UMBC

AN HONORS UNIVERSITY IN MARYLAND

CMSC 461, Database Management Systems
Spring 2018

Lecture 5 Chapter 3 – Introduction to SQL

These slides are based on “Database System Concepts” book and slides, 6th edition, and the 2009/2012 CMSC 461 slides by Dr. Kalpakis

Jennifer Sleeman

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

Logistics

- Phase 1 of project is due 2/15/2018

Lecture Outline

- Overview
- Data Definition Language
- Data Manipulation Language

Lecture Outline

- ***Overview***
- Data Definition Language
- Data Manipulation Language

Overview

- SQL – most widely used
- Used to:
  - Query database
  - Define structure of the data
  - Modify data in database
  - Specify security constraints

History

- Original Version
  - Called Sequel
  - Developed by IBM
  - Part of System R project in early 1970's
- Renamed Structured Query Language (SQL)
- ANSI and ISO standard SQL:
  - SQL-86, SQL-89, SQL-92
  - SQL:1999, SQL:2003, SQL:2008
- Commercial systems offer most, if not all, SQL-92 features, plus varying feature sets from later standards and special proprietary features.
  - Not all examples here may work on your particular system

SQL Language

- Data-definition language (DDL)
  - Define relation schemas, delete relations, modify relation schemas
- Data-manipulation language (DML)
  - Query information, insert tuples, delete tuples, modify tuples in database
- Integrity
- View Definition
- Transaction Control
- Embedded SQL and Dynamic SQL
  - Embed in programming languages
- Authorization

Lecture Outline

- Overview
- ***Data Definition Language***
- Data Manipulation Language

SQL Data Definition

- The SQL ***data-definition language*** (DDL) allows the specification of information about relations, including:
  - Schema for each relation
  - Types of values for attributes
  - Integrity constraints
  - Relation indices
  - Security and Authorization
  - Physical storage structure

Domain Types in SQL

- ***char(n)*** - Fixed length character string, with user-specified length n.
- ***varchar(n)*** - Variable length character strings, with user-specified maximum length n.
- ***int*** - Integer (a finite subset of the integers that is machine-dependent).
- ***smallint*** - Small integer (a machine-dependent subset of the integer domain type).
- ***numeric(p,d)*** - Fixed point number, with user-specified precision of p digits, with n digits to the right of decimal point.
- ***real, double precision*** - Floating point and double-precision floating point numbers, with machine-dependent precision.
- ***float(n)*** - Floating point number, with user-specified precision of at least n digits.
- More covered in Chapter 4

MySQL Exercise

Use your own computer for this exercise

MySQL Exercise

Login to MySQL

MySQL Exercise

mysql -u <username> -p

MySQL Exercise

Create a database called lecture5

MySQL Exercise

```
create database lecture5;
```

MySQL Exercise

Create a new user

MySQL Exercise

CREATE USER jenn IDENTIFIED BY 'jennspassword';

grant usage on *.* to jenn@localhost identified by 'jennspassword';

grant all privileges on lecture5.* to jenn@localhost;

MySQL Exercise

Login to MySQL as new user

MySQL Exercise

First: type exit

mysql -u jenn -p

MySQL Exercise

Connect to the database

MySQL Exercise

```
use lecture5;
```

MySQL Exercise

Look at what tables are defined in the lecture5 database

MySQL Exercise

```
show tables;
```

MySQL Exercise

Create a table called test_char

MySQL Exercise

```
create table test_char (capacity char(2));
```

MySQL Exercise

Look at the table you just created

MySQL Exercise

```
describe test_char;
```

MySQL Exercise

Insert into the test_char table

MySQL Exercise

insert into test_char (capacity) values (100);

MySQL Exercise

What happened?

Errors and Defining sizes

create table test_char (capacity ***char(2)***);

insert into test_char (capacity) values (**100**);

ERROR 1406 (22001): Data too long for column 'capacity' at row 1

MySQL Exercise

Let change the data type….

Errors and Defining Sizes

create table test_varchar (capacity ***varchar(2)***);

Errors and Defining Sizes

insert into test_varchar (capacity) values (***100***);

Errors and Defining Sizes

drop table test_varchar;

create table test_varchar (capacity ***varchar(3)***);

insert into test_varchar (capacity) values (***100***);

Create Table Construct

SQL relation is defined using the create table command:

create table r (A_1 D_1, A_2 D_2, ..., A_n D_n,
(integrity-constraint$_1$),
...,
(integrity-constraint$_k$))

r is the name of the relation
each A_i is an attribute name in the schema of relation r
D_i is the data type of values in the domain of attribute A_i

Create Table Construct

Example:

```
create table instructor (
ID            char(5),
name          varchar(20) not null,
dept_name     varchar(20),
salary        numeric(8,2));
```

How do you view the structure of the table you created?

How do you view the structure of the table you created?

```
describe instructor;
```

Integrity Constraints in Create Table

Primary Key (A_{j1}, A_{j2} ..., A_{jm})

Required not null

Require unique

Foreign Key (A_{k1}, A_{k2} ..., A_{kn}) references r

Not null

Specifies null not allowed

Integrity Constraints in Create Table

Example:

Declares *ID* as the primary key for *instructor, depart_name* as the foreign key and *name* as 'not null'.

create table *instructor* (
ID **char**(5),
name **varchar**(20) **not null,**
dept_name **varchar**(20),
salary **numeric**(8,2),
primary key (*ID*),
foreign key *(dept_name*) **references** *department (dept_name) on delete set null);*

Examples

```
create table student (
        ID              varchar(5),
        name            varchar(20) not null,
        dept_name       varchar(20),
        tot_cred        numeric(3,0),
        primary key (ID),
        foreign key (dept_name)
        references department (dept_name)
on delete set null );
```

Examples

```
create table takes (
        ID              varchar(5),
        course_id       varchar(8),
        sec_id          varchar(8),
        semester        varchar(6),
        year            numeric(4,0),
        grade           varchar(2),
        primary key (ID, course_id, sec_id, semester, year),
        foreign key (ID) references  student (ID) on delete set null,
        foreign key (course_id, sec_id, semester, year) references
section (course_id,sec_id, semester, year) on delete set null );
```

Examples

create table *course* (
    course_id **varchar**(8) **primary key**,
    title **varchar(**50),
    dept_name **varchar**(20),
    credits **numeric**(2,0),
    foreign key (*dept_name*) **references** *department (dept_name) on delete set null*);

Primary key declaration can be combined with attribute declaration as shown above

Can I do this?

```
create table course (
        course_id      varchar(8),
        title          varchar(50),
        dept_name      varchar(20)  primary key,
        credits        numeric(2,0),
        foreign key (dept_name) references
department  (dept_name) on delete set null);
```

Can I do this?

```
create table course (
        course_id       varchar(8),
        title           varchar(50),
        dept_name       varchar(20)  primary key,
        credits         numeric(2,0),
        foreign key (dept_name) references
department  (dept_name) on delete set null);
```

ERROR 1215 (HY000): Cannot add foreign key constraint

Can I do this?

```
create table course2 (
        course_id      varchar(8) primary key,
        title          varchar(50) primary key,
        dept_name      varchar(20),
        credits        numeric(2,0),
        foreign key (dept_name) references
(dept_name) on delete set null);
```

Can I do this?

```
create table course2 (
        course_id       varchar(8) primary key,
        title           varchar(50)  primary key,
        dept_name       varchar(20),
        credits         numeric(2,0),
        foreign key (dept_name) references
department (dept_name) on delete set null);
```

ERROR 1068 (42000): Multiple primary key defined

Can I do this?

```
create table course (
        course_id        varchar(8) primary key,
        title            varchar(50),
        dept_name        varchar(20),
        credits          numeric(2,0),
        foreign key (dept_name) references
department (dept_name) on delete set null);
```

insert into course (course_id, title, dept_name, credits) values ("BIO-101", "Intro to Bio", "Biology", 4);
insert into course (course_id, title, dept_name, credits) values ("BIO-101", "Intro to Bio", "Biology", 3);

Can I do this?

create table *course* (
    course_id **varchar**(8) **primary key**,
    title **varchar(**50),
    dept_name **varchar**(20),
    credits **numeric**(2,0),
    foreign key *(dept_name*) **references**
department (dept_name) on delete set null);

```
mysql> insert into course (course_id, title, dept_name, credits) values
("BIO-101", "Intro to Bio", "Biology", 4);
Query OK, 1 row affected (0.01 sec)

mysql> insert into course (course_id, title, dept_name, credits) values
("BIO-101", "Intro to Bio", "Biology", 3);
ERROR 1062 (23000): Duplicate entry 'BIO-101' for key 'PRIMARY'
```

Insert Construct

Newly created relation empty
Use insert command to add tuples

```
create table instructor (
        ID          char(5),
        name        varchar(20) not null,
        dept_name   varchar(20),
        salary      numeric(8,2));
```

Insert Construct

insert into instructor (ID,name,dept_name,salary) values (‘10211’, ’Smith’, ’Biology’, 66000);

insert into instructor (ID,name,dept_name,salary) values (‘10211’, null, ’Biology’, 66000);

Drop and Delete Construct

drop table student

- Deletes the all tuples and the schema
- Table must be recreated in order to insert tuples after a drop command

delete from student

- Deletes all tuples, but retains the relation

Alter Table Construct

alter table

- ***alter table*** r ***add*** A D
  - where A is the name of the attribute to be added to relation r and D is the domain of A.
  - All tuples in the relation are assigned null as the value for the new attribute.
- ***alter table*** r ***drop*** A
  - where A is the name of an attribute of relation r
  - Dropping of attributes not supported by **SOME** databases (most support it)

MySQL Alter Syntax

```
alter_specification:  table_options
  | ADD [COLUMN] col_name column_definition
      [FIRST | AFTER col_name ]
  | ADD [COLUMN] (col_name column_definition,...)
  | ADD {INDEX|KEY} [index_name]
      [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]] PRIMARY KEY
      [index_type] (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
      UNIQUE [INDEX|KEY] [index_name]
      [index_type] (index_col_name,...) [index_option] ...
  | ADD FULLTEXT [INDEX|KEY] [index_name]
      (index_col_name,...) [index_option] ...
  | ADD SPATIAL [INDEX|KEY] [index_name]
      (index_col_name,...) [index_option] ...
  | ADD [CONSTRAINT [symbol]]
      FOREIGN KEY [index_name] (index_col_name,...)
      reference_definition
```

MySQL Alter Syntax

```
  | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP
DEFAULT}
  | CHANGE [COLUMN] old_col_name new_col_name column_definition
        [FIRST|AFTER col_name]
  | MODIFY [COLUMN] col_name column_definition
        [FIRST | AFTER col_name]
  | DROP [COLUMN] col_name
  | DROP PRIMARY KEY
  | DROP {INDEX|KEY} index_name
  | DROP FOREIGN KEY fk_symbol
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO|AS] new_tbl_name
  | ORDER BY col_name [, col_name] ...
```

MySQL Alter Syntax

```
  | CONVERT TO CHARACTER SET charset_name [COLLATE
collation_name]
  | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=]
collation_name]
  | DISCARD TABLESPACE
  | IMPORT TABLESPACE
  | ADD PARTITION (partition_definition)
  | DROP PARTITION partition_names
  | COALESCE PARTITION number
  | REORGANIZE PARTITION [partition_names INTO
(partition_definitions)]
  | ANALYZE PARTITION {partition_names | ALL}
  | CHECK PARTITION {partition_names | ALL}
  | OPTIMIZE PARTITION {partition_names | ALL}
  | REBUILD PARTITION {partition_names | ALL}
  | REPAIR PARTITION {partition_names | ALL}
  | PARTITION BY partitioning_expression
  | REMOVE PARTITIONING
```

Lecture Outline

- Overview
- Data Definition Language
- ***Data Manipulation Language***

Basic Query Structure

The SQL data-manipulation language (DML) provides the ability to query information, and insert, delete and update tuples

A typical SQL query has the form:

select $A_1, A_2, \ldots, A_n$
from $r_1, r_2, \ldots, r_m$
where P

A_n represents an attribute
r_m represents a relation
P is a predicate
The result of a SQL query is a relation

The select Clause

- The ***select*** clause list the attributes desired in the result of a query
  - Corresponds to the ***projection*** operation of the relational algebra
  - Example - Find the names of all instructors:

    select name
    from instructor
- NOTE: SQL names are case insensitive (i.e., you may use upper- or lower-case letters.)
  - E.g. Name ≡ NAME ≡ name
  - Some people use upper case wherever we use bold font.

The select Clause

- SQL allows duplicates in relations as well as in query results.
  - To force the elimination of duplicates, insert the keyword ***distinct*** after select.

Find the names of all departments with instructor, and remove duplicates:

select distinct dept_name
from instructor

The select Clause

- The keyword ***all*** specifies that duplicates not be removed
  - Not necessary since the default is to allow duplicates

select all dept_name
from instructor

The select Clause

- An asterisk in the ***select*** clause denotes "all attributes"

  Select *

  from instructor
- The select clause can contain arithmetic expressions involving the operation, +, –, *, and /, and operating on constants or attributes of tuples.

The query:

select ID, name, ***salary/12***

from instructor

would return a relation that is the same as the instructor relation, except that the value of the attribute salary is divided by 12..

The select Clause - Examples

select *
from instructor

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	NULL	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	NULL	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

The select Clause - Examples

select name
from instructor

```
+------------+
| name       |
+------------+
| Srinivasan |
| Wu         |
| Mozart     |
| Einstein   |
| El Said    |
| Gold       |
| Katz       |
| Califieri  |
| Singh      |
| Crick      |
| Brandt     |
| Kim        |
+------------+
```

The select Clause - Examples

select name, salary
from instructor

name	salary
Srinivasan	65000.00
Wu	90000.00
Mozart	40000.00
Einstein	95000.00
El Said	60000.00
Gold	87000.00
Katz	75000.00
Califieri	62000.00
Singh	80000.00
Crick	72000.00
Brandt	92000.00
Kim	80000.00

The select Clause - Examples

select distinct(salary)
from instructor

```
+----------+
| salary   |
+----------+
| 65000.00 |
| 90000.00 |
| 40000.00 |
| 95000.00 |
| 60000.00 |
| 87000.00 |
| 75000.00 |
| 62000.00 |
| 80000.00 |
| 72000.00 |
| 92000.00 |
+----------+
```

The where Clause

- The ***where*** clause specifies conditions that the result must satisfy
  - Corresponds to the ***selection predicate*** of the relational algebra.
- Comparison results can be combined using the logical connectives and, or, and not.
- Comparisons can be applied to results of arithmetic expressions

The where Clause

To find all instructors in Comp. Sci. dept with salary > 80000

> ***select*** name
> ***from*** instructor
> ***where*** dept_name = ‘Comp. Sci.' ***and*** salary > 80000

Query Multiple Relations

- Accessing information across relations
  - List in the ***from*** clause each relation to access
  - Specify matching condition using the ***where*** clause
  - Matching attribute occurs in both relations

To find all instructors in Comp. Sci. dept with salary > 80000

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

The from Clause

- The ***from*** clause lists the relations involved in the query
  - Corresponds to the Cartesian product operation of the relational algebra.

Find the Cartesian product instructor X teaches

```
select *
from instructor, teaches
```

  - generates every possible instructor – teaches pair, with all attributes from both relations
- Cartesian product not very useful directly, but useful combined with where-clause condition (selection operation in relational algebra)

Cartesian Product instructors x teaches

instructor

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

teaches

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

inst.ID	*name*	*dept_name*	*salary*	*teaches.ID*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2009
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2009
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2010
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2010
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2009

The from Clause

- Think of ***from*** clause with multiple relations as iterative process
  - For each tuple t_1 in relation r_1
    - For each tuple t_2 in relation r_2
      - ….
- Resulting relation has all attributes from all relations in from clause
- Use prefixes if attribute names are the same across relations in from clause

SQL Query

1. Generate Cartesian product from relations in ***from*** clause
2. Apply predicates from ***where*** clause
3. For each tuple, output attributes from ***select*** clause
4. Implementations differ for efficiency

Joins

- For all instructors who have taught some course, find their names and the course ID of the courses they taught.

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

Joins

- Find the course ID, semester, year and title of each course offered by the Comp. Sci. department

select section.course_id, semester, year, title
from section, course
where section.course_id = course.course_id and
dept_name = 'Comp. Sci.'

Natural Join

- Natural join operates on two relations and produces a result relation
- Natural join matches tuples with the same values for all common attributes, and retains only one copy of each common column

Natural Join

So instead of writing:

Select name, course_id
from instructor, teaches
where instructor.ID = teaches.ID;

We can write:

Select name, course_id
from instructor ***natural join*** *teaches;*

Natural Join

Select name, course_id
from instructor natural join teaches;

ID	*name*	*dept_name*	*salary*	*course_id*	*sec_id*	*semester*	*year*
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010

Natural Join

- The from clause can have a combination of relations using natural join

select $A_1, A_2, \ldots, A_n$
from r_1 ***natural join*** r_2 ***natural join*** …
natural join r_m
where P;

- Even more generally, a from clause can be in the form of ***from*** $E_1, E_2 \ldots E_n$

Natural Join

Let's compare:

select name, title
from instructor ***natural join*** teaches,course
where teaches.course_id = course.course_id;

select name, title
from instructor ***natural join*** teaches ***natural join*** course;

Comparing Natural Joins

Instructor

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

Teaches

ID	course_id	sec_id	semester	year
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
10101	CS-101	1	Fall	2009
45565	CS-101	1	Spring	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
10101	CS-315	1	Spring	2010
45565	CS-319	1	Spring	2010
98345	CS-319	2	Spring	2010
10101	CS-347	1	Fall	2009
98345	EE-181	1	Spring	2009
12121	FIN-201	1	Spring	2010
32343	HIS-351	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009

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

Comparing Natural Joins

- select name, title from instructor ***natural join*** teaches,course
- where teaches.course_id = course.course_id;

```
+------------+---------------------------+
| name       | title                     |
+------------+---------------------------+
| Srinivasan | Intro. to Computer Science |
| Srinivasan | Robotics                  |
| Srinivasan | Database System Concepts  |
| Wu         | Investment Banking        |
| Mozart     | Music Video Production    |
| Einstein   | Physical Principles       |
| El Said    | World History             |
| Katz       | Intro. to Computer Science |
| Katz       | Image Processing          |
| Crick      | Intro. to Biology         |
| Crick      | Genetics                  |
| Brandt     | Game Design               |
| Brandt     | Game Design               |
| Kim        | Image Processing          |
| Kim        | Intro. to Digital Systems |
+------------+---------------------------+
15 rows in set (0.00 sec)
```

- select name, title
- from instructor ***natural join*** teaches ***natural join*** course;

```
+------------+---------------------------+
| name       | title                     |
+------------+---------------------------+
| Srinivasan | Intro. to Computer Science |
| Srinivasan | Robotics                  |
| Srinivasan | Database System Concepts  |
| Wu         | Investment Banking        |
| Mozart     | Music Video Production    |
| Einstein   | Physical Principles       |
| El Said    | World History             |
| Katz       | Intro. to Computer Science |
| Katz       | Image Processing          |
| Crick      | Intro. to Biology         |
| Crick      | Genetics                  |
| Brandt     | Game Design               |
| Brandt     | Game Design               |
| Kim        | Intro. to Digital Systems |
+------------+---------------------------+
14 rows in set (0.00 sec)
```

Lecture Outline

- Overview
- Data Definition Language
- Data Manipulation Language

In Class Exercise

create a table called department_L5
with attributes: department name, building, budget

create a primary key

create a table called course_L5
with attributes: course id, title, department name, credits

create a primary key
reference the department table

create a table called instructor_L5
with attributes: name, department name, salary

create a primary key
reference the department table

In Class Exercise

Add 3 departments:

Department Biology is in the Watson building and has a 90000 budget.
Department Computer Science is in the Taylor building and has a budge of 100000.
Department Electrical Engineering is in the Taylor building and has a budget of 85000.

Add 5 instructors:

Dr. Katz works in Computer Science and earns a salary of 75000
Dr. Brandt works in Computer Science and earns a salary of 92000
Dr. Kim works in Electrical Engineering and earns a salary of 80000
Dr. Crick works in Biology and earns a salary of 72000
Dr. Wu works in Finance and earns a salary of 90000

What happened? How do you fix this problem?

Add 6 courses:

Course BIO-101 is the Introduction to Biology offered in the Biology department and worth 4 credits.
Course BIO-399 is the Computational Biology offered in the Biology department and worth 3 credits.
Course CS-190 is the Game Design offered in the Computer Science department and worth 4 credits.
Course CS-315 is the Robotics offered in the Computer Science department and worth 3 credits.
Course FIN-201 is the Investment Banking offered in the Finance department and worth 3 credits.
Course HIS-351 is the World History offered in the History department and worth 3 credits.

In Class Exercise

Add 2 additional departments to fix the insert problem:

Department History is in the Painter building and has a 50000 budget
Department Finance is in the Painter building and has a budge of 120000 budget

In Class Exercise

Create the following queries:

1. Select instructors with a salary greater than 75000
2. For each instructor select all the courses they could teach based on their department using Cartesian product and a where clause
3. For each instructor select all the courses they could teach based on their department using natural join
4. Select instructors working in the Taylor building
5. Select instructor names who could teach 4 credit courses in the Computer Science department
6. Select instructors who could teach Robotics course or the World History course
7. Delete the Finance department from the Department table, what happens?
8. Drop table Department, what happens?