# 阶段4 - MySql数据库

### 上节课回顾

### 二、作业点评

### 三、本次课知识点，重点难点

【知识点目标】

• 数据库是什么

• 数据库中的数据是怎么进行保存的

• 创建使用Mysql数据库、表

• Mysql的基本操作

• Mysql中常见的数据类型

• 常用sql语句

• 查询语句：as，条件查询，模糊查询，连接查询，排序(升序降序)，分组，去重，范围

• 部分函数使用

【重点】

• 创建使用Mysql数据库、表

• 常用sql语句

【难点】

• 常用Sql语句 - 多表连接查询

### 四、本次课程任务知识点讲解

## 什么是数据库？

数据库（Database）是按照数据结构来组织、存储和管理数据的仓库，

每个数据库都有一个或多个不同的API用于创建，访问，管理，搜索和复制所保存的数据。

我们也可以将数据存储在文件中，但是在文件中读写数据速度相对较慢。

所以，现在我们使用关系型数据库管理系统（RDBMS）来存储和管理的大数据量。所谓的关系型数据库，是建立在关系模型基础上的数据库。

RDBMS即关系数据库管理系统(Relational Database Management System)的特点：

* 1.数据以表格的形式出现
* 2.每行为各种记录名称
* 3.每列为记录名称所对应的数据域
* 4.许多的行和列组成一张表单
* 5.若干的表单组成database

## RDBMS 术语

在我们开始学习MySQL 数据库前，让我们先了解下RDBMS的一些术语：

* **数据库:** 数据库是一些关联表的集合。
* **数据表:** 表是数据的矩阵。在一个数据库中的表看起来像一个简单的电子表格。
* **列:** 一列(数据元素) 包含了相同的数据, 例如邮政编码的数据。
* **行：**一行（=元组，或记录）是一组相关的数据，例如一条用户订阅的数据。
* **冗余**：存储两倍数据，冗余降低了性能，但提高了数据的安全性。
* **主键**：主键是唯一的。一个数据表中只能包含一个主键。你可以使用主键来查询数据。
* **外键：**外键用于关联两个表。
* **复合键**：复合键（组合键）将多个列作为一个索引键，一般用于复合索引。
* **索引：**使用索引可快速访问数据库表中的特定信息。索引是对数据库表中一列或多列的值进行排序的一种结构。类似于书籍的目录。
* **参照完整性:** 参照的完整性要求关系中不允许引用不存在的实体。与实体完整性是关系模型必须满足的完整性约束条件，目的是保证数据的一致性。

## Mysql数据库

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB公司开发，目前属于Oracle公司。MySQL是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

* Mysql是开源的，所以你不需要支付额外的费用。
* Mysql支持大型的数据库。可以处理拥有上千万条记录的大型数据库。
* MySQL使用标准的SQL数据语言形式。
* Mysql可以允许于多个系统上，并且支持多种语言。这些编程语言包括C、C++、Python、Java、Perl、PHP、Eiffel、Ruby和Tcl等。
* Mysql对PHP有很好的支持，PHP是目前最流行的Web开发语言。
* MySQL支持大型数据库，支持5000万条记录的数据仓库，32位系统表文件最大可支持4GB，64位系统支持最大的表文件为8TB。
* Mysql是可以定制的，你可以修改源码来开发自己的Mysql系统。

## Window 上安装Mysql

Window上安装Mysql相对来说会较为简单，你只需要在 [MySQL 下载](http://www.mysql.com/downloads" \t "_blank)中下载window版本的mysql安装包，并解压安装包。

双击 setup.exe 文件，接下来你只需要安装默认的配置点击"next"即可，默认情况下安装信息会在C:\mysql目录中。

接下来你可以通过"开始" =》在搜索框中输入 " cmd" 命令 =》 在命令提示符上切换到 C:\mysql\bin 目录，并输入一下命令：

mysqld.exe --console

如果安装成功以上命令将输出一些mysql启动及InnoDB信息。

# MySQL 连接

## 使用mysql二进制方式连接

您可以使用MySQL二进制方式进入到mysql命令提示符下来连接MySQL数据库。

### 实例

以下是从命令行中连接mysql服务器的简单实例：

[root@host]# mysql -u root -p

Enter password:\*\*\*\*\*\*

在登录成功后会出现 mysql> 命令提示窗口，你可以在上面执行任何 SQL 语句。

以上命令执行后，登录成功输出结果如下:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 2854760 to server version: 5.0.9

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

在以上实例中，我们使用了root用户登录到mysql服务器，当然你也可以使用其他mysql用户登录。

如果用户权限足够，任何用户都可以在mysql的命令提示窗口中进行SQL操作。

退出 mysql> 命令提示窗口可以使用 exit 命令，如下所示：

mysql> exit

Bye

## 创建数据库

使用普通用户，你可能需要特定的权限来创建或者删除 MySQL 数据库。

所以我们这边使用root用户登录，root用户拥有最高权限。

### 实例

以下命令简单的演示了创建数据库的过程，数据名为 RUNOOB:

[root@host]# mysqladmin -u root -p

Enter password:\*\*\*\*\*\*

[root@host]# create database runoob

以上命令执行成功后会创建 MySQL 数据库 RUNOOB。

# MySQL 删除数据库

使用普通用户登陆 MySQL 服务器，你可能需要特定的权限来创建或者删除 MySQL 数据库，所以我们这边使用 root 用户登录，root 用户拥有最高权限。

在删除数据库过程中，务必要十分谨慎，因为在执行删除命令后，所有数据将会消失。

## drop 命令删除数据库

drop 命令格式：

drop database <数据库名>;

例如删除名为 RUNOOB 的数据库：

mysql> drop database RUNOOB;

# MySQL 选择数据库

在你连接到 MySQL 数据库后，可能有多个可以操作的数据库，所以你需要选择你要操作的数据库。

## 从命令提示窗口中选择MySQL数据库

在 mysql> 提示窗口中可以很简单的选择特定的数据库。你可以使用SQL命令来选择指定的数据库。

### 实例

以下实例选取了数据库 RUNOOB:

[root@host]# mysql -u root -p

Enter password:\*\*\*\*\*\*

mysql> use RUNOOB;

Database changed

mysql>

执行以上命令后，你就已经成功选择了 RUNOOB 数据库，在后续的操作中都会在 RUNOOB 数据库中执行。

**注意:**所有的数据库名，表名，表字段都是区分大小写的。所以你在使用SQL命令时需要输入正确的名称。

# MySQL 数据类型

MySQL中定义数据字段的类型对你数据库的优化是非常重要的。

MySQL支持多种类型，大致可以分为三类：数值、日期/时间和字符串(字符)类型。

## 数值类型

MySQL支持所有标准SQL数值数据类型。

这些类型包括严格数值数据类型(INTEGER、SMALLINT、DECIMAL和NUMERIC)，以及近似数值数据类型(FLOAT、REAL和DOUBLE PRECISION)。

关键字INT是INTEGER的同义词，关键字DEC是DECIMAL的同义词。

BIT数据类型保存位字段值，并且支持MyISAM、MEMORY、InnoDB和BDB表。

作为SQL标准的扩展，MySQL也支持整数类型TINYINT、MEDIUMINT和BIGINT。下面的表显示了需要的每个整数类型的存储和范围。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **大小** | **范围（有符号）** | **范围（无符号）** | **用途** |
| TINYINT | 1 字节 | (-128，127) | (0，255) | 小整数值 |
| SMALLINT | 2 字节 | (-32 768，32 767) | (0，65 535) | 大整数值 |
| MEDIUMINT | 3 字节 | (-8 388 608，8 388 607) | (0，16 777 215) | 大整数值 |
| INT或INTEGER | 4 字节 | (-2 147 483 648，2 147 483 647) | (0，4 294 967 295) | 大整数值 |
| BIGINT | 8 字节 | (-9 233 372 036 854 775 808，9 223 372 036 854 775 807) | (0，18 446 744 073 709 551 615) | 极大整数值 |
| FLOAT | 4 字节 | (-3.402 823 466 E+38，-1.175 494 351 E-38)，0，(1.175 494 351 E-38，3.402 823 466 351 E+38) | 0，(1.175 494 351 E-38，3.402 823 466 E+38) | 单精度 浮点数值 |
| DOUBLE | 8 字节 | (-1.797 693 134 862 315 7 E+308，-2.225 073 858 507 201 4 E-308)，0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308) | 0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308) | 双精度 浮点数值 |
| DECIMAL | 对DECIMAL(M,D) ，如果M>D，为M+2否则为D+2 | 依赖于M和D的值 | 依赖于M和D的值 | 小数值 |

**[MySQL数据类型DECIMAL用法](http://www.cnblogs.com/owenma/p/7097602.html)**

MySQL DECIMAL数据类型用于在数据库中存储精确的数值。我们经常将DECIMAL数据类型用于保留准确精确度的列，例如会计系统中的货币数据。

要定义数据类型为DECIMAL的列，请使用以下语法：

|  |  |
| --- | --- |
| 1 | column\_name  DECIMAL(P,D); |

在上面的语法中：

* P是表示有效数字数的精度。 P范围为1〜65。
* D是表示小数点后的位数。 D的范围是0~30。MySQL要求D小于或等于(<=)P。

DECIMAL(P，D)表示列可以存储D位小数的P位数。十进制列的实际范围取决于精度和刻度。

与INT数据类型一样，DECIMAL类型也具有UNSIGNED和ZEROFILL属性。 如果使用UNSIGNED属性，则DECIMAL UNSIGNED的列将不接受负值。

如果使用ZEROFILL，MySQL将把显示值填充到0以显示由列定义指定的宽度。 另外，如果我们对DECIMAL列使用ZERO FILL，MySQL将自动将UNSIGNED属性添加到列。

以下示例使用DECIMAL数据类型定义的一个叫作amount的列。

|  |
| --- |
| amount DECIMAL(6,2); |

在此示例中，amount列最多可以存储6位数字，小数位数为2位; 因此，amount列的范围是从-9999.99到9999.99。

MySQL允许使用以下语法：

|  |
| --- |
| column\_name DECIMAL(P); |

这相当于：

|  |
| --- |
| column\_name DECIMAL(P,0); |

在这种情况下，列不包含小数部分或小数点。

此外，我们甚至可以使用以下语法。

|  |
| --- |
| column\_name DECIMAL; |

在这种情况下，P的默认值为10。

## 日期和时间类型

表示时间值的日期和时间类型为DATETIME、DATE、TIMESTAMP、TIME和YEAR。

每个时间类型有一个有效值范围和一个"零"值，当指定不合法的MySQL不能表示的值时使用"零"值。

TIMESTAMP类型有专有的自动更新特性，将在后面描述。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **大小 (字节)** | **范围** | **格式** | **用途** |
| DATE | 3 | 1000-01-01/9999-12-31 | YYYY-MM-DD | 日期值 |
| TIME | 3 | '-838:59:59'/'838:59:59' | HH:MM:SS | 时间值或持续时间 |
| YEAR | 1 | 1901/2155 | YYYY | 年份值 |
| DATETIME | 8 | 1000-01-01 00:00:00/9999-12-31 23:59:59 | YYYY-MM-DD HH:MM:SS | 混合日期和时间值 |
| TIMESTAMP | 4 | 1970-01-01 00:00:00/2038  结束时间是第 **2147483647** 秒，北京时间 **2038-1-19 11:14:07**，格林尼治时间 2038年1月19日 凌晨 03:14:07 | YYYYMMDD HHMMSS | 混合日期和时间值，时间戳 |

## 字符串类型

字符串类型指CHAR、VARCHAR、BINARY、VARBINARY、BLOB、TEXT、ENUM和SET。该节描述了这些类型如何工作以及如何在查询中使用这些类型。

|  |  |  |
| --- | --- | --- |
| **类型** | **大小** | **用途** |
| CHAR | 0-255字节 | 定长字符串 |
| VARCHAR | 0-65535 字节 | 变长字符串 |
| TINYBLOB | 0-255字节 | 不超过 255 个字符的二进制字符串 |
| TINYTEXT | 0-255字节 | 短文本字符串 |
| BLOB | 0-65 535字节 | 二进制形式的长文本数据 |
| TEXT | 0-65 535字节 | 长文本数据 |
| MEDIUMBLOB | 0-16 777 215字节 | 二进制形式的中等长度文本数据 |
| MEDIUMTEXT | 0-16 777 215字节 | 中等长度文本数据 |
| LONGBLOB | 0-4 294 967 295字节 | 二进制形式的极大文本数据 |
| LONGTEXT | 0-4 294 967 295字节 | 极大文本数据 |

CHAR 和 VARCHAR 类型类似，但它们保存和检索的方式不同。它们的最大长度和是否尾部空格被保留等方面也不同。在存储或检索过程中不进行大小写转换。

BINARY 和 VARBINARY 类似于 CHAR 和 VARCHAR，不同的是它们包含二进制字符串而不要非二进制字符串。也就是说，它们包含字节字符串而不是字符字符串。这说明它们没有字符集，并且排序和比较基于列值字节的数值值。

BLOB 是一个二进制大对象，可以容纳可变数量的数据。有 4 种 BLOB 类型：TINYBLOB、BLOB、MEDIUMBLOB 和 LONGBLOB。它们区别在于可容纳存储范围不同。

有 4 种 TEXT 类型：TINYTEXT、TEXT、MEDIUMTEXT 和 LONGTEXT。对应的这 4 种 BLOB 类型，可存储的最大长度不同，可根据实际情况选择。

# MySQL 创建数据表

创建MySQL数据表需要以下信息：

* 表名
* 表字段名
* 定义每个表字段

### 语法

以下为创建MySQL数据表的SQL通用语法：

CREATE TABLE table\_name (column\_name column\_type);

以下例子中我们将在 RUNOOB 数据库中创建数据表runoob\_tbl：

CREATE TABLE IF NOT EXISTS `runoob\_tbl`(

`runoob\_id` INT UNSIGNED AUTO\_INCREMENT,

`runoob\_title` VARCHAR(100) NOT NULL,

`runoob\_author` VARCHAR(40) NOT NULL,

`submission\_date` DATE,

PRIMARY KEY ( `runoob\_id` )

)ENGINE=InnoDB DEFAULT CHARSET=utf8;

实例解析：

* 如果你不想字段为 **NULL** 可以设置字段的属性为 **NOT NULL**， 在操作数据库时如果输入该字段的数据为**NULL** ，就会报错。
* AUTO\_INCREMENT定义列为自增的属性，一般用于主键，数值会自动加1。
* PRIMARY KEY关键字用于定义列为主键。 您可以使用多列来定义主键，列间以逗号分隔。
* ENGINE 设置存储引擎，CHARSET 设置编码。

## 通过命令提示符创建表

通过 mysql> 命令窗口可以很简单的创建MySQL数据表。你可以使用 SQL 语句 **CREATE TABLE** 来创建数据表。

### 实例

以下为创建数据表 runoob\_tbl 实例:

root@host# mysql -u root -p

Enter password:\*\*\*\*\*\*\*

mysql> use RUNOOB;

Database changed

mysql> CREATE TABLE runoob\_tbl(

-> runoob\_id INT NOT NULL AUTO\_INCREMENT,

-> runoob\_title VARCHAR(100) NOT NULL,

-> runoob\_author VARCHAR(40) NOT NULL,

-> submission\_date DATE,

-> PRIMARY KEY ( runoob\_id )

-> )ENGINE=InnoDB DEFAULT CHARSET=utf8;

Query OK, 0 rows affected (0.16 sec)

mysql>

**注意：**MySQL命令终止符为分号 (;) 。

# MySQL 删除数据表

MySQL中删除数据表是非常容易操作的， 但是你再进行删除表操作时要非常小心，因为执行删除命令后所有数据都会消失。

### 语法

以下为删除MySQL数据表的通用语法：

DROP TABLE table\_name ;

## 在命令提示窗口中删除数据表

在mysql>命令提示窗口中删除数据表SQL语句为**DROP TABLE**：

### 实例

以下实例删除了数据表runoob\_tbl:

root@host# mysql -u root -p

Enter password:\*\*\*\*\*\*\*

mysql> use RUNOOB;

Database changed

mysql> DROP TABLE runoob\_tbl

Query OK, 0 rows affected (0.8 sec)

mysql>

# MySQL 插入数据

MySQL 表中使用**INSERT INTO**SQL语句来插入数据。

你可以通过 mysql> 命令提示窗口中向数据表中插入数据，或者通过PHP脚本来插入数据。

### 语法

以下为向MySQL数据表插入数据通用的**INSERT INTO**SQL语法：

INSERT INTO table\_name ( field1, field2,...fieldN )

VALUES

( value1, value2,...valueN );

如果数据是字符型，必须使用单引号或者双引号，如："value"。

## 通过命令提示窗口插入数据

以下我们将使用 SQL **INSERT INTO**语句向 MySQL 数据表 runoob\_tbl 插入数据

### 实例

以下实例中我们将向 runoob\_tbl 表插入三条数据:

root@host# mysql -u root -p password;

Enter password:\*\*\*\*\*\*\*

mysql> use RUNOOB;

Database changed

mysql> INSERT INTO runoob\_tbl

-> (runoob\_title, runoob\_author, submission\_date)

-> VALUES

-> ("学习 PHP", "菜鸟教程", NOW());

Query OK, 1 rows affected, 1 warnings (0.01 sec)

mysql> INSERT INTO runoob\_tbl

-> (runoob\_title, runoob\_author, submission\_date)

-> VALUES

-> ("学习 MySQL", "菜鸟教程", NOW());

Query OK, 1 rows affected, 1 warnings (0.01 sec)

mysql> INSERT INTO runoob\_tbl

-> (runoob\_title, runoob\_author, submission\_date)

-> VALUES

-> ("JAVA 教程", "RUNOOB.COM", '2016-05-06');

Query OK, 1 rows affected (0.00 sec)

mysql>

**注意：** 使用箭头标记 **->** 不是 SQL 语句的一部分，它仅仅表示一个新行，如果一条SQL语句太长，我们可以通过回车键来创建一个新行来编写 SQL 语句，SQL 语句的命令结束符为分号 **;**。

在以上实例中，我们并没有提供 runoob\_id 的数据，因为该字段我们在创建表的时候已经设置它为 AUTO\_INCREMENT(自动增加) 属性。 所以，该字段会自动递增而不需要我们去设置。实例中 NOW() 是一个 MySQL 函数，该函数返回日期和时间。

接下来我们可以通过以下语句查看数据表数据：

## 读取数据表：

select \* from runoob\_tbl;

输出结果：



# MySQL 查询数据

MySQL 数据库使用SQL SELECT语句来查询数据。

你可以通过 mysql> 命令提示窗口中在数据库中查询数据，或者通过PHP脚本来查询数据。

### 语法

以下为在MySQL数据库中查询数据通用的 SELECT 语法：

SELECT column\_name,column\_name

FROM table\_name

[WHERE Clause]

[LIMIT N][ OFFSET M]

* 查询语句中你可以使用一个或者多个表，表之间使用逗号(,)分割，并使用WHERE语句来设定查询条件。
* SELECT 命令可以读取一条或者多条记录。
* 你可以使用星号（\*）来代替其他字段，SELECT语句会返回表的所有字段数据
* 你可以使用 WHERE 语句来包含任何条件。
* 你可以使用 LIMIT 属性来设定返回的记录数。
* 你可以通过OFFSET指定SELECT语句开始查询的数据偏移量。默认情况下偏移量为0。

## 通过命令提示符获取数据

以下实例我们将通过 SQL SELECT 命令来获取 MySQL 数据表 runoob\_tbl 的数据：

### 实例

以下实例将返回数据表 runoob\_tbl 的所有记录:

## 读取数据表：

select \* from runoob\_tbl;

输出结果：



# MySQL WHERE 子句

我们知道从 MySQL 表中使用 SQL SELECT 语句来读取数据。

如需有条件地从表中选取数据，可将 WHERE 子句添加到 SELECT 语句中。

### 语法

以下是 SQL SELECT 语句使用 WHERE 子句从数据表中读取数据的通用语法：

SELECT field1, field2,...fieldN FROM table\_name1, table\_name2...

[WHERE condition1 [AND [OR]] condition2.....

* 查询语句中你可以使用一个或者多个表，表之间使用逗号**,** 分割，并使用WHERE语句来设定查询条件。
* 你可以在 WHERE 子句中指定任何条件。
* 你可以使用 AND 或者 OR 指定一个或多个条件。
* WHERE 子句也可以运用于 SQL 的 DELETE 或者 UPDATE 命令。
* WHERE 子句类似于程序语言中的 if 条件，根据 MySQL 表中的字段值来读取指定的数据。

以下为操作符列表，可用于 WHERE 子句中。

下表中实例假定 A 为 10, B 为 20

|  |  |  |
| --- | --- | --- |
| **操作符** | **描述** | **实例** |
| = | 等号，检测两个值是否相等，如果相等返回true | (A = B) 返回false。 |
| <>, != | 不等于，检测两个值是否相等，如果不相等返回true | (A != B) 返回 true。 |
| > | 大于号，检测左边的值是否大于右边的值, 如果左边的值大于右边的值返回true | (A > B) 返回false。 |
| < | 小于号，检测左边的值是否小于右边的值, 如果左边的值小于右边的值返回true | (A < B) 返回 true。 |
| >= | 大于等于号，检测左边的值是否大于或等于右边的值, 如果左边的值大于或等于右边的值返回true | (A >= B) 返回false。 |
| <= | 小于等于号，检测左边的值是否小于于或等于右边的值, 如果左边的值小于或等于右边的值返回true | (A <= B) 返回 true。 |

如果我们想再 MySQL 数据表中读取指定的数据，WHERE 子句是非常有用的。

使用主键来作为 WHERE 子句的条件查询是非常快速的。

如果给定的条件在表中没有任何匹配的记录，那么查询不会返回任何数据。

## 从命令提示符中读取数据

我们将在SQL SELECT语句使用WHERE子句来读取MySQL数据表 runoob\_tbl 中的数据：

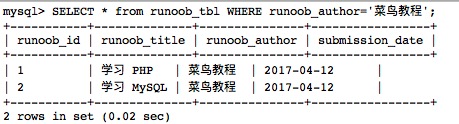
实例

以下实例将读取 runoob\_tbl 表中 runoob\_author 字段值为 Sanjay 的所有记录：

## SQL SELECT WHERE 子句

SELECT \* from runoob\_tbl WHERE runoob\_author='菜鸟教程';

输出结果：



MySQL 的 WHERE 子句的字符串比较是不区分大小写的。 你可以使用 BINARY 关键字来设定 WHERE 子句的字符串比较是区分大小写的。

如下实例:

## BINARY 关键字

mysql> SELECT \* from runoob\_tbl WHERE BINARY runoob\_author='runoob.com';

Empty set (0.01 sec)

mysql> SELECT \* from runoob\_tbl WHERE BINARY runoob\_author='RUNOOB.COM';

+-----------+---------------+---------------+-----------------+

| runoob\_id | runoob\_title | runoob\_author | submission\_date |

+-----------+---------------+---------------+-----------------+

| 3 | JAVA 教程 | RUNOOB.COM | 2016-05-06 |

| 4 | 学习 Python | RUNOOB.COM | 2016-03-06 |

+-----------+---------------+---------------+-----------------+

2 rows in set (0.01 sec)

实例中使用了 **BINARY** 关键字，是区分大小写的，所以 **runoob\_author='runoob.com'** 的查询条件是没有数据的。

# MySQL UPDATE 查询

如果我们需要修改或更新 MySQL 中的数据，我们可以使用 SQL UPDATE 命令来操作。.

### 语法

以下是 UPDATE 命令修改 MySQL 数据表数据的通用 SQL 语法：

UPDATE table\_name SET field1=new-value1, field2=new-value2

[WHERE Clause]

* 你可以同时更新一个或多个字段。
* 你可以在 WHERE 子句中指定任何条件。
* 你可以在一个单独表中同时更新数据。

当你需要更新数据表中指定行的数据时 WHERE 子句是非常有用的。

通过命令提示符更新数据

以下我们将在 SQL UPDATE 命令使用 WHERE 子句来更新 runoob\_tbl 表中指定的数据：

### 实例

以下实例将更新数据表中 runoob\_id 为 3 的 runoob\_title 字段值：

## SQL UPDATE 语句：

mysql> UPDATE runoob\_tbl SET runoob\_title='学习 C++' WHERE runoob\_id=3;

Query OK, 1 rows affected (0.01 sec)

mysql> SELECT \* from runoob\_tbl WHERE runoob\_id=3;

+-----------+--------------+---------------+-----------------+

| runoob\_id | runoob\_title | runoob\_author | submission\_date |

+-----------+--------------+---------------+-----------------+

| 3 | 学习 C++ | RUNOOB.COM | 2016-05-06 |

+-----------+--------------+---------------+-----------------+

1 rows in set (0.01 sec)

从结果上看，runoob\_id 为 3 的 runoob\_title 已被修改。

# MySQL DELETE 语句

你可以使用 SQL 的 DELETE FROM 命令来删除 MySQL 数据表中的记录。

你可以在 **mysql>** 命令提示符或 PHP 脚本中执行该命令。

### 语法

以下是 SQL DELETE 语句从 MySQL 数据表中删除数据的通用语法：

DELETE FROM table\_name [WHERE Clause]

* 如果没有指定 WHERE 子句，MySQL 表中的所有记录将被删除。
* 你可以在 WHERE 子句中指定任何条件
* 您可以在单个表中一次性删除记录。

当你想删除数据表中指定的记录时 WHERE 子句是非常有用的。

## 从命令行中删除数据

这里我们将在 SQL DELETE 命令中使用 WHERE 子句来删除 MySQL 数据表 runoob\_tbl 所选的数据。

### 实例

以下实例将删除 runoob\_tbl 表中 runoob\_id 为3 的记录：

## DELETE 语句：

mysql> use RUNOOB;

Database changed

mysql> DELETE FROM runoob\_tbl WHERE runoob\_id=3;

Query OK, 1 row affected (0.23 sec)

# MySQL LIKE 子句

我们知道在 MySQL 中使用 SQL SELECT 命令来读取数据， 同时我们可以在 SELECT 语句中使用 WHERE 子句来获取指定的记录。

WHERE 子句中可以使用等号 **=** 来设定获取数据的条件，如 "runoob\_author = 'RUNOOB.COM'"。

但是有时候我们需要获取 runoob\_author 字段含有 "COM" 字符的所有记录，这时我们就需要在 WHERE 子句中使用 SQL LIKE 子句。

SQL LIKE 子句中使用百分号 **%**字符来表示任意字符，类似于UNIX或正则表达式中的星号 **\***。

如果没有使用百分号 **%**, LIKE 子句与等号 **=** 的效果是一样的。

### 语法

以下是 SQL SELECT 语句使用 LIKE 子句从数据表中读取数据的通用语法：

SELECT field1, field2,...fieldN

FROM table\_name

WHERE field1 LIKE condition1 [AND [OR]] filed2 = 'somevalue'

* 你可以在 WHERE 子句中指定任何条件。
* 你可以在 WHERE 子句中使用LIKE子句。
* 你可以使用LIKE子句代替等号 **=**。
* LIKE 通常与 **%** 一同使用，类似于一个元字符的搜索。
* 你可以使用 AND 或者 OR 指定一个或多个条件。
* 你可以在 DELETE 或 UPDATE 命令中使用 WHERE...LIKE 子句来指定条件。

## 在命令提示符中使用 LIKE 子句

以下我们将在 SQL SELECT 命令中使用 WHERE...LIKE 子句来从MySQL数据表 runoob\_tbl 中读取数据。

### 实例

以下是我们将 runoob\_tbl 表中获取 runoob\_author 字段中以 **COM** 为结尾的的所有记录：

## SQL UPDATE 语句：

mysql> use RUNOOB;

Database changed

mysql> SELECT \* from runoob\_tbl WHERE runoob\_author LIKE '%COM';

+-----------+---------------+---------------+-----------------+

| runoob\_id | runoob\_title | runoob\_author | submission\_date |

+-----------+---------------+---------------+-----------------+

| 3 | 学习 Java | RUNOOB.COM | 2015-05-01 |

| 4 | 学习 Python | RUNOOB.COM | 2016-03-06 |

+-----------+---------------+---------------+-----------------+

2 rows in set (0.01 sec)

# MySQL 排序

我们知道从 MySQL 表中使用 SQL SELECT 语句来读取数据。

如果我们需要对读取的数据进行排序，我们就可以使用 MySQL 的 **ORDER BY** 子句来设定你想按哪个字段哪种方式来进行排序，再返回搜索结果。

### 语法

以下是 SQL SELECT 语句使用 ORDER BY 子句将查询数据排序后再返回数据：

SELECT field1, field2,...fieldN table\_name1, table\_name2...

ORDER BY field1, [field2...] [ASC [DESC]]

* 你可以使用任何字段来作为排序的条件，从而返回排序后的查询结果。
* 你可以设定多个字段来排序。
* 你可以使用 ASC 或 DESC 关键字来设置查询结果是按升序或降序排列。 默认情况下，它是按升序排列。
* 你可以添加 WHERE...LIKE 子句来设置条件。

## 在命令提示符中使用 ORDER BY 子句

以下将在 SQL SELECT 语句中使用 ORDER BY 子句来读取MySQL 数据表 runoob\_tbl 中的数据：

### 实例

尝试以下实例，结果将按升序及降序排列。

## SQL 排序

mysql> use RUNOOB;

Database changed

mysql> SELECT \* from runoob\_tbl ORDER BY submission\_date ASC;

+-----------+---------------+---------------+-----------------+

| runoob\_id | runoob\_title | runoob\_author | submission\_date |

+-----------+---------------+---------------+-----------------+

| 3 | 学习 Java | RUNOOB.COM | 2015-05-01 |

| 4 | 学习 Python | RUNOOB.COM | 2016-03-06 |

| 1 | 学习 PHP | 菜鸟教程 | 2017-04-12 |

| 2 | 学习 MySQL | 菜鸟教程 | 2017-04-12 |

+-----------+---------------+---------------+-----------------+

4 rows in set (0.01 sec)

mysql> SELECT \* from runoob\_tbl ORDER BY submission\_date DESC;

+-----------+---------------+---------------+-----------------+

| runoob\_id | runoob\_title | runoob\_author | submission\_date |

+-----------+---------------+---------------+-----------------+

| 1 | 学习 PHP | 菜鸟教程 | 2017-04-12 |

| 2 | 学习 MySQL | 菜鸟教程 | 2017-04-12 |

| 4 | 学习 Python | RUNOOB.COM | 2016-03-06 |

| 3 | 学习 Java | RUNOOB.COM | 2015-05-01 |

+-----------+---------------+---------------+-----------------+

4 rows in set (0.01 sec)

读取 runoob\_tbl 表中所有数据并按 submission\_date 字段的升序排列。

# MySQL GROUP BY 语句

GROUP BY 语句根据一个或多个列对结果集进行分组。

在分组的列上我们可以使用 COUNT, SUM, AVG,等函数。

### GROUP BY 语法

SELECT column\_name, function(column\_name)

FROM table\_name

WHERE column\_name operator value

GROUP BY column\_name;

## 实例演示

本章节实例使用到了以下表结构及数据，使用前我们可以先将以下数据导入数据库中。

-- ----------------------------

-- Table structure for `employee\_tbl`

-- ----------------------------

DROP TABLE IF EXISTS `employee\_tbl`;

CREATE TABLE `employee\_tbl` (

`id` int(11) NOT NULL,

`name` char(10) NOT NULL DEFAULT '',

`date` datetime NOT NULL,

`singin` tinyint(4) NOT NULL DEFAULT '0' COMMENT '登录次数',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------

-- Records of `employee\_tbl`

-- ----------------------------

BEGIN;

INSERT INTO `employee\_tbl` VALUES ('1', '小明', '2016-04-22 15:25:33', '1'), ('2', '小王', '2016-04-20 15:25:47', '3'), ('3', '小丽', '2016-04-19 15:26:02', '2'), ('4', '小王', '2016-04-07 15:26:14', '4'), ('5', '小明', '2016-04-11 15:26:40', '4'), ('6', '小明', '2016-04-04 15:26:54', '2');

COMMIT;

SET FOREIGN\_KEY\_CHECKS = 1;

导入成功后，执行以下 SQL 语句：

mysql> set names utf8;

mysql> SELECT \* FROM employee\_tbl;

+----+--------+---------------------+--------+

| id | name | date | singin |

+----+--------+---------------------+--------+

| 1 | 小明 | 2016-04-22 15:25:33 | 1 |

| 2 | 小王 | 2016-04-20 15:25:47 | 3 |

| 3 | 小丽 | 2016-04-19 15:26:02 | 2 |

| 4 | 小王 | 2016-04-07 15:26:14 | 4 |

| 5 | 小明 | 2016-04-11 15:26:40 | 4 |

| 6 | 小明 | 2016-04-04 15:26:54 | 2 |

+----+--------+---------------------+--------+

6 rows in set (0.00 sec)

接下来我们使用 GROUP BY 语句 将数据表按名字进行分组，并统计每个人有多少条记录：

mysql> SELECT name, COUNT(\*) FROM employee\_tbl GROUP BY name;

+--------+----------+

| name | COUNT(\*) |

+--------+----------+

| 小丽 | 1 |

| 小明 | 3 |

| 小王 | 2 |

+--------+----------+

3 rows in set (0.01 sec)

# Mysql 连接的使用

在前几章节中，我们已经学会了如何在一张表中读取数据，这是相对简单的，但是在真正的应用中经常需要从多个数据表中读取数据。

本章节我们将向大家介绍如何使用 MySQL 的 JOIN 在两个或多个表中查询数据。

你可以在 SELECT, UPDATE 和 DELETE 语句中使用 Mysql 的 JOIN 来联合多表查询。

JOIN 按照功能大致分为如下三类：

* **INNER JOIN（内连接,或等值连接）**：获取两个表中字段匹配关系的记录。
* **LEFT JOIN（左连接）：**获取左表所有记录，即使右表没有对应匹配的记录。
* **RIGHT JOIN（右连接）：** 与 LEFT JOIN 相反，用于获取右表所有记录，即使左表没有对应匹配的记录。

本章节使用的数据库结构及数据下载：[runoob-mysql-join-test.sql](http://static.runoob.com/download/runoob-mysql-join-test.sql)。

## 在命令提示符中使用 INNER JOIN

我们在RUNOOB数据库中有两张表 tcount\_tbl 和 runoob\_tbl。两张数据表数据如下：

### 实例

尝试以下实例：

## 测试实例数据

mysql> use RUNOOB;

Database changed

mysql> SELECT \* FROM tcount\_tbl;

+---------------+--------------+

| runoob\_author | runoob\_count |

+---------------+--------------+

| 菜鸟教程 | 10 |

| RUNOOB.COM | 20 |

| Google | 22 |

+---------------+--------------+

3 rows in set (0.01 sec)

mysql> SELECT \* from runoob\_tbl;

+-----------+---------------+---------------+-----------------+

| runoob\_id | runoob\_title | runoob\_author | submission\_date |

+-----------+---------------+---------------+-----------------+

| 1 | 学习 PHP | 菜鸟教程 | 2017-04-12 |

| 2 | 学习 MySQL | 菜鸟教程 | 2017-04-12 |

| 3 | 学习 Java | RUNOOB.COM | 2015-05-01 |

| 4 | 学习 Python | RUNOOB.COM | 2016-03-06 |

| 5 | 学习 C | FK | 2017-04-05 |

+-----------+---------------+---------------+-----------------+

5 rows in set (0.01 sec)

接下来我们就使用MySQL的**INNER JOIN(也可以省略 INNER 使用 JOIN，效果一样)**来连接以上两张表来读取runoob\_tbl表中所有runoob\_author字段在tcount\_tbl表对应的runoob\_count字段值：

## INNER JOIN

mysql> SELECT a.runoob\_id, a.runoob\_author, b.runoob\_count FROM runoob\_tbl a INNER JOIN tcount\_tbl b ON a.runoob\_author = b.runoob\_author;

+-------------+-----------------+----------------+

| a.runoob\_id | a.runoob\_author | b.runoob\_count |

+-------------+-----------------+----------------+

| 1 | 菜鸟教程 | 10 |

| 2 | 菜鸟教程 | 10 |

| 3 | RUNOOB.COM | 20 |

| 4 | RUNOOB.COM | 20 |

+-------------+-----------------+----------------+

4 rows in set (0.00 sec)

以上 SQL 语句等价于：

## WHERE 子句

mysql> SELECT a.runoob\_id, a.runoob\_author, b.runoob\_count FROM runoob\_tbl a, tcount\_tbl b WHERE a.runoob\_author = b.runoob\_author;

+-------------+-----------------+----------------+

| a.runoob\_id | a.runoob\_author | b.runoob\_count |

+-------------+-----------------+----------------+

| 1 | 菜鸟教程 | 10 |

| 2 | 菜鸟教程 | 10 |

| 3 | RUNOOB.COM | 20 |

| 4 | RUNOOB.COM | 20 |

+-------------+-----------------+----------------+

4 rows in set (0.01 sec)



## MySQL LEFT JOIN

MySQL left join 与 join 有所不同。 MySQL LEFT JOIN 会读取左边数据表的全部数据，即便右边表无对应数据。

### 实例

尝试以下实例，以 **runoob\_tbl** 为左表，**tcount\_tbl** 为右表，理解 MySQL LEFT JOIN 的应用：

## LEFT JOIN

mysql> SELECT a.runoob\_id, a.runoob\_author, b.runoob\_count FROM runoob\_tbl a LEFT JOIN tcount\_tbl b ON a.runoob\_author = b.runoob\_author;

+-------------+-----------------+----------------+

| a.runoob\_id | a.runoob\_author | b.runoob\_count |

+-------------+-----------------+----------------+

| 1 | 菜鸟教程 | 10 |

| 2 | 菜鸟教程 | 10 |

| 3 | RUNOOB.COM | 20 |

| 4 | RUNOOB.COM | 20 |

| 5 | FK | NULL |

+-------------+-----------------+----------------+

5 rows in set (0.01 sec)

以上实例中使用了 LEFT JOIN，该语句会读取左边的数据表 runoob\_tbl 的所有选取的字段数据，即便在右侧表 tcount\_tbl中 没有对应的 runoob\_author 字段值。



## MySQL RIGHT JOIN

MySQL RIGHT JOIN 会读取右边数据表的全部数据，即便左边边表无对应数据。

### 实例

尝试以下实例，以 **runoob\_tbl** 为左表，**tcount\_tbl** 为右表，理解MySQL RIGHT JOIN的应用：

## RIGHT JOIN

mysql> SELECT a.runoob\_id, a.runoob\_author, b.runoob\_count FROM runoob\_tbl a RIGHT JOIN tcount\_tbl b ON a.runoob\_author = b.runoob\_author;

+-------------+-----------------+----------------+

| a.runoob\_id | a.runoob\_author | b.runoob\_count |

+-------------+-----------------+----------------+

| 1 | 菜鸟教程 | 10 |

| 2 | 菜鸟教程 | 10 |

| 3 | RUNOOB.COM | 20 |

| 4 | RUNOOB.COM | 20 |

| NULL | NULL | 22 |

+-------------+-----------------+----------------+

5 rows in set (0.01 sec)

以上实例中使用了 RIGHT JOIN，该语句会读取右边的数据表 tcount\_tbl 的所有选取的字段数据，即便在左侧表 runoob\_tbl 中没有对应的runoob\_author 字段值。



## MySQL聚合函数

* [MySQL聚合函数](http://www.yiibai.com/mysql/aggregate-functions.html" \t "_blank" \o "MySQL聚合函数) - 提供最常用的MySQL聚合函数的简要概述。
* [avg()函数](http://www.yiibai.com/mysql/avg.html" \t "_blank" \o "avg()函数) - 计算一组值或表达式的平均值。
* [count()函数](http://www.yiibai.com/mysql/count.html" \t "_blank" \o "count()函数) - 计算表中的行数。
* [instr()函数](http://www.yiibai.com/mysql/instr.html" \t "_blank" \o "instr()函数) - 返回子字符串在字符串中第一次出现的位置。
* [sum()函数](http://www.yiibai.com/mysql/sum.html" \t "_blank" \o "sum()函数) - 计算一组值或表达式的总和。
* [min()函数](http://www.yiibai.com/mysql/min.html" \t "_blank" \o "min()函数) - 在一组值中找到最小值。
* [max()函数](http://www.yiibai.com/mysql/max-function.html" \t "_blank" \o "max()函数) - 在一组值中找到最大值。
* [group\_concat()函数](http://www.yiibai.com/mysql/group_concat.html" \t "_blank" \o "group_concat()函数) - 将字符串从分组中连接成具有各种选项(如DISTINCT，ORDER BY和SEPARATOR)的字符串。
* [MySQL标准偏差函数](http://www.yiibai.com/mysql/standard-deviation.html" \t "_blank" \o "MySQL标准偏差函数) - 显示如何计算人口标准偏差和样本标准偏差。

## MySQL字符串函数

* [concat()函数](http://www.yiibai.com/mysql/sql-concat-in-mysql.html" \t "_blank" \o "concat()函数) - 将两个或多个字符串组合成一个字符串。
* [length()函数&char\_length()函数](http://www.yiibai.com/mysql/string-length.html" \t "_blank" \o "length()函数&char_length()函数) - 以字节和字符获取字符串的长度。
* [left()函数](http://www.yiibai.com/mysql/left-function.html" \t "_blank" \o "left()函数) - 获取指定长度的字符串的左边部分。
* [replace()函数](http://www.yiibai.com/mysql/string-replace-function.html" \t "_blank" \o "replace()函数) - 搜索并替换字符串中的子字符串。
* [substring()函数](http://www.yiibai.com/mysql/substring.html" \t "_blank" \o "substring()函数) - 从具有特定长度的位置开始提取一个子字符串。
* [trim()函数](http://www.yiibai.com/mysql/trim.html" \t "_blank" \o "trim()函数) - 从字符串中删除不需要的字符。
* [find\_in\_set()函数](http://www.yiibai.com/mysql/find_in_set.html" \t "_blank" \o "find_in_set()函数) - 在逗号分隔的字符串列表中找到一个字符串。
* [format()函数](http://www.yiibai.com/mysql/format-function.html" \t "_blank" \o "format()函数) - 格式化具有特定区域设置的数字，舍入到小数位数。

## MySQL控制流函数

* [case()函数](http://www.yiibai.com/mysql/case-function.html" \t "_blank" \o "case()函数) - 如果满足WHEN分支中的条件，则返回THEN分支中的相应结果，否则返回ELSE分支中的结果。
* [if语句](http://www.yiibai.com/mysql/if-statement.html" \t "_blank" \o "if语句) - 根据给定的条件返回一个值。
* [ifnull()函数](http://www.yiibai.com/mysql/ifnull.html" \t "_blank" \o "ifnull()函数) - 如果第一个参数不为NULL，则返回第一个参数，否则返回第二个参数。
* [nullif()函数](http://www.yiibai.com/mysql/nullif.html" \t "_blank" \o "nullif()函数) - 如果第一个参数等于第二个参数，则返回NULL，否则返回第一个参数。

## MySQL日期和时间函数

* [curdate()函数](http://www.yiibai.com/mysql/curdate.html" \t "_blank" \o "curdate()函数) - 返回当前日期。
* [datediff()函数](http://www.yiibai.com/mysql/datediff.html" \t "_blank" \o "datediff()函数) - 计算两个DATE值之间的天数。
* [day()函数](http://www.yiibai.com/mysql/day.html" \t "_blank" \o "day()函数) - 获取指定日期月份的天(日)。
* [date\_add()函数](http://www.yiibai.com/mysql/date_add.html" \t "_blank" \o "date_add()函数) - 将时间值添加到日期值。
* [date\_sub()函数](http://www.yiibai.com/mysql/date_sub.html" \t "_blank" \o "date_sub()函数) - 从日期值中减去时间值。
* [date\_format()函数](http://www.yiibai.com/mysql/date_format.html" \t "_blank" \o "date_format()函数) - 根据指定的日期格式格式化日期值。
* [dayname()函数](http://www.yiibai.com/mysql/dayname.html" \t "_blank" \o "dayname()函数) - 获取指定日期的工作日的名称。
* [dayofweek()函数](http://www.yiibai.com/mysql/dayofweek.html" \t "_blank" \o "dayofweek()函数) - 返回日期的工作日索引。
* [extract()函数](http://www.yiibai.com/mysql/extract.html" \t "_blank" \o "extract()函数) - 提取日期的一部分。
* [now()函数](http://www.yiibai.com/mysql/now.html" \t "_blank" \o "now()函数) - 返回当前日期和时间。
* [month()函数](http://www.yiibai.com/mysql/month.html" \t "_blank" \o "month()函数) - 返回一个表示指定日期的月份的整数。
* [str\_to\_date()函数](http://www.yiibai.com/mysql/str_to_date.html" \t "_blank" \o "str_to_date()函数) - 将字符串转换为基于指定格式的日期和时间值。
* [sysdate()函数](http://www.yiibai.com/mysql/sysdate.html" \t "_blank" \o "sysdate()函数) - 返回当前日期。
* [timediff()函数](http://www.yiibai.com/mysql/timediff.html" \t "_blank" \o "timediff()函数) - 计算两个TIME或DATETIME值之间的差值。
* [timestampdiff()函数](http://www.yiibai.com/mysql/timestampdiff.html" \t "_blank" \o "timestampdiff()函数) - 计算两个DATE或DATETIME值之间的差值。
* [week()函数](http://www.yiibai.com/mysql/week.html" \t "_blank" \o "week()函数) - 返回一个日期的星期数值。
* [weekday()函数](http://www.yiibai.com/mysql/weekday.html" \t "_blank" \o "weekday()函数) - 返回一个日期表示为工作日/星期几的索引。
* [year()函数](http://www.yiibai.com/mysql/year.html" \t "_blank" \o "year()函数) - 返回日期值的年份部分。

## MySQL比较函数

* [coalesce()函数](http://www.yiibai.com/mysql/coalesce.html" \t "_blank" \o "coalesce()函数) - 返回第一个非NULL参数，这非常适合用于将值替换为NULL。
* [greatest()函数&least()函数](http://www.yiibai.com/mysql/greatest-least.html" \t "_blank" \o "greatest()函数&least()函数) – 使用n个参数，并分别返回n个参数的最大值和最小值。
* [isnull()函数](http://www.yiibai.com/mysql/isnull-function.html" \t "_blank" \o "isnull()函数) - 如果参数为NULL，则返回1，否则返回0。

### 五、作业讲解

### 六、当日课程内容总结

### 本次直播课(后)复习巩固补充题

## 练习题表结构和数据

INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (108 ,"曾华"

,'男' ,"1977-09-01",95033);

INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (105 ,"匡明"

,'男' ,"1975-10-02",95031);

INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (107 ,"王丽"

,'女' ,"1976-01-23",95033);

INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (101 ,"李军"

,'男' ,"1976-02-20",95033);

INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (109 ,"王芳"

,'女' ,"1975-02-10",95031);

INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (103 ,"陆君"

,'男' ,"1974-06-03",95031);

CREATE TABLE STUDENT

(SNO VARCHAR(3) NOT NULL,

SNAME VARCHAR(4) NOT NULL,

SSEX VARCHAR(2) NOT NULL,

SBIRTHDAY DATETIME,

CLASS VARCHAR(5));

CREATE TABLE COURSE

(CNO VARCHAR(5) NOT NULL,

CNAME VARCHAR(10) NOT NULL,

TNO VARCHAR(10) NOT NULL);

CREATE TABLE SCORE

(SNO VARCHAR(3) NOT NULL,

CNO VARCHAR(5) NOT NULL,

DEGREE NUMERIC(10, 1) NOT NULL);

CREATE TABLE TEACHER

(TNO VARCHAR(3) NOT NULL,

TNAME VARCHAR(4) NOT NULL, TSEX VARCHAR(2) NOT NULL,

TBIRTHDAY DATETIME NOT NULL, PROF VARCHAR(6),

DEPART VARCHAR(10) NOT NULL);

INSERT INTO COURSE(CNO,CNAME,TNO)VALUES ('3-105' ,'计算机导论',825);

INSERT INTO COURSE(CNO,CNAME,TNO)VALUES ('3-245' ,'操作系统' ,804);

INSERT INTO COURSE(CNO,CNAME,TNO)VALUES ('6-166' ,'数据电路' ,856);

INSERT INTO COURSE(CNO,CNAME,TNO)VALUES ('9-888' ,'高等数学' ,100);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (103,'3-245',86);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (105,'3-245',75);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (109,'3-245',68);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (103,'3-105',92);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (105,'3-105',88);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (109,'3-105',76);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (101,'3-105',64);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (107,'3-105',91);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (108,'3-105',78);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (101,'6-166',85);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (107,'6-106',79);

INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (108,'6-166',81);

INSERT INTO TEACHER(TNO,TNAME,TSEX,TBIRTHDAY,PROF,DEPART)

VALUES (804,'李诚','男','1958-12-02','副教授','计算机系');

INSERT INTO TEACHER(TNO,TNAME,TSEX,TBIRTHDAY,PROF,DEPART)

VALUES (856,'张旭','男','1969-03-12','讲师','电子工程系');

INSERT INTO TEACHER(TNO,TNAME,TSEX,TBIRTHDAY,PROF,DEPART)

VALUES (825,'王萍','女','1972-05-05','助教','计算机系');

INSERT INTO TEACHER(TNO,TNAME,TSEX,TBIRTHDAY,PROF,DEPART)

VALUES (831,'刘冰','女','1977-08-14','助教','电子工程系');

题目：

1、 查询Student表中的所有记录的Sname、Ssex和Class列。

2、 查询教师所有的单位即不重复的Depart列。

3、 查询Student表的所有记录。

4、 查询Score表中成绩在60到80之间的所有记录。

5、 查询Score表中成绩为85，86或88的记录。

6、 查询Student表中“95031”班或性别为“女”的同学记录。

7、 以Class降序查询Student表的所有记录。

8、 以Cno升序、Degree降序查询Score表的所有记录。

9、 查询“95031”班的学生人数。

10、查询Score表中的最高分的学生学号和课程号。

11、查询‘3-105’号课程的平均分。

12、查询Score表中至少有5名学生选修的并以3开头的课程的平均分数。

13、查询最低分大于70，最高分小于90的Sno列。

14、查询所有学生的Sname、Cno和Degree列。

15、查询所有学生的Sno、Cname和Degree列。

16、查询所有学生的Sname、Cname和Degree列。

17、查询“95033”班所选课程的平均分。

19、查询选修“3-105”课程的成绩高于“109”号同学成绩的所有同学的记录。

20、查询score中选学一门以上课程的同学中分数为非最高分成绩的记录。

21、查询成绩高于学号为“109”、课程号为“3-105”的成绩的所有记录。

22、查询和学号为108的同学同年出生的所有学生的Sno、Sname和Sbirthday列。

23、查询“张旭“教师任课的学生成绩。

24、查询选修某课程的同学人数多于5人的教师姓名。

25、查询95033班和95031班全体学生的记录。

26、查询存在有85分以上成绩的课程Cno.

27、查询出“计算机系“教师所教课程的成绩表。

28、查询“计算机系”与“电子工程系“不同职称的教师的Tname和Prof。

29、查询选修编号为“3-105“课程且成绩至少高于选修编号为“3-245”的同学的Cno、Sno和Degree,并按Degree从高到低次序排序。

30、查询选修编号为“3-105”且成绩高于选修编号为“3-245”课程的同学的Cno、Sno和Degree.

31、查询所有教师和同学的name、sex和birthday.

32、查询所有“女”教师和“女”同学的name、sex和birthday.

33、查询成绩比该课程平均成绩低的同学的成绩表。

34、查询所有任课教师的Tname和Depart.

35 查询所有未讲课的教师的Tname和Depart.

36、查询至少有2名男生的班号。

37、查询Student表中不姓“王”的同学记录。

38、查询Student表中每个学生的姓名和年龄。

39、查询Student表中最大和最小的Sbirthday日期值。

40、以班号和年龄从大到小的顺序查询Student表中的全部记录。

41、查询“男”教师及其所上的课程。

42、查询最高分同学的Sno、Cno和Degree列。

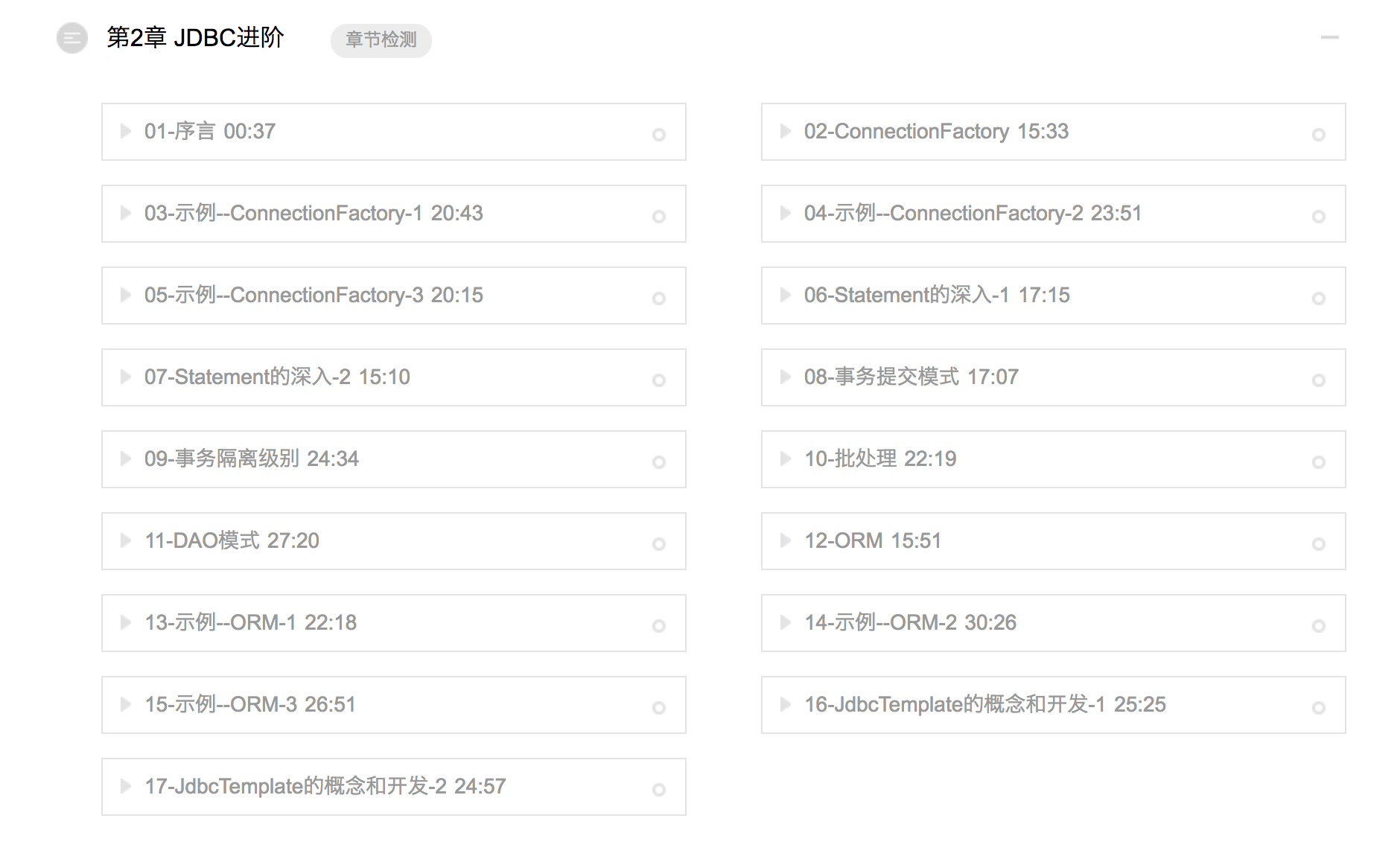
43、查询和“李军”同性别的所有同学的Sname.

44、查询和“李军”同性别并同班的同学Sname.

45、查询所有选修“计算机导论”课程的“男”同学的成绩表

### 八、下次自主学习任务布置





需要通过平台提交作业。完成平台上所布置的作业。