Introduction to Database Systems

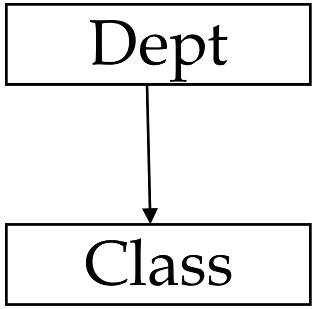
2023-Fall

2. Data Model

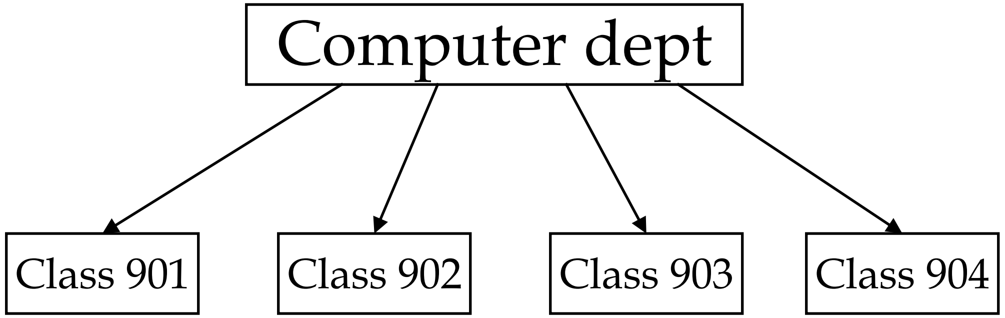
2.1 Hierarchical Data Model

Basic idea: because many things in real world are organized in hierarchy, hierarchical model manages to describe real world in a ***tree structure***.

- Record and field
- ***Parent-Child relationship (PCR):*** the most basic data relationship in hierarchical model. It expresses a **1:N** relationship between two record types.



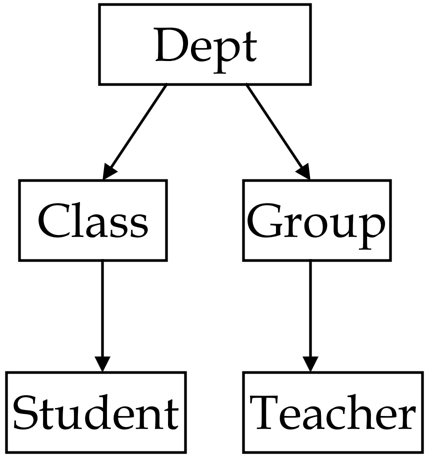
A PCR type



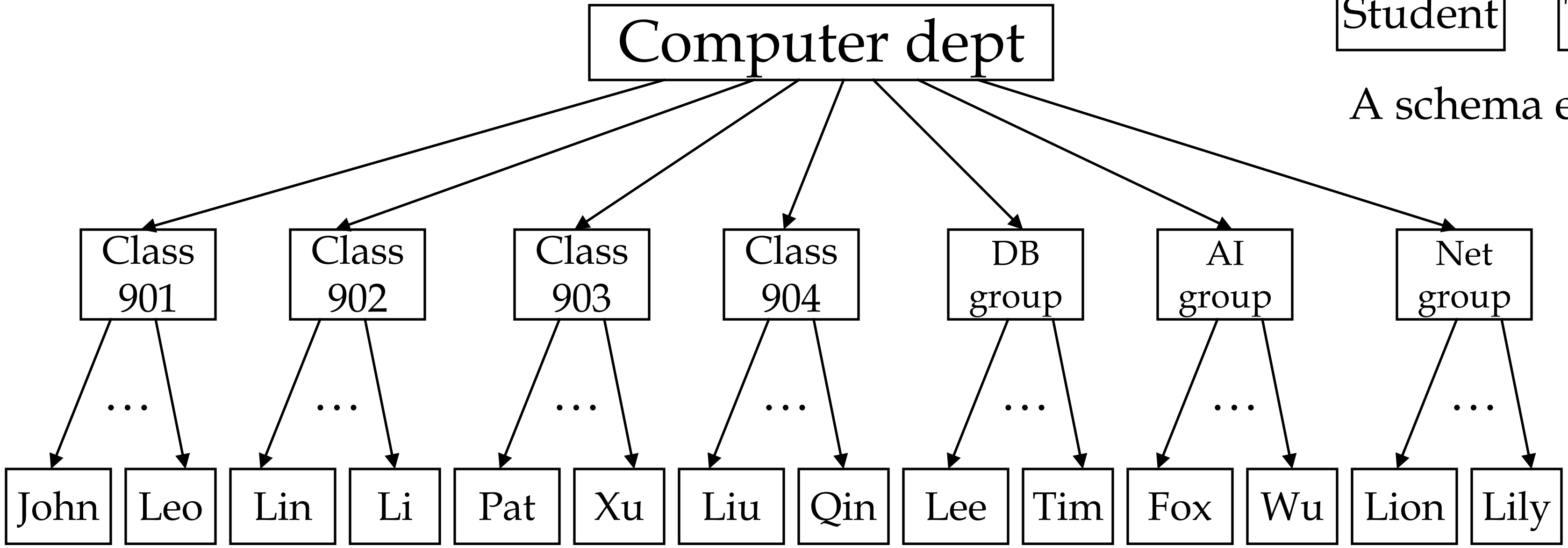
An instance of PCR

Hierarchical Data Schema

- A hierarchical data schema consists of PCRs.
- Every PCR expresses one 1:N relationship
- Every record type can only have one parent



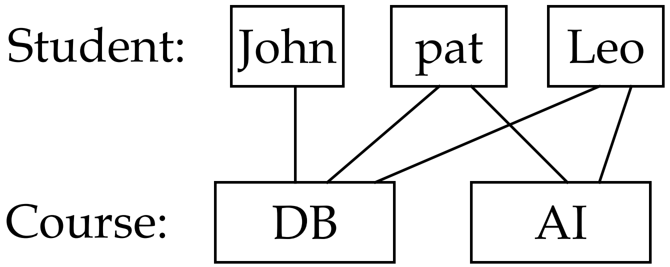
A schema example



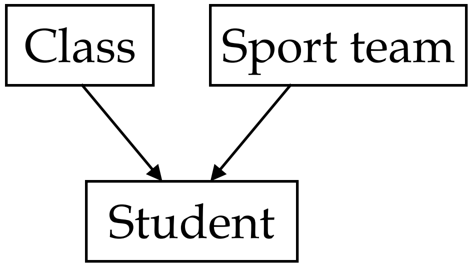
An instance of hierarchical data schema

Virtual Record

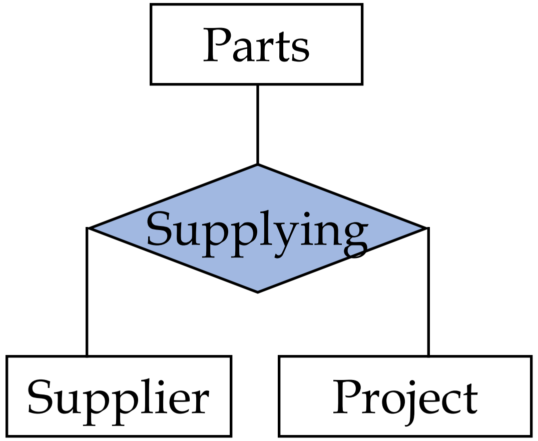
- In real world, many data are not hierarchical. It is hard to express them directly with PCR.
    - M:N relationship between different record types
    - A record type is the child of more than two PCRs.
    - N-ary relationship.



A M:N relationship



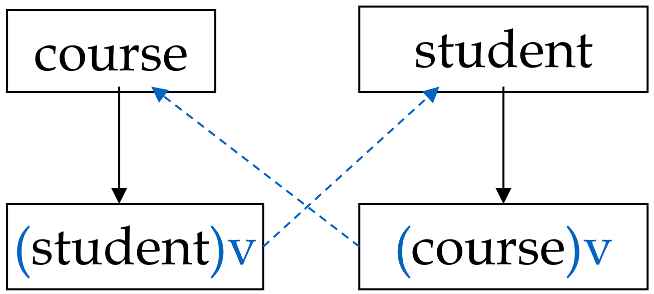
Multi parents



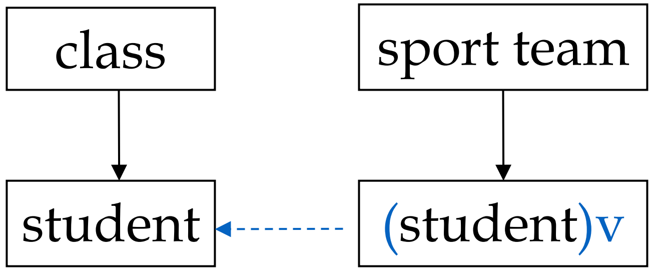
Ternary relationship

Virtual Record

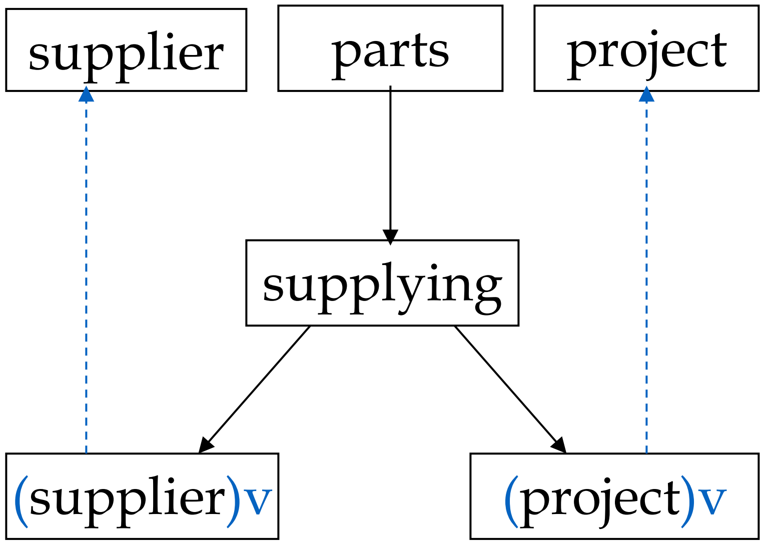
- To avoid redundant, virtual record is introduced to express above relationships. It is a pointer in fact.



M:N expressed with virtual record type



Multi parent expressed with virtual record type

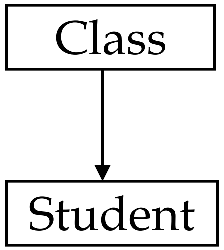


Ternary relationship expressed with virtual record type

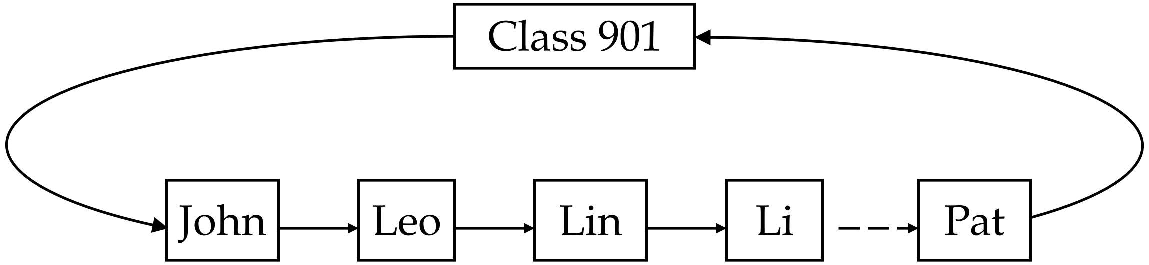
2.2 Network Data Model

- The basic data structure is "***set***", it represent a 1:N relationship between things in real world. "1" side is called owner, and "N" side is called member.
- One record type can be the owner of multi sets, and also can be the member of multi sets. Many sets form a network structure to express real world.
- It breaks through the limit of hierarchical structure, so can express non-hierarchical data more easy.
- Record and data items: data items are similar as field in hierarchical model, but it can be vector.
- ***Set*** : express the 1:N relationship between two record types.
- ***LINK record type***: used to express self relationship, M:N relationship and N-ary relationship.

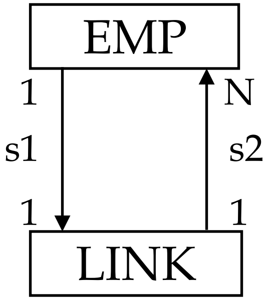
Example of Network Data Schema



C-S set

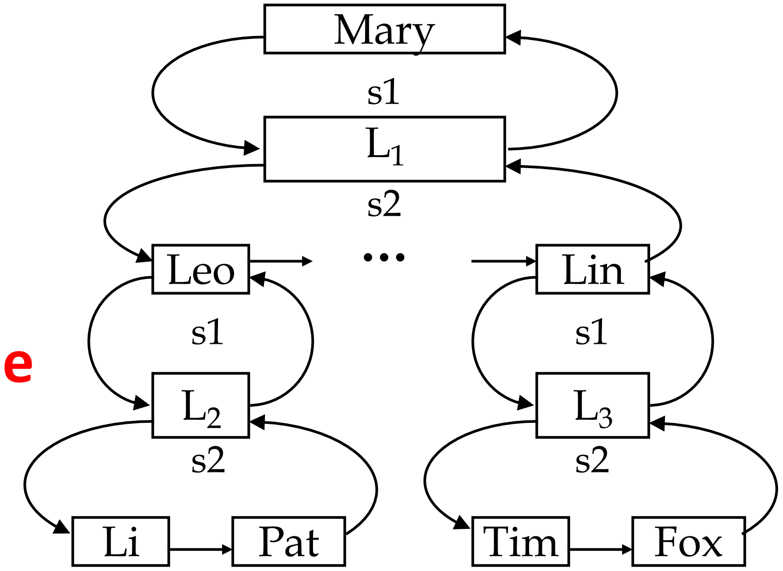


A value of C-S set



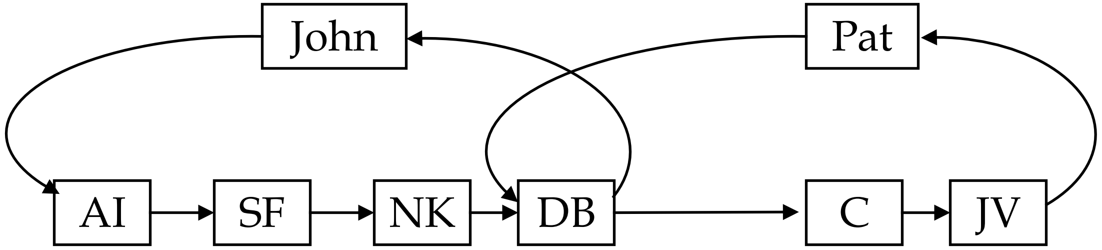
LINK record type

Leader relationship
between EMP itself

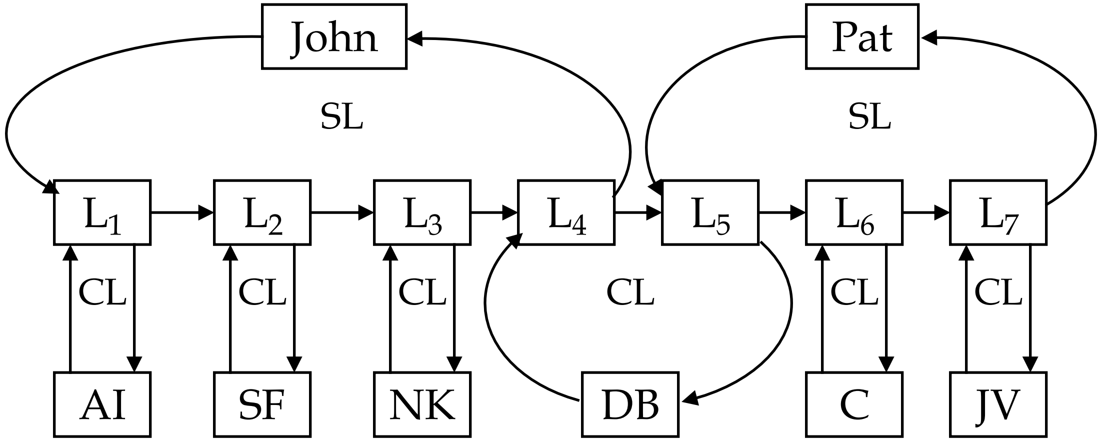


A value of EMP self link

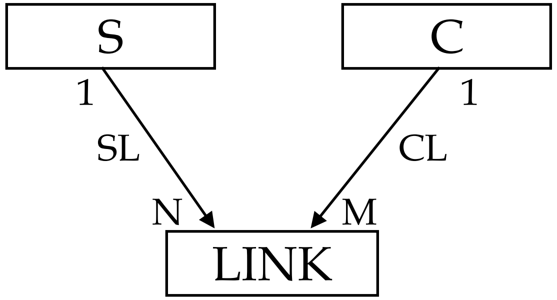
Example of Network Data Schema



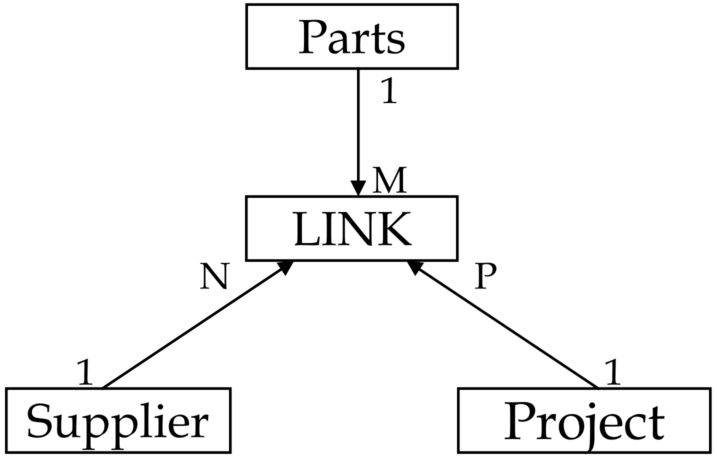
A value of M:N relationship between student and course



A value of M:N relationship between student and course



M:N relationship between student and course



Ternary relationship

2.3 Relational Data Model

- The basic data structure is “table”, or relation. The things and the relationships between them in real world are *all* expressed as tables, so it can be researched in strict mathematic methods. It raises the database technology to a theory height. Its features:

✓Based on set theory, high abstract level

✓Shield all lower details, simple and clear, easy to understand

✓Can establish new algebra system——relational algebra

✓Non procedure query language——SQL

✓***Soft link*** ——the essential difference with former data models

Understand *Soft link*

student

S#	SNAME	AGE	…

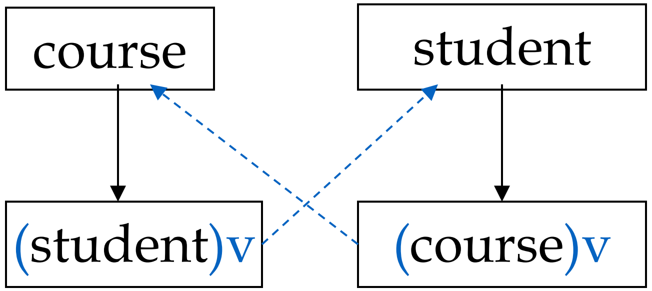
elective

S#	*C#*	*GRADE*

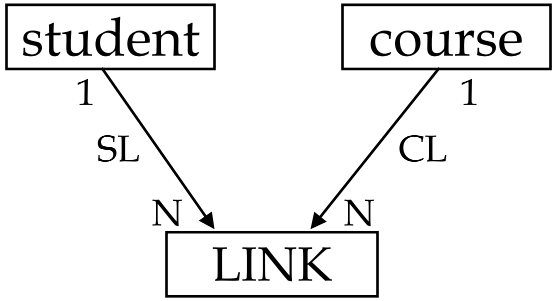
course

C#	CNAME	SCR	…

Compare with



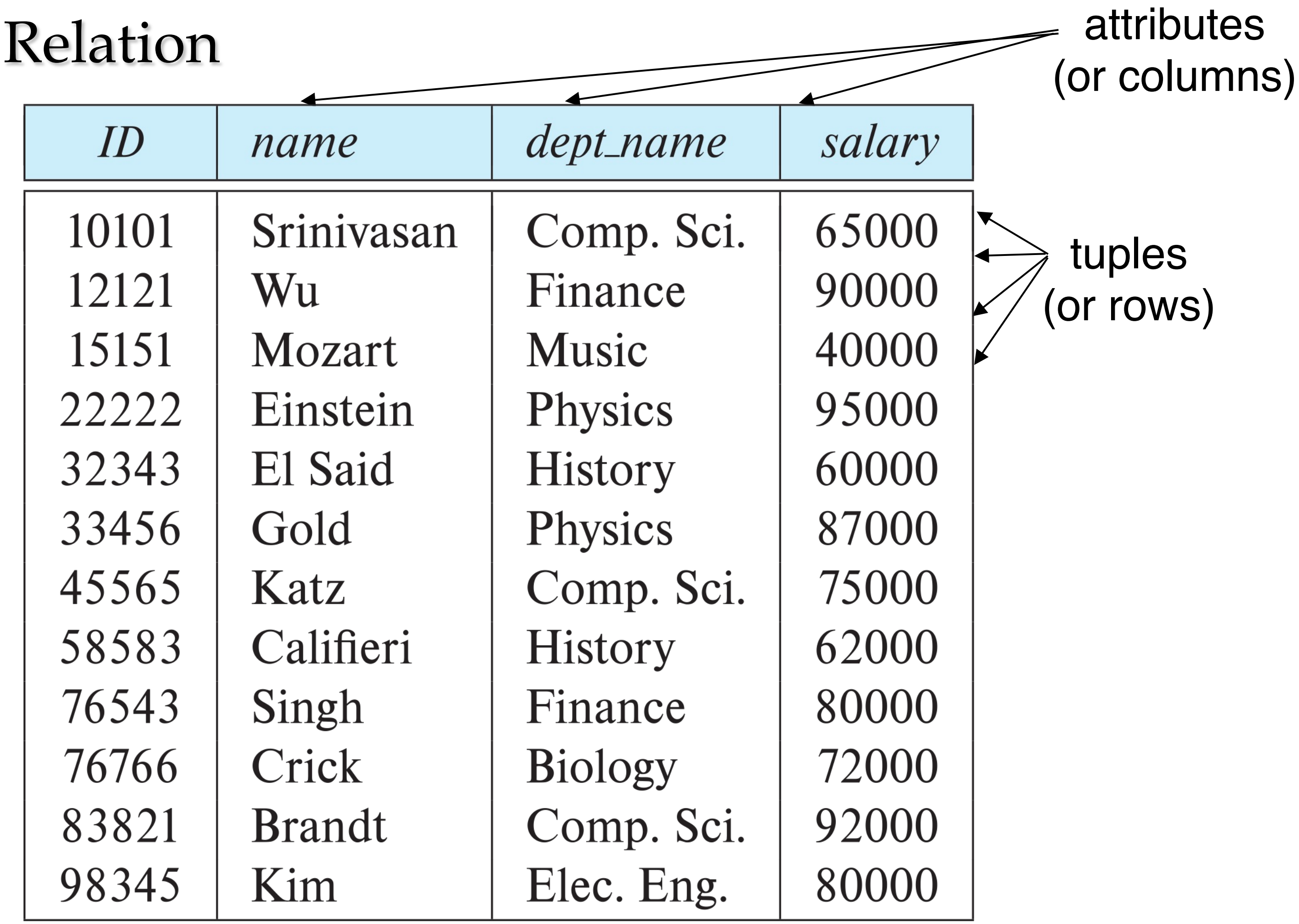
Expressed in hierarchical model



Expressed in network model

Relational Data Model

Example of a *Instructor* 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

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*)
- A relation instance *r* defined over schema *R* is denoted by *r* (*R*).
- The current values a relation are specified by a table
- An element ***t*** of relation ***r*** is called a *tuple* and is represented by a *row* in a table

Attributes

- 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

Attributes and Domain

- The features of an entity in real world are expressed as **attributes** in relational model

  E.g. a student can be described with the attributes such as *name, sid, gender, age, birthday, nationality,* etc.
- The value scope of an attribute is called its domain.
  - Atomic data
  - Null

Relation and Tuple

- An entity of real world can be expressed as one or more than one relations.
- A relation is a N-ary relationship defined on all of its attribute domain.

  Suppose a relation R with attributes $A_1, A_2, \ldots A_n$, the corresponding domains are $D_1, D_2, \ldots D_n$, then R can be expressed as:

  $R = (A_1/D_1, A_2/D_2, \ldots A_n/D_n)$, or

  $R = (A_1, A_2, \ldots A_n)$
- This is called the schema of R, and n is the number of attributes, called the degree of R. $A_i (1 \leq i \leq n)$ is attribute name.

Relation and Tuple

- An instance (value) of R can be expressed as r or $r(\mathrm{R})$, it is a set of n-tuple:

  $r = \{t_1, t_2, \ldots, t_m\}$

  every ***tuple*** *t* can be expressed as:

  $t = <v_1, v_2, \ldots, v_n>, v_i \in \mathrm{D}_\mathrm{i}, 1 \leq i \leq \mathrm{n}$

  that is:

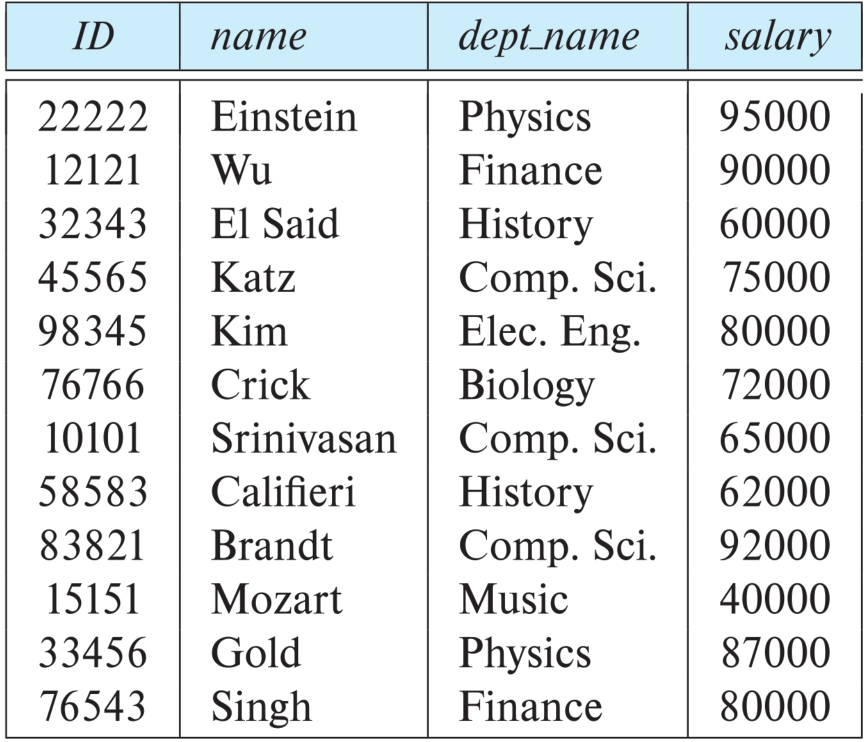
  $t \in \mathrm{D}_1 \times \mathrm{D}_2 \times, \ldots, \times \mathrm{D}_\mathrm{n}, 1 \leq i \leq \mathrm{n}$ (Cartesian Product)

  that is:

  $r \subseteq \mathrm{D}_1 \times \mathrm{D}_2 \times, \ldots, \times \mathrm{D}_\mathrm{n}, 1 \leq i \leq \mathrm{n}$
- Relation is also called *table*. ***Attribute*** is also called *column*, and ***tuple*** is also called *row*.

Relations are Unordered

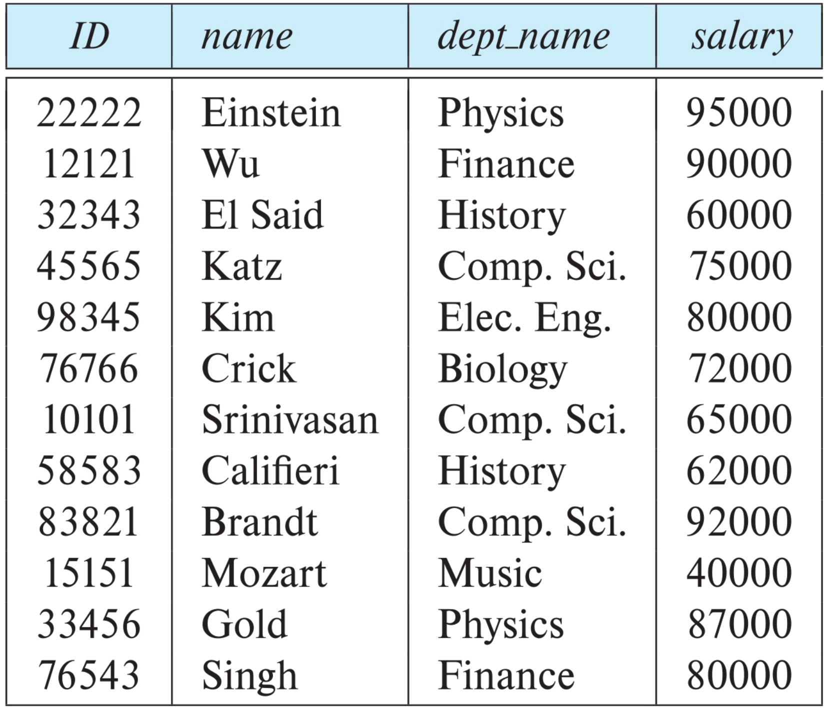
- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples



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 Schema

- Database schema -- is the logical structure of the database.
- Database instance -- is a snapshot of the data in the database at a given instant in time.
- Example:
  - schema: *instructor* (*ID, name, dept_name, salary*)
  - Instance:



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

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

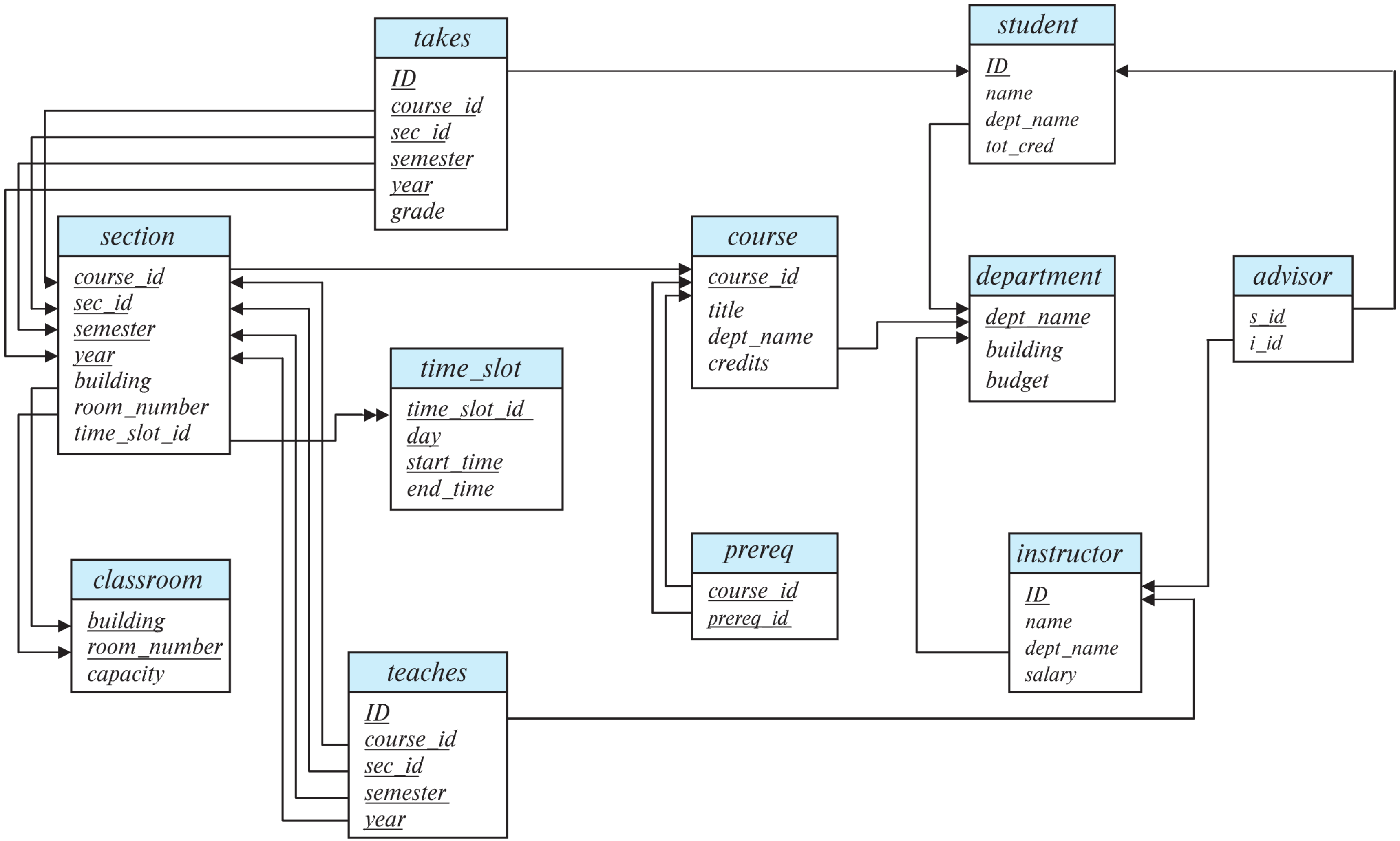
Example: {*ID*} is a candidate key for *Instructor*

- 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*

Primary Key

- A set of attributes is a ***candidate key*** for a relation if :
  1. No two distinct tuples can have same values in this set of attributes, and
  2. This (Part 1) is not true for any subset of this set of attributes.
     - Part 2 false? A ***superkey***.
     - If there's >1 key for a relation, one of the keys is chosen (by DBA) to be the ***primary key***, and the others are called ***alternate key***.
     - If the ***primary key*** consists of all attributes of a relation, it is called ***all key.***
- That means, the key can decide a tuple uniquely.
- E.g., sid is a key for Students. (What about *name*?) The set $\{sid, gpa\}$ is a superkey

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_number
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_number
capacity

prereq
course_id
prereq_id

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

Example 1

- Given Tables:
  - *Students(sid: string, name: string, login: string,*
    age: integer, gpa:real)
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:integer)*
- What's ***candidate key, superkey, primary key, alternate key?***

Example 2

- “Sailors”, “Reserves” and “Boats” relations for our examples.
- What’s ***candidate key, superkey, primary key, alternate key?***

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

B1

bid	bname	color
101	tiger	red
103	lion	green
105	hero	blue

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

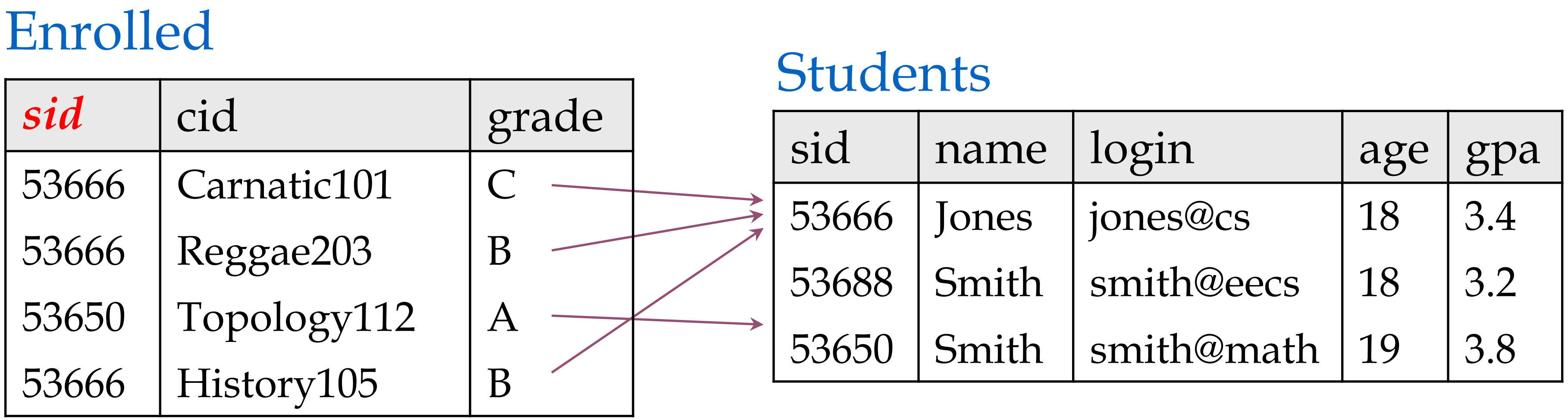
Foreign Keys, Referential Integrity

- ***Foreign key*** : Set of attributes in one relation that is used to ‘refer’ to a tuple in another relation. (Must correspond to primary key of the second relation.) Like a ‘logical pointer’.

- E.g. *sid* is a foreign key referring to Students:
  - Enrolled(*sid*: string, *cid*: string, *grade*: string)
  - If all foreign key constraints are enforced, ***referential integrity*** is achieved, i.e., no dangling references.
  - Have you forgotten ***soft link***?

An Example of Referential Integrity

- Only students listed in the Students relation should be allowed to enroll for courses.



Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Other Integrity Constraints

- Domain integrity constraint
  - An attribute's value must be a value in the domain of this attribute. This is the most basic constraint. All popular RDBMS are able to check domain integrity constraint automatically.
- Entity integrity constraint
  - Every relation should have a primary key. The value of primary key of each tuple must be unique. Primary key cannot be *NULL*. This is so-called entity integrity constraint.

Relational Query Languages

- Procedural versus 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 Turing-machine equivalent
  - Consists of 6 basic operations

Relational Algebra

- A procedural language consisting of a set of operations that take one or two relations as input and produce a new relation as their result.
- Six basic operators
  - select: σ
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ