YILDIZ TEKNİK ÜNİVERSİTESİ

ER-EER ile Veritabanı Kavramsal Tasarımı ve İlişkisel Şema Dönüşümü

Öğr. Gör. Dr. Yasemin Topuz

Yıldız Teknik Üniversitesi

BLM3041 – Veritabanı Yönetimi

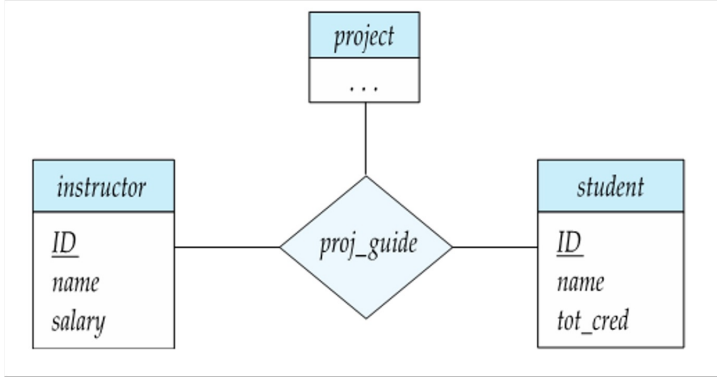
05 Aralık 2025 | YTÜ Bilgisayar Mühendisliği

Neler konuşacağız?

- Kardinalitelerin Eşlenmesi (Cardinality Constraints)
- Primary Keys (Birincil Anahtarlar)
- Varlık Kümelerindeki Gereksiz Niteliklerin Kaldırılması
- Zayıf Varlık Kümeleri (Weak Entity Sets)
- İlişkisel Şemalara Dönüştürme (Reduction to Relation Schemas)
- Extended ER

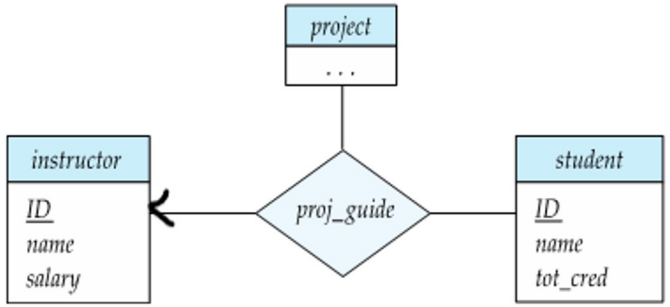
Cardinality Constraints on Ternary Relationships

- 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:

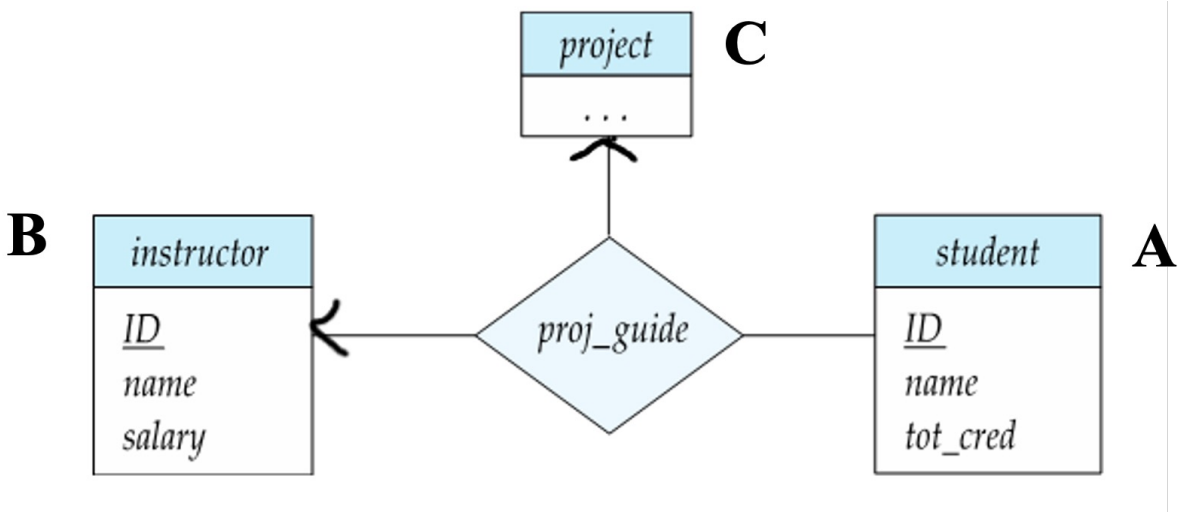
- Bir öğrencinin herhangi bir projede bir danışmanı olabilir.
- Bu öğrenci başka projeler de yapabilir ama aynı hocayla bir proje yapabilir.



Cardinality Constraints on Ternary Relationships

Birden fazla ok kullanınca iki farklı yorum oluşur:

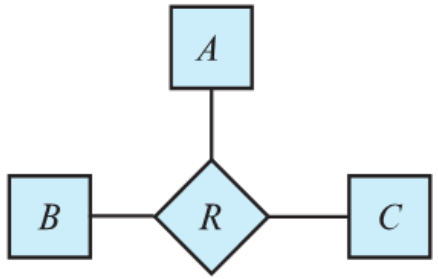
- Varsayalım R = (A, B, C) üçlü ilişkisi ve ok hem B'ye hem C'ye çizildi.
  - Anlam 1 (A → B ve A → C) Her A varlığı, benzersiz bir B ve benzersiz bir C ile eşleşir.
  - Anlam 2 ((A,B) → C ve (A,C) → B) Her (A,B) çifti benzersiz bir C ile eşleşir, ve her (A,C) çifti benzersiz bir B ile eşleşir.
- Bu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

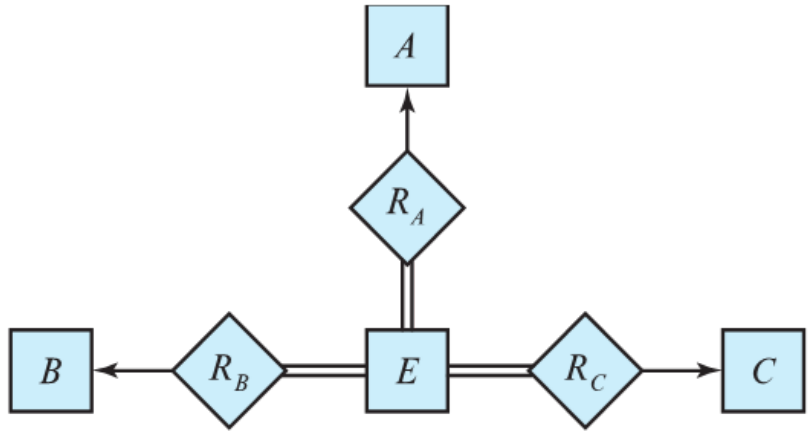


Cardinality Constraints on Ternary Relationships

- 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.





Primary Key

- Primary key, bir veritabanında her bir varlığı (entity) veya ilişkiyi (relationship) birbirinden ayırmak için kullanılan benzersiz tanımlayıcıdır.
- Her tabloya bakıp "bu satırı diğerlerinden ayıran şey ne?" sorusunun cevabıdır.

1) **Entity Sets (Varlık Kümeleri):** Bir entity set içindeki her varlığı ayırt etmek için bir primary key gerekir.

2) **Relationship Sets (İlişki Kümeleri):** Bazı ilişkiler için ilişkideki satırları ayırt edecek bir primary key gerekir.

3) **Weak Entity Sets (Zayıf Varlık Kümeleri):** Zayıf entity kendi başına primary key'e sahip değildir. Bu yüzden kendi partial key'inin + sahibi entity'nin primary key'inin birleşimi onun primary key'i olur.

Primary key for Entity Sets

- Her bir entity zaten birbirinden farklıdır.
  - Gerçek hayatta iki kişi tamamen aynı değildir. Veritabanında da her entity (ör. bir öğrenci, bir ürün, bir çalışan) diğerinden ayrı bir varlık olarak kabul edilir.
- Veritabanı açısından farkları özellikler (attributes) ile göstermemiz gerekir.
  - Bir veritabanı, entity'lerin farklı olduğunu özellik değerlerine bakarak anlar.
  - Örnek: Öğrencinin adı: Ali, Yaşı: 20, ID'si: 101 ("Ali"yi diğer öğrencilerden ayırabiliriz).
- Bir entity'yi benzersiz tanımlayan bir özellik kümesi olmalıdır.
  - Bir entity'nin attribute değerleri, onu tamamen tanıyabilmemizi sağlamalıdır. Yani veritabanında iki entity'nin tüm özellikleri birebir aynı olamaz.
- Primary key = Bir entity'yi diğer tüm entity'lerden ayırmaya yeterli olan attribute kümesidir.
  - Student (ID, name, tot_cred) → Primary key: ID
  - Car (plate_no, model, year) → Primary key: plate_no
  - Person (name, birth_date) → Primary key: YOK (çünkü aynı isimde ve doğumlu farklı kişiler olabilir)

Primary key for Relationship Sets

- Bir relationship set (ilişki kümesi), iki veya daha fazla entity arasındaki bağlantıları tutar. Bu ilişkileri birbirinden ayırmak için ilişkiye katılan entity'lerin primary key'leri kullanılır.

- Bir ilişkiyi benzersiz yapmak için, o ilişkiye katılan entity'lerin primary key'leri bir araya getirilir.
  - Yani, relationship set'in primary key'i: Katılan entity'lerin primary key'lerinin birleşimidir.

- Bir ilişkinin tanımı, o ilişkiye katılan entity'lerin kombinasyonudur.
  - Hangi öğrenci kimin danışmanı? Hangi çalışan hangi departmanda çalışıyor? Hangi müşteri hangi siparişi verdi? Hepsi iki entity'nin birleşimidir.

- Örnek: advisor ilişkisinde:
  - Bir instructor (primary key: instructor.ID). Bir student (primary key: student.ID) bir araya gelir.
  - Bu iki ID birlikte ilişkiyi benzersiz yapar. Relationship primary key = instructor.ID + student.ID

Choice of Primary Key for Binary Relationships

Binary relationship = iki entity arasında ilişki (A — R — B)

Burada primary key seçimi tamamen mapping cardinality'ye bağlıdır.

1) Many-to-Many (M–N) Relationship

- Örnek: Student — Enrolls — Course
  - Bir öğrenci birçok ders alır; bir ders birçok öğrenciye açık.
  - Primary key: Her iki entity'nin primary key'lerinin birleşimi (Composite key)
  - PK = (Student.ID + Course.ID) Bu ikisinin birleşimi ilişkiyi benzersiz yapar.

2) One-to-Many (1–N) Relationship

- Örnek: Instructor ⟵ advisor — Student
  - Bir instructor'ın birçok öğrencisi olabilir. Burada "Many" tarafı Öğrencidir.
  - Primary key: Many tarafının primary key'i. Student.ID zaten ilişkiyi benzersiz tanımlar.

Choice of Primary Key for Binary Relationships

3) Many-to-One (N–1) Relationship

- Örnek: Employee — WorksIn → Department (Employee = Many, Department = One)
  - Birçok çalışan bir departmanda çalışabilir.
  - Primary key: "Many" tarafının primary key'i (Employee.ID)

4) One-to-One (1–1) Relationship

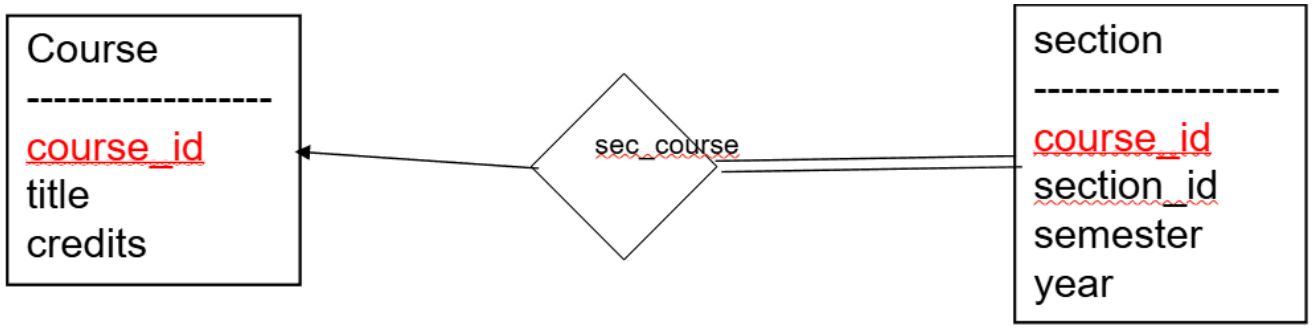
- Örnek: Person — Passport
  - Bir kişinin bir pasaportu, bir pasaportun bir sahibi olabilir. Her iki taraf da tek eşleşiyor.
  - Primary key: Her iki tarafın primary key'i de ilişkiyi benzersiz yapabilir.

Primary key seçimi, relationship set'in kardinalitesine bağlıdır.

Relationship türü	Primary Key
M–N (Many-to-Many)	Her iki tarafın PK'si (birleşim)
1–N (One-to-Many)	Many tarafının PK'si
N–1 (Many-to-One)	Many tarafının PK'si
1–1 (One-to-One)	Her iki tarafın PK'si olabilir

Weak Entity Sets (Zayıf Varlık Kümeleri)

- Bir varlığın tek başına benzersiz şekilde tanımlanamadığı, kimliğinin başka bir varlığa bağlı olduğu durumdur. Bu tür varlıklar weak entity (zayıf varlık) olarak adlandırılır.
- section ve course arasında *sec_course* diye bir ilişki daha oluşturulduğunda;



- Hem entity'nin içinde course_id var, hem ilişki setinde aynı bilgi tekrar ediyor.
- Bu gereksiz (redundant) veri tekrarına yol açar.
  - **Redundancy çözümü:** sec_course ilişkisini tamamen kaldırabiliriz. Çünkü section zaten course_id'yi içeriyor.
- Section hangi course ile ilişkili? → Bu bilgi ilişki olarak değil, sadece attribute olarak kalmış olur. Bu da ER modellemesinde pek tercih edilmez. Çünkü; Varlıklar arası ilişki kaybolur, model daha az anlaşılır olur.

Weak Entity Sets (Zayıf Varlık Kümeleri)

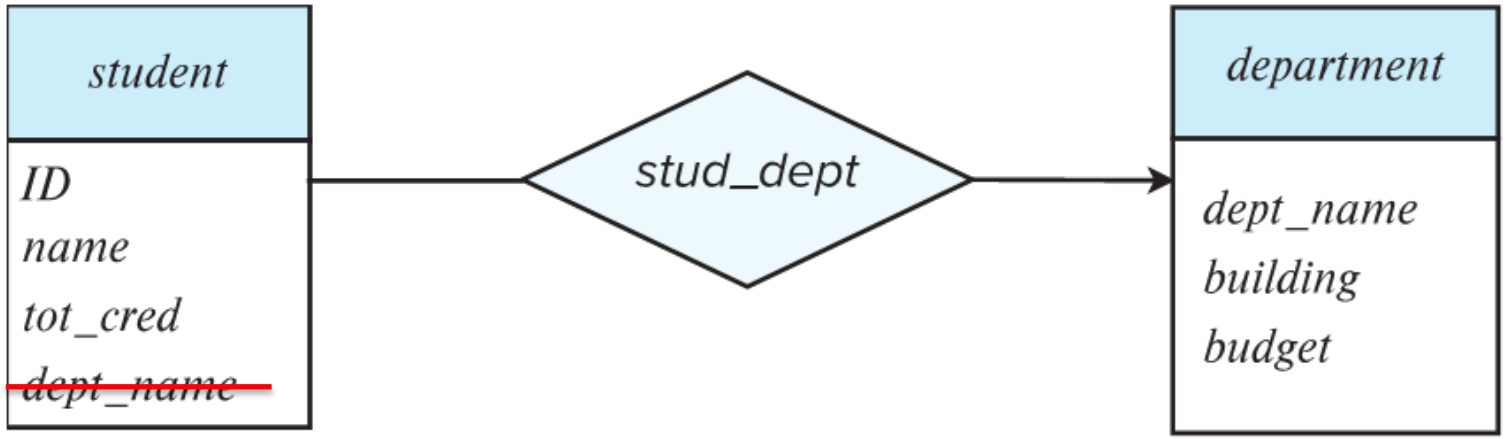
- **Önemli Not:** Section varlığının kimliği tek başına belirlenemiyorsa (mesela sec_id yeterli değilse), o zaman weak entity sayılır ve mutlaka bir owner entity (course) ve ilişki seti gerekir.
  - Owner entity → Course
  - Weak entity → Section
  - Identifying relationship → sec_course
- Bu gereksizliklerle başa çıkmanın alternatif bir yolu, bölüm varlığında course_id özniteliğini depolamamak ve yalnızca kalan section_id, yıl ve dönem özniteliklerini depolamaktır.
- **SORUN:** Section varlık kümesi, belirli bir bölüm varlığını benzersiz bir şekilde tanımlamak için yeterli özniteliğe sahip olmaz.
  - Bu sorunla başa çıkmak için, sec_course ilişkisini, bölüm varlıklarını benzersiz bir şekilde tanımlamak için gereken ek bilgileri (bu durumda course_id) sağlayan özel bir ilişki olarak ele alırız.
- Zayıf bir varlık kümesi, varlığı tanımlayıcı varlık adı verilen başka bir varlığa bağlı olan kümedir.
- Birincil anahtarı zayıf bir varlıkla ilişkilendirmek yerine, zayıf bir varlığı benzersiz bir şekilde tanımlamak için tanımlayıcı varlığı ve ayırıcı adı verilen ek öznitelikleri kullanırız.

Expressing Weak Entity Sets

- E-R diyagramlarında, zayıf bir varlık kümesi çift dikdörtgenle gösterilir.
- Zayıf bir varlık kümesinin ayırıcısını kesikli bir çizgiyle vurgularız.
- Zayıf varlık kümesini tanımlayıcı güçlü varlık kümesine bağlayan ilişki kümesi çift elmasla gösterilir.
- Bölüm için birincil anahtar – (sec_id, semester, year)

Redundant Attribute

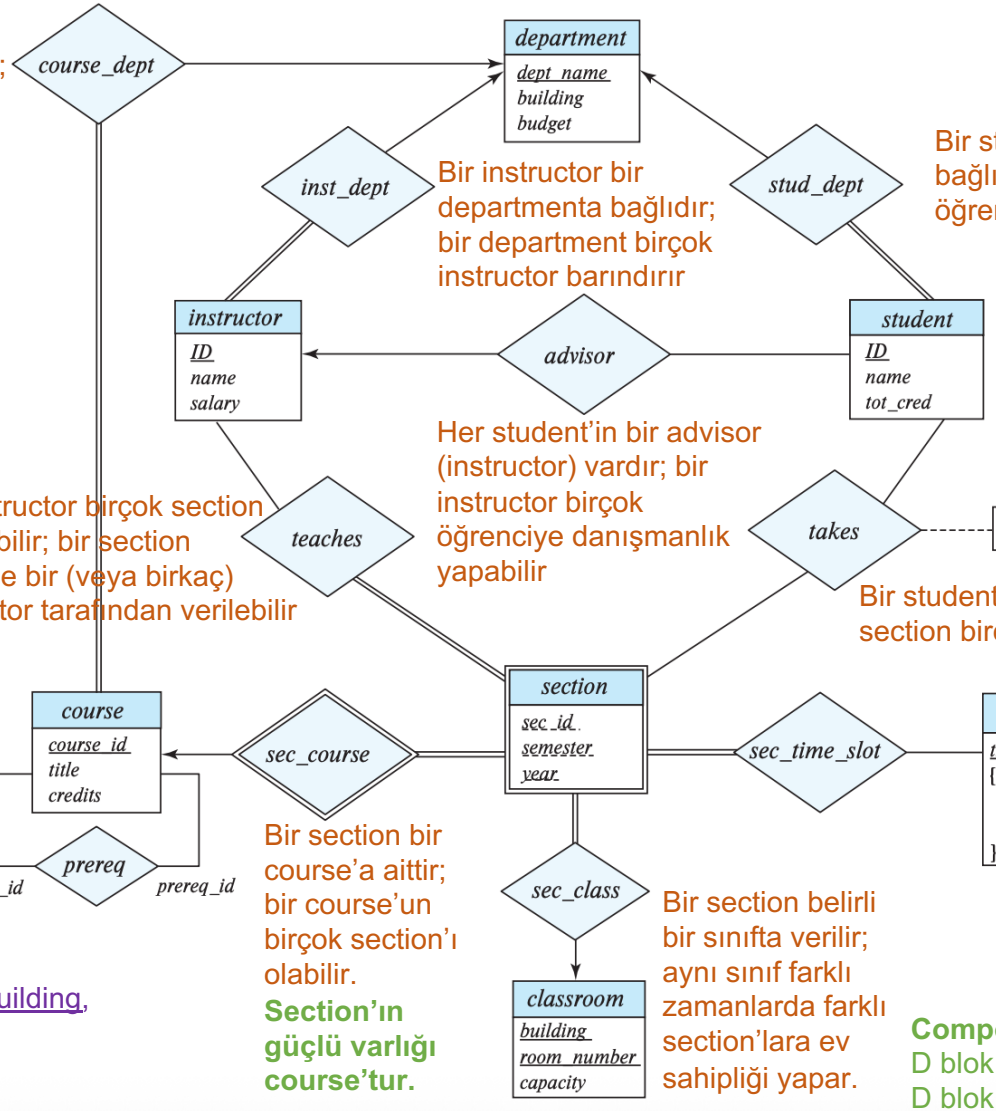
- Bir varlık kümesinde (entity set) bulunan bir özellik, zaten başka bir ilişki (relationship) üzerinden elde edilebiliyorsa, o özellik gereksizdir (redundant).
- Her öğrencinin bir departmanı vardır ⇒ stud_dept ilişkisi ile modellenir.
- Bu ilişki varken; öğrencinin departman bilgisi stud_dept ilişkisinden alınabilir.
  student tablosundaki dept_name bilgisi redundant (gereksiz)



- ***NOT:*** İlişkisel tabloya dönüşüm aşamasında (ER-to-Relational dönüşüm) bazen bu redundant öznitelik yeniden eklenir.

E-R Diagram for a University Enterprise

VARLIKLAR (Entites)
- Department
- İnstructor
- Student
- Course
- Section
- Classroom
- time_slot



Bir course bir department'a aittir; bir department birçok course'a sahip olabilir.

Bir instructor bir departmenta bağlıdır; bir department birçok instructor barındırır

Bir student bir department'a bağlıdır; bir department birçok öğrenciye sahip olabilir.

Her student'in bir advisor (instructor) vardır; bir instructor birçok öğrenciye danışmanlık yapabilir

Bir instructor birçok section öğretebilir; bir section genelde bir (veya birkaç) instructor tarafından verilebilir

ilişki-özniteliği: grade

Bir student birçok section alır; bir section birçok student tarafından alınır

Self-relationship: Bir dersin birçok önkoşulu olabilir; Bir ders birçok dersin önkoşulu olabilir

Bir section bir course'a aittir; bir course'un birçok section'ı olabilir.

Section'ın güçlü varlığı course'tur.

Bir section haftada birden çok time slot'a sahip olabilir; aynı time_slot birçok section için tekrar kullanılabilir.

Bir section belirli bir sınıfta verilir; aynı sınıf farklı zamanlarda farklı section'lara ev sahipliği yapar.

Multi-valued attribute: Dersin farklı oturumları olabilir.

BAĞINTILAR (Relationships)
- course_dept
- inst_dept
- stud_dept
- advisor
- teaches
- sec_course
- takes
- prereq (recursive)
- sec_time_slot
- sec_class

Section (courseid, sec_id, semester, year, building, room_number, time_slot_id)

Composite Primary Key
D blok (bulding), B11 (room_number)
D blok B11 EEF'de de olabilir MF'de de olabilir. Tek başına bir öznitelik Primary Key için yeterli değildir.

Reduction to Relation Schemas
(İlişki Şemalarına İndirgeme)

İlişkisel Şemalara Dönüştürme (Reduction to Relation Schemas)

- Varlık kümeleri (entity sets) ve ilişki kümeleri (relationship sets), veritabanının içeriğini temsil eden **ilişkisel şemalar (relation schemas)** olarak ifade edilebilir.

- Bir E-R diyagramına uygun bir veritabanı, bu diyagramı temsil eden **şema koleksiyonları** şeklinde modellenebilir.

- Her varlık kümesi ve her ilişki kümesi için, o kümeyi temsil eden benzersiz bir **şema** oluşturulur. Bu şemaya ilgili varlık veya ilişki kümesinin adı verilir.

- Her şema, genellikle söz konusu varlık veya ilişkiye ait özellikleri (attributes) karşılayan, **kendine özgü isimlere sahip sütunlar** içerir.

Varlık Kümelerinin Temsili (Representing Entity Sets)

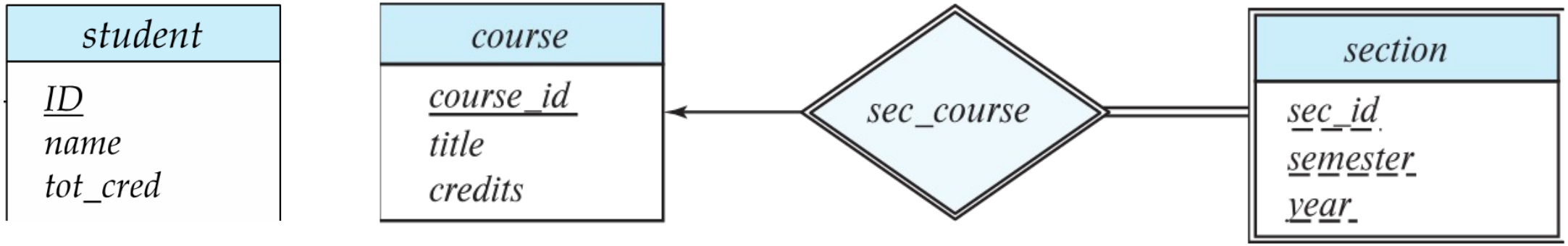
- **Güçlü varlık kümeleri** (strong entity set), sahip oldukları özelliklerle **birebir aynı özelliklere** sahip bir ilişkisel şemaya dönüştürülür.

  student(<u>ID</u>, name, tot_cred)

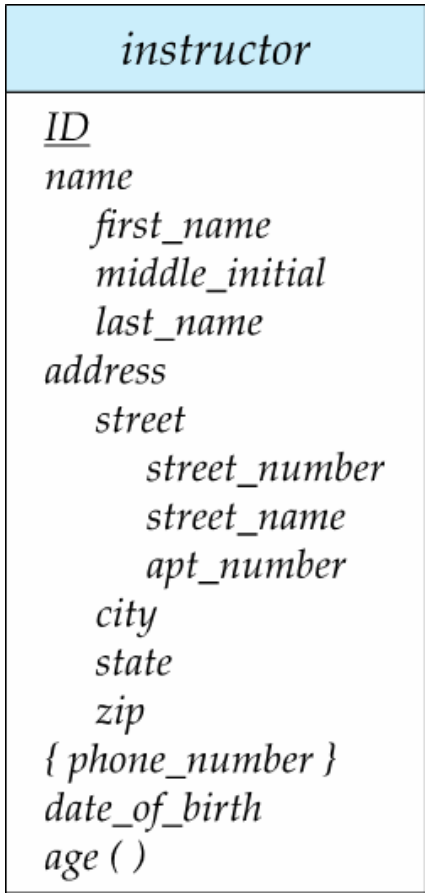
  Course (<u>course_id</u>, title, credits)

- **Zayıf varlık kümeleri** (weak entity set), kendilerini tanımlayan **güçlü varlık kümesinin birincil anahtarını** (primary key) içeren bir tabloya dönüştürülür.

  section(course_id, sec_id, sem, year)
  → Burada **course_id**, güçlü varlık kümesinden gelen anahtar niteliğidir.



Bileşik Öznitelikli (Composite Attributes) Varlık Kümelerinin Temsili



- Composite attributes, her bileşen için ayrı bir öznitelik oluşturularak düzleştirilir (flattening).
- **Örnek:** instructor varlık kümesinde **«name»** bileşik özniteliği

  → first_name, middle_initial, last_name bileşenlerinden oluşuyorsa, ilişkisel şema şu şekilde olur:

  ✓ name_first_name, name_middle_name, name_last_name

  → Eğer bir karışıklık yoksa, önek (prefix) kullanılmayabilir:

  ✓ name_first_name → first_name olarak yazılabilir.
- Multivalued attributes yok sayıldığında, genişletilmiş instructor şeması:

  → instructor(ID,

  first_name, middle_initial, last_name,
  street_number, street_name, apt_number,
  city, state, zip_code,
  date_of_birth)

Not: «age» derived attributes olduğu için ilişkisel şemaya taşınmaz.

Çok Değerli (Multivalued Attributes) Varlık Kümelerinin Temsili

- Bir varlık kümesi E içinde yer alan çok değerli bir öznitelik M, ayrı bir şema (EM) ile temsil edilir.
- **EM şeması**, E varlık kümesinin birincil anahtarını ve çok değerli öznitelik M'ye karşılık gelen bir sütunu içerir.

- **Örnek:** instructor varlık kümesindeki çok değerli **«phone_number»** özniteliği şöyle gösterilir:

  inst_phone(ID, phone_number)

- Çok değerli özniteliğin her bir değeri, EM şemasında ayrı bir satır (tuple) olarak karşılık bulur.

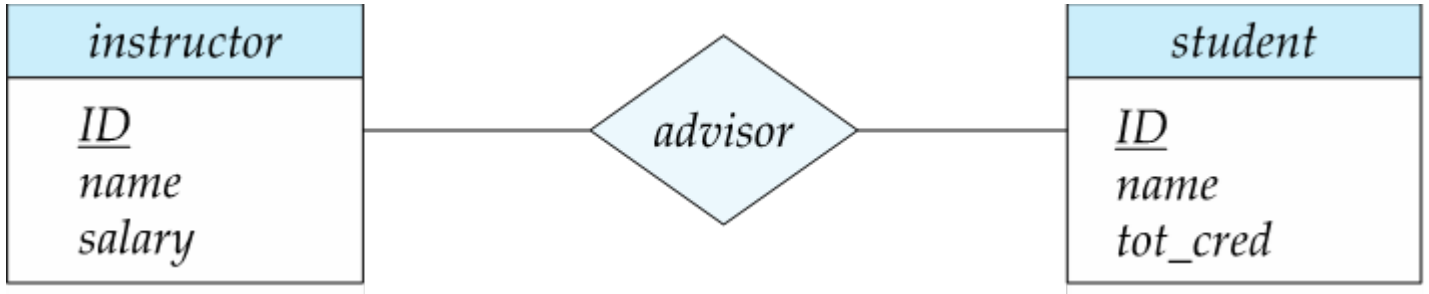
- **Örnek:** Bir instructor varlığının PK'si 22222, telefon numaraları ise 456-7890 ve 123-4567 olsun. Bu durumda iki ayrı satır oluşur:

  (22222, 456-7890)

  (22222, 123-4567)

İlişki Kümelerinin Temsili (Representing Relationship Sets)

- Çoktan-çoğa (many-to-many) bir ilişki kümesi, ilişkiye katılan iki varlık kümesinin birincil anahtarlarını içeren bir şema olarak temsil edilir.
- Ayrıca ilişki kümesine ait tanımlayıcı (descriptive) öznitelikler varsa onlar da bu şemaya eklenir.



- **Örnek:** advisor ilişki kümesi için oluşturulan şema:

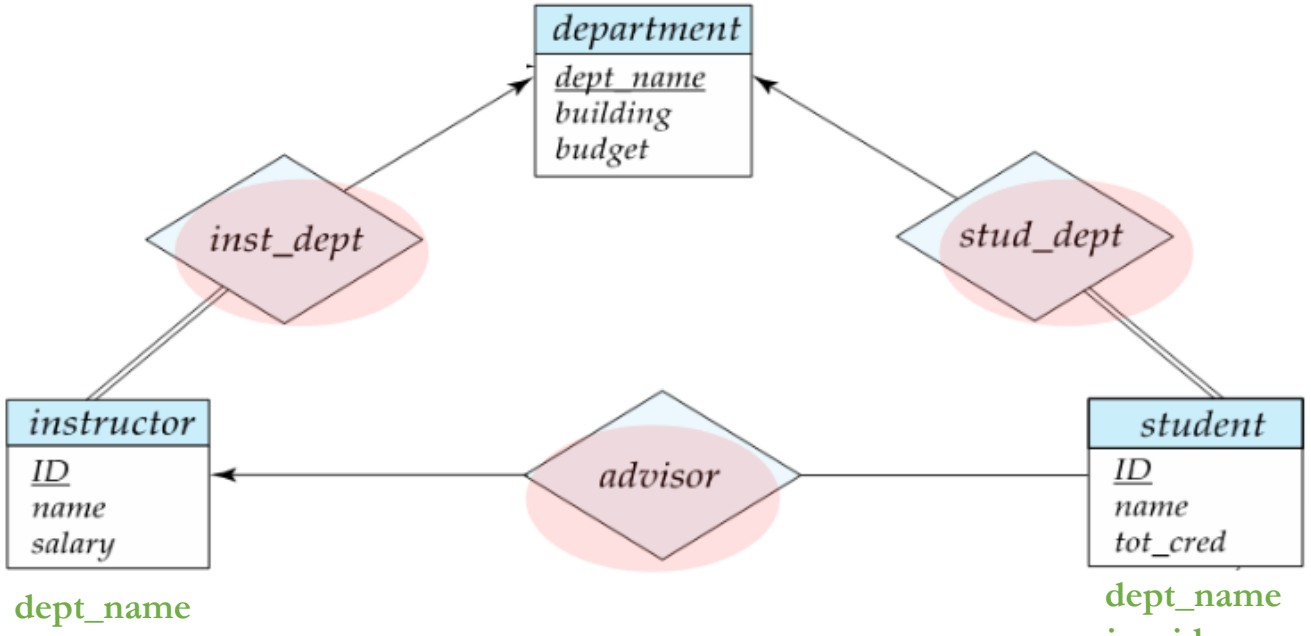
  advisor(s_id, i_id)

  s_id: student varlık kümesinin PK'si

  i_id: instructor varlık kümesinin PK'si

Şema Fazlalığı (Redundancy of Schemas)

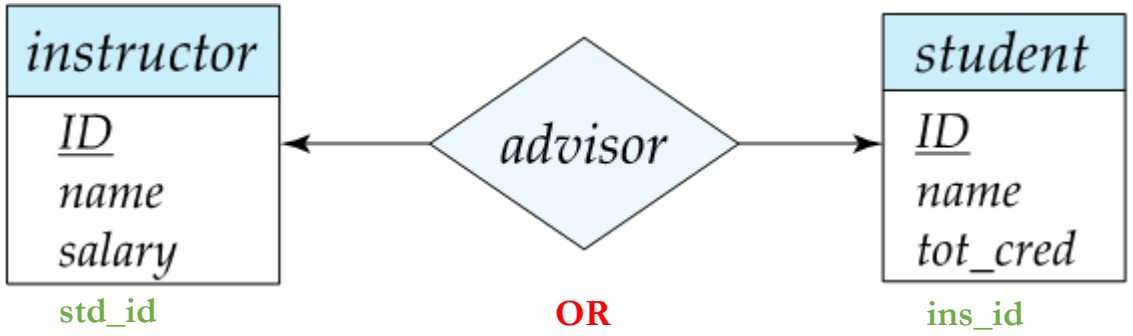
- Many-to-one veya one-to-many ilişki kümeleri, eğer ilişkinin "many" tarafı ilişkiye tamamen dahilse (total participation), ayrı bir ilişki şeması oluşturmaya gerek kalmadan temsil edilebilir.
  - "many" tarafına, "one" tarafının birincil anahtarını içeren bir yabancı anahtar (foreign key) eklenir.
- **Örnek: inst_dept** adlı ilişki için ayrı bir şema oluşturmak yerine, instructor varlık kümesinden türeyen şemaya **dept_name** özniteliğini (foreign key) eklemek yeterlidir.



Şema Fazlalığı (Redundancy of Schemas)

- Eğer katılım **"çok" tarafında kısmi (partial) ise** (önceki slaytta olduğu instructor-student ilişkisinde olduğu gibi) bir ilişkiyi temsil eden şemayı "çok" tarafındaki tabloya yabancı anahtar olarak eklemek NULL değerlere yol açabilir.
- Bu durumda iki seçenek vardır:
  - N-tarafının tam katılımlı olduğu senaryoda olduğu gibi (instructor-department durumla aynı)
  - İlişki için yeni bir tablo oluşturmak (yani ilişkiyi ayrı bir şema olarak modellemek)

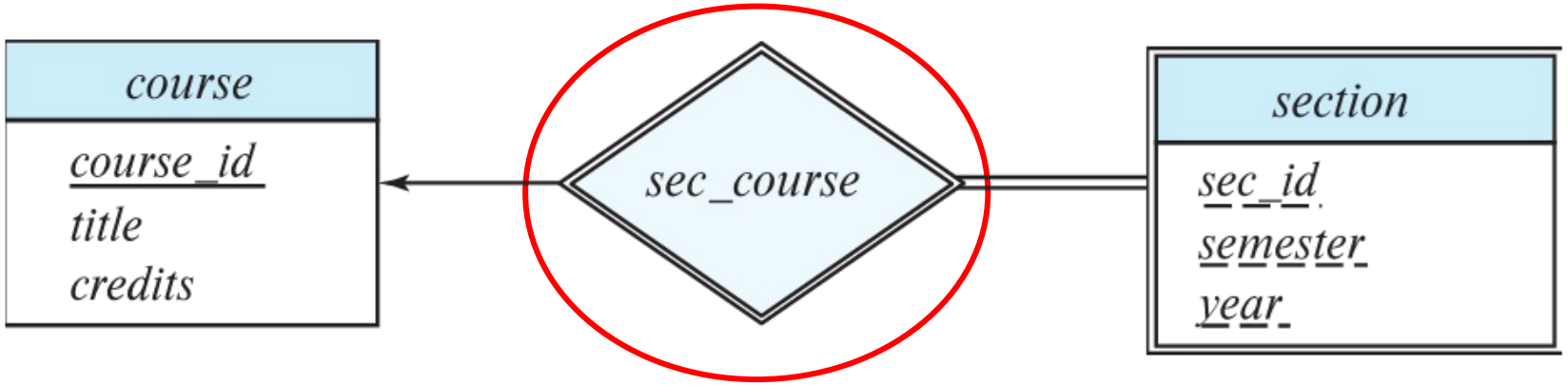
- **Bire bir (1–1) ilişkilerde**, iki taraftan herhangi biri "çok" tarafı gibi davranabilir. Yani, ilişkiyi temsil eden yabancı anahtar her iki tablodan birine eklenebilir.



- Total-total ya da partial-partial için hangi tablonun foreign key alan taraf olduğu önemli değildir.
- Ancak bir taraf TOTAL bir taraf PARTIAL ise PATIAL olan taraf foreing key alan taraf olmalıdır.

Şema Fazlalığı (Redundancy of Schemas)

- Zayıf bir varlık kümesini, onu tanımlayan güçlü varlık kümesine bağlayan ilişki için ayrı bir şema oluşturmak gereksizdir (redundant).
- Çünkü zayıf varlık tablosu zaten güçlü varlığa ait olan tanımlayıcı (identifying) öznitelikleri kendi içinde taşır.



redundant

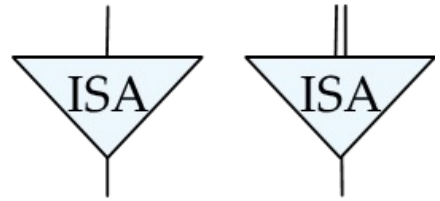
- section (dersin bir şubesi) bir zayıf varlıktır ve güçlü varlık olan course (ders) ile tanımlanır.
- section tablosu zaten şu öznitelikleri içerir: section_id (partial key), course_id (strong entity key → foreign key)
- Bu ilişkiyi ayrıca sec_course adıyla bir ilişki tablosu olarak modellemek gereksizdir.

Extended Entity-Relation Model
E-ER

Extended E-R Features – Specialization (Uzmanlaşma)

- **Üstten alta doğru (top-down) tasarım sürecidir.**
  - Bir varlık kümesi içinde, diğerlerinden farklı özelliklere sahip alt gruplar belirlenir.
  - Bu alt gruplar, üst düzey varlık kümesinde bulunmayan ek özniteliklere veya ilişkilere sahip alt düzey varlık kümeleri hâline gelir.

- ER diyagramında bu yapı, üzerinde ISA yazan bir üçgen ile gösterilir (örneğin: instructor "bir" person'dır → instructor ISA person).



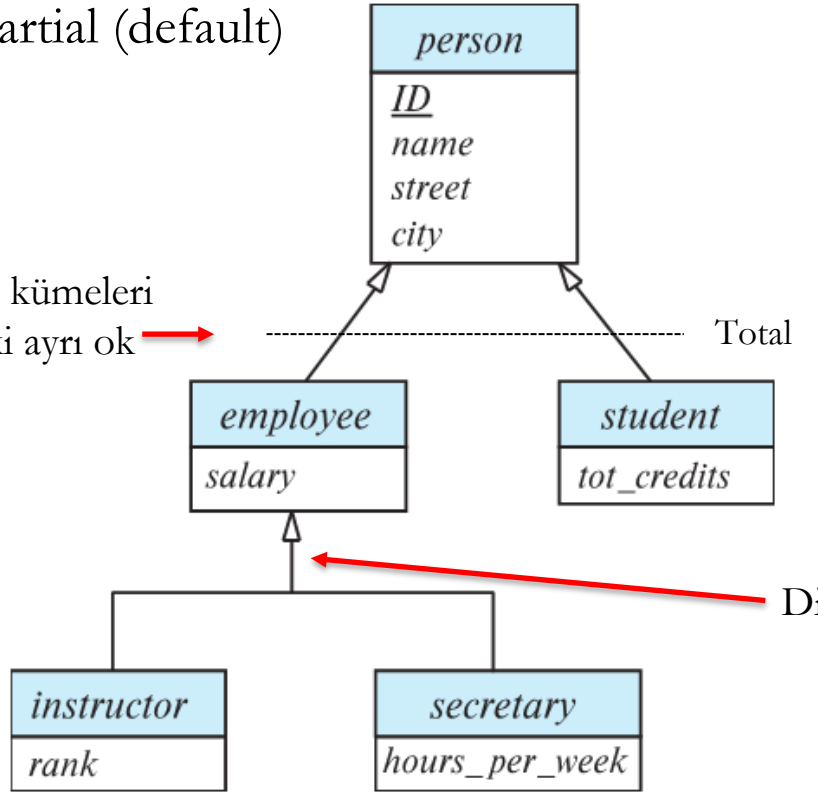
- **Öznitelik kalıtımı (Attribute inheritance):** Alt düzeydeki varlık kümesi, bağlı olduğu üst düzey varlık kümesinin tüm özniteliklerini ve ilişki katılımlarını miras alır.

- **Tek kalıtım vs. çoklu kalıtım (lattice):** Bir alt varlık yalnızca bir üst varlıktan mı türetilir, yoksa birden fazla üst varlık kümesinden mi öznitelik miras alabilir?

  (ER modellerinde çoklu kalıtım bazı durumlarda kafes yapısı—lattice—oluşturur.)

Extended E-R Features – Specialization (Uzmanlaşma) – Örnek

- Overlapping (Örtüşen) – Bir varlık aynı anda birden fazla alt varlık kümesine ait olabilir. employee ve student
- Disjoint (Ayrık) – Bir varlık en fazla bir alt varlık kümesine ait olabilir. instructor ve secretary
- Total ve Partial (default)



Mantık

- Bir varlık birden fazla uzmanlaşmış varlık kümesine mi ait?
  - Bir kişi hem çalışan hem öğrenci olabilir mi?
    Evet -> overlapping
- En fazla bir uzmanlaşmış varlık kümesine mi ait?
  - Bir kişi hem hoca hem sekreter olamaz mı?
    Evet -> disjoint

Representing Specialization via Schemas

Uzmanlaşmanın Şemalar Üzerinden Temsili – Yöntem 1

- Her varlık kümesi için bir şema oluşturulur ve bu şema:
  - Hem kendi yerel özniteliklerini,
  - Hem de üst düzey varlık kümesinden miras alınan tüm öznitelikleri içerir.

schema	attributes
person	ID, name, street, city
student	ID, name, street, city, tot_cred
employee	ID, name, street, city, salary

Dezavantajı

- Bir kişi hem student hem employee ise:
  - name, street ve city gibi ortak öznitelikler bu iki tabloda tekrarlı (redundant) olarak saklanmış olur.
- Bu yöntem sorguları kolaylaştırır fakat veri tekrarına ve güncelleme tutarsızlıklarına yol açabilir.

Representing Specialization via Schemas

Uzmanlaşmanın Şemalar Üzerinden Temsili – Yöntem 2

- **Adım 1:** Üst düzey varlık kümesi için bir şema oluşturulur.

  person(ID, name, street, city)

- **Adım 2:** Her alt düzey varlık kümesi için ayrı bir şema oluşturulur. Bu şemalar, üst düzey varlığın birincil anahtarını ve alt kümeye özgü öznitelikleri içerir.

  student(ID, tot_cred)
  employee(ID, salary)

schema	attributes
person	ID, name, street, city
student	ID, tot_cred
employee	ID, salary

Dezavantajı

- Bir employee hakkında bilgi almak için iki ayrı tabloya erişilmesi gerekir:
  - Üst düzey şema (person)
  - Alt düzey şema (employee)
- Bu yöntem, veri bütünlüğü sağlar ancak sorgulama maliyetini artırır.

Extended E-R Features – Genelleme (Generalization)

- **Aşağıdan yukarıya (bottom-up) tasarım süreci:**
  - Benzer özelliklere sahip birden fazla varlık kümesini bir araya getirerek daha üst düzey bir varlık kümesi oluşturma işlemidir.

- Specialization (uzmanlaşma) ve generalization (genelleme) birbirinin ters işlemidir.

- ER diyagramlarında her ikisi de aynı biçimde gösterilir.

- Bu iki terim uygulamada sık sık birbirinin yerine kullanılabilir.

EER – Bütünlük (Tamlık) Kısıtı — Completeness Constraint

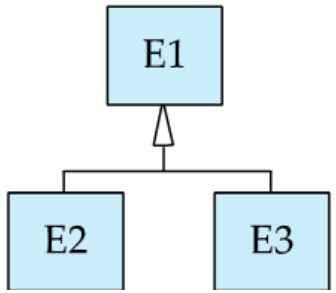
- Completeness constraint, üst düzey bir varlık kümesindeki bir varlığın, genelleme içinde yer alan alt düzey varlık kümelerinden en az birine ait olup olmaması gerektiğini belirtir.
  - **total:** Bir varlık mutlaka alt düzey varlık kümelerinden birine ait olmalıdır.
  - **partial:** Bir varlık alt düzey varlık kümelerinden hiçbirine ait olmak zorunda değildir.

- Her zaman aşağıdaki dört durumdan biri geçerlidir:
  - partial–overlapping
  - partial–disjoint
  - total–overlapping
  - total–disjoint

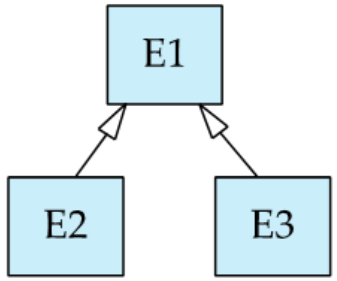
- Partial (kısmi) genellemede varsayılan davranıştır.
  - Üst düzey varlık kümesindeki bir varlığın alt düzey kümelere ait olması zorunlu değildir.

Bütünlük Kısıtı — Completeness Constraint

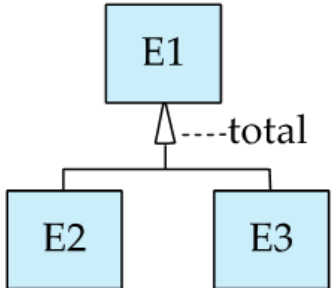
- Total genelleme belirtmek için, ER diyagramında total anahtar kelimesi eklenir ve ilgili alt kümelere doğru kesikli bir çizgiyle bağlanır:
  - disjoint bir genelleme ise çizgi tek bir oka,
  - overlapping bir genelleme ise çizgi birden fazla oka bağlanır.



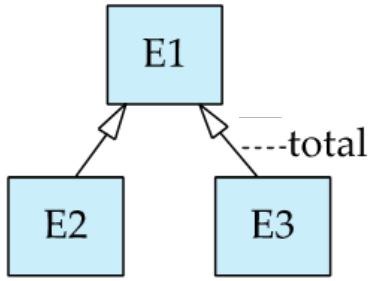
Partial-Disjoint



Partial-Overlapping



Total-Disjoint



Total-Overlapping

YILDIZ TEKNİK ÜNİVERSİTESİ

Yasemin Topuz

Yıldız Teknik Üniversitesi

ytopuz@yildiz.edu.tr