CSc 484
Database Management Systems

Ken Gamradt

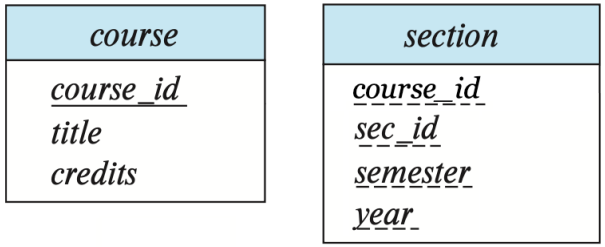
Spring 2024

Database Design using ER (III)

Weak Entity Sets and Strong Entity Sets

- **Weak** entity set: dependent on the other entity set
  - E.g., course entity set and section entity set
    - The existence of section is dependent on course
    - Section: weak entity, its identifying entity set is course

course is the **identifying entity set** of the **section** entity set

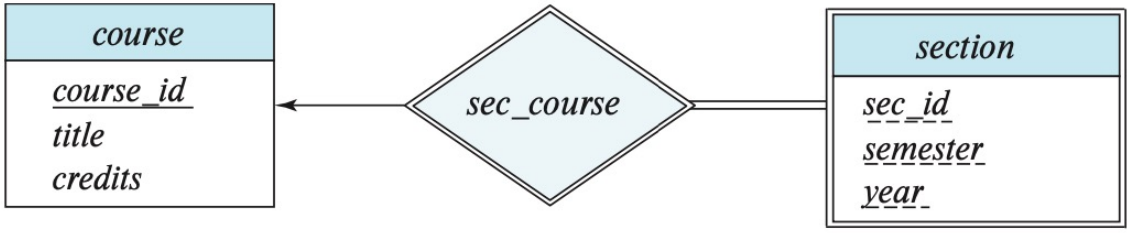


section is **dependent** on **course** entity set

- **Strong entity set**: does not have a dependent entity set
  - Such as course, student, instructor

Weak Entity Sets and Strong Entity Sets

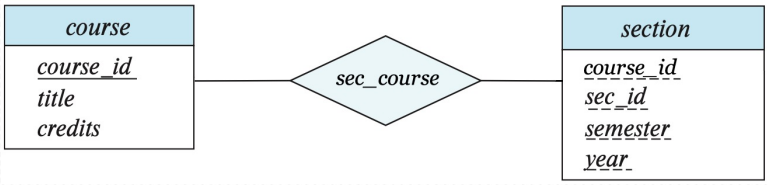
- Entity sets can be classified as a **strong entity set** and a **weak entity set**
- **Strong entity set**:
  - An entity set that is not existence-dependent on some other entity set
- **Weak entity set**:
  - An entity set that is existence-dependent on some other set
  - E.g., course entity set and section entity set



- Weak entity set must have total participation in the identifying relationship set
- Double rectangle used to indicate weak entity

Weak Entity Sets and Strong Entity Sets

- *course* and *section*



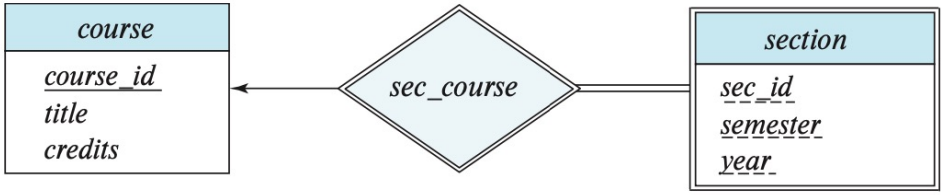
course_id	title	credits
CS101	ComputerScience	3
CS200	C++Programming	3
CS300	DataStructure	3

Information is **redundant** in the *sec_course* relationship set

course_id	sec_id	year	semester
CS101	1	2019	Fall
CS101	2	2019	Fall
CS101	1	2020	Spring
CS300	1	2019	Fall
CS300	1	2020	Spring
CS300	2	2020	Spring

Weak Entity Sets

- *course* and *section*
  - **Identifying relationship**



weak entity set
double rectangle

The primary key of weak entity set is determined by the primary key of identifying entity set and the extra attributes in the entity set to uniquely identify each weak entity

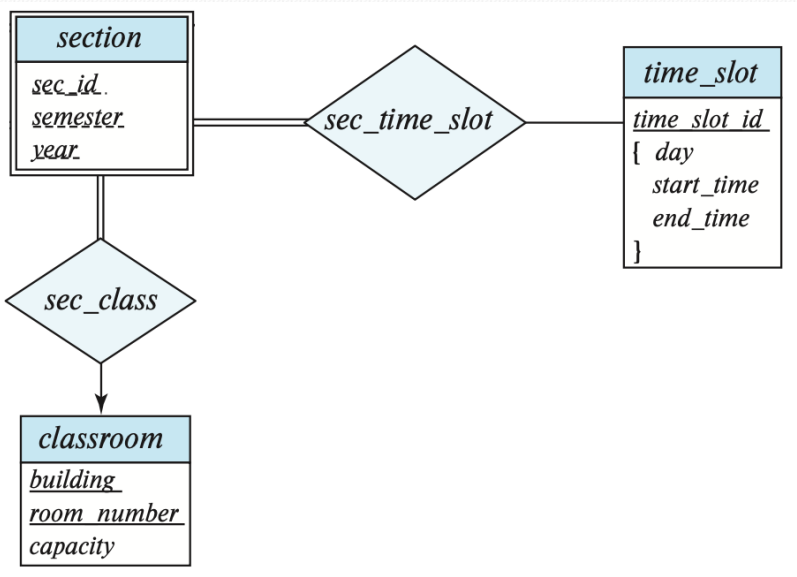
Primary key of *section*:
(**course_id**, **sec_id, year, semester**)

Cardinality: many-to-one from weak entity set to the identifying set
Participation: total participation for weak entity set

- **Primary key of the identifying set**
- **Extra attributes from weak entity set – discriminator attributes**

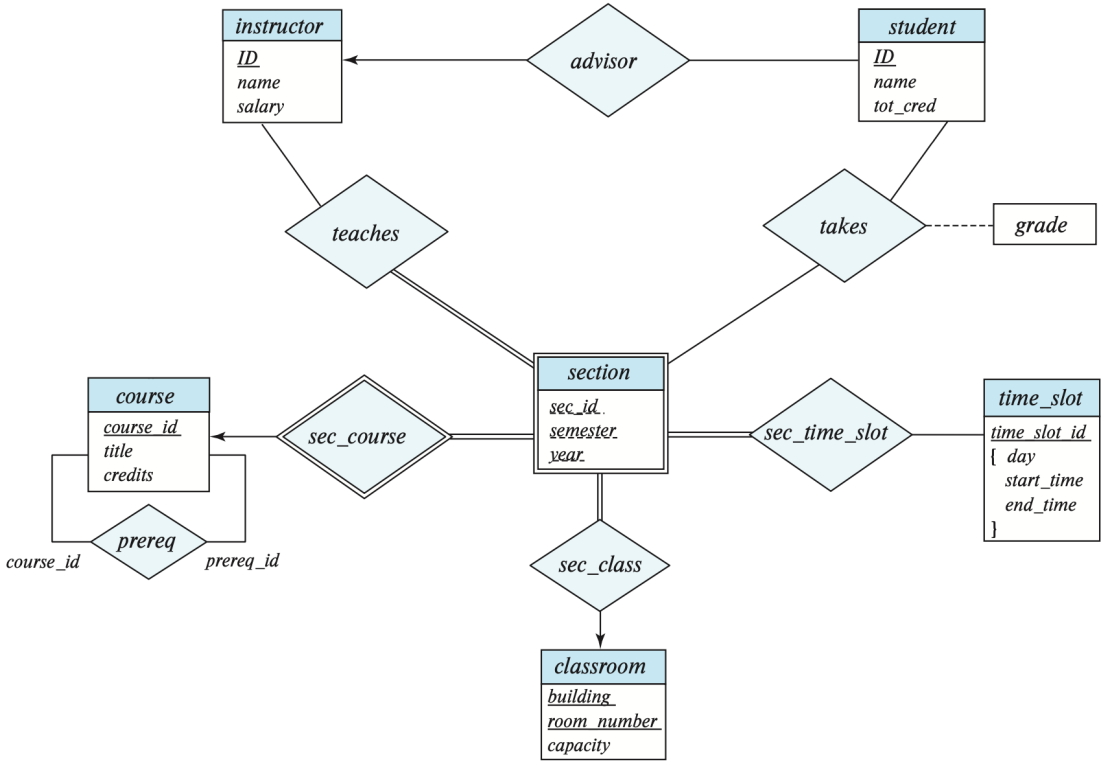
Weak Entity Sets

- The identifying relationship set should not have any descriptive attributes
  - Such attributes can be included in the weak entity set
- Weak entity set can participate in relationships other than in its identifying entity set



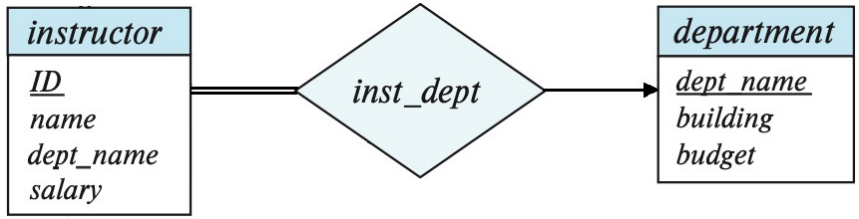
University Database Model

- Until now, the university database could be modeled as:

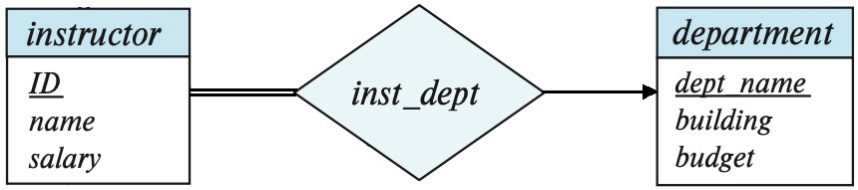


Redundant Attributes

- Each ***instructor*** and each ***student*** must associate to a ***department***

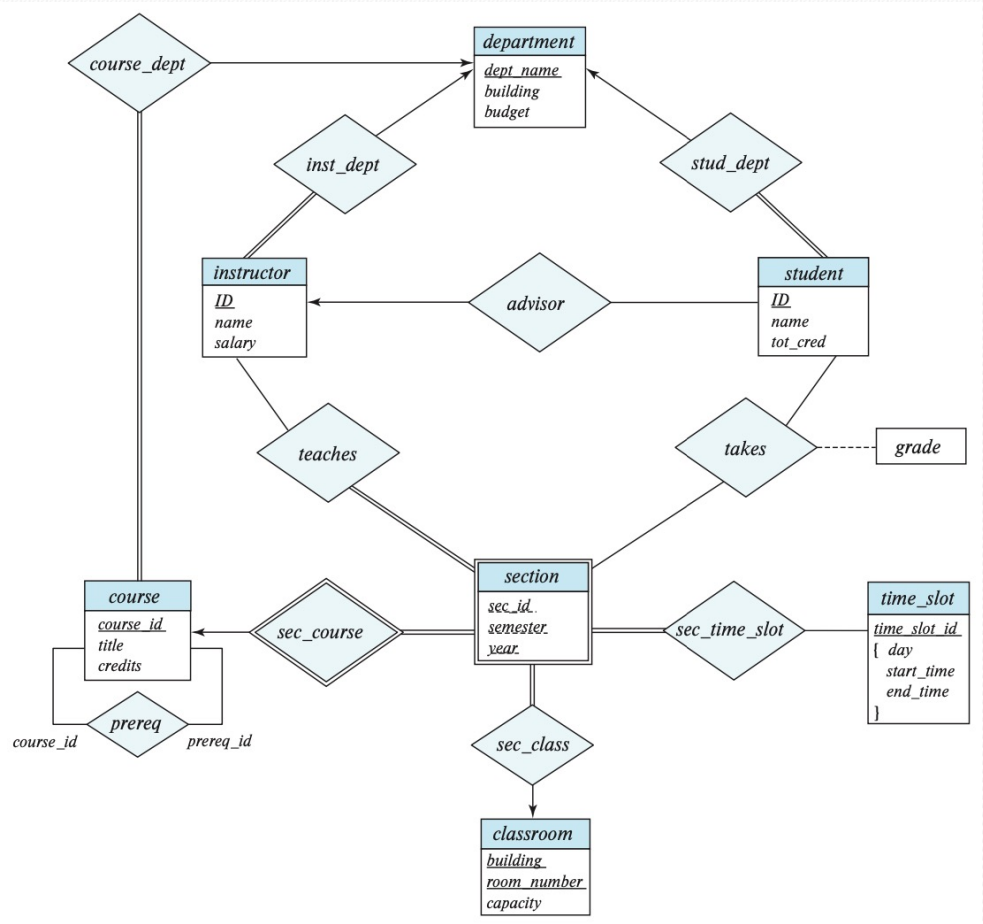


dept_name is redundant with the inst_dept relationship set



- **Note**: when converting back to tables, in some cases the attribute gets reintroduced, as we will see later

E-R model for university



A good entity-relationship design does not contain redundant attributes

Reducing E-R Diagrams to Relational Schemas

- Entity sets and relationship sets can be expressed uniformly as **relation schemas** that represent the contents of the database
- A database which conforms to an E-R diagram can be represented by a collection of schemas
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set
- Each schema has a number of columns (generally corresponding to attributes), which have unique names

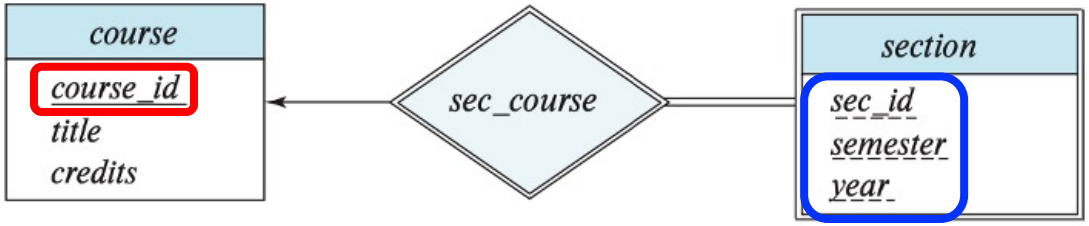
Weak Entity Sets

- A **strong** entity set reduces to a schema with the same attributes

  course (course_id, title, credits)

- A **weak** entity set becomes a table that includes
  - A column for the primary key of the identifying strong entity set
  - The columns of the weak entity set

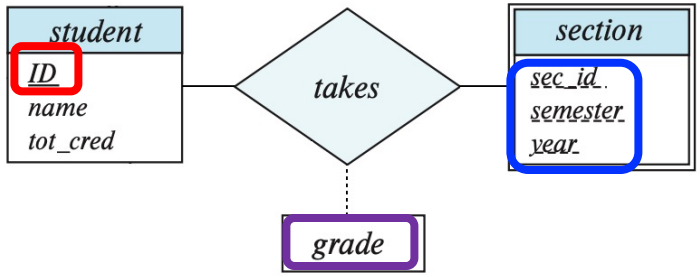
  section (course_id, sec_id, semester, year)



Relationship Sets

- A **weak** entity set becomes a table that includes
  - Foreign key, referencing the *student* relation
  - Foreign key, referencing the *section* relation
  - Descriptive attributes of the relationship set

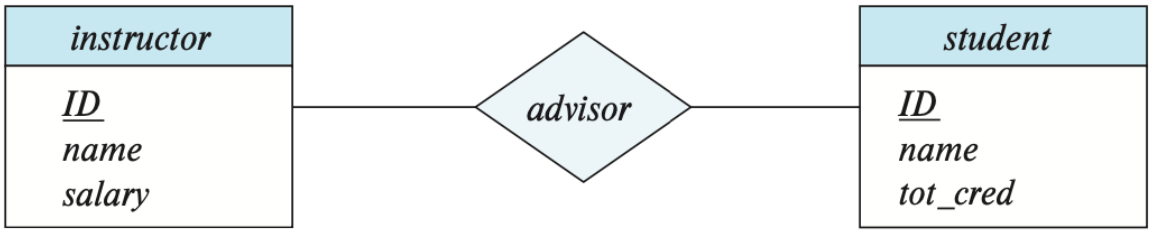
takes (*ID*, *sec_id*, *semester*, *year*, *grade*)



Relationship Sets

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set
- E.g., schema for relationship set *advisor*

advisor = (*s_ID*, *i_ID*)



Representing Entity with Multivalued Attributes

- Multivalued attribute M of an entity E is represented by a separate schema EM
- Schema EM has attributes corresponding to the primary key of E and an attribute corresponding to multivalued attribute M
- E.g., Multivalued attribute phone of student is represented by a schema:

  student (ID, name)

  student_phone (ID, phone)
- Each value of the multivalued attribute maps to a separate tuple of the relation on schema EM
- E.g., a student entity Bob, with primary key 10001 and phone numbers 605-611-2345 and 605-622-3456 maps to two tuples:

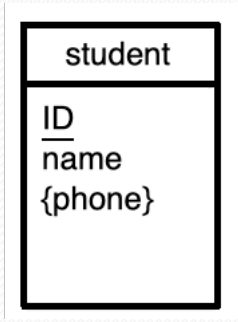
  student (10001, Bob)

  student_phone (10001, 605-611-2345)

  student_phone (10001, 605-622-3456)

Representing Entity with Multivalued Attributes

- Strong entity sets with complex attributes
  - Multivalued attributes



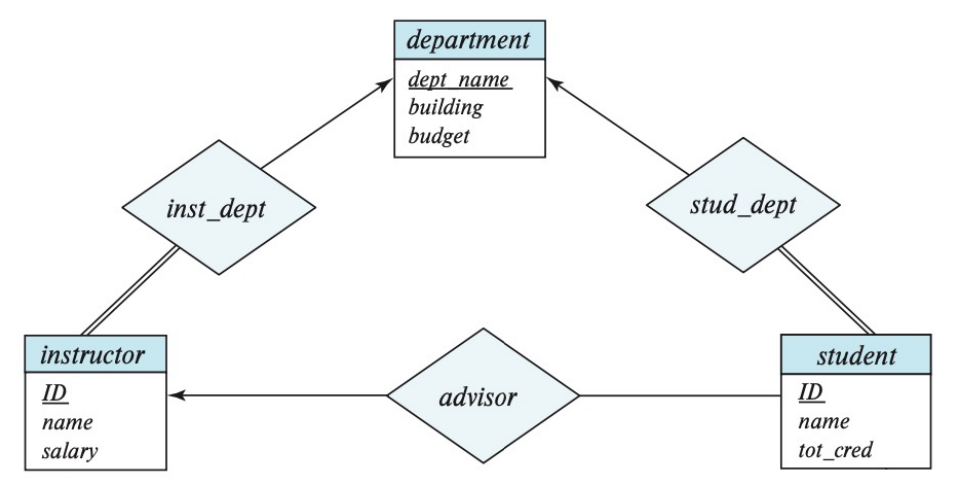
student (<u>ID</u>, name)

student_phone (<u>ID</u>, <u>phone</u>)

ID: 10001	*student (10001, Bob)*
Name: Bob	*student_phone(10001, 605-611-2345)*
Phone: 605-611-2345, 605-622-3456	*student_phone(10001, 605-622-3456)*

Redundancy of Schemas

- Many-to-one and one-to-many relationship sets that are total on the "many" side can be represented by adding an extra attribute to the "many" side
  - The primary key of the "one" side
- E.g., Instead of creating a schema for relationship set *inst_dept*, add an attribute *dept_name* to the schema arising from entity set *instructor*

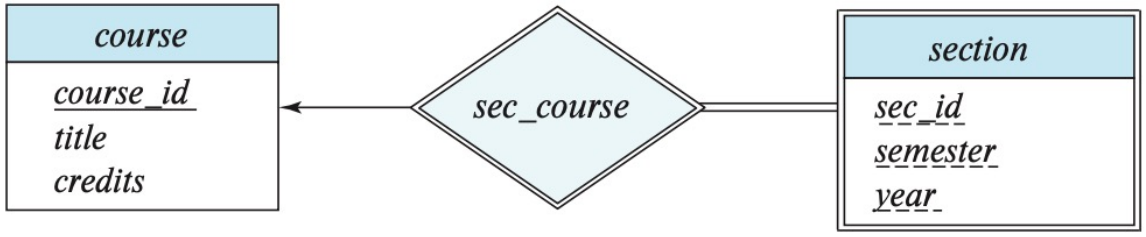


Redundancy of Schemas

- For one-to-one relationship sets, either side can be chosen to act as the "many" side
  - An extra attribute can be added to either of the tables corresponding to the two entity sets
- If participation is partial on the "many" side, replacing a schema by an extra attribute in the schema corresponding to the "many" side could result in null values

Redundancy of Schemas

- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant
- E.g., the *section* schema already contains the attributes that would appear in the *sec_course* schema

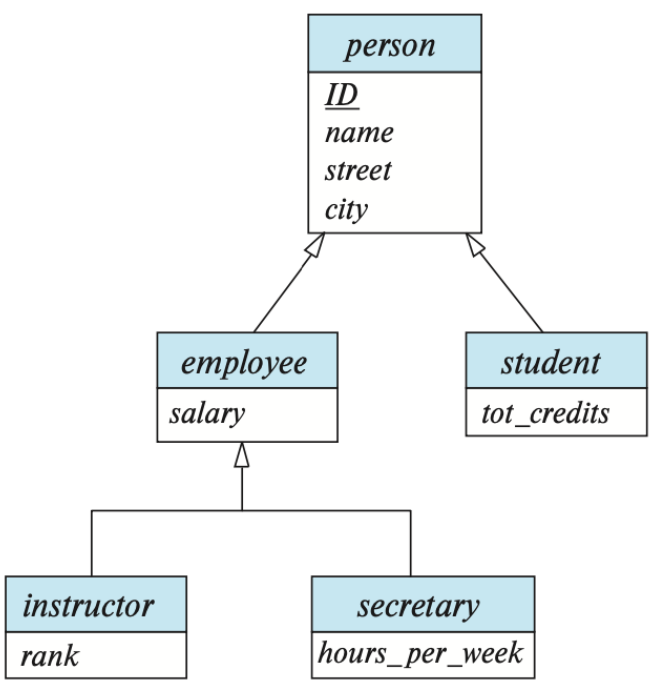


Specialization

- Top-down design process:
  - We designate sub-groupings within an entity set that are distinctive from other entities in the set
- These sub-groupings 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., *instructor* "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

Specialization Example

- **Overlapping** – *employee* and *student*
- **Disjoint** – *instructor* and *secretary*
- Total and partial



Representing Specialization via Schemas

- Method 1:
  - Form a schema for the higher-level entity
  - Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes

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

  - **Drawback**: getting information about, an *employee* requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema

Representing Specialization via Schemas

- Method 2:
  - Form a schema for each entity set with all local and inherited attributes

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

  - **Drawback**: *name, street* and *city* may be stored redundantly for people who are both students and employees

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

Completeness constraints

- To model an enterprise more accurately, the database designer may choose to place certain constraints on a particular generalization/specialization

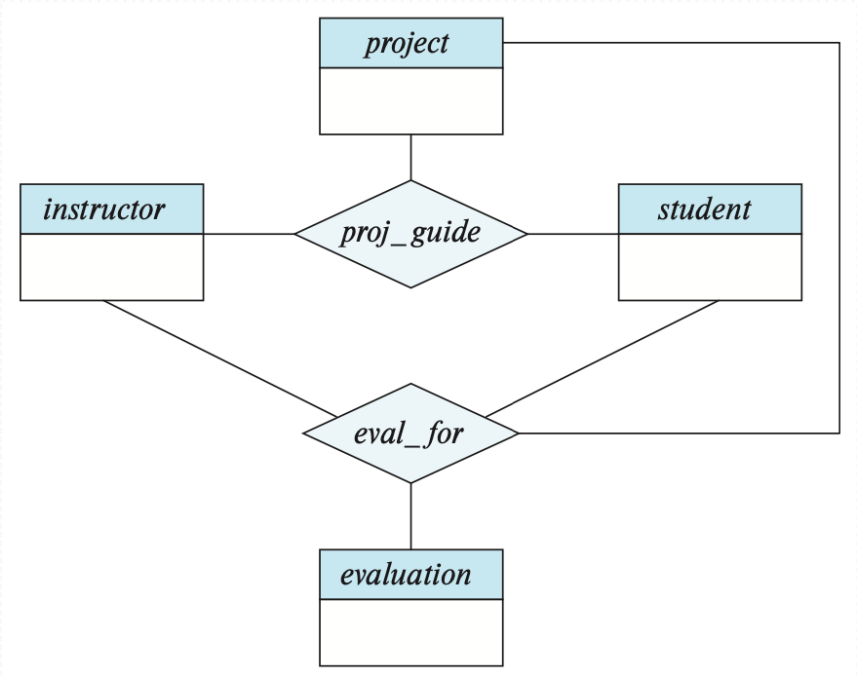
- **Completeness constraint** – specifies whether an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization
  - **Total specialization/generalization**:
    - Each higher-level entity must belong to a lower-level entity set
  - **Partial specialization/generalization**:
    - Some higher-level entities may not belong to any lower-level entity set

Completeness constraints

- Partial generalization is the default
- We can specify total generalization in an ER diagram by adding the keyword **total** in the diagram and drawing a dashed line from the keyword to the corresponding
  - hollow arrow-head to which it applies – **total** specialization
  - hollow arrow-heads to which it applies – **overlapping** specialization
- The *student* generalization is total
  - All student entities must be either graduate or undergraduate
  - Because the higher-level entity set arrived at through generalization is generally composed of only those entities in the lower-level entity sets, the completeness constraint for a generalized higher-level entity set is usually total

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

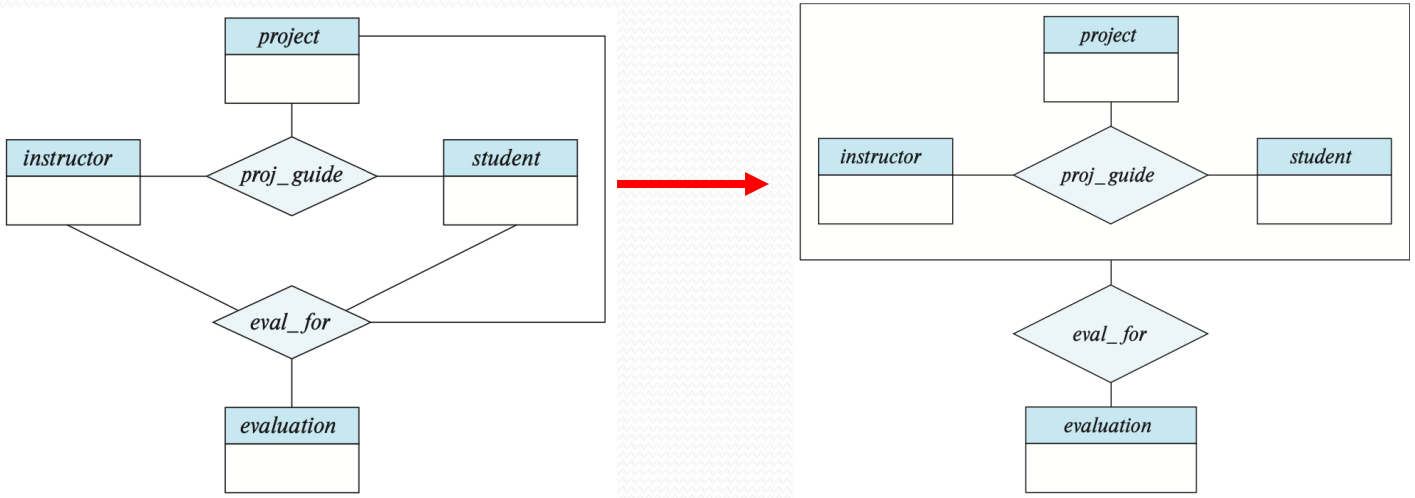


Aggregation

- 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
- Eliminate this redundancy via **aggregation**
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

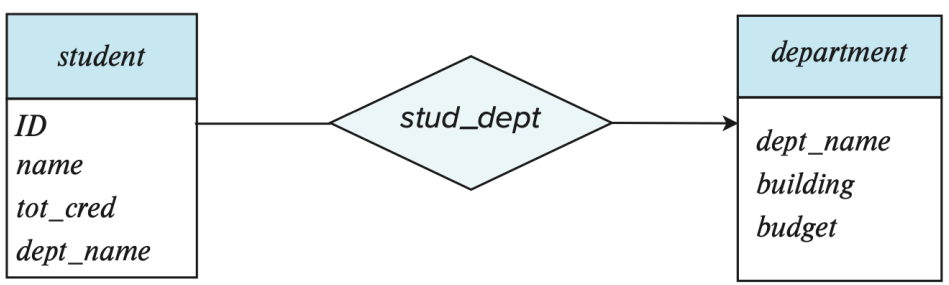
Aggregation

- Eliminate this redundancy via **aggregation** without introducing redundancy, 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



Common Mistakes in E-R Diagrams

- A common mistake when creating E-R models is the use of the primary key of an entity set as an attribute of another entity set, instead of using a relationship

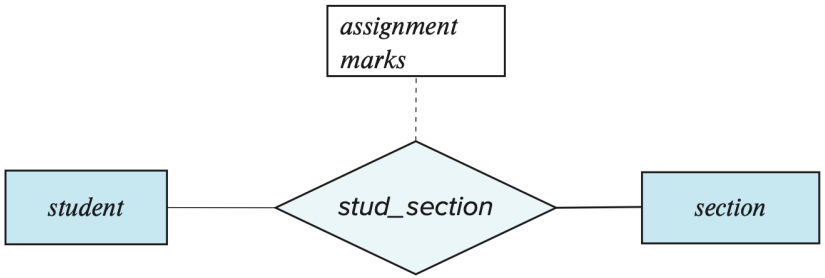


(a) Incorrect use of attribute

- The relationship stud_dept is the correct way to represent this information in the E-R model, since it makes the relationship between student and department explicit, rather than implicit via an attribute
- Having an attribute dept_name as well as a relationship stud_dept would result in duplication of information

Common Mistakes in E-R Diagrams

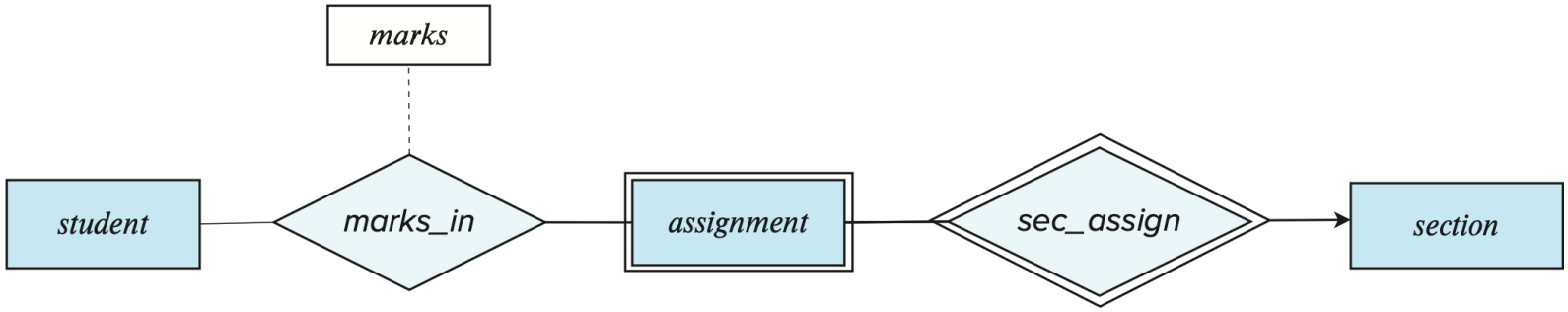
- A common mistake is to use a relationship with a single-valued attribute in a situation that requires a multivalued attribute



(b) Erroneous use of relationship attributes

- Suppose we decided to represent the marks that a student gets on different assignments of a course offering (section)
- A wrong way of doing this would be to add two attributes assignment and marks to the relationship takes
- We can only represent a single assignment for a given student-section pair

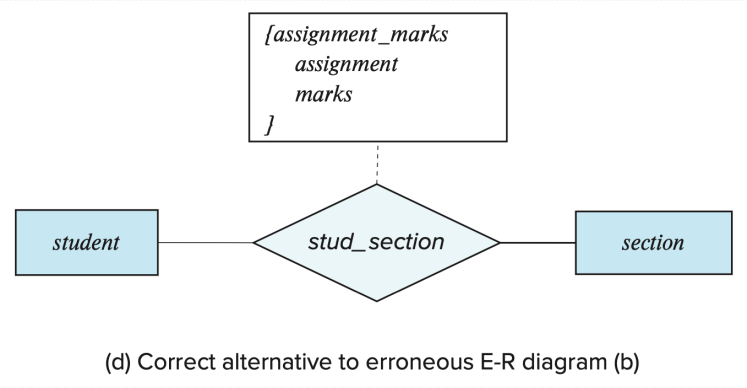
Common Mistakes in E-R Diagrams



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

- One solution to the problem is to model assignment as a weak entity identified by section, and to add a relationship marks in between assignment and student; the relationship would have an attribute marks

Common Mistakes in E-R Diagrams



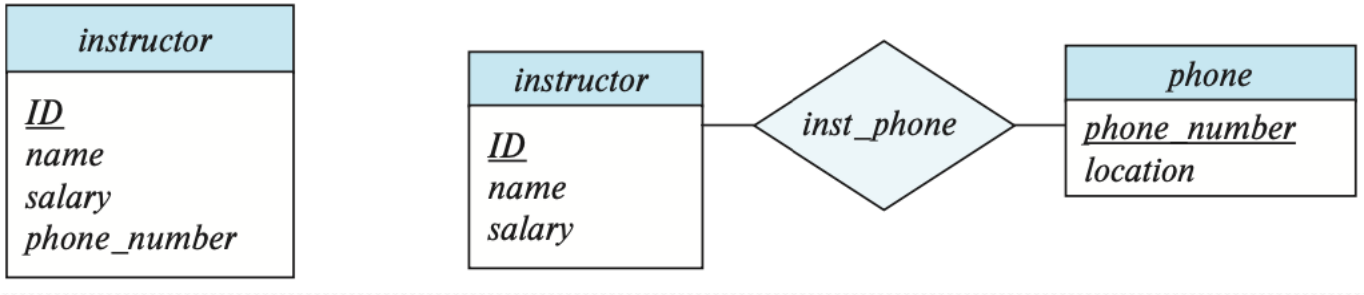
(d) Correct alternative to erroneous E-R diagram (b)

- An alternative solution is to use a multivalued composite attribute {assignment marks} to takes, where assignment marks has component attributes assignment and marks

- Modeling an assignment as a weak entity is preferable in this case, since it allows recording other information about the assignment, such as maximum marks or deadlines

Entities vs. Attributes

- Use of entity sets vs. attributes



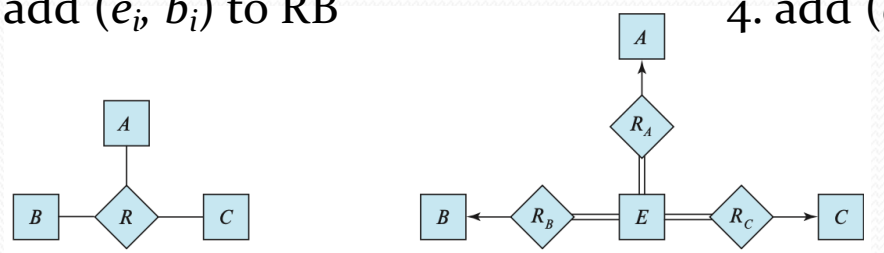
- Treating a phone as an entity better models a situation where one may want to keep extra information about a phone, such as: location or type
- Treating phone as an entity is more general than treating it as an attribute and is appropriate when the generality may be useful

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
  - E.g., 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
    - E.g., *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 RA
    3. add (e_i, b_i) to RB
    4. add (e_i, c_i) to RC



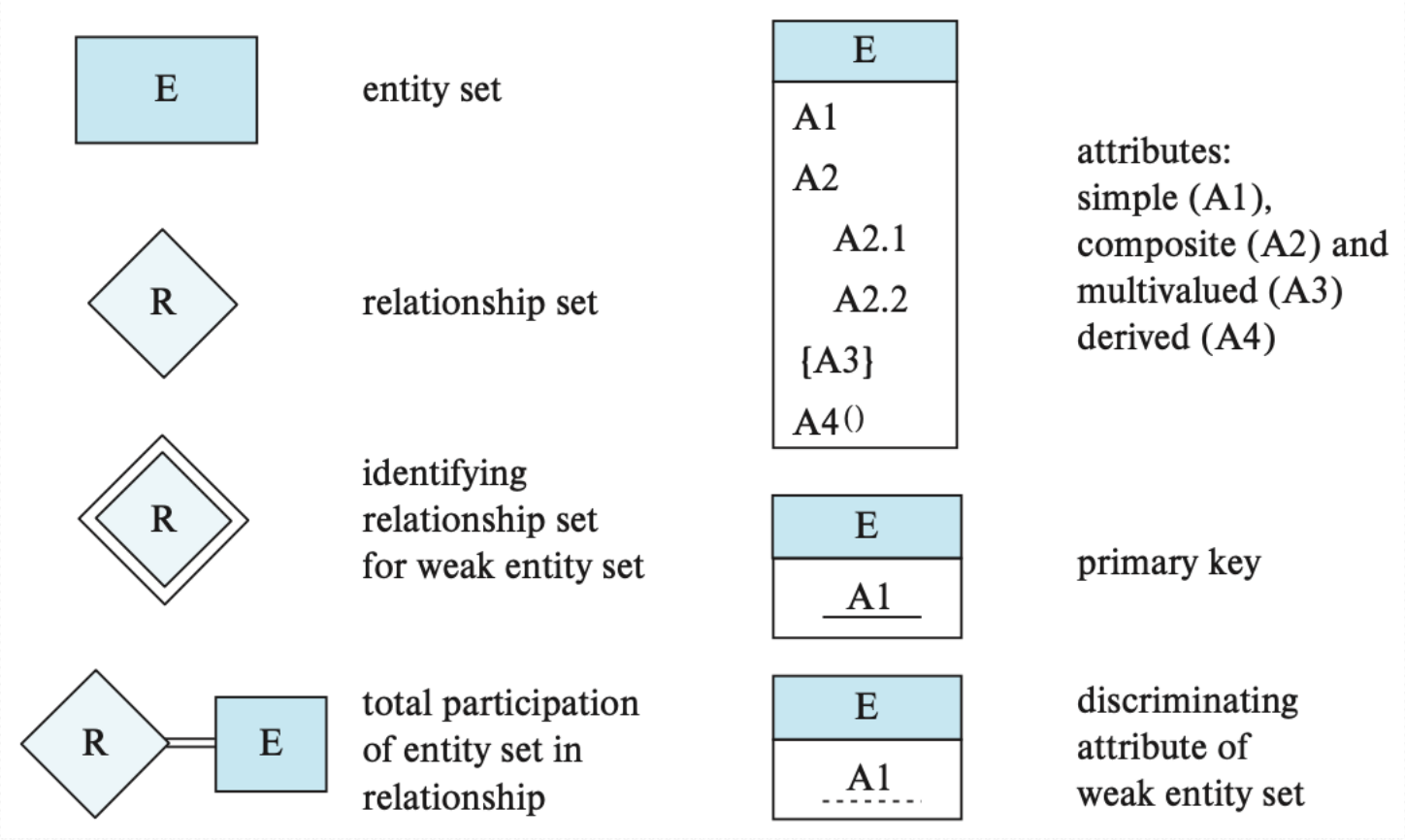
Converting Non-Binary Relationships to Binary Form

- 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 (described shortly) identified by the three relationship sets

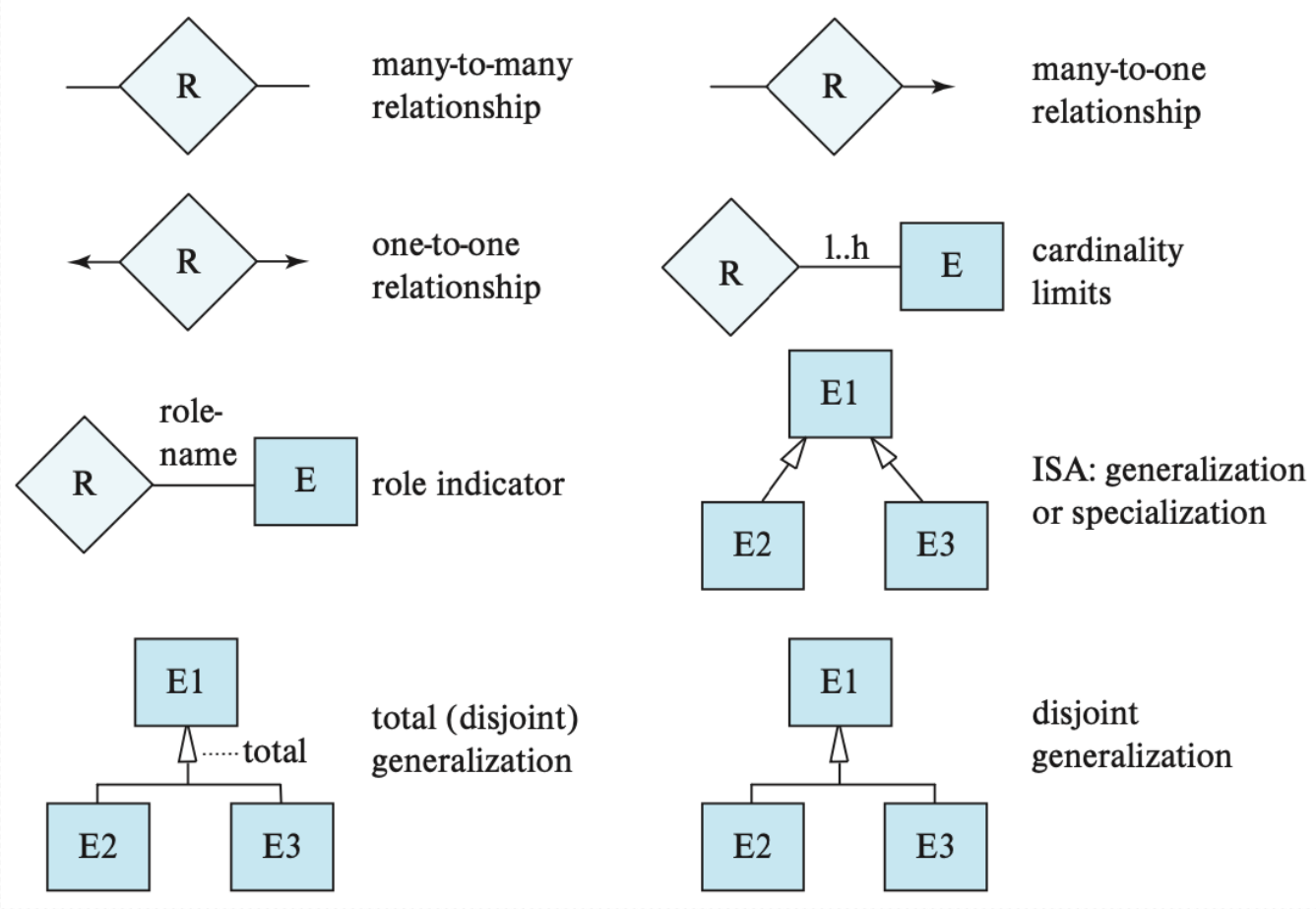
E-R Design Decisions

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

Summary of Symbols Used in E-R Notation

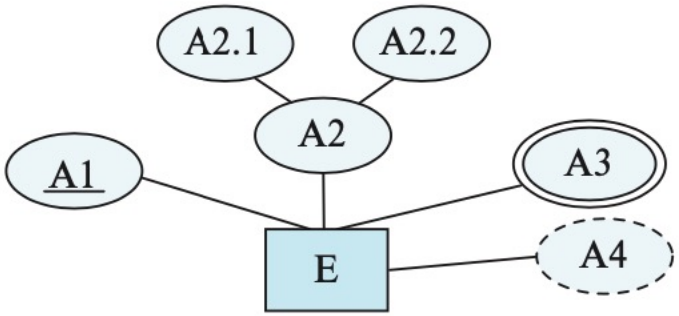


Summary of Symbols Used in E-R Notation



Alternative E-R Notations

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

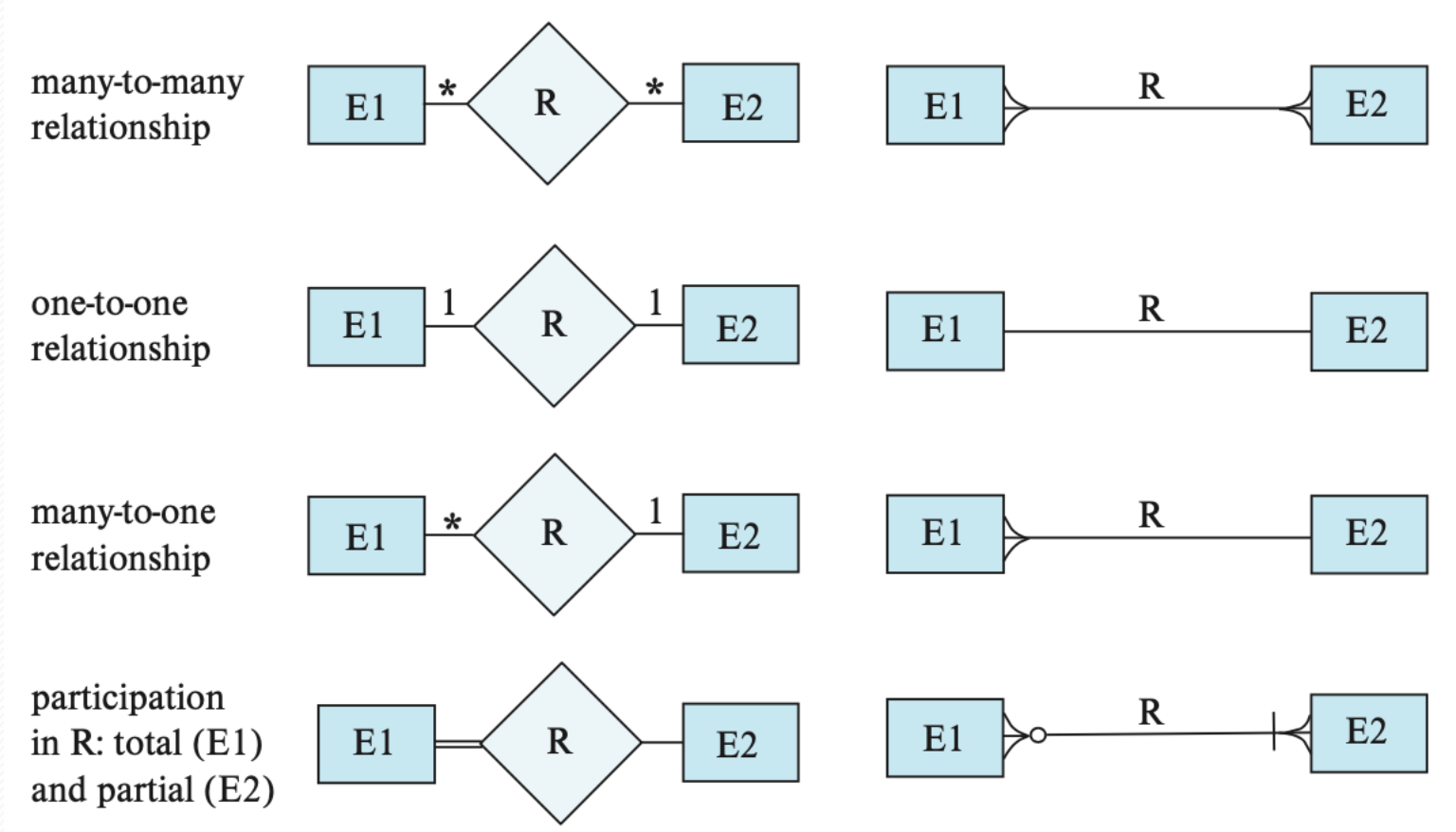


weak entity set

generalization

total
generalization

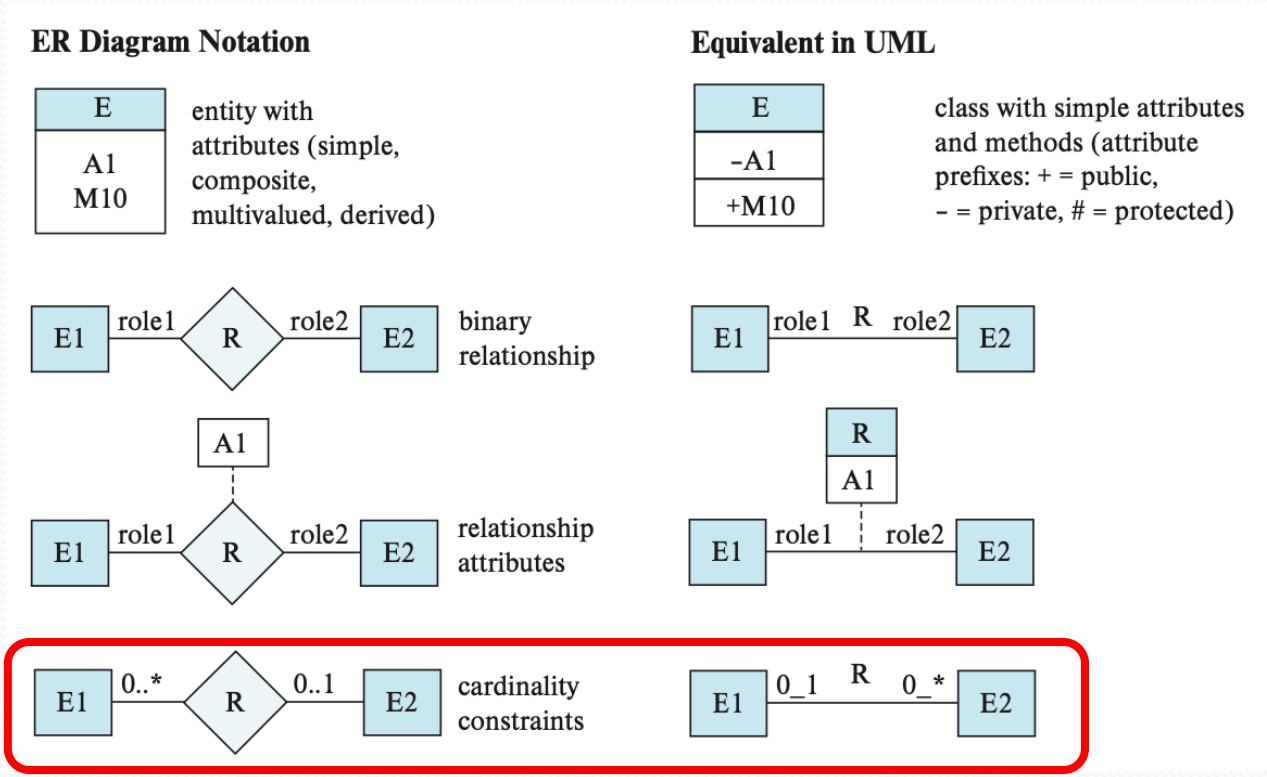
Alternative E-R Notations



UML

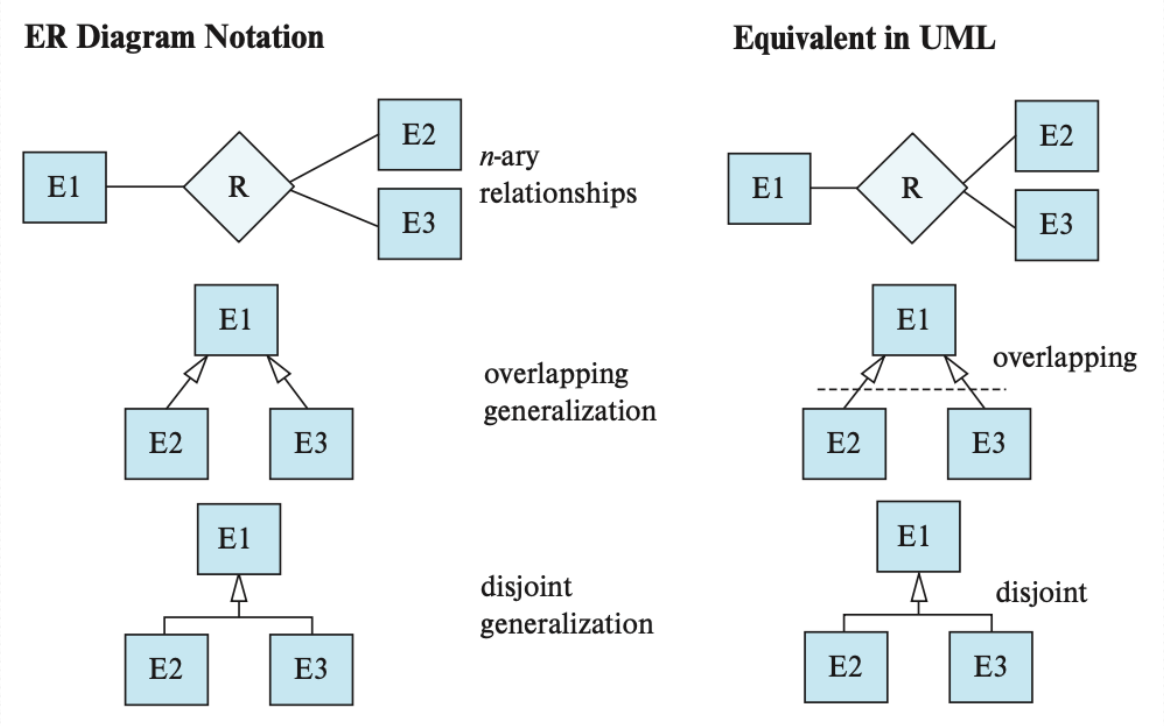
- **UML**: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- UML Class Diagrams correspond to E-R Diagram, but several differences

E-R vs. UML Class Diagrams



- Notice the reversal of position in cardinality constrain depiction

E-R vs. UML Class Diagrams

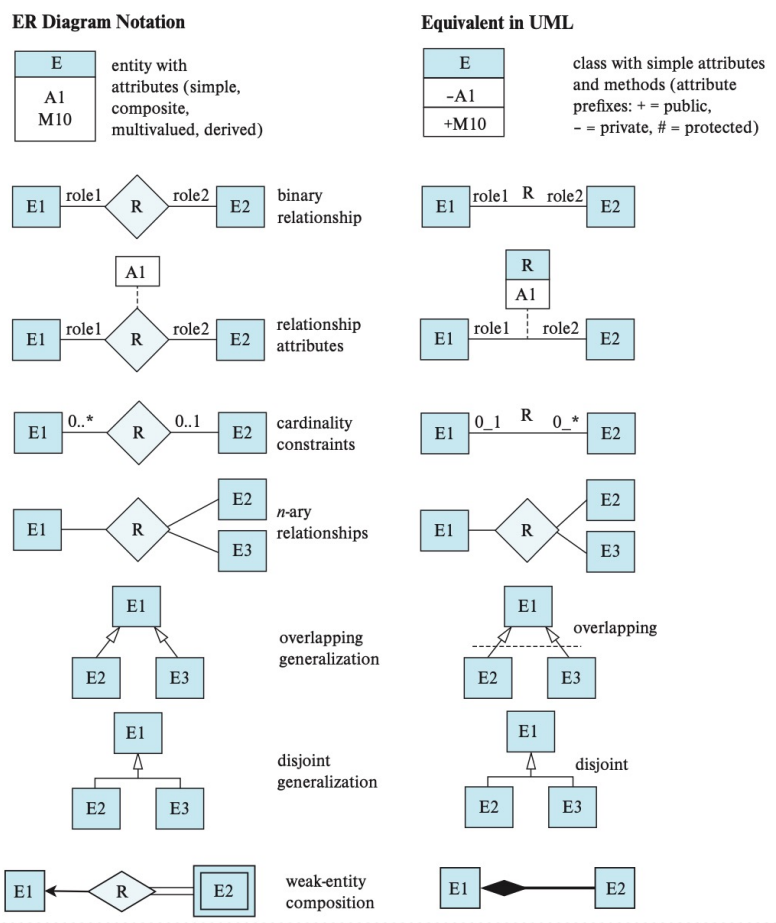


- Generalization can use merged or separate arrows independent of disjoint/overlapping

UML Class Diagrams

- 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

E-R vs. UML Class Diagrams



Other Aspects of Database Design

- Functional Requirements
- Data Flow, Workflow
- Schema Evolution

Acknowledgements

- Database design with UML and SQL, 4th edition
  - https://web.csulb.edu/colleges/coe/cecs/dbdesign/dbdesign.php?page=subclass.php