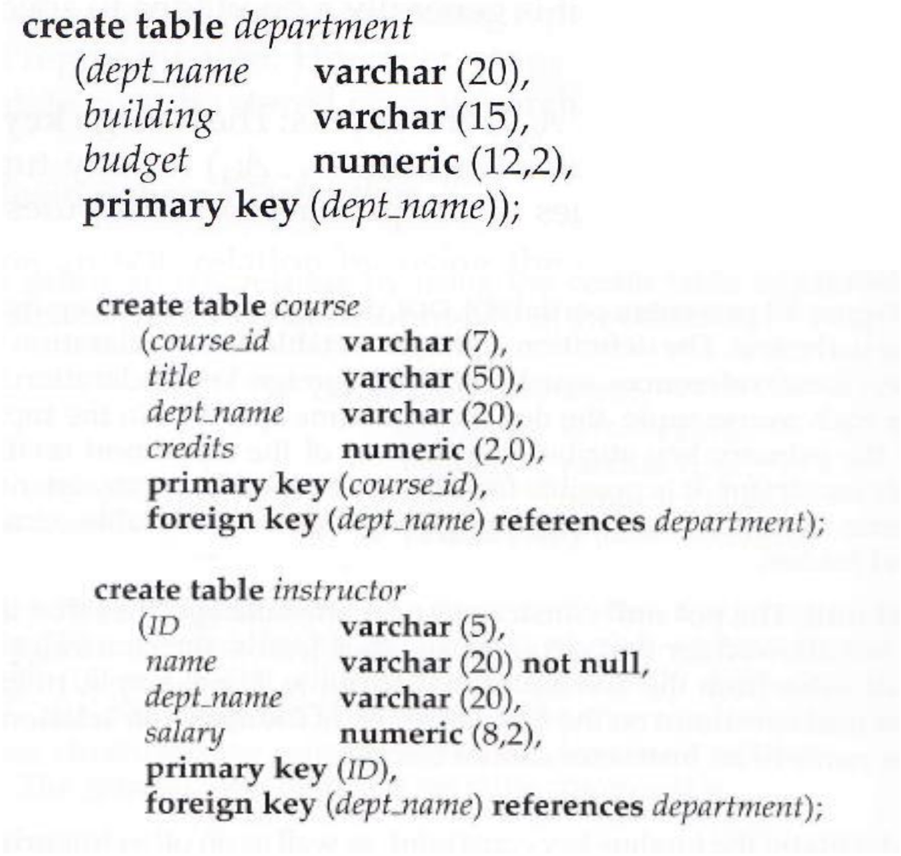
## 19CSE101 - Computer Systems Essentials

| Sl. No | Topics |
|---|---|
| 1 | Overview of SQL<br>Basic Data Types<br>Basic Schema Definition<br>Modification of the database<br>• Insertions<br>• Deletion<br>• Updates<br>Altering the Schema Definition<br>• Add a column<br>• Drop a column<br>• Modify the data type of a column |
| 2 | Queries on Single Relations<br>Queries on Multiple Relation - natural Join |
| 3 | Additional Basic Operations<br>• Rename Operations<br>• Attribute Specifications in select clause<br>• Where clause predicates |
| 4 | Set Operations |
| 6 | Aggregate Functions<br>• Aggregate Functions with grouping |

## List of Exercises and Evaluation Questions

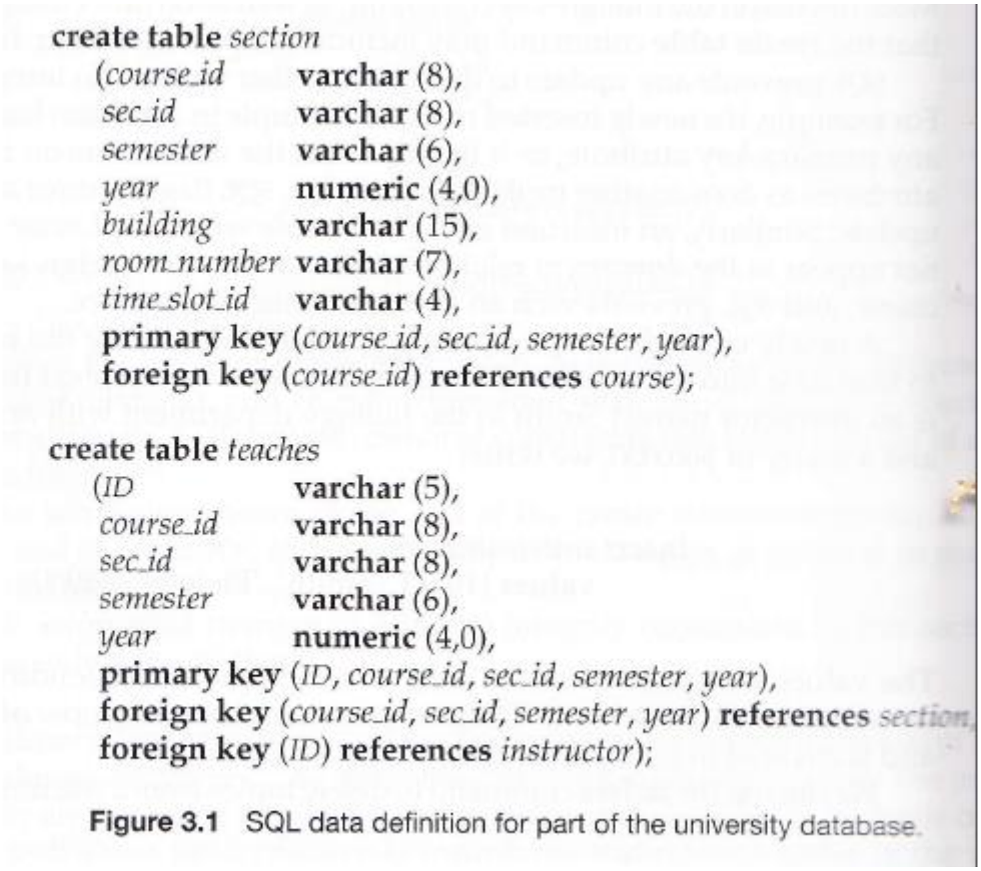
**<u>Exercise I</u> - Basic Schema Definition**



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

  create table course
      (course_id     varchar (7),
       title         varchar (50),
       dept_name     varchar (20),
       credits       numeric (2,0),
       primary key (course_id),
       foreign key (dept_name) references department);

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



```
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);

create table teaches
   (ID             varchar (5),
    course_id      varchar (8),
    sec_id         varchar (8),
    semester       varchar (6),
    year           numeric (4,0),
    primary key (ID, course_id, sec_id, semester, year),
    foreign key (course_id, sec_id, semester, year) references section,
    foreign key (ID) references instructor);
```

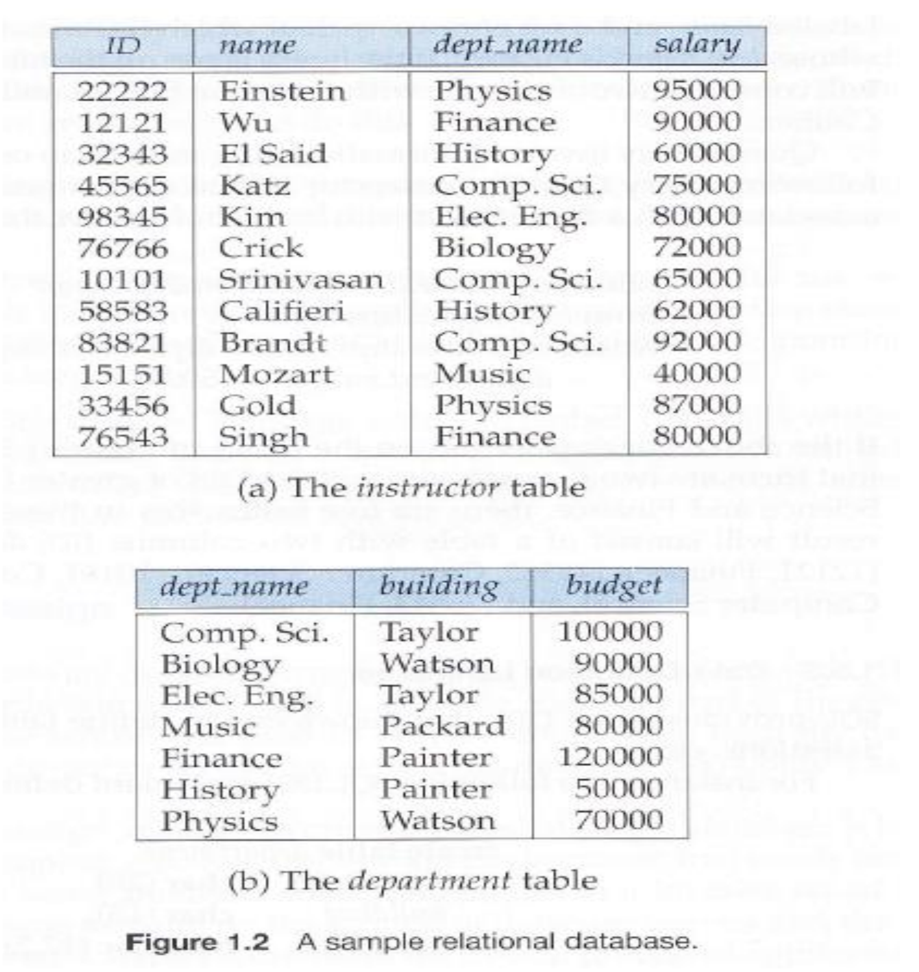
**Figure 3.1** SQL data definition for part of the university database.

**<u>Exercise II</u> - Insertion of Values.**

SQL >insert into instructor values (‘22222’,’Einsteing’,’Physics’,95000)

(or)

SQL >insert into instructor values (‘&ID’,’&name’,’&dept_name’,&salary)



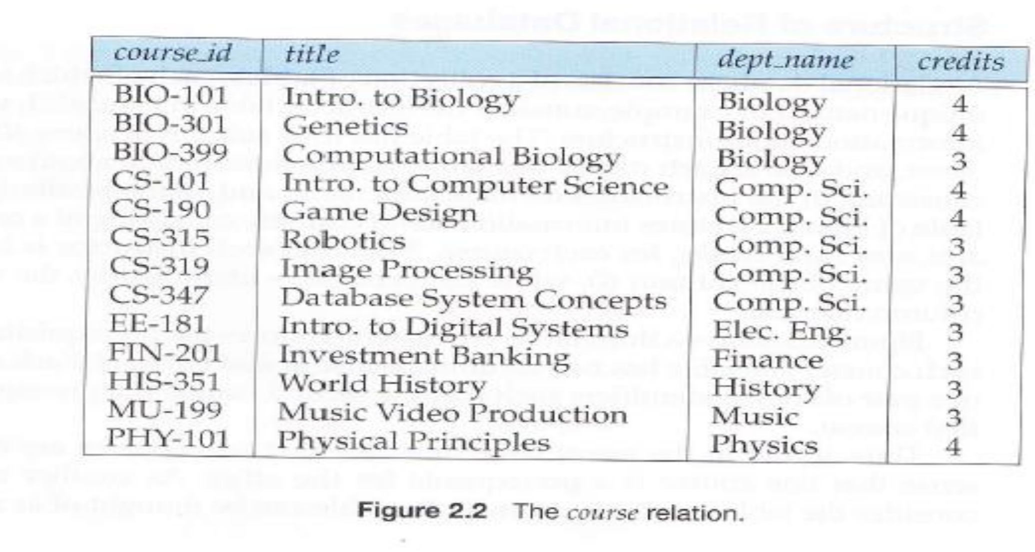
| *ID* | *name* | *dept_name* | *salary* |
|---|---|---|---|
| 22222 | Einstein | Physics | 95000 |
| 12121 | Wu | Finance | 90000 |
| 32343 | El Said | History | 60000 |
| 45565 | Katz | Comp. Sci. | 75000 |
| 98345 | Kim | Elec. Eng. | 80000 |
| 76766 | Crick | Biology | 72000 |
| 10101 | Srinivasan | 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

**Figure 1.2** A sample relational database.



| *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 |

**Figure 2.2** The *course* relation.

| *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 |

**Figure 2.3** The *prereq* relation.

| *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 |

**Figure 2.7** The *teaches* relation.

| *course_id* | *sec_id* | *semester* | *year* | *building* | *room_number* | *time_slot_id* |
|---|---|---|---|---|---|---|
| BIO-101 | 1 | Summer | 2009 | Painter | 514 | B |
| BIO-301 | 1 | Summer | 2010 | Painter | 514 | A |
| CS-101 | 1 | Fall | 2009 | Packard | 101 | H |
| CS-101 | 1 | Spring | 2010 | Packard | 101 | F |
| CS-190 | 1 | Spring | 2009 | Taylor | 3128 | E |
| CS-190 | 2 | Spring | 2009 | Taylor | 3128 | A |
| CS-315 | 1 | Spring | 2010 | Watson | 120 | D |
| CS-319 | 1 | Spring | 2010 | Watson | 100 | B |
| CS-319 | 2 | Spring | 2010 | Taylor | 3128 | C |
| CS-347 | 1 | Fall | 2009 | Taylor | 3128 | A |
| EE-181 | 1 | Spring | 2009 | Taylor | 3128 | C |
| FIN-201 | 1 | Spring | 2010 | Packard | 101 | B |
| HIS-351 | 1 | Spring | 2010 | Painter | 514 | C |
| MU-199 | 1 | Spring | 2010 | Packard | 101 | D |
| PHY-101 | 1 | Fall | 2009 | Watson | 100 | A |

**Figure 2.6** The *section* relation.

## Exercise III – Modification of the database

### Deletion:

```
        SQL >delete from instructor where dept_name = ‘Finance’;
        SQL >delete from instructor where salary between 1300 and 2000;
```

### Updates:

```
        SQL >update instructor set salary = salary + 1000 where salary<7000;
        SQL >update instructor set salary=1000 where name =’Raj’;
```

## Exercise IV - Altering the Schema Definition

```
SQL> create table AAA(a varchar(10));
SQL> desc AAA;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 A                                                  VARCHAR2(10)
```

### To add a column in a table:

```
SQL> alter table AAA add b varchar(10);

SQL> desc AAA;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 A                                                  VARCHAR2(10)
 B                                                  VARCHAR2(10)
SQL> alter table AAA add c numeric(10,3);

Table altered.

SQL> desc AAA;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 A                                                  VARCHAR2(10)
 B                                                  VARCHAR2(10)
 C                                                  NUMBER(10,3)
```

### To delete a column in a table

```
SQL> alter table AAA drop column b;

Table altered.

SQL> desc AAA;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 A                                                  VARCHAR2(10)
 C                                         NUMBER(10,3)
```

**<u>To change the data type of a column in a table,</u>**

```
SQL>alter table AAA modify c varchar(10);

SQL> desc aaa;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 A                                         VARCHAR2(10)
 C                                         VARCHAR2(10)
```

## <u>Exercise V</u> – Queries

### Queries on Single Relations

1) Find the name of all the instructors

   ```
   SQL >select name from instructors;
   ```

2) Find the department name of all the instructors

   ```
   SQL >select dept_name from instructors;
   ```

3) Select clause with arithmetic expression

   ```
   SQL >select name,salary*1.1 from instructors;
   ```

4) Where clause with predicates

   ```
   SQL >select name from instructor where dept_name=’Comp.Sci.’ and salary>7000;
   ```

### Queries on Multiple Relations

1) Retrieve the names of all instructors along with their department names and department building name.

   ```
   SQL >select name, instructors.dept_name,building from instructors, department where
   instructors.dept_name=department.dept_name;
   ```

2) **Understanding Natural Joins.**

   ```
   SQL >select * from Table1 natural join Table2;
   ```

3) **Using Natural Joins.**

```
SQL >select name, instructor from instructor natural join teaches;
```

## Exercise VI – Additional Basic Operations

### (a) Relational operations

1. **“Select the name of all the instructors”**

   ```
   SQL >Select name as instructor_name from instructor;
   ```

2. “For all the instructor in the university who have taught some course find their names and the course_id of all the course they taught”

   ```
   SQL >Select T.name, S.course_id from instructor T, teaches S where T.id = S.id;
   ```

3. “Find the names of all the instructor whose salary is greater than at least one instructor in the biology department”

   (to compare the tuples in the same relations)

   ```
   SQL >Select distinct T.name from instructor S, instructor T where T.salary >
   S.salary and S.dept_name='Biology’;
   ```

   **Patterns**

   % - matches any substring
   _ - matches any character

   Examples
   (1) ‘Intro%’ - matches any string beginning with ‘Intro’
   (2) ‘%Comp%’ - matches any string that has Comp as substring
   (3) ‘___’ - matches any string with exactly three characters
   (4) ‘___%’ - matches any string with at least three characters
   Additional Examples (using escape character)
   (5) ‘ab\%cd%’ - matches strings begin with *ab%cd*
   (6) ‘ab\\cd%’ - matches strings begin with *ab\cd*

### Example Queries

1. “Find the names of all departments whose building name includes the substring ‘Watson’;

   ```
   SQL >select dept_name from department where building like
   ‘%watson%’;
   ```

2.
   ```
   SQL >select name from instructor where name like ‘K%’;
   ```

3. SQL >select name from instructor where name like ‘K_’;

**(b) Attribute Specifications in select clause**

SQL > select * from instructor;
SQL > select instructor.* from instructor,teaches where instructor.id =teaches.id;

**(c) Where clause predicates**

1. SQL >Select name from instructor where dept_name =’physics’;

2. SQL >select dept_name from department where building=’Taylor’ or building=’Watson’;

3. SQL >select name from instructor where salary <=10000 and salary >=20000;

4. SQL >select name, course_id from instructor, teaches where instructor.id=salry.id and dept_name=’Biology’;

(or)

SQL >select name,course_id from instructor, teaches where (instructor.id, dept_name) =(teaches.id,’Biology’);

5. SQL >select name from instructor where salary between 10000 and 20000;

6. SQL >select name from instructor where salary in (60000,80000,40000);

7. SQL >select name from instructor where dept_name is null;

8. SQL >select name from instructor where dept_name is not null;

## Exercise VII – Set Operations

**Three operations**

- Union
- Intersection
- Minus

### A Simple Example

```
>create table AAA(a varchar(10));

>create table BBB(a varchar(10));

>insert into AAA values (‘&a’);
a
b
c
```

```
d
e
e
```

```
>insert into BBB values (‘&a’);
a
b
f
a
```

```
>(select * from AAA) union(select * from BBB);
```

**Result:**

```
a
b
c
d
e
f
```

```
>(select * from AAA) union all(select * from BBB);
```

**Result:**

```
a
b
c
d
e
e
a
b
f
a
```

```
>(select * from AAA) intersect (select * from BBB);
```

**Result:**

```
a
b
```

```
(select * from AAA) minus(select * from BBB);
```

**Result:**

```
c
d
e
```

**Example for set operations based on the university database**

(a) Union

**Example**

To find Set of all courses taught in Fall 2009 or Spring 2010 semesters.

(select course_id from section where semester=’Fall’ and Year=2009)union(select course_id from section where semester=’Spring’ and year=2010);

- The union operation eliminate duplicates, to retain all the duplicates we must use *union all*.

  (select course_id from section where semester=’Fall’ and Year=2009)union all(select course_id from section where semester=’Spring’ and year=2010);

(b) Intersection

**Example**

To find Set of all courses taught in Fall 2009 and Spring 2010 semesters.

(select course_id from section where semester='Fall' and Year=2009)intersect(select course_id from section where semester='Spring' and year=2010);

(c) Minus

**Example**

To find Set of all courses taught in Fall 2009 but not in Spring 2010 semesters.

(select course_id from section where semester=’Fall’ and Year=2009)minus (select course_id from section where semester=’Spring’ and year=2010);

**Exercise VIII – Aggregate Functions**

These functions take a collection of values as input and return a single value.

(a) avg (b) min (c) max (d) sum (e) count

**(a) Basic aggregation**

a. SQL>select avg(salary) from instructor;

b. SQL>select avg(salary) as avg_salary from instructor;

c. SQL>select count(distinct ID) from teaches where semester=’spring’ and year = 2010;

d. SQL>select count(*) from course

e. SQL>select min(salary) from instructor;

f. SQL>select max(salary) from instructor;

g. SQL>select sum(salary) from instructor;

### (b) Aggregation with Grouping

a. “To find the average salary of each department;

```
SQL>select dept_name, avg(salary)group_by dept_name;
```

b. “To find the number of instructors in each department who teach a course in the Spring 2010 semester”

```
SQL> select dept_name, count(distinct ID) from instructor natural join
teaches where semester=’Spring’and year =2010 group by dept_name;
```

Note : When using grouping the attribute that appear in the select clause outside the aggregate function should present in the group by clause.

Example: (understand the error in the following query)

```
SQL>select dept_name, ID, avg(salary) from instructor group by dept_name;
```