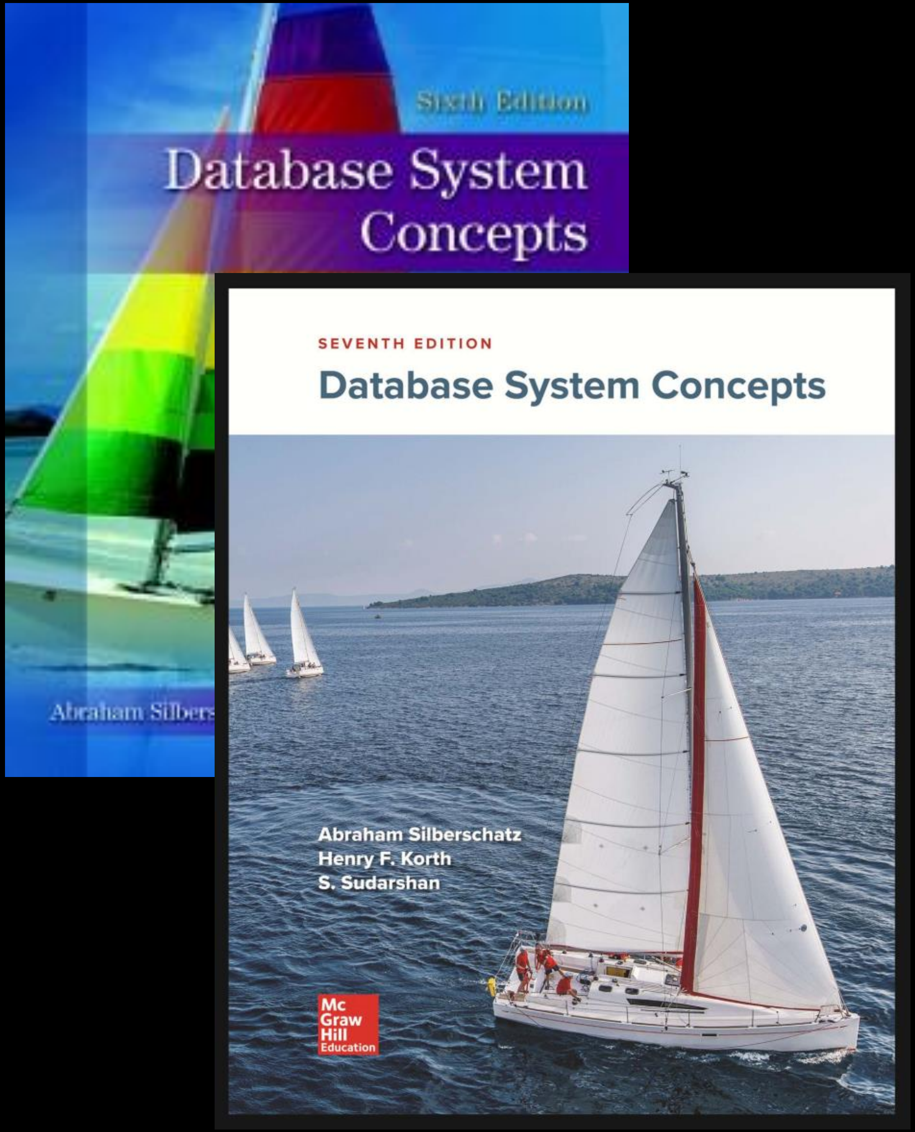
IF2140 – Pemodelan Basis Data / IF2240 – Basis Data

Database Design using E-R Model (p.3)

Modified form Silberschatz's slides, Database System Concept, 7th edition

KNOWLEDGE & SOFTWARE ENGINEERING



References

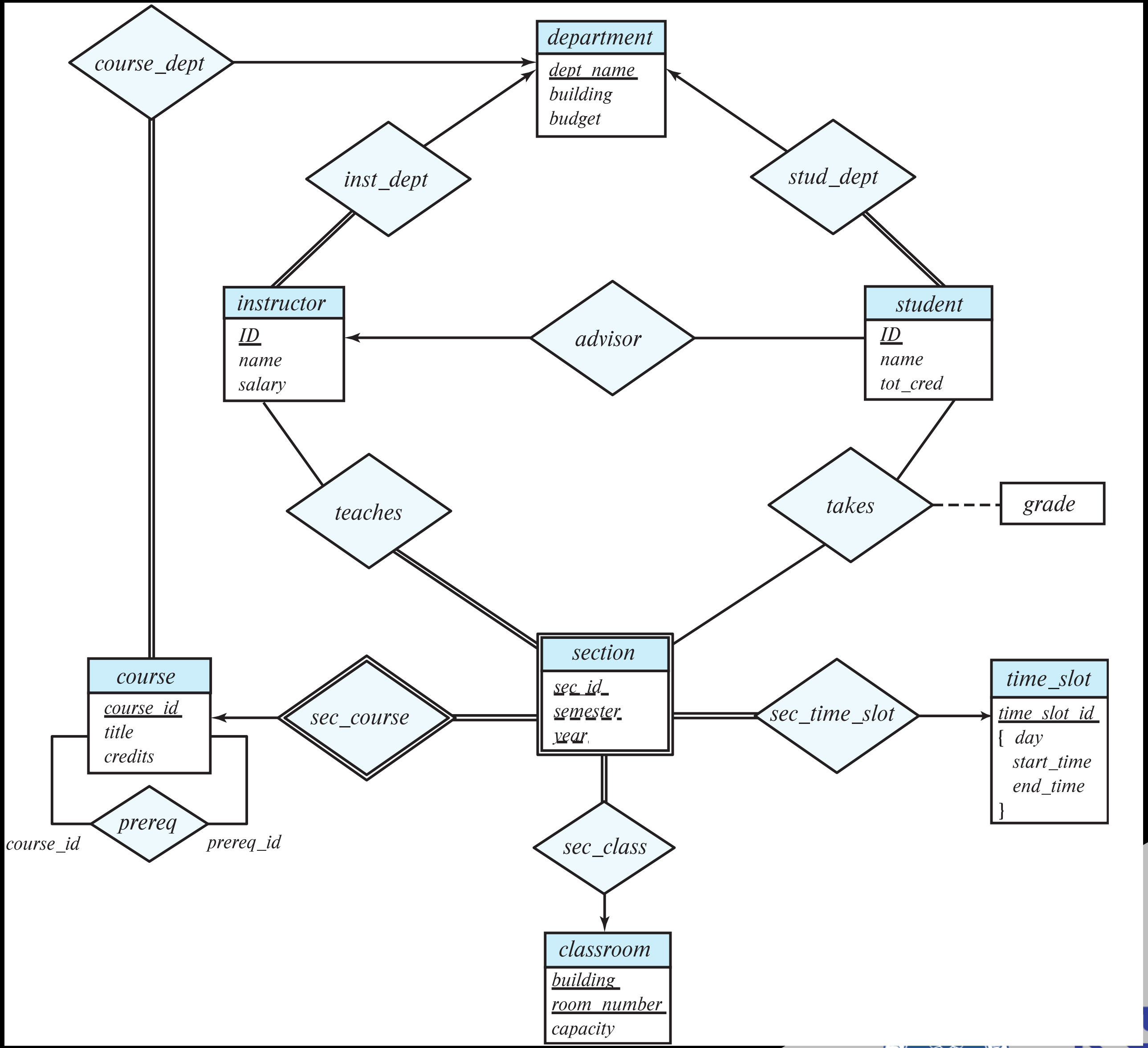
Abraham Silberschatz, Henry F. Korth, S. Sudarshan : "**Database System Concepts**", 6th Edition

- Chapter 7: Database Design and the E-R Model

Abraham Silberschatz, Henry F. Korth, S. Sudarshan : "**Database System Concepts**", 7th Edition

- Chapter 6: Database Design using E-R Model

E-R Diagram for a University Enterprise



KNOWLEDGE & SOFTWARE ENGINEERING

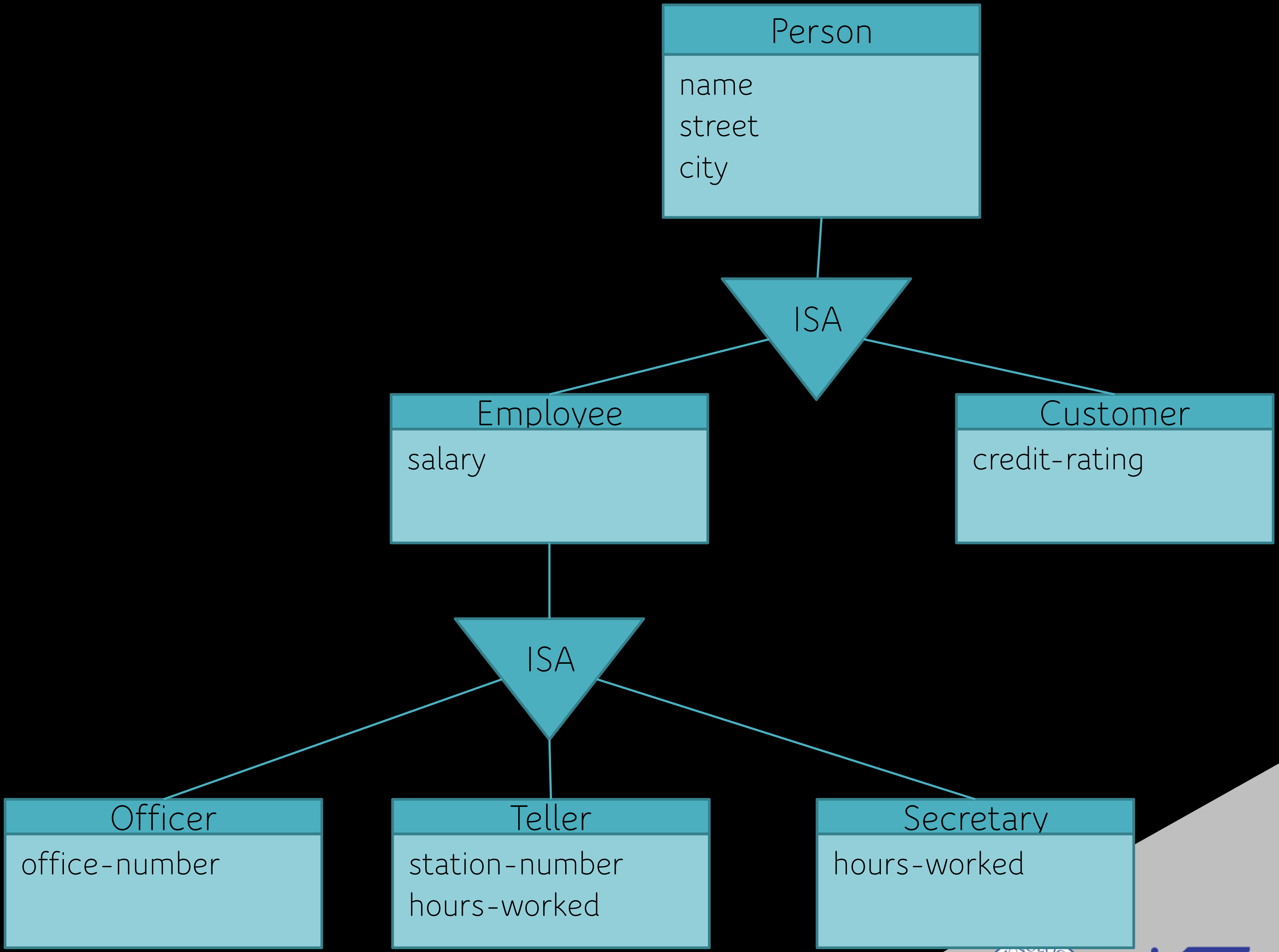
Specialization

Top-down design process: we designate subgroupings within an entity set that are distinctive from other entities in the set.

These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set

Depicted by a *triangle* component labeled ISA (E.g. *customer* "is a" *person*).

Attribute inheritance – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked



Generalization

A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set.

Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.

The terms specialization and generalization are used interchangeably

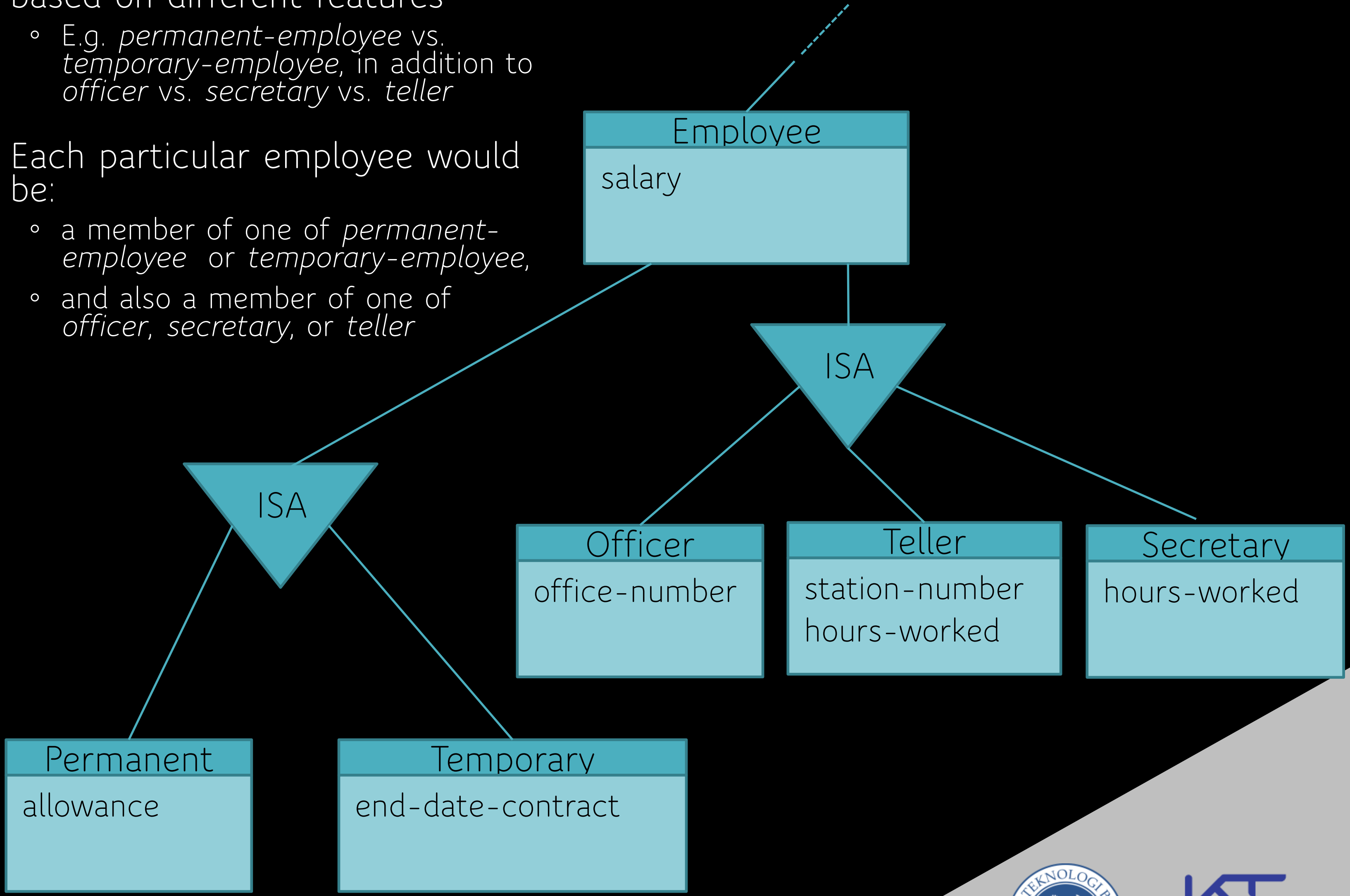
The ISA relationship also referred to as **superclass-subclass** relationship

Can have multiple specializations of an entity set based on different features

- E.g. *permanent-employee* vs. *temporary-employee*, in addition to *officer* vs. *secretary* vs. *teller*

Each particular employee would be:

- a member of one of *permanent-employee* or *temporary-employee*,
- and also a member of one of *officer*, *secretary*, or *teller*



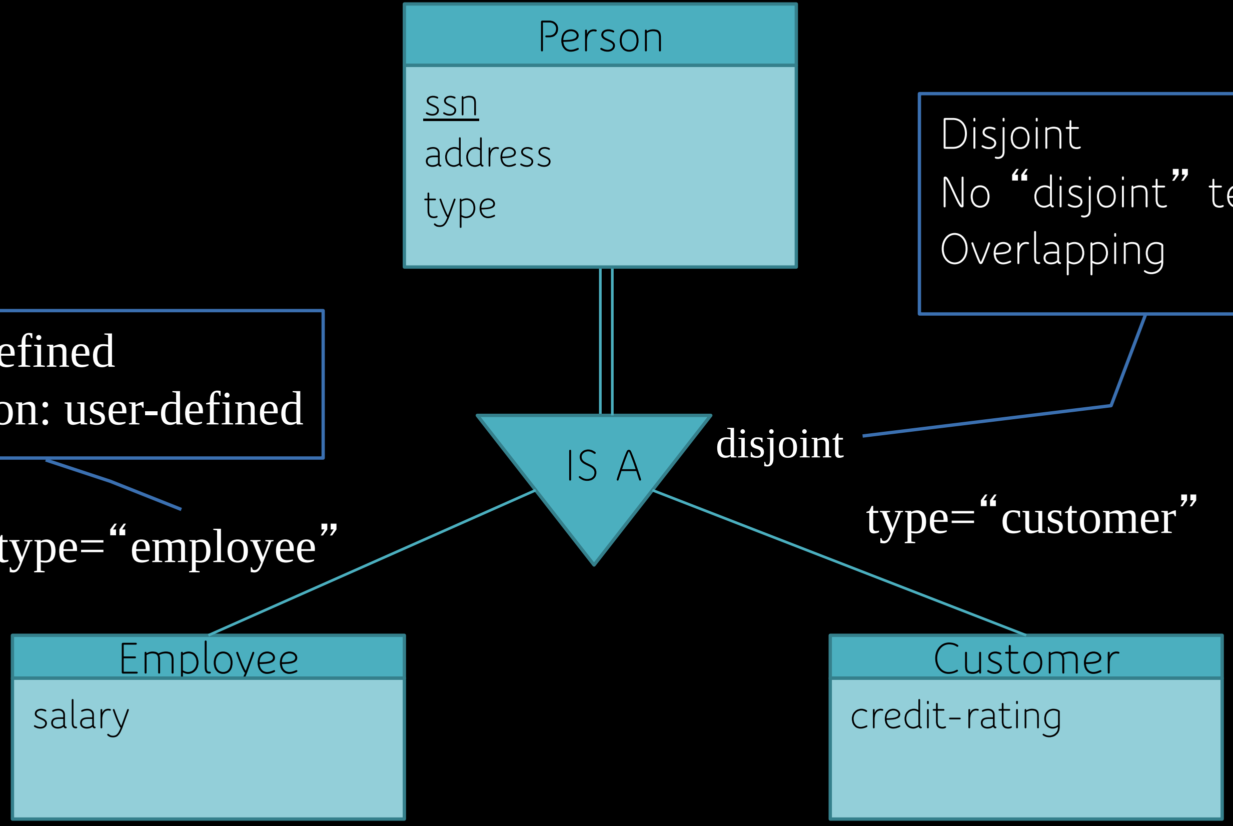
Design Constraints on a Specialization/ Generalization (1)

Constraint on which entities can be members of a given lower-level entity set

- condition-defined
- user-defined

Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization

- Disjoint
  - an entity can belong to only one lower-level entity set
  - Noted in E-R diagram by writing *disjoint* next to the ISA triangle
- Overlapping
  - an entity can belong to more than one lower-level entity set



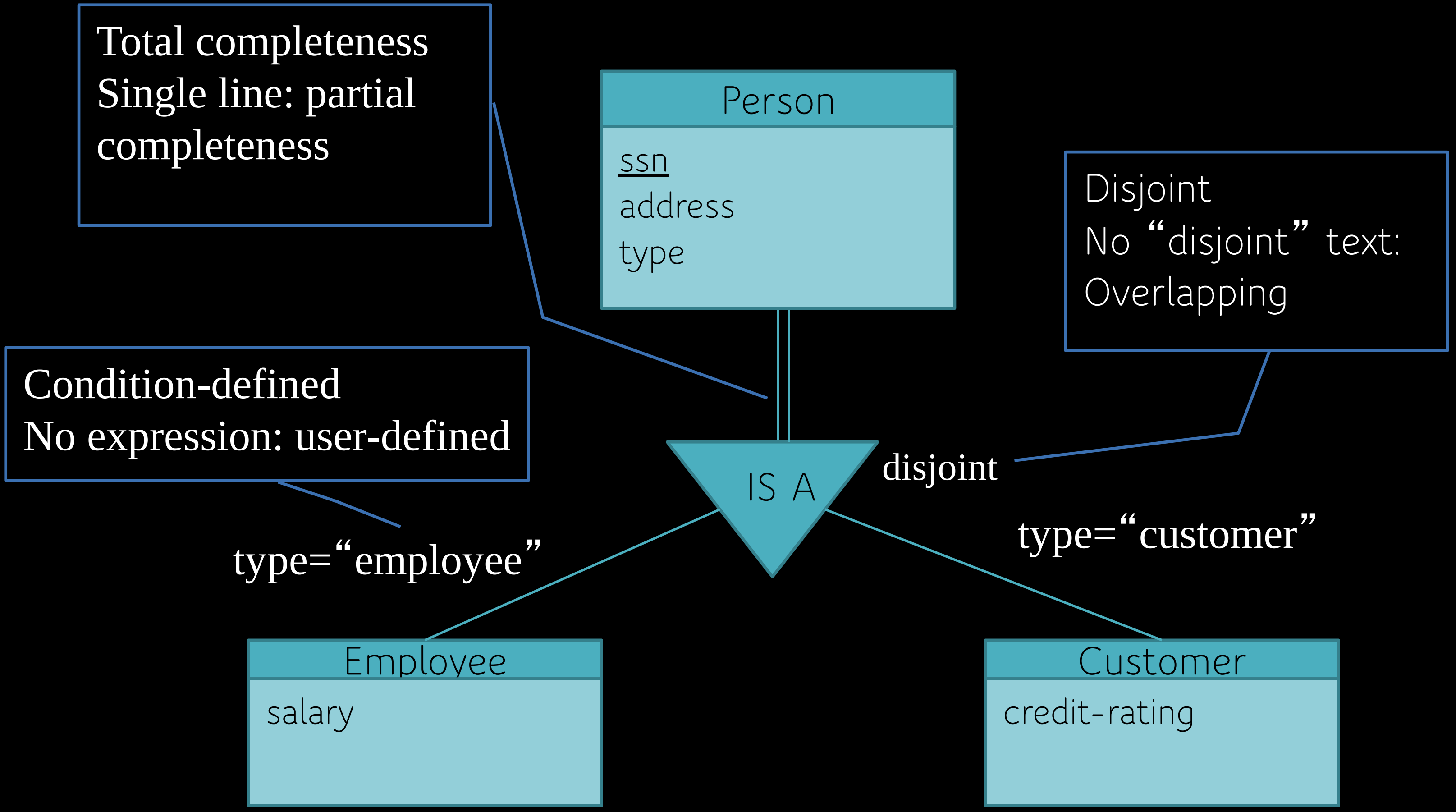
Disjoint
No "disjoint" text: Overlapping

Condition-defined
No expression: user-defined

Design Constraints on a Specialization/ Generalization (2)

Completeness constraint: specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.

- total: an entity must belong to one of the lower-level entity sets
- partial: an entity need not belong to one of the lower-level entity sets



Soal 01

Dalam database Dinas Sosial, disimpan data penduduk. Untuk setiap penduduk disimpan data NIK (unik untuk setiap orang), nama, tanggal lahir, no. telepon (setiap orang bisa punya >1 no. telepon), dan umur (dihitung berdasarkan tanggal lahir)

Setiap orang bisa dibedakan menjadi 2 jenis, yaitu tuna karya dan pekerja. Setiap tuna karya memiliki nomor kartu prakerja.

Setiap pekerja bekerja tepat di sebuah perusahaan. Untuk setiap perusahaan, dicatat id-perusahaan, nama, dan alamat.

Setiap orang harus merupakan tuna karya atau pekerja. Tetapi jika seseorang sudah tercatat sebagai tuna karya, dia tidak boleh tercatat sebagai pekerja, dan demikian pula sebaliknya.

Penentuan orang yang menjadi tuna karya dan pekerja dilakukan secara manual oleh pengguna.

Kuis Kecil…

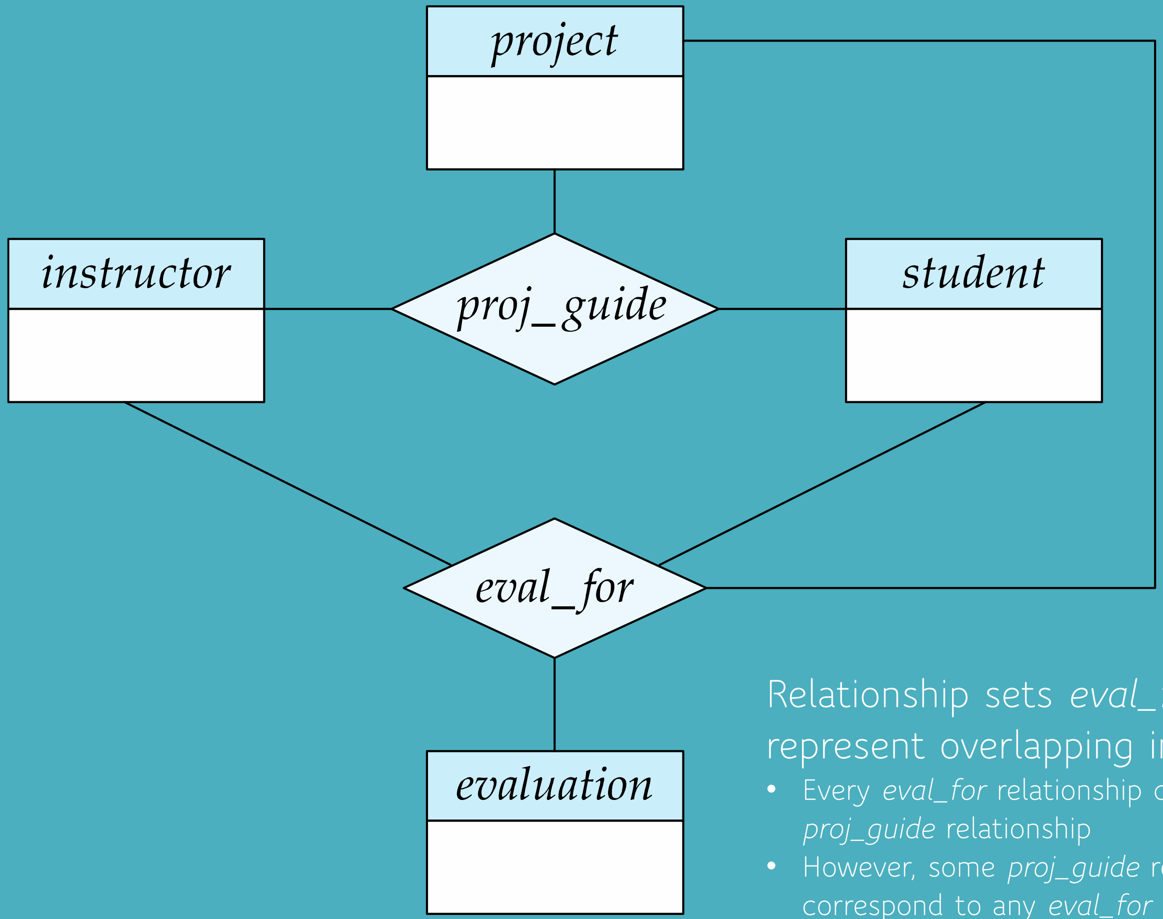
Apakah dibutuhkan spesialisasi/ generalisasi dalam persoalan ini?

Sebuah rental mobil mengklasifikasi kendaraan mereka menjadi empat katagori: compact, mid-size, full-size, dan sport utility. Tidak ada informasi khusus yang perlu disimpan untuk masing-masing katagori. Rental juga melakukan pencatatan data kendaraan berupa plat nomor (unik untuk setiap kendaraan), tahun pembuatan, model, tahun kepemilikan oleh rental, dan warna kendaraan. Seorang pelanggan dapat meminjam satu atau lebih kendaraan. Masing-masing pelanggan memiliki kode (unik untuk setiap pengguna), nama, alamat, dan email.

Aggregation

Consider the ternary relationship *proj_guide*, which we saw earlier

Suppose we want to record evaluations of a student by a guide on a project



Relationship sets *eval_for* and *proj_guide* represent overlapping information

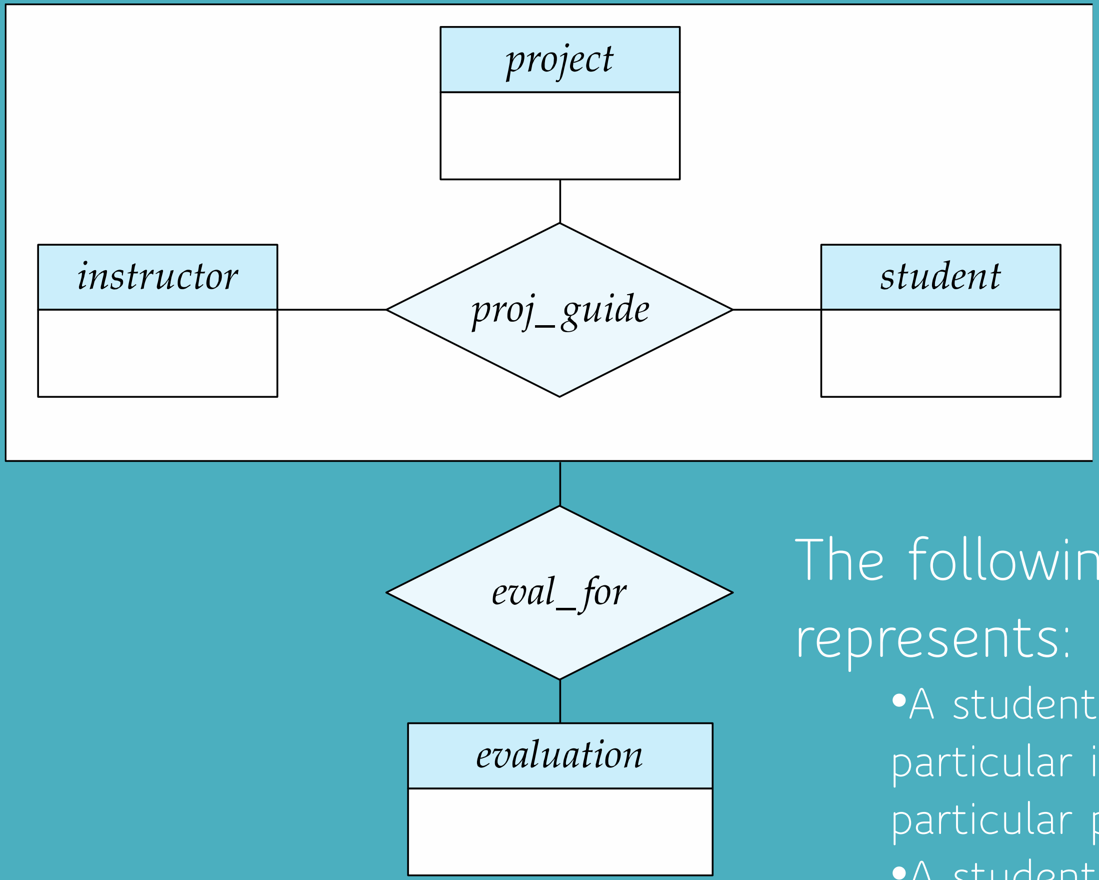
- Every *eval_for* relationship corresponds to a *proj_guide* relationship
- However, some *proj_guide* relationships may not correspond to any *eval_for* relationships

So we can't discard the *proj_guide* relationship

Aggregation

Eliminate this redundancy via *aggregation*

- Treat relationship as an abstract entity
- Allows relationships between relationships
- Abstraction of relationship into new entity



The following diagram represents:

- A student is guided by a particular instructor on a particular project
- A student, instructor, project combination may have an associated evaluation

Soal 02

Dalam sebuah database, disimpan data orang. Untuk setiap orang disimpan data no. KTP (unik untuk setiap orang, setiap orang punya KTP), nama, tanggal lahir, no. telepon (setiap orang bisa punya >1 no. telepon), dan umur (dihitung berdasarkan tanggal lahir)

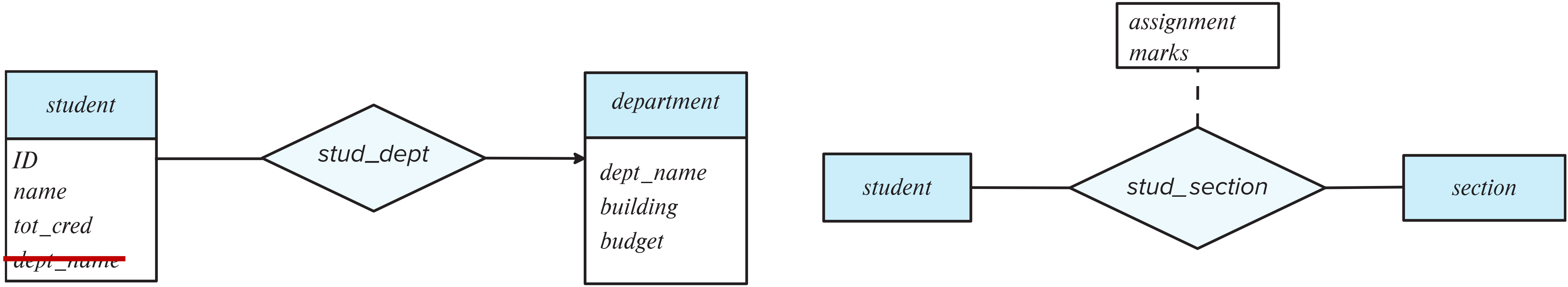
Disimpan pula data sejumlah organisasi. Setiap organisasi memiliki id organisasi yang unik, nama, tipe organisasi (perusahaan, LSM, organisasi pemerintah, dll), dan tanggal berdirinya.

Didefinisikan pula jabatan yang ada pada organisasi. Untuk setiap jabatan, dikenal id jabatan (unik untuk tiap jabatan), berikut deskripsinya. Pada sebuah organisasi terdefinisi beberapa jabatan, dan jabatan yang sama bisa ada di beberapa organisasi.

Suatu jabatan dalam organisasi pasti memiliki orang yang ditugaskan di sana. Namun, tidak semua orang harus tercatat menduduki jabatan tertentu di suatu organisasi. Ketika seseorang menjabat suatu jabatan tertentu di organisasi, dicatat tanggal mulainya. Seseorang boleh menduduki jabatan yang sama di organisasi yang berbeda dan menduduki jabatan yang berbeda di organisasi yang sama.

Common Mistakes in E-R Diagrams

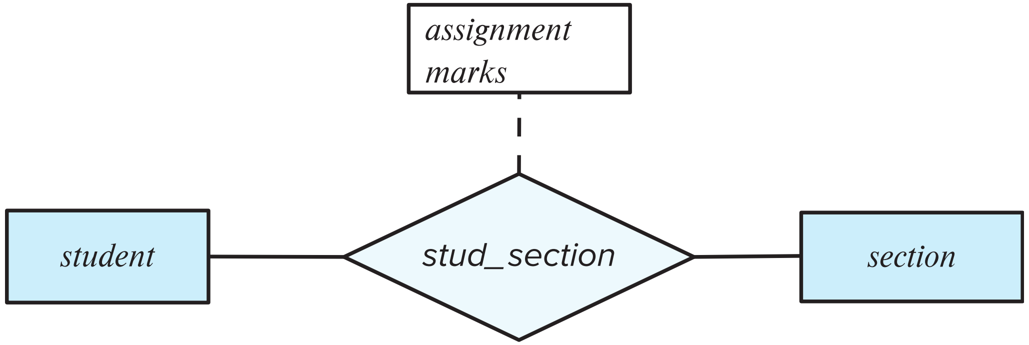
- Example of erroneous E-R diagrams



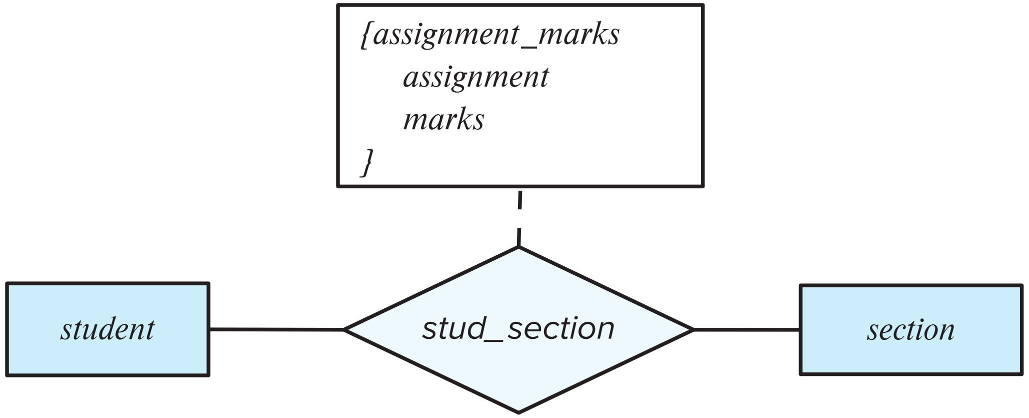
(a) Incorrect use of attribute

(b) Erroneous use of relationship attributes

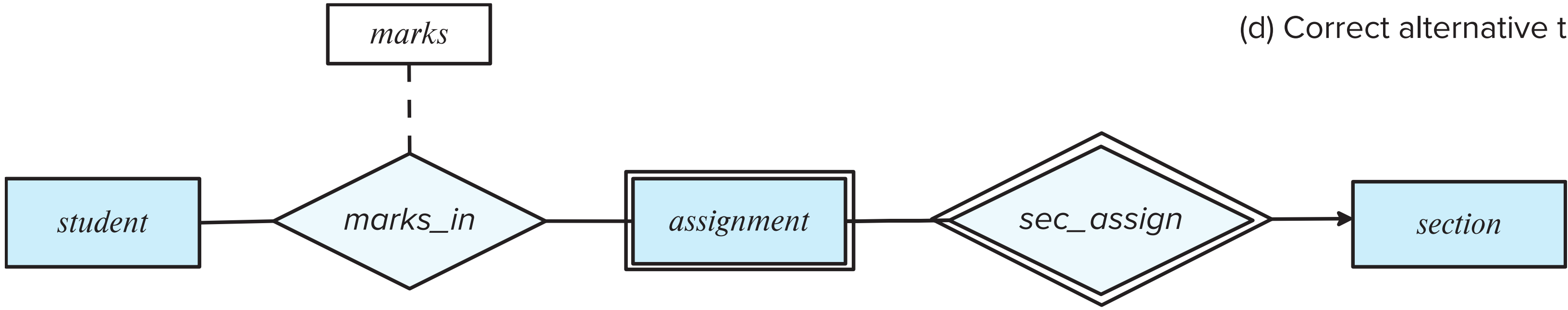
Common Mistakes in E-R Diagrams (Cont.)



(b) Erroneous use of relationship attributes



(d) Correct alternative to erroneous E-R diagram (b)



(c) Correct alternative to erroneous E-R diagram (b)

Redundant Attributes

Suppose we have entity sets:

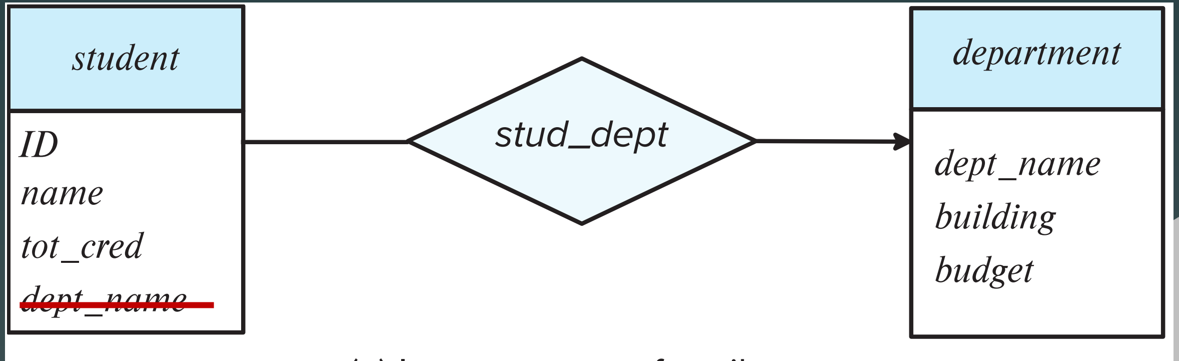
- *student*, with attributes: *ID, name, tot_cred, dept_name*
- *department*, with attributes: *dept_name, building, budget*

We model the fact that each student has an associated department using a relationship set *stud_dept*

The attribute *dept_name* in *student* below replicates information present in the relationship and is therefore redundant

- and needs to be removed.

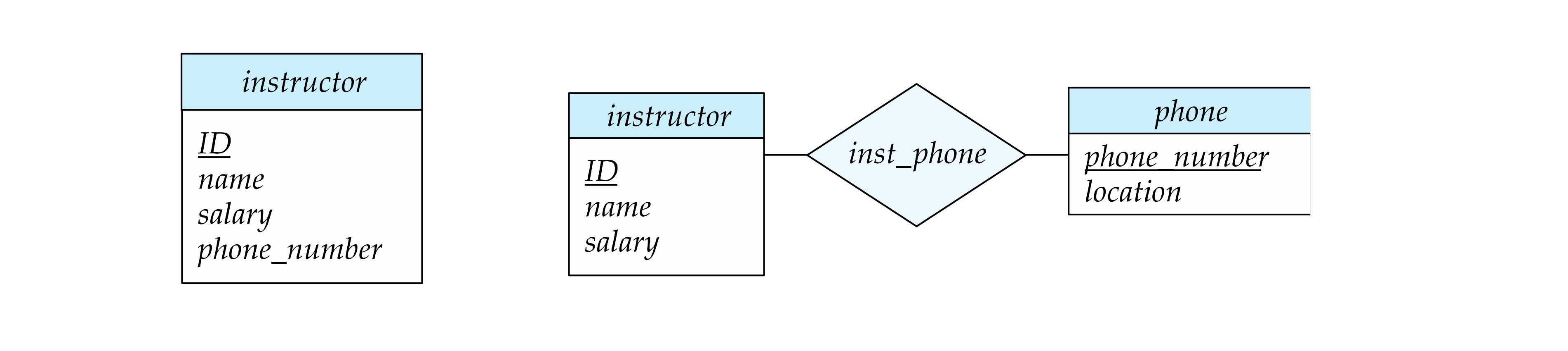
BUT: when converting back to tables, in some cases the attribute gets reintroduced, as we will see later (next topic).



(a) Incorrect use of attribute

Entities vs. Attributes

- Use of entity sets vs. attributes

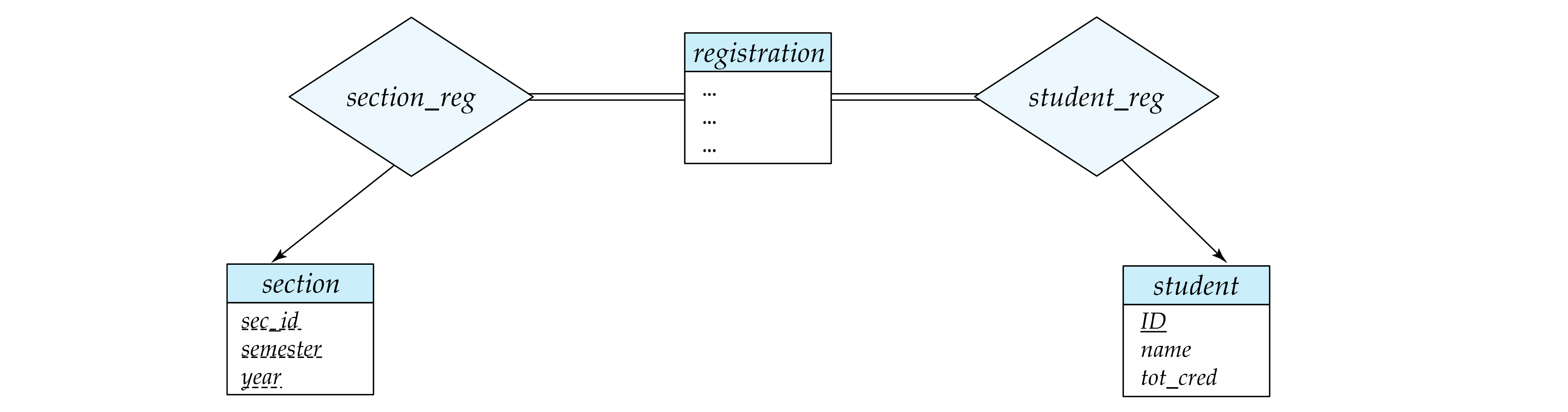


- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

Entities vs. Relationship sets

- Use of entity sets vs. relationship sets

  Possible guideline is to designate a relationship set to describe an action that occurs between entities



- Placement of relationship attributes

  For example, attribute date as attribute of advisor or as attribute of student

Binary Vs. Non-Binary Relationships

Although it is possible to replace any non-binary (*n*-ary, for $n > 2$) relationship set by a number of distinct binary relationship sets, a *n*-ary relationship set shows more clearly that several entities participate in a single relationship.

Some relationships that appear to be non-binary may be better represented using binary relationships

- For example, a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
  - Using two binary relationships allows partial information (e.g., only mother being known)
- But there are some relationships that are naturally non-binary
  - Example: *proj_guide*

Converting Non-Binary Relationships to Binary Form

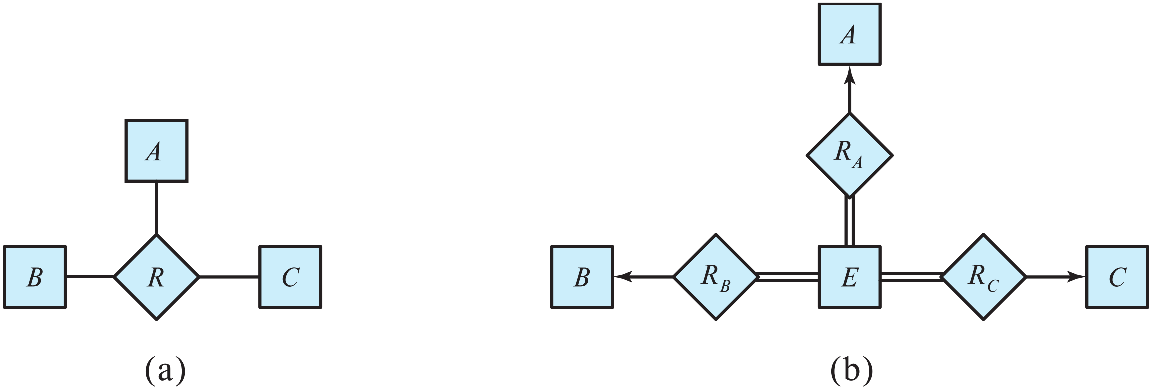
In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.

- Replace *R* between entity sets A, B and C by an entity set *E*, and three relationship sets:

  1. R_A, relating *E* and *A* 2. R_B, relating *E* and *B* 3. R_C, relating *E* and *C*

- Create an identifying attribute for *E and* add any attributes of *R* to *E*
- For each relationship (a_i , b_i , c_i) in *R*, create

  1. a new entity e_i in the entity set *E*
  2. add (e_i , a_i) to R_A
  3. add (e_i , b_i) to R_B
  4. add (e_i , c_i) to R_C



(a) (b)

Converting Non-Binary Relationships (Cont.)

Also need to translate constraints

- Translating all constraints may not be possible
- There may be instances in the translated schema that cannot correspond to any instance of *R*
  - Exercise: *add constraints to the relationships R_A, R_B and R_C* to ensure that a newly created entity corresponds to exactly one entity in each of entity sets *A*, *B* and *C*
- We can avoid creating an identifying attribute by making E a weak entity set identified by the three relationship sets

E-R Design Decisions

1. The use of an attribute or entity set to represent an object.
2. Whether a real-world concept is best expressed by an entity set or a relationship set.
3. The use of a ternary relationship versus a pair of binary relationships.
4. The use of a strong or weak entity set.
5. The use of specialization/generalization – contributes to modularity in the design.
6. The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.

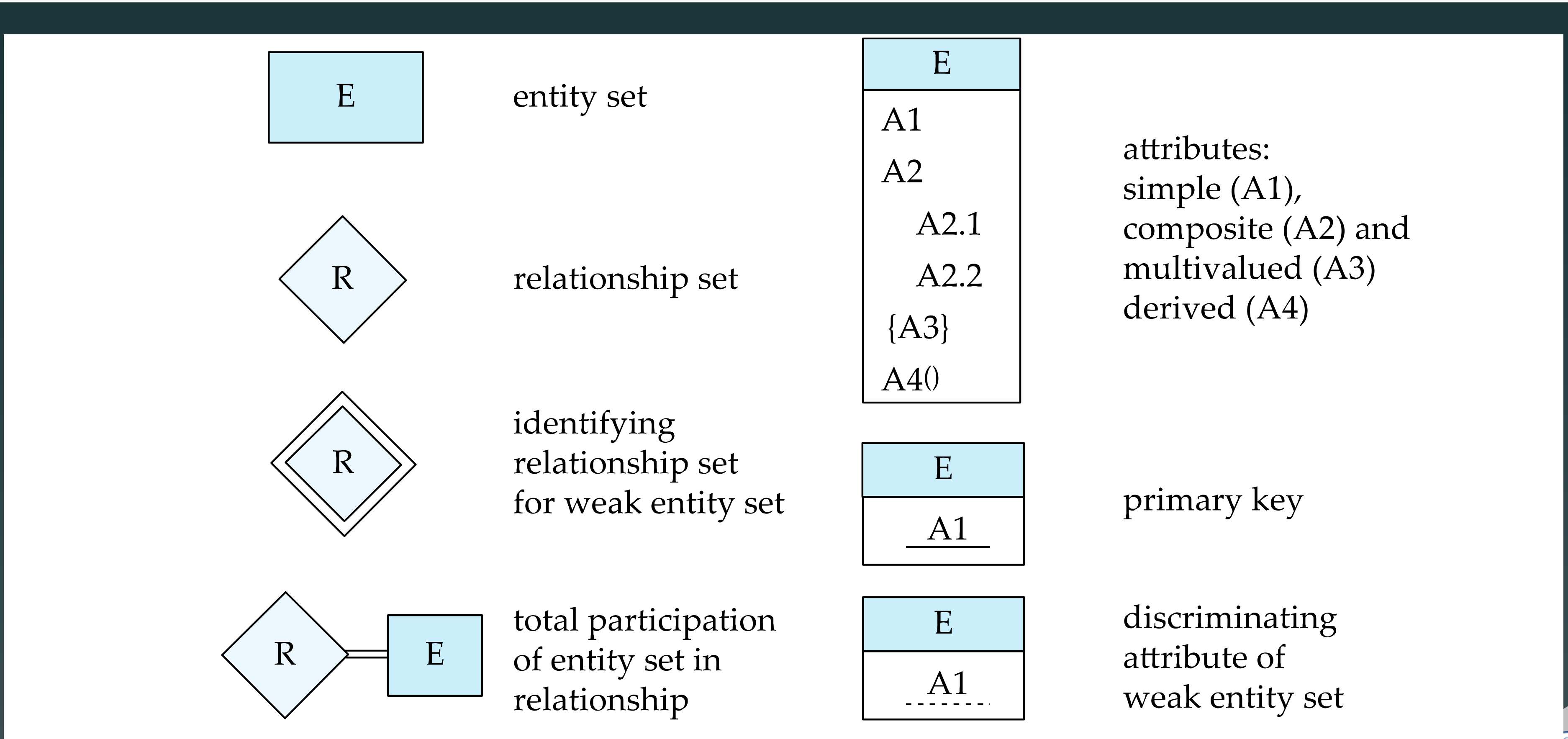
What if an E-R diagram becomes too big to draw in a single piece?

It makes sense to break it up into pieces, each showing part of the E-R model.

When an entity set is depicted in more than one page, attributes of the entity set should be shown only once, in its first occurrence.

- Subsequent occurrences of the entity set should be shown without any attributes
  - to avoid repeating the same information at multiple places, which may lead to inconsistency.

Summary of Symbols Used in E-R Notation



entity set

relationship set

identifying
relationship set
for weak entity set

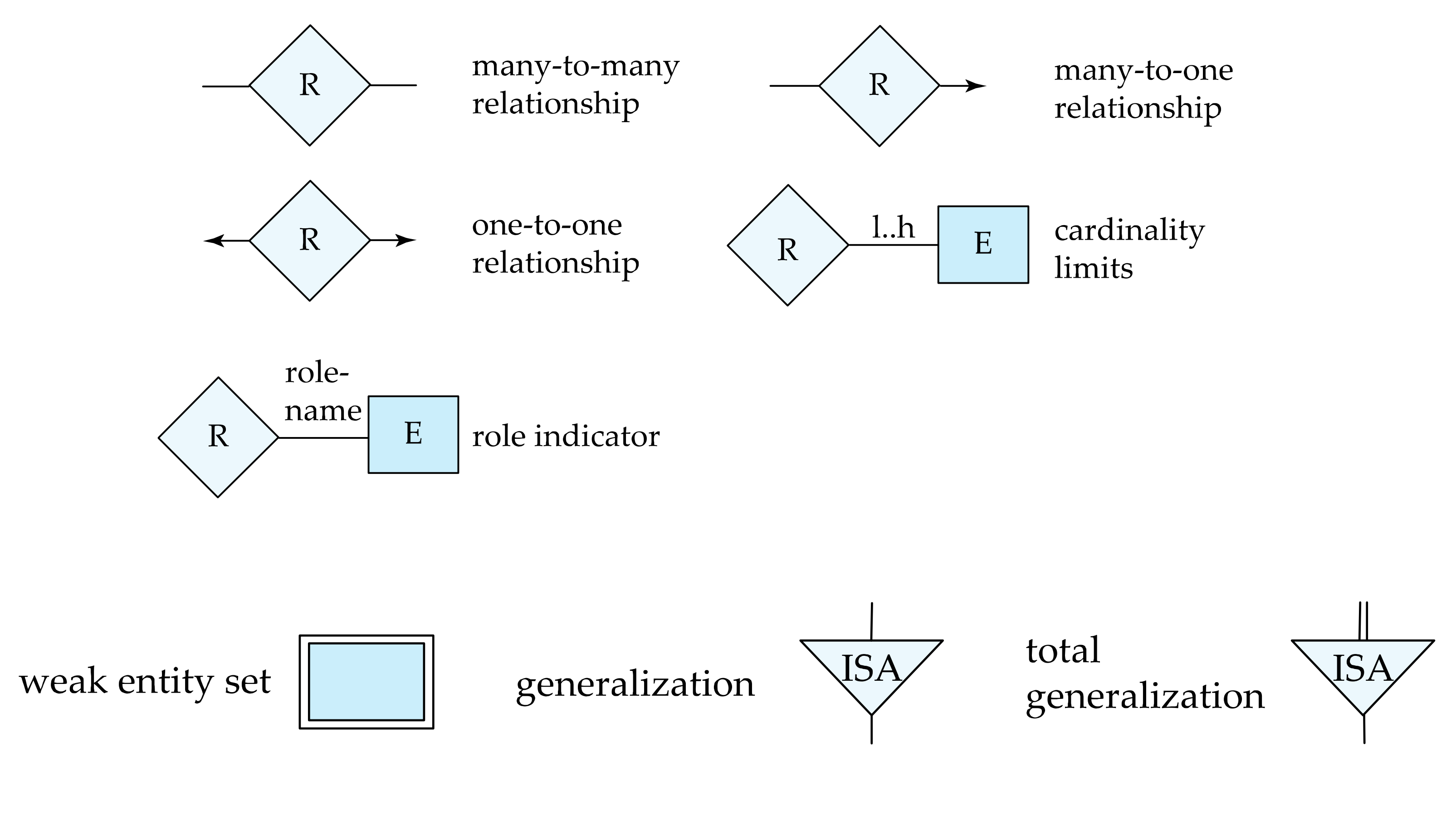
total participation
of entity set in
relationship

attributes:
simple (A1),
composite (A2) and
multivalued (A3)
derived (A4)

primary key

discriminating
attribute of
weak entity set

Symbols Used in E-R Notation (Cont.)



many-to-many relationship

many-to-one relationship

one-to-one relationship

cardinality limits

role indicator

weak entity set

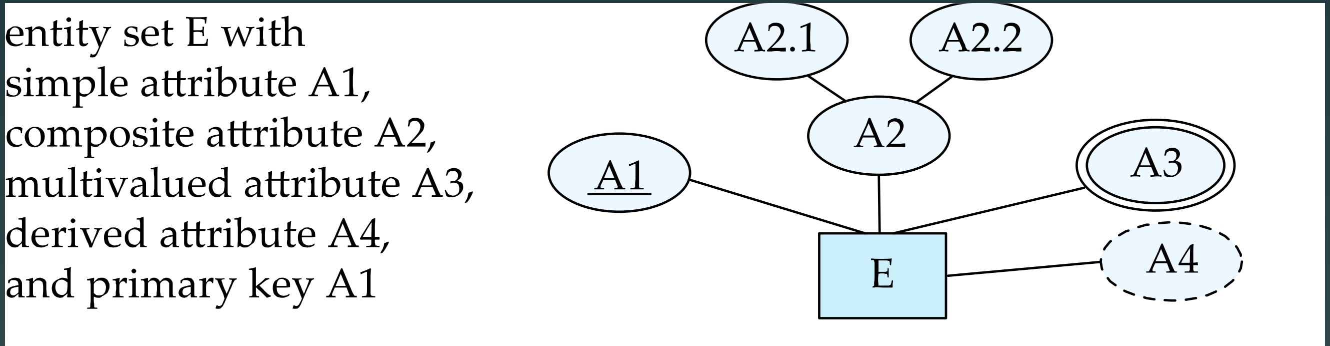
generalization

total generalization

Additional Materials

Alternative ER Notations

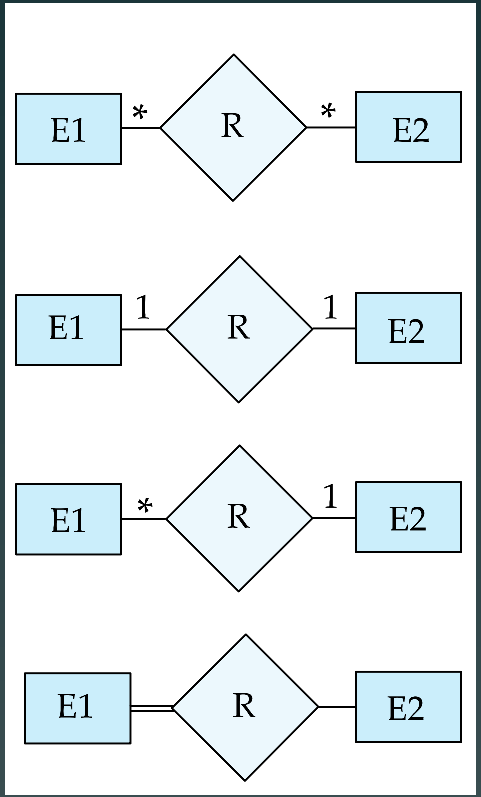
Chen, IDE1FX, …



entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1

Alternative ER Notations

CHEN



IDE1FX

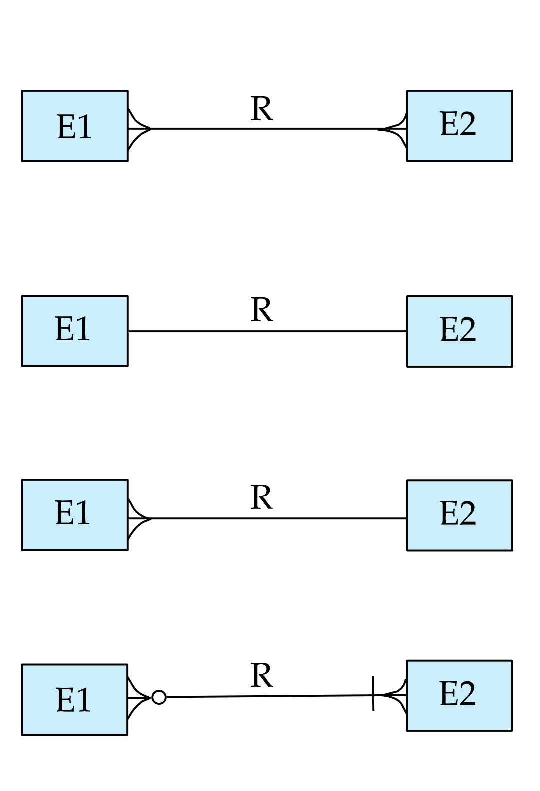
(CROWS FEET NOTATION)

many-to-many relationship

one-to-one relationship

many-to-one relationship

participation in R: total (E1) and partial (E2)



UML

UML: Unified Modeling Language

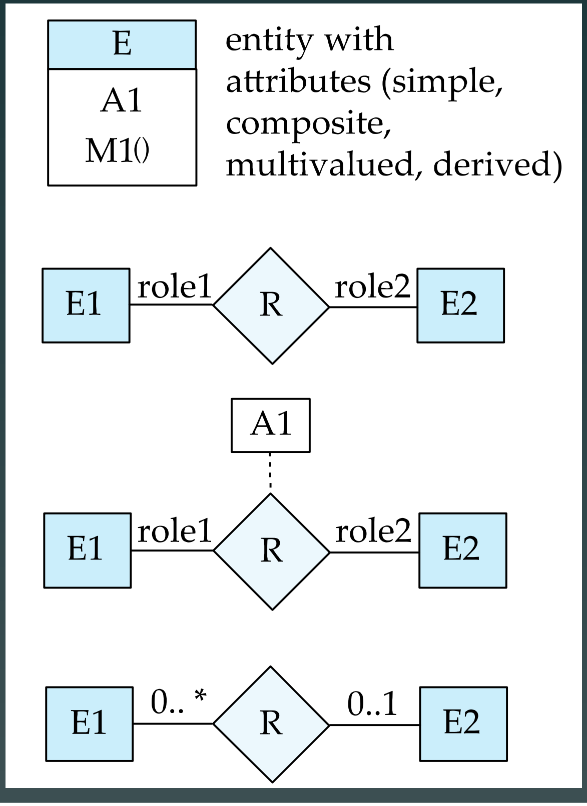
UML has many components to graphically model different aspects of an entire software system

UML Class Diagrams correspond to E-R Diagram, but several differences.

ER vs. UML Class Diagrams

ER DIAGRAM NOTATION

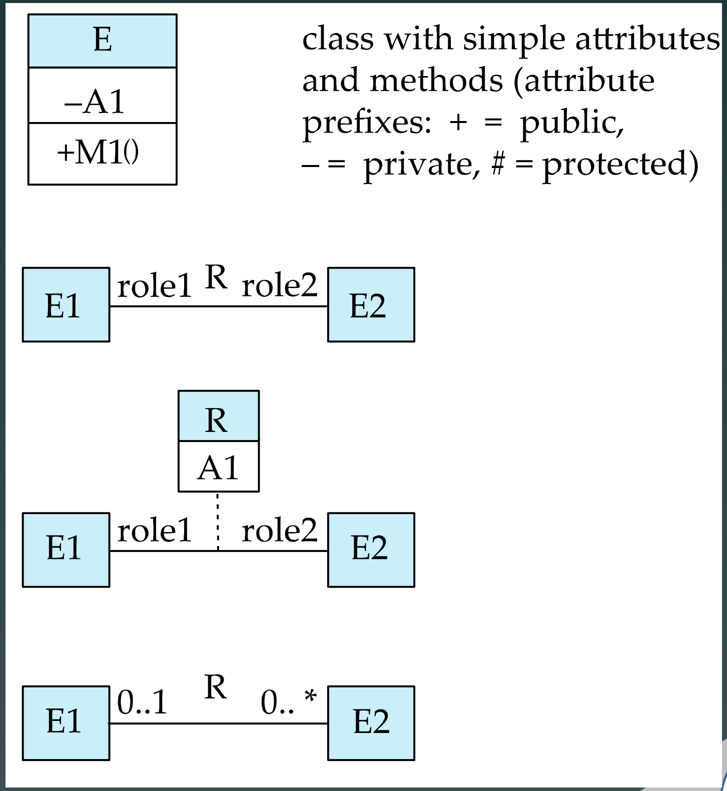
EQUIVALENT IN UML


E
A1
M1()
entity with attributes (simple, composite, multivalued, derived)
E1
role1
R
role2
E2
A1
E1
role1
R
role2
E2
E1
0.. *
R
0..1
E2

binary relationships

relationship attributes

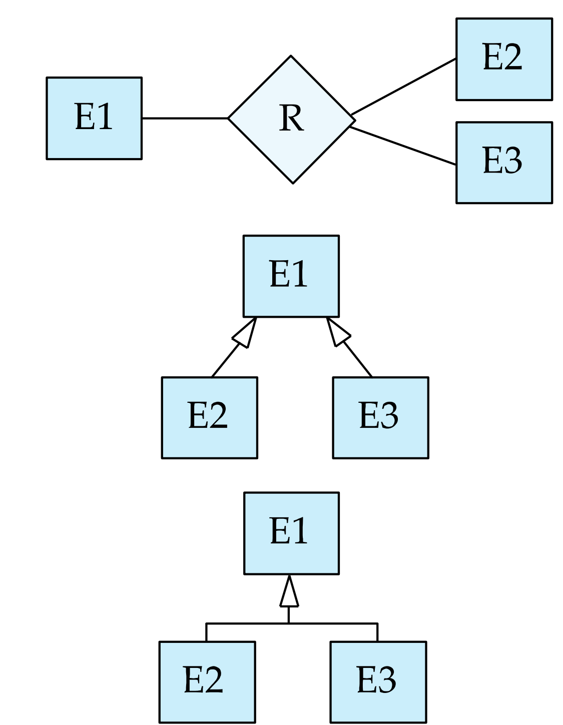
cardinality constraints


E
–A1
+M1()
class with simple attributes and methods (attribute prefixes: + = public, – = private, # = protected)
E1
role1
R
role2
E2
R
A1
E1
role1
role2
E2
E1
0..1
R
0.. *
E2

Note reversal of position in cardinality constraint depiction

ER vs. UML Class Diagrams (Cont.)

ER DIAGRAM NOTATION

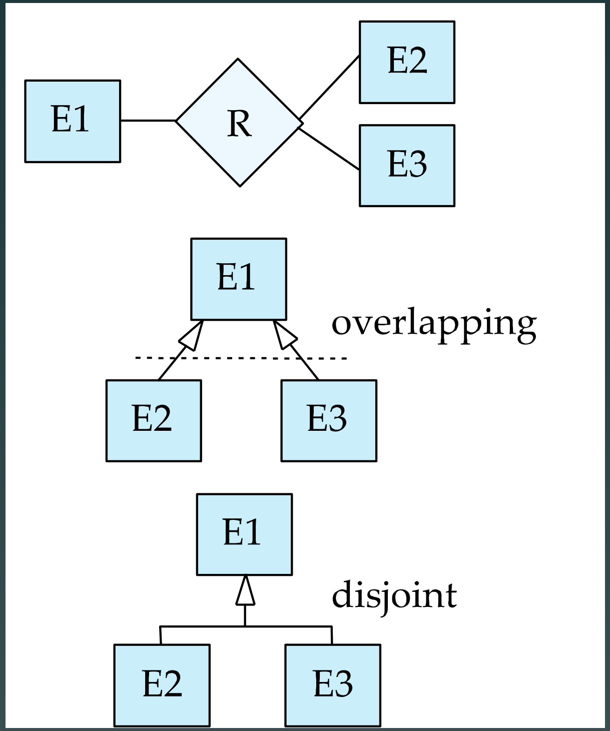


n-ary relationships

overlapping generalization

disjoint generalization

EQUIVALENT IN UML



Generalization can use merged or separate arrows independent of disjoint/overlapping

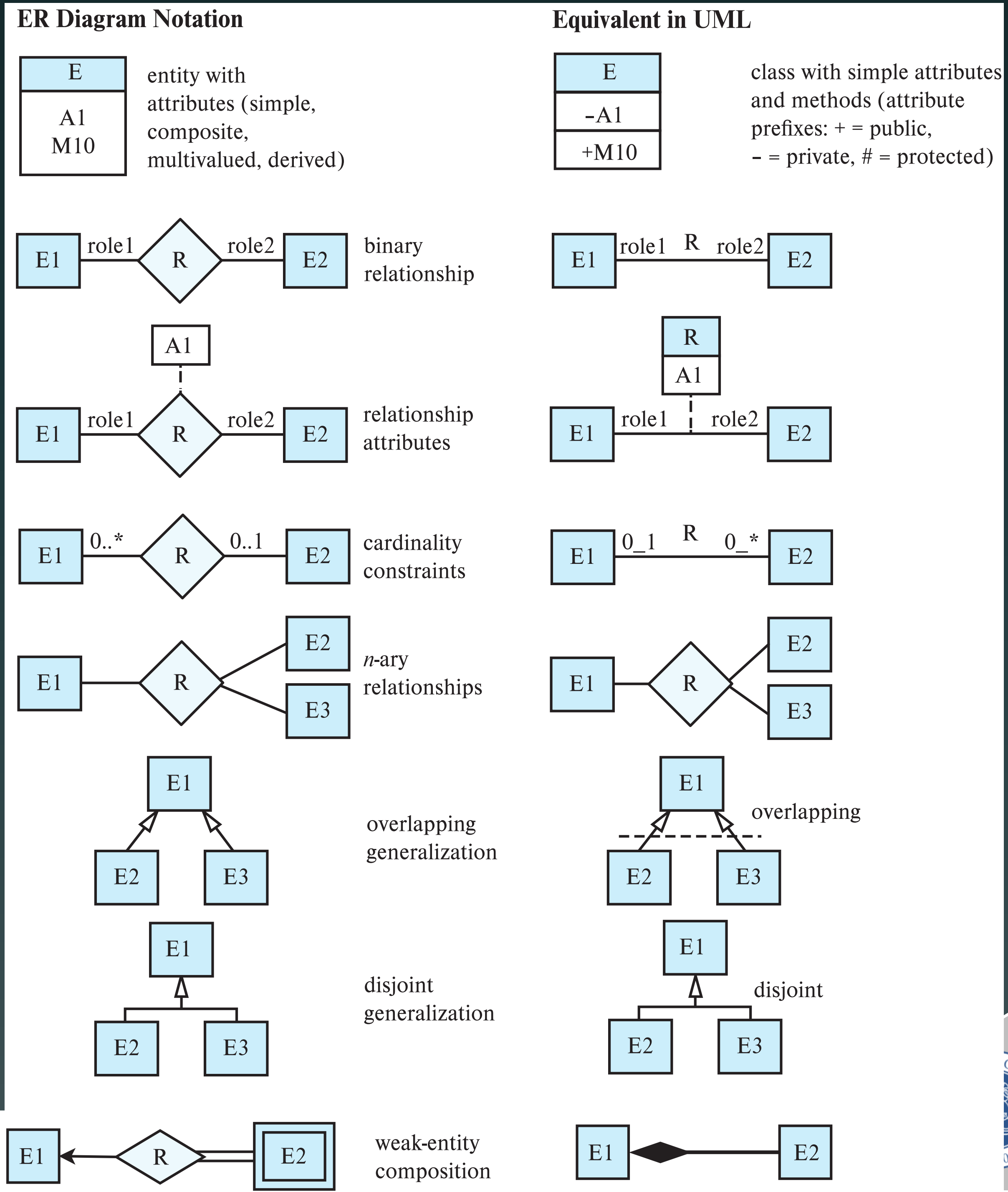
UML Class Diagrams (Cont.)

Binary relationship sets are represented in UML by just drawing a line connecting the entity sets. The relationship set name is written adjacent to the line.

The role played by an entity set in a relationship set may also be specified by writing the role name on the line, adjacent to the entity set.

The relationship set name may alternatively be written in a box, along with attributes of the relationship set, and the box is connected, using a dotted line, to the line depicting the relationship set.

ER vs. UML Class Diagrams



Other Aspects of Database Design

Functional Requirements

Data Flow, Workflow

Schema Evolution