YILDIZ TEKNİK ÜNİVERSİTESİ

ER ile Veritabanı Kavramsal Tasarımı

Öğr. Gör. Dr. Yasemin Topuz

Yıldız Teknik Üniversitesi

BLM3041 – Veritabanı Yönetimi

28 Kasım 2025 | YTÜ Bilgisayar Mühendisliği

Neler konuşacağız?

- Tasarım Sürecine Genel Bakış
- Varlık-İlişki Modeli
- Karmaşık Nitelikler
- Kardinalitelerin Eşlenmesi

Tasarım Aşamaları (Design Phases)

- **İlk aşama** – gelecekteki veritabanı kullanıcılarının veri ihtiyaçlarının tam olarak belirlenmesi.

- **İkinci aşama** – bir veri modeli seçilmesi
  - Seçilen veri modelinin kavramlarının çıkarılması.
  - Bu gereksinimlerin veritabanının kavramsal şemasına dönüştürülmesi.
  - Tam geliştirilmiş bir kavramsal şema, fonksiyonel gereksinimleri gösterir.
    - Veriler üzerinde yapılacak işlemlerin (veya işlemsel süreçlerin) türlerini açıklar.

Bilgi Notu: Kavramsal şema, sistemin hangi verileri tuttuğunu ve bu veriler arasındaki ilişkileri yüksek seviyede gösteren şemadır. Teknik detay içermez, daha çok tasarım mantığını ifade eder.

Tasarım Aşamaları (Design Phases)

- **Son Aşama –** Soyut bir veri modelinden veritabanının gerçek uygulanışına geçiş.
  - **Mantıksal Tasarım (Logical Design)** – Veritabanı şemasına karar verme.
    - ✓ Veritabanı tasarımı, iyi bir ilişki şemaları koleksiyonu bulmayı gerektirir.
    - ✓ ***İş kararı:*** Veritabanında hangi nitelikleri (attributes) saklamalıyız?
    - ✓ ***Bilgisayar bilimi kararı:*** Hangi ilişki şemalarına sahip olmalıyız ve bu nitelikleri bu şemalar arasında nasıl dağıtmalıyız?
    - ✓ Sonuç, VTYS'nin veri modeli içinde bir veritabanı şemasıdır.
  - **Fiziksel Tasarım (Physical Design)** – Veritabanının fiziksel yerleşimine karar verme.

Bilgi Notu:

- **Mantıksal tasarım:** Tablo yapılarının, alanların ve ilişkilerin planlandığı aşamadır.
- **Fiziksel tasarım:** Verilerin diskte nasıl tutulacağı, dosya organizasyonları, indeksler, erişim yolları ve veritabanı dosyaları için fiziksel tasarım parametreleri belirlenir.

Tasarım Alternatifleri

- Bir veritabanı şeması tasarlarken iki büyük hatadan kaçmamız gerekir:
  - **Redundancy (Fazlalık):** Kötü bir tasarım, bilgilerin tekrar etmesine neden olabilir.
    - ✓ Bilginin gereksiz tekrarlarla tutulması, farklı kopyalar arasında tutarsızlıklara yol açabilir.
  - **Incompleteness (Eksiklik):** Kötü bir tasarım, işletmenin bazı yönlerini modellemeyi zorlaştırabilir veya imkânsız hale getirebilir.
- Kötü tasarımlardan kaçınmak yeterli değildir. Seçim yapmamız gereken birçok iyi tasarım seçeneği olabilir.

Redundancy: Aynı bilginin birden fazla yerde tutulması.

Incompleteness: Gerekli bilgilerin veya ilişkilerin modele dahil edilmemesi.

ER Modeli – Veritabanı Modellemesi

- ER veri modeli, veritabanı tasarımını kolaylaştırmak için geliştirilmiştir. Bunun nedeni, veritabanının genel mantıksal yapısını temsil eden bir kurumsal şema (enterprise schema) tanımlamaya izin vermesidir.

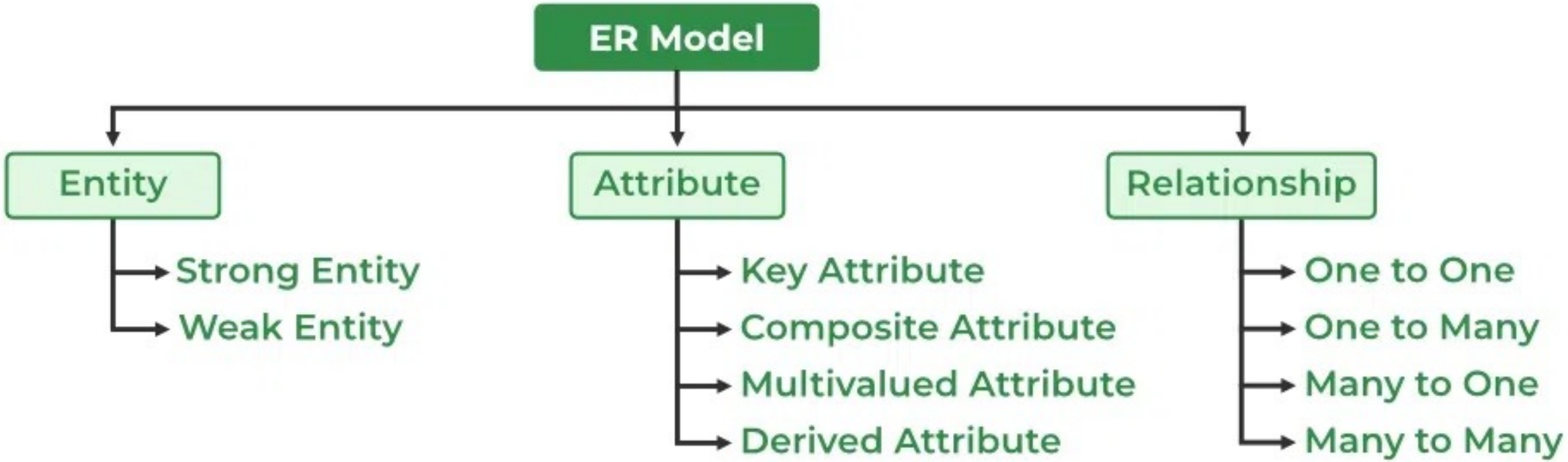
- **ER veri modeli üç temel kavram kullanır:**
  - Varlık kümeleri (entity sets),
  - İlişki kümeleri (relationship sets),
  - Öznitelikler (attributes).

- ER modelinin grafiksel bir gösterimi vardır: ER diyagramı. Bu diyagram, bir veritabanının genel mantıksal yapısını grafiksel olarak ifade eder.

Enterprise schema: Sistemdeki tüm varlıkları ve ilişkileri üst düzeyde gösteren şema.

ER diyagramı: Varlıkların kutularla, ilişkilerin ise çizgiler/diyagram sembolleriyle gösterildiği model.

ER Modeli


ER Model
Entity
Strong Entity
Weak Entity
Attribute
Key Attribute
Composite Attribute
Multivalued Attribute
Derived Attribute
Relationship
One to One
One to Many
Many to One
Many to Many

Tasarım Yaklaşımları

Varlık-İlişki (Entity–Relationship) Modeli

- Bir işletmeyi, **varlıklar (entities)** ve **ilişkiler (relationships)** topluluğu olarak modeller.
  - **Varlık (Entity):** İşletmedeki diğer nesnelerden ayırt edilebilen "şey" veya "nesne".
    - ✓ Bir **öznitelik (attribute)** kümesi ile tanımlanır.
  - **İlişki (Relationship):** Birden fazla varlık arasındaki bağlantıdır.
- Bu yapı, **varlık–ilişki diyagramı (ER diagram)** ile gösterilir.

Normalizasyon Teorisi

- Hangi tasarımların kötü olduğunu **biçimsel olarak tanımlar** ve bunları **test etmeyi** sağlar.

ER modeli: Sistemin nesnelerini (örneğin öğrenci, bölüm, sipariş) ve bunlar arasındaki ilişkileri gösteren temel tasarım yöntemidir.

Normalizasyon: Fazlalıkları azaltmak ve tutarlılığı artırmak için tablo yapılarının kurallara göre düzenlenmesidir.

Varlık Kümeleri (Entity Sets)

- **Varlık (entity),** var olan ve diğer nesnelerden ayırt edilebilen bir nesnedir.
  - Belirli bir kişi, Tek bir öğrenci, Tek bir araba

- **Varlık kümesi (entity set)**, aynı türden ve aynı özellikleri paylaşan varlıkların oluşturduğu kümedir.
  - Öğrenciler kümesi → Ali, Veli, Ayşe, Fatma … tüm öğrenciler
    Arabalar kümesi → Toyota, BMW, Renault… tüm arabalar

- Bir varlık, **özniteliklerden (attributes)** oluşan bir küme ile temsil edilir; yani varlık kümesindeki tüm nesnelerin sahip olduğu tanımlayıcı özelliklerdir.
  - instructor = (ID, name, salary)
  - course = (course_id, title, credits)

- Özniteliklerin bir alt kümesi, varlık kümesinin **birincil anahtarını (primary key)** oluşturur; yani kümedeki her varlığı benzersiz şekilde tanımlar.

Varlık Kümeleri (instructor and student)

76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

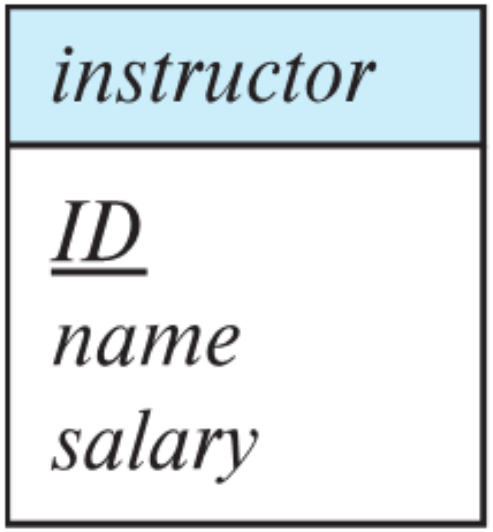
instructor

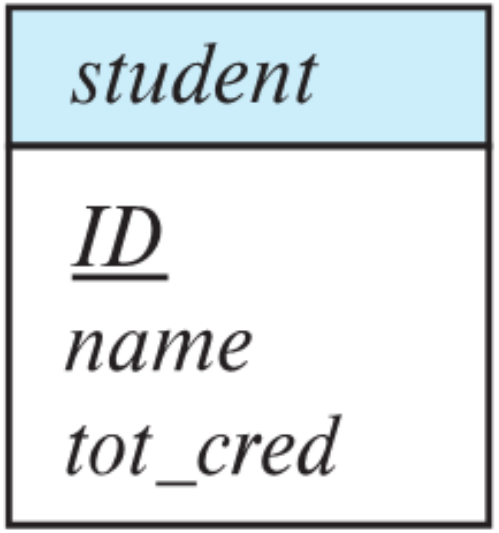
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

ER Diyagramında Varlık Kümelerinin Gösterimi

- Varlık kümeleri (entity sets) ER diyagramlarında aşağıdaki şekilde grafiksel olarak gösterilir:
  - **Dikdörtgenler**, varlık kümelerini temsil eder.
  - **Öznitelikler (attributes)**, bu dikdörtgenlerin içine yazılır.
  - **Altı çizili öznitelikler**, varlık kümesinin **birincil anahtarını (primary key)** gösterir.





Relationship Sets (İlişki Kümeleri)

- Bir ilişki (relationship), birden fazla varlık (entity) arasında kurulan bağlantıdır.
  - Öğrenci Peltier (ID: 44553) bir akademik danışmana sahiptir.
  - Bu danışman da Einstein (ID: 22222) olsun.
  - Bu durumda ilişki:
    - ✓ Peltier — advisor → Einstein

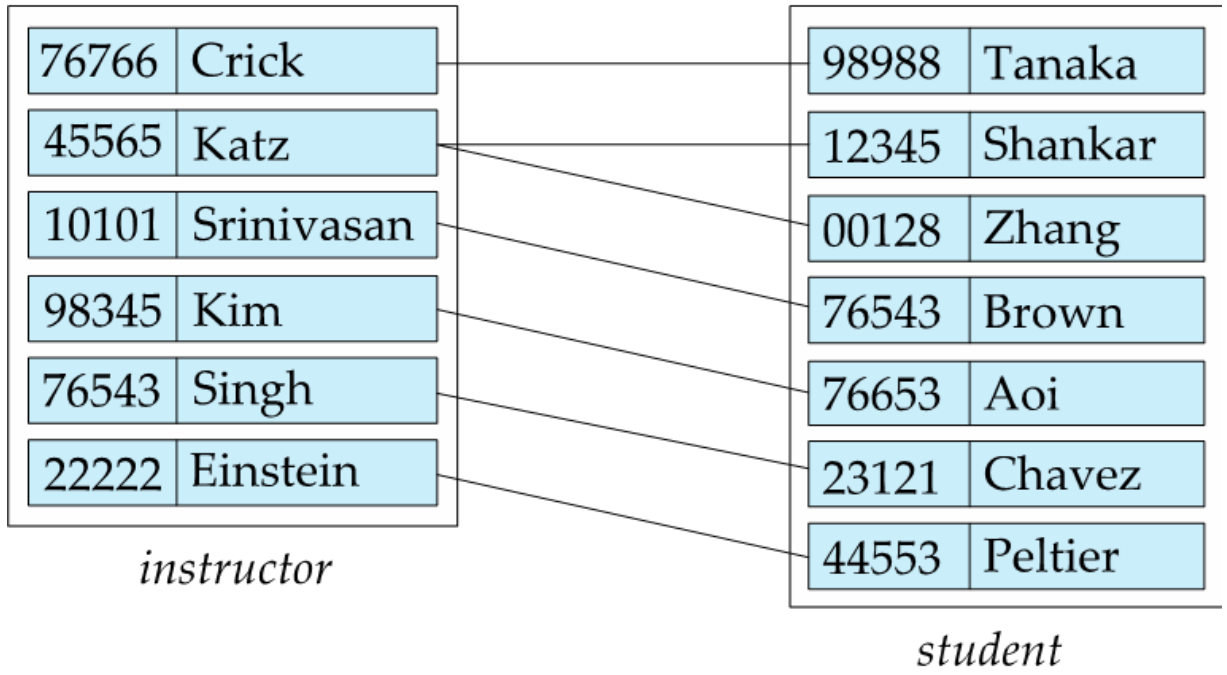
Burada:

- "Peltier" : student (öğrenci) varlığı
- "Einstein" : instructor (öğretim üyesi) varlığı
- "advisor" : ilişki türü

Relationship Sets (İlişki Kümeleri) – Örnek

Advisor (danışman) ilişki kümesi

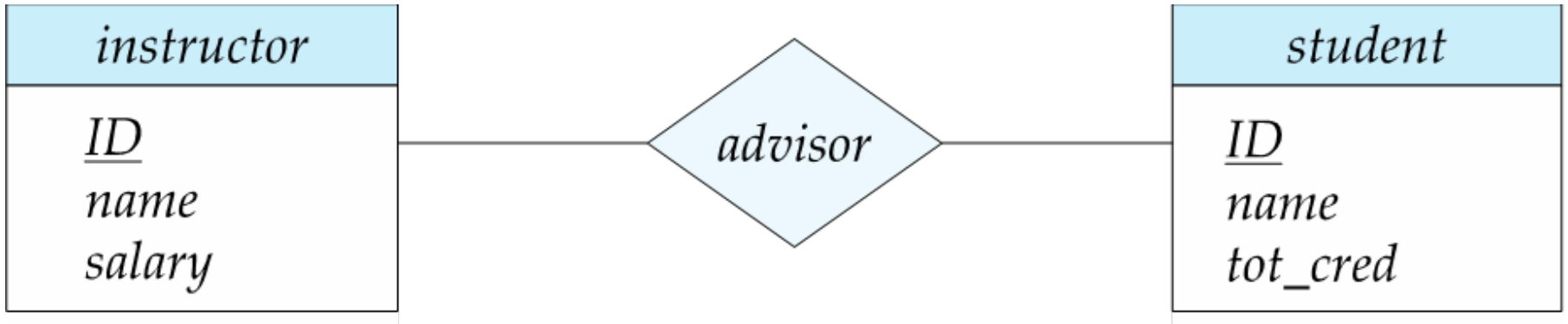
- Bu örnekte advisor adlı ilişki kümesini tanımlıyoruz. Bu ilişki kümesi, öğrenciler ile onların akademik danışmanı olan öğretim üyeleri arasındaki ilişkiyi gösterir.
- Her iki kümedeki varlıklar arasında ilişki olduğunda, ER diyagramında iki varlık arasına bir çizgi çizilir.



- **Bir** öğretim üyesi **birden fazla** öğrenciye danışmanlık yapabilir.
- **Bir** öğrenci sadece **bir** öğretim üyesine danışmana sahiptir.

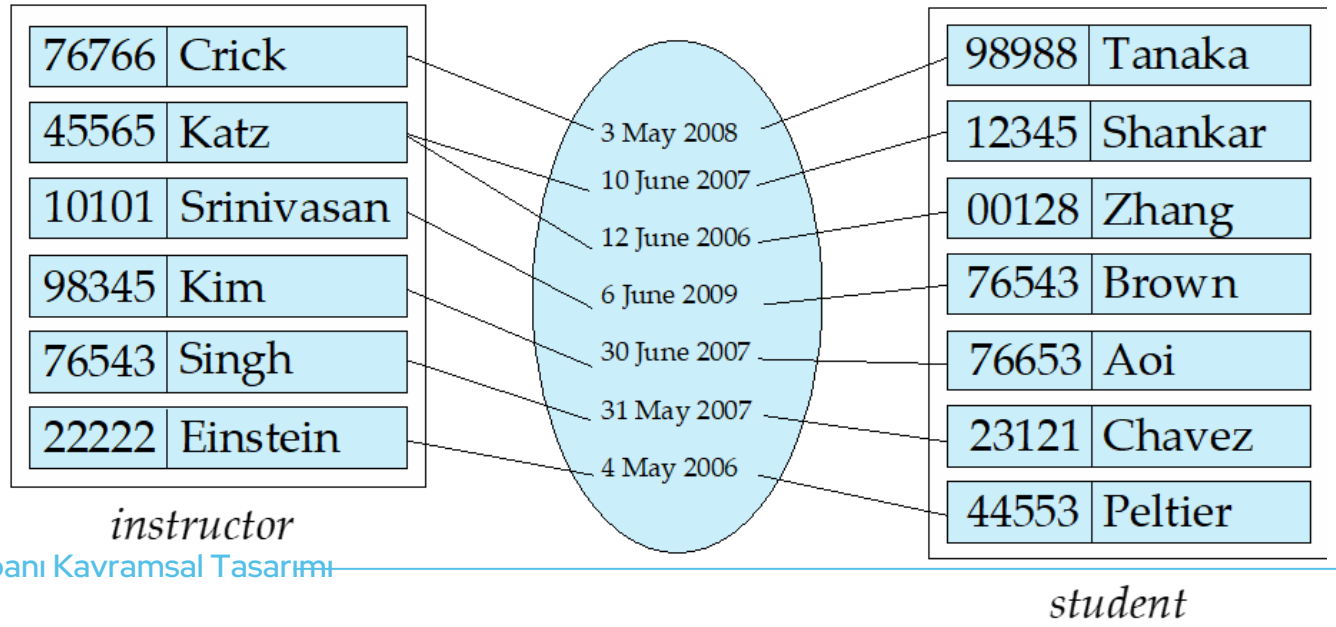
ER Diyagramlarında Relationship Set'lerin Gösterimi

- Elmas (♦) şekilleri relationship set'leri gösterir.
- Yani iki varlık kümesi arasındaki ilişkiyi göstermek için, aralarına bir elmas sembolü yerleştirilir ve bu elmasın içine ilişkinin adı yazılır.



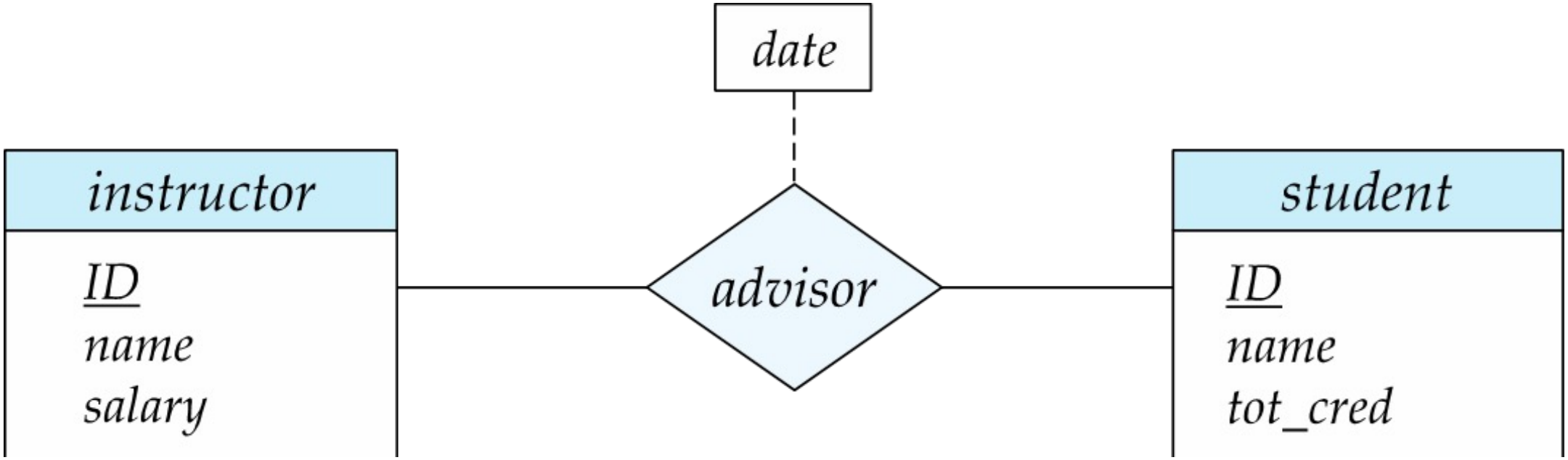
Relationship Sets (İlişki Kümeleri)

- Bir öznitelik (attribute), bir relationship set ile de ilişkilendirilebilir.
  Yani özellikler yalnızca varlık kümelerine (entity set) değil, ilişkilerin kendisine de ait olabilir.
- Örneğin, **advisor** relationship set'i (öğrenci–danışman ilişkisi)
  - instructor ve student varlık kümeleri arasındadır.
  - Bu ilişkiye **date (tarih)** adlı bir öznitelik eklenebilir.
    Bu tarih, öğrencinin danışmanı ile ilişkisinin başladığı günü gösterir.



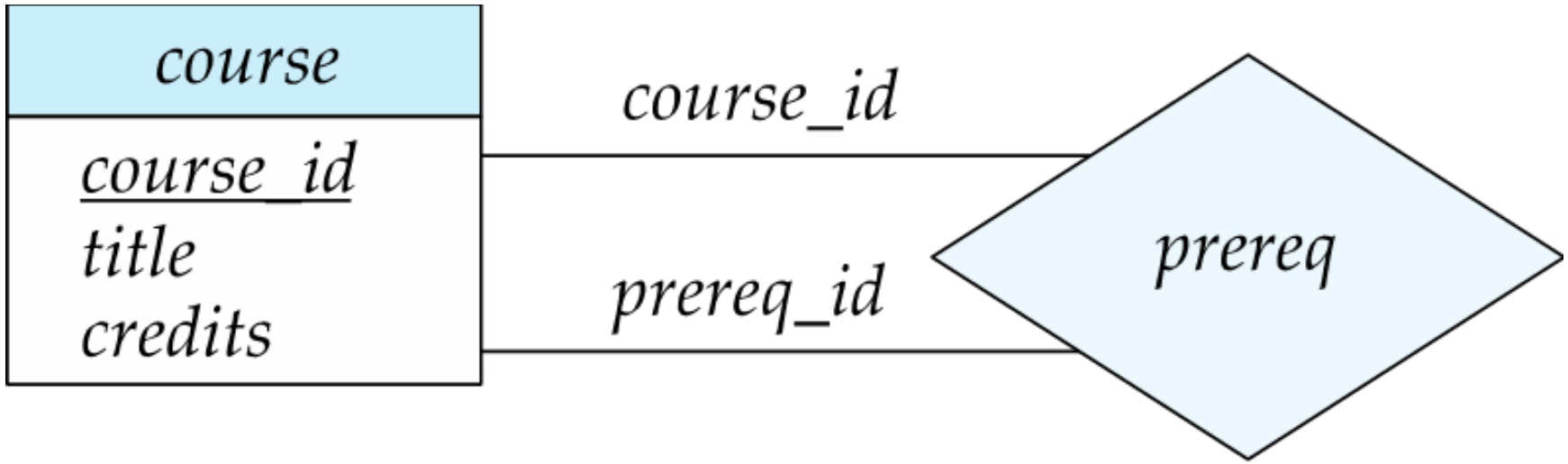
Bu şekilde, her öğrenci–danışman ilişkisine, ilişkiyi tanımlayan ek bilgiler (ör. başlangıç tarihi) eklenebilir.

Nitelikli İlişki Kümeleri



Roller (Roles)

- Bir ilişki kümesindeki (relationship set) varlık kümelerinin farklı olması gerekmez.
  - Yani aynı varlık kümesi bir ilişki içinde birden fazla kez yer alabilir.
- Bir varlık kümesinin her bir örneği, ilişkide bir "rol" oynar.
- "course_id" ve "prereq_id" etiketleri birer **rol** adıdır (roles)



Bir İlişki Kümesinin Derecesi (Degree of a Relationship Set)

1) Binary Relationship (İkili İlişki)

- Bir ilişki iki varlık kümesini içeriyorsa buna binary (derece 2) ilişki denir.
- Veritabanı sistemlerindeki ilişkilerin çoğu ikilidir.
  - student ↔ course
  - department ↔ instructor

2) İkiden Fazla Varlık Kümesi İçeren İlişkiler (Ternary Relationships)

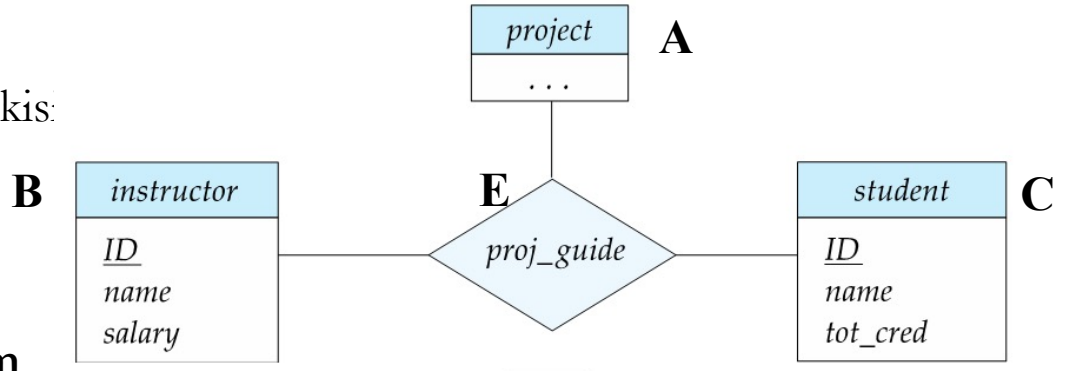
- Üç veya daha fazla varlık kümesi içeren ilişkiler nadirdir.
  - Üçlü İlişki (Ternary Relationship)
    - ✓ "Öğrenciler bir eğitmenin rehberliğinde araştırma projeleri üzerinde çalışırlar."
  - Bu durumda ilişkide üç varlık vardır:
    - ✓ student project instructor

    relationship proj_guide, eğitmen, öğrenci ve proje arasındaki üçlü (ternary) bir ilişkidir.

Non-binary Relationship Sets (İkiden Fazla Varlık İçeren İlişkiler)

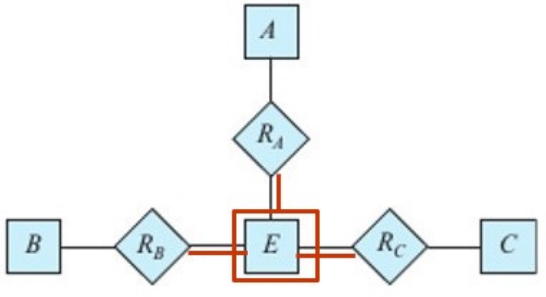
1) Ternary Relationship Örneği (proj_guide)

- Bu üç varlık (Instructor, Student, project) aynı anda proj_guide ilişkis
- "Bir öğrenci, bir projede bir hocanın rehberliğinde çalışır."
- Bu durumda üç varlık birlikte anlamlıdır, yani ilişki ayrılamaz.



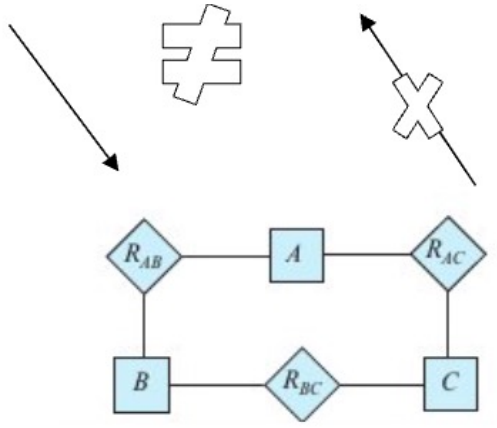
2) Ternary İlişkinin Binary İlişkilere Dönüştürülebileceği Durum

- Bu üçlü ilişki (A, B, C), eğer E varlığı (ilişkiyi temsil eden kutu) aracılığıyla anlamlı biçimde ayrılabiliyorsa:
- A — RA — E — RB — B. ve. E — RC — C
- Bu yapı ternary ile eşdeğer olabilir. Yani üçlü ilişkiyi parçalamak mümkündür.



3) Binary İlişkilere Bölünemeyen Durum (≠ gösterilen)

- Eğer A–B, A–C, B–C ilişkilerini ayrı ayrı kurarsak, asıl üçlü ilişkinin anlamı kaybolur.
- Yani üçlü ilişkiyi parçalamak doğru değildir.
- Bu nedenle o ilişki mutlaka ternary kalmalıdır.



Complex Attributes (Karmaşık Özellikler)

Bir varlığın (entity) özellikleri farklı türlerde olabilir. Bunları ayırmak modelleri doğru kurmak için kritik.

1) Simple ve Composite Attributes

Simple Attribute (Basit özellik): Bölünemez, tek parça bilgi içerir.

- Örnekler: salary, student_id

Composite Attribute (Bileşik özellik): Alt parçalara bölünebilir.

- Örnekler: address → street, city, zip name → first_name, middle_name, last_name

2) Single-Valued ve Multivalued Attributes

Single-valued (Tek değerli): Bir varlık için tek bir değer alır.

- Örnekler: birth_date, gender, passport_number

Multivalued (Çok değerli): Bir varlık birden fazla değer tutabilir.

- Örnekler: phone_numbers → bir kişinin birden fazla telefonu olabilir.,
  email_addresses (kişinin birden fazla e-postası olabilir),
  skills (çalışanın birden fazla becerisi olabilir), hobbies (birden çok hobi)

Complex Attributes (Karmaşık Özellikler)

3) Derived Attributes (Türetilmiş özellikler): Diğer özelliklerden hesaplanabilen değerlerdir.

- Örnekler: age, date_of_birth'ten hesaplanır.,
  total_price → quantity * unit_price,
  BMI → weight / (height²)
- Derived attribute genelde saklanmaz, ihtiyaç olduğunda hesaplanır.

4) Domain (Özellik Alanı): Bir özelliğin alabileceği izinli değerler kümesidir.

- Örnekler: gender → {Male, Female, Other},
  day_of_week → {Mon, Tue, Wed, Thu, Fri, Sat, Sun},
  grade → {AA, BA, BB, CB, CC, DD, FF},

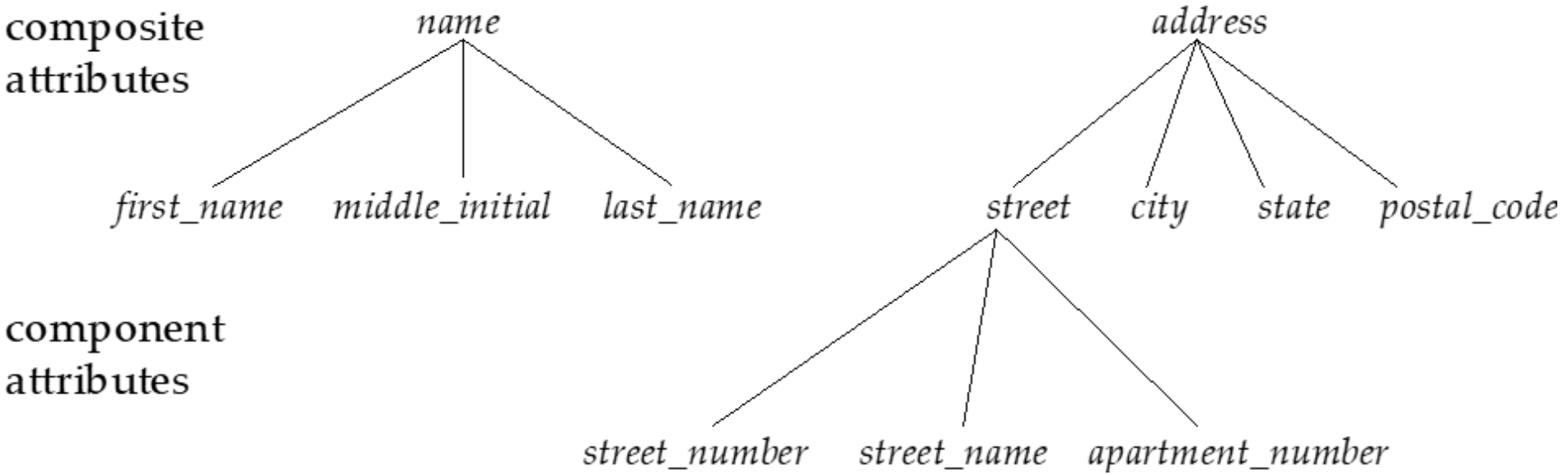
- Domain, veri doğruluğu ve geçerliliği için önemlidir.

Complex Attributes (Karmaşık Özellikler)

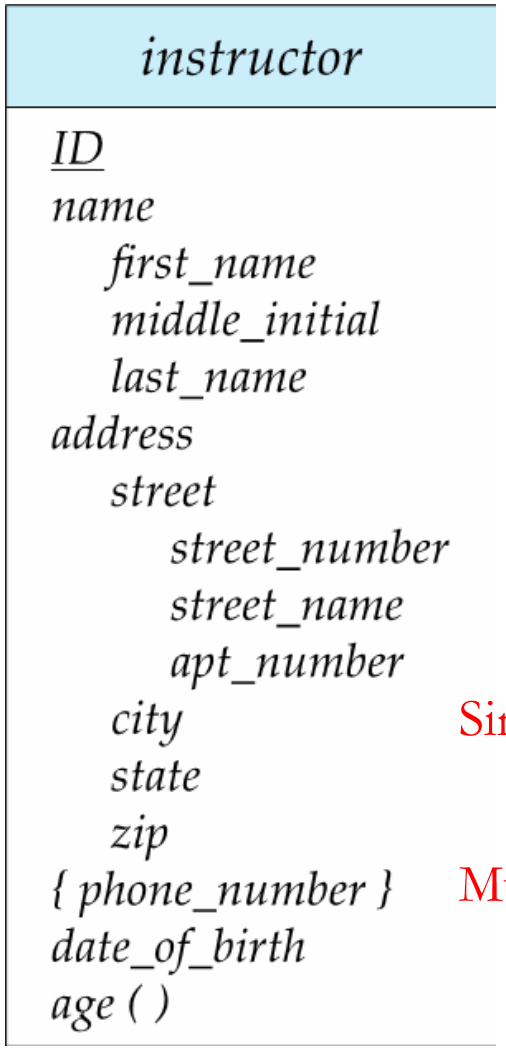
Attribute Türü	Açıklama	Örnek
Simple (Basit)	Bölünemez, tek parça bilgidir.	**salary**, gender, color, student_id
Composite (Bileşik)	Alt bileşenlere ayrılabilir.	address → (city, street, zip)
Single-valued	Tek bir değer alır.	email, national_id
Multivalued	Birden fazla değer alabilir.	phone_numbers
Derived	Diğer attribute'lardan hesaplanan değer.	age (date_of_birth'tan hesaplanır), total_price = quantity × unit_price
Domain	Bir attribute için izin verilen değer kümesi.	gender ∈ {male, female, other}

Composite Attributes (Bileşik Özellikler)

- Bileşik nitelikler, nitelikleri alt parçalara (diğer niteliklere) ayırmamızı sağlar.



ER Diyagramında Karmaşık Niteliklerin Temsili



Composite Attribute

Single Attribute (Primary Key)

Single valued Attribute

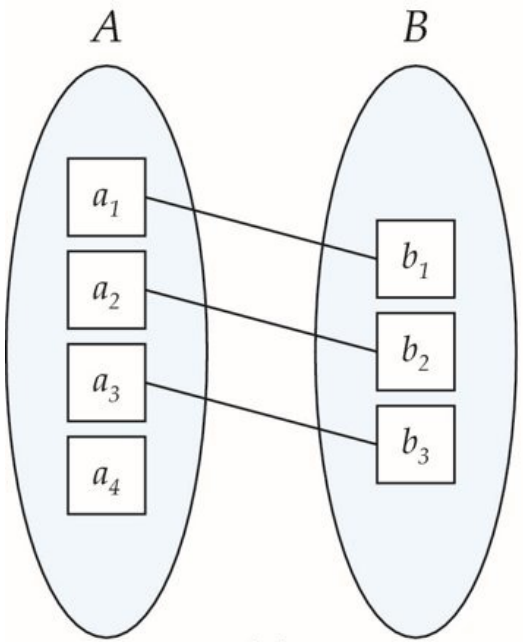
Multivalued Attribute

Derived Attributes

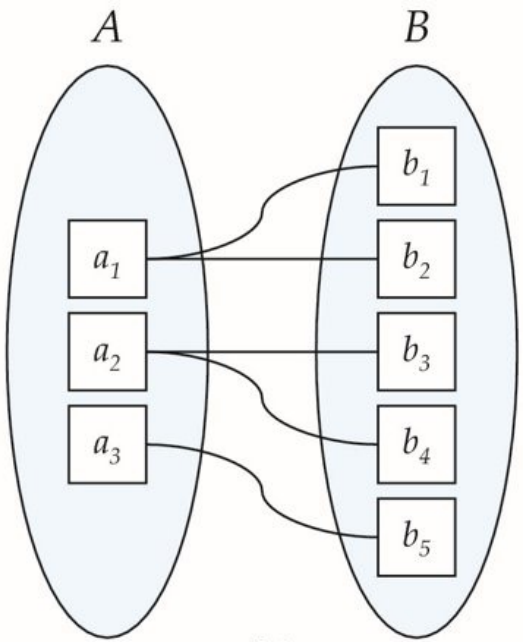
Mapping Cardinality Constraints (Eşleme Kardinalite Kısıtları)

- Bir ilişki kümesi aracılığıyla bir varlığın kaç farklı varlıkla ilişkilendirilebileceğini ifade eder.
- En çok ikili (binary) ilişki kümelerini açıklarken kullanışlıdır.
- Bir binary ilişki kümesi için mapping cardinality (eşleme kardinalitesi) aşağıdaki türlerden biri olmalıdır:
  - One to one (1–1)
  - One to many (1–N)
  - Many to one (N–1)
  - Many to many (N–M)

Mapping Cardinality Constraints (Eşleme Kardinalite Kısıtları)



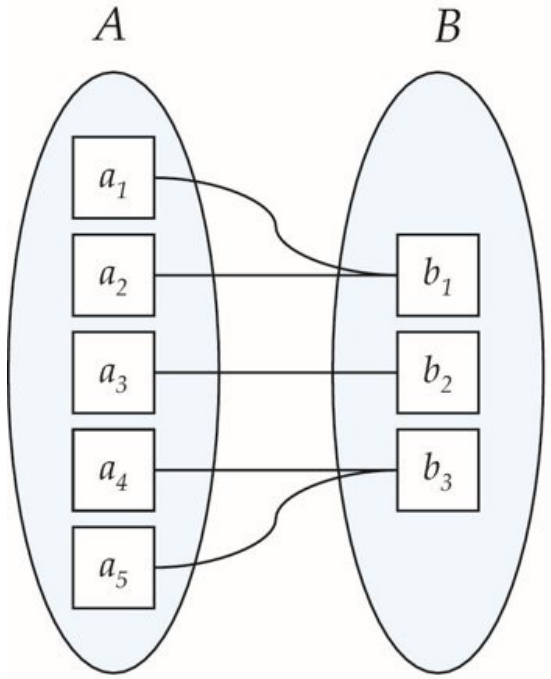
One to one



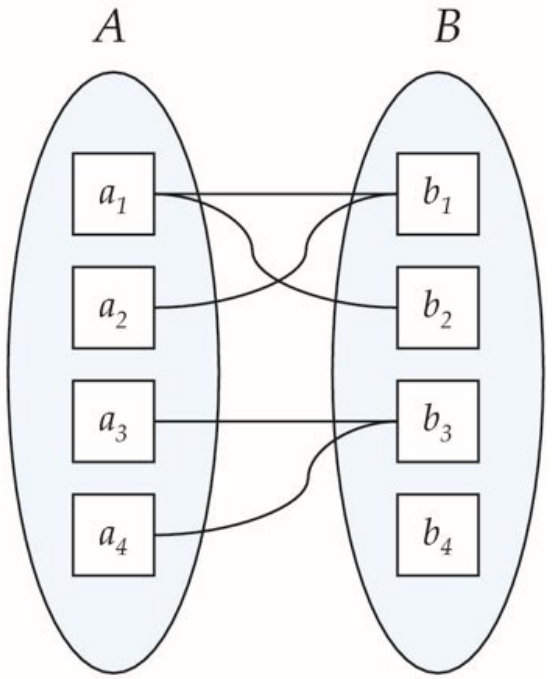
One to many

Not: A ve B'deki bazı elemanlar diğer kümedeki hiçbir elemana eşlenmeyebilir.

Mapping Cardinality Constraints (Eşleme Kardinalite Kısıtları)



Many to one



Many to many

Not: A ve B'deki bazı elemanlar diğer kümedeki hiçbir elemana eşlenmeyebilir.

ER Diyagramında Kardinalite Kısıtlarının Gösterimi

- Kardinalite kısıtlarını, ilişki ile varlık arasında çizilen çizgilerle ifade ederiz:
  - Yönlü çizgi (→) : "one" (bir) anlamına gelir.
  - Yönsüz düz çizgi (—) : "many" (çok) anlamına gelir.
- **Örnek:** instructor – student arasında **One-to-One** İlişki

  - Bir öğrenci (student), advisor ilişkisi üzerinden en fazla bir instructor ile ilişkilendirilebilir.
  - Bu nedenle ilişki 1–1 (one-to-one) olarak gösterilir.

One-to-Many Relationship

İlişkinin Mantığı: Bu ilişki, bir öğretim üyesinin birden fazla öğrencisi olabileceğini, ama bir öğrencinin sadece bir öğretim üyesine bağlı olabileceğini ifade eder.

- Instructor (1 tarafı): Bir instructor'ın, hiç öğrencisi olmayabilir veya birçok öğrencisi olabilir. (0, 1 veya daha fazla student)
- Student (N tarafı): Her öğrenci ise, danışman olarak yalnızca bir instructor seçebilir. (en fazla 1 instructor)

Instructor → Student : 1 → N

Many-to-One Relationship

İlişkinin Mantığı: Bu ilişki, bir öğrencinin birden fazla instructor ile ilişkili olabileceğini, ama bir instructor'ın en fazla bir öğrenci ile ilişkili olabileceğini ifade eder.

- Instructor (1 tarafı): Bir instructor, advisor ilişkisi üzerinden en fazla 1 öğrenci ile ilişkilendirilebilir.
- Student (N tarafı): Bir öğrenci ise advisor ilişkisi üzerinden hiç, bir veya birçok instructor ile ilişkili olabilir. (0, 1 veya daha fazla instructor)

Instructor → Student : 1 → N değil, 1 ← N

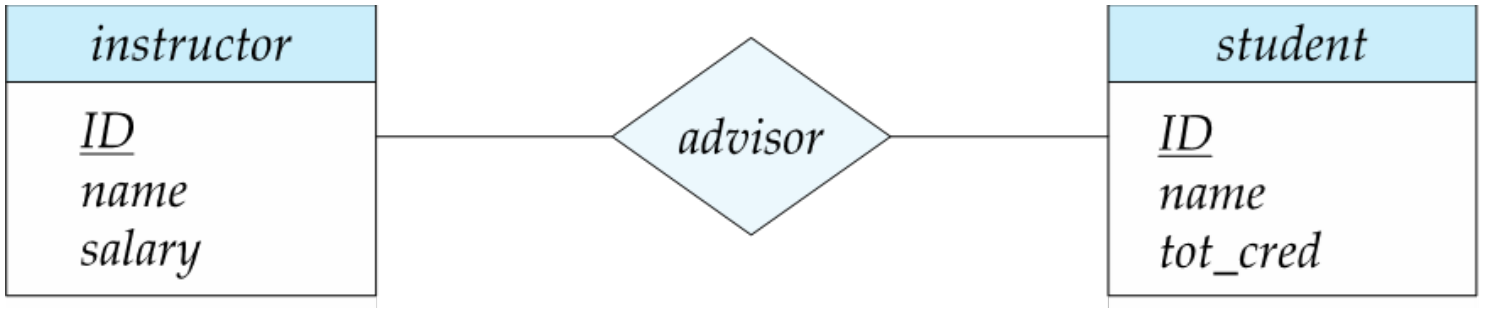
(Yani student tarafından instructor'a doğru many-to-one)

Student → Instructor : N → 1

Many-to-Many Relationship

İlişkinin Mantığı: Bu ilişki, hem instructor hem de student tarafında birden fazla eşleşmenin olabileceğini ifade eder.

- Instructor tarafı: Bir instructor, advisor ilişkisi üzerinden birçok öğrenciyle ilişkilendirilebilir (0, 1 veya daha fazla öğrenci).
- Student tarafı: Bir öğrenci, advisor ilişkisi üzerinden birçok instructor ile ilişkilendirilebilir (0, 1 veya daha fazla instructor).



Instructor → Student : N

Student → Instructor : N

Bu nedenle ilişki N–N (many-to-many)'dir.

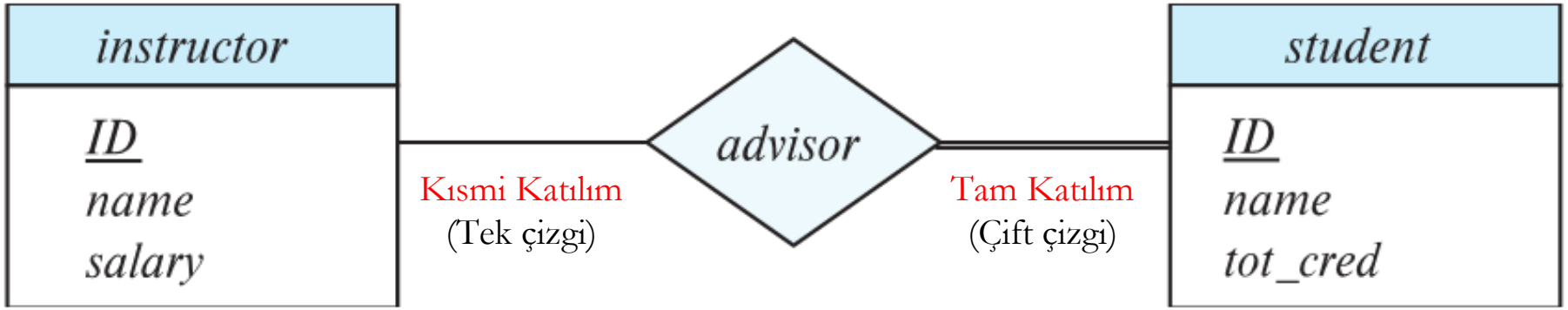
Total and Partial Participation (Tam ve Kısmi Katılım)

Total Participation: Bir varlık kümesindeki her bir varlık, ilişki kümesinde en az bir ilişkiye katılmak zorundadır.

- Her öğrencinin ilişkili bir eğitmeni olmalıdır.

Partial Participation: Bir varlık kümesindeki bazı varlıklar ilişkiye katılır, bazıları katılmayabilir.

- Eğitmenin danışman ilişkisine katılımı kısmidir.
  Eğitmenin danışman olduğu öğrencisi olmayabilir.



Karmaşık Kısıtlamaları İfade Etmek İçin Notasyon

Minimum–Maximum Cardinality (l..h Notasyonu): Bir ilişki çizgisinin yanında l..h şeklinde iki sayı yazılır: l = minimum katılım, h = maksimum katılım

- Bu gösterim hem kardinaliteyi (ilişki türü) hem de katılım zorunluluğunu (total/partial) ifade eder.

Minimum Değer (l)

- l = 0 → Katılım zorunlu değil (Partial participation)

Örnek: 0..* → Bu varlık ilişkide olabilir veya olmayabilir.

- l = 1 → Katılım zorunlu (Total participation)

Örnek: 1..1 → Bu varlık mutlaka ilişkide olmalı.

Maksimum Değer (h)

- h = 1 → En fazla 1 ilişkiye katılabilir

Örnek: 1..1 → Tam olarak 1 tane. h = * → Üst sınır yok, birden fazla olabilir

Örnek: 0..* → Sınırsız sayıda olabilir.

Karmaşık Kısıtlamaları İfade Etmek İçin Notasyon – Örnek

Instructor tarafı: 0..*

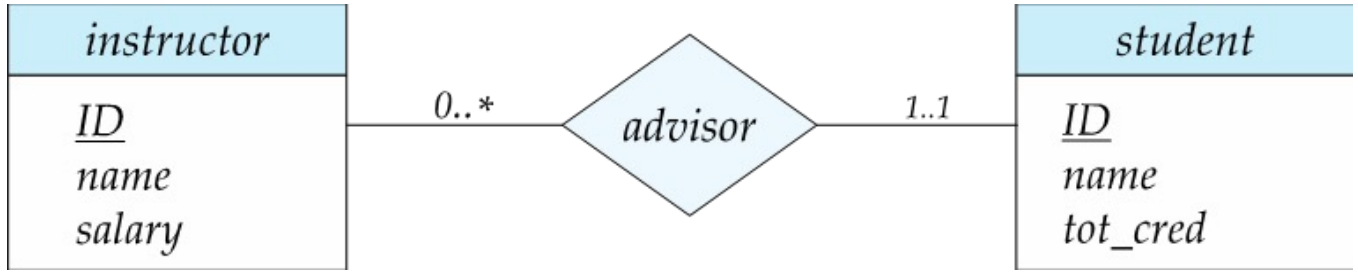
Bir instructor:

- 0 öğrenciye danışmanlık yapabilir
- 1 öğrenciye danışmanlık yapabilir
- Birçok öğrenciye danışmanlık yapabilir
- Katılım zorunlu değil (partial)
- Maksimum bir sınırı yok → * (çok)

Student tarafı: 1..1

Her student:

- Mutlaka bir advisor'a sahip olmalı
- Sadece tek bir advisor'ı olabilir
- Katılım zorunlu (total)
- Maksimum = 1 → aynı anda iki danışmanı olamaz.



- Eğitmen 0 veya daha fazla öğrenciye danışmanlık yapabilir. Bir öğrencinin 1 danışmanı olmalıdır; birden fazla danışmanı olamaz. 1-N ilişkisi gösterilmektedir.

Cardinality Constraints on Ternary Relationships

- Bir üçlü (ternary) ilişki, A–B–C gibi üç varlığı aynı anda bağlayan ilişkidir.
- Örnek: proj_guide (project – instructor – student)
  → Bir proje için bir öğrenciye bir eğitmen atanması gibi.
- Ternary ilişkide en fazla bir tane ok kullanılabilir:

Örnek: proj_guide ilişkisinde ok instructor'a doğruysa:

- Her student bir proje için en fazla bir instructor tarafından yönetilir. Yani bir öğrencinin bir projede birden fazla danışmanı olamaz.

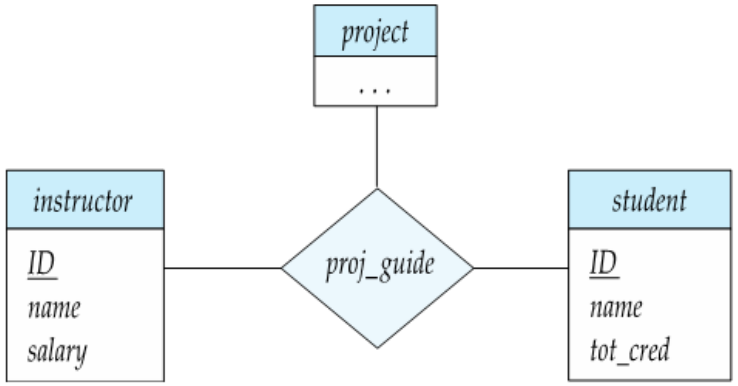
Birden fazla ok kullanınca iki farklı yorum oluşur:

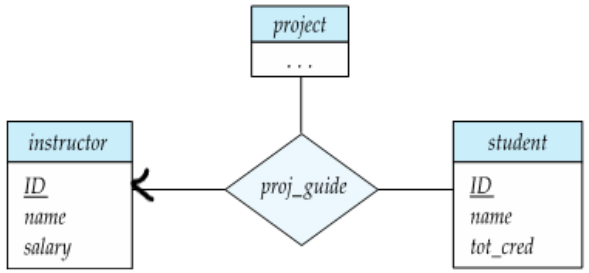
- Varsayalım R = (A, B, C) üçlü ilişkisi ve ok hem B'ye hem C'ye çizildi.
- Anlam 1 (A → B ve A → C) Her A varlığı, benzersiz bir B ve benzersiz bir C ile eşleşir.
- Anlam 2 ((A,B) → C ve (A,C) → B) Her (A,B) çifti benzersiz bir C ile eşleşir, ve her (A,C) çifti benzersiz bir B ile eşleşir.
- Bu iki anlam birbirinden tamamen farklıdır → o yüzden çoklu ok kullanılamaz.

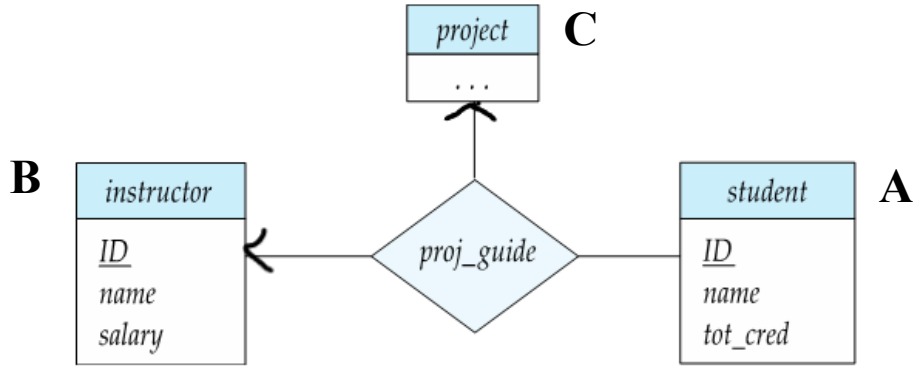
Cardinality Constraints on Ternary Relationships

Birden fazla ok kullanınca iki farklı yorum oluşur:

- Varsayalım R = (A, B, C) üçlü ilişkisi ve ok hem B'ye hem C'ye çizildi.
- Anlam 1 (A → B ve A → C) Her A varlığı, benzersiz bir B ve benzersiz bir C ile eşleşir.
- Anlam 2 ((A,B) → C ve (A,C) → B) Her (A,B) çifti benzersiz bir C ile eşleşir, ve her (A,C) çifti benzersiz bir B ile eşleşir.
- Bu iki anlam birbirinden tamamen farklıdır → o yüzden çoklu ok kullanılamaz.







Cardinality Constraints on Ternary Relationships

- Eğer çoklu ok gerektiren bir durumu ifade etmek istiyorsak non-binary relationship (ternary) ilişkiyi bir entity'ye dönüştürürüz.
  - Bu işleme relationship → entity dönüşümü denir.
- proj_guide bir ilişkiydi. Çoklu kısıt koymak istiyorsak bunu yeni bir varlık olarak çizeriz:
  - proj_guide artık bir entity olur.
  - instructor, student ve project bu entity'ye bağlanır.
  - Bu yöntem, karmaşık kısıtları iki tane ikili ilişkiye (binary) bölerek çözmemizi sağlar.

YILDIZ TEKNİK ÜNİVERSİTESİ

Yasemin Topuz

Yıldız Teknik Üniversitesi

ytopuz@yildiz.edu.tr