Bölüm 2: İlişkisel Model (Relational Model)'e Giriş

Database System Concepts, 7th Ed.

İlişkisel Veritabanlarının Yapısı

- İlişkisel bir veritabanı, her birine benzersiz bir ad atanan bir tablolar koleksiyonundan oluşur. Örneğin, eğitmenler hakkında bilgi depolayan Şekil 2.1'deki eğitmen tablosunu düşünün.
- **instructor** (eğitmen) tablosunu ele alalım, bu tablo eğitmenler hakkında bilgi depolar. Tablo dört sütun başlığına sahiptir: **ID**, **name** (isim), **dept name** (bölüm adı) ve **salary** (maaş). Bu tablonun her bir satırı, eğitmenin kimlik numarası, ismi, bölüm adı ve maaşından oluşan bilgiler kaydeder.

attributes
(öznitelikler)
(veya kolonlar)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples
(veya satırlar)

Course (Ders) İlişkisi Örneği

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

Genel olarak, bir tabloda yer alan bir satır, bir dizi değer arasındaki ilişkiyi temsil eder. Bir tablo, bu tür ilişkilerin bir koleksiyonu olduğundan, tablo kavramı ile matematiksel **relation** (tablo) kavramı arasında yakın bir bağ vardır. Bu ilişki modeli, ilişkisel (relational) veritabanı modeline adını vermektedir. Matematiksel terimlerle, bir **tuple** (satır), basitçe bir değer dizisi (veya liste) anlamına gelir. **n** değeri arasındaki bir ilişki, matematiksel olarak **n-tuple** (n-satır) ile ifade edilir, yani **n** değerden oluşan bir satır, bir tablodaki bir satıra karşılık gelir. Bu nedenle, ilişkisel modelde **relation** terimi bir tabloyu ifade ederken, **tuple** terimi bir satırı ifade eder. Benzer şekilde, **attribute** (öznitelik) terimi bir tablonun sütununu ifade eder.

Tablo (Relation) Şeması ve Instance

- *Instructor (Eğitmen)* ilişkisi dört niteliğe (attribute) sahiptir

  instructor = (*ID, name, dept_name, salary*)

- Relation instance (tablo örneği) terimi, belirli bir satır kümesi içeren bir ilişkinin belirli bir örneğini ifade etmek için kullanılır.
- *Instructor (Eğitmen)* örneği 12 satırdan (tuple) oluşur.

  12 adet eğitmen

attributes
(or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples
(or rows)

Sıralanmamış İlişkiler

- Bir ilişkide satırların sırası önemsizdir, çünkü bir ilişki satırların bir kümesidir. Her şekilde de aynı tuple kümesini içerdiğinden ilişkiler aynıdır.
  Örneğin; Sıralanmamış instructor ilişkisi

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Attributes

- Her bir ilişki özelliği için, bu özelliğin alabileceği izinli değerlerden oluşan bir küme vardır ve bu küme özelliğin etki alanı, ***domain*** olarak adlandırılır.

  Örneğin, **instructor** ilişkisinin **salary** (maaş) özelliğinin alanı, tüm olası maaş değerlerinin kümesidir. Benzer şekilde **name** (isim) özelliğinin lanı, tüm olası eğitmen isimlerinden oluşur.

- Attribute değerlerinin (normalde) ***atomic*** olması gerekir. Bir alan atomikse, o alanın elemanları bölünemez birimler olarak kabul edilir.

  Örneğin, **instructor** (eğitmen) tablosunda eğitmenle ilişkili bir telefon numaraları kümesini depolayan bir **phone number** (telefon numarası) özelliği olduğunu varsayalım. Bu durumda, **phone number** özelliğinin alanı atomik olmazdı çünkü alanın bir elemanı bir telefon numaraları kümesi olup, kümenin alt parçaları, yani bireysel telefon numaraları vardır. Burada önemli olan nokta, alanın ne olduğu değil, veritabanımızda alan elemanlarını nasıl kullandığımızdır. Şimdi de **phone number** özelliğinin tek bir telefon numarasını depoladığını düşünelim. Yine de, eğer telefon numarası değerini ülke kodu, alan kodu ve yerel numara gibi parçalara ayırırsak, onu atomik olmayan bir değer olarak işlemiş oluruz. Eğer her telefon numarasını bölünemez bir birim olarak ele alırsak, o zaman **phone number** özelliği atomik bir alana sahip olur.

- ***Null*** değeri, bir değerin bilinmediğini veya var olmadığını ifade eden özel bir değerdir. Örneğin, daha önce olduğu gibi, **instructor** (eğitmen) ilişkisine **phone number** (telefon numarası) özelliğini eklediğimizi varsayalım. Bir eğitmenin hiç telefon numarası olmayabilir ya da telefon numarası gizli olabilir. Bu durumda, değerin bilinmediğini veya var olmadığını belirtmek için **null** değerini kullanmamız gerekir

Database Schema (Veritabanı Şeması)

- Database schema (Veritabanı Şeması) → Veritabanının mantıksal görüntüsüdür.
- Database instance (Veritabanı örneği) → Belirli bir zamanda veritabanındaki verilerin anlık görüntüsüdür.
- Example:
  - schema: *instructor* (*ID, name, dept_name, salary*)
  - Instance:

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Relation Schema (Tablo Şeması)

- Relation (Tablo) → Programlama dillerindeki değişken kavramına karşılık gelir.
- Relation Schema (Tablo Şeması) → Programlama dillerindeki tip kavramına karşılık gelir.

➢ Genel olarak, bir ilişki şeması, bir dizi nitelik (attribute) ve bunların karşılık gelen alanlarından (domain) oluşur.

- Relation instance (Tablo örneği) → Programlama dillerinde bir değişkenin değerine karşılık gelir. Belirli bir değişkenin değeri zamanla değişebilir; benzer şekilde, bir ilişki örneğinin içeriği, ilişki güncellendikçe zamanla değişebilir. Buna karşılık bir ilişkinin şeması genellikle değişmez.

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Department (Bölüm) ilişkisinin şeması:

department (dept name, building, budget)

- **dept name** (bölüm adı) özelliğinin hem **instructor** (eğitmen) şemasında hem de **department** (bölüm) şemasında yer alır. Ortak özellikleri ilişki şemalarında kullanmak, farklı ilişkilerin **tuple** (kayıt)'larını ilişkilendirmenin bir yoludur.

Örneğin, **Watson** binasında çalışan tüm eğitmenler hakkında bilgi bulmak istediğimizi varsayalım. Öncelikle, **department** ilişkisine bakarak **Watson**'da yer alan tüm bölümlerin **dept name**'ini buluruz. Ardından, bu bölümler için **instructor** ilişkisinde ilgili **dept name** ile ilişkilendirilmiş eğitmen hakkında bilgi bulmak için arama yaparız.

Örnek

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

section (course id, sec id, semester, year, building, room number, time slot id)

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

teaches (ID, course id, sec id, semester, year)

Gerçek bir üniversite veritabanında daha pek çok ilişki bulunmaktadır. Daha önce listelenen **instructor**, **department**, **course**, **section**, **prereq** ve **teaches** ilişkilerine ek olarak aşağıdaki ilişkiler de kullanılacak:

- Öğrenci için → **student** (ID, name, dept name, tot cred)
- Danışman için → **advisor** (s id, i id)
- Dersin alınma bilgileri için → **takes** (ID, course id, sec id, semester, year, grade)
- Sınıf için → **classroom** (building, room number, capacity)
- Zaman aralığı için → **time slot** (time slot id, day, start time, end time)

Keys

- ***Superkey*** (üst anahtar), bir ilişkideki bir **tuple**'ı benzersiz bir şekilde tanımlamak için topluca alınan bir veya daha fazla özelliğin kümesidir.
  - ➢ Örneğin, **instructor** (eğitmen) ilişkisinin **ID** özelliği, bir eğitmen kaydını diğerlerinden ayırmak için yeterlidir. Dolayısıyla, **ID** bir **superkey**'dir. Öte yandan, **instructor**'ın **name** (isim) özelliği bir **superkey** değildir, çünkü birkaç eğitmen aynı isme sahip olabilir.

- Bir **superkey**, gereksiz özellikler içerebilir. Örneğin, **ID** ve **name** (isim) kombinasyonu, **instructor** ilişkisi için bir **superkey**'dir. Eğer K bir **superkey** ise, o zaman K'nın herhangi bir süper kümesi de bir **superkey**'dir. Genellikle, uygun alt kümesi olmayan **superkey**'lerle ilgileniriz. Bu tür minimal **superkey**'lere ***candidate key*** (aday anahtar) denir.
  - ➢ Birden fazla farklı özellik kümesinin aday anahtar olarak işlev görebileceği durumlar mümkündür. Örneğin, **name** (isim) ve **dept name** (departman ismi) kombinasyonunun **instructor** ilişkisi üyelerini ayırt etmek için yeterli olduğunu varsayalım. Bu durumda, hem {ID} hem de {name, dept name} aday anahtarlarıdır. **ID** ve **name** özellikleri bir arada **instructor tuple**'larını ayırt edebilse de, bu özelliklerin kombinasyonu {ID, name} bir aday anahtar oluşturmaz, çünkü **ID** özelliği tek başına bir aday anahtardır.

- **Primary key (Birinci anahtar)** terimi, bir veritabanı tasarımcısı tarafından bir ilişkideki **tuple**'ları tanımlamak için ana araç olarak seçilen **candidate key**'i belirtmek için kullanılır.
- Bir anahtar (birincil, aday veya süper anahtar olsun) tüm ilişkinin bir özelliğidir. İlişkinin herhangi iki bireysel **tuple**'ının, anahtar özelliklerinde aynı anda aynı değere sahip olması yasaktır. Anahtarın belirlenmesi, modelleme yapılan gerçek dünya işletmesinde bir kısıtlamayı temsil eder.
- Bir ilişki şemasının birincil anahtar özelliklerini, diğer özelliklerden önce listelenir; örneğin, **department** (departman) ilişkisi için **dept name** (departman ismi) özelliği birincil anahtar olduğu için ilk sırada listelenir. Birincil anahtar özellikleri de altı çizilir.

classroom (building, room number, capacity)

Burada birincil anahtar, altı çizilmiş olan iki özelliği, **building** (bina) ve **room number** (oda numarası) içerir. Tek başına hiçbir özellik bir sınıfı benzersiz şekilde tanımlayamaz; ancak birlikte, bir sınıfı benzersiz bir şekilde tanımlarlar.

time slot (<u>time slot id</u>, <u>day</u>, <u>start time</u>, end time)

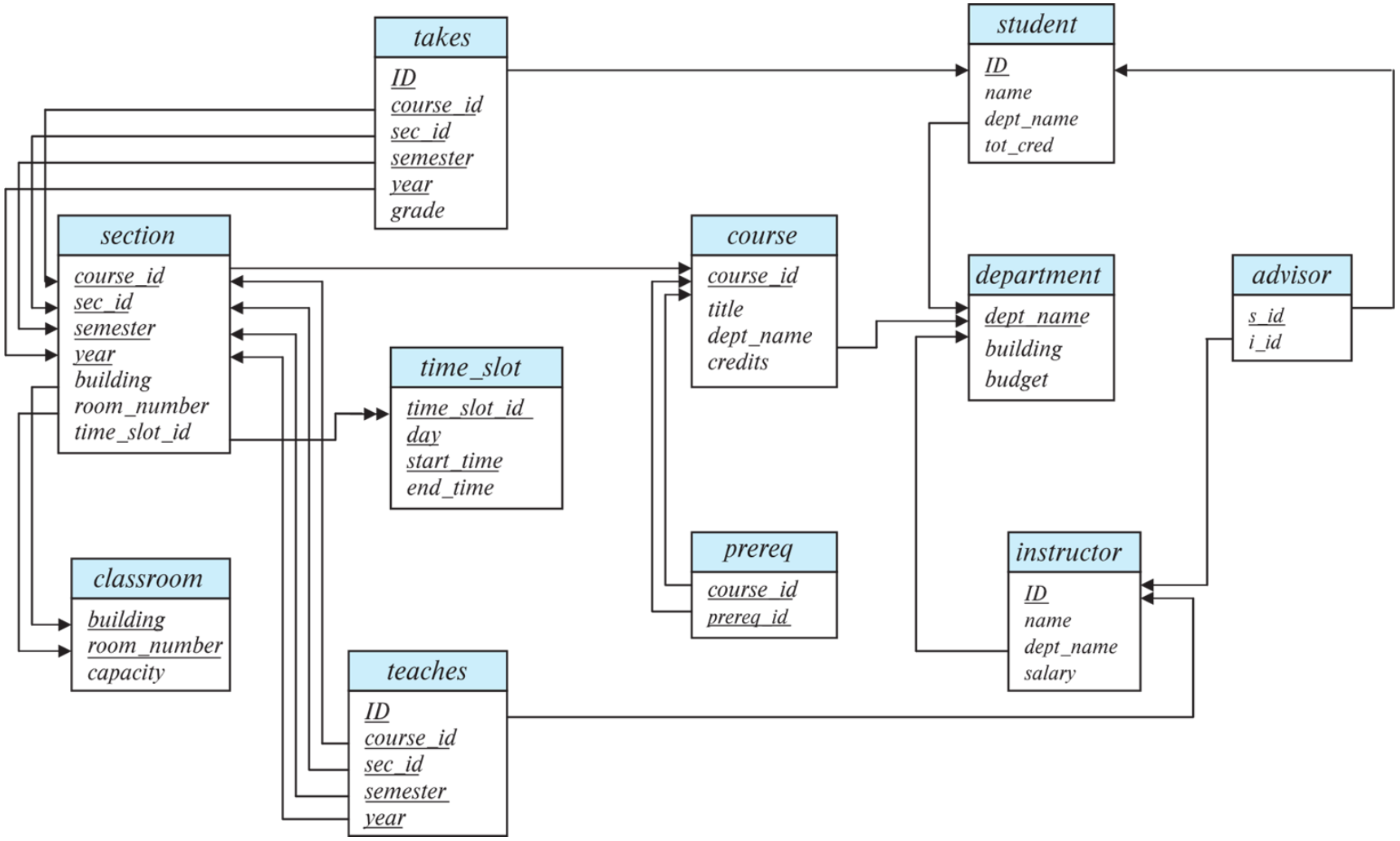
Her bölümün bir zaman dilimi kimliği ile ilişkili olduğunu varsayalım. **time slot** ilişkisi, belirli bir zaman dilimi kimliğinin hangi günlerde ve saatlerde bulunduğu hakkında bilgi sağlar. Örneğin, zaman dilimi kimliği **'A'**, Pazartesi, Çarşamba ve Cuma günleri sabah 8:00 ile 8:50 arasında olabilir. Bir zaman diliminin tek bir günde, farklı zaman dilimlerinde birden fazla oturuma sahip olması mümkündür; bu nedenle, zaman dilimi kimliği ve gün bir arada, **tuple**'ı benzersiz şekilde tanımlamaz. **time slot** ilişkisinin birincil anahtarı bu nedenle **time slot id** (zaman dilimi kimliği), **day** (gün) ve **start time** (başlangıç zamanı) özelliklerinden oluşur; çünkü bu üç özellik bir arada bir ders için bir zaman dilimini benzersiz bir şekilde tanımlar.

- **Foreign key, r1** ilişkisi içindeki **A** özellikleri ile **r2** ilişkisi içindeki birincil anahtar **B** arasında belirli bir ilişkiyi ifade eder. Yani, her **tuple** için **r1** içindeki **A** değerinin, **r2** içindeki bazı **tuple**'ların **B** değerleri ile aynı olması gerekir. **A** özellikleri **r1**'deki bir **foreign key** olarak adlandırılır ve **r1** ilişkisi **referencing relation**, **r2** ilişkisi ise **referenced relation** olarak adlandırılır.

➢ Örneğin, **instructor** tablosundaki **dept name** özelliği, **department** tablosunu referans alan bir foreign key'dir; burada **dept name** özelliği **department** tablosunun birincil anahtarıdır. Benzer şekilde, **section** ilişkisi içindeki **building** (bina) ve **room number** (oda numarası) özellikleri birlikte, **classroom** ilişkisini referans alan bir foreign key oluşturur.

- Bir foreign key kısıtlamasında, referans alınan özellikler, referans verilen ilişkinin birincil anahtarı olmalıdır.

University Veritabanı için Şema Diyagramı



Veritabanı şeması, birincil anahtar ve yabancı anahtar kısıtlamaları ile birlikte şema diyagramlarıyla tasvir edilebilir. Şekilde, üniversite organizasyonumuza ait şema diyagramını göstermektedir. Her bir ilişki, kutu şeklinde gösterilir; kutunun üst kısmında ilişki adı mavi renkte yer alırken, iç kısmında ise nitelikler listelenir.

Primary-key: Primary key niteliği altı çizili olarak gösterilmektedir.

Foreign -keyı: Foreign key, referans alan ilişki içindeki yabancı anahtar niteliğinden, referans verilen ilişki içindeki birincil anahtar niteliğine yönlendiren oklar olarak görünür.

Referans Bütünlüğü Kısıtlamaları: Yalnızca referans bütünlüğü kısıtlamasını gösteren iki başlı ok kullanılır; bu, foreign key olmayan bir durumu belirtir. Örneğin, şekildeki **section** ilişkisindeki **time slot id** niteliğinden, **time slot** ilişkisindeki **time slot id** niteliğine yönelen iki başlı ok, **section time - slot id** ile **time slot - time slot id** arasındaki referans bütünlüğü kısıtlamasını (**referential-integrity constraint)** temsil eder.

Birçok veritabanı sistemi, şema diyagramları oluşturmak için grafiksel kullanıcı arayüzü sunan tasarım araçları sağlar. Ayrıca, şemaların başka bir diyagramatik gösterimi olan **varlık-ilişki diyagramı** da vardır. Görünüm açısından bazı benzerlikleri olmasına rağmen, bu iki gösterim oldukça farklıdır ve birbirleriyle karıştırılmamalıdır.