2017-11-11

## 数据库介绍

什么是数据库?

数据库是为了实现一定的目的按某种规则组织起来的数据的集合, 简单的说,数据库就是存储数据的库.

举个例子来说明这个问题：每个人都有很多亲戚和朋友，为了保持与他们的联系，

我们常常用一个笔记本将他们的姓名、地址、电话等信息都记录下来，

这样要査谁的电话或地址就很方便了。

这个“通讯录”就是一个最简单的“数据库”，每个人的姓名、地址、电话等信息就是这个数据库中的“数据”。

我们可以在笔记本这个“数据库”中添加新朋友的个人信息，也可以由于某个朋友的电话变动而修改他的电话号码这个“数据”。

不过说到底，我们使用笔记本这个“数据库”还是为了能随时査到某位亲戚或朋友的地址、邮编或电话号码这些“数据”。

数据库是干什么用的?

存储和管理数据,便于程序开发.

oracle简介:

Oracle是殷墟（yīn Xu）出土的甲骨文（oracle bone inscriptions）的英文翻译的第一个单词。

Oracle公司成立与1977年，总部位于美国加州；

Oracle数据库是Oracle（中文名称叫甲骨文）公司的核心产品，Oracle数据库是一个适合于大中型企业的数据库管理系统。在所有的数据库管理系统中（比如：微软的SQL Server，IBM的DB2等），Oracle的主要用户涉及面非常广, 包括: 银行、电信、移动通信、航空、保险、金融、电子商务和跨国公司等。

Oracle数据库的一些版本有：Oracle7、Oracle8i、Oracle9i，Oracle10g到Oracle11g，Oracle12c， 各个版本之间的操作存在一定的差别，但是操作oracle数据库都使用的是标准的SQL语句，因此对于各个版本的差别不大。

2008年1月16日 收购bea，目的是为了得到weblogic（web服务器的框架，免费的对应的tomcat）。

2008年1月16日 sun公司收购了mysql 。

2009年4月20日 oracle收购了sun 。

常见的数据库有哪些？

1 oracle公司的oracle数据库

2 IBM公司的DB2数据库

3 Informix公司的Informix数据库

4 sysbase公司的sysbase数据库

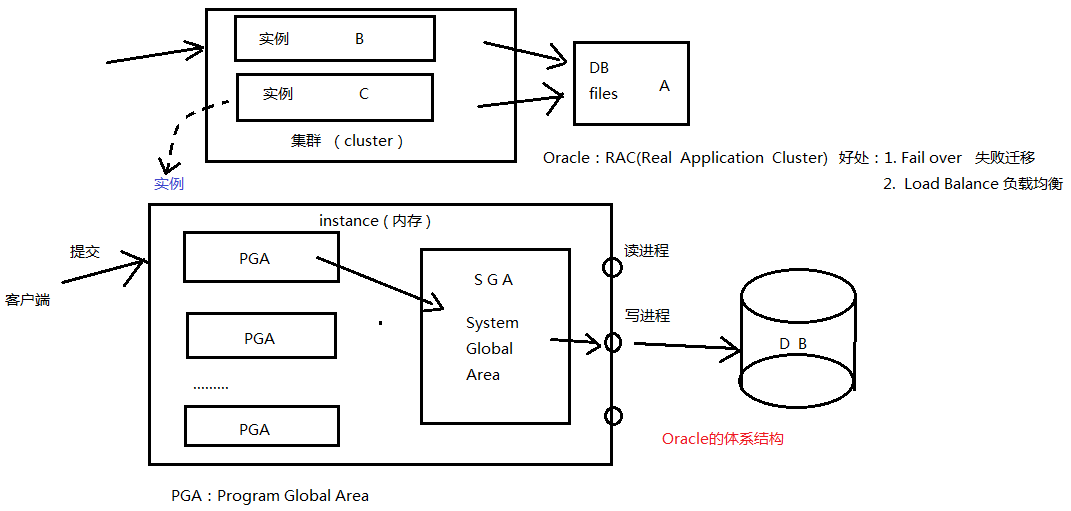
5 Microsoft公司的SQL Server

6 oracle的MySQL数据库（开始属于mysql公司，后来mysql被sun收购，sun又被oracle收购）

7 [MongoDB](https://baike.baidu.com/item/MongoDB%E6%9D%83%E5%A8%81%E6%8C%87%E5%8D%97/7338398)数据库

## 1 Oracle的体系结构

Oracle服务器：是一个数据管理系统(RDBMS)，它提供开放的, 全面的, 近乎完整的信息管理。由1个数据库和一个（或多个）实例组成。数据库位于硬盘上，实例位于内存中。



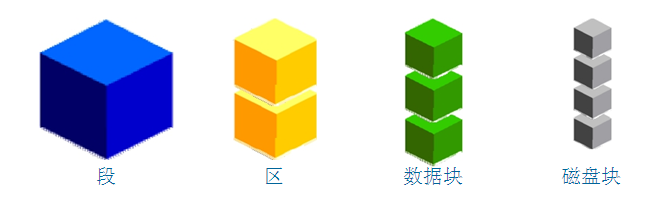
### 1.1 表空间和数据文件

逻辑概念：表空间是由数据文件组成，位于实例上，在内存中。

物理概念：数据文件，在磁盘上（/home/oracle\_11/app/oradata/orcl目录中的.DBF文件）；

一个表空间包含一个或者多个数据文件。

### 1.2 段、区、块



段存在于表空间中，段是区的集合，区是数据块的集合，数据块会被映射到磁盘块。

图请看讲义部分。

### 1.3 DBA

**数据库管理员**（Database Administrator，简称DBA），是从事管理和维护[数据库管理系统](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F)(DBMS)的相关工作人员的统称，属于[运维工程师](https://baike.baidu.com/item/%E8%BF%90%E7%BB%B4%E5%B7%A5%E7%A8%8B%E5%B8%88)的一个分支，主要负责业务数据库从设计、测试到部署交付的全生命周期管理。DBA的核心目标是保证数据库管理系统的稳定性、安全性、完整性和高性能。从时间开销上看：安装占用6%，创建和配置占用12%，装载和存档数据占6%, 软件维护占6%，管理数据库系统占55%，可见，管理数据库系统占用了大部分的时间开销。

### 1.4 如何启动数据库服务

Windows启动oracle数据库服务：

启动服务：services.msc，找到下列两个服务，并启动。

OracleServiceORCL： oracle数据库服务系统

home1TNSListene： 监听服务，用于远程连接的侦听

注意：若把数据库默认设置为自启动，则开机时间会延长。

Linux启动oracle数据库服务步骤(oracle数据库系统安装到linux系统上)

1 执行sqlplus / as sysdba或sqlplus sys/sys as sysdba进入到命令行界面

2 执行startup启动数据库服务

3 执行exit退出sqlplus命令行界面

4 执行lsnrctl start启动监听服务

注意：通过远程客户端连接oracle服务端必须要启动监听服务，否则客户端连接不上。

### 1.5 如何登陆数据库服务

1在本机登陆：

* 普通用户身份登陆

sqlplus ↙ 用户名 ↙ 密码 ↙

sqlplus 用户名/密码，如sqlplus scott/tiger

* 以管理员身份登陆

sqlplus / as sysdba（此处不用输入密码，在安装的时候已经输入密码）

sqlplus sys/sys as sysdba

2 通过网络使用客户端远程登陆

远程通过网络登陆数据库需要安装oracle客户端软件，并进行配置才能使用，可通过使用net manager进行配置，配置完成之后可以使用连接字符串进行登陆，连接字符串中包含了数据库服务的IP地址和端口，以及实例名。

注意：安装oracle客户端的时候，安装路径中不能出现中文和空格，安装的时候选择管理员模式。

D:\oracle\app\HGUANG\product\11.2.0\client\_1\network\admin\tnsnames.ora，下面是经过Net Manager进行配置后得到的一个文件内容：



客户端安装完成之后进行远程登陆之前最好先进行测试：

首先测试网络是否是通的： ping IP ， 然后tnsping IP 或者 tnsping oracle\_orcl。

* 普通用户登陆

sqlplus 用户名/密码@连接字符串，如sqlplus scott/tiger@oracle\_orcl

管理员用户登陆

sqlplus sys/sys@oracle\_orcl as sysdba

此外：还可以执行： sqlplus scott/tiger@//IP地址/实例名 进行登陆。

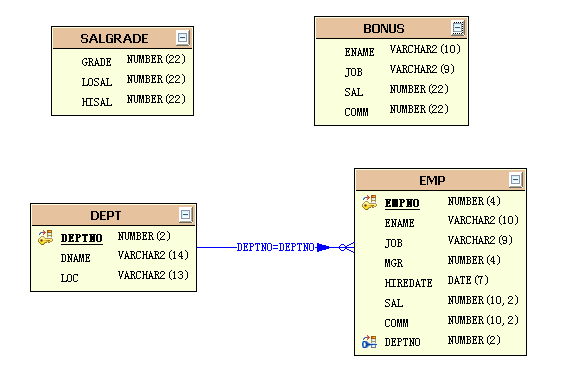
使用scott用户或者sys用户登陆完之后，可以使用show user测试一下，如果显示用户名就表明已经登陆成功了，或者是执行select \* from tab；进行一次查询, 有结果显示就表名已经登陆成功了.

* 解锁用户：alter user scott account unlock (管理员身份登陆，给scott用户解锁。用户默认锁定)
* 锁定用户：alter user scott account lock，（必须用管理员用户登陆）
* 修改用户密码：alter user scott identified by 新密码 (管理员身份登陆，给scott用户修改密码)
* 查看当前语言环境：select userenv('language') from dual;

### 1.6 贯穿这门课程的方案

请看oracle讲义。

scott用户的emp表, dept表和salgrade表.



## 2基本的SQL select语句

### 2.1 sqlplus的基本操作

显示当前用户： show user;

查看当前用户下的表：select \* from tab;

tab: 数据字典（记录[数据库](http://baike.baidu.com/view/1088.htm)和[应用程序](http://baike.baidu.com/view/330120.htm)源数据的目录），包含当前用户下的表。

查看员工表的结构： desc emp;

设置行宽：set linesize 120;

设置页面：set pagesize 100;

或者将上述两行写入如下两个配置文件，可永久设置：

C:\app\Administrator\product\11.2.0\client\_1\sqlplus\admin\glogin.sql

C:\app\Administrator\product\11.2.0\dbhome\_1\sqlplus\admin\glogin.sql

设置员工名列宽：col ename for a20 (a表示字符串)

设置薪水列为4位数子：col sal for 9999 (一个9表示一位数字)

### 2.2 基本的select语句

Select语句的整体形式：

select col1, col2…

from table\_name

where condition

group by col…

having condtion

order by col…

其语法格式为：

**SELECT \*|{[DISTINCT] *column*|*expression* [*alias*],...}** **FROM *table;***

**案例**

1查询所有员工的所有记录

select \* from emp;

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

说明：尽量使用列名，用列名代替\* (oracle 9i之前不同， 之后一样)

2 查询员工号、姓名、薪水

select empno, ename, sal from emp;

3 查询员工号、姓名、薪水和年薪

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

说明：select语句中可以使用表达式

注意：在执行sql语句的时候，执行 / 会执行上一次执行的语句。

若在写sql语句的时候, 写错了, 可以使用c（change）命令来修改. 默认, 光标闪烁位置指向上一条SQL语句的第一行。输入 2 则定位到第二行。

c /错误关键字/正确关键字 ↙

使用“/”来执行修改过的SQL语句

例如：错误输入: select empno, ename, sal, sal \* 12

form emp; （“from”书写错误，该错误位于整条SQL语句的第二行）

1. 输入：2↙ 终端提示：2\* from emp
2. 输入：c /form/from ↙ 终端提示：2\* from emp (意为改正后的sql语句样子)
3. 输入：/↙

也可以使用ed(或者edit)命令来修改

ed ↙弹出系统默认的文本编辑器，(如记事本)

修改、保存、退出、执行“/”。

* 使用edit打开文件编辑sql语句:

如果sql语句很长, 可以分行编写或者用ed命令打开一个文件,

然后在文件中编写sql语句, 注意, sql语句末尾不要加; , 然后换行加上/表示结束.

* 若想将显示结果保存到文件中:

spool命令:

spool d:\result.txt;

select \* from emp;

spool off;

**使用别名：**

别名：as

案例：查询员工号，姓名，月薪，年薪

select empno as "员工号", ename "姓名", sal 月薪, sal\*12 年薪 from emp;

说明：关键字as写与不写没有区别； ""有与没有取决于别名中是否有空格，建议在用别名的时候加上""。

DISTINCT---🡪重复记录只取一次

案例：

select deptno from emp;

select DISTINCT deptno from emp;

select job from emp;

select distinct job from emp;

select distinct deptno, job from emp;

说明: DISTINCT的作用范围不是距离它最近的列, 而是后面的所有的列。

### 2.3 SQL语句使用注意事项

* SQL 语言大小写不敏感。

sql语句对关键字的大小写不敏感, 如SELECT和select一样, 但是对于内容敏感。

* SQL可以写在一行或者多行
* 关键字不能被缩写也不能分行
* 各子句一般要分行写。
* 使用缩进提高语句的可读性

### 2.4 算数运算 + - \* /

* 乘除的优先级高于加减
* 优先级相同时, 按照从左至右运算
* 可以使用括号改变优先级

查询: 员工号、姓名、月薪、年薪、奖金、年收入。

select deptno, ename, sal, sal\*12, comm, comm+sal\*12 from emp;

结果不正确, 没有奖金的员工, 年收入不正确。

说明： 在程序开发过程中, 数据是核心. 程序再正确也没有用, 必须保证数据不能丢, 且正确, 对于上面的结果, 有对有错的情况是最危险的.

解决这个问题需要正确认识NULL值。

### 2.5 NULL值

* NULL值问题：

1. 包含NULL值的表达式都为空.
2. NULL不等于NULL

select \* from emp where NULL=NULL; 查不到任何记录。

解决：滤空函数：nvl(a, b) 如果a为NULL，返回b；

所以：使用sal \* 12 + nvl(comm, 0) 表示年收入。

1. 在SQL中, 判断一值是否等于NULL不用“=” 和“!=”而使用is和is not
   * + 查询奖金为NULL的员工信息：

select \* from emp where comm = NULL; (SQL中不使用==)

应该使用：select \* from emp where comm is NULL;

* + - 查询奖金不为NULL的员工信息：

select \* from emp where comm is not NULL;

总结: 空值是无效的, 未指定的, 未知的或不可预知的值, 空值不是空格或者0.

### 2.6 连接符

在oracle中使用 || 连接字符串

例如： select ename || ' is a ' || job from emp;

若要显示hello world字符串应该怎么办呢?

oracle中语法规定：select后面必须接from关键字，所有需要有一个表名用来满足oracle的SQL99语法规定，为此定义了一个伪表dual。

select concat('hello ', 'world') from dual;

若要连接三个字符串呢？

select concat(concat('hello ', 'world'), ' i love you') from dual;

注意：concat函数只支持两个参数，不支持三个参数形式。

也可以使用||连接三个以上的字符串：

select 'hello ' || 'world ' || 'i love you' from dual;

说明：使用||比concat函数要灵活，其实||就是用来替换concat函数的。

### 2.7 SQL和sqlplus

我们已经学习使用了select，应该知道还有update、delete、insert、create…

同时，我们学习了ed、c、set、col、desc….

SQL是语言，关键字不能缩写。

sqlplus是oracle提供的工具，可在里面执行SQL语句，它配有自己的命令(ed、c、set、col) 特点是缩写关键字。

SQL

* + 一种语言
  + ANSI 标准
  + 关键字不能缩写
  + 使用语句控制数据库中的表的定义信息和表中的数据

SQL\*Plus

* + 一种环境
  + Oracle 的特性之一
  + 关键字可以缩写
  + 命令不能改变数据库中的数据的值
  + 集中运行

## 3 过滤和排序数据

### 3.1 where条件过滤

在where条件中使用的列的值对大小写是敏感的, 如是字符串需要用单引号引起来, 如KING和king是不同的字符串。

如select \* from emp where ename= 'KiNg '；未选定行

select \* from emp where ename= 'KING '； 正确

说明： 对于列的值来说，大小写是敏感的。

[未选定行]不是发生了错误，而是查不到记录。

#### 日期格式

查询入职日期为1981年11月17日的员工

select \* from emp where hiredate= '1981-11-17 '；

会报错，ORA-01861: 文字与格式字符串不匹配。

首先查询一下emp使用的日期格式：

select \* from emp;

使用sysdate函数获取当前系统的日期：

select sysdate from dual;

查询得出格式为： DD-MON-RR

应该使用当前系统指定的日期格式来查询：

select \* from emp where hiredate = '17-11月-81 ';

如何获取当前系统的日期格式?

select \* from v$nls\_parameters;

若格式不太好看，可以执行col parameter for a30设置列的宽度。

如何修改日期格式?

alter session set NLS\_DATE\_FORMAT = 'yyyy-mm-dd ';

查看修改结果：

select \* from v$nls\_parameters;

验证：再次查询emp表：

select \* from emp where hiredate= '1981-11-17 ';

修改日期格式到秒

alter session set NLS\_DATE\_FORMAT = 'yyyy-mm-dd hh24:mi:ss';

执行select sysdate from dual;验证修改结果.

将日期格式改回默认设置

alter session set NLS\_DATE\_FORMAT = 'DD-MON-RR';

说明:

* 字符和日期要包含在单引号中。
* 字符大小写敏感, 日期格式敏感。
* 默认的日期格式是DD-MON-RR

#### 比较运算

* 普通比较运算符：

= 等于(不是==) > 大于

>= 大于等于 < 小于

<= 小于等于 <> 不等于(也可以是!=)

案例:

1 查询薪水不等于1250的员工信息

select \* from emp where sal <> 1250;

select \* from emp where sal != 1250;

* between…and:介于两值之间,闭区间,包含两边的值.

案例:

1 查询工资在1000-2000之间的员工：

select \* from emp where sal >=1000 and sal<=2000;

select \* from emp where sal between 1000 and 2000;

注意：1.包含边界 2. 小值在前，大值在后。 (对于日期也是如此)

2 查询81年2月至82年2月(不含2月)入职的员工信息：

select \* from emp where hiredate between '1-2月-81' and '30-1月-82';

* in：在集合中, not in 不在集合中

案例

1 查询部门号为10和20的员工信息：

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

select \* from emp where deptno in (10, 20);

2 查询部门号不是10和20的员工(除了10和20以外的部门)

select \* from emp where deptno not in (10, 20);

使用比较运算符该怎么写呢？

select \* from emp where deptno!=10 and deptno!=20;

如果是 ….. not in (10, 20, NULL) 可不可以呢?

☆NULL空值:如果结果中含有NULL,不能使用not in 操作符,但可以使用in操作符。

课后思考为什么???

因为: not in (10, 20, NULL)相当于:

deptno!=10 and deptno!=20 and deptno!=NULL包含NULL的表达式都为空.

* like：模糊查询

%匹配任意多个字符, \_匹配一个字符, 使用escape表示转义字符

案例:

1 查询名字以S开头的员工

select \* from emp where ename like 'S% '; (注意：S小写、大写不同)

2 查询名字是4个字的员工

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

增加测试例子：向表中插入员工：

3 insert into emp(empno, ename, sal, deptno) values(1001, ' TOM\_ABC ', 3000, 10);

查询名字中包含\_的员工：

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

查不到记录.

转义字符：

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

转义单引号本身使用两个单引号来完成转义

select 'hello '' world' from dual;

#### 逻辑运算

* 逻辑运算符

AND 逻辑并

OR 逻辑或

NOT 逻辑非

如果…..where 表达式1 and/or 表达式2;

…..where 表达式2 and/or 表达式1;

这两句SQL语句功能一样吗？效率一样吗？

※SQL优化：

SQL在解析where的时候，是从右至左解析的。

所以： and时应该将易假的值放在右侧;

or时应该将易真的值放在右侧.

这样的话可以提高效率.

案例:

1 查询部门为30且工种为SALESMAN的员工

select \* from emp where deptno=30 and job='SALESMAN';

2 查询部门为10或者工资大于2000的员工

select \* from emp where deptno=10 or sal>2000;

3 查询工种不是MANAGER或者PRISIDENT的员工

select \* from emp where job!='MANAGER' and job!='PRISIDENT';

select \* from emp where job not in ('MANAGER', 'PRESIDENT');

#### order by 排序

* 使用 ORDER BY 子句排序
  + ASC（ascend）: 升序。默认采用升序方式。
  + DESC（descend）: 降序

ORDER BY子句在SELECT语句的最末尾, 是对select查询的最后的结果进行排序.

**案例:**

1 查询emp表, 按照入职日期先后排序

select \* from emp order by hiredate;

2 查询员工信息, 按月薪排序

select \* from emp order by sal; ---从小到大排序, 默认方式.

select \* from emp order by sal desc; ---从大到小排序

* order by 之后可以跟那些内容呢？

order by + 列名, 序号, 表达式, 别名,

注意：语法要求order by子句应放在select的结尾。

案例:

1 查询员工信息, 按月薪排序----🡪使用列名排序的情况

select \* from emp order by sal;

2 按照工资进行排序--🡪使用序号进行排序的情况

select ename, sal, sal\*12, from emp order by 2 desc;

序号: 按照select后面列名出现的先后顺序, ename→1, sal→2, sal\*12→3

3 按照员工的年薪进行排序----🡪使用表达式排序的情况

select ename, sal, sal\*12 from emp order by sal \* 12 desc;

4 按照员工的年薪进行排序----🡪使用别名进行排序的情况

select ename, sal, sal\*12 "年薪" from emp order by "年薪" desc;

* 按照两列或者多列进行排序

案例:

1 按照部门和工资进行排序

select \* from emp order by deptno, sal;

order by后有多列时, 列名之间用逗号隔分, order by会同时作用于多列。上例的运行结果会在同一部门内升序, 部门间再升序。

select \* from emp order by deptno, sal desc;

注意: desc只作用于最近的一列, 两列都要降序排, 则需要两个desc。即：

select \* from emp order by deptno desc, sal desc;

2 查询员工信息, 按奖金由高到低排序：

select \* from emp order by comm desc;

结果前面的值为NULL, 数据在后面, 如果是一个100页的报表，这样显示肯定不

正确。较为人性化的显示应该将空值放在最后, 即：

select \* from emp order by comm desc nulls last(注意：是nulls 而不是null)

* 排序的规则
* 可以按照select语句中的列名排序
* 可以按照别名排序
* 可以按照表达式排序,如order by sal\*12+nvl(comm, 0)
* 可以按照select语句中的列名的顺序值(序号)排序
* 如果要按照多列进行排序，则规则是先按照第一列排序，如果相同，则按照第二列排序；以此类推
* desc和asc只作用于最近的一列.

## 4 单行函数

单行函数:只对一行进行变换, 产生一个结果。函数可以没有参数, 但必须要有返回值。如：concat、nvl

* 操作数据对象
* 接受参数返回一个结果
* **只对一行进行变换**
* **每行返回一个结果**
* 可以转换数据类型
* 可以嵌套
* 参数可以是一列或一个值

### 4.1 字符函数

操作对象是字符串。

* 大致可分为两大类: 一类是大小写控制函数, 主要有lower、upper、initcap:

案例:

1 大小写转换lower和upper, 首字母大写initcap函数测试

select lower('HeLlo, WORld') 转小写, upper('HellO, woRld') 转大写, initcap('hello, world') 首字母大写 from dual;

* 另一类是字符控制函数: 有CONCAT、SUBSTR、LENGTH/LENGTHB、INSTR、LPAD | RPAD、TRIM、REPLACE
* substr(a, b)：在字符串a中，从第b位开始取(计数从1开始)，取到结尾

案例:

1 从helloworld字符串中的第3位开始后面的所有字符

select substr('helloworld', 3) from dual; ---🡪得到lloworld

* substr(a, b, c)：从a中，第b位开始，向右取c位。

案例:

1 从hello world字符串中, 从第3位开始取连续取5个字符

select substr('helloworld', 3, 5) from dual;

* length:字符数, lengthb:字节数：

案例:

select length('hello world') 字符数, lengthb('hello world') 字节数 from dual;

select length('哈喽我的') 字符数, lengthb('哈喽我的') 字节数 from dual;

注意:对于length函数一个汉字是一个字符, 对于lengthb函数,一个汉字占两个,

这两个函数对于普通字符串没有什么区别.

* instr: 在母串中查找子串, 找到返回下标, 计数从1开始, 没有返回0

案例

1 查找hello world字符串中, llo子串首次出现的下标位置

select instr('hello world', 'llo'), instr('hello world', 'www') from dual;

注意: 下标是从1开始的.

* lpad：左填充，

参1: 待填充的字符串，

参2: 填充后字符串的总长度(字节)，

参3: 填充什么

rpad: 右填充, 参数同lpad.

案例

1 select lpad('abcd', 10, '\*') 左, rpad('abcd', 10, '#') 右 from dual;

结果显示: \*\*\*\*\*\*abcd abcd######

2 select lpad('abcd', 15, '你')左填充, rpad('abcd', 16, '我') 右填充 from dual;

结果显示: 你你你你你abcd abcd我我我我我我

注意: lpad('abcd', 15, '你')由于abcd本身占4个字节, 需要填充11个字节才能

够15个字节, 但是一个汉字占两个字节, 所以填充了1个空格+5个汉字.

* trim:去掉前后指定的字符(不去掉中间的)

案例

1 去掉' hello world '两端的空格

select trim(' hello world ') from dual;

2 去掉Hello worldH字符串前后的H字符

select trim('H' from 'Hello worldH') from dual;

* replace:替换

案例

1 将hello world字符串中的l替换成\*

select replace('hello world', 'l', '\*') from dual;

2 删除字符串'hello world'中的字符'l'

select replace('hello world', 'l', '') from dual;

### 4.2 数值函数

* ROUND: 四舍五入

ROUND(45.926, 2) 45.93

* TRUNC: 截断

TRUNC(45.926, 2) 45.92

* MOD: 求余

MOD(1600, 300) 100

* 案例:

1 round(45.926, 2) :2表达的含义是小数点向右保留两位并四舍五入，第二个参数如果是0可以省略.

select round(45.926, 2), round(45.926, 1), round(45.926, 0) , round(45.926), round(45.926, -1) , round(45.926, -2) from dual;

2 trunc函数, 正数表示小数点之后, 负数表示小数点之前的位数, 0可以不写.

select trunc(45.926, 2), trunc(45.926, 1), trunc(45.926, 0), trunc(45.926), trunc(45.926, -1) , trunc(45.926, -2) from dual;

3 mod函数: 求余数

select mod(1600, 600) from dual; --🡪结果为100

4 ceil函数:向上取整

floor函数:向下取整

select ceil(19.23), floor(19.23) from dual;

### 4.3 时间函数

* 在Oracle中日期型的数据，既有日期部分，也有时间部分.

案例:

1 select sysdate from dual;

这里没有时间部分，因为系统默认的格式中不显示时间

2 select to\_char(sysdate, 'yyyy-mm-dd hh24:mi:ss') from dual;

显示日期到秒

3 select to\_char(sysdate, 'day') from dual ;

可以显示当前日期星期几

* 日期加、减数字得到的结果仍为日期。单位：天
* 显示 昨天、今天、明天

select (sysdate-1) 昨天, (sysdate) 今天, (sysdate + 1) 明天 from dual;

select to\_char(sysdate-1, 'yyyy-mm-dd') 昨天, to\_char(sysdate, 'yyyy-mm-dd') 今天, to\_char(sysdate+1, 'yyyy-mm-dd') 明天 from dual; 🡪转换日期格式;

* 既然一个日期型的数据加上或者减去一个数字得到的结果仍为日期，两个日期相减，得到的就是相差的天数。
* 计算员工的工龄, 显示从入职以来的总天数, 星期数, 总月数, 总年数

select ename, hiredate, (sysdate - hiredate) 天, (sysdate - hiredate)/7 星期, (sysdate - hiredate)/30 月, (sysdate - hiredate)/365 年 from emp;

* 日期和日期可以相减表示相隔多少天, 但是不允许相加, 两个日期相加没有意义, 日期只能和数字相加----🡪类似于两个指针相加没有意义一样.

select sysdate+hiredate from emp;

报错: ORA-00975: 不允许日期 + 日期

### 4.4 日期函数

上面求取员工工龄的结果不精确，如果想将其算准确，可以使用日期函数来做。

* months\_between

两个日期值相差的月数(精确值)

select ename, hiredate, (sysdate-hiredate)/30 一, months\_between(sysdate, hiredate) 二 from emp;

months\_between函数更精确, 在表示月份差的时候要使用months\_between函数.

* add\_months

在某个日期值上，加上多少的月，正数向后计算，负数向前计算。

计算95个月以后是哪年、哪月、那天：

select add\_months(sysdate, 95) 哪一天 from dual;

结果是: 2025/10/11

* last\_day

日期所在月的最后一天, 要么30, 31或者28

select last\_day(sysdate) from dual;

上个月的最后一天

select last\_day(add\_months(sysdate, -1)) from dual;

下一个月的最后一天

select last\_day(add\_months(sysdate, 1)) from dual;

* next\_day:指定日期的下一个日期

从当前时间算起, 下一个星期一的日期

select next\_day(sysdate, '星期一') from dual;

从特定日期得到之后的第一个星期几的日期

select next\_day(to\_date('2017-11-11', 'yyyy-mm-dd'), '星期三') from dual;

* round、trunc 对日期型数据进行四舍五入和截断

select round(sysdate, 'month'), round(sysdate, 'year') from dual;

select trunc (sysdate, 'month'), trunc(sysdate, 'year') from dual;

### 4.5 转换函数

在不同的数据类型之间完成转换, 如将"123"转换为123, 有隐式转换和显示转换之分。

* 隐式转换(由oracle数据库来完成)

select \* from emp where hiredate = '17-11月-81';

若是显示转换:

select \* from emp where to\_char(hiredate, 'DD-MON-RR') = '17-11月-81';

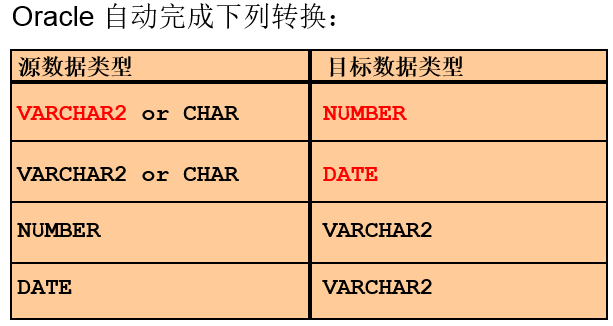
或者

select \* from emp where hiredate = to\_date('17-11月-81', 'DD-MON-RR');

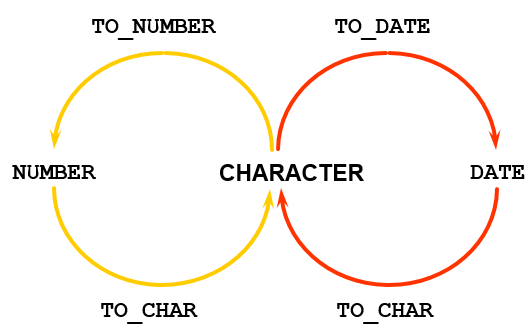
* 显示转换(通过转换函数来完成)

select to\_char(sysdate, 'yyyy-mm-dd hh24:mi:ss') from dual;

注意: 隐式转换，前提条件是：被转换的对象是可以转换的。（ABC→625 可以吗？）



显示转换：借助to\_char(数据，格式)、to\_number、to\_date函数来完成转换。



|  |  |  |
| --- | --- | --- |
| **格式** | **说明** | **举例** |
| YYYY | Full year in numbers | 2011 |
| YEAR | Year spelled out(年的英文全称) | twenty eleven |
| MM | Two-digit value of month 月份（两位数字） | 04 |
| MONTH | Full name of the month（月的全称） | 4月 |
| DY | Three-letter abbreviation of the day of the week(星期几) | 星期一 |
| DAY | Full name of the day of the week | 星期一 |
| DD | Numeric day of the month | 02 |

如果隐式转换和显示转换都可以使用，应该首选哪个呢？

注意：如果隐式、显示都可以使用，应该首选显示，这样可以省去oracle的解析过程。

练习：在屏幕上显示如下字符串：

2015-05-11 16:17:06 今天是 星期一

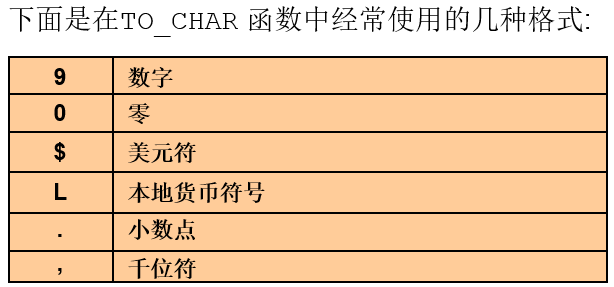
select to\_char(sysdate, 'yyyy-mm-dd hh24:mi:ss "今天是" day') from dual;

说明: 在固定的格式里加入自定义的格式，是可以的，必须要加””。

反向操作：已知字符串'2015-05-11 15:17:06 今天是 星期一'转化成日期.

使用to\_date函数将字符串转换成date类型

select to\_date('2015-05-11 15:17:06 今天是 星期一', 'yyyy-mm-dd hh24:mi:ss "今天是" day') from dual;



案例:

1查询员工的薪水: 使用2位小数, 本地货币代码, 千位符

select to\_char(sal, 'L9,999.99') from emp;

特别注意:'L9,999.99'之间没有空格

2 将￥2,975.00转化成数字：

select to\_number('￥2,975.00', 'L9,999.99') 转成数字 from dual;

### 4.6 通用函数(了解)

* 这些函数**适用于任何数据类型，同时也适用于空值**：
* NVL (expr1, expr2)
* NVL2 (expr1, expr2, expr3)
* NULLIF (expr1, expr2)
* COALESCE (expr1, expr2, ..., expr*n*)
* nvl2:是nvl函数的增强版。 nvl2(a, b, c) 当a = null 返回 c, 否则返回b

使用nvl2求员工的年收入：

select empno, ename, sal, sal\*12, sal \* 12 + nvl2(comm, comm, 0) 年薪 from emp;

* nullif: nullif(a, b) 当 a = b 时返回null, 不相等的时候返回a值。

select nullif('hello', 'hello') from dual; 返回空

select nullif('hello', 'world') from dual; 返回hello

select ename, nullif(comm, null) from emp;

* coalesce:coalesce(a, b, c, …, n) 从左向右找参数中第一个不为空的值。

select comm, sal, coalesce(comm, sal) 结果值 from emp;

### 4.7 条件表达式

例子：老板打算给员工涨工资, 要求：

总裁(PRESIDENT)涨1000, 经理(MANAGER)涨800, 其他人涨400. 请将涨前, 涨后的薪水列出。

涨后的薪水是根据job来判断的

思路： if 是总裁('PRESIDENT') then + 1000

else if 是经理('MANAGER') then + 800

else + 400

但是在SQL中无法实现if else 逻辑。当有这种需求的时候，可以使用case 或者 decode

case:是一个表达式，其语法为：

**CASE *expr* WHEN *comparison\_expr1* THEN *return\_expr1***

**[WHEN *comparison\_expr2* THEN *return\_expr2***

**WHEN *comparison\_exprn* THEN *return\_exprn***

**ELSE *else\_expr*]**

**END**

SQL> select ename, job, sal 涨前薪水,

case job

when 'PRESIDENT'

then sal+1000

when 'MANAGER'

then sal+800

else

sal + 400

end 涨后薪水

from emp;

注意语法：when then 与下一个when then以及end之间没有“,”分割符, 可以将when当成if.

decode：是一个函数，其语法为：

**DECODE(*col|expression, search1, result1***

**[*, search2, result2,...,*]**

**[*, default*])**

除第一个和最后一个参数之外，中间的参数都是成对呈现的 (参1, 条件, 值, 条件, 值, …, 条件, 值, 尾参)

select ename, job, sal 涨前薪水, decode(job, 'PRESIDENT', sal + 1000,

'MANAGER', sal + 800,

sal + 400) as 涨后薪水

from emp;

## 5 分组函数

### 5.1分组函数

多行函数也叫组函数，本章学习目标:

* 了解组函数。
* 描述组函数的用途。
* 使用GROUP BY 子句数据分组。
* 使用HAVING 子句过滤分组结果集。

分组函数作用于一组数据，并对一组数据返回一个值

常用的有5个函数: avg、count、max、min、sum操作的是一组数据，返回一个结果。

案例

1 求员工的工资总额

select sum(sal) from emp;

2 求员工人数

select count(\*) from emp;

3 平均工资

select sum(sal)/count(\*) 方式一, avg(sal) 方式二 from emp;

方式一和方式二结果一样, 当有空值得时候结果有可能不一样。如：奖金。

4 求员工的平均奖金

select sum(comm)/count(\*) 方式一, sum(comm)/count(comm) 方式二, avg(comm) 方式三 from emp;

结果：方式一结果不同，方式二 和 方式三结果一样。

说明: avg(comm)是求comm有值的几个的平均值, 跟方式二是相同的.

NULL空值：组函数都有自动滤空功能(忽略空值)，所以：

select count(\*), count(comm) from emp; 执行结果不相同。

说明: count(comm)返回不为空的comm的总个数

如何屏蔽组函数的滤空功能?

select count(\*), count(nvl(comm,0)) from emp;

但是实际应用中, 结果为14和结果为4都有可能对，看问题本身是否要求统计空值。

5 count函数

求emp表的部门总个数, 如果要求不重复的个数, 使用distinct。

select count(distinct deptno) from emp;

求emp表中的工种的总个数

select count(distinct job) from emp;

### 5.2 分组数据

分组数据使用group by关键字.

按照group by 后给定的表达式，将from后面的table进行分组。针对每一组，使用组函数, 即先分组, 再分组统计.

案例

1 查询“部门”的平均工资：

分析: 结合select \* from emp order by deptno; 结果分析分组

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

上SQL语句可以抽象成：select a, 组函数(x) from 表 group by a; 这样的格式。

如果select a, b, 组函数(x) …… group by 应该怎么写？

注意: 在select列表中所有没有包含在组函数中的列, 都必须在group by的后面出现.所以上面的问题应该写成group by a, b; 没有b就会出错, 不会执行sql语句。但, 反之可以。group by a，b，c; c可以不出现在select语句中。

group by后面有多列的情况：

2查询部门内部不同职位的平均工资:

select deptno, job, avg(sal) from emp group by deptno, job order by 1;

分析该SQL的作用：

因为deptno, job 两列没有在组函数里面，所以必须同时在group by后面。

该SQL的语义:按部门, 不同的职位统计平均工资。先按第一列分组, 如果第一列相同, 再按第二列分组, 所以查询结果中，同一部门中没有重复的职位。

关于分组函数常见的问题:

1 在select后面出现的列, 该列没有出现在分组函数中, 但是未出现在group by子句中.

如: select deptno, job, avg(sal) from emp group by deptno;

----job未出现在group by子句中

报错: ORA-00979: 不是 GROUP BY 表达式.

2 select count(\*) from emp;

分析: count是分组函数, 但是select后面没有其他列, 所以不必非出现group by子句,

该sql语句意思是统计emp表中员工总数;

当然: select count(\*) from emp group by deptno; 也对, 但是查询出来的结果并不知道是

哪个组的总数.

### 5.3 Having

* 使用 HAVING 过滤分组:

1. 行已经被分组。

2. 使用了组函数。

3. 满足HAVING 子句中条件的分组将被显示。

其语法：

**SELECT *column*, *group\_function***

**FROM *table***

**[WHERE *condition*]**

**[GROUP BY *group\_by\_expression*]**

**[HAVING *group\_condition*]**

**[ORDER BY *column*];**

* 查询平均薪水大于2000的部门 ：

分析：该问题实际上是在分组的基础上过滤分组。

select deptno, avg(sal) from emp group by deptno having avg(sal)>2000;

注意：having后面不能使用别名, 可以使用函数

* 特别注意: 不能在 WHERE 子句中使用组函数.
* 可以在 HAVING 子句中使用组函数。

从功能上讲，where和having都是将满足条件的结果进行过滤。但是差别是where子句中不能使用组函数, 所以上句中的having不可以使用where代替。

* 求10号部门的平均工资：

分析：在上一条的基础上，having deptno=10;

select deptno, avg(sal) from emp group by deptno having deptno=10;

使用where也可以做这件事

select deptno, avg(sal) from emp where deptno=10 group by deptno;

在子句中没有使用组函数的情况下，where、having都可以，应该怎么选择？

SQL优化： 尽量采用where。

如果有分组的话，where是先过滤再分组，而having是先分组再过滤。当数据量庞大如1亿条，where优势明显。

## 6 多表查询

理论基础：——笛卡尔集

笛卡尔集的行数 = table1的行数 x table2的行数

笛卡尔集的列数 = table1的列数 + table2的列数



在操作笛卡尔集的时候，应该避免使用“笛卡尔全集”，因为里面含有大量错误信息。

多表查询就是按照给定条件(连接条件)，从笛卡尔全集中选出正确的结果。

根据连接条件的不同可以划分为：等值链接、不等值链接、外链接、自连接

* **Oracle** 连接**:**
  + **Equijoin：等值连接**
  + **Non-equijoin：不等值连接**
  + **Outer** **join：外连接**
  + **Self** **join：自连接**
* **SQL:** **1999**
  + **Cross** **joins**
  + **Natural** **joins**
  + **Using** **clause**
  + **Full** **or** **two** **sided** **outer joins**

### 6.1 等值连接：

从概念上，区分等值连接和不等值连接非常简单，只需要辨别where子句后面的条件，是“=”为等值连接。不是“=”为不等值连接。

* 查询员工信息：员工号 姓名 月薪和部门名称

分析：这个问题涉及emp(员工号，姓名，月薪) 和dept(部门名称)两张表 ——即为多表查询。

通常在进行多表查询的时，会给表起一个别名，使用“别名.列名”的方式来获取数据，直接使用“表名.列名”语法上是允许的，但是实际很少这样用。

如果：select e.empno, e.ename, e.sal, e.deptno, d.dname, d.deptno from emp e, dept d;

直接得到的是笛卡尔全集。其中有错误结果。所以应该加 where 条件进行过滤.

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

如果有N个表，where后面的条件至少应该有N-1个, 才会不产生笛卡尔积.

### 6.2 不等值连接：

将上面的问题稍微调整下，查询员工信息：员工号 姓名 月薪 和 薪水级别(salgrade表)

select \* from salgrade;

看到员工总的薪水级别，共有5级，员工的薪水级别应该满足 >=当前级别的下限，<=该级别的上限：

过滤子句应该： where e.sal >= s.losal and e.sal <= s.hisal;

select e.empno, e.ename, e.sal, s.grade from emp e, salgrade s where e.sal >= s.losal and e.sal <= s.hisal;

更好的写法应该使用between…and:

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

where e.sal between s.losal and s.hisal

order by 1;

### 6.3 外链接：

* **按部门**统计员工人数，显示： 部门号 部门名称 人数

分析:

人数：一定是在emp表中，使用count()函数统计emp表中任一非空列均可。

部门名称：在dept表dname中, 直接读取即可。

部门号：任意，两张表都有, 两个表的联系是deptno.

所以:

select d.deptno 部门号, d.dname 部门名称, count(e.empno) 人数 from emp e, dept d where e.deptno=d.deptno group by d.deptno, d.dname;

注意：由于使用了组函数count()，所以组函数外的d.deptno和d.dname必须放到group by后。

得到查询结果，但是select \* from dept发现40号部门没有显示出来，原因是40号部门没有员工，where没满足。结果不对，40号部门没有员工，应该在40号部门位置显示0。

我们希望： 在最后的结果中，包含某些对于where条件来说不成立的记录 （外链接的作用）

* 左外链接：当 where e.deptno=d.deptno 不成立的时候，=左边所表示的信息，仍然被包含。

写法：与叫法相反：where e.deptno=d.deptno(+)

* 右外链接：当 where e.deptno=d.deptno 不成立的时候，=右边所表示的信息，仍然被包含。

写法：依然与叫法相反：where e.deptno(+)=d.deptno

以上我们希望将没有员工的部门仍然包含到查询的结果当中。因此应该使用外链接的语法。

写法1:

select d.deptno 部门号, d.dname 部门名称, count(e.empno) 人数 from emp e, dept d where e.deptno(+)=d.deptno group by d.deptno, d.dname;

写法2:

select d.deptno 部门号, d.dname 部门名称, count(e.empno) 人数 from emp e, dept d where d.deptno = e.deptno(+) group by d.deptno, d.dname;

这样就可以将40号部门包含到整个查询结果中。人数是0

注意：不能使用count(e.\*), 应该是某个表的具体的列.

思考: 能否使用count(\*), count(\*)与count(e.empno)有什么不同???

### 6.4 自连接：

核心，通过表的别名，将同一张表视为多张表。

例如: 查询员工信息：xxx的老板是 yyy

分析：执行select \* from emp; 发现，员工的老板也在员工表之中，是一张表。要完成多表查询我们可以假设，有两张表，一张表e(emp)只存员工、另一张表b(boss)只存员工的老板。—— from e, b;

老板和员工之间的关系应该是：where e.mgr=b.empno (即:员工表的老板 = 老板表的员工)

select e.ename || ' 的老板是 ' || b.ename from emp e, emp b where e.mgr=b.empno;

执行, 发现结果正确了, 但是KING没有显示出来. KING的老板是他自己. 应该怎么显示呢?

使用外连接:

select e.ename || ' 的老板是 ' || nvl(b.ename, '他自己' )

from emp e, emp b

where e.mgr=b.empno(+);

使用concat函数应该怎么做呢??

select concat(e.ename, concat(' 的老板是 ', nvl(b.ename, '他自己' )))

from emp e, emp b

where e.mgr = b.empno(+);

## 7 子查询

子查询语法很简单，就是select 语句的嵌套使用, 即sql嵌套sql。

查询工资比SCOTT高的员工信息.

分析：两步即可完成

1. 查出SCOTT的工资

select ename, sal from emp where ename='SCOTT'; 结果为3000

1. 查询比3000高的员工

select \* from emp where sal>3000;

通过两步可以将问题结果得到。子查询，可以将两步合成一步。

**——子查询解决的问题：问题本身不能一步求解的情况。**

select \*

from emp

where sal > (select sal

from emp

where ename='SCOTT');

子查询语法格式：

**SELECT *select\_list***

**FROM *table***

**WHERE *expr operator***

**(SELECT *select\_list***

**FROM *table*);**

本章学习目标:

* 描述子查询可以解决的问题
* 定义子查询(子查询的语法)
* 列出子查询的类型。
* 书写单行子查询和多行子查询。

### 7.1 定义子查询 需要注意的问题

1. 合理的书写风格 (如上例，当写一个较复杂的子查询的时候，要合理的添加换行、缩

进)

2. 小括号( )

3. 主查询和子查询可以是不同表，只要子查询返回的结果主查询可以使用即可

4. 可以在主查询的where、select、having、from后都可以放置子查询

5. 不可以在主查询的group by后面放置子查询 (SQL语句的语法规范)

6. **强调：**在from后面放置的子查询(\*\*\*), from后面放置是一个集合(表、查询结果)

7. 一般先执行子查询(内查询)，再执行主查询(外查询)；但是相关子查询除外

8. 一般不在子查询中使用order by, 但在Top-N分析问题中，必须使用order by

9. 单行子查询只能使用单行操作符；多行子查询只能使用多行操作符

10. 子查询中的null值

### 7.2 主、子查询在不同表间进行。

查询部门名称是“SALES”的员工信息

主查询：查询员工信息。select \* from emp;

子查询：负责根据部门名称(在dept表中)得到部门号。

select deptno from dept where dname='SALES'

SQL> select \*

from emp

where deptno= (select deptno

from dept

where dname='SALES')

主查询, 查询的是员工表emp, 子查询, 查询的是部门表dept。是两张不同的表。

将该问题使用“多表查询”解决

select e.\* from emp e, dept d where e.deptno=d.deptno and d.dname='SALES';

两种方式哪种好呢？

SQL优化: 理论上，既可以使用子查询，也可以使用多表查询，尽量使用“多表查询”。子查询有2次from, 与数据库服务的交互多.

不同数据库处理数据的方式不尽相同，如Oracle数据库中，子查询地位比较重要，做了深入的优化。有可能实际看到结果是子查询快于多表查询。

### 7.3 在主查询的where select having from 放置子查询

子查询可以放在select后，但，要求该子查询必须是单行子查询：(该子查询本身只返回一条记录，2+叫多行子查询)

select empno, ename, (select dname from dept) 部门 from emp;

注意：SQL中没有where是不可以的，那样是多行子查询。--但这是不符合实际情况的.

应该:

select empno, ename, (select dname from dept where deptno = 10) 部门

from emp

where deptno = 10;

进一步理解查询语句，实际上是在表或集合中通过列名来得到行数据，子查询如果是多行，select无法做到这一点。

在 having 后 和 where 类似。但需注意在where后面不能使用组函数。

在having后面使用子查询的例子:

查询部门平均工资高于30号部门平均工资的部门和平均工资

select deptno, avg(sal)

from emp

group by deptno

having avg(sal) > (select avg(sal) from emp where deptno = 30);

### 7.4 在from后面放置的子查询(\*\*\*)

表, 代表一个数据集合、查询结果(SQL)语句本身也代表一个集合, 把查询结果看成一个表.

查询员工的姓名、薪水和年薪：

说明：该问题不用子查询也可以完成。但如果是一道填空题：

select \* from \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

因为显示的告诉了，要使用select \*, 所以只能:

select \* from (select ename, sal, sal\*12 年薪 from emp);

将select 语句放置到from后面，表示将select语句的结果，当成表来看待. 这种查询方式在Oracle语句中使用比较频繁.

### 7.5 单行子查询只能使用单行操作符；多行子查询只能使用多行操作符

#### 7.5.1单行子查询：

单行子查询就是该条子查询执行结束时, 只返回一条记录(一行数据)。

使用单行操作符:

=、>、>=、<、<=、<>或者!=

单行子查询:

1. 单行子查询, 只能使用单行操作符

2. 在一个主查询中可以有多个子查询。

3. 子查询里面可以嵌套多层子查询。

4. 子查询也可以使用组函数。子查询也是查询语句，适用于前面所有知识。

案例:

1 查询emp表部门编号为10且高于本部门的平均工资的员工信息

select \* from emp where sal>(select avg(sal) from emp where deptno=10) and deptno=10;

2 查询emp表中与SMITH职位相同的员工信息

select \* from emp where job = (select job from emp where ename='SMITH');

3 查询薪水低于本公司的平均薪水且职位与SMITH职位相同的所有员工信息

select \* from emp where sal<(select avg(sal) from emp) and job=(select job from emp where ename='SMITH');

4 查询emp表中最低工资员工和最高工资员工的信息

select \* from emp where sal=(select max(sal) from emp) or sal=(select min(sal) from emp);

5 思考??,下列sql语句正确吗?

select \* from emp where sal=(select min(sal) from emp group by deptno);

注意: =、>、>=、<、<=、<>或者!=是单行操作符, 而后面的子查询会返回多条记录,所以会报错, 要解决这个问题要使用多行子查询.

#### 7.5.2多行子查询：

子查询返回2条记录以上就叫多行。

多行操作符有：

**IN**  **等于列表中的任意一个**

**ANY 和子查询返回的任意一个值比较**

**ALL 和子查询返回的所有值比较**

* IN(表示在集合中)：

解决上面的那个问题?

select \* from emp where sal in (select min(sal) from emp group by deptno);

* 查询部门名称为SALES和ACCOUNTING的员工信息。

分析：部门名称在dept表中，员工信息在emp表中,子查询应先去dept表中将SALES和ACCOUNTING的部门号得到，交给主查询得员工信息.

select \*

from emp

where deptno in

(select deptno from dept where dname in ('SALES', 'ACCOUNTING'));

也可以使用多表查询来解决该问题：

select e.\*

from emp e, dept d

where e.deptno = d.deptno

and (d.dname = 'SALES' or d.dname = 'ACCOUNTING');

红色语句也可以用 d.dname in ('SALES ', 'ACCOUNTING ')

这种解决方式，注意使用()来控制优先级。

如果查询不是这两个部门的员工，只要把in → not in就可以了，注意不能含有空值。

* ANY(表示和集合中的**任意**一个值比较)：
* 查询薪水比30号部门任意一个员工高的员工信息：

分析：首先查出30号部门的员工薪水的集合，然后>它就得到了该员工信息。

select \* from emp where sal > (select sal from emp where deptno=30); 正确吗？

这样是错的，子句返回多行结果。而‘>’是单行操作符。——应该将‘>’替换成‘> any’

实际上>集合的任意一个值，就是大于集合的最小值。

select \* from emp where sal > any(select sal from emp where deptno=30);

* 若将这条语句改写成单行子查询应该怎么写呢？

select \* from emp where sal > (select min(sal) from emp where deptno=30);

* ALL(表示和集合中的所有值比较)：

查询薪水比30号部门所有员工高的员工信息。

select \* from emp where sal > all (select sal from emp where deptno=30);

同样，将该题改写成单行子句查询：

select \* from emp where sal > (select max(sal) from emp where deptno=30);

对于any 和 all 来说，究竟取最大值还是取最小值，不一定。将上面的两个例子中的“高”换成“低”，any和all就各自取相反的值了。

### 子查询中null

判断一个值等于、不等于空，不能使用=和!=号，而应该使用is 和 not。

如果集合中有NULL值，不能使用not in。如： not in (10， 20， NULL)，但是可以使用in。为什么呢？

先看一个例子：

查询不是老板的员工信息：

分析：不是老板就是树上的叶子节点。在emp表中有列mgr，该列表示该员工的老板的员工号是多少。那么，如果一个员工的员工号在这列中，那么说明这员工是老板，如果不在，说明他不是老板。

select \* from emp where empno not in (select mgr from emp);

但是运行没有结果，因为有NULL

查询是老板的员工信息：只需要将not去掉。

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

还是我们之前null的结论：in (10, 20, null) 可以，not in (10, 20, null) 不可以

select \* from emp where deptno in (10, 20, null);-----可以

select \* from emp where deptno not in (10, 20, null);-----不可以

例如：a not in(10, 20, NULL) 等价于 (a != 10) and (a != 20) and (a != NULL)

因为, not in操作符等价于 !=All，最后一个表达式为假，整体假;

而a in (10, 20, NULL)等价于(a = 10) or (a = 20) or (a = null)只要有一个为真即为真。

in 操作符等价于 = Any

select \* from emp where deptno = any(10, 20, null); 等价于

select \* from emp where deptno in (10, 20, null);

继续，查询不是老板的员工信息, 只要将空值去掉即可。

select \* from emp where empno not in (select mgr from emp where mgr is not null);

注意: not in 后面的结合中不能有null

### 一般不在子查询中使用order by

一般情况下，子查询使用order by或是不使用order by对主查询来说没有什么意义。子查询的结果给主查询当成集合来使用，所以没有必要将子查询order by。

**但，在Top-N分析问题中，必须使用order by**

### 一般先执行子查询，再执行主查询

含有子查询的SQL语句执行的顺序是，先子后主。

**但，相关子查询例外.**

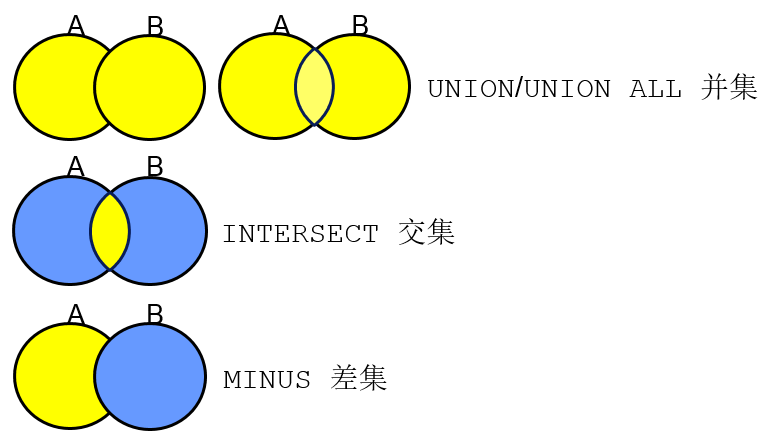
## 8 集合运算

* 查询部门号是10和20的员工信息? 有三种方法
* select \* from emp where deptno in(10, 20);
* select \* from emp where deptno=10 or deptno=20;
* 集合运算：
  + - Select \* from emp where deptno=10 加上
    - Select \* from emp where deptno=20

集合运算所操作的对象是两个或者多个集合，而不再是表中的列(select一直在操作表中的列)

### 8.1 集合运算符

集合运算的操作符。A∩B、A∪ B、A - B



select \* from emp where deptno=10

union

select \* from emp where deptno=20;

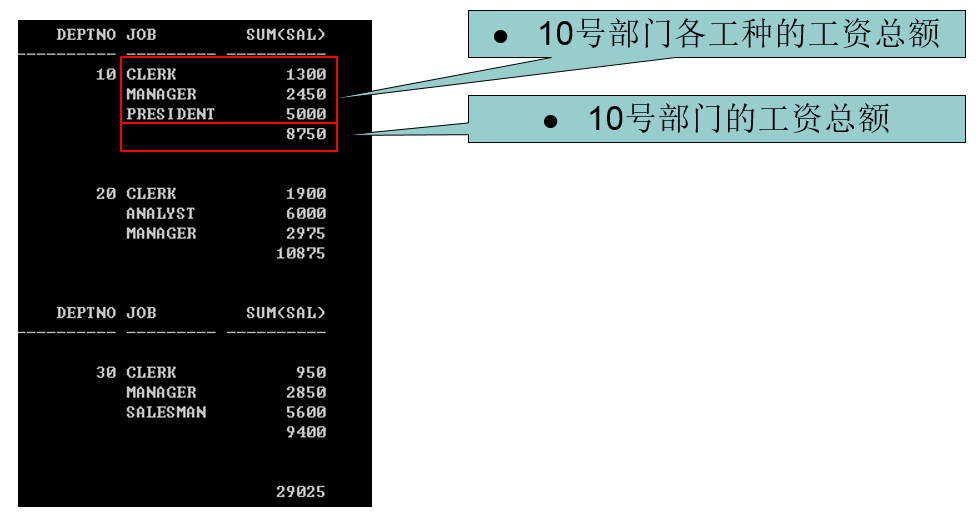
union和union all的区别: union会去掉重复的, 而union all会全部显示

注意：这是一条SQL语句。

### 8.2 集合运算需要注意的问题：

* **参与运算的各个集合必须列数相同，且对应每个列的类型一致**。
* 采用第一个集合的表头作为最终使用的表头.
* 可以使用括号()先执行后面的语句。

问题：按照部门统计各部门不同工种的工资情况，要求按如下格式输出：



分析SQL执行结果。

第一部分数据是按照deptno和job进行分组；select 查询deptno、job、sum(sal)

第二部分数据是直接按照deptno分组即可，与job无关；select 只需要查询deptno，sum(sal)

第三部分数据不按照任何条件分组，即group by null；select 查询sum(sal)

所以，整体查询结果应该= group by deptno,job + group by deptno + group by null

按照集合的要求,必须列数相同,类型一致,所以写法如下,使用null强行占位!

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

union

select deptno,to\_char(null),sum(sal) from emp group by deptno

union

select to\_number(null),to\_char(null),sum(sal) from emp;

交集和差集与并集类似，也要注意以上三点。只不过算法不同而已。

需要注意:集合运算的性能一般较差.

SQL的执行时间：

set timing on/off 默认是off

# 9 数据处理

### SQL语言的类型：

1. 数据库中，称呼增删改查，为DML语句。(Data Manipulation Language 数据操纵

语言)，就是指代：insert、update、delete、select这四个操作。

2. DDL语句。(Data Definition Language 数据定义语言)。

如：truncate table(截断/清空 一张表)

create table(表)、create view(视图)、create index(索引)、create sequence(序列)、

create synonym(同义词)、alter table、drop table。

3. DCL语句。DCL（Data Control Language数据控制语言）如：

commit(提交)、rollback(回滚)

### 插入数据insert：

* 使用 INSERT 语句向表中插入数据。其语法为：

**INSERT INTO *table* [(*column* [*, column...*])]**

**VALUES *(value* [*, value...*]);**

* 如果：values后面的值，涵盖了表中的所有列，那么table的列名可以省略不写。

desc emp; 查看员工表的结构，得到所有的列名。

insert into emp values (1001, 'Tom', 'Engineer', 7839, sysdate, 5000, 200, 10 )

insert into emp values (1005, 'Bone', 'Raphealy', 7829, to\_date('17-12月-82', 'DD-MON-RR'), NULL, 300, 20);

* 如果:插入的时候没有插入所有的列, 就必须显式的写出这些列的名字。

insert into emp(empno, ename, sal, deptno) values(1002, 'Marry', 6000, 20);

注意：字符串和日期都应该使用 ' '号引用起来.

* 没有写出的列自动填NULL, 这种方式称之为“隐式插入空值”。
* 显示插入空值: insert into emp(empno, ename, sal) values(1003, 'Jim', null);

#### “&” 地址符：

* 在insert语句中使用&可以让用户输入值:

insert into emp(empno, ename, sal, deptno) values(&empno, &ename, &sal, &deptno);

理论上“&”后面的变量名任意，习惯上一般与前面的列名相同，赋值的时候清楚在给谁赋值。

当再次需要插入新员工的时候直接输入“/”就可以继续输入新员工的值, /表示重复刚刚执行过的sql语句.

* 可以在DML的任意一个语句中输入“&”,

select empno, ename, sal, &t from emp ;

执行时，会提示你输入要查询的列名。当输入不同的列名时，显示不同的执行结果。

select \* from &t; 修改t的值，可以查看不同表。

* 总结: &类似于c语言中的宏替换, 只是进行简单的字符串替换操作, 但是select关键字不能用&指代. 如 &t \* from emp; 而from, 表名, where 是可以用&指代的, 如: select \* &t emp; select \* from &t; select \* from emp &t sal>2000;但是通常情况都是替换值.

#### 批处理：

一次插入多条数据, 使用一个新创建的表用来测试.

* 创建一张与emp完全相同的表，用于测试。

create table emp10 as select \* from emp where 1=2;

* select \* from tab;

select \* from emp10;

desc emp10;

* 一次性将emp表中所有10号部门的员工, 放到新表emp10中来。

insert into emp10 select \* from emp where deptno=10 ;

* 一次性将emp表中的指定列插入到表emp10中。

注意：insert的列名, 要和select的列名一致

insert into emp10(empno, ename, sal, deptno)

select empno, ename, sal, deptno from emp where deptno=10;

注意没有values关键字了, 但列名必须一一对应.

总结: 子查询可以出现在DML的任何语句中.

#### 更新数据update

* 格式: update 表名 set col=值 where condtion
* 对于更新操作来说，一般会有一个“where”条件，如果没有这限制条件，更新的就是整张表。

update emp10 set sal=4000, comm=300 where ename = 'CLARK';

注意：若没有where限定，会将所有的员工的sal都设置成4000，comm设置成300；

* 能否将某一列的值设置为null呢?

update emp10 set comm = null where empno=1000; ----可以

* 能否在查询的时候where条件中指定列的值为null吗?

select \* from emp where comm = null; ----不可以

## 10 删除数据delete

* 格式: delete from 表名 where condtion
* delete from emp10 where empno=7782;

注意: 如不加“where”会将整张表的数据删除。

“from”关键字在Oracle中可以省略不写，但MySQL中不可以;

但在使用的时候建议还是加上from.

### delete 和 truncate的区别：

1. delete 逐条删除表“内容”，truncate 先摧毁表再重建。

(由于delete使用频繁，Oracle对delete优化后delete快于truncate)

2. delete 是DML语句，truncate 是DDL语句。

DML语句可以闪回(flashback)，DDL语句不可以闪回。

（闪回： 做错了一个操作并且commit了，对应的撤销行为。了解）

3. 由于delete是逐条操作数据，所以delete会产生碎片，truncate不会产生碎片。

（同样是由于Oracle对delete进行了优化，让delete不产生碎片）。

两个数据之间的数据被删除，删除的数据——碎片，整理碎片，数据连续，行移动

4. delete不会释放空间，truncate 会释放空间

用delete删除一张10M的表，空间不会释放。而truncate会。所以当确定表不再

使用，应truncate

5. delete可以回滚rollback, truncate不可以回滚rollback。

### delete和truncate的时效性

【做实验sql.sql】：验证delete和truncate的时效性。 终端里@c:\sql.sql 可以执行脚本sql.sql

语句执行时间记录开关：set timing on/off

回显开关：set feedback on/off

【测试步骤】： 1. 关闭开关： SQL> set timing off; SQL> set feedback off;

2. 使用脚本创建表： SQL> @c:\sql.sql

3. 打开时间开关： SQL> set timing on;

4. 使用delete删除表内容： SQL> delete from testdelete;

5. 删除表： SQL>drop table testdelete purge;

6. 关闭时间开关： SQL> set timing off;

7. 使用脚本创建表： SQL> @c:\sql.sql

8. 打开时间开关： SQL> set timing on;

9. 使用truncate删除表内容： SQL> truncate table testdelete;

### 事务

联想现实生活中的银行转账业务, 从A账户把钱转给B账户.

* 数据库事务，是由有限的数据库操作序列组成的逻辑执行单元，这一系列操作要么全部执行，要么全部放弃执行。
* 数据库事务由以下的部分组成:
* **一个或多个DML 语句**
* 一个 DDL(Data Definition Language – 数据定义语言) 语句
* 一个 DCL(Data Control Language – 数据控制语言) 语句

事务的特点：要么都成功，要么都失败。

#### 事务的特性

* 事务4大特性(ACID) ：原子性、一致性、隔离性、持久性。
* 原子性 (Atomicity)：事务中的全部操作在数据库中是不可分割的，要么全部完成，要么均不执行。
* 一致性 (Consistency)：几个并行执行的事务, 其执行结果必须与按某一顺序串行执行的结果相一致。
* 隔离性 (Isolation)：事务的执行不受其他事务的干扰，当数据库被多个客户端并发访问时，隔离它们的操作，防止出现：脏读、幻读、不可重复读。
* 持久性 (Durability)：对于任意已提交事务，系统必须保证该事务对数据库的改变不被丢失，即使数据库出现故障。
* 事务的起始标志：oracle中自动开启事务，以DML语句为开启标志。
* 执行一个增删改查语句, 只要没有提交commit和回滚rollback, 操作都在一个事务中.
* 事务的结束标志: 提交、回滚都是事务的结束标志。
* 提交:
* 显示提交: commit
* 隐式提交

1. 有DDL语句，如：create table除了创建表之外还会隐式提交Create

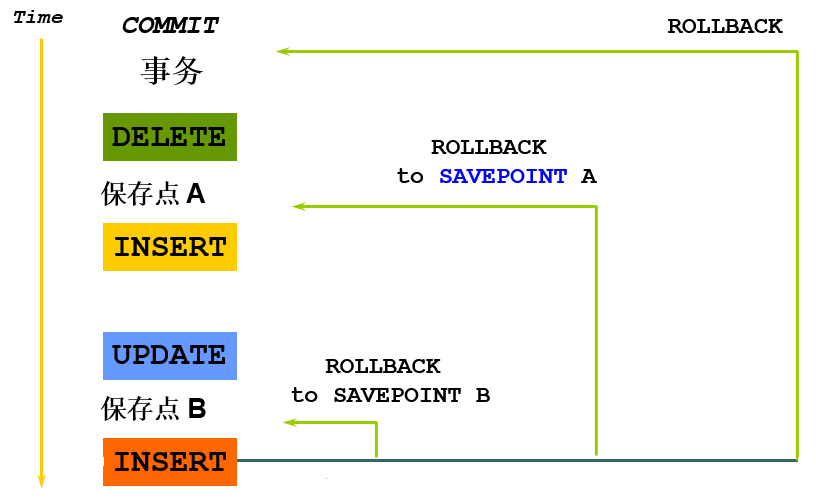
之前所有没有提交的DML语句。

2. 正常退出(exit / quit)

* + - 回滚:
* 显示回滚: rollback
* 隐式回滚: 掉电、宕机、非正常退出。

#### 控制事务

* 保存点（savepoint）可以防止错误操作影响整个事务，方便进行事务控制。



【示例】：1. create table testsp ( tid number, tname varchar2(20));

DDL语句会隐式commit之前操作.

2. set feedback on; 打开回显

3. insert into testsp values(1, 'Tom');

4. insert into testsp values(2, 'Mary');

5. savepoint aaa;

6. insert into testsp values(3, 'Moke'); 故意将“Mike”错写成“Moke”。

7. select \* from testsp; 三条数据都显示出来。

8. rollback to savepoint aaa; 回滚到保存点aaa

9. select \* from testsp; 发现表中的数据保存到第二条操作结束的位置

需要注意，前两次的操作仍然没有提交。如操作完成应该显示的执行 commit 提交。

savepoint主要用于在事务上下文中声明一个中间标记, 将一个长事务分隔为多个较小的部分, 和我们编写文档时, 习惯性保存一下一样, 都是为了防止出错和丢失。如果保存点设置名称重复，则会删除之前的那个保存点。一但commit之后，所有的savepoint将失效。

#### 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:
* **脏读**: 对于两个事物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
* **不可重复读**: 对于两个事物 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.
* **幻读**: 对于两个事物 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.
* 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.

SQL99定义4中隔离级别：

1. Read Uncommitted 读未提交数据。

2. Read Commited 读已提交数据。 （Oracle默认）

3. Repeatable Read 可重复读。 （MySQL默认）

4. Serializable 序列化、串行化。 （查询也要等前一个事务结束）

Oracle支持的隔离级别： Read Commited（默认）和 Serializable，以及Oracle自定义的Read Only三种。

Read Only：由于大多数情况下，在事务操作的过程中，不希望别人也来操作，但是如果将别人的隔离级别设置为Serializable（串行），但是单线程会导致数据库的性能太差。是应该允许别人来进行read操作的。

## 11 创建和管理表

### 常见的数据库对象

* 数据库的对象: 经常使用的数据库对象有 表、视图、索引、序列、同义词等.
* **表 基本的数据存储集合，由行和列组成。**
* **视图 从表中抽出的逻辑上相关的数据集合。**
* **序列 提供有规律的数值。**
* **索引 提高查询的效率**
* **同义词 给对象起别名**

### 表的基本操作

基本的数据存储集合，由行和列组成。表名和列名遵循如下命名规则：

* 必须**以字母开头**
* 必须在 1–30 个字符之间
* 必须只能包含 A–Z, a–z, 0–9, \_, $, 和 **#**
* 必须不能和用户定义的其他对象重名
* 必须不能是Oracle 的保留字
* Oracle默认存储是都存为大写
* 数据库名只能是1~8位, datalink可以是128位, 和其他一些特殊字符

#### 创建表

创建一张表必须具备：1. Create Table的权限 2. 存储空间。我们使用的scott/hr用户都具备这两点。

create table test1 (tid number, tname varchar2(20), hiredate date default sysdate);

default的作用是, 当向表中插入数据的时候, 没有指定时间的时候, 使用默认值sysdate。insert into test1(tid, tname) values(11, 'wangwu');

插入时没有指定Hiredate列，取当前时间。

创建表时， 列所使用的数据类型：



rowid：行地址 ——伪列

select rowid, empno, deptno from emp;

看到该列存储的是一系列的地址(指针), 创建索引用.

分析，之前我们使用过的创建表的语句：

create table emp10 as select \* from emp where 1=2;

在这条语句中，“where 1=2”一定为假。所以是不能select到结果的，但是将这条子查询放到Create语句中，可以完成拷贝表结构的效果。最终emp10和emp有相同的结构。

如果, “where”给定的是一个有效的条件, 就会在创建表的同时拷贝数据。如:

create table emp20 **as** select \* from emp where deptno=20;

这样emp20在创建之初就有5条数据。

创建一张表，要求包含：员工号 姓名 月薪 年薪 年收入 部门名称

分析：[员工号 姓名 月薪 年薪 年收入]在emp表中, 部门名称在dept表中, 两个表是通过deptno部门编号联系起来的.

根据要求，涉及emp和dept两张表(至少有一个where条件)，并且要使用表达式来计算年收入和年薪。

create table empincome as ↓

select e.empno, e.ename, e.sal, e.sal\*12 annualsal, e.sal\*12+nvl(comm, 0) income, d.dname from emp e, dept d where e.deptno = d.deptno;

**必须要给表达式取别名(语法要求)**

由于此时的“where”条件是有效的条件，就会在创建表的同时拷贝数据。

创建“视图”的语法与上边表的语法、顺序几乎完全一样，只是将“table”→“view”即可。

#### 修改表

* ALTER TABLE
* 追加一列: 向test1表中加入新列 image 类型是blob

alter table test1 add image blob;

desc test1

* 修改一列: 将tname列的大小有20→40.

alter table test1 modify tname varchar2(40);

* 删除一列: 将刚加入的新列image删除.

alter table test1 drop column image;

* 重命名一列: 将列tname重命名为username.

alter table test1 rename column tname to username;

注意: 若是修改表的字段的长度, 若是增加长度没有问题, 若是减少字段的长度, 有可能会报错.

#### 删除表

* 当表被删除：
* 数据和结构都被删除
* 所有正在运行的相关事物被提交
* 所有相关索引被删除
* DROP TABLE 语句不能回滚，但是可以闪回

select \* from tab; 查看当前用户下有哪些表, 拷贝保存表名。

drop table testsp; 将测试保存点的表删除。

select \* from tab; 再次查询跟刚刚保存的表名比对，少了testsp，但多了另外一张命名复杂的表。

Oracle的回收站：

1. 查看回收站：show recyclebin (sqlplus 命令)那个复杂的命名即是testsp在回收站中的名字。

select \* from testsp; 这样是不能访问的。

select \* from "BIN$+vu2thd8TiaX5pA3GKHsng==$0" 要使用“回收站中的名字”

2. 清空回收站：purge recyclebin

drop table test1 purge 表示直接删除表，不经过回收站。

将表从回收站里恢复，涉及“闪回”的知识，作为了解性知识点。

将表从回收站闪回

flashback table t2 to before drop;

清空回收站

purge recyclebin;

注意：并不是所有的用户都有“回收站”，对于没有回收站的用户（管理员）来说，删除操作是不可逆的.

#### 重命名表

rename test1 to test8

Truncate Table：DDL语句 ——注意：不能回滚(rollback)

### 约束：

#### 约束的种类

1. Not Null 非空约束

例如：人的名字，不允许为空。

2. Unique 唯一性约束

例如：电子邮件地址，不可以重复。

3. Primary Key 主键约束

通过这个列的值可以唯一的确认一行记录，主键约束隐含Not null + Unique

4. Foreign Key 外键约束

例如：部门表dept和员工表emp，不应该存在不属于任何一个部门的员工。用来约束两张表的关系。

注意：如果父表的记录被子表引用的话，父表的记录默认不能删除。解决方法：

1） 先将子表的内容删除，然后在删除父表。

2） 将子表外键一列设置为NULL值，断开引用关系，然后删除父表。

无论哪种方法，都要在两个表进行操作。所以**定义外键**时，可以通过**references**指定如下参数：

ON DELETE CASCADE：当删除父表时，如发现父表内容被子表引用，级联删除子表引用记录。

ON DELETE SET NULL：当发现上述情况，先把子表中对应外键值置空，再删除父表。

多数情况下，使用SET NULL方法，防止子表列被删除，数据出错。

5. Check检查性约束

如：教室中所有人的性别；工作后薪水满足的条件。

新建一个测试表: test7

create table test7

(tid number,

tname varchar2(20),

gender varchar(6) check (gender in ('男', '女')),

sal number check (sal > 0)

)

说明:

check (gender in ('男', '女')) 检查插入的性别是不是‘男’或‘女’（单引号）。

check (sal > 0) 检查薪水必须是一个正数。

测试: insert into test7 values(1, 'Tom', '男', 1000); 没有问题.

如果插入: insert into test7 values(2, 'Mary', '啊', 2000); 会报错.

ORA-02290：违反检查约束条件SCOTT.SYS\_C005523, gender的值只能取’男’或者’女’.

其中的“SYS\_C005523”是约束的名字，由于在定义约束时没有显式指定，系统默认给起了这样一个名称。所以我们建议，创建约束的时候，自定义一个见名知意的约束名。

constraint: 使用该关键字，来给约束起别名。

#### 约束举例

【约束举例】：

create table student

(

sid number constraint student\_PK primary key, --学生Id主键约束

sname varchar2(20) constraint student\_name\_notnull not null, --学生姓名非空约束

email varchar2(20) constraint student\_email\_unique unique --学生邮件唯一约束

constraint student\_email\_notnull not null, --同时邮件可再设非空，没有,

age number constraint student\_age\_min check(age > 10), --学生年龄设置check约束

gender varchar2(6) constraint gender\_female\_or\_male check(gender in ('男', '女')),

deptno number constraint student\_FK references dept (deptno) ON DELETE SET NULL

);

在定义学生deptno列的时候，引用部门表的部门号一列作为外键，同时使用references设置级联操作

——当删除dept表的deptno的时候，将student表的deptno置空。

查看student表各列的约束

desc student;

student的建表语句:

select dbms\_metadata.get\_ddl('TABLE', 'STUDENT') from dual;

测试用例：

insert into student values(1, 'Tom', 'tom@126.com', 20, '男', 10); 正确插入表数据。

insert into student values(2, 'Tom', 'tom@126.com', 15, '男', 10);

违反student\_email\_unique约束。

insert into student values(3, 'Tom3', 'tom3@126.com', 14, '男',100 ); 违反完整约束条件 (SCOTT.STUDENT\_FK) - 未找到父项关键字

……

问题：是不是父表的所有列，都可以设置为子表的外键？作外键有要求吗？

外键：必须是父表的主键.

查看指定表（如student）的约束, 注意表名必须大写。

select constraint\_name, constraint\_Type, search\_condition

from user\_constraints where table\_name='STUDENT';

## 12 其它数据库对象

### 视图：

视图是一种常见数据库对象, 它是**从表中抽出的逻辑上相关的数据集合。**

所以：1. 视图基于表。2. 视图是逻辑概念。3. 视图本身没有数据。

#### 创建视图

创建语法与创建表类似，只需要将table → view即可:

create view empincomeview

as

select e.empno, e.ename, e.sal, e.sal\*12 annualsal, e.sal\*12+nvl(comm, 0) income, d.dname

from emp e, dept d

where e.deptno = d.deptno;

出错提示：权限不足。因为创建视图需要“create view”的权限。默认scott用户没有该种权限。加之！

添加步骤：

1. 使用管理员登陆：sqlplus / as sysdba

2. 给scott用户增加权限

grant create view to scott;

3. 执行“/”可成功创建视图empincomeview.

4. 视图的操作和表的操作完全一样.

select \* from empincomeview;

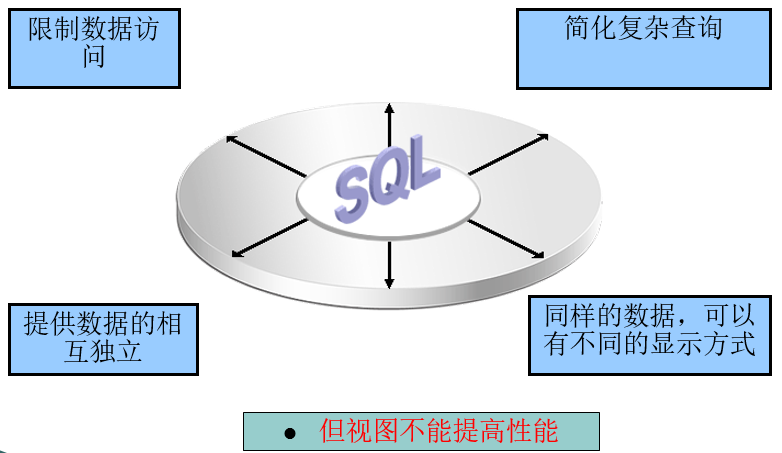
#### 视图的优点：

视图的优点

1. 简化复杂查询: 原来分组、多表、子查询等可以用一条select \* from xxxview代替,视图可以看做是表的复杂的SQL一种封装。

2. 限制数据访问: 只看视图的结构和数据是无法清楚视图是怎样得来的。可以限制数据的访问。例如：银行项目，所谓的各个“表”都是“视图”，并有可能只是“只读视图”

注意：1. 视图不能提高性能 2. 不建议通过视图对表进行修改。



#### 创建视图细节：

使用下面的语法格式创建视图：

**CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW *view***

**[(*alias*[, *alias*]...)]**

**AS *subquery***

**[WITH CHECK OPTION [CONSTRAINT *constraint*]]**

**[WITH READ ONLY [CONSTRAINT *constraint*]];**

1. 视图只能创建、删除、替换。（不能修改，修改即替换replace）

如：刚刚创建的empincomeview，其他语句不变，将create一行改写成：

create or replace view empincomeview 视图不存在则创建、存在则替换。

as

select…… from…..where…..

with read only; 可以将视图设为只读视图。

2. 别名: 可以写在子查询select各个列的后面，也可以写在视图的名字后面。

3. with read only 表示该视图为只读视图。

4. with check option 了解即可， 举例：

create view testview

as

select \* from emp where deptno=10

with check option;

insert into testview values(\*\*\*\*\*\*, 10);

不建议向视图插入，但可以做。向视图插入10号员工。

insert into testview values(\*\*\*\*\*\*, 20);

因为创建视图时加了“with check option”，所以失败。

视图中使用DML的规定：

一：

当视图定义中包含以下元素之一时不能使用delete:

* 组函数
* GROUP BY 子句
* DISTINCT 关键字
* ROWNUM 伪列

二：

当视图定义中包含以下元素之一时不能使用update :

* 组函数
* GROUP BY子句
* DISTINCT 关键字
* ROWNUM 伪列
* 列的定义为表达式

三：

当视图定义中包含以下元素之一时不能使用insert :

* 组函数
* GROUP BY 子句
* DISTINCT 关键字
* ROWNUM 伪列
* 列的定义为表达式
* 表中非空的列在视图定义中未包括

**总结一句话：**不通过视图做insert、update、delete操作。因为视图提供的目的就是为了简化查询。

删除视图:

drop view testview; ------🡪不加“purge”关键字。

### 序列：

可以理解成数组：默认，从[1]开始，长度[20] [1, 2, 3, 4, 5, 6, …, 20] 在内存中。

\*

由于序列是被保存在内存中，访问内存的速率要高于访问硬盘的速率。所以序列可以提高效率。

#### 序列的使用：

1. 初始状态下：指针\*指向1前面的位置。欲取出第一个值，应该将\*向后移动。每取出一个值指针都向后移。

2. 常常用序列来指定表中的主键。

3. 创建序列：create sequence myseq 来创建一个序列。

创建序列：

CREATE SEQUENCE *sequence*

[INCREMENT BY *n*]

[START WITH *n*]

[{MAXVALUE *n* | NOMAXVALUE}]

[{MINVALUE *n* | NOMINVALUE}]

[{CYCLE | NOCYCLE}]

[{CACHE *n* | NOCACHE}];

NOCACHE表示没有缓存，一次不产生20个，而只产生一个。

#### 创建序列

创建序列、表，以备后续测试使用：

create sequence myseq; 按默认属性创建一个序列。

create table tableA

(tid number, tname varchar2(20));

tid作为主键，准备使用序列来向表中插入值。

#### 序列的属性：

每个序列都有两个属性, nextval和currval.

NextVal 必须在CurrVal之前被指定。因为初始状态下，CurrVal指向1前面的位置，无值

对于新创建的序列使用select myseq.currval from dual; 得到出错。

但select myseq.nextval from dual; 可以得到序列的第一值1.

此时再执行select myseq.currval from dual; currval的值也得到1

使用序列给tableA表创建主键tid：

insert into tableA values(myseq.nextval, ‘aaa’)

只有nextval取完会向后移动，使用currval不会移动。

insert into tableA values(myseq.nextval, ‘bbb’)

继续使用nextval向表中添加主键tid

……

insert into tableA values(myseq.nextval, &name)

可以使用“&”和“/”来指定名字。

select \* from tableA;

由于前面测试currval和nextval关系时调用过nextval，所以tableA的tid起始从2开始。

查询序列的属性：select \* from user\_sequences; user\_sequences为数据字典视图。

修改序列：

* 必须是序列的拥有者或对序列有 ALTER 权限
* 只有将来的序列值会被改变
* 改变序列的初始值只能通过删除序列之后重建序列的方法实现

删除序列：drop sequence myseq;

使用序列需要注意的地方: 应首先执行nextval, 之后才能使用currval的值, 刚刚创建的序列不能直接使用currval的值.

#### 使用序列需要注意的问题：

* 序列是公有对象，所以多张表同时使用序列，会造成主键不连续。 如：[1, 2, 3, 4, 5, …, 20]

tableA: 1 2 4

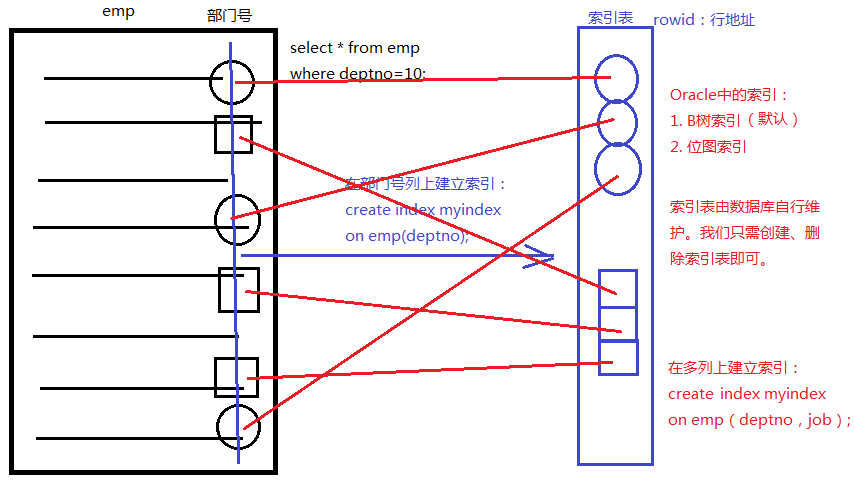
tableB: 3 5 A、B表有可能主键不连续。

* 回滚也可能造成主键不连续。 如：多次调用insert操作使用序列创建主键。但是当执行了rollback后再次使用insert借助序列创建主键的时候，nextval不会随着回滚操作回退。
* 掉电等原因，也可能造成不连续。由于代表序列的数组保存在内存中，断电的时候内存的内容丢失。恢复供电时候，序列直接从21开始。

### 索引：

索引，相当于书的目录，提高数据检索速度。提高效率（视图不可以提高效率）

* 一种独立于表的模式对象, 可以存储在与表不同的磁盘或表空间中
* 索引被删除或损坏, 不会对表产生影响, 其影响的只是**查询**的速度
* 索引一旦建立, Oracle 管理系统会对其进行自动维护, 而且由 Oracle 管理系统决定何时使用索引. 用户不用在查询语句中指定使用哪个索引
* 在删除一个表时, 所有基于该表的索引会自动被删除
* 通过指针**加速 Oracle 服务器的查询速度**
* 通过快速定位数据的方法，减少磁盘 I/O



上图中：

1. emp表中保存数据，其中包含部门号列。有10号部门，有20部门员工

2. 当 select \* from emp where deptno=10 的时候。由于10号部门员工不连续，没规律。

为了提高访问速度，可以在数据库中，依照rowid给deptno列建立索引

create index myindex on emp(deptno)

这样就建立了“索引表”可以通过rowid保存的行地址快速的找到表中数据。即使表中数据不连续。

3. 建立了索引以后，如果再执行select语句的时候，会先检查表上是否有索引表。如果有，可以通过有规律的rowid找到连续的数据。

4. Oracle的数据库中，索引有 B树索引（默认）和 位图索引两种。

5. 使用create index 索引表名 on 表名（列名1， 列名2…）;来创建索引表。由数据库自动进行维护。

使用主键查询数据最快速，因为主键本身就是“索引”，所以检索比较快。

索引使用的场景：

以下情况可以创建索引:

* 列中数据值分布范围很广
* 列经常在 WHERE 子句或连接条件中出现
* 表经常被访问而且数据量很大, 访问的数据大概占数据总量的2%到4%

下列情况不要创建索引:

* 表很小
* 列不经常作为连接条件或出现在WHERE子句中
* 查询的数据大于2%到4%
* 表经常更新

删除索引: drop index myindex;

### synonym同义词：

就是指表的别名。

如：scott用户想访问hr用户下的表employees。默认是不能访问的。需要hr用户为scott用户授权.

sqplus hr/11 或 conn hr/11（已登录界面， 切换登陆）

grant select on employees to scott;

hr用户为scott用户开放了employees表的查询权限。

这时scott用户就可以使用select语句，来查询hr用户下的employees表的信息了。

select count(\*) from hr.employees; (若用户名叫zhangsanfeng则zhangsanfeng.employees)

hr.employees名字过长，为了方便操作，scott用户为它重设别名:

create synonym hremp for hr.employees; 为hr.employees创建了同义词。

如有权限限制，那么切换管理员登录，给scott用户添加设置同义词权限。

conn / as sysdba

grant create synonym to scott;

select count(\*) from hremp; 使用同义词进行表查询操作。

同义词、视图 等用法在数据保密要求较高的机构使用广泛，如银行机构。好处是既不影响对数据的操作，同时又能保证数据的安全。

OCA

OCP

OCM 认证

select rownum, empno, ename, sal

from (select \*

from emp

order by sal desc)

where rownum <=3;

select \* from ( select rownum rn, empno, ename, sal

from (select \*

from emp

order by sal desc) ) where rn>=4 and rn<=7;

select d.deptno, e.empno, e.ename, d.avgsal, e.sal from emp e, (select deptno, avg(sal) avgsal from emp group by deptno) d where e.deptno=d.deptno and e.sal>d.avgsal;