```
SELECT a.tran_date,a.lc_amt,a.lc_no,a.TRXREF,a.who_create,a.WHO_MODI,b.scl_trandate
FROM IMLCEVENT A,SYS_CANCELLOG b where b.scl_trandate =
SUBSTR(A.WHEN_MODI,0,10)
UNION ALL
SELECT
a.tran_date,a.nego_amt,a.nego_no,a.TRXREF,a.who_create,a.WHO_MODI,b.scl_trandate FROM
IMBLEVENT A,SYS_CANCELLOG b where b.scl_trandate = SUBSTR(A.WHEN_MODI,0,10)
```

一.简单的 SQL 语句

1.创建一个简单的表：

```
create table test(
Sex Char(2),
Name varchar2(16),
Age number(3),
Birth date
);
create table exp_table_list(
table_name varchar2(20),
table_desc varchar2(20),
table_flds varchar2(3000)
);
```

Note:Char(n)类型存放长度完全相同的的字符串，即使字符串数据不同，在存储数据的时候也被存储成相同长度的字符串

Desc test：查看表的结构

字段间用“，”隔开，字段名称，字段类型,语句最终以“；”结束

2.SQL 语句分类

1.查询语句

select

2.DML 语句（数据操作语言）

Insert/Update/Delete/Merge(合并)

3.DDL 语句（数据定义语言）

Create/Alter/Drop/Truncate

Truncate:删除表中的所有行，但表结构及其列，约束，索引等保持不变

4.DCL 语句（数据控制语言）

Grant/Revoke

5.事务控制语句

Commit/Rollback/Savepoint

3.sqlplus 连接数据库的基本语法：

sqlplus 用户名/密码@网络数据库名

example：sqlplus scott/tiger@abc

4. 包含算术表达式的 select 语句

```
select last_name,salary,salary*(12+100) from employee;
```

包含连接表达式的 select 语句

```
select last_name||'work at'||job_id from employee;
```

字段别名

Note：使用别名强制大小写必须使用双引号扩起；

select last_name "Name",salary* 12 "Annual Salary" from employee

文本字符串

select last_name,'Today is'||'01-05' from employee;

select last_name,100 from employee;

去掉重复值

select distinct department_id from employee;

5.带有限制条件的查询

使用 where：

```
select lastname,salary,commission_pct
from employee
where salary>=1500;
where salary between 1000 and 2000;
where manager_id IN(7901,7923,2321);
where last_name is null;
where last_name like's%';
where last_name like'_s%';
```

like 用于做模糊查询，like 操作符与通配符配合使用；通配符有两个 1.“%”可以表示零或多个字符 2.“_”可以表示一个字符

where 中的逻辑运算符：and，or，not

```
select last_name,job_id,salary
from employee
where (job_id='salesman'
or job_id='president'
)
and salary>1500;
```

查询数据的排序

```
select last_name,job_id,hiredate
from employee
order by hiredate desc，salary(asc 升序可缺省)；
```

6.sql 中的函数

单行函数：字符函数，数字函数，日期函数，转换函数，其它函数。

常见的字符函数：

lower,upper,initcap,concat,substr,length,instr,lpad|rpad,trim,replace 等

lower 强制转换小写，upper 强制转换大写，initcap 首字母大写

concat 连接两个字符，substr 取出字符串中的一个字串，length 求出字符串的长度，instr 在字符串中找子串的位置，lpad 字符串填充补位函数，trim 截取字符串两端特殊字符，replace 替换字符串子串

```
select replace('oracle sql','oracle','training')test from from dual；
select last_name,concat(last_name,job)can,
length(last_name)len,instr(last_name,'a')ins
from employee
where substr(job_id,1,5)='sales';
```

数字类型函数
round（四舍五入），trunc（对数字进行截取），mod（求模）
select round(52.232,2) "小数后两位"，
round(52.232,0) "整数"， round(52.232,-1) "十位"
from sys.dual
select last_name,salary,MOD(salary,1000)
from employee
where job_id=‘saleman’；
日期类型函数
Oracle 取出当前日期：select sysdate from dual;
截取日期
trunc（sysdate，‘D’）截取到本周的第一天
trunc（sysdate，‘MM’）截取到本月的第一天
trunc（sysdate，‘DD’）截取到本日的第一天
trunc（sysdate，‘yyyy’）截取到今年的第一天
转换函数：
to_char,to_date,to_number
select last_name,
to_char(hire_date,'fmDD Month YYYY')
as hiredate
from employee;
select to_char(salary,'$00,000,000')salary
from employee
where last_name='scott';
select to_date('2004-1-2','YYYY-MM-DD')
from dual;
其它函数：
NVL(expr1,expr2)
NVL2(expr1,expr2,expr3)
NULLIF(expr1,expr2)
COALESCE(expr1,expr2,...exprn)
case expr
when comparison_expr1 then return_expr1
[
.....
when comparison_expr1 then return_exprn
]
end
分组函数：
avg();count();max();min();sum()
使用 group by 对数据分组
select department_id,AVG(salary),count(last name)
from employee
group by department_id;

使用 having 子句对分组结果进行限制

```
select department_id,Max(salary)
from employee
group by department_id
having max(salary)>10000
order by sum(salary);
```

7.多表查询和子查询

```
select table1 column,table2 column
from table1,table2
where table1.column1=table2.column2;
```

等值连接：

```
selecte.employee_id,e.last_name,e.department_id,d.department_id,d.location_id,l.city
from employee e,departments d,locations
where e.department_id=d.department_id
and d.location_id=l.location_id;
```

非等值连接：

```
select e.last_name,e.salary,j.grade_level
from employee e,job_grade j
where e.salary between j.lowest_sal and j.hightest_sal;
```

外连接

```
select e.last_name,e.department_id,d.departmen_name
from employee e,department d
where e.department_id(+)=d.department_id;(左外连接)
```

自连接

```
select worker.last_name||'work for'||manager.last_name
from employee worker,employee manager
where worker.manager_id=manager.employee_id;
```

8.数据操作与事务控制

添加：insert into department(department_id,department_name,location_id)
values(320,'财务部',1700);

修改：update departments
set salary=salary*1.1,com_pct=0.1
where department_id=10;

删除：delete departments
where department_id>150;

Merge(根据设置的条件将查询的数据目的表中执行修改或数据的操作，如果目的的表中存在相同的记录则执行 update，否则执行 insert。

```
merge into table_name table_alias
using(table|view|sub_query)alias
on (join condition)
when matched then
update set
col1=col1_val,
col2=col2_val...
```

when not matched then
insert(column_list)
values(column_values);
into 子句后是执行操作的目的表，也就是前面提到的历史表，用来存放历史数据的
using 子句是数据源，数据源包括表，试图，或者子查询
on 子句就是判断数据是否存在目的表中存放的判断条件，写法类似 where 语句中的条件语句，可以写上一个或多个连接在一起的条件。

9.其他

创建测试表：

```
drop sequence student_sequence；
create sequence student_sequence；
start with 10000
increment by 1;
drop table students;
create table students(
id number(5)primary key,
first_name varchar2(20),
last_name varchar2(20),
major varchar2(20),
current_credits number(3),
grade varchar2(20)
);
```

插入数据

```
insert into student(id,first_name,last_name,major,current_credits,grade)
values(student_sequence.NEXTVAL,'scott','smith','computer',98,null);
commit;
update students
set grade=(
select id,
case when current_credits>90 then 'a'
      when current_credits>80 then 'b'
else 'c'
end grade
from students) a
where a.id=students.id
）；
```

日期转换：

```
crete table date_test(d date);
insert into date_test
values(to_date('2004-02-09','YYYY-MM-DD'));
```