my.ini MySQL的配置文件

什么是sql，定义了操作所有关系数据库的规则

每一种数据库的操作方式存在不一样的地方，称为’方言’

sql的通用语法

1.sql语句可以单行或多行书写，以分号结尾。

2.mysql数据库的sql语句不区分大小写，关键字建议使用大小写

3.三种注释

单行注释： – 注释内容

#注释类内容

多行注释：/\*注释\*/

sql分类

DDL(操作数据库和表)

DML（增删改表中的数据）

DQL（查询表中的数据）

DCL（授权）

## 操作数据库

### 查询所有的数据库

show database;

### 查看数据库的创建语句

show create database 数据库名称；

### 创建数据库

create database 数据库名称;

create datadase if not exists 数据库名称;

### 创建数据库并设设置字符集

create database 数据库名称 character set utf8;

### 修改数据库的字符集

alter database 数据库名称 character set 字符集名称；

### 删除数据库

drop database 数据库名称;

drop database if exists 数据库名称；

### 查询当前正在使用的数据库

select database();

### 使用数据库

use 数据库名称；

## 操作表

### 查询某个数据库中所有的表名称

show tables ;

### 查询表结构

desc 表名；

### 创建表

crate table 表名（

列名1 数据类型，

。。。

列名2 数据类型

）；

### 数据类型

int 整数类型

double 小数类型

比如 score double（5，2）即999.99

日期类型 date 只包含年月日 yyyy-mm-dd

datetime 包含年月日时分秒

timestamp与datetime相似，区别timestamp如果不赋值或为null，则使用当前系统的默认时间，自动赋值

varchar 字符串

varchar(20)姓名最大20个字符 张三 2个字符

## 修改表

### 修改表名

alter table 表名 rename to 新的表名；

### 修改表的字符集

alter table 表名 character set 字符集名称；

### 添加一列

alter table 表名add 列名 数据类型；

### 修改类的名称 类型

alter table 表名change 列名 新列名 新数据类型；

alter table表名 modify 列名 新数据类型；

### 删除列

alter table 表名 drop 列名；

## 增删改表中的数据

### 添加数据

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

注意：列名和值要一一对应。

简化：如果不定义列，则默认给所有的列添加值

insert into 表名values(值1，值2)；

除了数字类型，其他类型用引号，单双都行

### 删除数据

delete from 表名 [where 条件]；

注意：如果不加条件，删除表中的所有记录

truncate table 表名； 删除表，然后创建一个一模一样的空表

### 修改数据

update 表名 set 列名1=值1，列名2=值2 [where 条件]；

注意如果不加条件，就会修改表中的所有记录

### 查询记录

语法：

select

字段列表

from

列表列名

where

条件列表

group

分组字段

having

分组之后的条件

order by

排序

limit

分页限定

## 基础查询

### 多字段的查询

select 字段，字段2 from 表名；

### 去除重复

select distanct字段 from表名；

### 计算列

一般可以使用四则运算计算一些列的值，一般字符自会进行数值的计算

select 字段 from 表名

例如 select math，english，math+english from student；

注意有null的计算，结果都为null

select math，english，math+ifnull（english，0） from student；

ifnull(表达式1，表达式2)

表达式1：判断该字符是否为null

表达式2：该字段为null 的替换值

### 起别名

select math，english，math+ifnull（english，0） as 总分 from student；

## 条件查询

where子句后面跟条件查询

### 运算符

### between。。。and

select \* from student where age between 20 and 30;

左闭右闭？

### in（集合）

select \* from student where age in (22,25,18);

### like

模糊查询

\_:单个任意字符

%：多个任意字符

select \* from student where name like “马%”;

### is null

select \* from student where english is null;

select \* from student where english is not null;

值为null 的不能使用预算符

and 或 &&

or 或 ||

not 或 ！

### 排序查询

order by 排序字段 拍寻方式，排序字段1 排序方式1

默认是升序asc

降序desc

select \* from student order by math desc；

如果有多个排序条件，则只会在前面条件排序是一样的才会判断第二条件

### 聚合函数

将一列数据作为一个整体，经行纵向的计算

count：计算个数

max：计算最大值

min：计算最小值

sum：计算和

avg：计算平均值

replace:替换 把'病假' 替换为 '--'：

UPDATE users SET username=REPLACE(username,'病假','--')  WHERE username LIKE '%病假%';

表 tb1里面的 f1字段的abc替换为def：

UPDATE tb1 SET f1=REPLACE(f1, 'abc', 'def');

select count（name） from student；

注意所有的函数都是排除null值

解决：选择非空的列，或使用ifnull

select count（ifnull（english，0）） from student；

mod(id,2) **id 为奇数**

dateDiff(a.RecordDate,b.RecordDate) = 1

算法 MySQL 使用 DATEDIFF 来比较两个日期类型的值

### 分组查询

语法：

group by 分组字段

GROUP BY X意思是将所有具有相同X字段值的记录放到一个分组里。

| gender | count(\*) |
| --- | --- |
| 女 | 3 |
| 男 | 3 |

GROUP BY X, Y意思是将所有具有相同X字段值和Y字段值的记录放到一个分组里。

select gender,score,count(\*) from user group by gender,score;

结果

| gender | score | count(\*) |
| --- | --- | --- |
| 女 | 92 | 1 |
| 女 | 98 | 2 |
| 男 | 60 | 1 |

注意：查询的字段：分组字段，聚合函数

按照性别分组，分别查询男，女同学的平局分

select sex，avg(math) from student group by sex;

按照性别分组，分别查询男，女同学的平局分 要求低于70的人，不参与分组

select sex，avg(math) from student where math>70 group by sex;

按照性别分组，分别查询男，女同学的平局分 要求低于70的人，不参与分组

,分组之后人数要大于2个人

select sex，avg(math) from student where math>70 group by sex having count（id）>2

where和having的区别

1.where在分组之前进行限定，如果不满足条件，则不参与分组。having在分组时候进行限定，如果不满足结果，则不会被查询出来

where 后不可以跟聚合函数，having可以经行集合函数的判断。

### 分页查询

语法：limit 开始的索引，每页数据条数

select \* from student limit 0，3；第一页

select \* from student limit 3，3；第二页

公式：开始的索引=（当前页面-1）\*每页条数；

分页操作时一个”方言”

语句1：select \* from student limit 9,4

// 语句1和2均返回表student的第10、11、12、13行

语句2：slect \* from student limit 4 offset 9

//语句2中的4表示返回4行，9表示从表的第十行开始

约束

是对表中的数据进行限定，保证数据的正确性，有效性和完整性

分类：

### 主键约束 primary key

主键约束，非空且唯一

一张表只能由一个字段为主键

主键是表中记录的唯一标识

删除主键

alter table stu drop primary key;

创建完表后，添加主键

alter table stu modficy id int primary key；

### 自动增长

如果某一列是数值类型，使用auto\_increment可以完成值的自动增长

insert into stu values(null,’ccc’);

也可以手动赋值id

删除自动增长

alter table stu modify id int;

因为主键不能删除，所以会删除自动增长

添加自动增长

alter table stu modify id int auto\_increment;

### 非空约束not null

一种方法：在创建表时添加约束

二种方法：alter stu modifcy name varchar（20 ）not null；

删除约束 更新列

alter table stu modficy name varchar（20）；

### 唯一约束 unique

值不能重复

null表示不确定

唯一限定的值可以有多个null

删除唯一约束

alter table stu drop index 列名

可以在创建表后添加约束

alter stu modifcy name varchar（20 ）unique；

### 外键约束 foreign key

数据冗余

表的拆分

在创建表时，可以添加外键,让表与表产生关系，从而保证数据的正确性

create table 表名(

。。。。

外键列

constraint 外键名称 foreign key （外键名称） references 主表的名称（主表列名称）

);

例如

create table employee(

in int primary key auto\_increment,

name varchar(20),

age int,

dep\_id int, -- 外键字段

contraint emp\_dept\_fk foregin key(dep\_id) references department(id)

)

查看表的建表语句可以查看外键

show create table 表名;

删除外键

alter 表名drop foreign key emp\_dept\_fk（外键名称）；

添加外键

alter table 表名 add contraint emp\_dept\_fk（外键名称） foregin key(外键字段) references 主表名称(主表列名称)

### 级联操作

外键可以为null但是不能为不存在的外键

添加外键，设计级联更新

alter table 表名 add contraint emp\_dept\_fk（外键名称） foregin key(外键字段) references 主表名称(主表列名称) on update cascade

级联删除

alter table 表名 add contraint emp\_dept\_fk（外键名称） foregin key(外键字段) references 主表名称(主表列名称) on delete cascade

级联删除和级联更新可以一起设置

## 数据库的设计

多表之间的关系

一对一

可以在任意一方添加唯一外键指向另一方的主键，一般不会创建两张表

一对多（多对一）

在多的一方建立外键，指向一定一方的主键。

多对多

需要借助第三张中间表，中间表至少包含两个字段，这两个字段作为第三张表的外键，分别指向两张表的主键。

数据的默认值设定 defalut

联合主键

primary key(rid,uid)

## 数据库设计的范式

设计数据库时需要遵循的规范

第一范式：每一列是不可分割的原子数据项

第二范式：在1NF的基础上，非码属性必须完全依赖于候选码（在1nf基础上消除非主属性对主码的部分函数依赖）

概念

函数依赖：如果通过a属性（属性组）的属性值，可以确定唯一的B属性值，则称B 依赖于a

例如：学号依赖于姓名 学号- - >姓名

属性组：通过学号和科目得到分数 （学号，课程名称）- - >分数

完全函数依赖 a- - >b 如果a是一个属性组，则b属性确定需要依赖a属性组中 的所有属性值

例如 （学号，课程名称）- - >分数

部分函数依赖 如果a是一个属性组，则b属性确定需要依赖a属性组中的某些属性第三范式

例如 （学号，课程名称）- - >姓名

传递函数依赖a- ->b ,b - - >c如果通过a属性（属性组）的属性值，可以确定唯一b属性的值，通过b属性（属性组）可以确定唯一c属性的值，则称c传递函数依赖

例如：学号- - >系名，系名 - - >系主任

码:如果在一个表中，如果一个属性或一个属性组，被其他所有属性完全依赖，则称这个属性（属性组）为该表的码

例如，通过学号和课程名称可以确定学生 的姓名，系名，系主任，课程名称，分数

主属性：码属性组中的所有属性

非主属性：除过码属性组的属性

第三范式：

在2nf基础上，任何非主属性不依赖于其他非主属性（在2nf上消除传递依赖）

数据库的备份和还原

1.命令行的方式

备份语法： mysqldump –u用户名 –p密码 数据库的名称>保存的路路径

还原：

登入数据库

常见数据库

使用数据库

执行文件。

source 文件路径

## 多表查询

查询多张表的结果集称为笛卡尔积：

有两个集合，A和B,取a和b所有的组合情况

消除无用的数据

多表查询的分类

### 内连接查询

INNER JOIN子句将它与t2表的每一行进行比较，以检查它们是否都满足连接条件。当满足连接条件时，INNER JOIN将返回由t1和t2表中的列组成的新行。

相连接的两张表地位平等

如果一张表中在另一张表中不存在对应数据，则不做连接

from子句后面直接出现多个表名，这种连接方式即属于内连接，是隐式连接

显示连接：select col\_list from table1[inner] join table2 on table1.col=table2.col

显示内连接比隐式内连接相同情况下更快。on是连接条件的限定

隐式内连接 :使用where 条件消除无用的数据

查询所有员工信息和对应的部门信息

select `name`,`age`,`sex`,`salary`, `hire\_date`,`dname` from`person`,`dept` where `dept`.did=`person`.id;

显式内连接

语法：select 字段列表 from 表名 1 inner join 表名2 on 条件

inner可以省略

注意事项:条件是什么

从那些表中查询

查询那些字段

### 外连接查询

左外连接：

语法：select 字段列表 from 表1 left outer join 表2 on 条件

outer可省略

查询的是左表的所有数据以及其交集部分

查询所有员工的信息，如果员工有部门则查询部门名称，没有部门则不显示部门名称

select t1.\*,t2.name from emp as t1 left join dept as t2 on t1.dept\_id =t2.id;

右外连接：

select 字段列表 from 表1 right outer join 表2 on 条件

### 子查询

查询中嵌套查询

查询工资最高的员工信息

1.查询最高的工资是多少

select max(salary) from emp;

2.查询员工信息，并且工资等于9000的

select \* from emp where salay=9000;

select \* from emp where salay=( select max(salary) from emp);

子查询不同的结果

1.子查询的结果是单行单列的

子查询可以作为条件，使用运算符取判断

查询员工工资小于平均工资的人

select \* from emp where salary<（select avg（salary）from emp）

2.子查询的结果是多行单列的

子查询可以作为条件,使用in来判断

查询财务部和市场部所有的员工信息

select id from dept where name=”财务部” or name=”市场部”;

select \* from emp where dept.id in(1,3);

select \* from emp where dept.id in (select id from dept where name=”财务部” or name=”市场部”);

3.子查询的结果是多行多列的

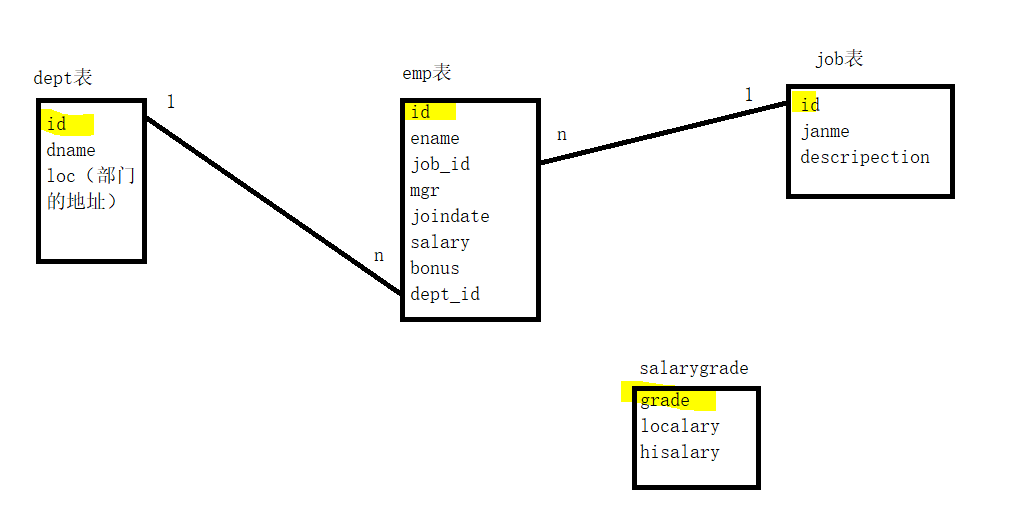
子查询可以作为一个虚拟表

查询员工入职日期是2011-11-11日之后的部门信息

select \* from emp where dept.date>’2011-11-11’

select \* from emp t1 ,(select \* from emp where dept.date>’2011-11-11’) t2 where t1.id=t2.dept.id

多表查询练习



查询所有员工信息，查询员工编号，员工姓名，工资，职位名称，职务描述

select t1.id,t1.ename,t2.jname,t2.description

from

emp t1,job t2

where

t1.job\_id=t2.id;

查询员工编号，员工姓名，工资，职务描述，部门名称，部门位置

select t1.id,t1.ename,t2.jname,t2.description,t3.dname,t3.loc

from

emp t1,job t2,dept t3

where

t1.job\_id=t2.id;

and

t1.dept\_id=t3.id;

查询员工姓名，工资，工资等级

select

t1.ename,

t1.salary

t2.grade

from

emp t1,salaryGrade t2

where

t1.salary between t2.losalary and t2.hissalary;

查询员工姓名，工资，职务名称，职务描述，部门名称，部门位置工资等级

select

from

emp t1,dept t2,job t3,salaryGrade t2

where

t1.job\_id=t2.id

and t1.dept\_id=t3.id

and t1.salary between t2.losalary and t2.hissalary;

查询出部门编号，部门的名称，部门的位置，部门人数

使用分组查询count（id）

select

t1.id,t2,dname

t1.loc,t2.total

from

dept t1,

(select

,dept\_id,count(id) as total

from

emp

group by(dept\_id)) t2,

where t1.id=t2.dept\_id;

查询所有员工的姓名及其上级的姓名，没有领导的员工也要查询。

分析

1.姓名 emp，直接上级的名称 emp

\*emp 表的id和mgr是自关联的

2.条件emp.id=emp.mgr;

3.查询左表的所有数据

select

t1.ename,

t2.mgr,

t2.id,

t2.ename

from

emp t1

left join emp t2 on

t1.mgr=t2.id

数据库事务

事务的基本概念

如果一个包含多个步骤的业务操作，被事务管理，那么这些操作要么同时成功，要么同时失败

操作

开启事务：strat transaction 要手动的滚回和提交

执行事务

发现问题，执行回滚事务命令，

回滚：rollback

没有问题，提交事务命令

提交：commit

在mysql数据库中事务默认自动提交

一条DML（增删改）语句会自动提交一次事务

事务的提交方式：手动提交：需要开启事务，Oracle是手动提

自动提交：mysql会自动提交

查看事务默认的提交方式

select @@autocommit;

1,代表自动提交

0，代表手动提交

修改提交方式

修改为手动提交

set @@autocommit=0

每次执行需要手动提交

事务的四大特征

1原子性：是不可分割的最小单位，要么同时成功，要么同时失败

2持久性：当事务提交和回滚后，数据库会持久化保存数据

3隔离性:多个事务之间相互独立

4.一致性：事务的操作数据总量不变

事务的隔离级别（了解）

概念：多个事务之间隔离的，相互独立的，但是多个事务操作同一批数据则会引发问题

设置不同的隔离级别就可以解决这些问题

存在问题：

脏读：一个事务读取到另一个事务中没有提交的数据（事务的数据没有提交就读到了）

不可重复读（虚读）：在同一事务中，两次读取到的数据不一样

幻读：一个事务操作（DML）数据表中所有的记录，另一个事务添加了一条数据，则第一个事务查询不到自己的修改

隔离级别

read uncommitted：读未提交

产生问题：脏读，不可重复读，幻读

read committed：读已提交。只有提交数据，另一个事务才会读到（oracle默认）

产生问题: 不可重复读，幻读

repeatable read :可重复读（mysql默认）

产生问题:幻读

serializable:串行化

可以解决所有问题

注意：隔离级别从小到大，安全性越来越高，效率越来越低

数据库设置隔离级别

set global transaction isolation level 级别字符串；

查询隔离级别

select @@tx\_isolation;

事务的隔离级别演示：

在一个窗口设置事务的隔离级别为read uncommitted 为A窗口，

打开另一个窗口，这时事务的隔离级别为read unCommitted，为窗口b

两边都开启事务

张三在A窗口转账500；

李四在b窗口查询：结果为500，1500；出现脏读：在另一个窗口看到未提及的事务结果

张三在A窗口回滚

李四在窗口查询：结果1000，1000；

两次查询不一样，出现不可重复读：

在一个窗口设置事务的隔离级别为read committed 为A窗口，

打开另一个窗口，这时事务的隔离级别为read Committed，为窗口b

两边都开启事务

张三在A窗口转账500；

李四在b窗口查询：结果为1000，1000；数据没有发生变化

张三在A窗口提交

李四在窗口查询：结果500，1500；

两次查询不一样，出现不可重复读：

在一个窗口设置事务的隔离级别为repeatable read为A窗口，

打开另一个窗口，这时事务的隔离级别为repeatable read，为窗口b

两边都开启事务

张三在A窗口转账500；

李四在b窗口查询：结果为1000，1000；数据没有发生变化

张三在A窗口提交

李四在窗口查询：结果1000，1000；

李四提交事务（关闭事务）：

李四再次查询：结果500，1500

在一个窗口设置事务的隔离级别为serializable为A窗口，

打开另一个窗口，这时事务的隔离级别为serializable，为窗口b

两边都开启事务

张三在A窗口转账500；

李四在b窗口查询：等待A窗口的提交

张三在A窗口提交

b窗口查询立即显示：结果500，1500；

DCL管理用户，授权

union

用于连接两个以上的 SELECT 语句的结果组合到一个结果集合中。多个 SELECT 语句会删除重复的数据。

查询和"01"号的同学学习的课程完全相同的其他同学的信息

重点是GROUP\_CONCAT( )函数，它可以分组，并将条件值连接起来

比如

SELECT GROUP\_CONCAT(c\_id) FROM Score WHERE s\_id = '01';

结果为

|  |
| --- |
| GROUP\_CONCAT(c\_id) |
| 01,02,03 |

SELECT

Student.\*

FROM

Student

WHERE

s\_id IN (select s\_id from score GROUP BY s\_id HAVING GROUP\_CONCAT(c\_id) = (

SELECT GROUP\_CONCAT(c\_id) FROM Score WHERE s\_id = '01')

);