CMSC 461, Database Management Systems
Spring 2018

Chapter 2 – Introduction to Relational Models

These slides are based on "Database System Concepts" book and slides, 6th edition, and the 2009 CMSC 461 slides by Dr. Kalpakis

Dr. Jennifer Sleeman

https://www.csee.umbc.edu/~jsleem1/courses/461/spr18/

Lecture Outline

- Recap Data Models
- Defining the Relational Model
- Keys
- Schema Diagram
- Relational Query Language
- Relational Operations
- Summary

Lecture Outline

- ***Recap Data Models***
- Defining the Relational Model
- Keys
- Schema Diagram
- Relational Query Language
- Relational Operations
- Summary

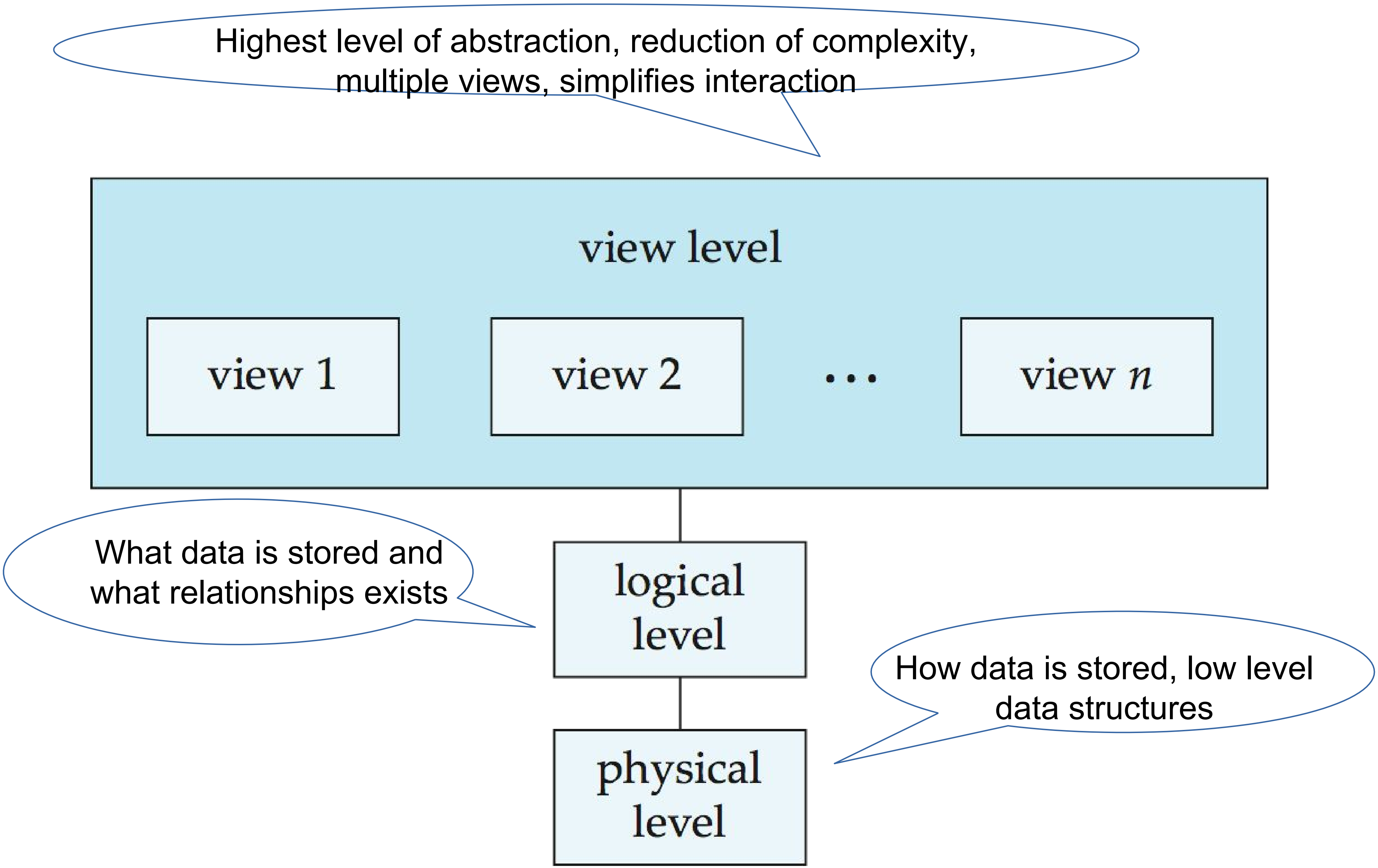
Data Models

What is a data model?

Data Models

- Underlying the structure of a database
- A collection of tools for describing:
  - Data
  - Data Relationships
  - Data Semantics
  - Consistency Constraints
- Provides a way to describe the design of a database at the physical, logical, and view levels

Reminder: View of Data



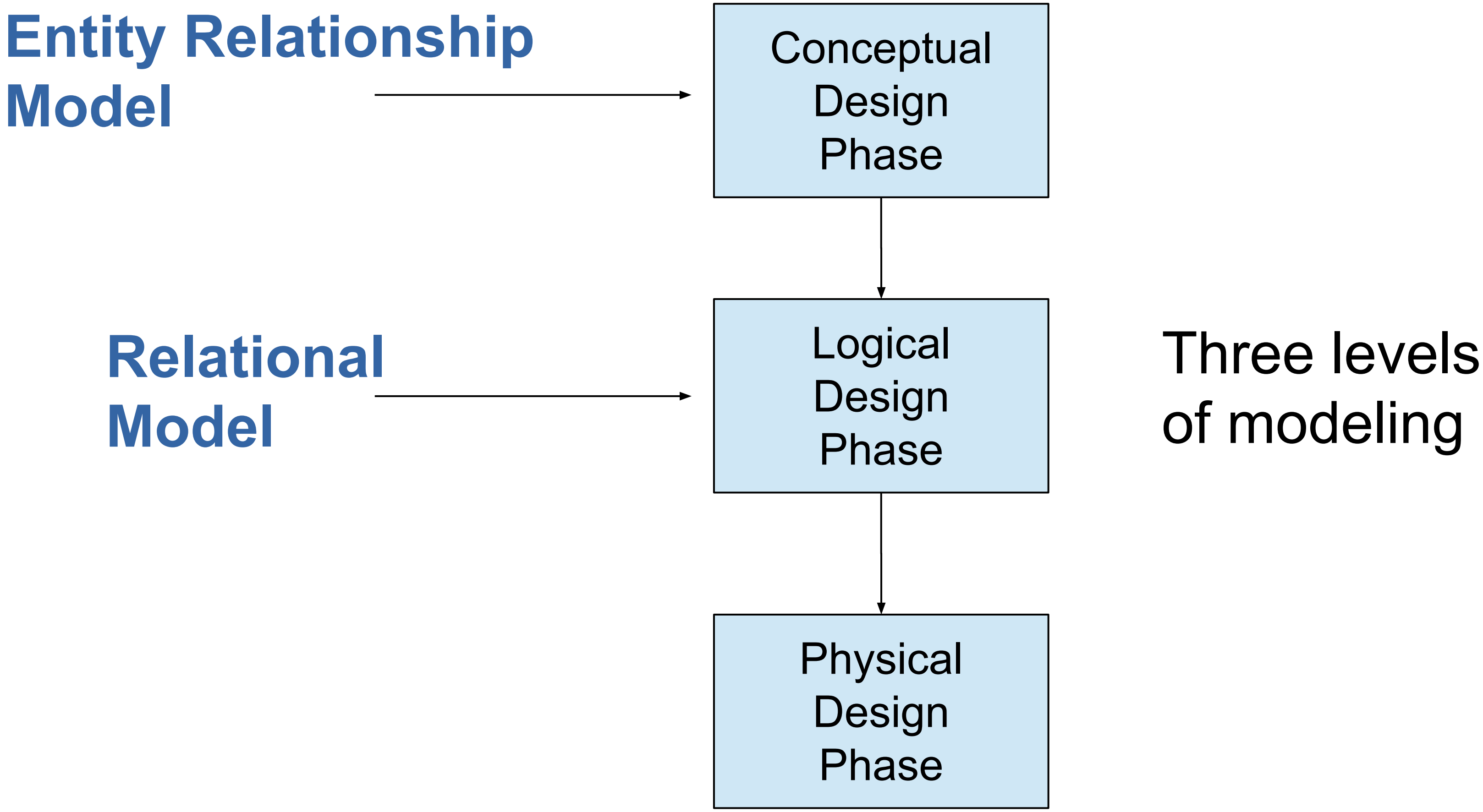
Data Models

- **Relational**
- **Entity-Relationship**
- Object-based - extends E-R model, combines object-oriented data model features and relational data model, See Chapter 22
- Semistructured – different sets of attributes, See Chapter 23
- Network - Used very little, See appendices D and E
- Hierarchical - Used very little, See appendices D and E

Data Models

- Think about data modeling for a moment, we perform modeling for software development, XML, and more.
- Why model, what does it give us?
- Can we model too much?

Database Design Phases



The Relational Model

- Uses a collection of tables to represent:
  - Data
  - Relationships among data
- Each table has multiple columns, each column has a unique name
- Record-based model
  - Data is structured in fixed format records of several types
  - Each table contains records of a particular type
  - Each record type defines a fixed number of fields
- Columns of a table correspond to attributes of the record type

An Example of a Table

Columns

The Instructor Table

Row

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

Lecture Outline

- Recap Data Models
- ***Defining the Relational Model***
- Keys
- Schema Diagram
- Relational Query Language
- Relational Operations
- Summary

Structure of Relational Databases

- A collection of tables
- Each table has a unique name
- A row in a table represents a relationship among a set of values
- Table conceptually like a mathematical relation

Structure of Relational Databases

- Tuple a sequence of values
- Relationship between *n* values represented by an n-tuple of values
- Corresponds to a row in a table

Structure of Relational Databases

- **Relation** - a table
- **Tuple** - a row
- **Attribute** - a column of a table
- **Relation instance** - a specific instance of a relation
  - Containing a specific set of rows

What is a relation?

- 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
  - *Example: Given sets:* $D_1 = \{1, 2, 3\}$, $D_2 = \{4,5\}$
  - $D_1 X D_2 = \{(1,4),(1,5),(2,4),(2,5),(3,4),(3,5)\}$

What is a relation?

- Formally, given sets D_1, D_2, …. D_n a **relation** ***r*** is a subset of $D_1 \times D_2 \times \ldots \times D_n$
  - *A **relation** **r** on $D_{1,} D_2$ is any subset of $\{(1,4),(1,5),(2,4),(2,5),(3,4),(3,5)\}$*
- Thus a relation is a set of ***n*-tuples**

What is a relation?

Example: if

depart_name = {Biology,Comp Sci.,Finance}
depart_building = {Watson,Taylor,Painter}
depart_budget = {90000,100000,120000}

Then r = {**(Biology, Watson, 90000),**
(Comp Sci., Taylor, 100000),
(Finance, Painter, 120000)}
is a relation over depart_name x depart_building x depart_budget

What is a relation?

Attributes

The Instructor Relation

Tuple

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

Attribute Types

- Each attribute of a relation has a name
- The set of allowed values for each attribute is called the **domain** of the attribute
  - Example: The domain of the salary attribute in the Instructor relation is the set of all possible salary values
- Attribute values are (normally) required to be **atomic**, that is, indivisible
  - E.g. multivalued attribute values are not atomic
  - E.g. composite attribute values are not atomic
-

Attribute Types

Are these attributes atomic?

ID	Name
1	Albert Einstein
2	Wolfgang Mozart

Attribute Types

ID	Name
1	Albert Einstein
2	Wolfgang Mozart

ID	First Name	Last Name
1	Albert	Einstein
2	Wolfgang	Mozart

Attribute Types

Are these attributes atomic?

ID	First Name	Last Name	Email
1	Albert	Einstein	aeinstein@google.com,aeinstein@yahoo.com
2	Wolfgang	Mozart	wmozart@msn.com;wmozart@google.com;wmozart@aol.com

Attribute Types

- The special value **null** is a member of every domain
- The null value causes complications in the definition of many operations
  - We shall ignore the effects of null values in our main discussion and consider their effects later

Attribute Types

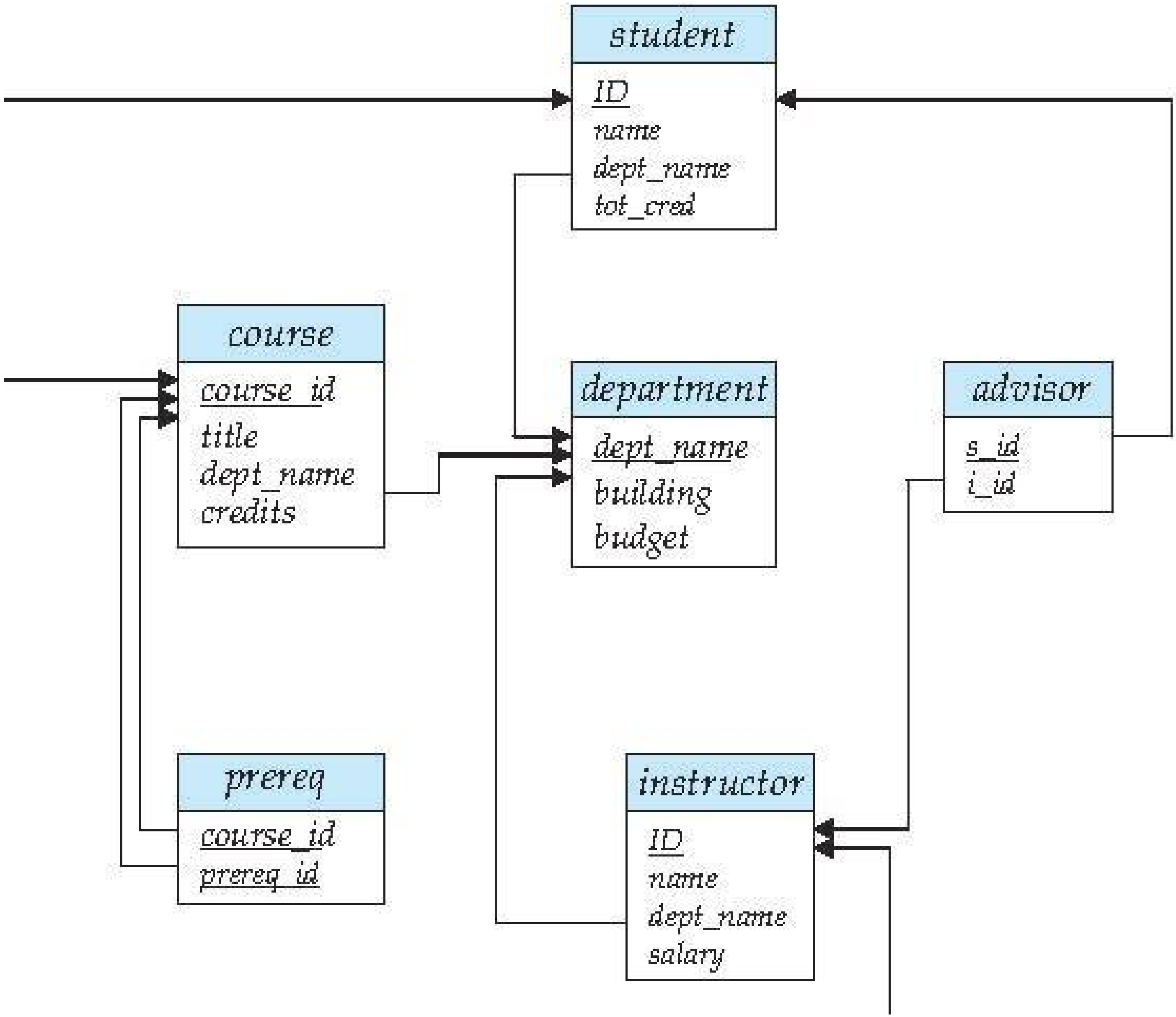
Suppose we do not have a room number yet?

ID	First Name	Last Name	Room Number
1	Albert	Einstein	100
2	Wolfgang	Mozart	NULL

Relation Schema

- Given that $A_1, A_2, \ldots, A_n$ are ***attributes***
- $R = (A_1, A_2, \ldots, A_n)$ *is a* ***relation schema***
-
  - *Example:* ***Instructor_schema = (ID, name, dept_name, salary)***
  -
- r(R) is a relation on the relation schema R
  - *Example:* ***instructor (Instructor_schema)***

Relation Schema



Relation Instance

- A relation instance of a relation *r* corresponds to a table *T*
- An element *t* of *r* is a tuple, and corresponds to a row in table *T*
- There is no implied order among the tuples
- The current values (relation instance) of a relation are specified by a table
-

Relation Instance

attributes
(columns)

tuples
(rows)

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

Instructor relation (table)

Database

- A database is a collection of relations
- Information about an enterprise is broken up into parts
  - instructor
  - student
  - advisor

Database

- 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

Database

What sort of problems might exist with this table?

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

Lecture Outline

- Recap Data Models
- Defining the Relational Model
- ***Keys***
- Schema Diagram
- Relational Query Language
- Relational Operations
- Summary

Keys

- How tuples are distinguished
  - Uniquely identify tuple - No two tuples in a relation allowed to have exactly the same values for all attributes
- A property of the entire relation
- Models a constraint in the real world enterprise
- Expressed in terms of attributes

What happens without them?

Why Identify?

The Instructor table

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

Suppose this was the table?

name	*dept_name*	*salary*
Einstein	Physics	95000
Wu	Finance	90000
El Said	History	60000
Katz	Comp. Sci.	75000
Kim	Elec. Eng.	80000
Crick	Biology	72000
Srinivasan	Comp. Sci.	65000
Califieri	History	62000
Brandt	Comp. Sci.	92000
Mozart	Music	40000
Gold	Physics	87000
Singh	Finance	80000

Why Identify?

What if we add a new instructor with the name Einstein who works in the Physics department and makes a salary of 95000?

name	*dept_name*	*salary*
Einstein	Physics	95000
Wu	Finance	90000
El Said	History	60000
Katz	Comp. Sci.	75000
Kim	Elec. Eng.	80000
Crick	Biology	72000
Srinivasan	Comp. Sci.	65000
Califieri	History	62000
Brandt	Comp. Sci.	92000
Mozart	Music	40000
Gold	Physics	87000
Singh	Finance	80000

Keys

What are examples of real world identifying attributes?

Why Constraints?

Assume no constraints between these tables

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
15152	Mozart2	Musik	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

Why Constraints?

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

If we add a constraint on the attribute dept_name in the Instructor table we ensures a dept_name cannot be entered into the instructor table unless it exists in the department table.

Keys

What are examples of real world constraints?

Keys

- Let R be a set of attributes
- Let K ⊆ 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*

Super Keys

Is Name a super key?

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

Instructor Table

Super Keys & Candidate Keys

What is/are the super key(s)?
What is/are the candidate key(s)?

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

Courses
Table

Super Keys & Candidate Keys

What is/are the super key(s)?
What is/are the candidate key(s)?

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

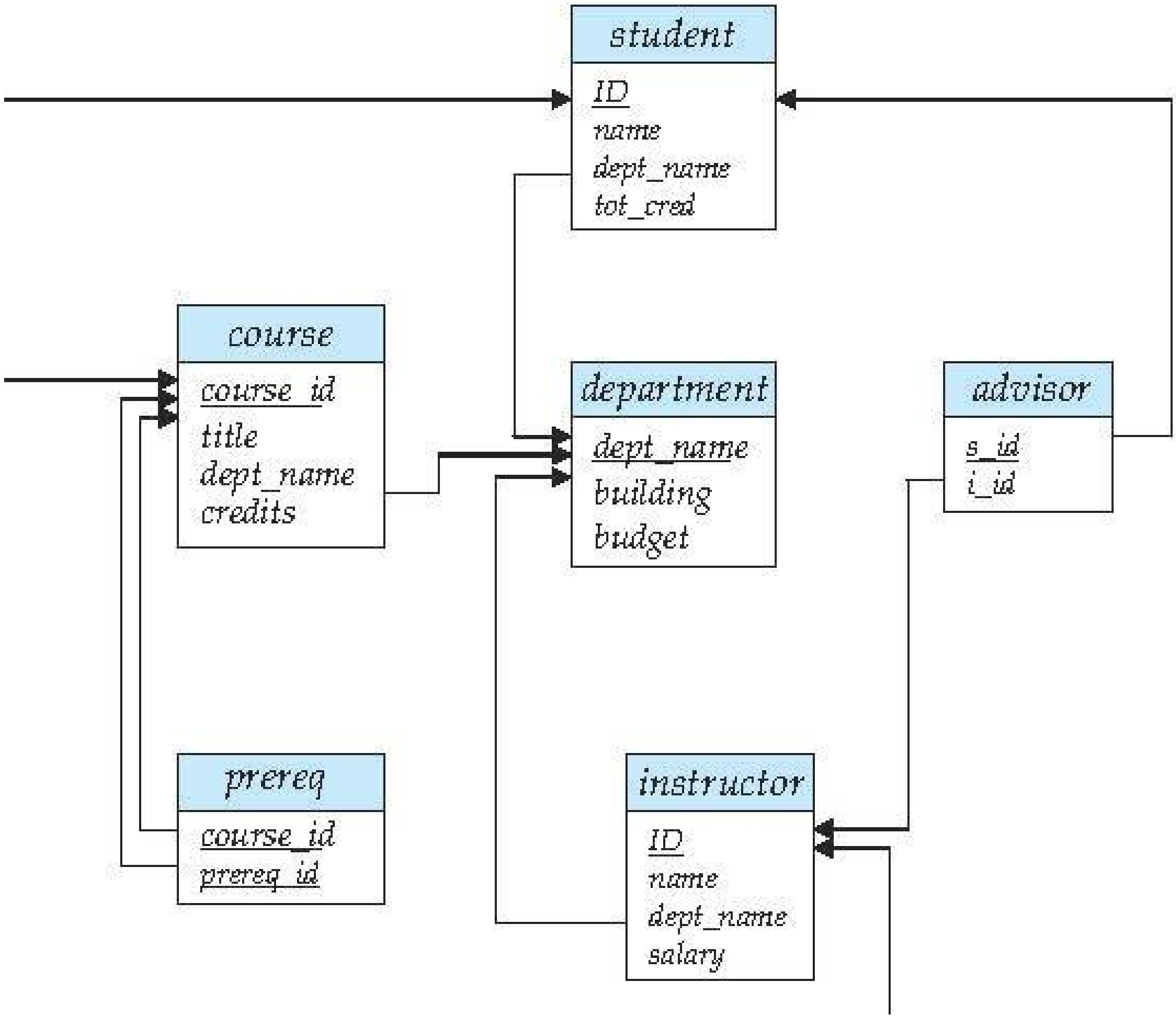
Teaches
Table

Primary Key

Primary key **-** A selected candidate key

- Chosen by DB designer
- Principle means of uniquely identifying tuples within a relation
- Choose with care
- Attribute values never or rarely change
- Can use a combination of other attributes as a key
- List the primary key attribute before others and underline in diagrams

Primary Key



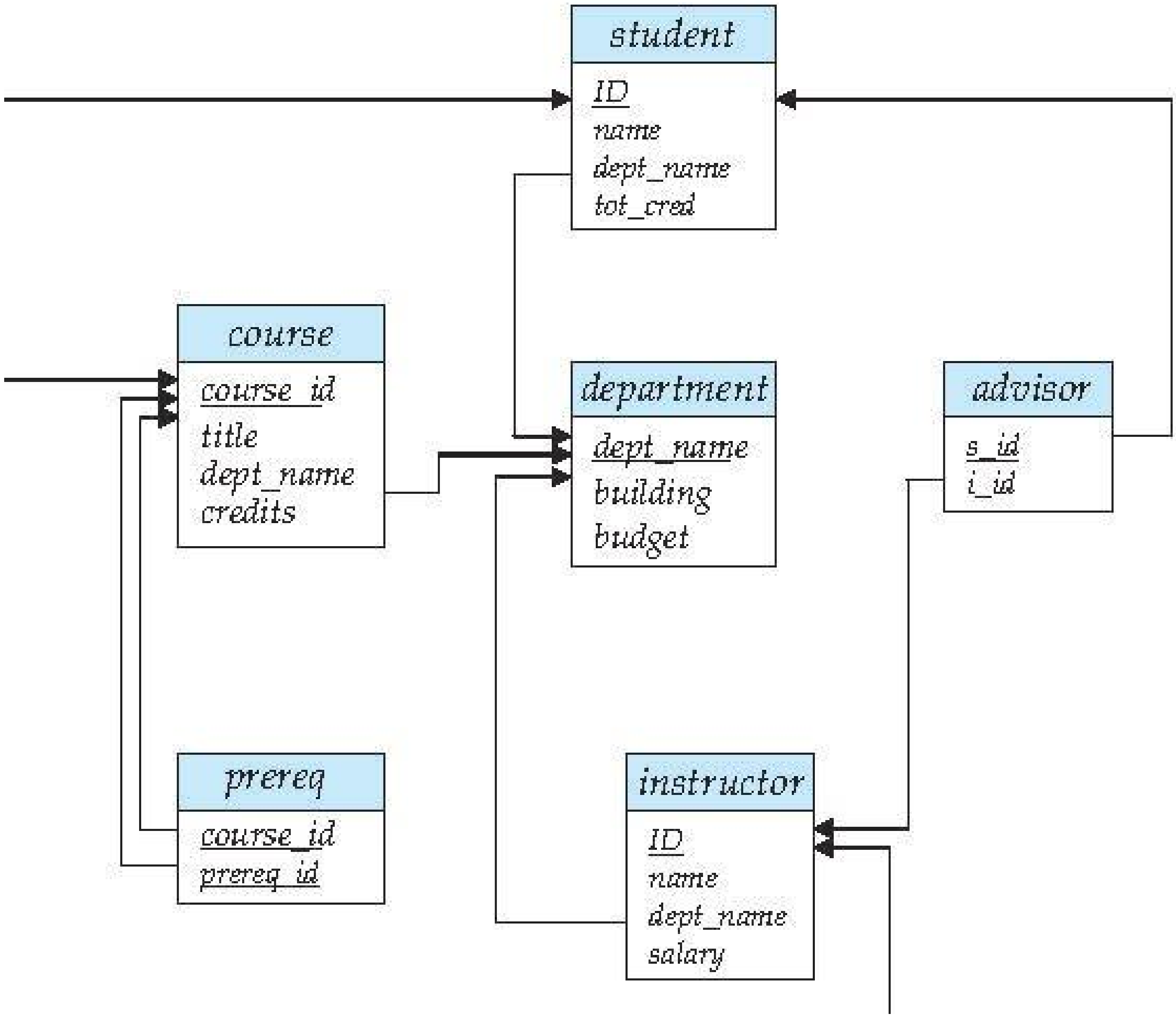
student
ID
name
dept_name
tot_cred
course
course_id
title
dept_name
credits
department
dept_name
building
budget
advisor
s_id
i_id
prereq
course_id
prereq_id
instructor
ID
name
dept_name
salary

Foreign Keys

Foreign key constraint - Value in one relation must appear in another

- A relation r_1 may include among its attributes the primary key of another relation r_2
- r_1 = referencing relation
- r_2 = referenced relation

Foreign Keys



Lecture Outline

- Recap Data Models
- Defining the Relational Model
- Keys
- ***Schema Diagram***
- Relational Query Language
- Relational Operations
- Summary

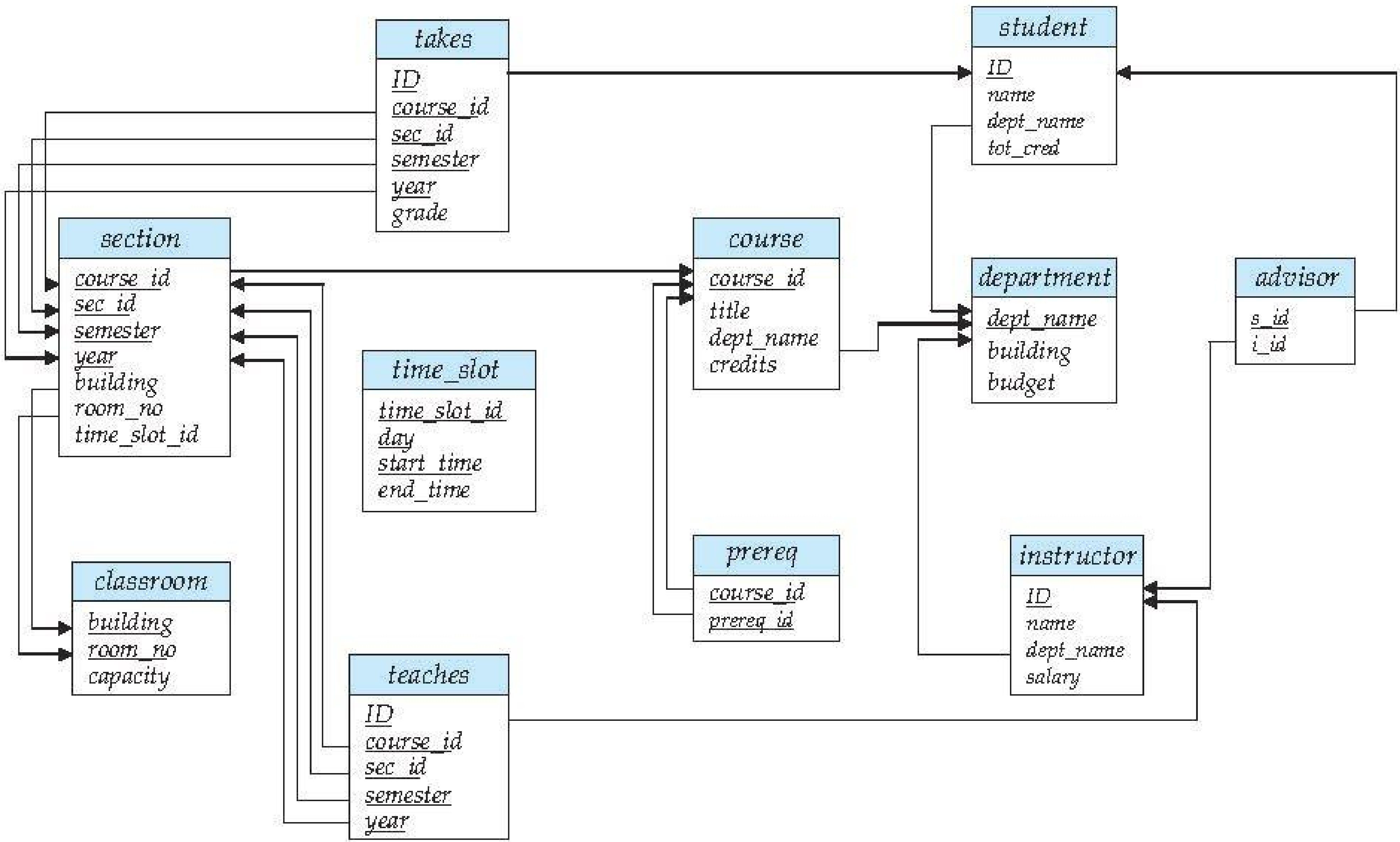
Schema Diagrams

- Depicts:
  - Schema
  - Primary keys
  - Foreign keys
- Relations as boxes
- Primary keys are underlined

Schema Diagrams

- Foreign key dependencies as arrows
  - Foreign key references relation with primary key
- Other referential integrity constraints not shown
  - E-R diagram

Schema Diagram for University Database



Lecture Outline

- Recap Data Models
- Defining the Relational Model
- Keys
- Schema Diagram
- ***Relational Query Language***
- Relational Operations
- Summary

Relational Query Language

- Query language – for requesting information from the database
  - Procedural – specifies sequence of operations to compute desired result
  - Non-procedural – specifies desired information without specifying procedure
- Relational algebra is procedural
- SQL is non-procedural

Lecture Outline

- Recap Data Models
- Defining the Relational Model
- Keys
- Schema Diagram
- Relational Query Language
- ***Relational Operations***
- Summary

Relational Operations

- A procedural query language based on the mathematical theory of sets that is the foundation of commercial DBMS query languages
- The operations typically take one or two relations as inputs and give a new relation as a result
- Can build expressions using multiple relational operations

Relational Operations

- Fundamental operations
  - Select
  - Project
  - Composition
  - Union
  - Set difference
  - Cartesian product
  - Rename

Lecture Outline

- Recap Data Models
- Defining the Relational Model
- Keys
- Schema Diagram
- Relational Query Language
- Relational Operations
- ***Summary***

Summary

- Understand how relational modeling fits into overall design
- Understand the mathematical concepts of a relation
- Understand attribute types, meaning of domain and atomic
- Understand difference between a relation schema and a relation instance
- Understand what contributes to a bad database design
- Understand how different types of keys and their function
- Understand the components of a schema diagram
- Understand procedural vs. non-procedural languages

Test Our Knowledge

A used car dealership has 5 cars in stock.

How would you start to design the relations to represent this problem?

What are some of the supporting relations for this problem?