Veritabanı Yönetim Sistemleri
1906003022015

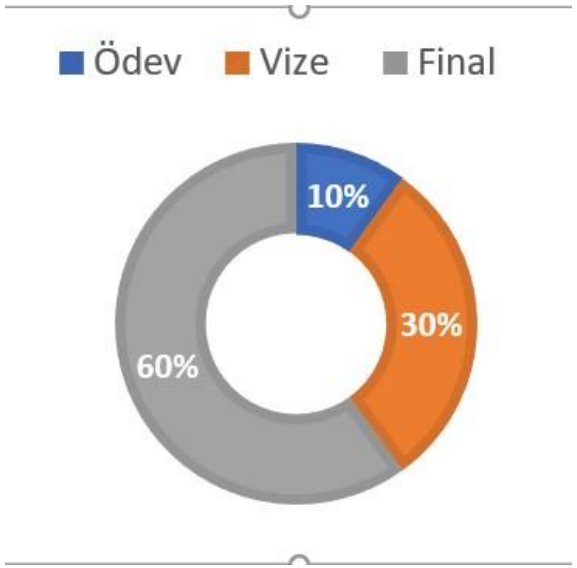
Dr. Öğr. Üy. Önder EYECİOĞLU
Bilgisayar Mühendisliği

Giriş

Ders Günü ve Saati:

Pazartesi: 09:15-12:45

- Devam zorunluluğu %70
- Uygulamalar MS SQL ve MongoDB üzerinde gerçekleştirilecektir.

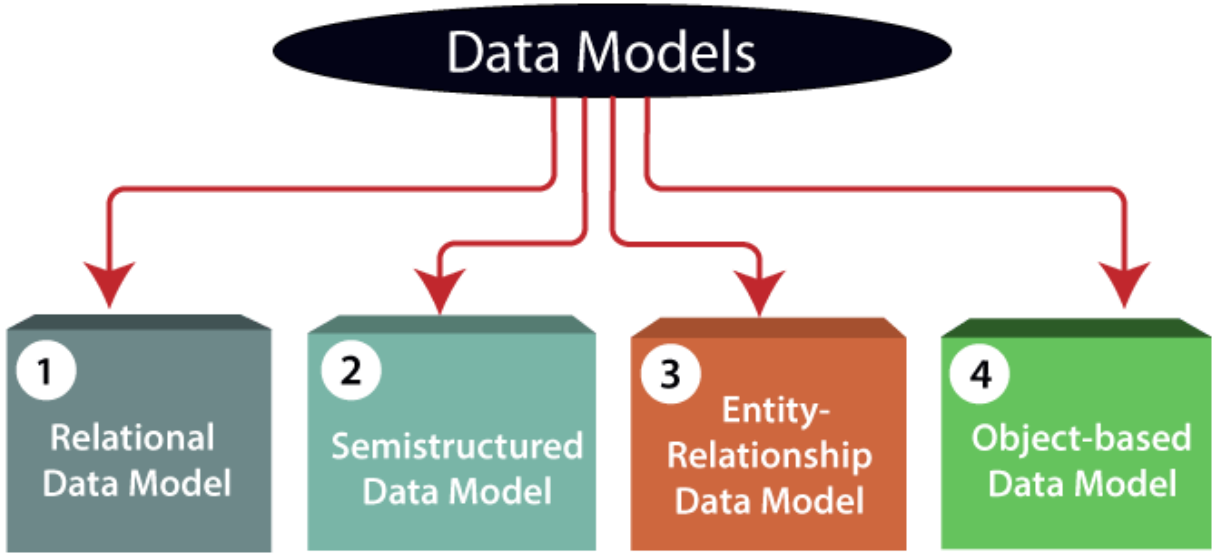


HAFTA	KONULAR
Hafta 1	VT ve VTYS'ne giriş
Hafta 2	ER Veri Modeli
Hafta 3	İlişkisel Modeller, İlişkisel model tasarımı
Hafta 4	İlişkisel Cebir ve Hesaplamalar
Hafta 5	İlişkisel Sorgular, SQL giriş
Hafta 6	SQL ile veri tabanı programlama
Hafta 7	SQL-Kısıtlar:Veri-tipi,birincil-anahtar,ikinci-anahtar,
Hafta 8	Vize
Hafta 9	İlişkisel Veri Tabanı Tasarımı ve Normalizasyon
Hafta 10	yarı-yapısal veri modelleri, XML
Hafta 11	JSON
Hafta 12	İlişkisel olmayan DB, NoSQL
Hafta 13	NoSQL
Hafta 14	DBMS -Eşzamanlılık (Concurrency) Kontrolü

ER Veri Modeli

DBMS Veritabanı Modelleri

Veri Modeli, veri tanımının, veri anlamının ve verilerin tutarlılık kısıtlamalarının modellenmesidir. Her veri soyutlama düzeyinde bir veritabanı tasarımını açıklamak için kavramsal araçlar sağlar. Bu nedenle, veritabanının yapısını anlamak için kullanılan aşağıdaki dört veri modeli vardır:

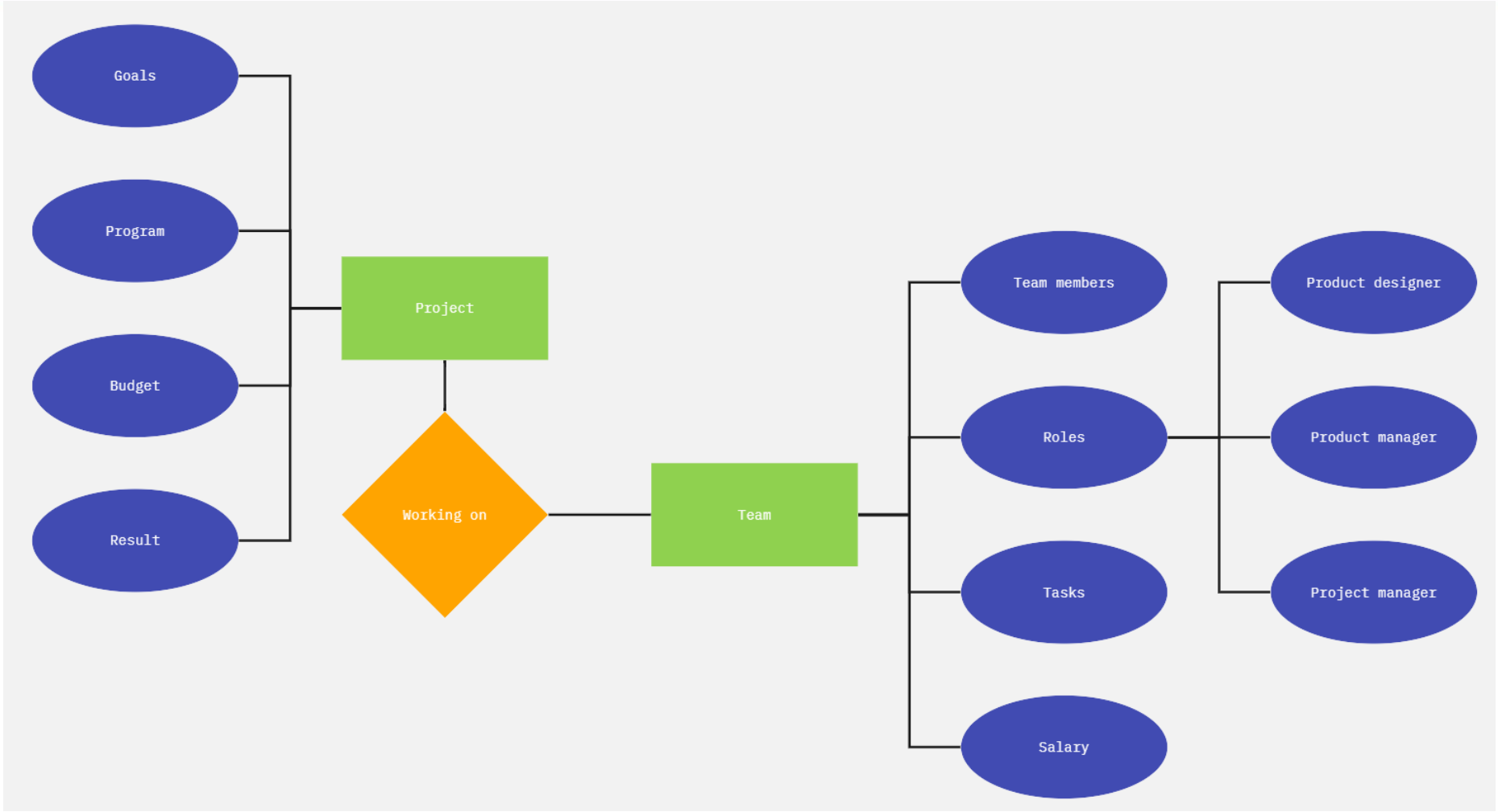


DBMS Veritabanı Modelleri

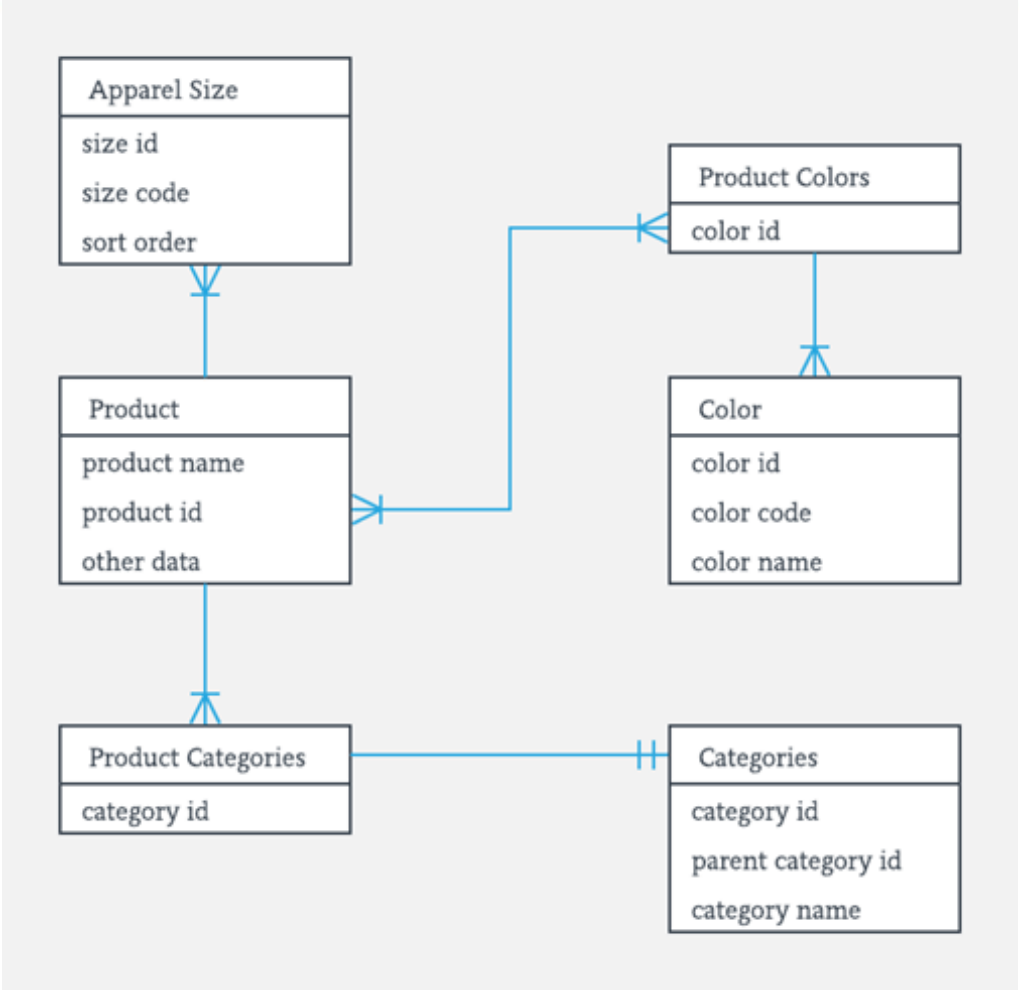
Veri modelleri, bir veritabanının mantıksal yapısının nasıl modellendiğini tanımlar. Veri Modelleri, bir DBMS'de soyutlamayı tanıtmak için temel varlıklardır. Veri modelleri, verilerin birbirine nasıl bağlandığını ve bunların sistem içinde nasıl işlenip depolanacağını tanımlar.

2) Varlık-İlişki Veri Modeli: Bir ER modeli, verilerin nesneler ve bunlar arasındaki ilişkiler olarak mantıksal temsilidir. Bu nesneler varlıklar olarak bilinir ve ilişki bu varlıklar arasındaki bir ilişkidir. Bu model Peter Chen tarafından tasarlandı ve 1976 makalelerinde yayınlandı. Veritabanı tasarımında yaygın olarak kullanılmıştır. Bir dizi nitelik varlıkları tanımlar. Örneğin, öğrenci_adı, öğrenci_kimliği 'öğrenci' varlığını tanımlar. Aynı tür varlıklardan oluşan bir dizi 'Varlık kümesi' olarak bilinir ve aynı türden ilişkiler kümesi 'ilişki kümesi' olarak bilinir.

Varlık-İlişki Veri Modeli



Varlık-ilişki Modeli



Varlık-ilişki Modeli

- ER Diyagramının amacı, varlık çerçeve altyapısını temsil etmektir.
- Varlık ilişkisi modellemeyle ilgili terimleri tanımlamanıza yardımcı olur
- Tüm tablolarınızın nasıl bağlanması gerektiğine, her tabloda hangi alanların olacağına dair bir önizleme sağlayın
- Varlıkları, nitelikleri, ilişkileri tanımlamaya yardımcı olur
- ER diyagramları, hızlı bir şekilde veritabanları oluşturmanıza olanak tanıyan ilişkisel tablolara çevrilebilir

ER Tasarım Sorunları

Veri modellemenin önceki bölümlerinde, bir ER diyagramı tasarlamayı öğrendik. Ayrıca, varlık kümelerini ve aralarındaki ilişkileri tanımlamanın farklı yollarını tartıştık. Bir ilişkiyi, bir varlığı ve onun niteliklerini temsil eden çeşitli tasarım şekillerini de anladık. Bununla birlikte, kullanıcılar genellikle ER diyagramının elemanlarının konseptini ve tasarım sürecini yanlış yönlendirirler. Böylece, ER diyagramının karmaşık bir yapısına ve gerçek dünya işletme modelinin özelliklerini karşılamayan bazı sorunlara yol açar.

ER Tasarım Sorunları

- 1) Varlık Kümesi ve Özniteliklerin Kullanımı

- Bir varlık kümesinin niteliklerini tanımlarken, bazen bir özelliğin bir nitelik olarak mı yoksa bir varlık kümesi olarak mı modellenmesi gerektiği (ve bir ilişki kümesi kullanılarak ilk varlık kümesiyle ilişkilendirilmesi) açık değildir. Örneğin, Çalışanlar varlık kümesine adres bilgisi eklemeyi düşünün. Bir seçenek, bir adres özniteliği kullanmaktır.

- Bu seçenek, çalışan başına yalnızca bir adres kaydetmemiz gerekiyorsa uygundur ve bir adresi bir dize olarak düşünmek yeterlidir. Bir alternatif, Adresler adlı bir varlık seti oluşturmak ve bir ilişki (örneğin, Has_Address) kullanarak çalışanlar ve adresler arasındaki ilişkileri kaydetmektir.

ER Tasarım Sorunları

- 1) Varlık Kümesi ve Özniteliklerin Kullanımı

- Bir varlık kümesinin veya özniteliğinin kullanımı, modellenmekte olan gerçek dünya kuruluşunun yapısına ve öznitelikleriyle ilişkili anlambilime bağlıdır. Kullanıcı, bir varlık kümesinin birincil anahtarını başka bir varlık kümesinin özniteliği olarak kullandığında bir hataya yol açar. Bunun yerine, bunu yapmak için ilişkiyi kullanmalıdır. Ayrıca, birincil anahtar nitelikler ilişki setinde örtüktür, ancak biz onu ilişki setlerinde belirleriz.

ER Tasarım Sorunları

- 2) Varlık Kümesi ile İlişki Kümelerinin Kullanımı

- Bir nesnenin bir varlık kümesi veya ilişki kümesi tarafından en iyi şekilde ifade edilip edilemeyeceğini incelemek zordur. Doğru kullanımı anlamak ve belirlemek için, kullanıcının varlıklar arasında meydana gelen bir eylemi açıklamak için bir ilişki seti belirlemesi gerekir. Nesneyi bir ilişki kümesi olarak temsil etme gerekliliği varsa, onu varlık kümesiyle karıştırmamak daha iyidir.

ER Tasarım Sorunları

- 2) Varlık Kümesi ile İlişki Kümelerinin Kullanımı
- ER modellemenin kesin olmayan doğası, bu nedenle, altta yatan varlıkları tanımayı zorlaştırabilir ve nitelikleri uygun varlıklar yerine ilişkilerle ilişkilendirebiliriz. Genel olarak, bu tür hatalar aynı bilgilerin gereksiz olarak depolanmasına yol açar ve birçok soruna neden olabilir.
- tablolardan fazlalıkları ortadan kaldırmak için normalleştirme adı verilen bir teknik tartışılacak.

ER Tasarım Sorunları

- 3) İkili ve n-ary İlişki Kümelerinin Kullanımı

- Genel olarak, veritabanlarında açıklanan ilişkiler ikili ilişkilerdir. Bununla birlikte, ikili olmayan ilişkiler birkaç ikili ilişki ile temsil edilebilir. Örneğin, bir çocuk, babası ve annesi ile ilgili olabilecek üçlü bir ilişki 'ebeveyn' oluşturabilir ve temsil edebiliriz. Bu tür bir ilişki aynı zamanda iki ikili ilişki ile de temsil edilebilir, yani anne ve baba, çocukları ile ilgili olabilir. Bu nedenle, ikili olmayan bir ilişkiyi bir dizi farklı ikili ilişki ile temsil etmek mümkündür..

ER Tasarım Sorunları

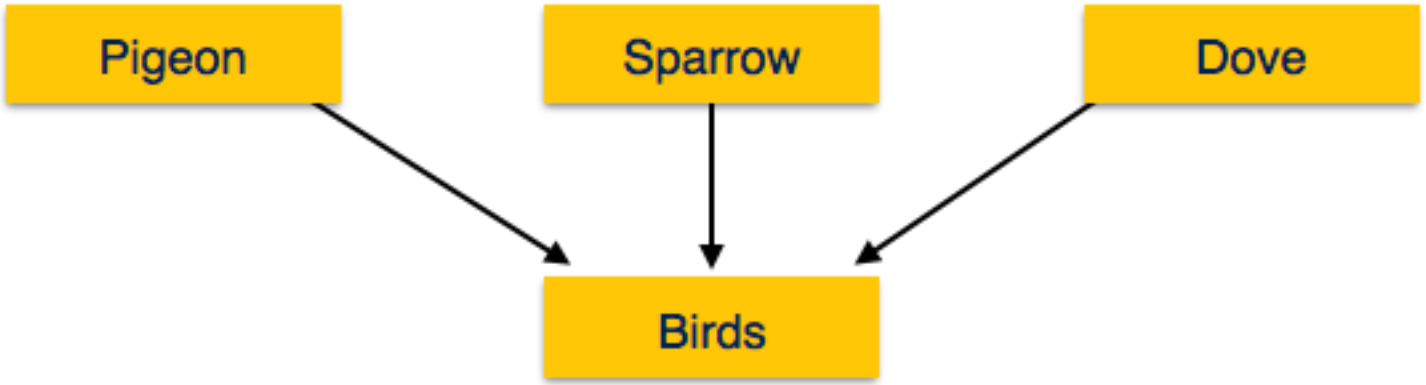
- 4) İlişki Niteliklerinin Yerleştirilmesi
- Kardinalite oranları, ilişki özelliklerinin yerleştirilmesinde etkili bir ölçü olabilir. Bu nedenle, bire bir veya bire çok ilişki kümelerinin özniteliklerini herhangi bir ilişki kümesi yerine herhangi bir katılımcı varlık kümesiyle ilişkilendirmek daha iyidir. Belirtilen özniteliği bir ilişki veya varlık özniteliği olarak yerleştirme kararı, modellenen gerçek dünya girişiminin özelliklerine sahip olmalıdır.

Genelleştirme

- ER Modeli, veritabanı varlıklarını kavramsal hiyerarşik bir şekilde ifade etme gücüne sahiptir. Hiyerarşi yükseldikçe, varlıkların bakış açısını genelleştirir ve hiyerarşinin derinliklerine doğru ilerlerken, bize dahil edilen her varlığın ayrıntılarını verir.
- Bu yapıda yukarı çıkmak , varlıkların daha genel bir görüşü temsil etmek için bir araya getirildiği genelleme olarak adlandırılır . Örneğin, Mira adlı belirli bir öğrenci tüm öğrencilerle birlikte genelleştirilebilir. Varlık öğrenci olmalı ve ayrıca öğrenci bir kişidir. Bunun tersine, bir kişinin öğrenci olduğu ve bu öğrencinin Mira olduğu uzmanlaşma denir.

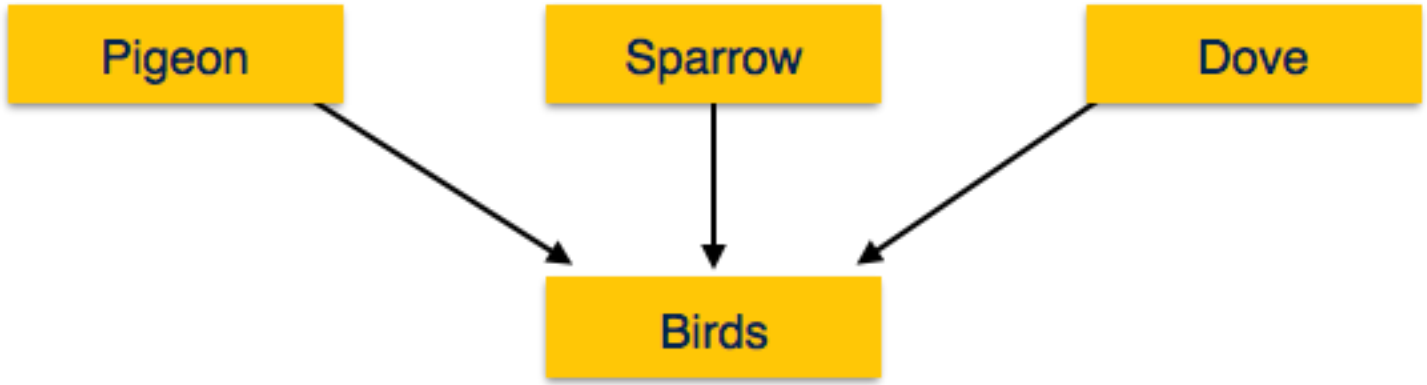
Genelleme

- Genelleştirilmiş varlıkların tüm genelleştirilmiş varlıkların özelliklerini içerdiği varlıkları genelleştirme sürecine genelleme denir. Genel olarak, birkaç varlık benzer özelliklerine göre tek bir genelleştirilmiş varlık olarak bir araya getirilir.



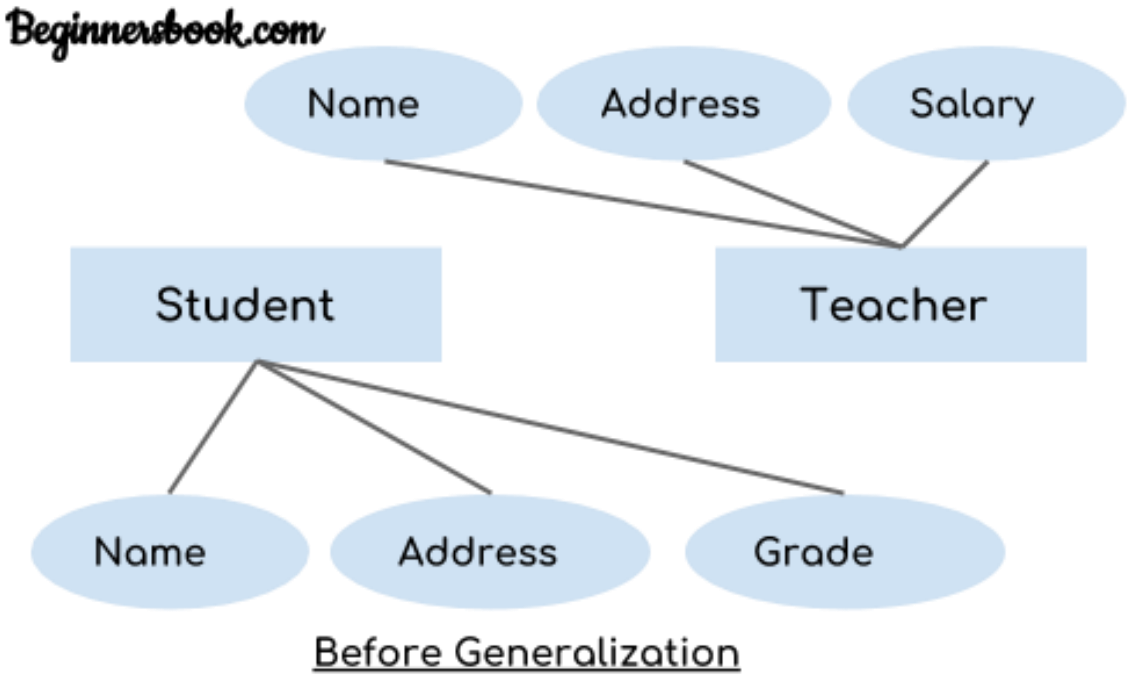
Genelleme

- 1. Genelleme, iki veya daha fazla alt seviyeli varlığın daha yüksek seviyede yeni bir varlık oluşturmak için bir araya geldiği aşağıdan yukarıya yaklaşımı kullanır.
- 2. Yeni genelleştirilmiş varlık, daha yüksek düzeyde genelleştirilmiş bir varlık oluşturmak için daha düşük seviyeli varlık ile birlikte birleşebilir.



Genelleme

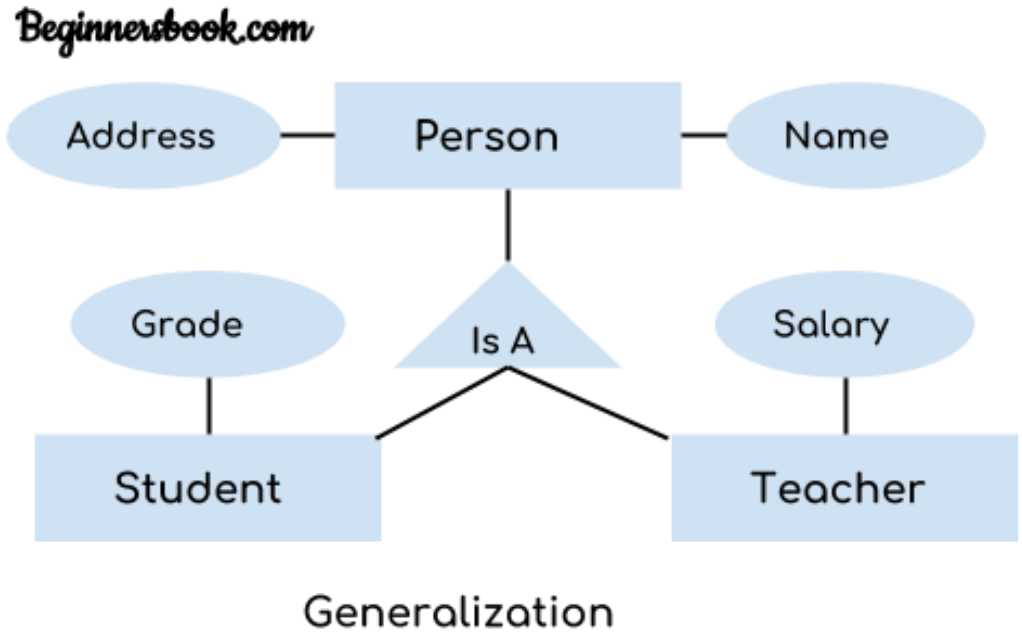
Genelleştirmeden önceki ER diyagramı şuna benzer:



Bu iki varlığın iki ortak özelliği vardır: Ad ve Adres, bu ortak özelliklerle genelleştirilmiş bir varlık yapabiliriz.

Genelleme

Genellemeden sonraki ER diyagramı:

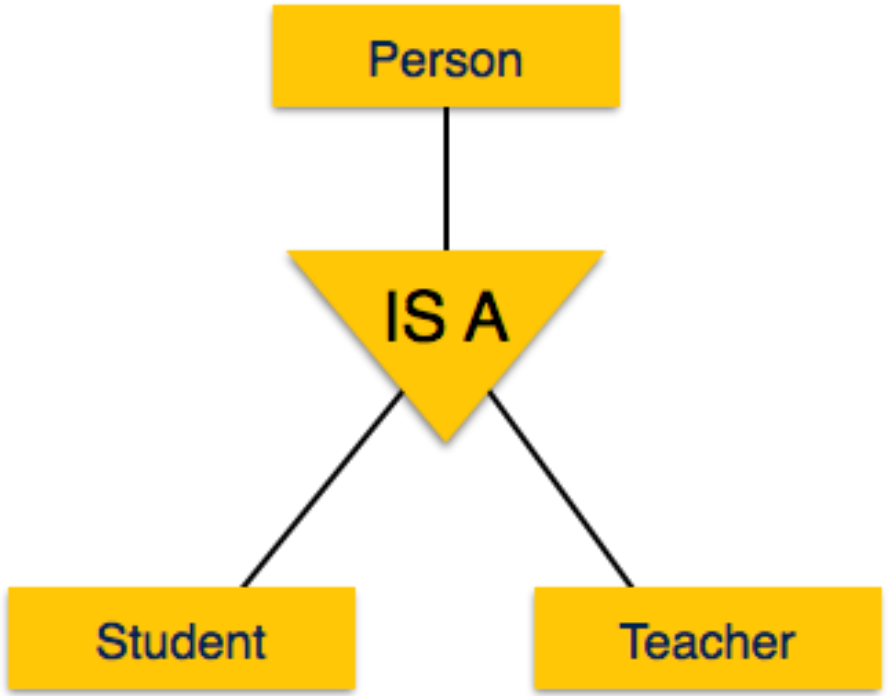


Generalization

Bu iki varlığın iki ortak özelliği vardır: Ad ve Adres, bu ortak özelliklerle genelleştirilmiş bir varlık yapabiliriz.

Özelleşme/ Uzmanlaşma

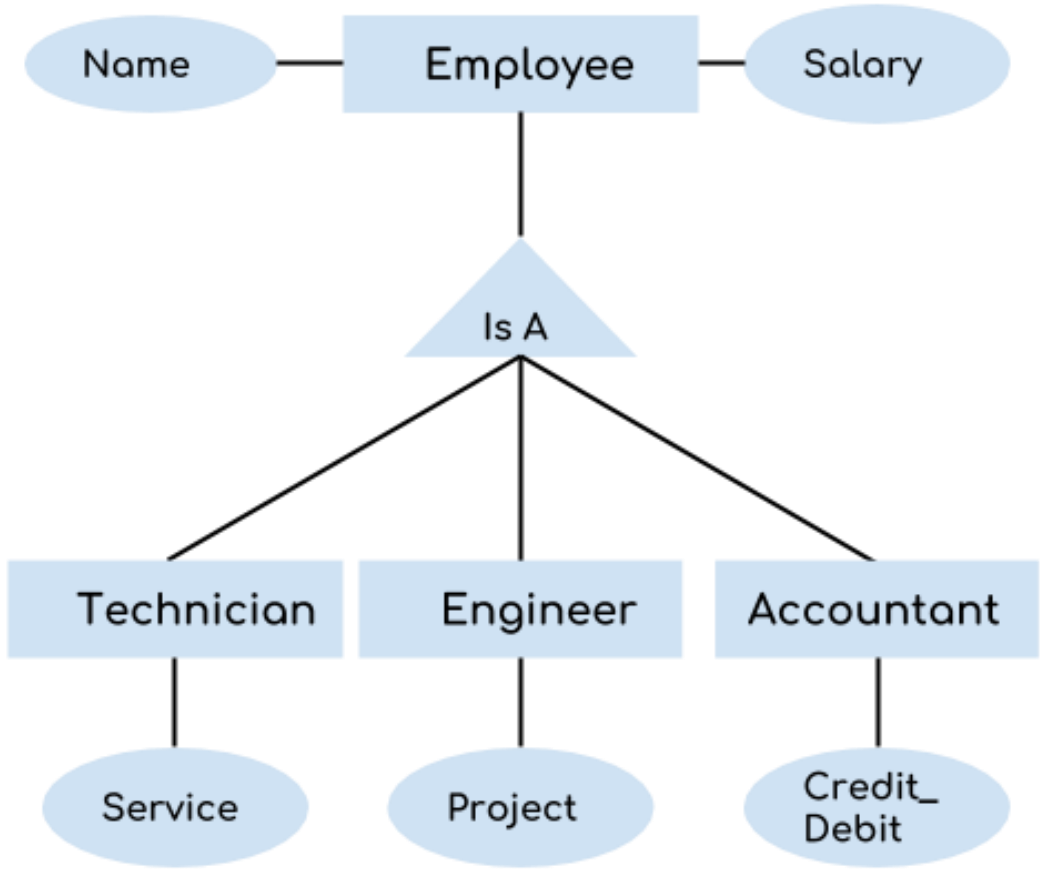
- Uzmanlık, genellemenin tam tersidir. Uzmanlaşmada, bir grup varlık, özelliklerine göre alt gruplara ayrılır. Örneğin bir 'Kişi' grubunu ele alalım. Bir kişinin adı, doğum tarihi, cinsiyeti vb. Vardır. Bu özellikler tüm insanlarda ortaktır. Ancak bir şirkette kişiler, şirkette oynadıkları role bağlı olarak çalışan, işveren, müşteri veya satıcı olarak tanımlanabilir.



Özelleşme/ Uzmanlaşma

- Uzmanlaşma , bir varlığın alt varlıklara bölündüğü bir süreçtir. Bunu ters bir genelleme süreci olarak düşünebilirsiniz, genellemede iki varlık yeni bir üst düzey varlık oluşturmak için bir araya gelir. Uzmanlaşma, yukarıdan aşağıya bir süreçtir.
- Uzmanlığın arkasındaki fikir, birkaç ayırt edici özelliğe sahip varlıkların alt kümelerini bulmaktır. Örneğin - Alt kuruluşlar Teknisyen, Mühendis ve Muhasebeci olarak sınıflandırılabilecek bir kuruluş çalışanı düşünün çünkü bu alt kuruluşların bazı ayırt edici özellikleri vardır.

Özelleşme/ Uzmanlaşma



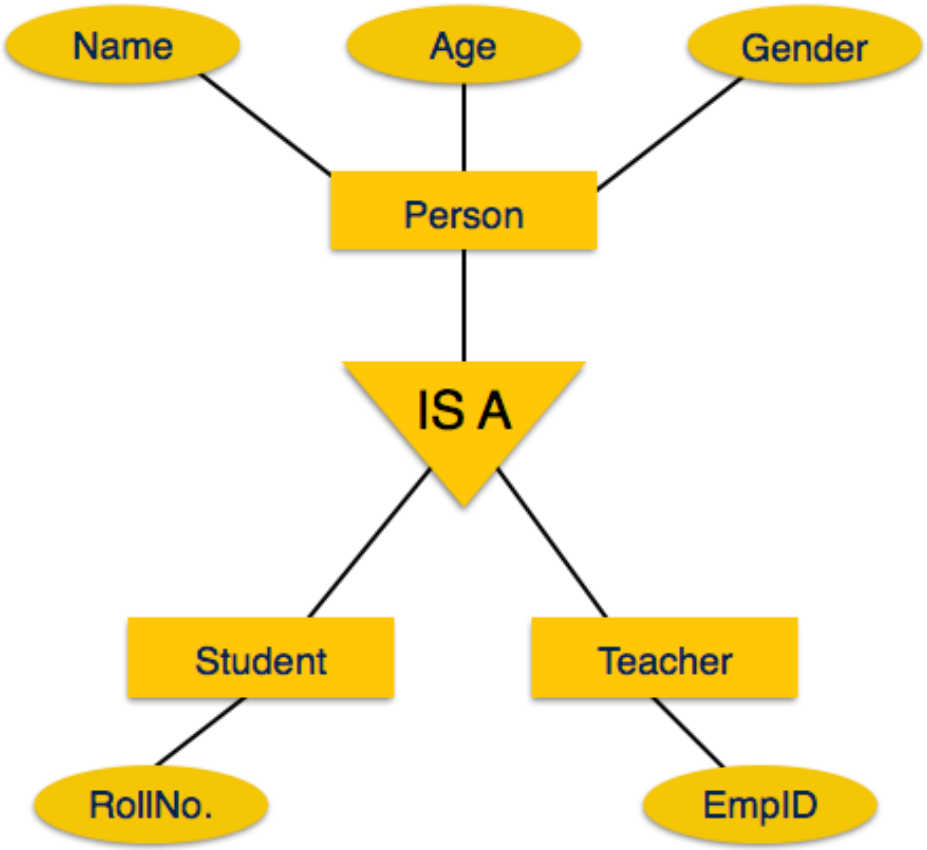
Specialization

Özelleşme/ Uzmanlaşma

- Yukarıdaki diyagramda, "Teknisyen", "Mühendis" ve "Muhasebeci" alt varlıklarına ayırdığımız, daha yüksek düzeyde bir "Çalışan" varlığımız olduğunu görebiliriz. Bunların hepsi sadece bir şirketin çalışanıdır, ancak rolleri tamamen farklıdır ve birkaç farklı özelliği vardır. Sadece örnek için, Teknisyenin servis taleplerini, Mühendisin bir proje üzerinde çalıştığını ve Muhasebecinin kredi ve borç ayrıntılarını ele aldığını gösterdim. Bu üç çalışan türünün tümü, yukarıdaki şemada gösterildiği gibi ana varlık "Çalışan" ile ilişkili bıraktığımız ad ve maaş gibi birkaç ortak özelliğe sahiptir..

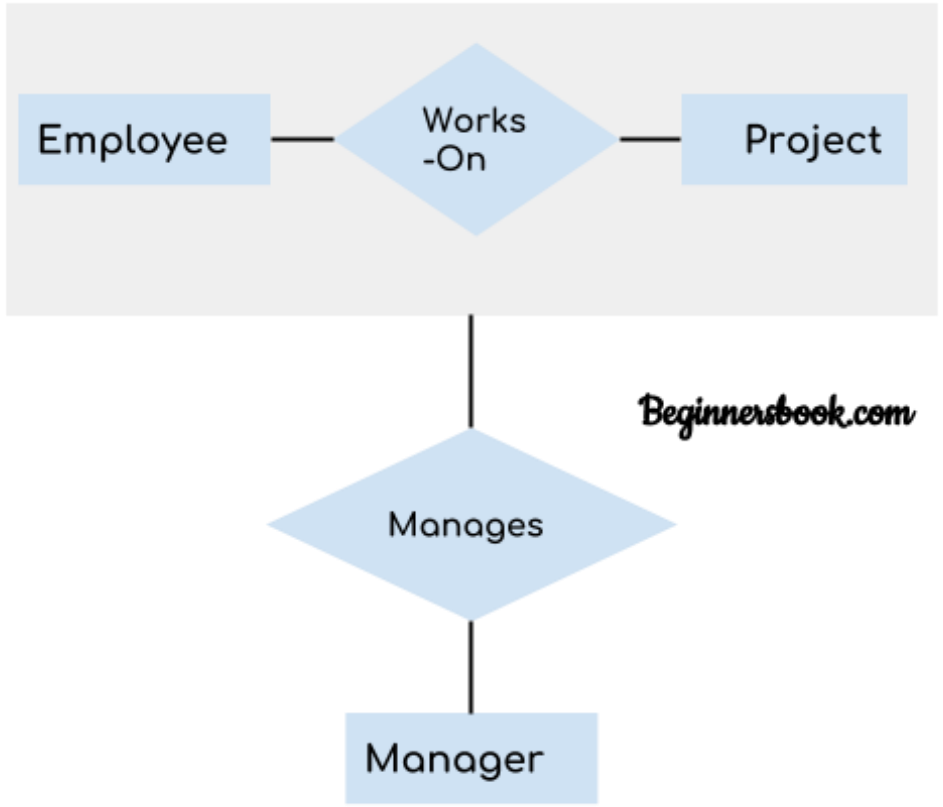
Miras

- Nesne yönelimli programlamada nesne sınıfları oluşturmak için ER-Model'in yukarıdaki tüm özelliklerini kullanıyoruz. Varlıkların ayrıntıları genellikle kullanıcıdan gizlenir; bu süreç soyutlama olarak bilinir .
- Kalıtım, Genelleme ve Uzmanlaşmanın önemli bir özelliğidir. Daha düşük seviyeli varlıkların daha yüksek seviyeli varlıkların niteliklerini devralmasına izin verir.



DBMS Aggregration/Toplama

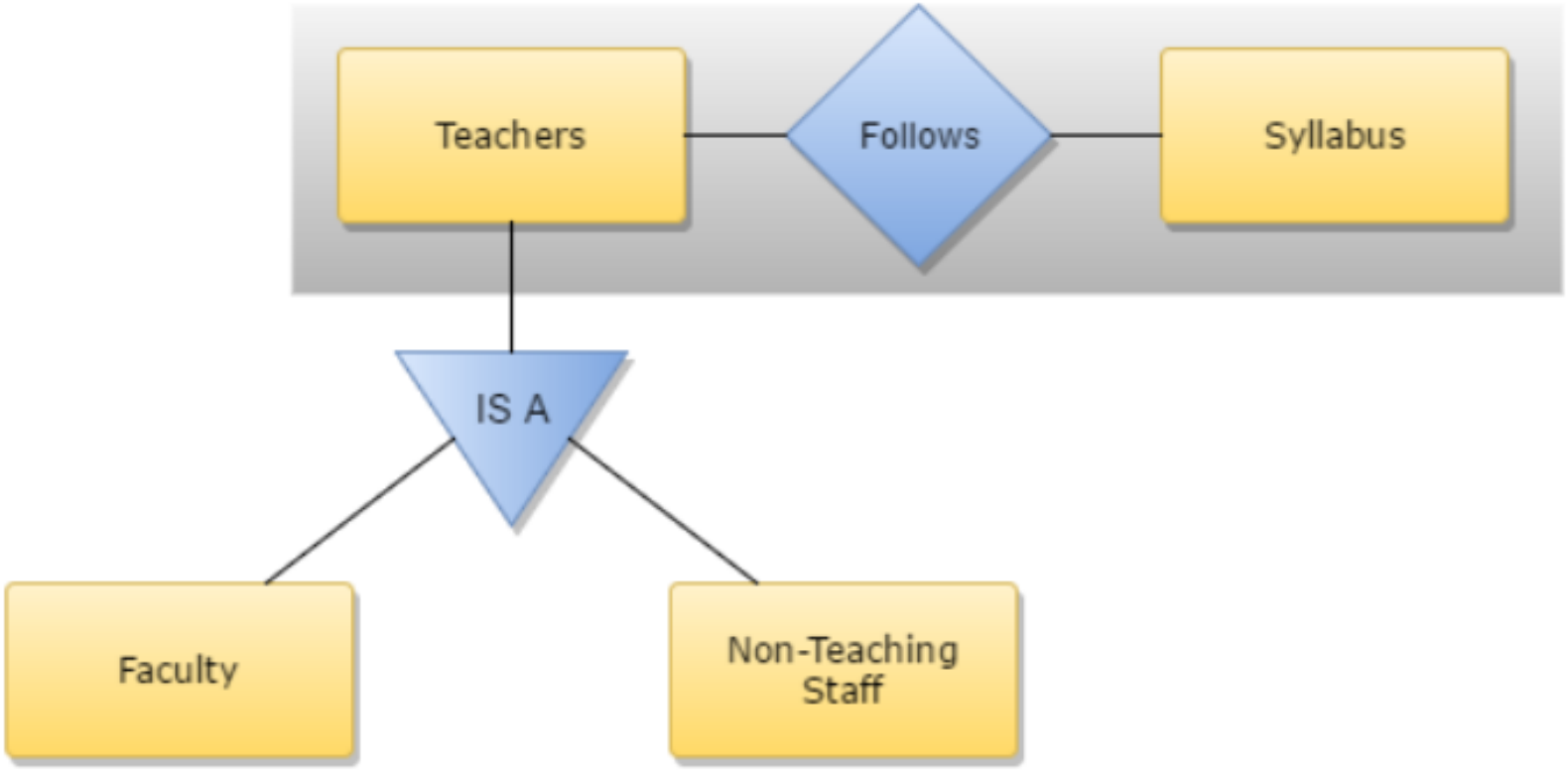
- Toplamada, iki varlık arasındaki ilişki tek bir varlık olarak kabul edilir. Toplamada, karşılık gelen varlıklar ile ilişki daha yüksek seviyeli bir varlık olarak toplanır.
- Toplama , tek bir varlığın bir ilişkide tek başına bir anlam ifade edemediği, dolayısıyla iki varlığın ilişkisinin tek bir varlık olarak hareket ettiği bir süreçtir.



DBMS Aggregration/Toplama

- Bir ilişki, kavramsal olarak aynı seviyede olan iki varlık türü arasındaki bir bağlantıyı temsil eder. Bazen, bir varlığın daha küçük varlıklardan ('bütün') oluşacak daha büyük bir varlığı ('bütün') temsil ettiği 'has-a', 'is-a' veya 'is-part-of' ilişkisini modellemek isteyebilirsiniz. 'parçalar'). Bu özel ilişki türü bir toplama olarak adlandırılır. Toplama, bütün ve parçaları arasındaki ilişki boyunca gezinmenin ve yönlendirmenin anlamını değiştirmez.

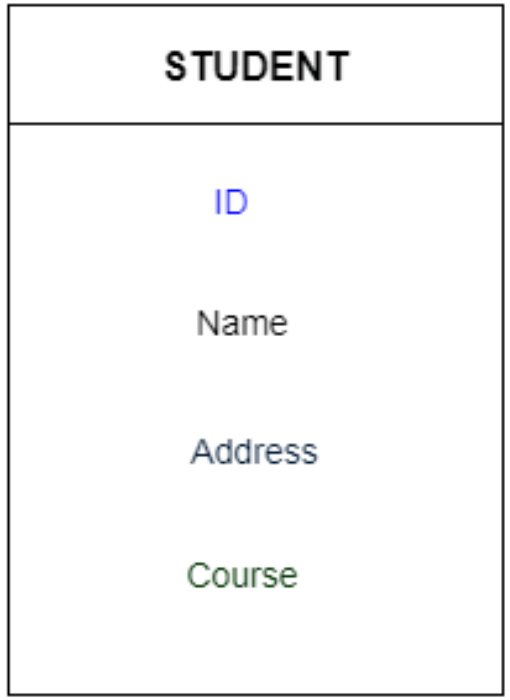
DBMS Aggregration/Toplama

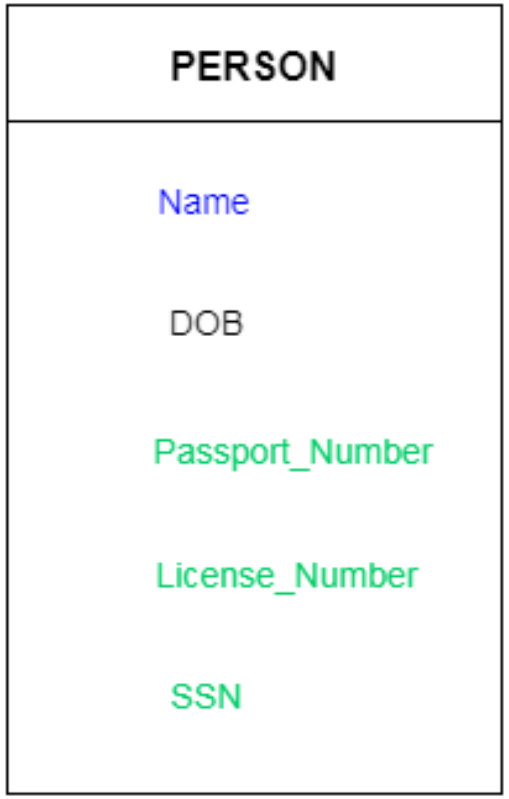


Specialization and Aggregation

DBMS Anahtarları

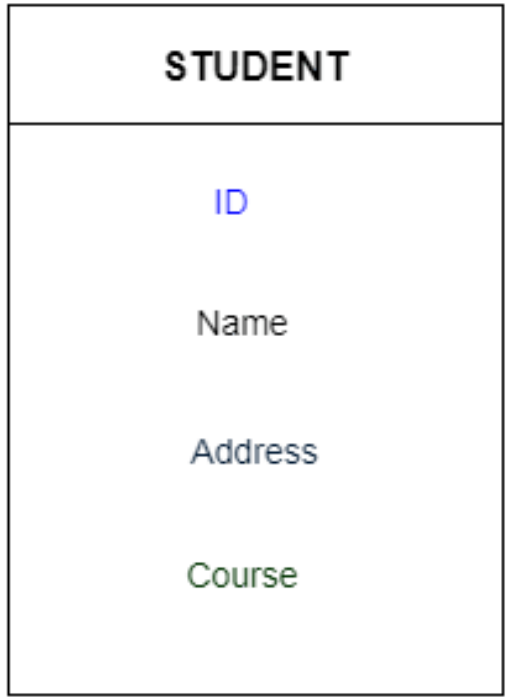
- Anahtarlar, ilişkisel veritabanında önemli bir rol oynar.
- Tablodaki herhangi bir kaydı veya veri satırını benzersiz şekilde tanımlamak için kullanılır. Ayrıca tablolar arasındaki ilişkileri kurmak ve tanımlamak için de kullanılır.

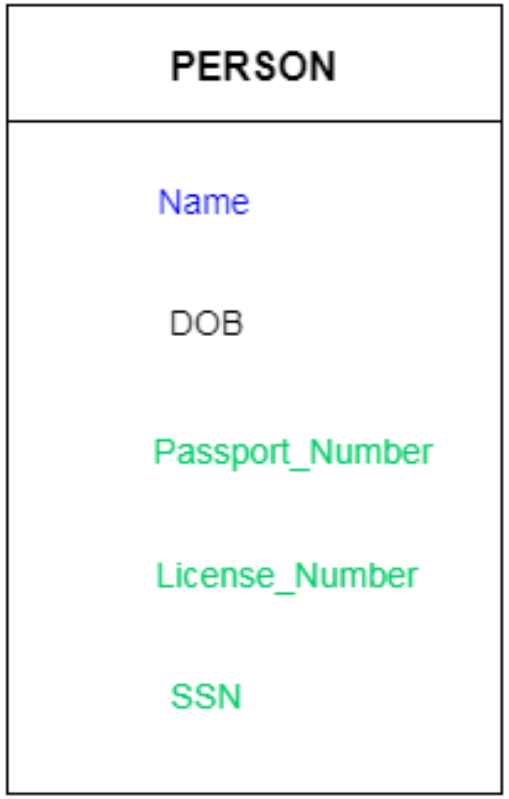




DBMS Anahtarları

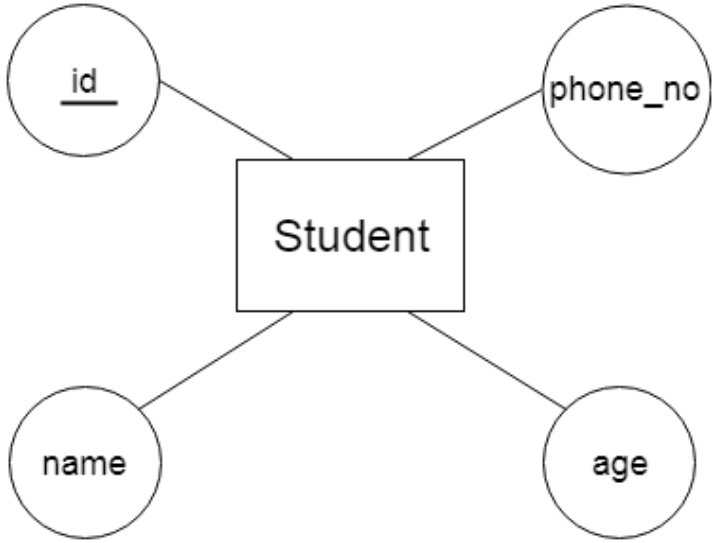
- Anahtarlar, ilişkisel veritabanında önemli bir rol oynar.
- Tablodaki herhangi bir kaydı veya veri satırını benzersiz şekilde tanımlamak için kullanılır. Ayrıca tablolar arasındaki ilişkileri kurmak ve tanımlamak için de kullanılır.



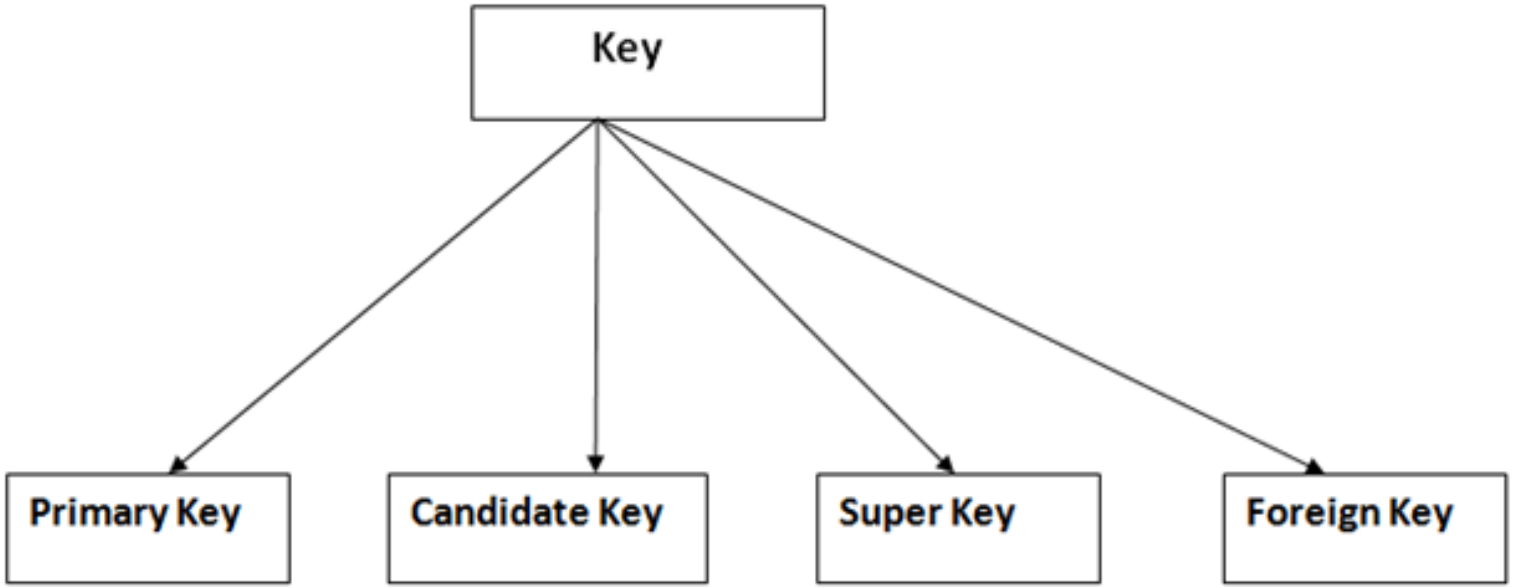


Öznitellikler

Anahtar Öznitelik - Anahtar öznitelik, bir varlığın temel özelliklerini temsil etmek için kullanılır. Bir birincil anahtarı temsil eder. Anahtar özellik, metni altı çizili olarak bir elips ile temsil edilir..



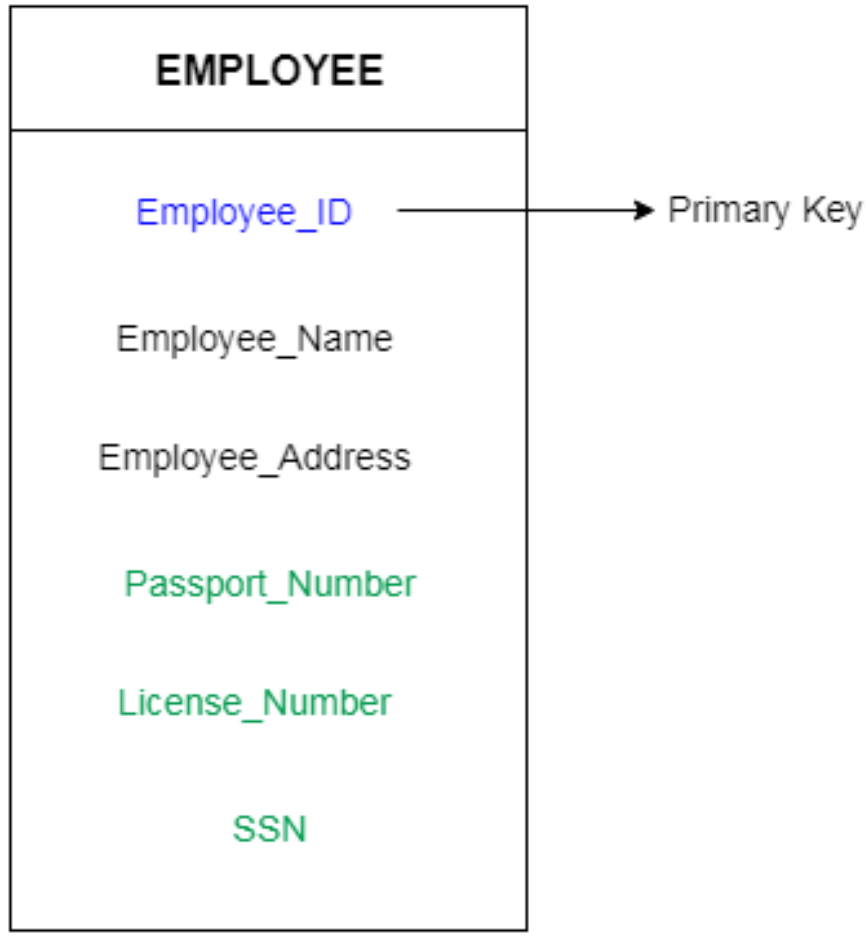
Anahtar türleri:



Anahtar türleri:

1. Birincil anahtar

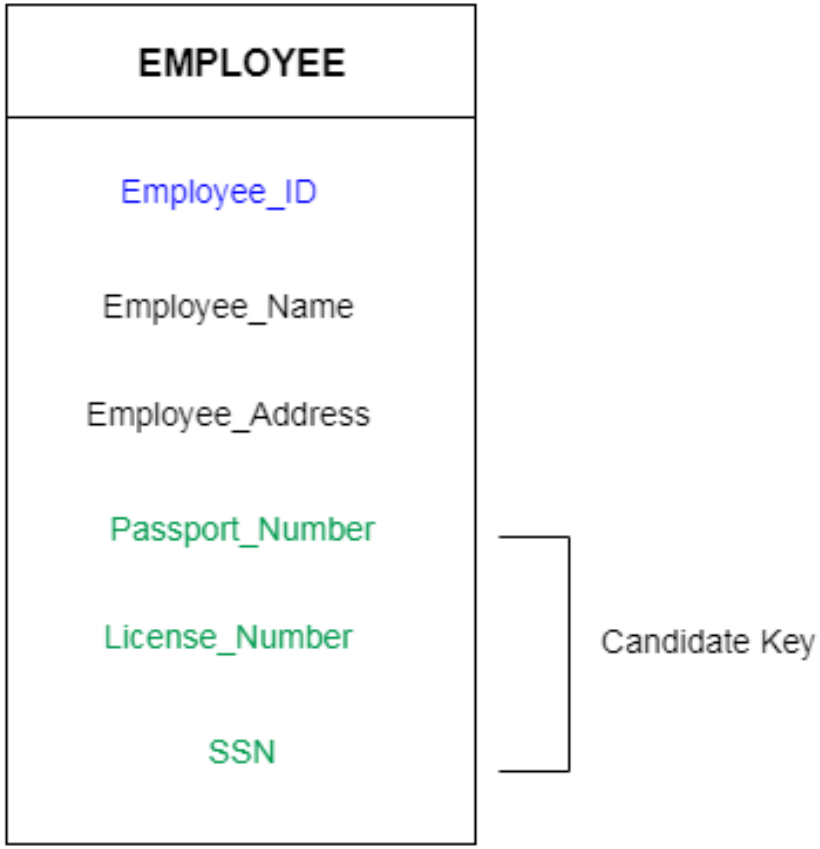
- Bir varlığın bir ve yalnızca bir örneğini benzersiz olarak tanımlamak için kullanılan ilk anahtardır. KİŞİSEL tablosunda gördüğümüz gibi bir varlık birden fazla anahtar içerebilir. Bu listelerden en uygun olan anahtar birincil anahtar olur.
- Her varlık için birincil anahtarın seçimi, gereksinime ve geliştiricilere bağlıdır.



Anahtar türleri:

2. Aday anahtarı

- Bir aday anahtar, bir demeti benzersiz bir şekilde tanımlayabilen bir öznitelik veya bir öznitelik kümesidir.
- Birincil anahtar dışında kalan öznitelikler bir aday anahtar olarak kabul edilir. Aday anahtarlar, birincil anahtar kadar güçlüdür.



Anahtar türleri:

3. Süper Anahtar

- Süper anahtar, bir demeti benzersiz şekilde tanımlayabilen bir öznitelik kümesidir. Süper anahtar, aday anahtarın üst kümesidir.
- Örneğin: Yukarıdaki EMPLOYEE tablosunda (EMPLOEE_ID, EMPLOYEE_NAME) için iki çalışanın adı aynı olabilir, ancak EMPLYEE_ID aynı olamaz. Bu nedenle, bu kombinasyon aynı zamanda bir anahtar olabilir.
- Süper anahtar EMPLOYEE-ID, (EMPLOYEE_ID, EMPLOYEE-NAME) vb. Olacaktır.

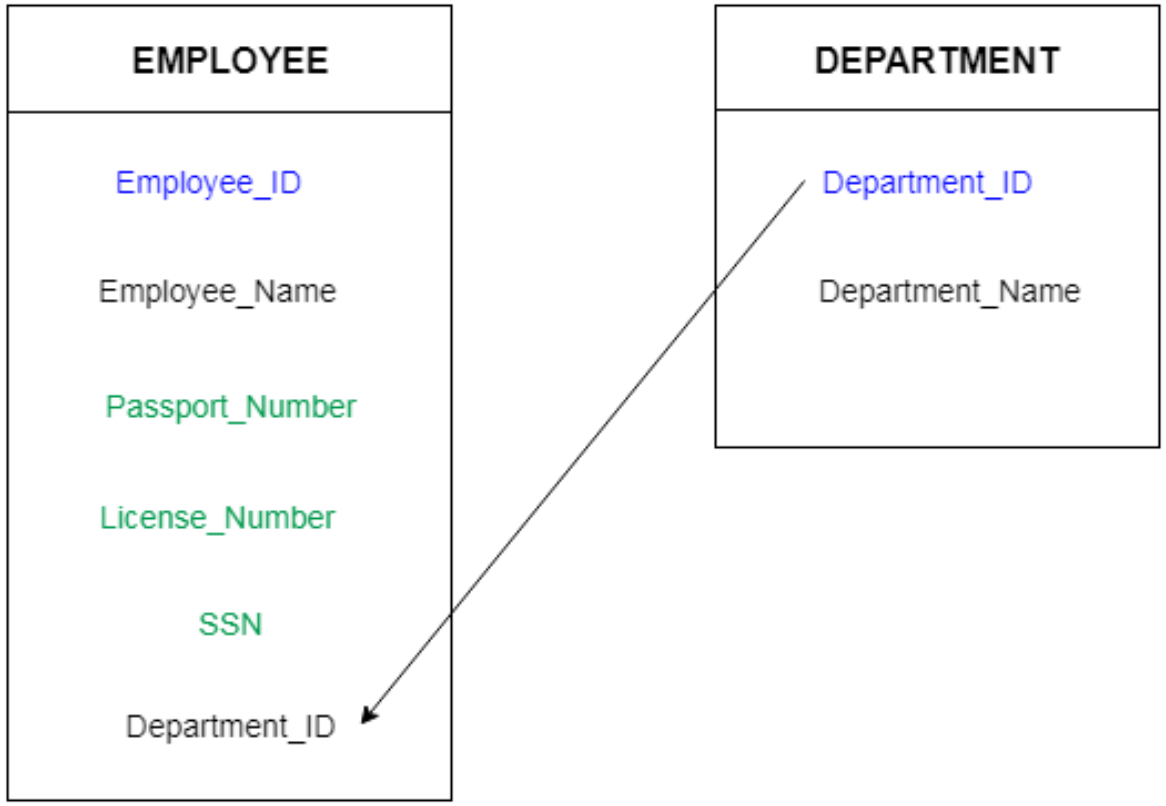
Anahtar türleri:

4. Yabancı anahtar

- Yabancı anahtarlar, başka bir tablonun birincil anahtarına işaret etmek için kullanılan tablonun sütunudur.
- Bir şirkette her çalışan belirli bir departmanda çalışır ve çalışan ve departman iki farklı varlıktır. Bu yüzden departmanın bilgilerini çalışan tablosunda saklayamayız. Bu nedenle, bu iki tabloyu bir tablonun birincil anahtarıyla birbirine bağlıyoruz.
- DEPARTMENT tablosunun birincil anahtarı olan Department_Id'yi EMPLOYEE tablosuna yeni bir öznitelik olarak ekliyoruz.
- Şimdi EMPLOYEE tablosunda, Department_Id yabancı anahtardır ve her iki tablo da ilişkilidir.

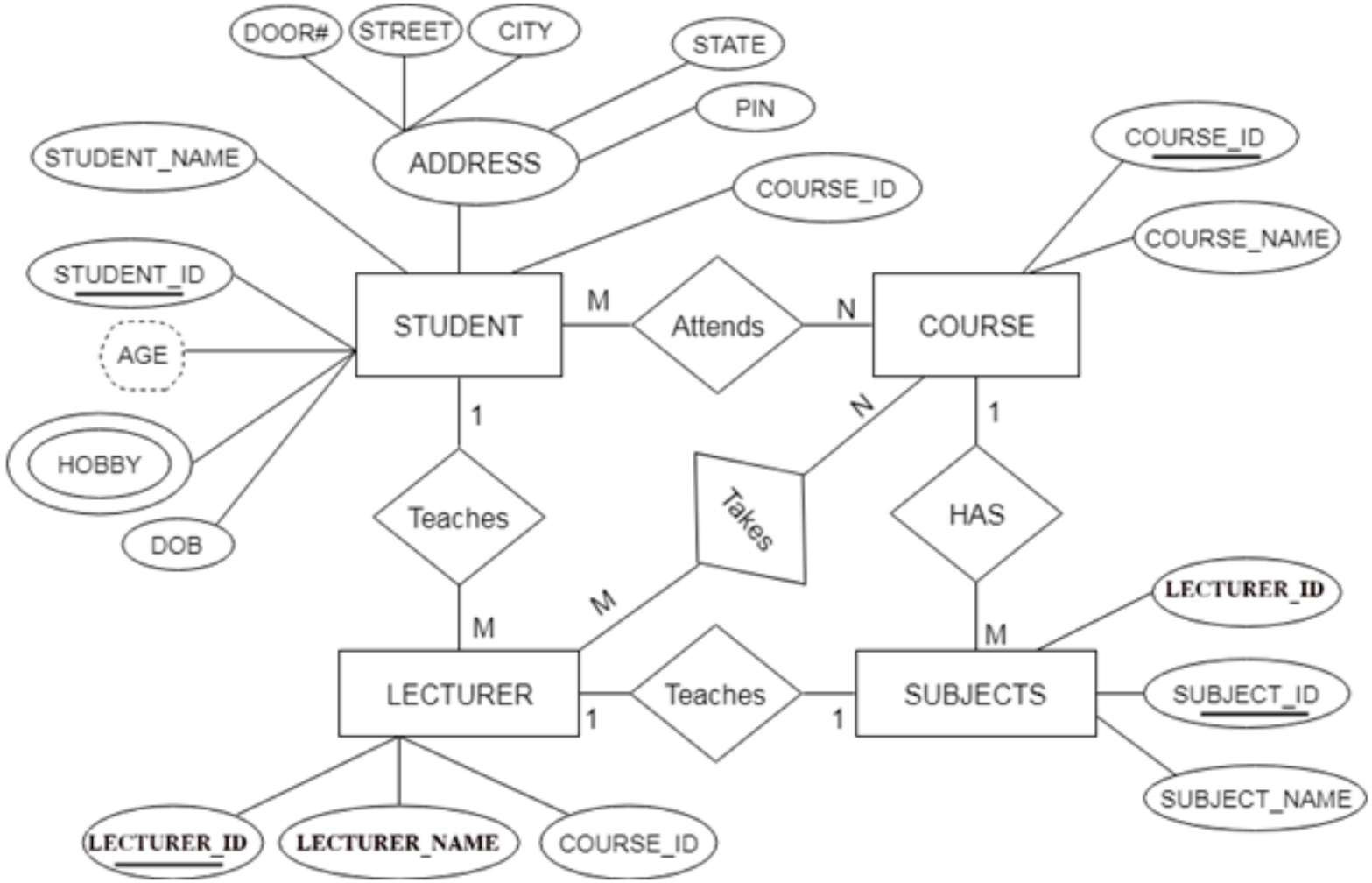
Anahtar türleri:

4. Yabancı anahtar



ER diyagramının Tabloya indirgenmesi

- Veritabanı, gösterimler kullanılarak temsil edilebilir ve bu gösterimler bir tablo koleksiyonuna indirgenebilir.
- Veritabanında, her varlık kümesi veya ilişki kümesi tablo biçiminde temsil edilebilir.

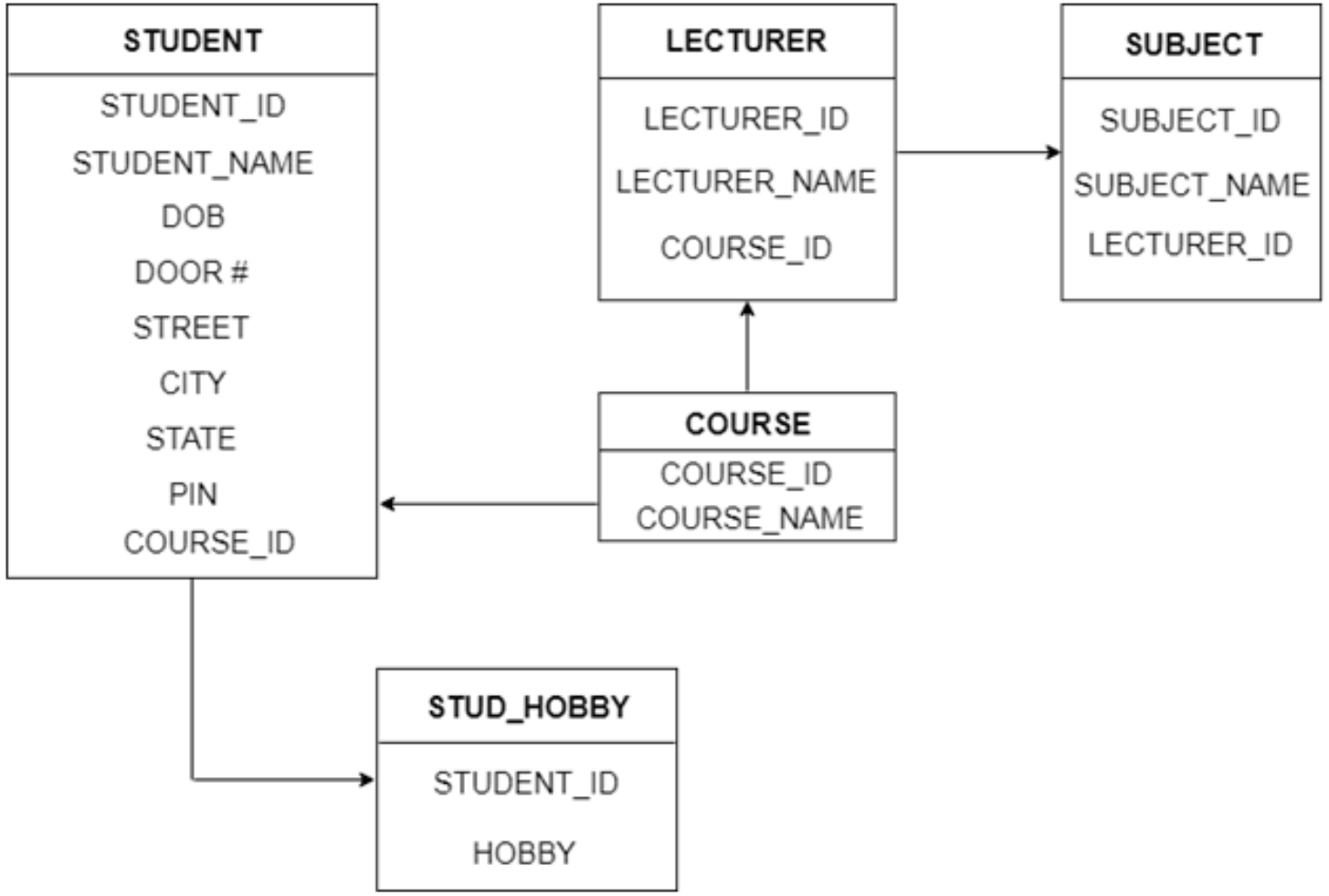


ER diyagramının Tabloya indirgenmesi

ER diyagramını tabloya dönüştürmek için bazı noktalar vardır:

- Varlık türü bir tablo haline gelir.
- Tüm tek değerli öznitelikler tablo için bir sütun haline gelir.
- Birincil anahtarla temsil edilen varlık türünün anahtar özelliği.
- Birden çok değerli öznitelik ayrı bir tabloyla temsil edilir.
- Bileşenlerle temsil edilen bileşik öznitelik.
- Tabloda türetilmiş özellikler dikkate alınmaz.

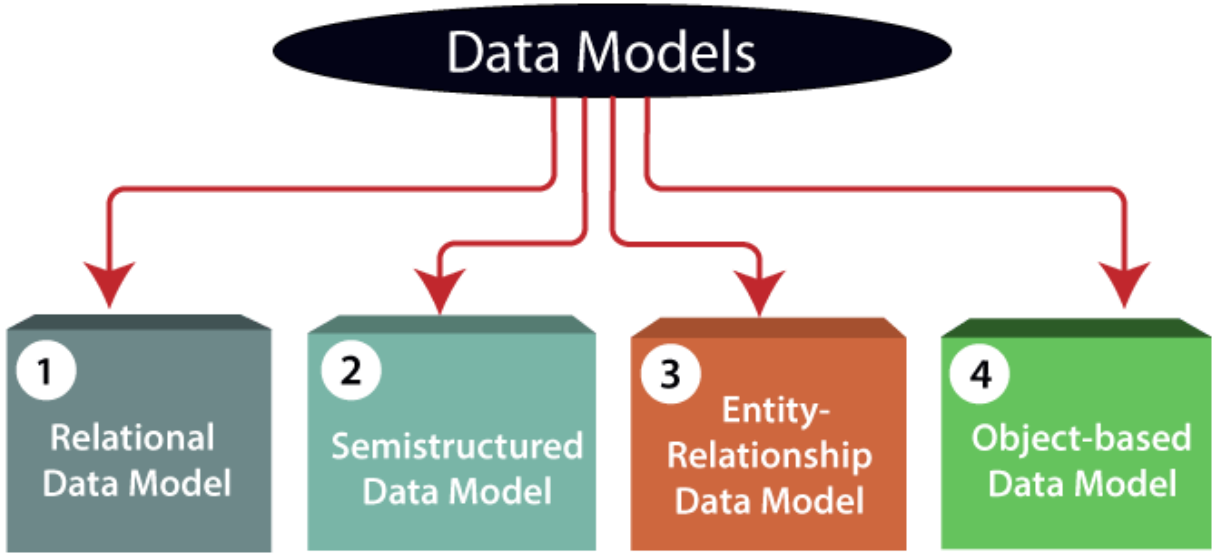
ER diyagramının Tabloya indirgenmesi



İlişkisel Model kavramı

DBMS Veritabanı Modelleri

Veri Modeli, veri tanımının, veri anlamının ve verilerin tutarlılık kısıtlamalarının modellenmesidir. Her veri soyutlama düzeyinde bir veritabanı tasarımını açıklamak için kavramsal araçlar sağlar. Bu nedenle, veritabanının yapısını anlamak için kullanılan aşağıdaki dört veri modeli vardır:



İlişkisel Model kavramı

Codd, 1970 yılında ilişkisel veri modelini önerdi. O zamanlar çoğu veritabanı sistemi, iki eski veri modelinden (hiyerarşik model ve ağ modeli) birine dayanıyordu; ilişkisel model, veritabanı alanında devrim yarattı ve büyük ölçüde bu önceki modellerin yerini aldı. Prototip ilişkisel veritabanı yönetim sistemleri, 70'lerin ortalarında IBM ve UC-Berkeley'deki öncü araştırma projelerinde geliştirildi ve kısa bir süre sonra birkaç satıcı ilişkisel veritabanı ürünleri sunmaya başladı. Bugün, ilişkisel model açık ara en baskın veri modelidir ve IBM'in DB2 ailesi, Informix, Oracle, Sybase, Microsoft'un Access ve SQLServer, FoxBase ve Paradox dahil önde gelen DBMS ürünlerinin temelidir.

İlişkisel Model kavramı

Dr Edgar F. Codd, Veritabanı sistemlerinin İlişkisel Modeli üzerine yaptığı kapsamlı araştırmanın ardından, kendisine göre bir veritabanının gerçek bir ilişkisel veritabanı olarak görülmesi için uyması gereken on iki kuralı ortaya attı.

Bu kurallar, depolanan verileri yalnızca ilişkisel yeteneklerini kullanarak yöneten herhangi bir veritabanı sistemine uygulanabilir. Bu, diğer tüm kuralların temelini oluşturan bir temel kuralıdır.

Codd'un kuralı, aslında bir DBMS'nin İlişkisel Veritabanı Yönetim Sistemi (RDBMS) haline gelmek için hangi kaliteyi gerektirdiğini tanımlar. Şimdiye kadar, 13 Codd'un kurallarına uyan neredeyse hiç ticari ürün yok. Oracle bile 13'ün yalnızca sekizini (8.5) takip ediyor.

Codd Kuralları

Kural 1: Bilgi Kuralı

Bir veritabanında depolanan veriler, kullanıcı verileri veya meta veriler olabilir, bazı tablo hücrelerinin bir değeri olmalıdır. Veritabanındaki her şey bir tablo formatında saklanmalıdır.

Kural 2: Garantili Erişim Kuralı

Her bir veri elemanının (değer), tablo-adı, birincil-anahtar (satır değeri) ve öznitelik-adı (sütun değeri) kombinasyonuyla mantıksal olarak erişilebilir olması garanti edilir. Verilere erişmek için işaretçiler gibi başka hiçbir yöntem kullanılamaz.

Kural 3: BOŞ Değerlerin Sistematik Olarak Değerlendirilmesi

Veritabanındaki NULL değerlere sistematik ve tek tip bir muamele verilmelidir. Bu çok önemli bir kuraldır çünkü bir NULL aşağıdakilerden biri olarak yorumlanabilir - veriler eksik, veriler bilinmiyor veya veriler geçerli değil.

Codd Kuralları

Kural 4: Etkin Çevrimiçi Katalog

Tüm veritabanının yapı açıklaması, yetkili kullanıcılar tarafından erişilebilen, veri sözlüğü olarak bilinen çevrimiçi bir katalogda saklanmalıdır . Kullanıcılar, veritabanına erişmek için kullandıkları kataloğa erişmek için aynı sorgu dilini kullanabilir.

Kural 5: Kapsamlı Veri Alt Dil Kuralı

Bir veritabanına yalnızca veri tanımını, veri işlemeyi ve işlem yönetimi işlemlerini destekleyen doğrusal sözdizimine sahip bir dil kullanılarak erişilebilir. Bu dil doğrudan veya bazı uygulamalar aracılığıyla kullanılabilir. Veritabanı bu dilin herhangi bir yardımı olmadan verilere erişime izin veriyorsa, bu bir ihlal olarak kabul edilir.

Kural 6: Güncelleme Kuralını Görüntüle

Bir veritabanının teorik olarak güncellenebilen tüm görünümleri de sistem tarafından güncellenebilir olmalıdır.

Codd Kuralları

Kural 7: Yüksek Düzeyli Ekleme, Güncelleme ve Silme Kuralı

Bir veritabanı, yüksek düzeyde ekleme, güncelleme ve silme işlemlerini desteklemelidir. Bu, tek bir satırla sınırlı olmamalıdır; yani, veri kaydı kümeleri elde etmek için birleşim, kesişim ve eksi işlemleri de desteklemelidir.

Kural 8: Fiziksel Veri Bağımsızlığı

Bir veritabanında depolanan veriler, veritabanına erişen uygulamalardan bağımsız olmalıdır. Bir veritabanının fiziksel yapısındaki herhangi bir değişikliğin, verilere harici uygulamalar tarafından nasıl erişildiği üzerinde herhangi bir etkisi olmamalıdır.

Codd Kuralları

Kural 9: Mantıksal Veri Bağımsızlığı

Bir veritabanındaki mantıksal veriler, kullanıcının görüşünden (uygulamasından) bağımsız olmalıdır. Mantıksal verilerdeki herhangi bir değişiklik, onu kullanan uygulamaları etkilememelidir. Örneğin, iki tablo birleştirilirse veya biri iki farklı tabloya bölünürse, kullanıcı uygulamasında herhangi bir etki veya değişiklik olmamalıdır. Bu, uygulanması en zor kurallardan biridir.

Kural 10: Bütünlük Bağımsızlığı

Bir veritabanı, onu kullanan uygulamadan bağımsız olmalıdır. Tüm bütünlük kısıtlamaları, uygulamada herhangi bir değişikliğe gerek kalmadan bağımsız olarak değiştirilebilir. Bu kural, bir veritabanını ön uç uygulamadan ve arayüzünden bağımsız kılar.

Codd Kuralları

Kural 11: Dağıtım Bağımsızlığı

Son kullanıcı, verilerin çeşitli konumlara dağıtıldığını görememelidir. Kullanıcılar her zaman verilerin yalnızca tek bir sitede bulunduğu izlenimini edinmelidir. Bu kural, dağıtık veritabanı sistemlerinin temeli olarak kabul edilmiştir.

Kural 12: Yıkmama Kuralı

Bir sistemin düşük düzeyli kayıtlara erişim sağlayan bir arabirimi varsa, arabirim sistemi bozmamalı ve güvenlik ve bütünlük kısıtlamalarını atlamamalıdır.

İlişkisel Model kavramı

İlişkisel model çok basit ve zariftir; veritabanı, her ilişkinin satırlar ve sütunlar içeren bir tablo olduğu bir veya daha fazla ilişki topluluğudur. Bu basit tablo gösterimi, acemi kullanıcıların bile bir veritabanının içeriğini anlamasını sağlar ve verileri sorgulamak için basit, yüksek seviyeli dillerin kullanılmasına izin verir. İlişkisel modelin eski veri modellerine göre en büyük avantajları, basit veri gösterimi ve karmaşık sorguların bile kolaylıkla ifade edilebilmesidir.

İlişkisel Model kavramı

Bir İlişkisel Veritabanı Yönetim Sistemi (RDBMS) EF Codd tarafından tanıtılan ilişkisel modeline dayalı bir veritabanı yönetim sistemidir. İlişkisel modelde, veriler ilişkilerde (tablolarda) depolanır ve tuple (satırlar) şeklinde temsil edilir .

RDBMS , İlişkisel veritabanını yönetmek için kullanılır. İlişkisel veritabanı , birbirleriyle ilişkili olan ve verilere kolayca erişilebilen organize tablolar topluluğudur. İlişkisel Veritabanı bu günlerde en yaygın kullanılan veritabanıdır.

İlişkisel veri modeli, veri depolama ve işleme için dünya çapında yaygın olarak kullanılan birincil veri modelidir. Bu model basittir ve verileri depolama verimliliği ile işlemek için gereken tüm özelliklere ve yeteneklere sahiptir.

İlişkisel model, sütunları ve satırları olan bir tablo olarak temsil edilebilir. Her satır bir demet olarak bilinir. Sütunun her tablosunun bir adı veya özniteliği vardır.

İlişkisel Model kavramı

İlişkisel modelde verileri temsil eden ana yapı bir ilişkidir. Bir ilişki, bir ilişki şemasından ve bir ilişki örneğinden oluşur. İlişki örneği bir tablodur ve ilişki şeması tablonun sütun başlıklarını tanımlar. İlk önce ilişki şemasını ve ardından ilişki örneğini tanımlıyoruz. Şema, ilişkinin adını, her alanın (veya sütunun veya özniteliğin) adını ve her alanın alanını belirtir. Bir etki alanı, bir ilişki şemasında etki alanı adıyla anılır ve bir dizi ilişkili değere sahiptir.

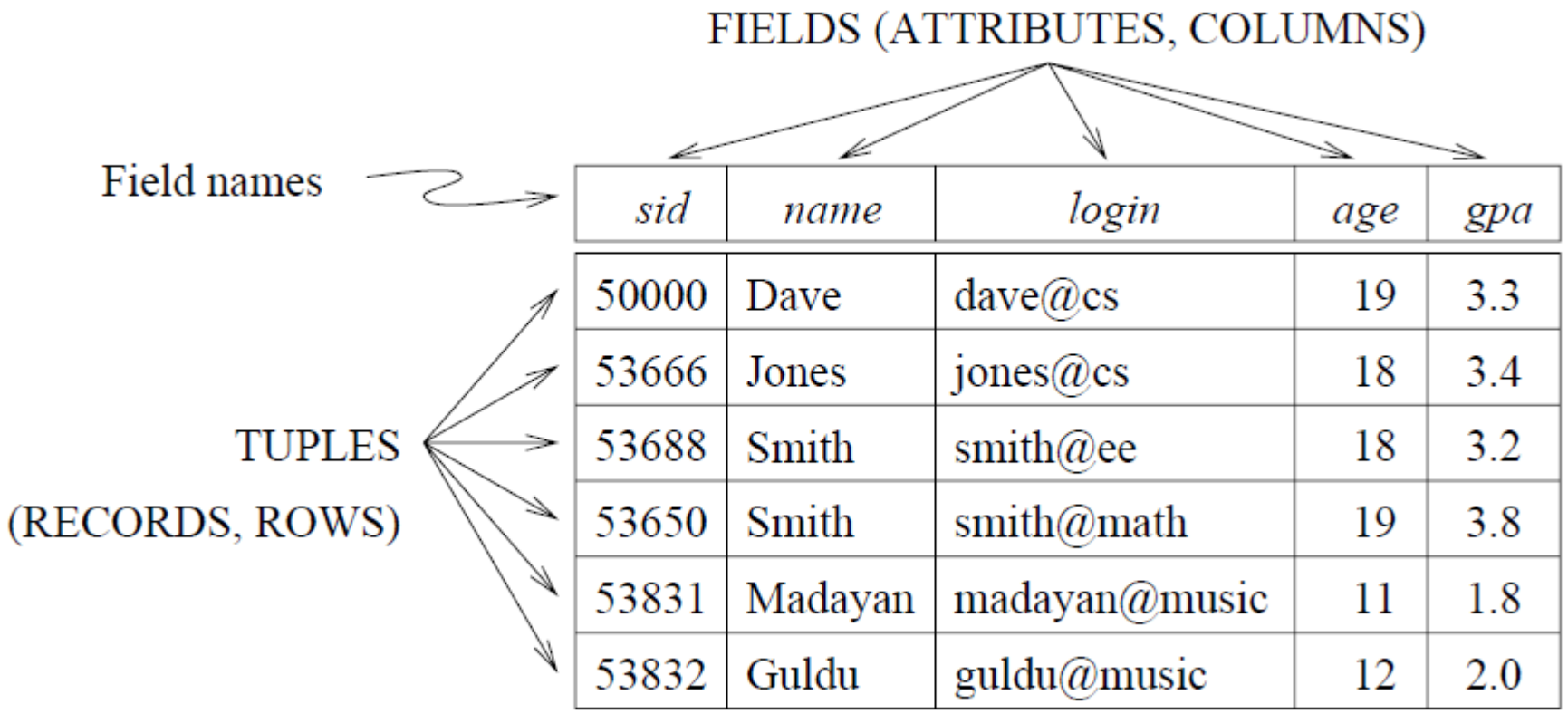
Students(*sid:* `string`, *name:* `string`, *login:* `string`, *age:* `integer`, *gpa:* `real`)

İlişkisel Model kavramı

Bir ilişkinin bir örneği (instance), her bir demetin (tuple) ilişki şemasıyla aynı sayıda alana sahip olduğu, kayıtlar (records) olarak da adlandırılan bir demetler kümesidir. Bir ilişki örneği, her bir demetin bir satır olduğu ve tüm satırların aynı sayıda alana sahip olduğu bir tablo olarak düşünülebilir. (İlişki örneği terimi, bir ilişkinin şeması gibi diğer yönleriyle herhangi bir karışıklık olmadığında, genellikle adil ilişki olarak kısaltılır.)

Students(*sid:* `string`, *name:* `string`, *login:* `string`, *age:* `integer`, *gpa:* `real`)

İlişkisel Model kavramı



sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Students(*sid:* `string`, *name:* `string`, *login:* `string`, *age:* `integer`, *gpa:* `real`)

İlişkisel Model kavramı

Bir ilişki şeması, ilişki örneğindeki her alanın veya sütunun alanını belirtir. Şemadaki bu etki alanı kısıtlamaları (domain constraints), ilişkinin her örneğinin karşılamasını istediğimiz önemli bir koşulu belirtir: Bir sütunda görünen değerler, o sütunla ilişkili etki alanından alınmalıdır. Bu nedenle, bir alanın etki alanı, programlama dili terimleriyle esasen o alanın türüdür ve alanda görünebilecek değerleri kısıtlar.

İlişkisel Model kavramı

More formally, let R(f_1:D1, ..., f_n:Dn) be a relation schema, and for each f_i, $1 \leq i \leq n$, let Dom_i be the set of values associated with the domain named Di. An instance of R that satisfies the domain constraints in the schema is a set of tuples with n fields:

$$\{ \langle f_1 : d_1, \ \ldots \ , f_n : d_n \rangle \ \mid \ d_1 \in Dom_1, \ \ldots \ , d_n \in Dom_n \ \}$$

The angular brackets $\langle \ldots \rangle$ identify the fields of a tuple. Using this notation, the first Students tuple shown in Figure 3.1 is written as $\langle$*sid:* 50000, *name:* Dave, *login:* dave@cs, *age:* 19, *gpa:* 3.3$\rangle$. The curly brackets $\{\ldots\}$ denote a set (of tuples, in this definition). The vertical bar | should be read 'such that,' the symbol $\in$ should be read 'in,' and the expression to the right of the vertical bar is a condition that must be satisfied by the field values of each tuple in the set. Thus, an instance of R is defined as a set of tuples. The fields of each tuple must correspond to the fields in the relation schema.

Kavramlar

RDBMS , ilişkisel veritabanı yönetim sistemi anlamına gelir. İlişkisel bir model, bir satır ve sütun tablosu olarak temsil edilebilir. İlişkisel bir veritabanı aşağıdaki ana bileşenlere sahiptir:

1. Tablo
2. Kayıt veya Tuple
3. Alan veya Sütun adı veya Öznitelik
4. Etki Alanı
5. Örnek
6. Şema
7. Anahtarlar

Kavramlar

1. Tablolar - İlişkisel veri modelinde ilişkiler Tablolar formatında kaydedilir. Bu format, varlıklar arasındaki ilişkiyi saklar. Bir tabloda, satırların kayıtları ve sütunların öznitelikleri temsil ettiği satırlar ve sütunlar vardır.

Tablo, satırlar ve sütunlar halinde temsil edilen bir veri koleksiyonudur. Her tablonun veritabanında bir adı vardır. Örneğin, aşağıdaki "STUDENT" tablosu öğrencilerin bilgilerini veri tabanında saklamaktadır.

İSİM	ROLL_NO	TELEFON YOK	ADRES	YAŞ
Veri deposu	14795	7305758992	Noida	24
Shyam	12839	9026288936	Delhi	35
Laxman	33289	8583287182	Gurugram	20
Mahesh	27857	7086819134	Ghaziabad	27
Ganesh	17282	9028 9i3988	Delhi	40

•In the given table, NAME, ROLL_NO, PHONE_NO, ADDRESS, and AGE are the attributes.
•The instance of schema STUDENT has 5 tuples.
•t3 = <Laxman, 33289, 8583287182, Gurugram, 20>

Kavramlar

2. Tuple/Kayıt - Bu ilişki için tek bir kayıt içeren bir tablonun tek satırına demet adı verilir.

Bir tablonun her satırı kayıt olarak bilinir. Tuple olarak da bilinir. Örneğin aşağıdaki satır, yukarıdaki tablodan aldığımız bir kayıttır.

Shyam	12839	9026288936	Delhi	35

3. Özellik/Atrribute: Belirli bir tablodaki bir sütunun adını içerir. Her Ai özniteliğinin bir etki alanı (dom (Ai)) olmalıdır.

İSİM	ROLL_NO	TELEFON YOK	ADRES	YAŞ

Kavramlar

4. Etki Alanı/Domain: Bir özniteliğin alabileceği bir dizi atomik değer içerir. Alan, tablodaki bir öznitelik için izin verilen değerler kümesidir. Örneğin, bir yılın ayı etki alanı Ocak, Şubat,… Aralık değerlerini değer olarak kabul edebilir, tarih etki alanı tüm olası geçerli tarihleri vb. Kabul edebilir. Tablo oluştururken öznitelik etki alanını belirleriz.

Bir öznitelik, etki alanlarının dışındaki değerleri kabul edemez.

İlişkisel örnek/Relational instance: İlişkisel veritabanı sisteminde, ilişkisel örnek sonlu bir demet kümesi ile temsil edilir. İlişki örneklerinin yinelenen demetleri yoktur.

İlişkisel şema: İlişkisel bir şema , ilişkinin adını ve tüm sütunların veya özniteliklerin adını içerir.

İlişkisel anahtar: İlişkisel anahtarda, her satırın bir veya daha fazla özniteliği vardır. İlişkideki satırı benzersiz bir şekilde tanımlayabilir.

İlişkilerin Özellikleri

- İlişkinin adı diğer tüm ilişkilerden farklıdır.
- Her ilişki hücresi tam olarak bir atomik (tek) değer içerir
- Her öznitelik farklı bir ad içerir
- Öznitelik alanının önemi yoktur
- tuple yinelenen değere sahip değil
- Demet sırası farklı bir sıraya sahip olabilir

Kısıtlamalar

Her ilişkinin, geçerli bir ilişki olması için tutması gereken bazı koşulları vardır. Bu koşullara İlişkisel Bütünlük Kısıtlamaları denir . Üç temel bütünlük kısıtlaması vardır -

- Anahtar kısıtlamalar
- Etki alanı kısıtlamaları
- Bilgi tutarlılığı kısıtlamaları