创建表和删除表

--dept表

DROP TABLE DEPT;

CREATE TABLE DEPT (DEPTNO NUMBER(2) CONSTRAINT PK\_DEPT PRIMARY KEY, DNAME VARCHAR2(14), LOC VARCHAR2(13) );

INSERT INTO DEPT VALUES (10,'ACCOUNTING','NEW YORK');

INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');

INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');

INSERT INTO DEPT VALUES (40,'OPERATIONS','BOSTON');

--emp表

DROP TABLE EMP;

CREATE TABLE EMP ( EMPNO NUMBER(4) CONSTRAINT PK\_EMP PRIMARY KEY,

ENAME VARCHAR2(10),

JOB VARCHAR2(9),

MGR NUMBER(4),

HIREDATE DATE,

SAL NUMBER(7,2),

COMM NUMBER(7,2),

DEPTNO NUMBER(2) CONSTRAINT FK\_DEPTNO REFERENCES DEPT);

INSERT INTO EMP VALUES (7369,'SMITH','CLERK',7902,to\_date('17-12-1980','dd-mm-yyyy'),800,NULL,20);

INSERT INTO EMP VALUES (7499,'ALLEN','SALESMAN',7698,to\_date('20-2-1981','dd-mm-yyyy'),1600,300,30);

INSERT INTO EMP VALUES (7521,'WARD','SALESMAN',7698,to\_date('22-2-1981','dd-mm-yyyy'),1250,500,30);

INSERT INTO EMP VALUES (7566,'JONES','MANAGER',7839,to\_date('2-4-1981','dd-mm-yyyy'),2975,NULL,20);

INSERT INTO EMP VALUES (7654,'MARTIN','SALESMAN',7698,to\_date('28-9-1981','dd-mm-yyyy'),1250,1400,30);

INSERT INTO EMP VALUES (7698,'BLAKE','MANAGER',7839,to\_date('1-5-1981','dd-mm-yyyy'),2850,NULL,30);

INSERT INTO EMP VALUES (7782,'CLARK','MANAGER',7839,to\_date('9-6-1981','dd-mm-yyyy'),2450,NULL,10);

INSERT INTO EMP VALUES (7788,'SCOTT','ANALYST',7566,to\_date('13-JUL-87')-85,3000,NULL,20);

INSERT INTO EMP VALUES (7839,'KING','PRESIDENT',NULL,to\_date('17-11-1981','dd-mm-yyyy'),5000,NULL,10);

INSERT INTO EMP VALUES (7844,'TURNER','SALESMAN',7698,to\_date('8-9-1981','dd-mm-yyyy'),1500,0,30);

INSERT INTO EMP VALUES (7876,'ADAMS','CLERK',7788,to\_date('13-JUL-87')-51,1100,NULL,20);

INSERT INTO EMP VALUES (7900,'JAMES','CLERK',7698,to\_date('3-12-1981','dd-mm-yyyy'),950,NULL,30);

INSERT INTO EMP VALUES (7902,'FORD','ANALYST',7566,to\_date('3-12-1981','dd-mm-yyyy'),3000,NULL,20);

INSERT INTO EMP VALUES (7934,'MILLER','CLERK',7782,to\_date('23-1-1982','dd-mm-yyyy'),1300,NULL,10);

--bouns表

DROP TABLE BONUS;

CREATE TABLE BONUS(ENAME VARCHAR2(10),JOB VARCHAR2(9) ,SAL NUMBER,COMM NUMBER);

--salgrade表

DROP TABLE SALGRADE;

CREATE TABLE SALGRADE( GRADE NUMBER,LOSAL NUMBER,HISAL NUMBER );

INSERT INTO SALGRADE VALUES (1,700,1200);

INSERT INTO SALGRADE VALUES (2,1201,1400);

INSERT INTO SALGRADE VALUES (3,1401,2000);

INSERT INTO SALGRADE VALUES (4,2001,3000);

INSERT INTO SALGRADE VALUES (5,3001,9999);

--提交

COMMIT;

表a(bm,mc);表b(bm,mc)删除a表中与b表重复的数据

方法一：

delete from a where exists(select 'X' from b where a.bm = b.bm and a.mc = b.mc);

方法二：

delete from a where (bm,mc) in (select bm,mc from b);

插入语句insert into

insert into 表名 (列1, 列2,...) values (值1, 值2,....)

--或：

insert into 表名values (值1, 值2,....)

更新语句 update

update 表名 set 列名 = 新值 where 列名 = 值

删除记录语句 delete

delete from 表名 where 列名 = 值

连接

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表连接 | 1.外连接 | 左外连接 | Left join...on... | Left out join...on... |
| 右外连接 | Right join...on... | Right out join...on... |
| 完整外连接 | Full join...on... | Full out join...on... |
| 2.内连接 |  | Join...on... | Inner join...on... |
| 3.交叉连接 |  | Cross join |  |

例表：

table1

|  |  |
| --- | --- |
| **id** | **name** |
| 1 | lee |
| 2 | zhang |
| 4 | wang |

table2

|  |  |
| --- | --- |
| **id** | **score** |
| 1 | 90 |
| 2 | 100 |
| 3 | 70 |

左连接left join...on...或left out join...on...

左向外联接的结果集包括 LEFT OUTER 子句中指定的左表的所有行，而不仅仅是联接列所匹配的行。

如果左表的某行在右表中没有匹配行，则在相关联的结果集行中右表的所有选择列表列均为空值(null)。

select \* from table1 left join table2 on table1.id=table2.id

结果集：包含table1的所有子句，根据指定条件返回table2相应的字段，不符合的以null显示。

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **id** | **score** |
| 1 | lee | 1 | 90 |
| 2 | zhang | 2 | 100 |
| 4 | wang | null | null |

**右连接：right join...on... 或 right outer join...on...**

 (1) 右向外联接是左向外联接的反向联接。将返回右表的所有行。如果右表的某行在左表中没有匹配行，则将为左表返回空值。

select \* from table1 right join table2 on table1.id=table2.id

结果集：包含table2的所有子句，根据指定条件返回table1相应的字段，不符合的以null显示

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **id** | **score** |
| 1 | lee | 1 | 90 |
| 2 | zhang | 2 | 100 |
|  |  |  |  |
| null | null | 3 | 70 |

**完整外部联接:full join...on... 或 full outer join...on...**

 (1)完整外部联接返回左表和右表中的所有行。当某行在另一个表中没有匹配行时，则另一个表的选择列表列包含空值。如果表之间有匹配行，则整个结果集行包含基表的数据值。

select \* from table1 full join table2 on table1.id=table2.id

结果集：返回左连接结果集与右连接结果集的并集。

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **id** | **score** |
| 1 | lee | 1 | 90 |
| 2 | zhang | 2 | 100 |
| 4 | wang | null | null |
| null | null | 3 | 70 |

**内连接 join...on... 或 inner join...on...**

1.概念：内联接是用比较运算符比较要联接列的值的联接

select \* from table1 join table2 on table1.id=table2.id

--等价于：

select a.\*,b.\* from table1 a,table2 b where a.id=b.id

结果集：只返回符合条件的table1和table2的列 。说明，连接的符号也可以使用其他比较运算符。

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **id** | **score** |
| 1 | lee | 1 | 90 |
| 2 | zhang | 2 | 100 |

**交叉连接（笛卡尔积） cross join**    说明：cross join 无on子句。

1.概念：没有 WHERE 子句的交叉联接将产生联接所涉及的表的笛卡尔积。

select \* from table1 cross join table2

--等价于：

select \* from table1,table2

select \* from table1 cross join table2 where table1.id=table2.id

--(注：cross join后加条件只能用where,不能用on)

结果集：table1和table2交叉连接产生3\*3=9条记录

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **id** | **score** |
| 1 | lee | 1 | 90 |
| 2 | zhang | 2 | 90 |
| 4 | wang | 3 | 90 |
| 1 | lee | 1 | 100 |
| 2 | zhang | 2 | 100 |
| 4 | wang | 3 | 100 |
| 1 | lee | 1 | 70 |
| 2 | zhang | 2 | 70 |
| 3 | wang | 3 | 70 |

select [distinct] \*|[column1 [as col1] , column2 [as col2] , …]

from table1 [tab1],table2 [tab2], …

where 查询条件

group by columnA, columnB , …

having 对分组的约束

order by字段x,字段y,…[ASC|DESC]

显示EMP表中雇员们,及其工资增加20%后的工资列

select ename,(1+0.2)\*sal from emp;

where子句的条件表达式中可以使用的运算符：

=  ； <>，!= ； >= ； <= ； > ； < ；

and ； or ； not ；

in(...)  ； not in(...) ；

between a and b ； not between a and b ；

is null ； is not null ； like ； not like

select ename,sal,hiredate from emp where hiredate>to\_date('1-1月-1982','dd-mm-yy')

select ename,sal,hiredate from emp where sal between 1000 and 2000

select ename,sal,hiredate from emp where ename like 'S%'

select ename,sal,hiredate from emp where sal in (1250,3000)

[not] like ‘匹配字符串’ [ESCAPE ‘换码字符串’]

like的通配符：

%    用于表示0个或多个字符。

\_     用于表示1个字符，称为位置标识符。

max

min

avg

count

variance  返回列或表达式的方差

stdeev  返回列或表达式的标准差

select avg(sal),avg(distinct sal),max(sal) ,min(sal),sun(sal),count(\*),count(sal),count(distinct sal),count(comm) from emp where deptno=30;

使用ROLLUP和CUBE限定词

在实际应用中，若需要生成横向，纵向的统计结果(两列分组时)。此时可以在group by子句中使用限定词rollup和cube来生成超级组合。

rollup用于生成横向统计结果。

cube用于生成纵向统计结果，横向统计结果。

SQL可以分为5类

1.查询语句

**SELECT**

2.数据操纵语句(Data Manipulation Language)

**INSERT :向表中添加行**

**UPDATE ：修改行的内容**

**DELETE ：删除行**

**MERGE ：合并**

3.数据定义语言

**CREATE ：创建数据库对象**

**ALTER ：修改数据库结构**

**DROP : 删除数据库结构**

**RENAME ：更改表明**

**TRUNCATE ：删除表的全部内容**

4.事务控制

**COMMIT ：永久性保存对行所作的修改**

**ROLLBACK ：取消对行所作的修改**

**SAVEPOINT ：设置一个保存点。可以将对行的修改回滚到此处**

5.数据控制语言

**GRANT ：授予其他用户对数据库结构的访问权限**

**REVOKE ：收回用户访问数据库结构的权限**

**注意：**

1.SQL关键字不区分大小写

2.对项名和列名不区分大小写

3.字符值和日期值区分大小写

4.在应用程序中编写SQL语句时，如果SQL语句文本很短，可以将语句文本放在一行；如SQL很长，则将文本分布到多行，可以使用缩进提高可读性。