Chapter 2: Intro to Relational Model

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan

Structure of Relational Databases

- Tables
- Attribute
- Record
- Tuples
- Relation
- Relation instance

Example of a Relation

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)

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.

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.

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”
- 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

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

Relation Schema and Instance

- $A_1, A_2, \ldots, A_n$ 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

SQL Data Definition

The SQL DDL allows specification of not only a set of relations, but also information about each relation, including:

- **The schema** for each relation.
- **The types** of values associated with each attribute.
- The **integrity constraints**.
- The set of **indices** to be maintained for each relation.
- The **security and authorization** information for each relation.
- The **physical storage structure** of each relation on disk.

Basic Types

- **char(*n*)**
- **varchar(*n*)**
- **Int**
- **smallint**
- **numeric(*p, d*)**
- **real, double precision**
- **float(*n*)**

Basic Schema Definition

create table ***r***

(*A1 D1,*

A2 D2,

. . . ,

An Dn,

integrity-constraint1

,

. . . ,

integrity-constraint*k*

);

SQL supports a number of different integrity constraints. In this section, we discuss only a few of them

- **primary key (*Aj1 , Aj2, . . . , Ajm)***
- **foreign key (*Ak1 , Ak2, . . . , Akn) references***
- **not null**

create table ***department***

(*dept name* ***varchar (20),***

building ***varchar (15),***

budget ***numeric (12,2),***

primary key (*dept name*));

create table ***course***

(*course id* ***varchar (7),***

title ***varchar (50),***

dept name ***varchar (20),***

credits ***numeric (2,0),***

primary key (*course id*),

foreign key (*dept name*) *references department*);

- **insert into *instructor* values (10211, ’Smith’, ’Biology’, 66000)**
- **drop table *r;***
- **delete from *r;***
- **alter table *r add A D;***
- **alter table *r drop A;***

Schema of the university database.

classroom(*building*, *room number*, *capacity*)

department(*dept name*, *building*, *budget*)

course(*course id*, *title*, *dept name*, *credits*)

instructor(*ID*, *name*, *dept name*, *salary*)

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

teaches(*ID*, *course id*, *sec id*, *semester*, *year*)

student(*ID*, *name*, *dept name*, *tot cred*)

takes(*ID*, *course id*, *sec id*, *semester*, *year*, *grade*)

advisor(*s ID*, *i ID*)

time slot(*time slot id*, *day*, *start time*, *end time*)

prereq(*course id*, *prereq id*)

Database

- A database consists of multiple relations
- Information about an enterprise is broken up into parts

  instructor
  student
  advisor

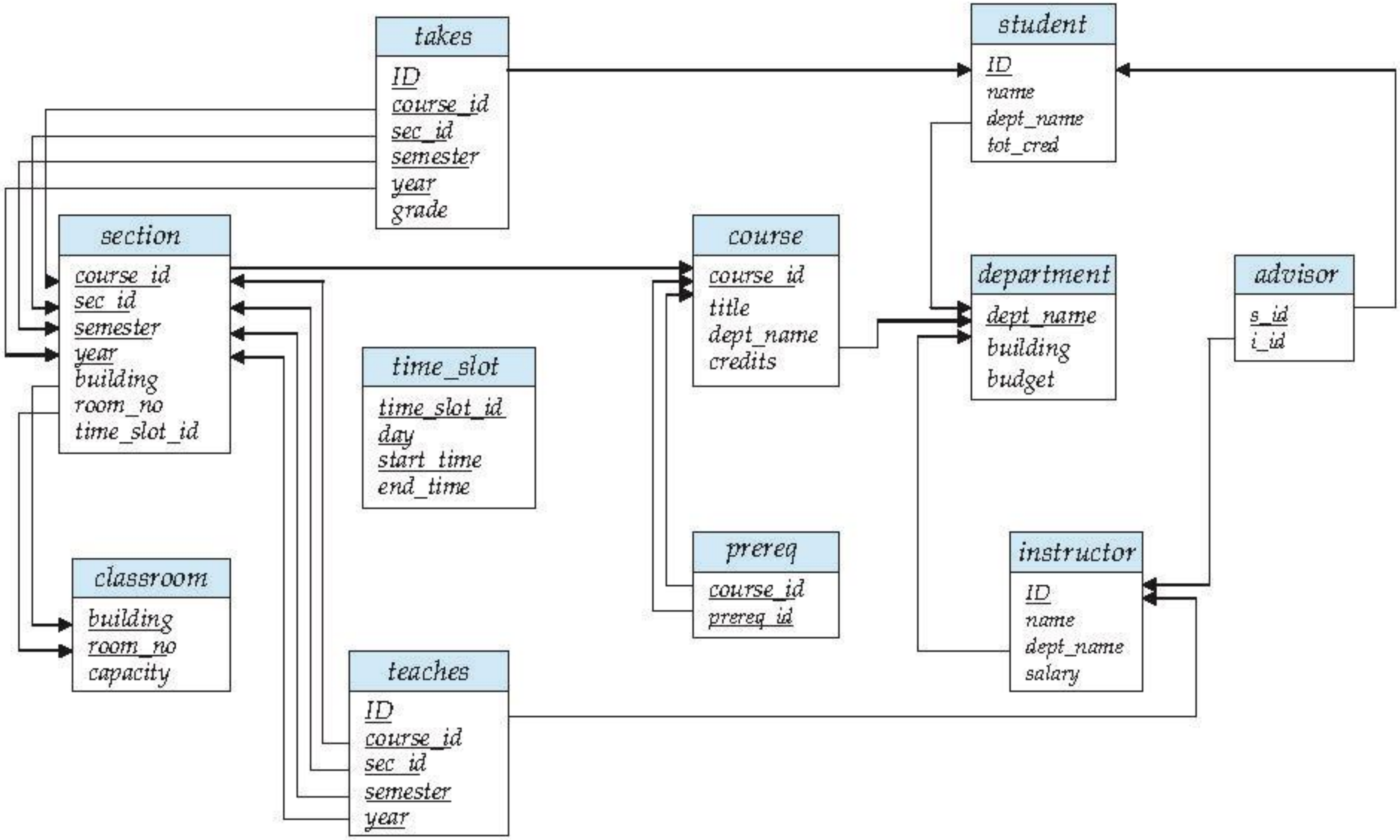
- Bad design:
  univ (*instructor -ID, name, dept_name, salary, student_Id*, ..)
  results in
  - repetition of information (e.g., two students have the same instructor)
  - the need for null values (e.g., represent an student with no advisor)
- Normalization theory (Chapter 7) deals with how to design “good” relational schemas

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
  - Example – *dept_name* in i*nstructor* is a foreign key from *instructor* referencing *department*

Schema Diagram for University Database



Relational Query Languages

- 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

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

- Select tuples with A=B and D > 5
  - $\sigma_{A=B \text{ and } D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Project Operation – selection of columns (Attributes)

- Relation *r*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

-

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

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

Union 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
α	2
β	1
β	3

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

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

- *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

Cartesian-product – naming issue

- Relations *r*, *s*:

A	B
α	1
β	2

r

B	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *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 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

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

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
  - $r \bowtie s$

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

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

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)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the *two* input relations.
– (Set Difference)	$\Pi_{name}(instructor)$ -- $\Pi_{name}(student)$
	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