Chapter 2: Intro to Relational Model

©Silberschatz, Korth and Sudarshan

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)

Attribute Types

- The set of allowed values for each attribute is called the **domain** or **data type** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
  - E.g., integer values
  - E.g., not address (street, city, zip code, state, country)
- The special value **null** is a member of every domain
  - Means *unknown* or *not applicable*
- The null value causes complications in the definition of many operations
  - Will be detailed later

Relation Schema and Instance

- $A_1, A_2, \ldots, A_n$ are ***attributes names***

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

  Example:

  instructor = (*ID, name, dept_name, salary*)

- Formally, given sets $D_1, D_2, \ldots. D_n$ of domains a **relation *r*** (or **relation instance)** 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 often specified in tabular form
  - Caveat: being a set, the tuples of the relation do not have any order defined as implied by the tabular representation
- An element ***t*** of ***r*** is a *tuple*, represented as a *row* in a table

Alternative Definitions

- A **<u>relation schema</u>** is often defined as a list of attribute-domain pairs
  - That is the data types of each attribute in the relation are considered as part of the relation schema
- Tuples are sometimes defined as functions from attribute names to values (order of attributes does not matter)
  - E.g., t(name) = ‘Bob’
- A relation **r** can be specified as a function
  - D_1 x D_2 x … x D_n -> *{true, false}*
  - **t** = (a_1, a_2, …, a_n) is mapped to *true* if **t** is in **r** and to *false* otherwise
- These alternative definition are useful in database theory
  - We will stick to the simple definition!

Relations are Unordered

- A relation is a **set** -> the elements of a set are not ordered per se
- From a practical perspective:
  - 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

Database

- A **database schema** *S* consists of multiple relation schema
- A **database instance** *I* for a schema S is a set of relation instances
  - One relation for each relation schema in S
- Information about an university 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 many null values (e.g., represent a student with no advisor or salary)
- Normalization theory (Chapter 7) deals with how to design "good" relational schemas avoiding these problems

Bad Design Example Revisited

- **Example:** Changing the budget of the 'Physics' department
  - Updates to many rows!
    - Easy to break **integrity**
    - If we forget to update a row, then we have multiple budget values for the physics department!
- Example: Deleting all employees from the 'Physics' department
  - How to avoid deleting the 'Physics' department?
  - Dummy employee's to store departments?
    - This is bad. E.g., counting the number of employees per department becomes more involved

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

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 (no subset of K is also a superkey)

  Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - which one? -> domain specific design choice
- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation

Keys

- Formally, a set of attributes $K \subseteq R$ is a superkey if for every instance r of R holds that
  - $\forall t, t' \in r: t.K = t'.K \Rightarrow t = t'$
- A superkey *K* is called a candidate key iff
  - $\forall K' \subset K: K'$ is not a superkey
- A foreign key constraint FK is quartuple (R, K, R', K') where R and R' are relation schemata, $K \subseteq R$, K' is the primary key of R', and $|K| = |K'|$
- A foreign key holds over an instance {r, r'} for {R,R'} iff
  - $\forall t \in R: \exists t' \in R': t.K = t'.K'$

Schema Diagram for the University Database

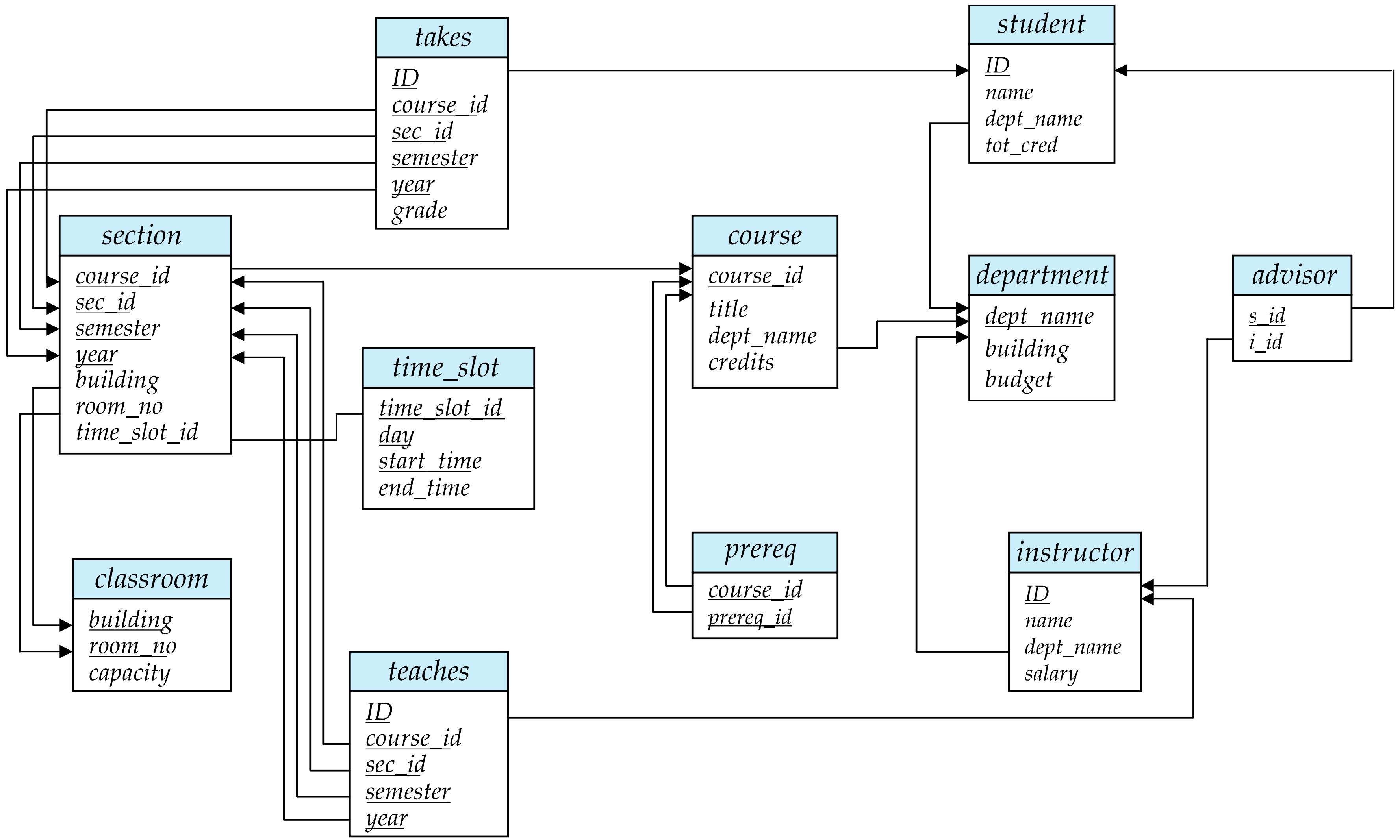

Figure 2.01

instructor

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 2.02

course

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

©Silberschatz, Korth and Sudarshan

Figure 2.03

prereq

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

©Silberschatz, Korth and Sudarshan

Figure 2.05

department

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

©Silberschatz, Korth and Sudarshan

Figure 2.06

section

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 2.07

teaches

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Recap

- **Database Schema** (or short schema)
  - Set of **relation schemata**
    - List of **attribute names**
- **Database Instance** (or short database)
  - Set of **relations instances**
    - Set of **tuples**
      - List of **attribute values**
- **Integrity Constraints**
  - **Keys** (Super-, Candidate-, Primary-)
    - For identifying tuples
  - **Foreign keys**
    - For referencing tuples in other relations

End of Chapter 2