50.043 Database and Big Data Systems

Relational Model

Roy Ka-Wei Lee
Assistant Professor, ISTD, SUTD

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

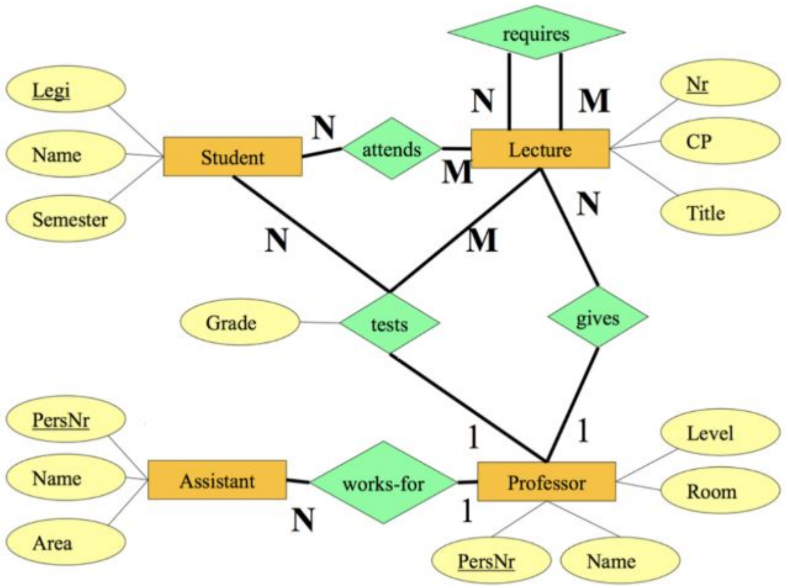
Learning Outcome

By the end of this lesson, you should be able to

- Describe the components of a relational model
- Translate an ER model into a relational model

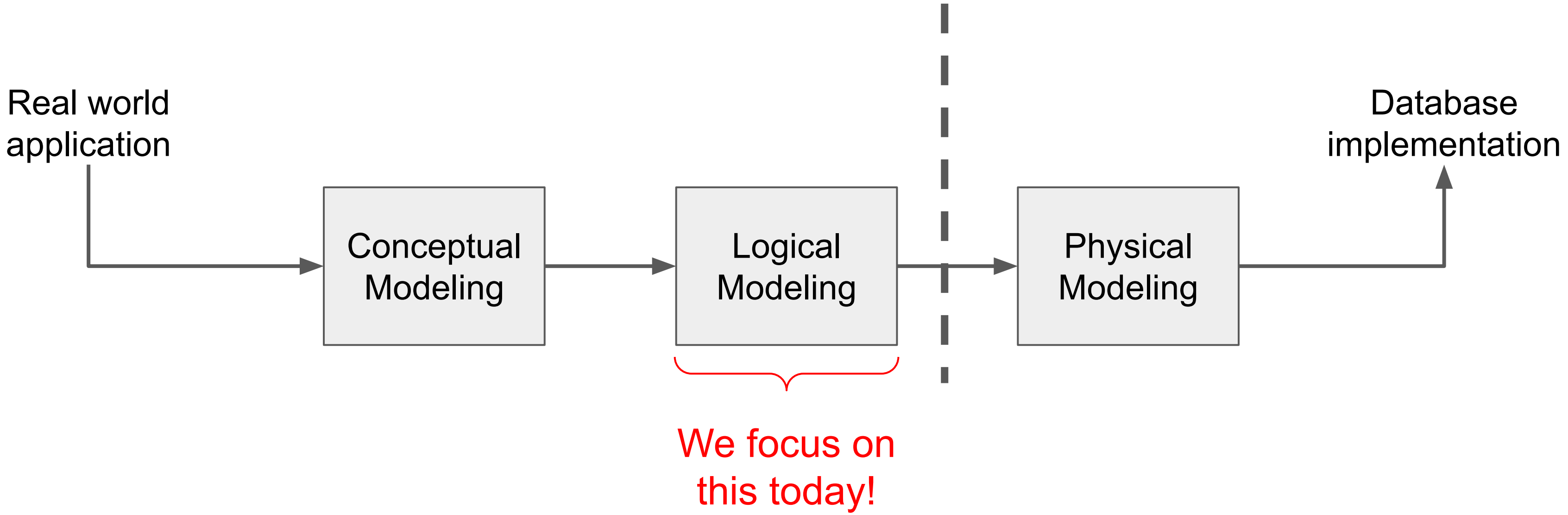
Recap - Where We Left Off…

- **ER Model** - Conceptual Model
- Describe **what data** the application has
- Building blocks:
  - Entity set: a collection of similar objects
  - Attribute: property of an entity
  - Relationship: connection between entity sets
    - Cardinality Constraints



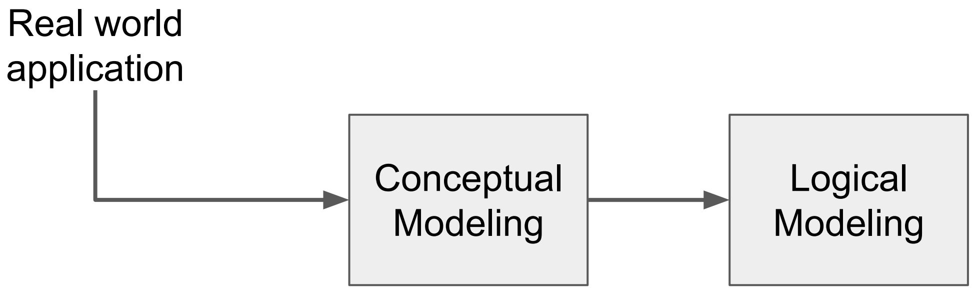
SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Recall This Diagram….



Recall This Diagram….

- Entity-Relationship (ER) Model: Conceptual
  - Describe **what data** the application has
  - Normally based on the user needs
- Relational Model: Logical
  - Describe **what structure** the data has
  - Your high level design!

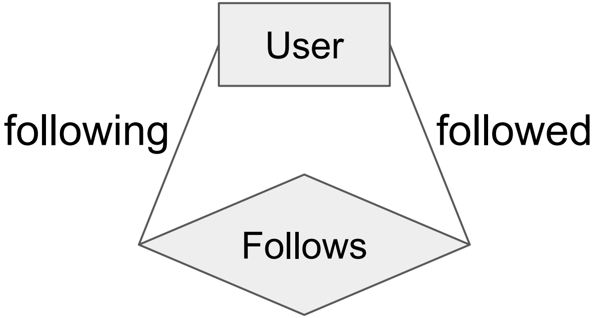


Data Model

- **Conceptual model** lets us describe *the data*
  - needed by the applications
  - Without worrying about how it is stored or processed
- **Logical model** let us describe *how data*
  - Is stored
  - Is processed
  - Without getting into the implementation details

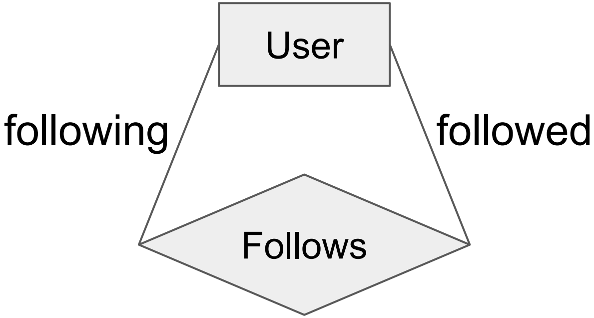
Data Model

- How do you implement this?

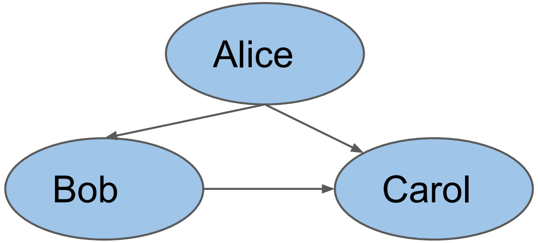


Data Model

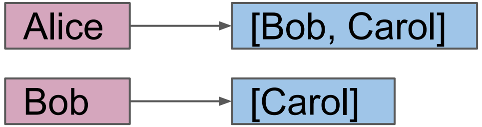
- How do you implement this?



Graph



Key-Value

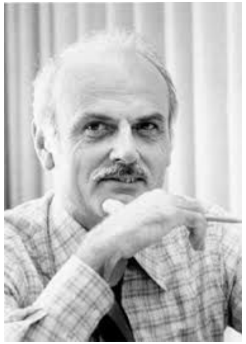


Relational

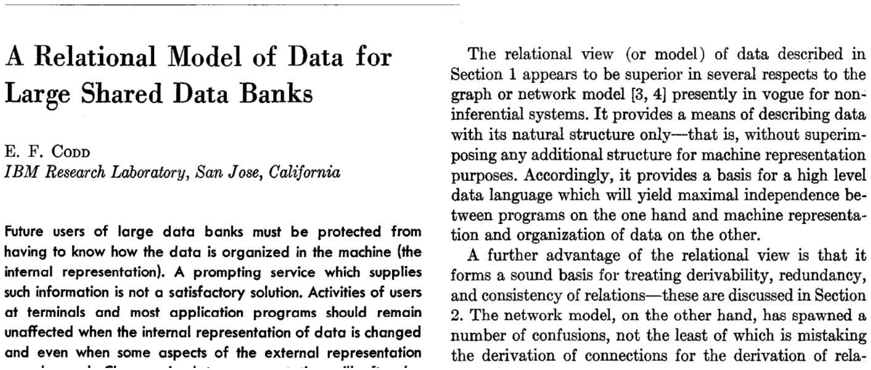
User	Following
Alice	Bob
Alice	Carol
Bob	Carol

Relational Model

- One of the most important ideas in computer science
  - Gave us 2 Turing award winners, Codd and Stonebraker



[Ted Codd]



A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of rela-

[1970]

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Relational Model

- Relation: an unordered set containing relationship of attributes
  - Attribute = field
- Tuple: sequence of attribute values in the relation
  - Relation = {tuples}

Relation: Student

Attribute (column)

Schema

Tuple (row)

Student Number	Name	Email	DoB
1234	James	james@istd	1/1/2000
5319	Vanessa	vanessa@epd	2/4/1999
3093	David	david@esd	3/7/2000

Relational Model

Attribute (column)

Relation: Ledger

Schema

Tuple (row)

Account	Account	Amount	ReferenceID
12343209	983620935	23	09309sdglkjwe
30937694	800967032	78	oiu093jhosgosi
09094672	390932623	89	209sdgkljoi390

Relational Model

- Relation is the same as table?
  - **No**, relation must be a set: no duplicate rows.
  - Other constraints (next few slides)
- Why call it a relation?
  - Because it is a ***mathematical relation***

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Relation Schema and Instance

- A_1, A_2, …, A_n are ***attributes***
- R = (A_1, A_2, …, A_n) is a ***relation schema***
  - Example: *instructor* = (*ID*, *name*, *dept_name*, *salary*)
- A **relation instance** *r* defined over schema *R* is denoted by *r* (*R*).
- The current values a relation are specified by a table
- An element ***t*** of relation ***r*** is called a ***tuple*** and is represented by a *row* in a table

Attributes

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value *null* is a member of every domain. Indicates that the value is “unknown”
    - The null value causes complications in the definition of many operations

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: instructor relation with unordered tuples

Student Number	Name	Email	DoB
1234	James	james@istd	1/1/2000
5319	Vanessa	vanessa@epd	2/4/1999
3093	David	david@esd	3/7/2000

Database Schema

- **Database schema** -- is the logical structure of the database.
- **Database instance** -- is a snapshot of the data in the database at a given instant in time.
- Example:
- schema: student (student number, name, email, DoB)
- Instance:

Student Number	Name	Email	DoB
1234	James	james@istd	1/1/2000
5319	Vanessa	vanessa@epd	2/4/1999
3093	David	david@esd	3/7/2000

Keys

- Let $K \subseteq R$
- *K* is a **superkey** of *R* if values for *K* are **<u>sufficient to identify a unique tuple</u>** of each possible relation *r(R)*
  - Example: {*Student Number*} and {*Student Number*, name} are both superkeys of *student.*
  - How do you know *K is* a superkey?
- Superkey *K* is a **candidate key** if *K* is minimal
  - Example: {*Student Number*} is a candidate key for *student*
- One of the candidate keys is selected to be the **primary key**.

Keys

- Let $K \subseteq R$
- *K* is a **superkey** of *R* if values for *K* are **sufficient to identify a unique tuple** of each possible relation *r(R)*
  - Example: {*Student Number*} and {*Student Number*, name} are both superkeys of *student.*
  - How do you know *K is* a superkey?
- Superkey *K* is a **candidate key** if *K* is minimal
  - Example: {*Student Number*} is a candidate key for *student*
- One of the candidate keys is selected to be the **primary key**.

Foreign Key

- **Foreign key** constraint: Value in one relation must appear in another
    - **Referencing** relation
    - **Referenced** relation
- Example:

Referencing

ID	Name	Dept_Name
1234	Roy	ISTD
5319	Dario	SMT
3093	Shaohui	EPD

Dept_Name	#Num_Faculty
ISTD	55
SMT	66
EPD	77

Foreign Key

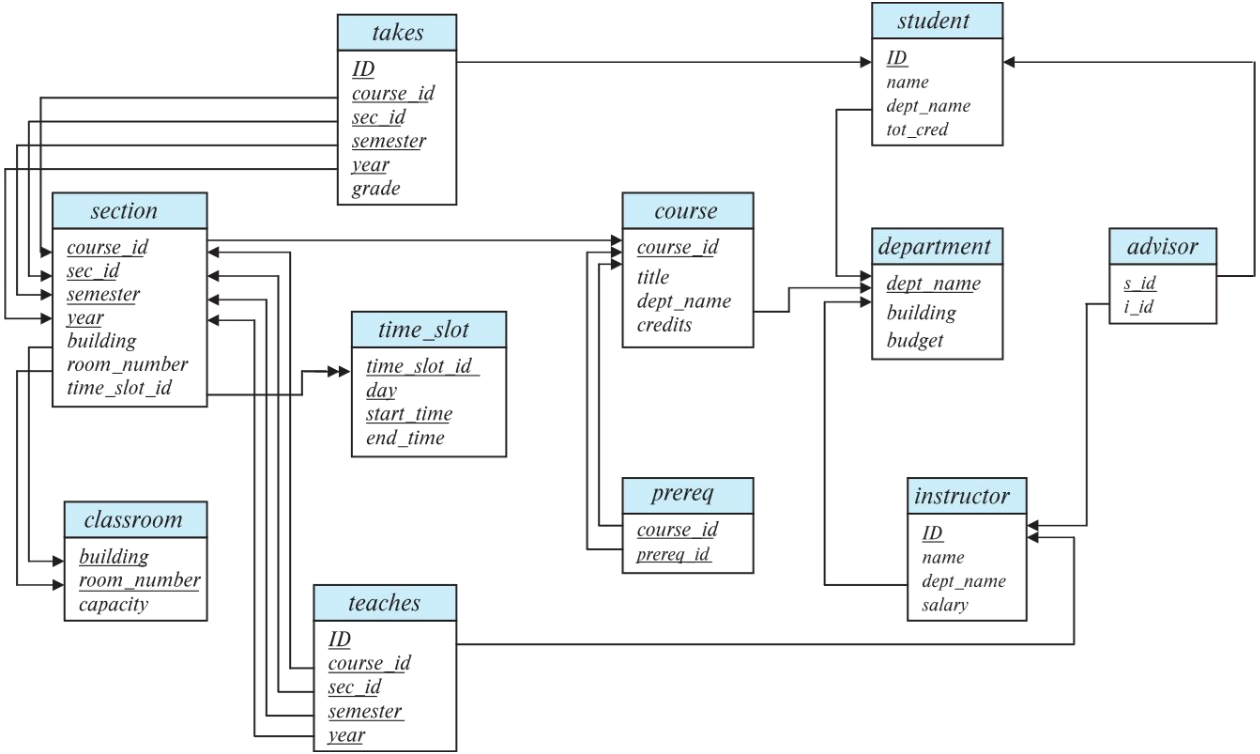
- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
- Example:

Referencing

<u>ID</u>	Name	Dept_Name
1234	Roy	ISTD
5319	Dario	SMT
3093	Shaohui	EPD

Dept_Name	#Num_Faculty
ISTD	55
SMT	66
EPD	77

Schema Diagram for University Database



SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

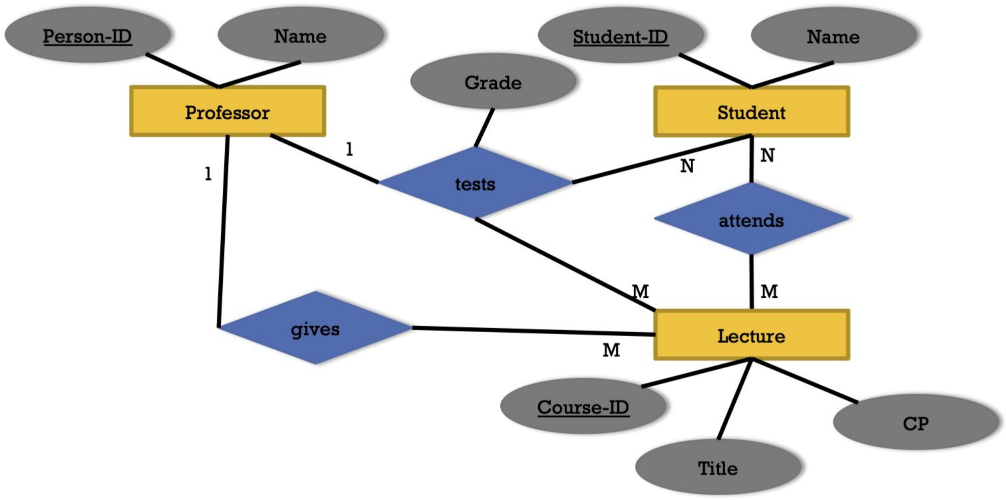
So… How do we convert
ER Diagram or
Relational Data Model?

ER To Relations

Rule 1: Entity set → Relation

- Preserve fields + primary key

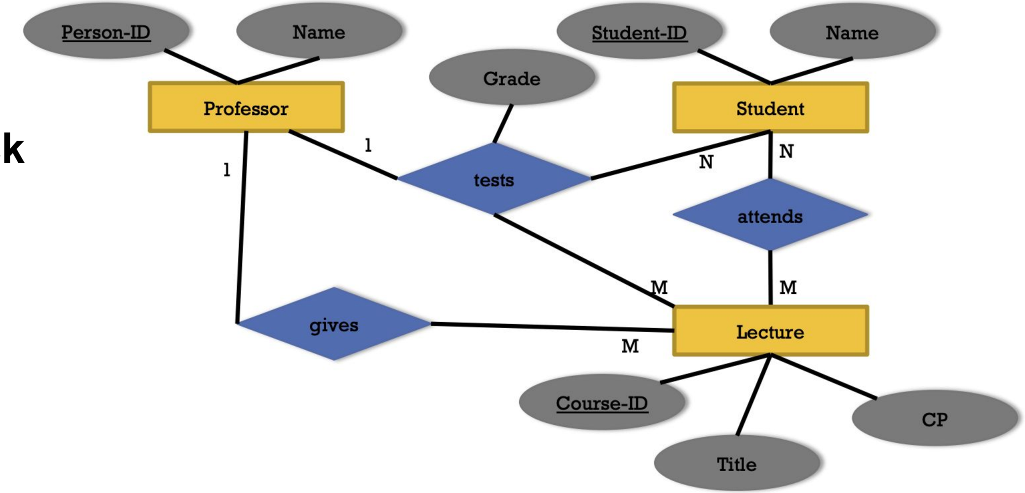
Professor(**<u>Person-ID</u>**,Name)
Student(**<u>Student-ID,</u>** Name)
Lecture(**<u>Course-ID</u>**, Title, CP)



ER To Relations

Rule 2: Relationship → Relation

- Combine all keys from entity sets to make a new primary key
- Many combination, **need to check for constraints**

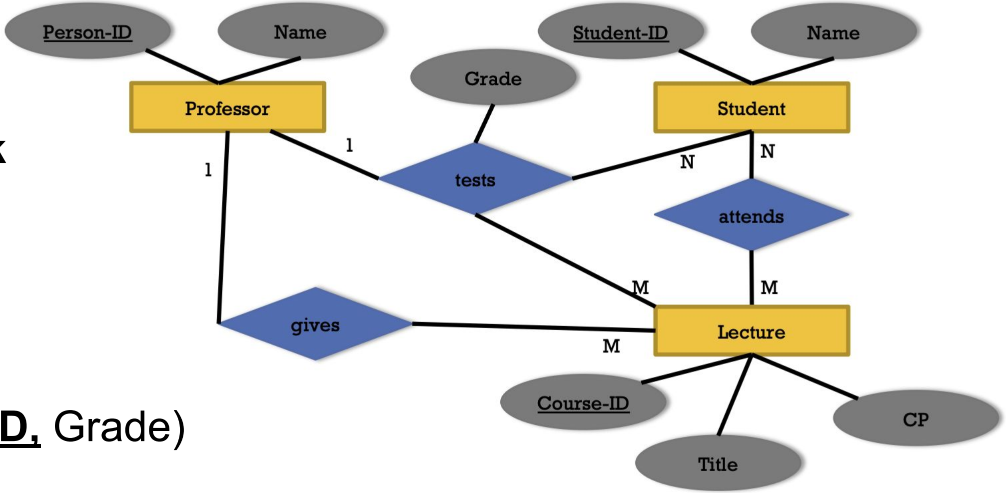


Gives(**<u>Person-ID, Course-ID</u>**)
Gives(Person-ID, **<u>Course-ID</u>**)
Gives(**<u>Person-ID</u>**, Course-ID)

ER To Relations

Rule 2: Relationship → Relation

- Combine all keys from entity sets to make a new primary key
- Many combination, **need to check for constraints**



Gives(Person-ID, **Course-ID**)

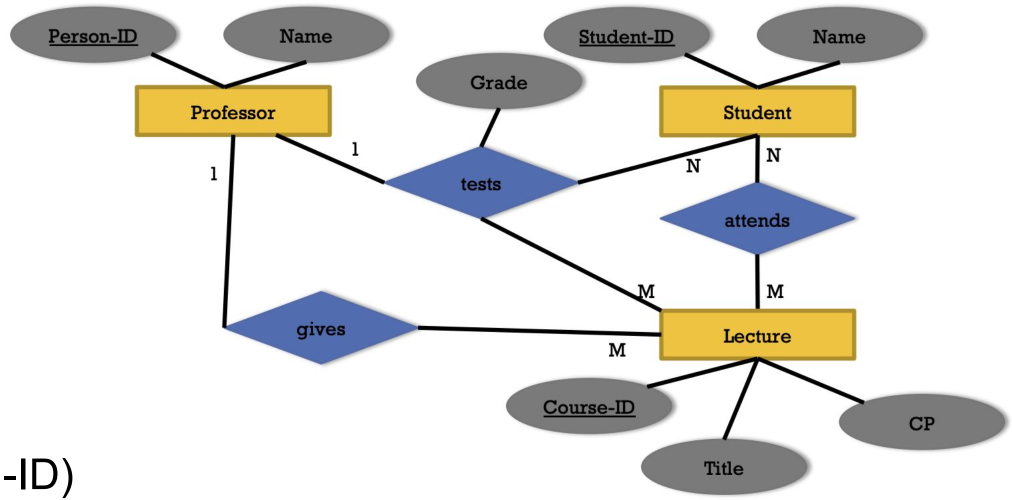
Tests(Person-ID, **Course-ID, Student-ID,** Grade)

Attends(**Student-ID, Course-ID**)

ER To Relations

Rule 3: Merge relations with same key

- But avoid data redundancy (NULL values at some fields)



Gives(Person-ID, **Course-ID**)

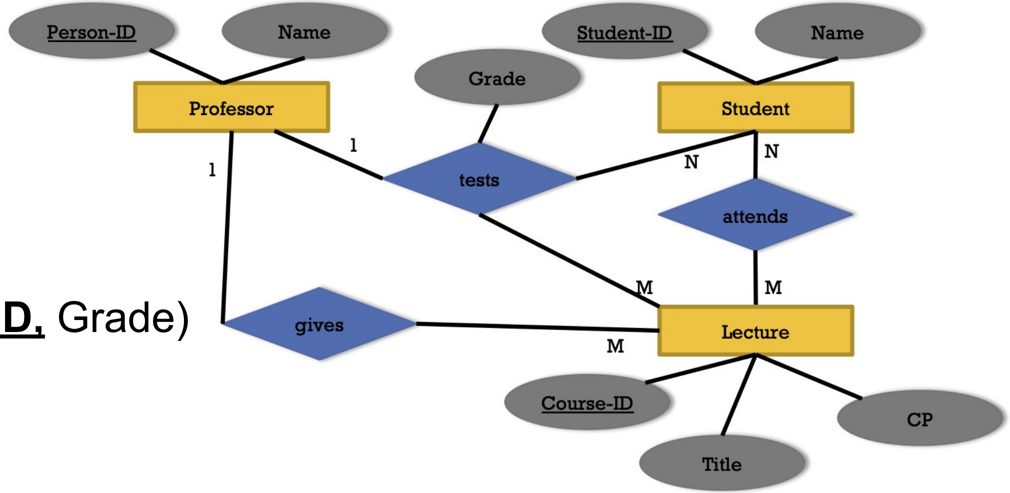
Lecture(**Course-ID**, Title, CP)

→ Lecture(**Course-ID**, Tile, CP, Person-ID)

ER To Relations

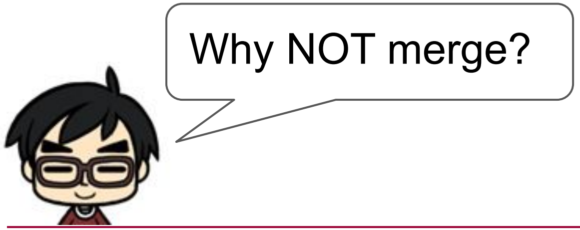
Rule 3: Merge relations with same key

- But avoid data redundancy (NULL values at some fields)



Tests(Person-ID, **<u>Course-ID, Student-ID,</u>** Grade)

Attends(**<u>Student-ID, Course-ID</u>**)



ER To Relations

Attends

StuID	CourseID
1	50043
2	50043

Student

Name	StuID
Aaron	1
Beatrice	2

Professor

PerID	Name
1	Anh
2	Kenny

Lecture

CourseID	Title
50043	DB
50042	Security

Merging **Attends** with **Tests** implies dropping **Attends**. As a result, we lost the information that *Aaron took the 50042 test w/o attending it*.

Tests

StudID	CourseID	PerID	Grade
1	50043	1	A
2	50043	1	A
1	50042	2	A

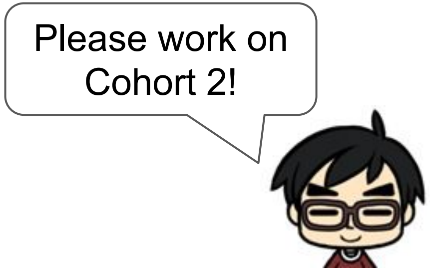
Gives

PerID	CourseID
1	50043
2	50042

What you should know?

- What are the components of a relational model?
- How do you translate an ER model into a relational model?

Reading Resources:
- https://sutd50043.github.io/notes/l2_relational_model/



Acknowledgement

- *The following material have been referenced or partially used:*
  - *MIT Database Systems (6.830)*
  - *University of Washington: Introduction to Data Management (CSE344)*
  - *CMU Database Systems (15-445/645)*
  - *ETH's Data Modeling and Databases (252-0063-00L)*
  - *ETH's Big Data For Engineers*
  - *Yale's Database System Concepts Seventh Edition (https://codex.cs.yale.edu/avi/courses/CS-437/slides/index.html)*