Test

1. 修改表名： alter table list rename lists;
2. 修改表的某一列名：alter table lists change jiangjin jiangjing double;
3. 添加某一列：alter table lists add ok char(10);
4. 删除表的某一列； alter table lists drop ok;
5. 查看现在处于哪个库中：select database();
6. 将表中的人按工资的多少排序：select name,salary from lists order by salary;
7. 将表中每个人的年薪分别算出来：select name, salary\*12 as nianxin from lists;
8. 算出表中低于平均工资的人：

select name,salary from lists where salary<(select avg(salary) from lists);

8.找出每个部门中的工资最高的人：

select deptno ,max(sal) from emp group by deptno;

9.找出年龄一样大的人中工资高于平均值的：

select age, salary from lists group by age having salary>avg(salary);

10.将表中的一个数据改掉：

A：update list set bumen=1 where name=tom;

B：update list set bumen=2 where name in('zhang','wang');

1. 删除某一行的数据： delete from list1 where id=4;
2. 去除重复的数据： select distinct salary from list;
3. 表的内连接： select name,other from list join list1 on list.bumen=list1.bumen1;
4. 找出每一个员工的部门名称，要求显示员工名和部门名：（表的等值连接）

select e.ename,d.dname from emp e join dept d on e.deptno=d.deptno;

1. 找出每个员工的工资等级和工资：（表的非等值连接）

select e.ename,e.sal,s.grade from emp e join salgrade s on e.sal between s.losal and s.hisal;

1. 找出每个员工的上级领导。（表的自连接）

select e.ename as emp,b.ename as boss from emp e left join emp b on e.mgr=b.empno;

1. 删除一个全是null的行：delete from student name is null;

34道经典的mysql题

1. 取得每个部门最高薪水的人员名单：select e.ename,m.maxsal from emp e right join (select max(sal) as maxsal from emp group by deptno) m on e.sal=m.maxsal;

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

| ename | maxsal |

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

| KING | 5000.00 |

| SCOTT | 3000.00 |

| FORD | 3000.00 |

| BLAKE | 2850.00 |

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

1. 那些员工的薪水在部门平均薪水之上：select e.ename,e.sal,t.avgsal,t.deptno from emp e join (select deptno,avg(sal) as avgsal from emp group by deptno) t on e.deptno=t.deptno where e.sal>t.avgsal;

3.取得部门中所有人的平均薪水等级（1先算部门的平均工资然后看等级）：select t.deptno,t.avgsal,s.grade from (select deptno,avg(sal) as avgsal from emp group by deptno) t join salgrade s on t.avgsal between s.losal and s.hisal;

3.取得部门中所有人的平均薪水等级（2先查出工资等级然后算平均）：select t.deptno,avg(t.grade) from (select e.ename, e.deptno, s.grade from emp e join salgrade s on e.sal between s.losal and s.hisal) t group by t.deptno;

4.

MySQL

1.端口：

IP是电脑的唯一编号，端口是软件的唯一编号。

Localhost等同于127.0.0.1表示的是本机的IP地址。

MySQL：3306 下载的端口：21 上网的端口：80 Oracle：1521

1. DB: DataBase （数据库，数据库实际在硬盘上以文件形式存在）

DBMS：DataBase Management System （数据库管理系统）

SQL：结构化查询语句，是个高级语言，通用的标准语言。

3.表：是数据库的基本组成单元。

一个表包括行和列，行是数据，列是字段。

每一个字段包括 字段名 数据类型 相关的约束。

相关约束： primary key auto\_increment 主键自增长

SQL的分类

DQL（数据查询语言）：查询语言，凡是select语句都是DQL。

DML（数据操作语言）：insert delete update，对表中的数据进行增删改。

DDL（数据定义语言）：create drop alter ，对表的结构的增删改查。

TCL（事务控制语言）：commit 提交事务，rollback回滚事务。

DCL（数据控制语言）：grant 授权，revoke撤销权限。

show databases 查看有哪些数据库

create database powernode 创建我们自己的数据库

use powernode 进入这个数据库

show tables 查看当前的数据库中有哪些表

source + .sql文件（sql脚本文件）（将文件拖进来就ok）

drop database powernode 删除库

desc +表名 查看表结构

\c 结束一条语句

Ctrl+c 强行停止

exit 退出MySQL

show create table +表名 查看创建表的语句

以上的都不是SQL语句

SQL语句：

查询

完整的DQL语句执行顺序

select 5

from 1

where 2

group by 3

having 4

order by 6

limit 7

简单查询：

select 字段名1，字段名2，字段名3， from 表名；

提示：

1.任何一条Sql语句都以 ；结尾。

2.Sql语句不区分大小写。

3.字符串最好用单引号括起来。（其实双引号也可以但是别的数据库可能用单引号所以养成习惯）

4.可以参与运算。（单行运算中只要有null参加运算结果必是null）

5.as 关键字可以省略。

select ename , sal\*12 as ‘年薪’ from emp；

条件查询：

select 字段，字段 from 表名 where 条件；执行顺序：先from，然后where，最后select

between 1000 and 3000 是闭区间[1100~3000]（必须左小右大）如果应用在字符上为左闭右开。

and 并且 or 或者 运算顺序不确定的时候加括号

拿到工资大于3000的人

select ename from emp where sal > 3000;

在数据库中null不是一个值。必须使用is null 或者 is not null

in（1000,5000）代表的是1000或者5000不是1000~5000

模糊查询

关键字 like

在模糊查询中%是任意多个字符，\_代表任意单个字符。

select ename from emp where ename like ‘%Y’;

排序

关键字 order

select ename，sal from emp order by sal 升序

select ename，sal from emp order by sal asc 升序

select ename，sal from emp order by sal desc 降序

按照工资的降序排序，当工资相同的时候在按照名字的升序排列（by后面可以跟数字，数字为第几列，然后按第几列排序）

select ename，sal from emp order by sal ，ename asc；

select ename，sal from emp order by 2；

注意：越靠前的字段越能起到主导作用，只有当前面的字段无法完成排序的时候，才会启用后面的字段。

找出工资工作岗位是SALESMAN的员工，并且要求按照薪资的降序排列。

Select ename，job，sal from emp where job =‘SALESMAN’ order by sal desc；

先执行from 然后 where 再 select 最后 order by

分组函数（）

count计数 sum求和 avg平均值 max 最大值 min最小值

count(\*) :不是统计某个字段中数据的个数，而是统计总记录条数。（和某个字段无关）

count(某个字段)：表示统计字段中不为null的数据总数量。

所有的分组函数都是对某一组数据进行操作的。

找出工资总和

select count(sal) from emp;

又叫多行处理函数，输入多行，输出一行。

分组函数自动忽略null。

分组函数不能直接使用在where语句当中。

select ename, sal from emp where sal > avg (sal);

以上这句会报错：无效的使用了分组语句。

因为group by 是在where 执行之后才会执行。

分组函数后面默认有一个缺省的group by，只是没有给任何分组的条件。

分组函数也可以组合起来用。

Select count(\*), sum(sal), avg(sal) from emp;

ifnull()空处理函数

ifnull（可能为null的数据，被当做什么处理）

计算每个人的年薪。

select ename ，(sal + ifnull(comm,0)) \*12 as ‘年薪’ from emp；

group by 和 having

group by : 按照某个字段或者某些字段进行分组。

having ：是对分组之后的数据进行再次过滤。

找出每个工作岗位的最高工资。

select max(sal) from emp group by job;

注意：分组函数一般会和group by 联合使用，这也是为什么它被称为分组函数的原因。

并且任何一个分组函数（count sum avg max min）都是在group by 语句执行结束之后才会执行的。当分组函数sql语句中没有group by是只会产生一组数据。

当一条语句中有group by的话，select后面只能跟分组函数和参与分组的字段。

如果可以在where 后面过滤的先过滤后再用group by分组 效率高

如果不行 就最后在使用having 过滤。having 是group by的搭档。

只有出现group by 在能使用having。

去重关键字distinct

distinct关键字只能出现在所有字段的最前面。

Select ename, distinct deptno ,job from emp

上述语法错误，因为字段ename和去重后的deptno的条数不一致无法显示出来，所以产生语法错误。

distinct表示后面的字段联系起来去重。

Select distinct deptno , job from emp

上述的sql语句的意思是将deptno和job两个字段联合起来去重。（就是既deptno相同且job相同的去掉重复）

连接查询

笛卡尔积现象：

select ename, dname from emp ,dept;

上述的sql语句执行完后会生成 ename和dname乘积的数目条数。

可以给起个别名，防止去别的表查找字段名，从而提高效率并增加可读性。

select e.ename, d.dname from emp e, dept d;

内连接之等值连接

特点：条件是等量关系。

语法： ...A join B on 连接条件 where ...

join 前面省略了一个inner，有inner 可以直观的知道是内连接。

（可以给表先起个别名 a和b，便于程序查找，提高效率）

select a.name,b.other from list a join list1 b on a.bumen=b.bumen1;

内连接中的非等值连接

找出每个员工的工资等级和工资：（表的非等值连接）

表的自连接

select e.ename,e.sal,s.grade from emp e join salgrade s on e.sal between s.losal and s.hisal;

找出每个员工的上级领导。（表的自连接）

select e.ename as emp,b.ename as boss from emp e join emp b on e.mgr=b.empno;

外连接

概念：其中有一个表为主表，一个表为副表，主要查询主表中的数据，捎带查询副表，当副表中的数据没有和主表中的数据匹配上，副表会自动模拟出null与之匹配。

左连接（left）：表示左边的这张表是主表。

右连接（right）：表示右边的这张表为主表。

案例：找出每个员工的上级领导。（所有员工都要查询出来最大的领导显示为null）

select e.ename as ‘员工’, b.ename as ‘领导’ from emp e left join emp b on e.mgr=b.empno;

（此处的left表示左连接，以左边的表为主表，主表的数据不能丢失）

三张表的连接

案例：找出每一个员工的部门名称以及工资等级。

select e.ename,d.dname,s.grade from emp e join dept d on e.deptno=d.deptno join salgrade s on e.sal between s.losal and hisal;

案例：找出每一个员工的部门名称，工资等级，以及上级领导。（因为king没有上级领导所以有可能会丢失king的数据，因此对于e1来讲把e设置为外连接它的主表）

select e.ename,d.dname,s.grade,e1.ename as boss from emp e join dept d on e.deptno=d.deptno join salgrade s on e.sal between s.losal and s.hisal left join emp e1 on e.mgr=e1.empno;

子查询

where 后面用嵌套子查询。（简单不做赘述）

from后面用嵌套子查询

案例： 找出每个部门的平均薪水的薪资等级。

第一步：现找到每个部门的平均薪资。

select d.dname ,avg(e.sal) as pinjun from emp e join dept d on e.deptno=d.deptno group by e.deptno;

第二步：将部门的平均薪资划分等级。

Select t.dname, t.pinjun, s.grade from (select d.dname ,avg(e.sal) as pinjun from emp e join dept d on e.deptno=d.deptno group by e.deptno) as t join salgrade s on t.pinjun between s.losal and s.hisal;

案例：找出每个部门的平均的薪资等级。

Select d.dname,avg(s.grade) from emp e join salgrade s on e.sal between s.losal and s.hisal join dept d on d.deptno=e.deptno group by e.deptno;

select后面嵌套子查询

Union（可以将查询结果相加）

案例：找出工作岗位是salesman 和 manager的员工

Select ename from emp where ename=’salesman’ union Select ename from emp where ename=’manager’;

将上述的两个查询结果相加，看起来好像按照工作岗位排序了。

注意：union使用的时候要先保证他们的列的数目是一致的，要不然会报错。

Limit（重点，分页查询全靠它了）

Limit是MySQL是特有的，其他的数据库中没有，不通用。（oracle中有一个相同的机制，叫做rownum）

语法：

Limit startIndex, length（length是取的数据的个数，不是结束的索引）

startIndex表示的起始位置，从0开始，0表示第一条数据。

取前5个：

Select ename ,sal, from emp order by sal desc limit 0,5;

前面的起始位置不写默认从0开始。

Select ename ,sal, from emp order by sal desc limit 5;

Limit语句是sql语句中最后执行的一个环节。

案例：找出工资排名在第四到第九名的员工？

Select ename from emp order by sal desc limit 3,6;

通用的分页显示：

每页显示pagesize条记录： 第pageNo页：（pageNo-1）\*pageSize, pageSize

DDL语句

创建表

建表语句的语法格式：

create table 表名( 字段名1 数据类型,字段名2 数据类型);

关于MySQL当中字段的数据类型？常见的

1. int 整数型
2. bigint 长整型（java中的long）
3. float 浮点型
4. char定长字符串
5. varchar 不定长字符串 （最长255）
6. date 日期类型 （对应java.sql.Date类型）
7. BLOB 二进制大对象（存储图片、视频等流体媒体信息）Binary Large Object（对应java中的Object）
8. CLOB 字符大对象（存储较大文本，比如，可以存储4G的字符串。对应java中的Object）

varchar和char的选择：

char适合于字段数据长度不发生改变的时候是定长的，例如：性别、生日等都是采用char

char(5)代表不管来几个数据都是给分配5个长度存储，前提也是不能超过框号中的数字5，否则就报错。

varchar(5)代表不管来几个数据先用if来判断，然后有几个数据给分配几个空间存储，前提也是不能超过框号中的数字5，否则就报错。（因为每次要用if判断几个字符，所以相对char来讲，效率较低）

insert语句插入数据语法格式

insert into 表名(字段1，字段2，...) values(值1，值2，...);

要求：字段的数量和值的数量相同，并且数据类型要对应相同。

当一条insert语句执行完后，表中必然多一行数据。

一次插入多行数据：insert into student values(10,’boy’),(11,’girl’);

表的复制

将查询结果当做新表创建出来。

create table 表名 as select语句;

create table student as select ename ,sal from emp;

将查询的结果插入一个现有的表中。（字段要统一有点像union）

insert into student select ename,sal from emp;

修改数据

语法格式：

update 表名 set 字段名1=值1，字段名2=值2.... where 条件；

update emp set comm=200 where ename='CLARK';

注意：没有条件将会把整个表更新。

删除数据

语法格式：

delete from 表名 where 条件；

注意：没有条件将全部删除。（只是删除的表中的数据，表的结构还在，叫清空数据比较准确）（删除的速度比较慢 ）

删除大表中的数据：truncate table 表名; // 表被截断，不可回滚，永久消失。

删除表：drop table 表名;

create（创建） delete（删除） update（修改） select（查询）

约束

非空约束（not null）：约束的字段不能为null，非空约束没有表级约束。

唯一约束（unique）：约束的字段不能重复，但可以为null。

在列后面直接加unique为列级约束。

For example：（name char(255) unique, code int unique）表示name 和code在各字段的分组中分别不能重复。

对个字段联合起来添加1个约束unique为表级约束。

（name char(255) , code int, unique(name,code)）表示name和code在表中的某一行不能连起来重复。

主键约束（primary key）：约束的字段既不能为null，也不能重复（简称PK）

一行数据的唯一标识。

主键自增后面加 auto\_increment （从1开始，以1递增）

外键约束（foreign key）：...（简称为FK）

语法：foreign key(子表中的某个字段) references 父表名(父表的某个字段（这个字段可以不是主键，但必须unique具有唯一性）)

子表的某个字段添加了外键约束，指向了父类的某个字段。

删除数据的时候，先删除子表，再删除父表。

添加数据的时候，先添加父表，再添加子表。

创建表的时候，先创建父表，再创建子表。

删除表的时候，先删除子表，再删除父表。

检查约束（check）：注意Oracle数据库有check约束，但是MySQL没有，目前MySQL不支持该约束。

存储引擎

存储引擎就是数据在数据库中的存储方式，默认的存储引擎是

常见的存储引擎

1. MyISAM

优点：可被压缩，节省空间。

这种的存储引擎不支持事务。

是MySQL最常用的存储引擎，但是这种引擎不是默认的。

采用三个文件组织一张表。

1. InnoDB

是缺省的默认引擎。

优点：支持事务、行级锁、外键等。

1. Memory

缺点：不支持事务，数据容易丢失，所有的数据和索引被存储在内存当中。

事务

\*一个事务是一个完整的业务逻辑单元，不可再分。（两条DML语句必须同时成功或者同时失败，不允许出现一条成功，一条失败）

\*和事务相关的语句只有：DML语句。（insert delete update）（改变数据的）

开启事务机制：（开始）

\*执行insert 语句——>（这个执行成功之后，把这个执行的记录到数据库的操作历史当中，并不会向文件中保存数据，不会真正的修改硬盘上的数据，暂时存在内存中）

\*执行update语句——>（同上）

\*...................

\*提交事务或者回滚事务。（结束）

以上从开始到结束为一个整体。（完成结束后提交到硬盘）

事务的TCL语句（commit提交 rollback回滚）

事务的四大特性：（ACID）

A：原子性：事务是最小的工作单元，不可再分。

C: 一致性：事务必须保证多条DML语句同时成功或者同时失败。

I：隔离性：事务A和事务B之间具有隔离。

D: 持久性：最终的数据必须持久化到硬盘文件中，事务才算成功的结束。

事务之间的隔离性

\*有4个级别

1. 读未提交（read uncommitted）（A事务不用结束比如改变数据后不commit 。B个事务就可以看到A事务改变后的数据）

2.读已提交（read committed）（A事务改变数据commit后，B事务才可以看到A事务改变后的数据）

3.可重复读（repeatable read）（A事务改变数据commit后，B事务进来看到的数据还是之前它进来时的数据（幻想数据），看不到A事务改变后的数据）

4.序列化读（serializable）（A事务没有commit，B事务根本查不了关闭一直在这里闪，等A事务提交后，B事务立马就可以查出来）

Oracle 数据库默认的隔离级别是：读已提交。

MySQL数据库默认的隔离级别是：可重复读。

Mysql事务默认是自动提交的。（执行一条DML语句就自动提交事务）（关闭自动提交start transaction）

使用两个事务的隔离级别

1. 设置事务的隔离级别

set global transaction isolation level + 隔离级别; 例如：read uncommitted

1. 查看事务的全局隔离级别

select @@global.tx\_isolation;

索引

通过索引可以快速的找到对应的资源。

添加索引的条件：

\*数据量庞大（根据客户的需求）

\*该字段很少的DML操作。（添加索引是给字段添加，数据被修改后，索引的要重新排序，影响效率，所以一般修改少的添加索引）

\*该字段经常出现在where语句中。（经常根据哪个字段查询）

注意：主键和具有unique约束的字段会自动添加索引。（根据主键查询效率较高，尽量根据主键检索）

在sql语句前面加explain 可以看到查询的具体步骤。

案例：给薪资sal字段添加索引。

create index emp\_sal\_index on emp(sal); emp\_sal\_index 为索引名随便起的

删除索引对象：drop index 索引名称 on 表名; oracle数据库不需要后面加（on 表名）就可以直接删除。

索引底层采用的数据结构是：B + Tree

索引的实现原理：

通过 B tree 缩小扫描的范围，底层索引进行了排序，分区，索引会携带数据在表中的“物理地址”，最终通过索引检索到数据之后，获取到关联的物理地址，通过物理地址定位表中的数据，效率是最高的。

索引的分类

单一索引：给单个字段添加索引

复合索引：给多个字段联合起来添加一个索引

主键索引：主键上会自动添加索引

唯一索引：有unique约束的字段会自动添加索引

索引的失效

模糊查询的时候第一个字段使用的是%，这个时候索引是失效的。

视图

视图：站在不同的角度去看到数据。（同一张表的数据，通过不同的角度去看待）

创建视图：create view myview as select empo ,ename from emp;

删除视图：drop view myview;

注意：只有DQL语句才能以视图的对象的方式创建出来。

对视图进行的增删改查，会影响到原表的数据。（通过视图影响原表数据的，不是直接操作的原表）

视图的作用：

视图可以隐藏表的实现细节。保密级别较高的系统，数据库只对外提供相关的视图，java程序员只对视图的对象进行CRUD（insert delete update select）

DBA（DataBase Administrator）命令

将数据库当中的数据导出

在windows的命令窗口中执行：

导出制定的库：

mysqldump bjpowernode >D:\bjpowernode.sql -uroot -p123456

导出制定的表：

mysqldump bjpowernode emp >D:\bjpowernode.sql -uroot -p123456

导入数据：

create database bjpowernode;

use bjpowernode;

source 将文件拖过来进dos窗口

数据库设计三范式

按照三范式设计的表不会出现数据冗余

第一范式；

任何一张表都应该有主键，并且每一个字段原子性不可再分。

第二范式：

建立在第一范式的基础之上，所有的非主键字段完全依赖主键，不能产生部分依赖。（复合主键就是不符合第二范式的经典案例）

多对多设计范式：三张表，关系表两个外键。

第三范式：

建立在第二范式的基础之上的，所有的非主键字段直接依赖主键，不能产生传递依赖。

一对多设计范式：两张表，多的表加外键。

注意：在实际的开发中，以满足客户的需求为主，有的时候会拿冗余换取执行速度。

（把一张大表拆成了两个小表，虽然解决了数据的冗余，但是在查表的时候会产生笛卡尔现象，导致执行的速度变低）

一对一设计：（将客户经常查询的数据建一个表，其他的信息建新一个表（比如用户登录表和用户详细信息表））

第一种方案：主键共享（一个表的主键作为另一个表的主键及外键）

第一种方案：外键唯一（给一个表添加一个字段，这个字段添加另一个表主键的外键约束，并且再加唯一性约束（unique））