CSc 484
Database Management Systems

Ken Gamradt

Spring 2024

Introduction to SQL (II)

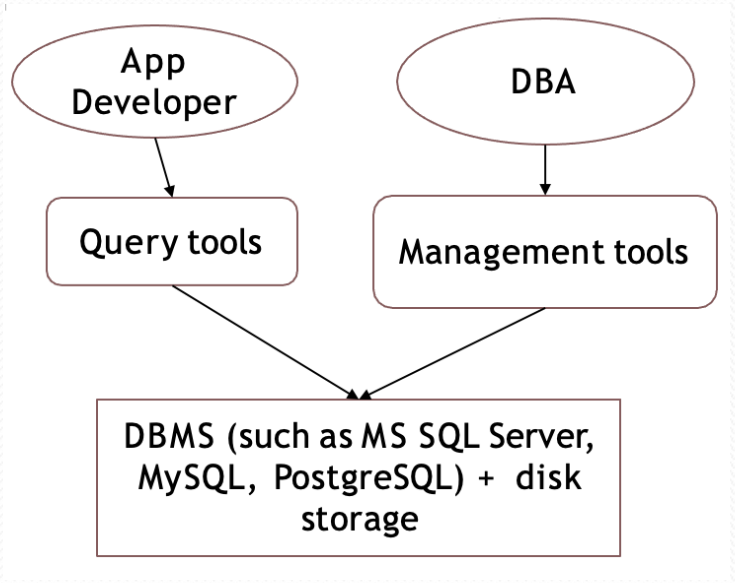
SQL

- A **standard** relational database language managed by **ANSI/ISO**
  - Most up-to-date SQL standard is SQL:2016
- Supported by most relational database systems
- Has two major components
  - **Data-Definition Language** (DDL)
    - Define the structure of the data
  - **Data-Manipulation Language** (DML)
    - Modify the data in the database, specify security constraints, …

SQL

- SQL can be used in two ways:
  - **Interactively** by entering statements at a terminal or management tools
  - **Embed** SQL statements in a general-purpose programming language

```
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
);
```


App Developer
DBA
Query tools
Management tools
DBMS (such as MS SQL Server, MySQL, PostgreSQL) + disk storage

SQL

- ISO specifies the **format** and **syntax rules** of SQL statements
  - Vendors implement them to process the SQL statement
- Supported by most database systems
- Different vendors may have different implementations of SQL
  - Features that are provided on top of the standard are called **extensions**
    - T-SQL (Transact-SQL) is the extension of SQL that is used in Microsoft
- Each implementation of SQL is called a **dialect**
- Might support non-standard features but omitted some of the more advanced and more recent standard features

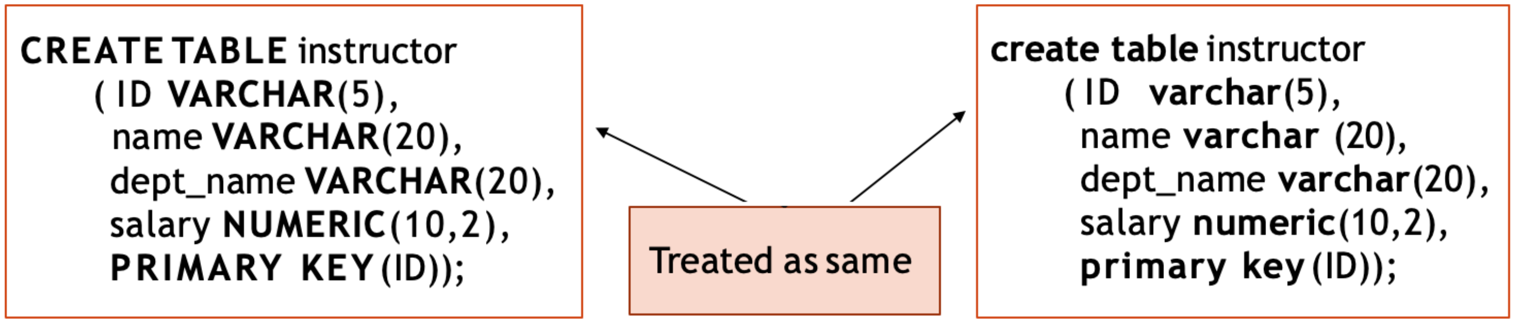
- We will learn SQL's fundamental constructs and concepts

Basic Data Types

- **Character data:**
  - **char**(n): with fixed-length of n. Full form: character(n)
  - **varchar**(n): with variable length of maximum length n
    - Full form: character varying (n)
- **Numeric data:**
  - **int**: an integer. Full form: integer
  - **smallint**: a small integer
  - **numeric(p, d)**:
    - **p**: total number of digits
    - **d**: total number of digits to the right of the decimal point
  - **float(n)**: float-pointing number with precision of at least n digits
  - **real**, **double precision**: float-pointing numbers

SQL Statements

- **Reserved words (key words)**: (select, from, and, or, where, insert, into, …)
  - **Not case-sensitive**
  - Usually written with uppercase
- **Identifiers**: to identify objects in the database
  - table names, view names, and columns
  - **Not case-sensitive** by default unless within quotes
  - Consists of [A…Z],[a…z],[0…9][_], start with a letter, length is no longer than 128 characters



```
CREATE TABLE instructor
      ( ID VARCHAR(5),
        name VARCHAR(20),
        dept_name VARCHAR(20),
        salary NUMERIC(10,2),
        PRIMARY KEY (ID));
```

```
create table instructor
      ( ID   varchar(5),
        name varchar (20),
        dept_name varchar(20),
        salary numeric(10,2),
        primary key (ID));
```

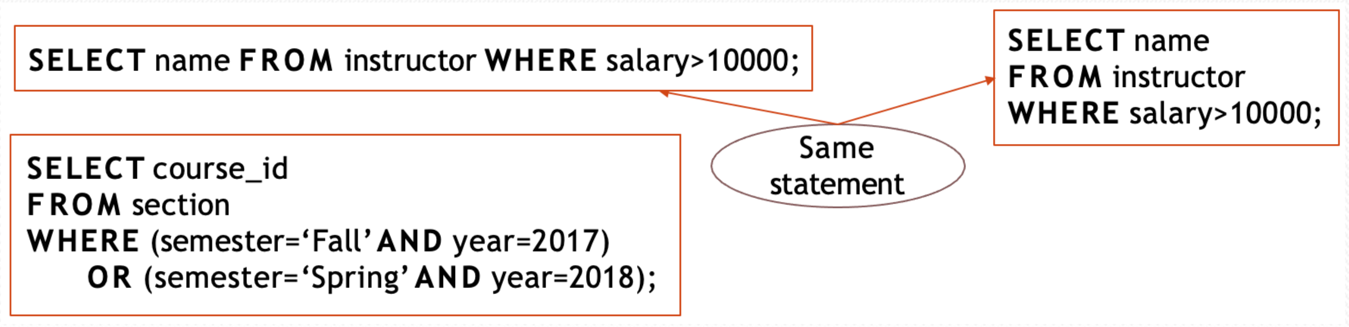
SQL Statements

- **Constants** (data to be updated to the database)
  - **Non-numeric**: within single quotes, case-sensitive
  - **Numeric**: without single quotes
- Many dialects of SQL require the use of a **statement terminator** to mark the end of each SQL statement
  - Usually use semicolon ";"

```
insert into instructor
        values ('10001', 'Gamradt', 'EECS', 100);
```

SQL Statements

- **Free-format**
  - The parts of the statements do not have to be typed at particular locations on the screen
- **Suggestions** on the format:
  - Each clause in a statement should begin on a new line
  - The beginning of each clause should line up with the beginning of other clauses
  - If a clause has several parts, they should each appear on a separate line indented under the start of the clause to show the relationship



```
SELECT name FROM instructor WHERE salary>10000;
```

```
SELECT name
FROM instructor
WHERE salary>10000;
```

```
SELECT course_id
FROM section
WHERE (semester='Fall' AND year=2017)
     OR (semester='Spring' AND year=2018);
```

Database Lifecycle

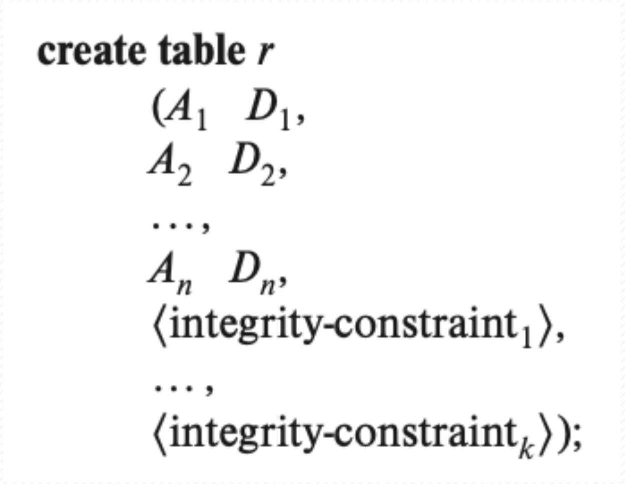
- Database design
- Create schema using DBMS commands (DDL)
  - Create database
  - Create relations
- Load data (DML)
- Be connected by applications to be retrieved and updated

Create Database

- ***create database*** *identifier;*
  - Supply whatever name you wish to call your database

Create Table

- General form for create table:



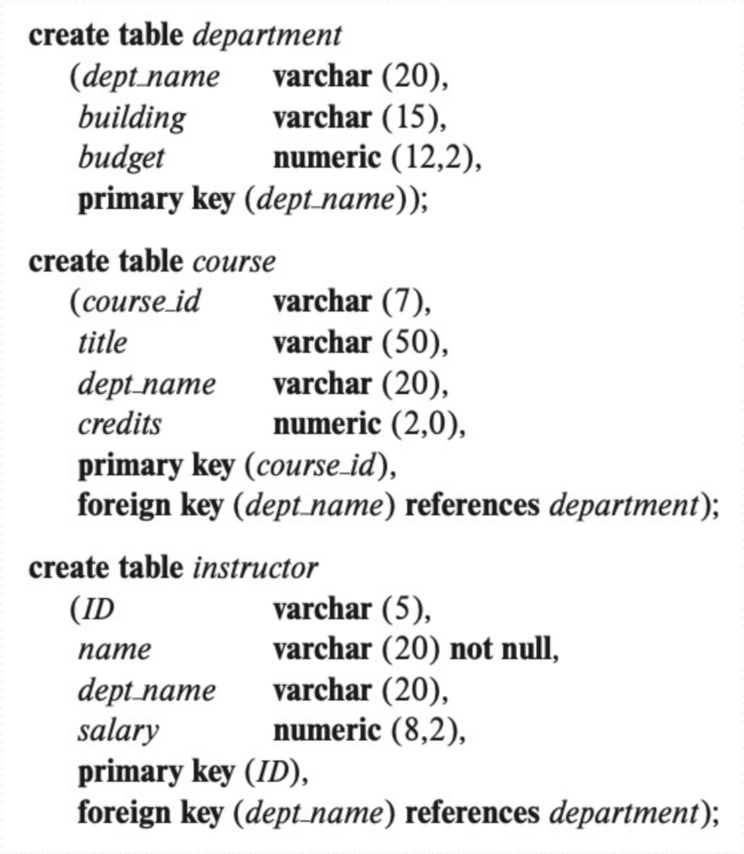
```
create table $r$
        ($A_1$   $D_1$,
        $A_2$   $D_2$,
        . . . ,
        $A_n$   $D_n$,
        ⟨integrity-constraint$_1$⟩,
        . . . ,
        ⟨integrity-constraint$_k$⟩);
```

- r: table name
- A_1, … , A_n: attributes
- D_1, … , D_n: attribute types

Create Table

- Integrity constraints:
  - **PRIMARY KEY** (Ak_1, …, Ak_m)
    - primary key attributes are required to be non-null and unique by default only one **PRIMARY KEY** clause is allowed per table
  - **FOREIGN KEY** (Ak_1, …, Ak_s) **REFERENCES** s (FK_1,..FK_n);
    - s: referenced relation
    - list of foreign key columns can be eliminated in some systems
    - foreign key can be null or any value from referenced relation's primary key
    - multiple **FOREIGN KEY** clauses allowed as needed
  - **NOT NULL**
    - specify the null value is not allowed for that attribute
- Foreign key list must have the same data type as referenced primary key list

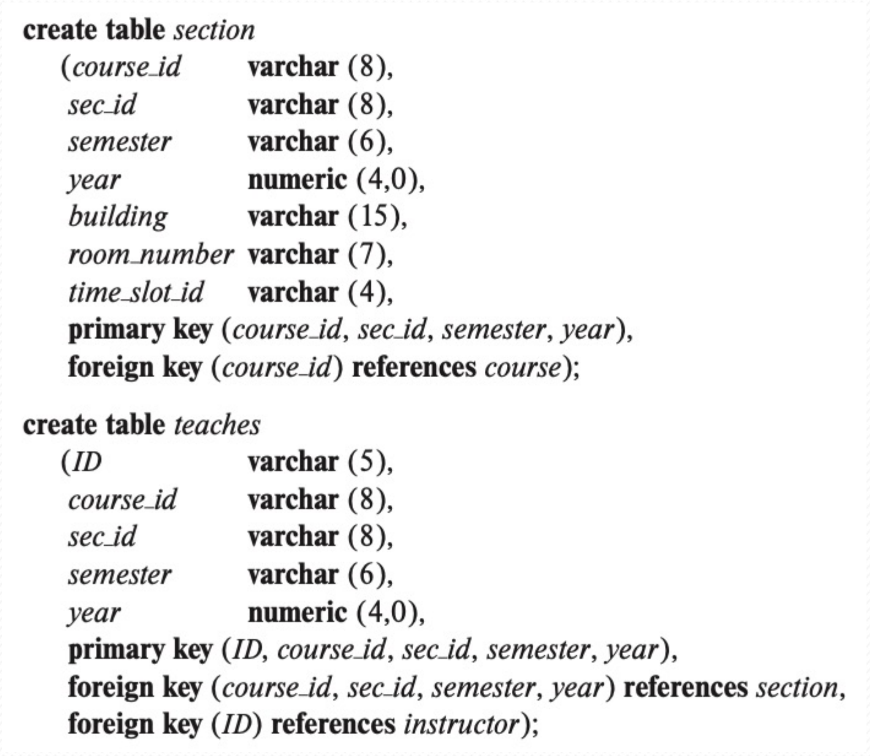
Create Table



```
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);
```

SQL Commands

create table TableName

{(columnName dataType [**not null**][**unique**]
[**default** defaultOption][**check** (searchCondition)][, ...]}
[**primary key** (listOfColumns),]
{[**unique** (listOfColumns)][, ...]}
{[**foreign key** (listOfForeignKeyColumns)
references ParentTableName [(listOfCandidateKeyColumns)]
[**match** {**partial** | **full**}
[**on update** referentialAction]
[**on delete** referentialAction]][, ...]}
{[**check** (serachCondition)][, ...]});

Delete From

- Remove a relation from an SQL database
  - Of course, the data is gone too

drop table *r;*

drop _table_ *department;*

Drop Table

- Remove a relation from an SQL database
  - Of course, the data is gone too

drop from *r;* **SAME** ***drop table*** *r;*

delete *r;*

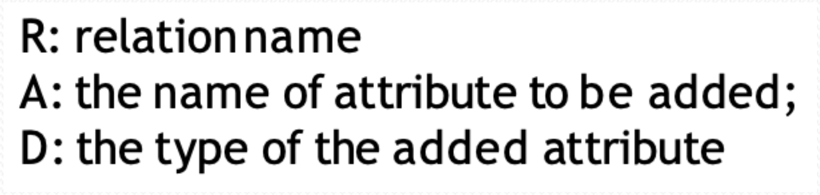
- Not able to operate ***delete*** commands directly on a relation whose primary key is used as a **foreign key** elsewhere

- Solution: remove the **foreign key** constraint

Alter Table

- Add attributes to an existing relation

  alter table *r* ***add*** *A D;*



R: relation name
A: the name of attribute to be added;
D: the type of the added attribute

- Drop attributes from an existing relation

  alter table *r* ***drop*** *A;*

  T-SQL: ***alter table*** *r* ***drop column*** *A;*

Insert

- Insert data into an existing relation

```
insert into table_name [(columnList)]
    values (dataValueList);

insert into course
    values ('csc-484', 'Database Systems', 'EECS', 3);

insert into course (course_id, title, dept_name, credit)
    values ('csc-484', 'Database Systems', 'EECS', 3);

insert into course (course_id, title, dept_name)
    values ('csc-484', 'Database Systems', 'EECS'),
           ('csc-300', 'Data Structures', 'EECS');
```

Insert

- columnList:
  - represents a list of one or more column names separated by commas
  - can be omitted if you insert values for all columns
  - attributes missing from the column list will have **null** inserted by default
    - if allowed
    - you can change default value
  - the order of attributes in column list can be different from the order in the relation schema
  - the order of data must match the order of the attributes listed

Acknowledgements

- WIKIPEDIA
  - https://en.wikipedia.org/wiki/SQL
- ANSI
  - ISO
  - https://www.ansi.org/iso/us-representation-in-iso/introduction
  - SQL
  - https://webstore.ansi.org/industry/software/technology-languages/SQL