《MySQL 入门很简单》学习笔记

目录

)ENGINE=MyISAM DEFAULT CHARSET=gbk;
ALTER TABLE product ENGINE=InnoDB DEFAULT CHARSET=gbk;
ALTER TABLE product CONVERT TO CHARSET gbk;

INSERT INTO student(num,name,sex,department,address) SELECT num,name,sex ,department,address FROM student0;
视图 **SELECT select_priv, create_view_priv from mysql.user WHERE user='root';**
SELECT user, select_priv, create_view_priv from mysql.user;
SELECT Drop_priv FROM mysql.user WHERE user='root';
SELECT * FROM information_schema.views \G
查看触发器 **SHOW TRIGGERS;** **SELECT * FROM information_schema. triggers;**
SELECT * FROM information_schema.triggers WHERE TRIGGER_NAME='dept_trig2' \G

USE information_schema
SELECT table_name, create_time FROM TABLES ORDER BY create_time;
SELECT table_name, create_time FROM TABLES ORDER BY table_name;
查看权限 **SELECT * FROM mysql.user \G** **SHOW CREATE FOR user** **SHOW CREATE FOR** 'test5'@'localhost';

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删是：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）　　DDL 语句：CREATE TABLE …

数据操作语言（Data Manipulation Language，简称为 DML）　　DML 语句：SELECT, INSERT, UPDATE ,DELETE

数据控制语言（Data Control Language，简称为 DCL）　　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．功能强大而且使用方便

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 平台下安装不配置 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 **gbk!!!**

"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("newPW") 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 平台下安装不配置 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) 字节数 1 无符号取值范围 0~255 有符号数的取值范围 –128~127
smallint(6) 字节数 2 无符号取值范围 0~65535 有符号数的取值范围 –32768~32767
mediumint(9) 字节数 3 无符号取值范围 0~16777215 有符号数的取值范围 –8388608~8388607
integer / int(11) 字节数 4 无符号取值范围 0~4294967295 有符号数的取值范围 –2147483648~214748
bigint(20) 字节数 8 无符号取值范围 0 ~18446744073709551615 有符号数的取值范围 –9223372036854775808~ 7
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，2^32，无符号最大能达到 4 亿多。
tinyint(4)相当于 bool 型

CREATE TABLE intdata2(a TINYINT(3) UNSIGNED ZEROFILL, b SMALLINT(5) UNSIGNED ZEROFILL,
c MEDIUMINT(8) UNSIGNED ZEROFILL, d INT(10) UNSIGNED ZEROFILL,
e BIGINT(20) UNSIGNED ZEROFILL);
DESC intdata1;

```
+-------+-------------------------------+------+-----+---------+-------+
| Field | Type                          | Null | Key | Default | Extra |
+-------+-------------------------------+------+-----+---------+-------+
| a     | tinyint(3) unsigned zerofill  | YES  |     | NULL    |       |
| b     | smallint(5) unsigned zerofill | YES  |     | NULL    |       |
```

```
| c     | mediumint(8) unsigned zerofill | YES  |     | NULL    |       |
| d     | int(10) unsigned zerofill      | YES  |     | NULL    |       |
| e     | bigint(20) unsigned zerofill   | YES  |     | NULL    |       |
+-------+--------------------------------+------+-----+---------+-------+
mysql> INSERT INTO intdata1 VALUES(1,1,1,1,1);
mysql> SELECT * FROM intdata1;
+------+-------+----------+------------+---------------------+
| a    | b     | c        | d          | e                   |
+------+-------+----------+------------+---------------------+
|  001 | 00001 | 00000001 | 0000000001 | 00000000000000000001 |
+------+-------+----------+------------+---------------------+
```

4.2、浮点数

float 字节数 4 负数取值范围 –3.402823466E+38~-1.175494351E-38 非负取值范围 0~3.402823466E+38

double 字节数 8 负数取值范围 1.7976931348623157E+308～-2.2250738585072014E-308 非负取值范围 0~ 0 和 2.2250738585072014E-308～1.7976931348623157E+308

decimal(m,d) 字节数 m+2 负数取值范围 同上 非负取值范围同上； 指总长 m 位，小数点后精确到 d 位 **精度高**

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 1 字节

(1) **$$$$ 1901-2155** ‘YYYY’ YYYY； (2) **$$ 1970-1999 2000-2069** ‘YY’； (3) **$$ 1970-1999 2001-2069 0-----0000** YY；

date 日期 4 字节 YYYY-MM-DD (1000-01-01 ~ 9999-12-31) **CURRENT_DATE NOW()**

(1) ‘YYYYMMDD’ ‘YYYY-MM-DD’ (1000-01-01 ~ 9999-12-31) 间隔字符 / . @ etc.

(2) ‘YYMMDD’ 同 YEAR----‘YY’； 间隔字符 / . @ etc.

(3) YYMMDD 同 YEAR----YY; 0---0000-00-00

time 时间 **HH:MM:SS--------(-838:59:59 -------- 838:59:59)** 3 字节 **CURRENT_TIME) (NOW()**

(1)’D HH:MM:SS’ D **0-34 /0-59 ’ HH:MM:SS’ ’HH:MM’ ’D HH:MM’ ’D HH’ ’SS’ HHMMSS ‘0’ 0(000000)**

datetime 日期时间 8 个字节 ‘YYYY-MM-DD HH:MM:SS’ NOW()

(1) ‘YYYY-MM-DD HH:MM:SS’ ‘YYYYMMDDHHMMSS’ 间隔字符 / . @ etc.

(2) ‘YY-MM-DD HH:MM:SS’ ‘YYMMDDHHMMSS’ 同 YEAR----‘YY’ 间隔字符 / . @ etc.

(3) YY-MM-DD HH:MM:SS YYMMDDHHMMSS 同 YEAR----YYYY YY 0---0000-00-00 00:00:00

timestamp 4 个字节 ‘YYYY-MM-DD HH:MM:SS’ **时间（时区）**，范围小(1970-01-01 ~ 2038-01-19 11:14:07)，支持时区

总体上同 datetime； **CURRENT_TIMESTAMP 或 输入 Null 或 无任何输入时， 都输入当前日期时间；**

INSERT INTO timestamp VALUES(); INSERT INTO timestamp VALUES(Null);

datetime 最通用，**year,date,time** 可以节省一些空间。

4.4、字符串 定义： **字符串类型（长度）**

char(m) 定长 0-255 字节 占用字节数——m 最后的空格 自动删除;

varchar(m) 不定长 0-65535 字节 占用字节数——m+1;(字符串结束标志占一位); 最后的空格 保存着;

tinytext 允许长度 0-255 字节 存储空间 值的长度+2 个字节

text 允许长度 0-65535 字节 存储空间 值的长度+2 个字节

mediumtext 允许长度 0-167772150 字节 存储空间 值的长度+3 个字节

longtext 允许长度 0-4294967295 字节 存储空间 值的长度+4 个字节

enum 枚举类型 MySQL 存入编号； 属性名 Enum(‘值 1’,’值 2’,…….’值 n’); 你<= 65535 存入编号 如果加上了 NOT NULL 属性，其默认值是列表的第一个元素，否则将允许插入 NULL（默认值）

set MySQL 存入编号； 属性名 SET(‘值 1’,’值 2’,…….’值 n’); n<=64 每个值可以是多个元素的组合，用 ‘,’ 隔开；顺序问题：INSERT INTO sets VALUES('C,B,D'); ------ | B,C,D |

enum,set 和其它库不兼容，可暂不用

4.5、二进制

binary(m) 字节数为 M，允许长度为 0-M 的定长二进制字符串 不足最大长度的空间由‘\0’补全；

varbinary(m) 字节数为值长度+1，允许长度为 0-M 的变长二进制字符串

bit(m) **M 位二进制数据，M<=64 （值：0-2^64-1）**
tinyblob **可变长二进制数据，最多 255 个字节**
blob **可变长二进制数据，最多(2^16+1)个字节**
mediumblob **可变长二进制数据，最多(2^24+1)个字节**
longblob **可变长二进制数据，最多(2^32+1)个字节**
Blob 可以用来保存图片；
SHOW WARNINGS;
如果要精确到小数点后 10 位，要选择 Double 类型；
浮点数、定点数会对其达不到精度的数值四舍五入；定点数默认整数位为 10，小数位为 0
Enum 类型只能从成员中选择一个，SET 类型可以选择多个，，适合‘爱好’等字段；
MySQL 中什么数据类型能够储存路径？——CHAR VARCHAR TEXT 但要过滤掉‘\’,用’\\’或’\’代替；
MySQL 中如何使用布尔类型？——BOOL(EAN) TINYINT(1)
MySQL 中如何存储 JPG 图片和 MP3 音乐？——BLOB 类型，或存路径；

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——表的类型； 插入式的存储引擎的概念，是 MySQL 的特点；
show engines \G mysql 支持的所有的 engine
(Engine: MyISAM 名称 Support: YES 支持 Comment: Default engine as of MySQL 3.23 with great performance 描述 Transactions: NO 不支持事务 XA: NO 不支持分布式交易处理的规范 Savepoints: NO 不支持保存点)
show variables like '%engine%'; 查看当前库的 **engine**
show variables like 'storage_engine'; 查看当前库默认的 **engine** **(my.ini: default- storage-engine= innodb)**
innodb 优点： 最常用，支持事务、提供崩溃修复能力和并发控制；回滚，自增（AUTO_INCREMENT），外键(FOREIGN KEY) 表结构存在.frm 文件中 数据和索引存在 innodb_data_home_dir 和 innodb_data_file_path 定义的表空间中
缺点： 读写效率稍差，占用空间大
myisam 曾经是默认 表存储成三个文件 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 ====== 不支持事务和开？？发 用于空间小、内存低，多插入、读取数据的情况；
memory **特殊的， 表存在硬盘中，数据存在内存中 处理快 但易丢失 表大小限制参数 max_rows max_hep_table_size** 演示系统 适于小巧的表；

表 5.1 存储引擎的对比

特 性	InnoDB	MyISAM	MEMORY
事务安全	支持	无	无
存储限制	64TB	有	有
空间使用	高	低	低
内存使用	高	低	高
插入数据的速度	低	高	高
对外键的支持	支持	无	无

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 (唯一、非空)

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 **子表的外键 --- 父表的主键;**

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(id INT PRIMARY KEY,
stu_id INT,
course_id INT,
CONSTRAINT c_fk FOREIGN KEY (stu_id, course_id) REFERENCES example2(stu_id, course_id));

ALTER TABLE 表名 1 ADD CONSTRAINT 外键名 FOREIGN KEY(属性名 1) REFERENCES 表名 2(属性名 2);

ALTER TABLE grade ADD CONSTRAINT grade_fk FOREIGN KEY(s_num) REFERENCES student(num);

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**

一个表只能有一个，而且为主键的一部分， 可以是任何整形;

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',
Math FLOAT DEFAULT 0,
Computer FLOAT DEFAULT 0);

复制表结构： **CREATE TABLE new_table SELECT * FROM old_table WHERE 0;**

复制表结构和数据： **CREATE TABLE new_table SELECT * FROM old_table;**

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(查看表的详细定义：字段名、字段数据类型、完整性约束条件默认存储引擎、字符编码)语句

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] 新名;** DESC example0; ALTER TABLE example0 RENAME 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、字段及数据类型的改、增、删以及改变位置

ALTER TABLE 表名 CHANGE 旧属性名 新属性名 新数据类型;

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 1 ADD CONSTRAINT 外键名 FOREIGN KEY(属性名 1) REFERENCES 表名 2(属性名 2);
ALTER TABLE 表名 DROP 属性名;
ALTER TABLE 表名 MODIFY 属性名 1 数据类型 FIRST|AFTER 属性名 2;
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
ALTER TABLE example1 CHANGE stu_name name VARCHAR(20); ALTER TABLE example1 CHANGE stu_sex sex INT(2);

ALTER TABLE user ADD phone VARCHAR(20); ALTER TABLE user ADD age INT(4) NOT NULL;
ALTER TABLE user ADD num INT(8) PRIMARY KEY FIRST;
ALTER TABLE user ADD address VARCHAR(30) NOT NULL AFTER phone;

ALTER TABLE grade ADD CONSTRAINT grade_fk FOREIGN KEY(s_num) REFERENCES student(num);

ALTER TABLE user DROP id;

ALTER TABLE user MODIFY name VARCHAR(30) FIRST; ALTER TABLE user MODIFY sex TINYINT(1) AFTER age;
6.3.7、更改表的存储引擎(表的存储类型) **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 ; (MyISAM 不支持外键)**
SHOW CREATE TABLE example3\G ALTER TABLE example3 DROP FOREIGN KEY c_fk; SHOW CREATE TABLE…..
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 CREATE 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 类型字段上，**只有 MyISAM 存储引擎支持之**。
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 3 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

注：建立主键的时候会自动建立一个唯一性索引， 即主键是一个索引！

7.2.1、创建表的时候创建索引 **CREATE TABLE 表名(属性名 数据类型[完整性约束条件], ……**

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
```

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。UNIQUE KEY `id` (`id`) UNIQUE KEY `index2_id` (`id`)

3、创建全文索引

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

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) );
INSERT INTO index5 VALUES(001,'123','woma'), (002,'a','man');
INSERT INTO index5 VALUES(003,'123','man'), (007,'a','woma');
EXPLAIN SELECT * FROM index5 WHERE name='a' AND sex='man' \G
EXPLAIN SELECT * FROM index5 WHERE name='a' \G
EXPLAIN SELECT * FROM index5 WHERE sex='man' \G
```

注： 必须使用索引中第一个字段！

6、创建空间索引

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

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(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(id);
3、创建全文索引 ALTER TABLE index15 ADD **FULLTEXT INDEX** index15_info(info);
4、创建单列索引 ALTER TABLE index16 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 **SPATIAL 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。安全！

8.1.1 视图的含义：在原有的表或者视图的基础上重新定义的虚拟表。数据库中只存放了视图的定义，而并没有存放视图中的数据。使用视图查询数据时，数据库系统会从原来的表中取出对应的数据。因此，视图中的数据是依赖于原来的表中的

数据的。一旦表中的数据发生改变，显示在视图中的数据也会发生改变。

8.1.2 视图的作用

1．使操作简单化： 起着类似于筛选的作用，简化操作，所见即所得；

2．增加数据的安全性：对用户没有用或者用户没有权限了解的信息 都直接屏蔽掉；

3．提高表的逻辑独立性： 修改了表的某些被引用列，视图随之做相应更改。

8.2、创建视图 创建在 n 张表上

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

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。

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；

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;

在多表上创建视图：

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;	DESC worker_view1;
SHOW TABLE STATUS LIKE ‘视图名’;	SHOW TABLE STATUS LIKE 'worker_view1' \G
SHOW CREATE VIEW 视图名;	SHOW CREATE VIEW worker_view1 \G
SELECT * FROM information_schema.views ;	SELECT * FROM information_schema.views \G

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	ALTER VIEW department_view2(department,name,sex,location)
CREATE OR REPLACE ALGORITHM=TEMPTABLE	AS SELECT d_name,worker.name,worker.sex,address
VIEW department_view1(department,function,location)	FROM department,worker WHERE department.d_id=worker.d_id
AS SELECT d_name,function,address FROM department;	WITH CHECK OPTION;

8.5、更新视图 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。超范围不能。

CREATE VIEW department_view3(name,function,address)

AS SELECT d_name,function,address FROM department WHERE d_id=1001;

UPDATE department_view3 SET name='科研部',function='新产品研发',address='3 号楼 5 层';

SELECT * FROM department_view3;

原则：尽量不要更新视图

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 SELECT Drop_priv FROM mysql.user WHERE user='root';

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

Drop VIEW worker_view1;

Drop VIEW IF EXISTS worker_view1, worker_view2;

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

CREATE TRIGGER dept_trig1 BEFORE INSERT
ON department FOR EACH ROW
INSERT INTO trigger_time VALUES(NOW());

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 &&
CREATE TRIGGER dept_trig2 AFTER DELETE
ON department FOR EACH ROW
BEGIN
INSERT INTO trigger_time VALUES('21:01:01');
INSERT INTO trigger_time VALUES('21:01:01');
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。

9.2、查看触发器

SHOW TRIGGERS ;

SELECT * FROM information_schema. triggers ;

SELECT * FROM information_schema.triggers WHERE TRIGGER_NAME='dept_trig2' \G

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;** DROP TRIGGER dept_trig1;

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

列出所有字段列出表的所有字段 | *

指定字段 属性列表”中列出所要查询的字段(注意顺序)

指定记录 WHERE 条件表达式 **=,<,>,!,** <> 及其组合

带 IN 关键字的查询 **[NOT]** IN (元素 1，元素 2，…，元素 n)

带 BETWEEN AND 的范围查询 **[NOT]** BETWEEN 取值 1 AND 取值 2

带 LIKE 的字符匹配查询 **[NOT]** LIKE '字符串' % _

查询空值 IS **[NOT]** NULL

带 AND 的多条件查询 条件表达式 1 **AND** 条件表达式 2 [… AND 条件表达式 n]

带 OR 的多条件查询 条件表达式 1 **OR** 条件表达式 2 […OR 条件表达式 n]

查询结果不重复 SELECT **DISTINCT** 属性名

给查询结果排序 ORDER BY 属性名 1 [ASC|DESC]，属性名 2 [ASC|DESC],…….，属性名 n [ASC|DESC]

分组查询 GROUP BY 属性名[HAVING 条件表达式] [WITHROLLUP]

1．单独使用 GROUP BY 关键字来分组 GROUP BY sex ----------2 条记录；

2．GROUP BY 关键字与 GROUP_CONCAT()函数（显示分组中具体信息）一起使用

SELECT sex, **GROUP_CONCAT**(name) FROM employee GROUP BY sex;

3．GROUP BY 关键字与集合函数一起使用 SELECT sex, COUNT(sex) FROM employee GROUP BY sex;

4．GROUP BY 与 HAVING 一起使用 SELECT sex, COUNT(sex) FROM e GROUP BY sex HAVING COUNT(sex)>=3;

5．按多个字段进行分组 GROUP BY id,sex;

6．GROUP BY 与 WITH ROLLUP 一起使用 WITH ROLLUP 一起使用，多一行，加统计

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

mysql> SELECT sex, GROUP_CONCAT(name) FROM employee GROUP BY sex WITH ROLLUP;

用 LIMIT 限制查询结果的数量 **LIMIT [初始位置,] 记录数** 不指定初始位置 LIMIT 2；指定初始位置 LIMIT **0,**2;

10.3、**使用集合函数查询** **COUNT() 个数/条数， AVG(), MAX(), MIN(), SUM()**

SELECT num,SUM(score) FROM grade WHERE num=1001;

SELECT num,SUM(score) FROM grade GROUP BY num;

SELECT AVG(age) FROM employee; SELECT course,AVG(score) FROM grade GROUP BY course;

SELECT num,AVG(score) FROM grade GROUP BY num;

10.4、**连接查询** 将两个或两个以上的表按某个条件（相同意义的字段）连接起来，从中选取需要的数据。

10.4.1、**内连接查询** select a.*, b.* from a, b where a.xid=b.xid

SELECT num, name, employee.d_id, age, sex, d_name, function

FROM employee, department

WHERE employee.d_id=department.d_id [AND age>20];

10.4.2、**外连接查询** 查询 n 个表。通过指定字段来进行连接，该字段取值不相等的记录也可以查询出来。

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 **-------左连接查询**

RIGHT JOIN 右表全记录，左表符合条件 **-------右连接查询**

SELECT num, name, employee.d_id, age, sex, d_name, function

FROM employee **LEFT JOIN** department

ON employee.d_id=department.d_id ;

10.5、**子查询** 将 1 查询语句嵌套在另 1 查询语句中。内层查询语句的查询结果为外层查询语句提供查询条件。实现多表之间的查询。子查询中可能包括 IN、NOT IN、ANY、ALL、EXISTS、NOT EXISTS 等关键字。子查询中还可能包含比较运算符，如“=”、“!=”、“>”和“<”等。

IN SELECT * FROM employee WHERE d_id **[NOT]** IN (SELECT d_id FROM department);

带比较运算符的 SELECT * FROM department WHERE d_id != (SELECT d_id FROM employee WHERE age=24);

EXISTS　表示存在，内层查询语句只返回一个真假值 (True|False)

SELECT * FROM employee　WHERE EXISTS　(SELECT d_name FROM department　WHERE d_id =1003);

SELECT * FROM employee　WHERE **age>24 AND** EXISTS　(SELECT d_name FROM department　WHERE...

ANY 满足其中任一条件(通常与比较运算符一起使用)

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

SELECT * FROM employee　WHERE d_id= ANY　(SELECT num FROM grade WHERE score>90);

ALL　满足所有条件　SELECT * FROM computer_stu　WHERE scrore >= ALL　(SELECT score FROM scholarship)

10.6、**合并查询结果**

SELECT　语句 1　　SELECT d_id FROM department

UNION | UNION ALL　　UNION　[ALL]

SELECT　语句 2　　SELECT d_id FROM employee **ORDER BY d_id;**

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

SELECT　语句 n　　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**

SELECT * FROM **department d**　WHERE d.d_id =1003;

SELECT **d_id AS department_id, d_name AS department_name** FROM department d　WHERE d.d_id =1003;

SELECT d.d_id AS department_id, d.d_name AS department_name, d.function FROM department d　WHERE…

10.8、使用正则表达式查询　　**属性名 REGEXP '匹配方式'**

用某种模式去匹配一类字符串的一个方式。查询能力比通配字符的查询能力更强大更灵活。可以应用于非常复杂查询。

表 10.2　正则表达式的模式字符

正则表达式的模式字符	含　义
^	匹配字符串开始的部分
$	匹配字符串结束的部分
.	代表字符串中的任意一个字符，包括回车和换行
[字符集合]	匹配“字符集合”中的任何一个字符
[^字符集合]	匹配除了“字符集合”以外的任何一个字符
S1 | S2 | S3	匹配S1、S2和S3中的任意一个字符串
*	代表多个该符号之前的字符，包括0和1个
+	代表多个该符号之前的字符，包括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，最多N次

^ 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

10.8.1 **查询以特定字符或字符串开头的记录**　使用字符“^”匹配　SELECT * FROM info WHERE name REGEXP '^L';

10.8.2 **查询以特定字符或字符串结尾的记录**　使用字符“$”匹配　SELECT * FROM info WHERE name REGEXP 'c$';

10.8.3 **用符号“.”来替代字符串中的任意一个字符**　用“.”来替代　查询以字母“L”开头，以字母“y”结尾，中间有两个任意字符的记录：SELECT * FROM info WHERE name REGEXP '^L..y$';

10.8.4 **匹配指定字符中的任意一个**　使用方括号（[]）可以将需要查询字符组成一个字符集。只要记录中包含方括号中的任意字符，该记录将会被查询出来。例如，通过“[abc]”可以查询包含 a、b、c 这三个字母中任何一个的记录。

SELECT * FROM info WHERE name REGEXP '[ceo]';　SELECT * FROM info WHERE name REGEXP '[0-9a-c]';

10.8.5 **匹配指定字符以外的字符**　使用“[^字符集合]”

查询包含 a 到 w 字母和数字以外的字符的记录：SELECT * FROM info WHERE name REGEXP '[^a-w0-9]';

10.8.6 **匹配指定字符串**　当表中的记录包含这个字符串时，就可以将该记录查询出来。多个字符串用“|”隔开。只要匹配这些字符串中的任意一个即可。　SELECT * FROM info WHERE name REGEXP 'ic|uc|ab';

SELECT * FROM info WHERE name REGEXP 'ic|uc　|ab';

10.8.7 **使用“*”和“+”来匹配多个字符** “*”和“+”都可以匹配多个该符号之前的字符。但是，“+”至少表示一个字符，而“*”可以表示零个字符。SELECT * FROM info WHERE name REGEXP 'a*c'; ….WHERE name REGEXP 'a+c';

10.8.8 **使用{M}或者{M,N}来指定字符串连续出现的次数** 正则表达式中，“字符串{M}”表示字符串连续出现 M 次；“字符串{M,N}”表示字符串联连续出现至少 M 次，最多 N 次。例如，“ab{2}”表示字符串“ab”连续出现两次。“ab{2,4}”表示字符串“ab”连续出现至少两次，最多四次。

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

第 11 章 插入、更新不删除数据

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 [(属性 1,属性 2, …,属性 n)] VALUES(值 1,值 2, …, 值 n);**

2、INSERT 语句中列出所有字段

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 m) VALUES(值 1,值 2, …, 值 m);**

11.1.3、同时插入多条数据 **INSERT INTO 表名[(属性列表)] VALUES(取值列表 1),(取值列表 2)… ,(取值列表 n) ;**

MySQL 特有的 INSERT INTO product(id,name,company) VALUES (1009,'护发 1 号','北京护发素厂'), (1010,'护发 2 号','北京护发素厂'), (1011,'护发 3 号','北京护发素厂');

11.1.4、查询结果插入表 **INSERT INTO 表名 1 (属性列表 1) SELECT 属性列表 2 FROM 表名 2 WHERE 条件表达式;**

INSERT INTO product(id,name,function,company,address) SELECT id,name,function,company,address FROM medicine;

11.2、更新数据 UPDATE 表名

SET 属性名 1=取值 1, 属性名 2=取值 2, …, 属性名 n=取值 n WHERE 条件表达式;

11.3、删除数据 **DELETE FROM 表名 [WHERE 条件表达式] ;** 删除所有记录 **DELETE FROM 表名**

DELETE FROM product WHERE id=1050;

第 12 章 MySQL 运算符

12.1、**算术运算符**

SELECT a, a+5+2, a-5-2, a*5*2 FROM t1;

SELECT a, a/3, a DIV 3, a*3, MOD(a,3) FROM t1;

SELECT 5/0, 5 DIV 0, 5%0, MOD(5,0) FROM t1; ->NULL

符号	表达式的形式	作用
+	x1+x2+…+xn	加法运算
-	x1-x2-…-xn	减法运算
*	x1*x2*…*xn	乘法运算
/	x1/x2	除法运算，返回 x1 除以 x2 的商
DIV	x1	除法运算，返回商。同“/”
%	x1%x2	求余运算，返回 x1 除以 x2 的余数
MOD	MOD(x1,x2)	求余运算，返回余数。同“%”

12.2、**比较运算符** 可以用于 Null 的运算符—— <=> NULL 结果只有 0 和 1

1. 运算符“=” SELECT a,a=24,a=20 FROM t1; -- 24, 1, 0 SELECT 'b'='b','b'='c',NULL=NULL FROM t1; ->1, 0, NULL

2. “<>”和“!=”

SELECT a,a<>23,a!=23,a!=24,a!=NULL FROM t1;-- 24, 1, 1, 0, NULL SELECT 'b'<>'b','b'!='c' -> 0,1

3. “<=>” SELECT 'b'<=>'b','b'<=>'c', NULL<=>NULL; -> 1, 0, 1

4. “>” 5. “>=”

6. “<” 7. “<=”

SELECT 'b'>'c','b'>'bb', NULL>NULL; -> 0, 0, NULL

8. “IS [NOT] NULL”

表 12.2 MySQL的比较运算符

符 号	表达式的形式	作 用
=	x1=x2	判断 x1 是否等于 x2
<>或!=	x1<>x2 或 x1!=x2	判断 x1 是否不等于 x2
<=>	x1<=>x2	判断 x1 是否等于 x2
>	x1>x2	判断 x1 是否大于 x2
>=	x1>=x2	判断 x1 是否大于等于 x2
<	x1<x2	判断 x1 是否小于 x2
<=	x1<=x2	判断 x1 是否小于等于 x2
IS NULL	x1 IS NULL	判断 x1 是否等于 NULL
IS NOT NULL	x1 IS NOT NULL	判断 x1 是否不等于 NULL
BETWEEN AND	x1 BETWEEN m AND n	判断 x1 的取值是否落在 m 和 n 之间
IN	x1 IN(值 1,值 2, …,值 n)	判断 x1 的取值是否值 1 到值 n 中的一个
LIKE	x1 LIKE 表达式	判断 x1 是否与表达式匹配
REGEXP	x1 REGEXP 正则表达式	判断 x1 是否与正则表达式匹配

SELECT NULL IS NULL, NULL IS NOT NULL; -> 1, 0

9. “BETWEEN AND” SELECT 'b' BETWEEN 'a' AND 'd','z' BETWEEN 'a' AND 'd'; -> 1, 0

10. “IN” SELECT 'a' IN('a','c','e'),'a' IN('b','c','d') FROM t1; -> 1, 0

11. “LIKE” SELECT s,s LIKE 'Beijing',s LIKE '____jing',**s LIKE 'b%'**,s LIKE 'z%' FROM t2; -> Beijing, 1, 0, **1**, 0

12. “REGEXP”正则表达式 SELECT s,s REGEXP '^B',**s REGEXP 'g$'**,s REGEXP 'y' FROM t2; -> Beijing, 1, **1**, 0(包含 y)

12.3、**逻辑运算符** 结果只有 0 和 1

1. 与运算&& AND **有 0->0，无 0 有 NULL->NULL，其他->1；** SELECT -1&&2&&3,**0&&NULL**,3&&NULL; ->1, **0**, Null

2. 或运算 || OR **All 操作数中 any 不是 0 且不是 NULL 则->1；只包含 0 和 NULL 则->NULL；如果只有 0 时则->0；**

3||NULL ->1 0||NULL ->NULL NULL||NULL ->NULL 0||0->0；

3. 非运算 ！ **!0 ->1 !x->0 !NULL->NULL；** SELECT !1, !0.3,**!-3**,!0,!NULL; -> 0, 0, **0**, 1, NULL

4. 异或运算 XOR **有 NULL->NULL，有 0 和非 0->1，若都是 0 或都是非 0->0**

SELECT NULL XOR 1, NULL XOR 0, 3 XOR 1, **1 XOR 0**, 0 XOR 0, 3 XOR 2 XOR 0 X ->NULL, NULL, 0, **1**, 0, 1

12.4、**位运算符 将十进制数据(CONV())转化为二进制->位计算->转化回十进制**

1. 按位与 & **SELECT 5&6,5&6&7; ->4, 4**

2. 按位或 | **SELECT 5|6,5|6|7;->7, 7**

3. 按位取反 ~ **SELECT ~1->** 18446744073709551614 **SELECT BIN(~1); ->** 1111….1111111111111111111110(共 64 位)

4. 按位异或 ^ **SELECT 5^6;-> 3**

5. 按位左移 << 和 按位右移 >> （右边补 0） **SELECT 5<<2,5>>2; ->20, 1**

表 12.5 MySQL运算符的优先级

优先级	运 算 符	优先级	运 算 符
1	!	8	|
2	~	9	=,<=>,<,<=,>,>=,!=,<>,IN,IS NULL,LIKE,REGEXP
3	^	10	BETWEEN AND,CASE,WHEN,THEN,ELSE
4	*,/,DIV,%,MOD	11	NOT
5	+,–	12	&&,AND
6	>>,<<	13	||,OR,XOR
7	&	14	:=

读者可以根据上表的内容来参考运算符的优先级。但是，实际使用中更多的使用“()”来将优先计算的内容括起来。这样用起来更加简单，而且可读性更强。

最小的是赋值语句

第 13 章 MySQL 函数

13.1、数学函数 **ABS(x)**用来求绝对值；**PI()**用来返回圆周率；**SQRT(x)**用来求平方根；**MOD(x,y)**用来求余数；**CEIL(x)和 CEILING(x)**这两个函数返回大于或等于 x 的最小整数；**FLOOR(x)**函数返回小于或等于 x 的最大整数；**RAND()和 RAND(x)**这两个函数都是返回 0～1 的随机数。但是 RAND()返回的数是完全随机的，而 RAND(x)函数的 x 相同时返回的值相同；**ROUND(x)**函数返回离 x 最近的整数，也就是对 x 进行四舍五入处理；**ROUND(x,y)**函数返回 x 保留到小数点后 y 位的值，截断时需要进行四舍五入处理；**TRUNCATE(x,y)**函数返回 x 保留到小数点后 y 位的值；**SIGN(x)**函数返回 x 的符号，x 是负数、0、正数分别返回-1、0、1；**POW(x,y)**和 **POWER(x,y)**这两个函数计算 x 的 y 次方；EXP(x)函数计算 e 的 x 次方；**LOG(x)**函数计算 x 的自然对数；**LOG10(x)**函数计算以 10 为底的对数；**RADIANS(x)**函数将角度转换为弧度；**DEGREES(x)**函数将弧度转换为角度；**SIN(x)**函数用来求正弦值，其中 x 是弧度；**ASIN(x)**函数用来求反正弦值。**ASIN(x)**中 x 的取值必须在-1 到 1 之间。否则返回的结果将会是 **NULL**；**TAN(x)**函数用来求正切值，其中 x 是弧度；ATAN(x)和 **ATAN2(x)**用来求反正切值；**COT(x)**函数用来求余切值。TAN(x)与 ATAN(x)、ATAN2(x)互为反函数.

SELECT RAND(),**RAND(),** RAND(2), RAND(2); ->, , , , , ,

SELECT POW(3,2), POWER(3,2), EXP(2); ->9, 9, 7.389

SELECT ABS(0.5),**ABS(-0.5),** PI();->0.5, **0.5,** 3.141593

SELECT SQRT(16),**SQRT(2),** MOD(5,2);

SELECT ROUND(2.3),**ROUND(2.5)**,ROUND(2.53,1),**ROUND(2.55,1),** TRUNCATE (2.53,1), TRUNCATE(2.59,1); ->2, **3**, 2.5, **2.6**,2.5, 2.5 SELECT SIN(0.535), ASIN(0.5);->, , , , , ,

SELECT ACOS(2); SELECT COT(1),1/TAN(1); SELECT ACOS(2); SELECT TAN(0. 785),ATAN(1), ATAN2(1);

SELECT SIN(0.535), ASIN(0.5); SELECT COS(1.0471975511), ACOS(0.5); SELECT SIN(0.535), ASIN(0.5);

SELECT CEIL(2.3),**CEIL(-2.3)**,CEILING(2.3),CEILING**(-2.3)**, FLOOR(2.3), FLOOR(-2.3); ->3, **-2**, 3, **-2**, 2 ,-3

表 13.1 MySQL的数学函数

函 数	作 用
ABS(x)	返回 x 的绝对值
CEIL(x),CEILING(x)	返回大于或等于 x 的最小整数
FLOOR(x)	返回小于或等于 x 的最大整数
RAND()	返回 0～1 的随机数
RAND(x)	返回 0～1 的随机数，x 值相同时返回的随机数相同
SIGN(x)	返回 x 的符号，x 是负数、0、正数分别返回–1、0 和 1
PI()	返回圆周率（3.141593）
TRUNCATE(x,y)	返回数值 x 保留到小数点后 y 位的值
ROUND(x)	返回离 x 最近的整数
ROUND(x,y)	保留 x 小数点后 y 位的值，但截断时要进行四舍五入
POW(x,y),POWER(x,y)	返回 x 的 y 次方（x^y）
SQRT(x)	返回 x 的平方根
EXP(x)	返回 e 的 x 次方（e^x）
MOD(x,y)	返回 x 除以 y 以后的余数
LOG(x)	返回自然对数（以 e 为底的对数）
LOG10(x)	返回以 10 为底的对数
RADIANS(x)	将角度转换为弧度
DEGREES(x)	将弧度转换为角度
SIN(x)	求正弦值
ASIN(x)	求反正弦值
COS(x)	求余弦值
ACOS(x)	求反余弦值
TAN(x)	求正切值
ATAN(x),ATAN2(x,y)	求反正切值
COT(x)	求余切值

SELECT SQRT(16),SQRT(2), MOD(5,2);->4, 1.414213562, 1 SELECT SIGN(-2),SIGN(0), SIGN(2);->, -1, 0, 1

SELECT LOG(7.3898568989), LOG10(100);->, , , , , , SELECT RADIANS(180), DEGREES(3.1415926535897);

13.2、字符串函数 重点 CONCAT(S1,S2…)

表 13.2　MySQL的字符串函数

函　数	作　用
CHAR_LENGTH(s)	返回字符串 s 的字符数
LENGTH(s)	返回字符串 s 的长度
CONCAT(s1,s2,…)	将字符串 s1,s2 等多个字符串合并为一个字符串
CONCAT_WS(x,s1,s2,…)	同 CONCAT(s1,s2,…)函数，但是每个字符串直接要加上 x
INSERT(s1,x,len,s2)	将字符串 s2 替换 s1 的 x 位置开始长度为 len 的字符串
UPPER(s),UCASE(s)	将字符串 s 的所有字母都变成大写字母
LOWER(s),LCASE(s)	将字符串 s 的所有字母都变成小写字母
LEFT(s,n)	返回字符串 s 的前 n 个字符
RIGHT(s,n)	返回字符串 s 的后 n 个字符
LPAD(s1,len,s2)	字符串 s2 来填充 s1 的开始处，使字符串长度达到 len
RPAD(s1,len,s2)	字符串 s2 来填充 s1 的结尾处，使字符串长度达到 len
LTRIM(s)	去掉字符串 s 开始处的空格
RTRIM(s)	去掉字符串 s 结尾处的空格
TRIM(s)	去掉字符串 s 开始处和结尾处的空格
TRIM(s1 FROM s)	去掉字符串 s 中开始处和结尾处的字符串 s1
REPEAT(s,n)	将字符串 s 重复 n 次
SPACE(n)	返回 n 个空格
REPLACE(s,s1,s2)	用字符串 s2 替代字符串 s 中的字符串 s1
STRCMP(s1,s2)	比较字符串 s1 和 s2
SUBSTRING(s,n,len)	获取从字符串 s 中的第 n 个位置开始长度为 len 的字符串
MID(s,n,len)	同 SUBSTRING(s,n,len)
LOCATE(s1,s),POSITION(s1 IN s)	从字符串 s 中获取 s1 的开始位置
INSTR(s,s1)	从字符串 s 中获取 s1 的开始位置
REVERSE(s)	将字符串 s 的顺序反过来
ELT(n,s1,s2,…)	返回第 n 个字符串
EXPORT_SET(x,s1,s2)	
FIELD(s,s1,s2,…)	返回第一个与字符串 s 匹配的字符串的位置
FIND_IN_SET(s1,s2)	返回在字符串 s2 中与 s1 匹配的字符串的位置
MAKE_SET(x,s1,s2,…)	按 x 的二进制数从 s1,s2,…,sn 中选取字符串

CHAR_LENGTH(s)计算字符数; LENGTH(s)计算长度 SELECT s, CHAR_LENGTH(s), LENGTH(s) FROM t2; -> Beijing,7,7

合并 CONCAT(s1,s2,…)　CONCAT_WS(x,s1,s2,…) SELECT CONCAT('bei','ji','ng'),CONCAT_WS('-','bei','ji','ng'); -> Beijing, bei-ji-ng

替换字符串的函数 INSERT(s1,x,len,s2)　SELECT s,INSERT(s,4,4,'fang') FROM t2;-> Beijing , Beifang

字母大小写 SELECT UPPER('mysql'),UCASE('mysql'),LOWER('MYSQL'),LCASE('MYSQL');->MYSQL,MYSQL,mysql,mysql

获取指定长度的字符串的函数 LEFT(s,n)和 RIGHT(s,n)　SELECT s, LEFT(s,3), RIGHT(s,4) FROM t2;-> Beijing, Bei, jing

填充 LPAD(s1,len,s2)和 RPAD(s1,len,s2) SELECT LPAD(s,10,'+-'), RPAD(s,10,'+-') FROM t2;-> +-+Beijing, Beijing+-+, Beijing

删除空格 LTRIM(s)　SELECT CONCAT('+',' me ','+'),CONCAT('+',LTRIM(' me '),'+');-> + me +, +me +

RTRIM(s)和 TRIM(s)　SELECT CONCAT('+',RTRIM(' me '),'+'),CONCAT('+',TRIM(' me '),'+');-> + me+, +me+

删除指定字符串的函数 TRIM(s1 FROM s)　SELECT TRIM('ab' FROM 'ababddddddabab');-> ddddddd

SELECT TRIM('ab' FROM 'ababaddddddaabab');-> addddddda

重复生成字符串的函数 REPEAT(s,n)　SELECT REPEAT('mysql-',2);-> mysql-mysql-

空格 SPACE(n)和替换 REPLACE(s,s1,s2) CONCAT('+',SPACE(2),'+'),REPLACE('MYSQL','SQL','HOOK');->+ +, MYHOOK

比较字符串大小的函数 STRCMP(s1,s2)　SELECT STRCMP('abc','abb'),STRCMP('abc','abc'),STRCMP('abc','abd');-> 1, 0, -1

获取子串函数 SUBSTRING(s,n,len)和 MID(s,n,len) SELECT s,SUBSTRING(s,4,3),MID(s,4,3) FROM t2;-> Beijing, jin, jin

匹配子串开始位置的函数 LOCATE(s1,s)、POSITION(s1 IN s)和 INSTR(s,s1)

SELECT s,LOCATE('jin',s),POSITION('jin'IN s),INSTR(s,'jin') FROM t2;-> Beijing, 4, 4, 4

字符串逆序的函数 REVERSE(s) SELECT s, REVERSE(s) FROM t2;-> Beijing, gnijieB

返回指定位置的字符串的函数 ELT(n,s1,s2,…) 返回第 n 个字符串 SELECT ELT(2,'me','my','he','she','mine');-> my

返回指定字符串位置的函数 FIELD(s,s1,s2,…) 返回 s 匹配字符串的位置 SELECT FIELD('she','me','my','he','she','mine');-> 4

返回子串位置的函数 FIND_IN_SET(s1,s2) SELECT FIND_IN_SET('she','me,my,he,she,mine');-> 4

选取字符串的函数 MAKE_SET(x,s1,s2,…) 按 x 的二进制数从 s1,s2,…,sn 中选取字符串

SELECT MAKE_SET(11,'a','b','c','d'),MAKE_SET(7,'a','b','c','d'); -> a, b, d a, b, c

13.3、日期和时间函数重点

表 13.3 MySQL的日期和时间函数

函 数	作 用
CURDATE(),CURRENT_DATE()	返回当前日期
CURTIME(),CURRENT_TIME()	返回当前时间
NOW(),CURRENT_TIMESTAMP(), LOCALTIME(),SYSDATE(), LOCALTIMESTAMP()	返回当前日期和时间
UNIX_TIMESTAMP()	以 UNIX 时间戳的形式返回当前时间
UNIX_TIMESTAMP(d)	将时间 d 以 UNIX 时间戳的形式返回
FROM_UNIXTIME(d)	把 UNIX 时间戳的时间转换为普通格式的时间
UTC_DATE()	返回 UTC（Universal Coordinated Time，国际协调时间）日期
UTC_TIME()	返回 UTC 时间
MONTH(d)	返回日期 d 中的月份值，范围是 1～12
MONTHNAME(d)	返回日期 d 中的月份名称，如 January,February
DAYNAME(d)	返回日期 d 是星期几，如 Monday,Tuesday 等
DAYOFWEEK(d)	返回日期 d 是星期几，1 表示星期日，2 表示星期一等
WEEKDAY(d)	返回日期 d 是星期几，0 表示星期一，1 表示星期二等
WEEK(d)	计算日期 d 是本年的第几个星期，范围是 0～53
WEEKOFYEAR(d)	计算日期 d 是本年的第几个星期，范围是 1～53
DAYOFYEAR(d)	计算日期 d 是本年的第几天
DAYOFMONTH(d)	计算日期 d 是本月的第几天
YEAR(d)	返回日期 d 中的年份值
QUARTER(d)	返回日期 d 是第几季度，范围是 1～4
HOUR(t)	返回时间 t 中的小时值
MINUTE(t)	返回时间 t 中的分钟值
SECOND(t)	返回时间 t 中的秒钟值
EXTRACT(type FROM d)	从日期 d 中获取指定的值，type 指定返回的值，如 YEAR,HOUR 等
TIME_TO_SEC(t)	将时间 t 转换为秒
SEC_TO_TIME(s)	将以秒为单位的时间 s 转换为时分秒的格式
TO_DAYS(d)	计算日期 d～0000 年 1 月 1 日的天数
FROM_DAYS(n)	计算从 0000 年 1 月 1 日开始 n 天后的日期
DATEDIFF(d1,d2)	计算日期 d1～d2 之间相隔的天数
ADDDATE(d,n)	计算起始日期 d 加上 n 天的日期
ADDDATE(d,INTERVAL expr type)	计算起始日期 d 加上一个时间段后的日期
DATE_ADD(d,INTERVAL expr type)	同 ADDDATE(d,INTERVAL n type)
SUBDATE(d,n)	计算起始日期 d 减去 n 天的日期
SUBDATE(d,INTERVAL expr type)	计算起始日期 d 减去一个时间段后的日期
ADDTIME(t,n)	计算起始时间 t 加上 n 秒的时间
SUBTIME(t,n)	计算起始时间 t 减去 n 秒的时间
DATE_FORMAT(d,f)	按照表达式 f 的要求显示日期 d
TIME_FORMAT(t,f)	按照表达式 f 的要求显示时间 t
GET_FORMAT(type,s)	根据字符串 s 获取 type 类型数据的显示格式

获取当前日期、时间的函数 CURDATE()和 CURRENT_DATE()　CURTIME()和 CURRENT_TIME() ->
获取当前日期和时间的函数 NOW()、CURRENT_TIMESTAMP()、LOCALTIME()和 SYSDATE() ->
UNIX 时间戳函数 UNIX_TIMESTAMP()函数以 UNIX 时间戳的形式返回当前时间;UNIX_TIMESTAMP(d)函数将时间 d 以 UNIX 时间戳的形式返回；FROM_UNIXTIME(d)函数把 UNIX 时间戳的时间转换为普通格式的时间。UNIX_TIMESTAMP(d)函数和 FROM_UNIXTIME(d)互为反函数 ->
UTC_DATE()返回 UTC 日期；UTC_TIME()返回 UTC 时间。其中，UTC 是 Universal Coordinated Time 国际协调时间 ->
获取月份的函数 MONTH(d) 取值范围是 1~12；和 MONTHNAME(d) 英文名称，如 January,February 等 d 日期[时间] ->
DAYNAME(d)返回日期 d 是星期几英文名，如 Monday,Tuesday 等；DAYOFWEEK(d)日期 d 是星期几，1 表示星期日，2 表示星期一，依次类推；WEEKDAY(d)返回日期 d 是星期几，0 表示星期一，1 表示星期二， d 日期[时间]。->
WEEK(d)函数和 WEEKOFYEAR(d)计算日期 d 是本年的第几个星期。返回值的范围是 1~53 ->
DAYOFYEAR(d)函数日期 d 是本年的第几天；DAYOFMONTH(d)函数返回计算日期 d 是本月的第几天 ->
YEAR(d)函数返回日期 d 中的年份值；QUARTER(d)函数返回日期 d 是本年第几季度，值的范围是 1~4；HOUR(t)函数返回时间 t 中的小时值；MINUTE(t)函数返回时间 t 中的分钟值；SECOND(t)函数返回时间 t 中的秒钟值 ->
EXTRACT(type FROM d)函数从日期 d 中获取指定的值。这个值是什么由 type 的值决定。type 的取值可以是 YEAR、MONTH、DAY、HOUR、MINUTE、SECOND。如果 type 的值是 YEAR，结果返回年份值；MONTH 返回月份值；DAY 返回是几号；HOUR 返回小时值；MINUTE 返回分钟值；SECOND 返回秒钟值->
TIME_TO_SEC(t)函数将时间 t 转换为以秒为单位的时间；SEC_TO_TIME(s)函数将以秒为单位的时间 s 转换为时分秒的格式。TIME_TO_SEC(t)和 SEC_TO_TIME(s)互为反函数 ->
计算日期和时间的函数 1. TO_DAYS(d)、FROM_DAYS(n)和 DATEDIFF(d1,d2)函数 2. ADDDATE(d,n)、SUBDATE(d,n)、ADDTIME(t,n)和 SUBTIME(t,n)函数 3. ADDDATE(d,INTERVAL expr type)和 DATE_ADD(d,INTERVAL expr type)

表 13.4　MySQL的日期间隔类型

类　型	含　义	expr 表达式的形式
YEAR	年	YY
MONTH	月	MM
DAY	日	DD
HOUR	时	hh
MINUTE	分	mm
SECOND	秒	ss
YEAR_MONTH	年和月	YY 和 MM 之间用任意符号隔开
DAY_HOUR	日和小时	DD 和 hh 之间用任意符号隔开
DAY_MINUTE	日和分钟	DD 和 mm 之间用任意符号隔开
DAY_SECOND	日和秒钟	DD 和 ss 之间用任意符号隔开
HOUR_MINUTE	时和分	hh 和 mm 之间用任意符号隔开
HOUR_SECOND	时和秒	hh 和 ss 之间用任意符号隔开
MINUTE_SECOND	分和秒	mm 和 ss 之间用任意符号隔开

将日期和时间格式化的函数 1．DATE_FORMAT(d,f)　2．TIME_FORMATE(t,f)　3．GET_FORMAT(type,s)->

表 13.4　MySQL的日期间隔类型

符号	含　义	取 值 示 例
%Y	以 4 位数字表示年份	2008,2009 等
%y	以 2 位数字表示年份	98,99 等
%m	以 2 位数字表示月份	01,02,…,12
%c	以数字表示月份	1,2,…,12
%M	月份的英文名	January,February,…,December
%b	月份的英文缩写	Jan,Feb,…,Dec
%U	表示星期数，其中 Sunday 是星期的第一天	00～52
%u	表示星期数，其中 Monday 是星期的第一天	00～52
%j	以 3 位数字表示年中的天数	001～366
%d	以 2 位数字表示月中的几号	01,02,…,31
%e	以数字表示月中的几号	1,2,…,31
%D	以英文后缀表示月中的几号	1st,2nd,…,
%w	以数字的形式表示星期几	0 表示 Sunday,1 表示 Monday,…
%W	星期几的英文名	Monday,…,Sunday
%a	星期几的英文缩写	Mon,…,Sun
%T	24 小时制的时间形式	00:00:00～23:59:59
%r	12 小时制的时间形式	12:00:00AM～11:59:59PM
%p	上午（AM）或下午（PM）	AM 或 PM
%k	以数字表示 24 小时	0,1,…,23
%l	以数字表示 12 小时	1,2,…,12
%H	以 2 位数表示 24 小时	00,01,…,23
%h,%I	以 2 位数表示 12 小时	01,02,…,12
%i	以 2 位数表示分	00,01,…,59
%S,%s	以 2 位数表示时	00,01,…,59
%%	标识符%	%

SELECT CURDATE(),CURRENT_DATE(),CURTIME(),CURRENT_TIME();->| 2014-03-13 | ---| 21:48:54| --- |

SELECT NOW(),CURRENT_TIMESTAMP(),LOCALTIME(),SYSDATE();->| 2014-03-13 21:50:34

SELECT NOW(),UNIX_TIMESTAMP(),UNIX_TIMESTAMP(NOW());->| 2014-03-13 21:53:42 | 1394718822 |1394718822 |

SELECT CURDATE(),UTC_DATE(),CURTIME(),UTC_TIME();->| 2014-03-13 | 2014-03-13 | 21:59:20 | 13:59:20|

SELECT MONTH(NOW()),MONTHNAME(NOW());->|3 | March |

SELECT DAYNAME(NOW()),DAYOFWEEK(NOW()),WEEKDAY(NOW());->| Thursday | 5 |3 |

SELECT WEEK(NOW()),WEEKOFYEAR(NOW());-> | 10 | 11 |

SELECT WEEK(CURDATE()),WEEKOFYEAR(CURDATE());-> | 10 | 11 |

SELECT DAYOFYEAR(CURDATE()),DAYOFMONTH(CURDATE());-> |72|13 |

SELECT YEAR(CURDATE()),QUARTER(CURDATE());->|2014 | 1 |

SELECT HOUR(NOW()),MINUTE(NOW()),SECOND(NOW());-> |22 | 8 |57 |

SELECT EXTRACT(MONTH FROM NOW()),EXTRACT(MINUTE FROM NOW());-> |3 | 10 |

SELECT TIME_TO_SEC(NOW()),SEC_TO_TIME(58559); -> |80038 | 16:15:59

SELECT TO_DAYS(NOW()),FROM_DAYS(734070),DATEDIFF(NOW(),'1984-09-02'); -> | 735670 | 2009-10-25 | 10784 |

SELECT ADDDATE(NOW(),3),SUBDATE(NOW(),3),ADDTIME(NOW(),5),SUBTIME(NOW(),5); -> 2014-03-16 22:21:36 | 2014-03-10 22:21:36 | 2014-03-13 22:21:41 | 2014-03-13 22:21:31 |

SELECT ADDDATE(NOW(),INTERVAL '1 1' YEAR_MONTH); ->| 2015-04-13 22:27:31

SELECT NOW(),ADDDATE(NOW(),INTERVAL '-1 -1' YEAR_MONTH); ->| 2014-03-13 22:28:47 | 2013-02-13 22:28:47

SELECT DATE_FORMAT(NOW(),'%b %D %Y'); ->| Mar 13th 2014|

SELECT DATE_FORMAT(NOW(),'%j %W %Y'); ->| 072 Thursday 2014

SELECT DATE_FORMAT(NOW(),'%r'); ->| 10:35:51 PM |

SELECT GET_FORMAT(DATETIME,'ISO'),GET_FORMAT(DATE,'EUR'),GET_FORMAT(TIME,'USA'); ->| %Y-%m-%d %H:%i:%s| %d.%m.%Y | %h:%i:%s %p

SELECT DATE_FORMAT(NOW(),GET_FORMAT(DATETIME,'INTERNAL')); ->| 20140313224438

SELECT DATE_FORMAT(NOW(),GET_FORMAT(DATETIME,'EUR')); ->| 2014-03-13 22.45.21|

SELECT DATE_FORMAT(NOW(),GET_FORMAT(DATE,'EUR')); ->| 13.03.2014

SELECT TIME_FORMAT(NOW(),GET_FORMAT(TIME,'USA')); ->| 10:46:47 PM |

13.4、条件判断函数

13.5.1 IF(expr,v1,v2)函数，如果表达式 expr 成立，返回结果 v1；否则，返回结果 v2。

SELECT id,grade,IF(grade>=60,'PASS','FAIL') from t6; SELECT 59,IF(59>=60,'PASS','FAIL');

13.5.2 IFNULL(v1,v2)函数，如果 v1 的不为空，就显示 v1 的值；否则就显示 v2 的值。

SELECT id,IFNULL(grade, 'NO GRADE') FROM t6; SELECT NULL,IFNULL(NULL,'FAIL');

13.5.3 CASE 函数

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

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

-> SELECT id,grade,
CASE WHEN grade>60 THEN 'GOOD' WHEN grade=60 THEN 'PASS' ELSE 'FAIL'END level
FROM t6;

-> SELECT id,grade,
CASE grade WHEN 98 THEN 'GOOD' WHEN 60 THEN 'PASS' ELSE 'FAIL'END level
FROM t6;

13.5、系统信息函数 查询 MySQL 数据库的系统信息。

13.5.1 获取 MySQL 版本号、连接数、数据库名的函数

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:

13.5.2 获取当前用户名称函数 **USER()、SYSTEM_USER()、SESSION_USER()、CURRENT_USER()和 CURRENT_USER**

13.5.3 获取字符串的字符集和排序方式的函数

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:

13.5.4 获取最后一个自动生成的 ID 值的函数 **LAST_INSERT_ID()函数返回最后生成的 AUTO_INCREMENT 值:**

SELECT VERSION(), CONNECTION_ID();->| 5.1.40-community | 3 |

SELECT DATABASE(),SCHEMA();-> | example | example |

SELECT USER(),SYSTEM_USER(),SESSION_USER(),CURRENT_USER(),CURRENT_USER; -> | root@localhost | root@localhost | root@localhost | root@localhost | root@local

SELECT CHARSET('aa'),COLLATION('aa'); ->| gbk | gbk_chinese_ci |

CREATE TABLE t8(id INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT); ->

SELECT LAST_INSERT_ID();-> | 3 |

13.6、加密函数 对数据进行加密

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。

SELECT PASSWORD('abcd');

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。 SELECT MD5('abcd');

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，字符串 pswd_str 应该与加密时的字符串 pswd_str 是相同的。 SELECT DECODE(ENCODE('abcd','aa'),'aa');

13.7、格式化函数

13.7.1 **格式化函数 FORMAT(x,n)** **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。**

例如 FORMAT(2.356,2)返回的结果将会是 2.36；FORMAT(2.353,2)返回的结果将会是 2.35。

13.7.2 **不同进制的数字进行转换的函数**

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。

13.7.3 **IP 地址与数字相互转换的函数**

INET_ATON(IP)将 IP 地址转换为数字表示； 其中，INET_ATON(IP)函数中 IP 值需要加上引号。互为反函数。

INET_NTOA(n)将数字 n 转换成 IP。

SELECT INET_ATON('59.65.226.15'),INET_NTOA(994173455); ->| 994173455 | 59.65.226.15|

13.7.4 **加锁函数和解锁函数**

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。 SELECT GET_LOCK('MYSQL',10); SELECT RELEASE_LOCK('MYSQL');

RELEASE_LOCK(name)解除名称为 name 的锁。如果解锁成功，返回 1；如尝试超时，返回 0；如解锁失败，返回 NULL；

IS_FREE_LOCK(name)判断是否使用名为 name 的锁。如果使用返回 0；否则返回 1。 SELECT IS_FREE_LOCK('MYSQL');

13.7.5 **重复执行指定操作的函数**

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

SELECT BENCHMARK(10000000,NOW());

13.7.6 **改变字符集的函数**

CONVERT(s USING cs)函数将字符串 s 的字符集变成 cs。【示例 13-68】下面将字符串“ABC”的字符集变成 gbk。

SELECT CHARSET('ABC'),CHARSET(CONVERT('ABC' USING gbk));

13.7.7 **改变字段数据类型的函数**

CAST(x AS type)和 **CONVERT(x,type)**将 x 变成 type 类型。只对 **BINARY、CHAR、DATE、DATETIME、TIME、SIGNED INTEGER、UNSIGNED INTEGER** 这些类型起作用。但只是改变了输出值数据类型，并没有改变表中字段的类型

SELECT d, CAST(d AS DATE),CONVERT(d,TIME) FROM t7; -> | 2014-03-14 18:20:16 | 2014-03-14 | 18:20:16 |

第 14 章 存储过程和函数 **拥有 EXECUTE 权限，存储在 information_schema 下面的 USER_PRIVILEGES 表中.**

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 **将经常使用的一组 SQL 语句组合一起，并将语句当作一个整体存储在 MySQL 服务器中。**

14.1.1、创建存储过程

CREATE PROCEDURE sp_name ([proc_parameter[,...]]) procedure 发音 [prə 'si:dʒə]

[characteristic ...] routine_body characteristic n. 特征；特性；特色

proc_parameter [IN|OUT|INOUT] param_name type **(- any data style);**

routine_body 由 SQL 组成(可以用 BEGIN…END 来标志的 SQL 代码)

characteristic 取值如下：

LANGUAGE SQL： 默认，说明 routine_boyd 部分是由 SQL 语言的语句组成，这也是数据库系统默认的语言；

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

```
DELIMITER &&
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    &&
  DELIMITER ;
```

14.1.2、创建存储函数

CREATE FUNCTION sp_name ([func_parameter[,...]])

RETURNS type

[characteristic ...] routine_body

ERROR 1418 (HY000): This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled (you *might* want to use the less safe log_bin_trust_function_creators variable)

SET GLOBAL log_bin_trust_function_creators=TRUE;

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

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]…**　　SET my_sql=30;

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

SELECT d_id **INTO my_sql**　FROM employee WHERE id=2;

14.1.4、定义条件和处理程序　事先定义程序执行过程中可能遇到的问题，且可在处理程序中定义解决这些问题的办法。

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　UNDO 目前 MySQL 不支持

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

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(游)标　存储过程|函数中，使用光标来逐条读取查询结果集中的记录。必声明在处理程序前，在变量和条件之后。

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 cursor_name 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　存储过程和存储函数中可以使用流程控制来控制语句的执行。

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;
IF @count=100 THEN
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 **CALL 语句**调用。而存储函数的使用方法与 MySQL 内部函数一样。执行存储过程和存储函数需要拥有 EXECUTE 权限，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[,…]]) ;** CALL num_from_employee(1002,@n); **S**ELECT @n;

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 SELECT name_from_employee(3);

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 ' ;
SHOW PROCEDURE STATUS LIKE'num_from_employee' \G
SHOW CREATE PROCEDURE num_from_employee \G
SELECT * FROM information_schema.Routines WHERE ROUTINE_NAME='num_from_employee' \G
SELECT SPECIFIC_NAME,SQL_DATA_ACCESS,SECURITY_TYPE FROM information_schema.Routines WHERE ROUTINE_NAME='num_from_employee';

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'

SELECT SPECIFIC_NAME,SQL_DATA_ACCESS,SECURITY_TYPE FROM information_schema.Routines WHERE ROUTINE_NAME='num_from_employee';

ALTER PROCEDURE num_from_employee
MODIFIES SQL DATA
SQL SECURITY INVOKER;

SELECT SPECIFIC_NAME,SQL_DATA_ACCESS,ROUTINE_COMMENT FROM information_schema.Routines WHERE ROUTINE_NAME='name_from_employee';

ALTER FUNCTION name_from_employee
COMMENT'FIND NAME';

14.5、删除存储过程和函数

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

```
DELIMITER &&
CREATE PROCEDURE food_price_count(IN price_info1 FLOAT, IN price_info2 FLOAT, OUT count INT)
    READS SQL DATA
    BEGIN
        DECLARE temp FLOAT;
        DECLARE match_price CURSOR FOR SELECT price FROM food;
        DECLARE EXIT HANDLER FOR NOT FOUND CLOSE match_price;
        SET @sum=0;
        SELECT COUNT(*) INTO count FROM food
            WHERE price>price_info1 AND price<price_info2;
        OPEN match_price;
        REPEAT
            FETCH match_price INTO temp;
              IF temp>price_info1 AND temp<price_info2
                  THEN SET @sum=@sum+temp;
              END IF;
    UNTIL 0 END REPEAT;
    CLOSE match_price;
    END &&
DELIMITER ;
```

第 15 章 MySQL 用户管理

15.1、权限表中最重要的是 user 表、db 表和 host 表。除此之外，还有 tables_priv 表、columns_priv 表、proc_priv 表等

15.1.1 user 表有 39 个字段，大致可以分为四类，分别是 1．用户列 2．权限列 3．安全列 4．资源控制列

15.1.2 db 表和 host 表 db 表中存储了某个用户对一个数据库权限。db 表、host 表表结构差不多 字段分为用户列和权限列。

15.1.3 tables_priv 表和 columns_priv 表

columns_priv 表可以对单个数据列进行权限设置。tables_priv 表包含 8 个字段，分别是 Host、Db、User、Table_name、Table_priv、Column_priv、Timestamp 和 Grantor。前四个字段分别表示主机名、数据库名、用户名和表名。Table_priv 表示对表进行操作的权限。这些权限包括 Select、Insert、Update、Delete、Create、Drop、Grant、References、Index 和 Alter。**Column_priv 表**示对表中的数据列进行操作的权限。这些权限包括 Select、Insert、Update 和 References。Timestamp 表示修改权限的时间。Grantor 表示权限是谁设置的。

15.1.4 procs_priv 表 可以存储过程和存储函数进行权限设置。procs_priv 表包含 8 个字段，分别是 Host、Db、User、Routine_name、Routine_type、Proc_priv、Timestamp 和 Grantor。前三个字段分别表示主机名、数据库名和用户名。Routine_name 字段表示存储过程或函数的名称。Routine_type 字段表示类型。该字段有两个取值，分别是 FUNCTION 和 PROCEDURE。FUNCTION 表示这是一个存储函数。PROCEDURE 表示这是一个存储过程。Proc_priv 字段表示拥有的权限。权限分为 3 类，分别是 Execute，Alter Routine 和 Grant。Timestamp 字段存储更新的时间。Grantor 字段存储权限是谁设置的

15.2、账户管理

15.2.1、**登录 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 双引号括
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。
mysql -h 219.225.16.17 -u root -p test; mysql -h 127.0.0.1 -u root –p mysql -h localhost -u root -p test -e "DESC employee";
退出 MySQL 服务器 EXIT QUIT \q
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]’pd’]]...
CREATE USER 'test2'@'localhost' IDENTIFIED BY 'test2';
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;
INSERT INTO mysql.user(Host,User,Password,ssl_cipher,x509_issuer,x509_subject)
VALUES ('localhost', 'test22',PASSWORD('test22'), '', '', ''); FLUSH PRIVILEGES; SELECT * FROM mysql.user \G
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’]]...
mysql> GRANT SELECT ON *.* TO 'test3'@'localhost' IDENTIFIED BY 'test3'; SELECT * FROM mysql.user WHERE User='test3' \G
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 'test2'@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;**
DELETE FROM mysql.user WHERE Host='hostname' AND User='test3'; 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”;**
C:\Users\Administrator>mysqladmin -u root -p password "new_pw"
2、修改 user 表
UPDATE mysql.user SET password=PASSWORD(“new_pd”) WHERE user=’root’ and host=’localhost’; FLUSH PRIVILEGES;
mysql>UPDATE mysql.user SET password=PASSWORD("****") WHERE user='root' and host='loc'; FLUSH PRIVILEGES;
3、使用 SET 语句来修改 root 用户的密码 **SET PASSWORD=PASSWORD(“new_password”);**
mysql> SET PASSWORD=PASSWORD("test");
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("test1") WHERE user='' and host=''; FLUSH PRIVILEGES;
mysql>UPDATE mysql.user SET password=PASSWORD("test1") WHERE user='test1' and host='localhost'; 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”);**
mysql> SET PASSWORD FOR 'test1'@'localhost'=PASSWORD("test11");
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’]]...
mysql> GRANT SELECT ON *.* TO 'test1'@'localhost' IDENTIFIED BY 'test11'; SELECT * FROM mysql.user WHERE User='test1' \G
15.2.6、普通用户修改密码 **SET PASSWORD=PASSWORD(“new_password”);**
mysql> SET PASSWORD=PASSWORD("test1");
15.2.7、root 用户密码丢失的解决办法 控制面板——管理工具——服务——MySQL（右键）—停止 **mysql -u root**
1、使用—skip-grant-tables 选项来启动 MySQL 服务 C:\Users\Administrator>**mysqld --skip-grant-tables**
#/etc/init.d/mysql start –mysqld –skip-grant-tables
2、登录 root，设置新密码
C:\Users\Administrator>**mysql -u root**

mysql> **update mysql.user set password=password("**new_pd **") where user='root' and host='localhost';** FLUSH PRIVILEGES;
3、加载权限表 **FLUSH PRIVILEGES;** **退出窗口—资源管理器中关闭"MySQLD"** 服务——MySQL（右键）—开启
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**

权 限 名 称	对应 user 表中的列	权限的范围
CREATE	Create_priv	数据库、表或索引
DROP	Drop_priv	数据库或表
GRANT OPTION	Grant_priv	数据库、表、存储过程或函数
REFERENCES	References_priv	数据库或表
ALTER	Alter_priv	修改表
DELETE	Delete_priv	删除表
INDEX	Index_priv	用索引查询表
INSERT	Insert_priv	插入表
SELECT	Select_priv	查询表
UPDATE	Update_priv	更新表
CREATE VIEW	Create_view_priv	创建视图
SHOW VIEW	Show_view_priv	查看视图
ALTER ROUTINE	Alter_routine_priv	修改存储过程或存储函数
CREATE ROUTINE	Create_routine_priv	创建存储过程或存储函数
EXECUTE	Execute_priv	执行存储过程或存储函数
FILE	File_priv	加载服务器主机上的文件
CREATE TEMPORARY TABLES	Create_tmp_table_priv	创建临时表
LOCK TABLES	Lock_tables_priv	锁定表
CREATE USER	Create_user_priv	创建用户
PROCESS	Process_priv	服务器管理
RELOAD	Reload_priv	重新加载权限表
REPLICATION CLIENT	Repl_client_priv	服务器管理
REPLICATION SLAVE	Repl_slave_priv	服务器管理
SHOW DATABASES	Show_db_priv	查看数据库
SHUTDOWN	Shutdown_priv	关闭服务器
SUPER	Super_priv	超级权限

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 OPTION: 该用户可以将这些权限赋予给别的用户；

MAX_QUERIES_PER_HOUR count: 设置每小时可以允许执行的 COUNT 次查询；

MAX_UPDATES_PER_HOUR count: 设置每小时可以允许执行的 COUNT 次更新；

MAX_CONNECTIONS_PER_HOUR count: 设置每小时可以允许建立 COUNT 连接次数；

MAX_USER_CONNECTIONS count: 设置每小时可以同时具有的的 COUNT 个连接数；

priv_type 权限类型 **column_list** 权限作用于哪个列上， 如果缺省则作用于整个表上； **user** **'user'@'host'**

database.table *.* **所有库的所有表**

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

GRANT SELECT,UPDATE ON *.* TO 'test5'@'localhost'IDENTIFIED BY 'test5' WITH GRANT OPTION;

SELECT Host,User,Password,Select_priv,Update_priv,Grant_priv FROM user WHERE user='test5' \G

15.3.3、收回权限 **保证数据库的安全**

REVOKE priv_type [(column_list)] [, priv_type [(column_list)]] ...

ON [object_type] {tbl_name | * | *.* | db_name.*}

FROM user [, user] ...

REVOKE UPDATE ON *.* FROM 'test5'@'localhost';

收回所有权限： **REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...**

REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'test5'@'localhost';

15.3.4 查看权限 1. **SELECT * FROM mysql.user \G 2. SHOW CREATE FOR user** SHOW CREATE FOR 'test5'@'localhost';

第16章 数据备份与还原

2．如何升级MySQL数据库？

升级 MySQL 数据库，可以按照下面的步骤进行：

（1）先使用 mysqldump 命令备份 MySQL 数据库中的数据。这样做的目的是为了避免误操作引起 MySQL 数据库中的数据丢失。

（2）停止 MySQL 服务。可以直接中止 MySQL 服务的进程。但是最好还是用安全的方法停止 MySQL 服务。这样可以避免缓存中的数据丢失。

（3）卸载旧版本的 MySQL 数据库。通常情况下，卸载 MySQL 数据库软件时，系统会继续保留 MySQL 数据库中的数据文件。

（4）安装新版本的 MySQL 数据库，并进行相应地配置。

（5）启动 MySQL 服务，登录 MySQL 数据库查询数据是否完整。如果数据不完整，使用之前备份的数据进行恢复。

16.1、数据备份

16.1.1、使用 mysqldump 命令备份 表的结构和表中的数据将存储在生成的文本文件中（结构、CREATE 所有记录 INSERT）

1．备份一个数据库 **mysqldump -u username –p dbname table1 table2...>BackupName.sql/txt**

2．备份多个数据库 **mysqldump -u username -p --databases dbname1 dbname2...>BackupName.sql/txt**

3．备份所有数据库 **mysqldump -u username -p** –all **databases >BackupName.sql/txt**

mysqldump [OPTIONS] database [tables]

mysqldump [OPTIONS] --databases [OPTIONS] DB1 [DB2 DB3...]

mysqldump [OPTIONS] --all-databases [OPTIONS]

C:\Users\Administrator>mysqldump -u root -p test employee>C:/employee.sql mysql -u root -p < C:\all.sql

C:\Users\Administrator>mysqldump -u root -p --databases test mysql>C:\backup.sql

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 对 InnoDB 存储引擎的表不适用。对于 MyISAM 存储引擎的表，很方便，最简单，速

度也最快。使用这种方法时，最好将服务器先停止。这样，可以保证在复制期间数据库中的数据不会发生变化。
大版本号 相同数据库数据库文件格式相同 **E:\Program Files\MySQL\MySQL server 5.1\data**
16.1.3、使用 mysqlhotcopy 工具快速备份
备份时不停止 MySQL 服务器，备份方式比 mysqldump 命令快。mysqlhotcopy 工具是一个 Perl 脚本，主要在 Linux 操作系统下使用。mysqlhotcopy 工具使用 LOCK TABLES、FLUSH TABLES 和 cp 来进行快速备份。其工作原理是：先将需要备份的数据库加上一个读操作锁，然后用 FLUSH TABLES 将内存中的数据写回到硬盘上的数据库中，最后把需要备份的数据库文件复制到目标目录。使用 mysqlhotcopy 的命令如下：
[root@localhost ~]# mysqlhotcopy [option] dbname1 dbname2 … backupDir
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 必须保证两个 MySQL 数据库的主版本号相同，才能保证这两个 MySQL 数据库的文件类型是相同。对 MyISAM 类型的表比较有效。对于 InnoDB 类型的表则不可用。在 Windows 操作系统下，MySQL 的数据库目录通常存放下面三个路径的其中之一。分别是 C:\mysql\date、C:\Documents and Settings\All Users\Application Data\MySQL\MySQL Server 5.1\data 或者 C:\Program Files\MySQL\MySQL Server 5.1\data。在 Linux 操作系统下，数据库目录通常在/var/lib/mysql/、/usr/local/mysql/data 或者/usr/local/mysql/var 这三个目录下
16.3、数据库迁移 原因是升级计算机，或者是部署开发的管理系统，或者升级 MySQL 数据库，或者换用其他数据库。
16.3.1、相同版本的 MySQL 数据库之间的迁移 MyISAM 类型 通过复制数据库目录来实现
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 MySQL 升级的原因。高版本的 MySQL 数据库通常都会兼容低版本，MySIAM 类型的表可以直接复制，也可以使用 mysqlhotcopy 工具。但是 InnoDB 类型的表不可以使用这两种方法。最常用的办法是使用 mysqldump 命令来进行备份，然后通过 mysql 命令将备份文件还原到目标 MySQL 数据库中。
16.3.3、不同数据库之间的迁移 从其他类型的数据库迁移到 MySQL，或从 MySQL 迁移到其他类型的。这种迁移没有普通适用的解决办法。例 微软 SQL Server 软件使用 T-SQL 语言(包含了非标准的 SQL 语句) 和 MySQLSQL 语句不能兼容。
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 可导出成**文本文件、XML 文件**或者 **HTML 文件**。相应的文本文件也可以导入 MySQL 数据库中。
16.4.1、用 SELECT …INTO OUTFILE 导出文本文件 能根据条件导出数据
SELECT [列名] FROM table [WHERE 语句]
INTO OUTFILE ‘目标文件’ [OPTION]
FIELDS TERMINATED BY '字符串': 设置字符串为字段的分隔符，默认值是\t;
FIELDS ENCLOSED BY '字符串': 设置字符来括上字段的值。默认不用符号;
FIELDS OPTIONALLY ENCLOSED BY '字符串': 设置字符来括上 CHAR VARCHAR TEXT 等字符型字段，默认不用符号;
FIELDS ESCAPED BY '字符串': 设置转义字符，默认是 \;
LINES STARTING BY '字符串': 设置每行开头的字符，默认无字符;
LINES TERMINATED BY '字符串': 设置每行的结束符，默认是\n;
SELECT * FROM test.teacher INTO OUTFILE 'C:/teacher1.txt'
FIELDS TERMINATED BY'\、' OPTIONALLY ENCLOSED BY '\"' LINES STARTING BY'\>'
TERMINATED BY '\r\n';
16.4.2、用 mysqldump 命令导出文本文件 备份数据库中的数据。
mysqldump -u root -pPassword -T 目标目录或文件 dbname table [option] ;
--fields-terminated-by=...，字符串 设置字符串为字段的分隔符，默认值是\t;
--fields-enclosed-by=...，字符 设置字符来括上字段的值;
--fields-optionally-enclosed-by=...，字符 设置字符来括上 CHAR VARCHAR TEXT 等字符型字段;
--fields-escaped-by=...，字符 设置转义字符;
--fields-terminated-by=...字符串 设置每行的结束符;
导出的是 txt + sql 文件
C:\Users\Administrator> mysqldump -u root -p -T C:\ test teacher "--fields-terminated-by=," "--fields-optionally-enclosed-by=""
C:\Users\Administrator>mysqldump -u root -p --xml test teacher > C:/teacher.xml
16.4.3、 mysql 命令导出 可以登录 MySQL 服务器，可以还原备份文件，可以导出文本文件、**XML 文件**或 **HTML 文件**：
mysql -u root -pPassword -e "SELECT 语句" dbname> C:/name.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

其中，“Password”表示密码；使用-e 选项就可以执行 SQL 语句；“SELECT 语句”用来查询记录；

C:\Users\Administrator> mysql -u root -p -e "SELECT * FROM teacher" test >C:/teacher2.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]

LOAD DATA INFILE 'C:/teacher.txt' INTO TABLE teacher

FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"';

16.4.5、mysqlimport 命令导入 txt **mysqlimport –u root –pPassword [--LOCAL] dbname file [OPTION]** （**mysqldump**）

mysqlimport -u root -p test C:\teacher.txt "--fields-terminated-by=," "--fields-optionally-enclosed-by=""

第 17 章 MySQL 日志

17.1、日志简介

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。可以查询 MySQL 数据库哪些改变。

错误日志

通用查询日志

慢查询日志

17.2、二进制日志 也变更日志（update log），

17.2.1、启动和设置 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]] **c:\mysqllog** 服务中 MySQL 重启服务 **mylog.000001, 2, 3…..**

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** cd c:\mysqllog\ mysqlbinlog mylog.000001

17.2.3、删除二进制日志 记录大量的信息。如果很长时间不清理二进制日志，将会浪费很多的磁盘空间

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 all 二进制文件

PURGE MASTER LOGS TO ‘mysqlbinlog mylog.000004’

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

暂停 MySQL 服务？？？ mysqlbinlog mysqlbinlog mylog.000005 | mysql -u root –p

17.2.5、暂时停止二进制日志功能 **SET SQL_LOG_BIN=0 / 1**

17.3、错误日志 主要用来记录 MySQL 服务的开启、关闭和错误信息。 默认开启的，而且，错误日志无法被禁止。

17.3.1、启动和设置错误日志 默认情况下，存储在 MySQL 数据库的数据文件夹下。通常的名称为 hostname.err。其中，hostname 服务器的主机名。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 Windows 可用文本文件查看器。Linux 可以使用 vi 工具或者使用 gedit 工具来查看。

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：

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 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。默认关闭

17.4.1、启动和设置通用查询日志 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： # my.cnf（Linux 操作系统下）或者 my.ini（Windows 操作系统下）

[mysqld] log [=DIR \ [filename]] log=C:/log/hjh

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。会直接覆盖旧的查询日志。

mysqladmin 命令的语法如下： **mysqladmin -u root -p flush-logs**

17.5、慢查询日志 记录执行时间超过指定时间的查询语句。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： # 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**

log-slow-queries = C:/log/hjh/slow long_query_time=3

17.5.2、查看慢查询日志 文本编辑/查看器 重复执行 n 次的运算 mysql> SELECT BENCHMARK(20000000,1*2); 暂停服务查看

17.5.3、删除慢查询日志 和通用查询日志的删除方法是一样的。可以使用 mysqladmin 命令和手工方式来删除。mysqladmin 命令的语法如下： **mysqladmin -u root -p flush-logs**

执行该命令后，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**。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’
错误日志	[mysqld] log-error[=DIR \ [filename]]	是	文本查看/编辑器	mysqladmin -uroot -p flush-logs
通用查询日志	[mysqld] log [=DIR \ [filename]]	否	同上	mysqladmin -uroot -p flush-logs
慢查询日志	log-slow-queries[=DIR \ [filename]] long_query_time=n	否	同上	mysqladmin -uroot -p flush-logs

第 18 章 性能优化

18.1、优化简介 通过某些有效的方法提高 MySQL 数据库的性能。目的是为了使 MySQL 数据库运行速度更快、占用的磁盘空间更小。性能优化包括很多方面，例如优化查询速度、优化更新速度、优化 MySQL 服务器等

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 了解查询语句的执行情况。MySQL 中，可以使用 EXPLAIN 语句和 DESCRIBE 语句来分析查询语句

18.2.1、分析查询语句 **EXPLAIN/DESC select;** EXPLAIN SELECT * FROM student \G

EXPLAIN SELECT * FROM student WHERE name='张三' \G

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

ref 表示多表查询时，后面的表使用了普通索引

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

unique_subquery 表示子查询中合作了 unique 或 primary key

eq_ref 表示多表连接时，后面的表使用了 unique 或 PRIMARY KEY

18.2.2、索引 如果查询时不使用索引，查询语句将查询表中的所有字段。这样查询的速度会很慢。如果使用索引进行查询，查询语句只查询索引字段。这样可以减少查询的记录数，达到提高查询速度的目的。

EXPLAIN SELECT * FROM student WHERE name='张三' \G

1、走单列索引 **Like 以%开头的不走** EXPLAIN SELECT * FROM student WHERE name LIKE '%四' \G

2、走多列索引　**多列索引第一个字段没有使用不走索引**　EXPLAIN SELECT * FROM student WHERE department='英语系' \G

Or 两边的列有一个没有建立索引不走索引　EXPLAIN SELECT * FROM student WHERE name='张三' or sex='女' \G

4、 不走索引的查询

18.2.3 优化子查询　子查询可以使查询语句很灵活，但子查询的执行效率不高。子查询时，MySQL 需要为内层查询语句的查询结果建立一个临时表。然后外层查询语句再临时表中查询记录。查询完毕后，MySQL 需要撤销这些临时表。因此，子查询的速度会受到一定的影响。如果查询的数据量比较大，这种影响就会随之增大。在 MySQL 中可以使用连接查询来替代子查询。连接查询不需要建立临时表，其速度比子查询要快。

18.3、优化数据库结构　数据库结构是否合理，存在冗余、对表的查询和更新的速度、表中字段的数据类型合理等内容

18.3.1、将字段很多的表分解成多个表　对于字段特别多且有些字段的使用频率很低的表，可以将其分解成多个表。

18.3.2、增加中间表　先分析经常需要同时查询哪几个表中的哪些字段。然后将这些字段建立一个中间表，并从原来那几个表将数据插入到中间表中。之后就可以使用中间表来进行查询和统计了。

18.3.3、增加冗余字段　设计数据库表的时候尽量让表达到三范式。 表的规范化程度越高，表与表之间的关系就越多。查询时可能经常需要多个表之间进行连接查询。而进行连接操作会降低查询速度，连接查询会浪费很多的时间。因此可以在 student 表中增加一个冗余字段 dept_name，该字段用来存储学生所在院系的名称。这样就不用每次都进行连接操作了。

反范式

空间换时间

18.3.4、优化插入记录的速度　插入记录时，索引、惟一性校验都会影响到插入记录的速度。而且，一次插入多条记录和多次插入记录所耗费的时间是不一样的。根据这些情况，分别进行不同的优化。

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　**(f1,f2….fn) VALUES (v1,v2….vn),…**代替多个 INSERT INTO

INSERT INTO table (f1,f2….fn) VALUES (v1,v2….vn),(f1,f2….fn) VALUES (v1,v2….vn),

18.3.5、分析、检查和优化表

分析表主要作用是分析关键字的分布　**ANALYZE TABLE table1[, table2…]**

检查表主要作用是检查表是否存在错误　**CHECK TABLE table1[, table2…]**

优化表主要作用是消除删除或者更新造成的空间浪费　**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　从硬件方面来进行优化。另一从 MySQL 服务的参数进行优化。通过这些优化方式，可以提供 MySQL 的运行速度。但是这部分的内容很难理解，一般只有专业的数据库管理员才能进行这一类的优化。

18.4.1、优化服务器硬件　CPU　磁盘，阵列　内存　配置（与用服务器，大内存配置）

服务器的硬件性能直接决定着 MySQL 数据库的性能。例如，增加内存和提高硬盘的读写速度，这能够提高 MySQL 数据库的查询、更新的速度。因为内存的读写速度比硬盘的读写速度快。可以在内存中为 MySQL 设置更多的缓冲区，这样可以提高 MySQL 访问的速度。如果将查询频率很高的记录存储在内存中，那么查询速度就会很快。如果条件允许，可以将内存提高到 4G。并且选择 my-innodb-heavy-4G.ini 作为 MySQL 数据库的配置文件。但是，这个配置文件主要主要支持 InnoDB 存储引擎的表。如果使用 2G 内存，可以选择 my-huge.ini 作为配置文件。而且，MySQL 所在的计算机最好是专用数据库服务器。这样数据库可以完全利用该机器的资源

18.4.2、优化 MySQL 参数　my.cnf 或者 my.ini 文件的[mysqld]组

内存中会为 MySQL 保留部分的缓存区。这些缓存区可以提高 MySQL 数据库的处理速度。缓存区的大小都是在 MySQL 的配置文件中进行设置的。