Database Design Using the E-R Model

Miao Qiao
The University of Auckland

Requirement Specification

- A car insurance company has customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars and has one or more premium payments associated with it. Each payment is for a particular period of time, and has an associated due date, and the date when the payment was received.

Design Process

- Conceptual Design – Capturing high-level data requirements
- Logical Design – Deciding on the database schema
- Physical Design – Deciding on the physical layout of the database

Conceptual Design Phases

- Initial phase -- characterize fully the data needs of the prospective database users.
- Second phase -- choosing a data model
- Final Phase -- moving from an abstract data model to the implementation of the database

- In designing a database schema, we must ensure that we avoid two major pitfalls:
  - Redundancy: a bad design may result in repeat information.
    - Redundant representation of information may lead to data inconsistency among the various copies of information
  - Incompleteness: a bad design may make certain aspects of the enterprise difficult or impossible to model.

ER model -- Database Modeling

- Entity Relationship Model: models an enterprise as
  - a collection of *entities* and *relationships*
    - **Entity**: a “thing” or “object” in the enterprise that is distinguishable from other objects
      - Described by a set of ***attributes***
    - **Relationship**: an association among several entities
  - Represented diagrammatically by an
    - *Entity-relationship diagram* (**ER diagram)**

- The ER data model employs three basic concepts:
  - entity sets,
  - relationship sets,
  - attributes.

Entity Sets

- An **entity** is an object that is *distinguishable* from other objects.
  - Example: specific person, company, event, plant
- An **entity set** is a set of entities of the same type (share the same properties).
  - Example: set of all persons, companies, trees, holidays
- An entity is represented by a set of attributes.
  - Example:

    instructor = (*ID, name, salary*)
    course= (*course_id, title, credits*)
- A subset of the attributes form a **primary key** of the entity set, uniquely identifying each member of the set.

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

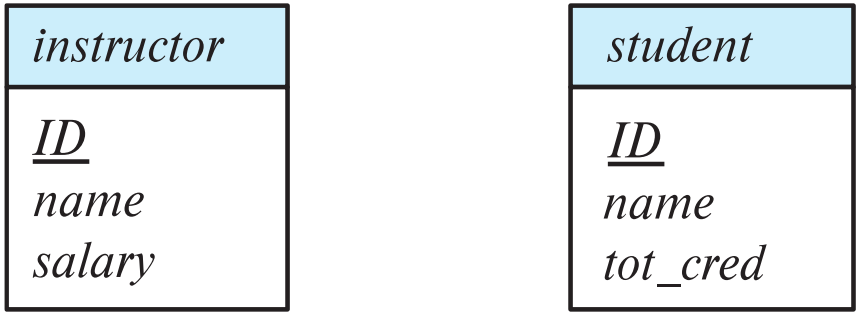
instructor

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

student

Representing Entity sets in ER Diagram

- Entity sets can be represented graphically as follows:
  - Rectangles represent entity sets.
  - Attributes listed inside entity rectangle
  - Underline indicates primary key attributes



Relationship Sets

- A **relationship** is an association among several entities

  Example:

  44553 (Peltier) — *advisor* — 22222 (Einstein)

  student entity — relationship set — *instructor* entity

- A **relationship set** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

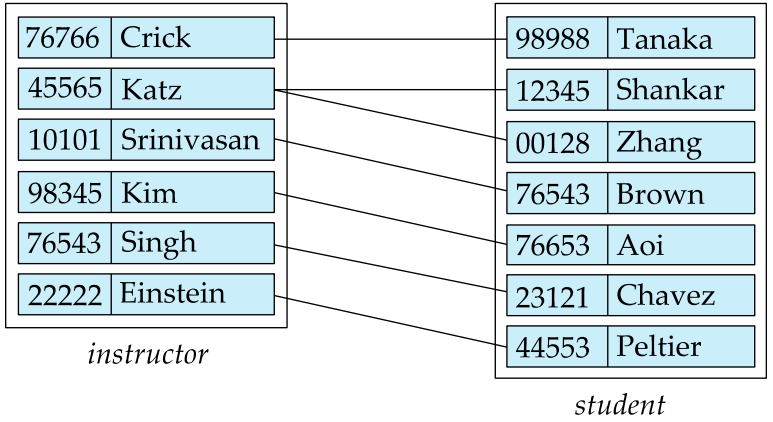
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

  where $(e_1, e_2, \ldots, e_n)$ is a relationship

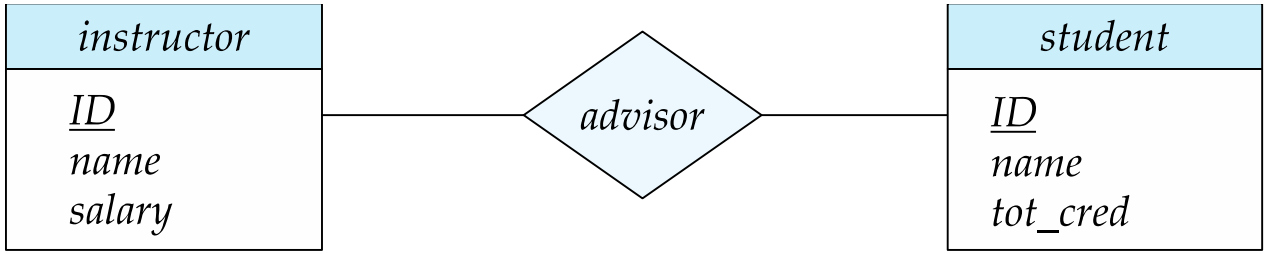
  - Example:

    $(44553,22222) \in advisor$

- Example: relationship set *advisor*.

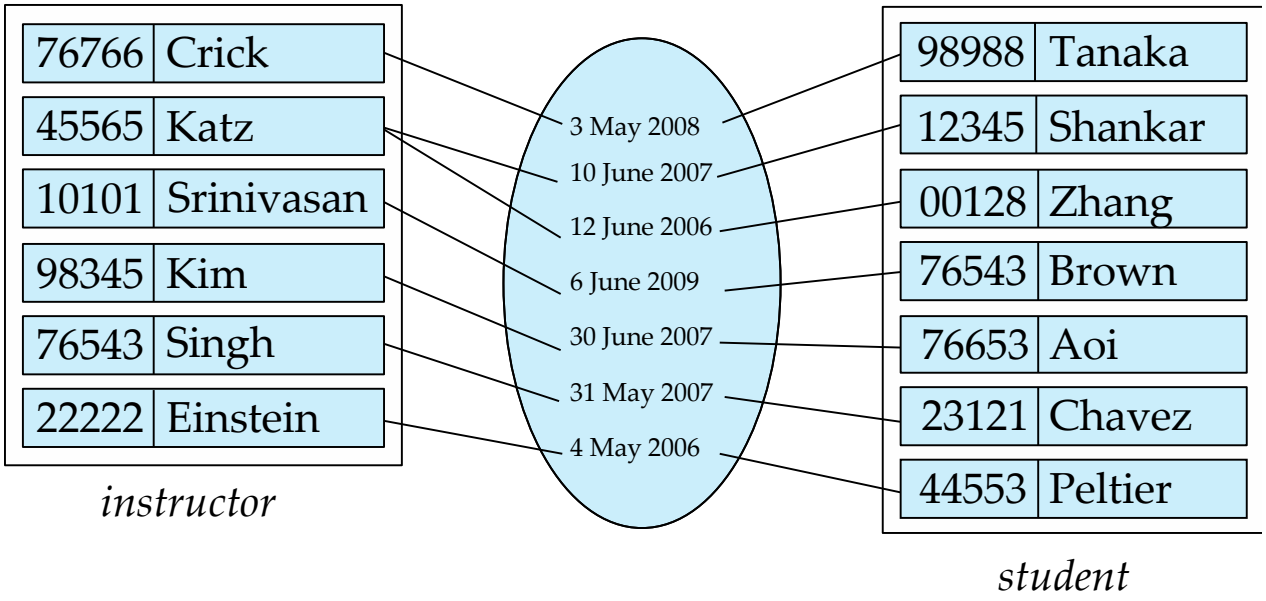


- ER Diagram
  - Diamonds represent relationship sets.

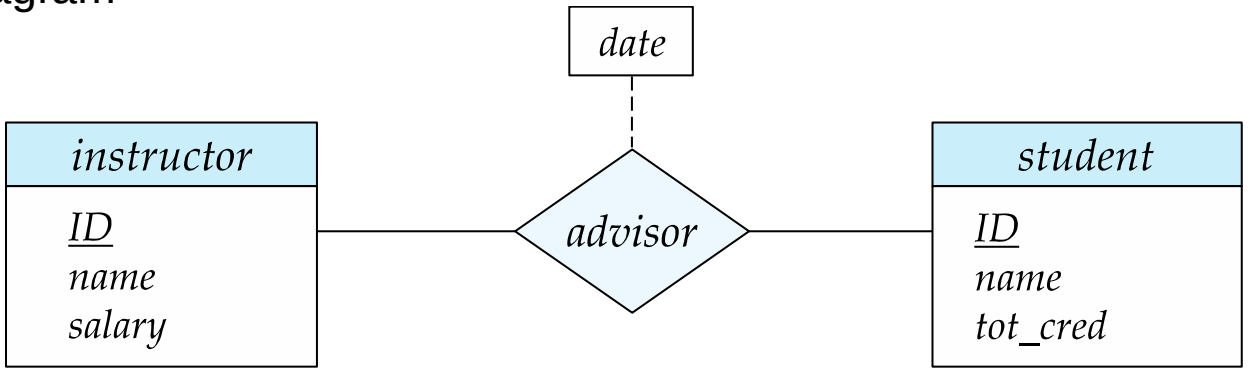


Relationship Sets (Cont.)

- An attribute can also be associated with a relationship set.

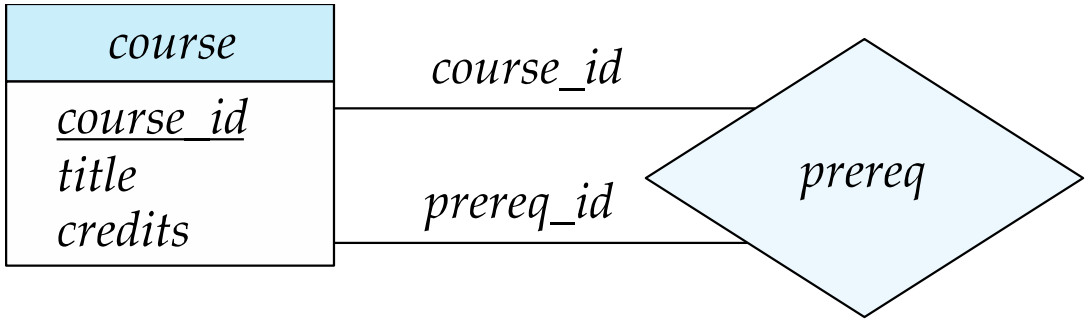


- ER Diagram



Roles

- Entity sets of a relationship need not be distinct
  - Each occurrence of an entity set plays a “role” in the relationship
- The labels “*course_id*” and “*prereq_id*” are called **roles**.

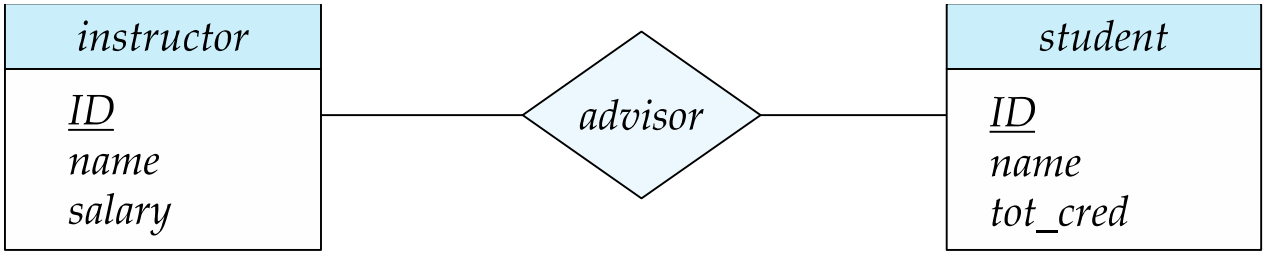


Degree of a Relationship Set

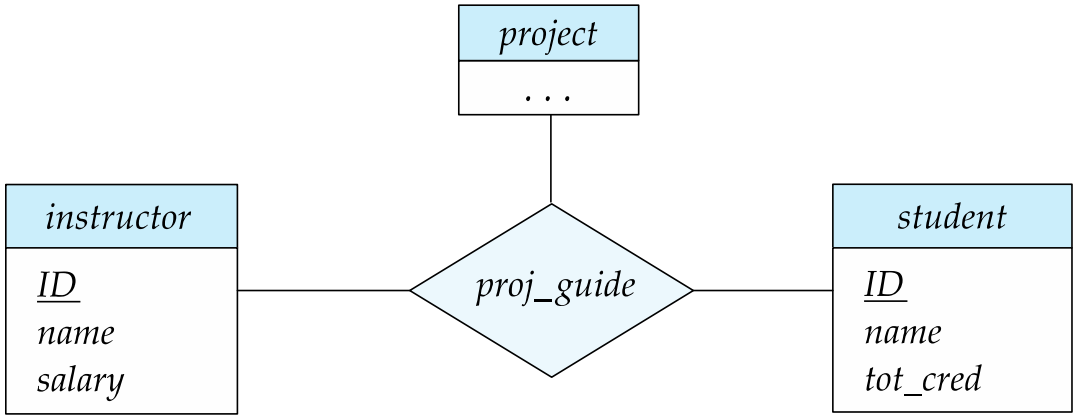
- Binary relationship
  - involve two entity sets (or degree two).
  - most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - relationship *proj_guide* is a ternary relationship between *instructor, student,* and *project*

Non-binary Relationship Sets

- Most relationship sets are binary



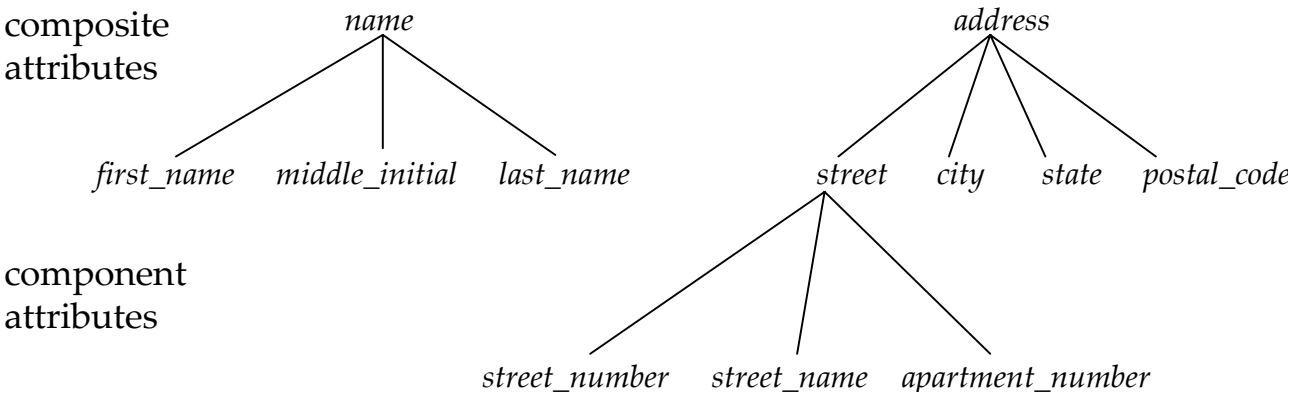
- E-R Diagram with a Ternary Relationship



Complex Attributes

- Attribute types: ER diagram =>
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes
    - Example: multivalued attribute: *phone_numbers*
  - **Derived** attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth
- **Domain** – the set of permitted values for each attribute
- **Composite attributes** allow us to divided attributes into subparts (other attributes).

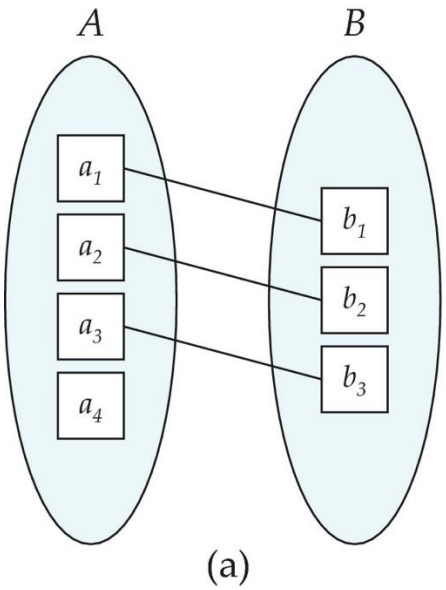
instructor
ID
name
first_name
middle_initial
last_name
address
street
street_number
street_name
apt_number
city
state
zip
{ phone_number }
date_of_birth
age ( )



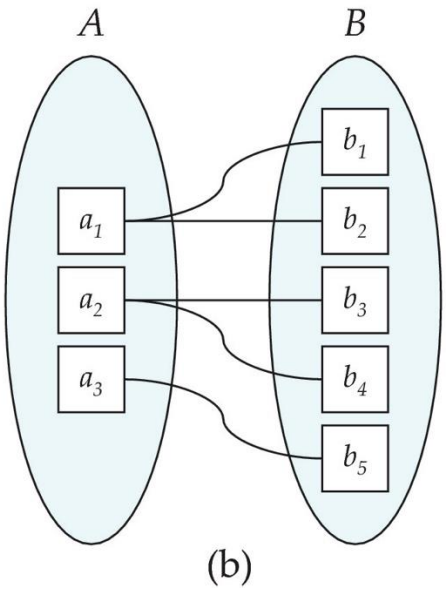
Mapping Cardinality Constraints

- Indicate how many instances of one entity can be linked to an instance of another entity through a relationship set.

- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many



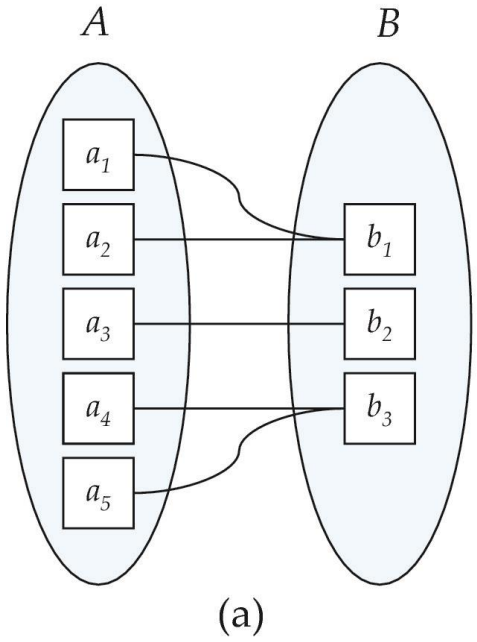
One to one



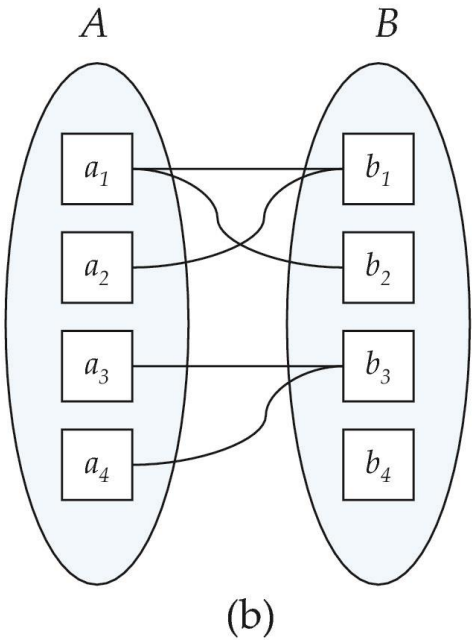
One to many

Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

Mapping Cardinalities



Many to one



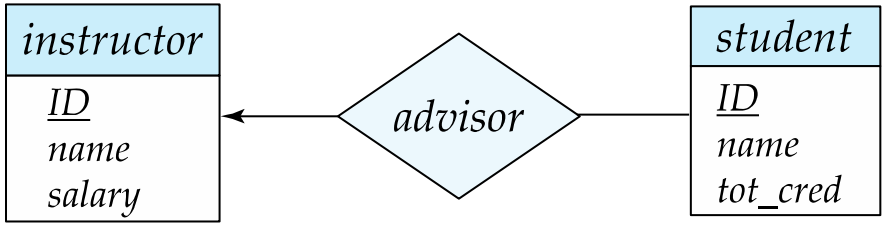
Many to many

Note: Some elements in A and B may not be mapped to any elements in the other set

Representing Cardinality Constraints in ER Diagram

- Between the relationship set and the entity set.
  - (→), signifying "one"
  - (—), signifying "many"
- One-to-one relationship between an *instructor* and a *student*:
  - A student is associated with at most one *instructor* via relationship *advisor*
  - A *student* is associated with at most one *department* via *stud_dept*

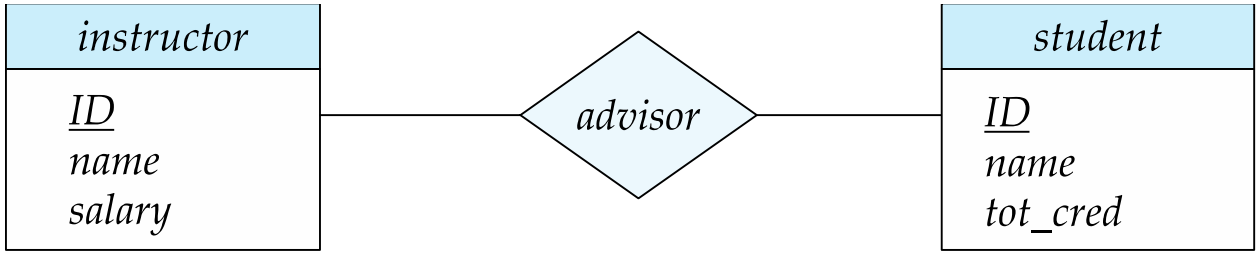
- One-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor*
  - a student is associated with at most one instructor via advisor,



Many-to-One Relationships

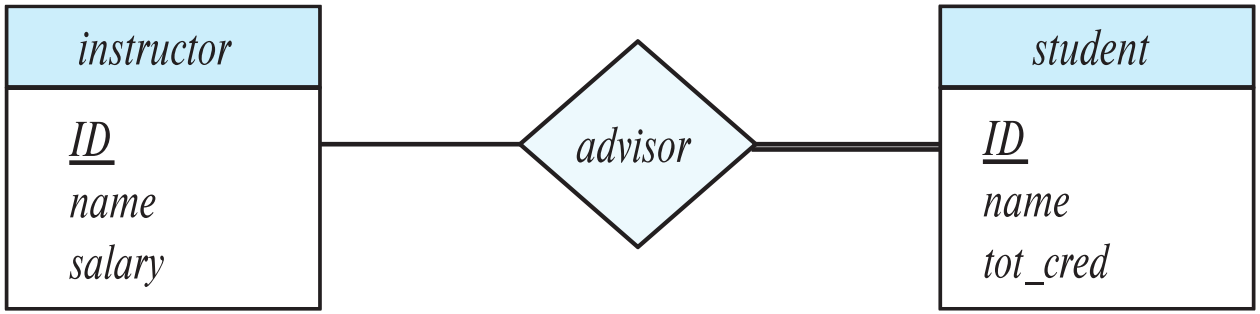
- In a many-to-one relationship between an *instructor* and a *student,*
  - an instructor is associated with at most one student via *advisor*,
  - and a student is associated with several (including 0) instructors via *advisor*

- In a many-to-many relationship between an *instructor* and a *student,*
  - an instructor is associated with several (possibly 0) students via *advisor*
  - a student is associated with several (possibly 0) instructors via *advisor*



Total and Partial Participation

- **Total participation** (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set



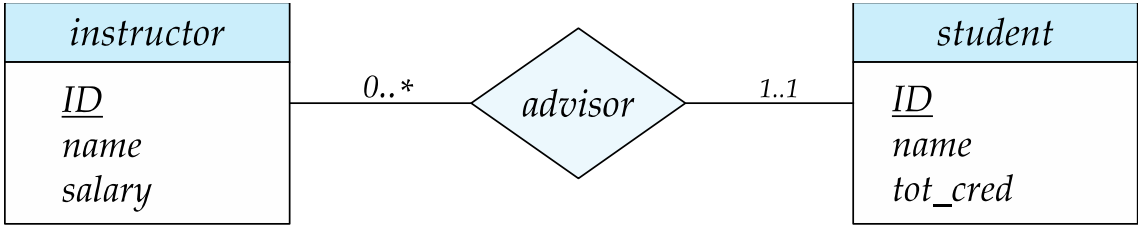
participation of *student* in *advisor* relation is total

  - every *student* must have an associated instructor

- **Partial participation**: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial

Notation for Expressing More Complex Constraints

- A line may have an associated minimum and maximum cardinality, shown in the form *l..h*, where *l* is the minimum and *h* the maximum cardinality
  - A minimum value of 1 indicates total participation.
  - A maximum value of 1 indicates that the entity participates in at most one relationship
  - A maximum value of * indicates no limit.
- Example



  - Instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

Cardinality Constraints on Ternary Relationship

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- For example, an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project
- To avoid confusion we outlaw more than one arrow

Primary Key

- Primary keys provide a way to specify how entities and relations are distinguished. We will consider:
  - Entity sets
  - Relationship sets (demonstration)
  - Weak entity sets

Primary key for Entity Sets

- By definition, individual entities are distinct.
- From database perspective, the differences among them must be expressed in terms of their attributes.
- The values of the attribute values of an entity must be such that they can uniquely identify the entity.
  - No two entities in an entity set are allowed to have exactly the same value for all attributes.
- A key for an entity is a set of attributes that suffice to distinguish entities from each other

Primary Key for Relationship Sets

- Use the primary keys of the entities in the relationship set
  - Let *R* be a relationship set involving entity sets E1, E2, .. En
  - The primary key for R consists of the union of the primary keys of entity sets E1, E2, ..En
  - If the relationship set *R* has attributes a1, a2, .., am associated with it, then the primary key of *R* also includes the attributes a1, a2, .., am
- Example: relationship set “advisor”.
  - The primary key consists of *instructor.ID* and s*tudent.ID*
- The choice of the primary key for a relationship set depends on the mapping cardinality of the relationship set.

Choice of Primary key for Binary Relationship

- Many-to-Many relationships. The preceding union of the primary keys is a minimal superkey and is chosen as the primary key.
- One-to-Many relationships . The primary key of the “Many” side is a minimal superkey and is used as the primary key.
- Many-to-one relationships. The primary key of the “Many” side is a minimal superkey and is used as the primary key.
- One-to-one relationships. The primary key of either one of the participating entity sets forms a minimal superkey, and either one can be chosen as the primary key.

Weak Entity Sets

- A **weak entity set** is one whose existence is dependent on another entity, called its **identifying entity**
- Instead of associating a primary key with a weak entity, we use the identifying entity, along with extra attributes called **discriminator** to uniquely identify a weak entity.

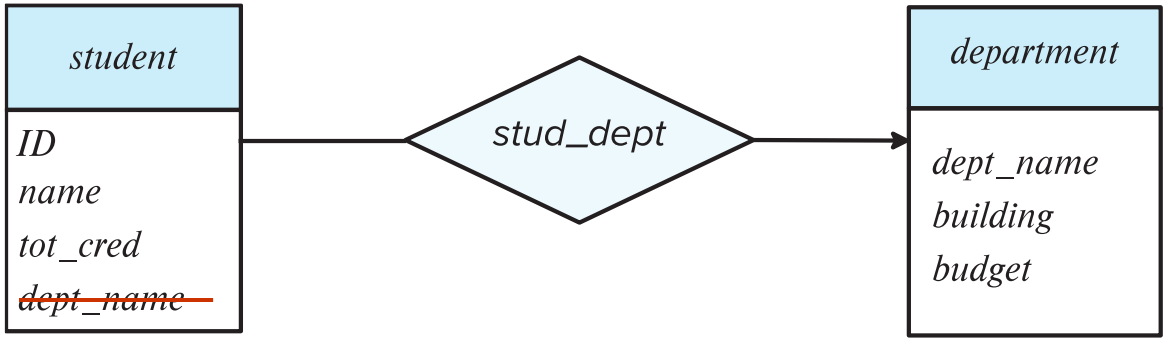
- In E-R diagrams, a weak entity set is depicted via a double rectangle.
- We underline the discriminator of a weak entity set with a dashed line.
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a double diamond.
- Primary key for *section* – (*course_id, sec_id, semester, year*)

Weak Entity Sets (Cont.)

- An entity set that is not a weak entity set is termed a **strong entity set**.
- Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set.
- The identifying entity set is said to **own** the weak entity set that it identifies.
- The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.
- Note that the relational schema we eventually create from the entity set *section* does have the attribute *course_id*, for reasons that will become clear later, even though we have dropped the attribute *course_id* from the entity set *section.*

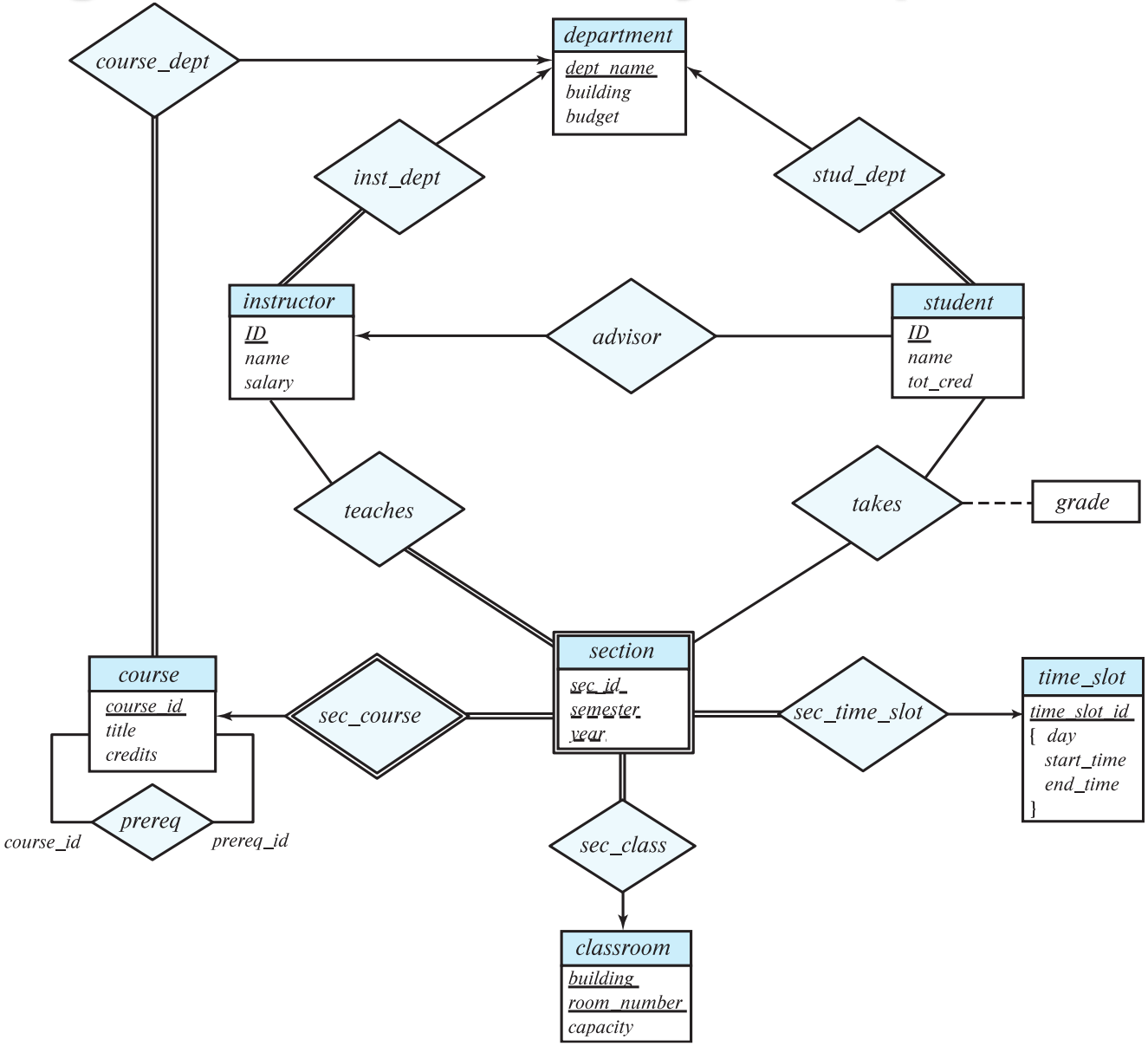
Redundant Attributes

- 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.



(a) Incorrect use of attribute

E-R Diagram for a University Enterprise



ER Diagram Challenge: Schema Sketch

Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars and has one or more premium payments associated with it. Each payment is for a particular period of time, and has an associated due date, and the date when the payment was received.

What we have discussed so far:

- The Entity-Relationship Model
- Complex Attributes
- Mapping Cardinalities
- Primary Key
- Removing Redundant Attributes in Entity Sets

Reduction to Relation Schemas

ER-to-Relational Mapping Algorithm

- Step 1: Mapping of Regular Entity Types
- Step 2: Mapping of Weak Entity Types
- Step 3: Mapping of Binary 1:1 (One to One) Relation Types
- Step 4: Mapping of Binary 1:N (One to Many, Many to One) Relationship Types.
- Step 5: Mapping of Binary M:N (Many to Many) Relationship Types.
- Step 6: Mapping of Multivalued attributes.
- Step 7: Mapping of N-ary Relationship Types.

GOALS during Mapping

- Preserve all information (that includes all attributes)
- Maintain the constraints to the extent possible (Relational Model cannot preserve all constraints – e.g., max cardinality ratio such as 1:10 in an ER relationship type).
- Minimize null values

The mapping procedure described has been implemented in many commercial tools.

- MySQL Workbench: allow you to transform your ER diagrams into SQL scripts or directly into a live database.
- dbdiagram.io: A simple tool to draw ER diagrams by writing code, which can then be exported as SQL statements.
- pgModeler: An open-source PostgreSQL Database Modeler that supports the creation of database models and the generation of SQL scripts
- Vertabelo

ER-to-Relational Mapping Algorithm

- **Step 1: Mapping of Regular Entity Types.**
  - For each regular (strong) entity type E in the ER diagram, create a relation R that includes all the simple attributes of E.
  - Composite attributes are represented by their ungrouped components.
  - Choose one of the key attributes of E as the primary key for R.
  - If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

- **Step 2: Mapping of Weak Entity Types**
  - For each weak entity type W in the ER diagram with owner entity type E, create a relation R & include all simple attributes (or simple components of composite attributes) of W as attributes of R.
  - Also, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
  - The primary key of R is the *combination of* the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

Representing Entity Sets

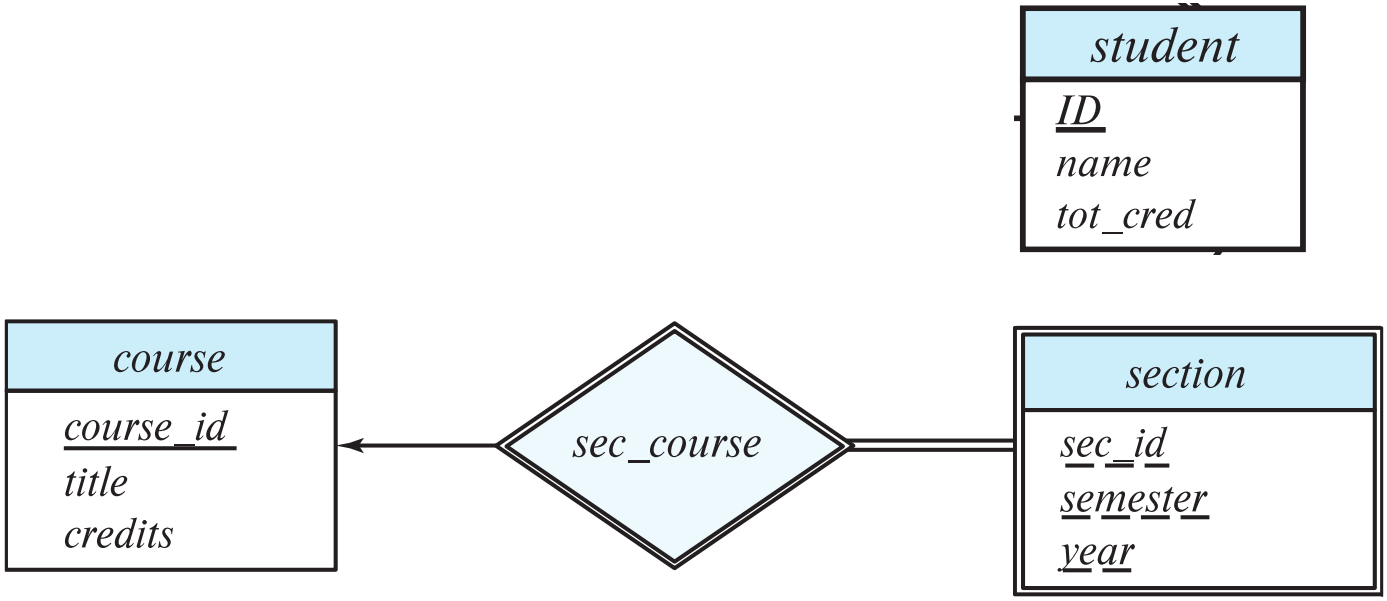
- A strong entity set reduces to a schema with the same attributes

  student(ID, name, tot_cred)

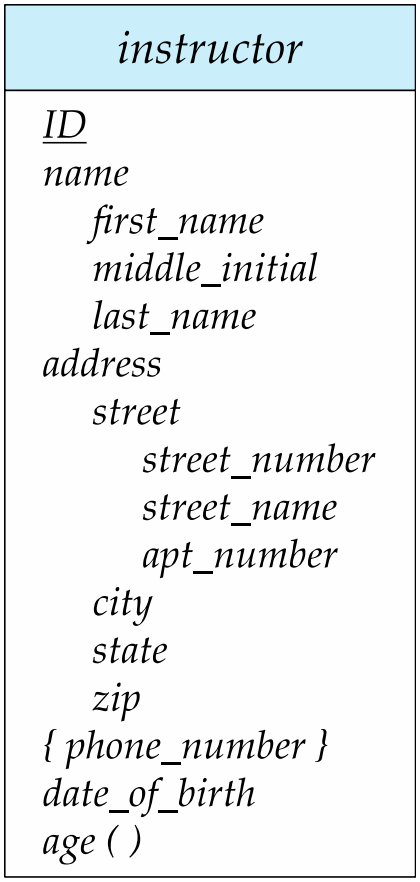
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

  section (course_id, sec_id, sem, year)

- Example



Representation of Entity Sets with Composite Attributes



- **Composite attributes are flattened out** by creating a separate attribute for each component attribute
  - Example: given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name* and *name_last_name*
    - Prefix omitted if there is no ambiguity (*name_first_name* could be *first_name)*
- **Ignoring multivalued attributes**, extended instructor schema is
  - *instructor(ID, first_name, middle_initial, last_name, street_number, street_name, apt_number, city, state, zip_code, date_of_birth)*

ER-to-Relational Mapping Algorithm

- **Step 3: Mapping of Binary 1:1 Relation Types**
  - For each binary 1:1 relationship type R in the ER diagram, identify the relations S and T that correspond to the entity types participating in R.
  - There are three possible approaches:
    - **Foreign Key (2 relations) approach:** Choose one of the relations (e.g., S) and include a foreign key in S the primary key of T. It is better to choose an entity type total participation in R; otherwise, there might be null values in the foreign key.
    - **Merged relation (1 relation) option:** An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.
    - **Cross-reference or relationship relation (3 relations) option:** The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

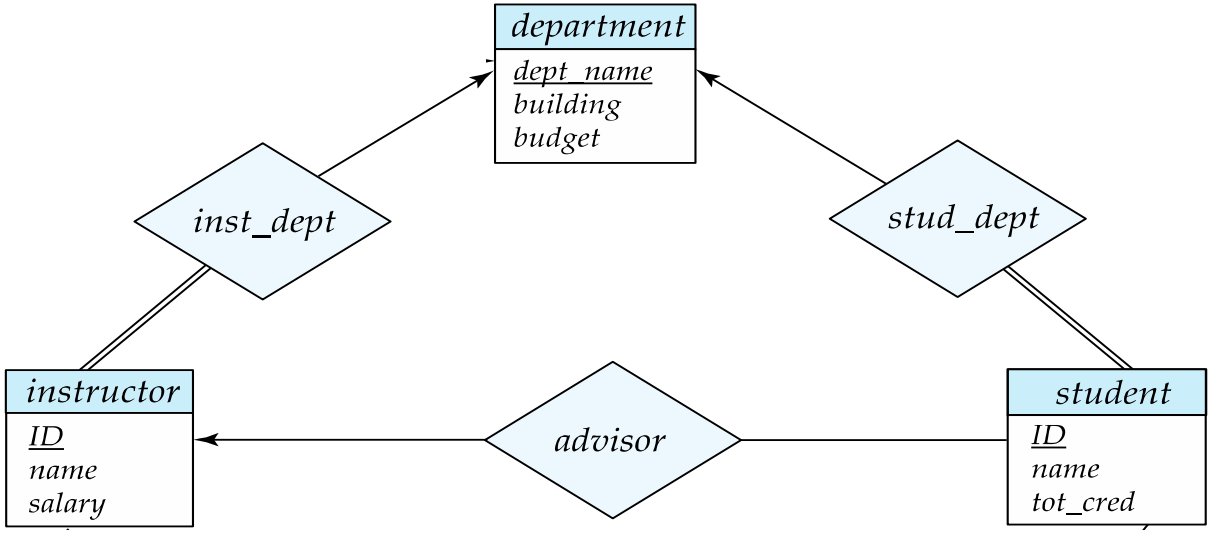
ER-to-Relational Mapping Algorithm

- **Step 4: Mapping of Binary 1:N (One to Many) Relationship Types.**
  - **The foreign key approach:**
    - For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side (many side) of the relationship type.
    - Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
    - Include any simple attributes of the 1:N relation type as attributes of S.
  - **The relationship relation approach**: An alternative approach uses a relationship relation (cross referencing relation) option as in the third option for binary 1:1 relationships.

Redundancy of Schemas

- **The foreign key approach**
- Example: Instead of creating a schema for relationship set *inst_dept*, add an attribute *dept_name* to the schema arising from entity set *instructor*
  - *instructor(ID, name, dept_name, salary,*

    foreign key (dept_name) references department (dept_name));



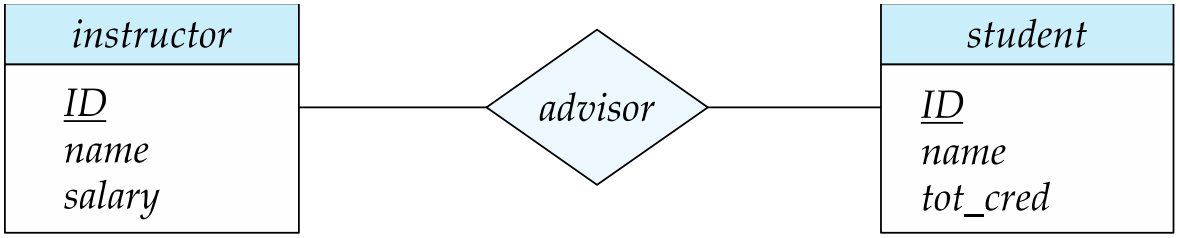
ER-to-Relational Mapping Algorithm

- **Step 5: Mapping of Binary M:N (Many to Many) Relationship Types.**
  - **The relationship relation (cross-reference) option:**
    - For each regular binary M:N relationship type R, *create a new relation* S to represent R. This is a *relationship relation.*
    - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; *their combination will form the primary key* of S.
    - Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

Example: Representing 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.
- Example: schema for relationship set *advisor*

advisor = (*s_id*, *i_id*)



ER-to-Relational Mapping Algorithm

- **Step 6: Mapping of Multivalued attributes.**
  - For each multivalued attribute A, create a new relation R.
  - This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
  - The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

Representation of Entity Sets with Multivalued Attributes

- A 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*
- Example: Multivalued attribute *phone_number* of *instructor* is represented by a schema:
  inst_phone= (*ID*, *phone_number*)
- Each value of the multivalued attribute maps to a separate tuple of the relation on schema *EM*
  - For example, an *instructor* entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples:
    (22222, 456-7890) and (22222, 123-4567)

ER-to-Relational Mapping Algorithm

- **Step 7: Mapping of N-ary Relationship Types.**
  - For each n-ary relationship type R, where n>2, create a new relationship S to represent R.
  - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
  - Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.

Summary of Mapping constructs and constraints

Table 9.1 Correspondence between ER and Relational Models

ER MODEL	RELATIONAL MODEL
Entity type	*Entity* relation
1:1 or 1:N relationship type	Foreign key (or *relationship* relation)
M:N relationship type	*Relationship* relation and *two* foreign keys
n-ary relationship type	*Relationship* relation and *n* foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

Summary

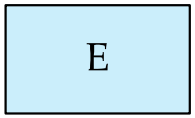
- The Relational Model has been implemented in many commercial and open source DBMSs.
- We present the algorithms for mappings form ER diagram into Relation Model, which has been implemented in many tools.
- **ER-to-Relational Mapping Algorithm**
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relation Types
  - Step 4: Mapping of Binary 1:N Relationship Types.
  - Step 5: Mapping of Binary M:N Relationship Types.
  - Step 6: Mapping of Multivalued attributes.
  - Step 7: Mapping of N-ary Relationship Types.

Redundancy of Schemas (Cont.)

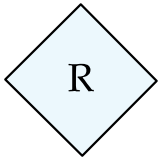
- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
- Example: The *section* schema already contains the attributes that would appear in the *sec_course* schema

Summary of Symbols Used in E-R Notation

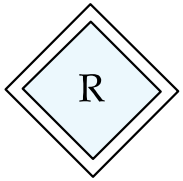
Summary of Symbols Used in E-R Notation



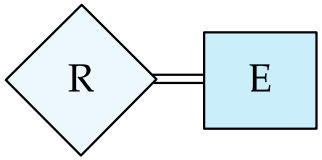
entity set



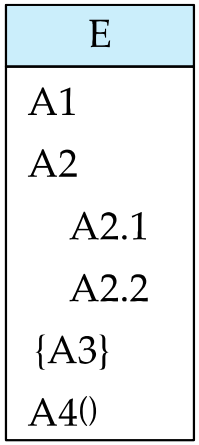
relationship set



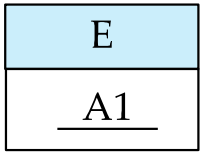
identifying relationship set for weak entity set



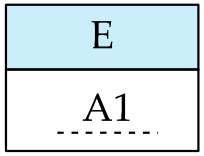
total participation of entity set in relationship



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

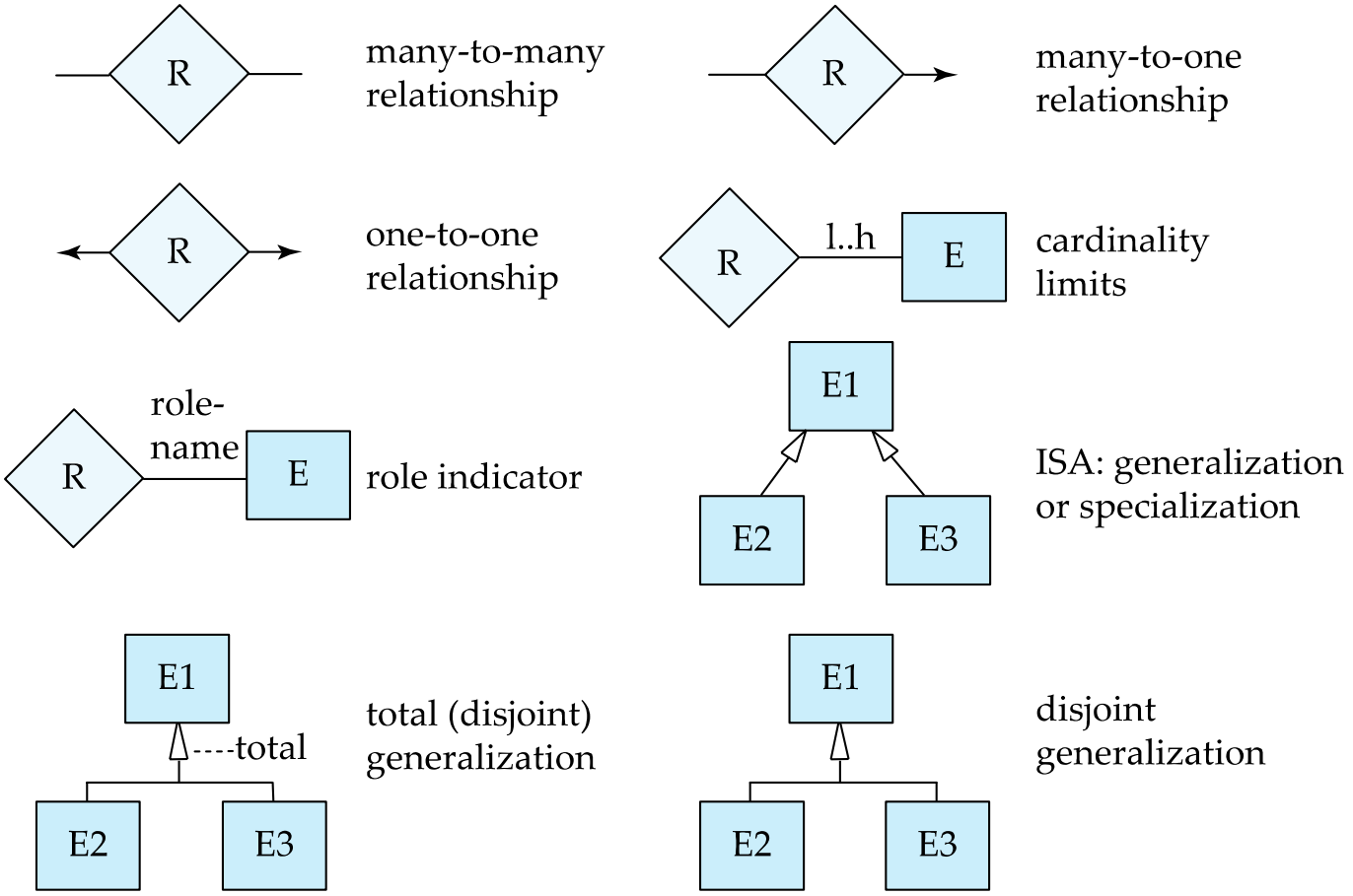


primary key



discriminating attribute of weak entity set

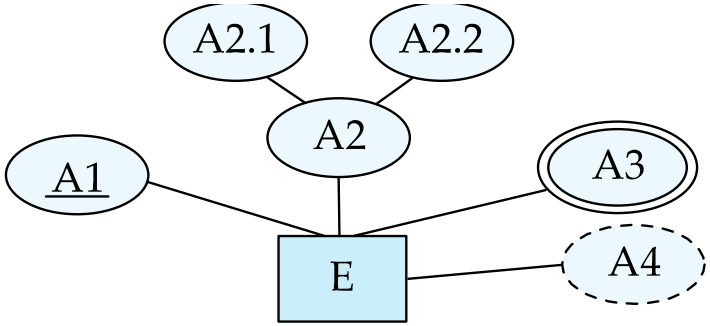
Symbols Used in E-R Notation (Cont.)



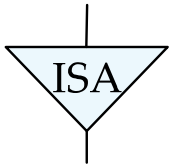
Alternative ER Notations

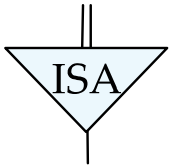
- Chen, IDE1FX, …

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

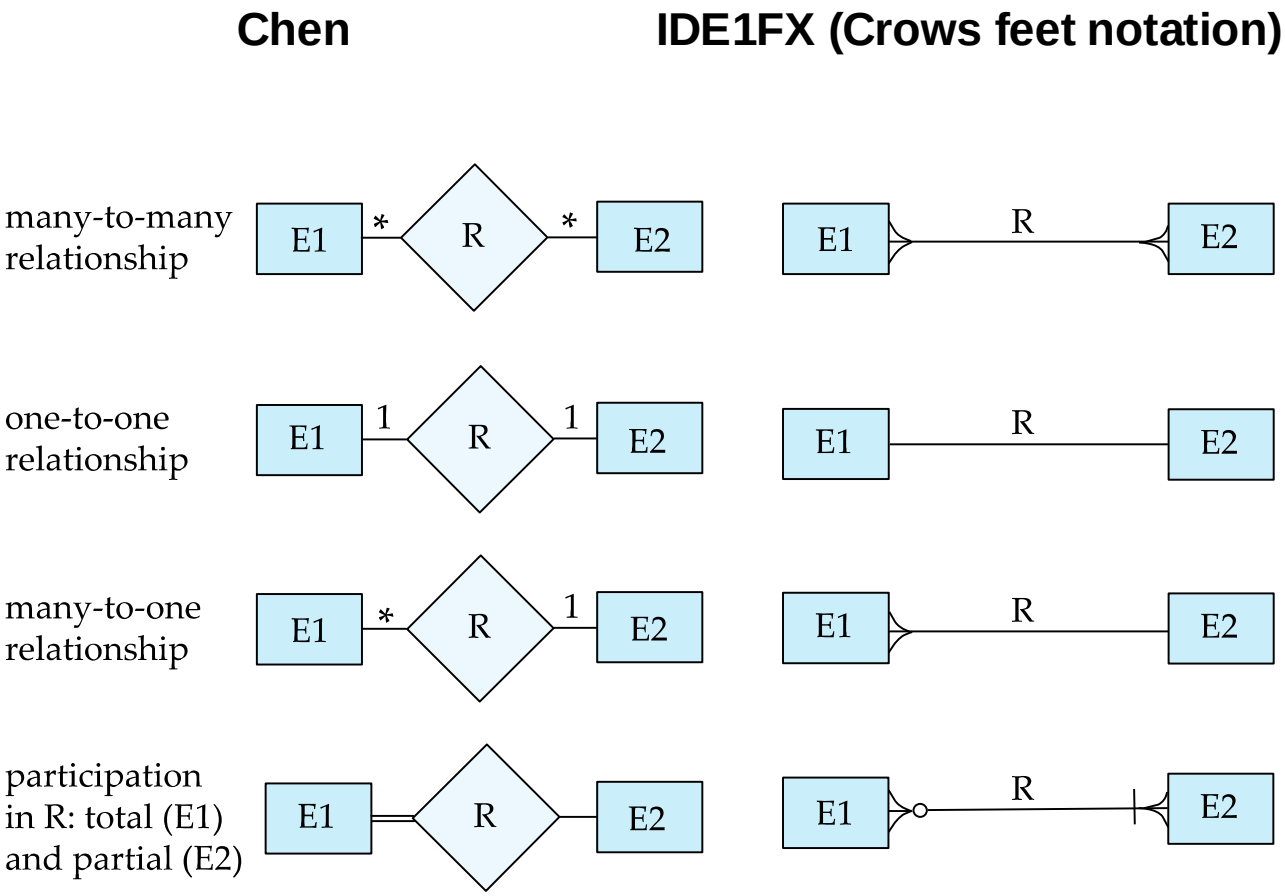


weak entity set 

generalization 

total generalization 

Alternative ER Notations



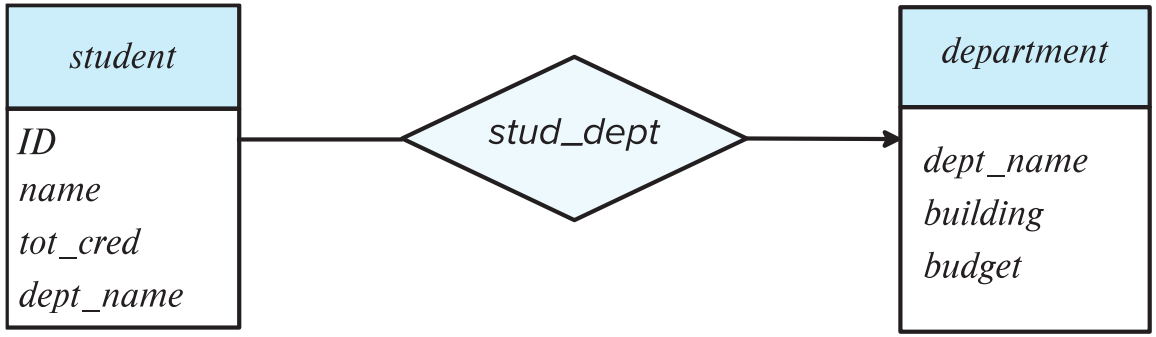
Other Aspects of Database Design

- Functional Requirements
- Data Flow, Workflow
- Schema Evolution

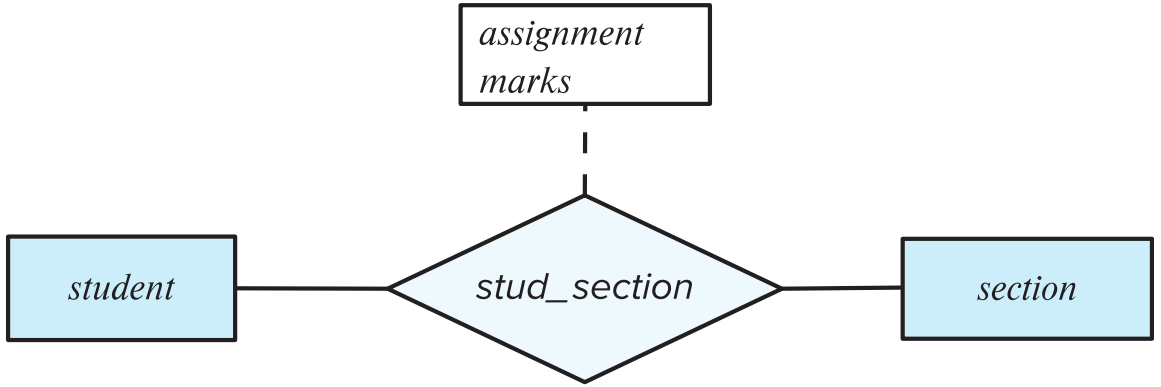
Design Issues

Common Mistakes in E-R Diagrams

- Example of erroneous E-R diagrams

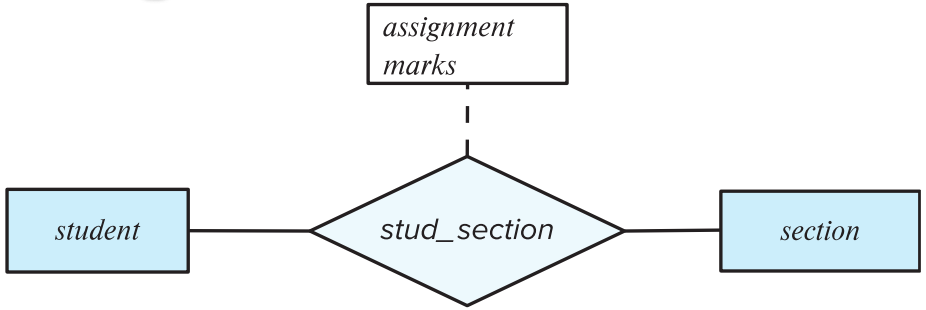


(a) Incorrect use of attribute

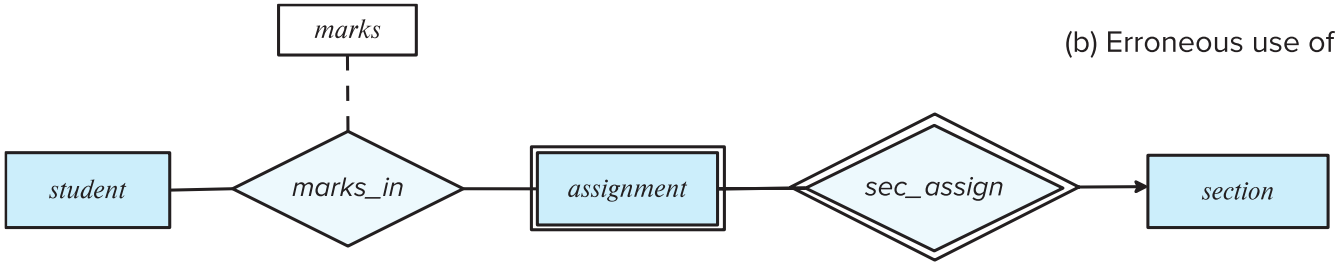


(b) Erroneous use of relationship attributes

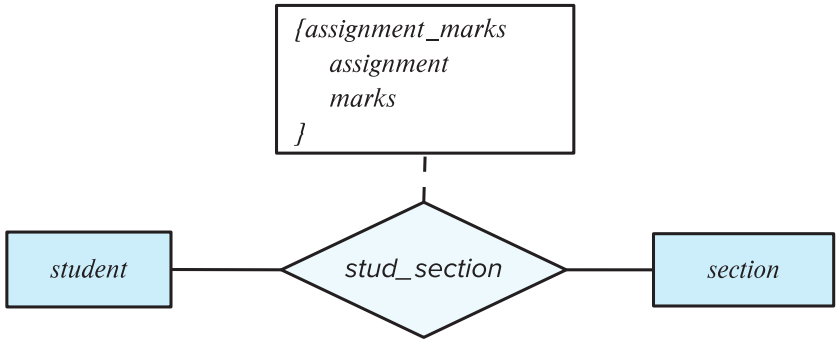
Common Mistakes in E-R Diagrams



(b) Erroneous use of relationship attributes



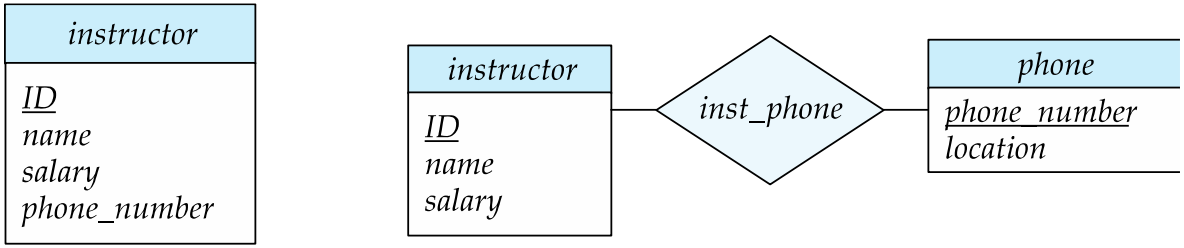
(c) Correct alternative to erroneous E-R diagram (b)



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

Entities vs. Attributes

- Use of entity sets vs. attributes

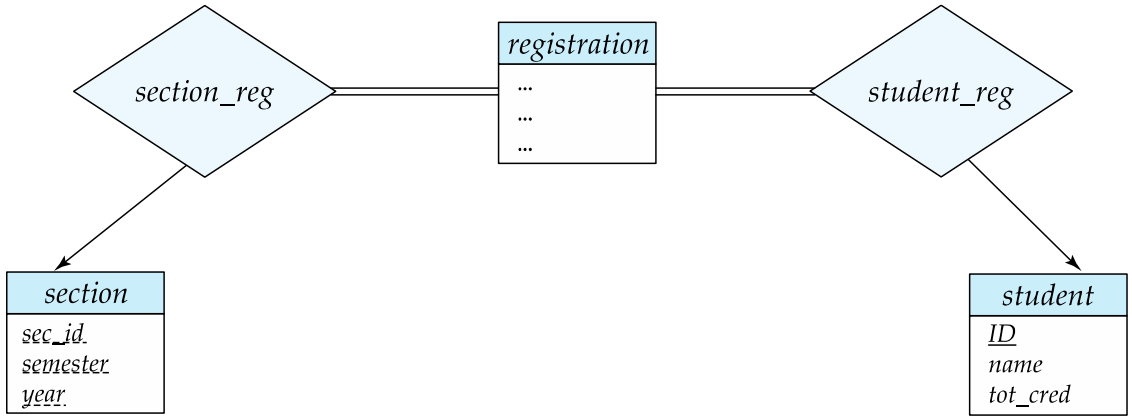


- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

Entities vs. Relationship sets

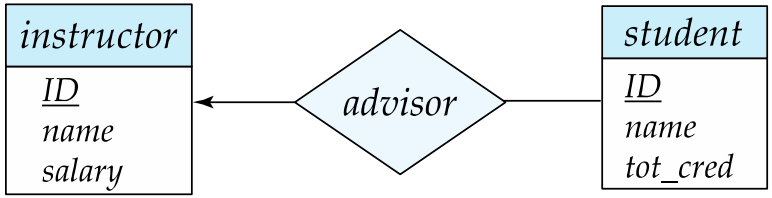
- **Use of entity sets vs. relationship sets**

  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*

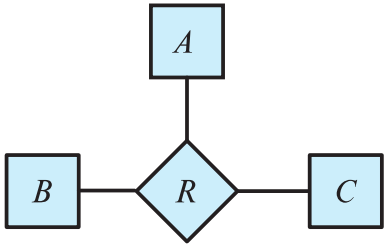
Converting Non-Binary Relationships to Binary Form

- 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:

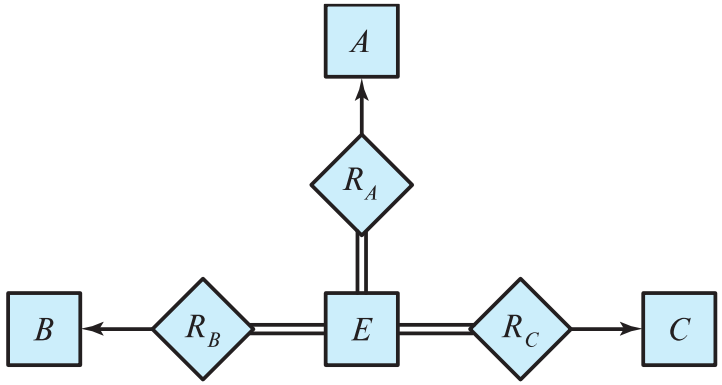
    1. R_A, relating *E* and *A*
    2. R_B, relating *E* and *B*
    3. R_C, relating *E* and *C*

  - 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 (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.

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

SCIENCE

FIN

Any questions?

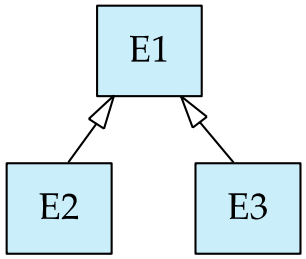
Extended E-R Features

Specialization

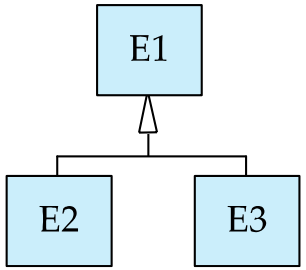
- Top-down design process; we designate sub-groupings within an entity set that are distinctive from other entities in the set.
- These sub-groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* 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

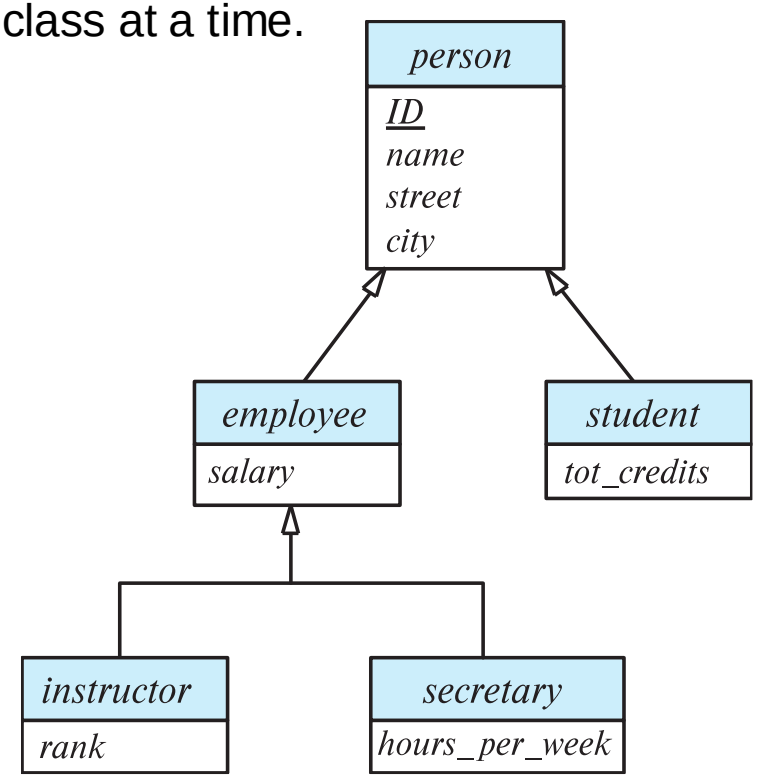
- **Overlapping** – *employee* and *student*
  - An entity can belong to multiple subclasses simultaneously.
  - Represented by allowing the same entity to be in multiple categories.
- **Disjoint** – *instructor* and *secretary*
  - An entity can belong to only one subclass at a time.
- Total and partial



ISA: generalization or specialization



disjoint generalization



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 as Schemas (Cont.)

- 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

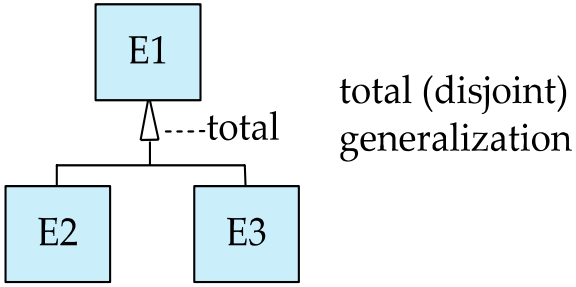
  - Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees

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 constraint

- **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

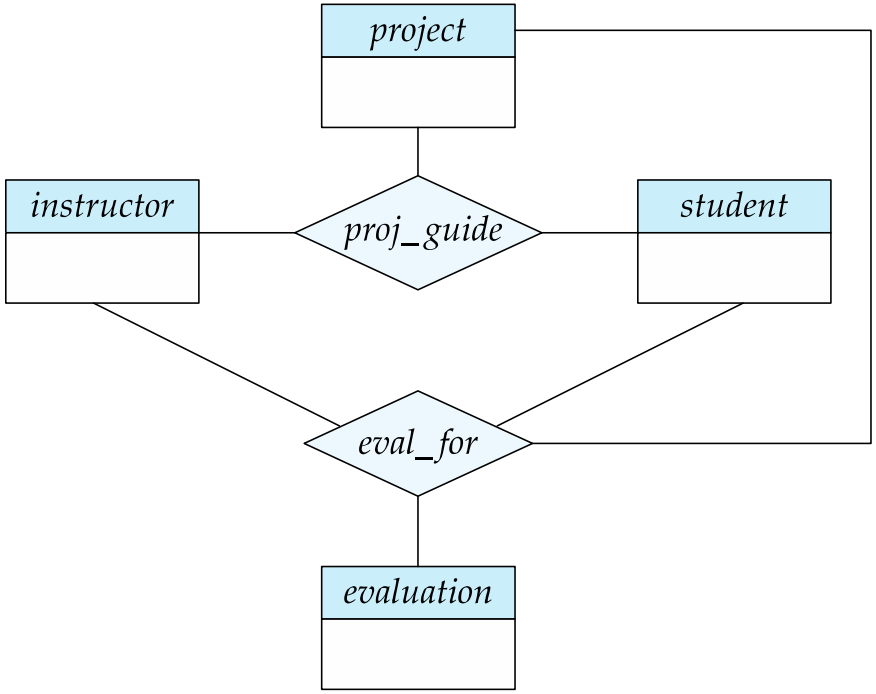


Completeness constraint (Cont.)

- 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 (for a total generalization), or to the set of hollow arrow-heads to which it applies (for an overlapping generalization).
- 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

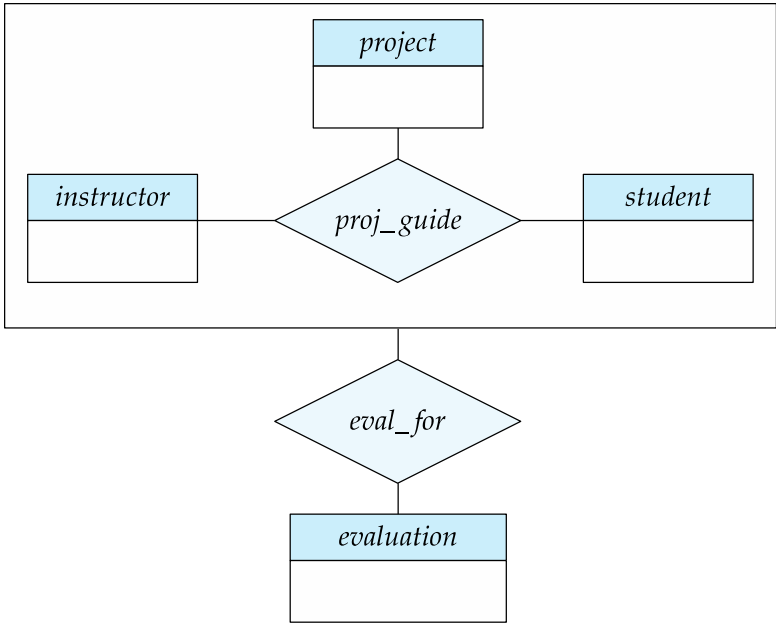


Aggregation (Cont.)

- 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 (Cont.)

- 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



Reduction to Relational Schemas

- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship,
  - The primary key of the associated entity set
  - Any descriptive attributes
- In our example:
  - The schema *eval_for* is:

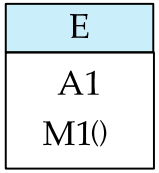
    eval_for (*s_ID, project_id, i_ID, evaluation_id*)
  - The schema *proj_guide* is redundant.

UML

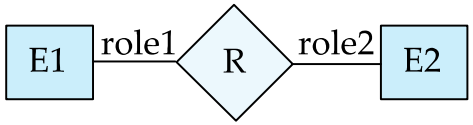
- **UML**: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- UML Class Diagrams correspond to E-R Diagram, but several differences.

ER vs. UML Class Diagrams

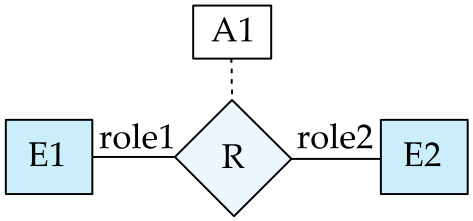
ER Diagram Notation



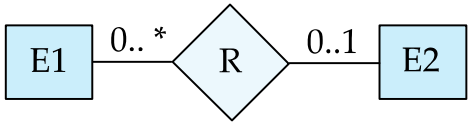
entity with attributes (simple, composite, multivalued, derived)



binary relationship

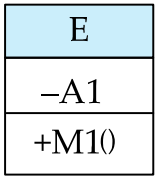


relationship attributes

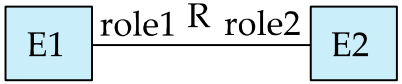


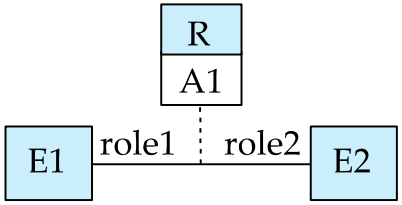
cardinality constraints

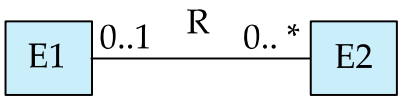
Equivalent in UML



class with simple attributes and methods (attribute prefixes: + = public, – = private, # = protected)

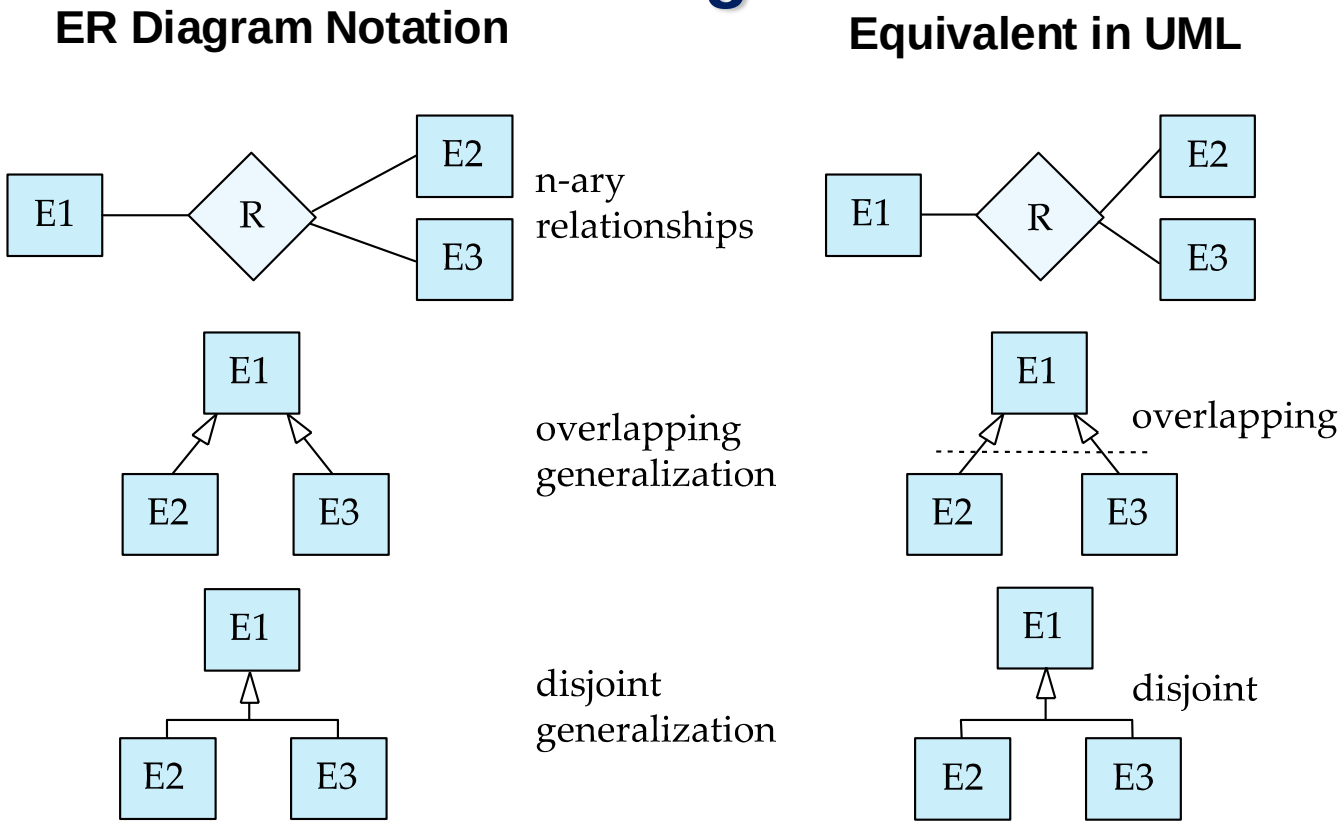






* Note reversal of position in cardinality constraint depiction

ER vs. UML Class Diagrams

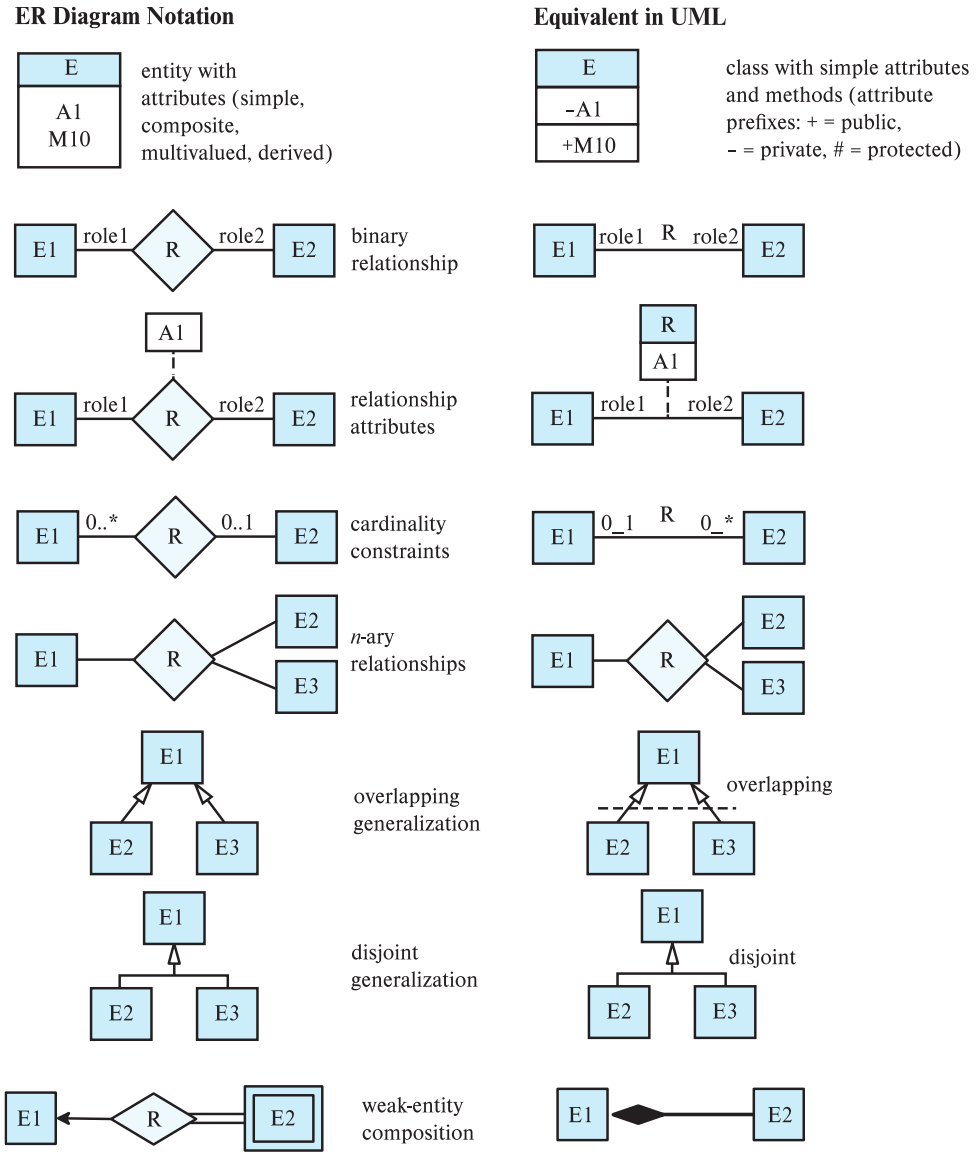


* Generalization can use merged or separate arrows independent of disjoint/overlapping

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



Database Design

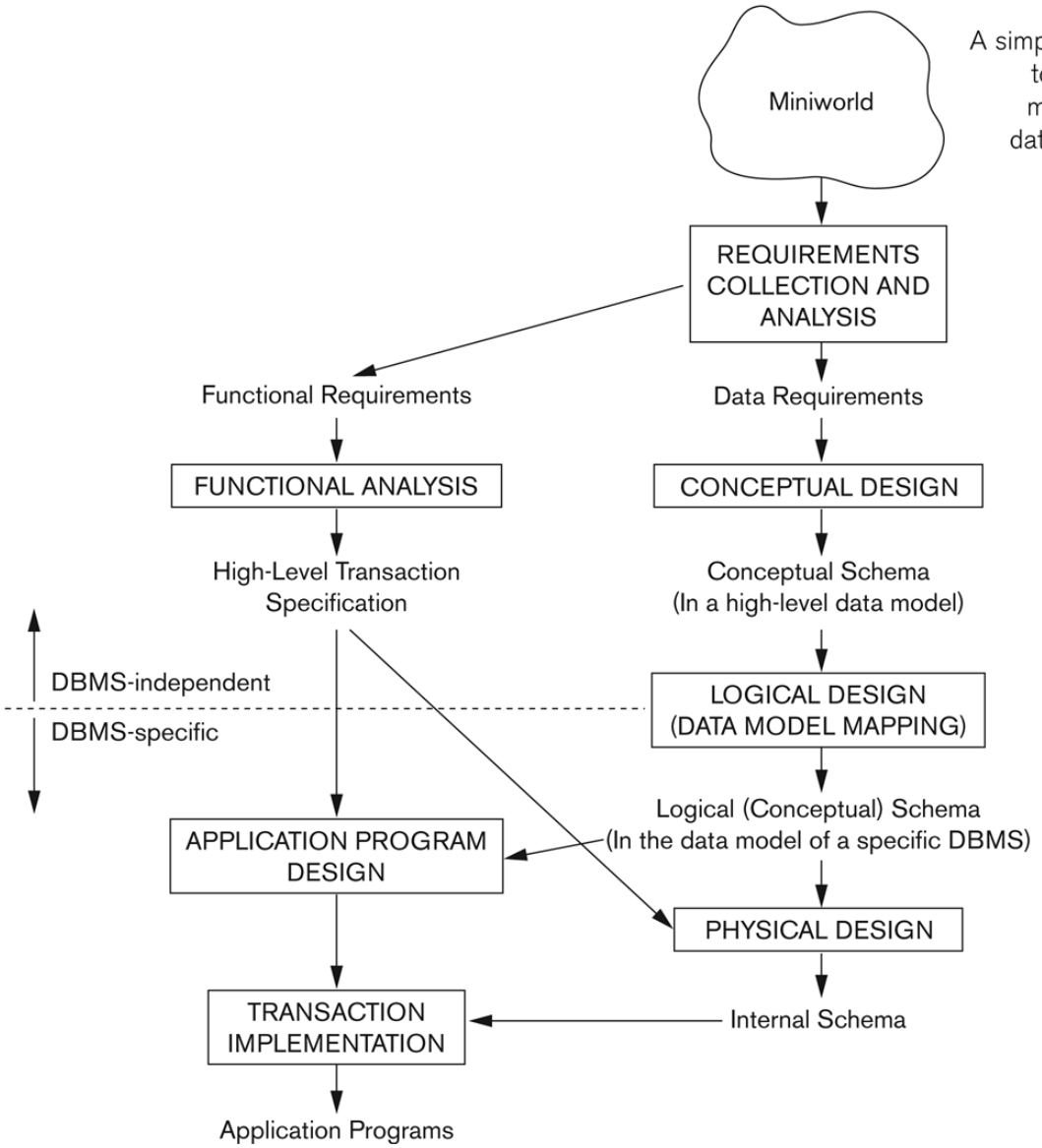
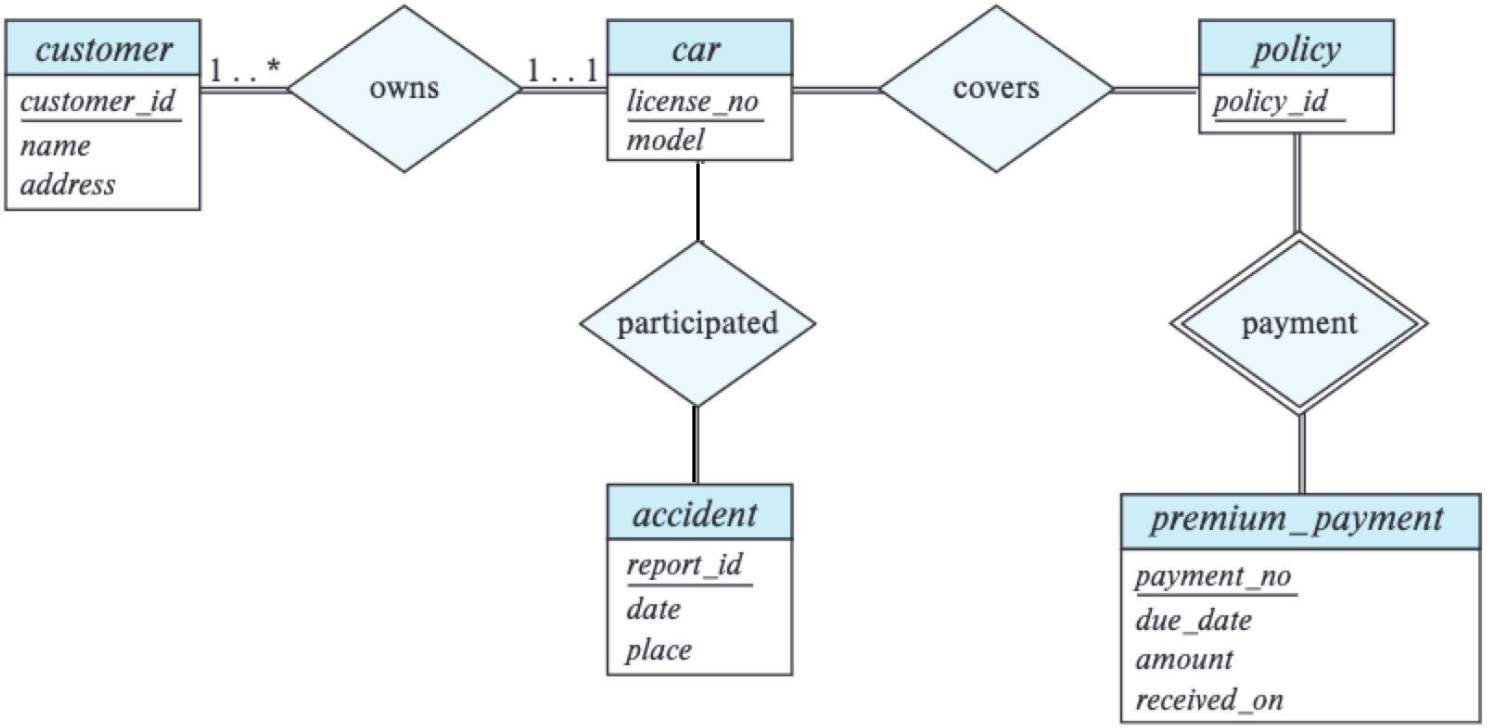


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Answer to ER Diagram Challenge



Enterprise Talk

- Weaviate
- ClickHouse

- **Book free download with three chapters**
  - ***Storage and Retrieval,***
  - ***Replication,***
  - ***and The Trouble with Distributed Systems***