

目录

[▪▪▪▪▪▪▪▪Day01.▪▪▪▪▪▪▪▪ 1](#_Toc505868298)

[1 数据库 1](#_Toc505868299)

[2 MySql 1](#_Toc505868300)

[3 MySql客户端工具 1](#_Toc505868301)

[3.1 连接服务器 1](#_Toc505868302)

[3.2 查看数据库 1](#_Toc505868303)

[3.3 进入数据库 1](#_Toc505868304)

[3.4 查看数据表列表 1](#_Toc505868305)

[3.5 退出客户端 1](#_Toc505868306)

[4 远程连接mysql 1](#_Toc505868307)

[n root@192.168.18.219 1](#_Toc505868308)

[n root@192.168.18.% 1](#_Toc505868309)

[5 中文问题 1](#_Toc505868310)

[6 数据库管理 1](#_Toc505868311)

[6.1 创建数据库 1](#_Toc505868312)

[6.2 查看数据库 1](#_Toc505868313)

[6.3 修改数据库 1](#_Toc505868314)

[6.4 删除数据库 1](#_Toc505868315)

[7 数据表管理 1](#_Toc505868316)

[7.1 创建表 1](#_Toc505868317)

[7.2 查看表 1](#_Toc505868318)

[7.3 修改表 1](#_Toc505868319)

[7.3.1 改表名 1](#_Toc505868320)

[7.3.2 改表的属性 1](#_Toc505868321)

[7.3.3 添加字段 1](#_Toc505868322)

[7.3.4 修改字段名 1](#_Toc505868323)

[7.3.5 修改字段类型 1](#_Toc505868324)

[7.3.6 修改字段顺序 1](#_Toc505868325)

[7.3.7 删除字段 1](#_Toc505868326)

[7.4 删除表 1](#_Toc505868327)

[7.5 截断表、重建表 1](#_Toc505868328)

[8 sql 1](#_Toc505868329)

[8.1 insert 插入数据 1](#_Toc505868330)

[8.2 update 1](#_Toc505868331)

[8.3 delete 1](#_Toc505868332)

[8.4 select 1](#_Toc505868333)

[8.5 术语 1](#_Toc505868334)

[8.5.1 row 1](#_Toc505868335)

[8.5.2 record 1](#_Toc505868336)

[8.5.3 column 1](#_Toc505868337)

[8.5.4 field 1](#_Toc505868338)

[8.5.5 table 1](#_Toc505868339)

[8.5.6 database 1](#_Toc505868340)

[8.5.7 schema 1](#_Toc505868341)

[9 数据存储引擎 1](#_Toc505868342)

[9.1 innodb 1](#_Toc505868343)

[9.2 myisam 1](#_Toc505868344)

[9.3 memory 1](#_Toc505868345)

[10 数据类型 1](#_Toc505868346)

[10.1 数字 1](#_Toc505868347)

[10.2 字符串 1](#_Toc505868348)

[10.3 日期 1](#_Toc505868349)

[11 京淘电商数据库 1](#_Toc505868350)

[11.1 商品表tb\_item 1](#_Toc505868351)

[12 作业 1](#_Toc505868352)

[▪▪▪▪▪▪▪▪Day02.▪▪▪▪▪▪▪▪ 1](#_Toc505868353)

[1 表字段的字节量限制(了解) 1](#_Toc505868354)

[2 约束 1](#_Toc505868355)

[2.1 主键 1](#_Toc505868356)

[2.1.1 添加主键 1](#_Toc505868357)

[2.1.2 查看主键 1](#_Toc505868358)

[2.1.3 删除主键 1](#_Toc505868359)

[2.1.4 自增主键 1](#_Toc505868360)

[2.2 外键 1](#_Toc505868361)

[2.2.1 添加外键 1](#_Toc505868362)

[2.2.2 查看外键 1](#_Toc505868363)

[2.2.3 取消外键 1](#_Toc505868364)

[2.3 非空 1](#_Toc505868365)

[2.3.1 添加非空约束 1](#_Toc505868366)

[2.3.2 查看非空约束 1](#_Toc505868367)

[2.3.3 取消非空约束 1](#_Toc505868368)

[2.4 唯一 1](#_Toc505868369)

[2.4.1 添加唯一约束 1](#_Toc505868370)

[2.4.2 查看唯一约束 1](#_Toc505868371)

[2.4.3 取消唯一约束 1](#_Toc505868372)

[2.5 检查 1](#_Toc505868373)

[3 表之间的关系 1](#_Toc505868374)

[3.1 一对一 1](#_Toc505868375)

[3.2 一对多 1](#_Toc505868376)

[3.3 多对多关系 1](#_Toc505868377)

[4 SQL - DML数据操作语言 1](#_Toc505868378)

[4.1 insert 1](#_Toc505868379)

[4.2 update 1](#_Toc505868380)

[4.3 delete 1](#_Toc505868381)

[5 作业 1](#_Toc505868382)

[▪▪▪▪▪▪▪▪Day03.▪▪▪▪▪▪▪▪ 1](#_Toc505868383)

[1 select 查询 1](#_Toc505868384)

[2 where 子句 1](#_Toc505868385)

[3 SQL 语句中的单引号 1](#_Toc505868386)

[3.1 SQL 注入攻击 1](#_Toc505868387)

[4 distinct 去除重复 1](#_Toc505868388)

[5 order by 子句 1](#_Toc505868389)

[6 字段别名 1](#_Toc505868390)

[7 函数 1](#_Toc505868391)

[7.1 字符串 1](#_Toc505868392)

[7.2 数字 1](#_Toc505868393)

[7.3 日期 1](#_Toc505868394)

[8 作业 1](#_Toc505868395)

[▪▪▪▪▪▪▪▪Day04.▪▪▪▪▪▪▪▪ 1](#_Toc505868396)

[1 函数（续） 1](#_Toc505868397)

[1.1 null值 1](#_Toc505868398)

[1.2 加密 1](#_Toc505868399)

[1.3 分支 1](#_Toc505868400)

[2 多行函数 1](#_Toc505868401)

[3 group by 子句 1](#_Toc505868402)

[4 having 子句 1](#_Toc505868403)

[5 MySql的分页查询 1](#_Toc505868404)

[6 子查询 1](#_Toc505868405)

[6.1 条件子查询 1](#_Toc505868406)

[6.2 from子查询 1](#_Toc505868407)

[6.3 字段列表中的子查询 1](#_Toc505868408)

[7 联合查询 1](#_Toc505868409)

[8 多表查询 1](#_Toc505868410)

[9 作业 1](#_Toc505868411)

[▪▪▪▪▪▪▪▪Day05.▪▪▪▪▪▪▪▪ 1](#_Toc505868412)

[1 标准表连接语法 1](#_Toc505868413)

[2 事务 1](#_Toc505868414)

[2.1 事务的四个特性ACID(理论) 1](#_Toc505868415)

[2.2 事务操作 1](#_Toc505868416)

[2.2.1 启动事务 1](#_Toc505868417)

[2.2.2 提交事务 1](#_Toc505868418)

[2.2.3 回滚事务 1](#_Toc505868419)

[2.3 数据访问冲突 1](#_Toc505868420)

[2.4 事务隔离级别 1](#_Toc505868421)

[3 视图 1](#_Toc505868422)

[3.1 创建视图 1](#_Toc505868423)

[3.2 查看视图 1](#_Toc505868424)

[3.3 删除视图 1](#_Toc505868425)

[4 索引 1](#_Toc505868426)

[4.1 创建索引 1](#_Toc505868427)

[4.2 查看索引 1](#_Toc505868428)

[4.3 删除索引 1](#_Toc505868429)

[4.4 索引的数据结构：B+Tree（了解） 1](#_Toc505868430)

[5 数据备份和恢复 1](#_Toc505868431)

[5.1 备份 1](#_Toc505868432)

[5.2 恢复 1](#_Toc505868433)

[6 数据库编程(了解) 1](#_Toc505868434)

[7 变量 1](#_Toc505868435)

[7.1 环境变量 1](#_Toc505868436)

[7.2 会话变量 1](#_Toc505868437)

[7.3 局部变量 1](#_Toc505868438)

[8 存储过程 1](#_Toc505868439)

[8.1 创建存储过程 1](#_Toc505868440)

[8.2 调用存储过程 1](#_Toc505868441)

[8.3 查看存储过程 1](#_Toc505868442)

[8.4 删除存储过程 1](#_Toc505868443)

[9 流程控制 1](#_Toc505868444)

[9.1 if 1](#_Toc505868445)

[9.2 case 1](#_Toc505868446)

[9.3 while 1](#_Toc505868447)

[9.4 loop 1](#_Toc505868448)

[9.5 repeat 1](#_Toc505868449)

[10 作业 1](#_Toc505868450)

[▪▪▪▪▪▪▪▪Day06.▪▪▪▪▪▪▪▪ 1](#_Toc505868451)

[1 函数 1](#_Toc505868452)

[1.1 创建函数 1](#_Toc505868453)

[1.2 查看函数 1](#_Toc505868454)

[1.3 删除函数 1](#_Toc505868455)

[2 练习 1](#_Toc505868456)

[3 SQL语句的效率检测 1](#_Toc505868457)

[4 触发器 1](#_Toc505868458)

[4.1 创建触发器 1](#_Toc505868459)

[4.2 查看触发器 1](#_Toc505868460)

[4.3 删除触发器 1](#_Toc505868461)



# 数据库

存储数据的仓库

关系型数据库

Oracle

MySql

DB2

Sybase

Sql Server

Sqlite

No Sql数据库

Redis

mongodb

互联网应用，多数采用MySql数据库

# MySql

* 开源免费数据库
* 芬兰MySql创始人，08年将MySql公司以10亿美金卖给Sun，一年后，Sun被Oracle收购
* Oracle收购MySql后，推出收费版MySql，也有开源社区版MySql，但社区被缩小，开发进度缓慢
* 开源社区认为MySql存在闭源风险，开源社区已经不再支持MySql
* MySql创始人从MySql开了一个新的分支，起名叫MariaDB，继续开发，整个开源社区，都转向支持MariaDB
* 阿里的MySql分支AliDB，也向MariaDB贡献代码

# MySql客户端工具

* MySql命令行客户端
* 窗口客户端工具
  + MySql的Workbench
  + Navicat
  + SqlYog
  + ...

## 连接服务器

进入mysql的bin目录下

**mysql 命令**

-u 指定用户名

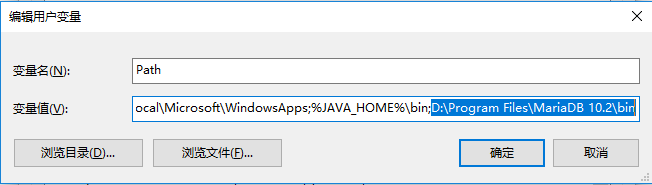
-p 填密码

-h 远程服务器ip

-P 端口，默认3306

**配置环境变量path**

在path中，添加bin目录的路径



mysql -uroot -p

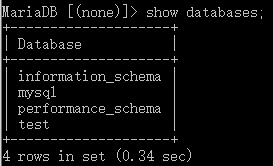
## 查看数据库

mysql服务器上可以有多个数据库

查看数据库列表

show databases;

show schemas;



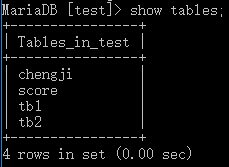
## 进入数据库

use test;

use mysql;

## 查看数据表列表

show tables;



## 退出客户端

quit 或 \q

# 远程连接mysql

* mysql用户是用“用户名@网络地址”来标识
  + root@localhost

只能在本机登录

* + [root@192.168.18.219](mailto:root@192.168.18.219)

只能从192.168.18.219指定的主机来连接

* + [root@192.168.18.%](mailto:root@192.168.18.%25)

可以从18网段内任意主机来连接

* + root@%

可以从任意主机来连接

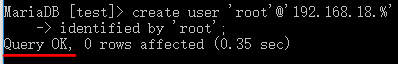
* 创建跨网络连接的用户
  + create user 'abc'@'localhost' identified by '123456'
  + create user 'abc'@'192.168.0.68' identified by ''
  + create user 'abc'@'192.168.0.%' identified by ''
  + create user 'abc'@'%' identified by ''
* 配置用户权限
  + grant select,insert,update,delete on test.\* to 'abc'@'%'
  + grant create,alter,drop on db1.\* to 'abc'@'localhost'
  + grant all privileges on \*.\* to 'abc'@'%'
  + grant select,insert on db1.t1 to 'abc'@'%'
  + grant select(id, name) on db1.t1 to 'abc'@'%

创建夸网络访问的root用户

-- 创建用户'root'@'192.168.18.%'

create user 'root'@'192.168.18.%'

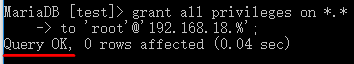
identified by 'root';



-- 对用户授权

grant all privileges on \*.\*

to 'root'@'192.168.18.%';



-- 连接旁边同学的mysql，对方要关闭防火墙

mysql -uroot -p -h192.168.地址 -P3306

# 中文问题

**GBK**

中 D6 D0

**UTF-8**

中 4E B8 AD

服务器端需要知道客户端传过来的中文采用的是什么编码，服务器才能做正确的转码运算

客户端需要告诉服务器，自己发送的中文编码是什么编码

-- windows命令行使用GBK编码

set names gbk;

# 数据库管理

## 创建数据库

create database db1;

utf8在mysql中有些特殊字符不支持，可以使用utf8mp4

alter database db2 charset utf8mb4;

create database db2

charset utf8;

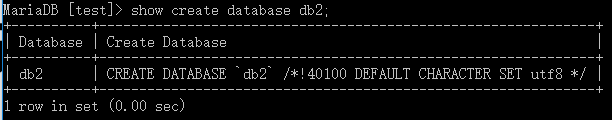
## 查看数据库

show databases;

-- 查看建库语句

show create database db1;

show create database db2;



## 修改数据库

只能修改数据库的默认字符编码

alter database db2 charset gbk;

show create database db2;

## 删除数据库

删除数据库不可恢复

drop database db2;

show databases;

# 数据表管理

## 创建表

-- 修改db1编码

alter database db1 charset utf8;

-- 进入db1库

use db1;

-- 创建表tb1

create table tb1(

name varchar(20),

gender char(1)

) engine=innodb charset=utf8;

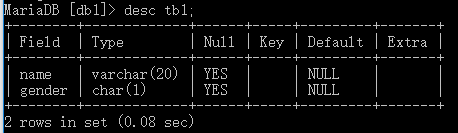
show tables;

## 查看表

show tables;

-- 描述describe，查看表的结构

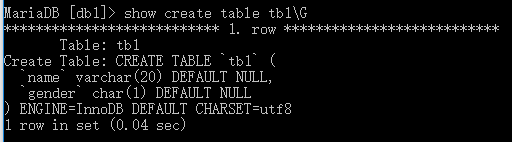
desc tb1;



-- 查看建表语句

-- \G纵向显示

show create table tb1\G



## 修改表

### 改表名

rename table tb1 to user;

show tables;

### 改表的属性

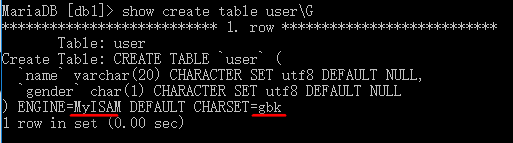
改引擎（innodb或myisam），

改字符编码（GBK 或 utf8）

alter table user

engine=myisam charset=GBK;

show create table user\G



### 添加字段

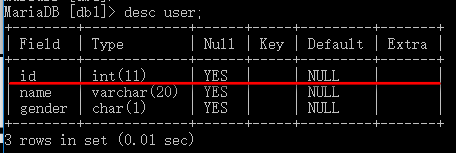
**first**  加在第一个

**after**  加在指定字段的后面

alter table user

add id int first;

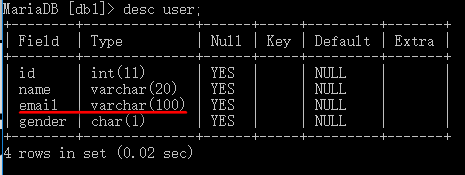
desc user;



alter table user

add email varchar(100) after name;

desc user;



--总共三位，小数位两位

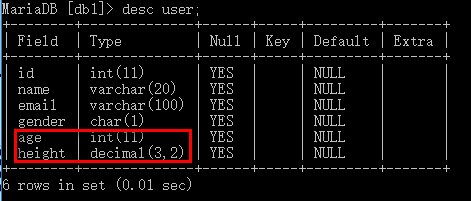
alter table user add(

age int,

height decimal(3,2)

);

desc user;



### 修改字段名

-- gender该名成sex

alter table user

change gender sex char(1);

desc user;

### 修改字段类型

alter table user

**modify** sex varchar(2);

desc user;

### 修改字段顺序

first 第一个

after 指定字段之后

alter table user

modify email varchar(100) after id;

alter table user

modify age int after name;

desc user;

### 删除字段

删除字段，同时删除其中的数据，不可恢复

alter table user

**drop** height;

desc user;

## 删除表

drop table user;

-- 如果存在，删除

drop table if exists user;

show tables;

## 截断表、重建表

先删除表，再重新创建

truncate table user;

# sql

结构化的查询语言

DDL 数据定义，建库建表

DML 数据操作，增删改

DQL 数据查询

## insert 插入数据

-- 通知服务器，客户端使用的是什么编码

set names gbk;

use db1;

drop table if exists user;

create table user(

id int,

name varchar(20)

)engine=innodb charset=utf8;

show tables;

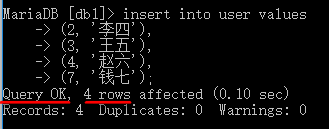
-- 向 user 表插入数据

insert into user values(1, '张三');

insert into user values

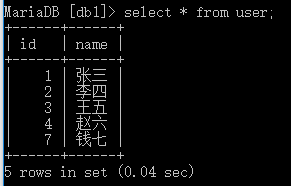
(2, '李四'), (3, '王五'),

(4, '赵六'), (7, '钱七');



-- 查询

select \* from user;



## update

修改数据

update user set

id=999, name='张三三'

where id=1;

select \* from user;

## delete

删除数据

delete from user

where id=4;

select \* from user;

## select

查询数据

select name from user;

select name,id from user;

select \* from user where id>4;

## 术语

### row

行，一行数据

### record

记录，一条记录，一行数据

### column

列

### field

字段，列

### table

表

### database

库

### schema

库

# 数据存储引擎

## innodb

默认存储引擎，多数情况下，都是用innodb

* 支持事务
* 支持外键
* 行级锁

## myisam

* 不支持事务
* 不支持外键
* 表级锁
* 查询多，增删改少，可以使用myisam

## memory

内存表

数据库重新启动，内存数据消失

# 数据类型

## 数字

unsigned 无符号，没有负数

zerofill 用0填充，结合位数设置使用

int(5) zerofill

查询结果中，显示成5位

不足5位前面补0,

超出5位，没效果

* tinyint 1字节
* smallint 2字节
* int 4字节
* bigint 8字节
* float 4字节
* double 8字节
* decimal 定点数，精确地表示金额

decimal(m,n) 字节量m+2

数字类型测试

drop table if exists tb1;

create table tb1(

a int(3) unsigned zerofill,

b decimal(3,2)

)engine=innodb charset=utf8;

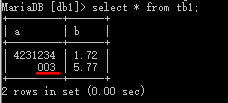
insert into tb1 values

(4231234, 1.72),(3, 5.7678);

insert into tb1 values(62, 754.88);



select \* from tb1;



## 字符串

* char(6)

定长字符串，最长不能超过255个字符

* varchar(6)

变长字符串，最长不能超过65535，

如果长度<=255，需要1个额外字节来保存数据长度

如果长度>255，需要2个额外字节来保存数据长度

* text

65535个字节

字符串测试

drop table if exists tb2;

create table tb2(

a char(6),

b varchar(6),

c text

)engine=innodb charset=utf8;

insert into tb2 values

('abcdef','abcdef','a'),

('阿宝拆的赋给','阿宝拆的赋给','b');

-- 超出长度报错

insert into tb2 values

('abcdefg','abcdefg','c');

## 日期

* datetime

年月日时分秒

* date

年月日

* time

时分秒

* timestamp

时间戳，年月日时分秒

* + 时间范围只到2038年
  + 第一个timestamp字段会随着其他字段的修改，自动更新为当前时间
  + 插入数据时，会自动填入时间

timestamp测试

drop table if exists tb3;

create table tb3(

a int,

b timestamp,

c timestamp

)engine=innodb charset=utf8;

insert into tb3(a) values(1);

select \* from tb3;

update tb3 set a=2;

select \* from tb3;

update tb3 set c='2038-1-1'; 可以修改

update tb3 set c='2039-1-1'; 超出范围

# 京淘电商数据库

## 商品表tb\_item

-- 创建数据库

create database jtdb

charset utf8;

-- 进入jtdb

use jtdb;

-- 创建商品表tb\_item

drop table if exists tb\_item;

create table tb\_item(

id bigint,

cid bigint,

brand varchar(50),

model varchar(50),

title varchar(100),

sell\_point varchar(500),

price bigint,

num int,

barcode varchar(30),

image varchar(500),

status tinyint,

created datetime,

updated datetime

)engine=innodb charset=utf8;

show tables;

desc tb\_item;

show create table tb\_item;

# 作业

重复练习数据表管理语句



# 表字段的字节量限制(了解)

一张表中，所有字段的字节量之和，不能超过65535

* 如果表中存在允许null值的字段，需要一个额外字节来表示 null 值
* varchar 255字节内，需要1个字节表示长度，大于255字节，需要2个字节表示长度
* text 只占用10个字节的字节量限制

字节量限制测试

use db1;

drop table if exists tb1;

create table tb1( -- 1

a int, -- 4

b double, -- 8

c varchar(65520) -- 2+65520

)engine=innodb charset=latin1;

drop table if exists tb2;

create table tb2( -- 1

a int, -- 4

b double, -- 8

c varchar(21840) -- 2+21840\*3

-- d tinyint -- 1

)engine=innodb charset=utf8;

drop table if exists tb3;

create table tb3( -- 1

a int, -- 4

b double, -- 8

c varchar(255), -- 2+255\*3

d varchar(21584), -- 2+21584\*3

e tinyint -- 1

)engine=innodb charset=utf8;

# 约束

限制字段的取值

* 主键
* 外键
* 非空
* 唯一
* 检查

## 主键

一行数据的唯一标识

* 不能重复
* 不能为空
* 会创建索引
* 一般使用业务无关，无意义数据作为主键
  + 自动增加的整数
  + 随机字符串，UUID
  + 按一定规则生成
* 存在 null 值，或有重复值
* 被其他外键引用时，不能删除

学生表

|  |  |  |  |
| --- | --- | --- | --- |
| id（主键） | 学号 | 姓名 | 性别 |
| 1 | 2008151129527 | xx | xx |
| 2 | 2008151679528 | xx | xx |

人员表

|  |  |  |  |
| --- | --- | --- | --- |
| id | 身份证号 | 姓名 | ... |
| 35a8f98c | 35234523 | xx | xx |
| 8f96c9de | 324563456 | xx | xx |

电信计费

|  |  |  |  |
| --- | --- | --- | --- |
| id | 手机号 | 金额 | .. |
| 1 | 32123235 | -2 | xx |
| 2 | 345346 | 34 | xx |

### 添加主键

* 创建表时添加
* 修改表时添加
* 双主键
  + 使用两个字段组合起来作为主键
  + 一般不使用双主键

create table tb4(

id int primary key,

name varchar(20)

)engine=innodb charset=utf8;

create table tb4(

id int,

name varchar(20),

primary key(id)

)engine=innodb charset=utf8;

alter table tb4

add primary key(id);

create table tb4(

a int,

b int,

c varchar(20),

primary key(a,b) -- 双主键

)engine=innodb charset=utf8;

-- 告诉服务器，客户端使用的是什么编码

set names gbk;

insert into tb4 values

(9527, '张三'),(18265, '李四');

insert into tb4 values(9527, '王五');

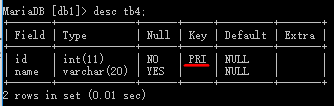


insert into tb4 values(null, '王五');

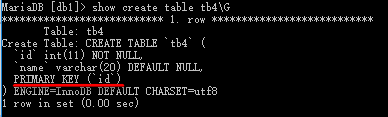


### 查看主键

desc tb4;



show create table tb4\G



### 删除主键

alter table tb4

drop primary key;

### 自增主键

插入数据时，主键值不用手动给出而是会自动产生

#### 添加自增主键

create table tb5(

id int primary key auto\_increment,

name varchar(20)

)engine=innodb charset=utf8;

create table tb5(

id int auto\_increment,

name varchar(20),

primary key(id)

)engine=innodb charset=utf8;

alter table tb5

modify id int auto\_increment;

insert into tb5(name) values

('a'),('b'),('c');

select \* from tb5;

-- 插入null时，也产生自增值填入

insert into tb5 values(null, 'd');

select \* from tb5;

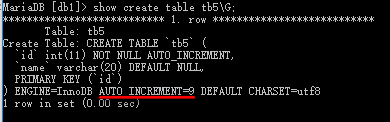
-- 插入一个指定的值

insert into tb5 values(1000, 'e');

select \* from tb5;

#### 查看自增主键

show create table tb5\G



#### 删除自增

-- 不加auto\_increment就是取消

alter table tb5

modify id int;

show create table tb5\G

#### 获得刚生成的主键值

last\_insert\_id() 函数

* 只得到**当前会话**所生成的自增值

**会话一：**

insert into tb5(name) values('e');

**会话二：**

insert into tb5(name) values('f');

**会话一：**

select \* from tb5;

select last\_insert\_id();

**会话二：**

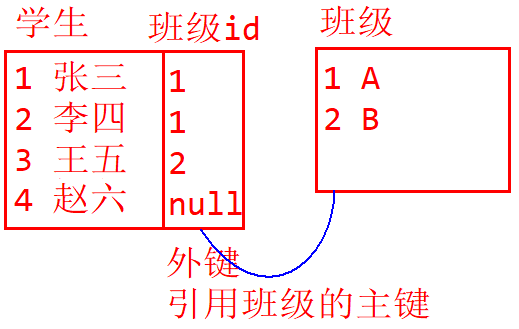
select \* from tb5;

select last\_insert\_id();

## 外键

限制字段的取值，只能取另一个主键中存在的值

* 允许null值
* 允许重复
* 自动创建索引
* 如果存在不符合外键约束的值，不能添加外键
* 删除父表数据，必须保证子表中没有引用这个主键值
* 删除父表，要保证没有其他表引用该表的主键



### 添加外键

-- 班级表，主表、父表

drop table if exists banji;

create table banji(

id int primary key auto\_increment,

name varchar(20)

)engine=innodb charset=utf8;

-- 学生表，子表

drop table if exists xuesheng;

create table xuesheng(

id int primary key auto\_increment,

num int,

name varchar(20),

ban\_id int,

foreign key(ban\_id)

references banji(id)

)engine=innodb charset=utf8;

-- 修改表添加外键

alter table xuesheng

add foreign key(ban\_id)

references banji(id);

insert into banji(name) values('A'),('B');

insert into xuesheng(num,name,ban\_id) values

(6433, 'a', 1),

(6434, 'b', 2),

(6435, 'c', 1),

(6436, 'd', 2),

(6437, 'e', null);

-- 引用不存在的ban\_id，违反外键约束

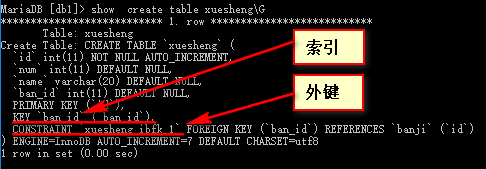
insert into xuesheng(num,name,ban\_id) values

(6433, 'a', 999);



### 查看外键

show create table xuesheng\G



### 取消外键

取消外键，索引不会被自动删除

-- 删除外键

alter table xuesheng

drop foreign key xuesheng\_ibfk\_1;

show create table xuesheng\G

-- 删除索引

alter table xuesheng

drop index ban\_id;

show create table xuesheng\G

## 非空

限制字段，不允许取 null 值

* 已经存在null值，不能添加非空

### 添加非空约束

create table banji(

id int primary key auto\_increment,

name varchar(20) not null

)engine=innodb charset=utf8;

alter table banji

modify name varchar(20) not null;

insert into banji(name) values(null);



查看非空约束

desc banji;

show create table banji \G

### 取消非空约束

-- 不写就是取消

alter table banji

modify name varchar(20);

-- 明确写出，允许null

alter table banji

modify name varchar(20) null;

desc banji;

show create table banji \G

## 唯一

限制字段取值，不能取重复的值

* 允许重复的null值
* 自动创建索引
* 已经存在重复的值，不能添加唯一约束

添加唯一约束

create table banji(

id int primary key auto\_increment,

name varchar(20) unique

)engine=innodb charset=utf8;

create table banji(

id int primary key auto\_increment,

name varchar(20),

unique key(name)

)engine=innodb charset=utf8;

-- 两个字段组合不重复

create table user(

id int primary key auto\_increment,

username varchar(20),

ip varchar(20),

unique key(username, ip)

) engine=innodb charset=utf8;

alter table banji

modify name varchar(20) unique;

alter table banji

add unique key(name);

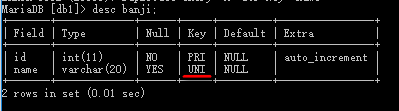
insert into banji(name) values

('A'),('B');

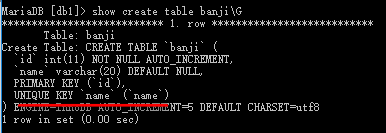


查看唯一约束

desc banji;



show create table banji\G



取消唯一约束

删除索引

alter table banji

drop index name;

## 检查

检查字段中的值，是否符合取值条件

MySql 没有实现

可以执行检查约束语法，但没有实现该功能

create table user(

...

age int,

gender char(1),

...,

check(age>=7 and age<=60),

check(gender in('M', 'F'))

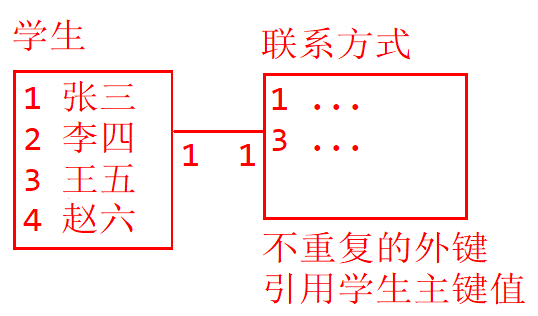
);

# 表之间的关系

* 一对一
* 一对多
* 多对多

## 一对一

通过不重复的外键，来体现一对一关系



学生和联系方式一对一关系

drop table if exists lianxi;

create table lianxi(

xs\_id int primary key,

tel varchar(20),

email varchar(50),

qq varchar(20),

foreign key(xs\_id)

references xuesheng(id)

)engine=innodb charset=utf8;

select \* from xuesheng;

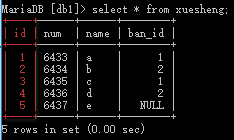
insert into lianxi(xs\_id, tel) values

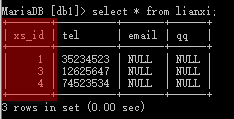
(1, '35234523'),

(3, '12625647'),

(4, '74523534');

select \* from lianxi;

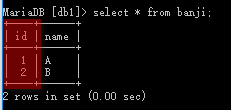




## 一对多

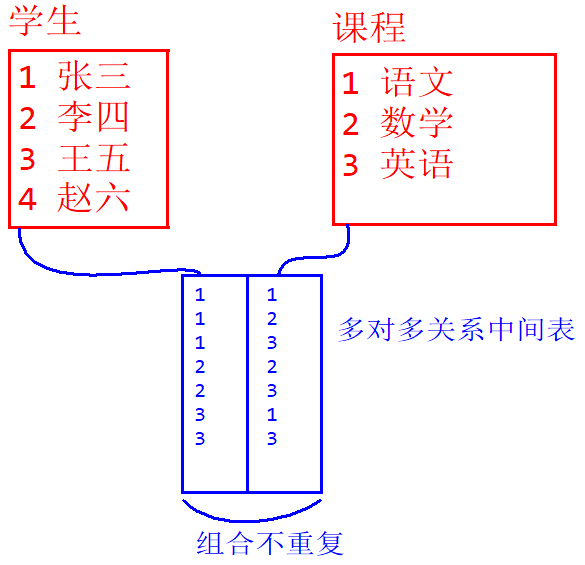
允许重复值的外键，来体现一对多关系





## 多对多关系

添加中间表，设置两个外键，分别引用两张表的主键



学生和课程的多对多关系

-- 课程表

drop table if exists kecheng;

create table kecheng(

id int primary key auto\_increment,

name varchar(20)

)engine=innodb charset=utf8;

-- 学生、课程关系中间表

drop table if exists xs\_kc\_link;

create table xs\_kc\_link(

xs\_id int,

kc\_id int,

primary key(xs\_id,kc\_id),

foreign key(xs\_id)

references xuesheng(id),

foreign key(kc\_id)

references kecheng(id)

) engine=innodb charset=utf8;

-- 课程数据

insert into kecheng(name) values

('语文'), ('数学'), ('英语');

-- 学生、课程关系数据

insert into xs\_kc\_link values

(1,1),(1,2),(1,3),

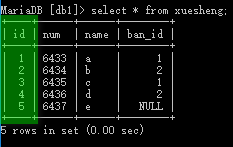
(2,2),(2,3),

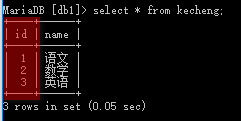
(5,1),(5,3);

select \* from xuesheng;

select \* from kecheng;

select \* from xs\_kc\_link;







# SQL - DML数据操作语言

* insert
* update
* delete

## insert

* insert into tb1 values(值1, 值2...)
* insert into tb1(c,a,b)   
  values(值3,值1,值3)
* 批量插入
  + insert into tb1(a,b,c)  
    values(1,2,3),(4,5,6),(7,8,9)
  + 将查询结果数据，批量插入到表中  
    insert into tb1(a,b,c)  
    select e,f,g from tb2
  + 将查询结果数据，创建成一张表  
    create table tb1  
    as -- 可以省略  
    select a,b,c from tb2

批量插入测试

create table k2

select name,id from kecheng;

select \* from k2;

insert into k2

select name,id from kecheng;

select \* from k2;

-- 自表的数据复制,自增id会出现批量的跳值

insert into kecheng(name) select name from kecheng;

## update

* update tb1 set a=1, b=2, c=3  
  where 条件
* 先按指定字段排序，再更新前几条数据  
  修改销量最低的前5个商品  
  update tb\_item set status=3  
  order by 销量 limit 5

-- 随机修改2个学生， 学号加10000

-- rand() 产生[0,1）范围的随机浮点数

-- 为每行数据产生一个随机值，然后按随机值排序

update xuesheng set num=num+10000

order by rand() limit 2;

select \* from xuesheng;

## delete

* delete from tb1  
  where 条件
* 先排序，然后删除前几条  
  delete from tb1  
  order by a limit 5

# 作业

4.2.1 商品分类表(tb\_item\_cat)

* id自增主键
* parent\_id外键，引用自己的主键
* sort\_order顺序号
* is\_parent 是否父分类

1 电脑 1 1

2 手机 3 1

3 电子产品 2 1

4 联想笔记本 1 2 0

5 戴尔笔记本 1 1 0

6 iPhone 2 1 1

7 华为 2 2 0

4.2.2 商品表(tb\_item)

* id 自增主键
* cid 分类id，非空，外键引用分类id

默认值和注释：

create table tb1(

a int default 0 comment '注释',

b varchar(10) default '' comment '注释'

);



# select 查询

select \* from tb1

查询全部字段

select c,a,b from tb1

查询指定的字段列表

# where 子句

对数据进行过滤查询

|  |  |
| --- | --- |
| =  <> | 等值过滤id=100  不等过滤 id<>100 |
| > >= < <= | 比较大小id>100 |
| between 小 and 大 | 范围判断[小, 大]  id between 100 and 120 |
| in | 从指定数据列表中取值 in(5,1,8,4,9) |
| is null | 判断是否是null值 |
| like | 模糊查询  \_ 通配单个任意字符  % 通配多个任意字符  \\_ 转义成斜杠字符  \% 转义成百分号字符  \\ 转义成一个斜杠字符 |
| not | not between 小 and 大  not in  is not null  not like |
| and | 并且,优先级高 |
| or | 或者 |

准备测试数据

source d:\hr\_mysql.sql

show tables;

select \* from employees;

employee\_id 员工id

first\_name 名

last\_name 姓

email email前缀

phone\_number 电话

hire\_date 入职时间

job\_id 工作岗位代码

salary 工资

commission\_pct 提成比例

manager\_id 主管id

department\_id 部门id

查询测试

1. 查询员工id,first\_name,salary

select employee\_id,first\_name,salary

from employees;

2. 查询员工id是120的员工

select employee\_id,first\_name,salary

from employees

where employee\_id=120;

3. 查询job\_id是 'IT\_PROG'

select

employee\_id,first\_name,salary,job\_id

from employees

where job\_id='IT\_PROG';

4. 查询 department\_id 是 50

select

employee\_id,first\_name,salary, department\_id

from employees

where department\_id =50;

5. 查询主管id，manager\_id是100

select

employee\_id,first\_name,salary, manager\_id

from employees

where manager\_id =100;

6. 部门不是50，排除50部门

select

employee\_id,first\_name,salary, department\_id

from employees

where department\_id<>50;

7. 工资 salary 小于等于 3000

select

employee\_id,first\_name,salary

from employees

where salary<=3000;

8. 工资是 [8000, 10000]

select

employee\_id,first\_name,salary

from employees

where salary between 8000 and 10000;

select

employee\_id,first\_name,salary

from employees

where

salary>=8000 and salary<=10000;

9. 员工id,   
employee\_id是 100,120,150,170

select

employee\_id,first\_name,salary

from employees

where

employee\_id in(100,120,150,170);

10. 岗位代码是 'IT\_PROG','SA\_MAN','HR\_REP'

select

employee\_id,first\_name,salary,job\_id

from employees

where

job\_id in('IT\_PROG','SA\_MAN','HR\_REP');

11. 主管id， manager\_id 是null

select

employee\_id,first\_name,salary,manager\_id

from employees

where manager\_id is null;

12. 部门id， department\_id 是 null

select

employee\_id,first\_name,salary, department\_id

from employees

where department\_id is null;

13. first\_name 中包含 'en'

select

employee\_id,first\_name,salary

from employees

where first\_name like '%en%';

14. first\_name第3和第4个字符是 'en'

select

employee\_id,first\_name,salary

from employees

where first\_name like '\_\_en%';

15. first\_name以'El' 开头

select

employee\_id,first\_name,salary

from employees

where first\_name like 'El%';

16. phone\_number中包含88

select

employee\_id,first\_name,salary,phone\_number

from employees

where phone\_number like '%88%';

17. 工资范围在 <3000 和 >15000

select

employee\_id,first\_name,salary

from employees

where salary not between 3000 and 15000;

select

employee\_id,first\_name,salary

from employees

where salary<3000 or salary>15000;

18. 排除 30,50,80,100 部门

select

employee\_id,first\_name,salary,department\_id

from employees

where

department\_id not in(30,50,80,100);

19. 提成 commission\_pct 不是 null

select

employee\_id,first\_name,salary, commission\_pct

from employees

where commission\_pct is not null;

20. first\_name不包含en

select

employee\_id,first\_name,salary

from employees

where first\_name not like '%en%';

21. phone\_number不包含44

select

employee\_id,first\_name,salary, phone\_number

from employees

where phone\_number not like '%44%';

22. 姓名中包含en，并且在80部门

select

employee\_id,first\_name, last\_name,

salary,department\_id

from employees

where (first\_name like '%en%' or

last\_name like '%en%') and

department\_id=80;

23. 90部门或者工作岗位后缀是'CLERK'

select

employee\_id,first\_name, last\_name,

salary,department\_id,job\_id

from employees

where department\_id=90 or

job\_id like '%CLERK';

# SQL 语句中的单引号

单引号用两个单引号，转义成一个单引号字符

'I'm super man' --> 'I''m super man'

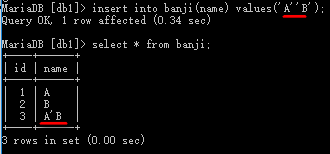
use db1;

-- 告诉服务器，客户端使用的是什么编码

set names gbk;

insert into banji(name) values('A''B');

select \* from banji;



## SQL 注入攻击

用单引号改变 sql 语句的结构

用户名：abc

密码：123

select \* from user

where username='abc' and password='123'

用户名：vfwe

密码：1' or '1'='1

select \* from user

where username='vfwe' and

password='1' or '1'='1'

解决SQL注入攻击的问题：

一个单引号替换成两个单引号

1' or '1'='1 --> 1'' or ''1''=''1

select \* from user

where username='vfwe' and

password='1'' or ''1''=''1'

sql注入

-- 新建user表

use db1;

drop table if exists user;

create table user(

id int primary key auto\_increment,

username varchar(20) unique not null,

password char(32) not null

)engine=innodb charset=utf8;

-- 插入用户数据，密码用md5加密

insert into user(username,password) values

('abc', md5('123')),

('def', md5('456')),

('ghi', md5('789'));

-- sql注入

select \* from user

where username='vfwe' and

password='1' or '1'='1';

-- 避免sql注入

select \* from user

where username='vfwe' and

password='1'' or ''1''=''1';

# distinct 去除重复

select **DISTINCT** a from ...

去除a字段的重复值

select **DISTINCT** a,b,c from ...

去除字段组合额重复值

distinct 测试

use hr;

24. 查询部门id，department\_id 去除重复值

select distinct department\_id

from employees

where department\_id is not null;

25. 不重复的job\_id

select distinct job\_id

from employees;

26. department\_id和job\_id组合不重复

select distinct department\_id,job\_id

from employees;

27. 查询有文员CLERK的部门id

select distinct department\_id

from employees

where job\_id like '%CLERK';

# order by 子句

排序

select ...

from ...

where ...

order by ...

* order by a

按a字段升序排序

* order by a,b

先按a字段升序排序，a的值相同，

再按b字段升序排序

* asc 升序排序（默认）

order by a asc

* desc 降序排序

order by a desc

order by a desc, b

order by a desc, b desc

order by 排序测试

28. 员工按薪水 salary 升序排序

select employee\_id,first\_name,salary

from employees

order by salary;

29. 员工按部门排序，相同部门按工资降序排序

select employee\_id,first\_name,salary,department\_id

from employees

order by department\_id,salary desc;

30. 薪水>=10000，按部门排序，相同部门按工资降序排序

select employee\_id,first\_name,salary,department\_id

from employees

where salary>=10000

order by department\_id,salary desc;

31. 查询50和80部门，按工资降序排序，工资相同，按工作岗位代码排序

select employee\_id,first\_name,salary,department\_id,job\_id

from employees

where department\_id in(50,80)

order by salary desc, job\_id asc;

# 字段别名

可以为字段，或者表达式起一个列的别名

* select abc as a from ...
* select abc a from ...

as可以省略

* select abc a from ... where abc=1

where 后面不能使用字段别名

* select abc a from ... where abc=1 order by a

order by 可以使用字段别名

* sql语句执行顺序：
  + - where 对数据进行过滤
    - select 选取字段
    - order by 排序

字段别名测试

32. 查询年薪，薪水\*12，年薪>=10万，并按年薪降序排序

select employee\_id,first\_name,salary,

salary\*12 sal

from employees

where salary\*12>=100000

order by sal desc;

33. 姓名连接起来显示，按姓名排序

select employee\_id,

concat(first\_name,' ',last\_name) name

from employees

order by name;

# 函数

* 字符串
* 数字
* 日期
* null值
* 加密
* 分支
* 多行函数

## 字符串

char\_length('a中') - 字符数

length('a中') - 字节数

concat('a','b','cde','fff') - 字符串连接，其他数据库可用 || 连接字符串，'abc' || 'def'

concat\_ws(';','abc','def','ggg') - 用分隔符连接字符串

instr('abcdefgdef','def') - 返回第一个子串的位置，从1开始，找不到返回0

locate('abc', '---abc---abc---abc-') - 返回第一个子串的位置，从1开始，找不到返回0

locate('abc', '---abc---abc---abc-',5) - 从指定位置向后找

insert('abcdefghijkl',2, 11, '---') - 用子串取代从2位置开始的11个字符

lower('AdFfLJf') - 变为小写

upper('AdFfLJf') - 变为大写

left('AdFfLJf',3) - 返回最左边的三个字符

right('AdFfLJf',3) - 返回最右边的三个字符

lpad('abc', 8, '\*') - 左侧填充，指定长度比源字符串少，相当于left

rpad('abc', 8, '\*') - 右侧填充，指定长度比源字符串少，相当于left

trim(' a bc ') - 去除两端空格

substring('abcdefghijklmn', 3) - 从3位置开始的所有字符

substring('abcdefghijklmn', 3, 6) - 从3位置开始的6个字符

repeat('abc', 3) - 重复三遍abc

REPLACE('Hello MySql','My','Your') - 子串替换

REVERSE('Hello') - 翻转字符串

SPACE(10) - 返回10个空格

字符串函数测试

set names gbk;

use db1;

select char\_length('a中');

select length('a中');

select concat('\*','a','b','c');

select concat\_ws('\*','a','b','c');

select locate('bc','abcabcabc');

select locate('bc','abcabcabc',3);

select insert('abcdef',2,4,'\*\*');

select left('abcdef',2);

select right('abcdef',2);

select lpad('abc',8,'\*');

select rpad('abc',8,'\*');

select trim(' abc ');

select substring('abcdef',3);

select substring('abcdef',3,3);

select repeat('abc',3);

select replace('abcabcabc','bc','#');

select reverse('abc');

use hr;

34.first\_name和last\_name首字母相同

select employee\_id,first\_name,last\_name

from employees

where left(first\_name,1)=left(last\_name,1);

select employee\_id,first\_name,last\_name

from employees

where substring(first\_name,1,1)=

substring(last\_name,1,1);

35. first\_name和last\_name字符长度相同

select employee\_id,first\_name,last\_name

from employees

where

char\_length(first\_name)=

char\_length(last\_name);

36.email不符合规则，first\_name首字母加last\_name

select employee\_id,first\_name,last\_name,email

from employees

where

concat(left(first\_name,1), replace(last\_name,' ', ''))<>email;

37.first\_name和last\_name连起来显示，空格居中

select employee\_id,

concat(

lpad(first\_name,20,' '),' ',last\_name) name

from employees

order by name;

## 数字

ceil(3.94) - 天花板，向上取整

floor(3.94) - 地板，向下取整

round(673.4974) - 四舍五入

round(673.4974, 2) - 四舍五入到小数点后两位

round(673.4974, -2) - 四舍五入到百

truncate(234.39, 1) - 舍去至小数点后1位

format(391.536, 2) - 数字格式化为字符串，###,###.###，四舍五入，第二个参数为小数位数

rand() - 随机浮点数,[0,1)

数字函数测试

select ceil(3.14);

select ceil(-3.14);

select floor(3.14);

select floor(-3.14);

select round(684.9574);

select round(684.9574, 2);

select round(6834.9574, -2);

select truncate(684.9574, 2);

select format(764364346.7453, 2);

-- [100, 200)范围的随机整数 rand()函数来运算产生

select 100+floor(rand()\*100);

38.涨工资 11.64%，向上取整到10位

select

employee\_id,first\_name,salary,

ceil(salary\*1.1164/10)\*10 new\_sal

from employees;

## 日期

NOW() 返回当前的日期和时间

CURDATE() 返回当前的日期

CURTIME() 返回当前的时间

DATE(时间) 提取日期或日期/时间表达式的日期部分

TIME(时间) 提取日期或日期/时间表达式的时间部分

EXTRACT(字段 From 日期) 返回日期/时间按的单独部分

字段的合法值：

MICROSECOND

SECOND

MINUTE

HOUR

DAY

WEEK

MONTH

QUARTER

YEAR

SECOND\_MICROSECOND

MINUTE\_MICROSECOND

MINUTE\_SECOND

HOUR\_MICROSECOND

HOUR\_SECOND

HOUR\_MINUTE

DAY\_MICROSECOND

DAY\_SECOND

DAY\_MINUTE

DAY\_HOUR

YEAR\_MONTH

DATE\_ADD(日期, INTERVAL 数量 字段) 给日期添加指定的时间间隔

字段的合法值同上

DATE\_SUB(日期, INTERVAL 数量 字段) 从日期减去指定的时间间隔

DATEDIFF(日期1, 日期2) 返回两个日期之间的天数

DATE\_FORMAT(日期, 格式) 用不同的格式显示日期/时间

格式字符： %Y-%m-%d %H:%i:%s

%d/%m/%Y

%Y年%m月%d日

%a 缩写星期名

%b 缩写月名

%c 月，数值

%D 带有英文前缀的月中的天

%d 月的天，数值(00-31)

%e 月的天，数值(0-31)

%f 微秒

%H 小时 (00-23)

%h 小时 (01-12)

%I 小时 (01-12)

%i 分钟，数值(00-59)

%j 年的天 (001-366)

%k 小时 (0-23)

%l 小时 (1-12)

%M 月名

%m 月，数值(00-12)

%p AM 或 PM

%r 时间，12-小时（hh:mm:ss AM 或 PM）

%S 秒(00-59)

%s 秒(00-59)

%T 时间, 24-小时 (hh:mm:ss)

%U 周 (00-53) 星期日是一周的第一天

%u 周 (00-53) 星期一是一周的第一天

%V 周 (01-53) 星期日是一周的第一天，与 %X 使用

%v 周 (01-53) 星期一是一周的第一天，与 %x 使用

%W 星期名

%w 周的天 （0=星期日, 6=星期六）

%X 年，其中的星期日是周的第一天，4 位，与 %V 使用

%x 年，其中的星期一是周的第一天，4 位，与 %v 使用

%Y 年，4 位

%y 年，2 位

LAST\_DAY(日期) - 返回当月最后一天

日期函数测试

select now();

select curdate();

select curtime();

select date(now());

select time(now());

select extract(year from now());

select extract(month from now());

select extract(day from now());

select date\_add(now(),interval 10 year);

select date\_add(now(),interval -10 year);

select datediff(now(),'1992-9-12');

select datediff('1992-9-12',now());

select dateformat(now(),'%Y-%m-%d %H:%i:%s');

select last\_day(now());

39.入职25年以上的员工

select employee\_id,first\_name,salary,hire\_date

from employees

where

hire\_date<date\_add(now(),interval -25 year);

select employee\_id,first\_name,salary,hire\_date

from employees

where

datediff(now(), hire\_date)>=25\*365+6;

40.97年上半年入职

select employee\_id,first\_name,salary,hire\_date

from employees

where hire\_date

between '1997-1-1' and '1997-6-30';

select employee\_id,first\_name,salary,hire\_date

from employees

where

extract(year from hire\_date)=1997 and

extract(month from hire\_date)<7;

41.不论哪一年，1月入职

select employee\_id,first\_name,salary,hire\_date

from employees

where extract(month from hire\_date)=1;

# 作业

4.2.3 商品描述表(tb\_item\_desc)

item\_id 主键，外键，引用商品主键

4.2.8 用户表(tb\_user)

id自增主键

username非空、唯一

password类型改成char(32)，非空

phone 唯一

email 唯一

4.2.9 收藏夹表(tb\_collect)

id 自增主键

user\_id非空，外键，引用用户id

item\_id非空，外键，引用商品id

status默认值1

4.2.10 购物车表(tb\_cart)

id 自增主键

user\_id非空，外键，引用用户id

item\_id非空，外键，引用商品id

num 默认1

复习今天的查询



# 函数（续）

## null值

ifnull(1, 2)

判断第1项是否是null，

不是null，就返回第1项，

是null，返回第2项

coalesce(1,2,3,4,5.....)

从左到右找到第一个非null值返回

null值函数测试

42.求年薪，加提成

select

employee\_id,first\_name,salary,

salary\*12\*(1+ifnull(commission\_pct,0)) sal

from employees;

## 加密

md5()

对密码加密常用的加密算法

得到32位16进制数字字符

sha()

比md5更安全的加密算法

## 分支

if(1, 2, 3)

判断第1项的真假

真，运算第2项

假，运算第3项

case

when 条件1 then ...

when 条件2 then ...

else ...

end

case 变量或表达式

when 值1 then ...

when 值2 then ...

else ...

end

分支测试

select if(rand()>0.5, '大', '小');

select

case truncate(rand()\*5, 0)

when 1 then '壹'

when 2 then '贰'

when 3 then '叁'

else '肆伍'

end a;

43.工资>=15000，有钱，>=8000小康，>=3000温饱，否则穷B

select

employee\_id,first\_name,salary,

case

when salary>=15000 then '有钱'

when salary>=8000 then '小康'

when salary>=3000 then '温饱'

else '穷B'

end a

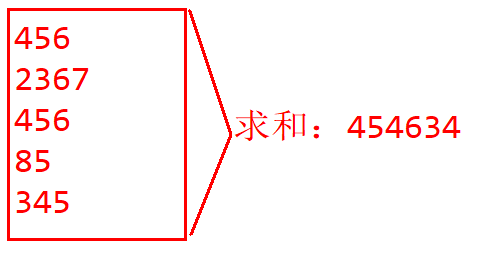
from employees

order by salary desc;

# 多行函数

多行函数、聚合函数

多行数据，运算产生一行结果



* count() 记数
* sum() 求和
* avg() 求平均
* max() 求最大值
* min() 求最小值
* 多行函数不能直接与其他字段一起查询

select avg(salary),first\_name from...

人名和平均工资一起查询没有意义，其他数据库会报错，MySql可以执行，把第一个人名查出来

* 多行函数会忽略null值
* count(\*)对行计数
* count(\*)数据量大时（>千万），性能差

多行函数测试

select avg(salary) from employees;

select avg(salary)~~, first\_name~~

from employees;

select

avg(salary),count(salary),

sum(salary),max(salary),min(salary)

from employees;

select count(department\_id)

from employees;

select count(commission\_pct)

from employees;

select

count(commission\_pct),

avg(commission\_pct),

sum(commission\_pct)/count(commission\_pct),

avg(ifnull(commission\_pct,0))

from employees;

44.有多少个部门

select count(distinct department\_id)

from employees;

45.有多少个工作岗位

select count(distinct job\_id)

from employees;

46.kecheng表中的数据行数

use db1;

select count(\*) from kecheng;

47.1997年入职的人数

select count(\*)

from employees

where hire\_date between '1997-1-1' and '1997-12-31';

select count(\*)

from employees

where

extract(year from hire\_date)=1997;

48.50部门的人数

select count(\*)

from employees

where department\_id=50;

49.最后入职的人的入职时间

select max(hire\_date) from employees;

50.员工122，他的手下人数

select count(\*)

from employees

where manager\_id=122;

# group by 子句

分组求多行函数

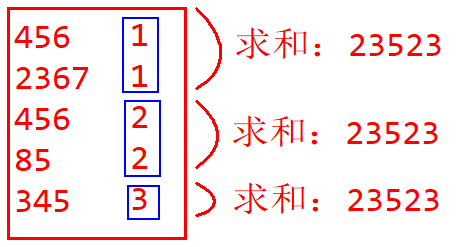
group by a

按a字段相同值分组

group by a,b

按a,b组合相同的值分组

* 分组字段，可以和多行函数一起查询



分组查询测试

51.每个部门的人数

select department\_id,count(\*) c

from employees

where department\_id is not null

group by department\_id

order by c;

52.每个工作岗位的平均工资

select job\_id,avg(salary) c

from employees

group by job\_id

order by c;

53.每年入职的人数

select

extract(year from hire\_date) year,

count(\*) c

from employees

group by year

order by c;

54.每个部门中每个主管的手下人数

select

department\_id,manager\_id,

count(\*) c

from employees

where

department\_id is not null and

manager\_id is not null

group by department\_id,manager\_id

order by department\_id,manager\_id,c;

# having 子句

分组求多行函数时，对多行函数结果进行过滤

where 和 having的区别：

* where 普通条件过滤
* where 最先执行
* having 多行函数结果过滤
* having 分组求出多行函数结果之后，才执行
* having 可以执行普通条件过滤，但效率低

having多行函数过滤测试

55.只有一个人的部门

select department\_id,count(\*) c

from employees

where department\_id is not null

group by department\_id

having c=1;

56.只有一个人入职的年份

select

extract(year from hire\_date) year,

count(\*) c

from employees

group by year

having c=1;

57.只有一个人的工作岗位

select job\_id,count(\*) c

from employees

group by job\_id

having c=1;

58.平均工资大于10000的工作岗位

select job\_id,avg(salary) c

from employees

group by job\_id

having c>10000;

# MySql的分页查询

sql语句没有标准的分页查询语法，数据库厂商各自定义自己的语法

* Oracle，用ruwnum加两层行内视图查询
* sql server，用top关键字
* MySql，用limit
* limit 10

前10行

* limit 0,10

从第一行（0）开始的10行

* limit 10,10

从第11行开始的10行

* limit 20,10

从第21行开始的10行

* 数据量大时（>千万）limit 性能差

limit 分页测试

select employee\_id,first\_name,salary

from employees

limit 10;

select employee\_id,first\_name,salary

from employees

limit 10,10;

select employee\_id,first\_name,salary

from employees

limit 20,10;

select employee\_id,first\_name,salary

from employees

limit 100,10;

select \* from kecheng

limit 100000000,10;

# 子查询

* 条件子查询
* from子查询
* 字段列表中的子查询

## 条件子查询

一个查询的查询结果，作为另一个查询的过滤条件

select .. from ..

where salary>(select ...)

* 单值子查询

=

<>

>

<

* 多值子查询

in

>all 大于最大值

>any 大于最小值

* 多列子查询

(a,b)=(select ...)

(a,b) in(select ...)

条件子查询测试

59.拿最低工资的员工

1）最低工资值

2）用工资过滤查询员工

select min(salary) from employees;

select employee\_id,first\_name,salary

from employees

where salary=(

select min(salary) from employees

);

60.工资低于平均工资

1）求平均工资

2）用平均工资过滤员工

select employee\_id,first\_name,salary

from employees

where salary<(

select avg(salary) from employees

);

61.只有一个人的部门中的员工

1)只有一个人的部门

2)用部门id过滤员工

select department\_id

from employees

where department\_id is not null

group by department\_id

having count(\*)=1;

select employee\_id,first\_name,salary,department\_id

from employees

where department\_id in(

select department\_id

from employees

where department\_id is not null

group by department\_id

having count(\*)=1

);

62.每个部门中拿最低工资的人

1)每个部门的最低工资值

2)用部门和工资值过滤员工

select department\_id,min(salary) s

from employees

where department\_id is not null

group by department\_id;

select employee\_id,first\_name,salary,department\_id

from employees

where (department\_id,salary) in(

select department\_id,min(salary) s

from employees

where department\_id is not null

group by department\_id

);

## from子查询

行内视图

把查询结果当做一张表，从查询结果中再查询

* MySql中，from子查询必须有别名

select .. from (select ...) t

from子查询测试

63.下属人数最多的主管

1)按主管分组查人数

2)得到人数的最大值

3)用最大值过滤主管id

4)用主管id过滤查主管信息

select manager\_id,count(\*) c

from employees

where manager\_id is not null

group by manager\_id;

select max(c) from (

select manager\_id,count(\*) c

from employees

where manager\_id is not null

group by manager\_id

) t;

select manager\_id

from employees

where manager\_id is not null

group by manager\_id

having count(\*)=(

select max(c) from (

select manager\_id,count(\*) c

from employees

where manager\_id is not null

group by manager\_id

) t

);

select employee\_id,first\_name,salary

from employees

where employee\_id=(

select manager\_id

from employees

where manager\_id is not null

group by manager\_id

having count(\*)=(

select max(c) from (

select manager\_id,count(\*) c

from employees

where manager\_id is not null

group by manager\_id

) t

)

);

## 字段列表中的子查询

select a,b,c,(select ...) from ...

字段列表中的子查询测试

64.员工信息后面添加一列平均工资

select employee\_id,first\_name,salary,

(

select avg(salary) from employees

) avg

from employees;

# 联合查询

将多个查询的结果组合成一个查询结果

例如：销量最低和销量最高的5件商品

* union

会去除重复行

* union all

保留重复行

(select ...)

union

(select ...)

union

(select ...)

联合查询测试

65.工资最高和最低各查5个员工

(select employee\_id,first\_name,salary

from employees

order by salary limit 5)

union

(select employee\_id,first\_name,salary

from employees

order by salary desc limit 5);

# 多表查询

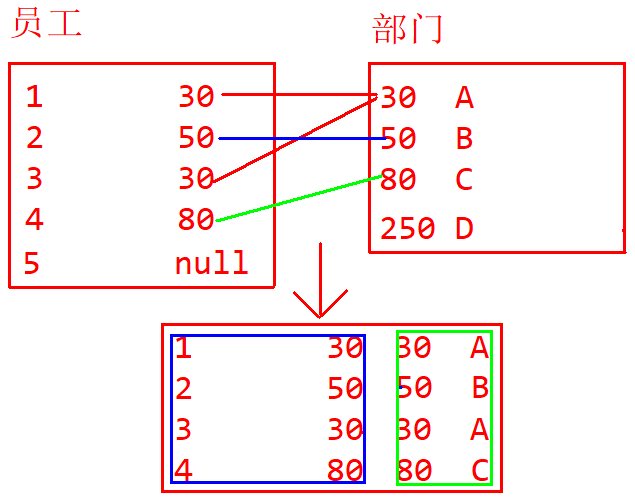
查询员工，同时查询部门名称

按部门id相等，来连接两张表，把两张表连接成一张大表，再查询

* 自连接

一张表看作是两张表，

自己的外键连接自己的主键



表连接查询测试

66.员工，部门名称

select

e.employee\_id,e.first\_name,e.salary,

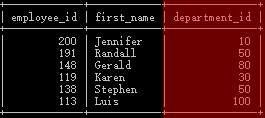
d.department\_id,d.department\_name

from

employees e,departments d

where

e.department\_id=d.department\_id;





67.部门，部门所在的城市

select

d.department\_id,d.department\_name,

l.city

from

departments d,locations l

where d.location\_id=l.location\_id;





68.部门，部门经理名

select

d.department\_id,d.department\_name,

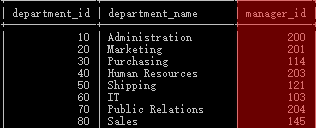
e.first\_name mgr\_name

from

departments d,employees e

where

d.manager\_id=e.employee\_id;





69.员工，主管名

select

e1.employee\_id,e1.first\_name,

e2.first\_name mgr\_name

from

employees e1, employees e2

where e1.manager\_id=e2.employee\_id;





70.员工，部门名，城市

select

e.employee\_id,e.first\_name,

d.department\_name,

l.city

from

employees e,

departments d,

locations l

where

e.department\_id=d.department\_id

and

d.location\_id=l.location\_id;

# 作业

工作代码 job\_id 不是 IT\_PROG、SH\_CLERK、PU\_CLERK、ST\_CLERK、

查询员工编号为 100,150,200,250 的员工

查询 薪水小于3000的员工，和100部门所有的员工

查询 薪水大于10000，且在100部门的 员工

查询 员工 first\_name 是 Bruce 的员工

薪水大于等于 10000 的员工，按姓名排序

只有一个下属的主管信息

平均工资最高的部门编号

1. 部门分组求平均
2. 求最大
3. 用最大值过滤部门编号

select department\_id,round(avg(salary),2) a

from employees

where department\_id is not null

group by department\_id;

select max(a) from (

select department\_id,round(avg(salary),2) a

from employees

where department\_id is not null

group by department\_id

) T;

select department\_id,round(avg(salary),2) a

from employees

where department\_id is not null

group by department\_id

having a=(

select max(a) from (

select department\_id,round(avg(salary),2) a

from employees

where department\_id is not null

group by department\_id

) T

);

平均工资最低的工种，查询做这些工作的人

1. 按岗位分组求平均工资
2. 最小工资值
3. 用最小工资过滤岗位
4. 用岗位过滤员工

select job\_id

from employees

where job\_id is not null

group by job\_id

having round(avg(salary),2)=(

select min(a) from (

select job\_id,round(avg(salary),2) a

from employees

where job\_id is not null

group by job\_id

) T

);

select employee\_id,first\_name,salary,job\_id

from employees

where job\_id in(

select job\_id

from employees

where job\_id is not null

group by job\_id

having round(avg(salary),2)=(

select min(a) from (

select job\_id,round(avg(salary),2) a

from employees

where job\_id is not null

group by job\_id

) T

)

);

查询员工，部门名，主管名，城市

select

e.employee\_id,e.first\_name,

d.department\_name,

e2.first\_name mgr\_name,

l.city

from

employees e,

departments d,

locations l,

employees e2

where

e.department\_id=d.department\_id

and

d.location\_id=l.location\_id

and

e.manager\_id=e2.employee\_id;



# 标准表连接语法

非标准语法，是事实标准，所有数据库都支持，但是它对外连接语法，每个数据库都有自己的扩展语法

标准语法就是对外连接进行语法规范

* 内连接，只查询连接条件以内的数据
* 外连接，可以查询连接条件以外的数据
  + - 左外连接

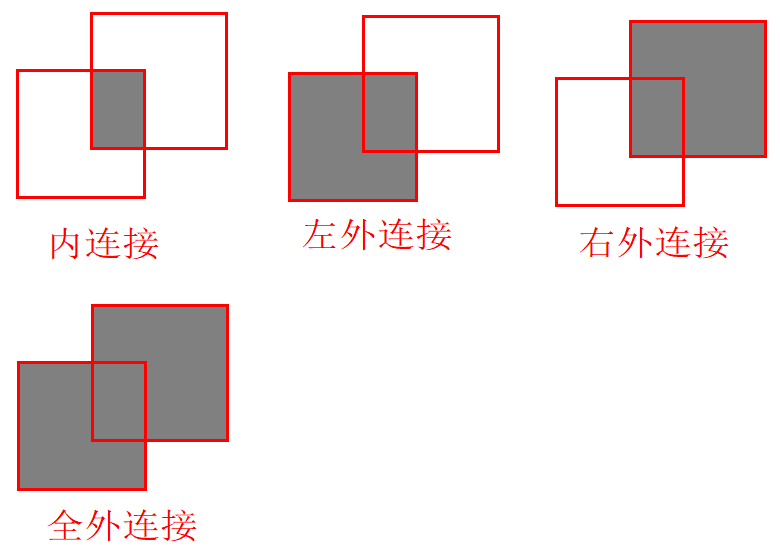
查询左侧表中条件外数据

* + - 右外连接

查询右侧表中条件外数据

* + - 全外连接

两张表中条件外数据



* select ...

from a

inner join b on(连接条件)

join c on(连接条件)

join d on(连接条件)

* select ...

from a

left outer join b on(连接条件)

right join c on(连接条件)

* select ...

from a

full outer join b on(连接条件)

* + - mysql没有实现全外连接

外连接测试

71.查询全部107个员工，显示部门名，没有部门显示null

select

e.employee\_id,e.first\_name,e.salary,

d.department\_name

from employees e

left join departments d

on(e.department\_id=d.department\_id);

# 事务

事务是由一组sql语句组成，这一组sql语句要么整体成功，要么整体失败

转账 A --> B

update user set money=money-100

where name='A'

update user set money=money+100

where name='B'

## 事务的四个特性ACID(理论)

A - 原子性 Atomic

数据操作的最小单元是事务，而不是SQL语句

C - 一致性 Consistency

转账前 a+b = 100

转帐后 a+b = 100

I - 隔离性 Isolation

一个事物进行中时，另一事物不能操作数据

D - 持久性 Durancy

提交事务之后，数据持久生效

## 事务操作

### 启动事务

* begin
* start trunsaction

### 提交事务

* commit
* 事务完成，数据生效

### 回滚事务

* rollback
* 事务结束，数据回退到事务之前的状态

事务测试

|  |  |
| --- | --- |
| 会话一 | 会话二 |
| use db1;  begin; | use db1;  begin; |
| insert into banji(name)  values('C'); |  |
| update banji  set name='AA' where id=1; |  |
| select \* from banji; |  |
|  | select \* from banji; |
| commit; |  |
|  | select \* from banji; |
|  | commit; |
|  | select \* from banji; |
| 会话一 | 会话二 |
| rollback;  begin; | rollback;  begin; |
| update banji  set name='AAA'  where id=1; |  |
|  | update banji  set name='AAAAAA'  where id=1; |
| commit; |  |
|  | update banji  set name='AAAAAA'  where id=1; |
|  | commit; |

## 数据访问冲突

多个事务并行运行时，同时访问数据，会造成数据访问冲突

* 脏数据

一个事务修改数据，还没有提交，

另一个事务就访问到这个数据

* 不可重复读

再次查询相同数据时，与第一次查询结果不一致

* 幻读

一个事务添加或删除了数据并提交,

另一个事务看不到新添加的数据，

或仍能看到删除的数据

## 事务隔离级别

为了兼顾性能和数据安全，数据库提供了4种隔离级别

set tx\_isolation='read-uncommitted';

set tx\_isolation='read-committed';

set tx\_isolation='repeatable-read';

set tx\_isolation='serializable';

* read-uncommitted

读取未提交

脏读、不可重复读、幻读

* read-committed

读取提交

不可重复读、幻读

* repeatable-read

可重复读，默认级别

幻读

* serializable

串行化，所有事务排队执行

事务隔离级别测试

|  |  |
| --- | --- |
| 会话一 | 会话二 |
| set tx\_isolation='read-uncommitted'; | |
| rollback;  begin; | rollback;  begin; |
| insert into banji(name)  values('D'); |  |
| update banji  set name='DD'  where id=1; |  |
|  | select \* from banji; |
| rollback; |  |
|  | select \* from banji; |
| 会话一 | 会话二 |
| set tx\_isolation='read-committed'; | |
| rollback;  begin; | rollback;  begin; |
|  | select \* from banji; |
| insert into banji(name)  values('D'); |  |
| update banji  set name='DD'  where id=1; |  |
|  | select \* from banji; |
| commit; |  |
|  | select \* from banji; |
| 会话一 | 会话二 |
| set tx\_isolation='repeatable-read'; | |
| rollback;  begin; | rollback;  begin; |
|  | select \* from banji; |
| insert into banji(name)  values('E'); |  |
| update banji  set name='EE'  where id=1; |  |
| commit; |  |
|  | select \* from banji; |
|  | update banji set  name=concat('#',name); |
|  | select \* from banji; |
| 会话一 | 会话二 |
| rollback;  begin; | rollback;  begin; |
|  | select \* from banji; |
| delete from banji  where id<4; |  |
| commit; |  |
|  | select \* from banji; |
|  | update banji set  name=concat('$',name); |
|  | select \* from banji; |
|  | commit; |
|  | select \* from banji; |
| 会话一 | 会话二 |
| set tx\_isolation='serializable'; | |
| rollback;  begin; | rollback;  begin; |
| select \* from banji; |  |
|  | select \* from banji; |
| insert into banji(name)  values('E'); |  |
|  | insert into banji(name)  values('F'); |
|  | 成功 |
| 死锁失败，自动回滚 |  |

set tx\_isolation='repeatable-read';

# 视图

将查询语句保存到数据库，起一个名字，就是视图，以后可以从这个视图查询

视图的作用：

* 简化查询
* 安全

## 创建视图

create view a

as

select ...

视图测试

use hr;

create view a

as

select

e.employee\_id,e.first\_name,

d.department\_name,

e2.first\_name mgr\_name,

l.city

from

employees e,

departments d,

locations l,

employees e2

where

e.department\_id=d.department\_id

and

d.location\_id=l.location\_id

and

e.manager\_id=e2.employee\_id;

select \* from a;

## 查看视图

show tables;

desc a;

show create table a\G

show create view a\G

## 删除视图

drop view if exists a;

# 索引

用来提高数据查询的过滤效率，它类似一个目录，查询数据时，通过“目录“先对数据进行定位，然后直接跳到数据所在的位置

## 创建索引

自动创建索引：

* 主键
* 外键
* 唯一

create index 索引名 on 表(字段)

create index 索引名 on 表(字段1, 字段2)

* 为a字段创建索引

where a='abc'

where a like 'abc%'

where a like '%abc' -- 不使用索引

* 为a,b字段创建索引

where a='abc' and b='def'

where a='abc'

where b='def' -- 不使用索引

创建索引测试

create index fname\_index

on employees(first\_name);

## 查看索引

show create table employees\G

## 删除索引

alter table employees

drop index fname\_index;

## 索引的数据结构：B+Tree（了解）



# 数据备份和恢复

系统命令行中的命令：

* mysqldump 备份
* mysql 恢复

## 备份

以下是一行命令，不能折行

mysqldump -uroot -p

--default-character-set=utf8

库名>d:\hr.sql

一行命令，中间不能折行

mysqldump -uroot -p

--default-character-set=utf8

hr>d:\hr.sql

## 恢复

mysql -uroot -p

--default-character-set=utf8

库名<文件路径

登录mysql，新建数据库

create database hr2 charset utf8;

\q 退出

一行命令，不能折行

mysql -uroot -p

--default-character-set=utf8

hr2<d:\hr.sql

# 数据库编程(了解)

在数据库中存储一段程序代码，对数据进行处理运算

* 存储过程
* 函数
* 触发器

存储过程

设置整段代码的结束符

delimiter //

use db1 //

create procedure p1()

begin

select now();

end

//

call p1() //

# 变量

## 环境变量

MySql的运行参数

* 查看环境变量

show variables //

show variables like 'tx\_isolation' //

show variables like '%char%' //

* 修改环境变量

set tx\_isolation='repeatable-read' //

只对当前会话有效

set global tx\_isolation='repeatable-read' //

全局有效，对当前回话需要重新连接才生效

## 会话变量

在会话期间，都可以使用的变量

* set @v1=434 //
* select @v1 //

## 局部变量

在 begin ... end 代码块内定义的变量，只能在这个代码块局部使用

begin

declare v2 int;

declare v3 double default 3.14;

end

# 存储过程

## 创建存储过程

* 存储过程参数
  + - in 输入参数
    - out 输出参数
    - inout 输入和输出参数

存储过程测试

产生[a,b)范围的随机整数

delimiter //

create procedure p2(

in a int, in b int, out r int)

begin

set r=truncate(a+rand()\*(b-a), 0);

end

//

## 调用存储过程

call p2(5,10,@v3) //

select @v3 //

## 查看存储过程

show procedure status\G

查看全部存储过程

show procedure status where db='db1'\G

查看db1库中的存储过程

show create procedure p2\G

## 删除存储过程

drop procedure if exists p2 //

存储过程练习

同时插入学生数据和学生的联系方式数据

1. 插入学生表
2. 查询刚插入的id值
3. 插入联系方式表

create procedure saveXs(

in num int,

in name varchar(20),

in tel varchar(20),

out id int)

begin

-- 局部变量保存新产生的学生id

declare xid int;

-- 插入学生数据

insert into xuesheng(num,name)

values(num,name);

-- 获得新产生的id存到xid

set xid=last\_insert\_id();

-- 插入练习方式表

insert into lianxi(xs\_id, tel)

values(xid, tel);

-- 为输出参数赋值

set id=xid;

end

//

call saveXs(3353,'张三三','45343345',@v4) //

select @v4 //

select \* from xuesheng //

select \* from lianxi //

# 流程控制

* if
* case
* while
* loop
* repeat

## if

if 条件 then

代码

end if

if 条件 then

代码1

else

代码2

end if

if测试

create procedure p3(in a int)

begin

if a=1 then

select '壹';

end if;

if a=2 then

select '贰';

else

select '叁';

end if;

end

//

call p3(1) //

call p3(2) //

call p3(3) //

call p3(4) //

## case

case 测试

create procedure p4(in a int)

begin

case a

when 1 then select '壹';

when 2 then select '贰';

else select '叁';

end case;

end

//

call p4(1) //

call p4(2) //

call p4(3) //

## while

while 条件 do

代码

end while

## loop

a: loop

代码

if 条件 then

leave a; -- 跳出名字叫a的循环

end if;

end loop

## repeat

repeat

代码

until 条件 end repeat

循环测试

drop table if exists tb1 //

create table tb1(

id int primary key auto\_increment,

a int

) //

create procedure p5(in n int)

begin

declare i int default 0;

while i<n do

insert into tb1(a) values(i+1);

set i=i+1;

end while;

set i=0;

lp: loop

insert into tb1(a) values(i+1);

if i=n-1 then

leave lp;

end if;

set i=i+1;

end loop;

set i=0;

repeat

insert into tb1(a) values(i+1);

set i=i+1;

until i=n end repeat;

end

//

call p5(5) //

# 作业

4.2.4 商品规格参数表(tb\_item\_param)

商品的规格参数模板

电脑

配置：

cpu

内存

硬盘

显卡

重量：

颜色：

品牌：

Item\_cat\_id 分类编号，非空，引用分类id

4.2.5 商品规格与商品的关系表(tb\_item\_param\_item)

具体一件商品的规格参数

华为 MateBook D

配置：

cpu xx

内存 xx

硬盘 xx

显卡 xx

重量： xx

颜色： xx

品牌： xx

item\_id 商品id，非空，外键

4.2.6 内容表(tb\_content)

广告、文章、软文、公告

category\_id 内容分类id，非空，外键

4.2.7 内容分类表(tb\_content\_category)



# 函数

函数有返回值

## 创建函数

create function f(输入参数)

returns varchar(100)

begin

代码

return 计算结果;

end;

求平方函数

delimiter //

create function fn\_pow(n int)

returns int

begin

declare r int;

set r=n\*n;

return r;

end

//

select fn\_pow(2) //

select fn\_pow(3) //

select fn\_pow(4) //

## 查看函数

show function status\G

show function status where db='db1'\G

show create function fn\_pow\G

## 删除函数

drop function if exists fn\_pow

# 练习

* 定义函数 fn\_randstr(a,b) 随机产生[a,b)长度的随机字符串
* 定义存储过程p\_gendata(n) 在表中插入n条随机字符串
* 在表中产生大量数据，测试索引

随机字符串函数

定义长度变量len保存长度值

备选字符串变量s0

最终结果字符串变量 s

循环次数控制变量 i

随机定位位置变量 j

循环len次，从s0挑选一个字符，连接到s

set names gbk //

drop function if exists fn\_randstr //

create function fn\_randstr(a int, b int)

returns varchar(255)

begin

declare len int;-- 长度值

-- 备选字符串

declare s0 varchar(800)

default 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ的一是在不了有和人这中大为上个国我以要他时来用们生到作地于出就分对成会可主发年动同工也能下过子说产种面而方后多定行学法所民得经十三之进着等部度家电力里如水化高自二理起小物现实加量都两体制机当使点从业本去把性好应开它合还因由其些然前外天政四日那社义事平形相全表间样与关各重新线内数正心反你明看原又么利比或但质气第向道命此变条只没结解问意建月公无系军很情者最立代想已通并提直题党程展五果料象员革位入常文总次品式活设及管特件长求老头基资边流路级少图山统接知较将组见计别她手角期根论运农指几九区强放决西被干做必战先回则任取据处队南给色光门即保治北造百规热领七海口东导器压志世金增争济阶油思术极交受联什认六共权收证改清己美再采转更单风切打白教速花带安场身车例真务具万每目至达走积示议声报斗完类八离华名确才科张信马节话米整空元况今集温传土许步群广石记需段研界拉林律叫且究观越织装影算低持音众书布复容儿须际商非验连断深难近矿千周委素技备半办青省列习响约支般史感劳便团往酸历市克何除消构府称太准精值号率族维划选标写存候毛亲快效斯院查江型眼王按格养易置派层片始却专状育厂京识适属圆包火住调满县局照参红细引听该铁价严龙飞';

-- 结果字符串

declare s varchar(255) default '';

-- 控制循环次数

declare i int default 0;

declare j int; -- 随机定位

-- 随机长度[a,b)

set len = truncate(a+rand()\*(b-a), 0);

while i<len do

-- 随机定位[1,char\_length(s0)+1)

set j=truncate(

1+rand()\*(char\_length(s0)), 0);

-- 取出s0的j位置字符，连接到s中

set s=concat(s,substring(s0,j,1));

-- i递增

set i=i+1;

end while;

return s;

end

//

select fn\_randstr(3,6) //

创建表

创建内存表，生成的数据先插入内存表，

再把内存数据，批量插入磁盘表

-- 内存表

create table tm(

a varchar(20)

)engine=memory charset=utf8 //

-- 磁盘表

create table t(

id int primary key auto\_increment,

a varchar(20)

)engine=innodb charset=utf8 //

存储过程 p\_gendata()

循环插入n条数据

create procedure p\_gendata(in n int)

begin

declare i int default 0;

-- 向tm表插入n条数据

while i<n do

insert into tm values(fn\_randstr(3,6));

set i=i+1;

end while;

-- tm表数据，批量插入t表

insert into t(a) select a from tm;

-- 清空内存表tm

delete from tm;

end

//

call p\_gendata(10) //

select \* from t //

测试千万数据查询效率

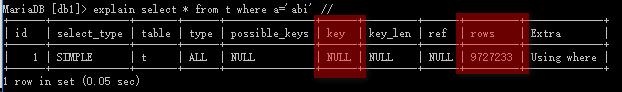
select \* from t where a like 'ab%' //

# SQL语句的效率检测

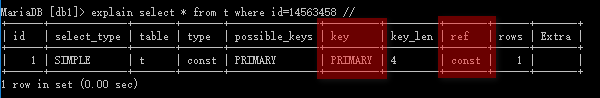
explain 解释sql 语句使用哪个索引，扫描数据的行数

explain 检测sql语句

explain select \* from t where a='abi' //



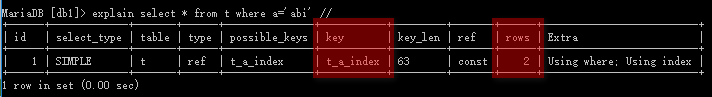
explain select \* from t where id=14563458 //



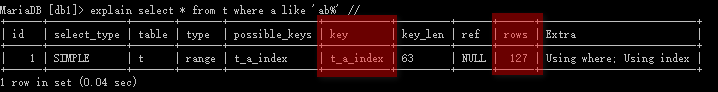
-- a字段创建索引

create index t\_a\_index on t(a) //

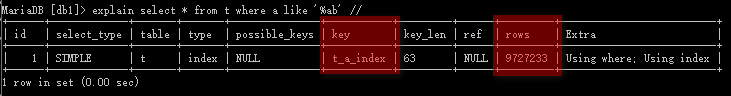
explain select \* from t where a='abi' //



explain select \* from t where a like 'ab%' //



explain select \* from t where a like '%ab' //



# 触发器

在对数据执行增删改操作时，可以触发一段代码执行

* 三种操作
  + - insert
    - update
    - delete
* 两个时间点
  + - before
    - after
* 六种触发器，  
  一张表中最多只能创建6个触发器
  + - before insert
    - before update
    - before delete
    - after insert
    - after update
    - after delete
* 两个隐含对象
  + - new 新行
    - old 旧行
    - insert
      * new 插入的新数据
      * old 没有
    - update
      * new 修改后的新数据
      * old 修改前的旧数据
    - delete
      * new 没有
      * old 被删除的数据
    - 访问字段：
      * new.a
      * old.b

## 创建触发器

create trigger 触发器名 时间点 on 表

for each row

begin

代码

end;

触发器测试

对用户表tb\_user添加触发器，

修改数据之前，先在updated字段中填入系统当前时间

use jtdb2 //

create trigger user\_before\_update

before update on tb\_user

for each row

begin

set new.updated=now();

end

//

update tb\_user set phone='123'

where id=14 //

select \* from tb\_user where id=14 //

对用户表tb\_user添加触发器，

插入数据之前，先在created字段和updated字段中填入系统当前时间

create trigger user\_before\_insert

before insert on tb\_user

for each row

begin

set new.created=now();

set new.updated=now();

end

//

insert into tb\_user(username,password)

values('aabbcc',md5('123')) //

select \* from tb\_user

order by id desc limit 1 //

tb\_item 商品表添加触发器，不允许删除商品，

执行delete 时，抛出错误

create trigger item\_before\_delete

before delete on tb\_item

for each row

begin

-- 抛出错误，阻止删除操作

-- mysql没有抛错语法，这里暴力产生错误

delete from You\_can\_not\_delete\_item;

end

//

delete from tb\_item where id=536563 //

select \* from tb\_item where id=536563 //

商品数量不能修改成负数，否则产生错误

create trigger item\_before\_update

before update on tb\_item

for each row

begin

if new.num<0 then

delete from

Item\_number\_can\_not\_be\_negtive;

end if;

end

//

update tb\_item set num=num-100000

where id=536563 //

select \* from tb\_item where id=536563 //

## 查看触发器

* 进入系统库 information\_schema

use information\_schema

* 查询 triggers 表

select \* from triggers\G

select \* from triggers

where TRIGGER\_SCHEMA='jtdb2'

and EVENT\_OBJECT\_TABLE='tb\_item'\G

## 删除触发器

drop trigger 触发器名

# 作业

商品表 tb\_item

有什么品牌

最贵商品的品牌、名称、价格

分页查看商品

商品名和商品描述

商品分类表 tb\_item\_cat

查询所有的顶层分类

查询 161 下的二层分类

查询 162 下的三层分类

用户表

订单表

订单编号是 20161001490698615071

查询没中类别的商品数量

查询 类别 163 的商品

查询商品价格不大于100的商品名称列表

查询品牌是联想,且价格在40000以上的商品名称和价格

查询品牌是三木,或价格在10以上的商品名称和价格

查询品牌是三木、广博、齐心的商品名称和价格

查询品牌不是联想、戴尔的商品名称和价格

查找品牌是联想且价格大于10000的电脑名称

查询联想或戴尔的电脑名称列表

查询联想、戴尔、三木的商品名称列表

查询不是戴尔的电脑名称列表

查询所有是记事本的商品品牌、名称和价格

查询品牌是末尾字符是'力'的商品的品牌、名称和价格

名称中有联想字样的商品名称

查询卖点含有'爆款'电脑名称

查询开头字母是A的电脑名称

将地址表中的城市、地区、详细地址组合到一起,

获取订单编号为 20161001490698615071 中的每种商品的标题、单价、数量、总价

统计商品表中各个品牌的商品数量

统计商品表中各个品牌的商品数量,并且按照数量从少到多排序

统计商品表中各个品牌的商品数量,去除品牌为NULL的行,并且按照数量从少到多排序

查询不同品牌中最贵的商品价格

将不同品牌中最贵的商品按照价格降序排序

找出不同品牌中最贵的商品的前三名

查询订购了10000028商品的客户姓名和联系方式(三层子查询)

首先查询订单表,根据10000028这个商品编号找到有该商品的订单编号

然后查询订单表,根据上面得到的订单编号找到订购此订单的用户编号

最后查询用户表,根据第二部中查到的用户编号,找到对应的用户名和联系方式

查询没中类别的商品数量

select cid,count(\*) from tb\_item group by cid

查询 类别 163 的商品

select id,title,price from tb\_item where cid=163;

查询商品价格不大于100的商品名称列表

select id,title,price from tb\_item where price<100;

查询品牌是联想,且价格在40000以上的商品名称和价格

select id,title,price from tb\_item where brand='联想' and price>40000;

查询品牌是三木,或价格在50以下的商品名称和价格

select id,brand,title,price from tb\_item where brand='三木' or price<50;

查询品牌是三木、广博、齐心的商品名称和价格

select id,brand,title,price from tb\_item where brand in('三木','广博','齐心');

查询品牌不是联想、戴尔的商品名称和价格

select id,brand,title,price from tb\_item where brand not in('联想','戴尔');

查找品牌是联想且价格大于10000的电脑名称

select id,brand,title,price from tb\_item where brand='联想' and price>10000;

查询联想或戴尔的电脑名称列表

select id,brand,title,price from tb\_item where brand='联想' or brand='戴尔';

查询联想、戴尔、三木的商品名称列表

select id,brand,title,price from tb\_item where brand in('联想','戴尔','三木');

查询不是戴尔的电脑名称列表

select id,brand,title,price from tb\_item where brand<>'戴尔';

查询所有是记事本的商品品牌、名称和价格

select id,brand,title,price from tb\_item where title like '%记事本%';

查询品牌是末尾字符是'力'的商品的品牌、名称和价格

select id,brand,title,price from tb\_item where brand like '%力';

名称中有联想字样的商品名称

select id,brand,title,price from tb\_item where title like '%联想%';

查询卖点含有'爆款'电脑名称

select id,brand,title,price from tb\_item where cell\_point like '%爆款%';

查询开头字母是A的电脑名称

select id,brand,title,price from tb\_item where title like 'A%';

将地址表中的城市、地区、详细地址组合到一起,

select concat(receiver\_city, receiver\_district,receiver\_address) addr from tb\_address;

获取订单编号为 20161001490698615071 中的每种商品的标题、单价、数量、总价

select title,price,num,total\_fee from tb\_order\_item where order\_id='20161001490698615071';

统计商品表中各个品牌的商品数量

select brand,count(\*) from tb\_item group by brand;

统计商品表中各个品牌的商品数量,并且按照数量从少到多排序

select brand,count(\*) c from tb\_item group by brand order by c;

统计商品表中各个品牌的商品数量,去除品牌为NULL的行,并且按照数量从少到多排序

select brand,count(\*) c from tb\_item where brand is not null group by brand order by c;

查询不同品牌中最贵的商品价格

select id,title,brand,price from tb\_item where (brand,price) in(

select brand,max(price) m from tb\_item where brand is not null group by brand);

将不同品牌中最贵的商品按照价格降序排序

select id,title,brand,price from tb\_item where (brand,price) in(

select brand,max(price) m from tb\_item where brand is not null group by brand)

order by price desc;

找出不同品牌中最贵的商品的前三名

select id,title,brand,price from tb\_item where (brand,price) in(

select brand,max(price) m from tb\_item where brand is not null group by brand)

order by price desc

limit 3;

查询订购了10000028商品的客户姓名和联系方式

首先查询订单表,根据10000028这个商品编号找到有该商品的订单编号

然后查询订单表,根据上面得到的订单编号找到订购此订单的用户编号

最后查询用户表,根据第二部中查到的用户编号,找到对应的用户名和联系方式

select id, username, phone, email from tb\_user where id in

(select user\_id from tb\_order where order\_id in

(select order\_id from tb\_order\_item where item\_id=10000028))

select

distinct u.id, u.username, u.phone, u.email

from

tb\_user u join tb\_order o on u.id=o.user\_id

join tb\_order\_item i on o.order\_id=i.order\_id

where

i. item\_id=10000028

商品表 tb\_item

select \* from tb\_item;

有什么品牌

select distinct brand from tb\_item;

最贵商品的品牌、名称、价格

select brand,title,price

from tb\_item

order by price desc

limit 1;

分页查看商品

select id,brand,title,price

from tb\_item

order by price desc

-- limit 0,10;

-- limit 10,10;

-- limit 20,10;

limit 30,10;

商品名和商品描述

select

title,item\_desc

from

tb\_item i join tb\_item\_desc d

on i.id=d.item\_id

limit 1;

商品分类表 tb\_item\_cat

select \* from tb\_item\_cat;

select \* from tb\_item\_cat

where name like '%电脑%' or

name like '%笔记本%' or

name like '%手机%';

查询所有的顶层分类

select \* from tb\_item\_cat

where is\_parent=1 and parent\_id is null;

``查询 161 下的二层分类

select \* from tb\_item\_cat

where parent\_id=161 order by sort\_order;

查询 162 下的三层分类

select \* from tb\_item\_cat

where parent\_id=162 order by sort\_order;

用户表

select \* from tb\_user;

订单表

select \* from tb\_order;

用户 id 是14的订单

select \* from tb\_order where user\_id=14;

订单编号是 20161001490698615071

查询这个订单的所有商品

select \* from tb\_order\_item

where order\_id=20161001490698615071;