**oracle**

# 数据库

## 数据库的相关概念、

### 数据库组成

数据：描述事物的符号。表现形式为L文本，图形，音频。

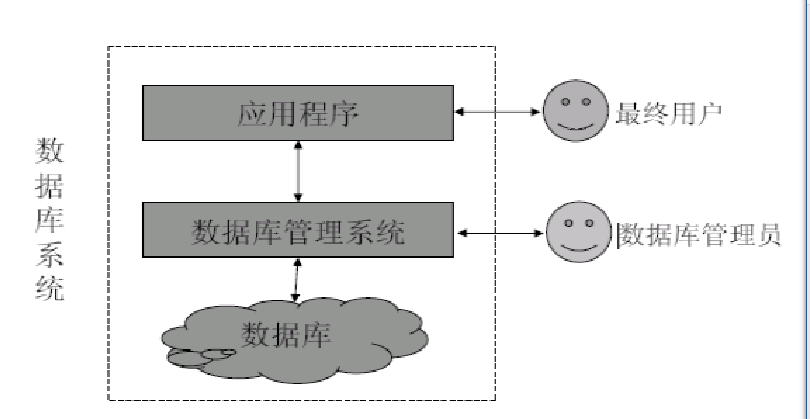
数据库：DB(database):存储数据的长裤，它保存了一系列有组织的数据。

数据库管理系统 (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）

数据库管理员(DBA:Database Administrator）: 数据库管理系统的操作者

最终用户：数据库的应用程序的使用者。

数据库系统(Database System，DBS):数据库+数据库管理系统+数据库应用程序+数据库管理员+最终用户。



### 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

关系型数据库模型是把复杂的数据结构归结为简单的二元关系（即二维表格形式）。在关系数据库中，对数据的操作几乎全部建立在一个或多个关系表格上，通过对这些关联表的表格分类、合并、连接或选取等运算来实现数据的管理。

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。

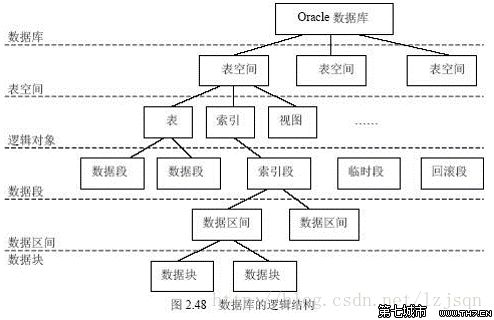
2、一个数据库中可以有多个表，每个表都有一个名字，用来表示自己。表名具有唯一性。

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”。

## 表空间



# Oracle启动和登陆

## Oracle安装问题

### 用户问题

当我们每创建一个数据库实例的时候，就会自动的生成三个用户：

sys 用户(超级管理员，权限最大)

system用户(管理员用户，权限仅次于 sys 用户)

scott 用户(普通用户，在默认情况下是锁定状态)

如果安装的时候，忘记对某个用户解锁，比如scott 可以通过 system 来对该用户解锁。步骤如下：

1、先使用 system 登录，运行sqlplus

2、使用命令：alter user scott account unlock

## Oracle开发工具和管理工具

### 开发工具

1、 sqlplusw 和 slqplus 工具是 oracle 自带的工具软件，

2、使用：

1、在开始 -- 程序 --oracle oradb\_home10g--application development--sql plus

2、在运行栏中输入： sqlplusw 即可，也可以使用 sqlplus 进入 oracle 数据库

### 管理工具

pl/sql developer，属于第三方软件，主要用于开发，测试，优化oracle pl/sql的存储过程比如：触发器，此软件 oracle 不带，需要单独安装。

## Oracle后台服务

OracleDBConsoleorcl 网络企业管理器服务。

OracleOraDb10g\_home1TNSListener 监听服务(监听端口1521) 。

OracleServiceORCL 数据库实例服务。

## sqlplus常用命令

## 登录

Win+R—cmd—>sqlplus sys//账号

账户密码：“ 密码 as sysdba” //以超级管理员登录

以sysdba用户登录，这样可以管理权限，添加用户等

Win+R—cmd—>sqlplus username/password //以指定用户名密码登录

win+R —> cmd —–> sqlplus //按照提示，输入用户名密码

## 创建用户

语法：create user username identified by password;

注：1)普通用户无法创建用户，只能由管理员创建。

2)可以在user 文件家中查看创建的用户。

3)在orcle中，数据库是唯一的，用户有多个，数据库的划分主要以用户来划分，不同的用户不能访问其他用户的表。

4)oracle规定用户的密码不能以数字开头。

### 权限，角色，用户的概念。

三者关系：

用户是最大的，某一个用户既可以给他单独的授予多个权限也可以给他授予几个特别的角色。

1.一个用户可以包含多个角色。

2.一个角色可以包含多个权限。

3.角色就是多个权限的集合。

### 修改密码

给当前用户修改密码：如果给自己修改密码可以直接使用sql>password 用户名

如果给其他用户修改密码则需要具有dba 的权限，或是拥有alter user的系统权限，也可以使用password用户名。

sql> password 用户名 ;

sql> alter user 用户名 identified by 新密码 ;

## orcle权限

对象权限，角色权限（较大），系统权限

格式：grant resource,connect to username;

注：

### 常用的角色权限

connect：连接角色。只能做登陆操作，临时用户。

resource，资源角色，可以做增删改查操作，正式用户。

dba：数据库管理员角色，最大的角色。

grant resource,connect to username;

resource:

create trigger

create sequence

create type

create procedure

create cluster

create operator

create indextype

create table

connect:

create session

### 撤销权限

格式：revoke 权限 from 用户名

前提：操作者必须为管理员，管理员可以撤销和添加自己的权限。

## 切换用户显示当前用户

conn username/password 切换用户

show user 显示当前用户名

5、使用$后面跟window命令符可以使用window命令

cls清屏cls清屏exit退出

6、在oracle软件中，启动数据库，需要以管理员身份启动

7、查询当前系统时间

select sysdate from dual;

8、导入数据文件（sql语句表数据之类的）

因为数据文件默认语言环境为英语，当前系统是中文，中英文时间显示不一致，导致倒入失败，所以 需要先修改语言环境

alter session set nls\_date\_language=english;

alter session set nls\_language=english;

然后开始导入文件使用：@ 文件全路径名 或者 start 文件全路径名

## 创建表

### 创建表

格式：CREATE TABLE [IF NOT EXISTS] TABLE\_NAME（

列名 列类型，（逗号分隔）

列名 列类型，

列名 列类型，

…

列名 列类型

);

例如：CREATE TABLE st\_stu1(

Number char(11),

Name varchar(50),

Age int,

Gender varchar(2)

);

插入数据：insert into TABLE st\_stu1 values(1001,’sx’,22,’女’)；

。

### 查看当前用户的所有表

select table\_name from user\_tables;

### 查看当前表结构

desc 表名字;//该语句为plsql语句 ，切换至命令窗口。

### 删除表

delete from 表名

### 修改表结构

1.添加列：给学生表添加一个address列，数据类型为varchar2(100);

alter table student add address varchar2(100);;

2.添加多列

alter table student add a number(10) add b number(10)

3.删除一列

alter table student drop column b

4.删除多列

alter table student drop (a,add) //删除多列不需要column

5.修改列长度

alter table student modify sname varchar2(100)

6.修改多列的长度

alter table student modify(age number(4), name varchar2(100))

7.删除表结构

drop table student

### 修改表名

rename student to student2;

11、Oracle服务意思

1.OracleDBConsoleorcl oem控制台的服务进程

2.OracleJobSchedulerXE 定时器的服务进程

3.OracleOraDb10g\_home1iSQL\*Plus isql\*plus的服务进程

4.OracleXETNSListener 监听器的服务进程

5.OracleServiceORCL 数据库服务进程

---------------------

## 导入到导出数据库

exp导入导出

exp导出文件(exp位置为\ORACLE\_HOME\BIN)

用户名：scott —>导入该用户的数据

或者直接在cmd中输入exp。

导入数据(Imp位置为\ORACLE\_HOME\BIN)

pl/sql导出

1、oracle导入导出表(工具-导入导出表)

2、sql 导入导出(工具-导入导出表,最常用)

导出：

导入：直接给.sql 文件

Oracle中的表的构成

关系数据库基本概念

-关系：整个二维表

-关系名：表格名称

-元组：行数据（记录）

-属性：列数据（字段）

-属性名：列名称（字段名）

-主键I唯一确定元组的属性组（关键字）

-域：属性的取值范围

## oracle 的数据类型

### 字符数据类型

char：不可变字符序列，固定长度字符串类型。字符长度固定，建议使用char，字符长度不固定建议使用varchar2()。char的最大指定长度为2000个字节。

例如char(10) 数据不做 10，则会补足到10长度，如果超过10长度，则会报错。

varchar2：可变长度字符串类型；varchar2(4000)，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。varchar2()最大的数据为4000个字节。

### 数值类型

number：既可以表示整数，也可以表示小数。

number(3)//长度小于999的都可以，例如：123.45

number(5,2)//逗号前面的表示总长度，逗号后面的表示保留的小数位数。

number(5,2) 1234.123是无法添加进去的。

### 日期类型

date :年月日，时分秒

timestamp：年月日，时分秒，微秒级别。

### 大对象的数据类型

BLOG：存储二进制，一般不会直接存储图片和视频对象，一般存储的是文件的地址符。

Clob：大文本数据对象。

# SQL语言

## sql

SQL是最重要的关系数据库操作语言，是所有关系数据库管理系统的标准语言。

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

### SQL的作用：

1、数据库数据的增删改查。

2、数据库对表的创建，修改和删除操作。

3、用户权限/角色的授予和取消。

4、事务控制。

### 3.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）。

### 4.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。

### 5、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！

CREATE TABLE：创建数据库表

            ALTER  TABLE：更改表结构、添加、删除、修改列长度

            DROP TABLE：删除表

            CREATE INDEX：在表上建立索引

            DROP INDEX：删除索引

本文来自 卖画的作家 的CSDN 博客 ，全文地址请点击：https://blog.csdn.net/ky415/article/details/80648590?utm\_source=copy

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：

GRANT：授予访问权限

             REVOKE：撤销访问权限

            COMMIT：提交事务处理

          ROLLBACK：事务处理回退

          SAVEPOINT：设置保存点

             LOCK：对数据库的特定部分进行锁定

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

INSERT：添加数据到数据库中

       UPDATE：修改数据库中的数据

       DELETE：删除数据库中的数据

       SELECT：选择（查询）数据

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

### DCL:对用户的创建，及授权。

## 2.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。

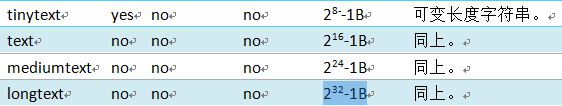
text(clob):字符串类型：mysql独有的类型，可以存储超大的字符数据。Clob为官方提供

I

>小

>中

>大



b1ob：字节类型：大的二进制字符（大字符）

>很小

>小

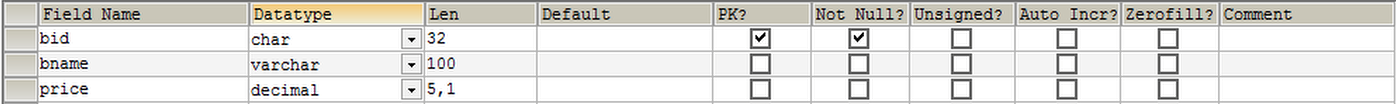
>中



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

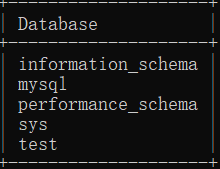
timestamp：时间戳类型：年月日时分秒



# DDL

## 数据库

\*查看所有数据库：SHOW DATABASES 1064 表示语法错误



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（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：

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：DROP DATABASE [IF EXISTS] database\_name；

\*修改数据库编码：ALTER DATABASE 数据库名 CHARACTER SET utf8（编码格式）

## 表

\*创建表：（创建表前要指定库）

CREATE TABLE [IF NOT EXISTS] TABLE\_NAME（

列名 列类型，（逗号分隔）

列名 列类型，

列名 列类型，

…

列名 列类型

);

例如：CREATE TABLE st\_stu1(

Number char(11),

Name varchar(50),

Age int,

Gender varchar(10)

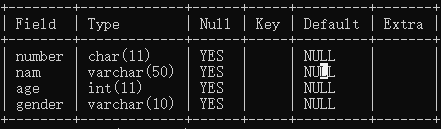
);

\*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（了解）；

查看表结构：DESC表名；(description)



删除表：DROP TABLE表名；

\*修改表：前缀：ALTERTABLE表名

>修改之添加列：

ALTER TABLE表名ADD(

列名 列类型，

列名 列类型，

列名 列类型，

….

）

>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….;

DQL:SELECT \*FROM 表名 查询表记录

Emp：

Dept：部门表

Salgrade：

# 、DML

>插入数据

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

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’

);

查看是否添加成功

Select\*from table\_name;

注：1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。

2、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’

);

>修改数据

\* UPDATE 表名SET列名1=列值1，列名2=列值2，..[WHERE条件]  
 例如：UPDATE student SET age=18 WHERE name=‘libin’ or id=‘11’；

UPDATE student SET name=’John’ where name=’Smith’

条件（条件可选的）：

>条件必须是一个boolean类型的值或表达式：

UPDATE person SET gender=’男’，age=age+1 WHERE sid='1’；

>运算符：

=、l=、<>( 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

>删除数据（记录）

DELETE FROM 表名[WHERE条件]；

\*TRUNCATE TABLE 表名：TRUNCATE是DDL语句，它是先删除drop该表，再create该表。而且无法回滚！！！

# 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。

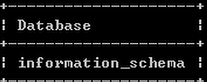
这个只能对这个数据库有权限，其他的数据库无法操作。

DCL（理解）

1.创建用户

\*CREATE USER用户名@ip地址 IDENTIFIED BY，‘密码’；

>用户只能在指定的IP地址上登录



创建后只能看见一个库。其他库得授权

\*CREATE USER用户名@‘%’ IDENTIFIED BY ’密码‘；

>用户可以在任意IP地址上登录。

2.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";

**2.1 登录MYSQL（有ROOT权限），这里以ROOT身份登录：**

　　@>mysql -u root -p

　　@>密码

**2.2 首先为用户创建一个数据库(testDB)：**

　　mysql>create database testDB;

**2.3 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";

**2.4 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

**2.5 授权test用户拥有所有数据库的某些权限：**

　　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。

3.撤销授权

\*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

4.查看权限

\*SHOW GRANTS FOR用户名@ip地址

>查看指定用户的权限

5.删除用户\*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;

6、修改用户密码

@>mysql -u root -p

  　　@>密码

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

  　　mysql>flush privileges;

# DQL

## 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。

### 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。）

### 查询常量值

SELECT 100 from dual ;

SELECT ‘john’ from dual

### 算术运算

select empno,ename,sal,sal\*1.08 from emp;

select empno,ename,sal,sal\*12 from emp;

--注：在select语句中，对数值型数据可以使用算术运算符创建表达式。只能对数值型的字符可以使用算术，不是数值型的不可以。

### 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。

### 连接运算符||

作用：可以将不同的字段连接起来显示。

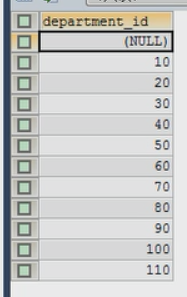
select empno||ename||sal from emp;

select "编 号:"||empno||' 姓名:'||ename||' 薪资:'||sal from emp; --字符串统一使用''括起来

### 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

1. 运算符：两个操作数都为数值型。
2. 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。

## 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

分类：

### 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

例3：查询员工职位是salesman的员工信息，字符串需要''括起来。

select \* from emp where job = 'SALESMAN'

例4：查询员工入职日期是1981/2/20

select \* from emp where hiredate = '1981/2/20'; --文字与格式不匹配

select \* from emp where hiredate = '20-2月-1981'; --固定格式，如果要是用自定义格式，需要使用函数解决

例5：查询员工部门编号不是20的员工信息。

select \* from emp where deptno!=20

select \* from emp where deptno<>20 -- <> 等同于 !=

### 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。

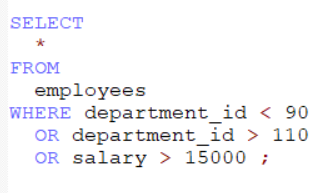
SELECT last\_name,alary,commission\_pct

FROM employees

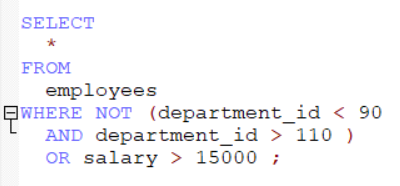
WHERE salary > 10000 AND salary < 20000 ;

作用：用于连接条件表达式。

案例1：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。

、

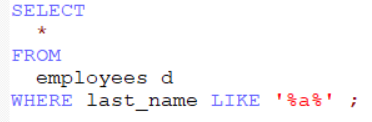
案例2（使用not）



### 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）

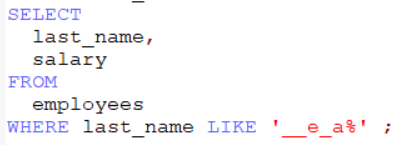


一般和通配符搭配使用

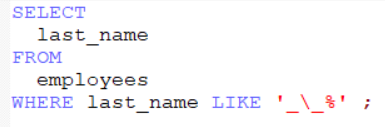
%：任意多个字符，包含0个字。

\_: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。

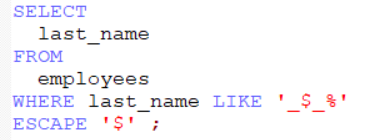


案例3：查询员工名中第二个字符为\_的员工名。（\_一般为通配符现在将它作为员工名查询））



可以将\_通过\转义。

案例3:也可以自己制定转义字符。



2、between and

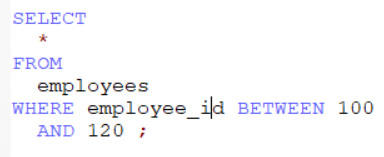
1）使用between and 可以提高语句的建解读

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。



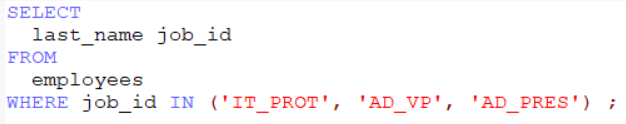
3、in关键字

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。

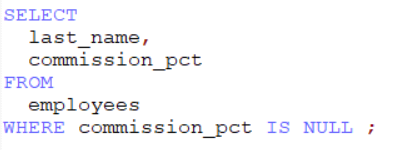


4、is null（判断该字段值是否为null）

注：=或<>不能御用判断null值。

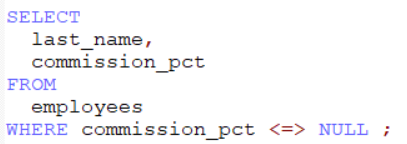
is null 或is not null 可以判断null值

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）

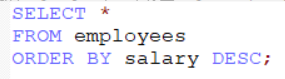
asc：由低到高。（默认情况下）（升序）

作用：将表中数据分成若干小组，然后对小组进行操作

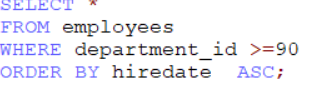
1、order by 句中可以支持单个字段、多个字段、表达式、函数、别名。

2、order by 子句一般房放在查询语句的最后面。（limit除外）

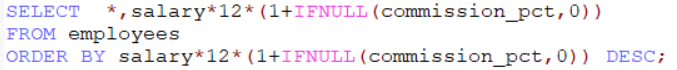
### 案例：查询员工的信息，要求工资从高到排序。



### 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）

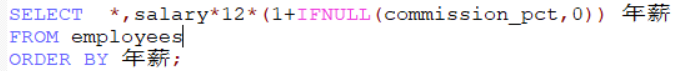


### 按年薪的高低显示员工的信息和年薪。（按表达排序）

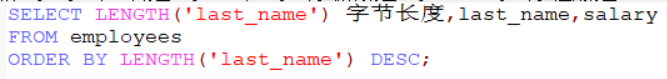


判断commission\_pct是否为0，如果是用0代替

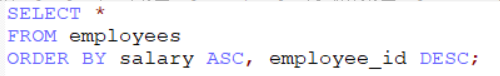
法二：也可以使用别名来操作。



### 按姓名的长度显示员工的姓名和工资。【按函数】



### 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。

### 查询员工表的所有信息，按照入职日期排序。(日期格式)进行排序

select \* from emp order by hiredate --按照入职从早到晚

select \* from emp order by hiredate desc --按照入职从晚到早

### 查询员工表的所有信息，按照姓名排序。

按字符串排序时，顺序是按字母表比较，如果字母相同，则比较下个字母。

select \* from emp order by ename

### 汉字按照字母顺序排列，但是没有实际意义

select \* from bonus order by ename;

### 多个字段同时排序 当第一个字段的值相同的时候，在依次按照后面的字段排序。

select \* from emp order by sal,ename

## 运算符的优先级

运算符的优先级：算数>连接>比较>逻辑

select \* from emp where job='SALESMAN' or job='CLERK' and sal>=1280;

select \* from emp where job='SALESMAN' or (job='CLERK' and sal>=1280);

select \* from emp where (job='SALESMAN' or job='CLERK') and sal>=1280;

# 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

1、if函数 ：if else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

2、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;

## 多行函数

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

分组前筛选：数据源为原始表 group by 字句的前面 where

分组后筛选：分组后的结果集 group by 字句的后面 having

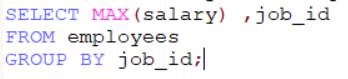
2、分组函数做条件肯定是放在having字句中

能用分组前筛选的，就有限考虑使用分组前筛选

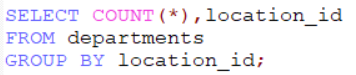
3、group by字句支持单个字段分组，多个字段分组（多个字段用逗号隔开没有顺序要求。

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

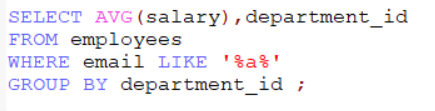


### 案例2：查询每个位置上的部门个数

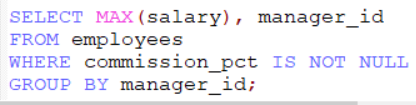


添加筛选条件

### 案例1：查询有相中包含a字符的，每个部门的平均工资



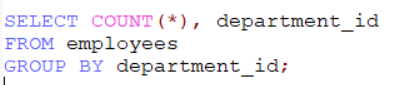
### 案例4：查询有奖金的每个领导手下员工的最高工资



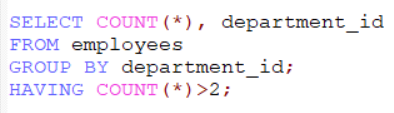
添加复杂的筛选条件

### 案例5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

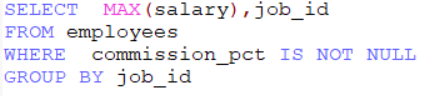


### 根据1的结果进行筛选，查询那个部门的员工个数>2，如果直接使用where COUNT(\*)查询，会报错，因为表中并没有该字段，无法对其进行查询。

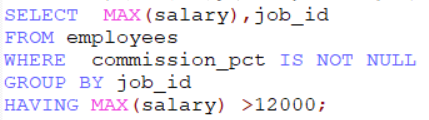


### 案例2：每个工种有奖金的员工的最高工资>12000的工种编号和其最高工资

1、查询每个工种有奖金的员工的最高工资

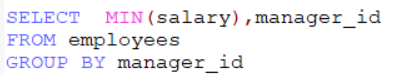


2、根据1的结果继续筛选，最高工资是否>12000

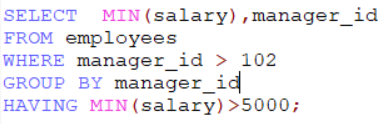


### 案例3：查询领导编号>102 的每个领导手下的最低工资>5000的领导编号是哪个，以及其最低工资

1、查询领导编号>102的领导手下的员工的最低工资



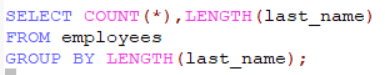
2、添加筛选条件



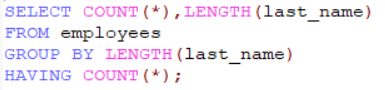
按表达式或函数分组

### 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

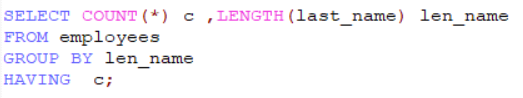
1、查询每长度的员工个数



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

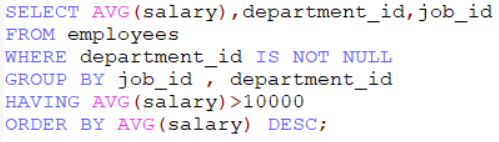
SELECT AVG(salary),department\_id,job\_id

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。



# SQL92多表查询

## 多表查询

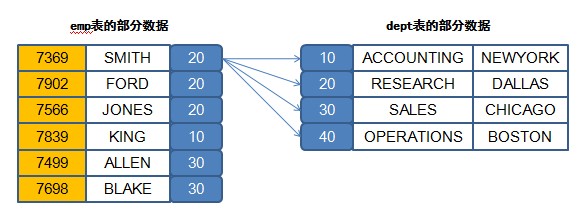
### 定义

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。

### 笛卡尔积

笛卡尔乘积现象：表1 有m行，表二有n行，结果为=m\*n行

结果列 = 列1+列2



但是现在查询之后发现一共产生了56条记录 = 雇员表的14条记录 \* 部门表的4条记录，之所以会造成这样的问题，主要都是由数据库的查询机制所决定的。

本问题在数据库的操作之中被称为笛卡尔积，就表示多张表的数据乘积的意思，但是这种查询结果肯定不是用户所希望的，那么该如何去掉笛卡尔积呢？

最简单的方式是采用关联字段的形式，emp表和dept表之间现在存在了deptno的关联字段，所以现在可以从这个字段上的判断开始

当在查询之中，不同的表中有了相同字段名称的时候，访问这些字段必须加上表名称，即“表.字段”。

SELECT COUNT(\*) FROM sales, costs

WHERE sales.prod\_id=costs.prod\_id;

这两张表即便消除了笛卡尔积的显示，但是本身也会有笛卡尔积的问题，所以最终的查询结果会很慢显示，甚至是不显示，所以通过这道程序一定要记住，多表查询的性能是很差的，当然，性能差是有一个前提的：数据量大。

引入：select \* from emp,dept；每个员工都要匹配每个部门。

select \* from emp,dept；每个部门都要匹配每个员工。

现实中，一般一个员工只能为一个部门、

发生原因：没有有效的连接条件

注：1)访问表中字段的时候使用的是“表.字段”名称，那么如果说现在假设表名称很长，所以一般在进行多表查询的时候往往都会为表起一个别名，通过别名.字段的方式进行查询。

## 分类

### 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）

### 功能分类

内连接：指连接结果仅包含符合连接条件的行，参与连接的两个表都应该符合连接条件

等值连接

非等值连接

自连接

外链接：连接结果不仅包含符合连接条件的行同时也包含自身不符合条件的行。包括左外连接、右外连接和全外连接。

左外连接

右外连接

全外连接：sql92没有该连接，sql99提供了该链接。

注：左外连接的+在连接条件的右边。

右外连接的+在连接条件的左边。

交叉连接

### 格式

SELECT [DISTINCT] \* | 字段 [别名] [,字段 [别名] ,…]

FROM 表名称 [别名], [表名称 [别名] ,…]

[WHERE 条件(S)]

[ORDER BY 排序字段 [ASC|DESC] [,排序字段 [ASC|DESC] ,…]];

## 等值连接

等值连接:将多个表中的共同字段使用=连接起来。也就是说，只有相同的两个表中相同的字段的信息才会显示出来。

### 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`;

### 案例3：查询部门10的员工的信息，包括部门名称

select emp.\*,dname

from emp,dept

where emp.deptno =dept.deptno --连接条件

and emp.deptno = 10 --筛选条件

给表起别名：

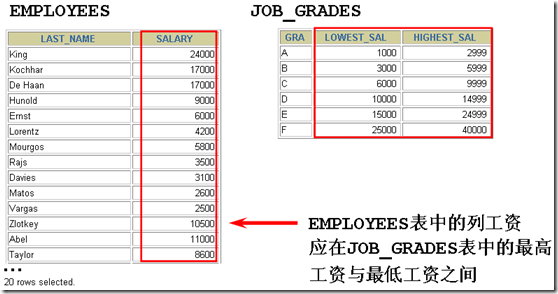
select e.\*,d.dname

from emp e ,dept d

where e.deptno = d.deptno and d.deptno = 10

## 非等值连接

非等值查询：2个表中没有相同的字段，通过范围值，确定连接条件



### 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';

### 案例2：查询部门10的员工的工资等级，员工编号，员工姓名，员工工资，员工等级

select empno,ename,sal,grade from emp e,salgrade s

where e.sal>=s.losal and e.sal<=s.hisal--连接条件

and e.deptno = 10 --筛选条件

## 自连接

作用：连接的表是同一张表，使用自连接可以将自身表的一个镜像当作另一个表来对待，从而能够得到一些特殊的数据。

### 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`;

### 案例2：

--查询每个员工的工号、姓名、经理编号

--查询每个员工的工号、姓名、经理姓名

--思路：1：每个员工的上级领导的编号实际上就是领导自身的员工编号。

2：利用员工的领导编号和员工编号做匹配。

select \* from emp e1,emp e2 //使用两个相同的表做等值连接。

select e1.empno,e1.ename,e2.ename from emp e1,emp e2

where e1.mgr = e2.empno

## 外连接

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

### 左外连接

案例1：查询员工所匹配的的信息，以及部门的所有信息。

select \* from dept d,emp e

where d.deptno = e.deptno(+) --左外连接 ，（+）在右边

### 右外连接

案例2: 查询员工的所有信息，以及部门所匹配的信息 -右外连接

select \* from dept d,emp e

where d.deptno(+) = e.deptno

## 经验

1、接触到许多新的数据库和数据表，如果此时数据量较小，则可以查询全部数据，如果数据量较大则不能直接使用SELECT查询。如果数据量大，可能造成无法查询数据，或者直接死机。

SELECT COUNT(\*) FROM 表名称;

## sql 99 语法

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

内连接：inner

外连接： 左外 ；left（outer） 右外right（outer） 全外（outer）

交叉连接cross

内连接：

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

1、案例1查询员工名、部门名

SELECT last\_name ,department\_name

FROM employees e

INNER JOIN departments d

ON e.`department\_id` = d.`department\_id`;(调换顺序不影响结果 )

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%';

**案例3查询部门个数>3的城市名和部门个数，（添加分组和筛选）**

**SELECT city,COUNT(\*) 部门个数**

**FROM departments d**

**INNER JOIN locations 1**

**ON d.'location\_id' = l.'location\_id'**

**GROUP BY city**

**HAVING COUNT(\*)>3**

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

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

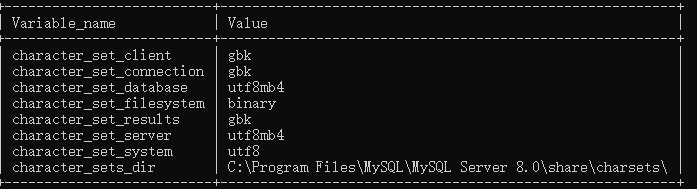
SELECT job,MAX(3al),MIN(sal),COUNT(\*)

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：

1.主键约束（唯一标识 例如身份证）

\*\*\*\*非空\*\*\*表示不能卫龙

\*\*\*\*唯一\*\*\* 表示不能相同

\*\*\*\*被引用\*\*\*\* 外键引用 不是主键不能被引用，不具有唯一

\*当表的某一列被指定为主键后，该列就不能为空，不能有重复值出现。

\*创建表时指定主键的两种方式：

CREATE TABLE stu(

sid CHAR(6） PRIMARY KEY，//在指定列后加PRIMARY KEY 该列为主键列

sname VARCHAR(20)，

age INT，

gender VARCHAR(10)

）；

例如：

Insert into stu values（1，‘hello’）；

Insert into stu values（1，‘hi‘’）；//dumlicate 在唯一的字段插入重复的值。且该值不能为null。

指定sid列为主键列，即为sid列添加主键约束

且创建的主键的值不能重复。

另一种创建的方法

CREATE TABLE stu(

sid CHAR(6），

sname VARCHAR(20)，

ageINT，

gender VARCHAR(10)，

PRIMARY KEY(sid) //主键是sid

）；

指定sid列为主键列，即为sid列添加主键约束

\*修改表时指定主键：

’ALTER TABLE stu

’ADD PRIMARY KEY(sid)；

例如：

\*删除主键：ALTER TABLE Stu DROP PRIMARY KEY； 不用指定主键名，因为只有一个主键

2.主键自增长

\*因为主键列的特性是：必须唯一、不能为空，所以我们通常会指定主键类为整型，然后设置其自动增长，这样可以保证在插入数据时主键列的唯一和非空特性。

\*创建表时指定主键自增长

CREATE TABLE stu(

sid INT PRIMARY KEY AUTO INCREMENT,

sname VARCHAR(20)，

age

INT，

gender VARCHAR(10）

\*修改表时设置主键自增长：ALTER TABLE 3tu CHANGE sid sid INT AUTO INCREMENT；

\*修改表时删除主键自增长：ALTER TABLE Stu CHANGE sid 3id INT；

测试主键自增长：

>INSERT INTO Stu VALUES(NULL,'zhangSan',23,'male')；

>INSERT INTO stu(sname,age,gender)VALUES('zhangSan',23,'male');

3.非空约束

\*因为某些列不能设置为NULl值，所以可以对列添加非空约束。

\*例如：

CREATE TABLE stu（

sid INT PRIMARY EY AUTO INCREMENT，

sname VARCHAR（20）NOT NULL，//加个not null

ageINT，gender VARCHAR（10）

）；

\*对sname列设置了非空约束

4.唯一约束

\*数据库某些列不能设置重复的值，所以可以对列添加唯一约束。

\*例如：

CREATE TABLE stu（

sid INT PRIMARY KEY AUTO INCREMENT，

Bnamel VARCHAR（20）NOT NULL UNIQUE，非空和唯一同时加到统一字段

ageINT，gender VARCHAR（10）

）；

\*对name列设置了非空约束

5、概念模型

Is-a

has a（关联）

>1对1

>1对多

>多对多

use a

当我们要完成一个软件系统时，需要把系统中的实体抽取出来，形成概念模型。

例如部门、员工都是系统中的实体

。概念模型中的实体最终会成为Java中的类、数据库中表。

实体之间还存在着关系，关系有三种：

\*1对多：例如每个员工都从属一个部门，而一个部门可以有多个员工，其中员工是多方，而部门是一方。\*1对1：例如老公和老婆就是一对一的关系，一个老公只能有一个老婆，而一个老婆只能有一个老公。

\*多对多：老师与学生的关系就是多对多，一个老师可以有多个学生，一个学生可以有多个老师。

概念模型在Java中成为实体类（javaBean）

类就使用成员变量来完成关系，一般都是双向关联！多对一双问中关联，即员工关联部门，部门也关联员工

class Employee{//多方关联一方

private Department department；

}

class Department{//一方关联多方

private List<Employee>employeeslI

class Husband{//d一对一关系

private Wife wife；

}

class Wife{//一对一关系

private Husband class Student

}

Class Student{

private List<Teacher> teachers

}

class Teacher{//多对多关系

private List<Student>students；

}

6.外键约束

.外键必须是另一表的主键的值（外键要引用主键！）

\*外键可以重复

\*外键可以为空

CREATE TABLE emp(

empno INT PRIVMARY KEY AUTO\_INCREMENT,

ename VARCHAR(50),

dno INT,

CONSTRAINT fk\_emp\_dept FOREIGN KEY(dno)REFERENCES dept(deptno)

7、建立关系模型中的三种关系

1-1

主表：

CREATE TABLE hushand(

hid INT PRIMARY KEY AUTO\_INCREMENT,

hname VARCHAR(50)

);

INSERT INTO hasband VALUES（NULL，'刘备’）；

INSERT INTO hasband VALUES（NULL，'关羽’）；

INSERT INTO hasband VALUES（NULL，'张飞'）；

从表：

CREATE TABLE wife(

Wif INT PRIMARY KEY AUTO\_INCREMENT,

wname VARCHAR(50),

CONSTRAINT fk\_wife hasband FOREIGN KEY(wif)REFERENCES hasband (hid)

);

wid：非空，唯一，引用hid

INSERT INTO wife VALUES（1，'杨贵妃’）；

INSERT INTO wife VALUES（2，'妲己’）；

从表的主键即是外键

多对多

CREATE TABLE student

（ sid INT PRIMARY KEY AUTO\_INCREMENT,主键

sname VARCHAR(50)

)

CREATE TABLE teacher(

tid INT PRIMARY KEY AUTO\_INCREMENT,主键

tname VARCHAR(50)

)

CREATE TABLE stu\_tea(

sid INT,

tid INT,

CONSTRAINT fk student FOREIGN KEY(sid)REFERENCES atudent(收、id),

CONSTRAINT fk teacher FOREIGN KEY(tid) REFERENCES teacher(tid)

);

INSERT INTO student VALUES（NULL，'刘德华）；

INSERT INTO student VALUES（NULL，'梁期伟’）；

INSERT INTO student VALUES（NULL，黄日华）；

INSERT INTO student VALUES（NULL，'首侨伟’）；

INSERT INTO student VALUES（NULL，'汤朕业）；

INSERT INTO teacher VALUES（NULL，'崔老师）；

INSERT INTO teacher VALUES（NULL，'刘老师’）；

INSERT INTO teacher VALUES（NULL，'石老师1）；

INSERT INTO stu\_ tea VALUES(1,2);

INSERT INTO stu\_tea VALUES(2,2);

INSERT INTO stu\_tea VALUES(3,2);

## 多表查询

### 合并结果集（合并两张表格）

\*要求被合并的表中，列的类型和列数相同（实质为查询的两个结果集相同）

\*UNION，去除重复行

UNION ALL 保留重复行

SELECT \*FROM ab

UNION ALL

SELECT \*FROM cd;

SELECT \*FROM ab

UNION //去掉all后，完全相同的结果集会被去掉

SELECT \*FROM cd;

\*UNIONALL，不去除重复行

### 连接查询：一次性查询多个表

子查询：查询里面包查询

# SQL99的多表查询

## cross join

作用：产生笛卡尔积，笛卡尔积本身并不是属于无用的内容，在某些情况下还是需要使用的。

select \* from dept,emp--sql92 笛卡尔积

select \* from dept cross join emp --sql99 交叉连接

## Natural join

作用：自动找到匹配的关联字段，消除掉笛卡尔积。

代码：select \* from emp natural join dept//会自动去搜索相同的列。

注：1) 受限的等值连接，2个表中必要要有同名列，否则结果就变为交叉连接。

2)两个表中相同列名的字段类型要相同。

3) 自然连接的结果不保留重复的属性。也就是系统只会保留一列同名列。

4)不能够在同名列的前面加表的别名。因为只会保留一列同名列，只用当前的字段名就可以了，如果使用表名.字段名，会报错。该列不属于任何表。

## Using 字句

如果不希望参照被连接表的所有同名列进行等值连接，自然连接将无法满足要求，可以在连接时使用USING子句来设置用于等值连接的列（参照列）名。使用using子句可以在有多个列满足条件的情况下进行筛选。

代码：select \* from emp join dept

using(deptno) --using 子句可以指定同名列做连接条件

## on字句

作用：自然连接的条件是基于表中所有同名列的等值连接，为了设置任意的连接条件或者指定连接的列，需要使用ON子句，连接条件与其它的查询条件分开书写。

### 案例1：查询部门编号是20的员工编号、姓名，薪资，部门名称

select empno,ename,sal,dname

from emp e,dept d

where e.deptno = d.deptno --连接条件

and d.deptno = 20 --筛选条件 （过滤条件） sql92

### 案例2：（99）

select empno,ename,sal,dname

from emp e join dept d

on e.deptno = d.deptno --连接条件写在on后面

where d.deptno = 20 --筛选条件写在where后面 sql99

### 案例3查询"01"课程比"02"课程成绩高的学生的信息及课程分数

select s.\*,sc1.score,sc2.score

from student s join score sc1 on s.stuid = sc1.stuid

join score sc2 on sc1.stuid = sc2.stuid

where (sc1.cid ='c1001'and sc2.cid = 'c1002') and sc1.score>sc2.score

## 外连接（outer）

### 左外连接（left outer join）

两个表在连接过程中除返回满足连接条件的行以外，还返回左表中不满足条件的行，这种连接称为左外连接。

select \* from emp e,dept d

where e.deptno = d.deptno(+) --sql92

select \* from emp e left join dept d

on e.deptno = d.deptno --sql99

### 右外连接（right outer join）

两个表在连接过程中除返回满足连接条件的行以外，还返回右表中不满足条件的行，这种连接称为右外联接。

select \* from emp e right join dept d

on e.deptno = d.deptno

### 全外连接

两个表在连接过程中除返回满足连接条件的行以外，还返回两个表中不满足条件的所有行，这种连接称为满外联接。

select \* from emp e full join dept d

on e.deptno = d.deptno

# 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=,<,>,>=,<=,<>

### 子查询出现的位置

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：★

标量子查询（单行） √

列子查询 （多行） √

行子查询

exists后面（相关子查询）

表子查询

## 单行子查询

如果子查询返回一个标量值（就一个值），那么外部查询就可以使用：=、>、<、>=、<=和<>符号进行比较判断；

案例1：查询工资最高的雇员名字和工资。（工资最高的雇员名字是未知的。）

select \* from emp where sal = ( select max(sal) from emp)

案例2：查询职务和SCOTT相同，比SCOTT雇佣时间早的雇员信息

SELECT \* from emp where job = (select job from emp where ename = 'SCOTT')

and hiredate<(select hiredate from emp where ename = 'SCOTT' )

案例3：查询工资比SCOTT高或者雇佣时间比SCOTT早的雇员的编号和名字

select \* from emp

where sal>(select sal from emp where ename = 'SCOTT' )

or hiredate<(select hiredate from emp where ename = 'SCOTT' )

## 多行子查询

如果子查询返回的不是一个标量值，而外部查询使用了比较符和子查询的结果进行了比较，那么就会抛出异常。子查询的结果有多行行记录 不可以使用=,<,>,>=,<=,<>。

-- ALL 和子查询返回的所有值比较

-- ANY 和子查询返回的任意一个值比较

-- IN 等于列表中的任何一个

### 案例：查询工资低于任意一个“CLERK”的工资的雇员信息。（any）

select \* from emp where sal< (select max(sal) from emp where job = 'CLERK') order by sal; --单行子查询

select \* from emp where sal<any(select sal from emp where job = 'CLERK') and job <>'CLERK' order by

### 案例：查询工资比所有的“SALESMAN”都高的雇员的编号、名字和工资。

select empno ,ename,sal

from emp

where sal> all(select sal from emp where job = 'SALESMAN')

### 案例：查询部门20中职务同部门10的雇员一样的雇员信息。

select \* from emp where job in( select job from emp where deptno = 10)

and deptno = 20

### 案例:查询在雇员中有哪些人是经理人

--思路：

--1：先查询表中所有的mgr的编号

--2：根据mgr和员工编号做匹配

select \* from emp where empno in (select distinct mgr from emp )

### 案例找出部门编号为20的所有员工中收入最高的职员

select \* from emp where sal>=all( select sal from emp where deptno = 20)

and deptno = 20

### 案例：查询每个部门的薪水等级（子查询可以出现在where和form以及having子句中）。

--1 首先将每个部门平均薪水求出来，然后把结果当成一张表。

--2 用这张结果表和salgrade表做连接，以此求得薪水等级。

select deptno,s.hisal,s.losal,grade from salgrade s,(select deptno,avg(sal) avg\_sal

from emp group by deptno) t

where t.avg\_sal>=s.losal and t.avg\_sal<=s.hisal

group by deptno,s.hisal,s.losal,grade

having grade > (select grade from salgrade where grade = 3)

### 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)=(  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

|  |
| --- |
| #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  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：

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`  );  #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`  ); |

# 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

--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

一种限制，用于限制表中的数据，为了保证表中的数据的准确和可靠性

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。

4、约束的分类：

列级约束：六大约束语法上都支持，但外键约束没有效果。

表级约束：除了非空、默认，其他的都支持。

## 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(mysql 中不支持，可以写，但是不起作用)

--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

用于限制两个表的关系，用于保证该字段的值必须来自于主表的关联列的值在从表添加外键约束，用于引用主表中某列的值。列类型也必须一致。

例如：学生编号、工种编号。

### DEFAULT:默认，用于保证该字段有默认值

比如：性别

### 添加约束的时机

1.创建表时

2.修改表时

## 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。

删除规则：

1.删除主表的记录之前，必须先删除子表所对应的

2.

3.

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。

);

### 级联删除

3.删除父表中的元素，首先要删除字表中的记录。否则出现，已找到子录。

级联删除：删除主表的同时删除子表的记录。

## 序列（sequence）

序列：序列是oracle专有的对象，他用来产生一个自动递增的数列。

1、序列的用法

--1：序列的初始值

--2：递增的值

--3：最大值

--4：是否循环

-给表中添加数据的时候使用

2、序列的2个关键属性

-nextval

--crlvaI

3、序列一般添加到逐渐上防止主键重复。

4、序列值一旦删除，无法恢复。删除1006 之后，下次添加会添加1007，无法返回到1007。

删除规则：

1：删除主表的记录之前，必须先删除子表所对应的记录 先删子表，后删主表 默认 RESTRICT

2：删除主表的同时，把子表的记录级联删除 on delete casecade

3：删除主表的同时，子表所对应的外键的值设为null on delete set null

### 创建序列

create sequence deq\_1

stat with 1000起始值

minvalue最小值为1

increment by 10 //递增指

### 查询序列的属性

第一次创建序列后，必须光调用nextva1，使序列有值，再调用currval

select seq\_1. nextval from dual;

select seq\_1.currval from dual;

### 使用序列

给表中添加数据的时候使用。

create table A{

a number(5) primary key,

b varchar2(20)

}

插入值：

inset into A values(seq\_1.nextval,’张三’);//1001

inset into A values(seq\_1.nextval,’张三’);//1002

inset into A values(seq\_1.nextval,’张三’);

inset into A values(seq\_1.nextval,’张三’);

### 删除序列

drop sequence seq\_1

# 索引（index）

索引：索引类似字典的和课本目录，是为了加快对数据的搜索速度尔设立的。索引有自己专门的存储空间，与表独立存放。

作用：提高查询效率。

缺点：索引会占用内存空间，特别是增删该查频繁的表不建议建立过多的索引。

--1索引会占用内存空间

-2素引政善检素操作的性能，但降低数帮插入、修改和删除的性能。

-3在执行这些操作时，DBIS必须动态地更新索引。

-4限制表中素引的数日。索引越多，在修改表时对素引做出修改的工作室越大

## rowid：标记数据的物理地址

可以通过rowid 直接查询数据

select \* from score where rowed = ‘数据的物理地址’。可以查询伪列(rowid),速度最快

但是不能修改rowid的值

supdate score set rowed = ‘数据的物理地址’ where rowed = ‘数据的物理地址’。

## 创建索引

### 自动创建

当在表上定义一个PRTMARY KEY 或者 UNIQLE 约束条件时Oracle数据库自动创建一个对应的 唯一索引。

### 手动：用户可以创建索引以加速查询:

create index index1 on student(sname)//给学生表中的姓名字段添加索引。

## 删除索引

drop index index1

# 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、

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。

## 创建视图

注：创建视图之前要添加权限

### 创建默认视图

create view view//可修改的视图

as

select \* from emp

### 创建只读视图

create view view1

as

select \* from emp

with read only

### 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

# 事务(transaction)

## 事务

事务(Transaction)是一个操作序列。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。

作用：保证数据库的完整性。

注：

## 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

### 幻想读

### 不可重复读取

### 脏读

# 其他内容

## 伪劣

### rowid

rowid在记录创建时生成，而且是不变，直接指向硬件上的存储位置

用rowid直接访问是最快的，但也是人力所无法做到的

只要记录没被搬动过，rowid是不变的

### rownum 行号

1.rownum是个伪列，查询的时候除非特别指定，否则不会显示。

select rownum from emp

2.主要的作用是控制查询返回的行数

3.只能使用：<,<=(不能是用有>号的，因为,rownum显示只能从1开始显示。)

Select rowid,rownum,e.\* from emp where rownum<=5//显示5行内容

案例：获取前五位的工资

select rownum，emp.\* from emp where rownum<=5 order by sal

上述的错误，先取前五行，后进行的排序。

思路：1.先对整个员工表按照工资排序

2.将排序后的结果作为一个新的表。

1、获取首页信息可以使用2层查询，获取前五条信息。

select rownum,t.\*

from (select rownum ,emp.\* from emp order by sal) t

where rownum<=5;

2、获取非首页信息 获取6-10 ，需要三层嵌套。

思路--2.1 先对整个员工表按照工资排序

--2.2 将排序后的结果作为一个新的表

--2.3 在按照rownum获取前10行

--2.4 将前10条记录再当做一个子表

--2.5 再最外层使用rownum的别名做>=操作

select rownum,t.\*

from (select rownum ,emp.\* from emp order by sal) t

where rownum<=10 and rownum>=6; rownum不能有>

select\*from(

select rownum r,t.\*from(

select rownum,emp.\* from emp order by sal)t where rownum<=10)

where r>5 //rownum 的别名可以使用>=操作

当rownum和order by一起使用时，会首先选出符合rownum条件的记录，然后再进行排序，这会给我们的查询带来难度。

作用：分页查询

示例

# 数据库的范式

## 范式(NF,NormalForm)

范式：范式是符合某一种设计要求的总结。要想设计一个结构合理的关系型数据库，必须满足一定的范式。

### 设计合理的数据库

1.结构合理

2.冗余较小

3.尽量避免插入删除修改异常

### 范式分类

第一范式：简单说就是要确保每列保持原子性，一列数据，不能写多个值。

第二范式：确保数据库表中的每一列都和主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）。即在一个数据库表中只能保存一种数据，不可以把多种数据保存在同一张数据库表中。

主要是解决多对多的关系

多对多实际是有2个一对多构成的

学生和课程M-n

学生表，课程表，选课表

学生一课表1-n一个学生可以学习多门课程 外键关联

课程一选表1-n一门旅程可以被多个学生学习 外键关联

学生一选课一课程mn多个学生可以学习多门课程

第三范式: 确保数据表中的每一列数据都和主键直接相关，而不能间接相关

属性不依赖于其他非主属性。

主要解决1-n的关系，靠外键建立关系。

班级和学生1-n，一个班级可以容纳多个学生。

第四范式

第五范式

## 第一范式

# PL/SQL中的一些功能

## 修改PL/SQL中字段的顺序

### 法一