CS 4092 Database Design and Development (DDD)

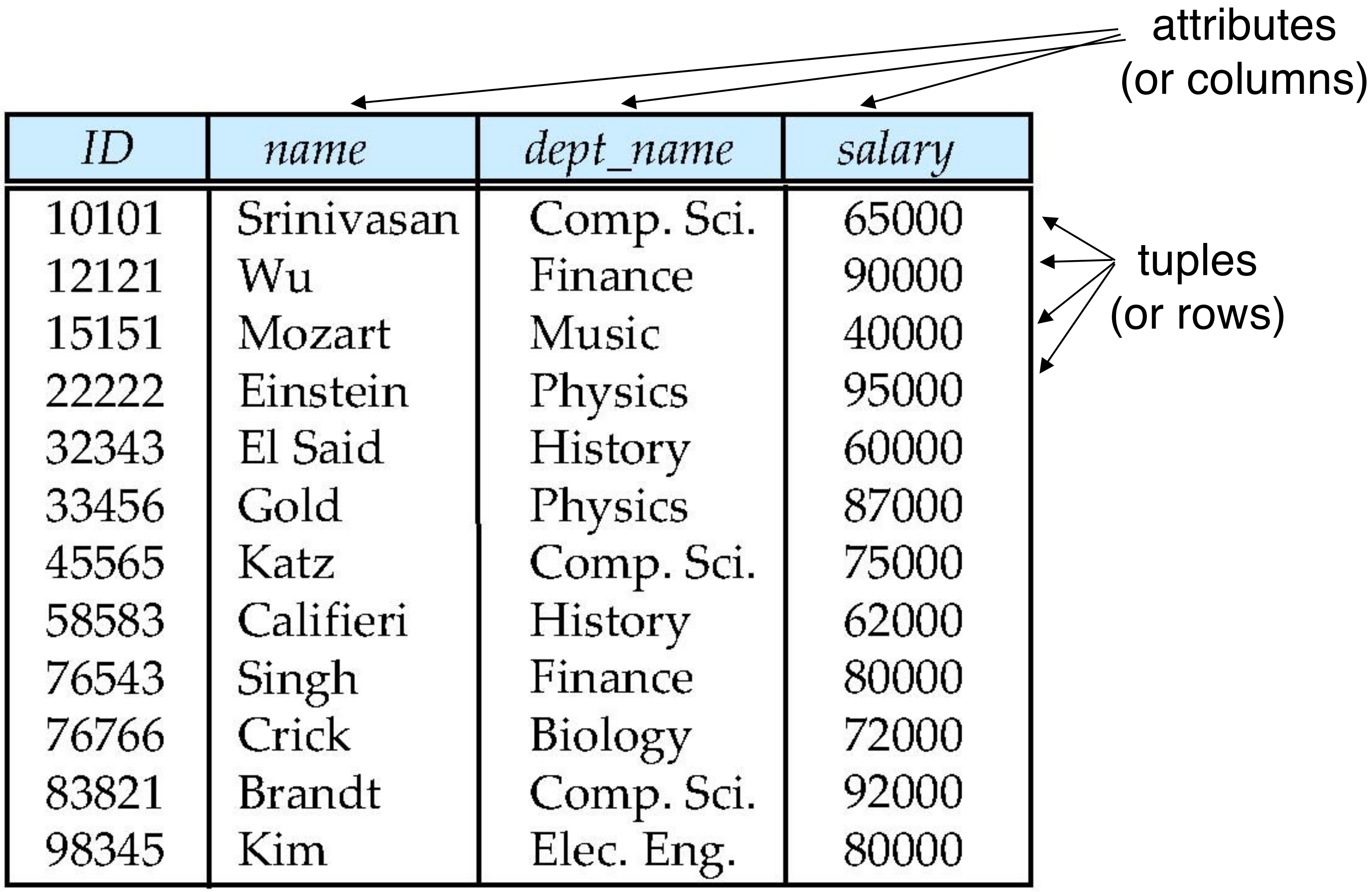
02: Relational Model

Seokki Lee

Slides are adapted from:

Database System Concepts, 6th & 7th Ed. ©Silberschatz, Korth and Sudarshan

Example of a Relation



attributes
(or columns)

tuples
(or rows)

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

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
  - Integer values?
  - Address?

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
  - Integer values
  - 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 x D_2 x … x D_n

  Thus, a relation is a **set** of ***n*-tuples** $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$

Relation Schema and Instance

- The current values (**relation instance**) of a relation are often specified in tabular form
  - Ordered or Unordered?

Relation Schema and Instance

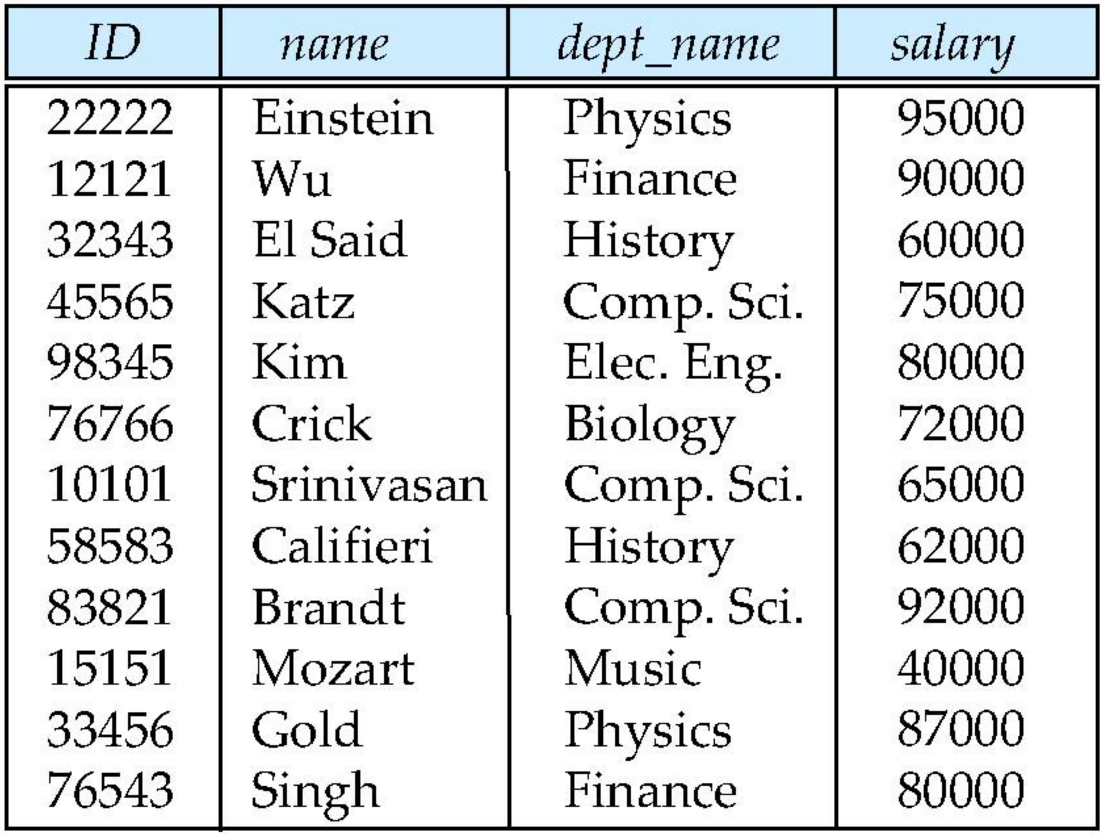
- The current values (**relation instance**) of a relation are often specified in tabular form
  - Caveat: being a set, the tuples of the relation do not have any order defined as implied by the tabular representation
- An element ***t*** of ***r*** is a tuple, represented as a row in a table

Alternative Definitions

- Tuples are sometimes defined as functions from attribute names to values (order of attributes does not matter)
  - E.g., t.name = ‘Bob’ or t(name) = ‘Peter’
- A relation **r** can be specified as a function
  - $D_1 \text{ x } D_2 \text{ x} \ldots \text{x } D_n \rightarrow \{true, false\}$
  - **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
- From a pratical perspective:
  - Order of tuples is irrelevant (tuples may be stored or returned in an arbitrary order) → “CS7071”
- 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 enterprise is broken up into parts

  instructor
  student
  advisor

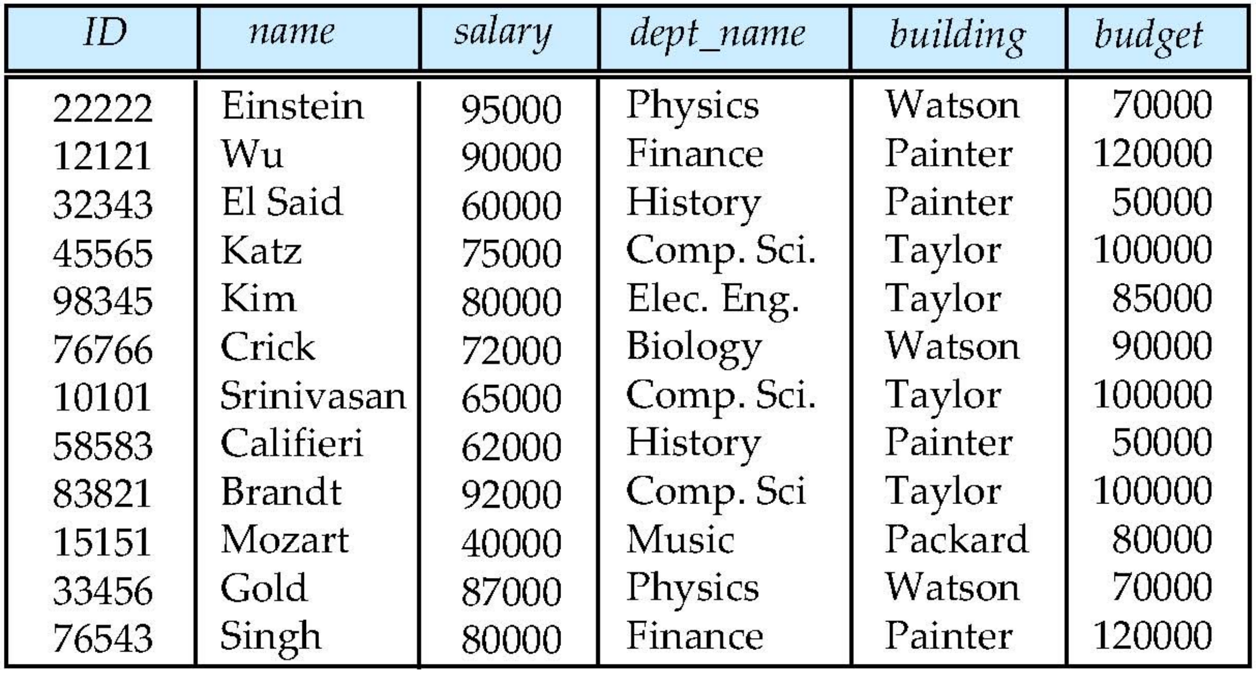
- Bad design:

  univ (*instructor -ID, name, dept_name, salary, student_Id*, ..)

  - repetition of information (e.g., two students have the same instructor)
  - the need for many null values (e.g., represent an student with no advisor)
- Normalization theory (Chapter 7) deals with how to design "good" relational schemas avoiding these problems

Bad Design Example Revisited

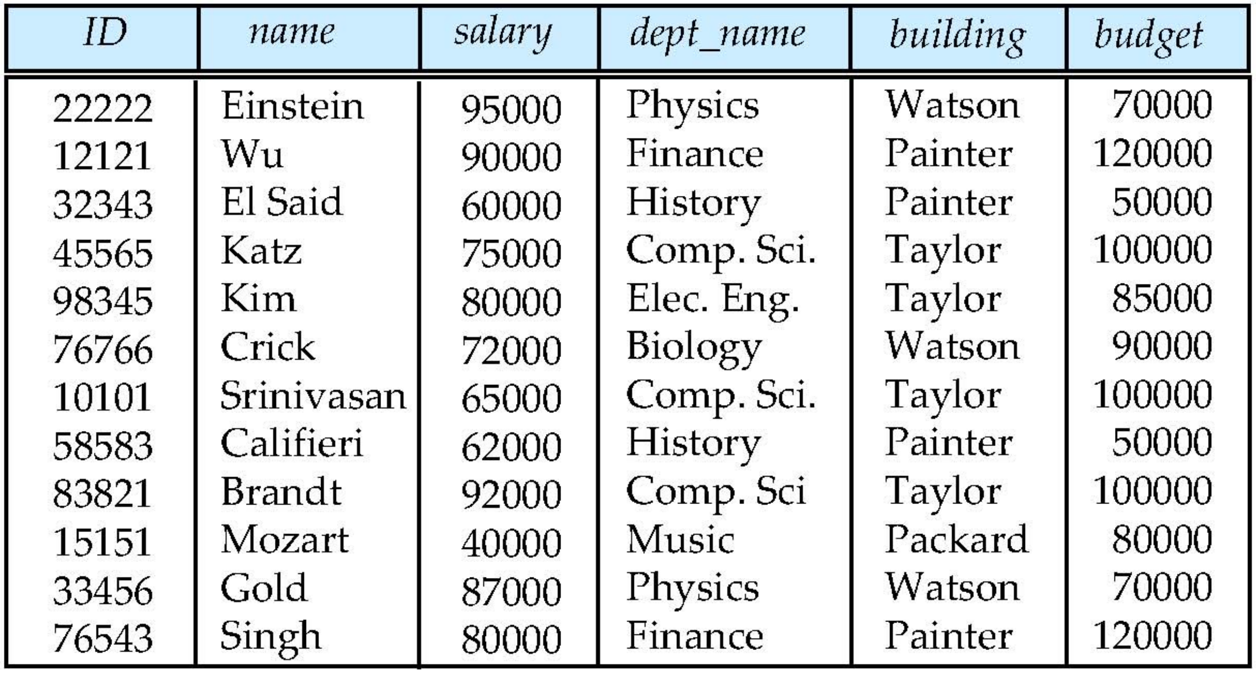
- **Changing** the budget of the 'Physics' department
  - Updates to many rows!
    - Easy to break **integrity**
    - If we forget to update a row, then we have multiple budget values for the physics department!



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

Bad Design Example Revisited

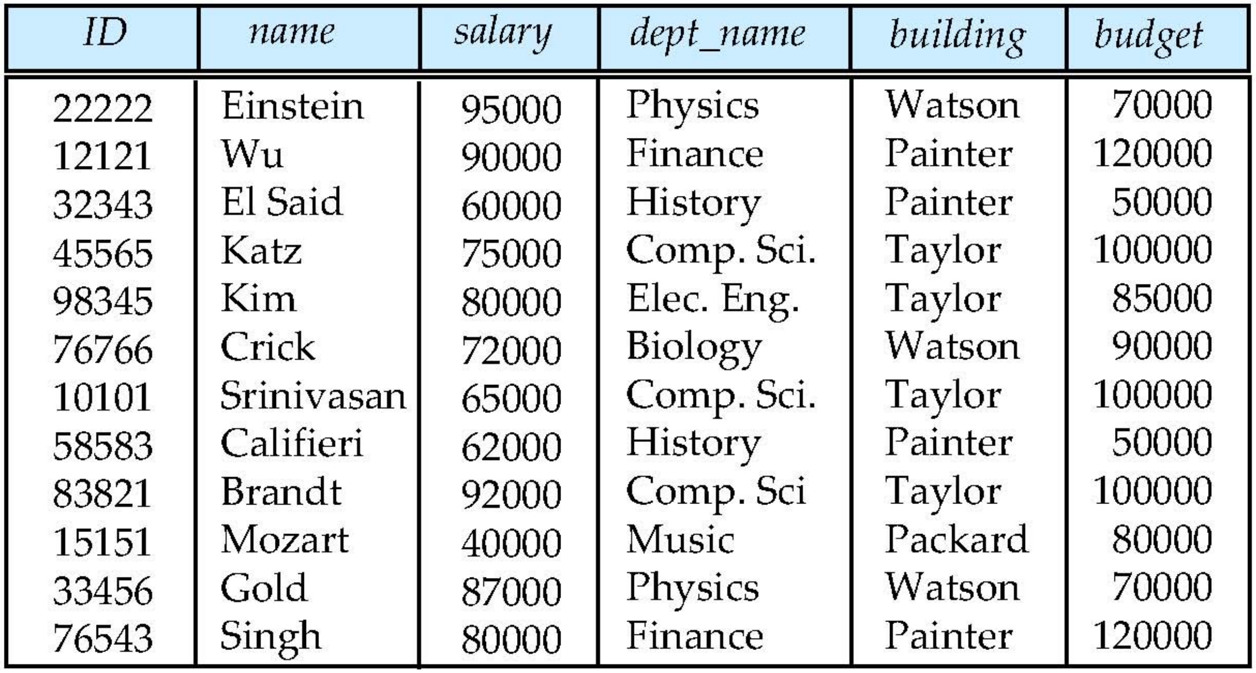
- **Deleting** all employees from the ‘Physics’ department



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

Bad Design Example Revisited

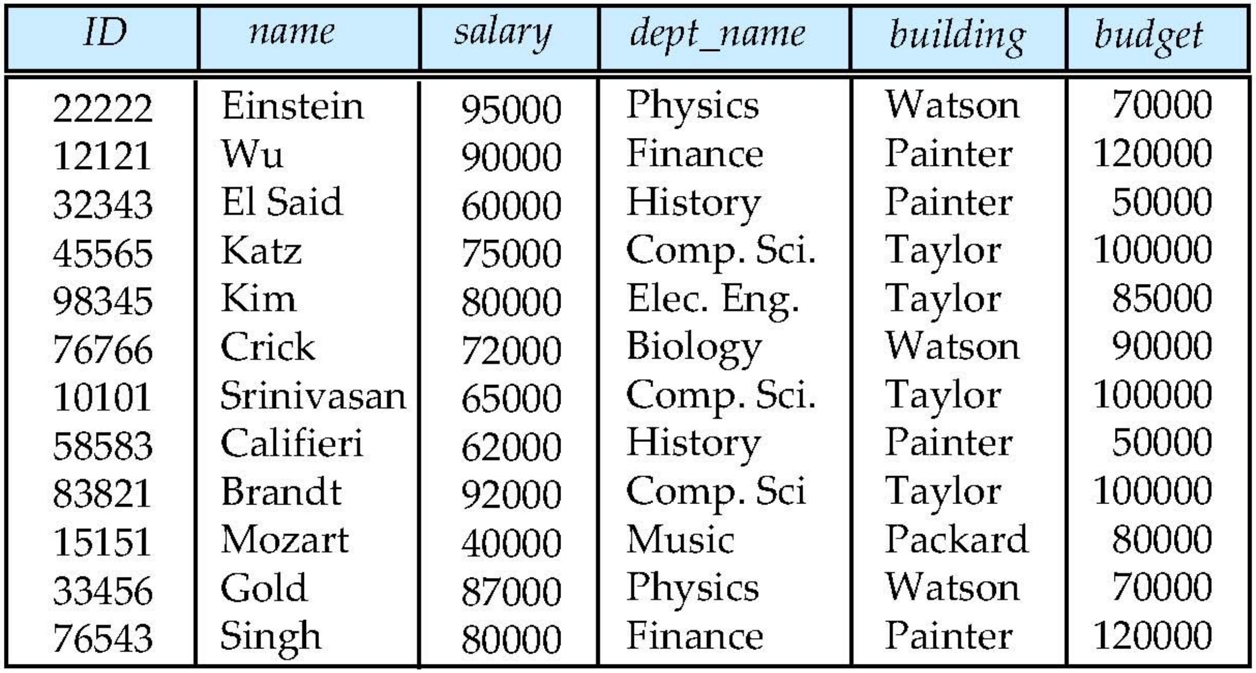
- **Deleting** all employees from the 'Physics' department
  - How to avoid deleting the 'Physics' department?



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

Bad Design Example Revisited

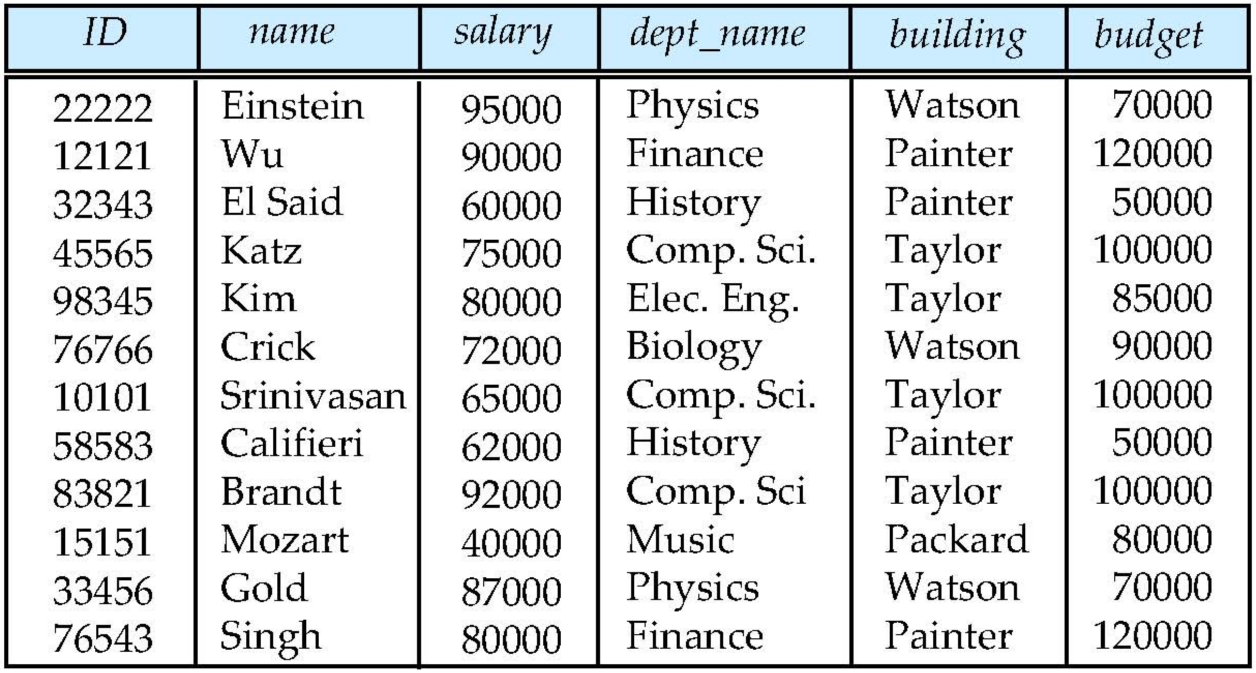
- **Deleting** all employees from the ‘Physics’ department
  - How to avoid deleting the ‘Physics’ department?
  - Dummy employee’s to store departments?



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

Bad Design Example Revisited

- **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

- Within a given relation, tuples should be distinguished.
- Typically expressed in terms of their attributes
  - Values of attribute values can ***uniquely identify*** each tuple.
  - In other words, ***no two tuples*** in a relation are allowed to have ***exactly the same*** value for all attributes.

Keys

- Superkey
- Candidatekey
- Primarykey

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)

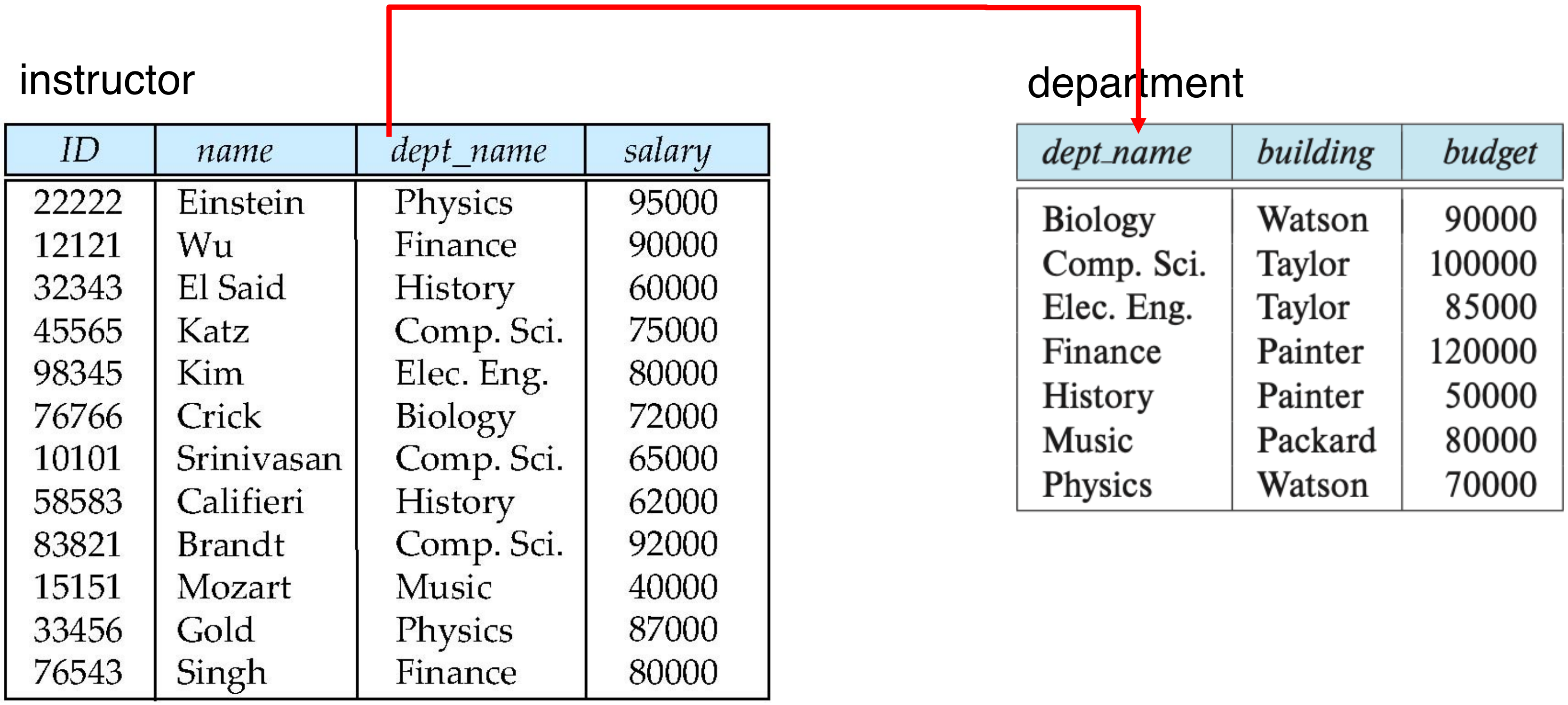
  Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - which one? → domain specific design choice

Keys

- 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
  - $\forall t \in R : \exists t' \in R' : t.K = t'.K'$

Keys

- **Foreign key** constraint: value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation



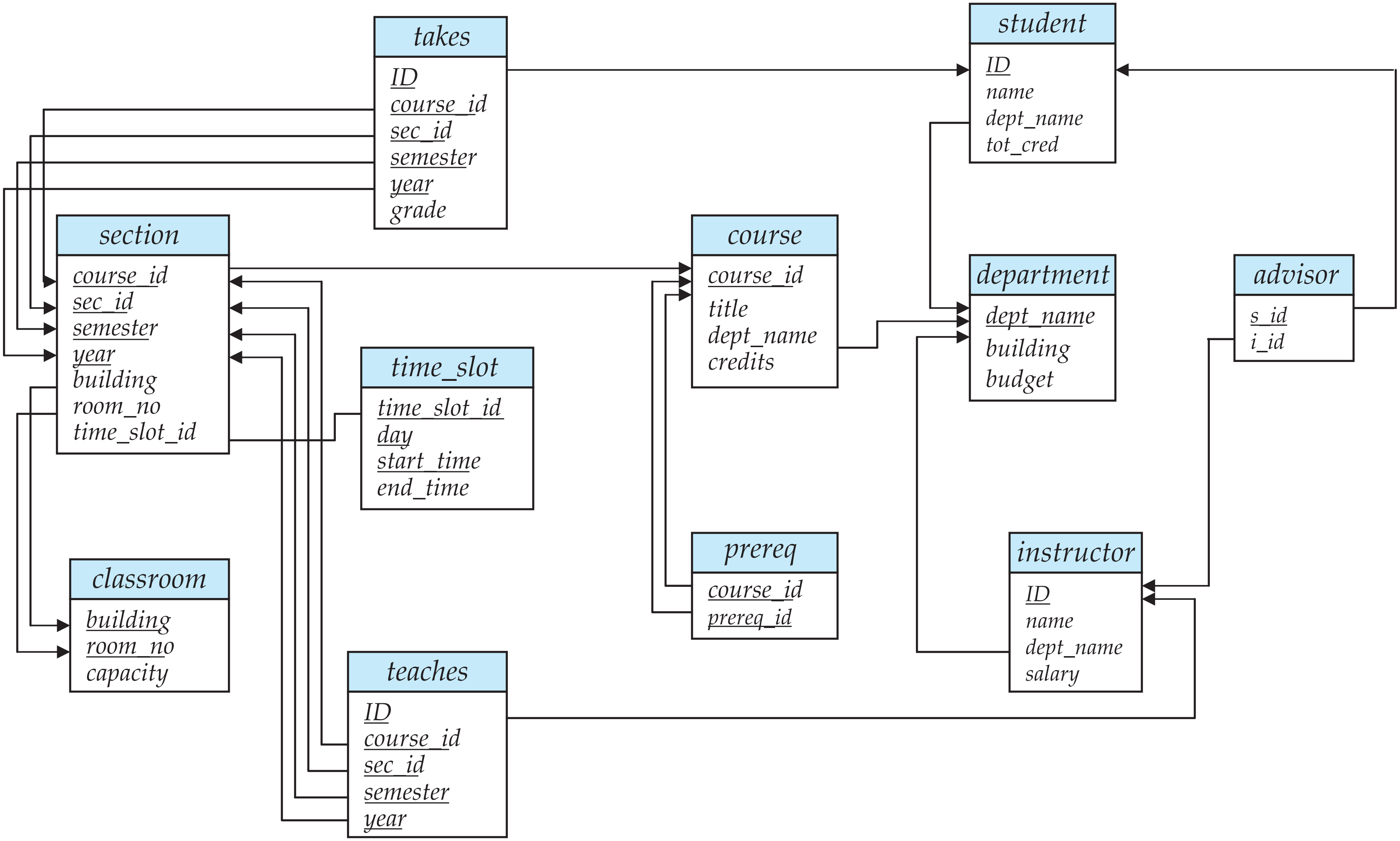
instructor

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

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

Schema Diagram for the University Database

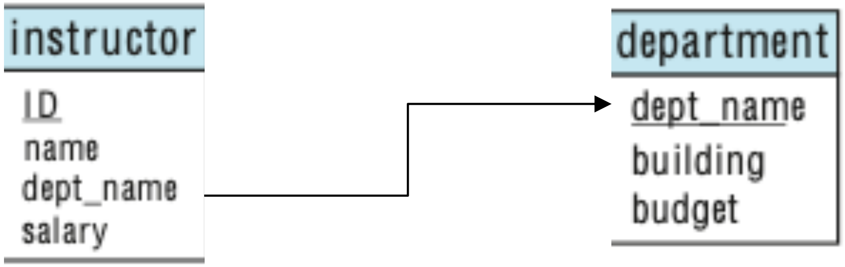


Recap

- **Database Schema** (or short schema)
  - Logical design of database
  - Set of **relation schemata**
    - List of **attribute names**
- **Database Instance** (or short database)
  - Set of **relations instances**
    - Set of **tuples**
      - List of **attribute values**

Recap

- **Integrity Constraints**
  - **Keys** (Super-, Candidate-, Primary-)
    - For identifying tuples (no two tuples are allowed)
    - A (set of) attribute value(s)
    - Superkey: sufficient to identify a tuple uniquely
    - Candidate key: superkeys that are minimal
    - Primary key: chosen from a candidate key
  - **Foreign keys**
    - For referencing tuples in other relations



Corresponding Reading Materials

- Relational Model
  - Database System Concepts 7th Edition
    - Chapter 2