RELATIONAL-MODEL

INTRODUCTION TO DATA SCIENCE

CARSTEN BINNIG
BROWN UNIVERSITY

teaching datascience .org

DATABASES FOR DATA SCIENTIST

Requirement Engineering → Conceptual Modeling → Logical & Physical Modeling → Ask Questions

Requirement Engineering	Conceptual Modeling	Logical & Physical Modeling	Ask Questions
Book of duty	Conceptual Design (ER)	• Logical design (schema) • Physical design (index, hints)	• Relational Algebra • SQL

MODELING ELEMENTS

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:string, acct_no:string, balance:double))

WHY ARE TABLES CALLED RELATIONS?

Relation:

- $R \subseteq D_1$ x ... x D_n
- D_1, D_2, ..., D_n are domains

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

WHY ARE TABLES 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 TABLES 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, Tel#:integer*]}

EXAMPLE: 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)

bname	bcity	assets
Brighton	Brooklyn	5M
Brighton	Boston	3M

Invalid!!

NULLS

NULL are special values that can be used for any data type

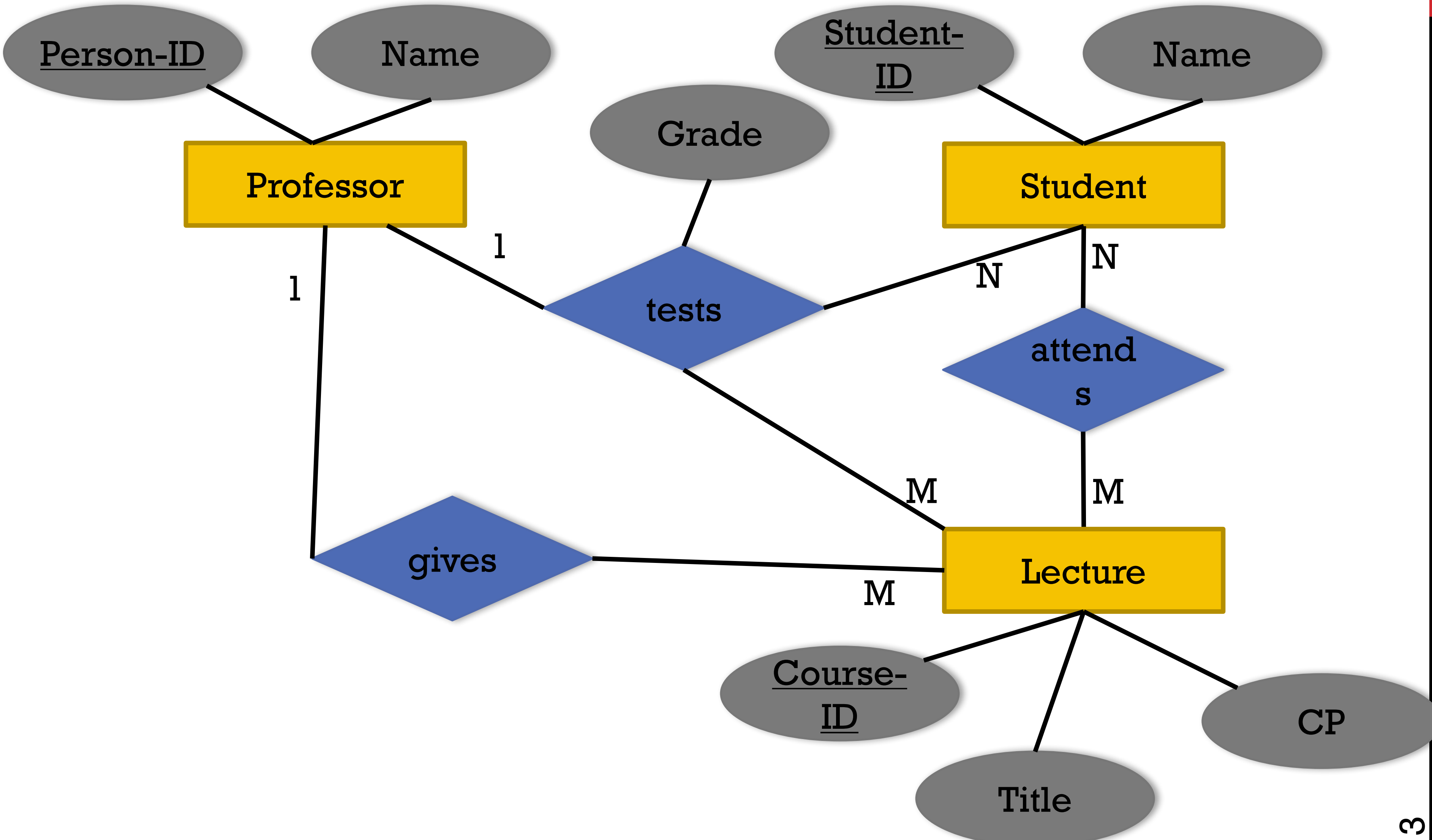
NULL typically means that value is not known

Keys can not be NULL

<u>**ISBN**</u>	**Title**	**Author**	**Genre**	**Publisher**	**Price**
0439708184	Harry Potter	J.K. Rowling	FANTASY	Scholastic	$6.70
0545663261	Mockingjay	Suzanne Collins	NULL	Scholastic	$7.39
…	….	…	…	…	…

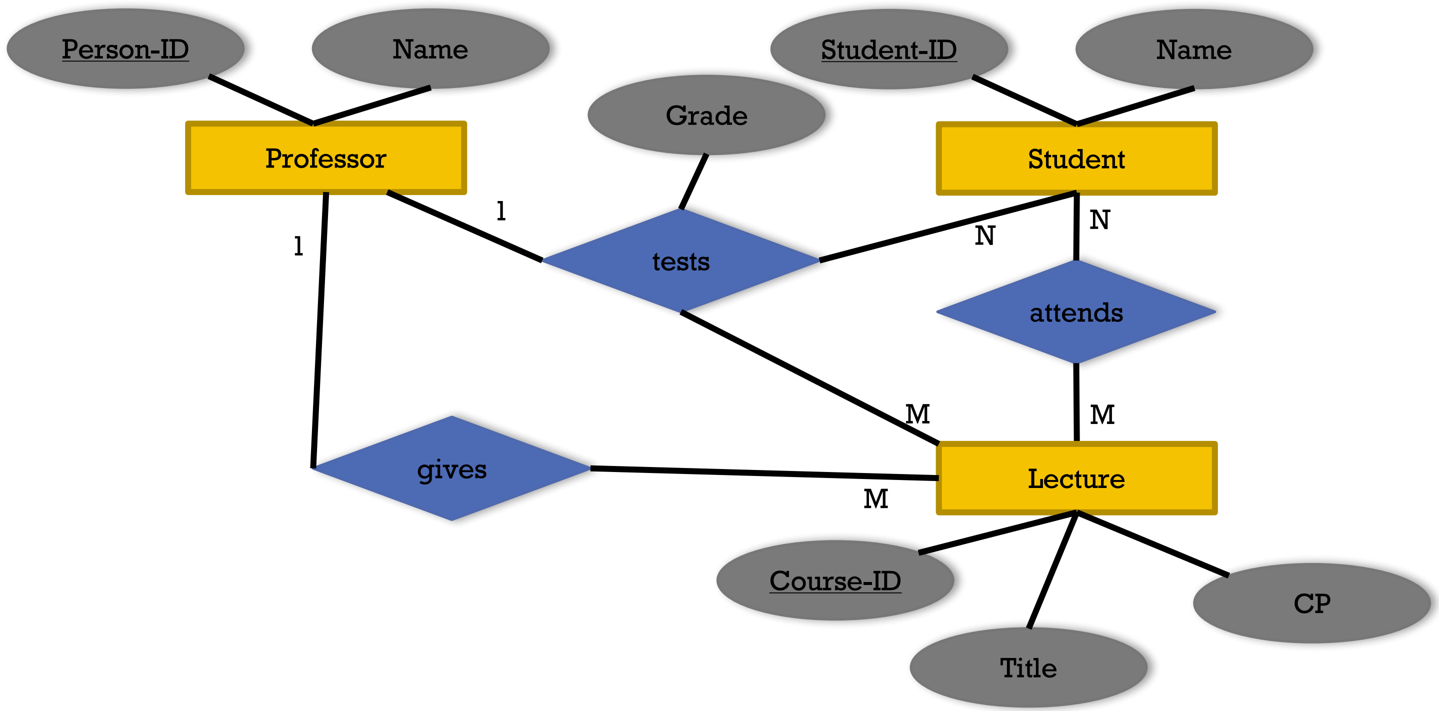
TRANSLATION OF ER TO RELATIONAL MODELS

HOW TO TRANSLATE ER TO RELATIONS?

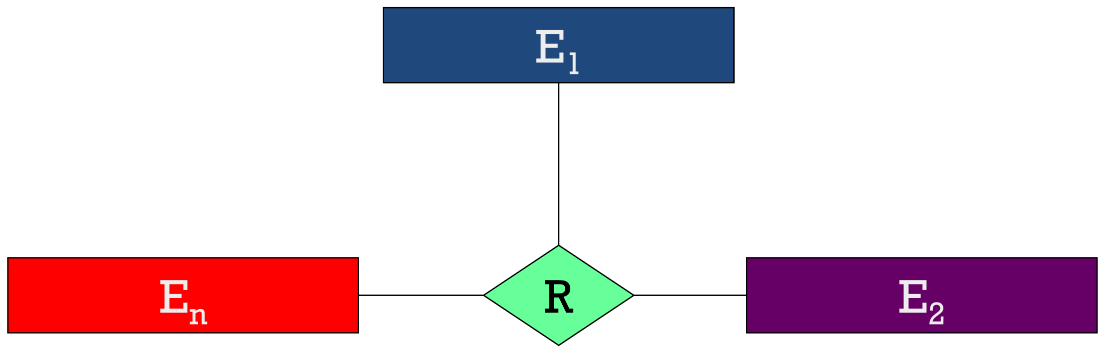


RULE #1: ENTITIES

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



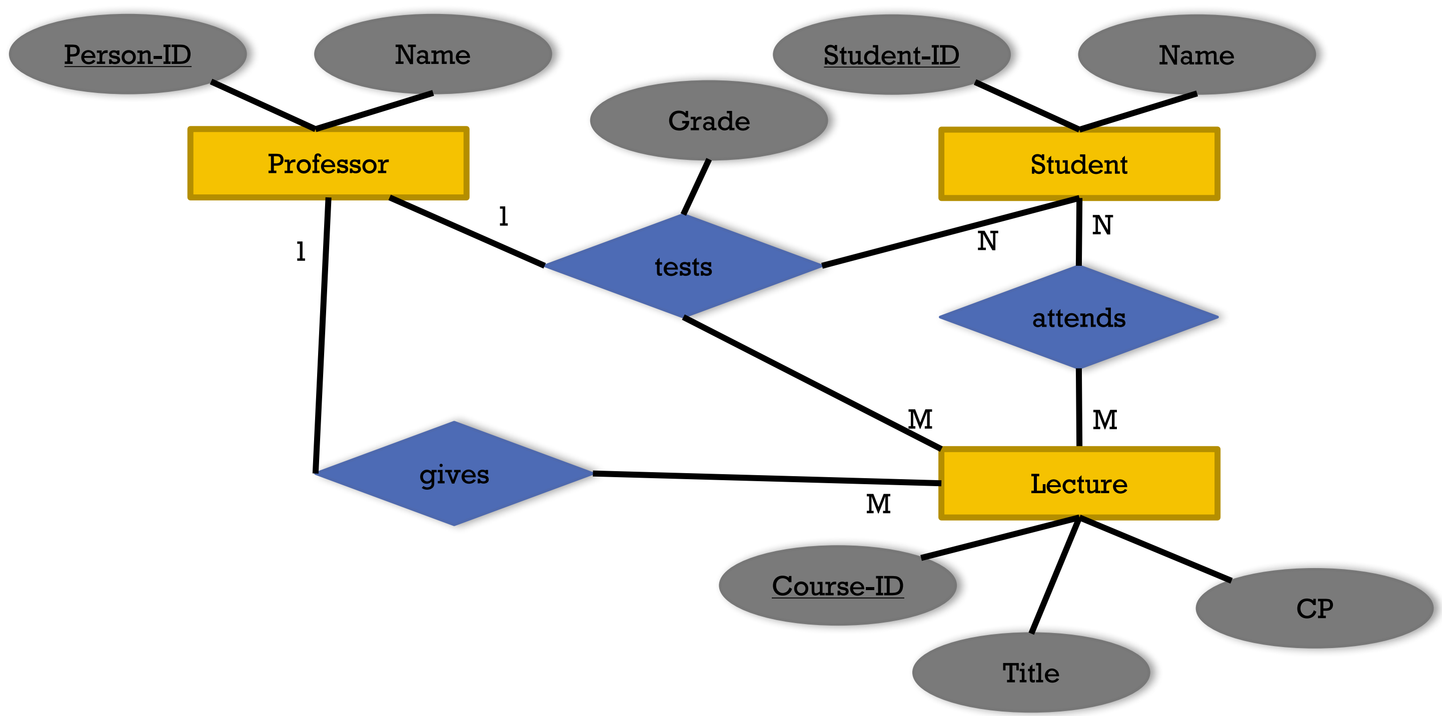
...

$$R:\{[\underbrace{A_{11}, \ldots, A_{1i}}_{\text{Keys of } E_1}, \underbrace{A_{21}, \ldots, A_{2j}}_{\text{Keys } E_2}, \ldots, \underbrace{A_{n1}, \ldots, A_{nl}}_{\text{Keys of } E_n}, \underbrace{A_{R1}, \ldots, A_{Rm}}_{\text{Attributes of } R}]\}$$

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?

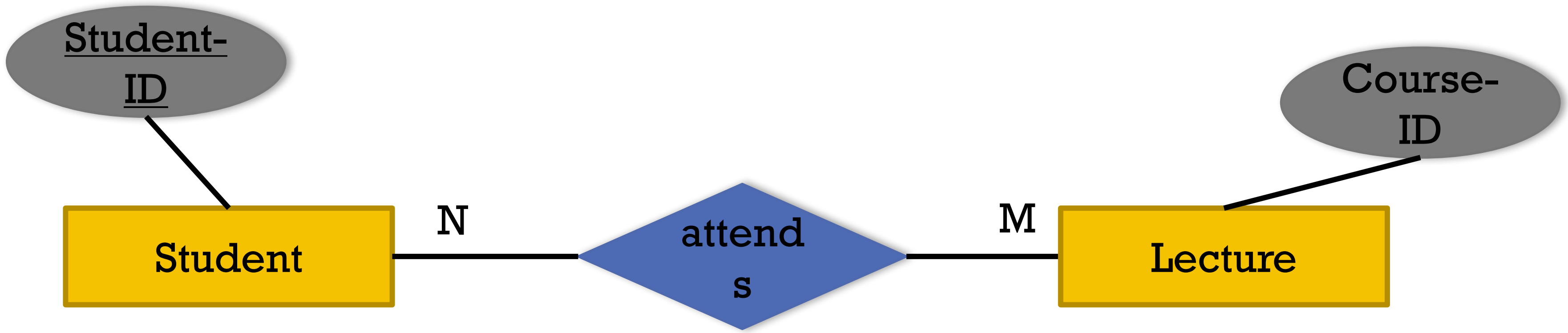


EXAMPLE: KEYS OF ATTENDS?

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 #2: RELATIONSHIPS

Keys depends on type of relationship

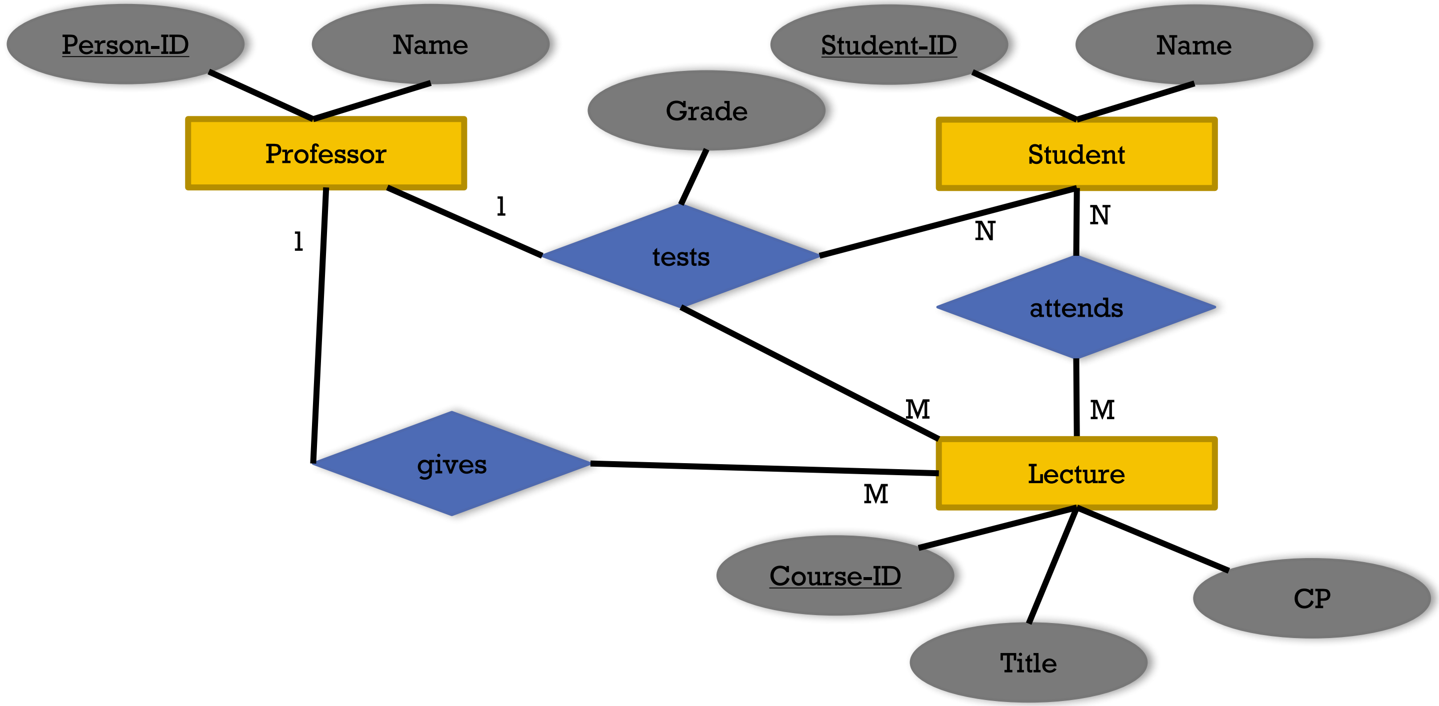
1:1 – Take key of one of the connected entities

1:N – Take key from the N-side

N:M – Combine keys of connected entities

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, <u>Course-ID:string</u>)
Attends(<u>Student-ID:integer, Course-ID:string</u>)
Tests(<u>Student-ID:integer, Course-ID:string</u>, Person-ID:integer, ,
Grade:string)



RULE #3: MERGE RELATIONS

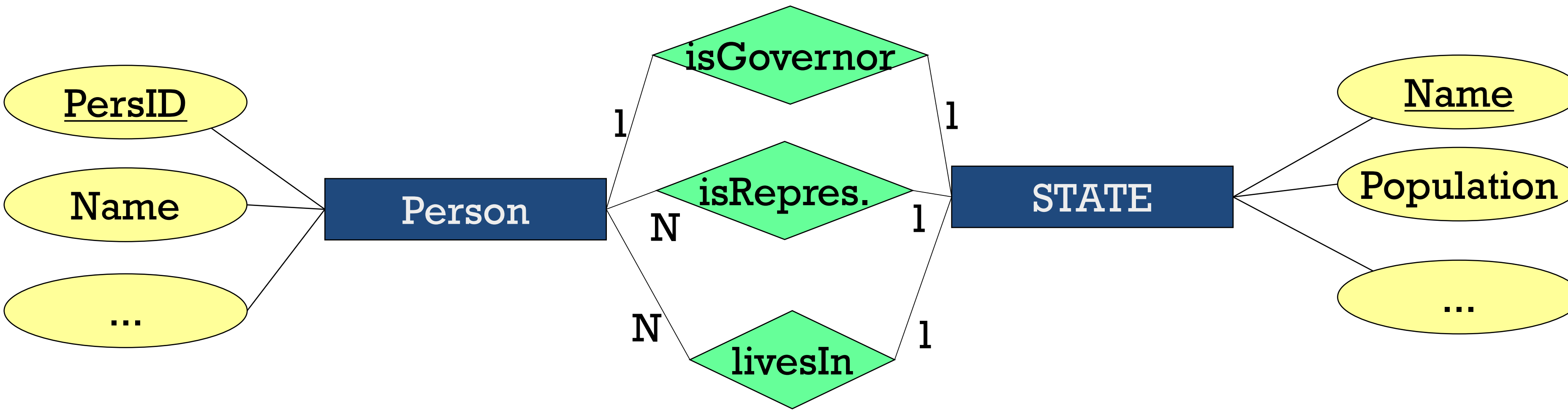
Merge relations of relationship into relations of entities

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

↓

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

RULE #3: EXCEPTION



Person					
PersID	Name	...	lives In	is Repr.	is Governor
4711	Binnig	...	RI	NULL	NULL
4813	Pyne	...	RI	NULL	NULL
...	...	...	...	...	
9011	Geller	...	RI	RI	NULL
9012	Raymondo	...	RI	RI	RI
...	...	...	...	...	

Problem: Many NULLs

Solution: Do no merge relations!

FINAL RELATIONAL MODEL

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**?

WHAT IS A GOOD RELATIONAL MODEL?

WHAT IS A BAD RELATIONAL MODEL?

Redundancy can lead to anomalies?

PersonID	**Name**	**…**	**CourseID**	**Title**
1	Upfal	…	1	Statistics
1	Upfal	…	2	Intro to Data Science
2	Felzen-schwalb	…	3	ML
3	Mr. X	…	NULL	NULL

- Update anomaly: not all values are updates
- Insert anomaly: NULL values must be inserted
- Delete anomaly: Information is lost

NORMALIZATION

Process of decomposing a relational schema to avoid redundancy

Decomposition must be lossless

Normal forms:

- **First and Second NF:** Contain some redundancy
- **Third NF:** Used in practice
- **Boyce-Codd NF**: Special form of Third NF
- **Other NFs:** Forth and Fifth NF

LOSSLESS DECOMPOSITION

PersonID	**Name**	**…**	**<u>CourseID</u>**	**Title**
1	Upfal	…	1	Statistics
1	Upfal	…	2	Intro to Data Science
2	Felzen-schwalb	…	3	ML

Decomposition

<u>PersonID</u>	**Name**
1	Upfal
2	Felzen-schwalb

<u>CourseID</u>	**Title**	**PersonID**
1	Statistics	1
2	Intro to Data Science	1
3	ML	2

Reconstruction possible?

FIRST NORMAL FORM

"all columns represent atomic (non-complex) values"

<u>PersonID</u>	**Name**	**…**	**Courses**
1	Upfal	…	{1, Stats}, {2, Intro to Data Science}
2	Felzen- schwalb	…	{3, ML}
3	Mr. X	…	NULL

Here: Courses is NOT atomic

SECOND NORMAL FORM

"all of a relation's nonkey attributes are dependent on all keys"

PersonID	**Name**	**…**	**CourseID**	**Title**
1	Upfal	…	1	Statistics
1	Upfal	…	2	Intro to Data Science
2	Felzen-schwalb	…	3	ML
3	Mr. X	…	NULL	NULL
4	Binnig	…	2	Intro to Data Science

Here: Name only depends on PersonID and not on CourseID

THIRD NORMAL FORM

"if it is in second normal form and has no transitive dependencies"

<u>**PersonID**</u>	**Name**	**Zip**	**City**
1	Upfal	02902	Providence
2	Felzen- schwalb	02905	Cranston
3	Mr. X	02902	Providence
…	…	…	…

Here: City depends on Zip and Zip on PersonID

DATABASES FOR DATA SCIENTIST

Requirement Engineering →	Conceptual Modeling →	Logical & Physical Modeling →	Ask Questions
Book of duty	Conceptual Design (ER)	• Logical design (relational schema) • Physical design (index, hints)	• Relational Algebra • SQL

REMINDER

1) **Clicker:** Register in Banner

2) **Assignment 1:** Due by 2/9

3) **Assignment 2:** Out 2/9

4) **Projects:** Dan will come in class 2/9

RELATIONAL ALGEBRA

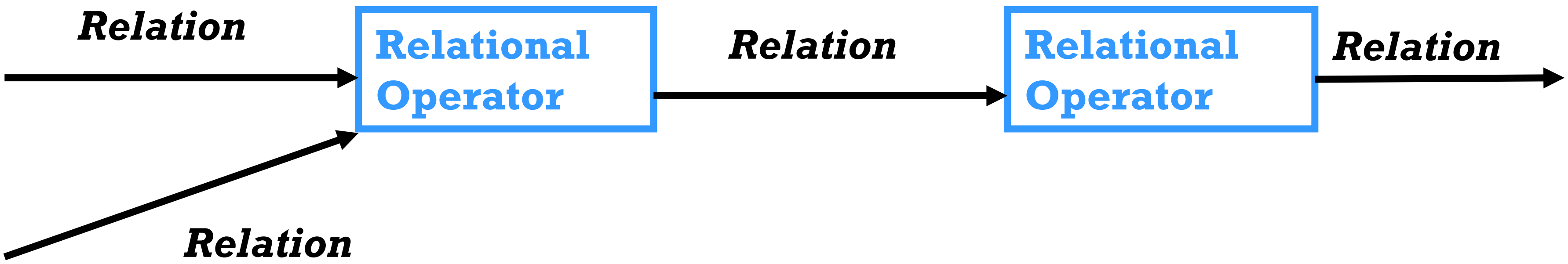
FORMAL DEFINITION OF REL. ALGEBRA

Operators (unary and binary):

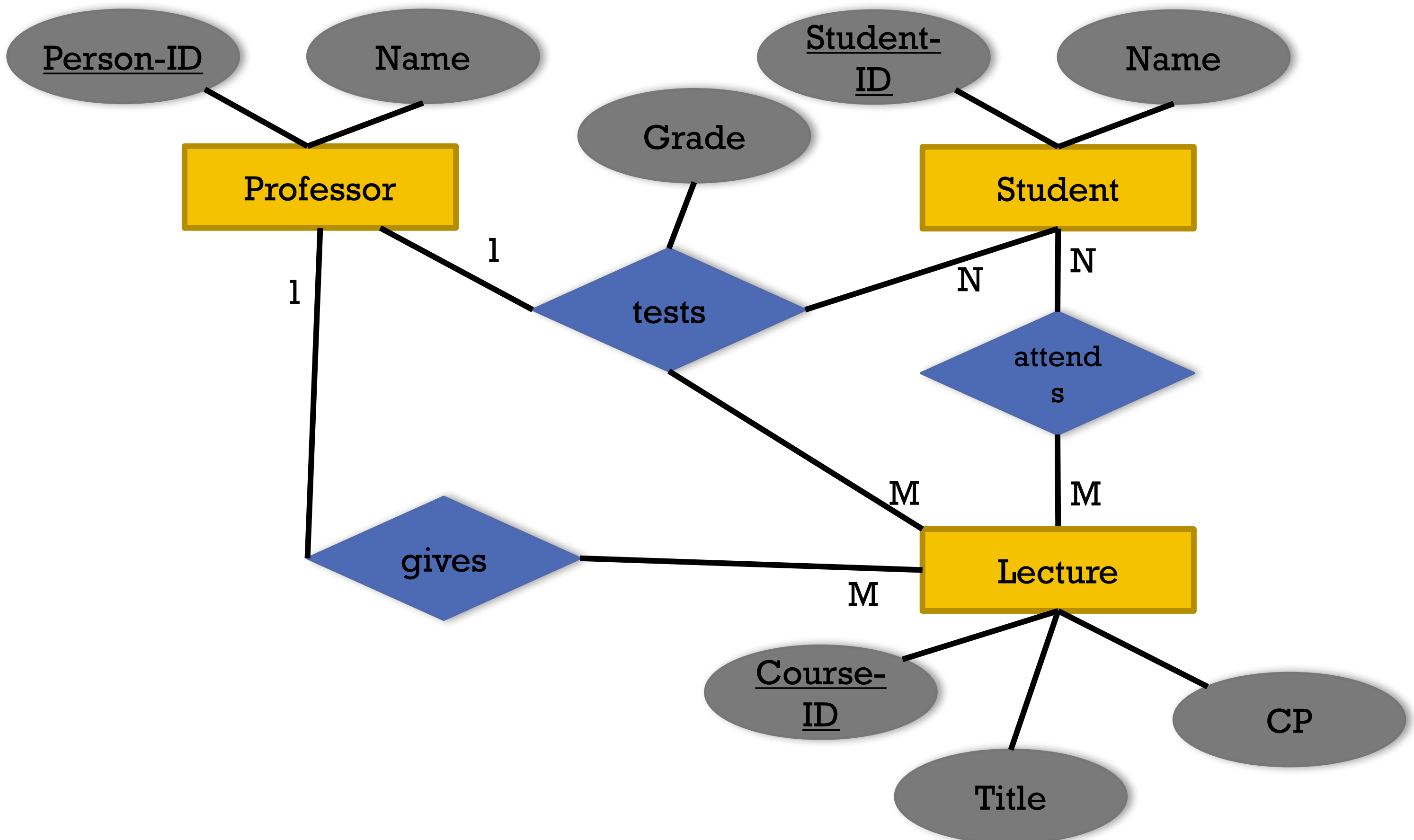
- **Selection**: σ (E1)
- **Projection**: Π (E1)
- **Cartesian Product**: E1 x E2
- **Rename**: ρ_V(E1), $\rho_{A \leftarrow B}$ (E1)
- **Union**: E1 $\cup$ E2
- **Minus**: E1 - E2
- **Other:** Joins, Aggregation, …

Input and Output: Relations

CLOSURE PROPERTY / COMPOSABILITY



Professor(<u>Person-ID:integer</u>, Name:varchar(30), Level:varchar(2))
Student(<u>Student-ID:integer</u>, Name:varchar(30), Semester:integer)
Lecture(<u>Course-ID:varchar(10)</u>, Title:varchar(50), CP:float)
Gives(Person-ID:integer, <u>Course-ID:varchar(10)</u>)
Attends(<u>Student-ID:integer</u>, <u>Course-ID:varchar(10)</u>)
Tests(<u>Student-ID:integer</u>, <u>Course-ID:varchar(10)</u>, Person-ID:integer, Grade:char(2))



SELECTION AND PROJECTION

Professor(<u>Person-ID:integer</u>, Name:varchar(30), Level:varchar(2))
Student(<u>Student-ID:integer</u>, Name:varchar(30), Year:integer)

Selection

$\sigma_{Year<5}$ (Student)

Student-ID	Name	Year
24002	Sally	2
25403	Emily	3

Projection

Π_{Name}(Professor)

Name
Eli
Stephanie

CARTESIAN PRODUCT

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

x

R	
D	E
d_1	e_1
d_2	e_2

=

Result				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_2	e_2
a_2	b_2	c_2	d_1	e_1
a_2	b_2	c_2	d_2	e_2

CARTESIAN PRODUCT (CTD.)

Student x attends?

Student		
Student-ID	Name	Year
26120	Sally	2
29555	Emily	3

Attends	
Student-ID	Course-ID
26120	5001
26120	5004
29555	5004
29555	5111

CARTESIAN PRODUCT (CTD.)

Student x attends

Student			**Attends**	
Student-ID	Name	Year	Student-ID	Course-ID
26120	Sally	2	26120	5001
26120	Sally	2	26120	5004
26120	Sally	2	29555	5004
26120	Sally	2	29555	5111
29555	Emily	3	26120	5001
...	...	...	...	...

- Huge result set (n * m)
- Usually only useful in combination with a selection (-> Join)

NATURAL JOIN

Two relations:

- $R(A_1,\ldots, A_m, B_1,\ldots, B_k)$
- $S(B_1,\ldots, B_k, C_1,\ldots, C_n)$

$$R \bowtie S = \Pi_{A1,\ldots,\, Am,\, R.B1,\ldots,\, R.Bk,\, C1,\ldots,\, Cn}(\sigma_{R.B1=S.B1 \wedge \ldots \wedge R.Bk = S.Bk}(R x S))$$

$R \bowtie S$											
$R - S$				$R \cap S$				$S - R$			
A_1	A_2	…	A_m	B_1	B_2	…	B_k	C_1	C_2	…	C_n
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

⋮

NATURAL JOIN

(Student ⋈ attends)

Student			Attends
Student-ID	Name	Year	Course-ID
26120	Sally	2	5001
26120	Sally	2	5004
29555	Emily	3	5004
29555	Emily	3	5011

THETA-JOIN

Two Relations:

- R(A1, …, An)
- S(B1, …, Bm)

$$R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$$

$R \bowtie_\theta S$							
R				S			
A_1	A_2	…	A_n	B_1	B_2	…	B_m
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

JOIN VARIANTS

- natural join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$\bowtie$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Result				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1

- left outer join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$⟕$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Result				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-

JOIN VARIANTS

- right outer join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⟖

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Result				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
-	-	c_3	d_2	e_2

JOIN VARIANTS

- (full) outer join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$⟗$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Result				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-
-	-	c_3	d_2	e_2

- left semi join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$\ltimes$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Result		
A	B	C
a_1	b_1	c_1

JOIN VARIANTS

- right semi join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$\rtimes$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat		
C	D	E
c_1	d_1	e_1

AGGREGATION OPERATOR

$\chi_{group,aggreation}$

orderID	custID	custName	orderTotal
1	1	Sally	100
2	1	Sally	500
3	1	Sally	400
4	2	Emily	333

$\downarrow$ $\chi_{\{custID,\ custName\},\ \{sum(oTotal),\ min(oTotal)\}}$

custID	custID	Sum	Min
1	Sally	1000	100
2	Emily	333	333

RENAME OPERATOR ρ

Renaming of relation names

- Needed to process self-joins and recursive relationships
- E.g., two-level dependencies of lectures („grandparents“)

$$\Pi_{\mathbf{L1.Prerequisite}}(\sigma_{\mathbf{L2.course\text{-}id=CS2270} \wedge \mathbf{L2.prerequisite=L1.course\text{-}id}}(\rho_{\mathbf{L1}}\ (\mathbf{Requires})\ \mathbf{x}\ \rho_{\mathbf{L2}}\ (\mathbf{Requires})))$$

Ρεθυιρεσ

course-id	prerequisite
CS1951A	CS160
CS160	CS320
CS2270	**CS1270**
CS1270	**CS160**

Renaming of attribute names

$$\rho_{\mathbf{Requirement} \leftarrow \mathbf{Prerequisite}}\ (\mathbf{requires})$$

SET DIFFERENCE (–)

Notation: $\mathit{Relation}_1$ - $\mathit{Relation}_2$

R - S valid only if:

1. *R, S have same number of columns (arity)*
2. *R, S corresponding columns have same domain (compatibility)*

Example:

$$(\Pi_{bname}\ (\sigma_{amount \geq 1000}\ (loan))) - (\Pi_{bname}\ (\sigma_{balance < 800}\ (account)))$$

loan

bname	lno	amount
Downtown	L-17	1000
Redwood	L-23	2000
Perry	L-15	1500
Downtown	L-14	500
Perry	L-16	300

account

bname	acct_no	balance
Mianus	A-215	700
Brighton	A-201	900
Redwood	A-222	700
Brighton	A-217	850

Result?

= (1)

bname
Mianus
Redwood

(3)

bname
Downtown
Redwood
Perry

(2)

bname
Downtown
Perry

SET DIFFERENCE (–)

Notation: $Relation_1$ - $Relation_2$

R - S valid only if:

1. *R, S have same number of columns (arity)*
2. *R, S corresponding columns have same domain (compatibility)*

Example:

$$(\Pi_{bname}\ (\sigma_{amount \geq 1000}\ (loan))) - (\Pi_{bname}\ (\sigma_{balance < 800}\ (account)))$$

loan

bname	lno	amount
Downtown	L-17	1000
Redwood	L-23	2000
Perry	L-15	1500
Downtown	L-14	500
Perry	L-16	300

account

bname	acct_no	balance
Mianus	A-215	700
Brighton	A-201	900
Redwood	A-222	700
Brighton	A-217	850

Result?

= (1)

bname
Mianus
Redwood

(3)

bname
Downtown
Redwood
Perry

(2)

bname
Downtown
Perry

INTERSECTION

$\Pi_{\text{Person-ID}}(\text{Lecture}) \cap \Pi_{\text{Person-ID}}(\sigma_{\text{Area=DB}}(\text{Professor}))$

Only works if both relations have the same schema

- Same attribute names and attribute domains

Intersection can be simulated with minus:

$\mathbf{R \cap S = R - (R - S)}$

Union should be trivial …

RELATIONAL DIVISION

Relational Division R ÷ S can be used for universal quantfication

Example: Find StudentIDs (Student), that attended all 2CP lectures

Algebra: attend ÷ S while $S = \Pi_{CourseID}(\sigma_{CP=2}(Course))$

Universal Quantfier: $\{s \mid s \in Student \wedge \forall l \in Course(c.CP=2 \Rightarrow \exists a \in attend(a.CourseID=c.CourseID \wedge a.StudentID=s.StudentID))\}$

EXAMPLE: RELATIONAL DIVISION

R: attend	
Student ID	Course ID
s_1	c_1
s_1	c_2
s_1	c_3
s_2	c_2
s_2	c_3

$\div$

S: $\Pi_{CourseID}(\sigma_{CP=2}$ (Course))
CourseID
c_1
c_2

$=$

$R \div S$
StudentID
s_1

CODD`S THEOREM

3 Languages:

- **Relational Algebra**
- **Tuple Relational Calculus** (safe expressions only)
- **Domain Relational Calculus** (safe expressions only)

are equivalent.

Impact of Codd`s theorem:

- SQL is based on the **relational calculus (but with duplicates!)**
- SQL implementation is based on **relational algebra**
- Codd`s theorem shows that **SQL is correct and complete.**

FURTHER MATERIAL

Not covered here

- **Aggregate Functions**
- **Codd´s Proof**
- **...**

But why not?

- **Sorting**
- **Duplicate Elicitation**

IN CLASS TASKS

Player

PlayerID	Name	Age	Team
1	Russel	27	Seahawks
…	…	…	…

Team

Team	State
Seahawks	Washington
…	…

Played

PlayerID	Date	Place	Score
1	2/1/15	Phoenix	3
…	…	…	…

In relational algebra:

1) Return all teams, who played at least once in Phoenix
2) Return all Seahawks player names, who did not play in the entire season

CLICKER QUESTION I

Player

PlayerID	Name	Age	Team
1	Russel	27	Seahawks
…	…	…	…

Team

Team	State
Seahawks	Washington
…	…

Played

PlayerID	Date	Place	Score
1	2/1/15	Phoenix	3
…	…	…	…

Return all teams, who played at least once in Phoenix

A) Π_{Team} ($\sigma_{Place=`Phoenix`}$ (Player X Played))

B) Π_{Team} (Player $\bowtie$($\sigma_{Place=`Phoenix}$ (Played)))

C) Π_{Team} ($\sigma_{Place=`Phoenix`}$ (Player $\bowtie$ Played))

CLICKER QUESTION I

Player

PlayerID	Name	Age	Team
1	Russel	27	Seahawks
…	…	…	…

Team

Team	State
Seahawks	Washington
…	…

Played

PlayerID	Date	Place	Score
1	2/1/15	Phoenix	3
…	…	…	…

Return all teams, who played at least once in Phoenix

A) Π_{Team} ($\sigma_{Place=`Phoenix`}$ (Player X Played))

B) Π_{Team} (Player $\bowtie$($\sigma_{Place=`Phoenix}$ (Played)))

C) Π_{Team} ($\sigma_{Place=`Phoenix`}$ (Player $\bowtie$ Played))

CLICKER QUESTION II

1. Π_{Team} $(\sigma_{Place=`Phoenix`}$ $(Player \bowtie Played))$
2. Π_{Team} $(Player \bowtie (\sigma_{Place=`Phoenix}$ $Played))$
3. Π_{Team} $(\sigma_{Place=`Phoenix`}$ $(Player \bowtie Played \bowtie Team))$

Which of these expressions are equivalent?

A) All
B) 1 and 2
C) 1 and 3
D) 2 and 3
E) None

CLICKER QUESTION II

1. Π_{Team} $(\sigma_{Place=`Phoenix`}$ $(Player \bowtie Played))$
2. Π_{Team} $(Player \bowtie (\sigma_{Place=`Phoenix}$ $Played))$
3. Π_{Team} $(\sigma_{Place=`Phoenix`}$ $(Player \bowtie Played \bowtie Team))$

Which of these expressions are equivalent?

A) All
B) 1 and 2
C) 1 and 3
D) 2 and 3
E) None

CLICKER QUESTION III

Player

PlayerID	Name	Age	Team
1	Russel	27	Seahawks
…	…	…	…

Team

Team	State
Seahawks	Washington
…	…

Played

PlayerID	Date	Place	Score
1	2/1/15	Phoenix	3
…	…	…	…

Return all Seahawks player names, who did not play so far

A) $\Pi_{Name}(\sigma_{Team=`Seahawks`}(Player ⟕ Played))$

B) $\Pi_{Name}(\sigma_{Team=`Seahawks`}(Player)) - \Pi_{Name}(\sigma_{Team=`Seahawks`}(Player \bowtie Played))$

C) $\Pi_{Name}(\sigma_{Team=`Seahawks`}(Player \ltimes Played))$

D) $\Pi_{Name}(\sigma_{Team=`Seahawks` \wedge Date\ is\ null}(Player ⟖ Played))$

CLICKER QUESTION III

Player

PlayerID	Name	Age	Team
1	Russel	27	Seahawks
…	…	…	…

Team

Team	State
Seahawks	Washington
…	…

Played

PlayerID	Date	Place	Score
1	2/1/15	Phoenix	3
…	…	…	…

Return all Seahawks player names, who did not play so far

A) $\Pi_{\text{Name}}(\sigma_{\text{Team=`Seahawks`}}(\text{Player} \text{ ⟕ } \text{Played}))$

B) $\Pi_{\text{Name}}(\sigma_{\text{Team=`Seahawks`}}(\text{Player})) - \Pi_{\text{Name}}(\sigma_{\text{Team=`Seahawks`}}(\text{Player} \bowtie \text{Played}))$

C) $\Pi_{\text{Name}}(\sigma_{\text{Team=`Seahawks`}}(\text{Player} \ltimes \text{Played}))$

D) $\Pi_{\text{Name}}(\sigma_{\text{Team=`Seahawks`} \wedge \text{Date is null}}(\text{Player} \text{ ⟖ } \text{Played}))$

SUMMARY

Relational Model

- **Tables / columns / rows**
- **set-oriented**
- **Keys**
- **NULL values**

Translation of ER-Models into Relational Model

Relational Algebra

- **Query Relations**
- **Composable**
- **Duplicate-free**