MySQL快速入门

1 什么是数据库？

- 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：Relational Database Management System）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，借助于集合代数等数学概念和方法来处理数据库中的数据。

  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，冗余可以使系统速度更快。

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。

```
1. #Mysql是开源的，所以你不需要支付额外的费用。
2. #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。
3. #MySQL使用标准的SQL数据语言形式。
4. #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。
5. #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。
6. #Mysql是可以定制的，采用了GPL协议，你可以修改源码来开发自己的Mysql系统。
```

2 MySQL 安装

所有平台的Mysql下载地址为： MySQL 下载. 挑选你需要的 MySQL Community Server 版本及对应的平台

2.1 ubuntu上安装Mysql

```
sudo apt-get install mysql-server
sudo apt-get install mysql-client
sudo apt-get install libmysqlclient-dev
```

- 重启mysql服务器

```
sudo /etc/init.d/mysqld restart
```

2.2 Window上安装Mysql

Window上安装Mysql相对来说会较为简单，你只需要载 MySQL 下载中下载window版本的mysql安装包，并解压安装包。

如果报错误2502或者2503，请按如下操作：

```
1、WIN+X键，命令提示符（管理员）；
2、msiexec /package   ‘msi文件路径’
```

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. 所需要的工具都在`C:\Program Files\MySQL\MySQL Server 5.7\bin` 目录中。

2.3 验证Mysql安装

```
mysqladmin --version
```

2.4 密码修改

```
[root@host] mysqladmin -u 用户名 -p 旧密码 password 新密码
```

2.5 MySQL 连接

```
[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。

3 创建数据库

- 可以使用 mysql mysqladmin 命令来创建数据库。

```
[root@host] mysqladmin -u root -p create TESTDB
Enter password:******
```

以上命令执行成功后会创建 MySQL 数据库 TESTDB。

- 登录mysql来创建

```
CREATE DATABASE TESTDB;
```

- 查看有哪些数据库

```
SHOW DATABASES;
```

4 选择数据库

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 TESTDB:

```
mysql> USE TESTDB;
Database changed
mysql>
```

5 删除数据库

- 使用 mysqladmin 删除数据库

```
[root@host] mysqladmin -u root -p drop TESTDB
Enter password:******
```

- 登录mysql删除数据库

```
DROP DATABASE TESTDB
```

6 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。

1. 数值类型

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,例如DECIMAL(5, 2) 范围为-999.99到999.99，占用7个字节，小数点与符号各占用一个字节	依赖于M和D的值	小数值

1. 日期和时间类型

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 00:00:00/2037 年某时	YYYYMMDD HHMMSS	混合日期和时间值，时间戳

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。

DATE可以通过CURDATE()来赋值当前的日期，TIME可以通过CURTIME()来赋值当前的时间，DATETIME与TIMESTAMP都可以通过函数NOW()来赋值当前的时间日期。

1. 字符串类型

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。它们只是可容纳值的最大长度不同。

有4种TEXT类型：TINYTEXT、TEXT、MEDIUMTEXT和LONGTEXT。这些对应4种BLOB类型，有相同的最大长度和存储需求。

7 创建数据表

7.1 创建数据表所需要信息

- 表名
- 表字段名
- 定义每个表字段

7.2 创建数据表的SQL通用语法

1. 简单的创建方式

```
CREATE TABLE table_name (column_name column_type);
```

1. 复杂创建方式

```
CREATE TABLE table_name (column_name column_type [NOT NULL] [DEFAULT default_value]
[AUTO_INCREMENT] [, PRIMARY KEY(column_name0, columen_name1...)]);
```

- 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，数值会自动加1, 默认初始值为0。一个表只能有一个自增字段，并且该字段必须是主键或者索引。

- DEFAULT用来定义默认值，DATE，TIME不能使用函数默认值。
- 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。主键不管有没NOT NULL修饰，都不能为NULL，主键值不能重复。主键可以由多个字段组成。例如：PRIMARY KEY (id, name)

7.3 实例

以下为创建数据表test_tb 实例:

```
mysql> use TESTDB;
Database changed
mysql> CREATE TABLE test_tb(
   -> id INT NOT NULL AUTO_INCREMENT,
   -> name VARCHAR(100) NOT NULL,
   -> age INT NOT NULL,
   -> sub_date DATETIME DEFAULT NOW(),
   -> PRIMARY KEY (id)
   -> );
Query OK, 0 rows affected (0.16 sec)
mysql>
```

注意：MySQL命令终止符为分号 (;) 。

7.4 查看数据库表

```
SHOW  TABLES;
```

7.5 查看数据库表结构

```
DESC TABLES;
```

8 删除数据库表

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。

- mysql语句删除

```
DROP TABLE table_name;
```

9 插入数据

MySQL 表中使用INSERT INTO SQL语句来插入数据。

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 ), (value1, value2,...valueN);
```

注意：如果数据是字符型，必须使用单引号或者双引号，如："value"。

- 实例

  以下实例中我们将想 test_tb表插入三条数据:

```
mysql> use TESTDB;
Database changed
mysql> INSERT INTO test_tb
     ->(name, age, sub_date)
     ->VALUES
     ->("Zhangsan", 20, NOW());
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO test_tb
     ->(name, age, sub_date)
     ->VALUES
     ->("Lisi", 21, NOW());
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO test_tb
     ->(name, age, sub_date)
     ->VALUES
     ->("Wangwu", 23, NOW());
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO test_tb
     ->(name, age, sub_date)
     ->VALUES
     ->("Zhaoliu", 25, '2007-05-06');
Query OK, 1 row affected (0.01 sec)
mysql>
mysql>INSERT INTO test_tb
     ->(name, age)
     ->VALUES
     ->("Zhangsi", 23),
      ->("Liwu", 26);
Query OK, 2 row affected (0.01 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 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。

10 查询数据

以下为在MySQL数据库中查询数据通用的 SELECT 语法：

```
SELECT column_name0,column_name1...
FROM table_name0,table_name1...
[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。
- 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
- 可以使用 WHERE 语句来包含任何条件。
- 可以使用 LIMIT 属性来设定返回的记录数。
- 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, OFFSET必须和LIMIT一起使用。

例如：查看整个数据库表

```
SELECT * FROM test_tb;
```

例如：跳过一条数据，显示3条数据

```
SELECT * FROM test_tb LIMIT 3 OFFSET 1;
```

例如：查询两个表

```
SELECT * FROM test_tb0,test_tb1;
```

11 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。

- 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...]
```

```
1.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。
2.你可以在WHERE子句中指定任何条件。
3.你可以使用AND或者OR指定一个或多个条件。
4.WHERE子句也可以运用于SQL的 DELETE 或者 UPDATE 命令。
5.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
IS NULL	判断是否为NULL	A IS NULL
IS NOT NULL	判断是否不为NULL	A IS NOT NULL

- 例如：查询test_tb表中age大于等于20的所有记录的name,age字段打印出来

```
SELECT name,age FROM test_tb WHERE age >= 20;
```

- 例如：查询test_tb表中age大于19小于22并且name为'Zhangsan'的记录name，age字段打印出来

```
SELECT name,age FROM test_tb WHERE age > 19 AND age < 22 AND name = "Zhangsan";
```

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。

```
mysql>SELECT name,age FROM test_tb WHERE  age > 19 AND age < 22 AND BINARY name = "Zhangsan";
```

注意：BINARY必须紧跟着字段名

- 例如: 查询test_tb表中subdate在2016-12-23 16:12分提交的所有数据的name，age字段。

```
SELECT name,age,sub_date FROM test_tb WHERE sub_date >= "2016-12-23 16:12:00" AND sub_date <
"2016-12-23 16:13:00";
```

例如：查询两个表中名字相同的数据

```
SELECT * FROM test_tb0,test_tb1 WHERE test_tb0.name = test_tb1.name
```

12 修改数据

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。

- 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]
  ```

- 例如：将test_tb中的“Lisi”年龄修改为25

  ```
  UPDATE test_tb SET age=25,sub_date=NOW() WHERE name="Lisi";
  ```

- 例如：将test_tb中所有年龄改为25

```
UPDATE test_tb SET age=25,sub_date=NOW();
```

13 删除数据

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

- 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]
```

- 例如：将test_tb中的“Zhangsan”这条记录删掉：

```
DELETE FROM test_tb WHERE name='Zhangsan';
```

14 LIKE 子句

LIKE子句主要用来匹配字符串的。

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 "name= 'Zhangsan'"。

但是有时候我们需要获取 name字段含有 "Zhang" 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。

- 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_name1, table_name2...
WHERE field1 LIKE condition1 [AND [OR]] filed2 = 'somevalue'
```

- 例如：把test_tb中，姓张的人都找出来:

```
SELECT * FROM test_tb WHERE name LIKE 'Zhang%';
```

15 排序

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中方式来进行排序，再返回搜索结果。

- 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 FROM table_name1, table_name2...
ORDER BY field1, [field2...] [ASC [DESC]]
```

```
1. 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。
2. 可以设定多个字段来排序。
3. 可以使用 ASC(ascending) 或 DESC(descending) 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。
4. 可以添加 WHERE...LIKE 子句来设置条件。
```

- 例如：test_tb按照age升序排列

```
SELECT * FROM test_tb ORDER BY age;
```

- 例如：test_tb按照age降序，相同age的记录再按照id降序

```
 SELECT * FROM test_tb ORDER BY age DESC, id DESC;
```

- 例如：查找出test_tb中name为“Zhang”开头的记录，并且按照id降序排列

```
 SELECT * FROM test_tb WHERE BINARY name LIKE 'Zhang%' ORDER BY id DESC;
```

- 例如：查找出test_tb中age=28的记录，并且按照name升序排序

```
 SELECT * FROM test_tb WHERE BINARY name LIKE 'Zhang%' ORDER BY id DESC;
```

注意：字符串升序排序按照26字母表的顺序进行比较，如果相同就比较第二个字母，以此类推。

- 如果使用LIMIT [M]语句，ORDER BY要写到LIMIT [M]前面

16 常用统计函数

常用的统计函数有COUNT, SUM, AVG。

```
COUNT(field_name)  //返回不同的非NULL值数目。count(*) 它返回检索行的数目， 不论其是否包含NULL值。
SUM(field_name)   //函数是用来计算记录中字段值的总和。
AVG(fidld_name)   //函数是用来计算记录中字段值的平均数。
```

- 例如：统计test_tb中所有记录的年龄总和，以及年龄平均数，以及记录条数

```
mysql> SELECT SUM(age),AVG(age),COUNT(*) FROM test_tb;
+----------+----------+----------+
| SUM(age) | AVG(age) | COUNT(*) |
+----------+----------+----------+
|      194 |  24.2500 |        8 |
+----------+----------+----------+
1 row in set (0.00 sec)

mysql>
```

- 对函数进行更名使用as例如：

```
mysql> SELECT SUM(age) as age_sum, AVG(age) as age_avg, COUNT(*) as records_count FROM test_tb;
+---------+---------+--------------+
| age_sum | age_avg | record_count |
+---------+---------+--------------+
|     194 | 24.2500 |            8 |
+---------+---------+--------------+
1 row in set (0.00 sec)

mysql>
```

17 分组

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;
```

注意：WHERE 与GROUP BY的先后顺序不能颠倒，如果有ORDER BY语句，GROUP BY要写前面，

- 看下面一个实例：

```
mysql> SELECT * FROM test_tb; //查看test_tb
+----+----------+-----+---------------------+
| id | name     | age | sub_date            |
+----+----------+-----+---------------------+
|  2 | Lisi     |  25 | 2016-12-23 16:49:28 |
|  3 | Wangwu   |  23 | 2016-12-23 16:13:07 |
|  4 | Zhaoliu  |  26 | 2016-12-23 16:13:24 |
|  5 | Zhaoliu  |  26 | 2016-12-23 16:13:35 |
|  6 | Zhangwu  |  28 | 2016-12-23 17:37:31 |
|  7 | Zhangsan |  28 | 2016-12-23 17:37:51 |
|  8 | zhangliu |  20 | 2016-12-23 18:13:48 |
|  9 | zhangba  |  18 | 2016-12-23 18:13:48 |
+----+----------+-----+---------------------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM test_tb GROUP BY age; //按照age进行分组后，发现age相同的记录只显示一条出来
+----+----------+-----+---------------------+
| id | name     | age | sub_date            |
+----+----------+-----+---------------------+
|  9 | zhangba  |  18 | 2016-12-23 18:13:48 |
|  8 | zhangliu |  20 | 2016-12-23 18:13:48 |
|  3 | Wangwu   |  23 | 2016-12-23 16:13:07 |
|  2 | Lisi     |  25 | 2016-12-23 16:49:28 |
|  4 | Zhaoliu  |  26 | 2016-12-23 16:13:24 |
|  6 | Zhangwu  |  28 | 2016-12-23 17:37:31 |
+----+----------+-----+---------------------+
6 rows in set (0.00 sec)

mysql> SELECT * FROM test_tb GROUP BY age WHERE age=28; //先后顺序不能颠倒
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to
your MySQL server version for the right syntax to use near 'WHERE age=28' at line 1
mysql> SELECT * FROM test_tb WHERE age=28 GROUP BY age; //这个才是正确的，还是只有一条记录
+----+---------+-----+---------------------+
| id | name    | age | sub_date            |
+----+---------+-----+---------------------+
|  6 | Zhangwu |  28 | 2016-12-23 17:37:31 |
+----+---------+-----+---------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM test_tb WHERE age=28 GROUP BY age;
+----+---------+-----+---------------------+
| id | name    | age | sub_date            |
+----+---------+-----+---------------------+
|  6 | Zhangwu |  28 | 2016-12-23 17:37:31 |
+----+---------+-----+---------------------+
1 row in set (0.00 sec)

mysql>
```

我们可以看到进行age分组之后，多条age相同的记录会分到一组中，并且只显示一条记录作为代表。我们可以调用COUNT, AVG, SUM等函数操作每个分组。

- 统计出age为28的记录数

```
mysql> SELECT *,COUNT(*) FROM test_tb WHERE age=28 GROUP BY age;
+----+---------+-----+---------------------+-------------+
| id | name    | age | sub_date            | COUNT(*) |
+----+---------+-----+---------------------+-------------+
|  6 | Zhangwu |  28 | 2016-12-23 17:37:31 |           2 |
+----+---------+-----+---------------------+-------------+
1 row in set (0.00 sec)

mysql>
```

我们看到这条查询结果会在最后面加一列表示该COUNT(name)来表示函数返回值。

- 统计出不同age的记录数

```
mysql> SELECT age,COUNT(*) FROM test_tb GROUP BY age;
+-----+-------------+
| age | COUNT(*) |
+-----+-------------+
|  18 |           1 |
|  20 |           1 |
|  23 |           1 |
|  25 |           1 |
|  26 |           2 |
|  28 |           2 |
+-----+-------------+
6 rows in set (0.00 sec)

mysql>
```

18 修改数据库表结构

当我们需要修改数据表名或者修改数据表字段时，就需要使用到MySQL ALTER命令。

18.1 删除，添加或修改表字段

- 删除字段

```
ALTER TABLE table_name DROP column_name;
```

如果数据表中只剩余一个字段则无法使用DROP来删除字段。

- 添加字段

```
ALTER TABLE table_name ADD column_name column_type;
```

如果你需要指定新增字段的位置，可以使用MySQL提供的关键字 FIRST (设定位第一列)， AFTER 字段名（设定位于某个字段之后）。

```
ALTER TABLE table_name ADD column_name column_type FIRST;
ALTER TABLE table_name ADD column_name column_type AFTER other_column_name;
```

- 修改字段类型及名称

  如果需要修改字段类型及名称, 你可以在ALTER命令中使用 MODIFY 或 CHANGE 子句 。

  例如，把字段name的类型从 VARCHAR(100) 改为 VARCHAR(20)，可以执行以下命令:

```
mysql> ALTER TABLE test_tb MODIFY name VARCHAR(20);
```

使用 CHANGE 子句, 语法有很大的不同。 在 CHANGE 关键字之后，紧跟着的是你要修改的字段名，然后指定新字段名及类型。尝试如下实例：

```
mysql> ALTER TABLE test_tb CHANGE name names VARCHAR(10);
mysql> ALTER TABLE test_tb CHANGE age age INT;
```

18.2 ALTER TABLE 对 Null 值和默认值的影响

当你修改字段时，你可以指定是否包含只或者是否设置默认值。

以下实例，指定字段 age为 NOT NULL 且默认值为20。

```
mysql> ALTER TABLE test_tb
    -> MODIFY age INT NOT NULL DEFAULT 20;
```

或者

```
mysql> ALTER TABLE test_tb
    -> CHANGE age age INT NOT NULL DEFAULT 20;
```

需要特别注意的是：如果要修改为NOT NULL的字段，已经存在NULL值得记录，则修改不成功。

18.3 修改字段默认值

- 修改默认值

```
ALTER TABLE table_name ALTER column_name SET DEFAULT column_value;
```

- 删除默认值：

```
ALTER TABLE table_name ALTER column_name DROP DEFAULT;
```

- 例如

```
ALTER TABLE test_tb ALTER age SET DEFAULT 10;
ALTER TABLE test_tb ALTER age DROP DEFAULT;
```

18.4 修改表名

```
ALTER TABLE table_name RENAME TO table_name_new;
```

18.5 删除主键

```
ALTER TABLE table_name DROP PRIMARY KEY;
```

18.6 添加一个主键

```
ALTER TABLE table_name ADD PRIMARY KEY(column_name);
```

18.7 添加AUTO_INCREMENT

添加AUTO_INCREMENT之前，首先你要确定将要添加为自增长的字段是否是主键或者索引，自增加的键只能有一个。

```
mysql> ALTER TABLE test_tb MODIFY id INT AUTO_INCREMENT;
```

18.8 修改AUTO_INCREMENT初始值

默认情况下为0，现在改为1000。

```
ALTER TABLE table_name AUTO_INCREMENT=1000;
```

19 索引

MySQL索引的建立对于MySQL的高效运行是很重要的，索引可以大大提高MySQL的检索速度。打个比方，如果合理的设计且使用索引的MySQL是一辆兰博基尼的话，那么没有设计和使用索引的MySQL就是一个人力三轮车。

索引分单列索引和组合索引。单列索引，即一个索引只包含单个列，一个表可以有多个单列索引，但这不是组合索引。组合索引，即一个索包含多个列。

创建索引时，你需要确保该索引是应用在 SQL 查询语句的条件(一般作为 WHERE 子句的条件)。

实际上，索引也是一张表，该表保存了主键与索引字段，并指向实体表的记录。

上面都在说使用索引的好处，但过多的使用索引将会造成滥用。因此索引也会有它的缺点：虽然索引大大提高了查询速度，同时却会降低更新表的速度，如对表进行INSERT、UPDATE和DELETE。因为更新表时，MySQL不仅要保存数据，还要保存一下索引文件。

建立索引会占用磁盘空间的索引文件。

19.1普通索引

- 创建索引

这是最基本的索引，它没有任何限制。它有以下几种创建方式：

```
CREATE INDEX indexName ON table_name(column_name(length));
```

如果是CHAR，VARCHAR类型，length可以小于字段实际长度；如果是BLOB和TEXT类型，必须指定 length。

例如：给test_tb创建了一个建立在age叫做index0的普通索引

```
CREATE INDEX index0 ON test_tb(age);
```

- 修改表结构来添加索引

```
ALTER table_name ADD INDEX [indexName] (column_name(length));
```

- 创建表的时候添加索引

```
CREATE TABLE mytable(

column_name type,

INDEX [indexName] (column_name(length))

);
```

- 删除索引的语法

```
DROP INDEX [indexName] ON table_name;
```

19.2 显示索引信息

你可以使用 SHOW INDEX 命令来列出表中的相关的索引信息。可以通过添加 \G 来格式化输出信息。

尝试以下实例:

```
mysql> SHOW INDEX FROM table_name\G
........
```

19.3 唯一索引

它与前面的普通索引类似，不同的就是：索引列的值必须唯一，但允许有空值。如果是组合索引，则列值的组合必须唯一。它有以下几种创建方式：

- 创建索引

```
CREATE UNIQUE INDEX indexName ON table_name(column_name(length))
```

- 创建表的时候直接创建索引

```
CREATE TABLE mytable(

column_name type,

UNIQUE [indexName] (column_name(length))

);
```

- 删除索引的语法

```
DROP INDEX [indexName] ON table_name;
```

20 事务

主要用于处理操作量大，复杂度高的数据。比如说，在人员管理系统中，你删除一个人员，你即需要删除人员的基本资料，也要删除和该人员相关的信息，如信箱，文章等等，这样，这些数据库操作语句就构成一个事务！

- 在MySQL中只有使用了Innodb数据库引擎的数据库或表才支持事务
- 事务处理可以用来维护数据库的完整性，保证成批的SQL语句要么全部执行，要么全部不执行
- 事务用来管理insert,update,delete语句

在Mysql控制台使用事务来操作

1. 开始一个事务

```
BEGIN
```

1. 操作

1. 可以回滚，可以提交，没有问题，就提交，有问题就回滚。

```
ROLLBACK    回滚
COMMIT      提交事务
```

例如：回滚

```
mysql>BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql>UPDATE test_tb SET name="lisi2" WHERE name="Lisi";
mysql>ROLLBACK;
```

例如：提交事务

```
mysql>BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql>UPDATE test_tb SET name="lisi2" WHERE name="Lisi";
mysql>COMMIT;
```

21 将数据表及数据库拷贝至其他主机

- 备份数据

  如果你需要将数据拷贝至其他的 MySQL 服务器上, 你可以在 mysqldump 命令中指定数据库名及数据表。

在源主机上执行以下命令，将数据备份至 dump.txt 文件中:

```
$ mysqldump -u root -p database_name table_name > dump.txt
password *****
```

如果完整备份数据库，则无需使用特定的表名称。

```
$ mysqldump -u root -p database_name > dump.txt
password *****
```

- 导入数据

  如果你需要将备份的数据库导入到MySQL服务器中，可以使用以下命令，使用以下命令之前你需要确认数据库已经创建：

```
$ mysql -u root -p database_name < dump.txt
password *****
```

22 复制表

- 首先需要复制表结构

```
CREATE TABLE new_table LIKE old_table;
```

- 然后将数据拷贝过来

```
INSERT INTO new_table SELECT * FROM old_table;
```