Databases

Relational Model & SQL

João R. Campos

Bachelor in Informatics Engineering
Department of Informatics Engineering
University of Coimbra
2024/2025

From Previous Less(ons)…

- Databases, DBMS and applications
- Models: relational model
- Design: conceptual model (ER) *vs* physical model (tables)
- Languages: SQL
- Engine: PostgreSQL, Oracle, Microsoft SQL Server
- System Architecture: storage manager, query processor, transaction manager
- Application Architecture: two-tier, three-tier, n-tier
- Users and administrators
- SQL basics

Outline

- Relational Model
  - Relational Databases
  - Relation and Database Schema
  - Keys (Primary and Foreign)
  - Schema Diagrams
  - Relational Query Languages
- Introduction to SQL
  - Overview of the SQL Query Language
  - SQL Data Definition Language (DDL)
  - Structure of SQL Queries (select)
  - Modifying the Data (insert, update, delete)

These slides use the following book as reference:
Abraham Silberschatz, Henry F. Korth and S. Sudarshan, "Database System Concepts", McGraw-Hill Education, Seventh Edition, 2019.

This class focuses mostly on **Chapter 2 & 3**

Register your presence at UCStudent!

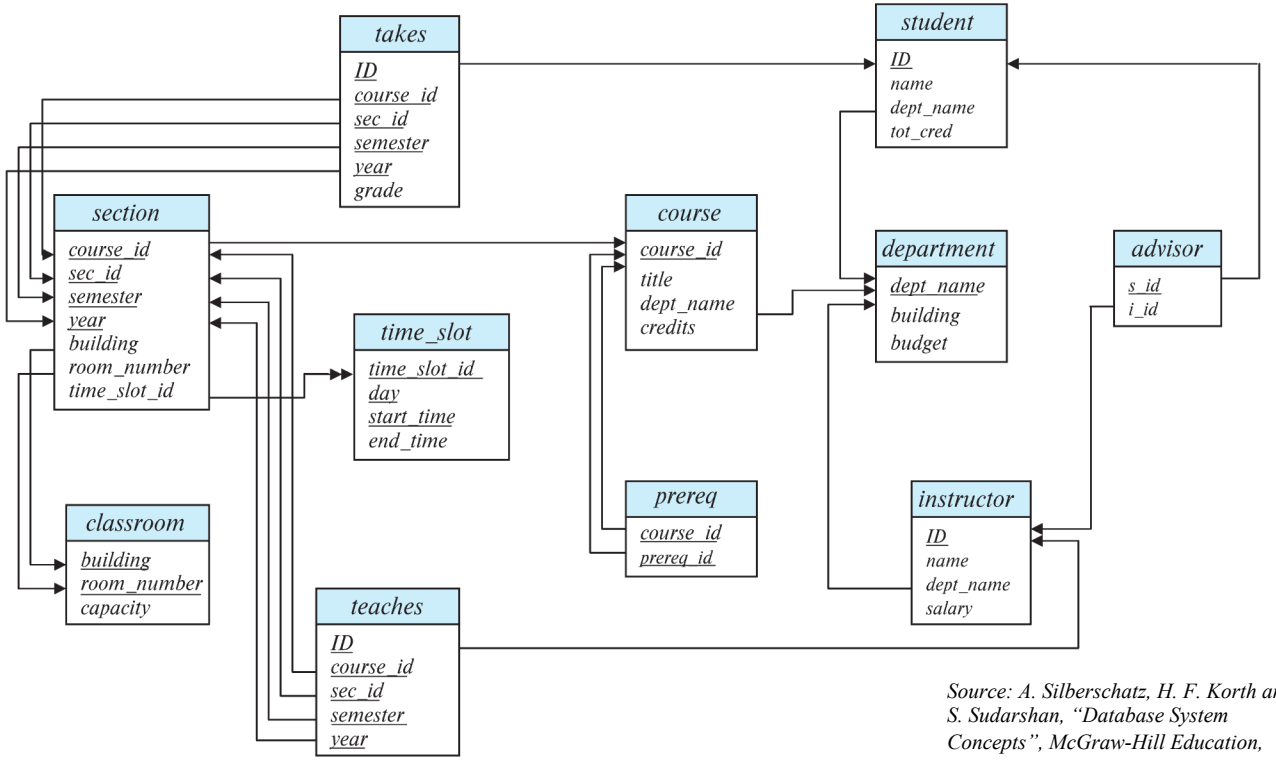
Databases

Relational Databases
Relation and Database Schema
Keys (Primary and Foreign)
Schema Diagrams
Relational Query Languages

RELATIONAL MODEL

Relational Model

- Primary model for data processing, proposed by Edgar F. Codd, 1970
- Simple, independent from any low-level data structures
- Incorporated new features and capabilities over time



Source: A. Silberschatz, H. F. Korth and S. Sudarshan, "Database System Concepts", McGraw-Hill Education, Seventh Edition, 2019.

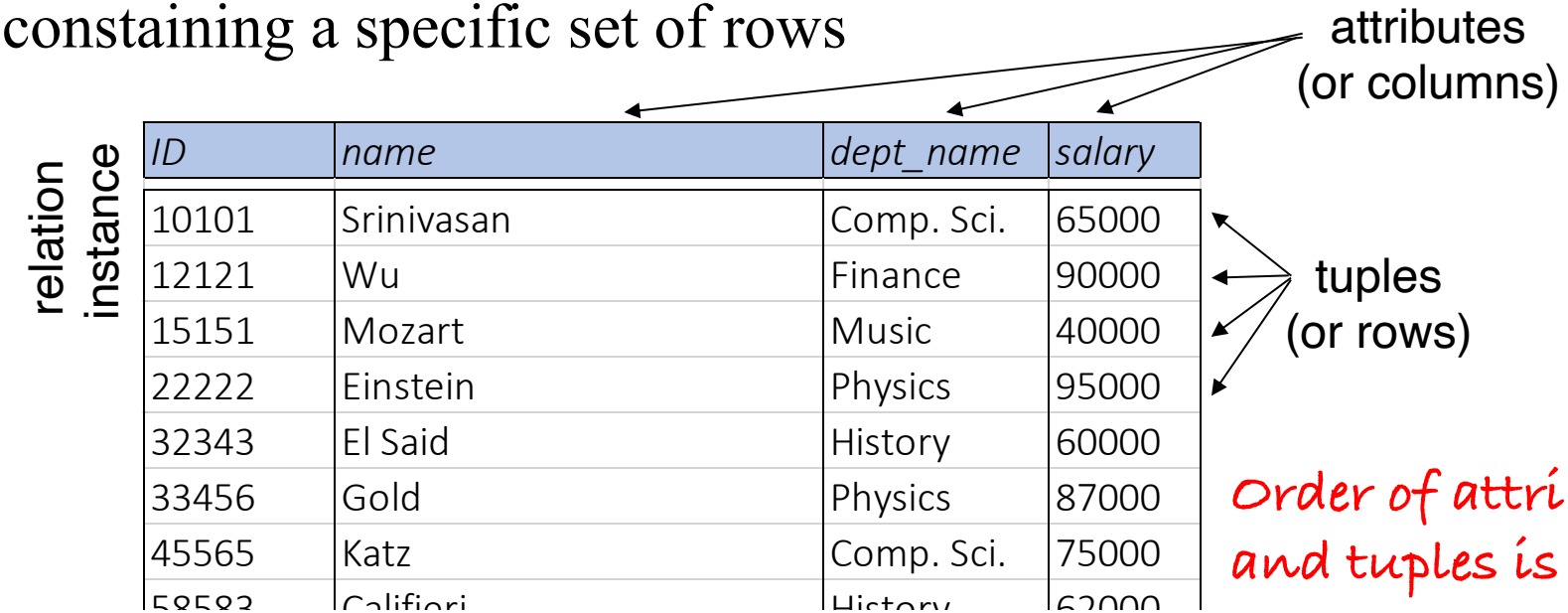
What is a Relation (aka Table)?

Relation of instructors in a university

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

Relational Databases

- A relational database is a collection of relations (tables)
  - Each table has an unique name and is a collection of related data
- A tuple (row) in a relation (table) represents a relationships among a set of values, each one representing an attribute (column)
- A relation instance refers to a specific instance of a relation, constaining a specific set of 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

Order of attributes and tuples is irrelevant!

Some more Relations…

university database

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

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

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

What is the meaning of the *prereq* relation?

Domain of the Attributes

- The domain is the set of permitted values for the attribute
  - e.g. the domain of the *salary* attribute of the *instructor* relation is the set of all possible salary values
  - What is the domain of the attribute *name* of the instructor relation?
- The domain of each attribute must be atomic, i.e. cannot be divided
  - e.g. the *salary* attribute of the *instructor* relation can store a single salary value
  - Is the *dept_name* attribute atomic? e.g., "Informatics Engineering"
  - Imagine that an attribute *phone_number* is added to the *instructor* relation, to represent the (several) phone numbers of each instructor. Is this atomic?
    - What if that that attribute is used to store a single phone number?

      It depends on how we treat this phone number. It can be divided in country code, area code, etc., but if we treat it as an indivisible unit, then it is atomic.

Domain of the Attributes

- The null value is a special case representing that the value is unknown or does not exist
  - e.g. if an instructor does not have a phone number, we would use a null value
  - Null values create a great deal of problems and should be avoided, if possible
- Let's be more specific…
  - What defines a domain? Data type?

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

What is the domain of the attributes in this relation?

Relation Schema and Instance

- Attributes: $A_1, A_2, \ldots, A_n$
- Relation schema is the logical structure of the relation:
  - $R = (A_1, A_2, \ldots, A_n)$
- Relation instance is a snapshot of the data in the table at a given instant in time
- Schema of the relation *instructor*:
  - *instructor = (ID, name, dept_name, salary)*
- An instance *r* defined over schema *R* is denoted by *r(R)*:

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000

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
- Schema of the relations in the *university* database:
  - *classroom(building, room_number, capacity)*
  - *department(dept_name, building, budget)*
  - *course(course_id, title, dept_name, credits)*
  - *instructor(ID, name, dept_name, salary)*
  - *section(course_id, sec_id, semester, year, building, room_number, time_slot_id)*
  - *teaches(ID, course_id, sec_id, semester, year)*
  - *student(ID, name, dept_name, tot_cred)*
  - *takes(ID, course_id, sec_id, semester, year, grade)*
  - *advisor(s_ID, i_ID)*
  - *time_slot(time_slot_id, day, start_time, end_time)*
  - *prereq(course_id, prereq_id)*

What's missing?

Keys

- How do we identify (univocally) a specific row in a table?
- How do we establish a relation between a row in one table and a row in another table?
- Keys!
  - e.g. how can we univocally identify a Portuguese person?
- Superkey is a set of one or more attributes that can be used to identify univocally a tuple in a relation
  - Let $K \subseteq R$, K is a superkey of R if values for K are sufficient to identify a unique tuple of each possible instance of R, i.e. *r(R)*
  - Example: {*ID*} and {*ID*,*name*} are both superkeys of *instructor*
  - Superkeys may contain extraneous attributes, so if K is a superkey, then so is any superset of K

Candidate Keys

- A candidate key is a minimal superkey, i.e. a superkey for each no proper subset is a superkey
  - No two tuples in a relation can have the same value for a candidate key
  - Is {*ID,name*} a candidate key for *instructor*?
  - What about {*ID*}?
- It is possible to have several candidate keys for a single relation
  - Suppose that no two courses can have the same name in the same department, what are the candidate keys of *course(course_id, title, dept_name, credits)*?
- It is important to identify all candidate keys
  - their values cannot repeat

Primary Key

- The primary key is one of the candidate keys
  - How to select the primary key? e.g, {*dept_number*} or {*dept_name*}?
  - For the course relation, should we use {*course_id*} or {*title,dept_name*}?
  - If there is no natural primary key it is possible to create synthetic keys
  - Primary keys are sometimes referred as primary key constraints
- Primary key attributes are customarily listed first and underlined:
  - *department(dept_name, building, budget)*
  - *course(course_id, title, dept_name, credits)*
- In case the primary key includes several attributes:
  - *classroom(building, room_number, capacity)*
  - *prereq(course_id, prereq_id)*

Foreign Keys

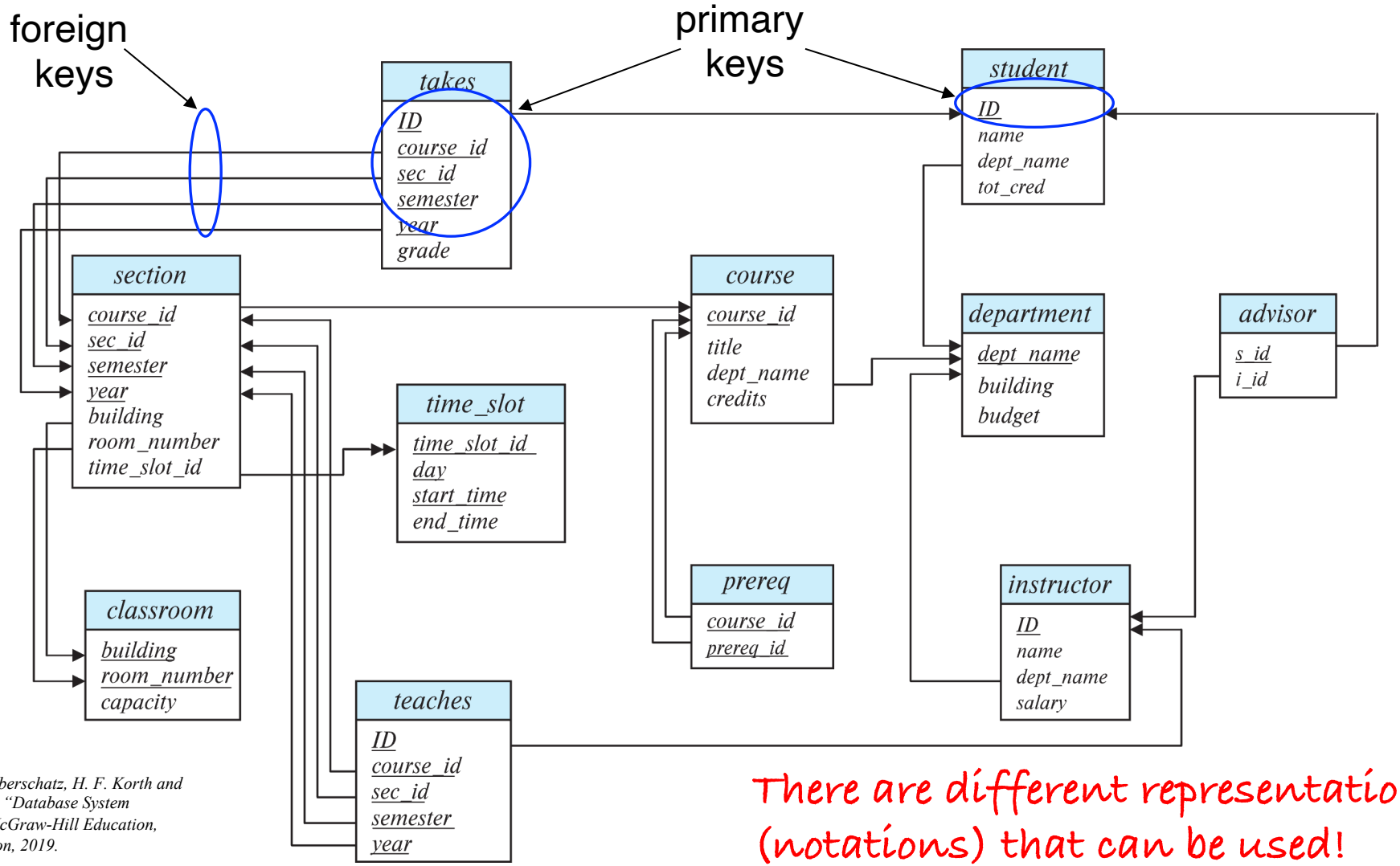
- A foreign-key constraint from attribute(s) A (foreign key) in r_1 to the primary key B of r_2 states that the values of A for each tuple in r_1 must be the value of B for some tuple in r_2
  - i.e., the value in one relation must appear in the other, e.g., the value of *dept_name* in *instructor* must exist in *department*
- The referencing relation is the one where the foreign key exists (r_1)
- The referenced relation is the one referenced by the foreign key (r_2)
  - Referenced attributes must be the primary key of the referenced relation
  - In referential integrity constraints the referenced attributes must be a candidate key (the requirement above is relaxed)
- *dept_name* in *instructor* is a foreign key from *instructor (*referencing relation) to *department* (referenced relation)

Schema of the University Database

- *classroom(building, room_number, capacity)*
- *department(dept_name, building, budget)*
- *course(course_id, title, dept_name, credits)*
- *instructor(ID, name, dept_name, salary)*
- *section(course_id, sec_id, semester, year, building, room_number, time_slot_id)*
- *teaches(ID, course_id, sec_id, semester, year)*
- *student(ID, name, dept_name, tot_cred)*
- *takes(ID, course_id, sec_id, semester, year, grade)*
- *advisor(s_ID, i_ID)*
- *time_slot(time_slot_id, day, start_time, end_time)*
- *prereq(course_id, prereq_id)*

What's missing?

Schema Diagrams



Source: A. Silberschatz, H. F. Korth and S. Sudarshan, "Database System Concepts", McGraw-Hill Education, Seventh Edition, 2019.

There are different representations (notations) that can be used!

Relational Query Languages

- A query language is used to request information from the database
  - Languages normally in a higher level than other programming languages
- Imperative query language: the programmer instructs the system to perform a sequence of operations to compute the result
- Functional query languages: computation expressed as the evaluation of functions that operate on data or on the results of other functions
  - e.g., relational algebra, which forms the base of SQL
- Declarative query languages: the programmer describes the desired information to be obtained and not the steps for obtaining it
  - e.g., tuple relational calculus and domain relational calculus
- SQL includes elements of imperative, functional and declarative

Databases

Overview of the SQL Query Language
SQL Data Definition Language (DDL)
Basic Structure of SQL Queries (select)
Modifying the Data (insert, update, delete)

INTRODUCTION TO SQL

Structured Query Language (SQL)

- SQL was initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce after learning about the relational model, 1970's
  - Several ANSI and ISO standards over the years: 1986, 1989, 1992, 1999, 2003, 2006, 2008, 2011 and 2016
- Although called a query language, it can be used to do more tham just query a database
  - Define the structure of the database
  - Modify data
  - Specify security constraints
  - …

Parts of the SQL Language

- Data Definition Language (DDL): defining relations, deleting relations, and modifying relations
- Data Manipulation Language (DML): query information from the database and insert, delete and modify tuples
- Integrity: DDL commands for specifying integrity constraints
- View definition: DDL commands for defining views
- Transaction control: specifying the beginning and end of transactions
- Authorization: specifying access rights to relations and views
- Embedded SQL and dynamic SQL: specifies how SQL can be embedded in general-purpose programming languages

SQL Data Definition Language (DDL)

- Use to specify not only a set of relations, but also information about each relation:
  - Schema for each relation
  - Data types of the values associated with each attribute
  - Integrity constraints
  - Set of indices to be maintained for each relation
  - Security and authorization information for each relation
  - Physical storage structure of each relation on disk

Basic Data Types

- char(n)
- varchar(n)
- smallint (2), integer (4), bigint (8)
- numeric(p, d)
- real, double precision
- …

create table…

- SQL DDL command used to define a relation:

```
create table r
        (A1 D1,
         A2 D2,
         ...,
         An Dn,
         ⟨integrity-constraint1⟩,
         ...,
         ⟨integrity-constraintk⟩);
```

```
create table department
        (dept_name varchar(20),
         building varchar(15),
         budget numeric(12,2),
         primary key (dept_name));
```

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

Integrity Constraints

- primary key (A_{j_1}, A_{j_2}, …, A_{j_n}): specifies the set of attributes that act as primary key, which are required to be *nonnull* and *unique*
- foreign key (A_{k_1}, A_{k_2},…,A_{k_n}) references *s*: defines a set of attributes that are a foreign key, where the attributes in *s* must be a primary key

```
create table section
       (course_id varchar(8),
        sec_id varchar(8),
        semester varchar(6),
        year numeric(4,0),
        building varchar(15),
        room_number varchar(7),
        time_slot_id varchar(4),
        primary key (course_id, sec_id, semester, year),
        foreign key (course_id) references course);
```

Integrity Constraints

- not null: specifies the attributes for which a value must exist
- unique (A_{j_1}, A_{j_2}, …, A_{j_n}): defines a set of attributes that form a superkey
- check(P): specifies a condition that must be satisfied for every tuple in the relation
- SQL prevents any update that violates an integrity constraint

synthetic PK (auto-increment)

```
create table instructor
        (ID INT primary key generated always as identity,
         name varchar(20) not null,
         dept_name varchar(20),
         salary numeric(8,2),
         foreign key (dept_name) references department,
         check (salary>0));
```

alter table…

- SQL DDL command used to alter a relation:

```
alter table instructor
        alter column name type varchar(50),
        alter column name set not null,
        alter column dept_name type varchar(20),
        alter column dept_name set not null;
```

- Fails if the new constraints cannot be satisfied by the existing data
  - e.g., if there are tuples in *instructor* where *dept_name* is null

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart		40000
22222	Einstein	Physics	95000

drop table…

- SQL DDL command used to remove a relation:

```
drop table course;
```

- Fails if the table is referenced by any other table with a foreign key
  - e.g., if table *section* has a foreign key to the table *course* (which is being dropped)
- Note that drop table is different from delete
  - *delete* is used to delete data from a table, but the table is not removed
  - *drop* is used to remove the table (data is obviously lost)

Basic Structure of SQL Queries (select)

- SQL DML command used to get data from the database:

```
select A1, A2,…, An
from r1, r2,…, rm
where P;
```

- $A_1, A_2, \ldots, A_n$ are the attributes to be obtained from the relations $r_1, r_2, \ldots, r_m$
- P is a predicate to be evaluated to decide which data to consider
  - If the *where* clause is omitted, then P is true

```
select name
from instructor;
```

name
Srinivasan
Wu
Mozart
Einstein
El Said
Gold
Katz
Califieri

Selecting from Multiple Relations

- Consider this form of a *select*:

```
select A1, A2,…, An
from r1, r2,…, rm;
```

- It is executed this way:

```
for each tuple t1 in relation r1
    for each tuple t2 in relation r2
        ...
        for each tuple tm in relation rm
            Concatenate t1, t2,…, tm into a single tuple t
            Add t into the result relation
```

- What is the result? How does it work?

How does it work?

- Consider the form of a typical *select*:

```
select *
from department, instructor;
```

Note that there is no where clause!

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

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

dept_name	building	budget	id	name	dept_name	salary
Biology	Watson	90000.00	10101	Srinivasan	Comp. Sci.	65000.00

How does it work?

- Consider the form of a typical *select*:

```
select *
from department, instructor;
```

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

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

dept_name	building	budget	id	name	dept_name	salary
Biology	Watson	90000.00	10101	Srinivasan	Comp. Sci.	65000.00
Biology	Watson	90000.00	12121	Wu	Finance	90000.00

How does it work?

- Consider the form of a typical *select*:

```
select *
from department, instructor;
```

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

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

dept_name	building	budget	id	name	dept_name	salary
Biology	Watson	90000.00	10101	Srinivasan	Comp. Sci.	65000.00
Biology	Watson	90000.00	12121	Wu	Finance	90000.00
Biology	Watson	90000.00	15151	Mozart	Music	40000.00

How does it work?

- Consider the form of a typical *select*:

```
select *
from department, instructor;
```

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

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

dept_name	building	budget	id	name	dept_name	salary
Biology	Watson	90000.00	10101	Srinivasan	Comp. Sci.	65000.00
Biology	Watson	90000.00	12121	Wu	Finance	90000.00
Biology	Watson	90000.00	15151	Mozart	Music	40000.00
...	...	...	...	...	...	...
...	...	...	...	...	...	...
Comp. Sci.	Taylor	100000.00	10101	Srinivasan	Comp. Sci.	65000.00

How does it work?

- Consider the form of a typical *select*:

```
select *
from department, instructor;
```

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

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

dept_name	*building*	*budget*	*id*	*name*	*dept_name*	*salary*
Biology	Watson	90000.00	10101	Srinivasan	Comp. Sci.	65000.00
Biology	Watson	90000.00	12121	Wu	Finance	90000.00
Biology	Watson	90000.00	15151	Mozart	Music	40000.00
...	...	...	...	...	...	...
...	...	...	...	...	...	...
Comp. Sci.	Taylor	100000.00	10101	Srinivasan	Comp. Sci.	65000.00
Comp. Sci.	Taylor	100000.00	12121	Wu	Finance	90000.00

How does it work?

- Consider the form of a typical *select*:

```
select *
from department, instructor;
```

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

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

dept_name	building	budget	id	name	dept_name	salary
Biology	Watson	90000.00	10101	Srinivasan	Comp. Sci.	65000.00
Biology	Watson	90000.00	12121	Wu	Finance	90000.00
Biology	Watson	90000.00	15151	Mozart	Music	40000.00
…	…	…	…	…	…	…
…	…	…	…	…	…	…
Comp. Sci.	Taylor	100000.00	10101	Srinivasan	Comp. Sci.	65000.00
Comp. Sci.	Taylor	100000.00	12121	Wu	Finance	90000.00
Comp. Sci.	Taylor	100000.00	15151	Mozart	Music	40000.00
…	…	…	…	…	…	…
…	…	…	…	…	…	…
Elec. Eng.	Taylor	85000.00	10101	Srinivasan	Comp. Sci.	65000.00
Elec. Eng.	Taylor	85000.00	12121	Wu	Finance	90000.00
Elec. Eng.	Taylor	85000.00	15151	Mozart	Music	40000.00
…	…	…	…	…	…	…
…	…	…	…	…	…	…
Finance	Painter	120000.00	10101	Srinivasan	Comp. Sci.	65000.00
Finance	Painter	120000.00	12121	Wu	Finance	90000.00
Finance	Painter	120000.00	15151	Mozart	Music	40000.00

Cartesian Product!
How many rows?

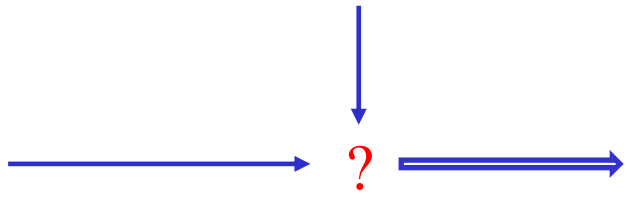
Joining Related Tables

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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



name	dept_name	building
Srinivasan	Comp. Sci.	Taylor
Wu	Finance	Painter
Mozart	Music	Packard
Einstein	Physics	Watson
El Said	History	Painter
Gold	Physics	Watson
Katz	Comp. Sci.	Taylor
Califieri	History	Painter
Singh	Finance	Painter
Crick	Biology	Watson
Brandt	Comp. Sci.	Taylor
Kim	Elec. Eng.	Taylor

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

name	dept_name	building

?

instructor.dept_name
=
department.dept_name

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

name	dept_name	building

?

instructor.dept_name
=
department.dept_name

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

?

department.dept_name
=
instructor.dept_name

name	dept_name	building
Srinivasan	Comp. Sci.	Taylor

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

?

department.dept_name
=
instructor.dept_name

name	dept_name	building
Srinivasan	Comp. Sci.	Taylor

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

?

department.dept_name
=
instructor.dept_name

name	dept_name	building
Srinivasan	Comp. Sci.	Taylor

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

?

department.dept_name
=
instructor.dept_name

name	dept_name	building
Srinivasan	Comp. Sci.	Taylor

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

?

department.dept_name
=
instructor.dept_name

name	dept_name	building
Srinivasan	Comp. Sci.	Taylor

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

?

department.dept_name
=
instructor.dept_name

name	dept_name	building
Srinivasan	Comp. Sci.	Taylor

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

?

department.dept_name
=
instructor.dept_name

name	dept_name	building
Srinivasan	Comp. Sci.	Taylor

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

?

department.dept_name
=
instructor.dept_name

name	dept_name	building
Srinivasan	Comp. Sci.	Taylor

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

?

department.dept_name
=
instructor.dept_name

name	dept_name	building
Srinivasan	Comp. Sci.	Taylor

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

?

department.dept_name
=
instructor.dept_name

name	dept_name	building
Srinivasan	Comp. Sci.	Taylor
Wu	Finance	Painter

How does it work?

- Consider the following example:

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

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

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

?

department.dept_name
=
instructor.dept_name

name	dept_name	building
Srinivasan	Comp. Sci.	Taylor
Wu	Finance	Painter
Mozart	Music	Packard
Einstein	Physics	Watson
El Said	History	Painter
Gold	Physics	Watson
Katz	Comp. Sci.	Taylor
Califieri	History	Painter
Singh	Finance	Painter
Crick	Biology	Watson
Brandt	Comp. Sci.	Taylor
Kim	Elec. Eng.	Taylor

Data from Different Tables and Conditions

```
select name, course_id
from instructor, teaches
where instructor.ID = teaches.ID;
```

name	course_id
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

```
select name, course_id
from instructor, teaches
where instructor.ID = teaches.ID
and instructor.dept_name = 'Comp. Sci.';
```

name	course_id
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Katz	CS-101
Katz	CS-319
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319

Rename Tables

```
select T.name, S.course_id
from instructor as T, teaches as S
where T.ID = S.ID;
```

name	course_id
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

```
select distinct T.name
from instructor as T, instructor as S
where T.salary > S.salary
and S.dept_name = 'Biology';
```

name
Einstein
Katz
Singh
Kim
Brandt
Wu
Gold

What?!

What happened here?

Alternate Syntax

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.dept_name;
```

The older comma join syntax is usually referred to as SQL-89

```
select name, instructor.dept_name, building
from instructor JOIN department ON
instructor.dept_name = department.dept_name;
```

The ON syntax was introduced in SQL-92; also called ANSI JOIN/syntax

Aggregate Functions

- Aggregate functions take a collection of values and return a single value, by performing some aggregation operation:
  - Average: avg
  - Minimum: min
  - Maximum: max
  - Total: sum
  - Count: count

```
select avg(salary)
from instructor
where dept_name = 'Comp. Sci.';
```

avg
77333.333333333333

Counting Rows

```
select count(ID)
from teaches
where semester = 'Spring'
and year = 2018;
```

count
7

Why?

```
select count(distinct ID)
from teaches
where semester = 'Spring'
and year = 2018;
```

count
6

```
select count (*)
from course;
```

count
13

insert… (One Row)

- SQL DML command used to insert one row in a relation:

```
insert into r(A1, A2, …, An)
values (v1, v2, …, vn);
```

- $A_1, A_2, \ldots, A_n$ are the attributes to be filled and $v_1, v_2, \ldots, v_n$ are the values for those attributes

```
insert into course
values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
```

```
insert into course (course_id, title, dept_name, credits)
values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
```

```
insert into course (title, course_id, credits, dept_name)
values ('Database Systems', 'CS-437', 4, 'Comp. Sci.')
returning course_id;
```

- The command may fail for several reasons, including data type mismatch and foreign key constraint violation

insert... (Several Rows)

- SQL DML command used to insert several rows in a relation:

```
insert into r
       select …
```

- The output of the *select* is inserted into table *r*

```
insert into instructor
       select ID, name, dept_name, 18000
       from student
       where dept_name = 'Music' and tot_cred > 144;
```

- Multi-row insert

```
insert into instructor (ID, name, dept_name, salary)
values
       (32423, 'Mark', 'Biology', 50000),
       (42323, 'Carl', 'Physics', 35000),
...
```

delete...

- SQL DML command used to delete data from a relation:

```
delete from r
where P;
```

- *P* is a predicate to be evaluated to decide which data to delete
  - If the *where* clause is omitted, all data is deleted

```
delete from instructor;
```

```
delete from instructor
where dept_name = 'Finance';
```

```
delete from instructor
where salary between 13000 and 15000;
```

- The command may fail if rows in other tables reference the rows being deleted
  - Unless "*on delete cascade*" was specified when creating the referencing table

update

- SQL DML command used to update data in a relation:

```
update r
set A1 = v1, A2 = v2, …, An = vn
where P;
```

- *P* is a predicate to be evaluated to decide which data to delete
  - If the *where* clause is omitted, all rows are updated

```
update instructor
set salary = salary * 1.05;
```

```
update instructor
set salary = salary * 1.05, name = upper(name)
where salary < 70000;
```

```
update instructor
set salary = salary * 1.05
where salary < (select avg(salary) from instructor);
```

Take-Away(s)

- Relational Model
  - Relational databases
  - Relation or table, tuple or row, attribute or column
  - Superkeys, candidate keys, primary key and foreign key
  - Integrity restrictions: primary key (entity), foreign key (referential), domain, …
- SQL
  - create table, alter table, and drop table
  - Query the database: select…
  - Cartesian product and joining tables
  - Aggregation functions
  - Modifying the data (insert, update, delete)

Next Lesson(s)

- Database Design Using Entity-Relationship Model
- Database Design Process
- Entities and Relationships
- Relationship Cardinalities and Participation
- Alternative Notations for E-R Diagrams
- From E-R Diagrams to Relational Schemas

Q&A

Databases

Relational Model & SQL

João R. Campos

Bachelor in Informatics Engineering
Department of Informatics Engineering
University of Coimbra
2024/2025