《MySQL 入门很简单》学习笔记

目录

整理者博客：http://blog.csdn.net/kimsoft
jinqinghua@gmail.com

第 1 章　数据库概述

1.1、数据存储方式

1．人工管理阶段
2．文件系统阶段
3．数据库系统阶段

1.2、数据库泛型

数据库泛型就是数据库应该遵循的规则。数据库泛型也称为范式。目前关系数据库最常用的四种范式分别是：
第一范式（1NF）、第二范式（2NF）、第三范式（3NF）和 BCN 范式（BCNF）

1.3、SQL 语言

SQL（Structured Query Language）语言的全称是结构化查询语言。数据库管理系统通过 SQL 语言来管理数据库中的数据。
SQL 语言分为三个部分：
数据定义语言（Data Definition Language，简称为 DDL）、
数据操作语言（Data Manipulation Language，简称为 DML）
数据控制语言（Data Control Language，简称为 DCL）。
DDL 语句： CREATE TABLE …
DML 语句：SELECT, INSERT, UPDATE ,DELETE
DCL 语句： GRANT, REVOKE

1.4、为什么要使用 MySQL

1．MySQL 是开放源代码的数据库
2．MySQL 的跨平台性
3．价格优势
4．功能强大且使用方便

1.5、常见数据库系统

1．甲骨文的 Oracle
2．IBM 的 DB2
3．微软的 Access 和 SQL Server
4．开源 PostgreSQL
5．开源 MySQL
6．文件数据库 SQLite,
7．内存数据库 HQL

第 2 章 Windows 平台下安装与配置 MySQL

2.1、msi 安装包

2.1.1、安装

特别要注意的是，安装前要删除原来的 my.ini 和原来的 data 目录，改名也行，不然在最后一步会“apply security settings”报个 1045 错误，原因 1，防火墙，原因 2，数据文件未清除。
一路 next，选 custom 安装
可以指定 data 的位置，不要指定到系统盘
顺便配置，选择"detailed configuration"
服务器类型和用途视开发还是生产环境
"best support for multilingualism"　支持大部分语系，默认字符集是 UTF-8，用这个吧
"add firewall exception for this port"　最好选上，尤其在开发机
"enabled strict mode"　生产机推荐，开发机可以不用，选的话，容易出现刚开始要求注意的问题
"include bin directory in windows path"　强烈建议选上，不然要手动配置 path 路径
"create anonymous account"　就不要了
没有意外的话，成功搞定

安装后 root 登录不了的解决办法
mysql -h localhost -u root -p

cmd
net stop mysql
mysqld --skip-grant-tables
#注意，net start mysql --skip-grant-tables，能启动，但是好象达不到效果
窗口可能死掉，不管，另开一个窗口
cmd

mysql -u root

发现直接进去了

use mysql

update user set password=password("新密码") where user='root' and host='localhost';

flush previliges;

OK 了，注意几点：

1、net start mysql --skip-grant-tables，能启动，但是好象达不到效果

2、mysql 是内置的数据库

3、user 表是 mysql 库里存储用户名密码和权限的表

4、密码要用 password()函数加密一下

5、host='localhost'这个条件可以不要的，那么 root 所有的密码都变了，不建议这样做，后面会简单讲一下 mysql 的用户

6、此时 set 方法 mysqlamdin -u root -p password "新密码"的修改密码方法是行不通的，只有直接修改数据库

2.1.2、卸载

1、可以在控制面板里卸载

2、最好通过原来安装包，双击，选"remove"卸载，较彻底

2.2、zip 文件（未验证）

2.2.1、安装

1、下载 mysql

2、解压到 c:/mysql

3、复制 my-large.ini 到 c:/windows/my.ini

4、修改 my.ini 文件

basedir="c:/mysql" 安装目录

datadir="c:/mysql/data" 数据目录

[WindowsMySQLServer]

Server="c:/mysql/bin/mysqld.exe"

5、安装服务

c:/mysql/bin/mysqld.exe --install

6、启动/关闭服务

net start/stop mysql

2.2.2、卸载

c:/mysql/bin/mysqld.exe --remove

2.3、命令常用参数及使用方法

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
? (\?) Synonym for `help'.
clear (\c) Clear the current input statement. （清除输入，不执行）
connect (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter.
ego (\G) Send command to mysql server, display result vertically. （垂直显示）
exit (\q) Exit mysql. Same as quit.
go (\g) Send command to mysql server.
help (\h) Display this help.
notee (\t) Don't write into outfile.
print (\p) Print current command.
prompt (\R) Change your mysql prompt.
quit (\q) Quit mysql.
rehash (\#) Rebuild completion hash.
source (\.) Execute an SQL script file. Takes a file name as an argument.
status (\s) Get status information from the server.
tee (\T) Set outfile [to_outfile]. Append everything into given outfile.
use (\u) Use another database. Takes database name as argument.
charset (\C) Switch to another charset. Might be needed for processing binlog with multi-byte charsets.
warnings (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.

For server side help, type 'help contents'

2.3.1、mysql

-h host
-u user
-p password(注意，一般不输密码，如果输，和-p 之间不能有空格)
-P port，一般是 3306 不常用
databasename 数据库名，相当于执行了 use database
-e "sql"执行语句
mysql -h localhost -u root -ppassword mysql -e "select user,host from user"

2.3.2、mysqladmin

a) 修改密码

mysqladmin -u root -p password "新密码"

注意：

1、password 相当于函数，必须要的

2、新密码要用双引号扩起来

第 3 章 Linux 平台下安装与配置 MySQL

3.1、RPM 文件安装

rpm -ivh mysql.rpm

3.2、二进制文件安装

groupadd mysql

useadd –g mysql mysql

tar –xzvf mysql.bin.tar.gz

scripts/mysql_install_db –user mysql

3.3、源码文件安装

groupadd mysql

useadd –g mysql mysql

tar –xzvf mysql.src.tar.gz

./configure –prfix=/usr/local/mysql

make

mack install

第 4 章 MySQL 数据类型

4.1、整数类型

tinyint(4)
smallint(6)
mediumint(9)
int(11)
bigint(20)
注意：后面的是默认显示宽度，以 int 为例，占用的存储字节数是 4 个，即 4*8=32 位，2 的 32 次方，无符号的最大能达到 4 亿多。
tinyint(4)相当于 bool 型

4.2、浮点数

float
double
decimal(m,d)
decimal(6,2)定义的数字形如 1234.56，指总长 6 位，小数点后精确到 2 位

4.3、日期和时间

year 年
date 日期
time 时间
datetime 日期时间
timestamp 时间（时区），范围小，支持时区
datetime 最通用，year,date,time 可以节省一些空间。

4.4、字符串

char(m) 定长
varchar(m) 不定长

enum,set 和其它库不兼容，可暂不用

tinytext
text
mediumtext
longtext

4.5、二进制

binary(m)
varbinary(m)
bit(m)
tinyblob
blob
mediumblob
longblob

第 5 章 操作数据库

假设已经登录
mysql -h localhost -uroot -proot

5.1、显示、创建、删除数据库

show databases;　　显示所有的数据库
create database xxx;　创建数据库
drop database xxx;　　删除数据库

5.2、数据库存储引擎

show engines \G　　mysql 支持的所有的 engine
show variables like '%engine%';　　查看当前库的 engine
innodb
　　最常用，支持事务，回滚，自增，外键
　　表结构存在.frm 文件中
　　数据和索引存在表空间中
　　读写效率稍差，占用空间大

myisam

表结构存在.frm 文件中

.myd 存储数据

.myi 存储索引

快速，占空间小，不支持事务和并发

memory

演示系统

第 6 章 创建、修改和删除表

6.1、创建表

6.1.1、创建表的语法形式

```
CREATE TABLE 表名（ 属性名 数据类型 [完整性约束条件],
属性名 数据类型 [完整性约束条件],
.....
属性名 数据类型
);
```

完整性约束条件表：

PRIMARY KEY	主键
FOREIGN KEY	外键
NOT NULL	不能为空
UNIQUE	唯一索引
AUTO_INCREMENT	自动增加
DEFAULT	默认值

```
CREATE TABLE example0 (id INT,
                       name VARCHAR(20),
                       sex    BOOLEAN
                       );
```

6.1.2、设置表的主键

单字段主键

属性名 数据类型 PRIMARY KEY

```
CREATE TABLE example1(stu_id INT PRIMARY KEY,
                      stu_name VARCHAR(20),
                      stu_sex BOOLEAN,
                      );
```

多字段主键

PRIMARY KEY(属性名 1，属性名 2...属性名 n)

```
CREATE TABLE example2(stu_id INT,
                      course_id INT,
                      grade FLOAT,
                      PRIMARY KEY(stu_id, course_id)
                      );
```

6.1.3、设置表的外键

CONSTRAINT 外键别名 FOREIGN KEY (属性 1.1，属性 1.2,...，属性 1.n)
REFERENCES 表名(属性 2.1，属性 2.2,...，属性 2.n)

```
CREATE TABLE example3(stu_id INT,
                      course_id INT,
                      CONSTRAINT c_fk FOREIGN KEY (stu_id, course_id)
                            REFERENCES example2(stu_id, course_id)
                      );
```

6.1.4、设置表的非空约束

属性名 数据类型 NOT NULL

```
CREATE TABLE example4(id INT NOT NULL PRIMARY KEY,
                      name VARCHAR(20) NOT NULL,
                      stu_id INT,
                      CONSTRAINT d_fk FOREIGN KEY(stu_id),
                            REFERENCES example1(stu_id)
                      );
```

6.1.5、设置表的唯一性约束

属性名 数据类型 UNIQUE

```
CREATE TABLE example5(id INT PRIMARY KEY,
```

```
                    stu_id INT UNIQUE,
                    name VARCHAR(20) NOT NULL
                    );
```

6.1.6、设置表的属性值自动增加

```
属性名 数据类型 AUTO_INCREMENT
CREATE TABLE example6(id INT PRIMARY KEY AUTO_INCREMENT,
                    stu_id INT UNIQUE,
                    name VARCHAR(20) NOT NULL
                    );
```

6.1.7、设置表的属性的默认值

```
属性名 数据类型 DEFAULT 默认值
CREATE TABLE example7(id INT PRIMARY KEY AUTO_INCREMENT,
                    stu_id INT UNIQUE,
                    name VARCHAR(20) NOT NULL.
                    english VARCHAR(20) DEFAULT   'zero' ,
                    computer   FLOAT DEFAULT 0
                    );
```

6.2、查看表结构

6.2.1、查看表基本结构语句 DESCRIBE

```
DESC 表名
desc example1
```

6.2.2、查看表详细结构语句 SHOW CREATE TABLE

```
SHOW CREATE TABLE 表名;
SHOW CREATE TABLE example example1\G;
```

6.3、修改表

6.3.1、修改表名

ALTER TABLE 旧表名 RENAME [TO] 新表名;

```
DESC example0;
ALTER TABLE example0 RENAME TO user;
```

6.3.2、修改字段的数据类型

ALTER TABLE 表名 MODIFY 属性名 数据类型;

```
DESC user;
ALTER TABLE user MODIFY name VARCHAR(30);
DESC user;
```

6.3.3~6.3.6、字段及数据类型的增、删，改以及改变位置

ALTER TABLE 表名 ADD 属性名 1 数据类型 [完整性约束条件] [FIRST|AFTER 属性名 2];
ALTER TABLE 表名 DROP 属性名;
ALTER TABLE 表名 CHANGE 旧属性名 新属性名 新数据类型;

利用上面的语句可以增加，删除，修改字段。修改字段名，数据类型，和位置

6.3.7、更改表的存储引擎

ALTER TABLE 表名 ENGINE=INNODB|MYISAM|MEMOERY;

```
SHOW CREATE TABLE user\G;
ALTER TABLE user ENGINE=MyISAM;
```

6.3.8、删除表的外键约束

ALTER TABLE 表名 DROP FOREIGN KEY 外键别名 ;

```
SHOW CREATE TABLE example3\G;
ALTER TABLE example3 DROP FOREIGN KEY c_fk;
```

```
SHOW CREATE TABLE example3\G;
```

6.4、删除表

6.4.1、删除没有被关联的普通表

```
DROP TABLE 表名;
DESC example5;
DROP TABLE example5;
DESC example5;
```

6.4.2、删除被其他表关联的父表

```
删除外键后再删除表
DROP TABLE example1; -- 报错
SHOW TABLE example4\G;
ALTER TABLE example4 DROP FOREIGN KEY d_fk;
SHOW TABLE example4\G;
DROP TABLE example1;
DESC example1;
```

第 7 章 索引

MySQL 中，所有的数据类型都可以被索引，包括普通索引，唯一性索引，全文索引，单列索引，多列索引和空间索引等。

7.1、索引简介

7.1.1、索引的含义和特点

BTREE 索引，HASH 索引

优点：提高查询，联合查询，分级和排序的时间

缺点：索引占空间，维护（创建，更新，维护）索引时需要耗费时间

7.1.2、索引的分类

1、普通索引

不加任何限制条件

2、唯一性索引

使用 UNIQUE 参数

3、全文索引

使用 FULLTEXT 参数，只能创建在 CHAR,VARCHAR,TEXT 类型的字段上，只有 MyISAM 存储引擎支持全文索引。

4、单列索引

在一个字段上建立的普通索引，唯一性索引或全文索引

5、多列索引

在多个字段上建立的普通索引，唯一性索引或全文索引

6、空间索引

使用 SPATIAL 参数，只有 MyISAM 存储引擎支持空间索引，必须建立在空间数据类型上，且必须非空，初学者很少用到。

7.1.3、索引的设计原则

1、选择唯一性索引

2、为经常需要排序、分组和联合操作的字段建立索引

如 ORDER BY、GROUP BY、DISTINCT，UNION 等操作的字段，特别是排序

3、为常作为查询条件的字段建立索引

4、限制索引的数目

避免过多地浪费空间

5、尽量使用数据量少的索引

6、尽量使用前缀来索引

如指索引 TEXT 类型字段的前 N 个字符(

Oracle 中有函数索引，这个是不是相当于 left(field, n)式的函数索引？！

7、删除不再使用或者很少使用的索引

7.2、创建索引

三种方式：

1、 创建表时创建索引

2、 已经存在的表上创建索引

3、 使用 ALTER TABLE 语句来创建索引

7.2.1、创建表的时候创建索引

```
CREATE TABLE 表名（属性名 数据类型 [完整约束条件],
                  属性名 数据类型 [完整约束条件],
                  ...
                  [UNIQUE|FULLTEXT|SPATIAL INDEX|KEY [别名] (属性名 1  [(长度)] [ASC|DESC])
                  );
```

1、创建普通索引

```
CREATE TABLE index1 (id INT,
                    name VARCHAR(20),
                    sex BOOLEAN,
                    INDEX(id)
                    );
SHOW CREATE TABLE index1\G;
```

2、创建唯一性索引

```
CREATE TABLE index2(id INT UNIQUE,
                    name VARCHAR(20),
                    UNIQUE INDEX index2_id(id ASC)
                    );
SHOW CREATE TABLE index2\G;
```

看到在字段 id 上建立了两个唯一索引 id 和 index2_id，当然这样是没有必要的。

3、创建全文索引

```
CREATE TABLE index3 (id INT,
                    info VARCHAR(20),
                    FULLTEXT INDEX index3_info(info)
                    ) ENGINE=MyISAM;
```

4、创建单列索引

```
CREATE TABLE index4 (id INT,
```

```
                    subject VARCHAR(30),
                    INDEX index4_st(subject(10))
                    );
```

注意：只索引 subject 前 10 个字符

5、创建多列索引

```
CREATE TABLE index5 (id INT,
                    name VARCHAR(20),
                    sex CHAR(4),
                    INDEX index5_ns(name, sex)
                    );

EXPLAIN select * from index5 where name='123'\G;
EXPLAIN select * from index5 where name='123'and sex='N'\G;
```

6、创建空间索引

```
CREATE TABLE index6 (id INT,
                    Space GEOMETRY NOT NULL,
                    SPATIAL INDEX index6_sp(space)
                    )ENGINE=MyISAM;
```

7.2.2、在已经存在的表上创建索引

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX 索引名 ON 表名 (属性名[(长度)] [ASC|DESC]);

1、创建普通索引

```
CREATE INDEX index7_id on example0(id);
```

2、创建唯一性索引

```
CREATE UNIQUE INDEX index_8_id ON index8(course_id);
```

3、创建全文索引

```
CREATE FULLTEXT INDEX index9_info ON index9(info);
```

4、创建单列索引

```
CREATE INDEX index10_addr ON index10(address(4));
```

5、创建多列索引

```
CREATE INDEX index11_na ON index11(name, address);
```

6、创建空间索引

```
CREATE SPATIAL INDEX index12_line on index12(line);
```

7.2.3、用 ALTER TABLE 语句来创建索引

ALTER TABLE 表名 ADD [UNIQUE|FULLTEXT|SPATIAL] INDEX 索引名 (属性名[(长度)] [ASC|DESC]);

1、创建普通索引

```
ALTER TABLE example0 ADD INDEX index12_name(name(20));
```

2、创建唯一性索引

```
ALTER TABLE index14 ADD UNIQUE INDEX index14_id(course_id);
```

3、创建全文索引

```
ALTER TABLE index15 ADD INDEX index15_info(info);
```

4、创建单列索引

```
ALTER TABLE index 16 ADD INDEX index16_addr(address(4));
```

5、创建多列索引

```
ALTER TABLE index17 ADD INDEX index17_na(name, address);
```

6、创建空间索引

```
ALTER TABLE index18 ADD INDEX index18_line(line);
```

7.3、删除索引

DROP INDEX 索引名 ON 表名;

```
DROP INDEX id ON index1;
```

第 8 章 视图

8.1、视图简介

视图由数据库中的一个表，视图或多个表，视图导出的虚拟表。其作用是方便用户对数据的操作。

8.2、创建视图

必须要有 CREATE VIEW 和 SELECT 权限

SELECT select_priv, create_view_priv from mysql.user WHERE user='root';

CREATE [ALGORITHM = { UNDEFINED | MERGE | TEMPTABLE }]

VIEW 视图名 [(属性清单)]
AS SELECT 语句
[WITH [CASCADED | LOCAL] CHECK OPTION] ;

ALGORITHM 参数表示视图选择的算法
UNDEFINED 未指定，自动选择
MERGE 表示将使用视图的语句和视图定义合并起来，使得视图定义的某一部分取代语句的对应部分
TEMPTABLE 表示将视图的结果存入临时表，然后使用临时表执行语句
LOCAL 参数表示更新视图时要满足该视图本身定义的条件即可；
CASCADED 参数表示更新视图时要满足所有相关视图和表的条件，默认值。
使用 CREATE VIEW 语句创建视图时，最好加上 WITH CHECK OPTION 参数和 CASCADED 参数。这样，从视图上派生出来的新视图后，更新新视图需要考虑其父视图的约束条件。这种方式比较严格，可以保证数据的安全性。

```
create view department_view1 as
  select * from department;

create view department_view2 (name, function, localtion) as
  select d_name, function, address from department;

create algorithm=merge view worker_view1(name,department, sex, age, address) as
  select name, department.d_name,sex, 2009-birthday, address from worker, department where worker.d_id=department.d_id
  with local check option;
```

8.3、查看视图

必须要有 SHOW VIEW 的权限

DESCRIBE | DESC 视图名 ;
SHOW TABLE STATUS LIKE '视图名' ;
SHOW CREATE VIEW 视图名;
SELECT * FROM information_schema.views ;

8.4、修改视图

CREATE OR REPLACE | ALTER [ALGORITHM = { UNDEFINED | MERGE | TEMPTABLE }]
VIEW 视图名 [(属性清单)]
AS SELECT 语句
[WITH [CASCADED | LOCAL] CHECK OPTION] ;

语法和 CREATE VIEW 基本一样

8.5、更新视图

更新视图是指通过视图来插入（INSERT）、更新（UPDATE）和删除（DELETE）表中的数据。因为是视图是一个虚拟表，其中没有数据。通过视图更新时，都是转换到基本表来更新。更新视图时，只能更新权限范围内的数据。超出了范围，就不能更新。

原则：尽量不要更新视图

语法和 UPDATE 语法一样

哪些视图更新不了：

1、视图中包含 SUM(), COUNT()等聚焦函数的

2、视图中包含 UNION、UNION ALL、DISTINCT、GROUP BY、HAVING 等关键字

3、常量视图

CREATE VIEW view_now AS SELECT NOW()

4、视图中包含子查询

5、由不可更新的视图导出的视图

6、创建视图时 ALGORITHM 为 TEMPTABLE 类型

7、视图对应的表上存在没有默认值的列，而且该列没有包含在视图里

8、WITH [CASCADED|LOCAL] CHECK OPTION 也将决定视图是否可以更新

LOCAL 参数表示更新视图时要满足该视图本身定义的条件即可；

CASCADED 参数表示更新视图时要满足所有相关视图和表的条件，默认值。

8.6 、删除视图

删除视图时，只能删除视图的定义，不会删除数据

用户必须拥有 DROP 权限

```
DROP  VIEW  [ IF EXISTS]  视图名列表  [ RESTRICT | CASCADE]
```

第 9 章　触发器

触发器（TRIGGER）是由事件来触发某个操作。这些事件包括 INSERT 语句、UPDATE 语句和 DELETE 语句。当数据库系统执行这些事件时，就会激活触发器执行相应的操作。MySQL 从 5.0.2 版本开始支持触发器

9.1、创建触发器

9.1.1、创建只有一个执行语句的触发器

```
CREATE  TRIGGER  触发器名  BEFORE | AFTER  触发事件
ON  表名  FOR  EACH  ROW  执行语句
```

9.1.2、创建有多个执行语句的触发器

```
DELIMITER &&
CREATE  TRIGGER  触发器名  BEFORE | AFTER  触发事件
ON  表名  FOR  EACH  ROW
BEGIN
执行语句列表
END
&&
DELEMITER  ;
```

DELIMITER，一般 SQL “;”结束，在创建多个语句执行的触发器时，要用到“;”，所以用 DELIMETER 来切换一下。

```
CREATE TRIGGER dept_tig1 BEFORE INSERT ON department
FOR EACH ROW
INSERT INTO trigger_time VALUES(NOWS());
```

9.2、查看触发器

```
SHOW  TRIGGERS ;
SELECT  *  FROM  information_schema. triggers ;
```

9.3、触发器的使用

MySQL 中，触发器执行的顺序是 BEFORE 触发器、表操作（INSERT、UPDATE 和 DELETE）、AFTER 触发器

触发器中不能包含 START TRANSACTION, COMMIT, ROLLBACK,CALL 等。

9.4、删除触发器

DROP　TRIGGER　触发器名；

第 10 章　查询数据

10.1、基本查询语句

```
SELECT 属性列表
    FROM 表名和视图列表
    [WHERE 条件表达式 1]
    [GROUP BY 属性名 1 [HAVING 条件表达式 2]]
    [ORDER BY 属性名 2[ASC|DESC]]
```

10.2、单表查询

列出所有字段

*

指定字段

指定记录

WHERE 条件表达式

=,<,>,!及其组合

[NOT] BETWEEN AND

[NOT] IN

[NOT] LIKE

%

_

IS [NOT] NULL

AND,OR

SELECT DISTINCT 属性名

ORDER BY 属性名 [ASC|DESC]

GROUP BY, GROUP_CONTACT()函数非常好用

```
SELECT sex, GROUP_CONTACT(name) FROM employee GROUP BY sex;
```

GROUP BY 与 WITH ROLLUP 一起使用，多一行，加统计

```
SELECT sex COUNT(sex) FROM employee GROUP BY sex WITH ROLLUP;
```

LIMIT [初始位置,] 记录数

10.3、使用集合函数查询

```
COUNT(), AVG(), MAX(), MIN(), SUM()
```

10.4、连接查询

10.4.1、内连接查询

```
select a.*, b.* from a, b where a.xid=b.xid
```

10.4.2、外连接查询

```
SELECT 属性名列表 FROM 表名 1 LEFT|RIGHT JOIN 表名 2
    ON 表名 1.属性 1 = 表名 2.属性 2；
```

LEFT JOIN 左表全记录，右表符合条件

RIGHT JOIN 右表全记录，左表符合条件

10.5、子查询

IN

EXISTS 表示存在，内层查询语句不返回查询的记录，而是返回一个真假值 （ true|false ）

ANY 任意一个值

```
SELECT * FROM computer_stu WHERE scrore >=ANY(SELECT score FROM scholarship)
```

ALL 满足所有条件

10.6、合并查询结果

```
SELECT 语句 1
    UNION | UNION ALL
SELECT 语句 2
...
```

UNION 所有的查询结果合并到一起，去掉重复项
UNION ALL 简单合并

10.7、为表和字段取别名

表名 表的别名
属性名 [AS] 属性的别名

10.8、使用正则表达式查询

属性名 REGEXP '匹配方式'

^	字符串开始
$	字符串结束
.	任意一个字符，包括回车和换行
[字符集合]	匹配字符集合中的任一字符
S1|S2|S3	三者之任一
*	任意多个
+	1+个
字符串{N}	字符串出现 N 次
字符串{M,N}	字符串出现至少 M 次，至多 N 次

```
SELECT * FROM info WHERE name REGEXP 'ab{1,3}';
```

第 11 章 插入、更新与删除数据

11.1、插入数据

11.1.1、为表的所有字段插入数据

1、INSERT 语句中不指定具体的字段名
insert into 表名 values (值 1,值 2...值 n)
2、INSERT 语句中列出所有字段
insert into 表名 (属性 1,属性 2...属性 n) values (值 1,值 2...值 n)

11.1.2、为表的指定字段插入数据

```
insert into 表名 (属性 1,属性 2,属性 3) values (值 1,值 2,值 3)
```

11.1.3、同时插入多条数据

```
insert into 表名 (属性 1,属性 2...属性 n) values (值 1,值 2...值 n),
                (属性 1,属性 2...属性 n) values (值 1,值 2...值 n),
                ...
                (属性 1,属性 2...属性 n) values (值 1,值 2...值 n);
```

MySQL 特有的

11.1.4、将查询结果插入到表中

```
insert into 表名 (属性 1,属性 2...属性 n)
      select 属性列表 from 表名 2 where 条件表达式
```

use mysql;

create table user1 (user char(16) not null);

insert into user1 (user) select user from user;

11.2、更新数据

```
update 表名
        set 属性 1=值 1,属性 2=值 2，...
        where 条件表达式;
```

11.3、删除数据

```
delete from 表名 [条件表达式];
```

第 12 章 MySQL 运算符

12.1、算术运算符

+
-
*
/或 DIV
%或 MOD

12.2、比较运算符

1．运算符“=”
2．运算符“<>”和“!=”
3．运算符“<=>”
4．运算符“>”
5．运算符“>=”
6．运算符“<”
7．运算符“<=”
8．运算符“IS NULL”
9．运算符“BETWEEN AND”
10．运算符“IN”
11．运算符“LIKE”
12．运算符“REGEXP”

12.3、逻辑运算符

1．与运算
2．或运算
3．非运算
4．异或运算

12.4、位运算符

1．按位与
2．按位或
3．按位取反
4．按位异或

5．按位左移与按位右移

第 13 章 MySQL 函数

13.1、数学函数

随机数可能会用到，其他基本无视。

13.2、字符串函数

重点 CONCAT(S1,S2….)

13.3、日期和时间函数

重点

13.4、条件判断函数

IF(expr,v1,v2)

IFNULL(v1,v2)

CASE

1．CASE WHEN expr1 THEN v1 [WHEN expr2 THEN v2…] [ELSE vn] END

2．CASE expr WHEN e1 THEN v1 [WHEN e2 THEN v2…] [ELSE vn] END

13.5、系统信息函数

VERSION()函数返回数据库的版本号；

CONNECTION_ID()函数返回服务器的连接数，也就是到现在为止 MySQL 服务的连接次数；

DATABASE()和 SCHEMA()返回当前数据库名。

USER()、SYSTEM_USER()、SESSION_USER()、CURRENT_USER()和 CURRENT_USER 这几个函数可以返回当前用户的名称

CHARSET(str)函数返回字符串 str 的字符集，一般情况这个字符集就是系统的默认字符集；

COLLATION(str)函数返回字符串 str 的字符排列方式。

LAST_INSERT_ID()函数返回最后生成的 AUTO_INCREMENT 值。

13.6、加密函数

PASSWORD(str)函数可以对字符串 str 进行加密。一般情况下，PASSWORD(str)函数主要是用来给用户的密码加密的。
MD5(str)函数可以对字符串 str 进行加密。MD5(str)函数主要对普通的数据进行加密。
ENCODE(str,pswd_str)函数可以使用字符串 pswd_str 来加密字符串 str。加密的结果是一个二进制数，必须使用 BLOB 类型的字段来保存它。
DECODE(crypt_str,pswd_str)函数可以使用字符串 pswd_str 来为 crypt_str 解密

13.7、格式化函数

FORMAT(x,n)函数可以将数字 x 进行格式化，将 x 保留到小数点后 n 位。这个过程需要进行四舍五入。

ASCII(s)返回字符串 s 的第一个字符的 ASCII 码；
BIN(x)返回 x 的二进制编码；
HEX(x)返回 x 的十六进制编码；
OCT(x)返回 x 的八进制编码；
CONV(x,f1,f2)将 x 从 f1 进制数变成 f2 进制数。

INET_ATON(IP)函数可以将 IP 地址转换为数字表示；INET_NTOA(n)函数可以将数字 n 转换成 IP 的形式。其中，INET_ATON(IP)函数中 IP 值需要加上引号。这两个函数互为反函数。

GET_LOCT(name,time)函数定义一个名称为 nam、持续时间长度为 time 秒的锁。如果锁定成功，返回 1；如果尝试超时，返回 0；如果遇到错误，返回 NULL。
RELEASE_LOCK(name)函数解除名称为 name 的锁。如果解锁成功，返回 1；如果尝试超时，返回 0；如果解锁失败，返回 NULL；
IS_FREE_LOCK(name)函数判断是否使用名为 name 的锁。如果使用，返回 0；否则，返回 1。

BENCHMARK(count,expr)函数将表达式 expr 重复执行 count 次，然后返回执行时间。该函数可以用来判断 MySQL 处理表达式的速度。

CONVERT(s USING cs)函数将字符串 s 的字符集变成 cs
CAST(x AS type)和 CONVERT(x,type)这两个函数将 x 变成 type 类型。这两个函数只对 BINARY、CHAR、DATE、DATETIME、TIME、SIGNED INTEGER、UNSIGNED INTEGER 这些类型起作用。但两种方法只是改变了输出值的数据类型，并没有改变表中字段的类型

第 14 章 存储过程和函数

避免编写重复的语句

安全性可控

执行效率高

14.1、创建存储过程和函数

14.1.1、创建存储过程

CREATE PROCEDURE sp_name ([proc_parameter[,...]])
[characteristic ...] routine_body

procedure 发音 [prə'si:dʒə]

proc_parameter IN|OUT|INOUT param_name type

characteristic n. 特征；特性；特色

LANGUAGE SQL 默认，routine_boyd 由 SQL 组成

[NOT] DETERMINISTIC 指明存储过程的执行结果是否是确定的，默认不确定

CONSTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA 指定程序使用 SQL 语句的限制

CONSTAINS SQL 子程序包含 SQL，但不包含读写数据的语句，默认

NO SQL 子程序中不包含 SQL 语句

READS SQL DATA 子程序中包含读数据的语句

MODIFIES SQL DATA 子程序中包含了写数据的语句

SQL SECURITY {DEFINER|INVOKER}，指明谁有权限执行。

DEFINER，只有定义者自己才能够执行，默认

INVOKER 表示调用者可以执行

COMMENT 'string' 注释信息

```
CREATE PROCEDURE num_from_employee (IN emp_id, INT, OUT count_num INT)
    READS SQL DATA
    BEGIN
        SELECT COUNT(*)  INTO count_num
        FROM employee
        WHERE d_id=emp_id;
    END
```

14.1.2、创建存储函数

CREATE FUNCTION sp_name ([func_parameter[,...]])
RETURNS type
[characteristic ...] routine_body

```
CREATE FUNCTION name_from_employee(emp_id INT)
    RETURNS VARCHAR(20)
    BEGIN
        RETURN (SELECT name FROM employee WHERE num=emp_id);
    END
```

14.1.3、变量的使用

1．定义变量

DECLARE var_name[,…] type [DEFAULT value]

DECLARE my_sql INT DEFAULT 10;

2．为变量赋值

SET var_name=expr[,var_name=expr]…

SELECT col_name[,…] INTO var_name[,…] FROM table_name WHERE condition

14.1.4、定义条件和处理程序

1．定义条件

DECLARE condition_name CONDITION FOR condition_value

condition value:

SQLSTATE [VALUE] sqlstate_value | mysql_error_code

对于 ERROR 1146(42S02)

sqlstate_value: 42S02

mysql_error_code:1146

//方法一

DECLARE can_not_find CONDITION FOR SQLSTATE ‘42S02’

//方法二

DECLARE can_not_find CONDITION FOR 1146

2．定义处理程序

DECLARE hander_type HANDLER FOR condition_value[,…] sp_statement

handler_type:

CONTINUE|EXIT|UNDO

condition_value:

SQLSTATE[VALUE] sqlstate_value | condition_name | SQLWARNING|NOTFOUND|SQLEXCEPTION|mysql_error_code

UNDO 目前 MySQL 不支持

1、捕获 sqlstate_value

DECLARE CONTINUE HANDLER FOR SQLSTATE ‘42S02’ SET @info=’CAN NOT FIND’;

2、捕获 mysql_error_code

DECLARE CONTINUE HANDLER FOR 1146 SET @info=’CAN NOT FIND’;

3、先定义条件，然后调用

DECLARE can_not_find CONDITION FOR 1146;

DECLARE CONTINUE HANDLER FOR can_not_find SET @info=’CAN NOT FIND’;

4、使用 SQLWARNING

DECLARE EXITHANDLER FOR SQLWARNING SET @info=’CAN NOT FIND’;

5、使用 NOT FOUND

DECLARE EXIT HANDLER FOR NOT FOUND SET @info=’CAN NOT FIND’;

6、使用 SQLEXCEPTION

DECLARE EXIT HANDLER FOR SQLEXCEPTION SET @info=’CAN NOT FIND’;

14.1.5、光标的使用

存储过程中对多条记录处理，使用光标

1．声明光标

DECLARE cousor_name COURSOR FOR select statement;

DECLARE cur_employee CURSOR FOR SELECT name, age FROM employee;

2．打开光标

OPEN cursor_name;

```
OPEN cur_employee;
```

3．使用光标

```
FETCH cur_employee INTO var_name[,var_name…];
```

```
FETCH cur_employee INTO emp_name, emp_age;
```

4．关闭光标

```
CLOSE cursor_name
```

```
CLOSE cur_employee
```

14.1.6、流程控制的使用

1．IF 语句

```
IF search_condition THEN statement_list
    [ELSEIF search_condition THEN statement_list]…
    [ELSE statement_list]
END IF
```

```
IF age>20 THEN SET @count1=@count1+1;
    ELSEIF   age=20 THEN @count2=@count2+1;
    ELSE @count3=@count3+1;
END
```

2．CASE 语句

```
CASE case_value
    WHEN when_value THEN statement_list
    [WHEN when_value THEN statement_list]…
    [ELSE statement_list]
END CASE
```

```
CASE
    WHEN search_condition THEN statement_list
    [WHEN search_condition THEN statement_list]...
    [ELSE statement_list]
END CASE
```

```
CASE age
    WHEN 20 THEN SET @count1=@count1+1;
    ELSE SET @count2=@count2+1;
END CASE;
```

```
CASE
    WHERE age=20 THEN SET @count1=@count1+1;
    ELSE SET @count2=@count2+1;
END CASE;
```

3．LOOP 语句

```
[begin_label:]LOOP
    statement_list
END LOOP[end_label]
```

```
add_num:LOOP
    SET @count=@count+1;
END LOOP add_num;
```

4．LEAVE 语句

跳出循环控制

LEAVE label

```
add_num:LOOP
    SET @count=@count+1;
    LEAVE add_num;
END LOOP add_num;
```

5．ITERATE 语句

跳出本次循环，执行下一次循环

```
ITERATE label
```

```
add_num:LOOP
    SET @count=@count+1;
    IF @count=100 THEN LEAVE add_num;
    ELSEIF MOD(@count,3)=0 THEN ITERATE add_num;
    SELECT * FROM employee;
END LOOP add_num;
```

6．REPEAT 语句

有条件循环，满足条件退出循环

```
[begin_label:]REPEAT
    statement_list
    UNTIL search_condition
END REPEAT[end_label]
```

```
REPEAT
    SET @count=@count+1;
    UNTIL @count=100;
END REPEAT;
```

7．WHILE 语句

```
[begin_label:]WHILE search_condition DO
    statement_list
END REPEAT[end_label]
```

```
WHILE @count<100 DO
    SET @count=@count+1;
END WHILE;
```

14.2、调用存储过程和函数

存储过程是通过 CALL 语句来调用的。而存储函数的使用方法与 MySQL 内部函数的使用方法是一样的。执行存储过程和存储函

数需要拥有 EXECUTE 权限。EXECUTE 权限的信息存储在 information_schema 数据库下面的 USER_PRIVILEGES 表中

14.2.1、调用存储过程

```
CALL   sp_name([parameter[,...]]) ;
```

14.2.2、调用存储函数

存储函数的使用方法与 MySQL 内部函数的使用方法是一样的

14.3、查看存储过程和函数

```
SHOW { PROCEDURE | FUNCTION } STATUS [ LIKE   ' pattern ' ] ;
SHOW CREATE { PROCEDURE | FUNCTION } sp_name ;
SELECT * FROM information_schema.Routines WHERE ROUTINE_NAME=' sp_name ' ;
```

14.4、修改存储过程和函数

```
ALTER {PROCEDURE | FUNCTION} sp_name [characteristic ...]
characteristic:
{ CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
| SQL SECURITY { DEFINER | INVOKER }
| COMMENT 'string'
```

14.5、删除存储过程和函数

```
DROP { PROCEDURE| FUNCTION } sp_name;
```

第 15 章 MySQL 用户管理

15.2、账户管理

15.2.1、登录和退出 MySQL 服务器

```
mysql –h hostname|hostIP –P port –u username –p[password] databaseName –e “SQL 语句”
```

-h	主机名或 ip	
-P	port[3306]	
-u	username	
-p	-p[password]	注意，之间没有空格
-e	执行 SQL 语句	SQL 用双引号括起

可以用此语句配合操作系统定时任务，达到自动处理表数据的功能，如定时将某表中过期的数据删除。

15.2.2、新建立普通用户

1、用 CREATE USER 语句新建

```
CREATE USER user [IDENTIFIED BY [PASSWORD]’ password’ ]
                [,user [IDENTIFIED BY [PASSWORD]’ password’ ]]...
```

2、用 INSERT 语句来新建普通用户

```
INSERT INTO mysql.user(host,user,password,ssl_cipher,x509_issuer,x509_subject)
VALUES ( ‘localhost’,’ test2’,PASSWORD( ‘test2’ ),’ ’ ,’ ’ ,’ ’ ,’ ’ );
FLUSH PRIVELEGES;
```

3、用 GRANT 语句来新建普通用户

```
GRANT priv_type ON database.table TO user [IDENTIFIED BY [PASSWORD]’ password
                                        [,user [IDENTIFIED BY [PASSWORD]’ password’ ]]...
```

15.2.3、删除普通用户

1、用 DROP USER 语句来删除普通用户

```
DROP USER user [,user]…;
```

例：drop user ‘test’@’localhost’;

2、用 DELETE 语句来删除普通用户

```
DELETE FROM mysql.user WHERE user=’username’ and host=’hostname’;
FLUSH PRIVILEGES;
```

15.2.4、root 用户修改自己的密码

1、使用 mysqladmin 命令来修改 root 用户的密码

```
mysqladmin –u username –p password “new_password”;
```

2、修改 user 表

```
UPDATE mysql.user SET password=PASSWORD(“new_password”) WHERE user=’root’ and host=’’;
FLUSH PRIVILEGES;
```

3、使用 SET 语句来修改 root 用户的密码

```
SET PASSWORD=PASSWORD(“new_password”);
```

15.2.5、root 用户修改普通用户密码

1、使用 mysqladmin 命令

不适用，mysqladmin 只能修改 root 用户密码

2、修改 user 表

```
UPDATE mysql.user SET password=PASSWORD(“new_password”) WHERE user=’’ and host=’’;
FLUSH PRIVILEGES;
```

3、使用 SET 语句来修改普通用户的密码

```
SET PASSWORD FOR ‘user’@’localhost’=PASSWORD(“new_password”);
```

4、用 GRANT 语句来修改普通用户的密码

```
GRANT priv_type ON database.table TO user [IDENTIFIED BY [PASSWORD]’password
                                        [,user [IDENTIFIED BY [PASSWORD]’password’]]...
```

15.2.6、普通用户修改密码

```
SET PASSWORD=PASSWORD(“new_password”);
```

15.2.7、root 用户密码丢失的解决办法

1、使用—skip-grant-tables 选项来启动 MySQL 服务

```
>mysqld –skip-grant-tables
```

```
#/etc/init.d/mysql start –mysqld –skip-grant-tables
```

2、登录 root，设置新密码

```
mysql –u root
update mysql.user set password=password(“new_password”) where user=’root’ and host=’localhost’;
```

3、加载权限表

```
fush privileges;
```

15.3、权限管理

15.3.1、MySQL 的各种权限

create, select, update, delete

all [privileges]　　指所有权限

15.3.2、授权

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON [object_type] {tbl_name | * | *.* | db_name.*}
    TO user [IDENTIFIED BY [PASSWORD] 'password']
        [, user [IDENTIFIED BY [PASSWORD] 'password']] ...
    [REQUIRE
        NONE |
        [{SSL| X509}]
        [CIPHER 'cipher' [AND]]
        [ISSUER 'issuer' [AND]]
        [SUBJECT 'subject']]
    [WITH with_option [with_option] ...]

object_type =
    TABLE
```

```
  | FUNCTION
  | PROCEDURE

with_option =
    GRANT OPTION
  | MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
  | MAX_CONNECTIONS_PER_HOUR count
  | MAX_USER_CONNECTIONS count
```

```
grant select, insert, update, delete on testdb.* to common_user@'%'
```

database.table　　*.*所有库的所有表
user　　‘user’@’host’

15.3.3、收回权限

```
REVOKE priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON [object_type] {tbl_name | * | *.* | db_name.*}
    FROM user [, user] ...

REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...
```

第 16 章 数据备份与还原

16.1、数据备份

16.1.1、使用 mysqldump 命令备份

```
mysqldump [OPTIONS] database [tables]
mysqldump [OPTIONS] --databases [OPTIONS] DB1 [DB2 DB3...]
mysqldump [OPTIONS] --all-databases [OPTIONS]
```

```
mysqldump –u root –p test student >c:/student.sql
mysqldump –u root –p test mysql > c:/multidb.sql
mysqldump –u root –p –all-databases > c:/all.sql
```

16.1.2、直接复制整个数据库目录

MyISAM 存储引擎的的表适用

大版本号相同数据库数据库文件格式相同

16.1.3、使用 mysqlhotcopy 工具快速备份

Linux 下备份，perl 脚本。

16.2、数据还原

16.2.1、使用 mysql 命令还原

```
mysql –u root –p [dbname] < backup.sql
mysql –u root –p < all.sql
```

16.2.2、直接复制到数据库目录

16.3、数据库迁移

16.3.1、相同版本的 MySQL 数据库之间的迁移

```
mysqldump –h host1 –u root –password=password1 –all-databases | mysql –h host2 –u root –password=password2
mysqldump –h host1 –u root –ppassword databasename | mysql –h host2 –u root –ppassword databasename
```

16.3.2、不同版本的 MySQL 数据库之间的迁移

mysqldump

16.3.3、不同数据库之间的迁移

1、工具，如 MS SQL Server 的数据库迁移工具

2、dump 出 sql 语句，然后手工修改 create 语句

16.4、表的导出和导入

16.4.1、用 SELECT...INTO OUTFILE 导出文本文件

```
SELECT [列名] FROM table [WHERE 语句]
      INTO OUTFILE  '目标文件'  [OPTION]
```

能根据条件导出数据

16.4.2、用 mysqldump 命令导出文本文件

```
mysqldump –u root –pPassword –T  目标目录或文件  dbname table [option];
```

--fields-terminated-by=... ,
--fields-enclosed-by=... ,
--fields-optionally-enclosed-by=... ,
--fields-escaped-by=... ,
--fields-terminated-by=...

导出的是 txt + sql 文件

16.4.3、用 mysql 命令导出文本文件

```
mysql –u root –pPassword –e  "sql"  dbname > c:/sql.txt
mysql –u root –pPassword   --xml   | -X -e  "sql"  dbname > c:/sql.txt
mysql –u root –pPassword   --html | -H -e  "sql"  dbname > c:/sql.txt
```

16.4.4、用 LOAD DATA INFILE 方式导入文本文件

```
LOAD DATA[LOCAL] INFILE file INTO TABLE table [OPTION]
LOAD DATA INFILE C:/student.txt INTO TABLE student [OPTION]
```

16.4.5、用 mysqlimport 命令导入文本文件

```
mysqlimport –u root –pPassword [--LOCAL] dbname file [OPTION]
```

第 17 章 MySQL 日志

17.1、日志简介

二进制日志
错误日志
通用查询日志
慢查询日志

17.2、二进制日志

二进制日志也叫作变更日志（update log），主要用于记录数据库的变化情况。通过二进制日志可以查询 MySQL 数据库中进行了哪些改变。

17.2.1、启动和设置二进制日志

默认关闭
my.cnf（Linux 操作系统下）或者 my.ini（Windows 操作系统下）
[mysqld]
log-bin [=DIR \ [filename]]

DIR 和 filename 可以不指定，默认为 hostname-bin.number，同时生成 hostname-bin.index 文件

17.2.2、查看二进制日志

mysqlbinlog filename.number

17.2.3、删除二进制日志

1．删除所有二进制日志
RESET MASTER;

2．根据编号来删除二进制日志

PURGE MASTER LOGS TO ‘filename.number’

清除编号小于 number 的所有二进制文件

3．根据创建时间来删除二进制日志

PURGE MASTER LOGS TO ‘yyyy-mm-dd hh:MM:ss’

删除指定时间之前的

17.2.4、使用二进制日志还原数据库

mysqlbinlog filename.number | mysql -u root –p

number 编号小的先还原

17.2.5、暂时停止二进制日志功能

SET SQL_LOG_BIN=0

17.3、错误日志

错误日志是 MySQL 数据库中最常用的一种日志。错误日志主要用来记录 MySQL 服务的开启、关闭和错误信息。

17.3.1、启动和设置错误日志

默认开启的，而且，错误日志无法被禁止。

默认情况下，错误日志存储在 MySQL 数据库的数据文件夹下。错误日志文件通常的名称为 hostname.err。其中，hostname 表示 MySQL 服务器的主机名。错误日志的存储位置可以通过 log-error 选项来设置。将 log-error 选项加入到 my.ini 或者 my.cnf 文件的[mysqld]组中，形式如下：

```
# my.cnf（Linux 操作系统下）或者 my.ini（Windows 操作系统下）
[mysqld]
log-error=DIR / [filename]
```

17.3.2、查看错误日志

文本编辑/查看器

17.3.3、删除错误日志

MySQL 数据库中，可以使用 mysqladmin 命令来开启新的错误日志。mysqladmin 命令的语法如下：
mysqladmin -u root -p flush-logs
执行该命令后，数据库系统会自动创建一个新的错误日志。旧的错误日志仍然保留着，只是已经更名为 filename.err-old。

17.4、通用查询日志

通用查询日志用来记录用户的所有操作，包括启动和关闭 MySQL 服务、更新语句、查询语句等。

17.4.1、启动和设置通用查询日志

默认情况下，通用查询日志功能是关闭的
通过 my.cnf 或者 my.ini 文件的 log 选项可以开启通用查询日志。将 log 选项加入到 my.cnf 或者 my.ini 文件的[mysqld]组中，形式如下：
my.cnf（Linux 操作系统下）或者 my.ini（Windows 操作系统下）
[mysqld]
log [=DIR \ [filename]]

17.4.2、查看错误日志

文本编辑/查看器

17.4.3、删除通用查询日志

可以使用 mysqladmin 命令来开启新的通用查询日志。新的通用查询日志会直接覆盖旧的查询日志，不需要再手动删除了。
mysqladmin 命令的语法如下：
mysqladmin -u root -p flush-logs

17.5、慢查询日志

慢查询日志用来记录执行时间超过指定时间的查询语句。通过慢查询日志，可以查找出哪些查询语句的执行效率很低，以便进行优化。

17.5.1、启动和设置慢查询日志

默认情况下，慢查询日志功能是关闭的。
通过 my.cnf 或者 my.ini 文件的 log-slow-queries 选项可以开启慢查询日志。通过 long_query_time 选项来设置时间值，时间以秒为单位。如果查询时间超过了这个时间值，这个查询语句将被记录到慢查询日志。将 log-slow-queries 选项和 long_query_time 选项加入到 my.cnf 或者 my.ini 文件的[mysqld]组中，形式如下：

```
# my.cnf（Linux 操作系统下）或者 my.ini（Windows 操作系统下）
[mysqld]
log-slow-queries [=DIR \ [filename] ]
long_query_time=n
```

17.5.2、查看慢查询日志

文本编辑/查看器

17.5.3、删除慢查询日志

慢查询日志的删除方法与通用查询日志的删除方法是一样的。可以使用 mysqladmin 命令来删除。也可以使用手工方式来删除。
mysqladmin 命令的语法如下：

```
mysqladmin -u root -p flush-logs
```

执行该命令后，命令行会提示输入密码。输入正确密码后，将执行删除操作。新的慢查询日志会直接覆盖旧的查询日志，不需要再手动删除了。数据库管理员也可以手工删除慢查询日志。删除之后需要重新启动 MySQL 服务。重启之后就会生成新的慢查询日志。如果希望备份旧的慢查询日志文件，可以将旧的日志文件改名。然后重启 MySQL 服务

17.6、小结

日志类型	配置 my.cnf 或 my.ini	默认 启动	查看	删除
二进制 日志	[mysqld] log-bin [=DIR \ [filename]]	否	mysqlbinlog filename.number	RESET MASTER; PURGE MASTER LOGS TO ‘filename.number’ PURGE MASTER LOGS TO ‘yyyy-mm-dd hh:MM:ss’
错误	[mysqld]	是	文本查看/编辑器	mysqladmin -uroot -p flush-logs

日志	log-error[=DIR \ [filename]]			
通用查询日志	[mysqld] log [=DIR \ [filename]]	否	文本查看/编辑器	mysqladmin -uroot -p flush-logs
慢查询日志	log-slow-queries[=DIR \ [filename]] long_query_time=n	否	文本查看/编辑器	mysqladmin -uroot -p flush-logs

第 18 章 性能优化

18.1、优化简介

SHOW STATUS LIKE ‘value’;

connections 连接数
uptime 启动时间
slow_queries 慢查询次数
com_select 查询操作次数
com_insert 插入操作次数
com_update 更新操作次数
com_delete 删除操作次数

18.2、优化查询

18.2.1、分析查询语句

EXPLAIN/DESC select;

type: 连接类型
system 表中只有一条记录
const 表中有多条记录，但只从表中查询一条记录
all 对表进行了完整的扫描
eq_ref 表示多表连接时，后面的表使用了 unique 或 PRIMARY KEY
ref 表示多表查询时，后面的表使用了普通索引
unique_subquery 表示子查询中合作了 unique 或 primary key
index_subquery 表示子查询中使用了普通索引
range 表示查询中给出了查询的范围

index　　表示对表中索引进行了完整的扫描

possible_key　　表示查询中可能使用的索引
key　　表示查询时使用到的索引

18.2.2、索引

1、走单列索引
2、走多列索引
3、不走索引的查询
　　Like 以 %开头的不走
　　Or 两边的列有一个没有建立索引不走索引
　　多列索引第一个字段没有使用不走索引

18.3、优化数据库结构

18.3.1、将字段很多的表分解成多个表

18.3.2、增加中间表

18.3.3、增加冗余字段

　　反范式
　　空间换时间

18.3.4、优化插入记录的速度

1、禁用索引
　　ALTER TABLE table DISABLE/ENABLE KEYS;
2、禁用唯一索引
　　SET UNIQUE_CHECK=0/1
3、优化 INSERT 语句
　　使用 INSERT INTO table (f1,f2….fn) VALUES (v1,v2….vn),
　　(f1,f2….fn) VALUES (v1,v2….vn),
　　(f1,f2….fn) VALUES (v1,v2….vn),
　　…

代替多个 INSERT INTO

18.3.5、分析、检查和优化表

ANALYZE TABLE table1[, table2…]

CHECK TABLE table1[, table2…]

OPTIMIZE TABLE table1[, table2…]

优化文本字段，消除更新操作带来的碎片，减少空间浪费

18.4、优化 MySQL 服务器

18.4.1、优化服务器硬件

CPU

磁盘，阵列

内存

配置（专用服务器，大内存配置）

18.4.2、优化 MySQL 参数

my.ini