Week 4: Relational Query Languages and Database Design

L4.1: *Formal Relational Query Languages - Part 1*

Relational Algebra

- It's a procedural query language
- Six basic operators:
  - Selection: σ
  - Projection: Π
  - Union: $\cup$
  - Difference: $-$
  - Intersection: $\cap$
  - Cartesian product: $\times$
  - Rename: ρ

Selection σ

- Notation: $\sigma_p(r)$
- p is called the selection predicate (a boolean expression - condition)
- r is a relation

$$\sigma_p(r) = \{t \mid t \in r \text{ and } p(t)\}$$

- where p is a formula in propositional calculus consisting of terms connected by $\wedge$ (`AND`) , $\vee$ (`OR`), and $\neg$ (`NOT`)

Projection Π

- Notation: $\Pi_{A_1, A_2, ..., A_n}(r)$
- $A_1, A_2, \ldots, A_n$ are attributes of r
- The result is defined as the relation of k columns obtained by erasing the columns that are not listed.
- Duplicate rows removed from result, sicne relations are sets.

Union $\cup$

- Notation: $r \cup s$

$$r \cup s = \{t \mid t \in r \vee t \in s\}$$

- For union:
  - r and s must have the same number of attributes.
  - The attributes must be of the same type. (eg: 3rd col of r must be of the same type as 3rd col of s)

Difference $-$

- Notation: $r - s$

$$r - s = \{t \mid t \in r \wedge t \notin s\}$$

- For difference:
  - r and s must have the same number of attributes.
  - The attributes must be of the same type. (eg: 3rd col of r must be of the same type as 3rd col of s)

Intersection $\cap$

- Notation: $r \cap s$

$$r \cap s = \{t \mid t \in r \wedge t \in s\}$$

- For intersection:
  - r and s must have the same number of attributes.
  - The attributes must be of the same type.
- **Note**: $r \cap s = r - (r - s)$

Cartesian Product $\times$

- Notation: $r \times s$

$$r \times s = \{t, q \mid t \in r \wedge q \in s\}$$

- Assume that attributes of r and s are disjoint ($r \cap s = \phi$)
- If attributes are not disjoint, we can rename them to make them disjoint.
- If r has n attributes and s has m attributes, then $r \times s$ has $n + m$ attributes.

Rename ρ

- Allows us to name the attributes of a relation.
- Notation: $\rho_X A_1, A_2, ..., A_n(E)$
- renames the attributes of E to $A_1, A_2, ..., A_n$ and the resulting relation is called X.

L4.2: *Formal Relational Query Languages - Part 2*

Predicate Logic

- **Predicate Logic** or **Predicate Calculus** is an extension of **Propositional Logic** or **Boolean Algebra**.
- **Tuple Relational Calculus** and **Domain Relational Calculus** are two variants of **Predicate Calculus**.
- It is a formal system for reasing and expressing statements in terms of predicates, quantifiers, and variables.
- It extends propositional logic by introducing quantifiers and variables, allowing for more precise and complex statements to be made.

Predicate

- A predicate is a statement that may be true or false depending on the values of its variables.

Example:

- $P(x)$: "x is a prime number"
- Here, x is the variable.
- "is a prime number" is the predicate.
- The predicate P can be considered as a function. It tells the truth value of the statement for a given value of x.

In general, a statement involving n variables $x_1, x_2, \ldots, x_n$ can be denoted by $P(x_1, x_2, x_3, \ldots, x_n)$ and is called an n-ary predicate or an n-place predicate.

Quantifiers

- Quantifiers are used to express the amount of truth in a statement.
- There are two types of quantifiers:
  - Universal Quantifier: $\forall$
  - Existential Quantifier: $\exists$
- **Universal Quantifier**: $\forall$
  - $\forall x P(x)$ is read as "For all x, $P(x)$ is true."
- **Existential Quantifier**: $\exists$
  - $\exists x P(x)$ is read as "There exists an x such that $P(x)$ is true."

Tuple Relational Calculus

- It is a non-procedural query language, where each query is of the form:

$$\{t \mid P(t)\}$$

- t are resulting tuples,
- $P(t)$ is a formula in predicate logic.

It can also use quantifiers:

- $\exists t \in r(Q(t))$ is "there exists a tuple t in r such that $Q(t)$ is true."

- $\forall t \in r(Q(t))$ is "for all tuples t in r, $Q(t)$ is true."

Example:

$$\{P \mid \exists S \in Students \wedge (S.CGPA > 8 \wedge P.name = S.name \wedge P.age = S.age)\}$$

- returns the name and age of students with CGPA greater than 8.

Domain Relational Calculus

- It is a non-procedural query language, where each query is of the form:

$$\{< t_1, t_2, \ldots, t_n > \mid P(t_1, t_2, \ldots, t_n)\}$$

- $t_1, t_2, \ldots, t_n$ represent domain variables.
- $P(t_1, t_2, \ldots, t_n)$ is a formula in predicate logic.

Equivalence of RA, TRC, and DRC

Operation	Relational Algebra	Tuple Calculus	Domain Calculus
Select	$\sigma_p(r)$	$\{t \mid t \in r \wedge p(t)\}$	$\{< a > \mid < a > \in r \wedge p(a)\}$
Project	$\Pi x(r)$	$\{t[x] \mid t \in r\}$	$\{< x > \mid < x > \in r[x]\}$
Union	$r \cup s$	$\{t \mid t \in r \vee t \in s\}$	$\{< a > \mid < a > \in r \vee < a > \in s\}$
Intersection	$r \cap s$	$\{t \mid t \in r \wedge t \in s\}$	$\{< a > \mid < a > \in r \wedge < a > \in s\}$
Difference	$r - s$	$\{t \mid t \in r \wedge t \notin s\}$	$\{< a > \mid < a > \in r \wedge < a > \notin s\}$
Natural join	$r \bowtie s$	$\{t \mid t[x] = s[y] \wedge t \in r \wedge s \in s\}$	$\{< x, y > \mid < x, y > \in r[x]$ $\wedge < x, y > \in s[y]\}$

L4.3: *Entity-Relationship Model - Part 1*

Design Process

A design:

- satisfies a given functional specification.
- conforms to limitations of the target medium.
- Meets implicit or explicit requirements on performance and resource usage.
- Satisfies implicit or explicit design criteria on the ofrm of the artifact.
- Satisfies restrictions on the design process itself, such as budget or time constraints or the tools available.

Role of Abstraction

- Abstraction is the process of removing irrelevant details from a problem.
- Example: A binary number is hard to remember. But if we convert it to decimal or hexadecimal, it becomes easier to remember.

Model Building

- A model is an abstraction of a real-world object or system.
- Each model describes some aspect of the object or system.

Design Approach

- **Requirement Analysis** - Analyse the data needs of the prospective database users.
    - Planning
    - System Definition
- **Database Designing** - Use a modeling framework to create abstraction of the real world.
    - **Logical model** - Deciding on a good database schema, business decisions, computer science decisions etc...
    - **Physical model** - Deciding on the physical laout of the database, storage, indexing etc...
- **Implementation**
    - Data conversion and loading
    - Testing and evaluation

Entity-Relationship Model

The Entity-Relationship (ER) model is a conceptual data model used to represent the structure and relationships between entities (objects, concepts, or things) within a domain. It provides a graphical representation of the database schema and serves as a foundation for designing and understanding relational databases. Here are key points about the ER model:

1. **Entities**
    - Entities are the fundamental building blocks of the ER model and represent real-world objects or concepts.
    - They are depicted as rectangles in the ER diagram and have attributes that describe their properties.
    - An **entity set** is a set of entities of same type that share the same properties. Example: set of all persons, companies, trees, holidays, etc.
2. **Attributes**

- Attributes are characteristics or properties of an entity.
- They provide details and information about the entity and are depicted as ovals connected to the respective entity.
- Attributes can be simple (e.g., name, age) or composite (composed of multiple sub-attributes) and can have a data type.
- Attribute types:
  - **Simple** (Single-valued)
    - An attribute that cannot be divided into smaller parts and is not composed of other attributes.
    - For example: "Name" or "Age" can be simple attributes.
  - **Composite**
    - An aatirbute that is composed of multiple sub-attributes.
    - For example: An address attribute can have sub-attributes like street, city, state, and postal code.
  - **Derived**
    - An aatribute whose value can be calculated or derived from other attributes.
    - For example: "Total Price" can be derived from "Price" and "Quantity".
  - **Multi-valued**
    - An attribute that can hold multiple values for a given instance of an entity.
    - For example: "Email" attribute in a "Person" entity can have multiple email addresses.

3. **Relationships**
   - Relationships represent associations or connections between entities.
   - They describe how entities interact or relate to each other.
   - Relationships can be one-to-one, one-to-many, or many-to-many, and they are depicted as diamonds connecting the related entities.

An entity set may be of two types:

Strong Entity Sets

- An entity set that contains sufficient attributes to uniquely identify all its entities.
- In other words, *a primary key exists for a strong entity set*.
- Example: The "Person" entity set can be uniquely identified by its "ID" attribute.

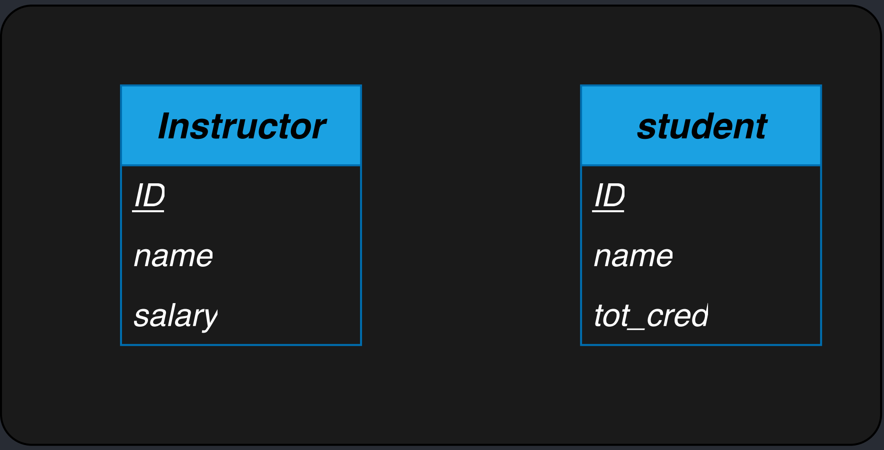
Weak Entity Sets

- An entity set that does not contain sufficient attributes to uniquely identify all its entities.
- In other words, *a primary key does not exist for a weak entity set*.
- However, it contains a partial key called as a **discriminator**.
- Discriminator can identify a group of entities from the entity set.
- Since, it doesn't have a primary key, it cannot independently exist in the ER model.
- It features in the model in relationship with another strong entity set. This is called the **Identifying relationship**.

L4.4: *Entity-Relationship Model - Part 2*

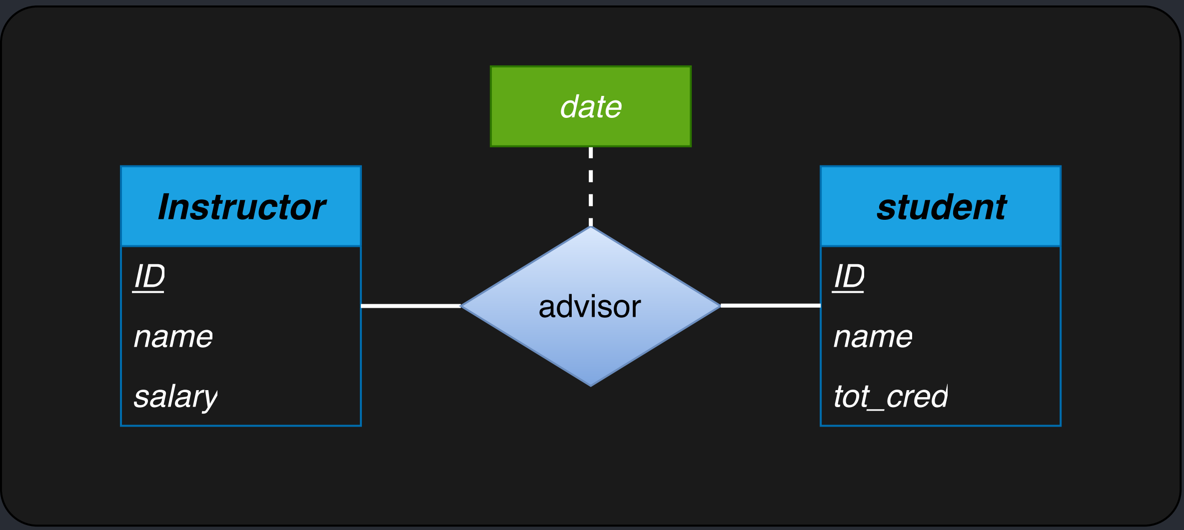
Entity Sets

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



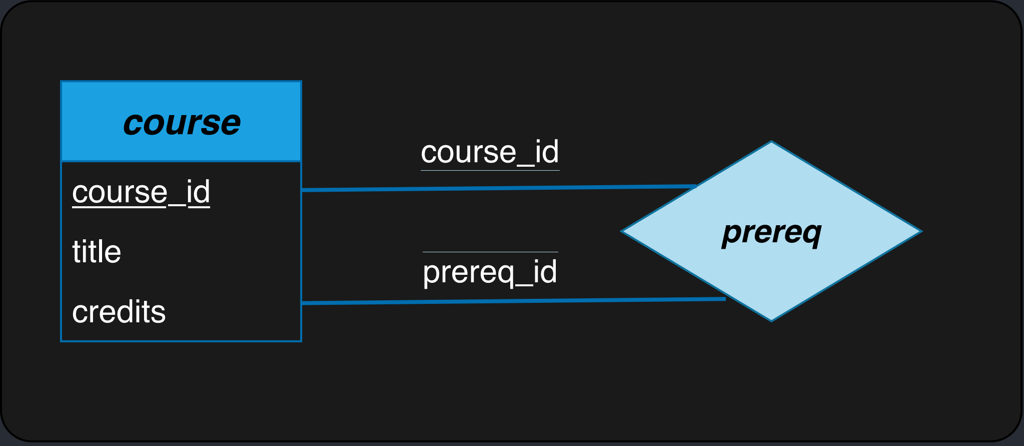
Diamond represent relationship sets.

- Relationships can be represented graphically as follows:
  - Diamonds represent relationship sets.
  - Lines link attributes to entity sets and entity sets to relationship sets.
  - Dashed lines indicate attributes of the key for the relationship set.



Roles

- A role represents the specific part or function played by an entity in a relationship.
- Entities can have different roles in different relationships, and the same entity can play different roles in the same relationship.
- Roles provide additional context and meaning to the relationship by indicating the responsibilities or behaviors of the entities involved.

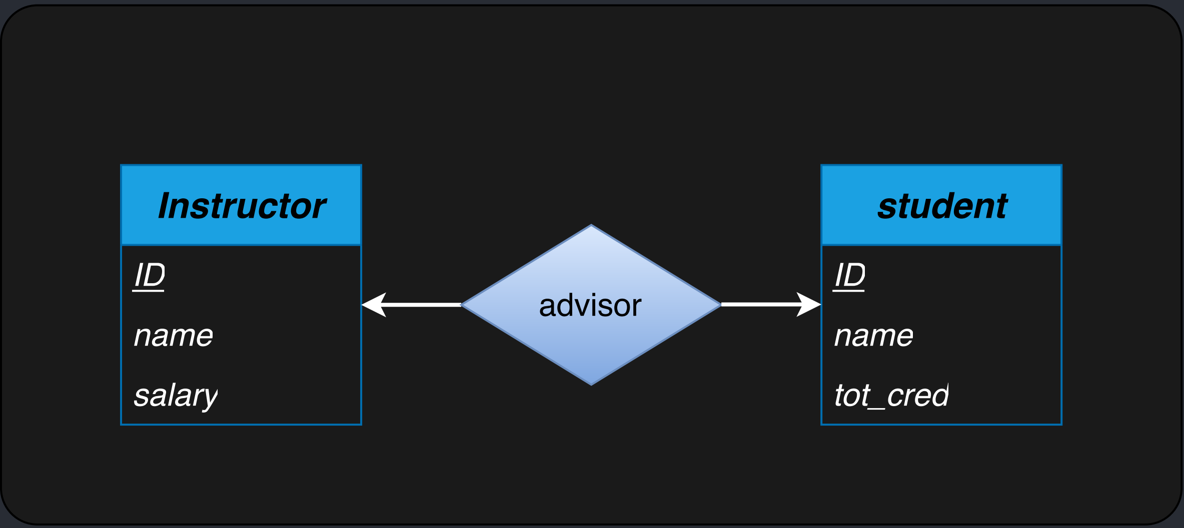


- Here, `prereq` is the role, which `course_id` and `prereq_id` play in the relationship `course`.

Cardinality Constraints

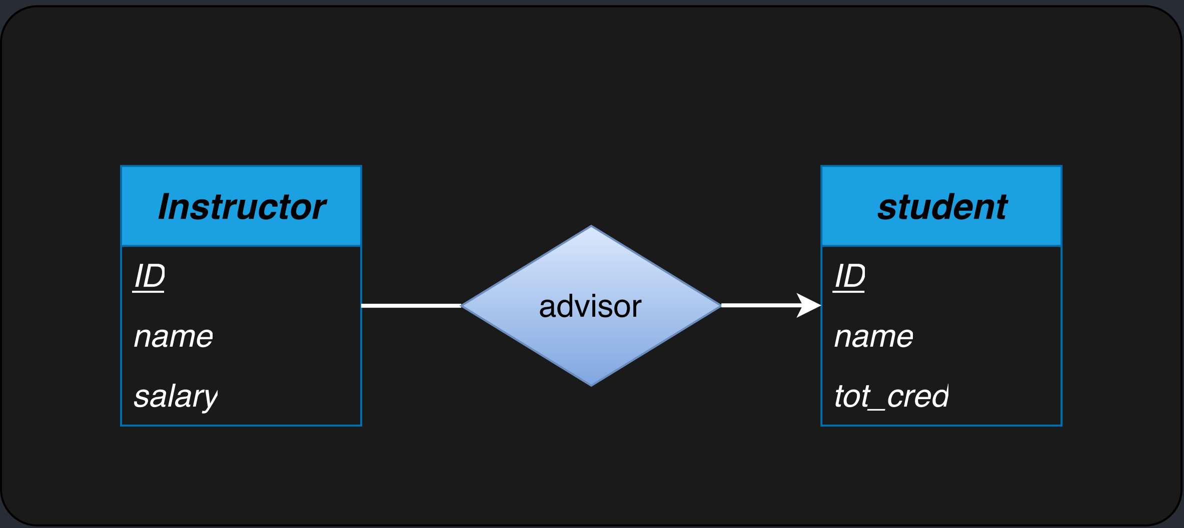
One-to-One [$\leftarrow$ x $\rightarrow$]

- A student is associated with at most one instructor via the relationship advisor.
- An instructor is associated with at most one student via the relationship advisor.



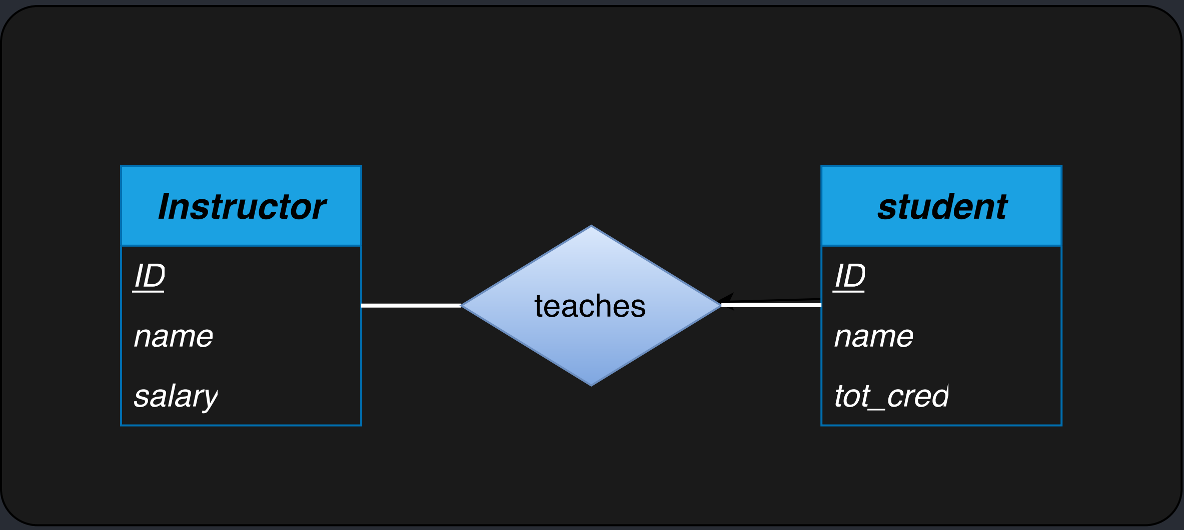
Many-to-One [$\rightarrow$]

- An instructor is associated with at most student via the relationship advisor.
- A student is associated with several (including 0) instructors via the relationship advisor.



Many-to-Many [——]

- An instructor is associated with several (including 0) students via the relationship teaches.
- A student is associated with several (possibly 0) instructors via the relationship teaches.

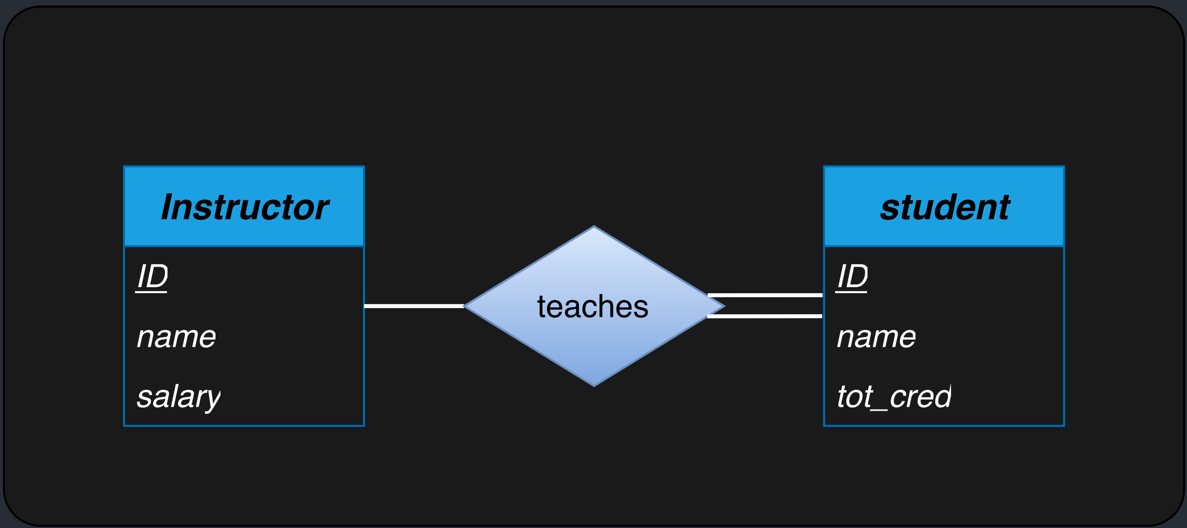


Total and Partial Participation

- **Total participation** (indicated by double line) means every entity in the entity set participates in at least one relationship in the relationship set.
- **Partial Participation** (indicated by single line) means some entities in the entity set may not participate in any relationship in the relationship set.

Example:

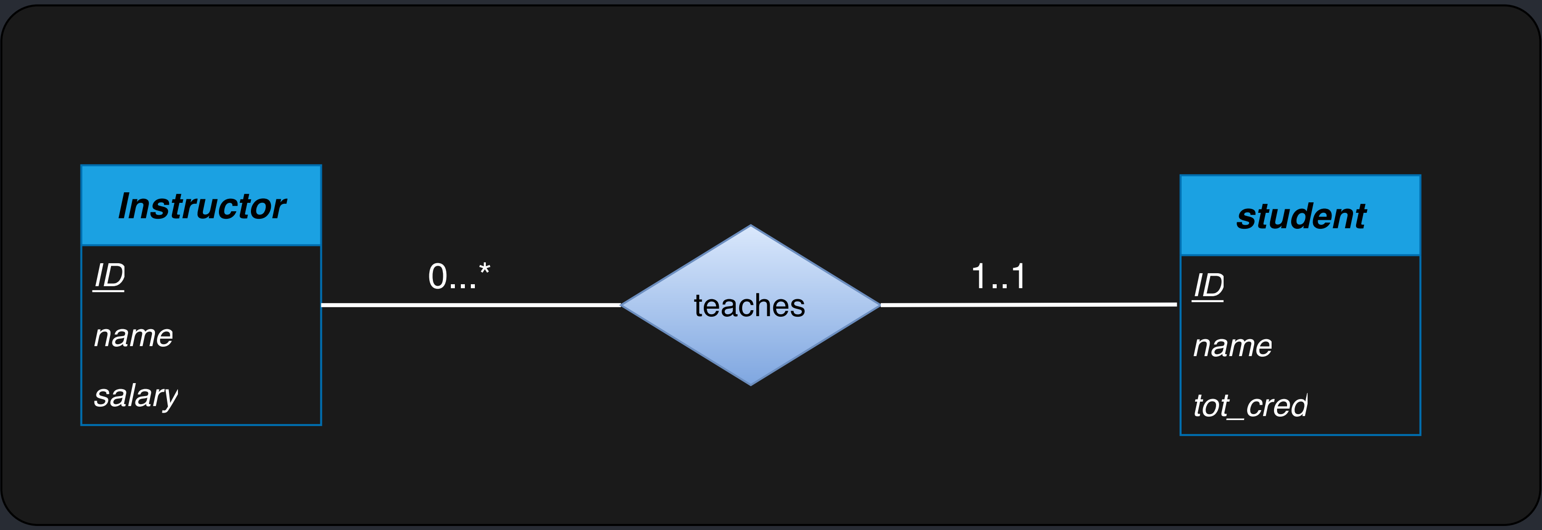
- Participation of student in teaches relation is total.
- Every student must be taught by at least one instructor.
- Participation of instructor in teaches is partial.



Notation for Expressing More complex constraints

- A line may have an associated minimum and maximum number of entities that must participate in a relationship.
- $I...h$ where I is the minimum and h is the maximum.
    - 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.



SYNTAX

```
(Simple attribute)
...
(Composite attribute)
    (Simple attribute)
    (Simple attribute)
    ...
{ Multi-valued attribute }
...
Derived attribute ()
...
```

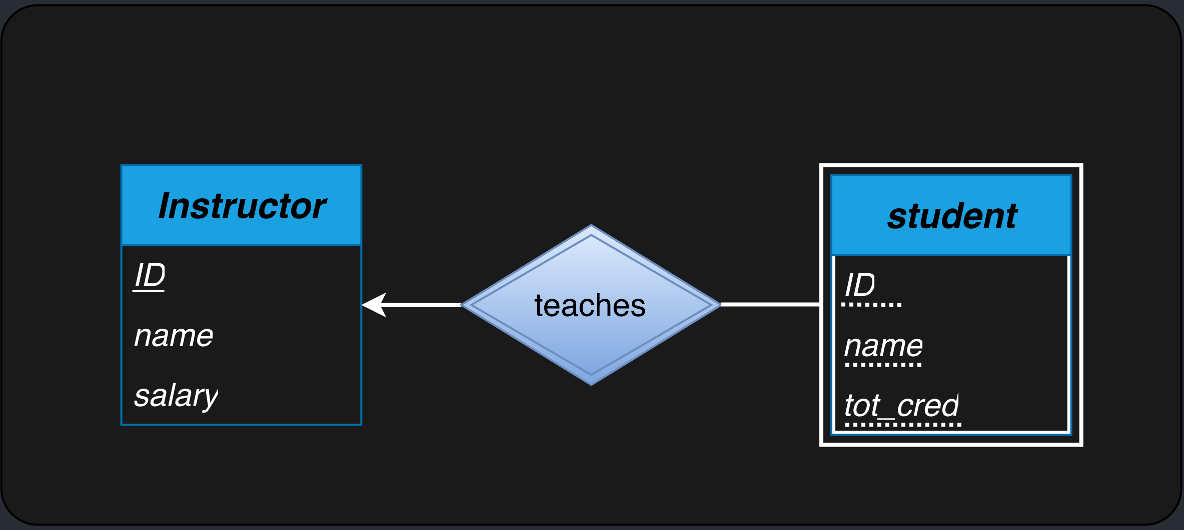
instructor

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

Expressing Weak Entity Sets

- In the ER diagram, a weak entity set is indicated by double rectangle.

- We underline the discriminator of a weak entity set with a dashed line.

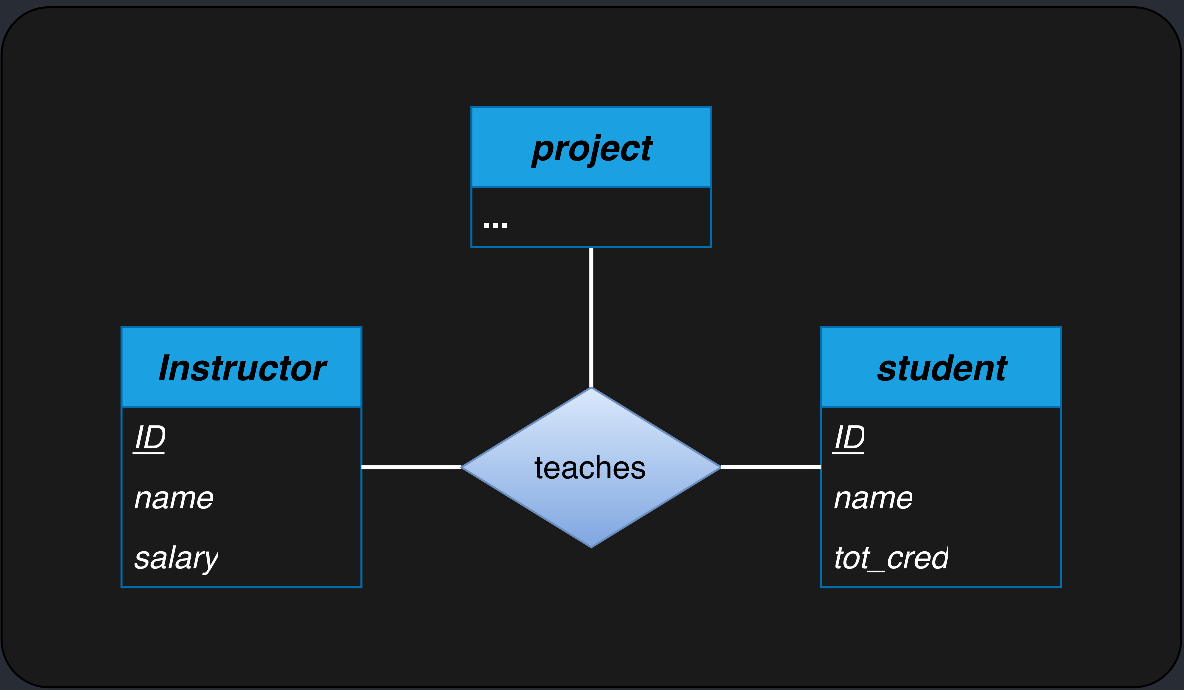


L4.5: *Entity-Relationship Model - Part 3*

ER features

Non-binary Relationship Sets

- Most relationships sets are binary.
- There are occasions when we need to represent a relationship set that involves more than two entity sets as non-binary relationship set.



Specialization: ISA

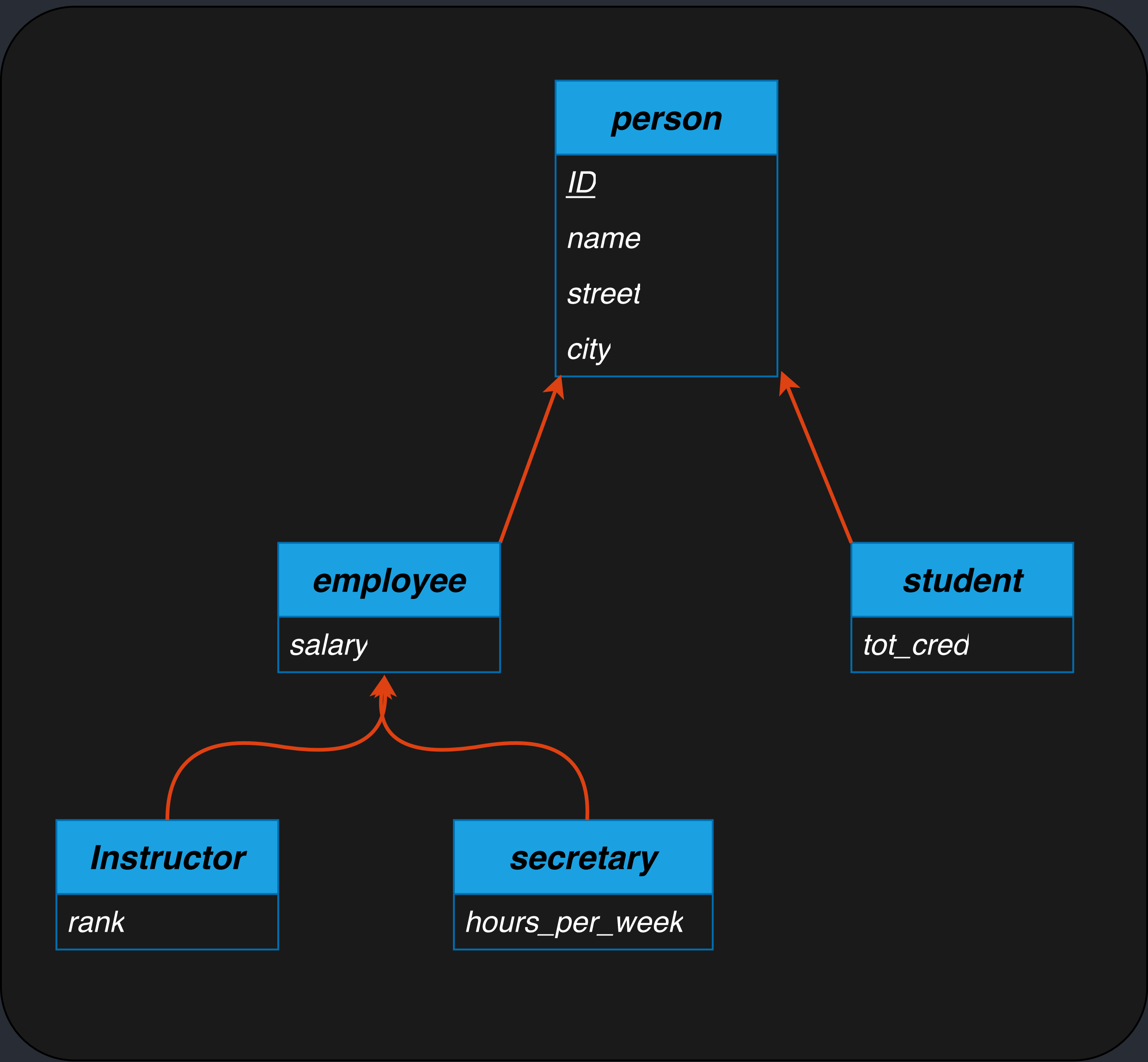
Top-down design process

- 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.

Attribute Inheritance

- A lower-level entity set inherits all the attribues and relationship participation of the higher-level entity set to which it is linked.



- **Overlapping**: employee and student
- **Disjoint**: instructor and secretary
- Total and Partial
- `instructor IS A employee`
- `secretary IS A employee`
- `student IS A person`
- `employee IS A person`

Representing specialization via Schema

- **Method 1**
  - Form a schema for the higher-level entity set.
  - Forma 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
instructor	ID, rank
secretary	ID, hours_per_week

- Drawback: Getting information about an employee requires accessing two relations.

- **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
instructor	ID, name, street, city, rank
secretary	ID, name, street, city, hours_per_week

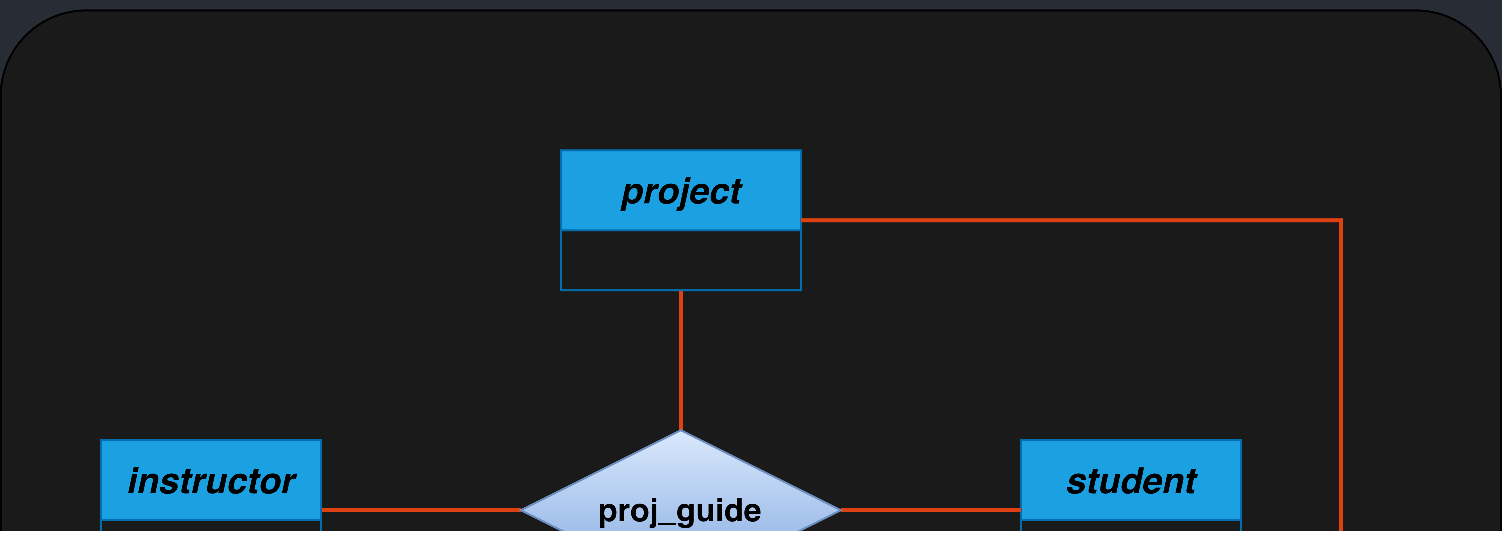
  - Drawback: Redundancy of attributes.

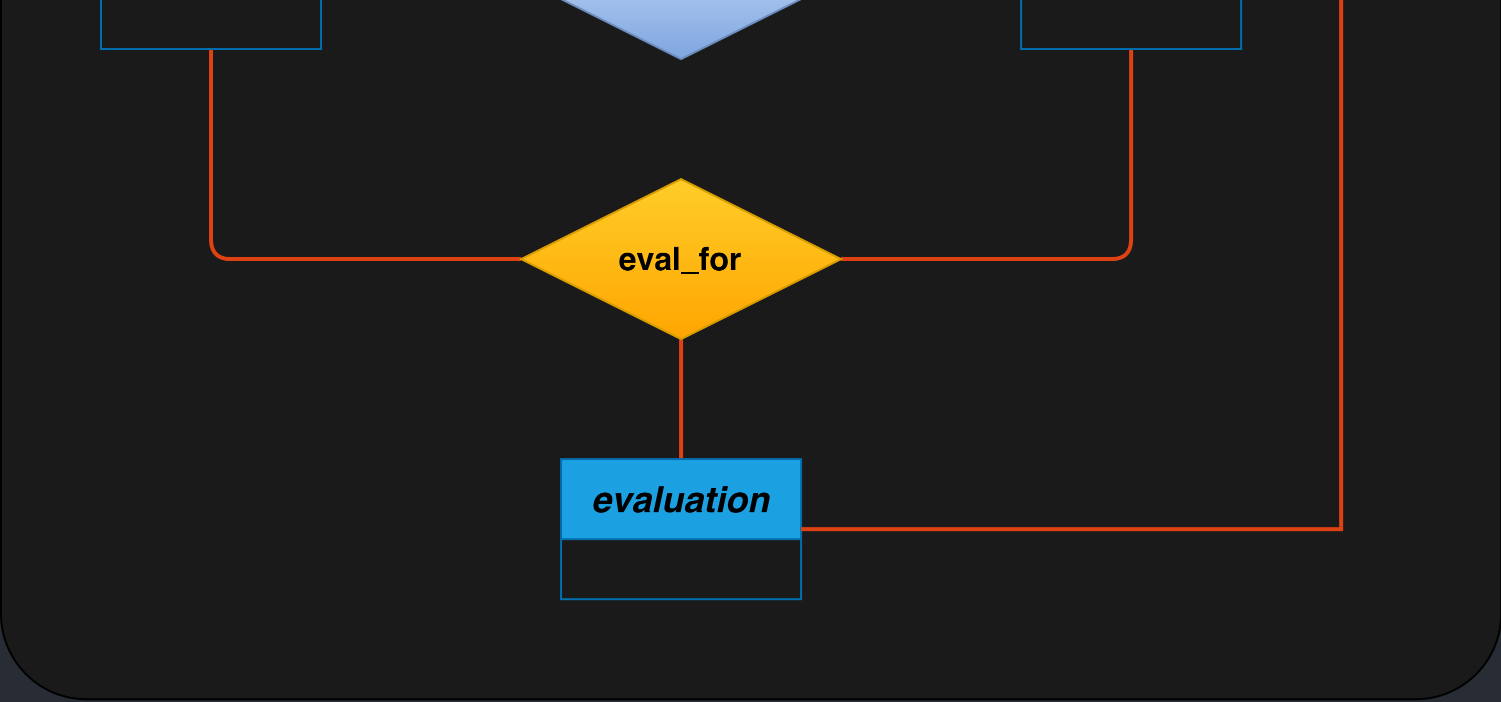
Bottom-up design process

- 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 the same way in an ER diagram.

Aggregation

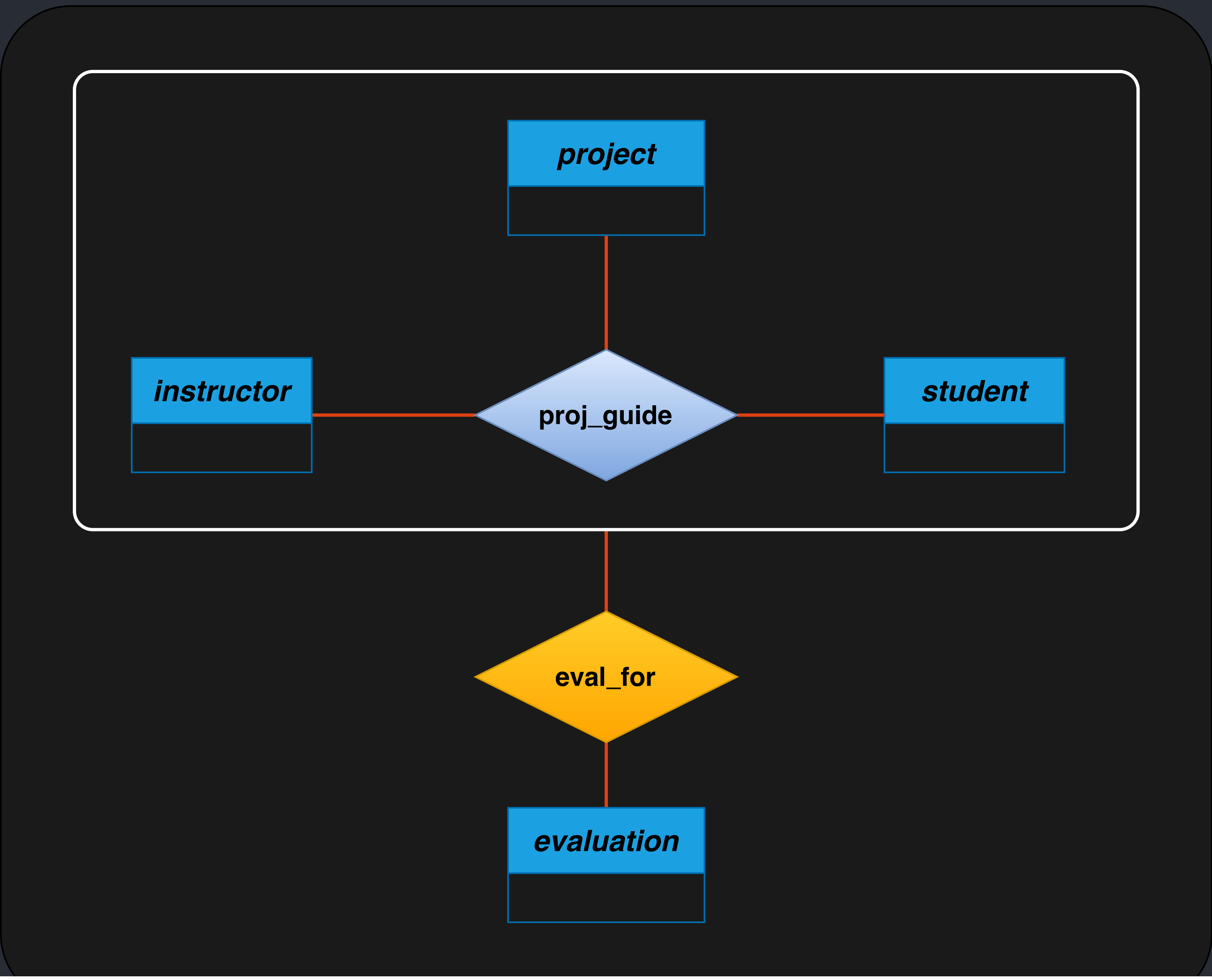
- Aggregation is a way of representing a relationship between two entities as a single entity.





- Relationship sets `eval_for` and `proj_guide` represents overlapping information.
  - Enever `eval_for` corresponds to `proj_guide` relationship.
  - However, some `proj_guide` relationships may not correspond to any `eval_for` relationships.

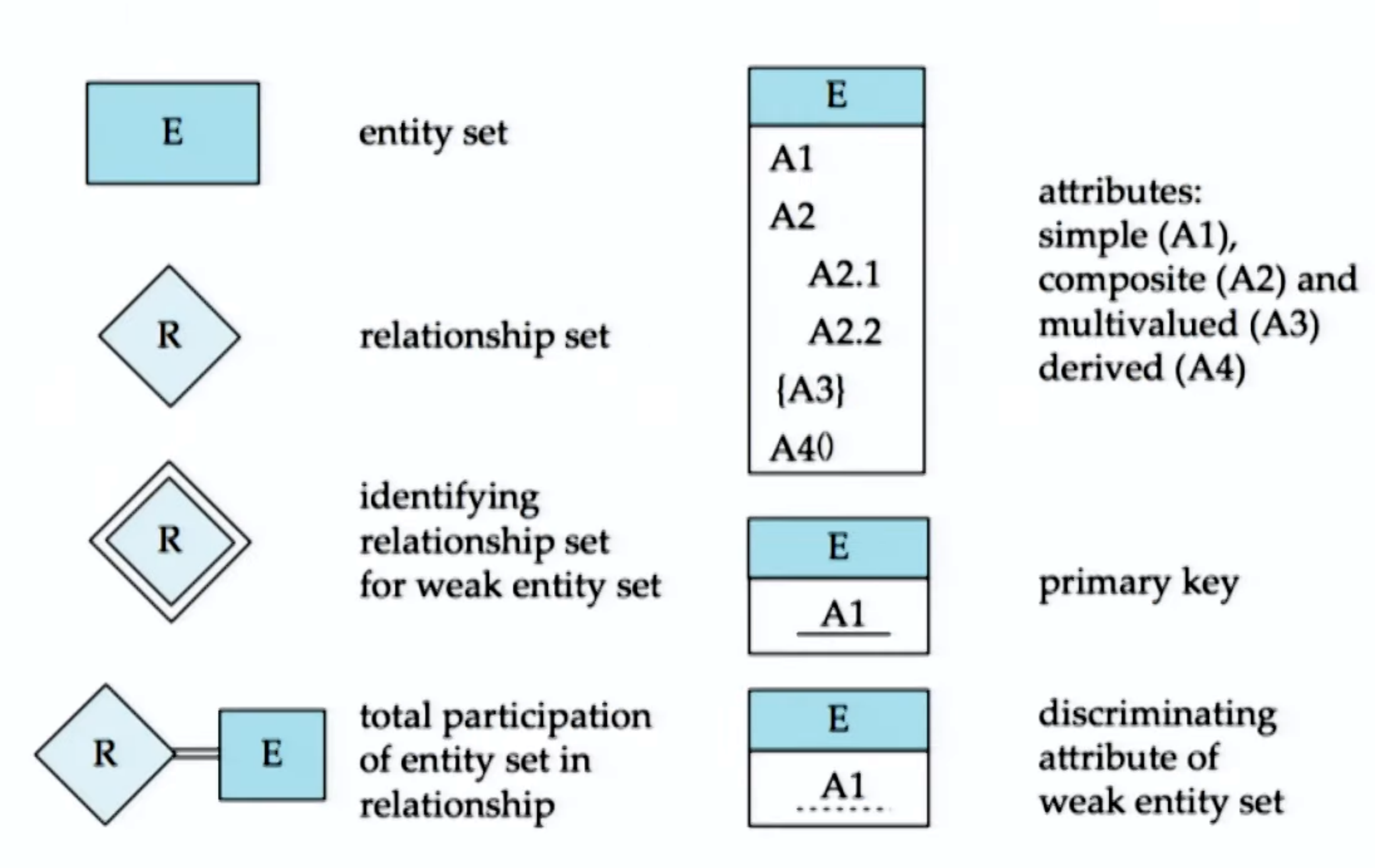
We can eliminate this redundancy via aggregation without introducing redundancy.

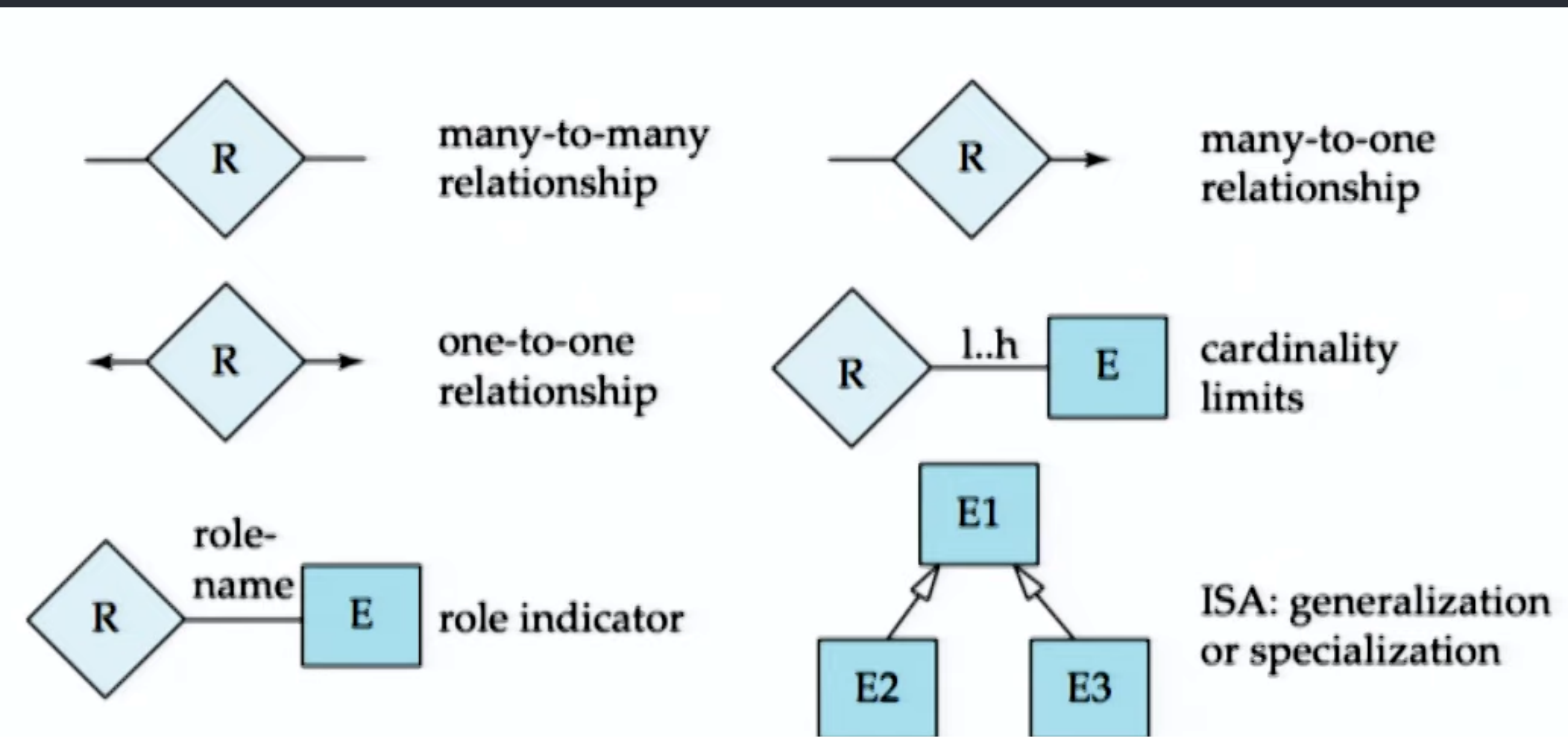


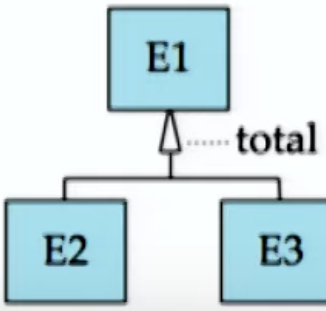
Now,

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

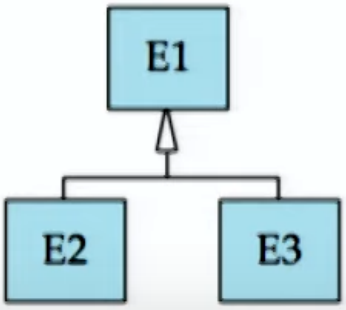
Symbols used in ER Notation





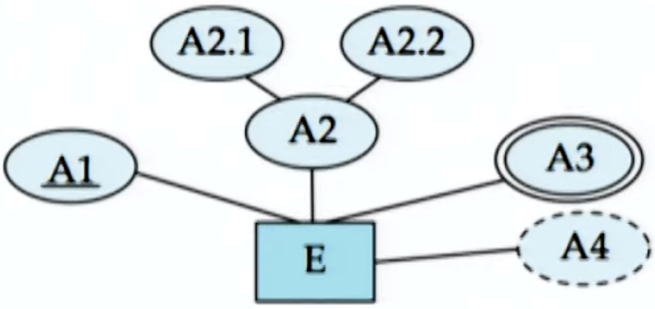


total (disjoint) generalization



disjoint generalization

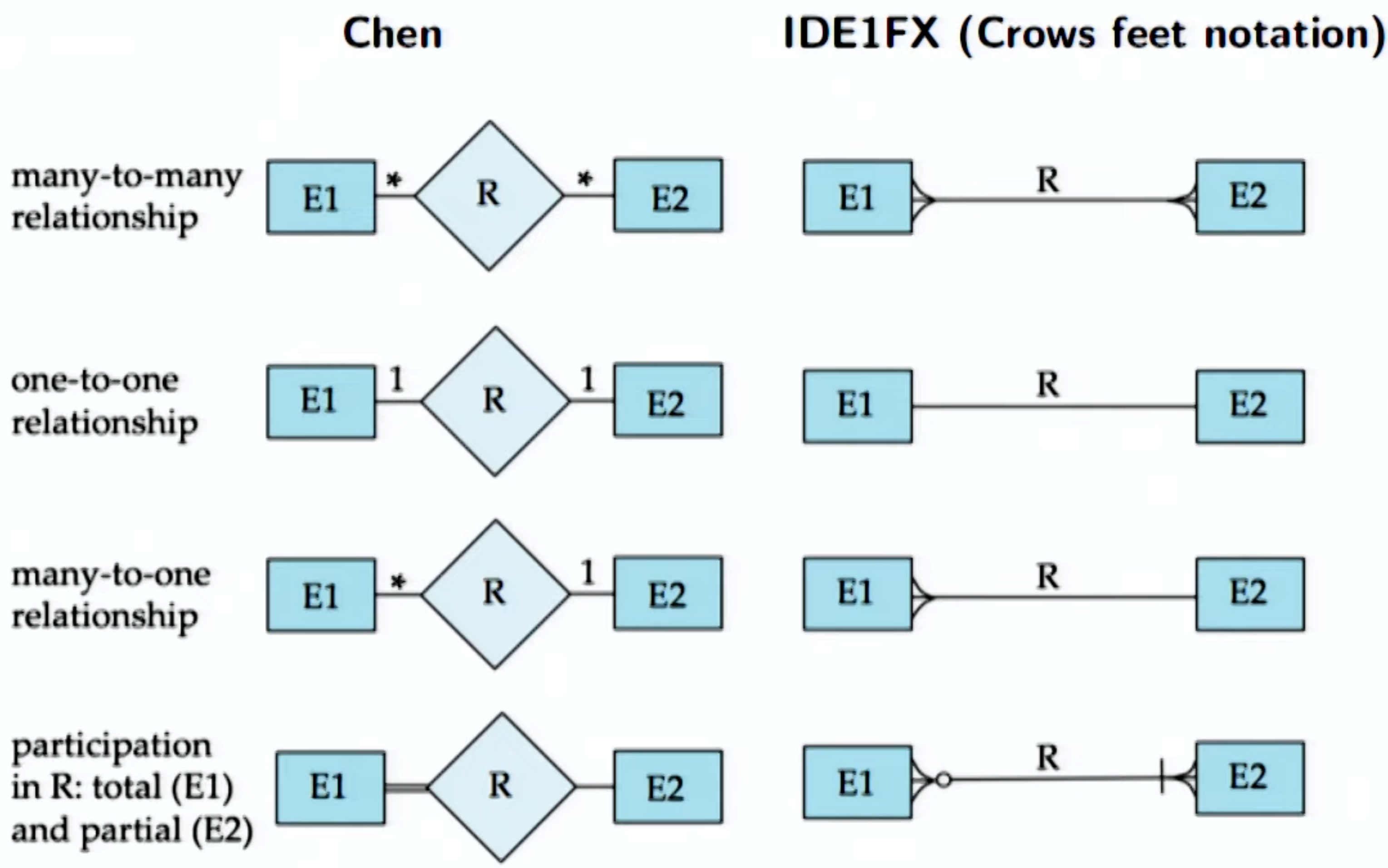
entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1



weak entity set 

generalization 

total generalization 



Tutorials