Chapter 2: Intro to Relational Model

Edited by Radhika Sukapuram. Original slides by

Database System Concepts, 6th Ed.

Relational Model Concepts

- The relational Model of Data is based on the concept of a Relation.

- A Relation is a mathematical concept based on the ideas of sets.

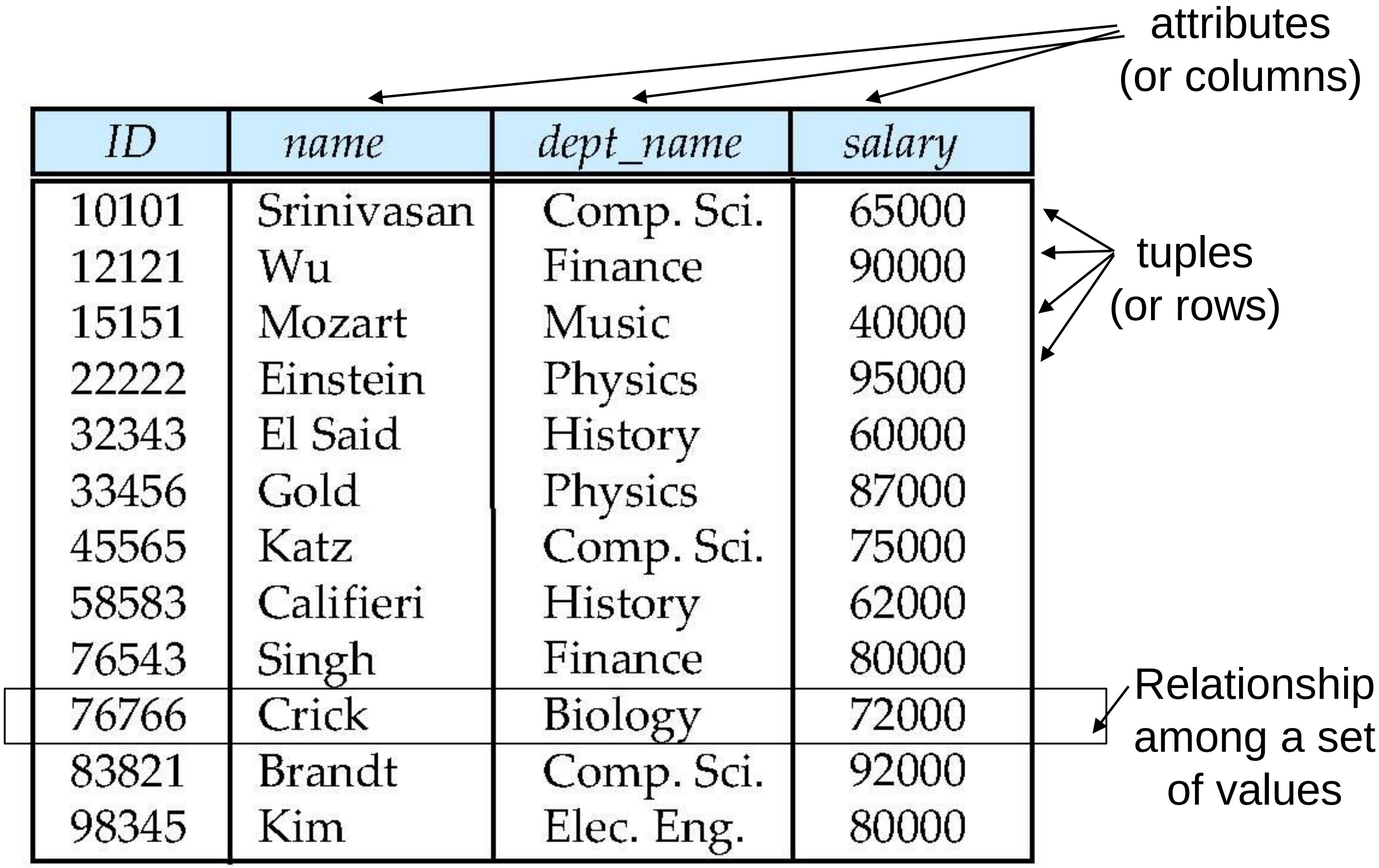
- The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations.

- The model was first proposed by Dr. E.F. Codd of IBM in 1970 in the following paper:
  “A Relational Model for Large Shared Data Banks,"
  Communications of the ACM, June 1970.

INFORMAL DEFINITIONS

- RELATION: A table of values
  - A relation may be thought of as a **set of rows**.
  - A relation may alternately be thought of as a **set of columns**.
  - Each row represents a fact that corresponds to a real-world **entity** or **relationship**.
  - Each row has a value of an item or set of items that uniquely identifies that row in the table.
  - Sometimes row-ids or sequential numbers are assigned to identify the rows in the table.
  - Each column typically is called by its column name or column header or attribute name.
  - Columns represent a property
- A relational database is a collection of two-dimensional tables
- Formal constraints of functional dependency and multi valued dependency used to develop a relational database

Example of a Relation



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

attributes
(or columns)

tuples
(or rows)

Relationship
among a set
of values

Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain. Indicating that the value is “unknown” or unspecified
- The null value causes complications in the definition of many operations

Example of a Relation

Employee

Empid	Name	Level	DOJ	Manager
101412	John	M3	4/10/98	101667
102235	Nancy	M4	1/23/01	101412
101398	Mike	S1	8/15/95	101667
101667	Jeff	M2	6/2/96	100351
103893	Cindy	M3	7/17/95	101284
101116	Rahul	S2	2/20/00	101412
102739	Scott	C1	4/13/01	101667

FORMAL DEFINITIONS

- Formally, given sets D_1, D_2, …. D_n a **relation** r 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, …, a_n) where each $a_i \in D_i$
- Example: If
  - *customer_name* = {Jones, Smith, Curry, Lindsay, …}

    /* Set of all customer names */
  - *customer_street* = {Main, North, Park, …} /* set of all street names*/
  - *customer_city* = {Harrison, Rye, Pittsfield, …} /* set of all city names */

  Then r = { (Jones, Main, Harrison),
  (Smith, North, Rye),
  (Curry, North, Rye),
  (Lindsay, Park, Pittsfield) }

  is a relation over

  customer_name x customer_street x customer_city

Relation Schema

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

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

  Example:

  Customer_schema = (*customer_name, customer_street, customer_city*)

- *r*(*R*) denotes a *relation r* on the *relation schema R*

  Example:

  customer (Customer_schema)

Relation Schema and Instance

- Formally, given sets D_1, D_2, …. D_n a **relation** ***r*** is a subset of

  D_1 x D_2 x … x D_n

  Thus, a relation is a set of tuples (a_1, a_2, …, 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

CHARACTERISTICS OF RELATIONS

- Notation:
  - We refer to **component values** of a tuple t by $t[A_i] = v_i$ (the value of attribute A_i for tuple t).
  - Similarly, $t[A_u, A_v, ..., A_w]$ refers to the subtuple of t containing the values of attributes A_u, A_v, ..., A_w, respectively.

Relational Model

Table - Terminology

In this document	Formal Terms	Many Database Manuals
Table	Relation	Table
Column	Attribute	Field
Row	Tuple	Record

Relational Model

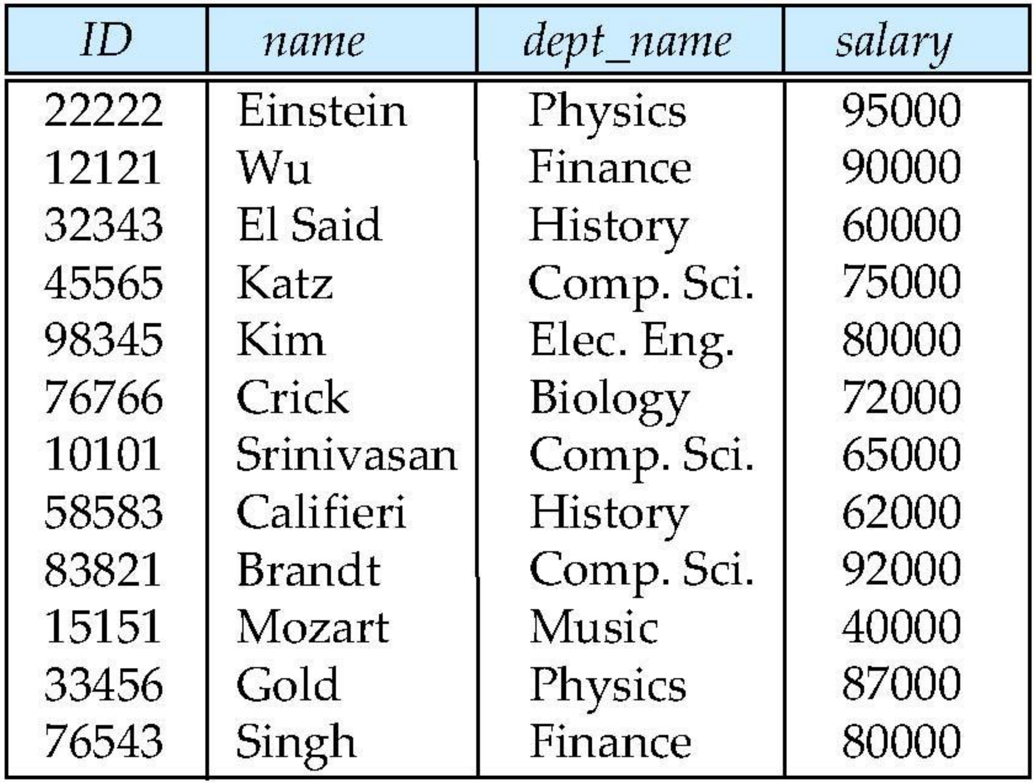
- Tables, Columns and Rows
- Relationships and Keys
- Data Integrity
- Normalization

Relational Model

- **Salient features of a relational table**
  - Values are atomic. A special **null** value is used to represent values that are unknown or inapplicable to certain tuples.
  - Column values are of the same kind (Domain)
  - Each Row is unique
  - Sequence of columns is insignificant
  - Sequence of rows is insignificant
  - Each column must have a unique name
  - Relationships and Keys
    - Keys - Fundamental to the concept of relational databases
    - Relationship - An association between two or more tables defined by means of keys

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

Database

- A database consists of multiple relations
- Information about an enterprise is broken up into parts, with each relation storing one part of the information

  account : stores information about accounts
  depositor : stores information about which customer owns which account
  customer : stores information about customers

- Storing all information as a single relation such as
  bank(*account_number, balance, customer_name*, ..)
  results in
  - repetition of information
    - e.g.,if two customers own an account (What gets repeated?)
  - the need for null values
    - e.g., to represent a customer without an account
- Normalization theory (Chapter 7) deals with how to design relational schemas

Relational Integrity Constraints

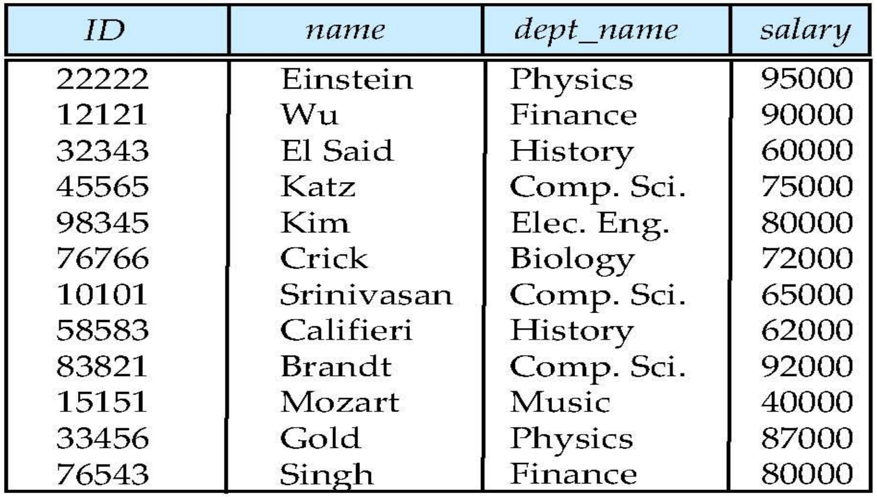
- Constraints are *conditions* that must hold on *all* valid relation instances. There are three main types of constraints:

1. **Key** constraints
2. **Entity integrity** constraints
3. **Referential integrity** constraints

Keys

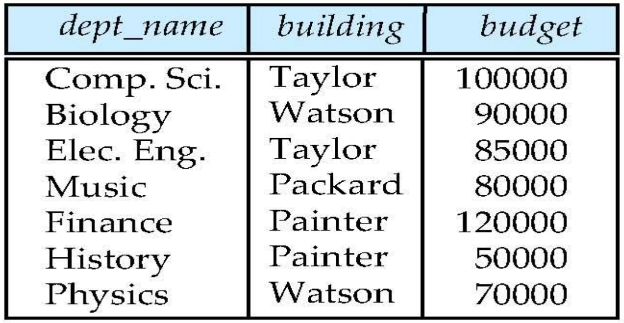
- Let R: set of attributes in the schema of relation *r*
- Let K $\subseteq$ R
- *K* is a **superkey** of *r* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID,name} are both superkeys of *instructor.*
- Superkey *K* is a **candidate key** if *K* is minimal.
  - No proper subset of K is a superkey
  - Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - The candidate key that is least likely to change is selected
- Keys are a property of a relation, not a tuple

Instructor and department relations



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

(a) The *instructor* table



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

(b) The *department* table

Entity Integrity

- **Relational Database Schema**: A set S of relation schemas that belong to the same database. S is the *name* of the **database**.

$$S = \{R_1, R_2, ..., R_n\}$$

- **Entity Integrity**: The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of r(R). This is because primary key values are used to *identify* the individual tuples.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- <u>Note:</u> Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity

- A constraint involving *two* relations (the previous constraints involve a *single* relation).
- Used to specify a *relationship* among tuples in two relations: the **referencing relation** and the **referenced relation**.
- Tuples in the *referencing relation* R_1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation* R_2. A tuple t_1 in R_1 is said to **reference** a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R_1.FK to R_2.

Referential Integrity Constraint

Statement of the constraint

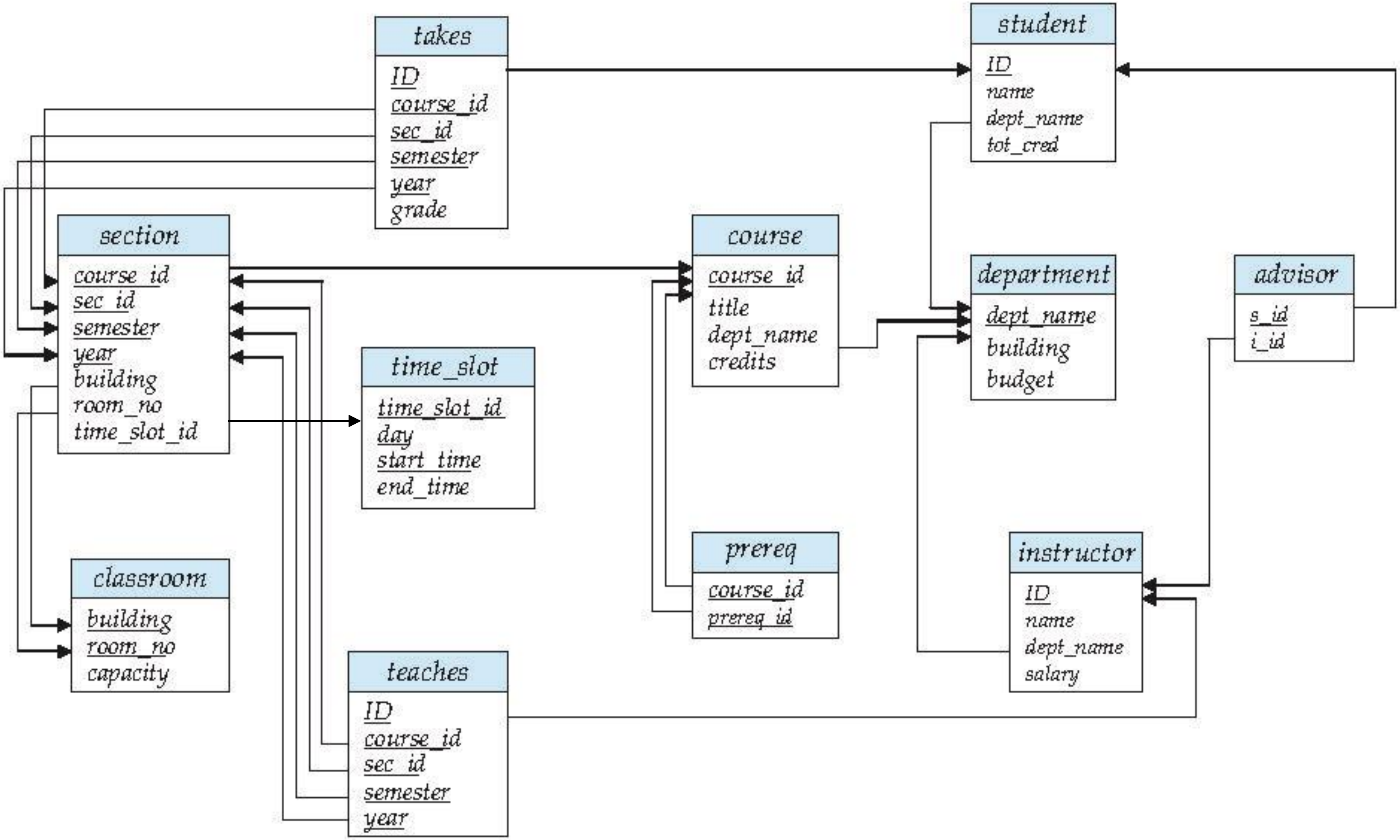
The value in the foreign key column (or columns) FK of the **referencing relation** R_1 can be either:

(1) a value of an existing primary key value of the corresponding primary key PK in the **referenced relation** $R_{2,}$, or..

(2) a null.

In case (2), the FK in R_1 should not be a part of its own primary key**.**

Schema Diagram for University Database



takes
ID
course_id
sec_id
semester
year
grade

student
ID
name
dept_name
tot_cred

section
course_id
sec_id
semester
year
building
room_no
time_slot_id

course
course_id
title
dept_name
credits

department
dept_name
building
budget

advisor
s_id
i_id

time_slot
time_slot_id
day
start_time
end_time

classroom
building
room_no
capacity

prereq
course_id
prereq_id

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

END HERE