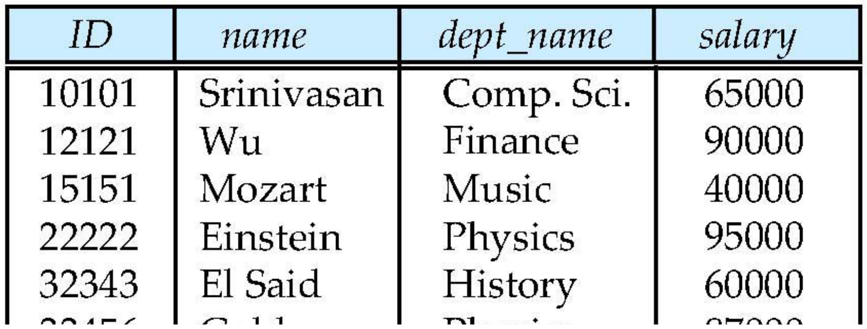
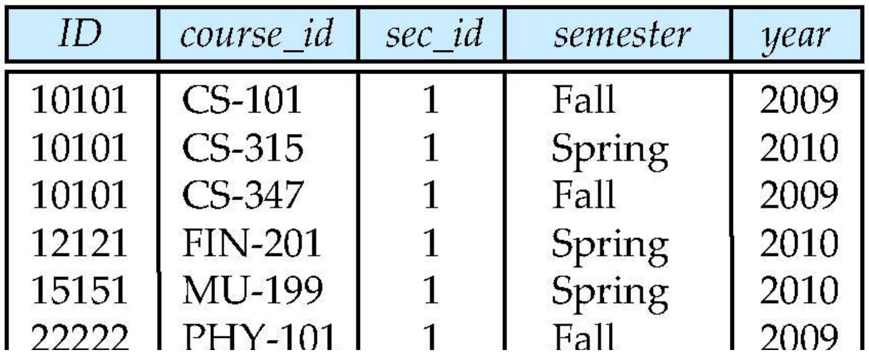
SQL Schema Changes and table updates

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

Table Creation

- **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) );
  ```
  - Primary key declaration can be combined with attribute declaration as shown above

Drop and Alter Table Constructs

- **drop table** *student*
  - Deletes the table and its contents
- **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 many databases

Modification of the Database

- Deletion of tuples from a given relation
- Insertion of new tuples into a given relation
- Updating values in some tuples in a given relation

Modification of the Database – Deletion

- Delete all instructors

  delete from *instructor*

- Delete all instructors from the Finance department

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

- Delete all tuples in the *instructor* relation for those instructors associated with a department located in the Watson building.

  delete from *instructor*
  where *dept_name* **in** (**select** *dept_name*
  from *department*
  where *building* = ' Watson');

Deletion (Cont.)

- Delete all instructors whose salary is less than the average salary of instructors

delete from *instructor*
where *salary* < (**select avg** (*salary*) **from** *instructor*);

- Problem: as we delete tuples from deposit, the average salary changes
- Semantics used in SQL: assume that
  1. you first, compute **avg** salary and find all tuples to delete
  2. then, you delete all tuples found above

  (without recomputing **avg** or retesting the tuples)

Modification of the Database – Insertion

- Add a new tuple to *course*

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

- or equivalently

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

- Add a new tuple to *student* with *tot_creds* set to null

  insert into *student*
      values ('3003', 'Green', 'Finance', *null*);

Insertion (Cont.)

- Add all instructors to the *student* relation with tot_creds set to 0

  insert into *student*
      select *ID, name, dept_name, 0*
      from *instructor*

- The **select from where** statement is evaluated fully before any of its results are inserted into the relation (otherwise queries like

      insert into *table*1 **select** * **from** *table*1

  would cause problems, if *table1* did not have any primary key defined).

Modification of the Database – Updates

- Increase salaries of instructors whose salary is over $100,000 by 3%, and all others receive a 5% raise
  - Write two **update** statements:

```
update instructor
     set salary = salary * 1.03
     where salary > 100000;
 update instructor
      set salary = salary * 1.05
      where salary <= 100000;
```

The order here is important !

  - A better way**:** use the **case** statement

```
update account
set balance = case  when balance <= 10000  then balance *1.05
                                     else balance * 1.06
                          end
```

String Operations

- SQL includes a string-matching operator for comparisons on character strings. The operator "like" uses patterns that are described using two special characters:
  - percent (%). The % character matches any substring.
  - underscore (_). The _ character matches any character.
- Find the names of all instructors whose name includes the substring "dar".

  select *name*
  from *instructor*
  where *name* **like** '%dar%'

String Operations (Cont.)

- Patters are case sensitive.
- Pattern matching examples:
  - 'Intro%' matches any string beginning with "Intro".
  - '%Comp%' matches any string containing "Comp" as a substring.
  - '_ _ _' matches any string of exactly three characters.
  - '_ _ _ %' matches any string of at least three characters.
- SQL supports a variety of string operations such as
  - concatenation (using "II")
  - converting from upper to lower case (and vice versa)
  - finding string length, extracting substrings, etc.