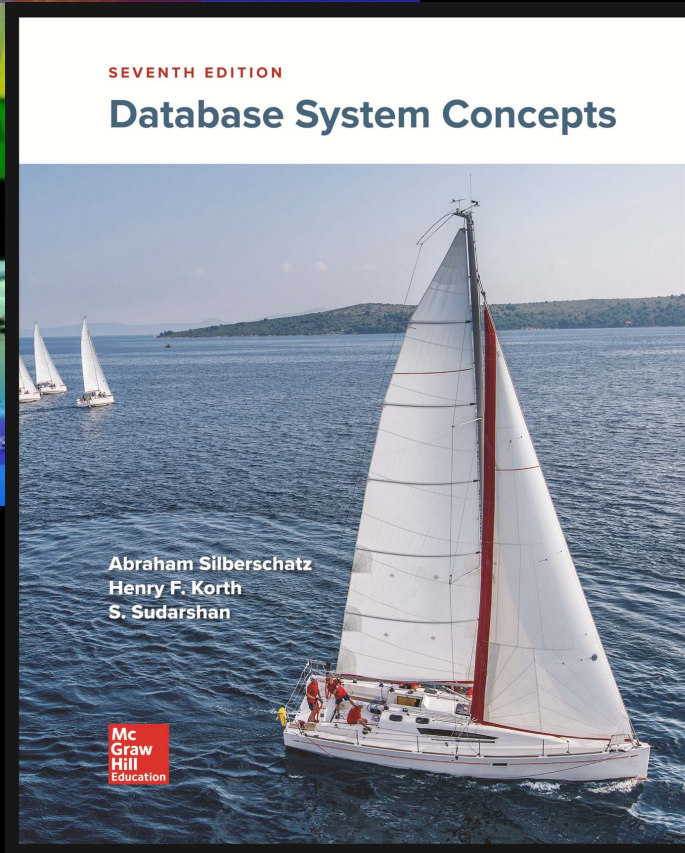
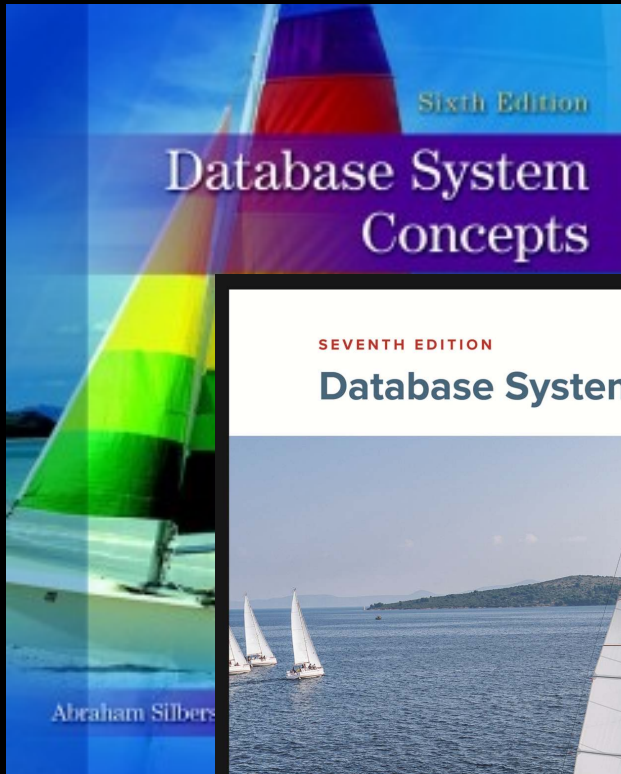




• IF2140 – Pemodelan Basis Data / IF2240 – Basis Data

Reducing E-R Diagrams to Relational Schemas



References

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

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

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

- Chapter 6: Database Design using E-R Model

Type of Mapping from ER to Relational Model

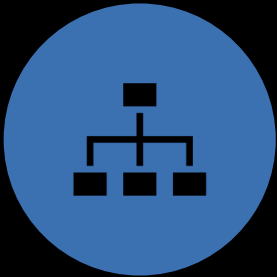
ER to Relational Mapping Algorithm:

- Mapping of Regular Entity Type
- Mapping of Weak Entity Type
- Mapping of Binary 1-1 Relationship
- Mapping of Binary 1-N Relationship
- Mapping of Binary M-N Relationship
- Mapping of Multivalued Attributes
- Mapping of N-ary Relationship

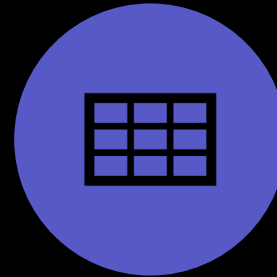
Mapping EER Model Constructs to Relations

- Options for mapping specialization or generalization
- Mapping of aggregation

Correspondence of Relational Model with E-R Model



Relations (tables) correspond with entity types and with many-to-many relationship types



Rows correspond with entity instances and with many-to-many relationship instances



Columns correspond with attributes

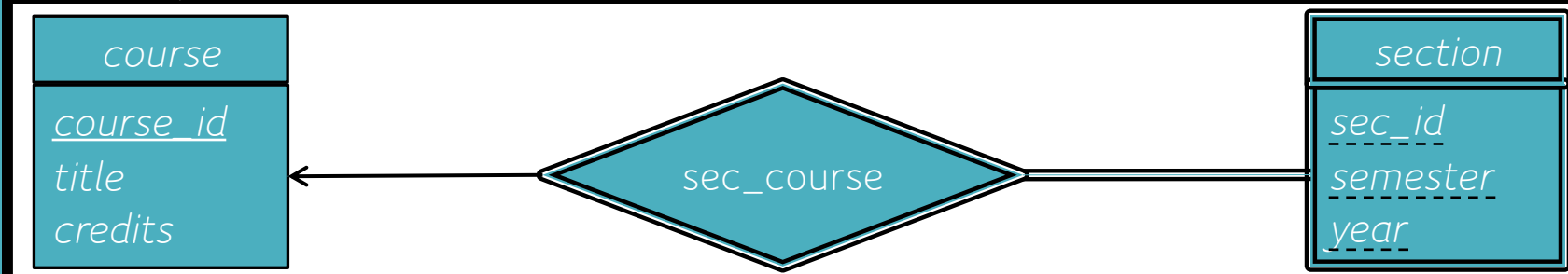


NOTE: The word *relation* (in relational database) is NOT the same as the word *relationship* (in E-R model)

Representing Entity Sets

- A **strong entity** set reduces to a schema with the same attributes
- A **weak entity** set becomes a relation schema that includes a column for the primary key of the identifying strong entity set

Example:



Strong entity:
course (*course_id*, title, credits)

Weak entity:
section (*course_id*, *sec_id*, *sem*, *year*)

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 and omitted derived attributes, extended *instructor* schema is

```
instructor(ID,  
  first_name, middle_initial, last_name,  
  street_number, street_name,  
  apt_number, city, state, zip,  
  date_of_birth)
```

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 ()



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:

$inst_phone (ID, \underline{phone_number})$

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)

instructor

ID

name

first_name

middle_initial

last_name

address

street

street_number

street_name

apt_number

city

state

zip

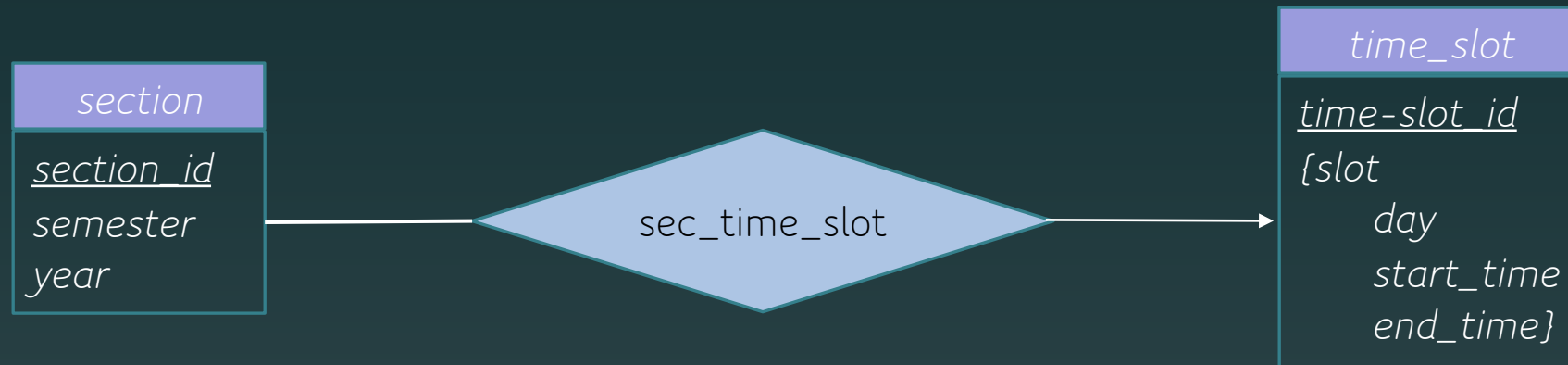
{ *phone_number* }

date_of_birth

age ()



Multivalued Attributes – Special Case



Entity *time_slot* has only one attribute other than the primary-key attribute, and that attribute is multivalued

- Optimization: No need to create the relation corresponding to the entity, just create the one corresponding to the multivalued attribute
 - $time_slot = (time_slot_id, day, start_time, end_time)$
- Caveat: *time_slot* attribute of *section* (from *sec_time_slot*) cannot be a foreign key due to this optimization. Why?

Derived Attributes

Derived attributes are omitted, they are not implemented in the relational schema

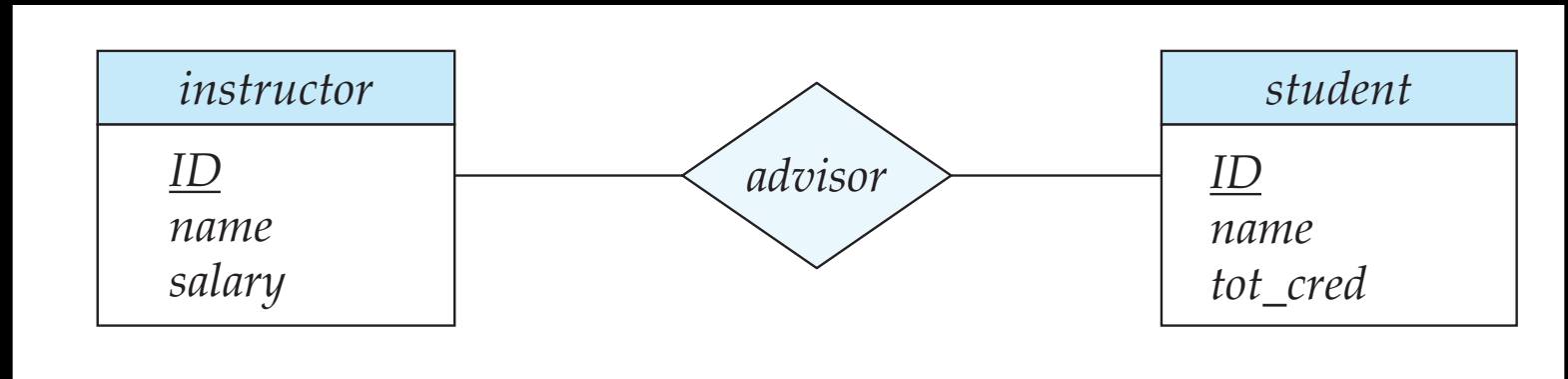
- Derived attributes are presented using *views*

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*

Example:



Many-to-many relationship set:
 $advisor = (\underline{s_id}, \underline{i_id})$

Redundancy of Schemas

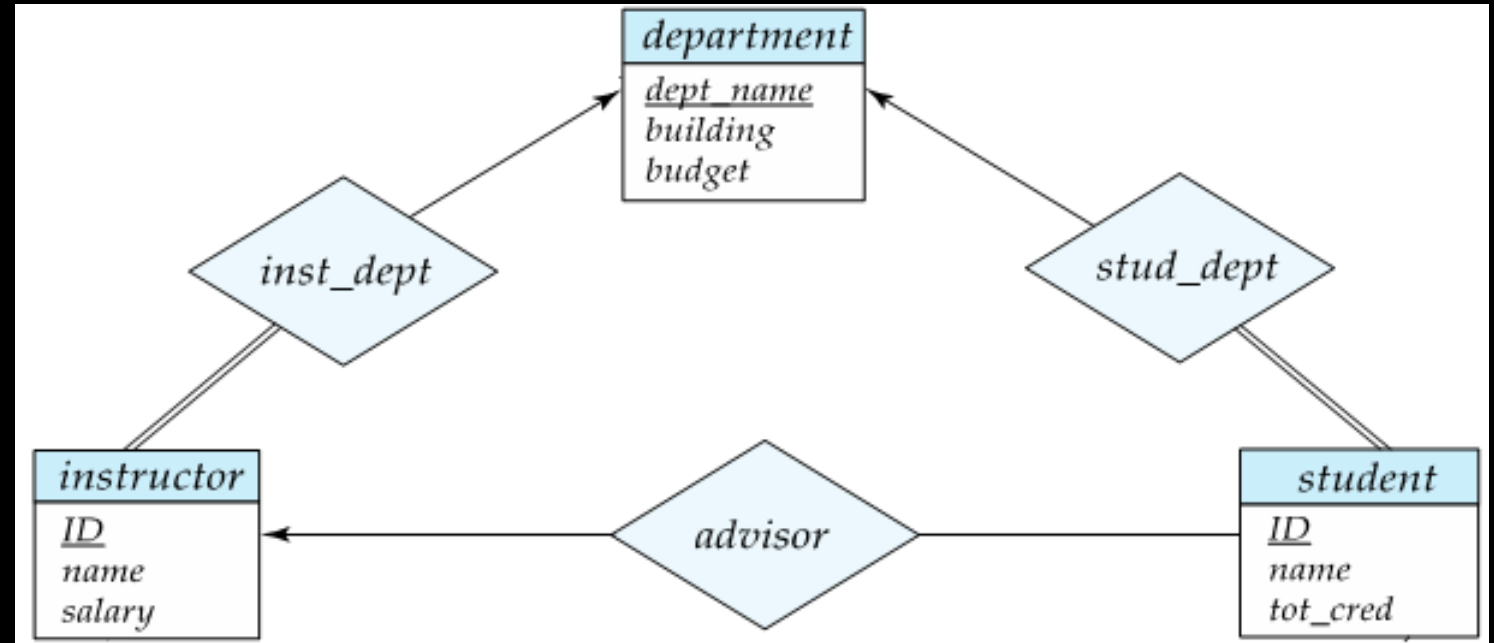
Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the “many” side, containing the primary key of the “one” side

For one-to-one relationship sets, either side can be chosen to act as the “many” side

- That is, 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 value

Example:

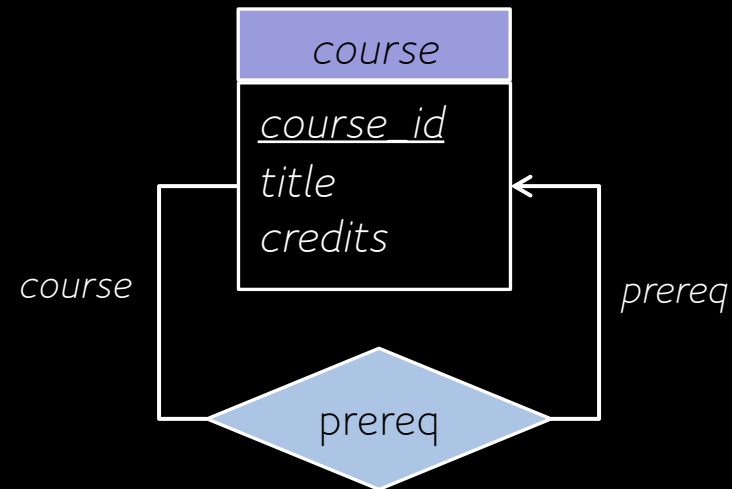


Example: Instead of creating a schema for relationship set *inst_dept*, add an attribute *dept_name* to the schema arising from entity set *instructor*

Instructor = (ID, name, salary, dept_name)

Representing Unary Relationship (1/2)

Example: One-to-Many: Recursive foreign key in the same relation



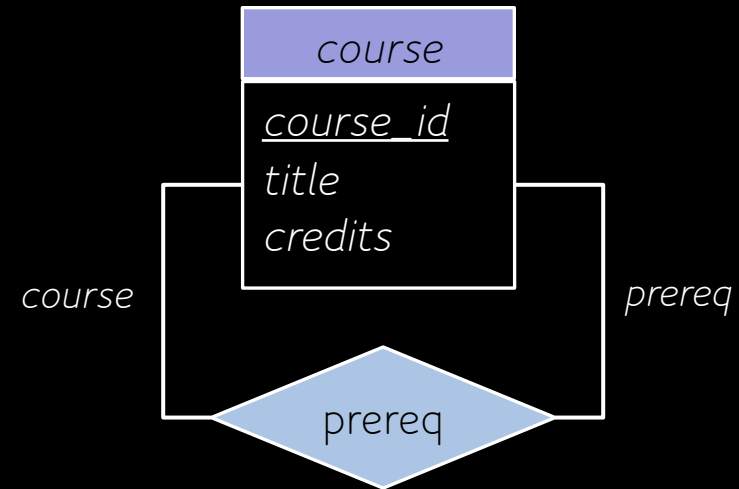
course = (*course_id*, *title*,
credits, *prereq_id*)

prereq_id is the recursive foreign
key that refers to primary key of
course (*course_id*)

Representing Unary Relationship (2/2)

Many-to-Many: Two relations

- One for the entity set
- One for the relationship in which the primary key has two attributes, both taken from the primary key of the entity set



course = (*course_id*, *title*, *credits*)

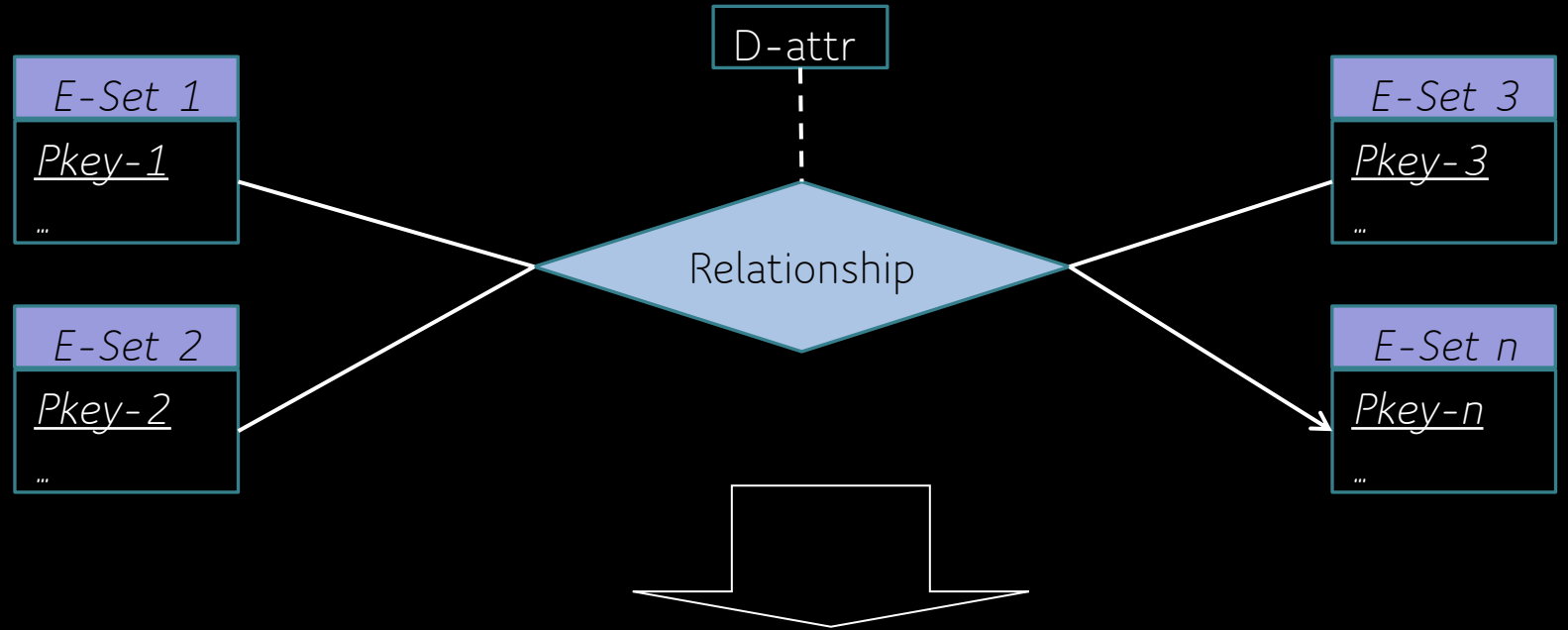
prereq = (*course_id*, *prereq_id*)

course_id and *prereq_id* are both foreign keys that refer to primary key of *course* (*course_id*)

Representing Ternary (and n-ary) Relationships

One relation for each entity set and one for the relationship set

The relation from the relationship set has foreign keys to each entity set in the relationship set



<u>Pkey 1</u>	<u>Pkey-2</u>	<u>Pkey-3</u>	Pkey-n	D-attr
9999	8888	7777	6666	Yes
1234	5678	9012	3456	No

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

- If specialization is total, the schema for the generalized entity set not required to store information
 - Can be defined as a “view” relation containing union of specialization relations
 - But explicit schema may still be needed for foreign key constraints
- Drawback: *name*, *street* and *city* may be stored redundantly for people who are both students and employees

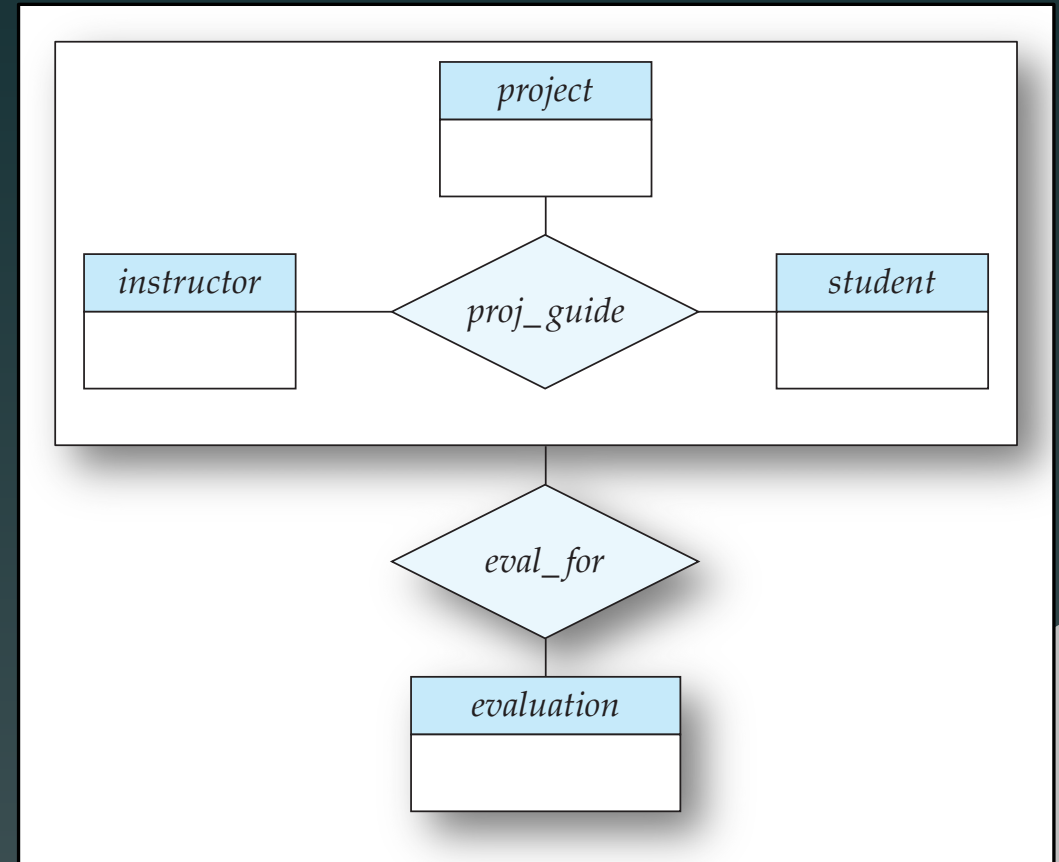
Reducing Aggregation to Relational Schemas

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.



Latihan - Lakukan translasi ER Model ke dalam Relational Model

