DATABASES

INTRODUCTION TO DATA SCIENCE

TIM KRASKA

teaching datascience .org

DATA PROCESSING PIPELINES

Colin Mallows:

1. Identify data to collect and its relevance to your problem
2. Statistical specification of the problem
3. Method selection
4. Analysis of method
5. Interpret results for non-statisticians

Peter Huber:

1. Inspection
2. Error checking
3. Modification
4. Comparison
5. Modeling and model fitting
6. Simulation
7. What-if analyses
8. Interpretation
9. Presentation of conclusions

Ben Fry:

1. Acquire
2. Parse
3. Filter
4. Mine
5. Represent
6. Refine
7. Interact

Our Definition

1. Preparing to run a model
2. Running the model
3. Communicating the results

CLICKER QUESTION

How well do you know databases

A. **What are they?**

B. **I used a relational database in the past, but don't really know how they work.**

C. **I know SQL and tables**

D. **I know SQL, ER diagrams, and the relational algebra**

E. **I know normalization (e.g., 4th normal form) and, star and snowflake schemas**

WHY DATABASES

Why not store everything in flat files?

WHY DATABASES

Why not store everything in flat files?

- Scalability → 100’s of nodes

WHY DATABASES

Why not store everything in flat files?

- Scalability → 100's of nodes

- Data redundancy and inconsistency

Name	Course	E-Mail	Grade
John Doe	CS112	jd@cs.brown.edu	B
C. Binnig	CS560		A
John Doe	CS560	John_doe@brown.edu	B

First Name	Last Name	Teaches	E-Mail	Grade
J.	Doe	CS112	jd@cs.brown.edu	B
Mike	Stonebraker	CS560	stonebraker@uni.edu	A
Carsten	Binnig		Carsten_binnig@brown.edu	B

Why is this a problem?

WHY DATABASES

Why not store everything in flat files?

- Scalability → 100's of nodes

- Data redundancy and inconsistency

Name	Course	E-Mail	Grade
John Doe	CS112	jd@cs.brown.edu	B
C. Binnig	CS560		A
John Doe	CS560	John_doe@brown.edu	B

First Name	Last Name	Teaches	E-Mail	Grade
J.	Doe	CS112	jd@cs.brown.edu	B
Mike	Stonebraker	CS560	stonebraker@uni.edu	A
Carsten	Binnig		Carsten_binnig@brown.edu	B

Why is this a problem?

- Wasted space (?)
- Potential inconsistencies

(e.g., multiple formats, John Smith vs Smith J.)

WHY DATABASES

Why not store everything in flat files?

- Scalability → 100's of nodes

- Data redundancy and inconsistency

- Data retrieval

- Find the student who took CS18
- Find the student who took CS18 and has a GPA > 3.5

WHY DATABASES

Why not store everything in flat files?

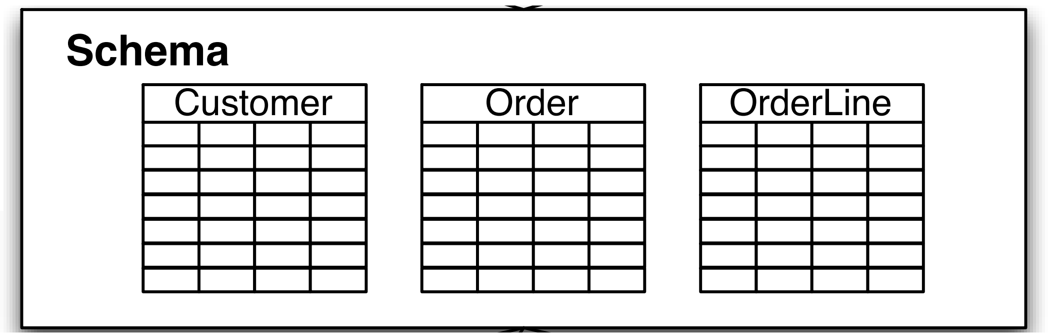
- Scalability → 100's of nodes

- Data redundancy and inconsistency

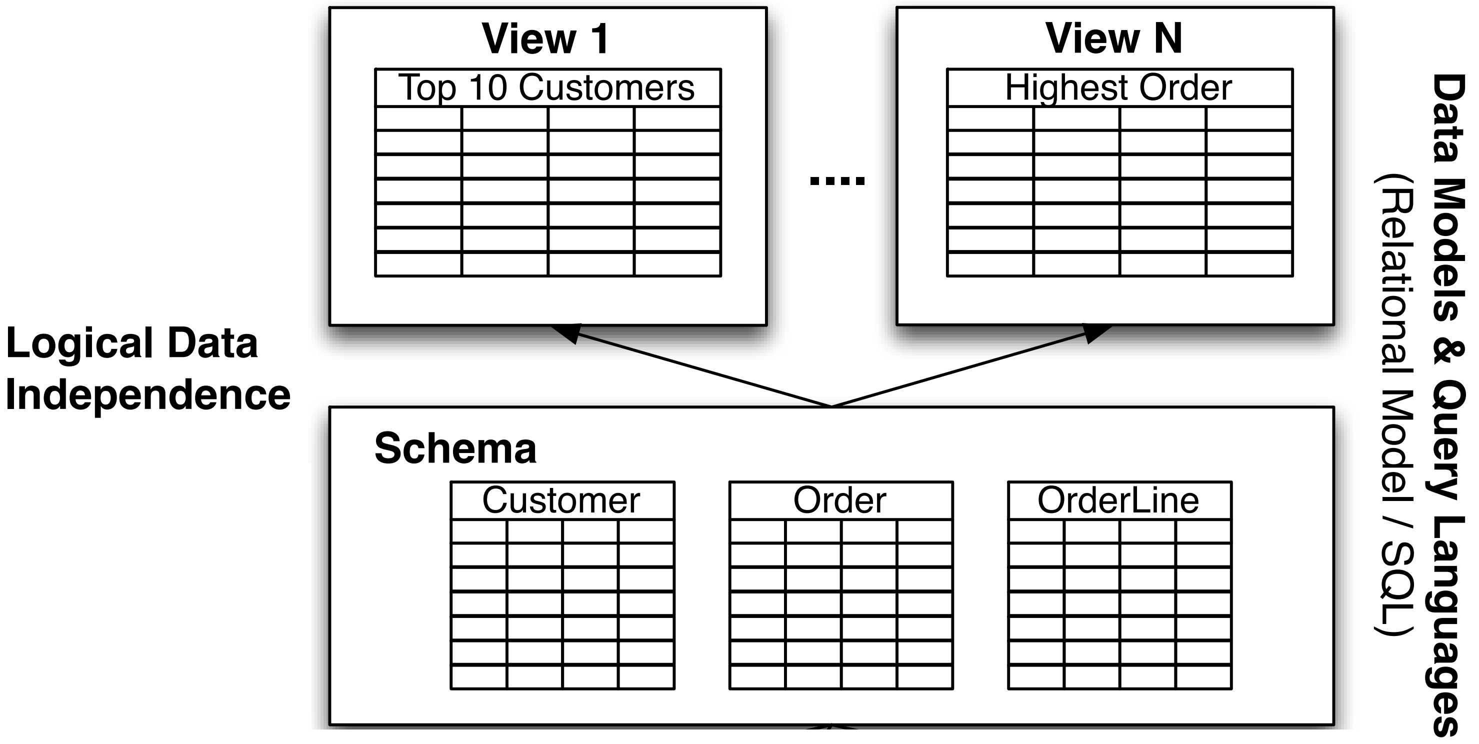
- Data retrieval

- Data-Independence

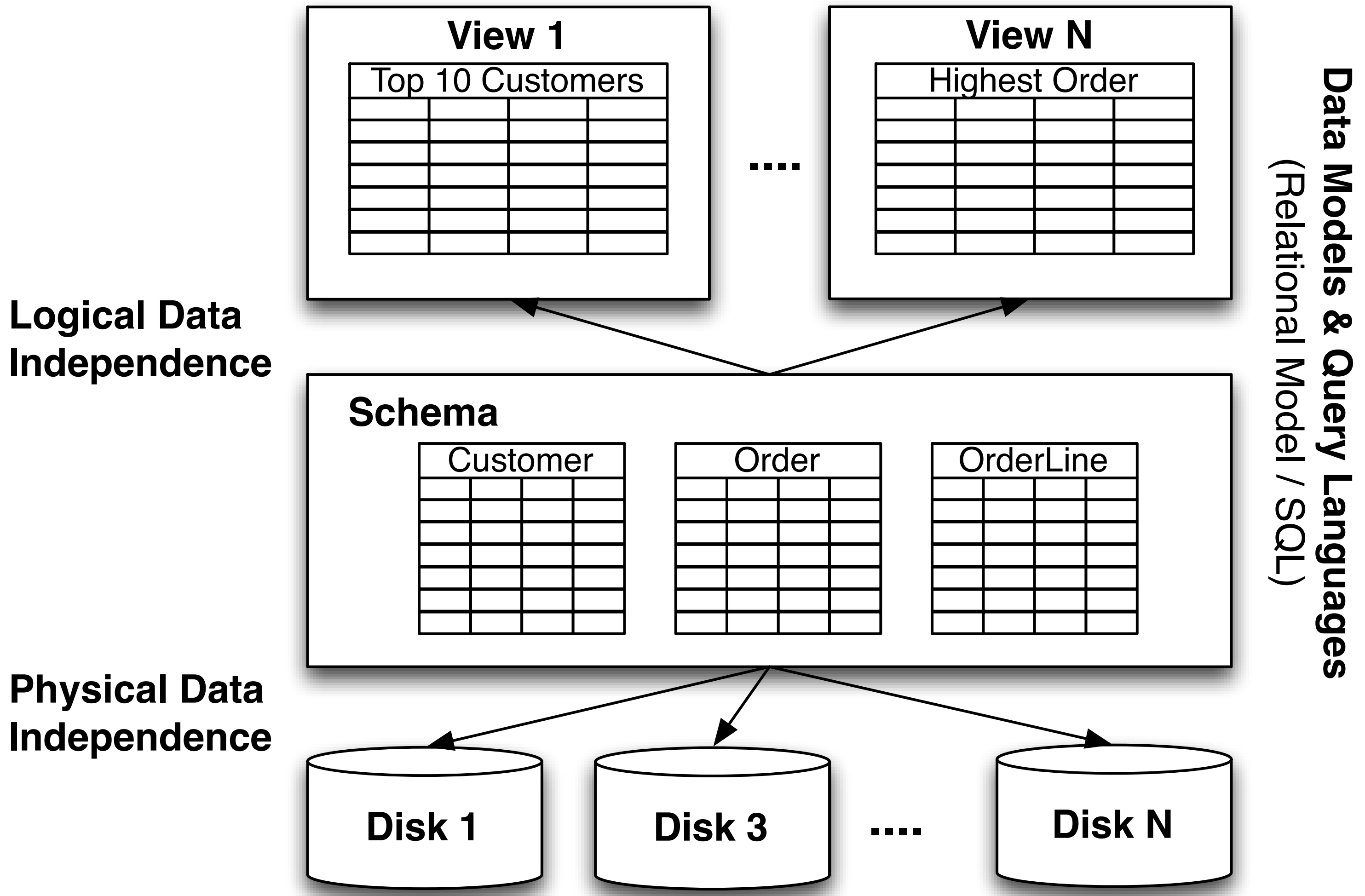
DATABASE OVERVIEW



DATABASE OVERVIEW



DATABASE OVERVIEW



WHY DATABASES

Why not store everything in flat files?

- Scalability → 100's of nodes

- Data redundancy and inconsistency

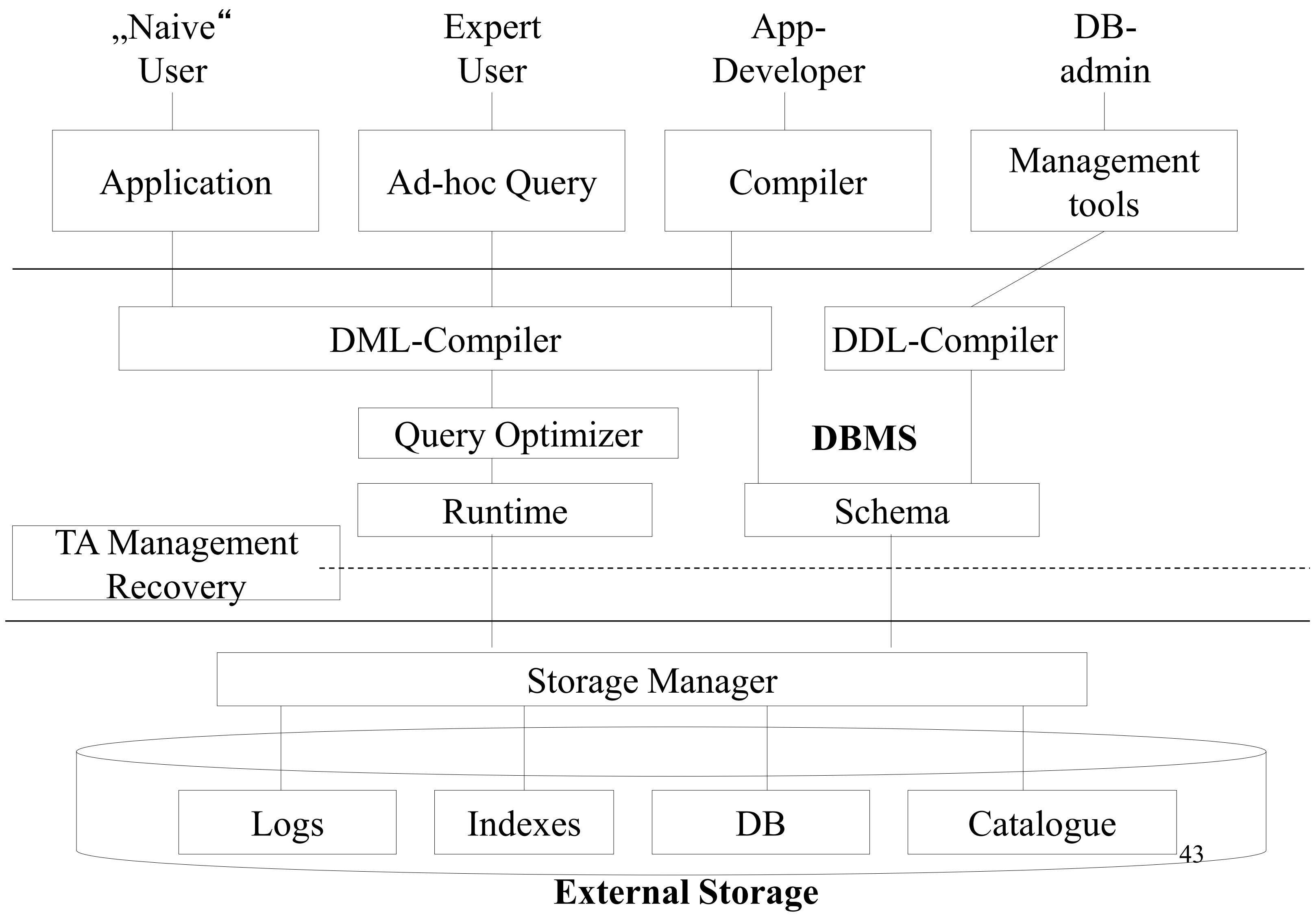
- Data retrieval

- Data-Independence

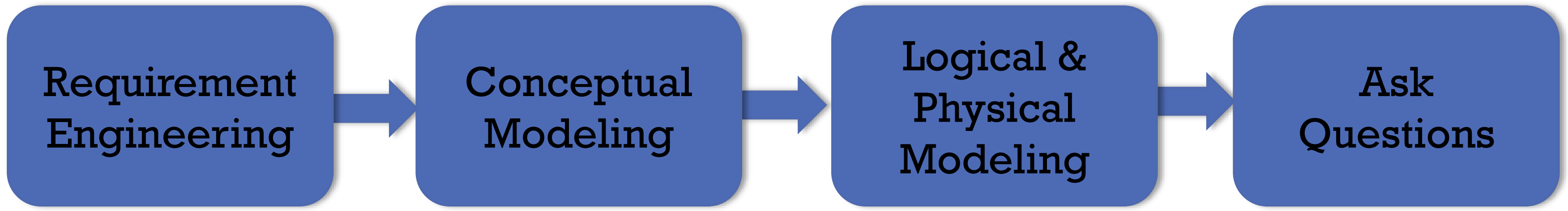
- Concurrent access

- Security and privacy

COMPONENTS OF A DATABASE SYSTEM



DATABASES FOR DATA SCIENTIST

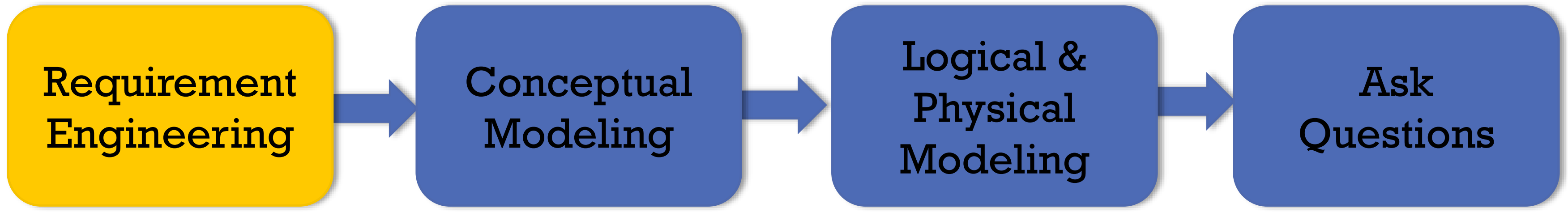


Book of duty

Conceptual Design (ER)

- Logical design (schema)
- Physical design (index, hints)

DATABASES FOR DATA SCIENTIST



Book of duty

Conceptual Design (ER)

- Logical design (schema)
- Physical design (index, hints)

BOOK OF DUTY

Describe information requirements

- **Objects used (e.g., student, professor, lecture)**
- **Domains of attributes of objects**
- **Identifiers, references / relationships**

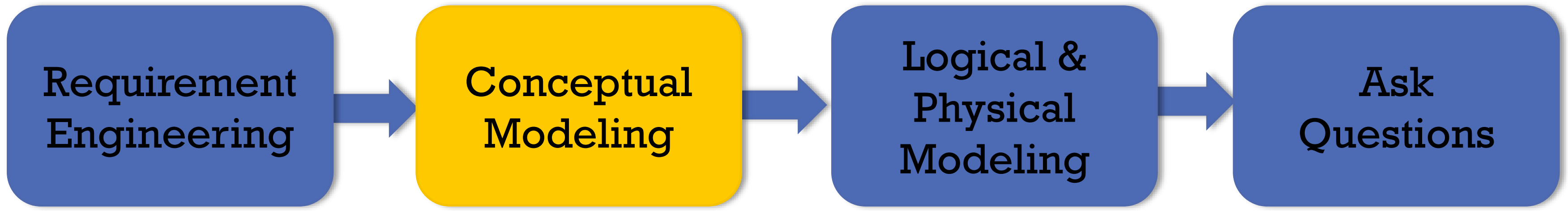
Describe processes

- E.g., examination, degree, register course

Describe processing requirements

- Cardinalities: how many students?
- Distributions: skew of lecture attendance
- Workload: how often a process is carried out
- Priorities and service level agreements

DATABASES FOR DATA SCIENTIST



Book of duty

Conceptual Design (ER)

- Logical design (schema)
- Physical design (index, hints)

ENTITY/RELATIONSHIP (ER) MODEL

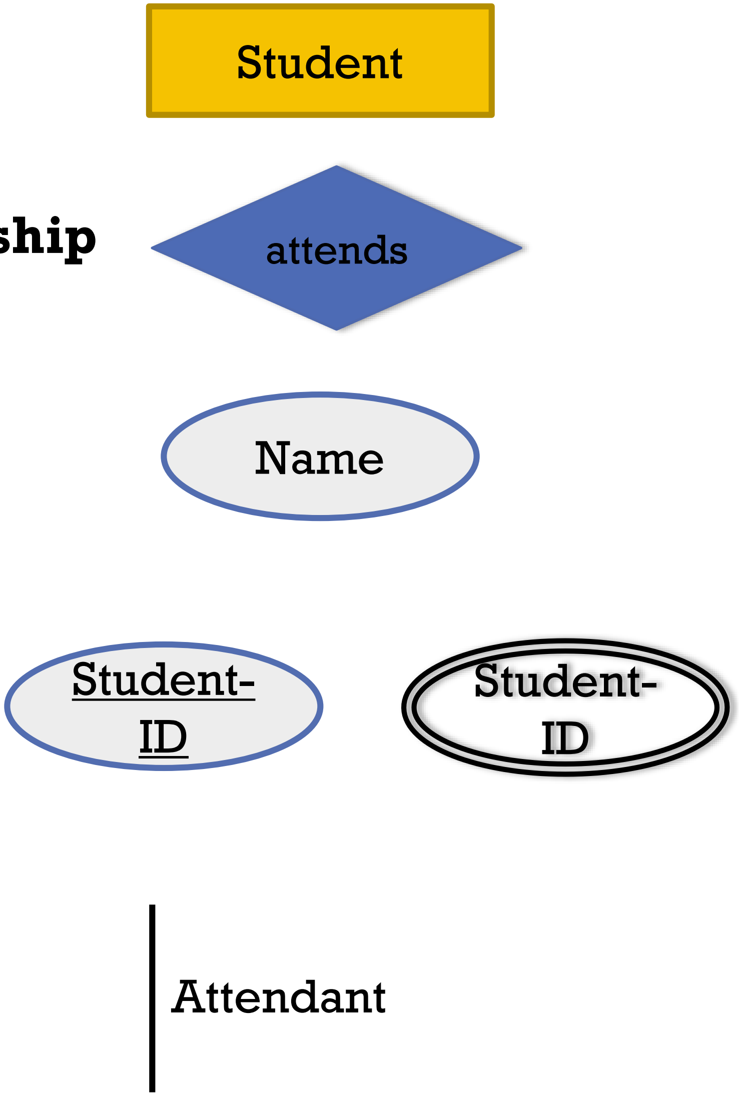
Entity

Relationship

Attribute

Key

Role



ENTITY/RELATIONSHIP (ER) MODEL

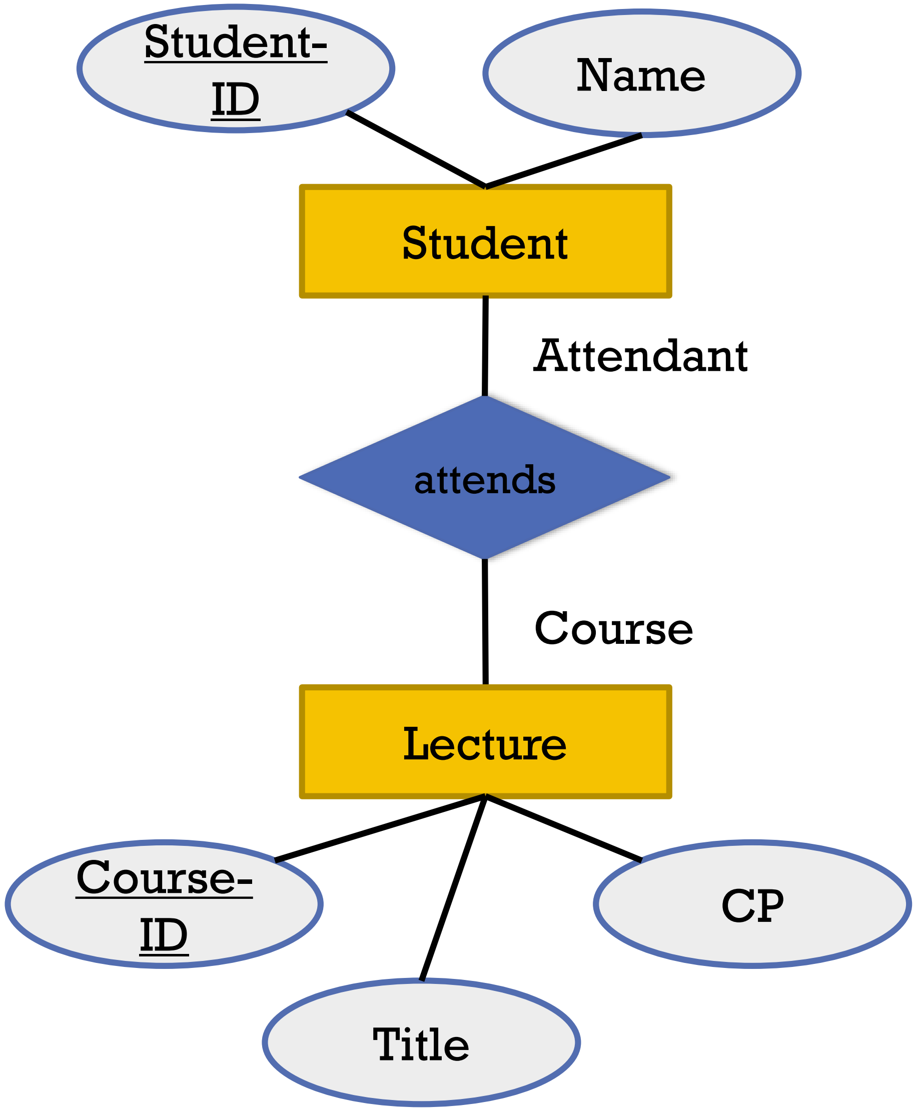
Entity

Relationship

Attribute

Key

Role



WHY ER

Advantages

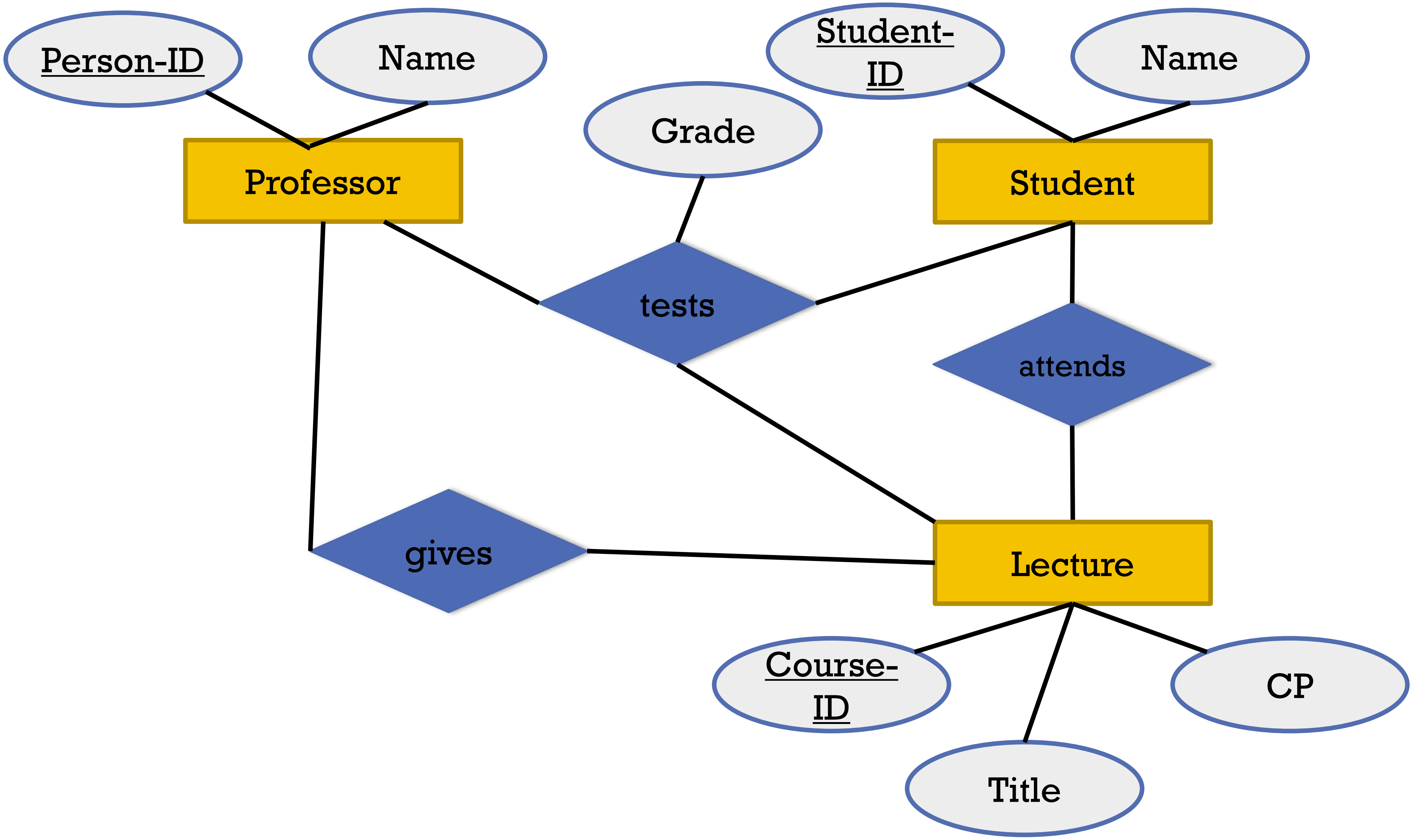
- ER diagrams are easy to create
- ER diagrams are easy to edit
- ER diagrams are easy to read (from the layman)
- ER diagrams express all information requirements

Other aspects

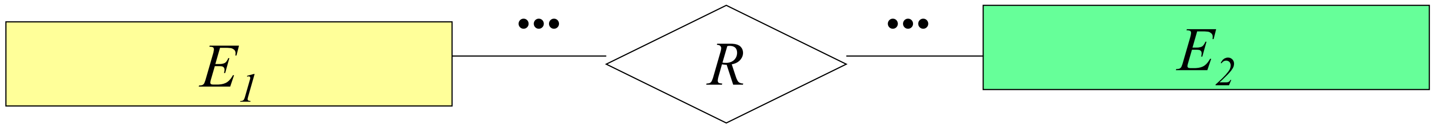
- Minimality
- Tools (e.g., Visio)
- Graphical representation

General

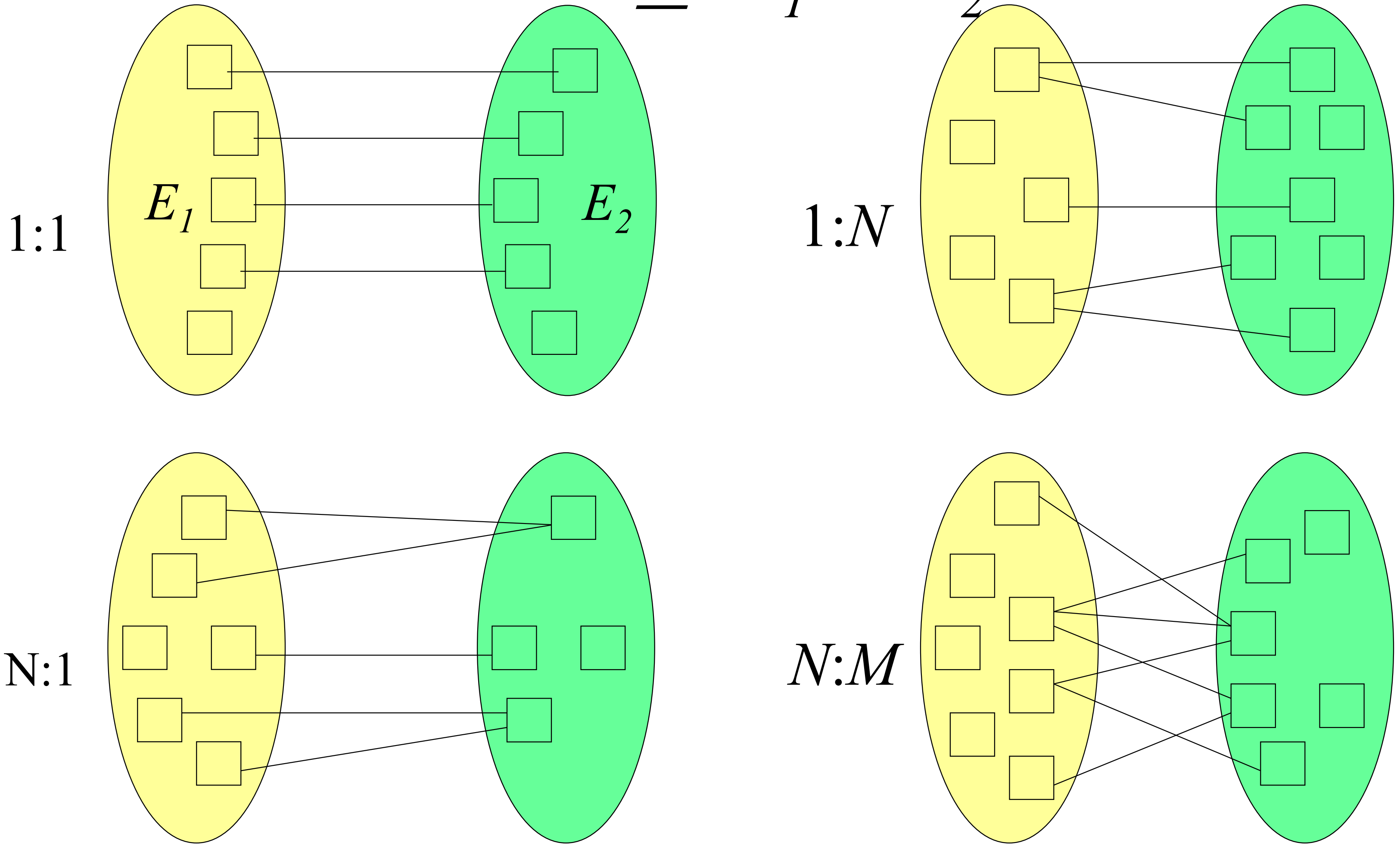
- Try to be concise, complete, comprehensible, and correct
- Controversy whether ER/UML is useful in practice
- No controversy that everybody needs to learn ER/UML


Person-ID
Name
Grade
Student-ID
Name
Professor
Student
tests
attends
gives
Lecture
Course-ID
Title
CP

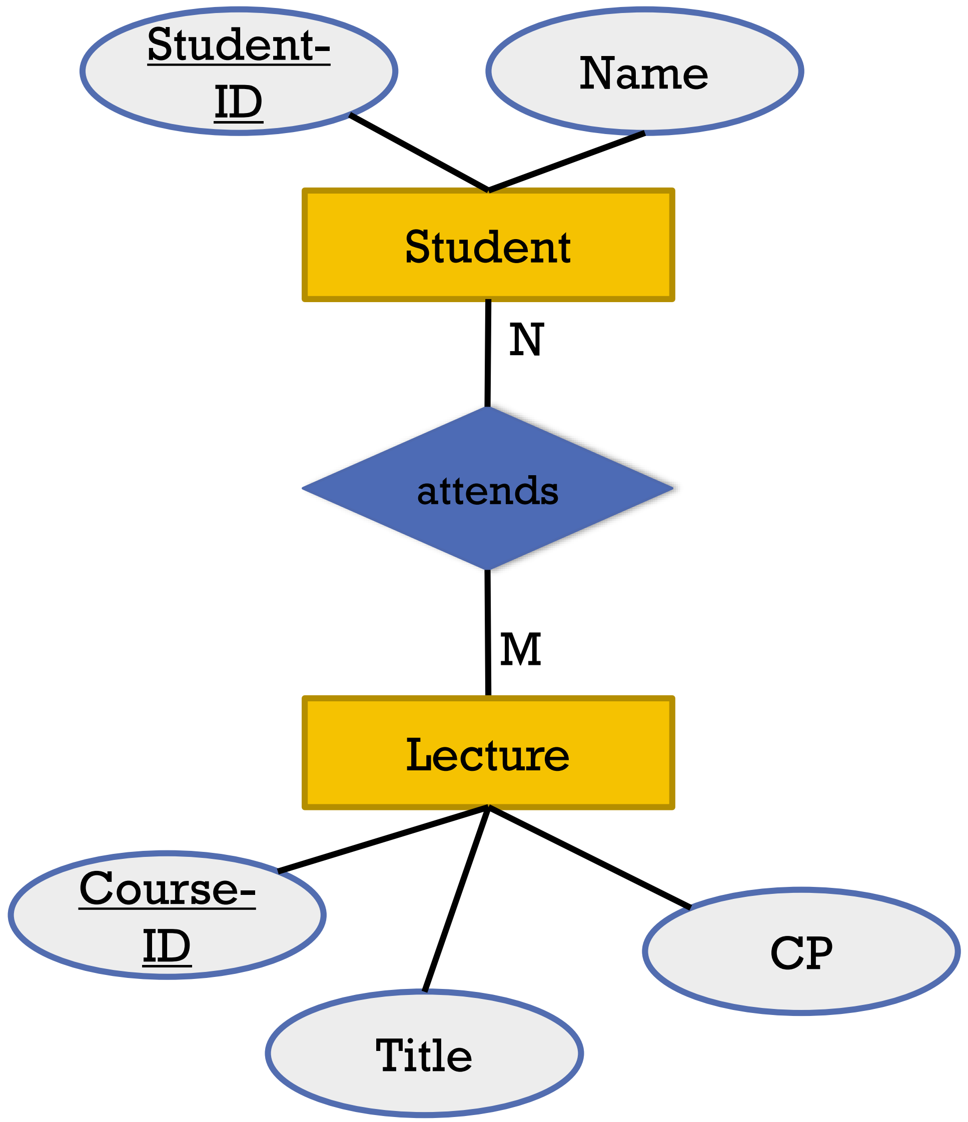
FUNCTIONALITIES



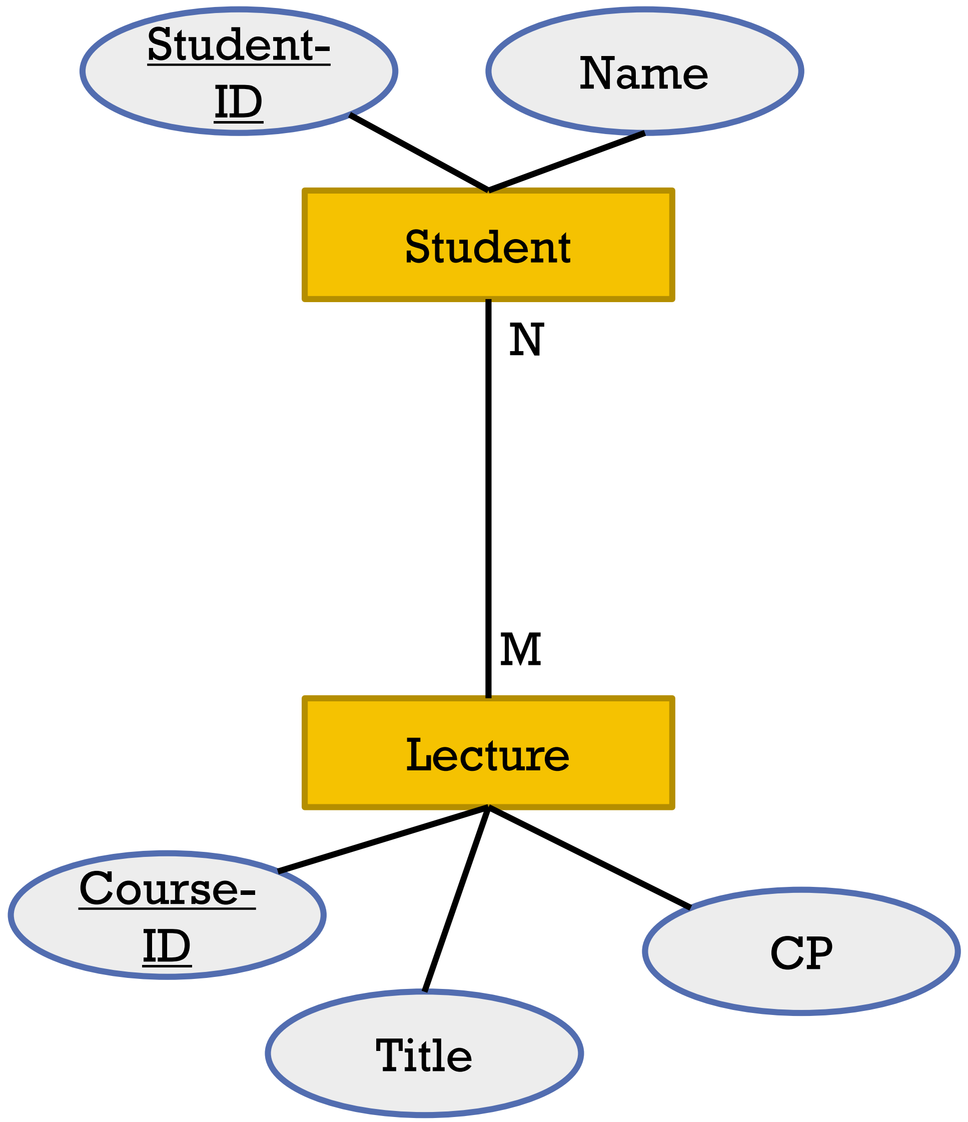
$$R \subseteq E_1 \times E_2$$



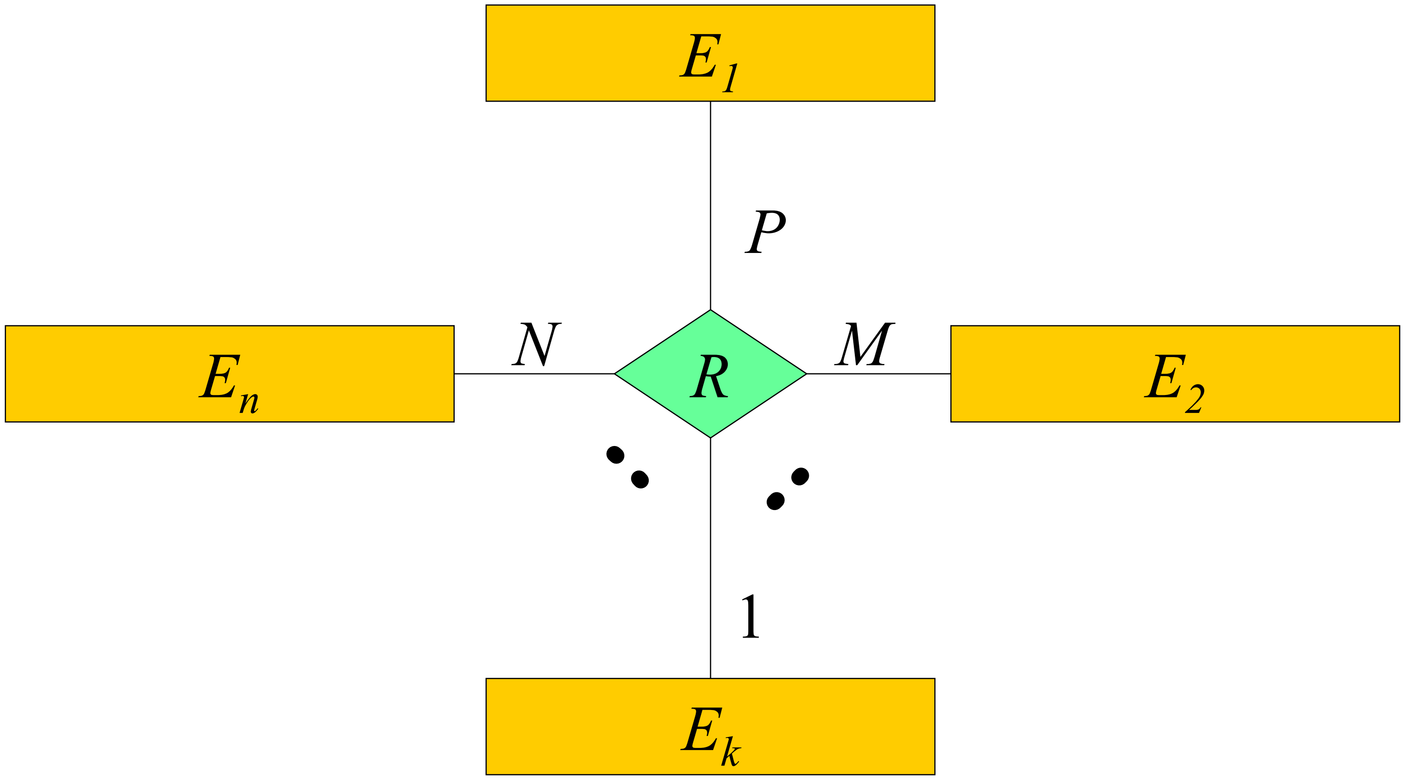
EXAMPLE: PROFESSOR <-> LECTURE



SOMETIMES ALSO SHOWN AS

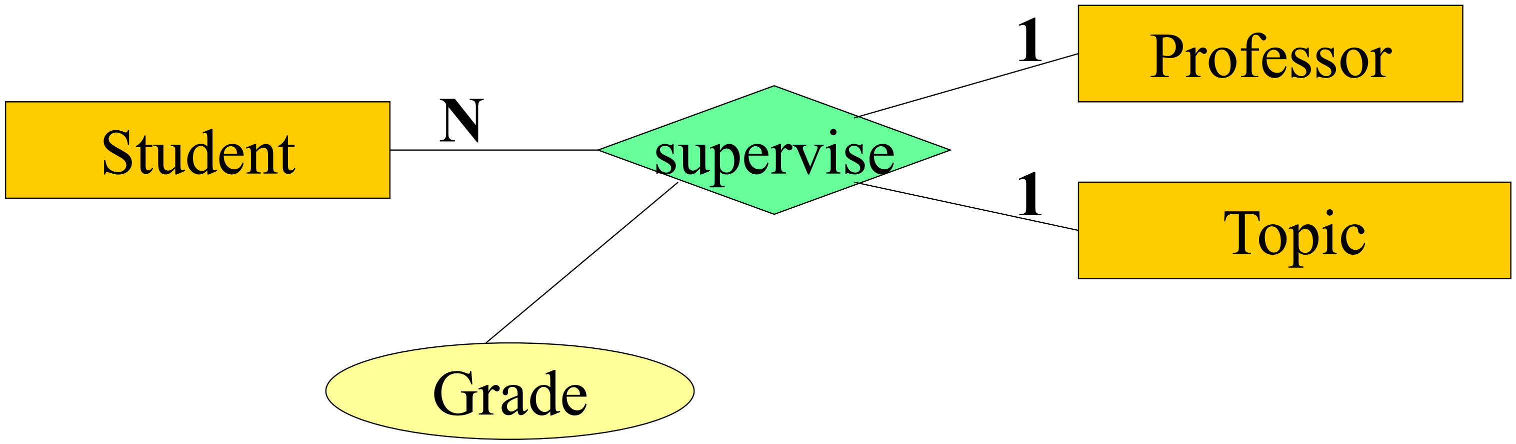


FUNCTIONALITIES OF N-ARY RELATIONSHIPS



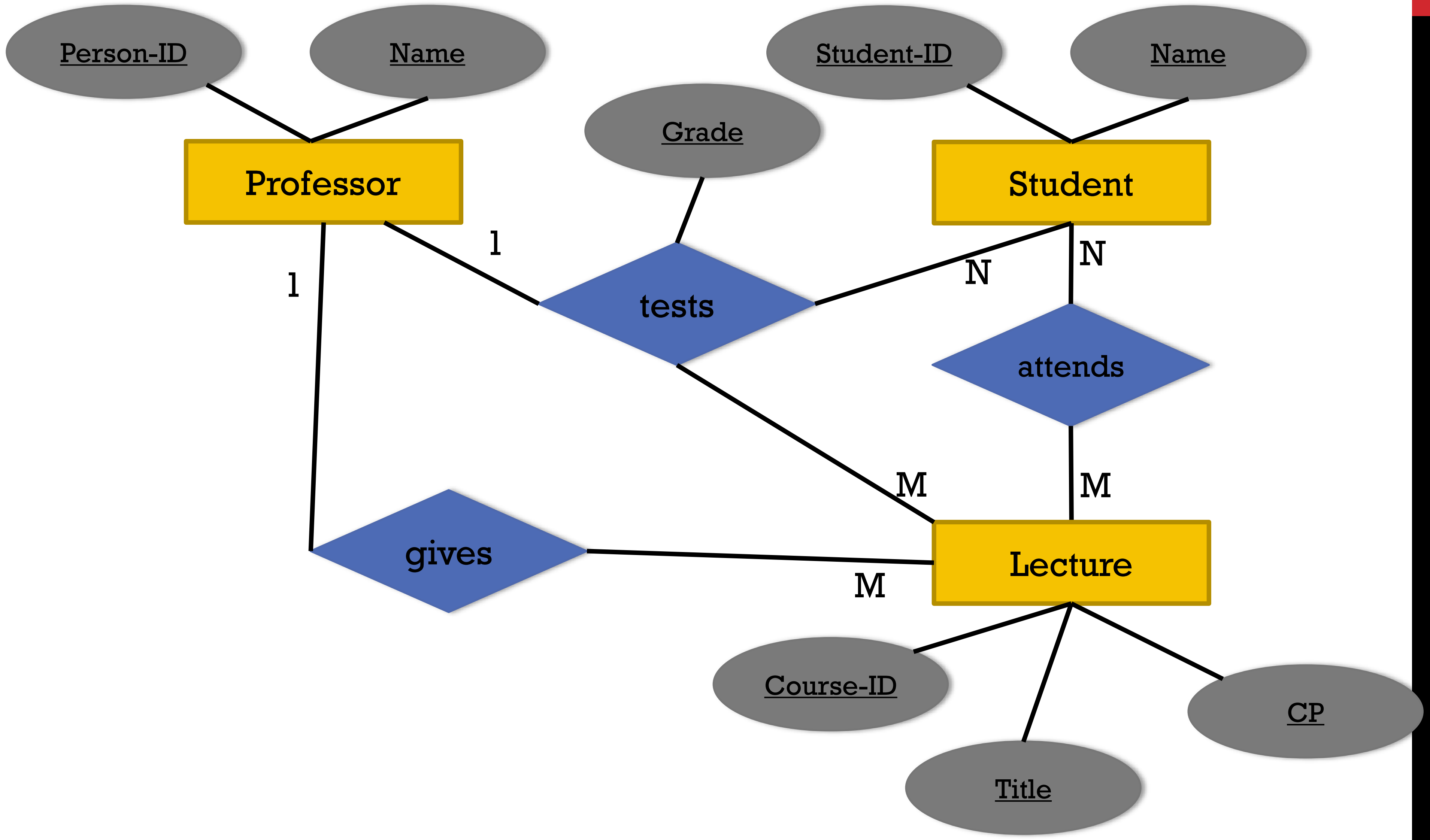
$$R : E_1 \times \ldots \times E_{k-1} \times E_{k+1} \times \ldots \times E_n \rightarrow E_k$$

EXAMPLE SEMINAR



supervise : Professor x Student → Topic

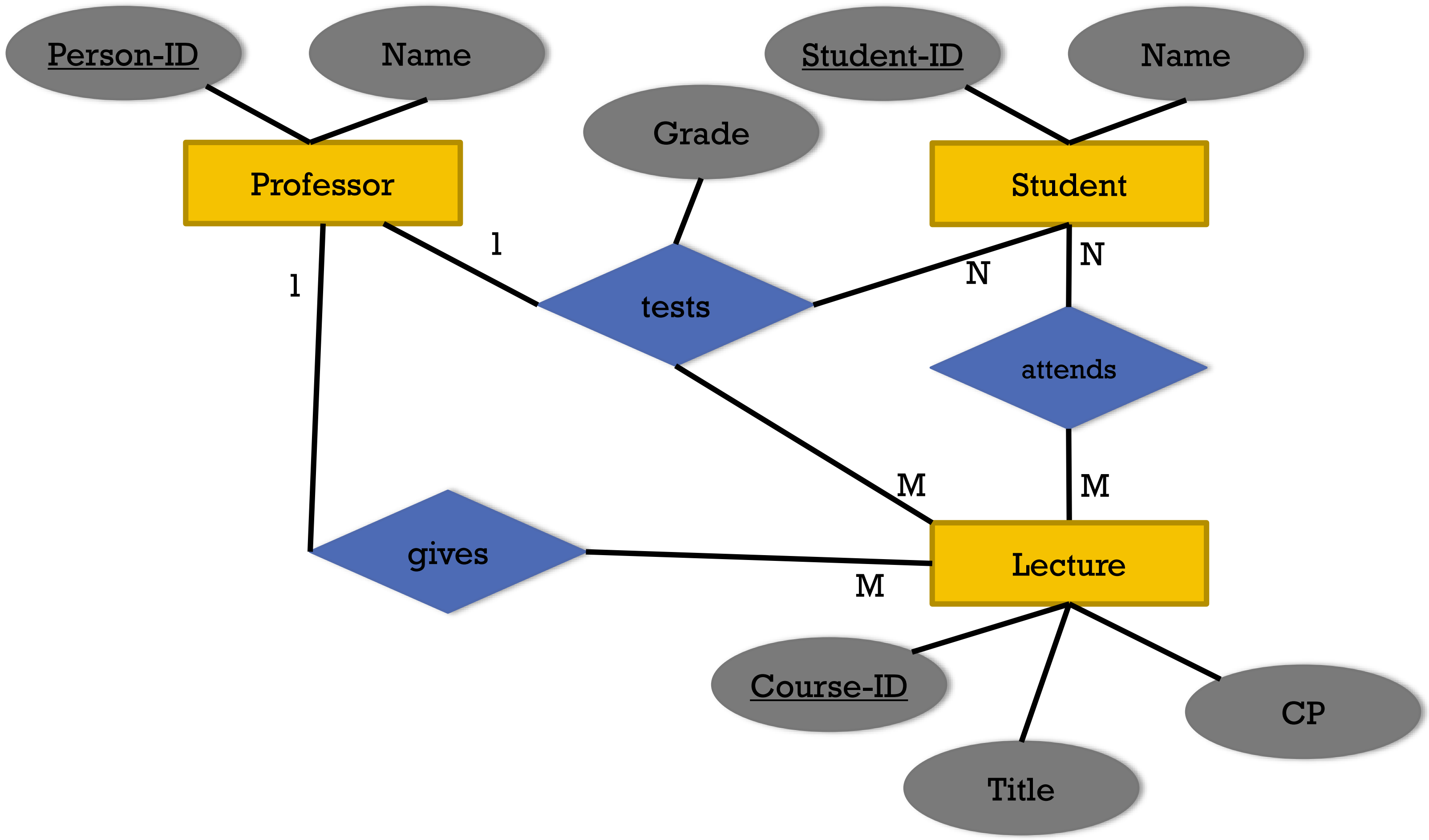
supervise : Topic x Student → Professor


Person-ID
Name
Student-ID
Name
Grade
Professor
Student
1
N
N
1
tests
attends
M
M
gives
M
Lecture
Course-ID
CP
Title

ATTRIBUTE VS ENTITY

Should the *grade* be an entity or attribute?

Should test be an entity or relationship?



RULES OF THUMB

Attribute vs. Entity

- Entity if the concept has more than one relationship
- Attribute if the concept has only one 1:1 relationship

Partitioning of ER Models

- Most realistic models are larger than a page
- Partition by domains (library, research, finances, …)

Good vs. Bad models

- Do not model redundancy or tricks to improve performance
- Less entities is better (the fewer, the better!)
- Remember the 5 C's (clear, concise, correct, complete, compliant)

NOT COVERED

(Min,Max) - Notation

Weak Entities

Generalization (i.e., inheritance)

Modeling limitations

Enhanced ERM

…

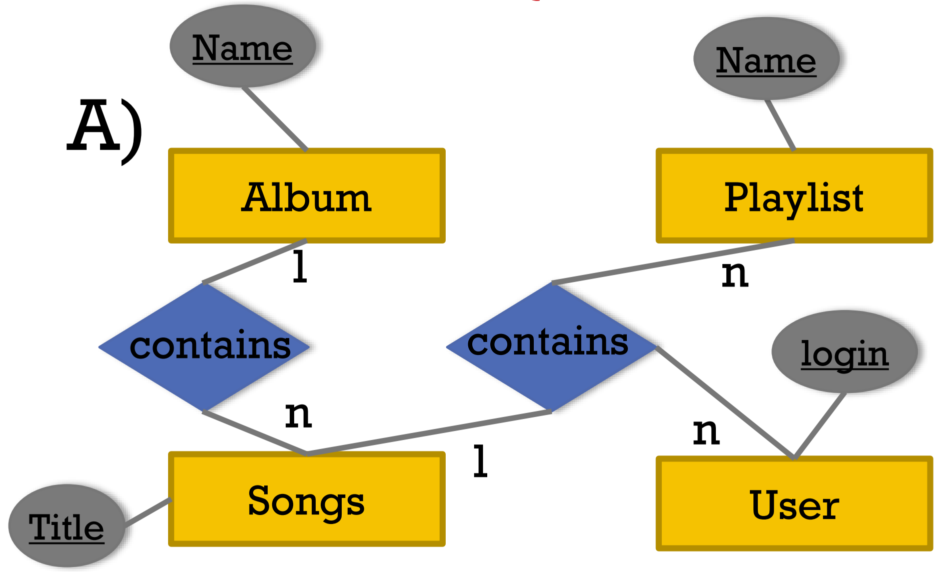
CLICKER QUESTION

Model a music record database

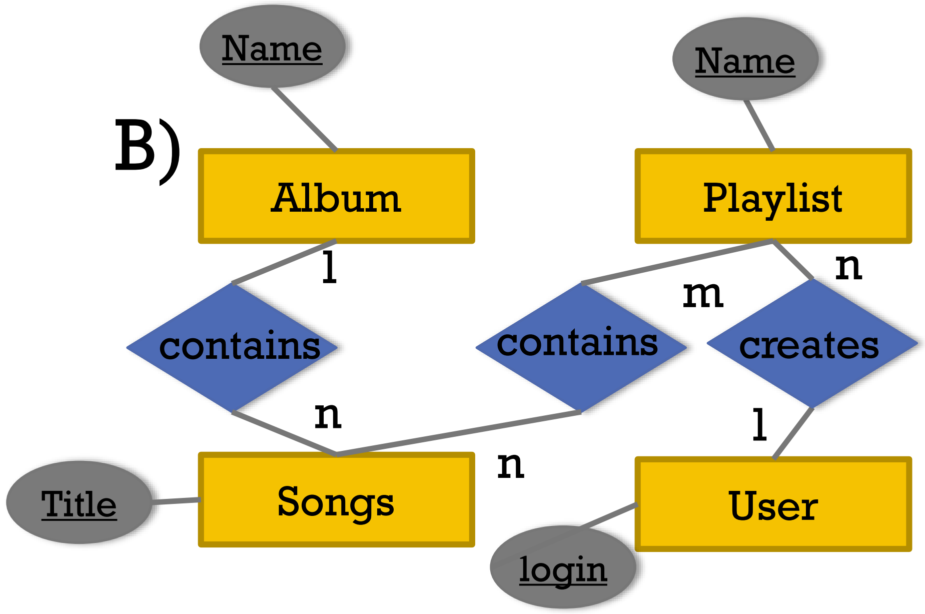
- An album has a unique name and songs have unique titles
- An album contains several songs
- A playlist has a unique name and is created by one user with a unique login
- A playlist contains several songs from potential different albums

CLICKER QUESTION

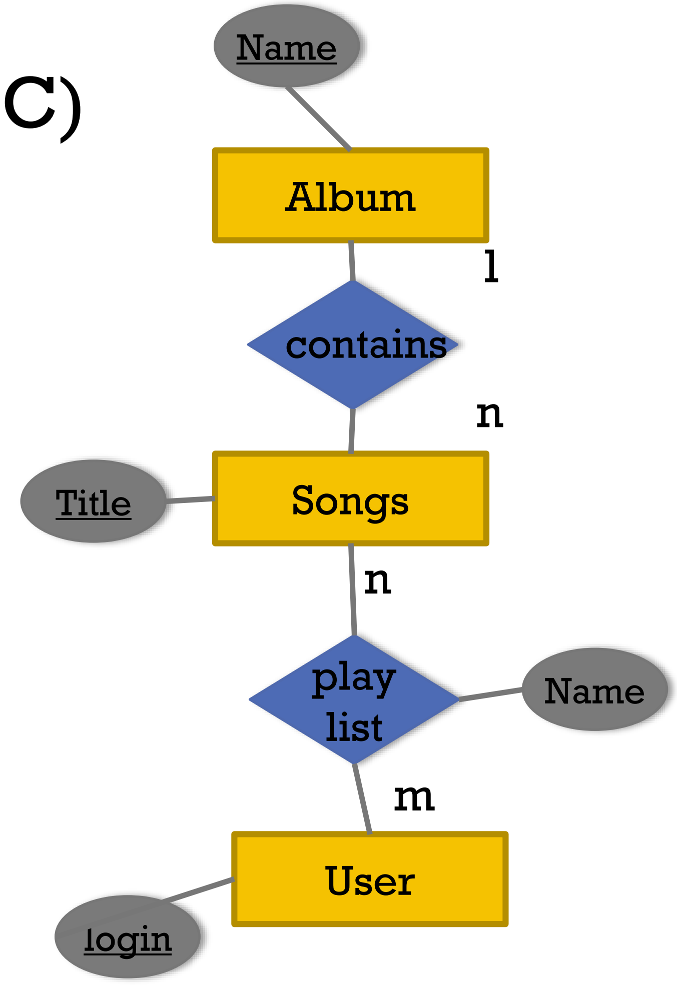
A)



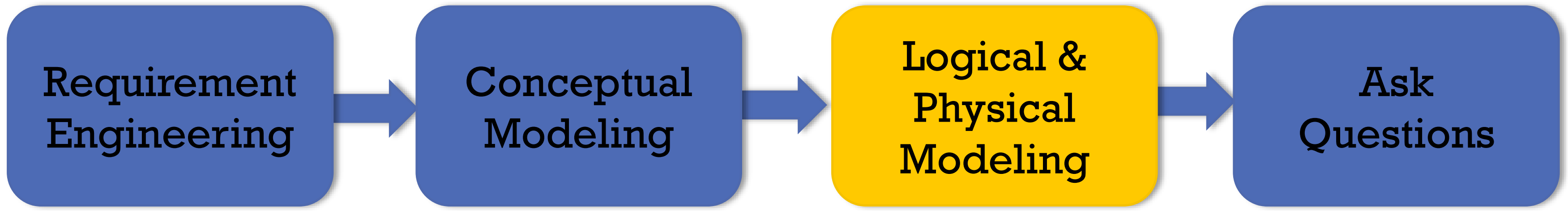
B)



C)



DATABASES FOR DATA SCIENTIST



Book of duty

Conceptual Design (ER)

- Logical design (schema)
- Physical design (index, hints)

RELATIONAL MODEL - TERMS

Account =

bname	acct_no	balance
Downtown	A-101	500
Brighton	A-201	900
Brighton	A-217	500

Table name

Attribute names

Terms

- Tables → Relations
- Columns → Attributes
- Rows → Tuples
- Schema (e.g.: Acct_Schema = (bname, acct_no, balance))

WHY ARE THEY CALLED RELATIONS?

Relation:

- $R \subseteq D_1 \times \ldots \times D_n$
- $D_1, D_2, \ldots, D_n$ are domains

Example: AddressBook $\subseteq$ string x string x integer

WHY ARE THEY CALLED RELATIONS?

Relation:

- $R \subseteq D_1 \times \ldots \times D_n$
- $D_1, D_2, \ldots, D_n$ are domains

Example: AddressBook $\subseteq$ string x string x integer

Tuple: $t \in R$

Example: t = („Mickey Mouse“, „Main Street“, 4711)

WHY ARE THEY CALLED RELATIONS?

Relation:

- $R \subseteq D_1 \times \ldots \times D_n$
- $D_1, D_2, \ldots, D_n$ are domains

Example: AddressBook $\subseteq$ string x string x integer

Tuple: $t \in R$

Example: t = („Mickey Mouse", „Main Street", 4711)

Schema: associates labels to domains

Example:

AddrBook: {[Name: string, Address: string, <u>Tel#:integer</u>]}

RELATIONS

bname	acct_no	balance
Downtown	A-101	500
Brighton	A-201	900
Brighton	A-217	500

Considered equivalent to…

{ (Downtown, A-101, 500),
(Brighton, A-201, 900),
(Brighton, A-217, 500) }

Relational database semantics are defined in terms of mathematical relations (i.e., sets)

KEYS AND RELATIONS

Kinds of keys

- Superkeys:
  set of attributes of table for which every row has distinct set of values
- Candidate keys:
  "minimal" superkeys
- Primary keys:
  DBA-chosen candidate key (marked in schema by underlining)

ISBN	Title	Author	Edition	Publisher	Price
0439708184	Harry Potter	J.K. Rowling	1	Scholastic	$6.70
0545663261	Mockingjay	Suzanne Collins	1	Scholastic	$7.39

KEYS AND RELATIONS

Kinds of keys

- Superkeys:
  set of attributes of table for which every row has distinct set of values
- Candidate keys:
  "minimal" superkeys
- Primary keys:
  DBA-chosen candidate key (marked in schema by underlining)

Act as Integrity Constraints

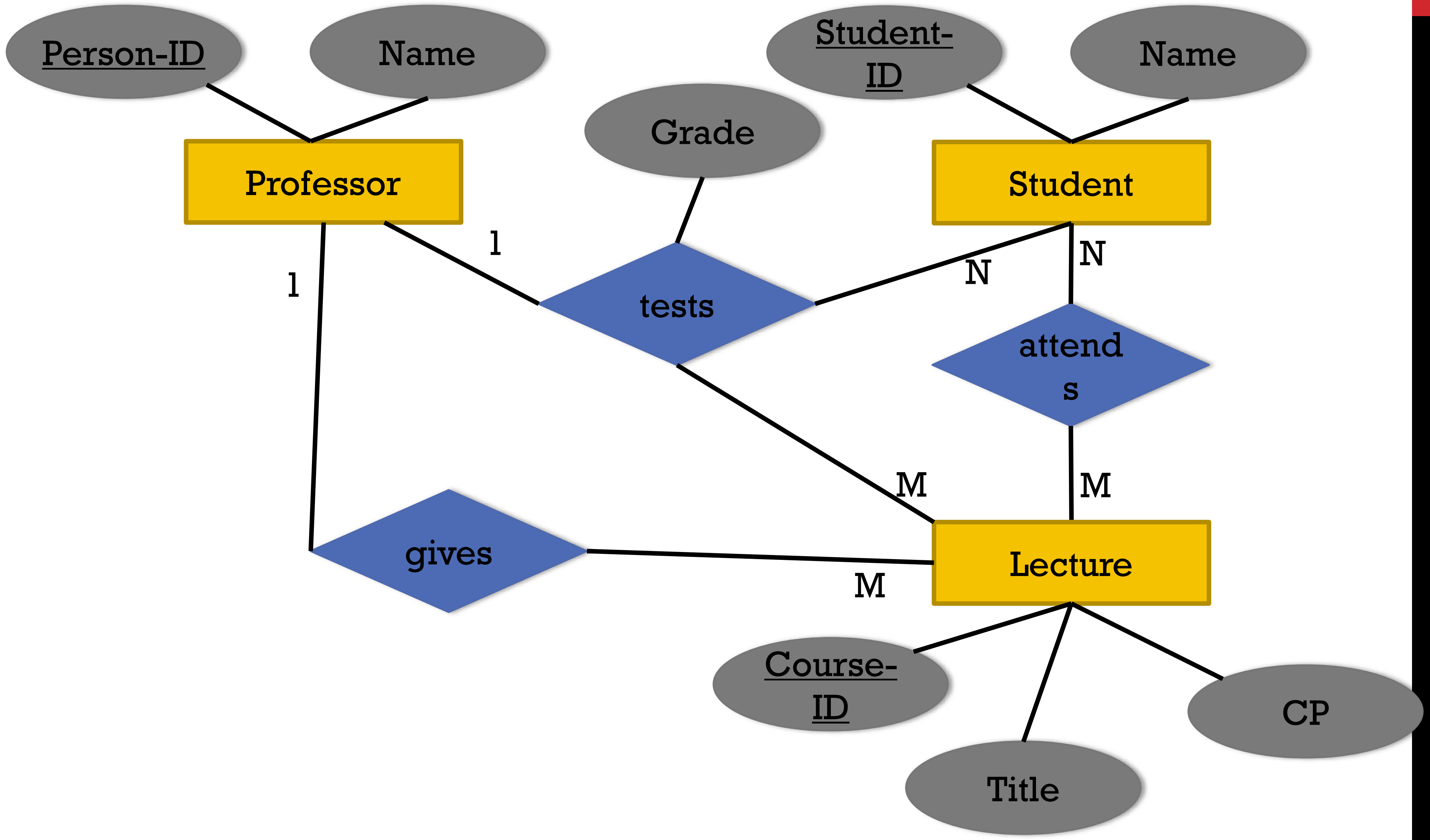
i.e., guard against illegal/invalid instance of given schema

e.g., Branch = (<u>bname</u>, bcity, assets) Þ

bname	bcity	assets
Brighton	Brooklyn	5M
Brighton	Boston	3M

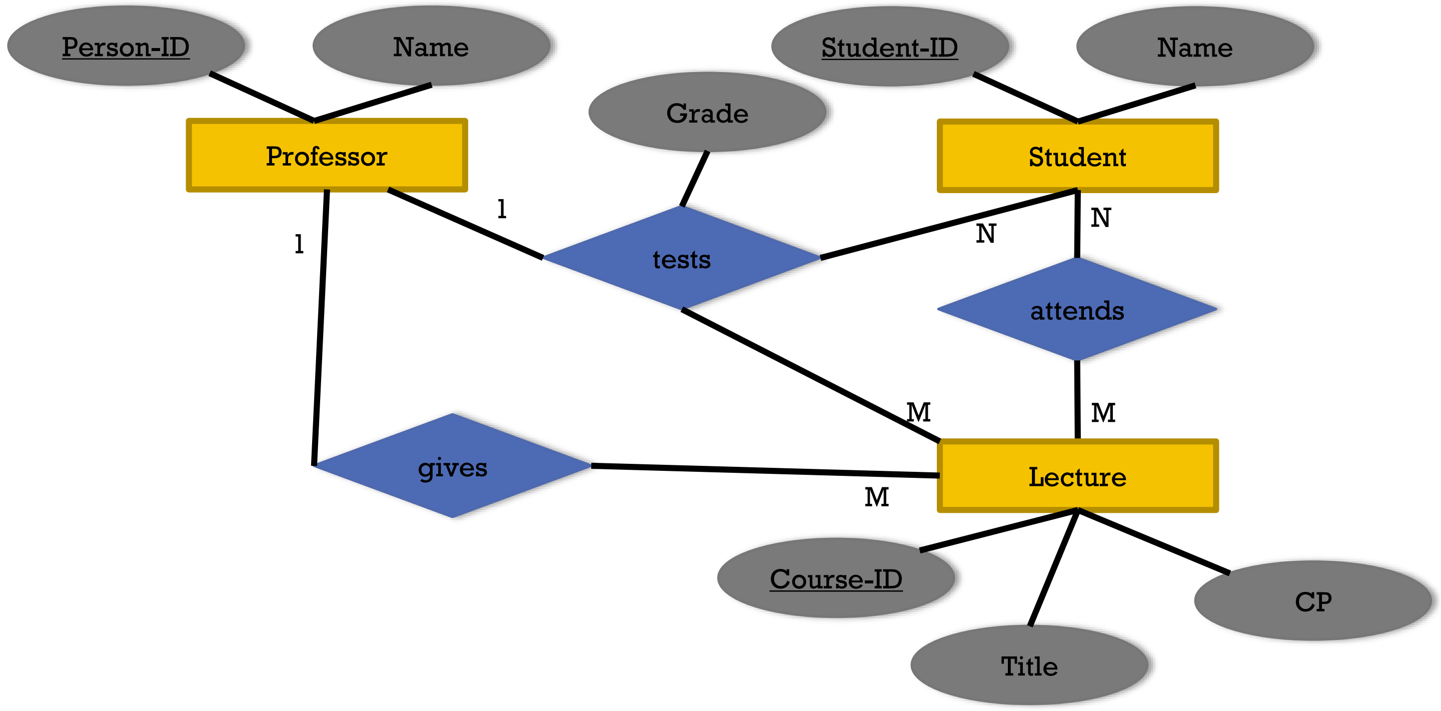
$\longleftarrow$ *Invalid!!*

HOW TO TRANSLATE ERM TO RELATIONS

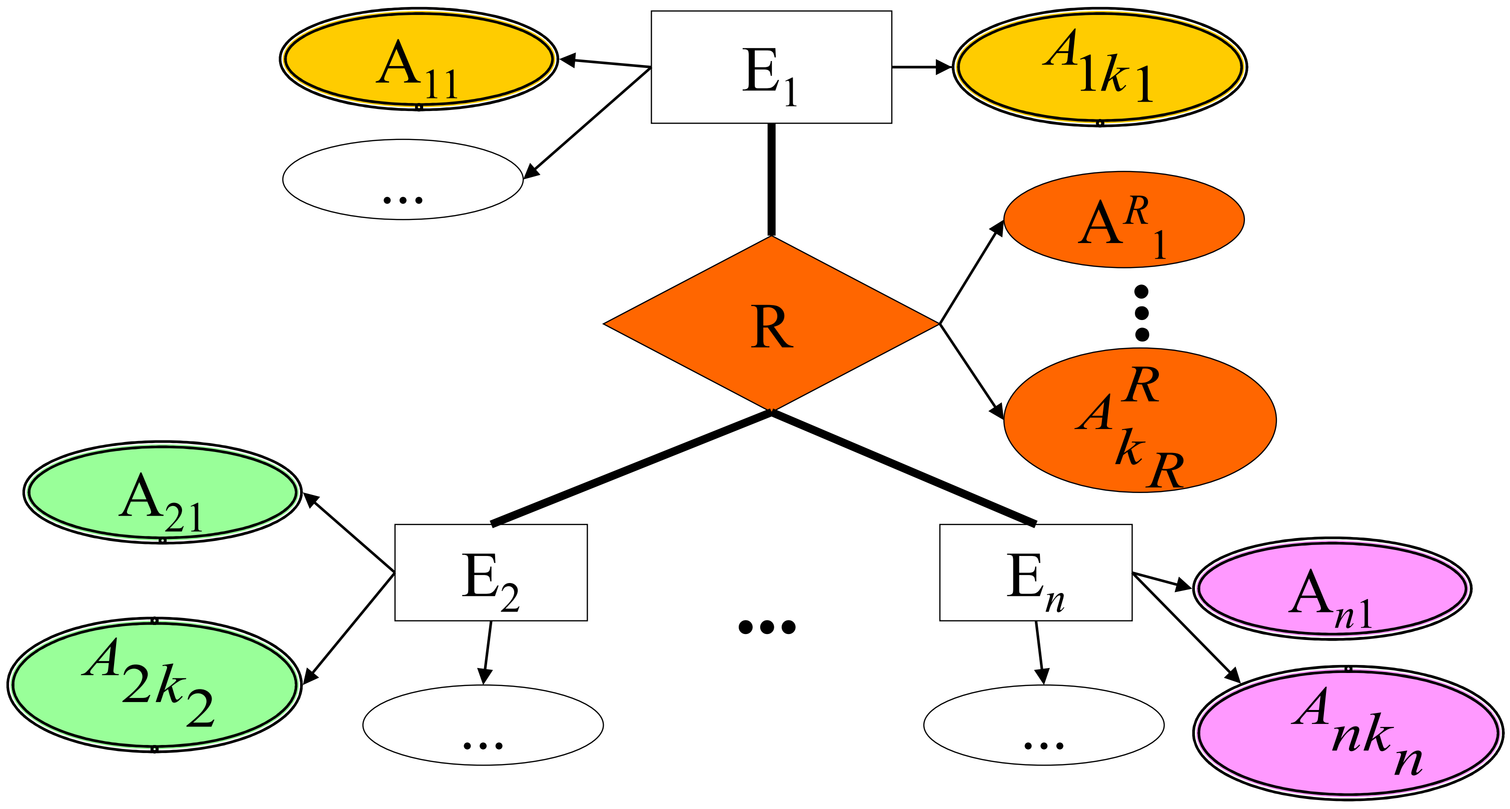


RULE #1: ENTITIES

Professor(<u>Person-ID:integer</u>, Name:string)
Student(<u>Student-ID:integer</u>, Name:string)
Lecture(<u>Course-ID:string</u>, Title:string, CP:float)



RULE #2: RELATIONSHIPS

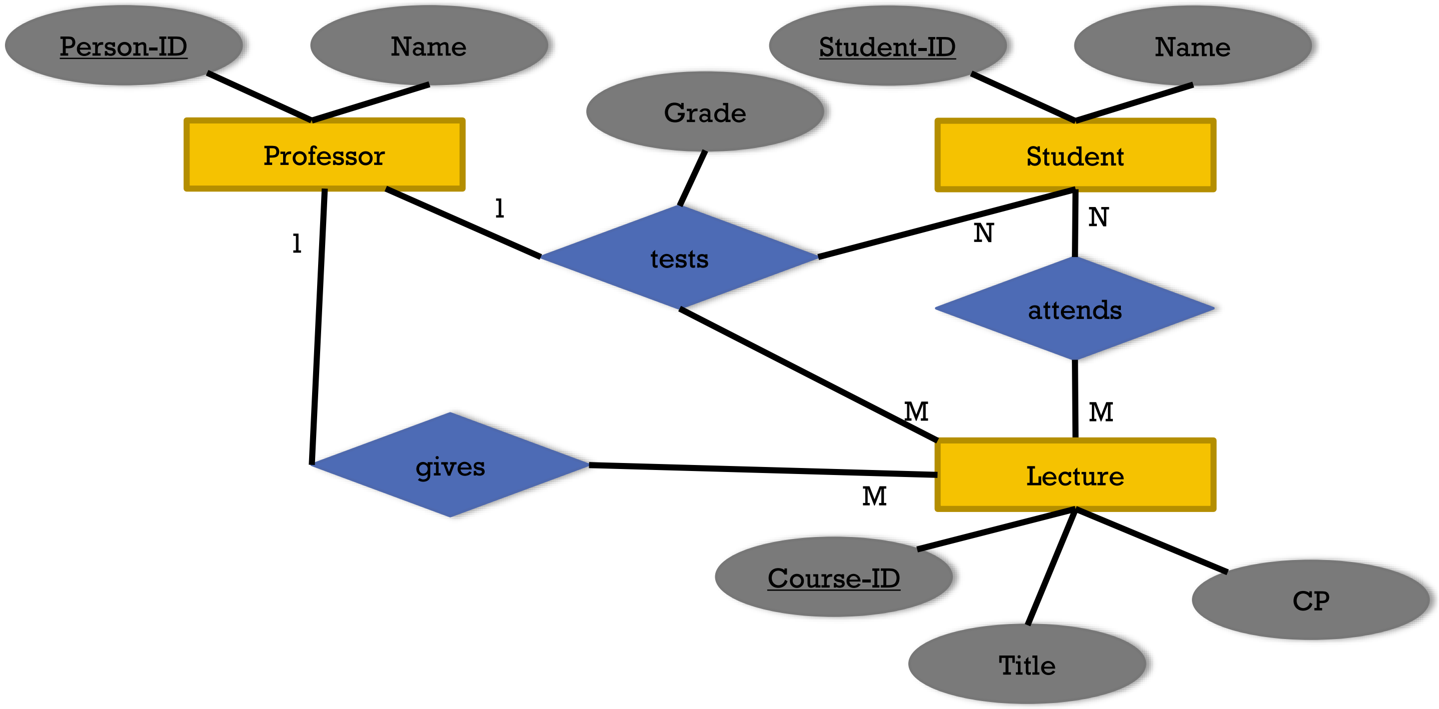


$$R: \{[\underbrace{A_{11},\ldots,A_{1k_1}}_{\text{Key of } E_1}, \underbrace{A_{21},\ldots,A_{2k_2}}_{\text{Key of } E_2}, \ldots, \underbrace{A_{n1},\ldots,A_{nk_n}}_{\text{Key of } E_n}, \underbrace{A^R_1,\ldots,A^R_{k_R}}_{\text{Attributes of R}}]\}$$

RULE #2: RELATIONSHIPS

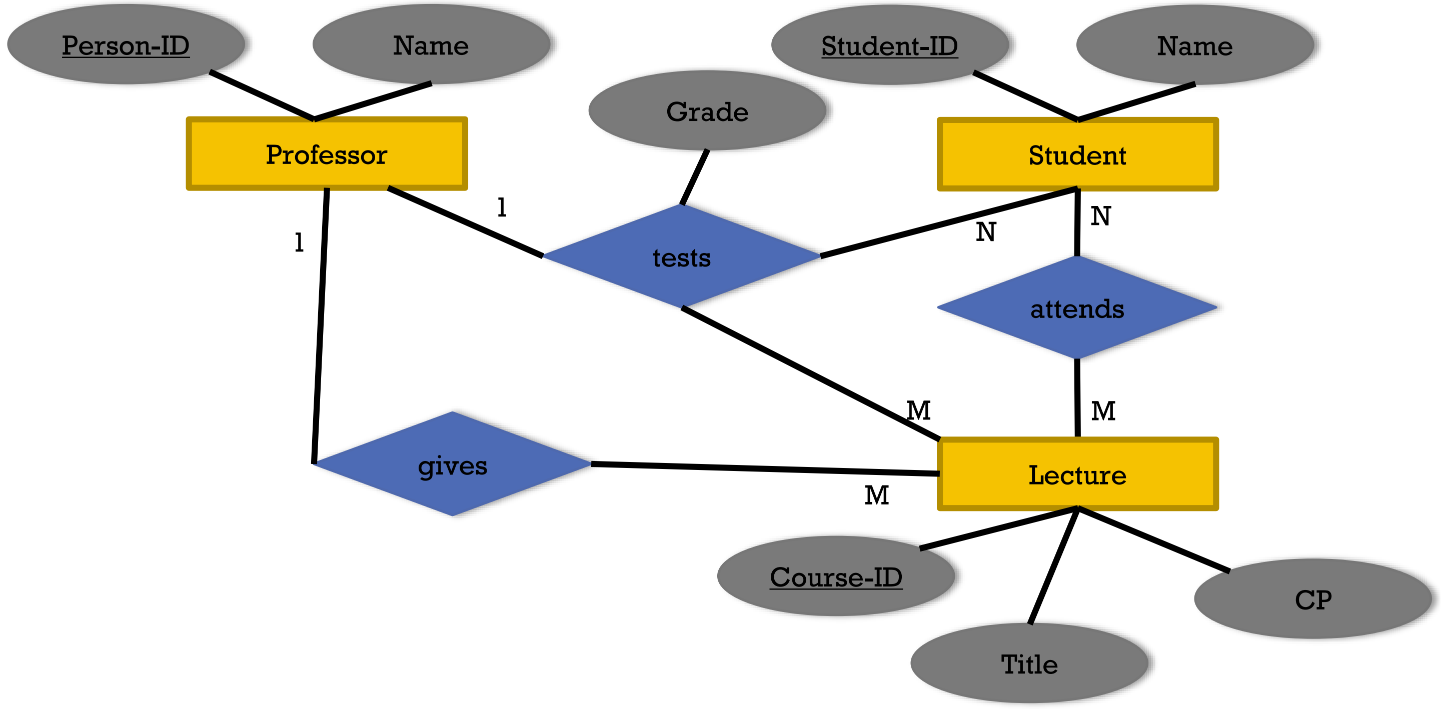
Professor(<u>Person-ID:integer</u>, Name:string)
Student(<u>Student-ID:integer</u>, Name:string)
Lecture(<u>Course-ID:string</u>, Title:string, CP:float)
Gives(Person-ID:integer, Course-ID:string)
Attends(Student-ID:integer, Course-ID:string)
Tests(Student-ID:integer, Course-ID:string, Person-ID:integer, Grade:String)

What about keys?



RULE #2: RELATIONSHIPS

Professor(Person-ID:integer, Name:string)
Student(Student-ID:integer, Name:string)
Lecture(Course-ID:string, Title:string, CP:float)
Gives(Person-ID:integer, Course-ID:string)
Attends(Student-ID:integer, Course-ID:string)
Tests(Student-ID:integer, Course-ID:string, Person-ID:integer, ,
Grade:string)

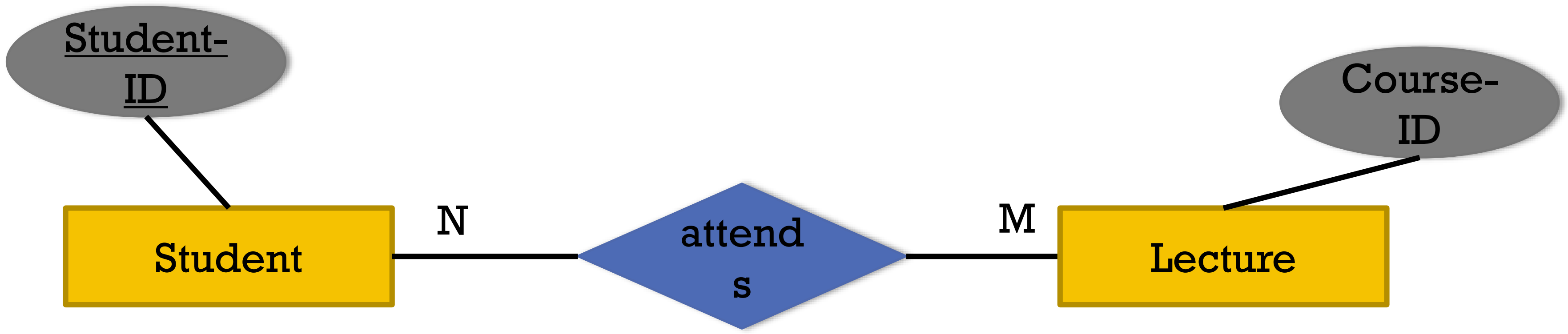


INSTANCE OF ATTENDS

Student	
Student-ID	…
1	…
2	…
4	…
5	…
6	…
10	…

Attends	
Student-ID	Course-ID
1	CS1951a
1	CS167
2	CS1951a
2	CS167
3	CS18
…	…

Lecture	
Course-ID	…
CS1951a	…
CS195w	…
CS18	…
CS17	…
CS142	…
CS167	…



RULE #3: MERGE RELATIONS WITH THE SAME KEY

Professor(Person-ID:integer, Name:string)
Lecture(**Course-ID:string**, Title:string, CP:float)
Gives(Person-ID:integer, **Course-ID:string**)

Professor(Person-ID:integer, Name:string)
Lecture(Course-ID:string, Title:string, CP:float, Person-ID:integer)

FINAL

Professor(Person-ID:integer, Name:string)

Student(Student-ID:integer, Name:string)

Lecture(Course-ID:string, Title:string, CP:float,
    Person-ID:integer)

Attends(Student-ID:integer, Course-ID:string)

Tests(Student-ID:integer, Course-ID:string,
    Person-ID:integer, Grade:string)

Why didn't we merge **Attends** and **Tests**?

SQL: CREATE TABLE

Professor(Person-ID:integer, Name:string)
Lecture(Course-ID:string, Title:string, CP:float, Person-ID:integer)
Student(Student-ID:integer, Name:string)

```
CREATE TABLE Student (
  Student-ID INT,
  Name VARCHAR(45));

CREATE TABLE Professor (
  Person-ID INT,
  Name VARCHAR(45));
```

```
CREATE TABLE Lecture(
    Course-ID INT,
    Title VARCHAR(45),
    CP REAL,
    Person-ID INT);
```

SQL: CREATE TABLE

Professor(Person-ID:integer, Name:string)
Lecture(Course-ID:string, Title:string, CP:float, Person-ID:integer)
Student(Student-ID:integer, Name:string)

```
CREATE TABLE Student (
  Student-ID INT,
  Name VARCHAR(45));

CREATE TABLE Professor (
  Person-ID INT,
  Name VARCHAR(45));
```

```
CREATE TABLE Lecture(
    Course-ID INT,
    Title VARCHAR(45),
    CP REAL,
    Person-ID INT);
```

INTEGRITY CONSTRAINTS IN CREATE TABLE

not null

primary key (*A_1*, ..., *A_n*)

Example: Declare *ID* as the primary key for *instructor*

```
CREATE TABLE Attends (
    Student-ID INT,
    Course-ID VARCHAR(6),
    PRIMARY KEY (Student-ID, Course-ID));
```

primary key declaration on an attribute automatically ensures **not null**

SINGLE ATTRIBUTE KEY

```
CREATE TABLE course (
        course_id  VARCHAR(8) PRIMARY KEY,
        title VARCHAR(50),
        cp NUMERIC(1,1));
```

Primary key declaration can be combined with attribute declaration as shown above

FOREIGN KEYS

```
CREATE TABLE `Attends` (
 `Student_ID` INT NOT NULL,
 `Course-ID` VARCHAR(6) NOT NULL,
 PRIMARY KEY (`Student_ID`, `Course-ID`),
 CONSTRAINT `fk_attend_Student`
  FOREIGN KEY (`Student_ID`)
  REFERENCES `Student` (`ID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
 CONSTRAINT `fk_attend_lecture`
  FOREIGN KEY (`Lecture_Course-ID`)
  REFERENCES `Lecture` (`Course-ID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;
```

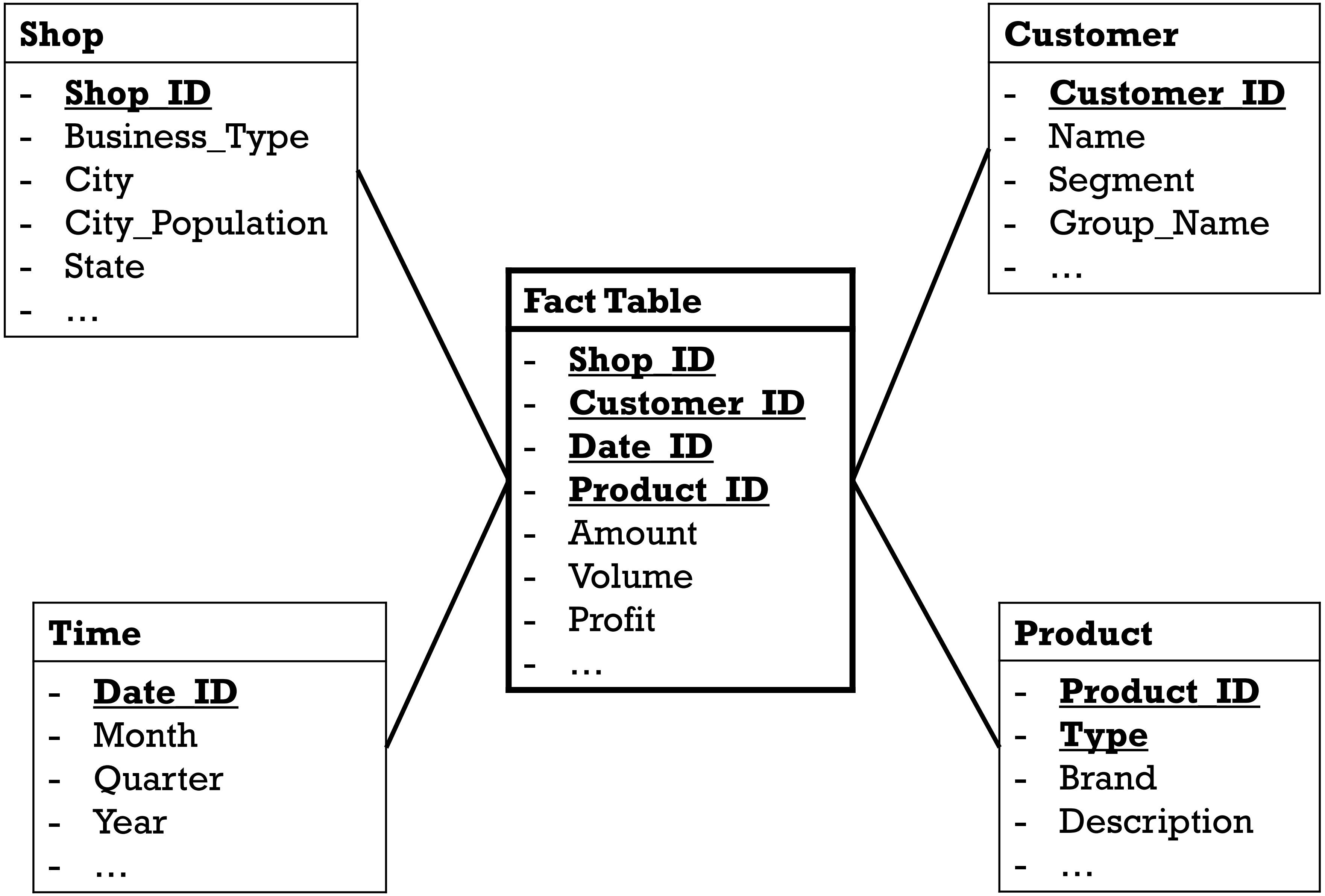
INDEX

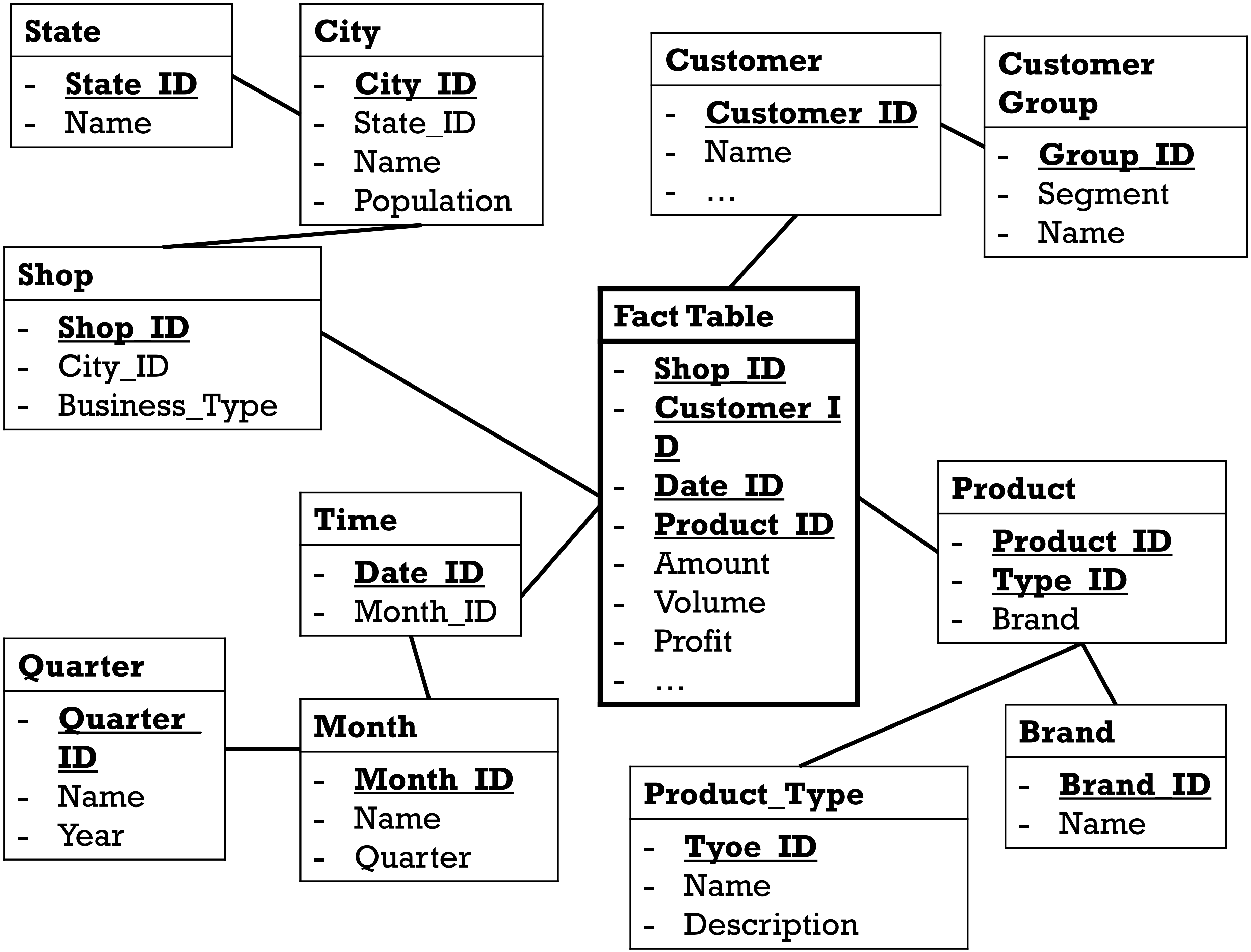
```
CREATE INDEX `fk_Student_has_Lecture_Lecture1_idx` ON `Attends`
(`Course-ID` ASC);

CREATE INDEX `fk_Student_has_Lecture_Student_idx` ON  `Attends`
(`Student-ID` ASC);
```

PROBLEM

- **You are the new Data Scientist at Evil Market**
- Evil Market is tracking all customer purchases with their membership card or credit card
- They also have data about their customers (estimated income, family status,…)
- Recently, they are trying to improve their image for young mothers
- As a start they want to know the following information for mothers under 30 for 2013:
    - How much do they spend?
    - How much do they spend per state?
    - How does this compare to all customers under 30?
    - What are their favorite products?
    - How much do they spend per year?

Your first project: Design the schema for Evil Market!


Shop
- Shop_ID
- Business_Type
- City
- City_Population
- State
- …
Customer
- Customer_ID
- Name
- Segment
- Group_Name
- …
Fact Table
- Shop_ID
- Customer_ID
- Date_ID
- Product_ID
- Amount
- Volume
- Profit
- …
Time
- Date_ID
- Month
- Quarter
- Year
- …
Product
- Product_ID
- Type
- Brand
- Description
- …


State
- State_ID
- Name
City
- City_ID
- State_ID
- Name
- Population
Customer
- Customer_ID
- Name
- …
Customer Group
- Group_ID
- Segment
- Name
Shop
- Shop_ID
- City_ID
- Business_Type
Fact Table
- Shop_ID
- Customer_ID
- Date_ID
- Product_ID
- Amount
- Volume
- Profit
- …
Time
- Date_ID
- Month_ID
Product
- Product_ID
- Type_ID
- Brand
Quarter
- Quarter_ID
- Name
- Year
Month
- Month_ID
- Name
- Quarter
Brand
- Brand_ID
- Name
Product_Type
- Tyoe_ID
- Name
- Description

OTHER ANNOUNCEMENTS

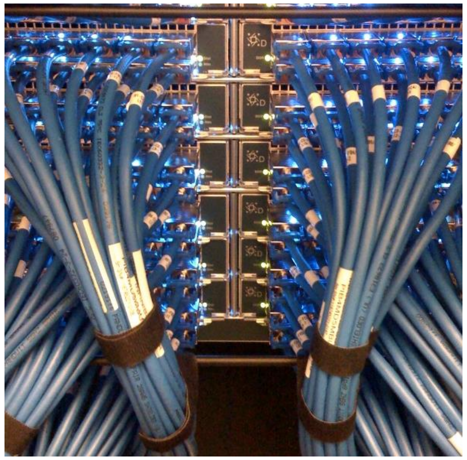
Want to get involved in research?

We are offering several independent studies and summer research internship.

Sign-up available on: http://database.cs.brown.edu/

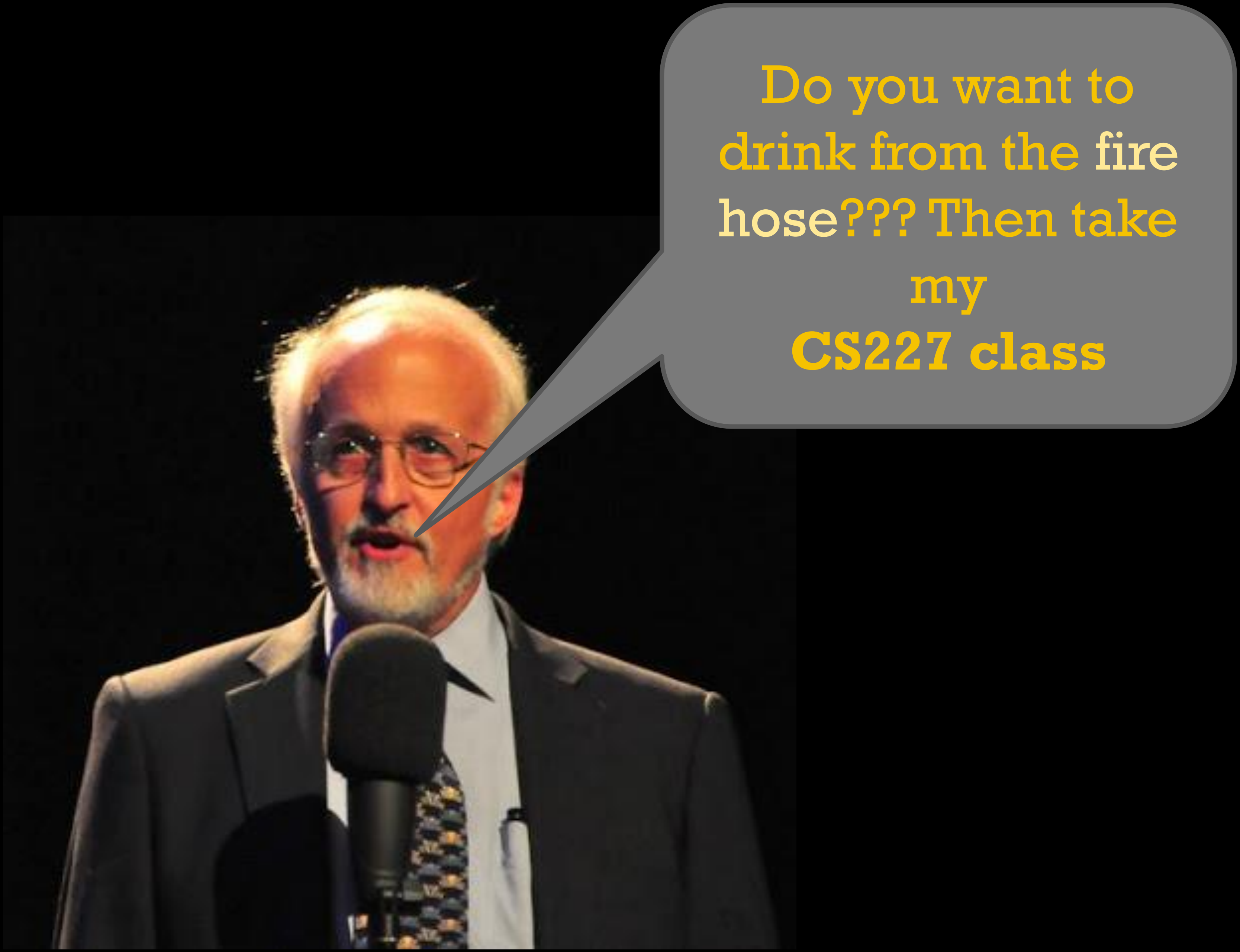
or directly: http://tinyurl.com/zxznf92

Possible Topics:


Infiniband

Tupleware

Interactive Data Exploration


Do you want to drink from the fire hose??? Then take my **CS227 class**