目录

[Mysql day02 2](#_Toc500776989)

[1 数据类型 2](#_Toc500776990)

[1.1 字符串 2](#_Toc500776991)

[1.2 数字 2](#_Toc500776992)

[1.3 日期 3](#_Toc500776993)

[1.4 Mysql的数据表所有字段字节量之和不能超过65535 4](#_Toc500776994)

[2 约束 5](#_Toc500776995)

[2.1 主键 5](#_Toc500776996)

[2.1.1 建表时添加主键 6](#_Toc500776997)

[2.1.2 修改表时添加主键 7](#_Toc500776998)

[2.1.3 查看主键约束 7](#_Toc500776999)

[2.1.4 删除主键约束 7](#_Toc500777000)

[2.1.5 自增主键 7](#_Toc500777001)

[2.2 外键 9](#_Toc500777002)

[2.2.1 建表时添加外键 9](#_Toc500777003)

[2.2.2 修改表时添加外键 10](#_Toc500777004)

[2.2.3 查看外键约束 11](#_Toc500777005)

[2.2.4 删除外键约束 11](#_Toc500777006)

[2.3 唯一约束 12](#_Toc500777007)

[2.3.1 创建表时添加唯一约束 12](#_Toc500777008)

[2.3.2 修改表时添加唯一约束 12](#_Toc500777009)

[2.3.3 添加字段组合唯一约束 12](#_Toc500777010)

[2.3.4 查看唯一约束 13](#_Toc500777011)

[2.3.5 删除唯一约束 13](#_Toc500777012)

[2.4 非空约束 14](#_Toc500777013)

[2.4.1 添加非空约束 14](#_Toc500777014)

[2.4.2 查看非空约束 15](#_Toc500777015)

[2.4.3 删除非空约束 15](#_Toc500777016)

[2.5 检查约束 15](#_Toc500777017)

[2.6 默认值 15](#_Toc500777018)

[3 表之间的关系 16](#_Toc500777019)

[3.1 一对一关系 16](#_Toc500777020)

[3.2 一对多 17](#_Toc500777021)

[3.3 多对多 18](#_Toc500777022)

[4 作业 19](#_Toc500777023)

Mysql day02

# 数据类型

## 字符串

char(长度)

定长字符串,最长255字符

存入的字符串长度不足,会补空格

varchar(最大长度)

可变长度字符串,最大65535字节,

如果最大长度是255字符内,在前面会添加一个字节,来表示字符串长度

如果最大长度超过255字符,在前面添加两个字节,来标识字符串长度

一般指定的最大长度不超过255

text

65535字节

blob

大对象类型

## 数字

标记:

unsigned 无符号,正数

zerofill 填充0

tinyint

1字节

smallint

2字节

int

4字节

int(5)

查询显示时,按5位显示,不足5位可以选择补0(zerofill),例如: 23 🡪 00023,超过5位没有影响

bigint

8字节

float

4字节浮点数

double

8字节浮点数

decimal(m, n)

定点数,表示钱,应该用 decimal

总共m位,小数位n位

字节量: m+2

## 日期

date 年月日

time 时分秒

datetime 年月日时分秒

timestamp

时间戳,最大只到2038年,修改数据时,表中第一个timestamp字段会自动更新成系统当前时间,timestamp不能放入空值

* 处理日期数据时,有时也会选择用 bigint 整数来作为日期数据,来标识1970-1-1 0点开始的毫秒值

## Mysql的数据表所有字段字节量之和不能超过65535

* 如果存在允许null(空值)的字段,表中需要一个额外字节来表示
* varchar,255内,需要一个额外字节表示长度,超过255,需要两个额外字节表示长度
* text类型,只占用表的字节量的 10 个字节

测试表的总字节量

mysql -uroot -p

set names gbk;

use test;

-- 创建表

drop table tb1;

create table tb1(

a int, -- 4字节

b double, -- 8字节

c varchar(65520) -- 65523-1-2

)charset=latin1; -- iso-8859-1

drop table tb2;

create table tb2( -- 空,1

a int(3) unsigned zerofill,-- 4

b varchar(255), -- 255\*3+1

c varchar(21587) -- (64764-2)/3

)charset=utf8;

测试timestamp

drop table tb3;

create table tb3(

a int(3) unsigned zerofill,

b timestamp,

c timestamp

);

insert into tb3(a) values(4213);

insert into tb3(a) values(13);

select \* from tb3;

update tb3 set a=14 where a=13;

select \* from tb3;

show create table tb3\G

# 约束

限制字段的取值

五种约束:

* + 主键约束
  + 外键约束
  + 唯一约束
  + 非空约束
  + 检查约束(mysql没有实现)
  + 默认值

## 主键

是一行数据的唯一标识

* + 不重复
  + 不能为空
  + 自动创建索引
  + 尽量使用业务无关的数据作为主键

学生表:

|  |  |  |  |
| --- | --- | --- | --- |
| id(PK) | 学号 | name | gender |
| 1 | 2012009 | xx | x |
| 2 | 2012010 | xx | x |
| 3 | 2012011 | xx | xx |

街道人员管理表:

|  |  |  |  |
| --- | --- | --- | --- |
| id(PK) | 身份证 | name | birth |
| 234526423 | xxxxxxxx | xx | xx |
| 8356345634 | xxxxxxxxx | xx | xx |

电信手机计费表:

|  |  |  |  |
| --- | --- | --- | --- |
| id(PK) | 手机号 | name | 预存 |
| 1 | xxxxx | xxx | xx |

### 建表时添加主键

create table tb4(

id int primary key,

...

);

create table tb4(

id int,

...,

primary key(id)

);

双主键,字段组合作为主键,尽量不用

create table tb4(

name varchar(20),

ip varchar(20),

...

primary key(name, ip)

);

create table tb4(

id int primary key,

name varchar(20)

);

insert into tb4 values

(1, 'a'), (2, 'b'), (3, 'c');

insert into tb4 values(null, 'a');

insert into tb4 values(1, 'a');

### 修改表时添加主键

alter table tb4

add primary key(id);

### 查看主键约束

desc tb4;

show create table tb4\G

### 删除主键约束

alter table tb4

drop primary key;

### 自增主键

向表中插入数据时,不需要提供自增主键的值,它会自动产生连续增加的值.

如果插入null,自动填入新的自增值.

如果插入较大值,以后会从插入的值继续向后递增.

#### 添加自增主键

mysql中,只有主键才允许自增

create table tb5(

id int primary key auto\_increment,

...

);

alter table tb5

modify id int auto\_increment;

新建表,添加自增主键

create table tb5(

id int primary key auto\_increment,

name varchar(20)

);

insert into tb5(name) values

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

select \* from tb5;

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

select \* from tb5;

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

select \* from tb5;

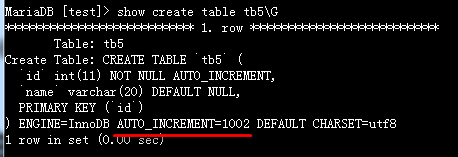
insert into tb5 values(null, 'f');

select \* from tb5;

#### 查看自增值

查看下一个自增值是几

show create table tb5\G



#### 取消自增(不删主键约束)

字段后面不加 auto\_increment 就是取消:

alter table tb5

modify id int;

#### 获得刚刚插入的自增值

使用函数: last\_insert\_id() 函数.

* 没有执行过insert语句,函数返回0
* 多个客户端都产生了自增值,只查询当前会话的自增值

select last\_insert\_id();

set names gbk; -- 通知服务器客户端字符集

use test;

-- 同时向班学生表和练习方式表插入数据

insert into xuesheng(name)

values('张三');

insert into lianxi(xs\_id,tel)

values(last\_insert\_id(),'234563462345');

select \* from xuesheng;

select \* from lianxi;

## 外键

外键引用主键中存在的值,

外键允许重复,也允许null值,

外键自动出创建索引

外键会降低表的访问效率,在大型互联网应用中,数据库表一般都不添加外键约束,用程序来保证数据的正确性

学生表

|  |  |  |
| --- | --- | --- |
| id(PK) | 姓名 | 性别 |
| 1 | 张三 | 男 |
| 2 | 李四 | 女 |
| 3 | 王五 | 男 |

学生练习方式表

|  |  |  |
| --- | --- | --- |
| 学生id(FK) | 电话 | qq |
| 1 | 1412341234 | 1234213 |
| 3 | 23423452345 | 3452345 |

### 建表时添加外键

create table tb5(

id int,

xs\_id int, -- 学生id,引用学生表的主键id

...,

foreign key(xs\_id) references xuesheng(id)

);

### 修改表时添加外键

alter table tb5

add foreign key(xs\_id) references xuesheng(id);

外键测试

-- 创建学生表

drop table xuesheng;

create table xuesheng(

id int primary key auto\_increment,

name varchar(20)

)engine=innodb charset=utf8;

-- 创建班级表

drop table banji;

create table banji(

id int primary key auto\_increment,

name varchar(20)

)engine=innodb charset=utf8;

-- 修改学生表,添加banji\_id外键,引用班级的主键

alter table xuesheng

add (

banji\_id int,

foreign key(banji\_id) references banji(id)

);

insert into banji(name)

values('一班'),('二班'),('三班');

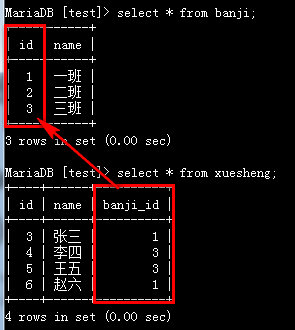
insert into xuesheng(name,banji\_id)

values('张三',1) , ('李四',3)

, ('王五',3) , ('赵六',1);

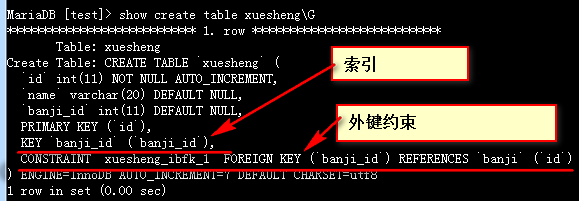
select \* from banji;

select \* from xuesheng;



### 查看外键约束

show create table xuesheng\G



### 删除外键约束

删除外键约束,不会自动删除它的索引

通过约束名(show create table看到的名字)删除外键:

alter table xuesheng

drop foreign key xuesheng\_ibfk\_1;

通过索引名再删除索引:

alter table xuesheng

drop index banji\_id;

## 唯一约束

保证字段中的值不重复,但允许重复的null值

唯一约束会创建索引,如果取消唯一约束,通过删除索引来取消

### 创建表时添加唯一约束

create table tb5(

id int primary key auto\_increment,

email varchar(100) unique,

...

);

create table tb5(

id int primary key auto\_increment,

email varchar(100)

...,

unique key(email)

);

### 修改表时添加唯一约束

alter table tb5

modify email varchar(100) unique;

alter table tb5

add unique key(email);

### 添加字段组合唯一约束

create table tb5(

...,

name ...,

ip ...,

...,

unique key(name, ip)

);

alter table tb5

add unique key(name, ip);

唯一约束测试

-- 班级表name唯一

-- 如果已经存在重复值,不能添加唯一约束

alter table banji

add unique key(name);

insert into banji(name)

values('a'),('b');

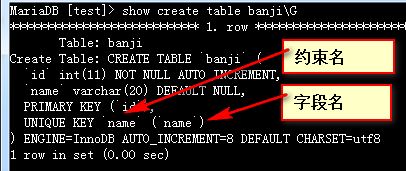
insert into banji(name)

values('a'),('b');



### 查看唯一约束

show create table banji\G



### 删除唯一约束

使用唯一约束的索引名来删除

alter table tb5

drop index name;

alter table banji

drop index name;

insert into banji(name)

values('a'),('b');

select \* from banji;

## 非空约束

约束字段不能取null(空)值

### 添加非空约束

create table tb5(

...,

email varchar(100) not null,

...

);

alter table tb5

modify email varchar(100) not null;

学生的name非空

-- 如果字段中已经存在null值,

-- 不能添加非空约束

alter table xuesheng

modify name varchar(20) not null;

-- 失败,name 非空

insert into xuesheng(name,banji\_id)

values(null, 2);



### 查看非空约束

desc xuesheng;

show create table xuesheng\G

### 删除非空约束

alter table tb5

modify email varchar(100) null;

-- 不写也可以

alter table tb5

modify email varchar(100);

## 检查约束

设置检查条件,判断存入字段中的值,是否符合要求

mysql支持检查约束的语法,但没有实现检查约束

对年龄范围进行检查: >=7 并且 <=60

对性别取值检查: ='男' 或者 ='女'

create table tb5(

...,

age int,

gender char(1),

...

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

check(gender='男' or gender='女')

);

## 默认值

设置了默认值的字段,如果插入数据时不提供数据,会自动插入默认值

create table tb5(

...,

deleted tinyint default 0,

...

);

默认值测试

-- 学生表添加status字段

-- 1-正常 2-休学 3-删除

alter table xuesheng

add status int default 1;

insert into xuesheng(name)

values('a'),('b');

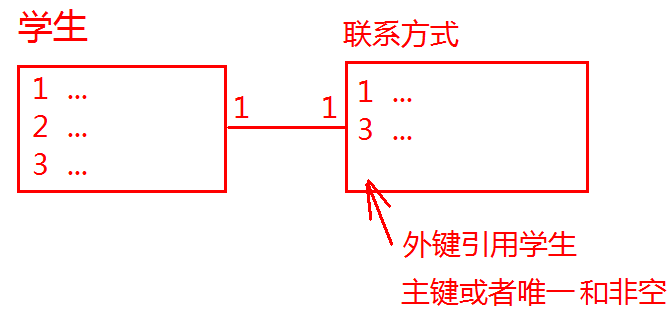
select \* from xuesheng;

# 表之间的关系

* 一对一
* 一对多(多对一)
* 多对多

## 一对一关系

通过具有唯一约束的外键来体现



学生的联系方式表

create table lianxi(

xs\_id int primary key,

tel varchar(20),

qq varchar(20),

foreign key(xs\_id)

references xuesheng(id)

)engine=innodb charset=utf8;

-- 为存在的学生添加联系方式

insert into lianxi(xs\_id, tel)

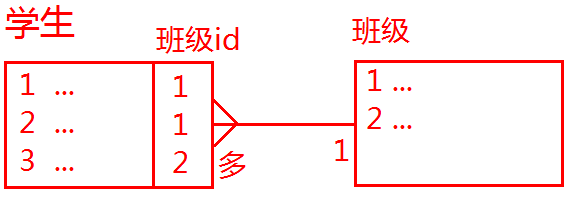
values(3,'234523'),

(5,'34256345'),

(6,'45674567');

## 一对多

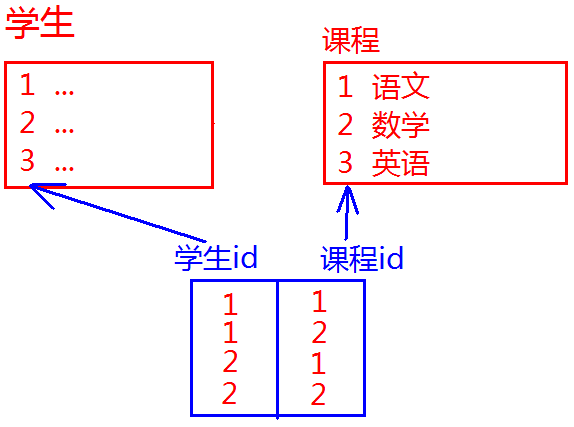
同过多方表中的外键来体现



## 多对多

同过多对多关系中间表体现

关系表中添加两个外键字段分别引用两张表的主键



多对多关系

-- 课程表和关系中间表

drop table if exists kecheng;

create table kecheng(

id int primary key auto\_increment,

name varchar(20)

)engine=innodb charset=utf8;

drop table if exists xs\_kc\_link;

create table xs\_kc\_link(

xs\_id int not null,

kc\_id int not null,

foreign key(xs\_id)

references xuesheng(id),

foreign key(kc\_id)

references kecheng(id),

unique key(xs\_id, kc\_id)

)engine=innodb charset=utf8;

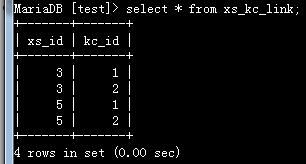
insert into kecheng(name)

values('语文'), ('数学');

insert into xs\_kc\_link

values(3,1), (3,2), (5,1), (5,2);

select \* from xs\_kc\_link;



# 作业

4.2.1 商品分类表(tb\_item\_cat)

id自增主键

parent\_id引用当前表主键id

status和is\_parent有默认值

4.2.2 商品表(tb\_item)

id自增主键

cid非空

status默认值

4.2.3 商品描述表(tb\_item\_desc)

item\_id主键,也是外键引用商品id

4.2.8 用户表(tb\_user)

4.2.9 收藏夹表(tb\_collect)

4.2.10 购物车表(tb\_cart)