MySQL–Creating a Database

Lecture 3
Section 4.1

Robb T. Koether

Hampden-Sydney College

Mon, Jan 21, 2013

Outline

Multiple Tables

- Suppose that we wish to add to the database the information about the departments, the employees' dependents, and the various projects that the employees are working on.
- We need additional information
  - Department's name
  - Department's manager
  - Dependent's name
  - Dependent's sex
  - Dependent's birthday
  - Project's name
  - Project's department
  - Employees working on each project
  - No. of hours worked by employees on each project

Relational Databases

- How should we incorporate this additional information into the database?

Relational Databases

- To add all of those attributes to the Employees relation would violate the *limited redundancy* principle.
- All of the related data for each department and project would be repeated for each employee working on that project.
- Furthermore, all the related data for each employee would be repeated for each dependent of that employee.

Relational Databases

- Clearly, the better solution is to create a new tables of departments, projects, dependents, and works.

```
Employees
(
    fname string,
    lname string,
    ssn string,
    bdate date,
    sex string,
    salary float,
    dept integer
)
```

Relational Databases

- Clearly, the better solution is to create a new tables of departments, projects, dependents, and works.

```
Departments
(
    dept integer,
    d_name string,
    mgr_ssn string
)
```

Relational Databases

- Clearly, the better solution is to create a new tables of departments, projects, dependents, and works.

```
Projects
(
    proj integer,
    p_name string,
    dept integer
)
```

Relational Databases

- Clearly, the better solution is to create a new tables of departments, projects, dependents, and works.

```
Dependents
(
    ssn string,
    dep_name string,
    dep_sex string,
    dep_bdate date
)
```

Relational Databases

- Clearly, the better solution is to create a new tables of departments, projects, dependents, and works.

```
Works
(
    ssn string,
    proj integer,
    hours float
)
```

Outline

MySQL Commands

- MySQL commands may be issued through a command line or from within a program (e.g., C, C++, Java, PHP).
- When issued through the command line, all MySQL commands end with a semicolon.
- When using MySQL through a programming language such as C++, a complication is that the internal structure of MySQL relations does not match any of the built-in datatypes.
- What to do?

Outline

Creating a Database

Creating a Database

```
CREATE DATABASE db_name;
```

- The `CREATE DATABASE` command will create a database.
- This creates a database with the name *db_name*, but it does not create any content.

Creating a Database

The Current Database

```
USE db_name;
```

- To work with a database, we execute the `USE` command to make it the current database.
- This makes *db_name* the current database.

Creating Tables

Creating a Table

```
CREATE TABLE table_name (attribute_list);
```

- The `CREATE TABLE` command will create a table (i.e., a relation) within a database.
- This creates the table *table_name* with the attributes specified in *attribute_list*.

Creating Attributes

- To specify an attribute, we need to provide the name and the data type.
- These may be followed by a number of modifiers that we will introduce shortly.
- For example, the employees table will have seven attributes:
  - `fname` – first name
  - `lname` – last name
  - `ssn` – social security number
  - `bdate` – birthday
  - `sex` – sex
  - `salary` – salary
  - `dept` – department

Creating Attributes

Creating an Attribute

```
fname CHAR(20)
```

- The `fname` attribute would be described as `CHAR(20)`.
- This states that the value of `fname` is a character string of length 20.
- We could use `VARCHAR(20)`, which will store *up to* 20 characters.

Creating Tables

Creating a Table

```
CREATE TABLE employees
(
    fname VARCHAR(20),
    lname VARCHAR(20),
    ssn CHAR(9),
    bdate DATE,
    sex CHAR(1),
    salary DECIMAL(10, 2),
    dept TINYINT(4),
    PRIMARY KEY (ssn)
);
```

- The other tables may be defined similarly.

Creating Tables

SHOW TABLES

```
SHOW TABLES;
+--------------------+
| Tables_in_company  |
+--------------------+
| employees          |
+--------------------+
```

- We can now use the `SHOW TABLES` command to see what tables exist.

Creating Tables

DESCRIBE

```
DESCRIBE employees;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| fname   | varchar(20)  | NO   |     | NULL    |       |
| lname   | varchar(20)  | NO   |     | NULL    |       |
| ssn     | char(9)      | NO   | PRI | NULL    |       |
| bdate   | date         | YES  |     | NULL    |       |
| sex     | char(1)      | YES  |     | NULL    |       |
| salary  | decimal(10,2)| YES  |     | NULL    |       |
| mgr_ssn | char(9)      | YES  | MUL | NULL    |       |
| dept    | tinyint(4)   | NO   | MUL | NULL    |       |
+---------+--------------+------+-----+---------+-------+
```

- We can use the `DESCRIBE` command to see a detailed description of a table.

Outline

MySQL Data Types

- Categories of data types
  - Integer
  - Fixed point
  - Floating point
  - Boolean
  - String
  - Time/Date
  - Generic

Integer Types

- Standard integer types
  - `SMALLINT` - 2-byte integers.
  - `INT` - 4-byte integers.
- Extended integer types
  - `TINYINT` - 1-byte integers.
  - `MEDIUMINT` - 3-byte integers.
  - `BIGINT` - 8-byte integers.
- Each integer type may be either signed (default) or unsigned.

Fixed-Point Types

- Standard fixed-point types
  - `DECIMAL` - Whole numbers up to 10 digits.
  - `DECIMAL(`n`)` - Whole numbers up to n digits.
  - `DECIMAL(`n`, `d`)` - Real numbers with n digits, d of which are after the decimal point.
- Fixed-point values are stored exactly.

Floating-Point Types

- Standard floating-point types
  - `FLOAT` - 4-byte floating-point numbers.
  - `DOUBLE` - 8-byte floating-point numbers.
- Floating-point values are stored approximately.

Boolean Types

- Standard boolean type
  - `BOOL` is equivalent to `TINYINT(1)`, i.e., a one-digit integer.
- The values `0` and `1` are interpreted as “false” and “true,” respectively.
- The symbols `FALSE` and `TRUE` are equivalent to 0 and 1, respectively.

String Types

- Standard string types
  - `CHAR(`n`)` - Exactly n characters ($0 < n \leq 255$).
  - `VARCHAR(`n`)` - Up to n characters ($0 < n \leq 255$).
  - `BINARY(`n`)` - Exactly n bytes.
  - `VARBINARY(`n`)` - Up to n bytes.
  - `TEXT` - Arbitrary amount of character data.
  - `BLOB` - Arbitrary amount of byte data.
  - `ENUM` - Any one string from a specified list.
  - `SET` - Any number of strings from a specified list.

Outline

Assignment

- Read Section 4.1, pages 87 - 94.
- Visit the websites
  - `http://dev.mysql.com/doc/refman/5.6/en/create-database.html`
  - `http://dev.mysql.com/doc/refman/5.6/en/use.html`
  - `http://dev.mysql.com/doc/refman/5.6/en/create-table.html`
  - `http://dev.mysql.com/doc/refman/5.6/en/data-types.html`