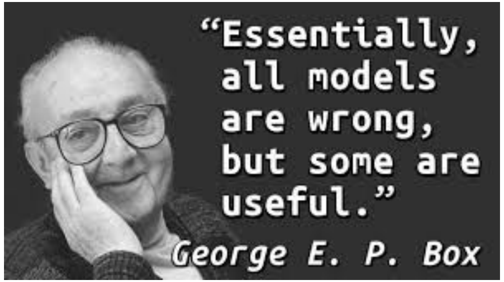
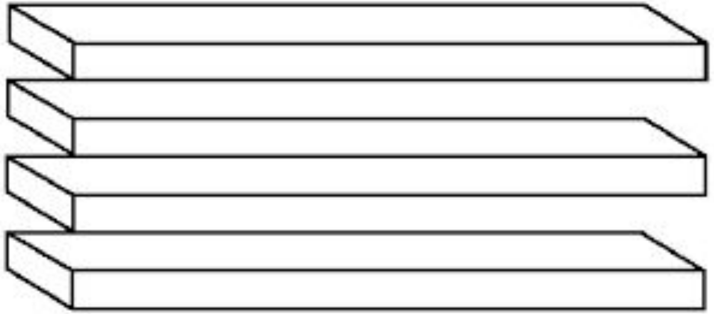
Databases and Big Data

Data Model

Conceptual Model

- How do you **<u>describe</u>** <u>the application to other users</u>?
  - Easy (for others) to understand
  - Without ambiguity (bad example: *“it stores student profiles”*)



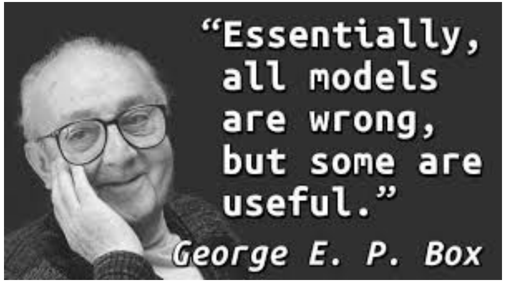


Real world application

Conceptual Modeling

Data Model

- How do you **describe the data to the database**?
  - In a language the database understand



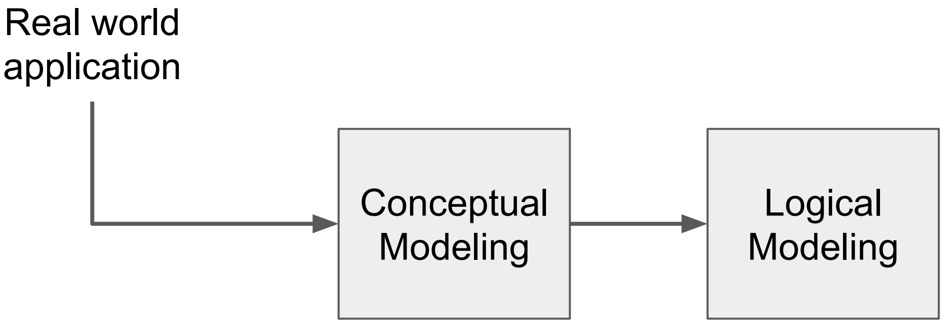
Real world application → Conceptual Modeling → Logical Modeling

Relational	Most database
Key/value Graph Document	No-SQL
Array/Matrix	Machine Learning

Today

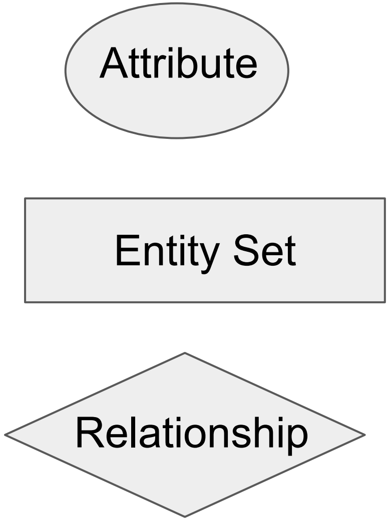
Entity Relationship Diagram

- ER Model: conceptual
  - Describe **what data** the application has
- Data Model: logical
  - Describe **what structure** the data has



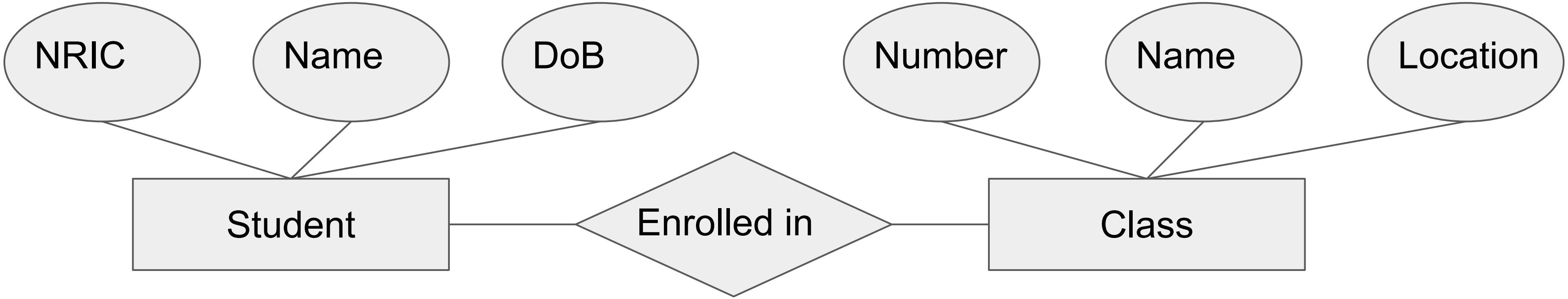
ER Model

- A graphical diagram
  - Because a picture worth a thousand words
- ER model consists of:
  - Entity: an object
  - Entity set: a collection of similar objects
  - Attribute: property of an entity
    - Entities in the same entity set have the same set of attributes
  - Relationship: connection between entity sets



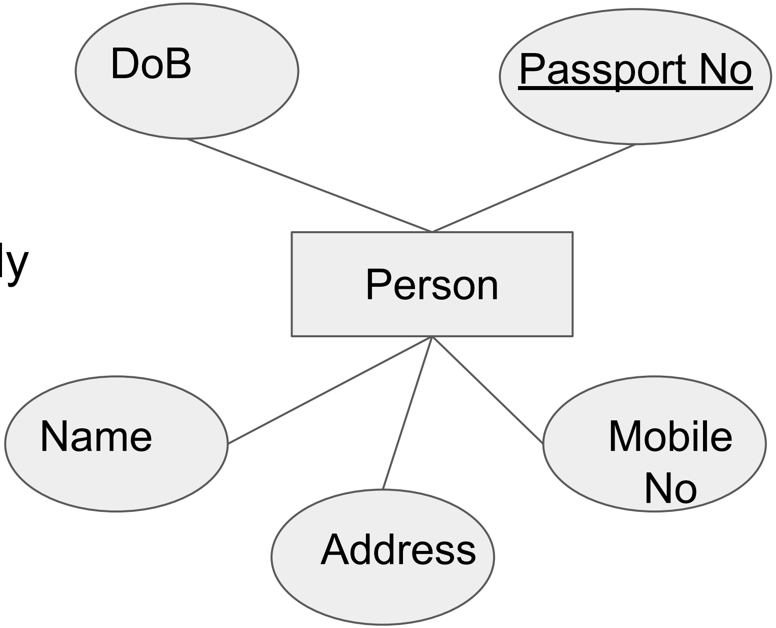
ER Model

A student has a name, date of birth, NRIC number. A student can enrol to a class. Each class has a name, a number, and is held at a lecture theater.



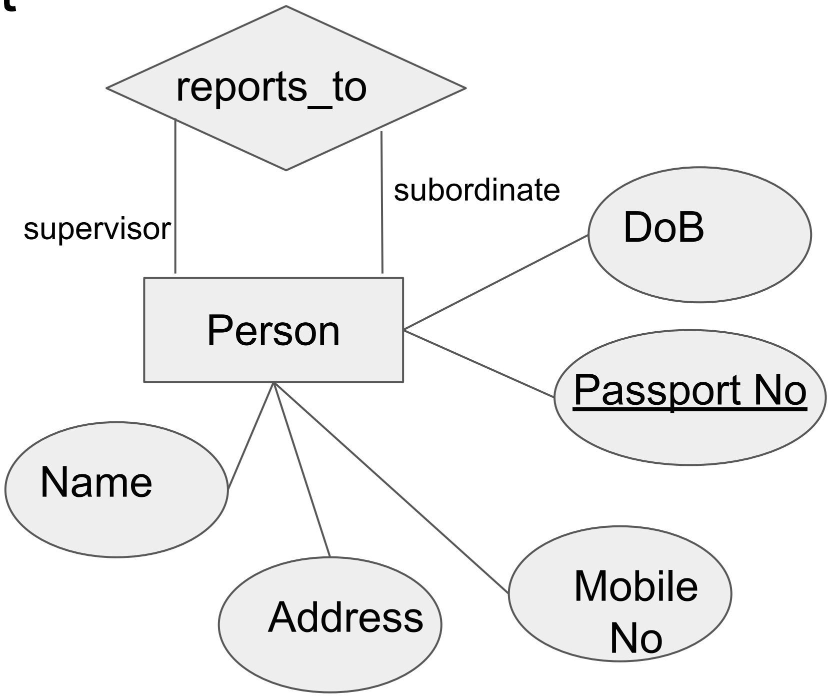
Entity Set

- It's a **set**
  - Every entity is unique
- <u>Primary Key</u>:
  - *minimal set of attributes* that uniquely identifies an entity in the set
- Example:
  - <u>Passport</u>
  - or (<u>Passport, Mobile, DoB, Name</u>) ?



Relationship

- Between entities of the same set
  - Prof / Student
  - Supervisor / Subordinate
  - etc.
- <u>Role:</u> model relationship in the same set



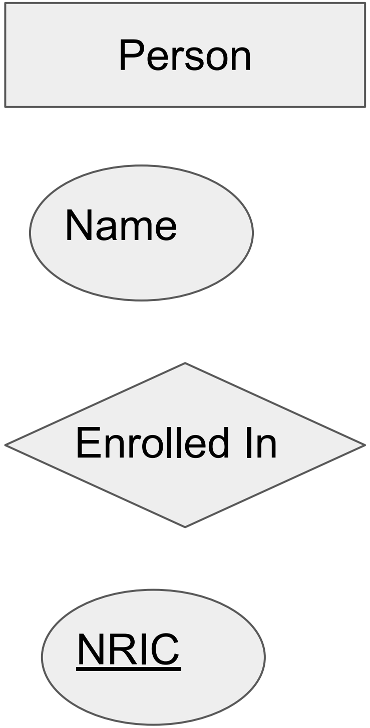
Again: ER Diagram

Entity Set

Attribute

Relationship

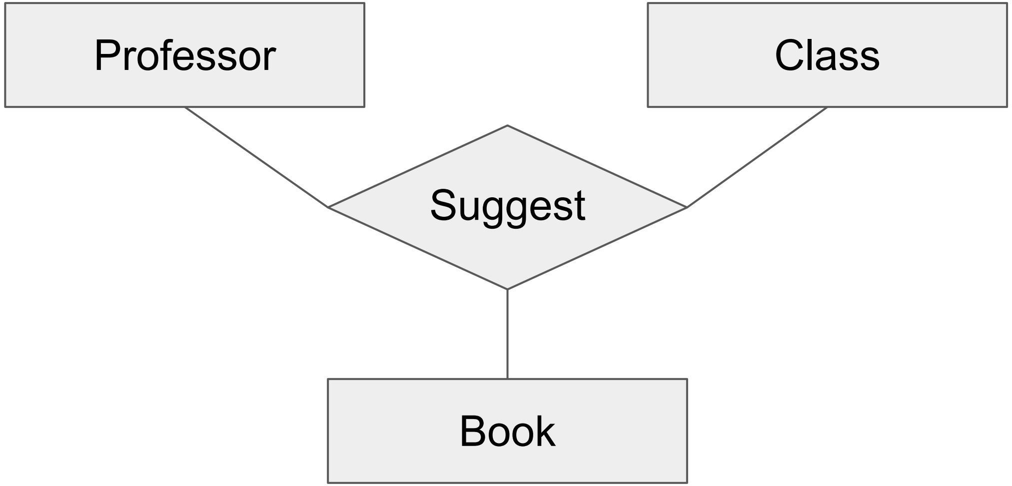
Pirmary Key



Role

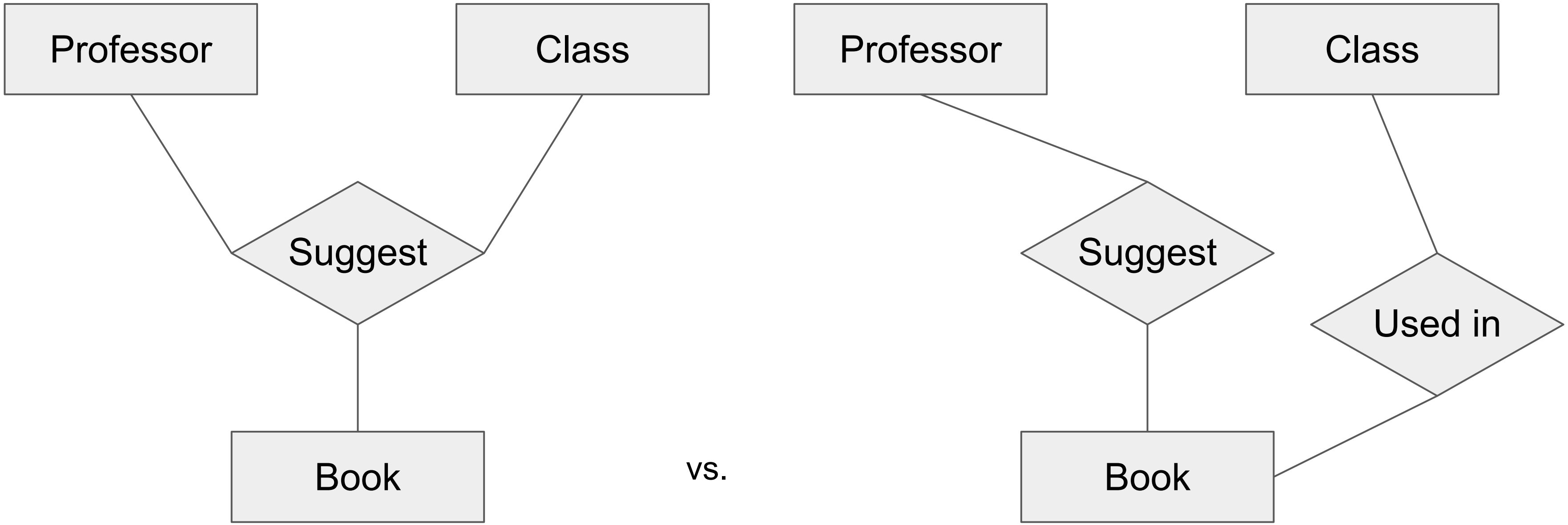
Multi-Way Relationship

- More than 2 entity sets
  - A professor can suggest books used in a class



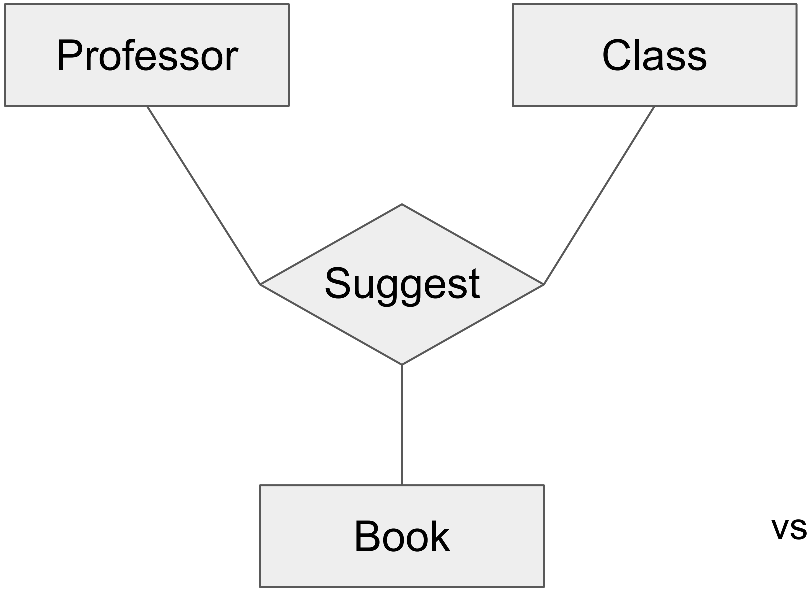
Which Model Is Better?

>> *ok*

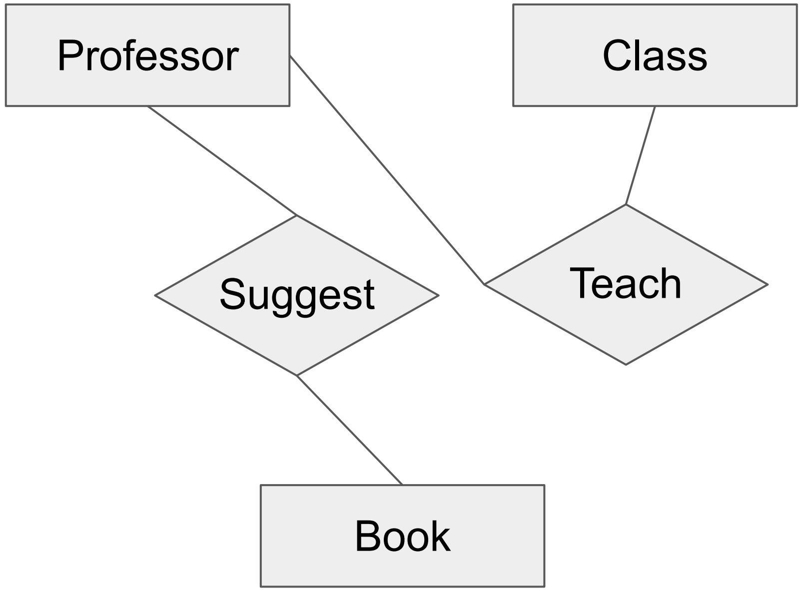


ER Models

>> no
relationship between book and
class is lost

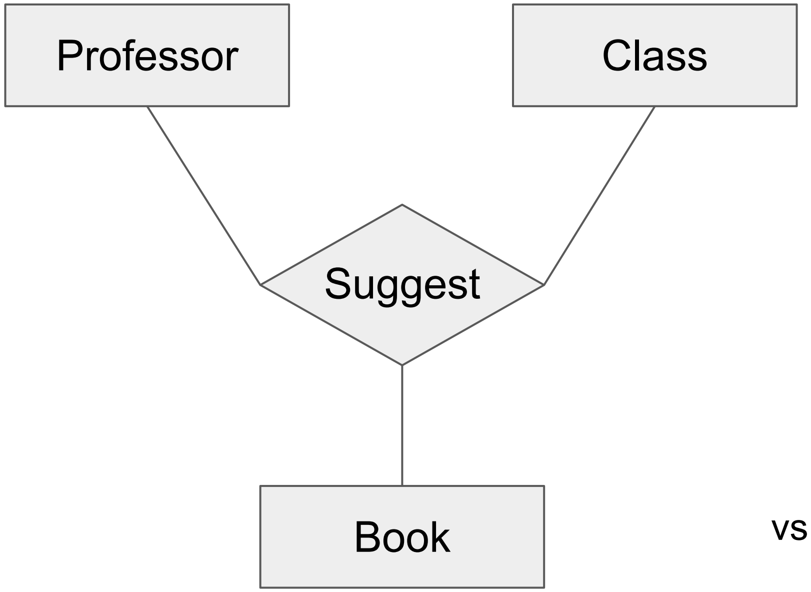


vs.

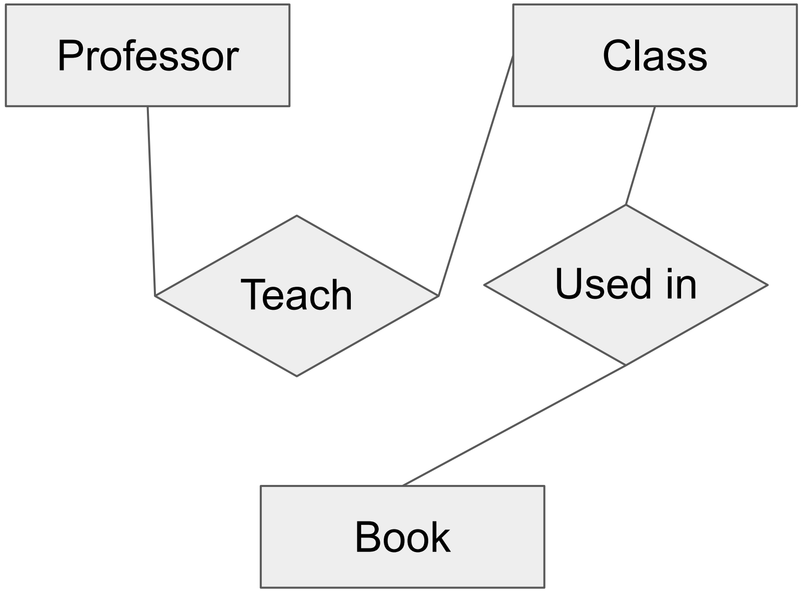


ER Models

>> no
lost meaning of the suggest relationship
Professor suggest books used in a class, not necessarily professor has to teach in that class and the book used in that class has to be suggested by the professor teaching.



vs.

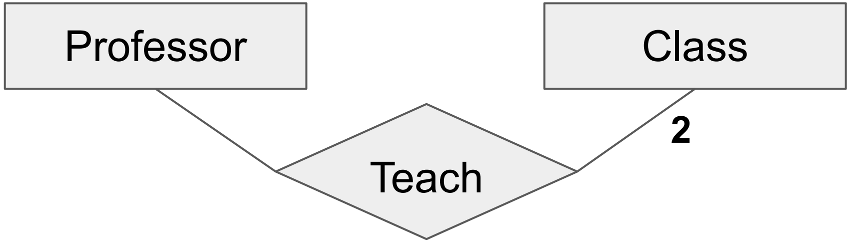


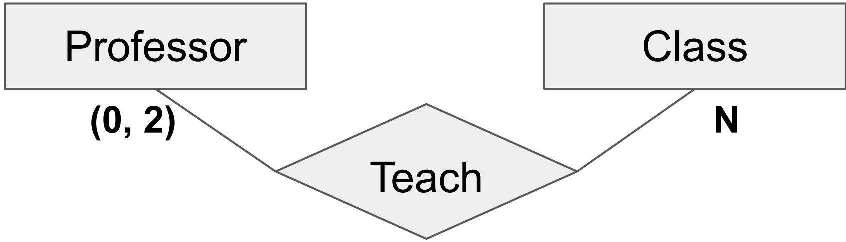
In this class, they won't ask 'exactly 2'. Only up to or N or a specific range.

Exactly 2 -> (2,2)

ER Model

- Cardinality constraints
  - Example 1: each professor teaches up to 2 classes
  - Example 2: each professor teaches many classes, each class has at most 2 professors

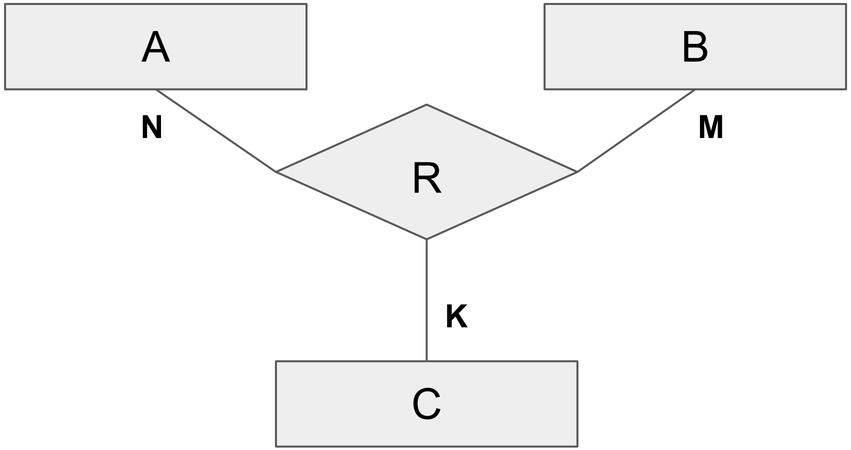




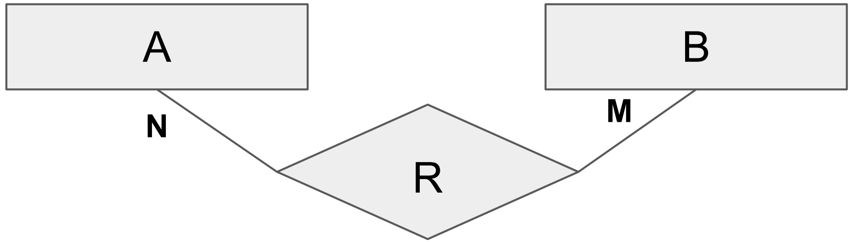
ER Model

- Cardinality constraints

1 *pair* of entity in (A,B) has relation R to **K** entities in C
1 *pair* of entity in (A,C) has relation R to **M** entities in B
1 *pair* of entity in (B,C) has relation R to **N** entities in A



1 entity in A has relation R to **M** entities in B
1 entity in B has relation R to N entities in A



ER Model

What does this say?

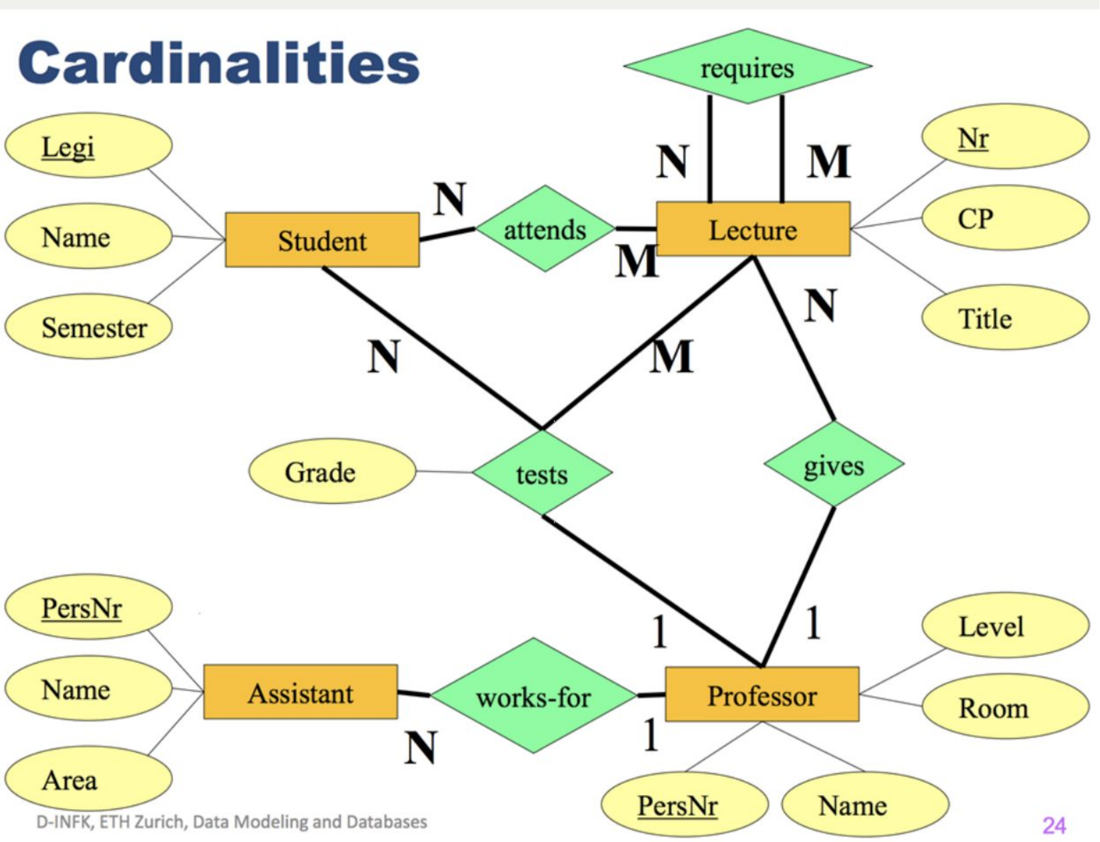
Important

For the relation Test:

*- A student in a lecture is tested by **1** professor*

*- A professor teaching in a lecture tests **N** students.*

*- A student & a Professor can have tests in **M** lectures.*



ER Recap

- ER diagram:
  - Entity Set, Attributes, Relationship, Primary Key, Role, Cardinality
- Advantages:
  - Quick to draw
  - Easy to understand
- Limitations:
  - Relationship between sets: person’s existence depends on a set of functioning organs
  - Negative relationship: Asst Prof cannot be Assoc Prof at the same time

__Limitations:__

- ER diagrams only for relational data (to show relationships)

- Not for unstructured data

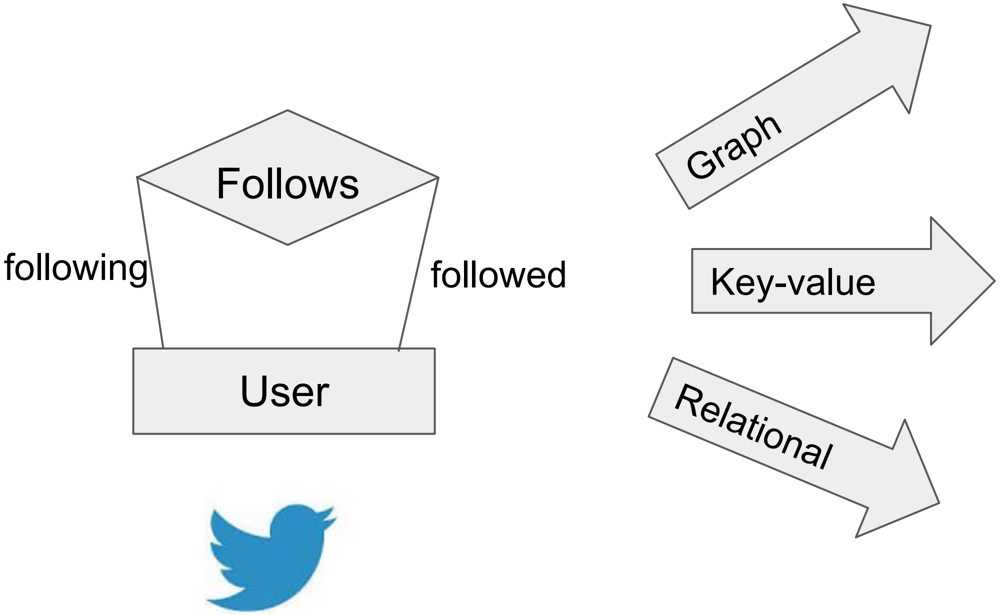
ER Recap

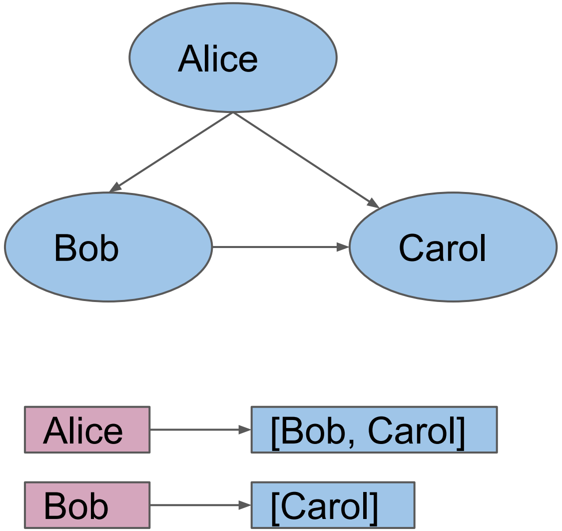
- ER in Database ~ UML in Software Engineering
- Are they useful in practice? Debatable
- Does everyone have to learn them? YES!

Data Model

- ER model lets us describe *the application*
- Data model lets us describe *the data*
  - Without worrying about how it is stored or queried.

Data Model





User	Following
Alice	Bob
Alice	Carol
Bob	Carol

Relational Model

- One of the most important ideas in computer science
- Relation: an unordered set containing relationship of attributes
- Tuple: sequence of attribute values in the relation
  - Relation = {tuples}

Relational Model

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

- Attribute also called Field

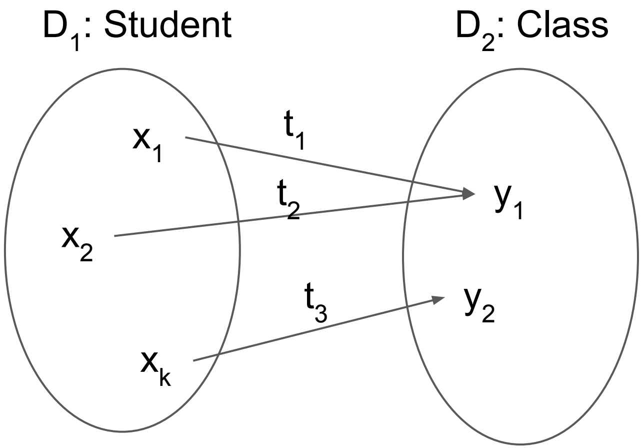
Relation = Table?

Are they just … tables?

- Well… yes, but:
  - Not any table (next few slides)
  - Must be a set: no duplicate rows.
  - And others
- Why call it a relation?
  - Because it is a *mathematical relation*

Relational Model

- Relation
  - $R \subseteq D_1 \times D_2 \times D_3 \times .. \times D_N$
  - $D_1, D_2, .., D_N$ are domains, or *fields*, or *attributes*
  - Each domain has a *type*
- Tuple = record
  - $t = (d_1, d_2, .., d_n \mid d_i \in D_i) \in R$
- Schema:
  - Associate domains to name
  - $S_R = \{D_i : name_i\}$



Relational Model

Database is a Set of Relations

- Let's agree on the terminologies:
  - Relation: the entire set of all possible tuple
    - $R \subseteq D_1 \times D_2 \times D_3 \times .. \times D_N$
    - When talk about the table in general
  - Relation instance: one subset
    - $I_R \in R$
    - When refer to a specific table
- Example:
  - Student $\subseteq$ Integer $\times$ String $\times$ String $\times$ Date
  - $I_{Student}$ = {(3093, David, david@esd, 3/7/2000), (1234, James, james@istd,1/1/2000), …}

Relational Model

- Mapping from relational model to table:
  - Relation → Table
  - Domain → Column
  - Tuple → Row
  - Schema → Column names
- Examples:
  - (**<u>Student number</u>**, Name, Email, DoB)
  - (Account number, Account number, Amount, **<u>Reference ID</u>**)
  - (**<u>Class number, Student number,</u>** Grade)

Table semantics = Relation semantics:

1. Set → must have a primary key
2. Domain has type → cannot put any value in
3. Flat → no complex object at each domain

Not all tables are relations

Relational Model

- Integrity constraints:
  - Conditions that must be met if a tuple is valid
- Primary key:
  - So that the relation is a set
- Foreign key:
  - Enforce relationship between relation
  - Value in field A of R_1 must exist in field A of R_2

Integrity Constraint

Relational model rejects it because same primary key.

<u>Account number</u>	Name	Balance
12343209	James	2093
30937694	Vanessa	3532
19047639	David	8073

Add *(19047639, Anh, 2321)*

REJECTED

Integrity Constraint

Account number	Name	Balance
12343209	James	2093
30937694	Vanessa	3532
19047639	David	8073

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

*Although **(09094672, Account)** is not a primary key, it is a foreign key (linked to the Account Number of another relation.*

Add (09094672, 12345678, *23, lksjgoiuwojg)*

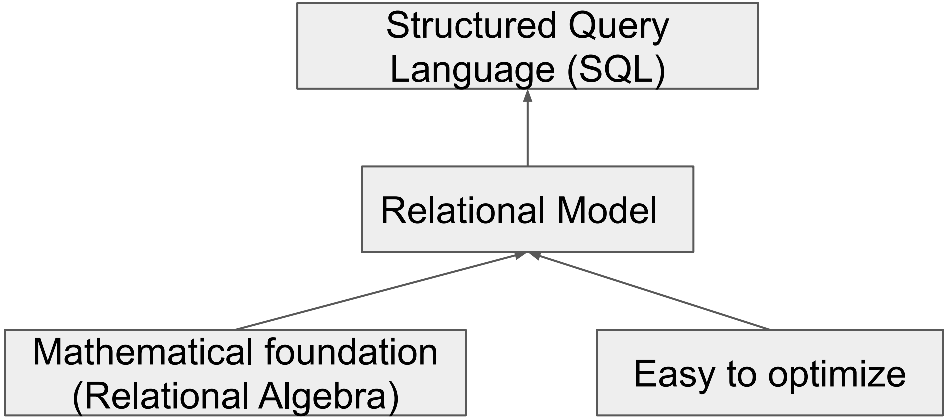
REJECTED

Hence, this is rejected because 09094672 is not found in the Account Relation.

Rmb:
Foreign key -- Value in field A of R1 must exist in field A of R2

Relational Model: Recap

- Beautifully simple:
  - Tables with some constraints
- Extremely powerful:
  - Manipulated with SQL
- Rigorous:
  - Built on strong mathematical foundation



Relational Model: Recap

- Many alternative models exist
- Relational remains on of the most popular
- Interested in history?

What Goes Around Comes Around

Michael Stonebraker
Joseph M. Hellerstein

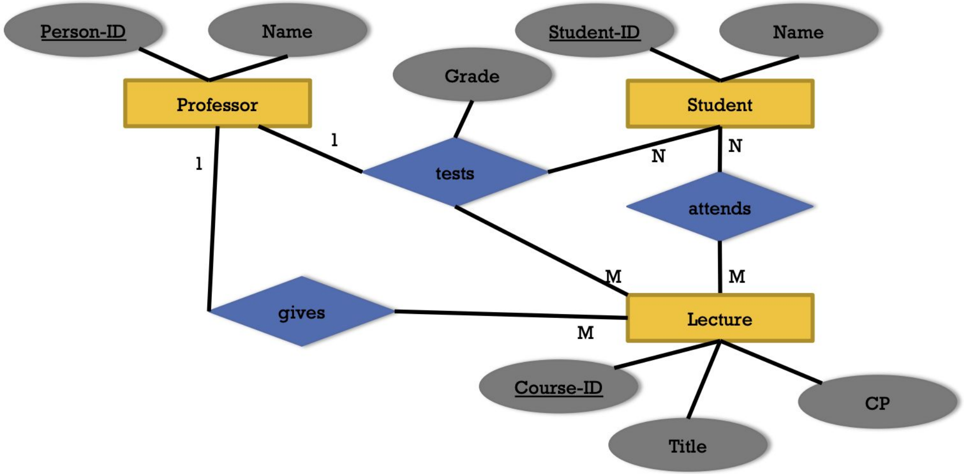
Abstract

This paper provides a summary of 35 years of data model proposals, grouped into 9 different eras. We discuss the proposals of each era, and show that there are only a few basic data modeling ideas, and most have been around a long time. Later proposals inevitably bear a strong resemblance to certain earlier proposals. Hence, it is a worthwhile exercise to study previous proposals.

ER Diagram

Relational Data Model

ER To Tables



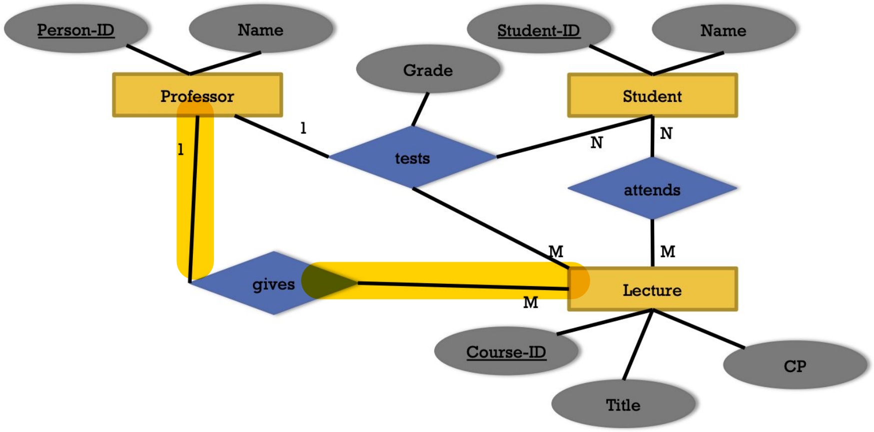
Rule 1: Entity set → Relation

Preserve fields and primary key

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

ER To Tables

*A prof gives **M** lectures*
*A lecture is given by **1** prof.*
*When **PersonID** is the **pk**, a prof can only teach 1 lecture **XXX***
*When **CourseID** is the **pk**, a course can only be taught by 1 prof*
YES



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**)

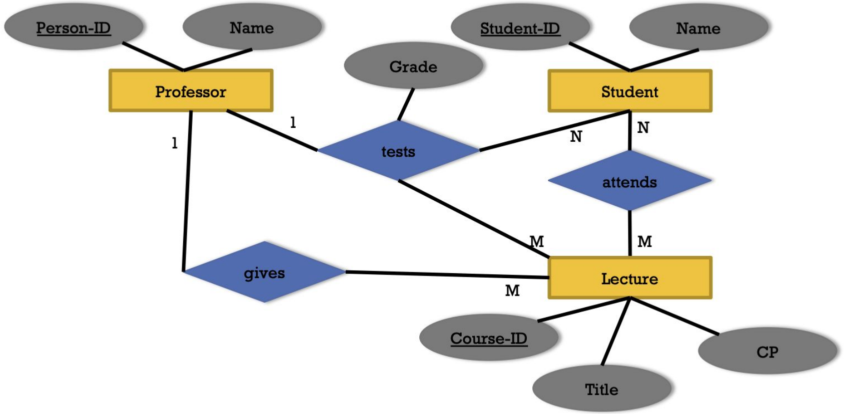
Gives(Person-ID, **Course-ID**)

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

ER To Tables

*- A student in a lecture is tested by **1 professor***
*- A professor teaching in a lecture tests **N students.***
*- A student & a Professor can have tests in **M lectures.***

***Observation:** When cardinality is **N/M**, that ID should be a **pk**.*



Rule 2: Relationship → Relation

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

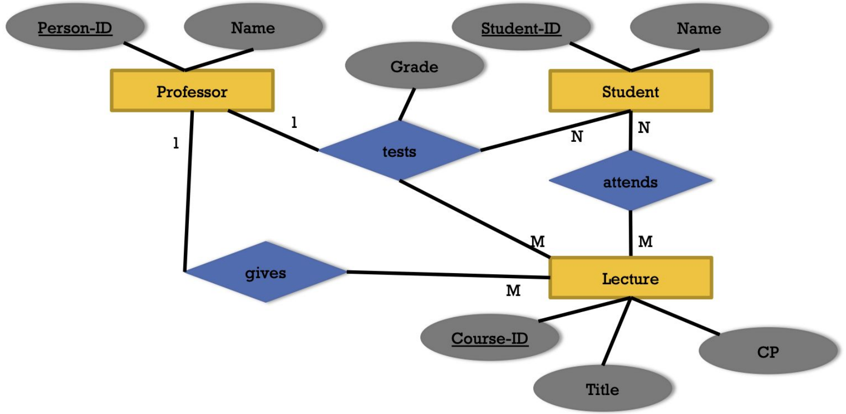
Composite PK

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

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

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

ER To Tables



Rule 3: Merge relations with same key

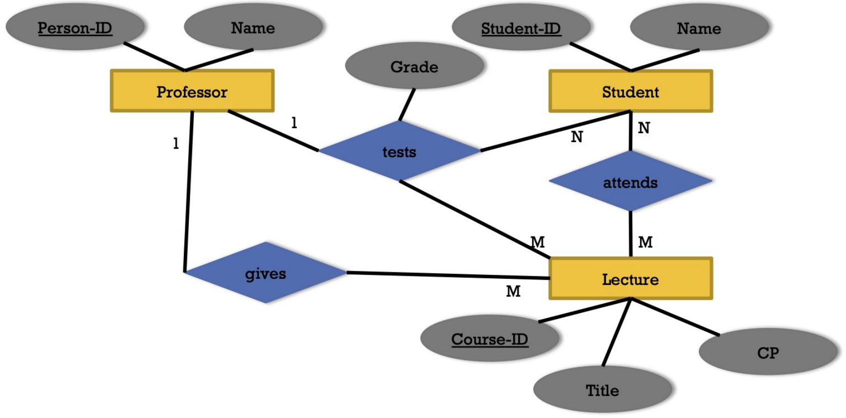
- Avoid data redundancy (NULL values at some fields)

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

Lecture(**<u>Course-ID</u>**, Title, CP)

→ Lecture(**<u>Course-ID</u>**, Tile, CP, Person-ID)

ER To Tables



Rule 3: Merge relations with same key

- Avoid data redundancy (NULL values at some fields)

Needs to make sure relations still make sense

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

→ Why NOT merge? (hint: many classes don't have tests)

ER Diagram

Relational Data Model

Rule 1: Entity set → Relation
Rule 2: Relationship → Relation
Rule 3: Merge relation

Summary

Remember this!