APAN PS5400: Managing Data

Week 3: Creating Relational Databases

Lecturer: François Scharffe

COLUMBIA UNIVERSITY
School of Professional Studies

Recap of last week

- Relational data model
- Relational databases
- Relational Database Management Systems (RDBMS)
- SQL as a query language

This week

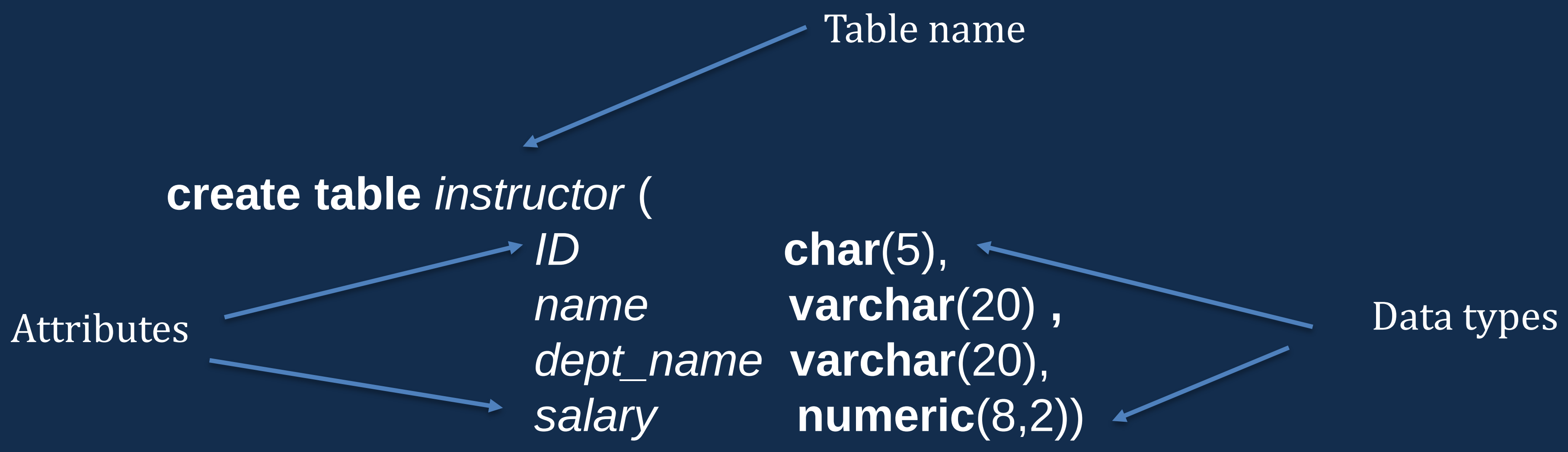
- Creating relational databases using SQL
- Primary key of a table
- Foreign key constraints and referential integrity
- Other types of constraints
- Insertion of data into tables
- Modifying data
- Deleting tuples
- Using Postgres SQL to create databases on your computer

SQL as Data Definition Language

Can be used to create relations as well as additional information about relations, such as:

- The schema of the relation
- The data types of the values to be stored in the relation
- The integrity constraints on the relation
- The set of indices on the relation
- Information about access privileges and security
- The physical storage structure of the relation on the disk

Create table



```
create table instructor (
        ID              char(5),
        name            varchar(20) ,
        dept_name       varchar(20),
        salary          numeric(8,2))
```

Most basic form. We can add various types of constraints to this basic form. See next few slides.

Create table with primary key constraint

```
create table instructor (
                ID           char(5),
                name         varchar(20) ,
                dept_name    varchar(20),
                salary       numeric(8,2),
                primary key (ID),
                  )
```

Primary key constraint

What is a primary key? See next.

Columbia University
School of Professional Studies

Primary keys

- Recall that a table (i.e., a relation) is just a set of tuples and a set cannot have duplicate tuples.
- This constraint on tables can be enforced by making a set, K, of attributes of the relation, R, such that no two tuples in R can have the same set of values on K.
    - In the *create table* statement of the previous slide we specify that no two tuples of the *instructor* table can have the same value on ID attribute.

Primary Keys: Formal definition

- Let K $\subseteq$ R, where R is the attributes of a relation
- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple in each possible instance of R.
  - Which implies that there cannot be two tuples in any instance of R with the same values on K, given that a set cannot have duplicates.
  - E.g., {ID} and {ID, name} are both superkeys of R
- A superkey K is a **candidate key** if no proper subset of it is itself a superkey (i.e., K is minimal)
- Any candidate key can be chosen as the **primary key** for R
  - ID is a candidate key and, thus, can be the primary key
  - If there are more than one candidate keys, then choose the one on which you want to build an index as the primary key

What if we don’t declare a primary key when creating a table? Then will the table allow duplicate rows?

How do you decide what should be primary key? (Recall we can have a PK that is a composite of several attributes.)
It depends on the rules of the organization (or the world) that the database is modelling?

Enrollment: (Student, Course, Section, Semester)

- What if you make Student the PK?
- What if you make {Student, Course} the PK?
- What if you make {Student, Course, Section} the PK?
- What if you make {Student, Course, Semester} the PK?
- What if you make {Student, Semester} the PK?
- What if you make {Student, Course, Section, Semester} the PK?

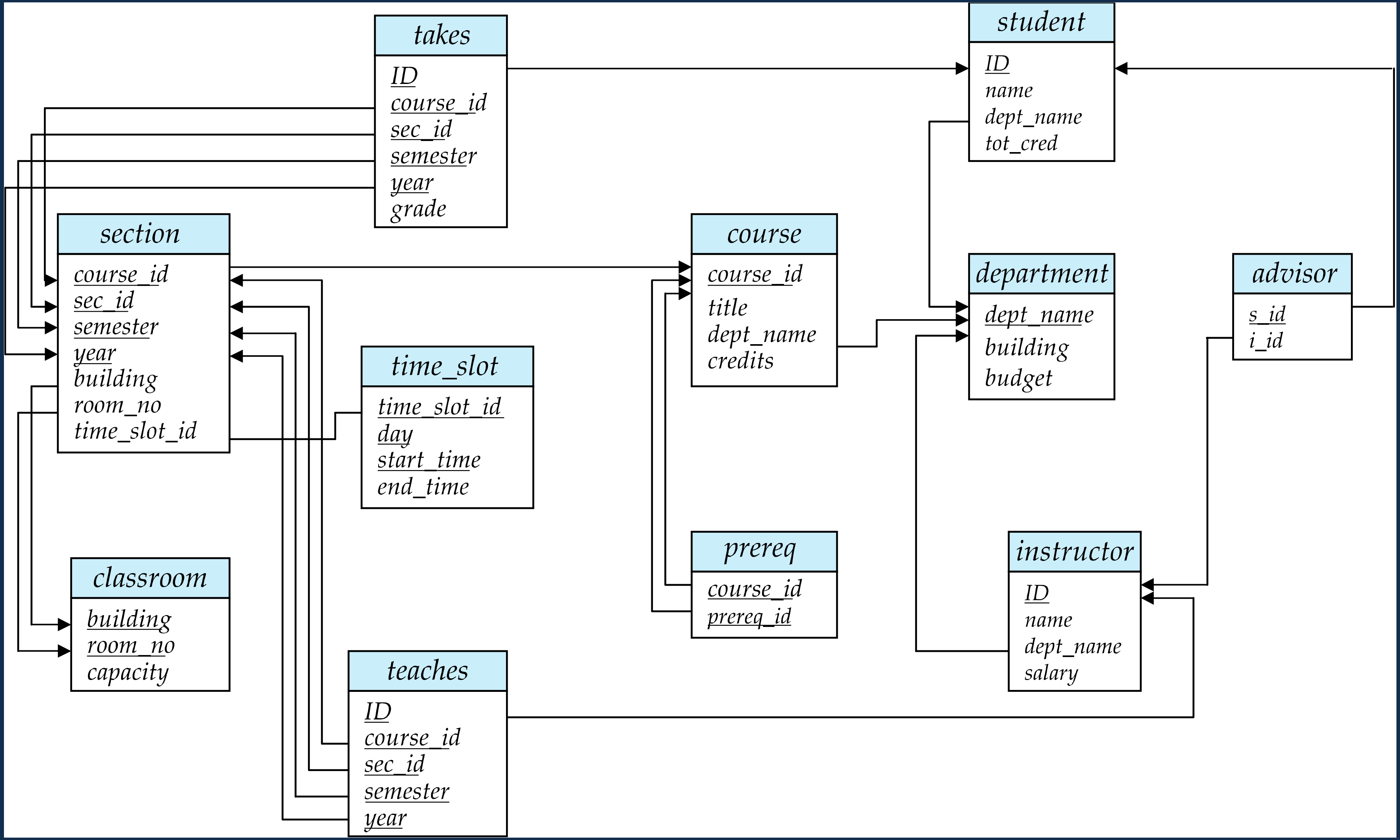
Create Table with Foreign Key

```
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)
```

Foreign key constraint

Foreign key constraint and Referential Integrity

- A table (e.g., *instructor)* may have an attribute (e.g., *department_name*) such that whether a value of that attribute in that table is a valid value or not has to be checked in terms of whether that value occurs in another table (e.g., *department)*
- The other table is regarded as the validator of such values in any other table.
- In this case we say there is a foreign key reference from any table which has that attribute to the validator table.
- A database in which all foreign key constraints are respected is said to have **referential integrity.**
- In the SQL code for creating a table we can specify what action to take if insertion of data violates a foreign key constraint on table: Block the insertion? Take a corrective action on the validator table? No action?
  - This will be covered in the elective course on SQL.



The schema of an RDB with foreign key constraints. An arrow from a table to another table indicates that there is a foreign key constraint from the first table to the second table.

Thus, we see that there is a foreign key reference from *course* to *department.*

Create table with not null constraint

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

This constraint specifies that *name* cannot take null values.
Primary key field is not allowed to take null values.

Null values

- Some of the attributes in some of the tuples in a relation can have **null values.**
- null denotes an unknown value or a value that doesn’t exist.
- Arithmetic expressions with a null value results in a null value
- Aggregate functions simply ignore null values
- P is not null return false if P is null, and true otherwise

Other constraints

- Check constraints
- Assertions
- Triggers
- Not all RDBMS support these constraints
- These will be covered in the SQL elective course

Inserting data,

- insert into course values (‘APAN5400’, ‘Managing Data’, ‘APAN’, 3), or equivalently
- insert into course (*course_id, title, dept_name, credits) values*

(‘APAN5400’, ‘Managing Data’, ‘APAN’, 3)

- Add a new tuple to student with tot_cred set to null

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

Updates

The general form is:

update *table_name*

set *attribute_value* = value

where *boolean condition*

Example:

update *instructor*

set *salary = salary * 1.5*

where *salary* < 50,000

Deletes

- General form:
  delete from *table* where *boolean condition*

- Example:
  delete from *instructor* where *salary > 75000*

Demo

- We will spend the rest of the class doing a demo of how to install Postgres on a computer, and
- Creating a sample database
- Pay attention—the first part of your first project (Project 1.1) requires you to master this.

Recap: What we covered this week

- SQL to create tables
- Primary key of a table
- Foreign key constraints and referential integrity
- Other types of constraints
- Insertion, update, delete

Next week

- Views
- Queries using views
- Functional dependencies
- BCNF
- Decomposition of tables to make database BCNF compliant