CDA Mysql基础讲义

讲师：刘志恒Carry.law

刘志恒

carry

计量经济学硕士

中金

人工智能算法工程师

研究领域: 自然语言处理

人工智能四大领域: 语音识别, 自然语言处理, 图像处理, 专家系统等

人工智能不仅是机器人, 还包括某些app中的一些模块.

工具: python做自然语言处理

其他工作: sas, spark, mysql, Hbase===>hadoop

学python和其它语言最好的方法就是去做项目

专注于某一块领域中, 不要专注于很多领域而每个领域都浅尝辄止

架构师mysql理解的更深刻

数据分析师的职业路径

初级数据分析师

使用sql, python, sas等工具进行数据清洗, 描述性统计, 数据挖掘(关联分析,回归分析), 形成报表给其他部门或上级使用

中级数据分析师

把初级数据分析师的工作系统化, 形成一套工具, 供其它业务或其他人使用, 即泛化的能力.

高级数据分析师

一般有自己的研究领域, 如人工智能, 一般重在研究如何把自己的研究成果产品化, 最终达到将产品向市场推广, 实现价值的目的. 或者是希望使用自己的产品增加乙方用户量或订单量的目的

经常有人会问, 自己什么时候才能从EXCEL上升到人工智能的阶段

很多行业中的大牛, 使用Excel表格就能做出来很好的线性拟合模型, 一是对统计挖掘的知识理解的特别深入, 二是有丰富的业务经验

埃森哲的数据报表, 40页的PPT报价70W, 数据分析师分成35W, 所以不要小看报表统计和描述性统计的功能, 最终的目的是给企业产生价值.

# 前言

明确不是database adminstator（DBA）数据库管理员，而是data analyst数据分析师，因此学习的侧重点会有所不同，前者在于数据库的运行和维护，后者在于数据库的使用和查询。

运维（了解）：

数据库建立（创建用户、创建表空间）

数据表管理（约束设置、变量增加与删除）

数据表存储（数据如何写入数据库）

查询（重点）：

数据表查询（单表查询、多表查询、子查询

条件查询、分组查询、模糊查询等）

1. 数据库简介与安装

2. 数据库与表空间

3. 约束与表管理

4. SQL查询语句

5. 案例解析

（预计用时：第1,2,3章0.7-0.9天，第4章1-1.2天，第5章0.1-0.2天）

# 1. 数据库简介与安装

1.1 数据库简介

数据库、数据库管理系统与SQL语言之间的联系？

数据库：相当于一个装有EXCEL表格的文件夹

数据库管理系统：管理这个文件夹的软件

SQL语言：执行这个软件的程序语言

SQL ==输入==>数据库管理系统==调用==>数据库

主流的数据库管理系统：

结构化Mysql、Oracle、SQL Server 关系型数据库

非结构化Hbase 非关系型数据库（例：微信聊天记录）

结构化与非结构化之间的区别？

SQL语言：select \* from table\_name;

（条件查询where，分组group by，分组后筛选having，排序order by）

随堂练习：

存在一张表名为stu\_inf的表

|  |  |  |  |
| --- | --- | --- | --- |
| id | stu\_name | stu\_sex | stu\_score |
| 1 | 张三 | 男 | 90 |
| 2 | 李四 | 男 | 80 |
| 3 | 王五 | 男 | 85 |
| 4 | 赵六 | 女 | 90 |
| 5 | 熊八 | 女 | 95 |

（1） 查询表中所有信息

Select \* from stu\_inf;

（2） 查询表中学生名为张三的信息

Select \* from stu\_infwhere stu\_name=”张三”;

（3） 查询表中成绩大于90的信息

Select \* fromstu\_infwhere stu\_score>90;

1.2数据库安装

（1）Mysql5.7数据库

（2）Workbench数据库可视化界面

（3）Sublime编辑器

在workbench中执行sql语句: ctrl+Enter

选择多条sql语句并执行: ctrl+shift+enter

# 2. 数据库与表空间

## 2.1数据库

-- 创建数据库

create database test1;

-- 查看数据库

show databases;

-- 选择数据库

use test1;

-- 删除数据库

drop database test1;

实例演示

-- 创建表

create table stu\_inf(id,stu\_name,stu\_sex,stu\_score); #错误，数据类型

create table stu\_inf(id int(10),stu\_name varchar(20),stu\_sex varchar(20),stu\_score float(10));

-- 查询表

show tables;

describe stu\_inf;

-- 修改表(修改表名称，添加变量，修改变量，删除变量)

alter table stu\_inf rename a1; #修改表名称

alter table a1 add column class; #错误，数据类型

alter table a1 add column class varchar(20); #添加变量

alter table a1 change column class stu\_class varchar(20); #修改变量

alter table a1 drop column class stu\_class; #删除变量

-- 删除表

drop table a1;

-- 添加索引

alter table stu\_inf add primary key(id); #主键索引

alter table stu\_inf add unique(stu\_name); #唯一索引

-- 查询索引

show index from stu\_inf;

-- 修改索引（先删除后添加）？？？？？

-- 删除索引, 删除索引时不删除变量??

alter table stu\_inf drop primary key; #删除主键索引

alter table stu\_inf drop index stu\_name; #删除其他索引

## 2.2表空间

表空间是什么？

|  |  |  |  |
| --- | --- | --- | --- |
| id | stu\_name | stu\_sex | stu\_score |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

表空间, 数据表中只有结构, 没有内容时, 像Excel中, 只有标题行, 没有数据.

纵向：列= 字段=变量

横向：行=观测

--创建表

create table stu\_inf(id,stu\_name,stu\_sex,stu\_score); #错误，数据类型

数据类型（3大类）：数值型、字符型、日期型

数值型：int 整数型, float 浮点型#精确到第15位，decimal

字符型：char, varchar #char与varchar之间的区别？

日期型：datetime #"YYYY-MM-DD hh:mm:ss"

create table stu\_inf(id int(10),stu\_name varchar(20),stu\_sex varchar(20),stu\_score float(10));

--查询表

show tables; #查看所有表名

describe stu\_inf; #查看单张表结构

--修改表(修改表名称、添加变量、修改变量、删除变量)

alter table stu\_inf rename a1; #修改表名称

修改数据表中的字段名时, 相当于把原来的字段删除了, 又新建一个新的字段, 所以要加上数据类型.

--alter table a1 add column class; #错误，数据类型？

alter table a1 add column class varchar(20); #添加变量

alter table a1 change column class stu\_class varchar(20); #修改变量

alter table a1 drop column class stu\_class; #删除变量

--删除表

drop table a1;

## 索引

包括普通索引、唯一索引、主键索引、组合索引、全文索引

主键索引（一张表里只能有一个），唯一索引（一张表里可能存在多个）

作用：提高后台的查询效率（为什么？）

约束与索引的区别？

约束: 检查数据的完整性 (业务逻辑), 如网站中要求用户名不能重复

索引: 实现数据查询的优化 (程序逻辑),

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | id | stu\_name | stu\_sex | stu\_score |
| >> | 1 | 王三 | 女 | 84 |
| >> | 2 | 李三 | 女 | 61 |
| >> | 3 | 赵六 | 女 | 77 |
| >> | 4 | 李五 | 女 | 64 |
| >> | 5 | 王六 | 男 | 92 |
| >> | 6 | 王四 | 男 | 82 |
| >> | 7 | 李四 | 女 | 76 |
| >> | 8 | 李六 | 男 | 87 |
| >> | 9 | 赵三 | 男 | 88 |
| >> | 10 | 张六 | 女 | 63 |
| >> | 11 | 李五 | 男 | 61 |
| >> | 12 | 张四 | 女 | 78 |
| >> | 13 | 王三 | 女 | 93 |
| >> | 14 | 王六 | 女 | 81 |
| >> | 15 | 王三 | 女 | 78 |

|  |  |
| --- | --- |
|  | stu\_name |
| >> | 王三 |
| >> | 李三 |
| >> | 赵六 |
| >> | 李五 |
| >> | 王六 |
| >> | 王四 |
| >> | 李四 |
| >> | 李六 |
| >> | 赵三 |
| >> | 张六 |
| >> | 李五 |
| >> | 张四 |
| >> | 王三 |
| >> | 王六 |
| >> | 王三 |

--添加索引

alter table stu\_inf add primary key(id);

alter table stu\_inf add unique(stu\_name);

--查询索引

show index from stu\_inf;

--修改索引(先删除后添加)

--删除索引

alter table stu\_inf drop primary key; #删除主键索引

alter table stu\_inf drop index stu\_name; #删除其他索引

主键索引, 一张表里只能对一个字段创建主键索引

唯一索引, 一张表里可以对多个字段创建唯一索引, 唯一索引的作用是保证这个字段里值的唯一性, 也就是这个字段里的值不能重复

-- 主键约束, 建立主键索引时会自动建立主键约束

create table s\_user(

u\_id int auto\_increment primary key,

u\_name varchar(20),

u\_pwd varchar(20)

);

-- 外键约束

-- 方法1, 创建表时添加外键约束, 要首先保证s\_user数据表存在

create table s\_order(

o\_id int auto\_increment primary key,

o\_buyer\_id int,

o\_totalprices float,

foreign key(o\_buyer\_id) references s\_user(u\_id)

);

-- 方法2, 修改表时添加外键约束

create table s\_order2(

o\_id int auto\_increment primary key,

o\_buyer\_id int,

o\_totalprices float

);

alter table s\_order2 add foreign key(o\_buyer\_id) references s\_user(u\_id);

-- 添加数据

insert into s\_user(u\_name,u\_pwd) values("carry","123");

insert into s\_user(u\_name,u\_pwd) values("carry","123"),("harry","456"),("marry","789");

-- 修改数据

# 设置数据库安全等级,

set sql\_safe\_updates=0; #高级 1

update s\_user set u\_pwd="963" where u\_name="harry";

???

# 数据库的安全等级不允许对使用主键之处的查询来修改表中的数据. 如果是进行主键查询, 则可以修改, 还是如果查询到的结果是多条记录的话就不允许进行修改?如.

update s\_user set u\_pwd="998" where id="1"

-- 删除数据

delete from s\_user where u\_name="marry";

/\*

## 单元测试：

创建一个数据库test\_stu

进入数据库test\_stu

在数据库test\_stu中建立四张表：

Student学生表（id, S\_name, S\_age, S\_sex, S\_number）

Score成绩表（S\_number, C\_name, score）

Teacher教师表（S\_number, C\_name, T\_name）

其中id为主键索引，S\_number为唯一索引

在工作中学会把需求进行分解, 分解为一块块的可以实现的小的需求

\*/

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| s\_name | s\_sex | s\_number | c\_name | score | teacher |
| 张三 | 男 | 141211 | 语文 | 90 | 李磊 |
| 张三 | 男 | 141211 | 数学 | 80 | 孙中 |
| 张三 | 男 | 141211 | 外语 | 90 | 信一 |
| 李四 | 男 | 141212 | 语文 | 75 | 李磊 |
| 李四 | 男 | 141212 | 数学 | 90 | 孙中 |
| 李四 | 男 | 141212 | 外语 | 75 | 阿哈 |
| 王五 | 女 | 141213 | 语文 | 85 | 李磊 |
| 王五 | 女 | 141213 | 数学 | 80 | 赵往 |
| 王五 | 女 | 141213 | 外语 | 80 | 信一 |

|  |  |  |  |
| --- | --- | --- | --- |
| student |  |  |  |
| id | s\_name | s\_sex | s\_number |
| 1 | 张三 | 男 | 141211 |
| 2 | 李四 | 男 | 141212 |
| 3 | 王五 | 女 | 141213 |

|  |  |  |
| --- | --- | --- |
| score |  |  |
| s\_number | c\_name | score |
| 141211 | 语文 | 90 |
| 141211 | 数学 | 80 |
| 141211 | 外语 | 90 |
| 141212 | 语文 | 75 |
| 141212 | 数学 | 90 |
| 141212 | 外语 | 75 |
| 141213 | 语文 | 85 |
| 141213 | 数学 | 80 |
| 141213 | 外语 | 80 |

|  |  |  |
| --- | --- | --- |
| teacher |  |  |
| s\_number | c\_name | teacher |
| 141211 | 语文 | 李磊 |
| 141211 | 数学 | 孙中 |
| 141211 | 外语 | 信一 |
| 141212 | 语文 | 李磊 |
| 141212 | 数学 | 孙中 |
| 141212 | 外语 | 阿哈 |
| 141213 | 语文 | 李磊 |
| 141213 | 数学 | 赵往 |
| 141213 | 外语 | 信一 |

-- 1.创建数据库

create database test\_stu;

-- 2.进入数据库

use test\_stu;

-- 3.建立student表

create table student(id int,s\_name varchar(20),s\_sex varchar(20),s\_number int);

-- 4.建立score表

create table score(s\_number varchar(20),c\_name varchar(20),score float);

-- 5.建立teacher表

create table teacher(s\_number varchar(20),c\_name varchar(20),t\_name varchar(20));

-- 6.添加主键索引

alter table student add primary key(id);

-- 7.添加唯一索引

alter table student add unique(s\_number);

alter table teacher add unique(s\_number);

alter table score add unique(s\_number);

## 映射关系图：E-R图

=INDIRECT("F"&RANDBETWEEN(2,5))&INDIRECT("G"&RANDBETWEEN(2,5))

专家图谱, 给机器一些知识图谱, 让它们有能理解现在文本的能力

知识图谱,

映射关系图, E-R图

数据字典, 其中包含一张关系, 每张表中是如何映射的.

原始的信息如下

s\_name s\_sex s\_number c\_name score teacher

张三 男 141211 语文 90 李磊

张三 男 141211 数学 80 孙中

张三 男 141211 外语 90 信一

李四 男 141212 语文 75 李磊

李四 男 141212 数学 90 孙中

李四 男 141212 外语 75 阿哈

王五 女 141213 语文 85 李磊

王五 女 141213 数学 80 赵往

王五 女 141213 外语 80 信一

如果直接把这张表保存在数据库中, 会造成数据库的冗余.

DBA在存储数据时, 会把它分解为3张数据表

student

id s\_name s\_sex s\_number

1 张三 男 141211

2 李四 男 141212

3 王五 女 141213

score

s\_number c\_name score

141211 语文 90

141211 数学 80

141211 外语 90

141212 语文 75

141212 数学 90

141212 外语 75

141213 语文 85

141213 数学 80

141213 外语 80

teacher

s\_number c\_name teacher

141211 语文 李磊

141211 数学 孙中

141211 外语 信一

141212 语文 李磊

141212 数学 孙中

141212 外语 阿哈

141213 语文 李磊

141213 数学 赵往

141213 外语 信一

而3张数据表以学生表中的s\_number这个字段连接到一起, 还可以分别把语文, 数学和外语的成绩分别放在三张数据表中.

student 1:n score

student 1:n teacher

score 1:1 teacher

score表和teacher表中因为有s\_number和c\_name这两个字段, 故是1:1的关系.

# 3. 约束与表管理

## 3.1约束

作用:

维护数据库完整性

1.防止将错误值插入数据表；2.保持表中数据的一致性

约束的类型

1. 防止将错误数据插入数据库;

非空约束, 唯一约束, 检查约束

2. 保持表中数据的一致性;

主键约束, 外键约束

SELECT keyboard\_name FROM keyboard ORDER BY price DESC limit 1;

SELECT keyboard\_name, payment\_link FROM keyboard ORDER BY price DESC limit 1;

|  |  |  |
| --- | --- | --- |
| 非空约束 | 唯一约束 | 检查约束 |
| 3000 | 张三 | 男 |
| 4000 | 张三 | 男 |
| 6000 | 李四 | 女 |
| null | 王五 | 女 |
| 5000 | 赵六 | 公 |

注意：与索引的区别（主键约束自动建唯一索引）

--主键约束

create table s\_user(

u\_id int primary key,

u\_name varchar(20),

u\_pwd varchar(20)

);

# 添加自增列

create table s\_user(

u\_id int auto\_increment primary key,

u\_name varchar(20),

u\_pwd varchar(20)

);

--外键约束

--方法1

create table s\_order(

o\_id int auto\_increment primary key,

o\_buyer\_id int,

o\_totalprices float,

foreign key(o\_buyer\_id) references s\_user(u\_id)

);

--方法2

create table s\_order2(

o\_id int auto\_increment primary key,

o\_buyer\_id int,

o\_totalprices float

);

alter table s\_order2 add foreign key(o\_buyer\_id) references s\_user(u\_id);

## 3.2表管理（数据管理）

--添加数据

insert into s\_user(u\_id,u\_name,u\_pwd) values(1,"carry","123");

#有自增列

insert into s\_user(u\_name,u\_pwd) values("carry","123");

insert into s\_user(u\_name,u\_pwd) values("carry","123"),("harry","456"),("marry","789");

--修改数据

# 设置数据库安全等级

set sql\_safe\_updates=0; #高级1

update s\_user set u\_pwd="963" where u\_name="harry";

--删除数据

delete from s\_user where u\_name="marry";

创建表emp员

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **员工号** | **员工姓名** | **职位** | **直属领导** | **雇佣时间** | **薪水** | **津贴** | **部门号** |
| empno | ename | job | mgr | hiredate | sal | comm | deptno |
| 7369 | smith | clerk | 7902 | 1980-12-17 00:00:00 | 800 | NULL | 20 |
| 7499 | allen | salesman | 7698 | 1981-02-20 00:00:00 | 1600 | 300 | 30 |
| 7521 | ward | salesman | 7698 | 1981-02-22 00:00:00 | 1250 | 500 | 30 |
| 7566 | jones | manager | 7839 | 1981-04-02 00:00:00 | 2975 | NULL | 20 |
| 7654 | martin | salesman | 7698 | 1981-09-28 00:00:00 | 1250 | 1400 | 30 |
| 7698 | blake | manager | 7839 | 1981-05-01 00:00:00 | 2850 | NULL | 30 |
| 7782 | clark | manager | 7839 | 1981-06-09 00:00:00 | 2450 | NULL | 10 |
| 7788 | scott | analyst | 7566 | 1987-04-19 00:00:00 | 3000 | NULL | 20 |
| 7839 | king | persident | NULL | 1981-11-17 00:00:00 | 5000 | NULL | 10 |
| 7844 | turner | salesman | 7698 | 1981-09-08 00:00:00 | 1500 | 0 | 30 |
| 7876 | adams | clerk | 7788 | 1987-05-23 00:00:00 | 1100 | NULL | 20 |
| 7900 | james | clerk | 7698 | 1981-12-03 00:00:00 | 950 | NULL | 30 |
| 7902 | ford | analyst | 7566 | 1981-12-03 00:00:00 | 3000 | NULL | 20 |
| 7934 | miller | clerk | 7782 | 1982-01-23 00:00:00 | 1300 | NULL | 10 |

把表单元测试3中的数据写入到test\_stu这个数据库的emp数据表中.

-- 1.创建表空间

create table emp (

empno int,

ename varchar(20),

job varchar(20),

leader int,

hiredate datetime,

sal float,

comm float,

deptno int

);

-- 2.添加表数据

insert into emp values(7369,"smith","clerk",7902,"1980-12-17",800,null,20),

(7499,"allen","salesman",7698,"1981-02-20",1600,300,30),

(7521,"ward","salesman",7698,"1981-02-22",1250,500,30),

(7566,"jones","manager",7839,"1981-04-02",2975,null,20),

(7654,"martin","salesman",7698,"1981-09-28",1250,1400,30),

(7698,"blake","manager",7839,"1981-05-01",2850,null,30),

(7782,"clark","manager",7839,"1981-06-09",2450,null,10),

(7788,"scott","analyst",7566,"1987-04-19",3000,null,20),

(7839,"king","persident",null,"1981-11-17",5000,null,10),

(7844,"turner","salesman",7698,"1981-09-08",1500,0,30),

(7876,"adams","clerk",7788,"1987-05-23",1100,null,20),

(7900,"james","clerk",7698,"1981-12-03",950,null,30),

(7902,"ford","analyst",7566,"1981-12-03",3000,null,20),

(7934,"miller","clerk",7782,"1982-01-23",1300,null,10);

# 4. SQL查询语句

## 4.1单表查询

select \* from table\_name;

where 条件查询

group by 分组查询

having 分组后查询

order by 结果排序

-- 查询emp表中所有的行和列

select \* from emp;

-- 为什么不推荐用\*？（涉及到变量的修改，添加与删除, DBA不会通知我们, 使用\*号是查询不出来的）

-- 从表中检索部分的变量

select empno,ename,job,leader,hiredate,sal,comm,deptno from emp;

-- 数据分析师还要与业务部分打交道, 要把查询的结果展示给他们. 这里的中文不需要加引号, 并且可以省略as

select empno as 员工编号,ename as 员工姓名,job as 职位,leader as 直属领导,hiredate as 雇用时间,

sal as 薪水,comm as 津贴,deptno as 部门编号 from emp;

select empno 员工编号,ename 员工姓名,job 职位,leader 直属领导,hiredate 雇用时间,

sal 薪水,comm 津贴,deptno 部门编号 from emp;

-- 查询去重, 查询部门编号

select distinct deptno from emp;

-- 条件查询, 也就是对行进行过滤

-- 从表中检索部分观测（字段的大小写不敏感）

select \* from emp where job='Salesman';

-- 从表中查找空值, 只对空值查询时才使用is, 因为null不属于任何的数据类型, 不能使用算术或逻辑运算符来计算或比较

select \* from emp where comm is null;

select \* from emp where comm is not null;

-- 将空值转化为实际值0, 不是在数据库中的操作, 而是在查询时把空值转换为0. 在使用这个操作时, 会在结果的临时表中建立一个coalesce(comm,0)的字段, 而不是在原comm字段上进行修改

select coalesce(comm,0) from emp;

-- 把查询的结果放到一张数据表中

create table a1 as

select comm,coalesce(comm,"张三") from emp;

desc a1;

-- 查询时可以把空值转换为字符, 但是这样就无法进行进一步的计算了

-- 多条件查询

select \* from emp where deptno=10 or deptno=20;

select \* from emp where deptno=10 or (deptno=20 and sal<=2000);

-- 模糊查询

-- %代表任意多个字符

-- \_代表任意一个字符

-- 查询员工姓名以'a'开头的

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

-- 查询员工姓名中包含'a'的

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

-- 查询员工姓名中第2个字符为'a'的

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

-- 分组查询(分组的目的是为了汇总计算)

-- 按部门分组并计算每个部门的薪资总和, 在结果中添加一列sum(sal)

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

-- 也可以使用\*来输出所有的字段, 但是这样做意义不大, 一般在结果中只输出进行分组的字段和汇总的字段即可.

select \*,sum(sal) from emp group by deptno;

-- 进行汇总的字段中存在空值, 可以先替换为0

select deptno,sum(comm) from emp group by deptno; #空值不计算

select deptno,sum(coalesce(comm,0)) from emp group by deptno; #空值变为0

-- 对每个部门中薪资大于2000的员工进行薪资的汇总, 先把表中所有薪资大于2000的员工提取出来, 再对这些员工进行分组汇总

select deptno,sum(sal) from emp where sal>2000 group by deptno;

select deptno,job,sum(sal) from emp group by deptno,job;

-- 分组后筛选, having必须要与group by配合使用（两句话的区别）

-- 表示把分组汇总的结果中职位为manager的选择了出来

select deptno,job,sum(sal) from emp group by deptno,job having job = 'manager';

-- 表示先把职位为manager的选择了出来, 再进行分组汇总

select deptno,job,sum(sal) from emp where job = 'manager' group by deptno,job;

-- 两个语句的结果是一样的, 但where条件查询更快, 因为是先筛选再进行的汇总

-- 为什么还要having, 如果是对汇总的字段进行聚合, 就只能使用having

select deptno,job,sum(sal) from emp group by deptno,job having sum(sal)>5000;

select deptno,job,new\_sal from (select deptno,job,sum(sal) as new\_sal from emp group by deptno,job) as a where new\_sal>5000;

-- 结果排序

select \* from emp order by sal;

select \* from emp order by sal desc; #注意descending和describe的区别

select \* from emp order by sal desc limit 5;

select deptno,job,sum(sal) as new\_sal from emp group by deptno,job order by deptno,new\_sal;

# 从第二行开始选取5行

方法一：

select \* from emp order by sal desc limit 1,5;

方法二：#显示带观测序号的列表

select @y:=@y+1 as num,ename,deptno,sal from (select @y:=0) a,emp;

select deptno,job,sum(sal) as new\_sal from emp group by deptno,job order by deptno,new\_sal;

-- 逻辑操作符:

and, or 且, 或

!=, <> 不等于

is null 查询空值

like 模糊查询

between b and c 表示[b,c), 即大于等于b, 小于c, 在不同的数据库中对上下限的处理不同, 所以一般不建议使用这个表达式

-- 逻辑表达式

case when ``` else ``` end

-- 在结果中添加一个字段status, 如果薪资小于等于2000, 显示为"过低", 薪资大于等于4000, 显示"过高", 其它的显示为"可行"

select ename,sal,case when sal<= 2000 then '过低'

when sal>= 4000 then '过高'

else '可行'

end as status

from emp;

-- 在结果中添加一个class字段, 如果工资大于等于1000并且小于2000, 显示为1, 其它的显示为2, 并且以class进行排序

-- 方法1

select empno,ename,case when sal >= 1000 and sal<2000 then 1

else 2

end as class

from emp order by class;

-- 方法2

select empno,ename,case when sal between 1000 and 2000 then 1

else 2

end as class

from emp

order by class;

-- 方法3

select empno,ename,case when sal < 1000 or sal >=2000 then 2

else 1

end as class

from emp order by class;

-- 方法4

select empno,ename,case when sal < 1000 then 2

when sal >=2000 then 2

else 1

end as class

from emp

order by class;

随堂练习

/\*

从emp表中选择

部门编号为20 且 工资小于2000的

部门编号为30 且 职位中包含"sale"的

按照部门编号升序, 薪资降序排列

\*/

select \* from emp

where deptno=20 and sal<2000 or (deptno=30 and job like "%sale%")

order by, deptno,sal desc

/\*

在结果中新建一个status字段

小于1500显示"过低",

大于1500小于4000, 显示"中等",

大于4000, 显示"过高"

输出员工姓名, 薪资和status

按薪资排序, 注意这里不能对新字段status进行排序, 因为是中文的, 排序时会以字母顺序排序

\*/

select ename,sal,case when sal<=1500 then "过低"

when sal<=4000 then "中等"

else "过高"

end as status

from emp

order by sal;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 雇员编号 | 姓名 | 性别 | 职称 | 出生日期 | 所在部门编号 | 部门名称 | 基本工资 | 职务工资 | 扣除 |
| emp\_id | name | gender | title | birthday | dep\_id | dep\_name | base\_salary | title\_salary | deduction |
| 1001 | 张三 | 男 | 高级工程师 | 1985/1/1 | 111 | 生产部 | 12000 | 1100 | 200 |
| 1002 | 李四 | 女 | 助工 | 1991/1/1 | 111 | 生产部 | 7200 | 200 | 100 |
| 1003 | 王五 | 男 | 工程师 | 1988/11/11 | 222 | 销售部 | 10000 | 700 | 200 |
| 1004 | 赵六 | 男 | 工程师 | 1989/8/1 | 222 | 销售部 | 9500 | 700 | 150 |
| 1005 | 张七 | 女 | 助工 | 1992/7/1 | 333 | 人事部 | 6000 | 200 | 100 |
| 雇员编号 | 姓名 | 性别 | 职称 | 出生日期 | 所在部门编号 | 部门名称 | 基本工资 | 职务工资 | 扣除 |
| emp\_id | name | gender | title | birthday | dep\_id | dep\_name | base\_salary | title\_salary | deduction |
| 1001 | 张三 | 男 | 高级工程师 | 1985/1/1 | 111 | 生产部 | 12000 | 1100 | 200 |
| 1002 | 李四 | 女 | 助工 | 1991/1/1 | 111 | 生产部 | 7200 | 200 | 100 |
| 1003 | 王五 | 男 | 工程师 | 1988/11/11 | 222 | 销售部 | 10000 | 700 | 200 |
| 1004 | 赵六 | 男 | 工程师 | 1989/8/1 | 222 | 销售部 | 9500 | 700 | 150 |
| 1005 | 熊七 | 女 | 助工 | 1992/7/1 | 333 | 人事部 | 6000 | 200 | 100 |

-- 创建数据表employee,并添加数据

create table employee(emp\_id int,name varchar(20),gender varchar(20),title varchar(20),

birthday date,dep\_id int,dep\_name varchar(20),base\_salary float,

title\_salary float,deduction float);

insert into employee values(1001,"张三","男","高级工程师","1985-01-01",111,"生产部",12000,1100,200),

(1002,"李四","女","助工","1991-01-01",111,"生产部",7200,200,100),

(1003,"王五","男","工程师","1988-11-11",222,"销售部",10000,700,200),

(1004,"赵六","男","工程师","1989-08-01",222,"销售部",9500,700,150),

(1005,"张七","女","助工","1992-07-01",333,"人事部",6000,200,100);

-- 查询每个雇员的姓名、职称、应发工资、实发工资

select name,title,base\_salary+title\_salary as 应发工资,base\_salary+title\_salary-deduction as 实发工资 from employee;

-- 查询姓"张"且年龄大于25岁的员工记录

select \* from employee where (year(now())-year(birthday))>25 and name like "张%"

select \*,(select year(curdate())-year(employee.birthday)) as 年龄 from emplyee where name like "张%" group by name like "张%" having 年龄>25;

-- 定义了一个子循环, 先从表中查询一个变量出来, 把这个变量命名为"年龄", 但尽量不要使用子查询, 速度会很慢

select \* from employee where round((curdate())-YEAR(birthday))>25 and name like "张%";

select year(now())-year(birthday) 年龄,name 姓名 from emplyee where name like "张%" group by name having 年龄>25;

select year(now())-year(birthday) 年龄,name 姓名 from emplyee where name like "张%" and (year(now())-year(birthday))>25;

select now(); #现在的日期和时间

select curdate(); #现在的日期

-- 如果想要精确的话, 还要取整

-- 统计各类职称的人数

select title,count(\*) as 人数 from employee group by title;

select title,count(emp\_id) as 人数 from emplyee group by title;

-- 统计各部门的部门名称、实发工资和平均工资

select dep\_name, sum(base\_salary+title\_salary-deduction) as 实发工资总和,avg(base\_salary+title\_salary-deduction) as 平均工资 from emplyee group by dep\_name

select \* from emp limit 1,3; #从第二行开始取, 取3行数据, 取前几行, 用来查看数据表中数据的结构. 如果想要查看数据的分布规律, 可以使用后面的数据统计的功能.

（1）查询emp表中部门编号为20和30，其中部门编号为20中只输出薪资小于等于2000的部分，部门编号为30中只输出职位包含字母sale的部分，按部门编号、薪资由大到小排序。

（2）建立新字段sta，当sal小于等于1500的时候，显示“过低”；当sal大于1500小于等于4000时，显示“中等”；当sal大于4000，显示“过高”，按薪资降序排列。

--7.（1）

select \* from emp where (deptno=20 and sal<=2000) or (deptno=30 and job like "%sale%")

order by deptno,sal desc;

--7.（2）

select ename,sal,case when sal<=1500 then '过低'

when sal>1500 and sal<=4000 then '中等'

else '过高'

end as sta

from emp order by sal desc;

## 4.2多表查询（注意：绘图）

表名:a1

|  |  |
| --- | --- |
| x | y |
| 1 | one |
| 2 | two |
| 2 | two |
| 3 | three |

表名：a2

|  |  |
| --- | --- |
| x | z |
| 1 | one |
| 2 | two |
| 4 | four |

drop table a1;

drop table a2;

create table a1(x int,y varchar(20));

create table a2(m int,n varchar(20));

insert into a1(x,y) values(1,"one"),(2,"two"),(2,"two"),(3,"three");

insert into a2(m,n) values(1,"one"),(2,"two"),(4,"four");

-- 纵向连接 连接观测----行 上下连接

-- 要求两个表的字段和字段的类型必须是一致的, 字段数量不能多也不能少, 这也是一般很少用纵向连接的一个原因

-- 纵向内连接, a1表中所有的行, 加上a2表中除去与a1表中重复的记录

select \* from a1

union

select \* from a2;

-- 纵向全连接

select \* from a1

union all

select \* from a2;

--加入条件语句

select x,y from a1 where x=1

union

select m,n from a2 where m=2

order by 1 desc;

-- 从emp中查询各部门工资排名前三的员工, 对各部门工资降序排列, 取其中前3名的员工, 查询部门10中工资排名前3的员工, 加上部门20中工资排名前3的员工, 加上部门30中工资排名前3的员工.

-- union和order by不能同时使用, 如

select \* from emp where deptno=10 order by sal desc limit 3

union all

select \* from emp where deptno=20 order by sal desc limit 3

union all

select \* from emp where deptno=30 order by sal desc limit 3;

-- 可以创建新的工作表, 把原表中的内容复制到新表中

create table m1 as

select \* from emp where deptno=10 order by sal desc limit 3;

create table m2 as

select \* from emp where deptn0=20 order by sal desc limit 3;

create table m3 as

select \* from emp where deptno=30 order by sal desc limit 3;

select \* from m1

union all

select \* from m2

union all

select \* from m3

-- 或者使用下面的语句, 每一个括号中的内容相当于是已经生成了一个新表, 把3个新表进行了连接

(select \* from emp where deptno=10 order by sal desc limit 3)

union all

(select \* from emp where deptno=20 order by sal desc limit 3)

union all

(select \* from emp where deptno=30 order by sal desc limit 3);

-- 横向连接 连接变量----列 左右连接, 因为数据是在不同的工作表中存储的, 要查询多个表中的数据

-- 内连接, 两个表的交集

select \* from a1 inner join a2 on a2.x=a2.m;

select \* from a1 join a2 on a2.x=a2.m; #推荐

select \* from a1,a2 where a1.x=a2.m; #不推荐

-- 左连接, 从左表中逐行取出数据与右表中进行匹配, 如果匹配到, 就把两个表中的数据拼接到一起, 如果匹配不到, 就只取左表中的数据

select \* from a1 left join a2 on a1.x=a2.m;

/\*

数组1 数组2

1 4

2 5

3 6

数组1和数组2的笛卡尔积=[(1,4),(1,5),(1,6),(2,4),(2,5),(2,6),(3,4),(3,5),(3,6)]

而进行横向连接时, 就是把两个表连接成笛卡尔积, 再从其中进行选择, 对于内连接, 选择两表中共同的部分, 对于左连接, 选择共同的数据, 再选择左表中独有的数据.

\*/

-- 右连接, 右表中的所有数据

-- 全连接, 两个表中的所有数据, mysql不支持全连接, 但可以使用左连接+右连接+纵向连接得到

select \* from a1 left join a2 on a2.x=a2.m

union

select \* from a1 right join a2 on a1.x=a2.m;

## 4.3子查询

select \* from emp where deptno=10;

可以在select 的部分中, from的部分中, deptno=的部分中插入子查询

创建测试表players, penalties

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **球员编号** | **球员姓名** | **姓首字母** | **生日** | **性别** | **加入时间** | **街道** | **房号** | **邮政号** | **乡镇** | **电话** | **联盟编号** |
| PLAYERNO | NAME | INITIALS | BIRTH\_DATE | SEX | JOINED | STREET | HOUSENO | POSTCODE | TOWN | PHONENO | LEAGUENO |
| 2 | Everett | R | 1948-09-01 | M | 1975 | Stoney Road | 43 | 3575NH | Stratford | 070-237893 | 2411 |
| 6 | Parmenter | R | 1964-06-25 | M | 1977 | Haseltine Lane | 80 | 1234KK | Stratford | 070-476537 | 8467 |
| 7 | Wise | GWS | 1963-05-11 | M | 1981 | Edgecombe Way | 39 | 9758VB | Stratford | 070-347689 | NULL |
| 8 | Newcastle | B | 1962-07-08 | F | 1980 | Station Road | 4 | 6584WO | Inglewood | 070-458458 | 2983 |
| 27 | Collins | DD | 1964-12-28 | F | 1983 | Long Drive | 804 | 8457DK | Eltham | 079-234857 | 2513 |
| 28 | Collins | C | 1963-06-22 | F | 1983 | Old Main Road | 10 | 1294QK | Midhurst | 010-659599 | NULL |
| 39 | Bishop | D | 1956-10-29 | M | 1980 | Eaton Square | 78 | 9629CD | Stratford | 070-393435 | NULL |
| 44 | Baker | E | 1963-01-09 | M | 1980 | Lewis Street | 23 | 4444LJ | Inglewood | 070-368753 | 1124 |
| 57 | Brown | M | 1971-08-17 | M | 1985 | Edgecombe Way | 16 | 4377CB | Stratford | 070-473458 | 6409 |
| 83 | Hope | PK | 1956-11-11 | M | 1982 | Magdalene Road | 16A | 1812UP | Stratford | 070-353548 | 1608 |
| 95 | Miller | P | 1963-05-14 | M | 1972 | High Street | 33A | 5746OP | Douglas | 070-867564 | NULL |
| 100 | Parmenter | P | 1963-02-28 | M | 1979 | Haseltine Lane | 80 | 6494SG | Stratford | 070-494593 | 6524 |
| 104 | Moorman | D | 1970-05-10 | F | 1984 | Stout Street | 65 | 9437AO | Eltham | 079-987571 | 7060 |
| 112 | Bailey | IP | 1963-10-01 | F | 1984 | Vixen Road | 8 | 6392LK | Plymouth | 010-548745 | 1319 |

|  |  |  |  |
| --- | --- | --- | --- |
| **支付编号** | **球员编号** | **支付时间** | **总数** |
| PAYMENTNO | PLAYERNO | PAYMENT\_DATE | AMOUNT |
| 1 | 6 | 1980-12-08 | 100 |
| 2 | 44 | 1981-05-05 | 75 |
| 3 | 27 | 1983-09-10 | 100 |
| 4 | 104 | 1984-12-08 | 50 |
| 5 | 44 | 1980-12-08 | 25 |
| 6 | 8 | 1980-12-08 | 25 |
| 7 | 44 | 1982-12-30 | 30 |
| 8 | 27 | 1984-11-12 | 75 |

--创建测试表

create table players(

playerno integer not null,

name varchar(15) not null,

initials varchar(10) not null,

birth\_date date,

sex varchar(10) not null,

joinedsmall int not null,

street varchar(30) not null,

houseno varchar(10),

postcode varchar(10),

town varchar(30) not null,

phoneno varchar(15),

leagueno varchar(10),

primary key (playerno)

);

create table penalties(

paymentno integer not null,

playerno integer not null,

payment\_date date not null,

amount decimal(7,2) not null,

primary key (paymentno)

);

--插入数据集

insert into players values (2, 'everett', 'r', '1948-09-01', 'm', 1975, 'stoney road','43', '3575nh', 'stratford', '070-237893', '2411'),

(6, 'parmenter', 'r', '1964-06-25', 'm', 1977, 'haseltine lane','80', '1234kk', 'stratford', '070-476537', '8467'),

(7, 'wise', 'gws', '1963-05-11', 'm', 1981, 'edgecombe way','39', '9758vb', 'stratford', '070-347689', null),

(8, 'newcastle', 'b', '1962-07-08', 'f', 1980, 'station road','4', '6584wo', 'inglewood', '070-458458', '2983'),

(27, 'collins', 'dd', '1964-12-28', 'f', 1983, 'long drive','804', '8457dk', 'eltham', '079-234857', '2513'),

(28, 'collins', 'c', '1963-06-22', 'f', 1983, 'old main road','10', '1294qk', 'midhurst', '010-659599', null),

(39, 'bishop', 'd', '1956-10-29', 'm', 1980, 'eaton square','78', '9629cd', 'stratford', '070-393435', null),

(44, 'baker', 'e', '1963-01-09', 'm', 1980, 'lewis street','23', '4444lj', 'inglewood', '070-368753', '1124'),

(57, 'brown', 'm', '1971-08-17', 'm', 1985, 'edgecombe way','16', '4377cb', 'stratford', '070-473458', '6409'),

(83, 'hope', 'pk', '1956-11-11', 'm', 1982, 'magdalene road','16a', '1812up', 'stratford', '070-353548', '1608'),

(95, 'miller', 'p', '1963-05-14', 'm', 1972, 'high street','33a', '5746op', 'douglas', '070-867564', null),

(100, 'parmenter', 'p', '1963-02-28', 'm', 1979, 'haseltine lane','80', '6494sg', 'stratford', '070-494593', '6524'),

(104, 'moorman', 'd', '1970-05-10', 'f', 1984, 'stout street','65', '9437ao', 'eltham', '079-987571', '7060'),

(112, 'bailey', 'ip', '1963-10-01', 'f', 1984, 'vixen road','8', '6392lk', 'plymouth', '010-548745', '1319');

insert into penalties values (1, 6, '1980-12-08',100),

(2, 44, '1981-05-05', 75),

(3, 27, '1983-09-10',100),

(4,104, '1984-12-08', 50),

(5, 44, '1980-12-08', 25),

(6, 8, '1980-12-08', 25),

(7, 44, '1982-12-30', 30),

(8, 27, '1984-11-12', 75);

子查询的类型

from型子查询, 表子查询

获取编号小于10的男性球员号码

select \* from players where playerno<10 and sex="m";

在做子查询时一定要注意把需求拆解开

编号小于10

select \*from players where playerno<10;

男性球员号码

select \* from players where sex="m"

子查询, 子查询后面必须要有as, 把子查询作为中间表

select \* from (select \* from players where sex="m") as a where playerno<10;

或者

select \* from (select \* from players where playerno<10) as a where sex="m";

注意这里的子查询不能放在select的后面, select的后面是字段, 而子查询的结果是一个表, 只能放在from的后面.

where型子查询 行子查询, where子查询返回的是行

获取和100号球员性别相同且居住在同一城市的球员号码

拆解需求, 先查询第100号球员的性别和城市

select sex,town from players where playerno=100;

因为players表中没有其它的字段, 所以这里的子查询不能使用as对sex和town进行重命名

select playerno,sex,town from players where (sex,town)=(select sex,town from players where playerno=100)

获取和27号球员出生在同一年的球员号码

select year(birth\_date) from players where playerno=27;

子查询的结果返回的是一条记录, 所以可以使用=

select playerno, birth\_date from players where year(birth\_date)=(select year(birth\_date) from players where playerno=27);

远程上课系统, 上课点名, 上课提问, 上课回答问题等的功能. 使用QQ群作为上课答疑群有很多的不方便.

子查询相当于是两个表的嵌套关系, 但实际中一般不使用子查询, 而是使用连接查询.

列子查询

select \*,(select year(curdate())-year(employee.birthday)) as 年龄 from employee where name like '张%' group by name like '张%' having 年龄>25;

列子查询和where子查询最大的区别是, 列子查询的返回值是一列, 因此不能使用 = > <, 而是使用

in 在指定项里

any 与子查询任何返回结果比较

all 与子查询所有返回结果比较

1. in

获取球员性别为女的所有球员的球员号, 名字及所在城市

select playerno,name,town from players where sex="f"

子查询的结果返回的是多条数据, 所以使用in, 这就是where型子查询和from型子查询之间最大的区别

select playerno,name,town from players where playerno in (select playerno from players where sex="f");

经验: 拿到一个需求之后, 不要一上来就去想怎么用子查询来实现, 而是想怎么使用最简单的方法来实现, 使程序的执行量最少.

如果有两张工作表, 直接使用连接查询连接为一个大表, 再使用where查询就可以了.

2. any

获取至少比同城的另一球员年轻的所有球员的号码, 日期和居住城市

select playerno,birth\_date,town from players as p1 where birth\_date>any(select birth\_date from players as p2 where p1.town=p2.town)

子查询返回的是多条记录, 但是是与p1表中的数据逐条进行比较的, p2表每返回一条数据, 就与p1表中的所有数据进行比较, 所以可以对any使用>号, 只要年龄比任一个球员的年龄大, 就会出现在结果集中, 也就是抛弃的是最大的年龄. 出生年份比较大, 年龄就会比较小.

3. all

获取最老的球员的号码, 名字及生日

select playerno,name,birth\_date from players where birth\_date<=all(select birth\_date from players);

select playerno,name,birth\_date from players order by birth\_date limit 1;

select playerno,name,min(birth\_date) from players;

一个需求可以有多种方法, 尽量使用最简单的方法. 如果数据量不超过500G的话, 一般不需要特别考虑查询的速度, 但子查询例外.

3. exists 存在型子查询

只要子查询中至少返回一个值, exists语句就为true

1. 获取那些至少支付了一次罚款的姓名和首字母, 只要penalties中存在players中的信息, 就表明这个球员支付过罚款.

select \* from players;

select \* from penalties;

select \* from penalties where penalites.playerno=playerno.playerno

select name,initials from players where exists (select \* from penalites where penalites.playerno=playerno.playerno)

2. 获取那些从未支付过罚款的球员名字和首字母

not exists

select name,initials from players where not exists (select \* from penalites where penalites.playerno=playerno.playerno)

## 4.4高级查询

### --数字运算

#每个部门平均值、最小值、最大值、工资合计、部门人数、获得提成人数, 在使用count计数时空值不计数.

select

deptno as 部门编号,

avg(sal) as 平均值,

min(sal) as 最小值,

max(sal) as 最大值,

sum(sal) as 工资合计,

count(ename) as 部门人数,

count(comm) as 获得提成人数

from emp group by deptno;

#每个部门人均提成的算法

select

deptno as 部门编号,

-- avg(comm) 错误的人均提成算法,

avg(coalesce(comm,0)) # 正确的人均提成算法. 因为空值加减乘除任何值的结果都为空, 所以要把空值转换为0. 但是可以对空值使用count来计数.

from emp group by deptno;

### --日期运算

select

hiredate as 聘用日期,

date\_add(hiredate,interval 1 day) as 日期加一天,

date\_add(hiredate,interval 1 month) as 日期加一月,

date\_add(hiredate,interval -1 month) as 日期减一月

from emp;

### --字符串运算

#截取字符串substring, ename表示截取的字段, 1表示从第1个字符开始, 4表示截取4个字符

select substring(ename,1,4) from emp;

#替换字符串

select replace(ename,"smith","SMITH") from emp;

#字符与数字合并

select concat(job,sal) as data from emp;

#数字转字符

select convert(empno,char(10)) from emp;

#如何确定字段更改成功了, 新建一张表

create table temp1 as

select convert(empno,char(10));

desc temp1;

# 虽然使用的char字段, 但是却得到了varchar类型的结果

### --正则表达式

#Mysql大小写不敏感，binary属性让大小写敏感

create table aa1(id int auto\_increment primary key,data varchar(20) binary);

insert into aa1(data) values ("123"),("abc"),("123abc"),("abc123");

insert into aa1(data) values ("DeF456"),("456dEf");

select data from aa1 where data regexp "[0-9]";

select data from aa1 where data regexp "[a-z]";

select data from aa1 where data regexp "def";

select data from aa1 where data regexp "[De][Ee][Ff]";

**单元测试**

1. 创建数据表customer（客户）、deposite（存款）、bank（银行），表结构如下：

customer的表结构:

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名称 | 类型与长度 | 中文含义 | 备注 |
| c\_id | char(6) | 客户标识 | 主键 |
| name | varchar(30) | 客户姓名 |  |
| location | Varchar(30) | 工作地点 |  |
| salary | decimal(8,2) | 工资 |  |

bank的表结构:

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名称 | 类型与长度 | 中文含义 | 备注 |
| b\_id | char(5) | 银行标识 | 主键，非空 |
| bank\_name | char(30) | 银行名次 | 非空 |

deposite的表结构:

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名称 | 类型与长度 | 中文含义 | 备注 |
| d\_id | int | 存款流水号 | 主键，自增 |
| c\_id | char(6) | 客户标识 | 外键，关联customer表的c\_id |
| b\_id | char(5) | 银行标识 | 外键，关联bank表的b\_id |
| dep \_date | date | 存入日期 |  |
| dep\_type | char(1) | 存款期限 | 1，3，5分别代表1年期、3年期和5年期 |
| amount | decimal(8,2) | 存款金额 |  |

2. 添加数据

customer的数据如下

|  |  |  |  |
| --- | --- | --- | --- |
| c\_id | name | location | salary |
| 101001 | 孙杨 | 广州 | 1234 |
| 101002 | 郭海 | 南京 | 3526 |
| 101003 | 卢江 | 苏州 | 6892 |
| 101004 | 郭惠 | 济南 | 3492 |
| 201001 | 张三 | 北京 | 6324 |

bank的数据如下：

|  |  |
| --- | --- |
| b\_id | bank\_name |
| B0001 | 工商银行 |
| B0002 | 建设银行 |
| B0003 | 中国银行 |
| B0004 | 农业银行 |

deposite的数据如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| d\_id | c\_id | b\_id | dep\_date | dep\_type | amount |
| 1 | 101001 | B0001 | 2011-04-05 | 3 | 42526 |
| 2 | 101002 | B0003 | 2012-07-15 | 5 | 66500 |
| 3 | 101003 | B0002 | 2010-11-24 | 1 | 42366 |
| 4 | 101004 | B0004 | 2008-03-31 | 1 | 62362 |
| 5 | 101001 | B0003 | 2002-02-07 | 3 | 56346 |
| 6 | 101002 | B0001 | 2004-09-23 | 3 | 353626 |
| 7 | 101003 | B0004 | 2003-12-14 | 5 | 36236 |
| 8 | 101004 | B0002 | 2007-04-21 | 5 | 26267 |
| 9 | 101001 | B0002 | 2011-02-11 | 1 | 435456 |
| 10 | 101002 | B0004 | 2012-05-13 | 1 | 234626 |
| 11 | 101003 | B0003 | 2001-01-24 | 5 | 26243 |
| 12 | 101004 | B0001 | 2009-08-23 | 3 | 45671 |

3. 按银行统计存款总数（显示信息b\_id,存款总数）

4. 查询孙杨的存款信息（显示信息：客户ID，客户姓名，银行名称，存款金额）

5. 查询日期为2011-04-05这一天进行过存款的客户ID，客户姓名，银行名称，存款金额

6. 存款在300000至500000之间的存款记录，并且地区为广州、苏州、济南，显示客户姓名name、银行名称bank\_name、存款金额amount.

7. 查询在农业银行存款第一名的客户存款信息（显示信息：客户姓名，银行名称，存款金额）

8. 查询姓“卢”的客户的存款信息（显示信息：客户姓名，银行名称，存款金额）

--单元测试

#1.创建数据表customer（客户）、deposite（存款）、bank（银行）

create table customer(

c\_id char(6) primary key not null,

name varchar(30) not null,

location varchar(30),

salary decimal(8,2)

);

create table bank(

b\_id char(5) primary key not null,

bank\_name char(30) not null

);

create table deposite(

d\_id int primary key not null auto\_increment,

c\_id char(6),

b\_id char(5),

dep\_date date,

dep\_type char(1),

amount decimal(8,2),

constraint fk\_cid foreign key(c\_id) references customer(c\_id),

constraint fk\_bid foreign key(b\_id) references bank(b\_id)

);

#2.添加数据

insert into customer values('101001','孙杨','广州',1234);

insert into customer values('101002','郭海','南京',3526);

insert into customer values('101003','卢江','苏州',6892);

insert into customer values('101004','郭惠','济南',3492);

insert into customer values('101005','张三','北京',6324);

insert into bank values('B0001','工商银行');

insert into bank values('B0002','建设银行');

insert into bank values('B0003','中国银行');

insert into bank values('B0004','农业银行');

insert into deposite values(null,'101001','B0001','2011-04-05','3',42526);

insert into deposite values(null,'101002','B0003','2012-07-15','5',66500);

insert into deposite values(null,'101003','B0002','2010-11-24','1',42366);

insert into deposite values(null,'101004','B0004','2008-03-31','1',62362);

insert into deposite values(null,'101001','B0003','2002-02-07','3',56346);

insert into deposite values(null,'101002','B0001','2004-09-23','3',353626);

insert into deposite values(null,'101003','B0004','2003-12-14','5',36236);

insert into deposite values(null,'101004','B0002','2007-04-21','5',26267);

insert into deposite values(null,'101001','B0002','2011-02-11','1',435456);

insert into deposite values(null,'101002','B0004','2012-05-13','1',234626);

insert into deposite values(null,'101003','B0003','2001-01-24','5',26243);

insert into deposite values(null,'101004','B0001','2009-08-23','3',45671);

#3.按银行统计存款总数（显示信息b\_id,存款总数）

select b\_id,sum(amount) as 存款总数 from deposite group by b\_id;

#4.查询孙杨的存款信息（显示信息：客户ID，客户姓名，银行名称，存款金额）

-- 确定使用连接查询

-- 先得到一张总的数据表

select \* from deposite

left join customer on customer.c\_id=deposite.c\_id

left join bank on bank.b\_id=deposite.b\_id;

-- 再从总的数据表中查找信息

select customer.c\_id,name,bank\_name,amount from deposite

left join customer on customer.c\_id=deposite.c\_id

left join bank on bank.b\_id=deposite.b\_id

where name='孙杨';

#5.查询日期为2011-04-05这一天进行过存款的客户ID，客户姓名，银行名称，存款金额

-- 先得到一张总的数据表

select \* from deposite d

left join customer c on c.c\_id=d.c\_id

left join bank b on b.b\_id=d.b\_id;

-- 再从总的数据表中查找信息

select c.name,c.location,b.bank\_name,d.amount,d.dep\_date from deposite d

left join customer c on c.c\_id=d.c\_id

left join bank b on b.b\_id=d.b\_id

where dep\_date='2011-04-05';

#6.存款在300000至500000之间的存款记录，地区为广州、苏州、济南，显示客户姓名name、银行名称bank\_name、存款金额amount.

-- 先得到一张总的数据表

select \* from customer c

left join deposite d on c.c\_id=d.c\_id

left join bank b on d.b\_id=b.b\_id

-- 再从总的数据表中查找信息

select c.name,c.location,b.bank\_name,d.amount from customer c

left join deposite d on c.c\_id=d.c\_id

left join bank b on d.b\_id=b.b\_id

where c.location in('广州','苏州','济南') and d.amount between 300000 and 500000;

对于重复的字段是否指明是哪个表中的问题, select中的字段, 如果有重复的, 必须要指明表名. 如果是条件中的语句, where中的, 或group by中的, 不管有无重复的字段, 都要指明表名.

#7.查询在农业银行存款第一名的客户存款信息（显示信息：客户姓名，银行名称，存款金额）

select c.name,c.location,b.bank\_name,d.amount from customer c

left join deposite d on c.c\_id=d.c\_id

left join bank b on d.b\_id=b.b\_id

where b.bank\_name = "农业银行" order by d.amount desc limit 1;

#8.查询姓“卢”的客户的存款信息（显示信息：客户姓名，银行名称，存款金额）

select c.name,c.location,b.bank\_name,d.amount from customer c

left join deposite d on c.c\_id=d.c\_id

left join bank b on d.b\_id=b.b\_id

where c.name like "卢%";

# 数据预处理

## --数据审核

完整性

"缺失值": "字段缺少",

"准确性": 记录方式是否符合业务逻辑"

拉货（买家已下单，商品晚一天到货）

在商品无货的状态下DBA对订单存储的两种方式

订单编号重复, 上面做求和, 下面第一条记录做剔除, 只留下第二条数据即可.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **订单编号** | **下单数量** | **实到数量** | **下单日期** | **到货日期** |
| 10100 | 5 | 0 | 2017/1/10 | 2017/1/13 |
| 10100 | 0 | 5 | 2017/1/10 | 2017/1/14 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **订单编号** | **下单数量** | **实到数量** | **下单日期** | **到货日期** |
| 10100 | 5 | 0 | 2017/1/10 | 2017/1/13 |
| 10100 | 5 | 5 | 2017/1/10 | 2017/1/14 |

要根据DBA的存储习惯修改自己的查询语句.

## --数据筛选

create table stu(

name varchar(20),

statistics int,

math int,

english int,

economic int

);

insert into stu values("张松","69","68","84","86"),

("王翔","91","75","95","94"),

("天宇","54","88","67","78"),

("李华","81","60","86","64"),

("赵寅","75","96","81","83"),

("宋元","83","72","66","71"),

("袁芳","75","58","76","90"),

("陈锋","87","76","92","77");

#统计学成绩为75的学生

select name,statistics from stu where statistics=75;

#英语成绩最高的前三名学生

select name,english from stu order by english desc limit 3;

#四门课程成绩都大于70的学生

select \* from stu where statistics>=70 and math>=70 and english>=70 and economic>=70;

--数据排序

select \* from stu where statistics >=70 and math >=70 and english>=70 and economic >=70

order by statistics desc, math desc, english desc, economic desc; #数据库读取规则

## --数据透视表

create table cost(

id int primary key auto\_increment,

sex varchar(20),

zone varchar(20),

avg\_live float,

avg\_cloth float,

element varchar(20)

);

insert into cost(sex,zone,avg\_live,avg\_cloth,element) values

("男","大型城市",800,200,"价格"),

("女","中小城市",600,180,"款式"),

("男","大型城市",1000,300,"品牌"),

("男","中小城市",400,40,"价格"),

("女","中小城市",500,150,"款式"),

("女","乡镇地区",800,80,"品牌"),

("男","中小城市",600,180,"品牌"),

("女","乡镇地区",400,120,"价格"),

("男","中小城市",1000,300,"款式"),

("女","大型城市",600,180,"款式"),

("女","中小城市",500,150,"价格"),

("男","乡镇地区",500,50,"价格"),

("男","乡镇地区",50,30,"价格"),

("女","中小城市",300,35,"价格"),

("女","中小城市",500,150,"款式");

#业务逻辑：

# 数据分析师探究的是数据的本质, 数据本身的规律.

#问题1: 女性对于衣服支出是否高于男性?

# 使用平均值

select sex,sum(avg\_cloth)/count(\*) as avg\_cost from cost group by sex;

#使用平均值得出的结论是否可行?

#考虑到支出占比（这就能说明女性对衣服支出低于男性么?注意：异常值）

select sex,sum(avg\_cloth)/sum(avg\_live) as avg\_cost from cost group by sex;

#考虑到地区问题

select sex,zone,sum(avg\_cloth)/sum(avg\_live) as avg\_cost from cost group by sex,zone order by zone,sex;

#问题2: 什么样的人更看重衣服的品牌和款式?

# 先看原始的数据表的结构

# element为用户的偏好

# 知道是哪些人, 把这些人挑出来才能进行进一步的比对

select \* from cost where element="品牌" or element="款式";

#性别占比(一维)

# 直接使用count(\*)是对以sex分组之后的数据进行的统计, 想要统计所有的人, 就要使用select count(\*) from cost

select sex,count(\*)/(select count(\*) from cost) as num\_sex

from cost

where element="品牌" or element="款式"

group by sex;

#性别和地区占比(二维)

select sex,zone,count(\*)/(select count(\*) from cost) as num\_sex

from cost

where element="品牌" or element="款式"

group by sex,zone;

#性别, 地区和平均生活费用占比(三维)

#即用户画像, 针对某一类客户, 从不同的维度研究用户的行为. 可以使用不同的工具得出, 可以用Excel, mysql, SPSS, R, Python等工具, 只是工具可以处理的数据的量级可能不同. 不要觉得哪个工具好, 只是不同的工具侧重点不同, 遇到一个问题, 需要知道使用哪个工具能更好更有效的去解决它.

select sex,zone,count(\*)/(select count(\*) from cost) as num\_sex,avg(avg\_live) as new\_live

from cost

where element="品牌" or element="款式"

group by sex,zone;

select sex,zone,count(\*)/(select count(\*) from cost) as num\_sex,sum(avg\_live)/count(\*) as new\_live

from cost

where element="品牌" or element="款式"

group by sex,zone;

select sum(sal) from emp group by job having sum(sal)>4000;

select sum(sal) as sum\_sal from emp group by job having sum\_sal>4000;

--电商数据库案例

--查看客户数据

select username,realname,registertime,ordercode,realreceivedmoney,totalmoney

from client a

join clientorder b on a.id=b.clientid;

select username,realname,registertime,ordercode,realreceivedmoney,totalmoney,cnname

from client a

join clientorder b on a.id=b.clientid

join clientorderdetail c on b.id=c.orderid

join GXproductName d on c.GXproductid=d.id;

select username,realname,registertime,ordercode,realreceivedmoney,totalmoney,cnname,firstclassname

from client a

join clientorder b on a.id=b.clientid

join clientorderdetail c on b.id=c.orderid

join GXproductName d on c.GXproductid=d.id

join GXProductFirstClass e on d.FirstClassid=e.id;

select firstclassname 一级类别名称,count(distinct username) 用户数,count(ordercode) 订单数,sum(realreceivedmoney) 实收总额,sum(totalmoney) 订单总额from client a

join clientorder b on a.id=b.clientid

join clientorderdetail c on b.id=c.orderid

join GXproductName d on c.GXproductid=d.id

join GXProductFirstClass e on d.FirstClassid=e.id

group by e.firstclassname;

--大客户购买（购买金额、购买频率==>客单价）

--购买金额前150

select top(150) b.username as 用户名,sum(a.TotalMoney) as 金额from clientorder a

left join client b on a.ClientID=b.ID

group by b.UserName

order by 金额desc;

--购买频数前150

select top(150) b.username as 用户名,count(a.OrderCode) as 购买次数from clientorder a

left joinclient b on a.ClientID=b.ID

group by b.UserName

order by 购买次数desc;

--客单价前150（利用RMF模型观察）

select top(150) b.username as 用户名,sum(a.TotalMoney) as 金额,count(a.OrderCode) as 购买次数,

sum(a.TotalMoney)/count(a.OrderCode) as 客单价

from clientorder a

left join clientb on a.ClientID=b.ID

group by b.UserName

order by 金额desc,购买次数desc,客单价desc;

课程回顾

1. 数据库, 数据库管理系统与mysql语句之间的联系?

SQL语句==输入==>数据库管理系统==调用==>数据库

2. 数据库创建与选择

create database database\_name;

use database\_name;

3. 创建表空间, 查询表结构(学生表:姓名为name, 年龄age, 入学时间time)

create table(name char(20),age int(10),time datetime);

show tables;

desc stu;

4. 索引的作用

提高后台查询的效率

5. 约束有哪几种, 分别的作用

非空约束, 唯一约束, 检查约束;

主键约束, 外键约束

1. 防止错误数据插入数据库, 2. 保持数据表之间的一致性

6. 往表中添加数据

insert into stu(name,age,time) values("qingshang",24,"2017-11-18");

7. SQL查询