**MYSQL面试宝典**

## 一.数据规范化

**1.第一范式**

**二维表格，不存在重复的行与列**

**2.第二范式**

**满足第一范式，不存在部分依赖**

**3.第三范式**

**满足第二范式，不存在传递依赖**

1. **二.sql分类**

**1.DDL:针对于数据库对象：数据库,表,列.关键字：create,alter,drop**

**2.DML:针对于表数据操作，关键字：insert ,delete,update**

**3.DCL:针对于访问权限和用户**

**4.DQL:针对于表查询，关键字：select ,from ,where**

## 三.命令

**DDL:**

**1.登陆mysql数据库**

**mysql -uroot -proot**

**-u:用户**

**-p:密码**

**2.查看有哪些数据库**

**show databases;**

**3.删除yz数据库**

**drop database yz;**

**4.使用哪个数据库**

**use mysql;**

**5.查看数据库下有哪些表**

**show tables;**

**6.查看user这张表字段**

**desc user;**

**7.创建数据库**

**create database yz;**

**8.创建表**

**create table teacher(name varchar(20) not null,age int,sex varchar(10) );**

**9.删除表**

**drop table student;**

**10.添加表字段**

**alter table teacher add column iphone int;**

**alter table teacher add column money decimal(5,2);//一共5位，小数点后2位**

**alter table teacher add column birthday datetime;**

**11.删除表字段**

**alter table teacher drop column iphone;**

**12.修改表字段**

**alter table teacher change birthday birth date;// birthday字段改为 birth**

**DML:**

**1.插入表数据**

**insert teacher values('aa',18,'a');**

**2.修改表数据**

**update teacher set age=38 where name='aa';**

**3.删除表数据**

**delete from teacher where name = 'cc';**

**4.清空表数据**

**delete from teacher;**

**truncate table teacher;**

**DQL:**

**1.查询数据**

**select \* from teacher;**

## 四.约束（保证数据的合理性）

**1.not Null 非空约束，不能为null**

**2.unique 唯一约束 ， 不能重复**

**3.primary key 主键约束 ，唯一标识这条数据**

**4.foreign key 外键约束，用来连接两张表，这张表的外键一定是另一张表的主键**

**5.check 检查约束，判断数据是否合理**

**6.default 默认约束 ，如果你不设置，则插入默认值**

## 五. unique 唯一约束 与 primary key 主键约束 的区别

**1.一张表里只能有一个主键，可以有多个唯一约束**

**2.唯一约束可以为空，但只能有一个空值，主键不可以**

**六.添加约束**

**1.为teahcer添加tno为主键**

**alter table teacher add column tno int primary key;**

**2.添加一个唯一约束**

**alter table teacher add constraint uk\_name unique(name);**

**3.删除唯一约束**

**alter table teacher drop index uk\_name;**

**4.删除主键**

**alter table teacher drop primary key;**

**5.新增一个主键**

**alter table teacher add constraint pk\_tno primary key(tno);**

**6.添加外键**

**alter table student add constraint fk\_tno foreign key(tno) references teacher(tno);**

**七.自增主键**

**create table student(**

**s\_no int primary key auto\_increment,**

**s\_name varchar(20),**

**s\_sex varchar(5),**

**s\_age int**

**)**

**添加一条信息**

**insert student(s\_name,s\_sex,s\_age) values('李文辉','男',20);**

## 八.数据库备份与还原

**mysqldump -uroot -proot --databases yz > /home/test/yz.sql**

**备份多个数据库：**

**mysqldump -uroot -proot --databases yz myschool > /home/test/yz.sql**

**数据库还原：**

**mysql -uroot -proot < /home/test/JavaCourse/mysql/xiaoshuai.sql**

**九.自增主键的重新排序（不存在外键关联）**

**alter table student drop column s\_no;**

**alter table student add column s\_no int primary key auto\_increment first;**

**十.普通查询**

**1.算术运算(as 起别名，可以省略)**

**select s\_name,s\_age\*3+1 as '你能活到多大' from student;**

**select \* from student where s\_no = 2 + 1;**

**2.比较运算符**

**select \* from student where s\_sex <> '男';**

**select \* from student where s\_age > 30;**

**select \* from student where s\_age >=20 and s\_age<=30;**

**select \* from student where s\_age between 20 and 30;**

**select \* from student where s\_age in (20,30,25);**

**select \* from student where s\_sex is null;**

**select \* from student where s\_name like '\_大%';**

**select \* from student where s\_name regexp '.大';**

**3.逻辑运算符**

**select \* from student where not s\_sex='男';**

**select \* from student where not s\_sex='男' and s\_age>20;**

**select \* from student where s\_age = 20 or s\_age = 30;**

**select \* from student where s\_age != 20 xor s\_age != 30;**

**4. in 与 or 的区别**

**没有区别，可以互换，当数据量小的时候用or,数据量大用in**

**select \* from student where s\_age in (20,30);**

**select \* from student where s\_age = 20 or s\_age = 30;**

**5./\*找出20岁以上男人，20岁以下女人\*/**

**select \* from student where s\_sex='男' xor s\_age<20;**

**select \* from student where (s\_sex='男' and s\_age>=20) or (s\_sex='女' and s\_age<20);**

**十一.字符串常用函数**

**1.连接字符串**

**select concat(s\_name,'学生') from student;**

**2.字符串替换（位置从1开始）**

**select insert(s\_name,1,2,'藏獒') from student where s\_name='葬爱勇少';**

**3.转换成大写**

**select upper(s\_name) from student where s\_name= 'a';**

**4.首尾拼接**

**select lpad(s\_name,7,'傻傻的') from student where s\_name='葬爱勇少';**

**select rpad(s\_name,7,'傻傻的') from student where s\_name='葬爱勇少';**

**5.截取**

**select substring(s\_name,1,2) from student where s\_name='葬爱勇少';**

## 十二.表连接

**1.内连接**

**1.1等值内连接**

**/\*李文辉是哪个班的\*/**

**select s.s\_name,c.c\_name from student s inner join class c on s.c\_no = c.c\_no where s.s\_name='李文辉';（第一种写法）**

**select s.s\_name,c.c\_name from student s,class c where s.c\_no = c.c\_no and s.s\_name='李文辉';（第二种写法）**

**/\*李文辉报考了哪些门课\*/**

**select s\_name,sb\_name from student s,result r ,subject sb where s.s\_no=r.s\_no and r.sb\_no = sb.sb\_no and s.s\_name='李文辉';（第一种写法）**

**select s\_name,sb\_name from (student s inner join result r on s.s\_no = r.s\_no)inner join subject sb on r.sb\_no = sb.sb\_no where s.s\_name='李文辉';（第二种写法）**

**/\*每一门课有多少人报考\*/**

**select sb\_name,count(\*) from student s,result r ,subject sb where s.s\_no=r.s\_no and r.sb\_no = sb.sb\_no group by sb\_name;**

**1.2不等值内连接**

**select s.s\_name,c\_name from student s,class c**

**where not s.c\_no > c.c\_no and not s.c\_no < c.c\_no and s.s\_name='李文辉';**

**2.外连接**

**2.1左外连接**

**/\*查看各门课的成绩\*/**

**select sb\_name,score from subject s left join result r on s.sb\_no = r.sb\_no;**

**/\*各老师的代课平均分\*/**

**select t\_name,avg(score) from teacher t left join subject sb on t.t\_no = sb.t\_no left join result r on r.sb\_no = sb.sb\_no group by t\_name**

**2.2 右外连接**

**/\*学生的分班情况\*/**

**select s\_name,c\_name from class c right join student s on s.c\_no = c.c\_no;**

## 十三.子查询

**1.单行子查询**

**/\*李文辉在哪个班\*/**

**select c.c\_name from class c where c\_no = (select c\_no from student where s\_name = '李文辉');**

**/\*极客巅峰有哪些学生\*/**

**select s\_name from student where c\_no = (select c\_no from class where c\_name='极客巅峰');**

**/\*查询出年龄最大的学生\*/**

**select s\_name from student where s\_age = (select max(s\_age) from student);**

**2.多行子查询**

**/\*查询大于5人的班级学生信息\*/**

**select \* from student where c\_no in (select c\_no from class where c\_num>5);**

**/\*查询比女生年龄都大的男生\*/**

**select \* from student where s\_age > ALL (select s\_age from student where s\_sex='女') and s\_sex = '男';**

**/\*查询比女生年龄大的男生\*/**

**select \* from student where s\_age > ANY (select s\_age from student where s\_sex='女') and s\_sex = '男';**

**2.多列子查询**

**/\*查询年龄大于25的男生\*/（不要在意这些细节，知识点重要）**

**select \* from student where (s\_sex,s\_age) in (select s\_sex,s\_age from student where s\_sex='男' and s\_age>25);**

**3.嵌套子查询**

**/\*嵌套在where后面\*/**

**select \* from student where c\_no = (select c\_no from class where c\_name='极客巅峰')**

**/\*嵌套在from后面\*/（不要在意这些细节，知识点重要）**

**select \* from (select s\_name from student) s;**

**/\*嵌套在select后面\*/（不要在意这些细节，知识点重要）**

**select (select s\_name from student where s\_name='李文辉') from student where s\_name='李文辉';**

**/\*极客巅峰JAVA谁考了第一\*/**

**select s\_name,sb\_name,score from student s,class c,subject sb,result r where s.c\_no=c.c\_no and sb.sb\_no = r.sb\_no and r.s\_no=s.s\_no and c.c\_name='极客巅峰' and sb.sb\_name='java' and r.score=(select max(score) from result r,subject sb where r.sb\_no = sb.sb\_no and sb.sb\_name='java' );**

**/\*每门学科的状元\*/**

**select s\_name,sb\_name,score from student s,result r,subject sb where s.s\_no = r.s\_no and r.sb\_no = sb.sb\_no and (sb\_name,score) in (select sb\_name,max(score) m from result r ,subject sb where r.sb\_no = sb.sb\_no group by sb.sb\_name);(第一种写法)**

**select s\_name,sb\_name,score from student s,result r,subject sb,(select sb\_name sn,max(score) ms from result r ,subject sb where r.sb\_no = sb.sb\_no group by sb.sb\_name) m where s.s\_no = r.s\_no and r.sb\_no = sb.sb\_no and sb.sb\_name = m.sn and m.ms = r.score;(第二种写法)**

## 十四. case when用法

## /\*男女性别互换\*/

**select s\_name,case s\_sex when '男' then '女' when '女' then '男' end from student;**

**/\*学生各门课的考试等级\*/**

**select s\_name,sb\_name,score,case when score>90 then 'A' when score>80 then 'B' when score>70 then 'C' else 'D' end '等级' from result r,student s,subject sb where r.s\_no = s.s\_no and r.sb\_no = sb.sb\_no;**

## 十五.索引

**1.相当于书的目录，加快查询速度**

**２.索引添加在字段上，建在经常where条件中使用的字段上(如，查询学生姓名where name=’xx’,可以在name创建索引)**

**３.创建索引能加快查询速度，但是耗费系统空间，因为需要很大一块空间来存放索引**

**４.经常插入修改数据的表，不建议创建索引，索引会降低插入和修改的效率**

**查看索引：**

**show index from student;**

**十六.索引分类**

**１.普通索引**

**创建普通索引：**

**create index name\_index on student(s\_name);**

**删除索引：**

**alter table student drop index name\_index;**

**２.唯一索引**

**如果创建了主键，唯一，外键约束，那就会自动创建主键索引，唯一索引，外键索引**

**创建唯一索引：**

**create unique index uq\_index\_no on student(s\_no);**

**３.组合索引**

**创建组合索引（适合多列子查询）：**

**create index index\_no\_name on student(s\_no,s\_name);**

**十七.视图**

**１.它是一张虚拟的表，基于select查询**

**２.简化复杂的查询(对非常复杂的查询创建视图，下次直接查询视图即可)**

**３.可以做权限控制，屏蔽掉一些敏感的字段**

**基于复杂的sql创建视图：**

**create view first\_view as select sb\_name,s\_name,score from student s,result r,subject sb where s.s\_no=r.s\_no and r.sb\_no=sb.sb\_no and (sb\_name,score) in (select sb\_name,max(score) from result r,subject sb where r.sb\_no=sb.sb\_no group by sb\_name);**

**基于原表创建视图（修改视图数据会影响原表数据）：**

**create view stu\_view as select s\_no,s\_name,s\_sex,c\_no from student;**

**查询视图就跟普通表一样：**

**select \* from first\_view where sb\_name='JAVA';**

**修改视图：**

**alter view stu\_view as select s\_no,s\_name,s\_sex from student;**

**删除视图：**

**drop view stu\_view;**

十八.触发器  
**１.被动，不是由用户执行，而是满足条件后自动执行一系列sql   
2.触发条件: insert ,update ,delete   
3.触发时间可以分为：after , before**

**创建触发器(student表插入一条数据触发在student\_bak表里备份一条数据):**

delimiter //

create trigger tr\_stu1

after insert on student

for each row

begin

insert student\_bak(s\_no,s\_name,c\_no,c\_name,score)

values(new.s\_no,new.s\_name,new.c\_no,new.c\_name,new.score);

end //

**创建触发器(student表删除数据触发在student\_bak表里备份数据):**

delimiter EOF

create trigger tr\_stu\_del\_bak

after delete on student

for each row

begin

insert student\_bak(s\_no,s\_name,c\_no,c\_name,score)

values(old.s\_no,old.s\_name,old.c\_no,old.c\_name,old.score);

end EOF

delimiter //:定义结束标识符为//，碰到//代表结束，也可以定义成其他的   
new ：代表新插入的数据   
old :代表删除的旧数据   
删除触发器:

drop trigger tr\_stu\_del\_bak;

**十九.存储过程**   
1．为了处理业务逻辑，相当于JAVA里的方法 

2．循环判断加上一堆sql语句

3.有参存储过程(求两个参数和)：

delimiter //

create procedure pro\_sum(i int ,j int)

begin

select i+j '求和';

end //

调用存储过程：call pro\_sum(7,8);

4.无参存储过程：

delimiter //

create procedure pro\_sum()

begin

declare i int;

declare j int;

set i = 5;

set j = 6;

select i+j '求和';

end //

调用存储过程：call pro\_sum();

5.插入5000条数据

delimiter //

create procedure auto\_addStu(num int)

begin

declare i int;

set i = 0;

while i < num do

insert student(s\_no,s\_name,s\_age) values(i+1,concat('Zara',i+1),round(rand()\*20+10));

set i = i + 1 ;

end while;

end//

调用存储过程：call auto\_addStu(5000);

**二十.事务**   
1.要不都执行，要么都不执行。   
2.要么提交commit，要么回滚rollback。

开启事务：

start transaction;

insert t\_user(name,sex,age,buildtime) values('张三','男',18,20171202125339);

insert t\_user(name,sex,age,buildtime) values('张三','男',18,20171202125339);

insert t\_user(name,sex,age,buildtime) values('张三','男',18,20171202125339);

insert t\_user(name,sex,age,buildtime) values('张三','男',18,20171202125331);

rollback;