Chapter 2: Intro to Relational Model

Database System Concepts, 6th Ed.

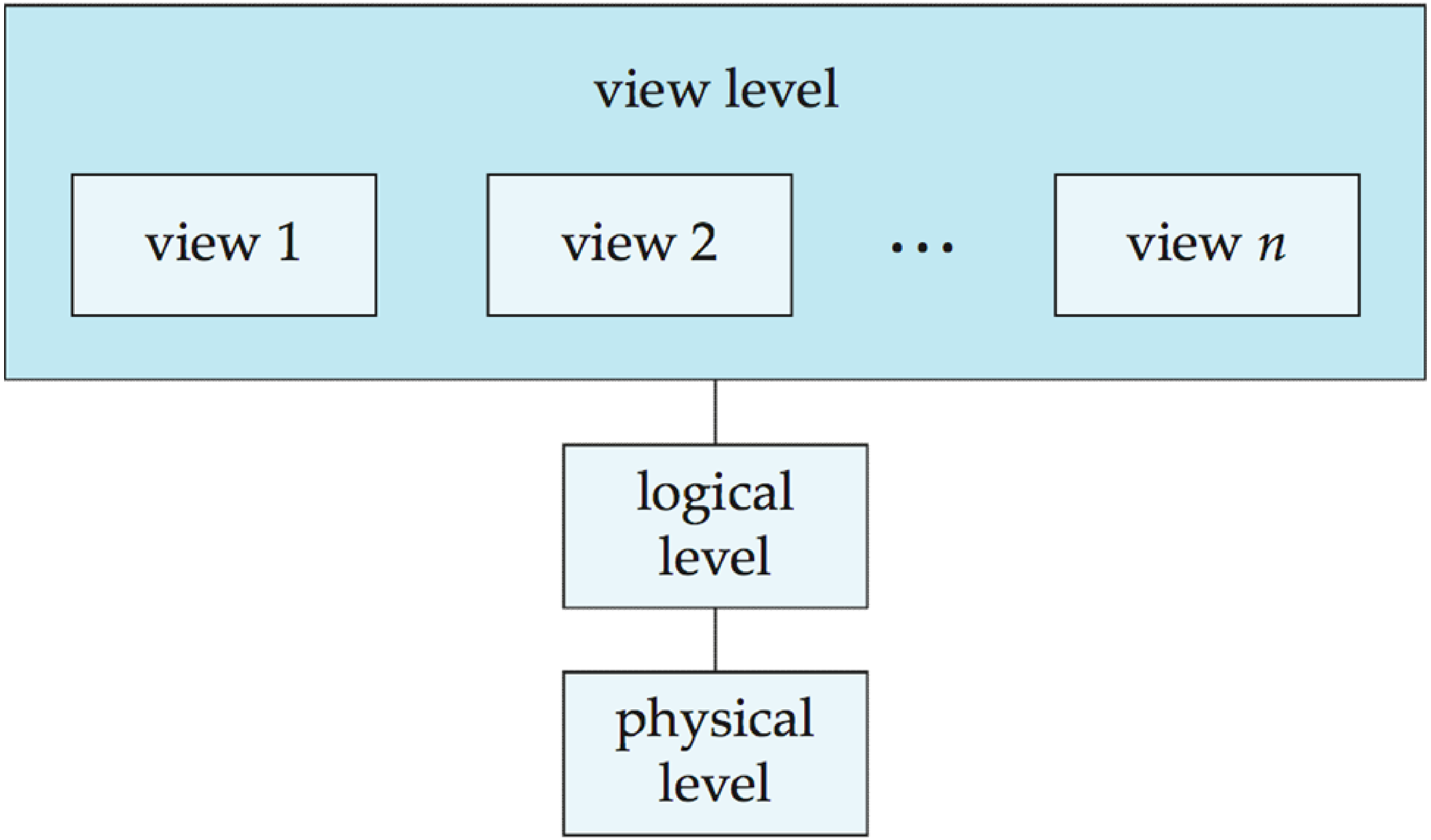
Review: Chapter 1

- Data Models
- Relational Databases -> Schema & Instance
- Database Design
- Storage Manager
- Query Processing
- Transaction Manager

Q1: Merits & Drawbacks of Database ?

Review: Chapter 1

Q2: Architecture for a database system?



Data Model

A data model is a collection of **conceptual tools** for describing ***data, data relationships, data semantics, and consistency constraints***.

- ***Relational model (our focus)***
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Example of a Relation

Relational data model uses a collection of tables to represent both data and the relationships

attributes
(or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples
(or rows)

Today a vast majority of DBMS products are based on the relational model.

Important issues

To make data from a relational database available to users, we have to address several issues

- The most important issue is how users specify requests for retrieving and updating data;

Chapters 3, 4 and 5 cover the SQL language

Chapter 6 covers three formal query languages, the relational algebra, the tuple relational calculus and the domain relational calculus

- data integrity and protection;

Chapter 4 also covers integrity constraints

Relational Model

Stucture of Relational Database

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 2.1 The *instructor* relation.

a row* in a table represents a relationship among *a set of values

a table is a collection of such relationships,

the term **relation** is used to refer to a table

the term **tuple** is used to refer to a row.

the term **attribute** refers to a column of a table.

Attribute Types

- 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. Indicated that the value is “unknown”/does not exist
- The null value causes **complications** in the definition of many operations

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 2.1 The *instructor* relation.

course_id	*prereq_id*
BIO-301	BIO-101
BIO-399	BIO-101
CS-190	CS-101
CS-315	CS-101
CS-319	CS-101
CS-347	CS-101
EE-181	PHY-101

Figure 2.3 The *prereq* relation.

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Figure 2.2 The *course* relation.

Relation Schema and Instance

A relation schema consists of a list of attributes and their corresponding domains.

- $A_1, A_2, \ldots, A_n$ are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  Example:

  instructor = (*ID*, *name*, *dept_name*, *salary*)
- Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation** ***r*** is a subset of

  $D_1 \times D_2 \times \ldots \times D_n$

  Thus, a relation is a set of *n*-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$
  - The current values (**relation instance**) of a relation are specified by a table
  - An element ***t*** of ***r*** is a *tuple*, represented by a *row* in a table

Relation Schema and Instance

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Figure 2.5 The *department* relation.

Schema

department (dept name, building, budget)

course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Figure 2.6 The *section* relation.

section (course id, sec id, semester, year, building, room number, time slot id)

Keys

Challenge: We must have a way to specify how tuples within a given relation are distinguished.

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Figure 2.6 The *section* relation.

Which attribute can be used to distinguish a tuple?

the values of the attribute values of a tuple must be such that they can ***uniquely identify*** the tuple

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: {*ID*} and {ID,name} are both superkeys of *instructor.*
- Superkey *K* is a **candidate key** if *K* is **minimal**

  Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - which one?
- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
  - a foreign key must be the primary key for the referenced relation
  - Example – *dept_name* in i*nstructor* is a foreign key from *instructor* referencing *department*

Priminary Keys

The primary key should be chosen such that its attribute values are ***never , or very rarely, changed***.

*For instance, the **address** field of a person should not be part of the primary key, since it is likely to change.*

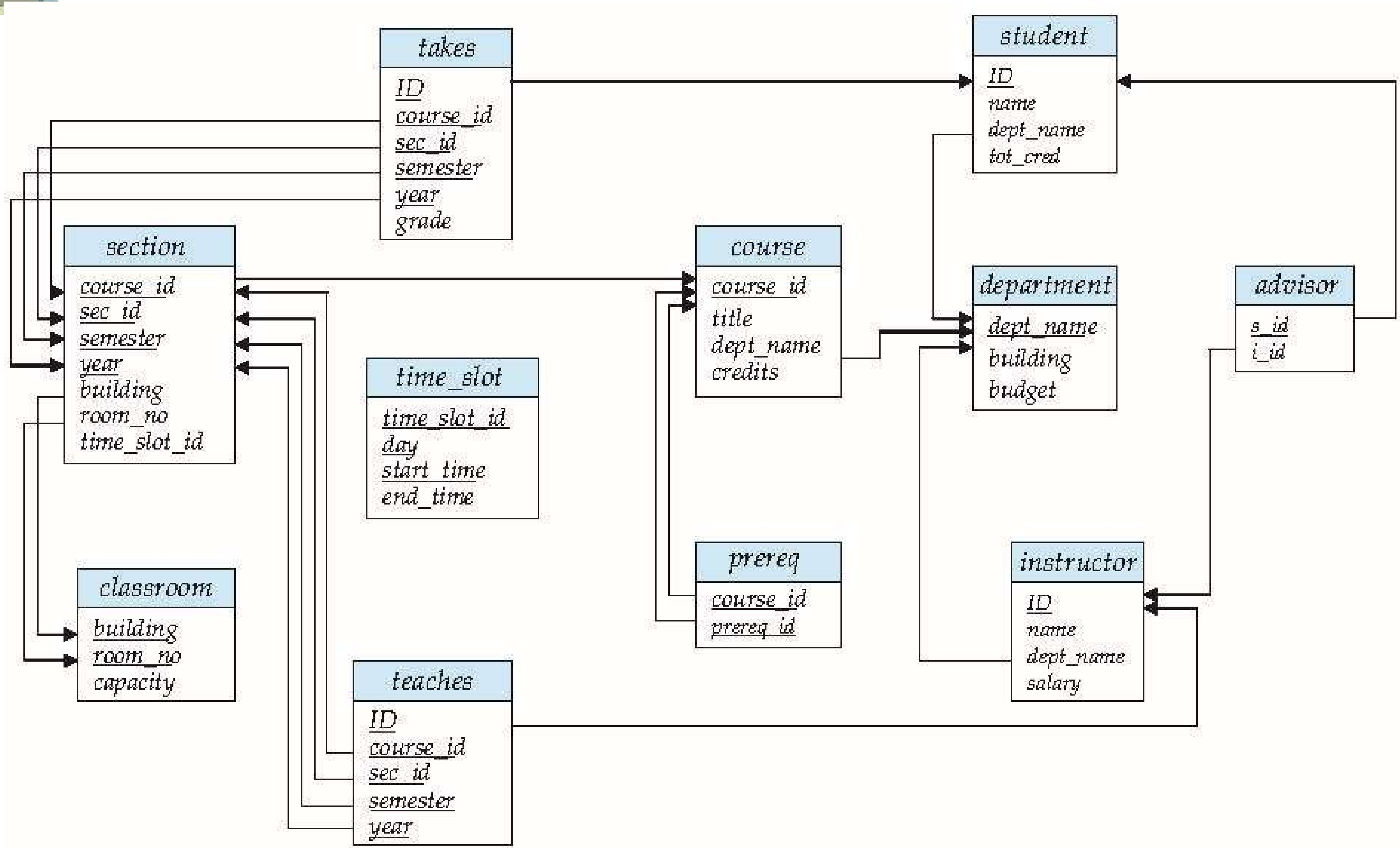
***Social-security** numbers, on the other hand, are guaranteed never to change.*

***Unique identifiers** generated by enterprises generally do not change, except if two enterprises merge; in such a case the same identifier may have been issued by both enterprises, and a reallocation of identifiers may be required to make sure they are unique.*

Schema Diagram

A database schema, along with primary key and foreign key dependencies, can be depicted by **schema diagrams**.

Schema Diagram for University Database



Relational Query Languages

A **query language** is a language in which a user requests information from the database. These languages are usually on a level higher than that of a standard programming language.

- Procedural vs .non-procedural, or declarative
- "Pure" languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- The above 3 pure languages are equivalent in computing power
- We will concentrate in this chapter on relational algebra
  - Not turning-machine equivalent
  - consists of 6 basic operations

Relational Operations

All procedural relational query languages provide ***a set of operations*** that can be applied to either **a single relation or a pair of relations**. These operations have the nice and desired property that ***their result is always a single relation***. This property allows one **to combine several of these operations** in a modular way.

+,-, /, X   The four operations are the basis of math.

Specifically, since the result of a relational query is itself a relation, relational operations can be applied to the results of queries as well as to the given set of relations.

Basic Relation Operations

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary}>=85000}(instructor)$ Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,\,salary}(instructor)$ Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
$\bowtie$ (Natural join)	$instructor \bowtie department$ Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
$\times$ (Cartesian product)	$instructor \times department$ Output all pairs of rows from the two input relations (regardless of whether or not they have the same values on common attributes)
$\cup$ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$ Output the union of tuples from the two input relations.

Select Operation – selection of rows (tuples)

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \land D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Select : Pick out specific tuples from a single relation that satisfies some particular predicate

Project Operation – selection of columns (Attributes)

A	B	C
a1	b1	C1
a1	b2	C2
a2	b2	c1

$\xrightarrow{\pi_{A,C}}$

A	C
a1	C1
a1	C2
a2	c1

Project: select certain attributes (columns) from a relation

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}\,(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

Union of two relations

- Relations *r, s:*

r

A	B
α	1
α	2
β	1

s

A	B
α	2
β	3

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

No duplicated records, because a relation is a set.

Union: performs a set union of two "similarly structured" tables

Set difference of two relations

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- *r* – *s:*

A	B
α	1
β	1

Set Difference: performs a set difference of two "similarly structured" tables

Set intersection of two relations

- Relation *r, s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

Note: $r \cap s = r - (r - s)$

Set Intersection: performs a set intersection of two "similarly structured" tables

joining two relations -- Cartesian-product

- Relations *r, s*:

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

the **natural join** operation on two relations matches tuples **whose values are the same on all attribute names that are common to both relations**.

- *r* x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

The ***Cartesian product*** operation combines tuples from two relations, but unlike the join operation, its result contains *all* pairs of tuples from the two relations,
regardless of whether their attribute values match.

The *join* operation allows the combining of two relations by merging pairs of tuples, one from each relation, into a single tuple.

Cartesian-product – naming issue

- Relations *r, s*:

r

A	B
α	1
β	2

s

B	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

- *r* x *s*:

A	$r.B$	$s.B$	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Renaming a Table

- Allows us to refer to a relation, (say E) by more than one name.

$$\rho_x(E)$$

returns the expression *E* under the name *X*

- Relations *r*

A	*B*
α	1
β	2

r

- $r \times \rho_s(r)$

r.A	*r.B*	*s.A*	*s.B*
α	1	α	1
α	1	β	2
β	2	α	1
β	2	β	2

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}\,(r \times s)$
- $r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- $\sigma_{A=C}\,(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

Joining two relations – Natural Join

- Let r and s be relations on schemas R and S respectively.
  Then, the "natural join" of relations R and S is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from r and t_s from s.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple t to the result, where
    - t has the same value as t_r on r
    - t has the same value as t_s on s

Natural Join Example

- Relations r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

- Natural Join
  - r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

$$\Pi_{A,\ r.B,\ C,\ r.D,\ E}\,(\sigma_{r.B\,=\,s.B\ \wedge\ r.D\,=\,s.D}\ (r \times s)))$$

the **natural join** operation on two relations matches tuples **whose values are the same on all attribute names that are common to both relations**.

Notes about Relational Languages

- Each Query input is a table (or set of tables)
- Each query output is a table.
- All data in the output table appears in one of the input tables
- Relational Algebra is not Turning complete
- Can we compute:
  - SUM
  - AVG
  - MAX
  - MIN

Summary of Relational Algebra Operators

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{\text{salary} >= 85000}$ (*instructor*)
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID,\ salary}$ (*instructor*)
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
x (Cartesian Product)	*instructor* x *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
∪ (Union)	Π_{name} (*instructor*) ∪ Π_{name} (*student*)
	Output the union of tuples from the *two* input relations.
– (Set Difference)	Π_{name} (*instructor*) -- Π_{name} (*student*)
	Output the set difference of tuples from the two input relations.
⋈ (Natural Join)	*instructor* ⋈ *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.

End of Chapter 2