

# Oracle数据库系统

深圳市门道信息咨询有限公司  
Shenzhen MT Information Consulting Co., LTD  
版权所有. 侵权必究

Chapter 1 数据库简介和Oracle入门

Chapter 2 如何登录Oracle

Chapter 3 pl/sql developer工具简介

Chapter 4 查询

Chapter 5 DML

Chapter 6 日期和函数

Chapter 7 伪列

Chapter 8 Oracle体系和其他对象

- 1.1 什么是数据库
- 1.2 数据库的分类
- 1.3 测试学习数据库的意义
- 1.4 关系型数据库的特点
- 1.5 Oracle名字的来源
- 1.6 Oracle的各版本
- 1.7 Oracle服务及卸载

# 1.1 什么是数据库

## ■ 什么是数据库

- 数据库（Database）是按照数据结构来组织、存储和管理数据的仓库。



仓库(货架)



数据库(表)

## 1.2 数据库的分类

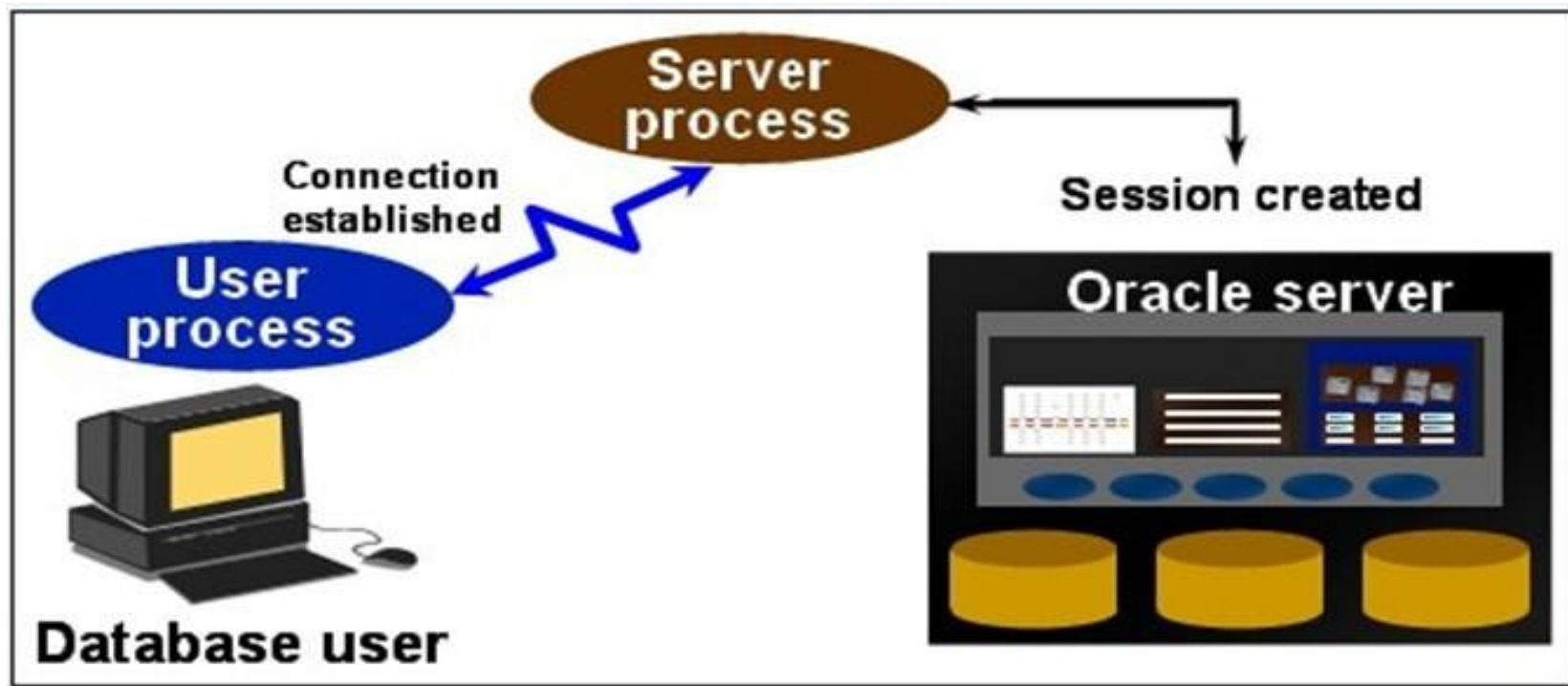
### ■ 数据库分类

- 小型数据库：Access
- 中型数据库：MySQL SQLServer
- 大型数据库：Oracle DB2 Sysbase



## 1.3 测试学习数据库的意义

- 我们在现实使用的系统，应用程序需要数据做依托，因此就需要数据库提供这样的服务。作为测试工程师，我们最主要的任务是（查）数据。





## 1.4 关系型数据库的特点

- 关系型数据库特点：
  - 实体、属性、关系



## 1.5 Oracle名字的来源

- Oracle原本的含义：“神喻”--指的是神说的话。
- 中国在商朝的时代，把一些刻在龟壳上的文字当成是上天的指示，所以在中国将Oracle也翻译成“甲骨文”。



## 1.6 Oracle的各版本

### ■ Oracle 9i

i代表internet，这一版本中添加了大量为支持internet而设计的特性。

### ■ Oracle 10g

g代表grid（网格技术）。这一版的最大的特性就是加入了网格计算的功能。

### ■ Oracle 11g

### ■ Oracle 12c

c表示Cloud（云技术）。

## 1.7 Oracle服务

- OracleDBConsoleorcl --可以不启动,用于管理Oracle的企业管理器的服务。
- OracleJobSchedulerORCL --可以不启动,用于定期操作任务的服务。
- OracleOraDb10g\_home2iSQL\*Plus --可以不启动,这是isqlplus服务,用于用网页执行sql的服务。
- OracleOraDb10g\_home2TNSListener **--必须启动,这是Oracle的监听服务。**
- OracleServiceORCL **--必须启动,这是Oracle的主服务。**

## 1.7 Oracle卸载

### ■ 如何手动完全删除Oracle(五步走)

1. 右击我的电脑→管理→服务停掉Oracle所有的服务;
2. 找到Oracle的安装路径并删除Oracle的安装文件;
3. 删除开始菜单中Oracle的所有目录;
4. 删除注册表中的Oracle项(有4个地方需要删除):
  - ① 在运行中输入regedit进入注册表
  - ② 在HKEY\_CLASSES\_ROOT中删除以Oracle开头的注册项
  - ③ 在HKEY\_CURRENT\_USER中的Software中删除以Oracle开头的注册项
  - ④ 在HKEY\_LOCAL\_MACHINE中的SOFTWARE中删除以Oracle开头的注册项
    - a. 在HKEY\_LOCAL\_MACHINE中的SYSTEM → ControlSet001 → services中删除以Oracle开头的注册项
    - b. 在HKEY\_LOCAL\_MACHINE中的SYSTEM → ControlSet002 → services中删除以Oracle开头的注册项
    - c. 在HKEY\_LOCAL\_MACHINE中的SYSTEM → CurrentControlSet → services中删除以Oracle开头的注册项
5. 重启电脑重新安装Oracle。

## 2.1 如何登录Oracle

## 2.1 如何登录Oracle

- 1、在开始菜单的Oracle目录里面的"应用程序开发"的sqlplus中登录。



```
Oracle SQL*Plus
文件(F) 编辑(E) 搜索(S) 选项(O) 帮助(H)

SQL*Plus: Release 10.2.0.1.0 - Production on 星期四 8月 16 16:57:40 2018

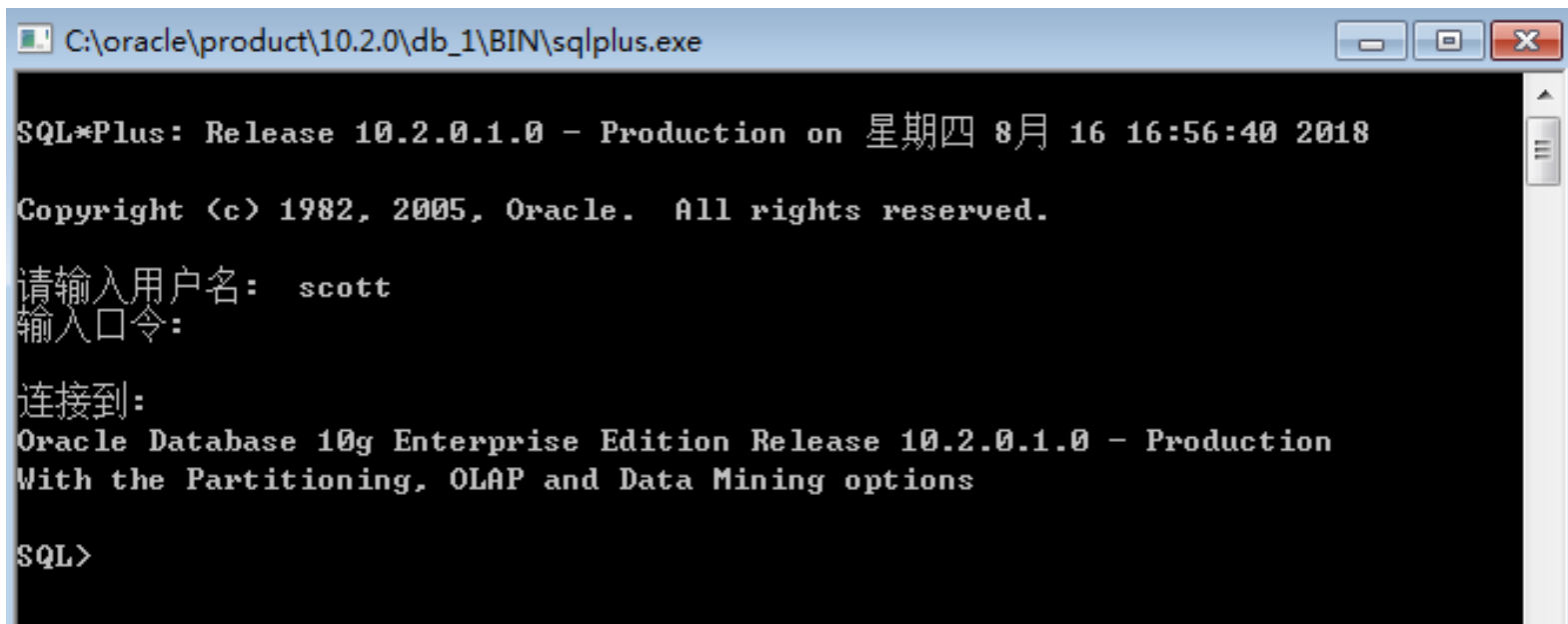
Copyright (c) 1982, 2005, Oracle. All rights reserved.

连接到:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> |
```

## 2.1 如何登录Oracle

2、在运行中输入cmd回车,然后输入sqlplus,调用sqlplus工具登录。



```
C:\oracle\product\10.2.0\db_1\BIN\sqlplus.exe

SQL*Plus: Release 10.2.0.1.0 - Production on 星期四 8月 16 16:56:40 2018

Copyright (c) 1982, 2005, Oracle. All rights reserved.

请输入用户名:  scott
输入口令:

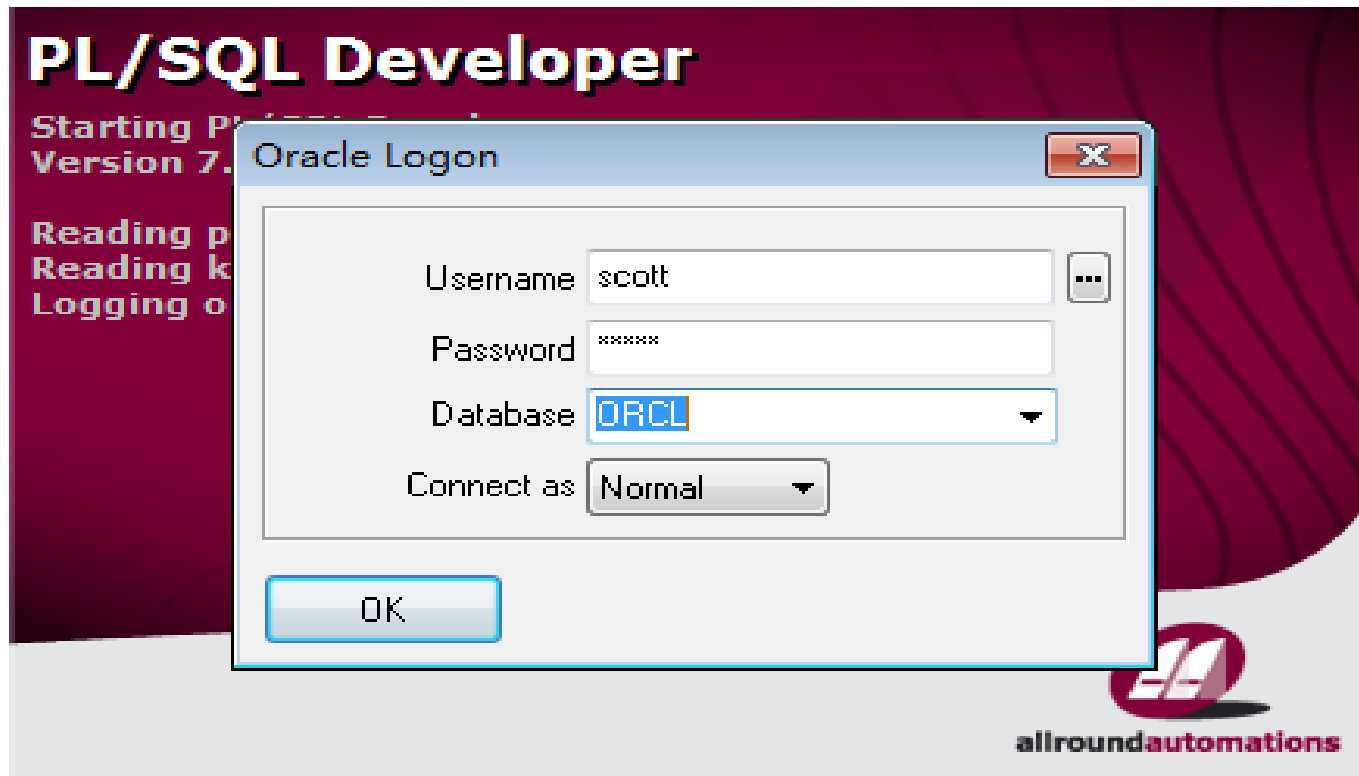
连接到:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL>
```



## 2.1 如何登录Oracle

3、使用可视化工具pl/sql developer登录。



## 3.1 PL/SQL简介

## 3.2 PL/SQL使用技巧

## 3.1 PL/SQL简介

- PL/SQL Developer是一个集成开发环境，专门面向Oracle数据库存储程序单元的工具。
- PL/SQL Developer侧重于易用性、代码品质和生产力，充分发挥Oracle应用程序开发过程中的主要优势。
- 简单理解： PL/SQL Developer是一个**图形界面工具**

## 3.2 PL/SQL使用技巧

### 1、标题栏

确定你是哪个用户登录的且你的身份是什么都显示出来。

### 2、小钥匙

灵活的切换用户。

### 3、all objects -> my objects

查看我当前登录的用户下有哪些属于“我的”对象。

### 4、小齿轮或按F8

执行SQL语句。

### 5、颜色和字体

tools -> Preferences -> Fonts。

### 6、tools -> windows list

方便多窗口之间进行切换。

### 7、全屏习惯

写代码方便，好看，符合人的阅读习惯。

### 8、窗口4/6开

4.1 基本查询

4.2 条件查询

4.3 分组查询

4.4 排序查询

4.5 子查询&多表查询

## 4.1 基本查询

### ■ 语法

```
select 字段 from 表;
```

### ■ 查询单个字段

--查询员工姓名

```
select ename from emp;
```

### ■ 查询多个字段

--查询员工姓名和薪水

```
select ename,sal from emp;
```

### ■ 查询全部字段

--查询员工的所有信息

```
select * from emp;
```



## 4.1 基本查询

### ■ 列上做运算和连接

运算:

--查询员工姓名和年薪

```
select ename, (sal+comm)*12 from emp;--结果不准确
```

```
select ename, (sal+nvl(comm,0))*12 from emp;--正确
```

注:nvl(字段,默认值)

连接:

--查询员工的姓名(姓名前拼接公司名)和薪水

```
select '门道信息_' || ename, sal from emp;
```

## 4.1 基本查询

### ■ 列别名和表别名

列别名：

--查询员工的姓名和年薪

```
select e.ename , (sal+nvl(comm,0))*12 as 年薪 from emp e;
```

--可以不写as, 同时别名尽量不用中文

```
select e.ename , (sal+nvl(comm,0))*12 year_sal from emp e;
```

--推荐写法, 别名做到“见名知意”

表别名：

--查询员工姓名和入职日期

```
select e.ename,e.hiredate from emp e;
```

## 4.1 SQL语句书写规范

注意：

- ① SQL 语言大小写不敏感。
- ② SQL 可以写在一行或者多行。
- ③ 各子句一般要分行写。
- ④ 使用缩进提高语句的可读性。

## 4.2.1 条件查询(WHERE)-概述

- 使用WHERE子句将不满足条件的行过滤掉，WHERE子句紧随FROM子句，语法格式如下：

```
select 字段 from 表 where 条件
```

- 字符大小写敏感

```
--查询职位为'CLERK'的员工的所有信息
```

```
select * from emp where job='CLERK' ;--查询正确
```

```
Select * from emp where job='clerk' ;--查询不出
```

- 数字直接写，字符串必须包含在单引号中，日期需要日期函数处理

```
--查询员工编号为7788的员工的所有信息
```

```
select * from emp where empno=7788;
```

```
--查询职位为'CLERK'的员工的所有信息
```

```
select * from emp where job='CLERK' ;
```

```
--查询入职日期为1987-08-20的员工的所有信息
```

```
select * from emp where hiredate=to_date('1987-08-20', 'yyyy-mm-dd') ;--默认时间格式为:yyyy-mm-dd hh24:mi:ss
```

## 4.2.2 关系运算符

操作符	含义
>	大于
<	小于
=	等于
>=	大于或者等于
<=	小于或者等于
!=	不等于
<>	不等于

## 4.2.3 条件查询(LIKE)

- 查询名字中包含字符'O'的员工信息

```
select * from emp where ename like '%O%';
```

- 查询名字中第三个字符为'O'的员工信息

```
select * from emp where ename like '%__O%';
```

注意:

- ① %: 匹配0-N个字符
- ② \_: 匹配1个字符



## 4.2.4 条件查询 (BETWEEN...AND)

- 查询工资在3000-5000之间的员工信息

```
select * from emp where sal between 3000 and 5000
```

注意：

**BETWEEN...AND条件查询包含边界值。**

例：上面的查询工资等于3000或者5000的员工信息也是符合条件的。

## 4.2.5 条件查询(IN)

- 查询empno在7369或者7499中的任意一个的员工信息

```
select * from emp where empno in (7369,7499)
```

注意：

IN查询表示只要是多个值中的任意一个就满足条件。

## 4.2.6 条件查询(NULL)

### ■ 查询没有奖金的员工信息

```
select * from emp where comm is not null
```

注意：

NULL只能通过is not或者not (is) 来判断，不能使用关系运算符来判断。

## 4.2.7 逻辑运算符

### ■ 逻辑运算符

操作符	含义
AND	并且(逻辑与)
OR	或者 (逻辑或)
NOT	非 (逻辑否)

注意：

- ① AND：只有AND的两边都成立才满足条件
- ② OR：只要OR的两边有一边成立就满足条件
- ③ NOT：表示与原来的条件相反

## 4.2.8 逻辑运算符的用法

### ■AND

```
select studentid,sname,sex,classid  
from student  
where classid = 1 AND sname LIKE '%O%';
```

### ■OR

```
select studentid,sname,sex,classid  
from student  
where classid = 2 OR sname LIKE '%O%';
```

### ■NOT

```
select studentid,sname,sex,classid  
from student  
where classid NOT IN (1,2);
```

## 4.3 分组查询

- 分组函数
- GROUP BY手动分组
- DISTINCT应用于分组函数
- NVL函数应用于分组函数
- 分组后过滤HAVING



## 4.3.1 分组函数

### ■ 什么是分组函数？

- 分组函数作用于一组数据，返回一个结果。

例：以下是一个公司所有员工的信息，SAL为工资

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7934	MILLER	CLERK	7782	1982/1/23	1300.00		10
7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
7839	KING	PRESIDENT		1981/11/17	5000.00		10
7369	SMITH	CLERK	7902	1980/12/17	800.00		20
7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20
7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
7902	FORD	ANALYST	7566	1981/12/3	3000.00		20
7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20

查询员工的最高工资：5000

## 4.3.2 分组函数的使用

- AVG--平均值、SUM--和、MAX--最大值、MIN--最小值、COUNT--个数

```
select avg(sal) from emp;  
select sum(comm) from emp;  
select max(hiredate) from emp;  
select min(ename) from emp;  
select count(*) from emp;
```

- AVG和SUM函数只能作用于number类型的字段
- MAX和MIN函数可以作用于任何类型
- 当字段的值为NULL时不会被COUNT函数统计
- 分组函数单独使用的时候可以任意组合

```
select avg(ename) from emp;  
select count(comm) from emp;  
select avg(sal),sum(sal),max(sal),min(sal),count(comm)  
from emp;
```

## 4.3.3 GROUP BY

### ■ 什么是GROUP BY手动分组

– **GROUP BY**作用于一组数据，返回0-N个结果。

例：以下是一个公司所有员工的信息，  
SAL为工资，deptno为部门。

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7934	MILLER	CLERK	7782	1982/1/23	1300.00		10
7782	CLARK	MANAGER	7839	1981/6/9	2450.00		10
7839	KING	PRESIDENT		1981/11/17	5000.00		10
7369	SMITH	CLERK	7902	1980/12/17	800.00		20
7876	ADAMS	CLERK	7788	1987/5/23	1100.00		20
7566	JONES	MANAGER	7839	1981/4/2	2975.00		20
7902	FORD	ANALYST	7566	1981/12/3	3000.00		20
7788	SCOTT	ANALYST	7566	1987/4/19	3000.00		20

查询每个部门的最高工资：比如10这个部门的最高工资为5000，20这个部门的最高工资为3000。

## 4.3.4 GROUP BY的使用

### ■ GROUP BY单独使用

--查询公司有哪些岗位

```
select job from emp group by job;
```

### ■ GROUP BY和分组函数一起使用

--查询公司每个部门的平均工资

```
select deptno,avg(sal) from emp group by deptno;
```

### ■ 根据多个字段分组

--查询每个部门每个岗位的最高工资

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

注意:

- ① 在SELECT列表中所有未包含在组函数中的列都应该包含在GROUP BY子句中。
- ② 包含在GROUP BY子句中的列不必包含在SELECT列表中。

## 4.3.5 DISTINCT关键字

### ■ DISTINCT单独使用表示去重

--查询公司有哪些岗位

```
select job from emp group by job;
```

### ■ DISTINCT应用于分组函数

--查询公司的岗位个数

```
select count(distinct job) from emp;
```

## 4.3.6 HAVING分组后过滤

### ■ HAVING分组以后过滤

--查询部门工资>2000的人数超过2个的部门人数

```
select deptno,count(*)
```

```
from emp
```

```
where sal>2000      --分组前过滤
```

```
group by deptno
```

```
having count(*)>=2    --分组后过滤
```

--查询部门工资>2000的人数超过2个的部门人数

```
select sid,sname,count(cid),max(score)
```

```
from student
```

```
where score>=60 group by sid,sname
```

```
having count(cid)>=2
```

注意:

- ① WHERE是分组前过滤，HAVING是分组后过滤。
- ② WHERE中不能使用分组函数，HAVING可以使用分组函数。

### ■ HAVING分组以后过滤

```
select ename,avg(sal) from emp;  
  
select deptno,count(*)  
from emp  
where sal>2000 and count(*)>=2  
group by deptno;
```

注意：

- ① 在SELECT 列表中所有未包含在组函数中的列都应该包含在 GROUP BY 子句中。
- ② WHERE中不能使用分组函数，HAVING可以使用分组函数。

## 4.4 排序查询

■ 逻辑数据都已写好，按照指定规则挪动数据就称为排序

```
select * from emp order by sal asc;  
select * from emp order by ename desc;  
select ename,sal*12 nianxin from emp order by nianxin asc;  
select ename,sal*12 from emp order by 2 asc;  
select * from emp order by deptno desc,sal  
asc,comm,empno,ename desc;
```

注意：

- ① **ORDER BY**子句永远在语句的最后。
- ② **ASC**是升序，**DESC**是降序，默认升序，可以不写。
- ③ 多字段排序时，先按照第一个规则排，排完后如果还有相同，按照第二个规则排，如果还有相同数据，继续按照第三个规则排，以此类推。



## 4.5 子查询&多表查询

4.5.1 使用子查询解决问题

4.5.2 子查询的特点

4.5.3 单行子查询

4.5.4 单行子查询示例

4.5.5 多行子查询

4.5.6 多行子查询示例

4.5.7 多表查询（内连接）

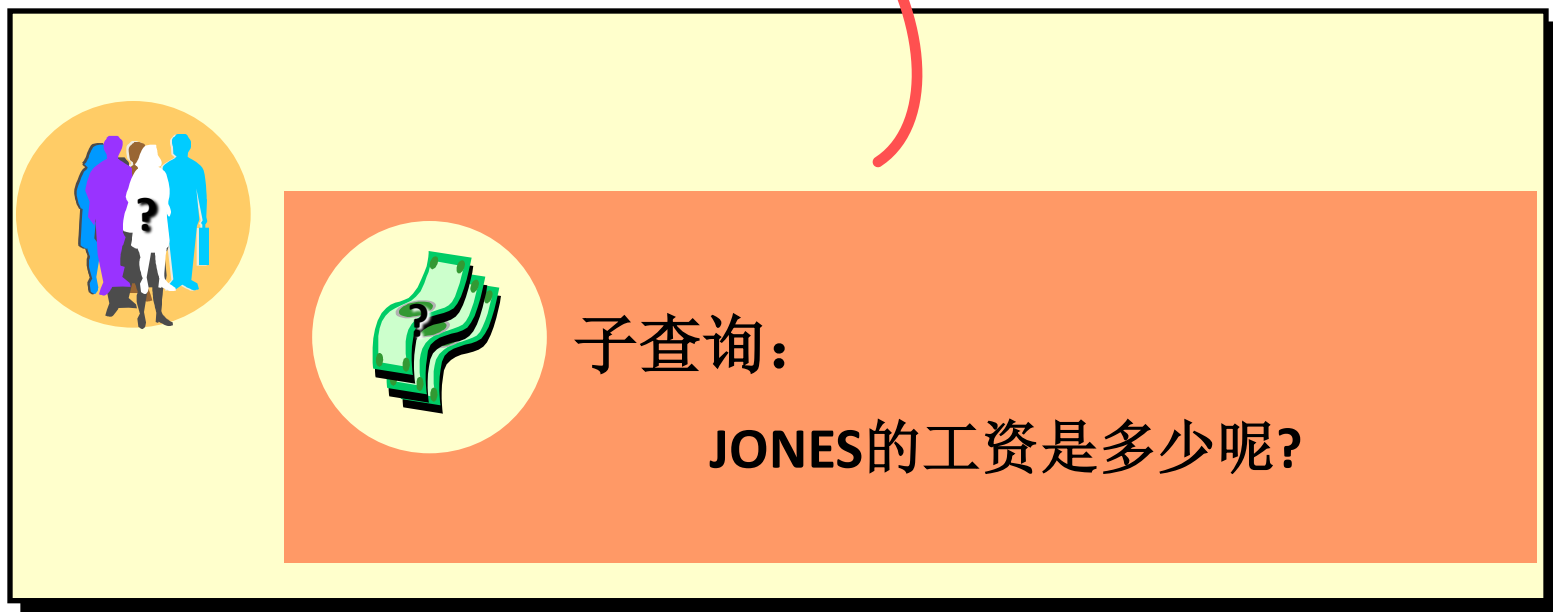
4.5.8 多表查询（左/右连接）

4.5.9 多表查询（其他连接）

## 4.5.1 使用子查询解决问题

- 查询比JONES的工资还要高的员工信息

主查询：谁的工资比JONES高？



注意：

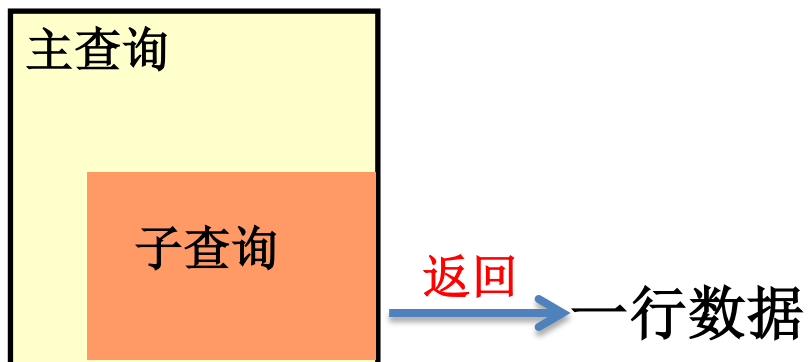
- ① 子查询 (内查询) 在主查询之前一次执行完成。
- ② 子查询的结果被主查询使用 (外查询)。

## 4.5.2 子查询的特点

- 子查询要包含在括号内
- 将子查询放在比较条件的右侧
- 子查询分为单行子查询和多行子查询
- 子查询一般情况下应用于where子句或者是from子句

## 4.5.3 单行子查询

### ■ 单行子查询



单行比较运算符	含义
=	等于
>	大于
>=	大于或等于
<	小于
<=	小于或等于
<>	不等于

注意：

- ① 只返回一行
- ② 使用单行比较操作符

## 4.5.4 单行子查询示例

### ■ 单行子查询示例

--查询比JONES的工资还要高的员工信息

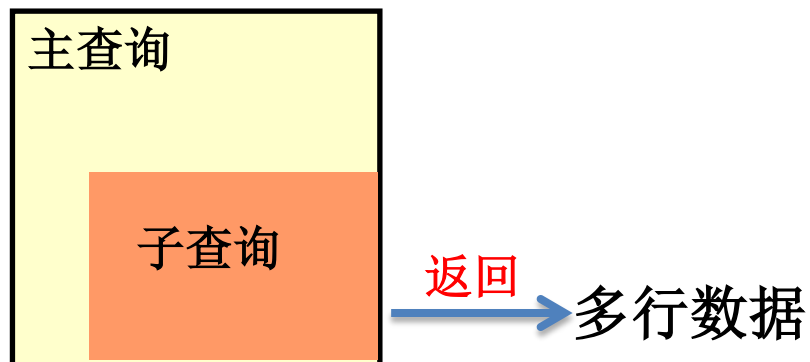
```
select * from emp where sal > (select sal from emp where  
ename = 'JONES');
```

--查询和SMITH在同一个部门的员工信息

```
select * from emp where deptno = (select deptno from emp  
where ename = 'SMITH') and ename <> 'SMITH';
```

## 4.5.5 多行子查询

### ■ 多行子查询



多行比较运算符	含义
IN	等于其中的一个
NOT IN	不等于其中的一个

注意：

- ① 返回多行
- ② 使用多行比较操作符

## 4.5.6 多行子查询示例

### ■ 多行子查询示例

--查询包含s的岗位并查询在这些岗位上的员工信息

```
select * from emp where job in (select distinct job  
from emp where job like '%S%')
```

--查询没有员工的部门信息

```
select * from dept where deptno not in (select  
distinct deptno from emp)
```

## 4.5.7 多表查询（内连接）

### ■ 内连接

查询员工姓名、工资、部门名称、部门编号信息

#### – 第一种写法

```
select e.ename,e.sal,d.dname,d.deptno from emp e,dept  
d where e.deptno=d.deptno;
```

#### – 第二种写法

```
select e.ename,e.sal,d.dname,d.deptno from emp e  
inner join dept d on e.deptno=d.deptno;
```

注意：

如果d.deptno不加前面的d别名则会报未明确定义的列错误。



## 4.5.8 多表查询（左/右连接）

- 左连接：以左边的表为基准，连接后的结果左表里面的数据全部都有，如果关联的字段在右表中有值，那么就会显示关联的右表的数据，如果关联的字段在右表中没有值，则显示null。

--查询员工姓名,工资,部门名称,部门编号信息

```
select e.ename,e.sal,d.dname,d.deptno from emp e  
left join dept d on e.deptno=d.deptno;
```

- 右连接：与左连接刚好相反。

--查询员工姓名,工资,部门名称,部门编号信息

```
select e.ename,e.sal,d.dname,d.deptno from emp e  
right join dept d on e.deptno=d.deptno;
```

## 4.5.9 多表查询（其他连接）

### ■ 满连接 full join

满连接的结果就是左连接和右连接的结果去掉重复记录以后加在一起。

### ■ 交叉连接 cross join

交叉连接的结果就是笛卡尔积，注意没有条件。

### ■ 三个或者三个表以上连接

```
--求班级名称, 学生姓名, 选修的课程名, 课程的成绩  
select bj.bname, st.sname, cs.kname, cs.scroe  
from classes_student cs, classes bj, student st  
where cs.bid=bj.bid and st.sid=cs.sid;
```

5.1 DML(INSERT插入语句)

5.2 DML(DELETE删除语句)

5.3 DML(UPDATE修改语句)

## 5.1 DML (INSERT插入语句)

■ INSERT(向表中插入数据)。语法如下：

```
insert into 表(字段) values (值);
```

— 向表中所有字段插入数据：

```
insert into course values (5, 'sql', 'sql语言');
```

— 向表中部分字段插入数据：

```
insert into course(courseid,coursename)  
values (5, 'mysql');
```

— 向表中手动插入空值：

```
insert into course values (6, 'db2', null);
```

注意：

- ① 事务是由完成若干项工作的DML语句组成的。
- ② 事务是最小的执行单位。
- ③ COMMIT：提交事务，ROLLBACK：回滚事务。

## 5.2 DML (DELETE删除语句)

■ DELETE(删除表中的数据), 语法如下:

```
delete from 表 [where 条件];
```

— 删除一条记录:

```
delete from course c where c.courseid=7;
```

— 删除所有记录(可以回滚):

```
delete from course;
```

— 删除所有记录(清空数据,不能回滚):

```
truncate table classes;
```

— 在删除中使用子查询:

```
delete from student where score in (select score from  
student s where s.score<60);
```

## 5.3 DML (UPDATE修改语句)

■ UPDATE(修改表中的数据),语法如下:

```
update 表 set 字段=新值[,字段=新值] where 条件;
```

— 修改一条记录:

```
update student set name='张三' where studentid=4;
```

— 修改所有记录:

```
update student set name='李四';
```

6.1 Oracle日期

6.2 Oracle日期函数

6.3 Oracle函数

## 6.1 Oracle日期

- Oracle中的日期格式: yyyy-mm-dd hh24:mi:ss
- 查询系统当前日期

```
select sysdate, current_date from dual;
```

注意:

dual:不存在的虚拟表

- 操作日期
  - 获得10天以后的日期
  - 查询员工做了多少天
  - 查询1981.03.03以后入职的员工

```
select sysdate+10 from dual;
```

```
select sysdate-hiredate from emp;
```

```
select * from emp where hiredate>'03-3月-1981';
```



## 6.2 Oracle日期函数

### ■ 日期函数

- `add_months()`：在当前日期上加多少个月

例：查询所有员工的转正日期

```
select add_months(hiredate,3) from emp;
```

- `months_between()`：计算两个日期之间的月份数

例：查询所有员工的工龄

```
select months_between(sysdate,hiredate)/12 from emp;
```

- `last_day()`：返回输入日期当前月的最后一天

```
select last_day('1-2月-2016') from dual;
```

## 6.3 Oracle函数

6.3.1 TO\_CHAR

6.3.2 TO\_DATE

6.3.3 TO\_NUMBER

6.3.4 MOD

6.3.5 ROUND

6.3.6 TRIM

6.3.7 REPLACE

6.3.8 其他函数

# TO\_CHAR

### ■ 转换字符

字符TO\_CHAR(c): 将nchar、nvarchar2、clob、nclob类型转换为char类型。

### ■ 转换时间

字符TO\_CHAR(d[,fmt]): 将指定的时间(data,timestamp,timestamp with time zone)按照指定格式转换为varchar2类型。

### ■ 转换数值

字符TO\_CHAR(n[,fmt]): 将指定数值n按照指定格式fmt转换为varchar2类型并返回。

# 转换字符

```
SQL> SELECT TO_CHAR('AABBCC') FROM DUAL;
```

```
TO_CHAR('AABBCC')
```

```
-----
```

```
AABBCC
```

# 转换时间

```
SQL> SELECT TO_CHAR(sysdate,'yyyy-mm-dd hh24:mi:ss') FROM DUAL;
```

```
TO_CHAR(SYSDATE,'YYYY-MM-DDHH24:MI:SS')
```

```
-----  
2013-05-13 10:43:23
```

```
SQL> select to_char(systimestamp(9),'yyyymmddhh24missff') from dual;
```

```
TO_CHAR(SYSTIMESTAMP(9),'YYYYMMDDHH24MISSFF')
```

```
-----  
20130513104536926486000
```

# 转换数值

```
SQL> SELECT TO_CHAR(-100, 'L99G999D99MI') FROM DUAL;
```

```
TO_CHAR(-100,'L99G999D99MI')
```

```
-----
```

```
¥ 100.00-
```

# TO\_DATE

- 将char、nchar、varchar2、nvarchar2转换为日期类型，如果fmt参数不为空，则按照fmt中指定格式进行转换。注意这里的fmt参数。如果ftm为‘J’则表示按照公元制(Julian day)转换，c则必须为大于0并小于5373484的正整数。

### TO\_DATE示例

```
SQL> SELECT TO_DATE(2454336, 'J')  
,TO_DATE('2007-8-23 23:25:00', 'yyyy-mm-dd hh24:mi:ss')  
FROM DUAL; 2 3
```

```
TO_DATE(2454 TO_DATE('200
```

```
-----
```

```
23-AUG-07 23-AUG-07
```



### TO\_DATE示例

- DATE类型的取值范围是公元前4712年1月1日至公元9999年12月31日。

```
SQL> SELECT TO_CHAR(TO_DATE('9999-12-31','yyyy-mm-dd'),'j')  
FROM DUAL;
```

```
TO_CHAR(TO_DATE  
-----  
5373484
```

```
SQL> SELECT TO_DATE('10001-12-31','yyyy-mm-dd') FROM dual;  
SELECT TO_DATE('10001-12-31','yyyy-mm-dd') FROM dual  
*
```

ERROR at line 1:

ORA-01861: literal does not match format string

# TO\_NUMBER

- 将char、nchar、varchar2、nvarchar2型字符串按照fmt中指定格式转换为数值类型并返回。

```
SQL> SELECT TO_NUMBER('-1000.00') FROM DUAL;
```

```
TO_NUMBER('-1000.00')
```

```
-----
```

```
-1000
```

## MOD(n1,n2)

- 返回n1除n2的余数，如果n2=0则返回n1的值。

```
SQL> SELECT MOD(24, 5) FROM DUAL;
```

```
MOD(24, 5)
```

```
-----
```

```
4
```

```
SQL> SELECT MOD(24, 0) FROM DUAL;
```

```
MOD(24, 0)
```

```
-----
```

```
24
```

## ROUND(n1 [,n2])

- 返回四舍五入小数点右边n2位后n1的值，n2缺省值为0，如果n2为负数就舍入到小数点左边相应的位上(虽然oracle documents上提到n2的值必须为整数，事实上执行时此处的判断并不严谨，即使n2为非整数，它也会自动将n2取整后做处理)。

```
SQL> SELECT ROUND(23.56), ROUND(23.56, 1) FROM DUAL;
```

```
ROUND(23.56)  ROUND(23.56, 1)
```

```
-----
```

```
24
```

```
23.6
```

# **TRIM ([[LEADING | | TRAILING | | BOTH] c2FROM]c1)**

- 如果没有指定任何参数则去除c1头尾空格。
- 如果指定了c2，并以c2为参数去掉同c2相同的字符，头尾有连续的c2继续去掉，直到头尾找不到c2为止。

### TRIM示例

```
SQL> SELECT TRIM(' How are you! ') FROM DUAL;  
TRIM(' HOWAREYOU!')
```

-----  
How are you!

### **REPLACE(c1,c2[,c3])**

- 将c1字符串中的c2替换为c3，如果c3为空，则从c1中删除所有c2。

### REPLACE示例

将所有的H的替换成MM

```
SQL> select replace('How are you','H','MM') from dual;  
REPLACE('HOWAREYOU','H',
```

```
-----  
MMow are you
```

将所有的o替换成MM

```
SQL> select replace('How are you','o','MM') from dual;  
REPLACE('HOWAREYOU','O','M
```

```
-----  
HMMw are yMMu
```

把所有的H去掉

```
SQL> select replace('How are you','H','') from dual;  
REPLACE('HOWAREYOU',
```

```
-----  
ow are you
```



### 其他函数

操作符	作用
Lower	转换小写
upper	转换大写
substr	取子串
length	取长度
initcap	首字母大写
case	同decode
decode	同case

7.1 伪列(ROWID)

7.2 伪列(ROWNUM)

## 7.1 伪列 (ROWID)

### ■ 伪列 (ROWID)

- 1、ROWID是一个伪列，是用来确保表中行的唯一性，它并不能指示出行的物理位置，但可以用来定位行。
- 2、ROWID是存储在索引中的一组既定的值（当行确定后）。我们可以像表中普通的列一样将它选出来。
- 3、利用ROWID是访问表中一行的最快方式。
- 4、ROWID与磁盘驱动的特定位位置有关，因此，ROWID是获得行的最快方法。但是，行的ROWID会随着卸载和重载数据库而发生变化，因此建议不要在事务中使用ROWID伪列的值。例如，一旦当前应用已经使用完记录，就没有理由保存行的ROWID。不能通过任何SQL语句来设置标准的ROWID伪列的值。
- 5、ROWID需要10个字节来存储，显示为18位的字符串。

## 7.1 伪列(查询ROWID)

### ■ 查询ROWID

```
SQL> SELECT sname,ROWID FROM student;
```

SNAME	ROWID
-----	-----
张三	AAAM0XAAIAAAAAOAAA
学员2	AAAM0XAAIAAAAAOAAB
学员3	AAAM0XAAIAAAAAOAAC
学员4	AAAM0XAAIAAAAAOAAD
学员5	AAAM0XAAIAAAAAOAEE
学员6	AAAM0XAAIAAAAAQAAA
学员7	AAAM0XAAIAAAAAQAAB

7 rows selected.

## 7.1 伪列 (ROWID条件查询)

### ■ ROWID条件查询

```
SQL> SELECT studentid, sname  
FROM student  
WHERE rowid = 'AAAM0XAAIAAAAAOAAA';
```

STUDENTID	SNAME
-----------	-------

1	张三
---	----

# ROWNUM

ROWNUM是ORACLE数据库顺序分配为从查询结果集返回行的编号，返回的第一行分配的是1，第二行是2，依此类推，这个伪字段可以用于限制查询返回的总行数，而且ROWNUM不能以任何表的名称作为前缀。

## 7.2 ROWNUM简单查询

### ■ ROWNUM简单查询

```
SQL> SELECT studentid, sname,ROWNUM  
FROM student;
```

STUDENTID	SNAME	ROWNUM
1	张三	1
2	学员2	2
3	学员3	3
4	学员4	4
5	学员5	5
6	学员6	6
7	学员7	7

7 rows selected.

## 7.2 ROWNUM分页查询

### ■ 分页查询示例

- 查询公司员工表emp中第6条到第10条的员工记录

```
select * from  
  (select rownum rm,t.* from emp t where rownum<=10) temp  
 where rm>=6 and rm<=10;
```

注意：

ROWNUM只能应用于<和<=，不能应用于>和>=。



8.1 Oracle角色

8.2 Oracle表空间、用户及授权

8.3 创建表、修改表、查询表

8.4 序列(SEQUENCE)

8.5 视图(VIEW)

8.6 索引

8.7 存储过程

8.8 约束

8.9 导入导出

### ■ Oracle常用角色

#### – **CONNECT** ROLE(连接角色)

临时用户，特别是那些不需要建表的用户，通常只赋予他们CONNECT role。CONNECT是使用Oracle的简单权限。

#### – **RESOURCE** ROLE (资源角色)

更可靠和正式的数据库用户可以授予RESOURCE role。RESOURCE提供给用户额外的权限以创建他们自己的表、序列、过程(procedure)、触发器(trigger)、和索引(index)等。

#### – **DBA** ROLE (数据库管理员角色)

DBA role拥有所有的系统权限包括无限制的空间限额和给其他用户授予各种权限的能力。

## 8.2 Oracle表空间、用户及授权

### ■ 创建表空间(表空间文件不能手动删除)

#### – 创建表空间tests

```
create tablespace tests  
datafile 'C:\app\Administrator\oradata\orcl\testss.DBF'  
size 100M --文件的初始大小是100M  
autoextend on next 50M --当文件达到100M时,自动增加50M
```

### ■ 创建用户

#### – 创建用户mtesting, 密码123

```
create user mtesting identified by 123;
```

### ■ 给用户授权

#### – 给用户mtesting分别赋予connect, resource, dba的权限

```
grant connect to mtesting  
grant resource to mtesting ;  
grant dba to mtesting ;
```

## 8.3 创建表

### ■ 创建表

- 必须具备的条件
  - CREATE TABLE的权限
  - 存储空间
- 创建学生信息表

```
CREATE TABLE student
(
  studentid integer, --学生学号, 唯一标识学生的标志
  sname    varchar2(50), --学生姓名
  sex      varchar2(30), --性别
  birthday date --出生年月
);
```

## 8.3 创建表(常用数据类型)

常用数据类型	描述
<code>VARCHAR2(size)</code>	可变长字符数据
<code>CHAR(size)</code>	定长字符数据
<code>NUMBER(p,s)</code>	可变长数值数据
<code>DATE</code>	日期型数据

### ■ 修改表

#### — 追加新的列

```
alter table student add (score number);
```

#### — 修改现有的列

- 修改字段类型

```
alter table student modify (score varchar2(50));
```

- 修改字段名称

```
alter table student rename column score to aaa;
```

#### — 删除一个列

```
alter table student drop (aaa);
```

### ■ 删除表

```
drop table student;
```

注意：

- ① 数据和结构都被删除
- ② 所有正在运行的相关事务被提交
- ③ 所有相关索引被删除
- ④ DROP TABLE 语句不能回滚

## 8.4 序列 (SEQUENCE)

### ■ 创建序列

```
create sequence student_seq
```

### ■ 查询序列

- 查询序列的当前值（第一次不能查询）

```
select student_seq.nextval from dual;
```

- 查询序列的下一个值

```
select student_seq.currval from dual;
```

### ■ 使用序列

- 使用序列实现主键自增

```
insert into student values(student_seq.nextval,'张三');
```



## 8.5 视图 (VIEW)

### 8.5.1 视图简介

### 8.5.2 视图的好处

### 8.5.3 创建和查询视图

- 视图是基于一个表或多个表或视图的逻辑表，本身不包含数据，通过它可以对表里面的数据进行查询和修改。视图基于的表称为基表。视图是存储在数据字典里的一条select语句。通过创建视图可以提取数据的逻辑上的集合或组合。

## 8.5.2 视图的好处

- 对数据库的访问更便捷，因为视图可以有选择性的选取数据库里的一部分。
- 用户通过简单的查询可以从复杂查询中得到结果。
- 维护数据的独立性，视图可从多个表检索数据。
- 对于相同的数据可产生不同的视图。

## 8.5.3 创建和查询视图

### ■ 创建视图

```
create or replace view de_view  
as  
select dname,ename from emp e,dept d where e.deptno=d.deptno
```

### ■ 查询视图

```
select * from de_view;
```

8.6.1 索引简介

8.6.2 索引的好处

8.6.3 索引的不足

8.6.4 索引的分类

8.6.5 创建索引的语法

- 索引就是加快检索表中数据的方法。数据库的索引类似于书籍的索引。在书籍中，索引允许用户不必翻阅完整的书就能迅速地找到所需要的信息。在数据库中，索引也允许数据库程序迅速地找到表中的数据，而不必扫描整个数据库。

## 8.6.2 索引的好处

- 通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。
- 可以大大加快数据的检索速度，这也是创建索引的最主要的原因。
- 可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。
- 在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。
- 通过使用索引，可以在查询的过程中，搞高SQL语句的处理速度，提高系统的性能。

## 8.6.3 索引的不足

- 创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。
- 索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间。
- 当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。



## 8.6.4 索引的分类

- B树(B-Tree)索引(默认索引类型)
- 唯一(Unique)索引
- 位图(Bitmap)索引
- HASH索引
- 反转键(Reverse Key)索引
- 函数索引
- 分区索引
- 位图连接索引

## 8.6.5 创建索引的语法

### ■ 创建索引语法

```
CREATE UNIQUE | BITMAP INDEX <schema>.<index_name>  
ON <schema>.<table_name>  
    (<column_name> | <expression> ASC | DESC,  
     <column_name> | <expression> ASC | DESC,...)  
TABLESPACE <tablespace_name>
```

### ■ 相关说明

- 1、UNIQUE | BITMAP：指定UNIQUE 为唯一值索引， BITMAP 为位图索引，省略为B-Tree 索引。
- 2、<column\_name> | <expression> ASC | DESC：可以对多列进行联合索引，当为expression 时即 “基于函数的索引”
- 3、TABLESPACE：指定存放索引的表空间 (索引和原表不在一个表空间时效率更高)。

## 8.7 存储过程

8.7.1 存储过程简介

8.7.2 存储过程的好处

8.7.3 创建存储过程语法

8.7.4 创建存储过程

## 8.7.1 存储过程简介

- 存储过程(Procedure)是一组为了完成特定功能的SQL 语句集，存储在数据库中，经过第一次编译后再次调用不需要再次编译。
- 用户通过指定存储过程的名字并给出参数(如果该存储过程带有参数)来执行它。存储过程是数据库中的一个重要对象，任何一个设计良好的数据库应用程序都应该用到存储过程。

## 8.7.2 存储过程的好处

- **重复使用：**存储过程可以重复使用，从而可以减少数据库开发人员的工作量。
- **减少网络流量：**存储过程位于服务器上，调用的时候只需要传递存储过程的名称以及参数就可以了，因此降低了网络传输的数据量。
- **安全性：**参数化的存储过程可以防止SQL注入式攻击，而且可以将Grant、Deny以及Revoke权限应用于存储过程。

## 8.7.3 存储过程的语法

```
create or replace procedure 存储过程名 (param1 in type, param2 out type)
as
变量1 类型 (值范围);
变量2 类型 (值范围);
Begin
    Select count(*) into 变量1 from 表A where 列名=param1;
    If (判断条件) then
        Select 列名 into 变量2 from 表A where 列名=param1;
        Dbms_output.Put_line('打印信息' );
    Elsif (判断条件) then
        Dbms_output.Put_line('打印信息' );
    Else
        Raise 异常名 (NO_DATA_FOUND) ;
    End if;
Exception
    When others then
        Rollback;
End;
```

## 8.7.4 创建存储过程

### ■ 创建存储过程

```
create or replace procedure proc1(  
    p_para1 varchar2  
)as  
    v_name varchar2(20);  
begin  
    v_name := '张三丰';  
    p_para1 := v_name;  
    dbms_output.put_line('p_para1:' || p_para1);  
end;
```

### ■ 调用存储过程

```
SQL>call proc1();a
```

- 主键primary key约束：不能为空, 必须唯一。
- 外键foreign key约束：不能增加外键对应的字段中没有的值。
- unique唯一性约束：可以为空, 必须唯一。
- not null约束：表示字段不能为空。



### ■ 导出用户对象 export user objects

只能导出用户对象（表，视图，序列.....）的结构而不能导出数据。

### ■ 导出用户表 export user tables

可以导出两种格式：.sql和.dmp文件，并且可以导出表里面的数据。

### ■ 导入

先导入表结构，再导入表数据。

# 联系我们

电话：0755-83221336/13928429246（微信同步）

邮箱：[service@mtesting.net](mailto:service@mtesting.net)

官网：[www.mtesting.net](http://www.mtesting.net)

学习社区：[www.mtesting.cn](http://www.mtesting.cn)

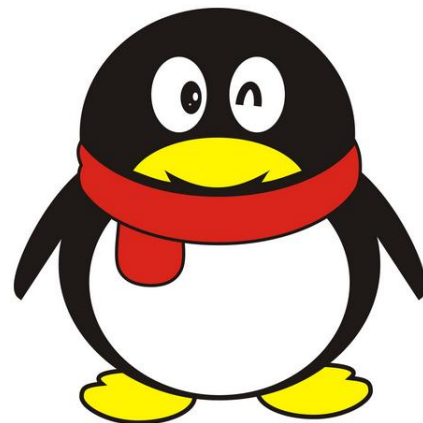
地址：深圳市福田区彩田南路深圳青年大厦五楼



手机扫描访问官网



手机扫描关注公众号



QQ群号:15233368

感谢您对我们的关注



Thanks&Best wishes for you!

**多动脑 勤动手 定成功**