MySql 核心技术手册 create by ferris，2017.5

SQL语句和函数

Chap 3 基础

创建用户

grant on命令格式：grant select on 数据库.* to 用户名@登录主机 identified by “密码”;
grant all on *.* to 'ferris' @'localhost' identified by '***';

grant all on *.* to 'ferris' @'localhost' identified by '***';

登录

mysql (-h localhost) -u user -p

mysql 命令本身对大小写不敏感，数据库和表名则取决于操作系统，如linux就敏感

创建

建立数据库 ： create datebase bookstore;

建立表 ： create table books (book_id int, title varchar(50), description text, genre enum('novel', 'poetry','drama') , author_id int auto_increment primary key);

显示

显示数据库 show databases；

显示表 ： show tables (from databaes);

显示表的内容： describe table ;

显示表的状态等: SHOW TABLES STATUS FROM tables LIKE tables;

切换

切换数据库use database；

插入数据

insert into books （ book_id, genre, author_id) values (888, 'novel', 1);

选择数据

显示所有行 select * from table

特定行： SELECT book_id, title FROM books WHERE genre = 'novel';

关联其他表join： SELECT book_id, title CONCAT (author_first, ' ', author_last) AS author FROM books JOIN author USING (author_id) WHERE author_last = '

TOM';

查询LIKE： SELECT book_id, title CONCAT (author_first, ' ', author_last) AS author FROM books JOIN author USING (author_id) WHERE author_last LIKE = '% TOM% ' and title = ' %winter% ' ； %通配符表示可以有零个火多个字符

排序 分组 限制

排序：SELECT book_id, title CONCAT (author_first, ' ', author_last) AS author FROM books JOIN author USING (author_id) WHERE author_last = 'TOM' ORDER BY title ; （默认字母数字顺序升序）降序加上DESC 在 title 后面

限制： 在上面最后加上 LIMIT 20； 显示记录在20行之内。 LIMIT 带有两个参数，第一个是将要略过的行数或取回数据的起始点，第二个是将要显示记录数。

分组： 最后加上 GROUP BY

分析和处理

SELECT COUNT (*) FROM books JOIN author USING (author_id) WHERE author_last = 'Tolstoy';

SELECT SUM(sale_amount) AS 'Tolstoy Sales' FROM .....;

显示时间： SELECT DATE_FORMAT(purchase_date, "%M %d,， %Y“） AS ‘Date’ FORM .....

修改数据

修改： UPDATE books SET years = '1938' WHERE book_id = 2;

重复替换： REPLACE INTO books (title, isbn, genre) VALUES （‘apple book, 11101，‘novel’），（‘cpp prime' , 22010, 'novel');

删除数据

子查询语句: DELETE FROM books WHERE author_id = (SELECT author_name FROM authors WHERE author_last = 'Tom')

设置变量形式： SET @potter = (SELECT author_name FROM authors WHERE author_last = 'Tom'); DELETE FROM books WHERE author_id = @potter;

批量导入数据P40

Chap 4 用户安全管理

与安全和用户有关

用户访问权限 在MyISAM表中

user 全局层级权限

db 数据库层级权限

tables_priv 表层级权限

columns_priv 列层级权限

CREATE USER

CREATE USER 'user' [@'localhost'] IDENTIFIED BY 'new_password' PASSWORD EXPIRE;

RENAME USER

RENAME USER old_user TO new_user [, old_user TO new_user]

SET PASSWORD

SET PASSWORD [FOR 'jeffrey'@'localhost'] = PASSWORD('mypass')

DROP USER

DROP USER [IF EXISTS] user [, user] ...

删除全局权限用户，要先 撤销用户权限 REVOKE ALL ON *.* FROM 'user'@'host'

GRANT 授权

GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ... ON [object_type] priv_level TO user [auth_option] [, user [auth_option]] ... [REQUIRE {NONE | tls_option [[AND] tls_option] ...}] [WITH {GRANT OPTION | resource_option} ...]

限制连接类型

连接次数和数目

SHOW GRANTS；

撤销授权 REVOKE

REVOKE priv_type [(column_list)] [, priv_type [(column_list)]] ... ON [object_type] priv_level FROM user [, user] ... REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ... REVOKE PROXY ON user FROM user [, user] ...

SHOW PRIVILEGES；

FLUSH 清除并重载MySQL的临时缓存

FLUSH [NO_WRITE_TO_BINLOG | LOCAL] flush_option [, flush_option] ...

与函数有关 SELECT +

AES_DECRYPT(string， password); Advanced Encryption Standard 解密文本

AES_ENCRYPT(string, password) 加密文本

DECODE （string，password） 二进制格式

ENCODE （string，password）

DES_DECRYPT(string, [key]) 128位长度关键字解密

DES_ENCRYPT(string, [key]) 128位长度关键字加密文本

ENCRYPT（string[，seed]) 调用c语言crypt， 加密不能解密

SHA=SHA1（string) 为指定的字符串返回安全散列算法160位校验和 Secure Hash Algorithm;

MD5（string）使用md5 128位校验和返回32位二进制字符串的1321标准的散列值

PASSWORD（string）用mysql中的password列加密密码，不能解密

SESSION）DES_DECRYPT(string, [key]) 128位长度关键字解密

SESSION_USER()=SYSTEM_USER()=USER();当前用户名和主机的组合

Chap5 数据库和表模式

建立数据库 mysql 中 DATABASE等价于SCHEMA

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name [create_specification]

建立表CREATE TABLE

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name [(create_definition,...)] [table_options] [partition_options] select_statement [LIKE old_tbl_name | (LIKE old_tbl_name]

添加索引 CREATE INDEX

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name [index_type] ON tbl_name (index_col_name,...) [index_option] [algorithm_option | lock_option]

创建视图CREATE VIEW

CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}] [DEFINER = { user | CURRENT_USER }] [SQL SECURITY { DEFINER | INVOKER }] VIEW view_name [(column_list)] AS select_statement [WITH [CASCADED | LOCAL] CHECK OPTION]

创建服务CREATE SERVER

CREATE SERVER server_name FOREIGN DATA WRAPPER wrapper_name OPTIONS (option [, option] ...)

option: { HOST character-literal | DATABASE character-literal | USER character-literal | PASSWORD character-literal | SOCKET character-literal | OWNER character-literal | PORT numeric-literal }

修改数据库设置

ALTER {DATABASE | SCHEMA} [db_name] [DEFAULT] CHARACTER SET [=] charset_name | [DEFAULT] COLLATE [=] collation_name

修改连接参数

ALTER SERVER server_name OPTIONS (option [, option] ...)

修改表的结构和其他属性 ALTER TABLE tbl_name [alter_specification [, alter_specification] ...] [partition_options]

ADD子句

适用列：| ADD [COLUMN] col_name column_definition [FIRST | AFTER col_name] | ADD [COLUMN] (col_name column_definition,...)

适用标准索引：ADD {INDEX|KEY} [index_name] [index_type] (index_col_name,...) [index_option] ...

适用FULLTEXT（为text，char，varchar添加索引)ADD FULLTEXT [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...

适用SPATIAL索引： ADD SPATIAL [INDEX|KEY] [index_name] (index_col_name,...) [index_option] ...

适用外键 foreign key 索引： ADD [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)[index_option] ... | ADD [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY] [index_name] ADD [CONSTRAINT [symbol]] FOREIGN KEY [index_name] (index_col_name, ...)reference_definition

CHANGE 子句

设置列的默认值 ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}

CHANGE [COLUMN] old_col_name new_col_name column_definition[FIRST|AFTER col_name] 改变列的定义，名称等

MODIFY [COLUMN] col_name column_definition [FIRST | AFTER col_name] 改变列的定义

DROP 子句

删除指定列及其数据 DROP [COLUMN] col_name

删除索引

删除标准索引|DROP PRIMARY KEY

删除主键索引|DROP {INDEX|KEY} index_name

删除外键 DROP FOREIGN KEY fk_symbol

综合子句

转换并设置字符集 CONVERT TO CHARACTER SET charset_name [COLLATE collation_name] | [DEFAULT] CHARACTER SET [=] charset_name [COLLATE [=] collation_name]

禁用或启用主键 DISABLE KEYS | ENABLE KEYS 大量插入行，可先禁用索引，完成后再启用

导入删除表空间DISCARD TABLESPACE | IMPORT TABLESPACE

记录行 ORDER BY col_name [, col_name]

重命名表：RENAME [TO|AS] new_tbl_name

修改分区子句

Mysql支持水平分区，并不支持垂直分区; 水平分区：指将同一表中不同行的记录分配到不同的物理文件中； 垂直分区：指将同一表中不同列的记录分配到不同的物理文件中；

ADD PARTITION (partition_definition)

| DROP PARTITION partition_names

REORGANIZE PARTITION partition_names INTO (partition_definitions)

REMOVE PARTITIONING

分区管理子句

ANALYZE PARTITION {partition_names | ALL}

| CHECK PARTITION {partition_names | ALL}

| OPTIMIZE PARTITION {partition_names | ALL}

| REBUILD PARTITION {partition_names | ALL}

| REPAIR PARTITION {partition_names | ALL}

修改视图： ALTER VIEW

ALTER [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}] [DEFINER = { user | CURRENT_USER }] [SQL SECURITY { DEFINER | INVOKER }] VIEW view_name [(column_list)] AS select_statement [WITH [CASCADED | LOCAL] CHECK OPTION]

显示表的列信息DESCRIBE = EXPLAIN

{EXPLAIN | DESCRIBE | DESC} tbl_name [col_name | wild]

重命名表

RENAME TABLE tbl_name TO new_tbl_name

显示信息SHOW

创建数据库SHOW CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name

创建表SHOW CREATE TABLE tbl_name

列SHOW [FULL] COLUMNS {FROM | IN} tbl_name [{FROM | IN} db_name] [LIKE 'pattern' | WHERE expr]

视图SHOW CREATE VIEW view_name

表SHOW [FULL] TABLES [{FROM | IN} db_name] [LIKE 'pattern' | WHERE expr]

表的状态SHOW TABLE STATUS [{FROM | IN} db_name] [LIKE 'pattern' | WHERE expr]

安装的字符集SHOW CHARACTER SET [LIKE 'pattern' | WHERE expr]

字符集的校验SHOW COLLATION [LIKE 'pattern' | WHERE expr]

索引SHOW {INDEX | INDEXES | KEYS} {FROM | IN} tbl_name [{FROM | IN} db_name] [WHERE expr]

删除数据库

DROP {DATABASE | SCHEMA} [IF EXISTS] db_name

删除表

DROP [TEMPORARY] TABLE [IF EXISTS] tbl_name [, tbl_name] ... [RESTRICT | CASCADE]

删除索引

DROP INDEX index_name ON tbl_name [algorithm_option | lock_option] .

删除服务器

DROP SERVER [IF EXISTS] server_name

删除视图

DROP VIEW [IF EXISTS] view_name [, view_name] ... [RESTRICT | CASCADE]

Chap6 数据操纵语句和函数

语句

事务：事务是服务器执行的语句集合

必须满足4个条件（ACID）： Atomicity（原子性）、Consistency（稳定性）、Isolation（隔离性）、Durability（可靠性）

InnoDB,BDB,BDB CLuster支持，MyISAM不支持

开始事务

START TRANSACTION [transaction_characteristic [, transaction_characteristic] ...]

= BEGIN

提交事务

COMMIT [WORK] [AND [NO] CHAIN] [[NO] RELEASE]

SET autocommit = {0 | 1}，0为禁止

增加数据行 INSERT

INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE] [INTO] tbl_name [PARTITION (partition_name,...)] [(col_name,...)] {VALUES | VALUE} ({expr | DEFAULT},...),(...),... [ON DUPLICATE KEY UPDATE col_name=expr [, col_name=expr] ...]

检索并显示SELECT

[ALL | DISTINCT | DISTINCTROW] [HIGH_PRIORITY] [MAX_STATEMENT_TIME = N] [STRAIGHT_JOIN] [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT] [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS] select_expr [, select_expr ...] [FROM table_references [PARTITION partition_list] [WHERE where_condition] [GROUP BY {col_name | expr | position} [ASC | DESC], ... [WITH ROLLUP]] [HAVING where_ condition] [ORDER BY {col_name | expr | position} [ASC | DESC], ...] [LIMIT {[offset,] row_count | row_count OFFSET offset}] [PROCEDURE procedure_name(argument_list)] [INTO OUTFILE 'file_name' [CHARACTER SET charset_name] export_options | INTO DUMPFILE 'file_name' | INTO var_name [, var_name]] [FOR UPDATE | LOCK IN SHARE MODE]]

查询结果合并UNION

SELECT ... UNION [ALL | DISTINCT] SELECT ... [UNION [ALL | DISTINCT] SELECT ...]

删除表中数据行 DELETE

DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name [PARTITION (partition_name,...)] [WHERE where_condition] [ORDER BY ...] [LIMIT row_count]

替换数据行 REPLACE

REPLACE [LOW_PRIORITY | DELAYED] [INTO] tbl_name [PARTITION (partition_name,...)] [(col_name,...)] {VALUES | VALUE} ({expr | DEFAULT},...),(...),

插入新行，但有相同的值时替换已存在的行 相当于 INSERT + DELECT

修改已有数据UPDATE

UPDATE [LOW_PRIORITY] [IGNORE] table_reference SET col_name1={expr1|DEFAULT} [, col_name2={expr2|DEFAULT}] ... [WHERE where_condition] [ORDER BY ...] [LIMIT row_count]

控制表达式结果输出DO

DO expr [, expr]

显示列信息EXPLAIN

{EXPLAIN | DESCRIBE | DESC} tbl_name [col_name | wild]

{EXPLAIN | DESCRIBE | DESC} tbl_name SELECT ...

句柄 handle 提供一个直接访问表的一个通道（智能指针），比SELECT快

HANDLER tbl_name OPEN [[AS] alias]

HANDLER tbl_name READ index_name { = | <= | >= | < | > } (value1,value2,...) [WHERE where_condition] [LIMIT ...]

基于共用数据的列连在一起JOIN

SELECT ... |UPDATE... | UPDATE...

table_reference [INNER | CROSS] JOIN table_factor [join_condition] | table_reference STRAIGHT_JOIN table_factor | table_reference STRAIGHT_JOIN table_factor ON conditional_expr | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference join_condition | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor

join_condition: ON conditional_expr （=来指示） | USING (column_list)

索引提示 index_hint: USE {INDEX|KEY} [FOR {JOIN|ORDER BY|GROUP BY}] ([index_list]) | IGNORE {INDEX|KEY} [FOR {JOIN|ORDER BY|GROUP BY}] (index_list) | FORCE {INDEX|KEY} [FOR {JOIN|ORDER BY|GROUP BY}] (index_list)

限制返回行数 LIMIT

LIMIT count

LIMIT [offset,] cont

LIMIT count OFFSET offset

文本导入LOAD DATA

LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name' [REPLACE | IGNORE] INTO TABLE tbl_name [PARTITION (partition_name,...)] [CHARACTER SET charset_name] [{FIELDS | COLUMNS} [TERMINATED BY 'string'] [[OPTIONALLY] ENCLOSED BY 'char'] [ESCAPED BY 'char']]

保存点 SAVEPOINT

标志没完成事务

SAVEPOINT identifier

释放SAVEPOINT

RELEASE SAVEPOINT；

回滚SAVEPOINT

ROLLBACK [WORK] TO [SAVEPOINT] identifier

设置变量 SET

SET variable_assignment [, variable_assignment]

显示错误或警告

错误： SHOW ERRORS [LIMIT [offset,] row_count] SHOW COUNT(*) ERRORS

警告：SHOW WARNINGS [LIMIT [offset,] row_count] SHOW COUNT(*) WARNINGS

XA分布式事务

XA {START|BEGIN} xid [JOIN|RESUME]

XA END xid [SUSPEND [FOR MIGRATE]]

XA PREPARE xid

XA COMMIT xid [ONE PHASE]

XA ROLLBACK xid

XA RECOVER [CONVERT XID]

函数

对结果表分析ANALYSE（）

SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max_elements,[max_memory]])

评估服务器性能BENCHMARK（）

BENCHMARK(count,expr)

返回会话中的数据库（）

SELECT DATABASE()

SELECT SCHEMA（）

返回插入最后行的标志号

SELECT LAST_INSERT_ID()

返回修改的行数

SELECT ROW_COUNT()

Chap7 表和数据库管理语句和函数

语句

服务SERVER

创建服务CREATE SERVER

CREATE SERVER server_name FOREIGN DATA WRAPPER wrapper_name OPTIONS (option [, option] ...)

option: { HOST character-literal | DATABASE character-literal | USER character-literal | PASSWORD character-literal | SOCKET character-literal | OWNER character-literal | PORT numeric-literal }

修改服务ALTER SERVER；

ALTER SERVER server_name OPTIONS (option [, option] ...)

表TABLE

分析和存储信息 ANALYZE TABLE

ANALYZE [NO_WRITE_TO_BINLOG | LOCAL] TABLE tbl_name [, tbl_name]

对MyISAM表备份,不可靠

BACKUP TABLE table[,...] TO '/path'

恢复表RESTORE TABLE

RESTORE TABLE table [,...] FORM '/path'

检查表的错误CHECK TABLE

CHECK TABLE tbl_name [, tbl_name] ... [option]

option = {FOR UPGRADE| QUICK | FAST | MEDIUM| EXTENDED | CHANGED}

MyISAM表的活性校验和

CHECKSUM TABLE tbl_name [, tbl_name] ... [QUICK | EXTENDED]

锁定表LOCK TABLES，可提高速度

LOCK TABLES tbl_name [[AS] alias] lock_type [, tbl_name [[AS] alias] lock_type]

解锁UNLOCK TABLES
UNLOCK TABLES

优化表中的数据OPTIMIZE TABLE
OPTIMIZE [NO_WRITE_TO_BINLOG | LOCAL] TABLE tbl_name [, tbl_name

修复破坏的 MyISAM 表
REPAIR [NO_WRITE_TO_BINLOG | LOCAL] TABLE tbl_name [, tbl_name] … [QUICK] [EXTENDED] [USE_FRM]

清除临时缓存FLUSH
FLUSH [NO_WRITE_TO_BINLOG | LOCAL] flush_option [, flush_option] …

终止客户端连接KILL

先用SHOW PROCESSLIST 查看运行的线程
KILL [CONNECTION | QUERY] processlist_id

提前将索引加载到关键字缓存
LOAD INDEX INTO CACHE tbl_index_list [, tbl_index_list]

SHOW

存储引擎的细节信息
SHOW ENGINE engine_name {STATUS | MUTEX}

列举可用的表类型和引擎信息
SHOW [STORAGE] ENGINES

显示表缓存中打开表的list
SHOW OPEN TABLES [{FROM | IN} db_name] [LIKE 'pattern' | WHERE expr]

显示运行的线程
SHOW [FULL] PROCESSLIST

显示服务器信息及变量值
SHOW [GLOBAL | SESSION] STATUS [NOT | LIKE 'pattern' | WHERE expr]

显示表的状态信息
SHOW TABLE STATUS [{FROM | IN} db_name] [LIKE 'pattern' | WHERE expr]

显示系统变量
SHOW [GLOBAL | SESSION] VARIABLES [LIKE 'pattern' | WHERE expr]

显示插件

SHOW PLUGINS

函数

MySQL线程标识符
SELECT CONNECTION_ID()

获取第一个参数指定名字的一个锁
GET_LOCK (string, seconds)

释放GET_LOCK创建的锁
RELEASE_LOCK(string)

确定锁名是否可用
IS_FERR_LOCK(string)

锁名是否正在使用
IS_USED_LOCK(string)

返回一个通用唯一标识符UUID 128位，独一无二的数值
UUID（）
Universal Unique Identifier

Chap8 复制函数与语句

复制过程note
show global variables like '%bin%'

sudo lsof -nc mysqld | grep -vE '(.so(..*)?$|.frm|.MY?|.ibd|ib_logfile|ibdata|TCP)'

只有修改数据的语句才会记录在二进制日志文件中bin.log

ubuntu mysql 通过 mysqlbinlog 管理二进制日志
http://www.ilanni.com/?p=7816

(1) master将改变记录到二进制日志(binary log)中（这些记录叫做二进制日志事件，binary log events）；(2) slave将master的binary log events拷贝到它的中继日志(relay log)；(3) slave重做中继日志中的事件，将改变反映它自己的数据。

http://www.cnblogs.com/hustcat/archive/2009/12/19/1627525.html

在主从服务器都创建复制用户帐号

GRANT REPLICATION SLAVE，REPLICATION CLIENT ON *.* TO 'replicant'@'slave_host' IDENTIFIED BY 'my_pwd'

设置服务器

/etc/ my.ini 或my.cnf

复制数据库或启动复制

使用mysqldump工具

mysqldump [OPTIONS] database [tables]

转到操作系统级

FLUSH TABLE WITH READ LOCK 获得全局读锁定

备份 MySQL 下的data目录

LOAD DATA FROM MASTERR 不推荐

开始复制

START SLAVE 主从服务器建立连接

使用复制备份

阻止从服务器复制 STOP SLAVE

复制函数和语句

启动从服务器

START SLAVE [thread_types] [until_option] [connection_options] [channel_option]

修改在从服务器上与主服务器和复制相关的设置

CHANGE MASTER TO option [, option] ... [channel_option]

option: MASTER_BIND = 'interface_name' | MASTER_HOST = 'host_name' | MASTER_USER = 'user_name' | MASTER_PASSWORD = 'password' | MASTER_PORT = port_num | MASTER_CONNECT_RETRY = interval | MASTER_RETRY_COUNT = count | MASTER_DELAY = interval | MASTER_HEARTBEAT_PERIOD = interval | MASTER_LOG_FILE = 'master_log_name' | MASTER_LOG_POS = master_log_pos | MASTER_AUTO_POSITION = {0|1} | RELAY_LOG_FILE = 'relay_log_name' | RELAY_LOG_POS = relay_log_pos | MASTER_SSL = {0|1} | MASTER_SSL_CA = 'ca_file_name' | MASTER_SSL_CAPATH = 'ca_directory_name' | MASTER_SSL_CERT = 'cert_file_name' | MASTER_SSL_CRL = 'crl_file_name' | MASTER_SSL_CRLPATH = 'crl_directory_name' | MASTER_SSL_KEY = 'key_file_name' | MASTER_SSL_CIPHER = 'cipher_list' | MASTER_SSL_VERIFY_SERVER_CERT = {0|1} | MASTER_TLS_VERSION = 'protocol_list' | IGNORE_SERVER_IDS = (server_id_list) channel_option: FOR CHANNEL channel server_id_list: [server_id [, server_id] ...]

控制主从服务器的同步

MASTER_POS_WAIT(log_name,log_pos[,timeout][,channel_name])

删除主从服务器的二进制日志

BINARY = MASTER

PURGE { BINARY | MASTER } LOGS { TO 'log_name' | BEFORE datetime_expr }

删除主服务器所有二进制日志
RESET MASTER

删除从服务器所有二进制日志
RESET SLAVE

跳过主服务器中给定的事件数
SET GLOBAL sql_slave_skip_counter = N

启动或禁用当前链接的SQL日志
SET sql_log_bin = {0|1}

显示SHOW

显示日志中的事件
SHOW BINLOG EVENTS [IN 'log_name'] [FROM pos] [LIMIT [offset,] row_count]

显示主服务器创建日志列表
SHOW BINARY LOGS

显示当前日志的状态
SHOW MASTER STATUS

显示主服务器注册的从服务器的列表
SHOW SLAVE HOSTS

显示从服务器的线程信息
SHOW SLAVE HOSTS

终止从服务器
STOP SLAVE [thread_types]

复制状态
SHOW PROCESSLIST

主服务器的BinLog Dump线程状态

从服务器的I/O线程状态

从服务器的SQL线程状态

Chap 9存储过程

事件EVENT： 指定的时间和日期执行安排的事件

创建CREATE [DEFINER = { user | CURRENT_USER }] EVENT [IF NOT EXISTS] event_name ON SCHEDULE schedule [ON COMPLETION [NOT]

PRESERVE] [ENABLE | DISABLE | DISABLE ON SLAVE] [COMMENT 'comment'] DO event_body;

显示创建SHOW CREATE EVENT event_name

修改ALTER [DEFINER = { user | CURRENT_USER }] EVENT event_name [ON SCHEDULE schedule] [ON COMPLETION [NOT] PRESERVE] [RENAME TO new_event_name] [ENABLE | DISABLE | DISABLE ON SLAVE] [COMMENT 'comment'] [DO event_body]

显示列表 SHOW EVENTS [FROM database] [LIKE 'pattern' | WHERE expression]

删除DROP EVENT [IF EXISTS] event_name

自定义函数FUNCTION （实为SQL语句集，带返回值）

创建CREATE [AGGREGATE] FUNCTION function_name RETURNS {STRING|INTEGER|REAL|DECIMAL}

显示创建SHOW CREATE FUNCTION func_name

修改ALTER FUNCTION func_name [characteristic ...] characteristic: COMMENT 'string' | LANGUAGE SQL | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA } | SQL SECURITY { DEFINER | INVOKER }

显示函数信息SHOW FUNCTION STATUS [LIKE 'pattern' | WHERE expression]

显示内部代码SHOW FUNCTION CODE func_name

删除DROP FUNCTION function_name

子程序PROCEDURE（SQL语句集）

创建CREATE [DEFINER = { user | CURRENT_USER }] PROCEDURE sp_name ([proc_parameter[,...]]) [characteristic ...] routine_body

显示创建SHOW CREATE PROCEDURE proc_name

修改ALTER PROCEDURE proc_name [characteristic ...]

显示内部代码SHOW PROCEDURE CODE proc_name

显示PROCEDURE的信息SHOW PROCEDURE STATUS [LIKE 'pattern' | WHERE expression]

删除DROP {PROCEDURE | FUNCTION} [IF EXISTS] sp_name

触发器TRIOGGER

创建CREATE [DEFINER = { user | CURRENT_USER }] TRIGGER trigger_name trigger_time trigger_event ON tbl_name FOR EACH ROW [trigger_order] trigger_body

删除DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name

显示触发器列表SHOW TRIGGER [FROM database] [LIKE 'pattern' | WHERE expression]

没有ALTER TRIGGER，只能删除再重建

修改结束字符DELIMITER

DELIMITER character

一般不为反斜线\

复合语句BEGIN....END

使用前一般修改DELIMITER不为‘；’

使用后再修改回来

调用子程序CALL

CALL store_procedure[([parameter[,...]])]

声明DECLARE，用在BEGIN 和END之间

局部变量

DECLARE var_name [, var_name] ... type [DEFAULT value]

条件

DECLARE condition_name CONDITION FOR condition_value condition_value: mysql_error_code | SQLSTATE [VALUE] sqlstate_value

光标

DECLARE cursor_name CURSOR FOR select_statemen

句柄

DECLARE handler_action HANDLER FOR condition_value [, condition_value] ... statement

游标CURSOR（标志数据读的地方）

特性： 1,只读的，不能更新的。 2,不滚动的 3,不敏感的，不敏感意为服务器可以活不可以复制它的结果表

声明DECLARE cursor_name CURSOR FOR select_statemen

打开OPEN cursor_name

读取下一行到变量FETCH cursor_name INTO var_name [, var_name] .

关闭CLOSE cursor_name

预处理prepared statement （执行多个相同的语句，提高性能）

创建PREPARE stmt_name FROM preparable_stmt

执行EXECUTE stmt_name [USING @var_name [, @var_name] ...]

删除DEALLOCATE PREPARE stmt_name

Chap10 聚集子句函数与子查询

聚集函数

特性：

1.当聚集函数遇到错误时，会返回NULL 2.使用时一般包含GROUP BY，若无，则对所有行起作用

位运算

与运算BIT_AND(expr)

或运算BIT_OR(expr)

异或运算BIT_XOR(expr)

返回列的平均值AVG()

AVG （[DISTINCT] column)

DISTINCT 使重复值不计算

返回列的行数count()

COUNT ([DISTINCT] expr)

GROUP_CONCAT() 返回一组非NULL值，由GROUP BY关联

GROUP_CONCAT([DISTINCT] expr [,expr ...] [ORDER BY {unsigned_integer | col_name | expr} [ASC | DESC] [,col_name ...]] [SEPARATOR str_val])

CONCAT(str1,str2,...) 返回结果为连接参数产生的字符串

最大值MAX()

MAX([DISTINCT] expr)

最小值MIXN)

MIN([DISTINCT] expr)

总体标准差STDDEV()

STDDEV(expr)

STD(expr)

STDDEV_POP(expr)

样本标准差STDDEV_SAMP(expr)

方差

基于一个总体挑选的行

VAR_POP(expression)

VARIANCE(expression)

基于总体样本查询的行

VAR_SAMP(expression)

子查询

指嵌套在另外一个SQL语句中的SELECT或其他语句

可以在 SELECT、INSERT、UPDATE 和 DELETE 语句中， 同 =、<、>、>=、<=、IN、BETWEEN 等运算符一起使用。

与JOIN或UNION可以得到相同效果，

但子查询更易读且排错

可以使用BENCHMARK()比较二者的异同

单子段子查询

返回一个标量或单一值

应用情景：1.带有'='的WHERE语句 2.其他合法表达式的单一值

多字段子查询

对多个值匹配，子句使用：(NOT) ALL,ANY(=SOME),EXITS,IN

（=ANY)可替换IN

子查询结果集

在FROM子句中使用子查询可以看成数据库的另外一个表

Chap11 字符串函数

字符集和校对函数

返回字符串的字符集CHARSET(string)

返回逗号最左边的非NULL字符串或列 COALESECE(column[,...])

返回可压缩值CORECIBILITY(string)

返回字符集的排序方式COLLATION(string)

类型转换

返回相应的ASCII码值　ASCII(string)

返回与数字代码值相对应的字符串的值CHAR（与ASCII 刚好相反）

CHAR(ascii[,...]) [USING character_set)

返回二进制字符串BIN

BIN(number)

CONV(N,10,2)

处理二进制字符串 BINARY string !不需要圆括号

将一种数据类型的值转换为另外一种类类型的值CAST

CAST(expr AS type [CHARACTER SET character_set])

将给定的字符串集转换为USING后的指定值
CONVERT(expr USING transcoding_name)

压缩，得到的为二进制字符串，需要编译果zlib
COMPRESS(string_to_compress)

解压缩，从压缩字符串得到原来字符串
UNCOMPRESS(string_to_uncompress)

表示每个给定数目的位EXPORT_SET (大端模式)
EXPORT_SET(bits,on,off[,separator[,number_of_bits]])

返回16进制的数值HEX
HEX(str), HEX(N)

将16进制数值转换成相应字符UNHEX
UNHEX(str)

返回一个由二进制数设置的值列表MAKE_SET

MAKE_SET(bits,str_LSB,str2_MSB,...)，bit为10进制，会自动转换二进制

返回经典算法比较两个类似字符串的结果 SOUNDEX
SOUNDEX(str)

返回最左边字符的ascii字符集的ascii值ORD
ORD(str)

(1st byte code) + (2nd byte code * 256) + (3rd byte code * 2562) ...

格式化函数

将字符串或列连接到一个结果字段中CONCAT，多与AS连用
CONCAT(str1,str2,...)

使用第一个参数字符作为分隔符的CONCAT CONCAT_WS
CONCAT_WS(separator,str1,str2,...)

将字符转换为小写字符
LCASE(string)

LOWER(string)

将字符转换为大写字符
UCASE(string)

UPPER(string)

返回字符串中的字符数，用字节衡量
LENGTH(str)

OCTET_LENGTH(str)

octet是字节的同义词

从字符串的左边位置开始填充字符LPAD
LPAD(str,len,padstr)

去掉空格TRIM

前导空格LTRIM(str)

结尾空格RTRIM(str)

前导和结尾空格TRIM(str)

返回一个以单引号括起来的字符串QUOTE
QUOTE(str)

返回由空格 间隔符的字符串SPACE
SPACE(N)

表达式函数

返回字符串的比特长度BIT_LENGTH
BIT_LENGTH(string)

返回字符串的字符长度
CHAR_LENGTH(string)

CHARACTER_LENGTH(string)

返回32位计算CRC的比特值CRC32
CRC32(string)

查找字符列表的第一个字符并返回数字位置FIELD
FIELD(str,str1,str2,str3,...)

返回第一个参数在一个以逗号分隔符的字符串中的位置FIND_IN_SET
FIND_IN_SET(str,strlist)

返回子字符串在第一个字符串首次出现的位置INSTR
INSTR(str,substr)

返回子字符串在第二个字符串首次出现的位置LOCATE(POSITION)
LOCATE(substr,str)

LOCATE(substr,str,pos)，pos用于指定起始位置
POSITION(substr IN str)

返回N在N1～Nn序列中的位置INTERVAL
INTERVAL(N,N1,N2,N3,...)

搜索列中的字符串MARCH()AGAINST()
MATCH (col1,col2,...) AGAINST (expr [search_modifier])

按照ascii比较字符串STRCMP
STRCMP(expr1,expr2)

返回解压字符串在解压前字符串的长度
UNCOMPRESSED_LENGTH(compressed_string)

提取函数

返回需要数量的 leftmost 最左边字符LEFT
LEFT(str,len)

返回需要数量的 rightmost 最右边字符RIGHT
RIGHT(str,len)

返回指定位置和数量字符串
SUBSTRING(str,pos), SUBSTRING(str FROM pos), SUBSTRING(str,pos,len),

SUBSTRING(str FROM pos FOR len)

SUBSTR = SUBSTRING

MID(str,pos,len)

返回定界符分割的子字符串SUBSTRING_INDEX
SUBSTRING_INDEX(str,delim,count)

读取文件内容，并将文件作为字符串返回LOAD_FILE
LOAD_FILE(file_name)

字符串操纵函数

将最后的参数字符插入到第一个字符中INSERT
INSERT(string, position, length,new_string)

重复多个字符串REPEAT
REPEAT(string, count)

替换列中的旧字符串REPLACE
REPLACE(string, old_element, new_element)

倒置字符串顺序REVEASE
REVERSE(str)

Chap12 日期和时间函数

五种数据类型

DATE 记录日期信息 yyyy-mm-dd

TIME 记录时间 hh:mm:ss

计算时候为hhmmss

记录日期和时间复合型DATETIME yyyy-mm-dd hh:mm:ss

TIMESTAMP 类似DATETIME，但限制较多

起止时间UNIX创始时间 1970-01-01.终结于2037年

YEAR记录年份 yy或yyyy

确定日期或时间函数

以2形式返回当前系统时间
CURTIME()

CURRENT_TIME()

以1形式返回当前系统日期
CURDATE()

CURRENT_DATE()

以3形式返回当前日期时间
CURRENT_TIMESTAMP()

NOW()

返回SQL语句开始执行的时间
LOCALTIMESTAMP()

LOCALTIME()

SYSDATE （）

返回其被调用时间，其表现时间会在NOW之前

计算从UNIX纪元以来的时间数(以秒度量)
UNIX_TIMESTAMP(), UNIX_TIMESTAMP(date)

以1形式返回当前日期的标准日期
UTC_DATE()

以2形式返回当前UTC时间(Universal Time Coordinated)
UTC_TIME()

以3形式返回当前的UTC日期和时间
UTC_TIMESTAMP()

提取并格式化日期或时间函数

从表达式等提取日期DATE
DATE(expre)

根据格式化字符串 格式化日期
DATE_FORMAT(date,format)

从表达式中提取时间值
TIME(time)

更具格式化字符 格式化时间
TIME_FORMAT(time,format)

合并日期和时间为形式3的timestamp
TIMESTAMP(date, time)

返回指定数据类型的格式
GET_FORMAT({DATE|TIME|DATETIME}, {'EUR'|'USA'|'JIS'|'ISO'|'INTERNAL'})

其格式也适用于DATE_FORMAT

返回时间参数

返回指定日期的年份
YEAR(date)

返回指定日期的季度值
QUARTER(date)

range 1 to 4

返回参数中的月份
MONTH(date)

in the range 1 to 12 for January toDecember

返回参数中的月份名
MONTHNAME(date)

返回指定日期在某月中的天数
DAY(date)

DAYOFMONTH(date)

以1形式返回指定日期月份中的最后一天
LAST_DAY(date)

返回一年中的第几周
WEEK(date[,mode]

YEARWEEK(date), YEARWEEK(date,mode)

WEEKOFYEAR(date)

range from 1 to 53.

返回指定日期的工作日
DAYNAME(date)

返回指定日期的工作日索引
DAYOFWEEK(date)

(1 = Sunday, 2 = Monday, ..., 7 =Saturday)

WEEKDAY(date)

(0 = Monday, 1 = Tuesday, ... 6 =Sunday).

返回指定日期在一年中对应的天数
DAYOFYEAR(date)

range 1 to 366

返回给定时间的小时数
HOUR(time)

0 to 23

返回给定时间的分钟数

MINUTE(time)

range 0 to 59

返回给定时间中的秒数

SECOND(time)

range 0 to 59

返回给定时间的微秒数

MICROSECOND(expr)

range from 0 to 999999

从格式化的日期等表达式中提取时间日期信息

EXTRACT(unit FROM date)

把给定的time数值转化为2形式

MAKETIME(hour,minute,second)

把给定的年分和对应天数转化为1形式

MAKEDATE(year,dayofyear)

把给定的秒数返回2形式

SEC_TO_TIME(seconds)

计算和修改日期或时间函数

把日期时间增加或减去一个日期间隔值

DATE_ADD(date,INTERVAL expr unit)

ADDDATE(date,INTERVAL expr unit), ADDDATE(expr,days)

DATE_SUB(date,INTERVAL expr unit)

SUBDATE(date,INTERVAL expr unit), SUBDATE(expr,days)

计较两个日期的日差值(day)

DATEDIFF(expr1,expr2)

形式相同

日期时间增加或减去时间

ADDTIME(expr1,expr2)

exp1可为形式2或3，exp2只能为形式2

SUBTIME(expr1,expr2)

比较两个时间的差值(返回形式2）
TIMEDIFF(expr1,expr2)

形式要相同

增加datetime的时间日期
TIMESTAMPADD(unit,interval,datetime_expr)

MICROSECOND， SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, or YEAR.

比较时间日期的差值
TIMESTAMPDIFF(unit,datetime_expr1,datetime_expr2)

把指定月份加日期
PERIOD_ADD(yearmonth , mon_number)

计算两个日期的月差值

PERIOD_DIFF(yearmonth，yearmonth)
in the format YYMM or YYYYMM

把时区1时间改为其他时区
CONVERT_TZ(dt,from_tz,to_tz)

给定一个day的值，形式1返回日期
FROM_DAYS(N)

标准日历值
FROM_UNIXTIME(unix_timestamp), FROM_UNIXTIME(unix_timestamp,format)

将给定的秒数返回形式2的时间值
SEC_TO_TIME(seconds)

将形式2的时间值转为秒数
TIME_TO_SEC(time)

与SEC_TO_TIME(seconds)相反

暂停SQL的执行(second)
SLEEP(second)

成功执行返回0，否则返回1

Chap13 数学函数

发生错误，所有的数学函数返回NULL

产生0～1随机数
RAND([seed])

弧度与度转换
弧度转为度
DEGREES(num)

度转为弧度
RADIANS

返回pi值
PI

默认5位，可以加掩码0.0000000

三角函数
余弦
COS(num)

反余弦ACOS(num)

num绝对值大于1返回null

正弦
SIN(num)

反正弦ASIN(num)

num绝对值大于1返回null

正切
TAN(num),num为弧度

反正切ATAN(num[,...]

以弧度返回反正切ATAN2(num,num）

与TAN相反

取余数
MOD(N,M), N % M, N MOD M

平方根
SQRT(num)

计算以e为底的幂

EXP(num)

计算对数值

计算对数值

LOG(num[,base]

当没输入base时，等效为LN

计算自然对数值

LN(num）

计算以2为基数的对数值

LOG2(num)

计算以10为基数的对数值

LOG10(num)

返回绝对值

ABS(number）

判断正负数

SIGN(num)

负数为-1，正数为1，零为0

返回 expr 的二进制表达式中”1“的个数

BIT_COUNT(expr)

取近似值

四舍五入取值

FORMAT(num,decimal)

ROUND(num,[precission])

取有效值

TRUNCATE(num, decimal)

不会四舍五入

浮点数的向上取整

CEIL(num)

CEILING(num)

浮点数向下取整

FLOOR(num)

数制转换

CONV(num,from_base,to_base)

返回最大值

GREATEST(value1,value2,...)

返回最小值

LEAST(value1,value2,...)

IP与整数转换

将IP地址转为整数值

INET_ATON(expr)

SELECT INET_ATON('209.207.224.40'); -> 3520061480 产生的数字总是按照网络字节顺序。如上面的例子，数字按照 209×2563 + 207×2562 + 224×256 + 40 进行计算。

大端模式存储

将整数值转为IP地址

INET_NTOA(expr)

Chap14 控制流程函数

CASE

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

IF

IF(condition,result1,result2)

condition 不为0或null，返回result1；否则，返回result2

IFNULL

IFNULL(condition,result)

condition为null,返回result；否则，返回condition

ISNULL

ISNULL(expr)

expr为NULL，返回1；否则，返回0

NULLIF，两参数相同返回NULL

NULLIF(expr1,expr2)

服务器和客户端工具

Chap15 服务器俄客户程序

配置文件order by /etc/my.cnf /etc/mysql/my.cnf ~/.my.cnf

客户端

mysql options [database]

man mysql

服务器

mysqld [options]

mysqld --verbose --help

在配置文件中应该使用长表并忽略前面的双划线

如 命令行中 --basedir=/data/mysql ,配置文件为basedir=/data/mysql

存储引擎(表类型)

MyISAM -默认引擎
InnoDB

运行多个MySQL服务器 mysqld_multi

mysqld_multi [options] {start|stop|report} [GNR[,GNR] ...]

可以在不同的socket和端口

安全性推荐工具 mysqld_safe

mysqld_safe options

Chap16 命令行使用工具

检测，修复MyISAM myisamchk

myisam options table[.MYI] [...]

直接作用于表文件

检查用户的访问权限 mysqlaccess

mysqlaccess [host [user [database]]] [options]

允许从命令行执行MySQL服务器管理任务mysqladmin

mysqladmin [option] command [command_option]

检查，修复MyISAM表 mysqlcheck
mysqlcheck [options] database [table]

z直接与MySQL服务器交互

导出数据和表结构 mysqldump
mysqldump [OPTIONS] database [tables]

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

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

将数据导入指定的数据库musqlimport
mysqlimport [option] database filename [...]

获得数据库、表等信息列表 mysqlshow
mysqlshow [option] [database [table [column]]]

模拟多用户并发，用以检测系统时序mysqlslap
mysqlslap [options] database

将C API 转换为MySQL相应的函数msql2mysql
msql2mysql program.c

不会创建原文件的副本

也不能转换所有的mSQL函数

转换存储引擎mysql_convert_table_format
mysql_convert_table_format options database

创建存放已有表信息的表 mysql_tableinfo
mysql_tableinfo options new_database [existing_database [existing_table]

更新版本 mysql_upgrade
mysql_upgrade options

原始版本 mysql_fix_privilege_tables ，功能少

服务器运行时备份数据库mysqlhotcopy
mysqlhotcopy database [path]

仅作用于MyISAM和ISAM

配置解析文件，将键值换为命令行对应的选项my_printf_defaults
my_printf_defaults options filename

编译错误代码与可用格式映射的文件comp_err
  comp_err source destination

查找包含SQL语句的文本文件 mysql_find_rows
  mysql_find_rows options filename

扫描提取myisam.log 信息 myisamlog
  myisam options [filename [table ...]]

创建制度压缩表 myisampack
  myisampack optios /path/table[,MYI]

终止指定模式的进程 mysql_zap
  mysql_zap [options] pattern

等待进程结束mysql_waitpid
  mysql_waitpid options PID wait_time

查询并替换简单文件中的文本replace
  replace options filename

将MyISAM 表文件扩展名由小写转为大写 mysql_fix_extensions
  mysql_fix_etensions path

提交出错信息bug mysqlbug
  mysqlbug

显示慢速查询日志的摘要mysqldumpslow
  mysqldumpslow [options] [filename]

显示收到的系统错误代码描述信息perror
  perror [options] code

将主机名转为ip地址resolveip
  resolveip [options] host ...

将地址和数字数据分解到指定的栈resolve_stack_dump
  resolve_stack_dump options symbols_filename [numeric_dump_file]

从源码创建二进制发行版
  make_binary_distrubution

显示与MyISAM表的FULLTEXT索引信息myisam_ftdump

myisam_ftdump options table index_nbr

C API

c 编程 include 编译

gcc -I/usr/include/mysql *.c -L/usr/lib/mysql -lmysqlclient -o *

http://www.cnblogs.com/awy-blog/p/3390514.html

gcc $(mysql_config --cflags) xxx.c -o xxx $(mysql_config --libs)

http://www.2cto.com/database/201203/123227.html

数据类型

MYSQL

mysql_init 创建的句柄结构，可使用mysql_close ()释放内存

MYSQL_RES

查询语句获取的查询结果集的结构，可被取回函数使用，mysql_free_result()释放内存

MYSQL_ROW

存放结果集的行数据。可使用mysql_fetch_row()检索结构中的数据

MYSQL_FIELD

存放结果集的字段信息。通过mysql_fetch_field()创建存放字段信息的数组，数组元素包括缺省的name、table 、以及def

MYSQL_FIELD_OFFSET

记录结果集的指针偏移位置。 mysql_row_tell()可取得偏移量，可被mysql_row_seek()使用

my_ulonglong

可变类型，用于存放函数获取的行数。 可用来打印变量的值，并复制到其他无符号长整型变量中。

可以由mysql_affected_rows()、mysql_num_rows()、 mysql_insert_id() 创建

函数

分配并初始化MYSQL对象mysql_init

MYSQL *mysql_init(MYSQL *mysql)

int mysql_library_init(int argc, char **argv, char **groups）可以完成相同功能，可用于多线程

配置连接选项 mysql_options

int mysql_option(MYSQL *mysql, enum mysql_option option, const char *value)

建立MYSQL 连接

msql_real_connect

MYSQL *mysql_real_connect(MYSQL *mysql, const *host, const char *user, const char *password, const char *user, const char *password, const char *

database, unit port, const char *user, const char *password, const char *database, unit port, const char *unix_socket, uint flag)

地址-连接主机名，用户名，用户密码- 数据库名称，端口- 套接字-客户机标记

mysql_connect

MYSQL *mysql_connect(MYSQL *mysql, const *host, const char *user, const char *password)

更改会话用户mysql_change_user()

mysql_change_user(MYSQL *mysql, const char *user, const char *password, const char *database)

选择其他数据库mysql_select_db

int mysql_select_db(MYSQL *mysql, const char *database)

成功返回0

执行SQL查询语句，成功返回0

int mysql_query(MYSQL *mysql, const char *query)

int mysql_real_query(MYSQL *mysql, const char *query, unsigned int length)

可查询二进制数据，可使用strlen确定字节数

返回最近查询中受影响的记录行数mysql_affected_rows

my_ulonglong mysql_affected_rows(MYSQL * mysql）

INSERT、UPDATE、DELETE语句有效，而SELECT返回为0

创建新的数据库mysql_create_db {不推荐}

int mysql_create_db(MYSQL *mysql, const char *database)

推荐在mysql_query() 使用 DROP DATABASE

MYSQL 线程

初始化mysql_thread_init

my_bool mysql_thread_init(void)

被自动调用，成功返回0

返回线程标识码mysql_thread_id

unsigned long mysql_thread_id(MYSQL *mysql)

确定是否线程安全mysql_thread_safe

unsigned int mysql_thread_safe(void)

安全返回1，否则返回0

释放内存 mysql_thread_end
void mysql_thread_end(void)

没有返回值，也不能自动调用

关闭服务器连接mysql_close()
void mysql_close (MYSQL *mysql)

void mysql_library_end(void)

关闭服务器mysql_shutdown
int mysql_shutdown(MYSQL *mysql)

成功返回0

释放函数分配的内存 mysql_free_result

void mysql_free_result(MYSQL_RES *result)

删除数据库mysql_drop_db {不推荐}
int mysql_drop_db(MYSQL *mysql, const char *database)

推荐在mysql_query() 使用 CREATE DATABASE

结束服务器上运行的线程mysql_kill
int mysql_kill(MYSQL &mysql, unsigned long identifier

判断连接是否处于开启状态mysql_ping
int mysql_ping(MYSQL *mysql)

返回服务器上所有数据库的集合mysql_list_dbs
MYSQL_RES *mysql_list_db(MYSQL *mysql, const char *wild)

wild指定表达式，可用通配符%或_来

返回指定表的所有字段mysql_list_fields
MYSQL_RES *mysql_list_fields(MYSQL *mysql, const char *table, const char *wild)

返回服务器上所有进程或服务器线程 mysql_list_processes
MYSQL_RES *mysql_list_processes(MYSQL *mysql)

返回数据库中的所有表 mysql_list_tables
MYSQL_RES *mysql_list_tabl(MYSQL *mysql, const char *expression)

expression指定表达式，可用通配符%或_来

返回最后一次调用API函数的出错信息
char *mysql_error (MYSQL *mysql)

成功执行返回空字符串

返回最后一次调用API函数的出错代码
unsign int mysql_errno(MYSQL *mysql)

成功执行返回0

返回前面查询遇到的警告信息数mysql_warning_count
unsigned int mysql_warning_count(MYSQL *mysql)

返回结果集中的相关信息
MYSQL_FIELD *mysql_fetch_field(MYSQL_RES *result)

MYSQL_FIELD *mysql_fetch_field_direct (MYSQL_RES *result, unsigned int fieeld_nbr)可指定字段

返回结果集的一个数组 MYSQL_FIELD *mysl_fetch_fields(MYSQL_RES *result)

返回结果集某一行中每一列的长度 unsigned long *mysql_fetch_lengths(MYSQL *result)

取结果集下一行 MYSQL_ROW mysql_fetch_row(MYSQL_RES *result)

返回结果集中的列数 unsigned int mysql_field_count (MYSQL *mysql)

将结果集中当前字段的指针移动到偏移处 MYSQL_FIELD_OFFSET mysql_field_seek(MYSQL_RES *result, MYSQL_FIELD_OFFSET offset)

与mysql_fetch_field连用

返回取结果函数 MYSQL_FIELD_OFFSET mysql_field_tell(MYSQL_RES *result)

读取并储存结果集 mysql_store_result
MYSQL_RES *mysql_store_result (MYSQL *mysql)

一次检索并储存所有数据

读取查询结果，每次一行mysql_use_result
MYSQL_RES *mysql_use_result (MYSQL *mysql)

适用结果集庞大且对处理速度要求高

检测结果集是否结束

检测结果集是否有更多的结果 mysql_more_result
mybool mysql_more_result(MYSQL *mysql)

有更多结果返回1，否则返回0

从结果集中读取下一行数据mysql_next_result

int mysql_next_result(MYSQL *mysql)

执行成功返回0，检索成功但没有更多结果行返回-1，结果集没有数据返回则失败，返回非零值

返回结果集中每一行的字段数mysql_num_fields
unsigned int mysql_num_fields(MYSQL_RES *result)

返回结果集中的行数mysql_num_rows
int mysql_num_rows(MYSQL_RES *result)

刷新缓存和数据表，重置备份服务器mysql_refresh
int mysql_refresh(MYSQL *mysql, unsigned int options)

成功返回0

返回c/s信息

返回客户机库的版本mysql_get_client_info
char *mysql_get_client_info(void)

返回数字格式的客户机库版本 mysql_get_client_version
unsigned long *mysql_get_client_version(void)

返回主机名和当前连接类型mysql_get_host_info
char *mysql_get_host_info (MYSQL *mysql)

返回当前连接的协议版本 mysql_get_proto_info
unsigned int mysql_get_proto_info(MYSQL *mysql)

返回数据库的默认字符集信息mysql_get_character_set_info
void mysql_get_character_set_info(MYSQL *mysql, MY_CHARSET_INFO *cs)

返回服务器上连接的MYSQL的版本mysql_get_server_info
char *mysql_get_server_info(MYSQL *mysql)

返回服务器当前连接的MYSQs数值格式的版本信息 mysql_get_server_version
unsigned long mysql_get_server_version(MYSQL *mysql)

返回给出的信息字符串mysql_info
char *mysql_info(MYSQL *mysql)

仅作用于5类SQL语句： INSERT, INTO…SELECT, INSERT INTO… VALUES…， LOAD DATA INFLIE, ALTER TABLE以及UPDATE。 其他语句会返回NULL

重新装载授权表mysql_reload
int mysql_reload(MYSQL *mysql)

不推荐，可在mysql_query使用FLUSH PRIVILEDES

成功返回0

提交当前事务mysql_commit
my_bool mysql_commit (MYSQL *mysql)

创建跟踪调试文件mysql_debug
void mysql_debug(const char *debug)

确定是否取回了最后一行记录mysql_eof
my_bool mysql_eof(MYSQL *result)

没有执行到最后一行返回0，否则返回一个非零值

返回当前连接下，使用INSERT插入最后一条记录的主键标识码mysql_insert_id
my_ulonglong mysql_insert_id(MYSQL *mysql)

表中必须有AUTO_INCREMENT,否则返回0

返回当前连接最后一次发生错误的错误代码mysql_sqlstate
const char *mysql_sqlstate(MYSQL *mysql)

回退当前事务mysql_rollback
my_bool mysql_rollback(MYSQL *mysql)

成功返回0

启用或禁用自动提交模式mysql_autocommit
my_bool mysql_autocommit(MYSQL *mysql, my_bool mode)

自动提交模式可令服务器在增删改语句后自动更新

返回正在使用的默认字符集名称mysql_character_set_name
const char *mysql_character_set_name (MYSQL *mysql)

将取回结果的当前指针移动到指定的偏移位置mysql_data_seek
void mysql_data_seek(MYSQL_RES *result, my_ulonglong offset)

将出错信息写进MySQL服务器日志文件mysql_dump_debug_info
int mysql_dump_debug_info（MYSQL *mysql)

成功返回0，失败返回非零值

返回代SSL密码名称的字符串mysql_get_ssl_cipler
const char *mysql_get_ssl_cipher(MYSQL *mysql)

使用SSL建立一个安全连接mysql_ssl_set

my_bool mysql_ssl_set(MYSQL *mysql, const char *key_path, const char *cert_path, const char *ca_path, const char *pem_path, const char *cipher)

成功返回0

将十六进制字符串转为SQL语句格式 mysql_hex_string

unsigned long mysql_hex_string (char *to,const char *from, unsigned long length)

为当前连接设置默认字符集mysql_set_character_set

int mysql_set_character_set((MYSQL *mysql, const char *char_set)

成功返回0