create table orders(

id int not null auto\_increment primary key COMMENT '订单号',

cid int ,

rid varchar(6),

price int default 20,

foreign key(cid) references clients(cid),

foreign key(rid) references rooms(rid)

)default charset=utf8;

auto\_increment必须是主键，插入时insert into orders values(null, 4, '312', 96)

填null就可以。自动设置为id列最大值+1

ENGINE=InnoDB DEFAULT CHARSET=utf8;

comment注释

UPDATE tbl SET title='学习 C++' WHERE runoob\_id=3;

增加字段：

alter table 表名 ADD 字段 类型 约束 [默认值 注释]

ALTER TABLE video ADD category\_id int(11) unsigned not null DEFAULT '0' COMMENT '视频分类id';

在column1后增加字段

ALTER TABLE 表名 ADD 字段名 字段类型 AFTER column1;

在表的开始位置增加字段

ALTER TABLE user ADD id INT(10) FIRST;

删除字段

alter table t1 drop column age;

修改字段名：

alter table 表名 rename column A to B

ALTER TABLE video RENAME COLUMN category\_id TO cid;

修改字段类型：

alter table 表名 modify column 字段名 类型 约束 [默认值, 注释];

ALTER TABLE video MODIFY COLUMN category\_id smallint(5) unsigned not null DEFAULT '0' COMMENT '视频分类id';

修改字段默认值

alter table 表名 alter column 字段名 drop default; --(若本身存在默认值，则先删除)

alter table 表名 alter column 字段名 set default 默认值; --(若本身不存在则可以直接设定)

ALTER TABLE video ALTER COLUMN sort SET DEFAULT '50';

修改表名

alter table user rename new\_user

删除表内数据

delete from student where T\_name = "张三";

清除表内数据，保存表结构

truncate table student;

删除表

drop table student;

* 查看数据库**编码格式**

mysql> show variables like 'character\_set\_database';

* 查看数据表的编码格式

mysql> show create table <表名>;

* 创建数据库时指定数据库的字符集

mysql>create database <数据库名> character set utf8;

* 修改数据库的编码格式

mysql>alter database <数据库名> character set utf8;

* 修改数据表格编码格式

mysql>alter table <表名> character set utf8;

* 修改字段编码格式

mysql>alter table <表名> change <字段名> <字段名> <类型> character set utf8;

mysql>alter table user change username username varchar(20) character set utf8 not null;

我建表的时候没有设置默认编码为utf8，把数据库、表格、字段的编码格式都改了以后才生效

* MySQL 获得当前时间戳函数：current\_timestamp, current\_timestamp()

SELECT NOW(),CURDATE(),CURTIME()

* CREATE **TRIGGER** trigger\_name trigger\_time trigger\_event ON tb\_name FOR EACH ROW trigger\_stmt

trigger\_name：触发器的名称

tirgger\_time：触发时机，为BEFORE或者AFTER

trigger\_event：触发事件，为INSERT、DELETE或者UPDATE

tb\_name：表名

trigger\_stmt：触发器的程序体，可以是一条SQL语句或者是用BEGIN和END包含的多条语句

所以可以说MySQL创建以下六种触发器：

BEFORE INSERT BEFORE DELETE BEFORE UPDATE

AFTER INSERT AFTER DELETE AFTER UPDATE

DELIMITER //

CREATE TRIGGER t1 BEFORE INSERT ON client FOR EACH ROW

BEGIN

END //

DELIMITER ;

* 查看触发器

SHOW TRIGGERS;

* 删除触发器

DROP TRIGGER t1;

存储程序：

1.存储过程procedure

2.存储函数function

创建存储过程

CREATE PROCEDURE sp\_name ( [proc\_parameter] ) routine\_body

proc\_parameter参数列表

形式为[ IN | OUT| INOUT] param\_name type

mysql> DELIMITER //

mysql> Create procedure p1(OUT param1 INT)

-> begin

-> select count(\*) INTO param1 from clients;

-> end //

Query OK, 0 rows affected (0.39 sec)

mysql> dilemiter;

delimiter 作用将mysql结束符设置为//注意是斜杠不是反斜杠。因为mysql默认语句结束符号为 ; ，为了避免与存储过程中SQL语句冲突，需要使用Delimiter改变储存过程的结束符，以end //结束存储过程。delimiter; 恢复默认结束符

创建存储函数

DELIMITER //

CREATE FUNCTION p2(id INT)

RETURNS CHAR(10)

BEGIN

RETURN (SELECT name FROM CLIENTS WHERE cid=id);

END //

delimiter ;

调用存储过程

CALL p1( 100, num);

num如果是一个OUT的参数，结果就存在num里了

调用存储函数

SELECT p2(101);

删除

DROP FUNCTION p2;

DROP PROCEDURE p1;

可以在存储过程中使用的局部变量

DECLARE var1 int DEFAULT 100;

SET var1=200;//使用Set赋值

SELECT cid INTO var1 FROM clients where name='zhangxianseng';

光标Cursor，网上叫游标

DECLARE c1 CURSOR FOR SELECT id, name FROM clients;

打开光标

OPEN c1;

使用

FETCH c1 INTO var1;

关闭光标

CLOSE c1

貌似只能在函数中使用

流程控制语句

* IF var1 IS NULL

THEN SELECT 'var1 is null';

ELSE SELECT 'var1 is not null';

END IF;

* CASE var1

WHEN 100 THEN SELECT 'var1 is 100';

WHEN 200 THEN SELECT 'var1 is 200';

ELSE SELECT 'var1 is not 100 or 200';

END CASE

DECLARE id INT DEFAULT 0;

* l1 : LOOP

SET id=id+1;

IF id>=10 THEN LEAVE l1;

END LOOP l1;

leave 可以退出任何被标注的流程控制构造

ITERATE label;

出现在LOOP、REPEAT、WHILE语句内。相当于continue应该

Mysql变量

1.SET @sum=0;

会话变量。在当前客户端中有效

变量名必须是@开头

2.DECLARE var1 INT DEFAULT 199;

局部变量，只能用在begin和end之间，end之后没了

变量名不能有@

1. 数据定义语言(DDL ，Data Defintion Language)语句：数据定义语句，用于定义不同的数据段、数据库、表、列、索引等。常用的语句关键字包括create、drop、alter等。
2. 数据操作语言(DML ， Data Manipulation Language)语句：数据操纵语句，用于添加、删除、更新和查询数据库记录，并检查数据的完整性。常用的语句关键字主要包括insert、delete、update和select等。
3. 数据控制语言(DCL， Data Control Language)语句：数据控制语句，用于控制不同数据段直接的许可和访问级别的语句。这些语句定义了数据库、表、字段、用户的访问权限和安全级别。主要的语句关键字包括grant、revoke等。

**索引**

索引用于快速找出在某个列中有一特定值的行。

索引越多写入，修改的速度越慢。因为写入修改数据时也要修改索引

* 普通索引

创建索引

CREATE INDEX indexName ON mytable(username);

修改表结构(添加索引)

ALTER table tableName ADD INDEX indexName(columnName)

删除索引的语法

DROP INDEX [indexName] ON mytable;

* 唯一索引

索引列的值必须唯一，但允许有空值。如果是组合索引，则列值的组合必须唯一

创建索引

CREATE UNIQUE INDEX indexName ON mytable(username(length))

修改表结构

ALTER table mytable ADD UNIQUE [indexName] (username)

* 全文索引

全文索引是目前搜索引擎使用的一种关键技术。它能够利用"分词技术"等多种算法智能分析出文本文字中关键字词的频率及重要性，然后按照一定的算法规则智能地筛选出我们想要的搜索结果

创建article表，其中有主键ID(id)、文章标题(title)、文章内容(content)三个字段

在title和content两个列上创建全文索引

CREATE TABLE article (

id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,

title VARCHAR(200),

content TEXT,

FULLTEXT (title, content)

);

在article表的title和content列中全文检索指定的查询字符串的SQL语句：

SELECT \* FROM article WHERE MATCH(title, content) AGAINST('查询字符串')

* MySQL自带的全文索引只能用于数据库引擎为MyISAM的数据表。此外，MySQL自带的全文索引只能对英文进行全文检索，目前无法对中文进行全文检索。
* 目前，使用MySQL自带的全文索引时，如果查询字符串的长度过短将无法得到期望的搜索结果。MySQL全文索引所能找到的词的默认最小长度为4个字符。另外，如果查询的字符串包含停止词，那么该停止词将会被忽略。
* 尽量先创建表并插入所有数据后再创建全文索引，而不要在创建表时就直接创建全文索引，因为前者比后者的全文索引效率要高。
* 主键索引

添加主键的同时自动添加索引

ALTER TABLE money ADD PRIMARY KEY(id);

创建表时声明索引：  
INDEX [索引名] (字段)  
FULLTEXT [索引名] (字段)  
UNIQUE[索引名] (字段)

中括号中的索引名，代表可选

CREATE TABLE test (  
    id INT NOT NULL ,   
     username VARCHAR(20) NOT NULL ,   
     password INT NOT NULL ,   
     content INT NOT NULL ,   
     PRIMARY KEY (id),   
     INDEX pw (password),   
     UNIQUE (username),  
    FULLTEXT (content)  
 )

显示索引信息

SHOW INDEX 命令列出表中的相关的索引信息。添加 \G 格式化输出信息。

SHOW INDEX FROM table\_name; \G

查询单个字段不重复记录

select distinct age from student;

* 结果集排序

select 字段 from 表 order by 字段 排序关键词

select name, age, id from student order by id desc;

asc 升序 ascend

desc 降序排列 descend

select id, name, age from student order by id desc, age asc;

id一样就根据age排

* 结果集限制

select \* from students limit 5;

显示5个学生

select id,name from student order by balance desc limit 5;

* 结果集区间选择

select 字段 from 表 limit 偏移量,数量

select \* from student limit 0,3;

从第一条开始取三条记录

* 第一条记录为0。

select \* from student limit 3,3;

* 统计类函数
* sum
* count
* max
* min
* avg

mysql> select count(id) as zongshu from money;

+---------+

| zongshu |

+---------+

| 12 |

+---------+

1 row in set (0.00 sec)

* 分组

select \* from student group by age;

按照年龄分组

年龄字段相同的不显示

mysql> select count(1), age from student group by age;

统计不同年龄学生的人数

count(1)和count(\*)和count(999)都显示一样

如果student有字段score，可以为null，那么

select count(\*) from student; /\* 5 \*/

select count(score) from student; /\* 3 \*/

这样写不统计score为null的记录

select count(\*), name from student group by age with rollup;

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

| count(\*) | name |

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

| 9 | 林然 |

| 3 | Jennifer |

| 12 | Jennifer | /\*加个with rollup多了一行这个, 统计总数\*/

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

* 结果过滤

having 是筛选组,而where是筛选记录

select count(\*) as result, age from student group by age having result > 2;

顺序

SELECT

[字段1 [as 别名1],[函数(字段2) ,]......字段n]

FROM 表名

[WHERE where条件]

[GROUP BY 字段]

[HAVING where\_contition]

[order 条件]

[limit 条件]