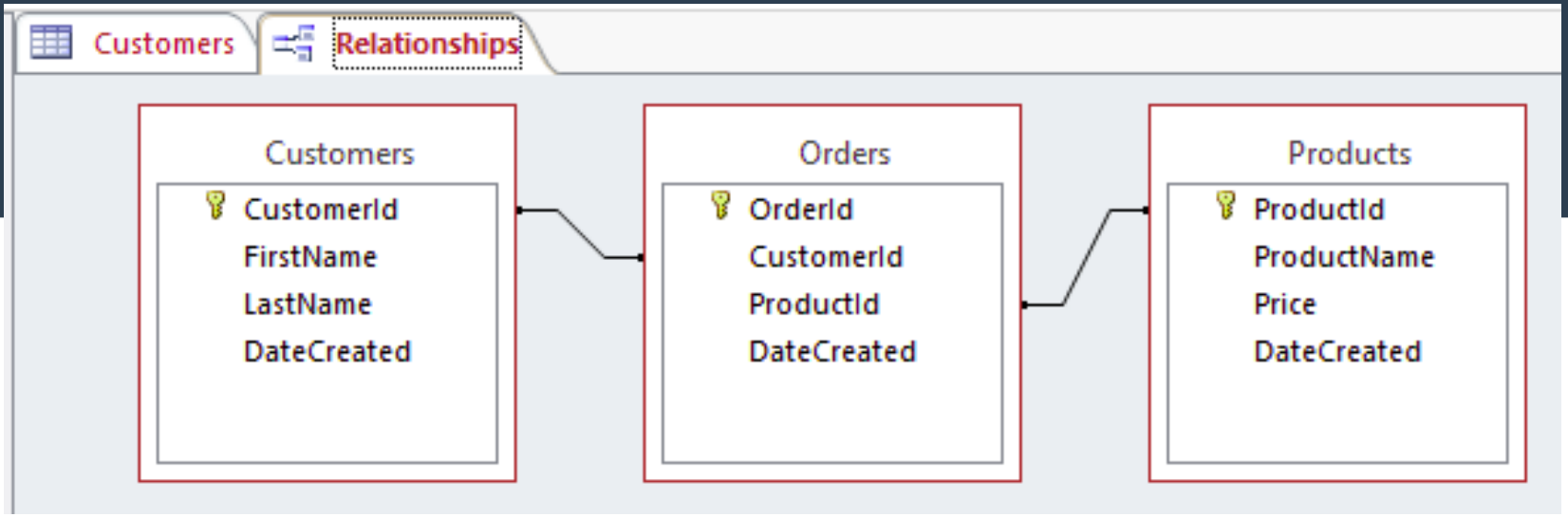
Many-to-Many



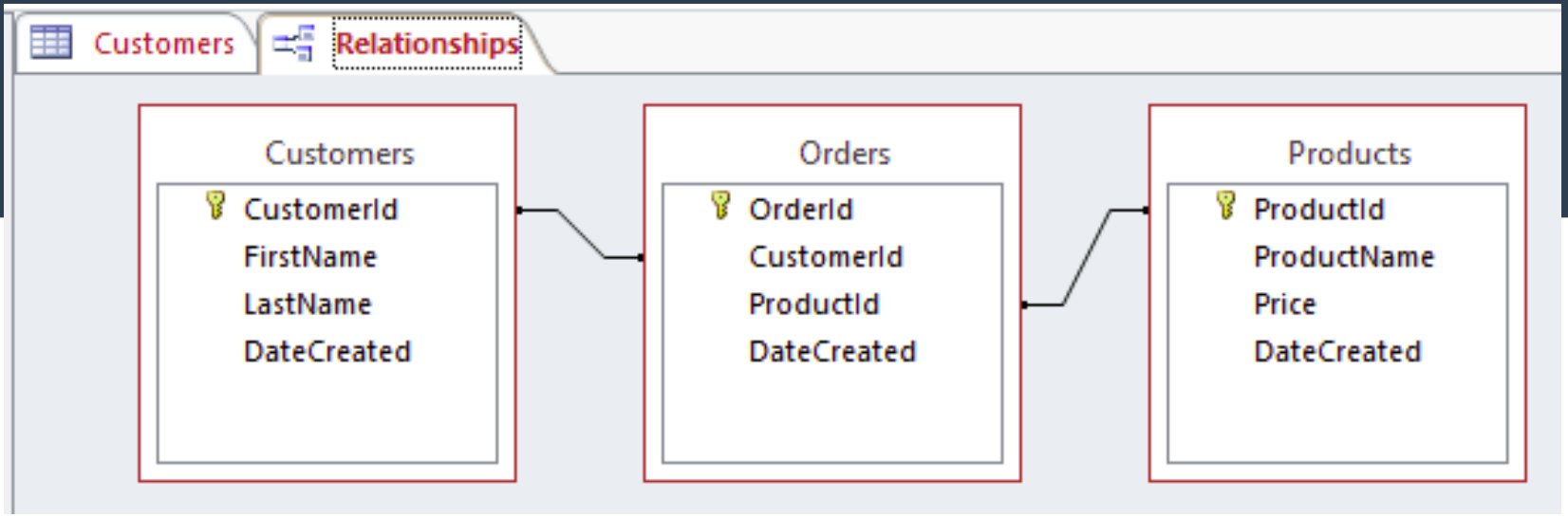
- **Bu örnekte de Customers (müşteriler) ve Products (ürünler) arasındaki many-to-many ilişkisini anlatabilmek için yeni bir Orders (siparişler) tablosu oluşturulmuştur.**



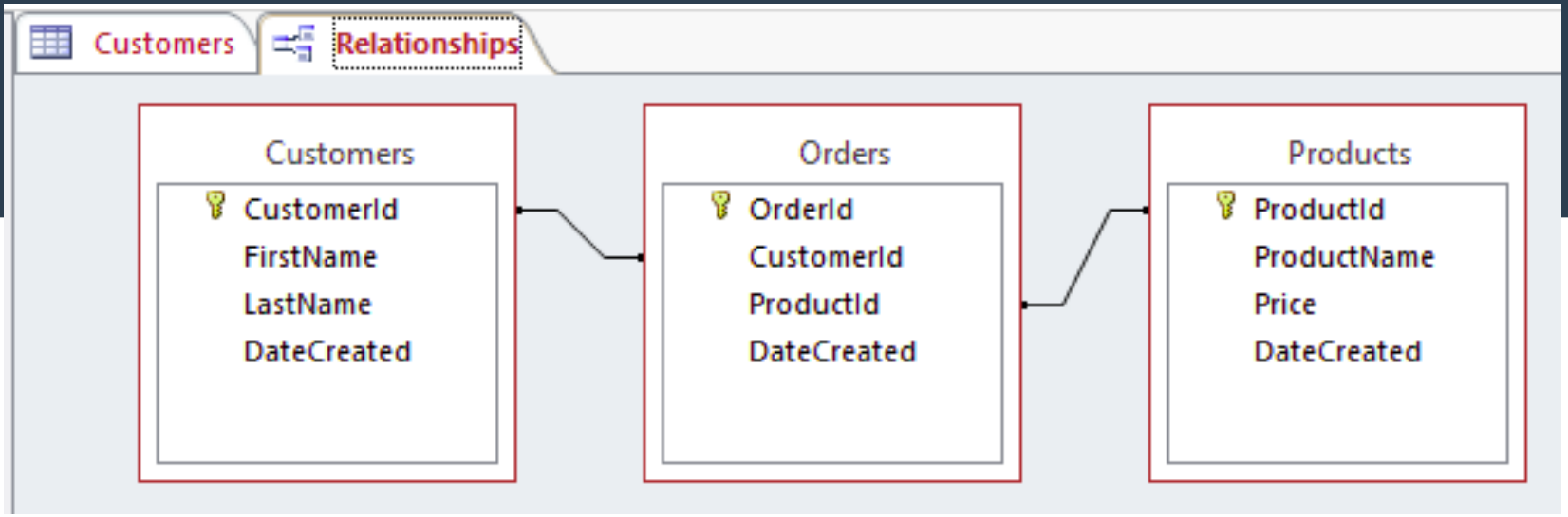
Customers tablosunda CustomerID, Products tablosunda da ProductID isimli bir alan mevcuttur.
Bu alanların içerdiği değerler Orders tablosunda ilgili alandaki değere karşılık gelmelidir.
Bu yüzden Orders.CustomerId'de verilen herhangi bir değer de Customer.CustomerId alanında bulunmalıdır.



Eğer bu koşullar gözetilmezse var olmayan müşteriler için sipariş almış olurduk. Aynı şekilde var olmayan ürünler için sipariş almış olabilirdik. Bu da "referential integrity" olarak adlandırılan tablolar arası ilişkilerde veri tutarlılığımızın kötü olduğu anlamına gelirdi.



Çoğu veritabanı sistemi, veritabanının referans bütünlüğünün korunmasına ilişkin destek sunar. Bu nedenle, bir kullanıcı (veya bir işlem) primary key alanında bulunmayan bir foreign key değerini eklemeye çalıştığında, hata alınır.



Bizim örneğimizde de Orders.CustomerID alanı Customers.CustomerID alanının foreign key'idir. (Customers.CustomerID bir primary key'dir)

Aynı şekilde Orders.ProductID değeri de Products.ProductID değerinin foreign key'idir. (Products.ProductID bir primary key'dir.)

ERD nedir?

- **ERD (Entity Relationship Diagram), bir uygulamanın/programın veritabanını inşa ederken veritabanındaki tabloların özelliklerinin, türlerinin vb. Belirlendiği ve tablolar arası ilişkilerin şekillendirildiği taslak çalışmasıdır. Veritabanları bir kez oluşturulduktan sonra tasarımlarında değişiklik yapmak zor olduğundan, temel vazifesi gören bu aşamanın dikkatlice yapılması sağlıklı bir uygulama yazabilmek için çok önemlidir.**
- **ERD örnekleri için tıklayın.**

Ekstra Okumalar

Veritabanları konusunda daha derin bilgi sahibi olmak isteyenler için okuma listesi:

- **Read-Write Conflicts**
- **ACID Properties**
- **Python ve MongoDB**
- **Full Stack Python - Database (+ General Database Resources kısmı)**