MySQL

# 数据库

## mysql数据库的优势

1、持久化数据到本地

2、可以实现结构话查询，方便管理。

3、体积小、速度快、成本低

## 数据库的相关概念

DB(database):存储数据的长裤，它保存了一系列有组织的数据。

DBMS(Database Management System)数据库管理系统。数据库是通过DBMS创建和操作的容器。（操作DB）

常见的数据库库管理系统：

Oracle（神喻）：甲骨文（最高！）;（价格贵）

DB2:IBM;（处理海量的数据）

SQL Server:微软;（只能安装到wwindows）

Sybase：赛尔斯;

MySQL:甲骨文;

关系型数据库主要有 MySQL、 Oracle、 DB2, Infomix、 SQL Server 等数据库。

非关系型数据库主要有 NoSQL、 VoltDB 等数据库。

SQL(Structure Query Language)结构化查询语言：专门用来和数据库通信的语言。

是某个特定数据库供应商转悠的语言，几乎所有的DBMS都支持SQL。

## 数据库的特点

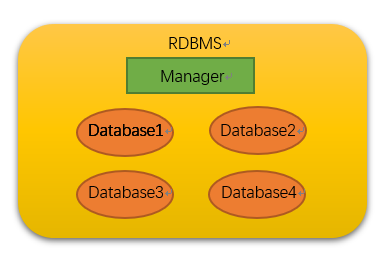
1、将数据放到表中，表在放到库中。

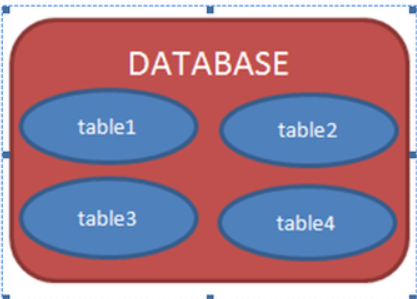
2、一个数据库中可以有多个表，每个表都有一个名字，用来表示自己。表名具有唯一性。

3、表具有一些特性，这些特性定义了数据在表中如何存储，类似java中“类”的设计。

4、表由列组成，也称之为字段。所有表都是由一个或多个列组成，每一列类似java中的属性。

5、表中的数据是按行存储，每一行雷士java中的“对象”。





# MySQL的版本

### MySQL的类别

MySQL Community Server：社区版完全免费， 但是官方不提供技术支持．

MySQL Enterprise Server：企业版能够为企业提供高性能的数据库应用，以及高稳定性的数据库系统，提供完整的数据库提交、回滚以及锁机制等功能，但是该版本收费，官方只提供电话支持。

MySQL Cluster 主要用于建立数据库集群服务器，需要在以上两个版本的基础上使用。

### 版本命名规则

MySQL 的命名机制由 3 个数字组成， 例如： MySQL-5.6.15

第1个数字5是主版本号，用于描述文件格式，表示所有版本5的发行版都有相同的文件格式。

第二个数字6是发行级别，他与主版本号组合在一起构成了发行序列号。

第3个数字15是发行系列的版本号。

# Mysql的文件的安装和配置

## Mysql的安装

注意点：安装的文件路径不能有中文。

### 卸载mysql

1、停止MySQL。

2、添加删除程序中卸载MySQL

3、到安装目录删除MySQL

4、删除：

C:\Documents and Settings\All Users\Application Data\MySQL

C：\ProgramData\MySQL

5、查看注册表：

cmd输入regedit

HKEY LOCAL MACHINE\SYSTEM\CurrentControlSet\Services

HKEY LOCAL MACHINE\SYSTEM\ControlSet001\Services

HKEY LOCAL MACHINE\SYSTEM\ControlSet002\Services

### 安装后的文件

1.MysQL安装成功后会在两个目录中存储文件：

>D:\Program Filea\MySQL\MySQL Server 5.1:DBMS管理程序

>C：\ProgramData\MysQL \MysQL Server 5.1\data:DBMs数据库文件（卸载MysQL时不会删除这个目录，需要自己手动删除）

2.MysQL重要文件

>D:\Program File3\MySQL\MySQIL Server 5.1\bin\myaql.exe:客户端程序，用来操作服务器。但必须保证服务器已开启才能连接上！

>D:\Program Files\MySQL\MySQL Server 5.1\bin\mysqld.exe:服务器程序，必须先启动它，客户端才能连接上服务器。

>D:\Program Files\MySQL\MySQL Server 5.1\bin\my.ini:服务器配置文件（里面的默认端口为3306，如没有必要不要修改）

3.C:PTOgramData\MySQL\MysQL Server 5:1\dataL（隐藏目录，都是数据库，一个文件夹都是一个数据库，里面的frm是表。该目录下的每个目录表示一个数据库，例如该目录下有一个mySQL目录，那么说明你的DBMS中有一个名为mySQL的database。

>在某个数据库目录下会有0~N个扩展名为frm的文件，每个frm文件表示一个table。你不要用文本编辑器打开它，它是由DBMS来读写的！

4.my.ini，MysQL最为重要的配置文件配置MySQL的端口：默认为3306，没有必要去修改它;

>配置字符编码：

\*[client]下配置客户端编码：default-character-set=gbk

\*[mySQLd]下配置服务器编码：character-set-server=utf8配置二进制数据大小上限：

\*在[mySQLd]下配置：maxallowed packet=8M

C：\ProgramData\MysQL\MysQLServer 5.5\data目录下：有一个目录就有一个数据库！！！每个数据库目录下，有一个frm文件，就说明有一张表！

## MYSQL启动和登陆

客户端可以通过命令行或者图形化界面登陆数据库。

### 服务器操作（该操作必须在管理员模式下）

1.开启服务器（必须保证mysql为windows服务）：net start mysql。

>查看进程表中是否存在：mysql.exe进程（存在）

例如：net start +服务器名 net start mysql80

2.关闭服务器（必须保证mysql为windows服务）：net stop mysql。

>查看进程表中是否存在：mysql.exe进程不存在客户登操作cmd

3.登录服务器：mysql -uroot -p123 - hlocalhost -P 3306

mysql 【-h主机名 -P端口号】 -u用户名 -p密码（括号可选）

>-u：后面跟随用户名

>-p：后面跟随密码

>-h：后面跟随IP 127.0.01 给出IP地址

>-P :端口号

通过MySQL文件下的MySQL 8.0 Command Line Client 登录，但该方式只支持root用户。

4.退出服务器：exit或quit或者ctrl+c

# SQL语言

## sql语言

### SQL：结构化查询语言（Structured Query Language)

### SQL的作用：客户端使用SQL来操作服务器。

>启动mysql.exe，连接服务器后，就可以使用SQL来操作服务器了。

>将来会使用Java程序连接服务器，然后使用SQL来操作服务器。

### SQL标准（例如SQL99，即1999年制定的标准）。

>由国际标准化组织（ISO）制定的，对DBMS的统一操作方式（例如相同的语句可以操作：mySQL、oracle等）。

### SQL方言

某种DEMS不只会支持SQL标准，而且还会有一些自己独有的语法，这就称之为方言！例如limit语句只在MySQL中可以使用。

### SQL语法规则

1.sql语句可以在单行或多行书写，以分号结尾。

2.可使用空格和缩进来增强语句的可读性。

3.MySQL不区别大小写，但建议关键字大写，表名，列名小写。

4.注释：

单行注释：#注释文字

单行注释：-- 注释文字

多行注释：/\* 注释文字 \*/

5、sql中是不区分字符和字符串的。

6、查看mysql的版本：

SELECT version();

status

mysql –-version（不登录的情况下）

mysql -V（不登录的情况下）

## SQL语句分类

1、DDL（Data Definition Language):数据定义语言，用来定义数据库对象：库、表、列等。

>创建、删除、修改：库、表结构！

2、DML（Data Manipulation Language):数据操作语言，用来定义数据库记录（数据）：地等节录>增、删、改：

3、DCL（Data Control Language)：数据控制语言，用来定义访问权限和安全级别：

4、DQLData Query Language）：数据查询语言，用来查询记录（数据）。

5事物查询语言。

DDL:数据库或表的结构操作

DML：对表的记录进行更新（增、删、改）

DQL:对包的记录的查询 难点

## 数据类型（创建表时指定列类型）

int：整型

double：浮点型，例如:double(5，2）表示最多5位，其中必须有2位小数，即最大值为999.99;

decimal：浮点型，在表单钱方面使用该类型，因为不会出现精度缺失问题;

char：固定长度字符串类型;char(255) 数据不做 255，则会补足到255长度，如果超过255长度，则会报错。

varchar：可变长度字符串类型;varchar(65535)，zhangSan 原来数据的长度。

但是要专门一个字节要存储的多长。如果长度大于一个字节，要增加到两个字节。

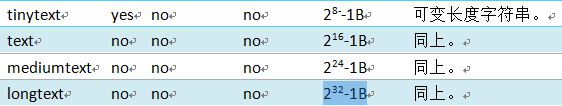
text(clob):字符串类型：mysql独有的类型，可以存储超大的字符数据。Clob为官方提供

I

>小

>中

>大



b1ob：字节类型：大的二进制字符（大字符）

>很小

>小

>中



date：日期类型，格式为：yyyy-04-dd;年月日

time：时间类型，格式为：hh:mm:sg 时分秒

timestamp：时间戳类型：年月日时分秒

# mysql中的数据类型

数值型：整型

小数：定点数和浮点数

字符型：

较短的文本：char、varchar

较长的文本：text、blob（较长的二进制数据）

日期型：

选择数据的原则：所选择的类型越简单越好，能保存数值的类型越小越好。

## 整形



分类：

tinyint、smallint、mediumint、int/integer、bigint

特点：

1、 如果不设置无符号还是有符号，默认是有符号，如果想设置无符号，需要添加unsigned关键字

2、如果插入的数值超出了整型的范围,会报out of range异常，并且插入临界值，也就是整形的最大能够存储的最大数值。

如果不设置长度，会有默认的长度

长度代表了显示的最大宽度，如果不够会用0在左边填充，但必须搭配zerofill使用！

### 如何设置无符号和有符号

DROP TABLE IF EXISTS tab\_int;

CREATE TABLE tab\_int(

t1 INT(7) ZEROFILL,//使用ZEROFILL 进行0填充。

t2 INT(7) ZEROFILL unsigned,//使用0填充之后，只支持无符号，所以unsigned没有效果。

t3 INT(7) unsigned //无符号的数字。

);

INSERT INTO tab\_int VALUES(-123456);//如果是有符号，可以插入负数。

INSERT INTO tab\_int VALUES(-123456,-123456);

INSERT INTO tab\_int VALUES(2147483648,4294967296);//插入的数值超过整形的范围。

INSERT INTO tab\_int VALUES(123,123);//显示为0000123,0000123，一共显示7位，也就是int(7)显示大小为7位，并不是代表int值的大小。

## 小数

1.浮点型

float(M,D)

double(M,D)



2.定点型

dec(M，D)

decimal(M,D)

3、特点

①

M：整数部位+小数部位

D：小数部位

如果超过各部位的范围，则插入临界值

②

M和D都可以省略

如果是decimal，则M默认为10，D默认为0

如果是float和double，则会根据插入的数值的精度来决定精度，不能超过最大的存储值。

③定点型的精确度较高，如果要求插入数值的精度较高如货币运算等则考虑使

DROP TABLE tab\_float;

CREATE TABLE tab\_float(

f1 FLOAT(5,2),

f2 DOUBLE(5,2),

f3 DECIMAL(5,2)，默认整数10位，小数位0为位。在没有设置大小的情况下，设值为123.4，实际的值为123.

f1 FLOAT //

);

SELECT \* FROM tab\_float;

DESC tab\_float;

INSERT INTO tab\_float VALUES(123.4523,123.4523,123.4523);

INSERT INTO tab\_float VALUES(123.456,123.456,123.456);

INSERT INTO tab\_float VALUES(123.4,123.4,123.4);

INSERT INTO tab\_float VALUES(1523.4,1523.4,1523.4);

## 字符型

较短的文本：char、varchar

较长的文本：text、blob(较大的二进制)

特点：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 写法 | M的意思 | 特点 | 空间的耗费 | 效率 |
| char | char(M) | 最大的字符数，可以省略，默认为1 。 | 固定长度的字符 | 比较耗费 | 高 |
| varchar | varchar(M) | 最大的字符数，不可以省略给值。 | 可变长度的字符 | 比较节省 | 低 |

注：1、char 和varchar的大小都是存储的字符，例如存储数字‘1’和‘a’和‘宋’都是字符。

2、选择原则：存储的大小的值适合char，选择不固定。

案例1：

CREATE TABLE tab\_char(

c1 ENUM('a','b','c')//插入枚举，只能插入

);

INSERT INTO tab\_char VALUES('a');//成功

INSERT INTO tab\_char VALUES('b'); //成功

INSERT INTO tab\_char VALUES('c'); //成功

INSERT INTO tab\_char VALUES('m'); //失败

INSERT INTO tab\_char VALUES('A'); //失败

案例2:

CREATE TABLE tab\_set(

s1 SET('a','b','c','d')

);

INSERT INTO tab\_set VALUES('a');

INSERT INTO tab\_set VALUES('A,B');

INSERT INTO tab\_set VALUES('a,c,d');

## 日期型

分类：

1、date只保存日期

2、time保存时间

3、year只保存年

4、datetime保存日期+时间（常用）

5、timestamp：时间戳，保存日期+时间（常用）

特点：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 字节 | 范围 | 时区等的影响 |
| datetime | 8 | 1000——9999年 | 不受 |
| timestamp | 4 | 1970-2038年 | 受时区影响 |

练习：

CREATE TABLE tab\_date(

t1 DATETIME,

t2 TIMESTAMP

);

INSERT INTO tab\_date VALUES(NOW(),NOW());//返回当前的时间、

SELECT \* FROM tab\_date;

SHOW VARIABLES LIKE 'time\_zone'; 查看时区

SET time\_zone='+9:00';//修改时区，修改为东九区，对时间戳会有影响，里面的值会自动修改为东九区的时间。datetime存储的时间不会改变。北京为东八区。

### timestamp

timestamp有两个属性，分别是CURRENT\_TIMESTAMP 和ON UPDATE CURRENT\_TIMESTAMP两种，使用情况分别如下：

1： 如果定义时DEFAULT CURRENT\_TIMESTAMP和ON UPDATE CURRENT\_TIMESTAMP子句都有，列值为默认使用当前的时间戳，并且自动更新。

2： 如果不使用DEFAULT或ON UPDATE子句，那么它等同于DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP。

3： 如果只有DEFAULT CURRENT\_TIMESTAMP子句，而没有ON UPDATE子句，列值默认为当前时间戳但不自动更新。

4： 如果没用DEFAULT子句，但有ON UPDATE CURRENT\_TIMESTAMP子句，列默认为0并自动更新。

5： 如果有一个常量值DEFAULT，该列会有一个默认值，而且不会自动初始化为当前时间戳。如果该列还有一个ON UPDATE CURRENT\_TIMESTAMP子句，这个时间戳会自动更新，否则该列有一个默认的常量但不会自动更新。

换句话说，你可以使用当前的时间戳去初始化值和自动更新，或者是其中之一，也可以都不是。（比如，你在定义的时候可以指定自动更新，但并不初始化。）下面的字段定义说明了这些情况：

|  |
| --- |
| 以下是代码片段：  CREATE TABLE test`(  ww VARCHAR( 9 ) NOT NULL ,  t1 TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP ,  t2 TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00'  ) ENGINE = MYISAM  自动初始化和更新:  ts TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP  只自动初始化:  ts TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  只自动更新  ts TIMESTAMP DEFAULT 0 ON UPDATE CURRENT\_TIMESTAMP  只是给一个常量（注：0000-00-00 00:00:00）  ts TIMESTAMP DEFAULT 0 |

# DDL

## 数据库操作

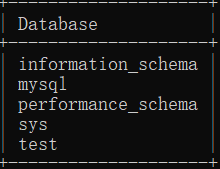
创建： create

修改： alter

删除： drop

### 查看所有数据库

格式：SHOW DATABASES 1064 表示语法错误



默认数据库：

Mysql:保存入库信息

Information\_schema:源数据信息

Performance\_schema:性能信息

test：测试数据库，没有表

### 切换数据库

格式：切换（选择要操作的）数据库：USE数据库名

成功显示：Database changed

注：操作前要先选择库。

### 创建数据库：

格式：CREATE DATABASE [IE NOT EXISTS ] database\_name [CHARSET=utf8]

[IF NOT EXISTS] :数据库不存时创建

[CHARSET=utf8] ：(编码方式)不提供编码时默认为utf8 安装时指定为utf8 括号中的内容是可选的

### 删除数据库

格式：DROP DATABASE [IF EXISTS] database\_name;

### 修改数据库编码

格式：ALTER DATABASE 数据库名 CHARACTER SET utf8（编码格式）

### 案例

|  |
| --- |
| #1、库的创建  语法：容错性的处理，使用if not exists  create database [if not exists]库名;  #案例：创建库Books  CREATE DATABASE IF NOT EXISTS books ;  #2、库的修改  RENAME DATABASE books TO 新库名;//已经废弃，修改可能会导致数据丢失，一般找到该库的文件夹进行修改文件名。  #更改库的字符集  ALTER DATABASE books CHARACTER SET gbk;  #3、库的删除  DROP DATABASE IF EXISTS books;//重复执行会报错，如果存在，就删除。 |

## 表操作

### 创建表

格式：创建表：（创建表前要指定库）

CREATE TABLE [IF NOT EXISTS] TABLE\_NAME（

列名 列的类型【(长度) 约束】，

列名 列的类型【(长度) 约束】，

列名 列的类型【(长度) 约束】，

…

列名 列的类型【(长度) 约束】

);

例如：CREATE TABLE st\_stu1(

Number char(11),

Name varchar(50),

Age int,

Gender varchar(10)

);

### 查看数据库

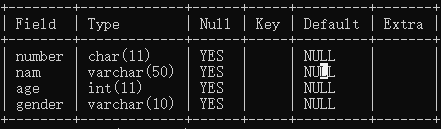
查看当前数据库中所有表名称：SHOW TABLES;

在当前的数据库查看别的数据库的表：SHOW TABLES FORM mysql

查看指定表的创建语句：SHOW CREATETABLE表名

### 查看表结构

查看表结构：DESC表名;(description)



### 删除表

表的删除：提高容错性，不清楚服务器端有没有改表。

DROP TABLE IF EXISTS 表名;

SHOW TABLES;

通用的写法：先删除数据库中库或表名，然后在新建一个表或库。

DROP DATABASE IF EXISTS 旧库名;

CREATE DATABASE 新库名;

DROP TABLE IF EXISTS 旧表名;

CREATE TABLE 表名();

### 修改表

语法：alter table 表名 add|drop|modify|change column 列名 【列类型 约束】;

注：change的column可以不加，一般情况下都加上。

>添加新列

ALTER TABLE 表名 ADD 列名 列类型;

ALTER TABLE author ADD COLUMN annual DOUBLE;

>修改之修改列类型（如果被修改的列已存在数据，那么新的类型可能会影响到已存在数据）：ALTER TABLE表名 MODIFY列名 列类型;

>修改之修改列名（字段名）：

ALTER TABLE 表名 CHANGE 【COLUMN】原列名 新列名 新列类型;

>修改之删除列：ALTER TABLE表名DROP列名;

>修改表名称：ALTER TABLE原表名 RENAME TO新表名;

>删除数据表:drop table [if exists] table\_name1,table\_name2….;

|  |
| --- |
| #①修改列名  ALTER TABLE book CHANGE COLUMN publishdate pubDate DATETIME;  #②修改列的类型或约束  ALTER TABLE book MODIFY COLUMN pubdate TIMESTAMP;  #③添加新列  ALTER TABLE author ADD COLUMN annual DOUBLE;  #④删除列  ALTER TABLE book\_author DROP COLUMN annual;  #⑤修改表名  ALTER TABLE author RENAME TO book\_author; |

### 复制表

1.仅仅复制表的结构

CREATE TABLE copy LIKE author;

2.复制表的结构+数据

CREATE TABLE copy2

SELECT \* FROM author;

3.只复制部分数据

CREATE TABLE copy3

SELECT id,au\_name

FROM author

WHERE nation='中国';

4.仅仅复制某些字段

CREATE TABLE copy4

SELECT id,au\_name

FROM author

WHERE 0;

# DML

DML（数据操作语言，它是对表记录的操作（增、删、改）！

### 插入数据

格式 INSERT INTO表 名（列名1，列名2，…）VALUE（列值1，列值2，...）;

注1、数据中所有的字符串类型，必须使用单引，不能使用双引！日期类型也要用单引。

例如：INSERT INTO STU(

Number，name，age，gender

);

VALUES(

‘ITCAST\_0001’,’zhangsan’,28,’male’

);

2、查看是否添加成功

Select\*from table\_name;

3、在表名后给出要插入的列名，其他没有指定的列等同与插入null值。所以插入记录总是插入一行，不可能是半行。

4在VALUES后给出列值，值的顺序和个数必须与前面指定的列对应。

INTERT INTO 表名 VALUES（列值1，列值2）

>没有给出要插入的列，那么表示插入所有列。

>值的个数必须是该表列的个数。

>值的顺序，必须与表创建时给出的列的顺序相同。

例如：INSERT INTO STU VALUES(

‘ITCAST\_0001’,’zhangsan’,28,’male’

);

5.插入的值的类型要与列的类型一致或兼容(隐式的转换)

INSERT INTO beauty(id,NAME,sex,borndate,phone,photo,boyfriend\_id)

VALUES(13,'唐艺昕','女','1990-4-23','1898888888',NULL,2);

6.不可以为null的列必须插入值。可以为null的列如何插入值

#方式一：添加列名，值为null

INSERT INTO beauty(id,NAME,sex,borndate,phone,photo,boyfriend\_id)

VALUES(13,'唐艺昕','女','1990-4-23','1898888888',NULL,2);

#方式二：不添加列名，也不添加值。

INSERT INTO beauty(id,NAME,sex,phone)

VALUES(15,'娜扎','女','1388888888');

7.列的顺序是否可以调换：列的顺序可以颠倒，但是字段名和值要一一对应。

INSERT INTO beauty(NAME,sex,id,phone)

VALUES('蒋欣','女',16,'110');

8.列数和值的个数必须一致。

INSERT INTO beauty(NAME,sex,id,phone)

VALUES('关晓彤','女',17,'110');

第二种插入方式：

9.可以省略列名，默认所有列，而且列的顺序和表中列的顺序一致

INSERT INTO beauty

VALUES(18,'张飞','男',NULL,'119',NULL,NULL);

### 插入两种方式的比较

1、方式一支持插入多行,方式二不支持，一条语句，批量的插入三行。

INSERT INTO beauty

VALUES(23,'唐艺昕1','女','1990-4-23','1898888888',NULL,2)

,(24,'唐艺昕2','女','1990-4-23','1898888888',NULL,2)

,(25,'唐艺昕3','女','1990-4-23','1898888888',NULL,2);

2、方式一支持子查询，方式二不支持

INSERT INTO beauty(id,NAME,phone)

SELECT 26,'宋茜','11809866';//可以在这里直接加入select语句，查询某一列来自哪个表。查询的结果集，插入到里面。

INSERT INTO beauty(id,NAME,phone)

SELECT id,boyname,'1234567'

FROM boys WHERE id<3;

## 修改数据

### 修改单表的记录

语法：update 表名

set 列=新值,列=新值,.../

where 筛选条件;

注：1、值的类型要与字段类型相同或者兼容。

2、中间用逗号隔开。

3、如果没有加条件，所有的值都会更新。

4、执行顺序：先找到表，然后筛选符合条件的，然后去修改值。

5、日期和字符要用单引号引注。

#案例1：修改beauty表中姓唐的女神的电话为13899888899

UPDATE beauty SET phone = '13899888899'

WHERE NAME LIKE '唐%';

#案例2：修改boys表中id好为2的名称为张飞，魅力值 10

UPDATE boys SET boyname='张飞',usercp=10

WHERE id=2;

### 修改多表记录

语法：sql92语法：(只支持内联)

update 表1 别名,表2 别名

set 列=值,...

where 连接条件

and 筛选条件;

语法sql99语法：

update 表1 别名

inner|left|right join 表2 别名

on 连接条件

set 列=值,...

where 筛选条件;

案例 1：修改张无忌的女朋友的手机号为114

UPDATE boys bo

INNER JOIN beauty b ON bo.`id`=b.`boyfriend\_id`

SET b.`phone`='119',bo.`userCP`=1000

WHERE bo.`boyName`='张无忌';

案例2：修改没有男朋友的女神的男朋友编号都为2号

UPDATE boys bo

RIGHT JOIN beauty b ON bo.`id`=b.`boyfriend\_id`

SET b.`boyfriend\_id`=2

WHERE bo.`id` IS NULL;

## 运算符：

运算符:=、！=、<>( 不等于)、>、<、>=、<=、BETWEEN...AND、IN(...）、IS NULL、NOT、OR、AND

BETWEEN...AND

例如:UPDATE student SET age=18 WHERE BETWEEN 18 and 40;//修改在18-40之间。或者是

UPDATE student SET age=18 WHERE age>= 18 and age<= 40;

IN(…)

UPDATE student SET age=18 WHERE name IN(‘zhangsan ’ ,’lisi’);条件为在in中里面的。

Not

UPDATE student SET age=18 WHERE not age>= 18 and age<= 40; 满足相反条件的。

IS Null

UPDATE student SET age=18 WHERE age=NULL;//错误的表示方法，=返回值为false

UPDATE student SET age=18 WHERE age IS NULL;//age的值为null都会修改

## 删除语句

### 方式一：delete

语法：

1、单表的删除【★】

delete from 表名 where 筛选条件

方式一：delete

注：不加筛选条件，会将整个表删除。

案例：删除手机号以9结尾的女神信息

DELETE FROM beauty WHERE phone LIKE '%9';

SELECT \* FROM beauty;

2、多表的删除【补充】

sql92语法：

delete 表1的别名,表2的别名

from 表1 别名,表2 别名

where 连接条件

and 筛选条件;

sql99语法：

delete 表1的别名,表2的别名 //表1和表2 级联删除，这里直接写表名，筛选条会筛选出。

from 表1 别名

inner|left|right join 表2 别名 on 连接条件

where 筛选条件;

2.多表的删除

#案例：删除张无忌的女朋友的信息

DELETE b

FROM beauty b

INNER JOIN boys bo ON b.`boyfriend\_id` = bo.`id`

WHERE bo.`boyName`='黄晓明';

#案例：删除黄晓明的信息以及他女朋友的信息

DELETE b,bo //级联删除，删除女朋友的同时，也删除黄晓明

FROM beauty b

INNER JOIN boys bo ON b.`boyfriend\_id`=bo.`id`

WHERE bo.`boyName`='黄晓明';

### 方式二：truncate

语法：truncate table 表名;

注：truncate中不能使用where条件，直接用来清空数据，它的清空效率比delete高。

案例：将魅力值>100的男神信息删除

TRUNCATE TABLE boys ;

### delete和truncate的区别。

1.delete 可以加where 条件，truncate不能加。

2.truncate删除，效率高些。

3.假如要删除的表中有自增长列，如果用delete删除后，再插入数据，自增长列的值从断点开始，而truncate删除后，再插入数据，自增长列的值从1开始。

4.truncate删除没有返回值，delete删除有返回值。这个返回值是被删除的行数。

5.truncate删除不能回滚，delete删除可以回滚.。

# DCL

一个项目创建一个用户，一个项目对应的数据库只有一个。这个只能对这个数据库有权限，其他的数据库无法操作。

## 创建用户

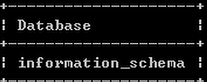
格式：\*CREATE USER用户名 @ip地址 IDENTIFIED BY密码;//在指定的ip登录

\*CREATE USER用户名 @% IDENTIFIED BY 密码;// 用户可以在任意IP地址上登录。

注：

1、用户只能在指定的IP地址上登录。

2、创建后只能看见一个库。其他库得授权。



## 给用户授权

授权格式：grant 权限 on 数据库.\* to 用户名@登录主机 identified by "密码";

>权限、用户、数据库

，给用户分派在指定的数据库上的指定的权限

>例如：GRANT CREATE，ALTER，DROP,INSERT,UPDATE,DELETE,SELECT ON mydb1.\* TO user1@localhost:

\*给user1用户分派在mydb1数据库上的create、alter、drop、insert、update、delete、select权限

\*GRANT ALL ON数据库.\*TO用户名IP地址;

>给用户分派指定数据库上的所有权限

例如：授权格式：grant 权限 on 数据库.\* to 用户名@登录主机 identified by "密码";

## ****登录MYSQL（有ROOT权限）****

　　格式：>mysql -u root -p

　 　 >密码

### **首先为用户创建一个数据库(testDB)：**

mysql>create database testDB;

## 数据库授权

### **授权**

**授权test用户拥有testDB数据库的所有权限（某个数据库的所有权限）：**

mysql>grant all privileges on testDB.\* to test@localhost identified by '1234';

 　 mysql>flush privileges;//刷新系统权限表

　　格式：grant 权限 on 数据库.\* to 用户名@登录主机 identified by "密码";

### 如果想指定部分权限给一用户，可以这样来写:

　　mysql>grant select,update on testDB.\* to test@localhost identified by '1234';

　　mysql>flush privileges; //刷新系统权限表

### **授权某些权限：**

　　mysql>grant select,delete,update,create,drop on \*.\* to test@"%" identified by "1234";

     //test用户对所有数据库都有select,delete,update,create,drop 权限。

　 //@"%" 表示对所有非本地主机授权，不包括localhost。（localhost地址设为127.0.0.1，如果设为真实的本地地址，不知道是否可以，没有验证。）

　//对localhost授权：加上一句grant all privileges on testDB.\* to test@localhost identified by '1234';即可。

### 撤销授权

格式：REVOKE权限1，…，权限n on数据库.\* FROM 用户名 自rB地址：

>撤消指定用户在指定数据库上的指定权限

>例如：REVOKE CREATE,ALTER，DROP ON mydb1.\* FROM user1@localhost:

\*撤消user1用户在mydb1数据库上的create、alter、arop权限

### 查看权限

格式:SHOW GRANTS FOR用户名@ip地址

>查看指定用户的权限

### 删除用户

格式：DROP USER用户名@ip地址

　@>mysql -u root -p

　　@>密码

 　　mysql>Delete FROM user Where User='test' and Host='localhost';

 　　mysql>flush privileges;

 　　mysql>drop database testDB; //删除用户的数据库

删除账户及权限：>drop user 用户名@'%';

　　　　　　　　>drop user 用户名@ localhost;

### 修改用户密码

修改之前先登录：

  　mysql>update mysql.user set password=password('新密码') where User="test" and Host="localhost";

  　　mysql>flush privileges;

# DQL

查询不会对数据库修改

## 字段（列）控制或者属性

语法：select 查询列表 from 表名。

查询的内容：表中的字段（属性）、常量、表达式和函数。

查询的结果：是一个 虚拟的表格，不会对表的内容进行修改。

注意点：1、操作数据之前要先打开相应得库。

2、` 符号，数字1旁边的，使用他来表示字段，例如`name` 。目的是有些字段可能与sql 的关键字相同。

3、

### 查询单个字段

select last\_name from employees;

### 查询表中的多个字段

SELECT last\_name,salary,email FROM employees;

注：字段之间用逗号隔开，且数量和属性不受限制。

### 查询表中的所有字段

SELECT last\_name,salary,email ,……FROM employees;

SELECT \* FROM employees;（\*表示所有字段，顺序为数据库存储的顺序。）

### 查询表达式

SELECT 100%98;

### 查询常量值

SELECT 100;

SELECT ‘john’

### 查询函数

SELECT VERSION();调用该方法并得到它的返回值。

### 起别名

方式一：:使用AS

SELECT 100%98 AS 结果;



SELECT last\_name AS 姓,first\_name AS 名 FROM employees;

作用：1、便于理解

2、如果要查寻的字段有重名的情况，使用别名可以区分开来。

方式二：使用空格

例如：SELECT last\_name 姓,first\_name 名 FROM employees;

案例：查询salary 结果显示为out put

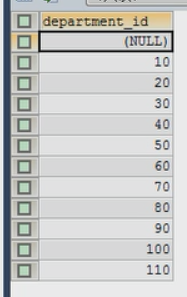
SELECT salary AS OUT put FROM employees;

别名有空格，有系统字段，或者有#号，可以用双引号讲别名标注。

### 去重

案例：查询员工表中涉及到的所有的部门编号（只是显示部门编号）

select distinct department\_id from employees;



### +号的作

java中+号的作用

运算符：两个操作数都为数值型。

连接符：只有一个操作数为字符串。

mysql中的+号：仅仅只有运算符。

例如：SELECT 90+100; 两个操作数都为数值型，则做加法运算。

SELECT ‘123’+90;其中一个为字符型，它会试图将字符型数值转换为数值型，如果转换成功，则继续做加法运算。如果转换失败，则将字符型数值转换成0。

SELECT ‘john’ +90;

SELECT null +10; 只要其中一方为null，则结果为null。

### 连接字符

concat(字段，字段，字段…..)

select

concat(last\_name , first\_name) AS 姓名

from

employees ;

## 条件查询

语法：select 查询列表 from 表名 where 筛选条件; 在这个条件下，去查询。

1. 先去执行表名。
2. 然后筛选条件。
3. 然后去查询列表

注：1、where字句不能跟别名。

### 按条件表达式筛选

条件运算符：>、< 、+、！=、<>（不等于） 、 >= 、<=

例1：查询工资>12000的员工信息

select \* from where salary>12000

例2：部门编号不等于90号的员工名和部门编号

**SELECT last\_name 名,department\_id 部门编号 FROM employees WHERE department\_id<>90**

### 按逻辑表达筛选

逻辑表达式：&& 、|| 、！

and、or、not（mysql中）

&&和and：两个条件都为true，结果为true，反之为false。

||或or：只要有一个条件为true，结果为true，繁殖为false。

案例1：查询工资在10000到20000之间的员工名、工资以及奖金。

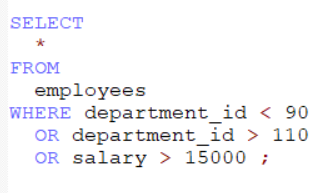
SELECT last\_name,alary,commission\_pct

FROM employees

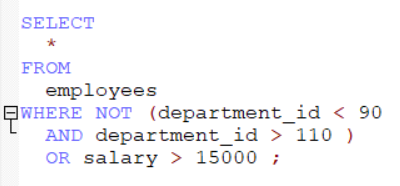
WHERE salary > 10000 AND salary < 20000 ;

作用：用于连接条件表达式。

案例2：查询部门编号不是在90到110之间，或者工资高于15000的员工信息。

、

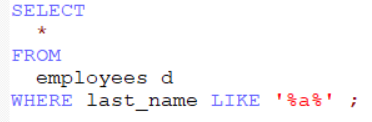
案例2（使用not）



### 模糊查询：

like 、 between and、 in 、is null

案例1：查询员工名中包含字符a的信息。（字符用‘’括起来）

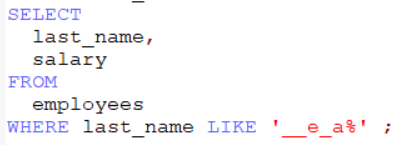


一般和通配符搭配使用

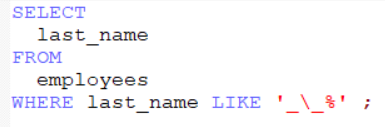
%：任意多个字符，包含0个字。

\_:任意单个字符

案列2：查询员工名中第三个字符为e，第五个字符为a的员工名和工资。

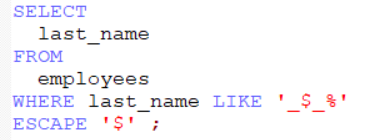


案例3：查询员工名中第二个字符为\_的员工名。（\_一般为通配符现在将它作为员工名查询））



可以将\_通过\转义。

案例3:也可以自己制定转义字符。



2、between and

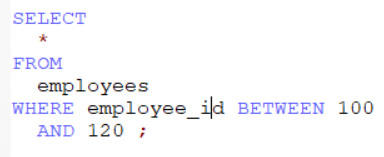
1）使用between and 可以提高语句的建解读

2）包含临界值，即100和120.

3）两个临界值不能颠倒。颠倒后悔产生下面的效果。

:

案例1：查询员工编号在100到120之间的员工信息。



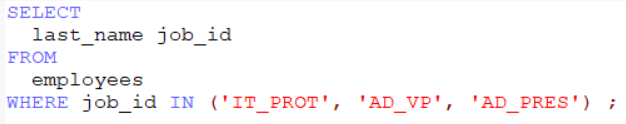
3、in关键字

作用：判断某字段的值是否-始于in列表中的某一项。

特点：1、使用in提高语句的简介度。

2、in列表的值类型必须一致或兼容

案例1：查询员工的工种编号是：IT\_PROG 、AD\_VP、AD\_PRES的一个的员工名和工种编号。

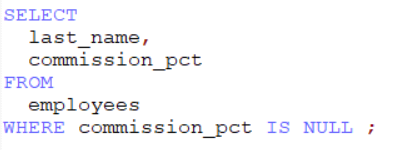


4、is null（判断该字段值是否为null）

注：=或<>不能御用判断null值。

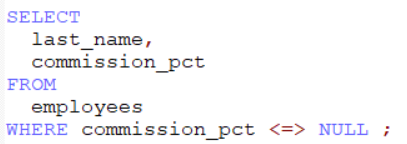
is null 或is not null 可以判断null值

案例：查询没有奖金的员工名和奖金率



5、安全等于 <=>

安全等于：判断是否与该值相同



is null 和<=>

is null:仅仅可以判断null值，可读性比较高。

<=>;既可以判断null值，又可以判断普通的数值，可读性较低

### 排序查询

语法：select 查询列表

from 表

【where 筛选条件】

order by 排序列表 【asc|desc】

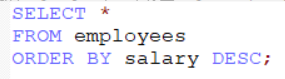
desc;由高到低。（降序）

asc：由低到高。（默认情况下）（升序）

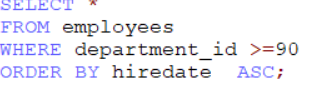
1、order by 句中可以支持单个字段、多个字段、表达式、函数、别名。

2、order by 子句一般房放在查询语句的最后面。（limit除外）

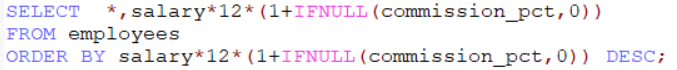
1、案例：查询员工的信息，要求工资从高到排序。



2、案例：查询编号>=90的员工信息，按入职时间的先后排序。（添加筛选条件）

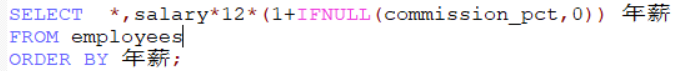


案例3：按年薪的高低显示员工的信息和年薪。（按表达排序）

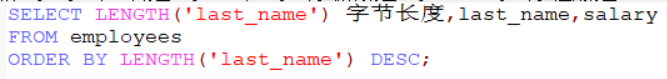


判断commission\_pct是否为null，如果是用0代替

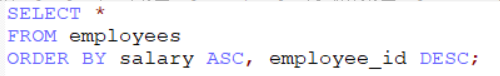
法二：也可以使用别名来操作。



案例五：按姓名的长度显示员工的姓名和工资。【按函数】



案例六：查询员工信息，先按工资排序，再按编号排序。（多个字段排序）



结果：工资由低到高，工资相同时，按id排序。

## 常见函数

概念：类似java的方法，将一组逻辑语句封装到方法体中对外暴露方法名，。

好处：1、隐藏了实现的细节。

2、提高代码的重用性。

调用：select 函数名(实参) from 表;

特点：1、函数名。 2、函数的功能

分类：1）单行函数：如 concat 、length、ifnull 等。功能：传入一个值，返回一个值。

2） 分组函数：功能：传入一组织，然后返回一个值。一般用来做统计。

常见函数：length、concat、substr、instr、trim、upper、lower、lpad、rpad、replace

数学函数：round、ceil、floor、truncate、mod

日期函数、now、curdate、curtime、year、month、monthname、day、hour、minute、second、str\_to\_date、date\_format

其他函数:version、database、user、

控制函数：if、case

### 字符函数：

#length获取参数值得字节个数。;

SELECT LENGTH('john')

SELECT LENGTH ('张三丰jajaja');

SHOW VARIABLES LIKE '%char%'; //显示系统所用的编码种类。、

#concat 拼接字符串

SELECT CONCAT(last\_name,' ',first\_name) 姓名 FROM employees;

#3.upper 变大写、lower 变小写

SELECT UPPER('john')

SELECT LOWER('joHn')

实例：将姓变大写，名变小写，然后拼接。

SELECT CONCAT(UPPER(last\_name),' ',LOWER(first\_name)) 姓名 FROM employees;

#substr 、substring

截取从指定索引除后面所有字符

SELECT SUBSTR('李莫愁爱上了陆展元',7) out\_put;截取从1开始，获取目标后的字符

截取从指定索引处指定字符长度的字符。

SELECT SUBSTR('李莫愁爱上了陆展元',1,3) out\_put;//李莫愁

案例：将姓名中首字符大写，其他字符小写然后用下划线拼接，显示出来。

SELECT CONCAT(UPPER(SUBSTR(last\_name,1,1)),'\_',LOWER(SUBSTR(first\_name,2))) out\_put FROM employees;

#instr 返回子串第一次出现的索引，如果找不到，返回0

SELECT INSTR('杨不悔爱上了殷六侠','殷六侠') AS out\_put ;

#trim

SELECT TRIM(' 张翠山 ') AS out\_put; //去除字符串中的空格

SELECT TRIM('a' FROM 'aaaaaaaaa张aaaaa翠aaaa山') AS out\_put; 去除字符串前后的 字符‘a’，中间的无法去掉。

LPAD 用指定的字符实现左填充指定长度

SELECT LPAD('应速速',10,'\*') AS OUT\_PUT; 最终总长度为10，将10改为2时，只剩速速。

rPAD 用指定的字符实现右填充指定长度

SELECT RPAD('应速速',10,'\*') AS OUT\_PUT; 最终总长度为10，将10改为2时，只剩应速。

#replace 替换

将指定字符串或为指定的字符。

SELECT REPLACE ('周芷若周芷若周芷若周芷若周芷若','周芷若','赵敏') AS OUT\_PUT;

2、数学函数

SELECT ROUND(1.65)#2

SELECT ROUND(-1.15)#-1

SELECT ROUND(1.567,2);#小数点后保留两位 结果为1.57

#ceil向上取整,返回>=该参数的最小整数

SELECT CEIL(1.52);#2

SELECT CEIL(1.02);#2

#floor 向下取整，返回小于等于该参数的最大整数

SELECT FLOOR(9.99);#-10

#truncate 截断

SELECT TRUNCATE(1.699999,2);#第二个参数是截断后小数的长度。

#mod 取余(余数的符号取决于被余数)

SELECT MOD(10,3);

SELECT MOD(-10,3);

### 时间函数

#now 返回当前系统日期和时间

SELECT NOW();

#curdate 返回当前系统日期不包含时间

SELECT CURDATE();

#curtime 返回当前时间，不包含日期

SELECT CURTIME();

#可以活期指定的部分，年、月、日、小时、分钟、秒

SELECT YEAR(NOW()) 年;

SELECT YEAR('1998-1-1') 年;

SELECT YEAR(hiredate) 年 FROM employees;

SELECT MONTH(NOW()) 月; #返回当前月

SELECT MONTHNAME(NOW()) 月;

#str\_to\_date 将字符通过指定的格式转换成日期

SELECT STR\_TO\_DATE('1998-3-2','%Y-%c-%d') AS OUT\_PUT;

#查询入职日期为1992-4-3的员工信息,但是从网站拿到的是字符串，字符串的格式可能与该格式不一样，所以需要对日期进行转换。

SELECT\*FROM employees WHERE hiredate = '1992-4-3';

SELECT\*FROM employees WHERE hiredate =STR\_TO\_DATE('4-3 1992','%c-%d %Y');



#date\_format：将日期转换为字符串

SELECT DATE\_FORMAT(NOW(),'%y年%m月%d日') AS out\_put;

#查询有奖金的员工名和入职日期(xx月/xx日 xx年)

SELECT last\_name ,DATE\_FORMAT(hiredate,'%m/%d日 %y年') 入职日期

FROM employees

WHERE commission\_pct IS NOT NULL;

### 其他函数

SELECT VERSION();#产看当前的版本号

SELECT DATABASE();#查看当前的数据库

SELECT USER(); #查看当前的用户

## #流程控制函数

### if函数

SELECT IF(10>5,'大','小');

SELECT last\_name,commission\_pct,IF(commission\_pct IS NULL,'没奖金','有奖金') 备注

FROM employees

### case函数

使用一：switch CASE 的效果

mysql中

格式：

case：要判断的字段或表达式

WHEN 常量1 THEN 要显示的值1或语句1;

WHEN 常量2 THEN 要显示的值1或语句2;

WHEN 常量3 THEN 要显示的值1或语句3;

...

ELSE 要显示的值n或语句n;

END

案例：产讯员工的工资，要求

部门号=30，显示的工资为1.1倍

部门号为=40，显示的工资为1.2倍

部门号问=50，显示的工资为1.3倍

其他部门，显示的工资为原工资

SELECT salary 原始的工资,department\_id,

CASE department\_id

WHEN 30 THEN salary\*1.1

WHEN 40 THEN salary\*1.2

WHEN 50 THEN salary\*1.3

ELSE salary

END AS 新工资

FROM employees;

使用二：类似于多重if

mysql中：

CASE

WHEN 条件1 THEN 要显示的值1或语句1

WHEN 条件2 THEN 要显示的值2或语句2

...

ELSE 要显示的

案例;查询员工的工资情况

如果工资>20000,显示A级别

如果工资>15000,显示B级别

如果工资>10000,显示C级别

否则，显示D级别

SELECT salary,

CASE

WHEN salary>20000 THEN 'a'

WHEN salary>15000 THEN 'b'

WHEN salary>10000 THEN 'c'

ELSE 'd'

END AS 工资级别

FROM employees;

## 分组函数

功能：用作同使用，又称为聚合函数或同级函数或组函数。

分类：sum求和 avg 平均值 max最大值 min最小值 count计算个数

#1、简答的使用

SELECT SUM(salary) FROM employees;

SELECT AVG(salary) FROM employees;

SELECT MIN(salary) FROM employees;

SELECT MAX(salary) FROM employees;

SELECT CONCAT(salary) FROM employees;

SELECT SUM(salary) 和, AVG(salary) 平均, MAX(salary) 最高, MIN(salary) 最低 , COUNT(salary) 个数

注：1、sum、avg 一般用于处理数值型。

max、min、count可以处理任何类型

2、以上的分组函数都忽略null值。

SELECT SUM(last\_name),AVG(last\_name) FROM employees;

SELECT MAX(last\_name),MIN(last\_name) FROM employees;

SELECT MAX(hiredate),MIN(hiredate) FROM employees;

SELECT COUNT(commission\_pct) FROM employees;

3、可以和一个关键字搭配使用（distinct）

SELECT SUM(DISTINCT salary) ,SUM(salary)FROM employees;

SELECT SUM(DISTINCT salary) ,SUM(salary)FROM employees;

SELECT COUNT(DISTINCT salary), COUNT(salary) FROM employees;

4、count函数的详细介绍

SELECT COUNT(salary) FROM employeesd;

SELECT COUNT(\*) FROM employees;//统计行数、一般用这个

SELECT COUNT(1) FROM employees; //也是统计行数，加了一列的2，然后统计2的个数。综上所述，count()可以常量值。

效率问题 ：

mysam存储引擎下，count(\*)的效率高

innodb存储引擎下，count(1)和count(\*)的效率差不多，比count(字段)要高一些。

所有用count(\*)比较多。

5和分组函数一同查询的字段有限制（和分组函数在一起的字段一般是group后的字段）

SELECT AVG(salary) 平均工资,employee\_id FROM employees;



## 分组查询

语法：

select 分组函数，列(要求出现在group by 的后面)

from 表名

where 筛选条件

order by子句

注：查询列表必须特殊，要求是分组函数和group by后出现的字段。

引入:查询每个部门的平均工，将同一个部门的成员分到一个小组。

特点：1、分组查询中的筛选条件分为两类

组前筛选：数据源为原始表 group by 字句的前面 where

组后筛选：分组后的结果集 group by 字句的后面 having

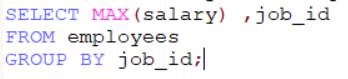
2、分组函数做条件肯定是放在having字句中

能用分组前筛选的，就有限考虑使用分组前筛选

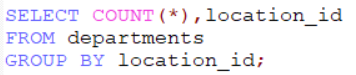
3、group by字句支持单个字段分组，多个字段分组（多个字段用逗号隔开没有顺序要求。

4、也可一添加排序，排序一般放到最后。

案例1：查询每个工种的最高工资

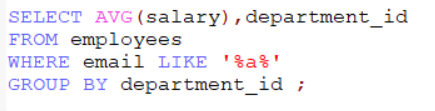


案例2：查询每个位置上的部门个数

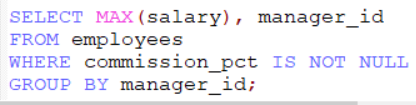


添加筛选条件

案例1：查询有相中包含a字符的，每个部门的平均工资



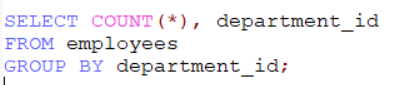
案例2：查询有奖金的每个领导手下员工的最高工资



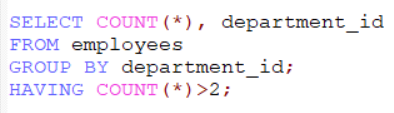
### 添加复杂的筛选条件

案例1;查询那个部门的员工数>2

1、先去查询每个部门的员工个数

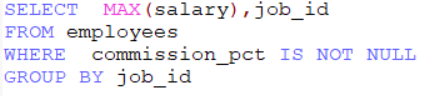


2、根据1的结果进行筛选，查询那个部门的员工个数>2，如果直接使用where COUNT(\*)查询，会报错，因为表中并没有该字段，无法对其进行查询。

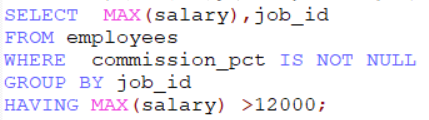


案例2：每个工种有奖金的员工的最高工资>12000的工种编号和其最高工资

1、查询每个工种有奖金的员工的最高工资

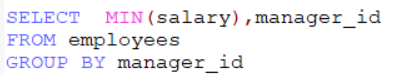


2、根据1的结果继续筛选，最高工资是否>12000

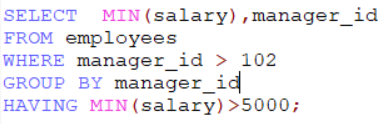


案例3：查询领导编号>102 的每个领导手下的最低工资>5000的领导编号是哪个，以及其最低工资

1、查询领导编号>102的领导手下的员工的最低工资



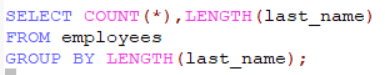
2、添加筛选条件



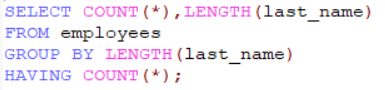
### 按表达式或函数分组

案例1：按员工姓名的长度分组，查询每一组的员工个数，筛选员工>5的有那些

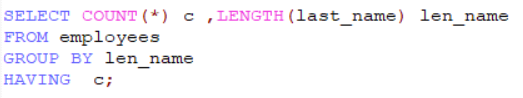
1、查询每长度的员工个数



2、添加筛选条件



3、支持添加别名



案例：查询每个部门每个工种的员工的平均工资

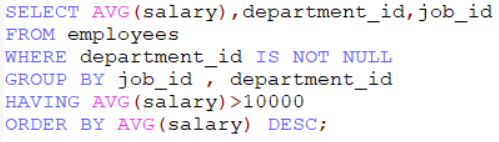
SELECT AVG(salary),department\_id,job\_id

FROM employees

GROUP BY department\_id ,job\_id

### 分组查询添加顺序

案例：查询每个部门每个工种的员工的平均工资，并且按平均工资高低显示。



## 连接查询（多表查询）

含义：又称多表查询，当查询的字段来余多个表时，就会用到连接查询。

笛卡尔乘积现象：表1 有m行，表二有n行，结果为=m\*n行

发生原因：没有有效的连接条件

分类

1、按年份分类

sql92标准（仅仅支持内连接）

sql99标准（支持所有的内、外连接（左外和右外）+交叉连接）

2、功能分类

内连接：

等值连接

非等值连接

自连接

外链接：

左外连接

右外连接

全外连接

交叉连接

### 等值连接

案例1：查询女神名和男神名

select name,boyName from boys,beauty

where beauty.boyfriend\_id = boys.id;（分别依次匹配条件是否满足。

例如：

SELECT NAME,boyName FROM boys,beauty

WHERE beauty.boyfriend\_id = boys.id;



案例2、查询员工名和对应的部门名

SELECT last\_name ,department\_name

FROM employees,departments

WHERE employees.`department\_id`=departments.`department\_id`;

### 非等值连接

案例1:查询员工工资和工资级别

**SELECT salary, grade\_level**

**FROM employees, job\_grades g**

**WHERE salary BETWEEN g.'lowest\_sal' AND g.'highest\_sal';**

### 自连接

案例1：查询员工名、上级的名称

根据员工中上级的id 第二次在搜查表中管理id进行匹配

SELECT e.employee\_id ,e.last\_name,m.employee\_id ,m.last\_name

FROM employees e,employees m

WHERE e.`manager\_id`=m.`employee\_id`;

（sql99）

## sql99

### 语法标准

1999年推出的标准

语法：select 查询列表

from 表1 别名 [连接类型] join 表2 别名

on 连接条件

where 筛选条件

group by 分组

having 筛选条件

order by 排序列表

### 连接类型

内连接：inner

外连接： 左外:left[outer]

右外:right[outer]

全外:full[outer]

交叉连接cross

## 内连接（sql99）：

语法：

select 查询列表

from 表1 别名inner join 表2 别名

on 连接条件（关联列的类型最好一致，也可以隐式的转换过去，长度不影响）

分类：等值连接 非等值连接 自连接

### 等值连接

特点：

①添加排序、分组、筛选。

②inner可以省略。

③筛选条件放在where后面，连接条件放在on后面，提高分离性，便于阅读。

④inner join连接和sql92语法中的等值连接效果是一样的，都是查询多表的交集。

案例查询员工名、部门名

|  |
| --- |
| SELECT last\_name,department\_name  FROM departments d  JOIN employees e  ON e.`department\_id` = d.`department\_id`; |

案例2.查询名字中包含e的员工名和工种名（添加筛选）

|  |
| --- |
| **SELECT last\_name,job\_title**  **FROM employees e**  **INNER JOIN jobs j**  **ON e.`job\_id`= j.`job\_id`**  **WHERE e.`last\_name` LIKE '%e%';** |

查询部门个数>3的城市名和部门个数，（添加分组+筛选）

#①查询每个城市的部门个数

#②在①结果上筛选满足条件的

|  |
| --- |
| SELECT city,COUNT(\*) 部门个数  FROM departments d  INNER JOIN locations l  ON d.`location\_id`=l.`location\_id`  GROUP BY city  HAVING COUNT(\*)>3; |

#案例4.查询哪个部门的员工个数>3的部门名和员工个数，并按个数降序（添加排序）

#①查询每个部门的员工个数

|  |
| --- |
| SELECT COUNT(\*),department\_name  FROM employees e  INNER JOIN departments d  ON e.`department\_id`=d.`department\_id`  GROUP BY department\_name |

#② 在①结果上筛选员工个数>3的记录，并排序

|  |
| --- |
| SELECT COUNT(\*) 个数,department\_name  FROM employees e  INNER JOIN departments d  ON e.`department\_id`=d.`department\_id`  GROUP BY department\_name  HAVING COUNT(\*)>3  ORDER BY COUNT(\*) DESC; |

查询员工名、部门名、工种名，并按部门名降序（添加三表连接）

|  |
| --- |
| SELECT last\_name,department\_name,job\_title  FROM employees e  INNER JOIN departments d ON e.`department\_id`=d.`department\_id`  INNER JOIN jobs j ON e.`job\_id` = j.`job\_id`  ORDER BY department\_name DESC; |

### 非等值连接

|  |
| --- |
| #查询员工的工资级别  SELECT salary,grade\_level  FROM employees e  JOIN job\_grades g  ON e.`salary` BETWEEN g.`lowest\_sal` AND g.`highest\_sal`;      #查询工资级别的个数>20的个数，并且按工资级别降序  SELECT COUNT(\*),grade\_level  FROM employees e  JOIN job\_grades g  ON e.`salary` BETWEEN g.`lowest\_sal` AND g.`highest\_sal`  GROUP BY grade\_level  HAVING COUNT(\*)>20  ORDER BY grade\_level DESC; |

### 自连接

|  |
| --- |
| #查询员工的名字、上级的名字  SELECT e.last\_name,m.last\_name  FROM employees e  JOIN employees m  ON e.`manager\_id`= m.`employee\_id`;    #查询姓名中包含字符k的员工的名字、上级的名字  SELECT e.last\_name,m.last\_name  FROM employees e  JOIN employees m  ON e.`manager\_id`= m.`employee\_id`  WHERE e.`last\_name` LIKE '%k%'; |

## 外连接（sql99）

外连接：使用外连接可以看到参与连接的某一方不满足连接条件的记录，而不仅仅是满足连接。

分类：左外连接和右外连接

左外连接：显示左表的记录，以及右表匹配的记录。

右外连接：显示右表的记录，以及左表匹配的记录。

应用场景：用于查询一个表中有，另一个表没有的记录

特点：

1、外连接的查询结果为主表中的所有记录

如果从表中有和它匹配的，则显示匹配的值

如果从表中没有和它匹配的，则显示null

外连接查询结果=内连接结果+主表中有而从表没有的记录

2、左外连接，left join左边的是主表

右外连接，right join右边的是主表

3、左外和右外交换两个表的顺序，可以实现同样的效果

4、全外连接=内连接的结果+表1中有但表2没有的+表2中有但表1没有的

|  |
| --- |
| #引入：查询男朋友 不在男神表的的女神名    SELECT \* FROM beauty;  SELECT \* FROM boys;    #左外连接  SELECT b.\*,bo.\*  FROM boys bo  LEFT OUTER JOIN beauty b  ON b.`boyfriend\_id` = bo.`id`  WHERE b.`id` IS NULL;    #案例1：查询哪个部门没有员工  #左外  SELECT d.\*,e.employee\_id  FROM departments d  LEFT OUTER JOIN employees e  ON d.`department\_id` = e.`department\_id`  WHERE e.`employee\_id` IS NULL;    #右外  SELECT d.\*,e.employee\_id  FROM employees e  RIGHT OUTER JOIN departments d  ON d.`department\_id` = e.`department\_id`  WHERE e.`employee\_id` IS NULL;    #全外  USE girls;  SELECT b.\*,bo.\*  FROM beauty b  FULL OUTER JOIN boys bo  ON b.`boyfriend\_id` = bo.id;  #交叉连接    SELECT b.\*,bo.\*  FROM beauty b  CROSS JOIN boys bo;    #sql92和 sql99pk  /\*  功能：sql99支持的较多  可读性：sql99实现连接条件和筛选条件的分离，可读性较高  \*/ |

## 联合查询

union 联合 合并：将多条查询语句的结果合并成一个结果

语法：

查询语句1

union

查询语句2

union

...

应用场景：

要查询的结果来自于多个表，且多个表没有直接的连接关系，但查询的信息一致时

特点：★

1、要求多条查询语句的查询列数是一致的！

2、要求多条查询语句的查询的每一列的类型和顺序最好一致

3、union关键字默认去重，如果使用union all 可以包含重复项

|  |
| --- |
| #引入的案例：查询部门编号>90或邮箱包含a的员工信息  SELECT \* FROM employees WHERE email LIKE '%a%' OR department\_id>90;;  SELECT \* FROM employees WHERE email LIKE '%a%'  UNION  SELECT \* FROM employees WHERE department\_id>90;  #案例：查询中国用户中男性的信息以及外国用户中年男性的用户信息  SELECT id,cname FROM t\_ca WHERE csex='男'  UNION ALL  SELECT t\_id,tname FROM t\_ua WHERE tGender='male'; |

1）查询所有列

SELECT\*FROM表名;

SELECT \*FROM emp;

-->其中\*表示查询所有列

2）查询指定列

SELECT列1[，列2，..列N] FROM表名;（至少一列）

SELECT empno，ename，salcomm FROM表名;（只看姓名和工资）

3）完全重复的记录只一次

当查询结果的的多行记录一模一样时，只显示一行。一般查询所有列时很少会有这种情况，但只查询一列（或几列）时，这种可能就大了！

SELECT DISTINCT\*|列1，列2，...列N]FRoM表名;

SELECT DISTINCT sal FROM emp;

-->保查询员工表的工资，如果存在相同的工资只显示一次！

4）列运算

I数量类型的列可以做加、减、乘、除运算（不可以使用+来连接字符串）

SELECT sal\*1.5 FROM emp;无法转换成整数的都当成0，例如 name+100=100

SELECT sal+comm FROM emp;任何结果和null相加都为null

SELECT sal\*IFNULL<comm,0> FROM emp 判断是否为null，如果是，则用0代替。

II字符串类型可以做连续运算

SELECT CONCAT('$'，sa1)FRoM emp;可以将$符号与sal连接起来

II 转换NUll值

有时需要把ULL转换成其它值，例如com+1000时，如果com列存在NUL值，那么NULL+1000还是NULI，而我们这时希望把NULL当前o来运算。

SELECT IENULI(Comm,0)+1000 FROM emp;

-->TENULI(comm,0):如果comm中存在NULL值，那么当成0来运算。

v 给列起别名

你也许已经注意到了，当使用列运算后，查询出的结果集中的列名称很不好看，这时我们需要给列名起个别名，这样在结果集中列名就显示别名了

SELECT IFNULL(comm,0)+1000 As 奖金FROM emp;

->其中As可以省略

2.条件控制

蔡件查询II

与前面介绍的UPDATE和DELETE语句一样，SELECr语句也可以使用WHERE子句来控制记录。

\*SELECT empno,ename,sal,comm FROM emp WHERE sal>10000 AND comm IS NOT NULL;

\*SELECI empno,ename,sal FROM emp WHERE sal BETWEEN 20000 AND 30000;

\*SELECI empno,ename,job FROM ep WHERE job IN(1经理，，董事长）;

2）模糊查询

当你想查询姓张，并且姓名一共两个字的员工时，这时就可以使用模糊查询

\*SELECT\*FROM emp WHERE ename LIKE ’张\_’;匹配的字数为两个人

->模糊查询需要使用运算符：LIKE，其中\_匹配一个任意字符，注意，只匹配一个字符而不是多个。

的品工

->上面语句查询的是姓张，名字由两个字组成的

\*SELECT \* FROM emP WHERE ename LIKE: 姓名由3个字组成的员工

如果我们想查询姓张，名字几个字可以的员工时就要使用“%”了。

SELECT\*FROM emp WHERE ename LIKE‘张%’;

-一>其中匹配0~N个任意字符，所以上面语句查询的是姓张的所有员工。

SELECT \*FROM emp WHERE ename LIKE1s阿s！;

-->千万不要认为上面语句是在查询姓名中间带有阿字的员工，因为a匹配0~N个字符，所以姓名以阿开头和结尾的员工也都会查询到。

SELECT\*FROM emp WHERE ename LIKE's';

-->这个条件等同与不存在，但如果姓名为NULL的查询不出来！

排序

二、排序

1）升序

SELECT \* FROM WHERE emp ORDER BY sal ASC（排序列） ;//升序

-->按sal排序，升序！

-->其中ASC是可以省略的

2）降序

SELECT \* FROM WHERE emp ORDER BY sal DESC;

-->按comm排序，降序！

-->其中DESC不能省略

3）使用多列作为排序条件(排序列存在相同值)

SELECT \* FROM WHERE emp ORDER BY sal ASC ，comm DESC:sal升序，当然sal相同时，comm使用降序排

## 分页查询

应用场景：当要显示的数据，一页显示不全，需要分页提交sql请求。

语法：

select 查询列表

from 表

【join type join 表2

on 连接条件

where 筛选条件

group by 分组字段

having 分组后的筛选

order by 排序的字段】

limit 【offset,】size;

offset要显示条目的起始索引（起始索引从0开始）

size 要显示的条目个数

特点：

①limit语句放在查询语句的最后

②公式

要显示的页数 page，每页的条目数size

|  |
| --- |
| #案例1：查询前五条员工信息  SELECT \* FROM employees LIMIT 0,5;  SELECT \* FROM employees LIMIT 5;  #案例2：查询第11条——第25条  SELECT \* FROM employees LIMIT 10,15;  #案例3：有奖金的员工信息，并且工资较高的前10名显示出来  SELECT  \*  FROM  employees  WHERE commission\_pct IS NOT NULL  ORDER BY salary DESC  LIMIT 10 ; |

# 子查询

## 子查询

### 问题引入

如何查得所有比“CLARK”工资高的员工的信息

select \* from emp

where sal>(select sal from emp where ename='CLARK');

思考：查询工资高于平均工资的雇员名字和工资。

思考：查询和SCOTT同一部门且比他工资低的雇员名字和工资。

### 子查询定义

子查询：子查询允许把一个查询嵌套在另一个查询当中。子查询，又叫内部查询，相对于内部查询，包含内部查询的就称为外部查询。

子查询可以包含普通select可以包括的任何子句，比如：distinct、 group by、order by、limit、join和union等；但是对应的外部查询必须是以下语句之一：select、insert、update、delete、set或 者do。

子查询的位置：select 中、from 后、where 中.group by 和order by 中无实用意义。

### 子查询的使用场合

1 在查询是基于未知值时应考虑使用子查询

2 子查询必须包含在括号内

3 建议将子查询放在比较运算符的右侧，以增强可读性。

4 除非进行分页分析，否则不要在子查询中使用ORDER BY 子句。

### 子查询的分类

　 1. 标量子查询：返回单一值的标量，最简单的形式。

　　2. 列子查询：返回的结果集是 N 行一列。

　　3. 行子查询：返回的结果集是一行 N 列。

　　4. 表子查询：返回的结果集是 N 行 N 列。

1：单行子查询：子查询的结果只有一行记录可以使用（比较运算符） =,<,>,>=,<=,<>

2: 多行子查询：子查询的结果有多行行记录 不可以使用=,<,>,>=,<=,<>

### 子查询的位置

select后面：

仅仅支持标量子查询

from后面：

支持表子查询

where或having后面：★(重点)

标量子查询（单行子查询） √

列子查询 （多行子查询 ） √

行子查询

exists后面（相关子查询）

表子查询

## where或having后面

1、标量子查询（单行子查询）

2、列子查询（多行子查询）

3、行子查询（多列多行）

特点：

①子查询放在小括号内

②子查询一般放在条件的右侧

③标量子查询，一般搭配着单行操作符使用

> < >= <= = <>

列子查询，一般搭配着多行操作符使用in/not in、any/some、all

in：等于列表准的任意一个

any/some：和子查询返回的值作比较。

=any：等于any中任何一个。

>any()大于any中的最小值就可以。

<any()小于any中的最大值就可以了。

all：和子查询的所有值作比较，例如：>all():大于all中的最大值。

④子查询的执行优先于主查询执行，主查询的条件用到了子查询的结果

### 标量子查询

|  |
| --- |
| #1.标量子查询★  #案例1：谁的工资比 Abel 高?  #①查询Abel的工资  SELECT salary  FROM employees  WHERE last\_name = 'Abel'  #②查询员工的信息，满足 salary>①结果  SELECT \*  FROM employees  WHERE salary>(  SELECT salary  FROM employees  WHERE last\_name = 'Abel'  );  #案例2：返回job\_id与141号员工相同，salary比143号员工多的员工 姓名，job\_id 和工资  #①查询141号员工的job\_id  SELECT job\_id  FROM employees  WHERE employee\_id = 141  #②查询143号员工的salary  SELECT salary  FROM employees  WHERE employee\_id = 143  #③查询员工的姓名，job\_id 和工资，要求job\_id=①并且salary>②（可以放两个子查询）  SELECT last\_name,job\_id,salary  FROM employees  WHERE job\_id = (  SELECT job\_id  FROM employees  WHERE employee\_id = 141  ) AND salary>(  SELECT salary  FROM employees  WHERE employee\_id = 143  );  #案例3：返回公司工资最少的员工的last\_name,job\_id和salary(查询中带有分组函数)  #①查询公司的 最低工资  SELECT MIN(salary)  FROM employees  #②查询last\_name,job\_id和salary，要求salary=①  SELECT last\_name,job\_id,salary  FROM employees  WHERE salary=(  SELECT MIN(salary)  FROM employees  );  #案例4：查询最低工资大于50号部门最低工资的部门id和其最低工资  #①查询50号部门的最低工资  SELECT MIN(salary)  FROM employees  WHERE department\_id = 50  #②查询每个部门的最低工资  SELECT MIN(salary),department\_id  FROM employees  GROUP BY department\_id  #③ 在②基础上筛选，满足min(salary)>①  SELECT MIN(salary),department\_id  FROM employees  GROUP BY department\_id  HAVING MIN(salary)>(  SELECT MIN(salary)  FROM employees  WHERE department\_id = 50  );  #非法使用标量子查询  SELECT MIN(salary),department\_id  FROM employees  GROUP BY department\_id  HAVING MIN(salary)>(  SELECT salary  FROM employees  WHERE department\_id = 250  ); |

### 列子查询（多行子查询）

|  |
| --- |
| #案例1：返回location\_id是1400或1700的部门中的所有员工姓名  #①查询location\_id是1400或1700的部门编号  SELECT DISTINCT department\_id  FROM departments  WHERE location\_id IN(1400,1700)  #②查询员工姓名，要求部门号是①列表中的某一个  SELECT last\_name  FROM employees  WHERE department\_id <>ALL(  SELECT DISTINCT department\_id  FROM departments  WHERE location\_id IN(1400,1700)  );  #案例2：返回其它工种中比job\_id为‘IT\_PROG’工种任一工资低的员工的员工号、姓名、job\_id 以及salary  #①查询job\_id为‘IT\_PROG’部门任一工资  SELECT DISTINCT salary  FROM employees  WHERE job\_id = 'IT\_PROG'  #②查询员工号、姓名、job\_id 以及salary，salary<(①)的任意一个  SELECT last\_name,employee\_id,job\_id,salary  FROM employees  WHERE salary<ANY(  SELECT DISTINCT salary  FROM employees  WHERE job\_id = 'IT\_PROG'  ) AND job\_id<>'IT\_PROG';  #或  SELECT last\_name,employee\_id,job\_id,salary  FROM employees  WHERE salary<(  SELECT MAX(salary)  FROM employees  WHERE job\_id = 'IT\_PROG'  ) AND job\_id<>'IT\_PROG';  #案例3：返回其它部门中比job\_id为‘IT\_PROG’部门所有工资都低的员工的员工号、姓名、job\_id 以及salary  SELECT last\_name,employee\_id,job\_id,salary  FROM employees  WHERE salary<ALL(  SELECT DISTINCT salary  FROM employees  WHERE job\_id = 'IT\_PROG'  ) AND job\_id<>'IT\_PROG';  #或  SELECT last\_name,employee\_id,job\_id,salary  FROM employees  WHERE salary<(  SELECT MIN( salary)  FROM employees  WHERE job\_id = 'IT\_PROG'  ) AND job\_id<>'IT\_PROG'; |

### 行子查询（结果集一行多列或多行多列）

|  |
| --- |
| #案例：查询员工编号最小并且工资最高的员工信息  SELECT \*  FROM employees  WHERE (employee\_id,salary)=(//前后正好对应。  SELECT MIN(employee\_id),MAX(salary)  FROM employees  );  #①查询最小的员工编号  SELECT MIN(employee\_id)  FROM employees  #②查询最高工资  SELECT MAX(salary)  FROM employees  #③查询员工信息  SELECT \*  FROM employees  WHERE employee\_id=(  SELECT MIN(employee\_id)  FROM employees  )AND salary=(  SELECT MAX(salary)  FROM employees  ); |

## select后面

仅仅支持标量子查询（只能是一行一列）

|  |
| --- |
| #案例：查询每个部门的员工个数  SELECT d.\*,(  SELECT COUNT(\*)  FROM employees e  WHERE e.department\_id = d.`department\_id`  ) 个数  FROM departments d;      #案例2：查询员工号=102的部门名    SELECT (  SELECT department\_name,e.department\_id  FROM departments d  INNER JOIN employees e  ON d.department\_id=e.department\_id  WHERE e.employee\_id=102    ) 部门名; |

## from后面

将子查询结果充当一张表，要求必须起别名

|  |
| --- |
| #案例：查询每个部门的平均工资的工资等级  #①查询每个部门的平均工资  SELECT AVG(salary),department\_id  FROM employees  GROUP BY department\_id  SELECT \* FROM job\_grades;  #②连接①的结果集和job\_grades表，筛选条件平均工资 between lowest\_sal and highest\_sal  //部门，工资 与等级做笛卡尔积，然后判断条件为，工资在最高工资与最低工资之间。  SELECT ag\_dep.\*,g.`grade\_level`  FROM (  SELECT AVG(salary) ag,department\_id  FROM employees  GROUP BY department\_id  ) ag\_dep  INNER JOIN job\_grades g  ON ag\_dep.ag BETWEEN lowest\_sal AND highest\_sal; |

## exists后面（相关子查询）

语法：判断exists()中的子查询有没有值，有值返回1，没值返回0;

exists(完整的查询语句)

结果：1或0

|  |
| --- |
| SELECT EXISTS(SELECT employee\_id FROM employees WHERE salary=300000);  #案例1：查询有员工的部门名  #in  SELECT department\_name  FROM departments d  WHERE d.`department\_id` IN(  SELECT department\_id  FROM employees  )  #exists  SELECT department\_name  FROM departments d  WHERE EXISTS(  SELECT \*  FROM employees e  WHERE d.`department\_id`=e.`department\_id`//这里使用到了外层查询d。  );  #案例2：查询没有女朋友的男神信息  #in  SELECT bo.\*  FROM boys bo  WHERE bo.id NOT IN(  SELECT boyfriend\_id  FROM beauty  )  #exists  SELECT bo.\*  FROM boys bo  WHERE NOT EXISTS(  SELECT boyfriend\_id  FROM beauty b  WHERE bo.`id`=b.`boyfriend\_id`  ); |

# 练习

练习： 1. 查询出部门编号为30的所有员工

SELECT \*

FROM emp

WHERE deptno=30;

2. 查询所有销售员的姓名、编号和部门编号。

SELECT ename, empno, deptno

FROM emp

WHERE job='销售员';

3. 找出奖金高于工资的员工。

SELECT \*

FROM emp

WHERE comm > sal

4. 找出奖金高于工资60%的员工。

SELECT \*

FROM emp

WHERE comm > sal\*0.6;

5. 找出部门编号为10中所有经理，和部门编号为20中所有销售员的详细资料。

SELECT \*

FROM emp

WHERE (deptno=10 AND job='经理') OR (deptno=20 AND job='销售员')

6. 找出部门编号为10中所有经理，部门编号为20中所有销售员，还有即不是经理又不是销售员但其工资大或等于20000的所有员工详细资料。

SELECT \*

FROM emp

WHERE (deptno=10 AND job='经理') OR (deptno=20 AND job='销售员') OR (job NOT IN ('经理', '销售员') AND sal >= 20000)

8. 无奖金或奖金低于1000的员工。

SELECT \*

FROM emp

WHERE comm IS NULL OR comm < 1000

9. 查询名字由三个字组成的员工。

SELECT \*

FROM emp

WHERE ename LIKE '\_\_\_'

10.查询2000年入职的员工。

SELECT \*

FROM emp

WHERE hiredate LIKE '2000-%' 模糊查询

11. 查询所有员工详细信息，用编号升序排序

SELECT \*

FROM emp

ORDER BY empno

12. 查询所有员工详细信息，用工资降序排序，如果工资相同使用入职日期升序排序

SELECT \*

FROM emp

ORDER BY sal DESC, hiredate ASC

13.查询每个部门的平均工资

SELECT deptno,AVG(3al)

FROM emp

GROUP BY deptno

14.查询每个部门的底员数量。

SELECT deptno,COUNT(\*)

FROM emp

GROUP BY deptno

15.查询每种工作的最高工资、最低工资、人数

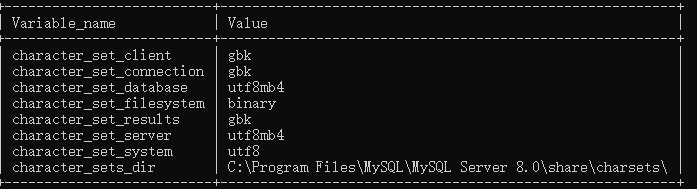
SELECT job,MAX(3al),MIN(sal),COUNT(\*)

FROM emp

GROUP BY job

1. 查看MYSQL数据库的编码

SHOW VARIABLES LIKE 'char%';



character\_set\_client=utf8,无论客户端发送的是什么编码的数据，mysq1都当成是utf8的数据！

>若客卢端发送的是GBK

>服务器会当成utf8对待

>总结：必然乱码！

处理问题的手段有两种;

>让客户端发送ut8的数据（行不通）

>把character\_set\_client修改为gbk

set character setclient=gbk;-->只在当前窗口内有效，也就是说，关闭窗口后，再打开，又回到utf8了。

character set results=utf8,把数据用什么编码发送给客户端！

>若服务器发送给客户端的是utf8的数据

>客户端会把它当成gbk，因为我们的小黑屏，只能显示gbk

>总结：必然乱码！

处理问题的手段有两种：

>让服务器发送gbk的数据：set character\_set\_results=gbk

>让小黑屏使用utf8来解读（行不通）

my.ini

在总配置文件中进行配置，可以一劳永逸

[client]

port=3306

[mysq1]

default-character-set=gbk/\*它可以一劳永逸！它可以修改三个变量：client、results、condection\*/

# mysql数据的备份和恢复

1.数据库导出SQL脚本 （不用登陆数据库）

>myaqldump -u用户名-p密码数据库名生成的脚本文件路径

例如：mysqldump-uroot-p123mydb1>c:/mydb1.sg1(与mysgl.exe和mysqld.exe一样，、都在bin目录下）

>注意，不要打分号，不要登录mysq1，直接在cmd下运行

>注意，生成的脚本文件中不包含create database语句

2、恢复数据库内容，如果将数据库删掉，可以将数据库的内容恢复，无法恢复数据库

2.执行sQ脚本

第一种方式

>myaq1-u用户名-p密码数据库<脚本文件路径

>例如：

\*先删除mydb1库，再重新创建mydb1库

\*mysq1-uroot-p123 mydb1<C:\mydbl.sql

>注意，不要打分号，不要登录mysq1，直接在cmd下运行

第二种方式I

>登录mysq1

>source SQL脚本路径

>例如：

\*先删除mydb1库，再重新创建mydb1库

\*切换到mydb1库

\*source c:\\mydb1.sq1

mysq1dump-uroot-p123 mydb3>c:/a.sq1-->备份

mysq1-uroot-p123 mydb3<c:/a.sql-->恢复

source c:/a.sql-->恢复

# 约束

## 问题引入

1)创建学生表：

create table student(

sno number(5), --学号

sname varchar2(20), --姓名

sex char(2), --性别

age number(3), --年龄

enterdate date, --入学日期

clazz number(5), --班级编号

email varchar2(50) --邮箱

);

2.插入数据

insert into student values(1001,'张三','男',20,'02-5月-2016',1,'12345@qq.com');

I nsert into student values(1001,'','你',500,'02-5月-2016',1,'12345@qq.com');

3.产生的问题1

--问题

--1：学号相同 添加主键 主键约束

--2：姓名为空 not null 非空约束

--3：性别错误 只能是男或女，默认是男 检查约束

--4：年龄过大 年龄18-30 检查约束

--5：邮箱相同 唯一 唯一约束

4、问题2

班级编号随便写，没有实际意义，需要再建立一张班级表，班级表中规定好特定班级编号，学生表中按照班级表的班级编号插入数据。

## 数据库表的约束（constraints）

1、概念

约束条件是一些规则，在对数据进行插入，删除和修改时要对规则进行验证，从而起到约束作用。

且一个表只能有一个主键，但是一个主键可以由多个字段构成，称为联合主键。

2、作用

3、数据的完整性

1：域完整性约束（非空not null，检查check），字段约束。

2：实体完整约束（唯一unique，主键primary key）。行与行之间的约束。唯一，

3：参照完整性约束（外键foreign key） 表与表的约束。

## 创建表时添加约束

### 添加列级约束

|  |
| --- |
| USE students;  DROP TABLE stuinfo;  CREATE TABLE stuinfo(  id INT PRIMARY KEY,#主键  stuName VARCHAR(20) NOT NULL UNIQUE,#非空  gender CHAR(1) CHECK(gender='男' OR gender ='女'),#检查  seat INT UNIQUE,#唯一  age INT DEFAULT 18,#默认约束，添加默认值。  majorId INT REFERENCES major(id)#外键  );  #查看stuinfo中的约束  DESC stuinfo    #查看stuinfo中的所有索引，包括主键、外键、唯一  SHOW INDEX FROM stuinfo; |

### 表级约束

语法：在各个字段的最下面

【constraint 约束名】 约束类型(字段名)

注：非空和默认不支持表级约束。

|  |
| --- |
| CREATE TABLE stuinfo(  id INT,  stuname VARCHAR(20),  gender CHAR(1),  seat INT,  age INT,  majorid INT,    CONSTRAINT pk PRIMARY KEY(id),#主键  CONSTRAINT uq UNIQUE(seat),#唯一键  CONSTRAINT ck CHECK(gender ='男' OR gender = '女'),#检查  CONSTRAINT fk\_stuinfo\_major FOREIGN KEY(majorid) REFERENCES major(id)#外键  //这里的名字也可以不起，系统默认为主键为primary，其他为各个字段名。    #通用的写法：外键方法到表级，其他为列级。  CREATE TABLE IF NOT EXISTS stuinfo(  id INT PRIMARY KEY,  stuname VARCHAR(20),  sex CHAR(1),  age INT DEFAULT 18,  seat INT UNIQUE,  majorid INT,  CONSTRAINT fk\_stuinfo\_major FOREIGN KEY(majorid) REFERENCES major(id) |

## 修改时添加约束

1、添加列级约束

alter table 表名 modify column 字段名 字段类型 新约束;

2、添加表级约束

alter table 表名 add 【constraint 约束名】 约束类型(字段名) 【外键的引用】;

|  |
| --- |
| DROP TABLE IF EXISTS stuinfo;  CREATE TABLE stuinfo(  id INT,  stuname VARCHAR(20),  gender CHAR(1),  seat INT,  age INT,  majorid INT  )  DESC stuinfo;  #1.添加非空约束  ALTER TABLE stuinfo MODIFY COLUMN stuname VARCHAR(20) NOT NULL;  #2.添加默认约束  ALTER TABLE stuinfo MODIFY COLUMN age INT DEFAULT 18;  #3.添加主键  #①列级约束  ALTER TABLE stuinfo MODIFY COLUMN id INT PRIMARY KEY;  #②表级约束  ALTER TABLE stuinfo ADD PRIMARY KEY(id);  #4.添加唯一  #①列级约束  ALTER TABLE stuinfo MODIFY COLUMN seat INT UNIQUE;  #②表级约束  ALTER TABLE stuinfo ADD UNIQUE(seat);  #5.添加外键//也可以添加名字  ALTER TABLE stuinfo ADD CONSTRAINT fk\_stuinfo\_major FOREIGN KEY(majorid) REFERENCES major(id); |

## 修改表时删除约束

|  |
| --- |
| #三、修改表时删除约束  #1.删除非空约束  ALTER TABLE stuinfo MODIFY COLUMN stuname VARCHAR(20) NULL;//也可以不写。  #2.删除默认约束  ALTER TABLE stuinfo MODIFY COLUMN age INT ;//不写  #3.删除主键  ALTER TABLE stuinfo DROP PRIMARY KEY;  #4.删除唯一  ALTER TABLE stuinfo DROP INDEX seat;  #5.删除外键  ALTER TABLE stuinfo DROP FOREIGN KEY fk\_stuinfo\_major;  SHOW INDEX FROM stuinfo; |

## 约束的分类

### 主键约束primary key （非空+唯一）

-- 1个表只能有一个主键，但是一个主键可以有多个字段构成（联合主键）

-- 联合主键只能由表级约束来定义，列级不允许

--列级约束：直接添加在列名称后面，不能起名

--表级约束：添加在所有的字段后面，可以起名，也可以在修改表的时候，添加约束

### 非空约束not null 确保字段值不允许为空

--只允许出现在列级，不允许出现在表级

### 唯一约束unique 确保字段唯一可以为空，而且多次为空

--列级约束：直接添加在列名称后面，不能起名

--表级约束：添加在所有的字段后面，可以起名，也可以在修改表的时候，添加约束

### 检查约束 check()check中所有筛选条件可以写的，都行。

--列级约束：直接添加在列名称后面，不能起名

--表级约束：添加在所有的字段后面，可以起名，也可以在修改表的时候，添加约束

### 外键约束foreign key 对多表进行约束

## 主键约束

### 列级约束（在）

create table student(

sno number(5) primary key ，//添加主键，列级约束，这里无法给约束命名。

sname varchar2(20), --姓名

sex char(2), --性别

);

### 表级约束

create table student(

sno number(5) primary key ，

sname varchar2(20), --姓名

sex char(2), --性别

constraints pk\_student\_sno primary key(sno);

//如果添加相同的编号，会返回pk\_student\_sno信息，知道错的原因。

);

### 联合主键

create table student(

sno number(5) primary key ，

sname varchar2(20), --姓名

sex char(2), --性别

constraints pk\_student\_sno primary key(sno,name);--联合主键，该主键由多个字段构成，且只能由表级元素构成，列级不容许。

);

### 修改表时添加主键约束

alter table student add constraints pk\_student\_sno primary key(sno)

该方式相当于表级约束。

### 删除主键约束

alter table student drop constraints pk\_student\_sno

## 非空约束

not null :指该字段的内容非空。

create table student(

sno number(5) primary key ，

sname varchar2(20) not null , //添加非空约束

sex char(2), --性别

constraints pk\_student\_sno primary key(sno);

);

注：该约束只能出现在列级约束，不能为表级约束。

## 唯一约束

作用：确保字段唯一，可以为空，而且多次为空。

not null :指该字段的内容非空。

### 列级约束

create table student(

sno number(5) primary key ，

sname varchar2(20) unique, //添加唯一约束

sex char(2), --性别

constraints pk\_student\_sno primary key(sno)；

);

### 表级约束

create table student(

sno number(5) primary key ，

sname varchar2(20) unique, //添加唯一约束

sex char(2), --性别

constraints pk\_student\_sno primary key(sno)，

constraints uk\_student\_emain unique(email) ;

//唯一约束。

);

### 添加唯一约束

alter table student add constraints uk\_student\_emain unique(email) ;

## 检查约束

### 列级约束

create table student(

sno number(5), --学号

sname varchar2(20), --姓名

sex char(2), --性别

age number(3) check (age>=18 and age<=30) , --列级约束

);

### 表级约束

create table student(

sno number(5), --学号

sname varchar2(20), --姓名

sex char(2), --性别

age number(3) check (age>=18 and age<=30) , --列级约束

constraints ck\_student\_age check (age>=18 and age<=30)

);

### 添加check约束

add constraints ck\_student\_age check (age>=18 and age<=30)

## 默认值default

### 列级约束

create table student(

sno number(5), --学号

sname varchar2(20), --姓名

sex char(2) default ‘男’ check(sex=’男’ or sex = ‘女’),

//同时添加检查和默认值。

age number(3) check (age>=18 and age<=30) , --列级约束

);

添加该默认值

insert into student values(1001，‘张三’，default,13)

### 表级约束

create table student(

sno number(5), --学号

sname varchar2(20), --姓名

sex char(2) default ‘男’ check(sex=’男’ or sex = ‘女’),

//同时添加检查和默认值。

age number(3) check (age>=18 and age<=30) , --列级约束

constraints ck\_student\_age check (age>=18 and age<=30)

)

添加该默认值

insert into student values(1001，‘张三’，default,13)

## 外键约束

定义：外键是关联其他表的一种约束，至少需要两张表。

注意点：

1.外键可以为null，可多次为空。

2.创建此表之前，要先创建主表。

3、要求在从表设置外键关系。

4、从表的外键列的类型和主表的关联列的类型要求一致或兼容，名称无要求。

5、主表的关联列必须是一个key（一般是主键或唯一）。

6、插入数据时，先插入主表，再插入从表。

7、删除主表的记录之前，必须先删除子表所对应的

班级表(主表或者父表) 一般现有主表后有字表。

create table clazz(

cno number(5),

cname varchar2(20) not null,

constraint pk\_clazz\_cno primary key(cno)

)

### 列级约束

学生表(子表)：

create table student(

sno number(5), --学号

sname varchar2(20), --姓名

sex char(2), --性别

age number(3), --年龄

enterdate date, --入学日期

email varchar2(50) --邮箱

clazz number(5) references clazz(cno), --班级编号//

//直接指向 references clazz(cno)

);

### 表级约束

create table student(

sno number(5), --学号

sname varchar2(20), --姓名

sex char(2), --性别

age number(3), --年龄

enterdate date, --入学日期

email varchar2(50) --邮箱

clazz number(5) references clazz(cno), --班级编号//

constraints fk\_student\_cno foreign key(cno) references clazz(cno)

表级约束。

## 主键和唯一的对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 保证唯一性 | 是否允许为空 | 一个表中可以有多少个 | 是否允许组合 |
| 主键 | √ | × | 至多有1个 | √，但不推荐 |
| 唯一 | √ | √ | 可以有多个 | √，但不推荐 |

注：1、虽然唯一键可以null值，但是不能同时又两个的值为null。

2、两个列是否组合成一个主键，组合唯一键。

## 标识列

标识列：又称为自增长列

含义：可以不用手动的插入值，系统提供默认的序列值。

特点：

1、标识列必须和主键搭配吗？不一定，但要求是一个key

2、一个表可以有几个标识列？至多一个！

3、标识列的类型只能是数值型。int foloat 或者是double

4、标识列可以通过 SET auto\_increment\_increment=3;设置步长可以通过手动插入值，设置起始值。

|  |
| --- |
| CREATE TABLE tab\_identity(  id INT ,  NAME FLOAT UNIQUE AUTO\_INCREMENT,//添加表示列，添加值时自动增长。  seat INT  );  插入值：  INSERT INTO tab\_identity(id,NAME) VALUES(NULL,'john');//有自增长的列，插入null  INSERT INTO tab\_identity(NAME) VALUES('lucy');  设置开始的增长值，但是在mysql中不能设置初始值，但是可以设置步长。  SHOW VARIABLES LIKE '%auto\_increment%';  SET auto\_increment\_increment=3; |

### 解决不能设置初始值的方法。

INSERT INTO tab\_identity(id,NAME) VALUES(10,'john');//有自增长的列，插入null

### 修改表示删除表示

ALTER TABLE 表名 MODIFY COLUMN id int；

# 事务

## 事务

# 事务(transaction)

## 事务

事务(Transaction) 一个或一组sql语句组成一个执行单元，这个执行单元要么全部执行，要么全部不执行。这列操作要么多做，要么多不做，是一个不可分割的工作单位，是数据库环境中的逻辑工作单位。

事务：事务由单独单元的一个或多个sQL语句组成，在这个单元中，每个MySQL语句是相互依赖的。而整个单独单元作为一个不可分割的整体，如果单元中某条SQL语句一且执行失败或产生错误，整个单元将会回滚。所有受到影响的数据捋返回到事物开始以前的状态；如果单元中的所有SQL语句均执行成功，则事物被顺利执行。事务控制语言(Transaction Control Language )

作用：保证数据库的完整性。

例如：案例：转账

张三丰 1000

郭襄 1000

update 表 set 张三丰的余额=500 where name='张三丰'

意外，导致郭襄的钱没有执行增加操作。

update 表 set 郭襄的余额=1500 where name='郭襄'

## 事务过程

事务开启：起始于一条dml语句（insert，update，delete）

例如：update emp1 set sal = 300 from emp where ename = ‘scott’

上述操作只是编译器中对数据进行操作，而在数据库中没有修改他的值。

事务回滚：rollback;取消dml语句产生的结果,回滚到dml语句之前。

事务提交：commit;将数据写到数据库中

结束事务

1、当用户显式执行 Commit语句提交操作或rollback语句回退。

commit：表示事务成功地结束，此时告诉系统，数据库要进入一个新的正确状态，该事务对数据库的所有更新都以交付实施。每个Commit语句都可以看成是一个事务成功的结束，同时也是另一个事务的开始。

Rollback表示事务不成功的结束，此时告诉系统，已发生错误，数据库可能处在不正确的状态，该事务对数据库的更新必须被撤销，数据库应恢复该事务到初始状态。每个Rollback语句同时也是另一个事务的开始。

一旦执行了commit语句，将目前对数据库的操作提交给数据库（实际写入DB），以后就不能用rollback进行撤销。

2、当执行DDL (create,alter、drop)语句事务自动提交(commit)

3.、用户正常断开连接时，Transaction自动提交。例如：关闭软件，切换用户

例如：ATM取钱操作

1、输入金钱

2、确认金钱

3、取钱

异常情况，中途断电，前面的操作

## 事务的ACID(acid)属性

### 原子性(Atomicty)

原子性是指事务是一个不可分割的工作单位，事务中的操作要么都发生，要么都不发生。

### 一致性（Consistency)

事务必须使数据库从一个一致性状态变换到另外一个一致性状态。例如，操作前的钱数总和为2000，操作后的钱的总和也为2000。

### 隔离性（lsolation)

事务的隔离性是指一个事务的执行不能被其他事务干扰，即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。

### 持久性（Durability)

持久性是指一个事务一旦被提交，它对数据库中数据的改变就是永久性的，接下来的其他操作和数据库故障不应该对其有任何影响。事务提交之后，就是永久性的。

## 事务的创建’

隐式事务：事务没有明显的开启和结束的标记

比如insert、update、delete语句

delete from 表 where id =1;//只要执行该语句，会默认开启事务。当有多条语句时要在同一事务语句时，要使用显式事务

显式事务：事务具有明显的开启和结束的标记

前提：必须先设置自动提交功能为禁用 语句： set autocommit=0;

注：

1、关闭之后支队当前事务有效。

2、SHOW VARIABLES LIKE 'autocommit'; 查看事务是否开启、

例如：

开启事务的语句;

update 表 set 张三丰的余额=500 where name='张三丰'

update 表 set 郭襄的余额=1500 where name='郭襄'

结束事务的语句;

### 演示事务的使用步骤

|  |
| --- |
| set autocommit=0;  步骤1：开启事务  set autocommit=0;  start transaction;可选的  步骤2：编写事务中的sql语句(select insert update delete)  语句1;  语句2;  ...  步骤3：结束事务  commit;提交事务  rollback;回滚事务//回滚值能配合jdbc使用。  savepoint 节点名;设置保存点  #1.演示事务的使用步骤  #开启事务  SET autocommit=0;  START TRANSACTION;  #编写一组事务的语句  UPDATE account SET balance = 1000 WHERE username='张无忌';  UPDATE account SET balance = 1000 WHERE username='赵敏';  #结束事务  ROLLBACK;//数据回滚，数据不发生变化，相当于存到内存中，到那时还没有写到磁盘中。  #commit;//提交事务，数据会变  #2演示savepoint 的使用  SET autocommit=0;  START TRANSACTION;  DELETE FROM account WHERE id=25;  SAVEPOINT a;#设置保存点名字为a，只搭配ROLLBACK TO使用，如果回滚的话回滚这个点。  DELETE FROM account WHERE id=28;  ROLLBACK TO a;#回滚到保存点 |

## 事务的状态

### 提交或回滚前数据的状态

1、以前的数据可恢复

2、当前的用户可以看到DML操作的结果，只有自己能看到当前的信息。

3、其他用户不能看到DML操作的结果

4、被操作的数据被锁住,其他用户不能修改这些数据。当前用户对数据库使用dml语言对数据进行操作，其他用户无法对该表进行操作。即使对该数据进行dml操作者，也是子啊当前用户执行完事务之后的才会对该表进行其他操作。

### 提交后数据的状态

1、数据的修改被永久写在数据库中.

2、数据以前的状态永久性丢失.

3、所有的用户都能看到操作后的结果.

4、记录锁被释放,其他用户可操作这些记录.

### 回滚后数据的状态

1、语句将放弃所有的数据修改

2、修改的数据被回退.

3、恢复数据以前的状态.

4、行级锁被释放.

## 事务的隔离级别

对于同时运行的多个事务，当这些事务访问数据库中相同的数据时，如果没有采取必要的隔离机制，就会导致各种并发问题。例如线程的安全问题，多个线程去访问统一个资源。

1、幻读

幻读：对于两个事务T1，T2，T1从一个表中读取了一个字段，然后T2在该表中插入了一些新的行.之后，如果T1再次读取同一个表，就会多出几行受影响。

2、不可重复读取

不可重复读：对于两个事务T1，T2，T1读取了一个字段，然后T2更新了该字段，之后，T1再次读取同一个字段，值就不同了.

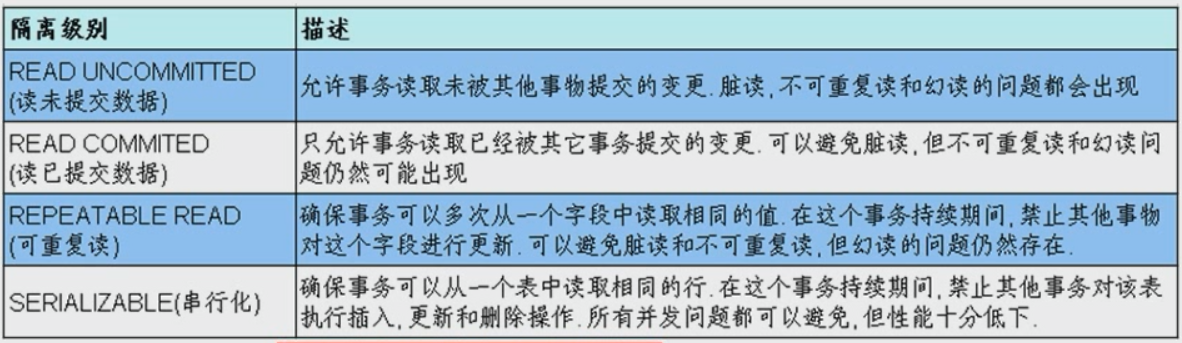
3、脏读

脏读：对于两个事务T1，T2，T1读取了已经被T2更新但还没有被提交的字段.之后，若T2回滚，T1读取的内容就是临时且无效的。、

数据库事务的隔离性：数据库系统必须具有隔离并发运行各个事务的能力，使它们不会相互影响，避免各种并发问题.

一个事务与其他事务隔离的程度称为隔离级别.数据库规定了多种事务隔离级别，不同隔离级别对应不同的干扰程度，隔离级别越高，数据一致性就越好，但并发性越弱.

### 隔离级别



Oracle支持的2种事务隔离级别：READ COMMITED，SERIALIZABLE。Oracle默认的事务隔离级别为：READCOMMITED

Mysql支持4种事务隔离级别.Mysql默认的事务隔离级别为：REPEATABLE READ

查看隔离级别：SELECT @@tx\_isolation

所有隔离级别：

1）read uncommitted : 读取尚未提交的数据 ：哪个问题都不能解决

2）read committed：读取已经提交的数据 ：可以解决脏读 ---- oracle默认的

3）repeatable read：重读读取：可以解决脏读 和 不可重复读 ---mysql默认的

4）serializable：串行化：可以解决 脏读 不可重复读 和 虚读---相当于锁表，只有当前事务的结束之后，另一个事务才能执行添加完的操做。即使做了添加操作，也不会执行插入操作。

设置： set session transaction isolation level事务隔离级别;

设置数据库系统的全局的隔离级别：

set global transaction isolation level read committed;

|  |  |  |  |
| --- | --- | --- | --- |
| 事务的隔离级别： | 脏读 | 不可重复读 | 幻读 |
| read uncommitted | √ | √ | √ |
| read committed | ×√ | √ | √ |
| repeatable read | × | × | √ |
| serializable | × | × | × |

mysql中默认 第三个隔离级别 repeatable read

oracle中默认第二个隔离级别 read committed

查看隔离级别

select @@tx\_isolation;

设置隔离级别

set session|global transaction isolation level 隔离级别;

savepoint;

# 视图

视图View：视图是从若干基本表和其他视图构造出来的表。（虚拟的表）

在创建一个视图时，只是存放的视图的定义，也即是动态检索数据的查询语句，而并不存放视图对应的数据。视图其实就是一个select 的查询结果集，真正的数据还是保存在基表中。

视图的作用

1、可以限制对数据的访问，可以给用户授予表的特定部分的访问权限而不是整个表的访问权限。

2、提供了对相同数据的不同显示。

3、一般复杂的查询使用视图。

4、修改原表会对视图产生影响。默认创建的视图，修改视图同时也会修改原表。

|  |
| --- |
| #案例：查询姓张的学生名和专业名  SELECT stuname,majorname  FROM stuinfo s  INNER JOIN major m ON s.`majorid`= m.`id`  WHERE s.`stuname` LIKE '张%';  CREATE VIEW v1  AS  SELECT stuname,majorname  FROM stuinfo s  INNER JOIN major m ON s.`majorid`= m.`id`;  //查询时之需要查询VIEW视图。  SELECT \* FROM v1 WHERE stuname LIKE '张%'; |

## 创建视图

注：创建视图之前要添加权限

### 创建默认视图

语法：

create view 视图名

as

查询语句;

|  |
| --- |
| #1.查询姓名中包含a字符的员工名、部门名和工种信息  #①创建  CREATE VIEW myv1  AS  SELECT last\_name,department\_name,job\_title  FROM employees e  JOIN departments d ON e.department\_id = d.department\_id  JOIN jobs j ON j.job\_id = e.job\_id;  #②使用  SELECT \* FROM myv1 WHERE last\_name LIKE '%a%';  #2.查询各部门的平均工资级别  #①创建视图查看每个部门的平均工资  CREATE VIEW myv2  AS  SELECT AVG(salary) ag,department\_id  FROM employees  GROUP BY department\_id;  #②使用  SELECT myv2.`ag`,g.grade\_level  FROM myv2  JOIN job\_grades g  ON myv2.`ag` BETWEEN g.`lowest\_sal` AND g.`highest\_sal`;  #3.查询平均工资最低的部门信息  SELECT \* FROM myv2 ORDER BY ag LIMIT 1;  #4.查询平均工资最低的部门名和工资  CREATE VIEW myv3  AS  SELECT \* FROM myv2 ORDER BY ag LIMIT 1;  SELECT d.\*,m.ag  FROM myv3 m  JOIN departments d  ON m.`department\_id`=d.`department\_id`; |

### 多表视图

create view view3

as

select deptno,dename,max(),min from emp e,dept d

where e.depteno = d.depteno

group by depteno,dename

having

## 使用视图

select \* from view1//如何查询表就如何查询视图

## 删除视图

drop view view1

# 练习题：

1、查询员工号为176的员工的姓名和部门号和年薪。

SELECT

last\_name,

department\_id,

salary\*12\*(1+IFNULL(commission\_pct,0)) AS 年薪 （这里要排除奖金为null的）

FROM employees;

1. 查询没有奖金和工资小于18000的salary，last\_name

SELECT salary ,last\_name

FROM employees

WHERE commission\_pct IS NULL

AND salary<18000;

1. 查询employees表中，job\_id部位‘IT’或者工资为12000的员工成员信息;

SELECT \*

FROM employees

WHERE job\_id <> 'IT' AND salary=12000;

4、查询部门deparements的表结构

DESC departments;

5、查询部门departments表中涉及到了那些位置编号

SELECT DISTINCT location\_id

FROM departments;

6、访问select \* form employees;和select \*from emploees where “commission\_pct” like ‘%%’ and last\_name like ’%%’ 是否一样。

答：不一样，判断的字段可能有null

7、查询员工的姓名和部门年薪，按年薪降序，按姓名升序。

SELECT last\_name ,department\_id,salary\*12\*(1+IFNULL(commission\_pct,0)) 年薪

FROM employees

ORDER BY 年薪 DESC, last\_name ASC;

8、选择工资不在8000到17000的员工给的姓名和工资，按工资降序。

SELECT last\_name,salary

FROM employees

WHERE salary NOT BETWEEN 8000 AND 12000;

order by salary desc;

9、查询邮箱中包含e的员工信息，并先按邮箱的字节数降序，在按部门号升序。

SELECT \*

FROM employees

WHERE email LIKE '%e%'

ORDER BY LENGTH (email) DESC,department\_id ASC;

10、显示系统时间（注：日期+时间）

select now();



11、查询工号，姓名，工资，以及工资提高20%后的结果（new salary）

SELECT employee\_id,salary,last\_name,salary\*1.2 'new salary'

FROM employees;

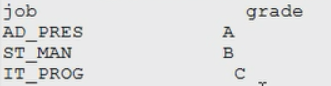
11、将员工的姓名首字母排序，写出姓名的长度（length）

SELECT LENGTH(last\_name) 长度,SUBSTR(last\_name,1,1) 首字符,last\_name

FROM employees

ORDER BY 首字符;

12、使用case-when 实现下面的条件



SELECT last\_name, job\_id AS job,

CASE job\_id

WHEN 'AD\_PRES' THEN 'A'

WHEN 'ST\_MAN' THEN 'B'

WHEN 'IT\_PROG' THEN 'C'

WHEN 'SA\_PRE' THEN 'D'

WHEN 'ST\_CLERK' THEN 'E'

END AS Grade

FROM employees

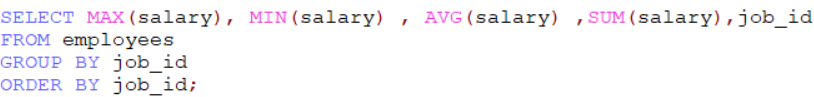
WHERE job\_id = 'AD\_PRES';

13、查询公司员工工资的最大值、最小值、平均值、总和

SELECT MAX(salary) mx\_sal, MIN(salary) mi\_sal,ROUND(AVG(salary),2) ag\_sal,SUM(salary) sm\_sal

FROM employees;

14、查询个job\_id的员工工资的最大值，最小值，总和，并按job\_id升序。

15、工最高工资和最低工资的差距（difference）

select MAX(salary)- MIN(salary) DIFFERENCE

FROM employees;

16、管理者手下员工的最低工资，其中最低工资不能低于6000，没有管理者的员工不计算在内。

SELECT MIN(salary) ,manager\_id

FROM employees

WHERE manager\_id IS NOT NULL

GROUP BY manager\_id

HAVING MIN(salary)>=6000;

17、查询所有部门的编号，员工数量，和平均工资值，并按平均工资排序。

SELECT department\_id ,COUNT(\*),AVG(salary)

FROM employees

GROUP BY department\_id

ORDER BY AVG(salary);  
18、选择具有各个job\_id的员工人数

SELECT COUNT(\*) 个数,job\_id

FROM employees

GROUP BY job\_id;

19、显示员工表中最大工资，工资平均值

SELECT MAX(salary) ,AVG(salary) FROM employees;

20、2 、查询员工表的employe\_id,job\_id last\_name 按deparment\_id降序、salary升序

SELECT

employee\_id,

job\_id,

last\_name

FROM

employees

ORDER BY department\_id DESC,

salary ASC ;

21、查询员工表的job\_id中包含a和e的，并且a在e的前面

SELECT job\_id FROM employees WHERE job\_id LIKE '%a%e%' ;

22、显示当前时间，以及去前后空格，截取字符串的函数。

select now（）

## SQL删除重复数据只保留一条数据

|  |
| --- |
| CREATE TABLE tablezzl(  id int,  name VARCHAR(255)  ); |

表结构



1、查询出重复的数据：

SELECT name FROM tablezzl GROUP BY name HAVING COUNT(name)>1

2、查询出要保留的重复数据：

SELECT MIN(id) ids FROM tablezzl GROUP BY NAME HAVING COUNT(name)>1

3、最终的SQL

DELETE FROM tablezzl

WHERE 1=1

AND `name` in (SELECT \* FROM ( (SELECT `name` `names` FROM tablezzl GROUP BY name HAVING COUNT(`name`)>1) ) a)

AND id not in (SELECT \* FROM ( (SELECT MIN(id) ids FROM tablezzl GROUP BY NAME HAVING COUNT(`name`)>1) ) b)