**MySql数据库**

* + 掌握MYSQL基本知识
  + 学习数据库管理和操作
  + 掌握数据库设计方法
  + 熟练运用SQL编程语言

**数据库**

* + 对大量信息进行管理的高效解决方案
  + 按照数据结构来组织、存储和管理数据的库

数据库=数据+管理工具

DBS = DB + DBMS

Database System = Database + database Manager System

**关系型数据库**：建立在关系模型上的数据库

当将日常生活中的实体和实体的属性保存到数据库中时，该如何

处理该实体结构

关系模型体现在两个方面：

1.数据结构可以规定（同类数据结构一致，典型的二维表格）

2.数据之间的关系可以指定

**概念**

数据库：数据的仓库

表：数据是存储在表内的，保存在一个表内的数据应该具有相同的数据结构

记录（行）：一行代表一个完整的记录（记录数据）

字段（列）：表示记录（实体）的属性(规定数据格式)

SQL：结构化查询语言，用来管理数据的语言

**Mysql安装配置**

2017年7月5日

14:22

步骤

Step1:解压缩mysql-5.7.17-winx64.zip文件到指定目录

Step2:将my-default.ini文件改为my.ini（mysql配置文件）

basedir = D:\\mysql-5.7.17-winx64

datadir = D:\\mysql-5.7.17-winx64\\data

basedir代表mysql的安装路径

datadir：数据存放目录（需要自己新建一个用于保存数据文件的文件夹）

Step3:初始化mysql：

进入mysql安装目录的bin目录：执行mysqld --initialize

Step4:安装进windows的服务管理器中，随着windows开机自动启动

Mysqld --install

设置root用户密码

1.跳过认证信息（开启服务）

Mysqld --skip-grant-tables

2.跳过认证信息连接mysql服务器

Mysql -hlocalhost -P3306 -uroot -p

3.选择mysql数据库

Use mysql;

4.重新设置root用户的密码

update user set authentication\_string = password('886887') where user='root';

5.刷新权限

Flush privileges;

6.退出登录，重新登录

以认证方式登陆后，如果出现

You must reset your password using ALTER USER statement before executing this statement

需执行该语句：

Set password = password('886887');

Tip：

连接信息

如果地址默认为localhost，端口默认为3306，"-hlocalhost -P3306"参数可以省略

设置环境变量

**常用操作**

**C/S：client/server 客户端/服务端**

开启服务端：

1.dos窗口下执行mysqld命令

--console

--debug

2.通过windows的服务管理进行启动(services.msc)

a.

b.通过命令行（高大上）

net start mysql（启动mysql服务）

net stop mysql

如何查看服务是否开启

1.可以使用netstat -ano查看默认端口3306是否启动

2.任务管理器：taskmgr

停止：

1.taskkill /pid 3196 /f

2.通过任务管理器终止进程

客户端连接流程

1.Mysql -hlocalhost -P3306 -uroot -p

2.向mysql发送SQL指令

3.负责接收sql指令并执行

4.生成执行结果并发回给客户端

5.客户端显示执行结果

2,3,4步骤可以重复执行

我们当前已经安装配置好了mysql，进入bin

守护进程(daemon)

服务器端：

客户端：

mysql常用工具：

step1：先开启服务

开启后是否有一个守护进程啊

要通信，首先要连接

**SQL：Structured Query Language**

2017年7月24日

13:48

SQL：结构化查询语言，用来管理数据的语言

操作对象不同，可以分为：

数据管理语言（DML）

数据定义语言（DDL)

数据控制语言 (DCL)

数据库操作

2017年7月24日

13:57

**创建数据库**：

语法：**create database 数据库名称** [数据库选项信息];

选项信息：

1.字符集charset

2.校对规则collation

容错方式创建：

create database [if not exists] 数据库名称 [数据库选项信息]

.opt文件：保存数据库选项信息

Tip:

每条语句以;或者\G结束

Tip:

可以使用字符限定符将标识符限定起来，可以使用空格，中文（需要字符集支持），关键字来作为标识符

Create database `create`;

**修改数据库**

语法：alter database 数据库名称 [修改指令]

**查看数据库的创建信息**

**show create database** test;

标识符（数据库名）命名规则：

1.大小写取决于当前操作系统

2.见名知意（使用下划线）

[原则上可以使用任意的字符]

查看数据保存目录

每当创建一个数据库都会创建一个相应的文件夹

因为大小写的问题，建议使用下划线分割

**数据库查询**

Show databases;

**带有条件的查询**

Show databases **like** '%e%'

Show databases like '\_e%'

**通配符：**

%：任意位数的任意字符

\_:一个位数的任意字符（不包括空字符）

系统自身维护的一些数据库，保存一些状态，维护信息等，非常重要，不要删除

**删除数据**：

**Drop database test;**

**Drop database [if exists]** test;

**表操作**

2017年7月24日

14:49

**创建表**

**create table [if not exists] 表名称(列结构）[表选项]**

列结构：

列名 数据类型(长度) [列的属性，约束]

简单的数据类型：

数值型

字符串型

日期类型

表选项

存储引擎

默认字符集

数据库是表的容器，在对表操作之前，需要指定数据库操作：

**Use 数据库名称**

**多个列定义用,分割 , 最后一个列定义不能有,**

**查看表的创建信息**

**Show create table** student;

**查看表**

**Show tables;**

**删除表**

Drop table 表名;

Drop table if exixts 表名

**修改表**

**a.修改表列定义（修改表结构）**

Alter table student [add|modify|change|drop] 列定义

1.增加字段：alter table student add stu\_name varchar(20);

2.修改列定义：alter table student modify stu\_no char(6);

可以同时修改字段名可以采用change

Alter table student change stu\_no stu\_id char(6);

3.删除列字段：

Alter table student drop stu\_id;

**b.修改表选项**

Alter table student [新的表选项]

Alter table student charset=gbk;

c.修改表名

Rename table student(旧表名称) to students(新表名称);

**查看表结构信息**

**Desc 表名;**

**数据类型**

2017年8月3日

14:54

Date：日期

格式：YYYY-MM-DD

取值范围：1000-01-01到9999-12-31

定义方法：出生日期 dete

支持2位的年

70-99:1970-1999

00-69:2000-2069

insert into test3(sex,born\_date) values('male','92-10-20');

Time，时间类型

取值范围：-838:59:59-838:59:59

包含两层含义：

1.一天中的某个时间

2.时间间隔，可以定义天

Datetime 日期时间类型

格式：YYYY-MM-DD HH:MM:SS

Year,年类型

格式：YYYY

取值范围：1901到2155

Timestamp，时间戳

取值范围:1970-01-01 00:00:00’到2038-01-19 03:14:07

存储时：int

表示时：日期时间的格式

1.如果插入值得时候没有给timestamp指定值，那么它会插入默认值current\_timestamp

2.修改时会触发on update current\_timestamp

3.一个表里只能有一个timestamp字段设置为current\_timestamp

4.如果表里只有一个字段设为timestamp，会自动变成current\_timestamp

字符串

2017年8月3日

14:55

|  |  |  |
| --- | --- | --- |
| 字符串 | Char(8) | Varchar(8) |
| '' | 8 | 1 |
| 'a' | 8 | 2 |
| 'abcdefghi' | 8 | 9 |

char：定长字符串

格式：char(L)

L代表长度，小于等于255

varchar：可变长度字符串

格式：varchar(L)

L代表长度

L<65535,具体的长度和编码有关

Text

大文本类型

格式：text

L < 65535

Blob：

保存二进制大对象数据

Enum枚举类型

实际上也是字符串，通常用于保存单选数据

定义：sex enum('male','female','mars');

枚举的值是有索引的，是从1开始

Set：集合类型

一般用于复选数据

Hobby set('movie','novel','beauty')

对于包含多个SET元素的值，当插入值时元素所列的顺序并不重要。在值中一个给定的元素列了多少次也不重要。当以后检索该值时，值中的每个元素出现一次，根据表创建时指定的顺序列出元素。

**数值型**

2017年8月3日

14:56

数值型-整型

**Tinyint**

1个字节

范围-128~127

|  |  |  |  |
| --- | --- | --- | --- |
| **类型** | **字节** | **最小值** | **最大值** |
|  |  | **(带符号的/无符号的)** | **(带符号的/无符号的)** |
| TINYINT | 1 | -128 | 127 |
|  |  | 0 | 255 |
| SMALLINT | 2 | -32768 | 32767 |
|  |  | 0 | 65535 |
| MEDIUMINT | 3 | -8388608 | 8388607 |
|  |  | 0 | 16777215 |
| INT | 4 | -2147483648 | 2147483647 |
|  |  | 0 | 4294967295 |
| BIGINT | 8 | -9223372036854775808 | 9223372036854775807 |
|  |  | 0 | 18446744073709551615 |

**无符号的定义**

可以使用**unsigned**标识没有符号

定义显示宽度，可以达到统一显示的目的

需要0填充：前导0填充

**Zerofill（一定会带上unsigned属性）**

Tinyint(M) zerofill,

**M代表显示宽度**

**布尔值**

可以使用tinyint(1)来表示

非0值表示为真，0表示为假

浮点型

float单精度

Float(M,D)

M代表总位数

D代表小数位

浮点型

double双精度

定点数

DECIMAL（M,D）

M：总位数，如果不指定，默认为10

D：小数位数如果不指定，默认为0

小数也可以zerofill

可以使用科学计数法

**数据操作（重点）**

2017年8月3日

14:58

Tip：

1.可以不定义字段列表，那么代表我们的值必须和定义字段时的顺序进行相对应

2.字符串值需要用''包裹起来

**创建数据（插入数据）**

**Insert into** 表名（字段列表） values （值列表）

**Insert into 表名 set 列=值，列=值**

-- 插入指定列数据

insert into student(stu\_no) values('9527');

-- 如果吧不定义列，那么插入的值必须和定义列时的顺序要求一致，字符串用‘’

insert into student values ('9528','华安');

-- 插入多条数据

insert into student(stu\_no,stu\_name) values('1001','张无忌'),

('1002','金毛狮王'),

(null,null);

insert into student set stu\_no='1004',stu\_name='刘德华';

create table if not exists student(

stu\_no char(6),

stu\_name varchar(20)

);

**获得数据**

Select 字段列表 from 表名 [查询条件]

通配符\*代表所有字段

查询条件：**where**

如果不指定查询条件，那么默认代表为真： where 1

-- 查询所有信息（通配符\*代表所有列字段）

select \* from student;

select \* from student where 1;

select stu\_name from student;

select stu\_name from student where stu\_no='9528';

**删除数据**

**Delete from 表名 [删除条件]**

如果不指定删除条件，那么默认为：where 1

需要在逻辑上严格限定条件，否则为全部删除

**truncate table 表名**

清空表数据，保留表的结构定义

delete from student where 1;

delete from student where stu\_name is null;

truncate table student;

修改数据

**Update 表名 set 列=值，列=值[where 条件]**

**如果不指定条件:where 1**

update student set stu\_name = '华安' where stu\_no='9527';

update student set stu\_name='张三丰';

update student set stu\_name='张三丰' where 1;

**字符集及校对规则（了解）**

2017年8月3日

14:59

字符集

定义：*字符集*是一套符号和编码

查看服务器所所有的字符集：

Show variables like '%character%

可以在my.ini配置文件中配置服务器端字符编码（默认的字符编码）

在[mysqld]

character\_set\_server=utf8

校对规则

每一套字符集都有与之相关的校对规则

什么是校对规则：

在当前的字符集下，字符之间的比较顺序

规则:

字符集\_地区\_ci|cs|bin

Ci:不区分大小写

Cs:区分大小写

bin：字节比较（编码比较）

每一个字符都有默认的校对规则，如果没有指定校对规则，会采用该字符集下的默认校对规则

create database d1 charset=gbk collate=gbk\_bin;

use d1;

create table t(c varchar(2));

insert into t values('B'),('a'),('c');

select \* from t order by c;

create table t1(c varchar(2)) charset=gbk collate=gbk\_chinese\_ci;

insert into t1 values('B'),('a'),('c');

select \* from t1 order by c;

**列属性（\*）**

2017年8月3日

15:00

**列属性**

**列定义**

列名 数据类型 列属性

**默认值**

定义时： 列名 数据类型 **default** ''

stu\_sex enum('male','female','mars') default 'male'

插入值：

insert into student values ('张三丰',**default**);

**主键**

Primary key

指定方式：

1.定义列时指定主键信息：id int **primary key**

2.定义完列信息后指定

Primary key(列名）

**主键特征：**

1. **Not null**
2. **唯一索引**

一个表里面只能存在一个主键

主键可以是一列，也可以是多列组成

作用：用来为唯一标识（区分）表实体数据

通用设计：

在每一张都应该存在一个主键，利用一个与业务实体无关的属性，作为唯一标识

主键值不参与业务逻辑（该值做到最少修改原则）

**自增长**

Auto\_increment

为每条记录提供一个唯一的标识，每次插入记录，将设置为自增长的列字段值+1

需要整型，同时需要一个索引

插入值得时候，可以不指定该列，还可以插入null，还可以使用default

可以使用auto\_increment = 开始值指定自增顺序

**唯一**

**Unique**

1.该列值必须唯一，比如学号

**非空**

**Not null**

注释

列注释，解释该列属性

**设计范式（重点，多看看设计思路）**

2017年8月3日

15:02

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 姓名 | 课程名称 | 课程学分 |
| 9527 | 华安 | 数据库原理 | 5 |
| 9528 | 张三丰 | 数据库原理 | 5 |
| 9530 | 秦始皇 | 操作系统 | 6 |

**1NF**

属性不可再分，字段保证原子性

问题：理解同2NF

**2NF**

在满足第一范式的基础上（每一个字段都和主键字段相关）

实现：增加一个用于标识实体对象的列字段（主键）

出现的问题：

1.数据冗余

如多个学生同时选修一门课程，一个学生选择多门课程，会出现大量的冗余数据

2.更新异常

如果需要调整某一门课程的学分信息，会更新大量同课程学分信息

3.插入异常

如新增一门课程还没有学生进行选修（学号为空），插入时违反了学号非空约束，不能新增课程

4.删除异常

如果一批学生完成该课程的选修，需要将该课程的选修信息删除，那么课程信息也会被删除

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | 学号 | 姓名 | 课程名称 | 课程学分 |
| 1 | 9527 | 华安 | 数据库原理 | 5 |
| 2 | 9528 | 张三丰 | 数据库原理 | 5 |
| 3 | 9530 | 秦始皇 | 操作系统 | 6 |

**3NF**

在满足第二范式的基础上，不能出现传递依赖（该表字段不能出现其他表非主键字段）

实现：将出现传递依赖的字段信息单独抽取成新表，用一个外键字段指向新表的主键

|  |  |  |  |
| --- | --- | --- | --- |
| Id | 学号 | 姓名 | 课程id |
| 1 | 9527 | 华安 | 1 |
| 2 | 9528 | 张三丰 | 1 |
| 3 | 9530 | 秦始皇 | 2 |

|  |  |  |
| --- | --- | --- |
| Id | 课程名称 | 学分 |
| 1 | 操作系统 | 6 |
| 2 | 数据库原理 | 4 |
| 3 | 高等数学 | 12 |

实体之间的关系（重点）

2017年8月3日

15:00

**1:1一对一**

两个实体表内，存在相同的主键字段

如果一个记录的主键值等于另一个关系表内的主键值，他们的关系1:1

常用设计：

从表主键引用主表的主键（主外键约束）：从表主键依赖于主表主键

从表新增外键列引用主表的主键，同时指定该字段为unique的（特殊的1：M）

常用业务：

1.表的垂直分割

2.person-idcard

**1:M一对多**

一个实体，对应其他多个实体

例如：一个班级对应多个学生

一个部门对应多个员工

设计：在多的一端（学生，员工）增加一个外键字段，用于指向所属的另外一个

实体标识（一的一方的主键）

**M:N多对多**

设计：利用一个中间表，表示实体间的对应关系

中间表的每个记录代表一个关系

|  |  |  |
| --- | --- | --- |
| stu\_id | stu\_name | Stu\_age |
| 1 | 张三 | 18 |
| 2 | 张三丰 | 20 |
| 3 | 张四丰 | 30 |

|  |  |  |
| --- | --- | --- |
| Id | Stu\_id | Sourse\_id |
| 1 | 1 | 1 |
| 2 | 1 | 2 |
| 3 | 2 | 1 |
| 4 | 2 | 2 |

|  |  |  |
| --- | --- | --- |
| Course\_id | Course\_name | Scource\_credit |
| 1 | javaSE | 6 |
| 2 | Mysql | 5 |
| 3 | Oracle | 10 |

**外键 foreign key**

**概念：**

如果一个实体（employees）的某个字段（employees:dep\_id)引用另一个实体(department)的主键（department:dep\_id),这个字段（employees:dep\_id)我们就称为外键

**被指向的实体，称之为主实体（主表），也叫父实体**

**负责指向的实体，称之为从实体（从表）**

**作用**

用于约束处于关系内的实体

1：增加子表记录时，是否有与之对应的父表记录

2：删除/更新主表记录时，从表应该如何处理相关数据

**定义**

**Constraint 外键名 foreign key(外键字段） references 主表（主表的主键字段）**

**级联操作**

当主表对应的数据发生改变时，与之对应的从表记录如何处理

主表更新

ON UPDATE

主表删除

ON DELETE

允许的级联操作

Cascade ：

关联操作，如果主表关联的记录被更新或者删除，与之对应的从表记录执行相同的操作

Set null：设置为空，表示从表不指向任何的主表记录（前提是外键字段不能设置为not null）

restrict：拒绝主表的相关操作（严格限制）

NO ACTION

**事务（重点）**

2017年8月3日

15:03

**概念**

一组sql语句操作单元，一个事务作为一个整体，要么成功，要么失败，如果某一事务完成，则在该事务中所有数据更改均会提交，称为数据库中永久组成部分，如果事务遇到错误且必须取消或回滚，则所有数据更改均被清除

**事务特性**

**原子性**

事务是一个完整的操作，事务的各个元素不可分（原子的）事务中所有元素必须作为一个整体提交或回滚

**一致性**

当事务完成时，数据必须处于一致状态

**隔离性**

事务之间是独立的，不应以任何方式依赖或影响其他事务

**持久性**

事务完成后，对系统的影响是永久的

|  |  |
| --- | --- |
| 1.设置自动提交事务关闭，改为手动提交 | Set autocommit=0 |
| 2.显式的声明事务 | **Start transaction** |
|  | begin |

mysql中默认自动提交事务的，因此需要关闭自动提交功能

存在一个系统变量：autocommit=1

关闭自动提交：set autocommit=0

事务提交：commit

事务回滚：rollback

该方式问题：如果事务结束后，当前的系统为事务手动提交，以后新的操作必须手动提交事务

常见事务指令

BEGIN或START TRANSACTION；显示地开启一个事务

关闭自动提交，事务结束后，回到start时的状态

**事务的保存点**

SAVEPOINT identifier

SAVEPOINT允许在事务中创建一个保存点，一个事务中可以有多个SAVEPOINT

RELEASE SAVEPOINT identifier

删除一个事务的保存点，当没有指定的保存点时，执行该语句会抛出一个异常

ROLLBACK TO identifier

把事务回滚到标记点

结合jdbc

隔壁老王

重庆

成都

中途在绵阳见网友

截图

**事务的隔离级别**

SET TRANSACTION ISOLATION LEVEL

用来设置事务的隔离级别。InnoDB存储引擎提供事务的隔离级别有

|  |  |  |  |
| --- | --- | --- | --- |
| 隔离级别 | 脏读 | 不可重复读 | 幻读 |
| READ UNCOMMITTED | 可能出现 | 可能出现 | 可能出现 |
| READ COMMITTED | 不会出现 | 可能出现 | 可能出现 |
| REPEATABLE READ | 不会出现 | 不会出现 | 可能出现 |
| SERIALIZABLE | 不会出现 | 不会出现 | 不会出现 |

**默认隔离级别**

**Select @@tx\_isolation;//全局变量**

结合jdbc

两个事务同时操作一个数据，相当于并发一样的，相当于两个线程同时去操作我们的账户信息，这个就和我们的隔离级别有关系

一般四种，和具体的数据库相关

最低级别：有可能别人还没有提交我有可能就看到了，有可能读到错误的数据（或者两次读到的数据部一样==不可重复读）（第一次查出5条，第二次多了==幻读）

隔离级别越高肯定数据正确性的保证越好，但是这是有代价的，需要消耗系统性能的，会影响系统并发效率，所以也不要设置的太高

**select（重点中的重点）**

2017年8月3日

15:04

**Where**

1.查询20号部门的所有员工信息。

select \* from emp where depno = 20;

2.查询所有工种为CLERK的员工的工号、员工名和部门名。

select empno,ename,depno from emp where job like 'CLERK';

3.查询奖金COMM高于工资SAL的员工信息。

select \* from emp where comm > sal;

4.查询奖金高于工资的20%的员工信息。

select \* from emp where comm > (sal\*0.2);

5.查询10号部门中工种为MANAGER和20号部门中工种为CLERK的员工的信息。

select \* from emp

where (depno = 10 and job like 'MANAGER') or (depno = 20 and job like 'CLERK');

6.查询所有工种不是MANAGER和CLERK，且工资大于或等于2000的员工的详细信息。

select \* from emp where job not in ('MANAGER','CLERK') and sal >= 2000

8.查询所有员工工资和奖金的和。

select ename,sal+comm as salcomm from emp;

9.查询没有奖金或奖金低于100的员工信息。

select \* from emp where (comm is null or comm < 100)

**各种查询**

查询工资高于本部门平均工资的员工的信息。

select emp.\* from emp join (select depno,avg(sal) avg\_sal from emp group by depno) t on emp.depno=t.depno and sal>avg\_sal

查询工资高于本部门平均工资的员工的信息及其部门的平均工资。

select emp.\*,avg\_sal from emp join

(select depno,avg(sal) avg\_sal from emp group by depno) t on emp.depno=t.depno and sal>avg\_sal

查询工资、奖金与10号部门某个员工工资、奖金都相同的员工的信息。

select emp.\* from emp join (select sal,comm from emp where depno = 10) t on emp.sal=t.sal and nvl(emp.comm,0)=nvl(t.comm,0) and emp.depno != 10;

查询部门人数大于5的部门的员工的信息。

select \* from emp where depno in

(select depno from emp group by depno having count(\*)>5);

查询所有员工工资都大于1000的部门的信息。

select \* from dept where depno in (select distinct depno from emp where depno not in (select distinct depno from emp where sal < 1000))

查询所有员工工资都大于1000的部门的信息及其员工信息。

select \* from emp join dept on dept.depno in

(select distinct depno from emp where depno not in

(select distinct depno from emp where sal < 1000)) and dept.depno=emp.depno;

查询所有员工工资都在 900~3000 之间的部门的信息。

select \* from dept where depno in

(select distinct depno from emp where depno not in

(select distinct depno from emp where sal not between 900 and 3000))

**模糊查询like**

**%：匹配任意数量的任意字符**

**\_：匹配一个数量的任意字符**

查询员工名正好为6个字符的员工的信息。

select \* from emp where ename like '\_\_\_\_\_\_';

查询员工名字中不包含字母“S”员工。

select \* from emp where ename not in (select ename from emp where ename like '%S%')

select \* from emp where ename not like %S%

查询员工姓名的第2个字母为“M”的员工信息。

select \* from emp where ename like '\_M%';

**Distinct**

去除重复记录

重复条件:distinct后跟的所有字段都一致，认为重复记录

查询有奖金的员工的不同工种。

select distinct job from emp where comm is not null;

查询出所有的工作信息

select distinct depno from emp;

**分页limit**

Limit offset,size

offset：偏移量，数据起始位置，索引从0开始。[可以省略，即代表为0]

Size:取size条数据

类似于javascript的substr(offset,size)

只能出现在所有的条件，分组，排序确定后

**排序 Order by**

按字段名称排序，可以指定多个字段，排序规则为先按照前一个字段进行排序，如果不能确定再按照第二个字段进行排序，依次类推

Order by 字段 [asc|desc], 字段 [asc|desc]…...

Asc:升序排列[默认]

desc：降序排列

查询员工的姓名和入职日期，并按入职日期从先到后进行排列。

select ename,hiredate from emp order by hiredate asc

显示所有的姓名、工种、工资和奖金，按工种降序排列，若工种相同则按工资升序排列。

select ename,job,sal,comm from emp order by job desc,sal asc

**聚合函数**

聚合函数（统计函数）是对一组值执行计算并返回单一的值的函数，它经常与SELECT语句的GROUP BY子句一同使用

AVG(列名) 返回指定组中的平均值，空值被忽略

COUNT(列名) 返回指定组中项目的数量，会过滤掉NULL值

COUNT(1)

COUNT(\*)

 COUNT(\*) 跟COUNT(1) 的结果一样，都包括对NULL的统计，而count(column) 是不包括NULL的统计,

假如表沒有主键(Primary key), 那么count(1)比count(\*)快， 如果有主键的話，那主键作为count的条件时候count(主键)最快,如果你的表只有一个字段的话那count(\*)就是最快的

MAX(列名) 返回指定数据的最大值

MIN(列名) 返回指定数据的最小值

SUM(列名) 返回指定数据的和，只能用于数字列，空值被忽略。

查询各种工作的最低工资。

select job,min(sal) from emp group by job

查询各个部门中的不同工种的最高工资。

select max(sal),job,depno from emp group by depno,job order by depno,job

查询10号部门员工以及领导的信息。

select \* from emp where empno in (select mgr from emp where depno=10) or depno = 10

查询各个部门的人数及平均工资。

select depno,count(\*),avg(sal) from emp group by depno

查询工资为某个部门平均工资的员工信息。

select \* from emp where sal in (select avg(sal) avg\_sal from emp group by depno)

查询工资高于 20号部门某个员工工资的员工的信息。

select \* from emp where sal > any(select sal from emp where depno=20);

统计各个工种的人数与平均工资。

select job,count(\*),avg(sal) from emp group by job

统计每个部门中各个工种的人数与平均工资。

select depno,job,count(\*),avg(sal) from emp group by depno,job order by depno,job;

**Group by**

根据一个或多个列对结果集进行分组,然后针对若干个小组进行数据处理（一般配合聚合函数进行统计工作）。

**Having**

聚合函数不能作为where条件

分组后的条件过滤

找出平均工资大于2000的部门

Select avg(sal),depno from emp

Group by depno

Having avg(sal) > 2000

**Group By 和Having, Where ,Order by语句的执行顺序**

首先WHERE

将最原始记录中不满足条件的记录删除(所以应该在where语句中尽量的将不符合条件的记录筛选掉，这样可以减少分组的次数)，然后通过

Group By

关键字后面指定的分组条件将筛选得到的视图进行分组，接着系统根据

Having

关键字后面指定的筛选条件，将分组视图后不满足条件的记录筛选掉，然后按照

Order By

语句对视图进行排序，这样最终的结果就产生了

**子查询**

子查询就是指的在一个完整的查询语句之中，嵌套若干个不同功能的小查询，从而一起完成复杂查询

**在WHERE子句中使用子查询**

**单行单列子查询**

查询比张无忌工资高的员工

SELECT \*

FROM emp

WHERE sal > (SELECT sal FROM emp WHERE ename='张无忌')

**多行单列子查询**

**IN操作**

查询出与每个部门中最低工资相同的全部雇员信息

SELECT \*  
FROM emp e  
WHERE e.sal IN(  
 SELECT MIN(sal)  
 FROM emp  
 GROUP BY depno  
);

**单行多列子查询**

查找出与张无忌从事同一工作，并且工资高于鹿杖客的全部雇员信息,

SELECT \*  
FROM emp e  
WHERE e.job=(  
 SELECT job  
 FROM emp   
 WHERE ename='张无忌')   
 AND e.sal>(  
 SELECT sal  
 FROM emp   
 WHERE ename='鹿杖客');

查询至少有一个员工的部门信息。

select \* from dept where depno in (select distinct depno from emp where mgr is not null)

查询工资比SMITH员工工资高的所有员工信息。

select \* from emp where sal > (select sal from emp where ename like 'SMITH')

查询所有员工的姓名及其直接上级的姓名。

select staname,ename supname from

(select ename staname,mgr from emp) t join emp on t.mgr=emp.empno

查询入职日期早于其直接上级领导的所有员工信息。

select \* from emp where empno in

(select staempno from (select empno staempno,hiredate stahiredate,mgr from emp) t join emp

on t.mgr=emp.empno and stahiredate < hiredate)

查询最低工资大于2500的各种工作。

select job from (select min(sal) min\_sal,job from emp group by job) where min\_sal > 2500

查询最低工资低于2000的部门及其员工信息。

select \* from emp where depno in

(select depno from (select min(sal) min\_sal,depno from emp group by depno) where min\_sal < '2000')

查询在SALES部门工作的员工的姓名信息。

select ename from emp where depno = (select depno from dept where dname like 'SALES');

查询工资高于公司平均工资的所有员工信息。

select \* from emp where sal > (select avg(sal) from emp)

查询与SMITH员工从事相同工作的所有员工信息。

select \* from emp where job in

(select job from emp where ename like 'SMITH') and ename not like 'SMITH'

列出工资等于30号部门中某个员工工资的所有员工的姓名和工资。

select ename,sal from emp where sal =any (select sal from emp where depno = 30)

查询工资高于30号部门中工作的所有员工的工资的员工姓名和工资。

select ename,sal from emp where sal >all (select sal from emp where depno = 30)

**多表查询-连接查询（较重点）**

2017年8月3日

15:05

**概念及使用场景**

每个实体，一张表

一个业务逻辑，使用多个实体的数据

多张表应该一起使用，将多张表的数据连接起来

**连接的分类**

根据连接的条件不同，可以分为

**内连接**

**外连接**

**自然连接**

**内连接**

数据内部的连接

要求

连接的多个数据都必须存在

**[Inner] join**

如果内连接不指定条件，就是一个交叉连接，可以指定为cross join

共存在M \* N条记录（笛卡尔积）

此时可以指定连接条件

**Where**

**On**

**Using**(字段名)：作为连接条件的两表的字段名必须相同

建议使用ON来连接条件

如果发生数据连接不上，数据丢掉

**外连接**

如果负责连接的数据不真实存在，可以采用外连接

左外连接

Emp **Left join** dep on

出现左表的数据连接不上右表的数据，左表的数据保留，右表放弃

右外连接

Emp **right join** dep on

出现左表的数据连接不上右表的数据，右表的数据保留，左表放弃

全外连接

mysql暂不支持

查询所有部门及其员工信息，包括那些没有员工的部门。

select \* from dept left join emp on emp.deptno=dept.deptno order by dept.deptno

**外连接不能使用where作为连接条件**

**自然连接**

通过Mysql自己的判断完成连接过程，所以不需要指定连接条件（连接数据同内外连接）

自然内连接：natural join 等效于内连接的using

自然外连接

左：natural left join

右：natural right join

查询所有工种为CLERK的员工的姓名及其部门名称。

select ename,dname from emp join dep on job like 'CLERK' and emp.depno=dep.deptno

查询从事同一种工作但不属于同一部门的员工信息。

select distinct t1.empno,t1.ename,t1.depno from emp t1

join emp t2 on t1.job like t2.job and t1.depno <> t2.depno

查询各个部门的详细信息以及部门人数、部门平均工资。

Select dep.\*,person\_num,avg\_sal from dep,

(select count(\*) person\_num,avg(sal) avg\_sal,depno from emp group by depno) t

where dep.depno = t.depno

视图

2017年8月3日

15:08

触发器

2017年8月3日

15:08

存储过程

2017年8月3日

15:08

函数

2017年8月3日

15:09

自定义函数

2017年8月3日

15:09

索引

2017年8月3日

15:09