如果是默认的校对规则，mysql是不会显示的

比较排序时，校对规则会影响排列顺序

show collation ;

设置字符集有两种方式:

1.alter table table\_name character set gbk;

2.alter table table\_name charset=gbk;

数据类型：

1.整型

tinyint 1字节

smallint 2字节

mediumint 3字节

int 4字节

largeint 8字节

1字节=8位，2^8=256 表示有256中组合，所以可以表示有符号整数-128-127，（在计算机内部是以补码形式存放数据，-0的位置用来存放-128）

这里的范围是指 存放数的范围，如果是字符，他是按照字节来存放的，比如char(33)可以存放33个字符

tinyint int 默认是有符号的，unsigned

如果存放的数据超过字段所定义的数据类型的范围，则保存的是最大的那个数

定义显示宽度：

int(4) tinyint(3) zerofill这里的3 4 就是表示字段的最小宽度（如果数据的宽度大于这里定义的宽度，他会原封不动的显示，他并不会改变源数据），zerofill前置0 填充

效果图如下：

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

| id | name | sex |

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

| 0001 | zmh | 男 |

| 0127 | zmh | na |

| 0002 | zmh | na |

| 0003 | zmh | na |

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

浮点型：

精度概念：从左边第一个不为零开始，有多少位

单精度 float 4字节 精度越高，存放的数据越准确

分成两部分 精度空间+10^5 在磁盘上是这样存放的

浮点数是计算机算出来的，123.456在计算机中存放的是：123456 3 ->0.123456\*10^3,就是取出精度部分经过计算得出0.123456，再乘10^3,所谓精度的位数就是精度部分可以存放多大范围的整型数据

float只能保存5-6位有效精度位数，double可以保存15-16位有效精度位数

| a | float | YES | | |

| b | double | YES |

| a | b |

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

| 1234570000 | 1234567890.0123458 |

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

双精度 double 8字节

Type(M,D)M 表示总位数，D表示小数位数，M-D表示整数位数

这样来控制范围，比如：float(5,2)表示的范围是：-999.99-999.99 整数部分只能有三位，如果超过范围，他会警告，然后插入最近进的最大数

支持科学计数法 1.234E2

定点数 小数点是固定的，每9位用4字节来保存 ，他是原封不动的保存，不存在精度损失（他是用变长字节来保存数据的，到底用多大字节取决于数据的大小） decimal

单/双精度都会存在精度损失

decimal(M,D)小数部分超过范围按照四舍五入存放数据

小数也支持zerofill

时间型：

create table dt\_1(a datetime,b timestamp);

对于时间戳：显示时和datetime一样，保存时，是以整型int 4字节存放的

插入时，使用时间的字符串形式，'2016-12-30 10:40:30'

时间戳是只能保存‘1970-01-01 00:00:00’到2038-01-19 03:14:07 范围的时间

因为他是四个字节的，超过这个范围是存不进去的

datetime的范围比时间戳大，可以用来做万年历

检索时，通过+0 可以显示时间戳

datetime和 timestamp支持任意分隔符

time类型

1.表示当前时间

2.表示一个时间间隔

字符串类型：

char(M) M是固定长度，M直接就是字符的个数

char与varchar的区别：

char()的效率高，varchar()空间优化的好，因为varchar要先计算字符串的长度，在开辟空间

char一般用于字符长度固定的比如手机号，身份证号，varchar用于存放可变长度的，比如姓名，地区

varchar(M) M是允许的最大长度，是一个字节范围，还要根据不同字符编码计算字符的个数

记录的总长度也是有限制的65535，他会保留两三个字节来保存一些额外的信息，比如null也是占用空间的，如果没有设置not null 那么他就会占用三个字节来保存null信息

所以varchar的真实长度是65533或者65532

char的真实长度是255，这里的长度是指字符个数还是字节数呢？

答案应该是字节数，对于不同的字符集，一个字符占用的字节数是不一样的，如果是gbk，一个汉字占两个字节长度，如果是utf8，一个汉字占用三个字节

所以varchar(M)能保存的字符个数就是：M字节数/该字符集一个字符占用的字节数

如果varchar中的M值超过65532，那么他会自动转成mediumtext类型

create table demo4 (a varchar(65533) )character set latin1;

/\*这样是不能创建成功的，超过了记录的最大长度\*/

create table demo5 (a varchar(65533) not null)charset=latin1;

/\*这里加多了not null 就可以正常创建了\*/

latin1字符集是一个字符占用一个字节的

如果是char(256)则会创建失败

create table demo9 (a char(255) )charset=latin1;

为什么是65533或者65532 ？

因为 存放null需要一个字节，需要两个字节来保存记录的长度，用于计算。注意这里记录的长度是指所有字段的总长度

有时候需要所有字段都为 not null 才符合你想插入的个数，只有所有的字段都设置为not null 这一个字节才可以省略

失败

create table demo10 (a varchar(65532),b tinyint)charset=latin1;

成功

create table demo10 (a varchar(65532) not null ,b tinyint not null)charset=latin1;

整条记录的所有null都保存在一个字节中

一次删除多张表

drop table demo2,demo3,demo4,demo5,demo6,demo7,demo8,demo9,demo10;

text类型，相当于varchar的最大长度 2^16 65535字节 他所有的字节都可以用来保存数据，不用指定长度，可以自己指定

text 也有tinytext mediumtext longtext

text类型字段的数据是不会被统计到记录总长度中去的

枚举类型：相当于单选

create table demo1(gender enum('female','male'));

如果插入的不是里面的内容，则会插不进去

枚举类型看似 字符串，实际上是以整型形式存放，按顺序从1开始

集合类型： set 相当于多选 可以表示64中组合，因为他占用8字节，共64位

create table demo2(hobby set('music','篮球','足球'));

insert into demo2 values('music,篮球');

mysql> select hobby+0 from demo2;

+---------+

| hobby+0 |

+---------+

| 3 | 集合下标也是从1开始，这里返回一个下标之 和，表示所选内容

文本编辑器只能显示字符流的文件，想图片、视频等二进制文件是以字节流形式存放的，文本编辑器找不到对应的字符编码来对应显示，所以产生一堆乱码

如何保存二进制文件？

binary varbin 但是实际上是不会这样做的！如果插入图片，只需保存图片的路径即可

尽量精确，能用小的尽量用小的

列属性：

not null default '';

所有的字段都可以设置默认值；

主键：为了提高检索速度，可以将唯一的字段设置成字段，比如学号，ID号...

利用一个与实体信息不相关的属性作为唯一标识，他与业务逻辑不相关，只是用来标识记录

主键必须唯一，而且不能为null

可以将两个字段组成为一个主键，比如 primary key(name,class)这样就是name和class组合作为一个唯一主键，而不是将两个字段都设置成字段

修改主键：先删除原来主键 alter table table\_name drop primary key;

然后新建主键： alter table table\_name add primary key(id);

自动增长：插入记录是自动将字段值加1。

如何添加auto\_increment?

设置自动增长的初始值

alter table table\_name auto\_increment 10;从10开始自增长

使用delete只是删除了表的内容，那条记录还是占用了原来的空间，可以往这个空间重新插入新的记录

第十五讲

实体：每个实体对应数据库的一条记录

记录与记录之间的关系就是实体间的对应关系

对应关系有：一对一 多对一 一对多 多对多

一对一：一条记录的主键值 等于 另一张表另一条记录的主键值 这是优化表最常用的方法，称之为垂直分割，也就是把本来属于同一张表，同一条记录的内容分到另外一张表中 如果一条记录中的记录较多，就应该将其拆分为 常用 和 不常用 两大类，再进行垂直分表 只要每个表有相同的主键即可

一对多：一个班级对应多个学生 典型设计方案：在多的一端（学生表）增加一个字段，用来表示少的表（班级表）的主键值

多对多：老师和班级的对应关系，一个老师教多个班级，一个班级被多个老师教 也就是老师和班级是多对多的关系 设计：借助中间表，表示这两张表之间的关系，从而实现将多对多拆分成一对多 在多的一端有个字段指向另一张表的主键

多对多关系的典型设计方法：利用一个中间表

比如老师和班级之间的对应关系，就是典型的多对多；

把两个表中的主键抽出来，放在中间表中

第十七讲

外键（foreign key）约束：一个实体去约束另一个实体

概念：如果一个实体的某个字段指向另一个实体的主键，那么这个实体的这个字段就称为另一个实体的外键（他是在另一个实体外部的）

外键的语法：foreign key(cid) references class(id);

被指向的实体称为主表、父表，负责指向的实体称为从表、字表。

作用:用于约束处于关心内的实体

增加子表记录时，是否有与之对应的父表记录,如果没有则不能创建成功

在删除或者更新主表记录时，字表应该如何处理相关记录

例子：

班级表

drop table if exits class;

create table class(

class\_id int primary key auto\_increment comment'班级号',

class\_name varchar(12) not null default 'itclass'

)character set utf8;

学生表

drop table if exits student ;

create table student (

stu\_id int primary key auto\_increment,

stu\_name varchar(12) not null default '' comment '学生名字',

class\_id int ,

foreign key(class\_id) references class(class\_id)

)character set utf8;

insert into student values(null,'朱明浩',1);

以上记录不能插入成功，因为主表中没有class\_id=1 的班级，只有在class表中插入一条class\_id=1的记录才能在student字表中插入以上记录

如果子表中还有记录指向父表的记录，那么父表是无法更新或者被删除的

也就是：只要还有学生，就一定要有相应的班级

可以在主表中插入从表没有的记录，但从表中不能插入主表没有的记录

设置级联操作

在主表数据发生改变时，与之关联的从表数据应该如何处理

主表更新：on update 只有更新主表的主键值时，从表的相关字段才会改变

主表删除：on delete

允许的级联动作：

cascade：如果主表被更新或删除，那么从表也就会执行相应的操作

set null:设置为null表示从表不指向任何主表级联

Restrict： 拒绝主表的相关操作

先删除外键，才能新建外键

alter table student drop foreign key student\_ibfk\_1;

alter table student add foreign key(class\_id) references class(class\_id) on delete set null ;/\*通过这样设置，再删除主表记录时，从表的class\_id字段将会置为null，前提是该字段允许为null\*/

外键名称可以在创建时指定，也可以由系统自动生成

以下是系统自动生成的外键名

CONSTRAINT `student\_ibfk\_1` FOREIGN KEY (`class\_id`) REFERENCES `class` (`clas

s\_id`)

如果是cascade呢？

先来维护一下数据

要主表中有class\_id，才能在从表中插入

insert into class values(1,'php基础班');

update student set class\_id=1 where stu\_id=2;

alter table student drop foreign key student\_ibfk\_1;

alter table student add foreign key(class\_id) references class(class\_id) on delete cascade ;

结果是：从表的所有外键相关记录随着主表记录的删除而删除了

第十九讲

存储引擎

表是需要生成一个文件存储在磁盘上，不同的文件有不同存储的结构，这就是存储引擎

其实存储引擎是表的数据类型，与列类型类似

不同的存储引擎，功能是不一样的

默认的存储引擎是Innodb支持外键，外键不是必须的，也可以通过程序来控制，以达到一种外键的关系

innoDB&myisam：

数据和索引的保存的文件不同：MyISAM是分开保存，而innodb是保存到表空间

MyIASM支持索引压缩，而Innodb索引和数据是绑定保存不压缩，体积大。

Innodb很多时候是行级锁，而myisam是表级锁，innodb的并发高。

InnoDB不支持FULLTEXT类型的索引（新版本的InnoDB也支持）。

Innodb支持事务，外键，数据完整性约束要强。而MYISAM不支持。

第二十讲

排序order by 排序取决于字符集的校对规则，因为校对规则是与字符集绑定的!

允许多字段排序，当一个字段不能区分时，再采用第二个字段进行排序

语法格式：... order by 字段一 asc，字段二 desc;

select \* from xxx\_table where xxx order by xxx asc/desc;默认是升序

去除重复记录

distinct：重复记录指的是，字段值 都相同的记录

所以检索出来的记录中 没有一条重复的记录 没有一条记录的字段完全相同

select distinct ziduan1,ziduan2 from xxxxxxx

union：联合查询

只需要在两个select语句之间通过union连接即可，最好用小括号将两个子查询语句括起来

如果在子查询中需要用到排序，则括号是必须的，而且order by 还要结合limit来使用，否则忽略子查询的排序

如果order by 放在最后一个select子查询语句之后，则表示整体排序

union使用场景：

获得数据的条件，出现逻辑冲突，或者很难在一个

逻辑内表示，就可以拆分成 多个逻辑，分别实现

最后将结果合并到一起。也就是拆成两个条件不同的select子句，最后联合结果

union自带distinct 去重功能，要想显示所有记录，包括重复的，要用 union all

注意：union连接的两个子查询子句中字段的个数和类型必须一致，而且检索结果的列名称取决于第一条select子句的列名

子查询：语句内部的查询语句就是子查询语句

比如：select t\_name, gender from teacher\_class where days=(select max(days) from teacher\_class);

此句表示：查询授课天数最多的老师的信息，把最大的天数作为一个子查询

子查询的分类：不同的分类有不同的使用方式

分类标准：子查询的位置、子查询的返回值形式、

返回值的分类：

单一值：标量子查询 使用：获得一个标量值用关系运算符对其进行运算 > < = >= <=

一列多行：也就是一类数据的集合。所以要用到操作集合的运算符 in =any

not in <>all

一行多列：可以通过distinct或者limit 1 来限制子查询只返回一行 使用：在where后面通过括号()来构建与子查询字段相同的一行，然后用 = 连接构建的行和子查询

语法：select xxx from table where (gender,class) = (select gender ,class from table limit 1);

多行多列(表)：通常用在from子查询里 select \* from (table) where cond;

table 部分可以用一个返回一张表的子查询来表示！！！注意：要给子查询返回的结果设置别名 用 as 关键字

外部查询的字段有子查询决定，也就是在子查询内同样可以通过 as 来设置临时表中字段的别名

出现位置：where型

from型、exist型

Exists(subquery)

判断依据：

如果子查询的 可以返回数据，则认为 exists 表达式 返回真。

否者，返回假。

解决思路是不一样的：

exists： 先获得每一条 teacher\_class的数据，然后获得id字段，去teacher表内查找对应值，找到，

说明符合条件。

Exists

Exists(subquery)

判断依据：

如果子查询的 可以返回数据，则认为 exists 表达式 返回真。

否者，返回假。

下面的两个语句完成的是同样的事情：

select \* from teacher\_class where exists (select \* from teacher where teacher\_class.id=t\_id);

select \* from teacher\_class where id in (select t\_id from teacher);

解决思路是不一样的：

exists： 先获得每一条 teacher\_class的数据，然后获得id字段，去teacher表内查找对应值，找到，

说明符合条件，再检索出相应字段。

深入理解exist子查询：相当于双重循环，先获得外查询的一条记录，然后比对外循环的指定字段和内循环的所有指定字段值，若相同，取出外循环相应字段

in：先获得所有的id的可能性。再在检索teacher\_class数据时，判断当前的id是否在id集合内。

连接查询：也叫链接查询

分类：

内连接：inner join 要求连接的多个数据都必须存在才能进行连接 也就是两个表的记录都要能够匹配得上

外连接：left/right [outer] join 一条记录在另外一张表中没有与之对应的记录，也就是连到了一个不存在的数据 称为外连接

自然连接：natrual join 既有内连接，又有外连接

连接的是两个表的数据

语法：select xxx from table\_left inner join table\_right on 条件;

这里的条件是：table\_left的某个字段 = table\_right 的某个字段

注意：查询的字段可以来自两个表的任何字段，要注明 表名

eg：select join\_teacher.t\_name,join\_teacher\_class.begin\_date,join\_teacher\_class.days from join\_teacher inner join join\_teacher\_class on join\_teacher.id = join\_teacher\_class.t\_id;

上面语句如果改为 left outer join 则会出现有记录连接到不存在的记录，null

执行流程：首先根据指定条件连接两个表的记录，组成一个结果集，这个结果集包含这两个表的所有字段，然后根据要求检索出指定字段

可以给join连接的左右表设置别名 as xxx 然后通过别名来取表的字段，这样更方便

内连接是可以省略条件的，共存在MxN个连接，这种连接称为交叉连接或者笛卡尔积，此时可以用cross join来代替inner join

在mysql中inner join是默认的连接，也就是可以不写 inner

不带条件的多表查询也是一个笛卡尔积

最常用的是left join 左外连接，保留左表虚拟右表，并用null表示虚拟字段

外连接不能没有条件

外连接不能使用where，where一般表示过滤的条件，on用于连接的条件，内外连接都适用

如果要实现全连接，可以用union连接左右两个外连接

区别：这里的内外连接是两个表的横向连接，而union是查询结果的纵向连接

规律：内外连接的工作原理和exist子查询的工作原理类似，都是双重循环，比对条件，条件符合，则返回true将其检索出来，或者连接成功

where的写法：数据过滤，理解上，数据安装交叉连接连接完成后，再做数据过滤。

on的写法：在连接时，就对数据进行判断。

Using的写法。Using 要求，负责连接的两个实体之间的字段名称，一致。

建议是，在有同名字段时，使用using ，而在通用条件时，使用on。

在数据过滤时（不是指的连接过滤） 使用where。

using(重复字段);作为条件，会去掉结果中重复字段，并放在前面，而on condition 是会显示所有字段，不具有去重功能

自然连接：包含内外连接

自然内连接：natural join

自然左连接：natural left join

自然右连接：natural right join

其特点是：与使用using条件的内/外连接一样

自然连接不需要指定 连接条件。

mysql会使用多表内的，相同的字段，作为连接条件。

在写连接的时候，尽量避免使用笛卡尔积

连接时支持多表连接，一个表可以被多次连接，也可以自己连接自己。由于多次连接到同一个表产生的表名冲突可以通过设置别名来区分!!!

子查询和连接查询往往都可实现同样的效果，都支持自身或者跨表操作

多表查询：即是在from后面有多个表名，每张表用逗号隔开，并且设置别名，若有n个表，至少要有n-1个条件限制，条件之间用and连接

学习进度31

将检索出来的结果保存到服务器文件

语法：select \* into outfile 'e:/amp/test' from table\_name; 他是可以自动创建文件的！！！但是不能重写已存在的文件

相当于将查询的数据备份，生成的文件格式：默认每条记录独占一行，制表符来区分字段

区分行和字段的分隔符是可以修改的：

如果要导出二进制文件，可以使用 into dumpfile 文件路径 from xxx ;这样可以确保二进制没有被转义等其他额外操作

insert使用细节：

插入数据时，如果主键冲突会如何？

默认有 主键约束，不会插入成功

但是可以在insert语法内，控制

在主键冲突时，改成执行更新操作。

insert into teacher\_class (id, t\_name, c\_name) values

(13, '杨露婵', '太极')

on duplicate key update

t\_name='杨露婵', c\_name='太极'

;

插入的数据源：可以是查询出来的记录，也就是select子查询结果作为作为新内容插入，蠕虫复制

replace和insert是可以替换的，只是如果主键冲突时，replace可以将原来的记录替换，否则直接插入

它的语法和insert完全相同

注意：自动增长必须建立在主键索引上，如果主键是自动增长，主键是不能被删除的，要先修改modify字段为不是自动增长，然后再drop主键

导出数据时，为了方便以后导入，一定要把自增主键的字段用null表示，否则以后导入会有冲突。

例如：select null，name from table\_name;

导入数据：load data infile 文件地址 into table\_name;

update使用细节：

支持与delete一样的语法 where limit order by join(组成一个大表)

不适用外键的多表更新：update one join two using(public\_file) set one\_data=2,two\_data=3 where one\_id=3;

这样就可以一次性更新两个表的记录

delete使用细节：

允许使用条件where(删除符合条件的数据)

允许使用limit限制删除的记录数 limit N

常见的是limit配合order by使用:先排序，再删除固定数量的记录数

如果不限制，则是全部删除

允许连接删除：同时删除多个表内的记录

语法：delete from 表一,表二 using one join two on 连接条件 where 删除条件;

truncate 相当于 delete from table\_name;也就是清空表的所有记录，并不会删除表的结构

注意：使用delete不会清空原来自增的id

但是truncate 是完完整整的清空所有，包括自增id

delete 是逐行删除

truncate是先删除表，再重新建一个结构一样的空表

所以truncate的效率更高，如果针对数据层面的，使用删除delete，初始化表时使用truncate

备份还原：

方案一：

如果是myisam表直接复制表的三个文件即可

但innodb的所有表都存放在一个文件中

方案二：

将建表结构和插入数据的sql语句生产并保存，下次如果需要该结构和数据，重新执行一次sql语句即可

1.备份整个数据内的表

mysqldump -uroot -p php\_one > e:/php\_one.sql

在mysql客户端，直接执行即可：

如何执行 保存在文件内的sql语句：

使用 source 指令，可以指定需要执行sql语句的源代码文件：

2，备份数据库内的某张表：

mysqldump -uroot -p php\_one teacher\_class > e:/php\_one\_teacher\_class.sql

mysqldump -uroot -p php\_one teacher\_class tbl\_name1 tbl\_name2 tbl\_name3 > e:/php\_one\_teacher\_class.sql

创建视图：

create view view\_name as select id，name from info\_teacher;

视图就是存在数据库中的一张虚拟表，可以通过show tables;查看到视图表

视图本身没有数据，只是每次视图查询时通过select子查询获得真实表的数据，相当于一个定义好的表子查询，这个子查询只有在每次操作视图时才被执行

所以可以通过定义视图来实现哪些字段不被看见，而操作视图实际上操作的是真实表的数据

视图主要是用来给别人看的，不建议在视图上直接操作数据，因为有可能插入不完整信息或者无法直接插入

视图本质上就是一个定义好的select语句

视图操作

drop view [if exists]view\_name;

修改视图，实际上就是修改select语句

语法：alter view view\_name as select id , name , salary from stu\_info;

修改视图内部字段使用的名称：

alter view view\_name (v1,v2) as select id,name from stu\_info;

这里的v1、v2就是视图表显示的名称，类似于别名

使用技巧：在视图中用join连接多个表，虚拟出一张新表，往视图插入数据就相当于往他所连接的三张表插入了数据

缩减业务逻辑：

通过视图还可以，是复杂的业务逻辑，简单的完成，先使用视图完成一定的逻辑，在在视图的基础上，完成另外的逻辑。

通常，视图完成的逻辑，都是相对来说比较基础的逻辑。

create view join\_info as select tc.id as tc\_id, t\_name, c\_name, days from join\_teacher\_class as tc left join join\_teacher as t on tc.t\_id=t.id left join join\_class as c on tc.c\_id=c.id;

事务：

简介：一组sql语句操作单元，组内所有sql语句完成一个业务，如果整组成功，意味着全部sql都实现，如果任何一个sql语句执行失败，意味着整个操作都失败。

失败意味着，整个过程是没有意义的，应该是数据库回到操作前的初始状态。

以上就是事务的特性。

如何处理？

1.失败后，可以回到开始位置

2.没成功之前，别的用户（进程，会话）是不能看到操作内的数据修改的。

成功：就让别人看到数据的修改，失败则回到记号位置

思路：就是在一组操作之前，设计一个记号，备份点，还原点

实现：利用innodb存储引擎的事务日志功能；

执行语句时，分成两个阶段，首先保存在事务日志内，如果成功，则将结果提交到数据库内，这就意味着更改成功

1.执行阶段

2.将执行结果提交的阶段

其中我们的事务日志，就是保存执行结果的阶段，如果用户选择提交才将执行结果提交到数据库

默认的执行方式叫自动提交，执行完毕，自动提交执行结果，因此需要关闭自动提交功能，这样才能将执行结果保存在日志内

存在一个系统变量：autocommit可以对自动提交进行配置

set autocommit = 0 ;关闭自动提交

show variables like 'autocommit%';

在关闭自动提交的基础上，执行完所有的sql语句。

判断是否都成功

成功：将结果提交

利用commit

失败：回到开始位置

rollback

常见的事务指令：

开启事务：

start transaction 可以使用begin来代替，但不建议使用

关闭自动提交，如果事务结束了，无论成或败，都会将自动提交机制恢复到初始状态

事务的特点：ACID 分别对应一下四点

1.原子性，不可再分，也就是一组sql语句

2.一致性 也就是在一个事务还没有执行完成时，其他的会话连接是不能够操作这个表的！！！

3.隔离性 事务之间的隔离性，两个会话中开启的事务互不影响

4.持久性 执行结果一旦提交不能回滚

触发器

监听数据进行操作:

在当前操作的表上,设置一个对每行数据的监听器,监听相关事件,当事件发生时,会执行一段有sql完成的功能代码

查询时没有触发器的

创建触发器:

create trigger 名字 事件 执行代码

事件：insert update delete

时间的时机：执行前before 执行后after

所以有六种MySQL事件 before insert ...

create trigger jianbanfei after update on table1 for each row update table2 set money=money+30;

可执行代码：sql语句组成的代码

触发：当对table1执行update操作时，触发可执行代码

比如：update table1 set xxx；

注意：触发器不能重名；目前mysql只支持一类事件设置一个触发器

删除触发器：drop trigger 触发器名 ;

show triggers;

在触发器内获得触发该触发程序时的数据

利用 new old 关键字 分别表示新 旧 记录 指的是一行

new.money - old.money

注意：insert事件不能使用old

delete事件不能使用new

如果一个触发器由多条sql语句组成

应该：

1、语句组成语句块，用begin end来标识语句块

2、语句块要有独立的语句结束符，分号

所以要设置命令行的结束符：触发器开始之前delimiter $$

触发器结束后delimiter ;

使用PHP操作MySQL

此时PHP相当于客户端

首先mysql要开放接口让其他语言连接，也就是在安装时勾选了开发主键development components组件

sql编程

注释：

#行注释

-- 行注释

块注释：

/\*\*/

只有在命令行才有语句结束符这种说法，在PHP中是不用使用语句结束符分号;的

\g \G 可以通过delimiter来修改语句结束符

变量：字段名就是变量

定义一个变量：

1、set 变量名 = 变量值

注意：为了区分系统变量和字段与用户自定义变量，需要在用户变量之前加@表示用户自定义变量

select @var 可以查看变量

2、select 字段 表达式 into 变量列表

可以一次性设置多个变量并赋值

前面部分是变量值，后面部分是变量名称

同样需要@符号

可以把字段值注入到一个变量中

select name from stu\_info where id = 3 into @stu\_name;

注意：以上select语句只能是返回一行，也就是要一一对应

3、在MySQL的select语句中等号= 是关系运算，:=才是赋值运算

在set语句中可以使用 = 和 := 作为赋值运算符

变量有效期：会话结束

变量的类型就是字段的类型

运算符：

函数

1.内置函数

rand() 0-1

formate()格式化

时间日期：

now()

unix\_timestamp() 获得unix时间戳

from\_unixtime() 时间戳转成格式化时间

字符串函数：

concat()字符串连接

substring(原字符串，开始位置，截取长度)开始位置为1

char\_length()字符数

length()字节数

lpad()左边补足 lpad('1',3,'0') 001

2.用户自定义函数

create function 函数名 (参数列表)

返回值类型 函数体

delimiter $$

create function sayhello() returns varchar(10)

begin

return 'hello world!';

end

$$

delimiter ;

带参数形式

delimiter $$

drop function if exists sayhello ;

create function sayhello(name varchar(10)) returns varchar(10)

begin

return concat('hello',name);

end

$$

delimiter ;

调用 select sayhello();

注意：自定义函数是与当前数据库绑定的

流程控制：

分支语句

if 条件 then 执行代码块

else if 条件 then 代码块

else 代码块

end if

循环语句

while 条件 do

循环体

end while

由于mysql中没有{}所以只可以用end xxx来代替，作为分割

写一个判断白天晚上的函数

delimiter $$

create function func1() returns varchar(5)

begin

if hour(now())>18 then

return '晚上';

else

return '白天';

end if ;

end

$$

delimiter ;

用循环实现输出1+2+...10的和

drop function if exists func2;

delimiter $$

create function func2() returns int

begin

-- 注意函数体内，每一句都要用分号表示结束

set @i = 0;

set @sum = 0;

w:while @i<10 do

set @i=@i+1;

if @i = 5 then

#leave w;跳出整个循环

iterate w;#跳出当前循环

end if;

set @sum= @sum+@i;

end while w;

return @sum;

end

$$

delimiter ;

mysql中的sql编程语言是属于强类型的，因为字段是强类型的，一个字段只能存放一种类型数据

跳出当前循环

leave 终止循环，这里的循环需要用一个标签包起来

lab:while do ...end while lab;

leave lab;就是表示退出上面循环

iterate lab; 跳出正在执行的循环并继续下一个循环

函数内使用的变量

@var 形式相当于全局变量，函数内外都可使用

函数的参数：

函数声明的局部变量：

在函数内部使用：declare i int default 0;

建海量表方法：

往表中插入该表现有的所有记录

inser into table\_name(字段1,字段2,..) select ziduan1,ziduan2 from table\_name;

结果是往表中插入查询出来的记录

练习：如何调换两个表的内容

way：创建第三个表，使用rename来进行移动，代码如下：

rename table tb1 to tb3,tb2 to tb1,tb3 to tb2;

这里的第三张表是不需要创建的

多表查询

燕十八mysql基础

在MySQL 5.1中，采用二进制格式保存DECIMAL列的值，将9个十进制数字打包在4字节中。

DECIMAL(18,9)列在小数点的每一侧均有9位数字，因此，整数部分和小数部分均需要4字节。DECIMAL(20,10)列在小数点的每一侧均有10位数字。对于每一部分，9位数字需要4字节，剩余的1位数字需要1字节。

就是利用了字节保存数据的范围来表示

比如1字节=255 2位数字

2字节=65536 3-4位数字 一次类推

4字节 8-9位数字

日期和时间类型

如果出现incorrect string value 说明是客户端的字符编码和服务器的编码不对应引起的，只要设置 set names gbk 即可

date time datetime 为什么要用这些而不直接用字符串，因为这样更节省空间，可以查手册

timestamp 时间戳

CURRENT\_TIMESTAMP 自动获取当前时间 形成时间戳格式，显示格式与datetime相同

表的优化：

1.表中定长的字段越多，检索速度越快，像char(1) decimal(9,2) date datetime int tinyint

2.name char(10)多浪费点空间，提高速度 值！！！

如果浪费太大是不行的，比如intro(1400),尽量避免

在开发中,会员的信息优化往往是 把频繁用到的信息,优先考虑效率,存储到一张表中.

不常用的信息和比较占据空间的信息,优先考虑空间占用,存储到辅表中.

Alter table 表名 add 列名称 列类型 列参数 after 某列 [把新列加在某列后]

Alter table 表名 add 列名称 列类型 列参数 first [把新列加在最前面]