

oracle

1.安装和使用oracle

1-1.解压已经安装oracle的xp

解压xp虚拟机(已安装oracle)到没有中文名称的目录

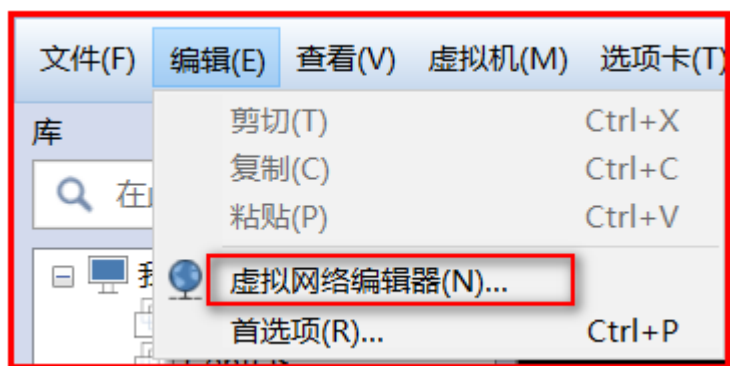
LENOVO (D:) > develop >				
名称	修改日期	类型	大小	
xp虚拟机 (已安装oracle) .7z	2018/8/28 20:20	好压 7Z 压缩文件	2,736,423	
xp_oracle	2018/8/30 9:56	文件夹		

1-2.虚拟机中打开解压的xp系统

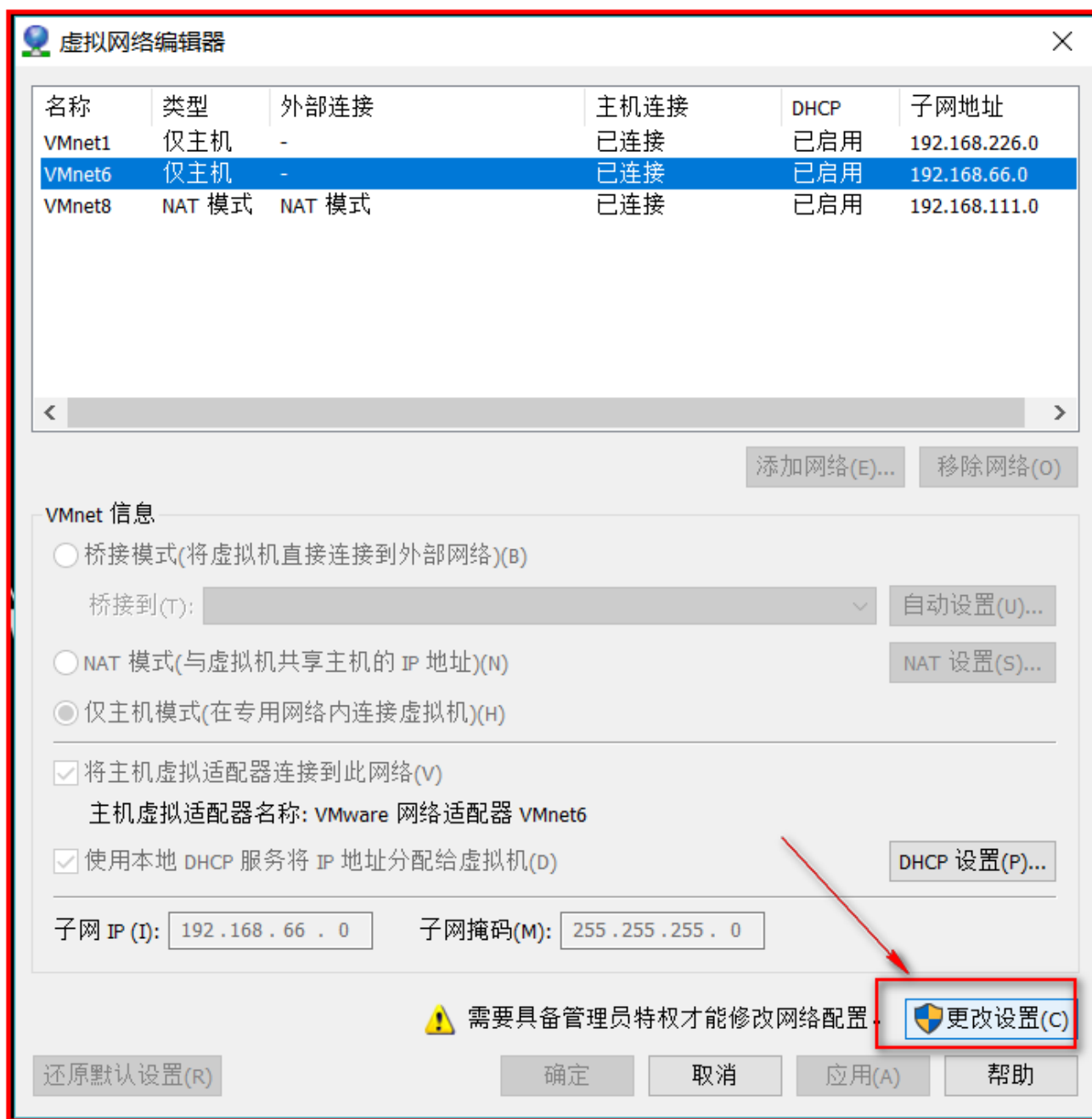
LENOVO (D:) > develop > xp_oracle >				
名称	修改日期	类型	大小	
564d4d10-fd19-7cc6-6209-fec60b7d...	2018/8/30 9:55	文件夹		
cache	2018/8/22 19:28	文件夹		
xp_oracle.vmdk.lck	2018/8/30 9:55	文件夹		
xp_oracle.vmx.lck	2018/8/30 9:55	文件夹		
564d4d10-fd19-7cc6-6209-fec60b7d...	2018/8/30 9:55	VMEM 文件	1,048,576	
vmware.log	2018/8/30 9:55	文本文档	0 KB	
vmware-0.log	2018/8/29 22:59	文本文档	225 KB	
vmware-1.log	2018/8/29 21:55	文本文档	223 KB	
vmware-2.log	2018/8/29 21:48	文本文档	225 KB	
vprintproxy.log	2018/8/22 16:50	文本文档	17 KB	
xp_oracle.nvram	2018/8/29 22:59	VMware 虚拟机...	9 KB	
xp_oracle.vmdk	2018/8/30 9:55	VMware 虚拟磁	6,148,032	
xp_oracle.vmsd	2016/5/28 17:51	VMware 快照元	0 KB	
xp_oracle.vmx	2018/8/30 9:56	VMware 虚拟机	3 KB	
xp_oracle.vmx	2016/5/28 18:37	VMware 组成员	4 KB	

1-3.设置虚拟机ip

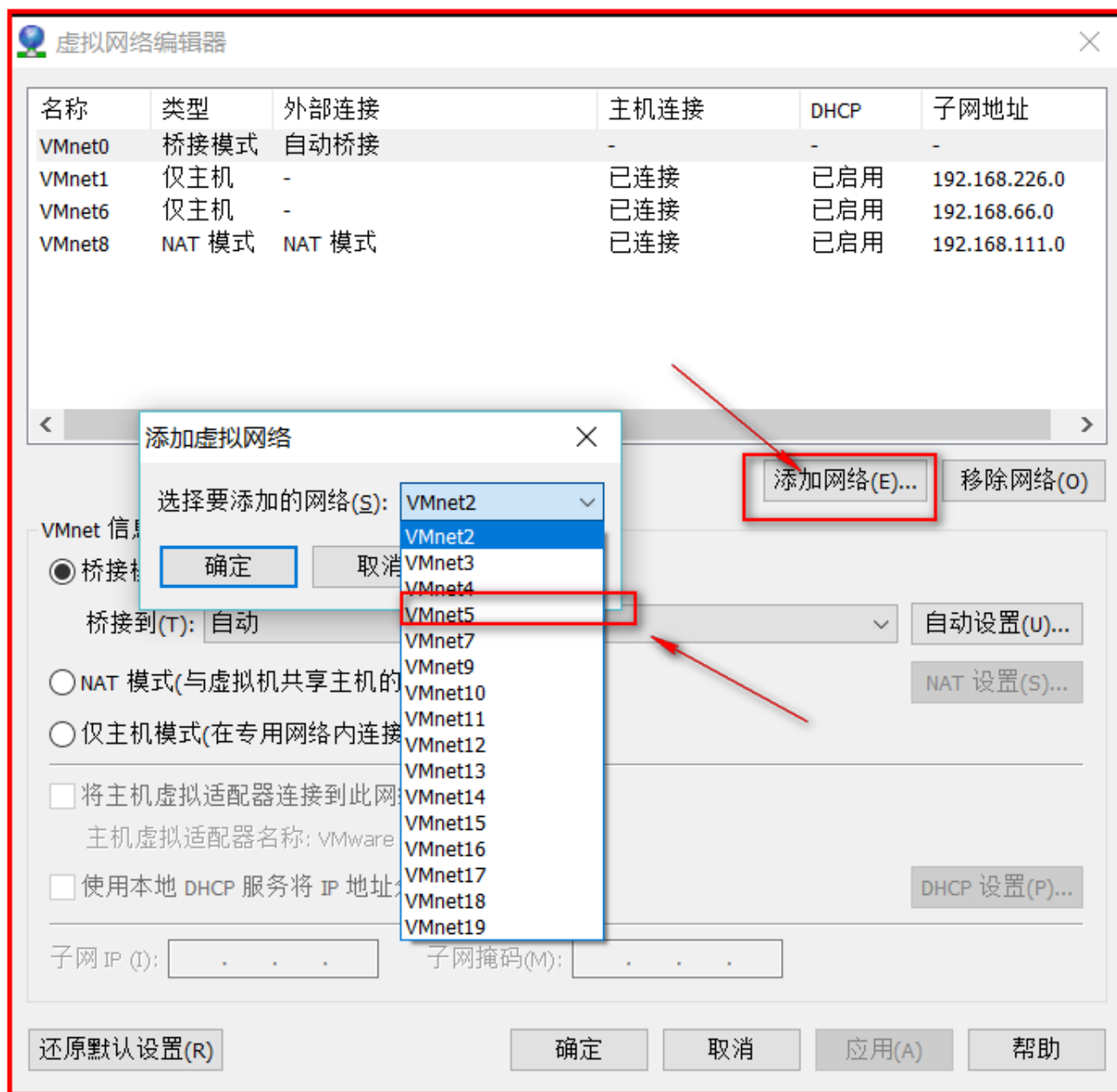
虚拟机网卡：解决本机和虚拟机之间通讯使用(仅主机模式) 编辑-->虚拟网络编辑器



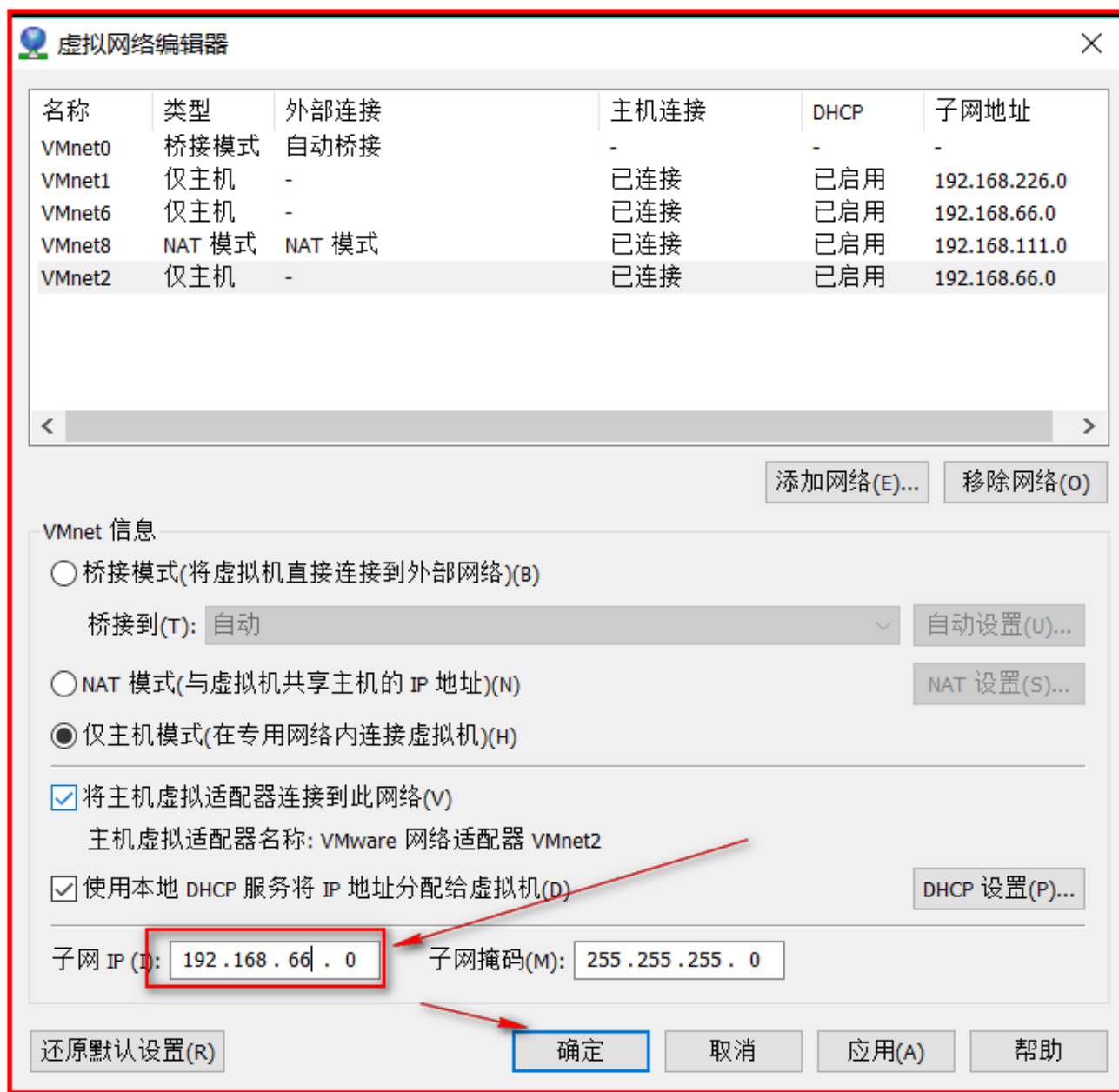
选择更改设置



添加网络-->选择要添加的网络-->确定

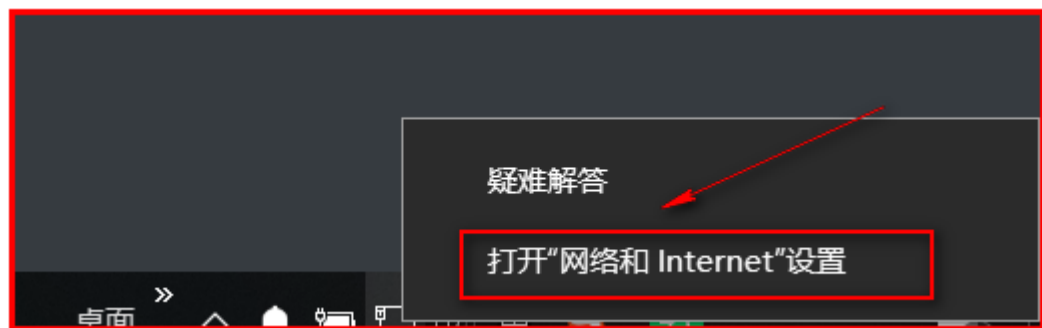


子网ip: 192.168.66.0

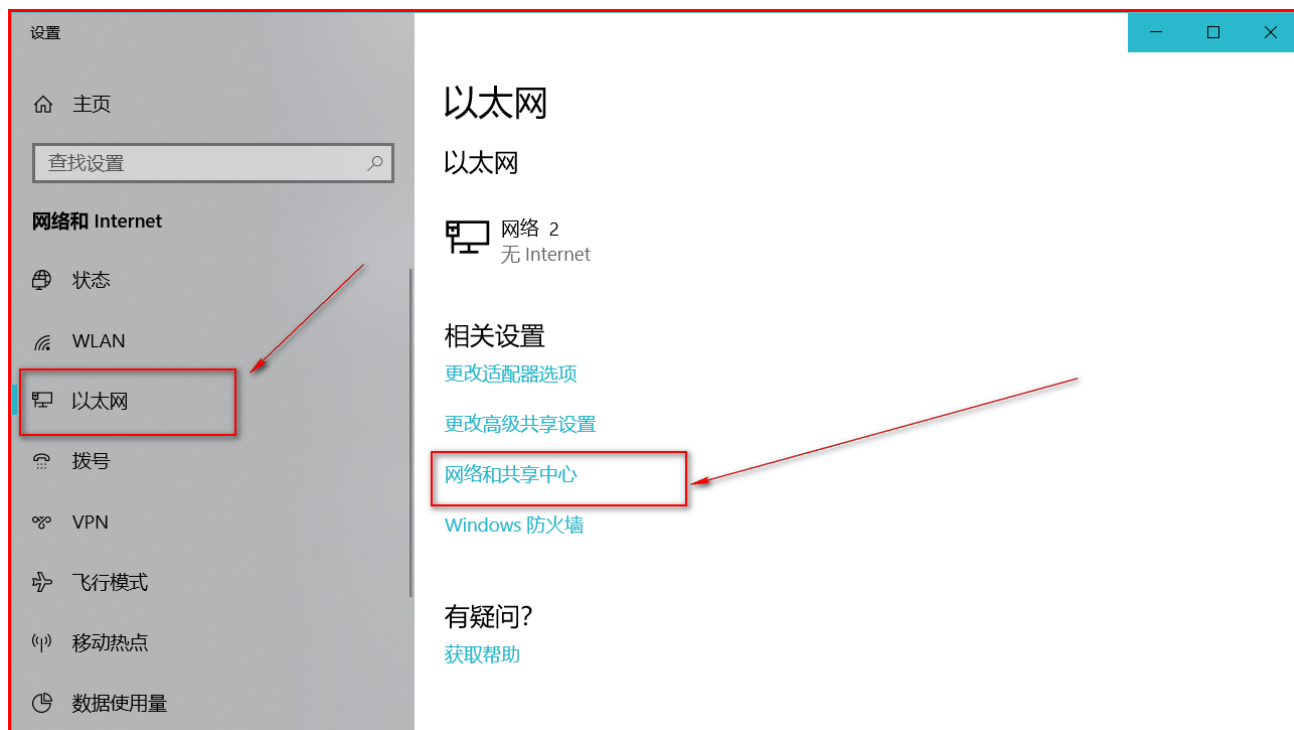


1-4.检查本机和虚拟机是否是同一个网段

win10: 打开网络和Internet设置



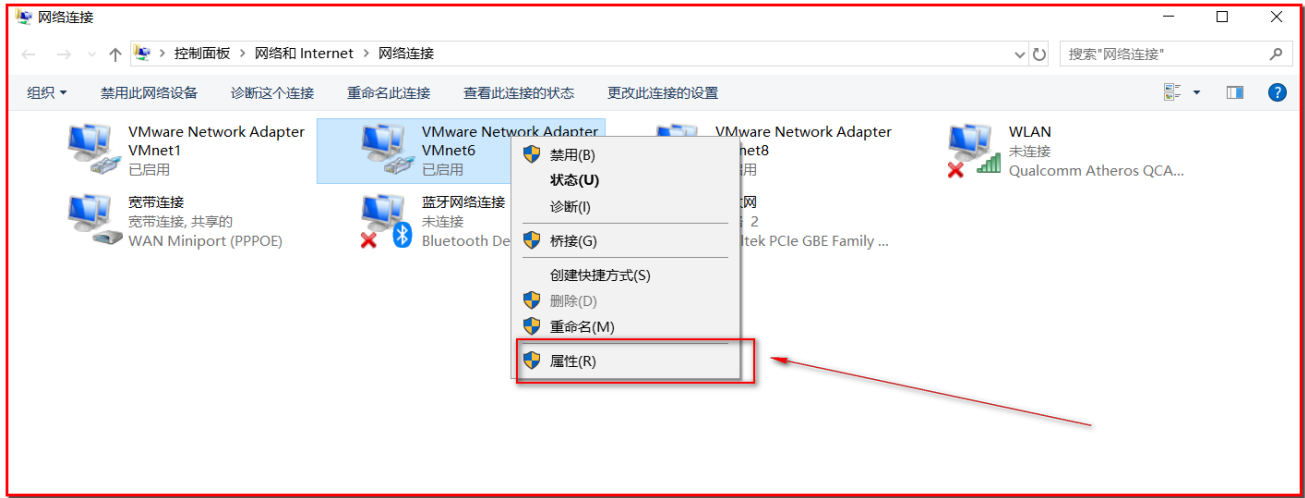
选择以太网-->网络和共享中心



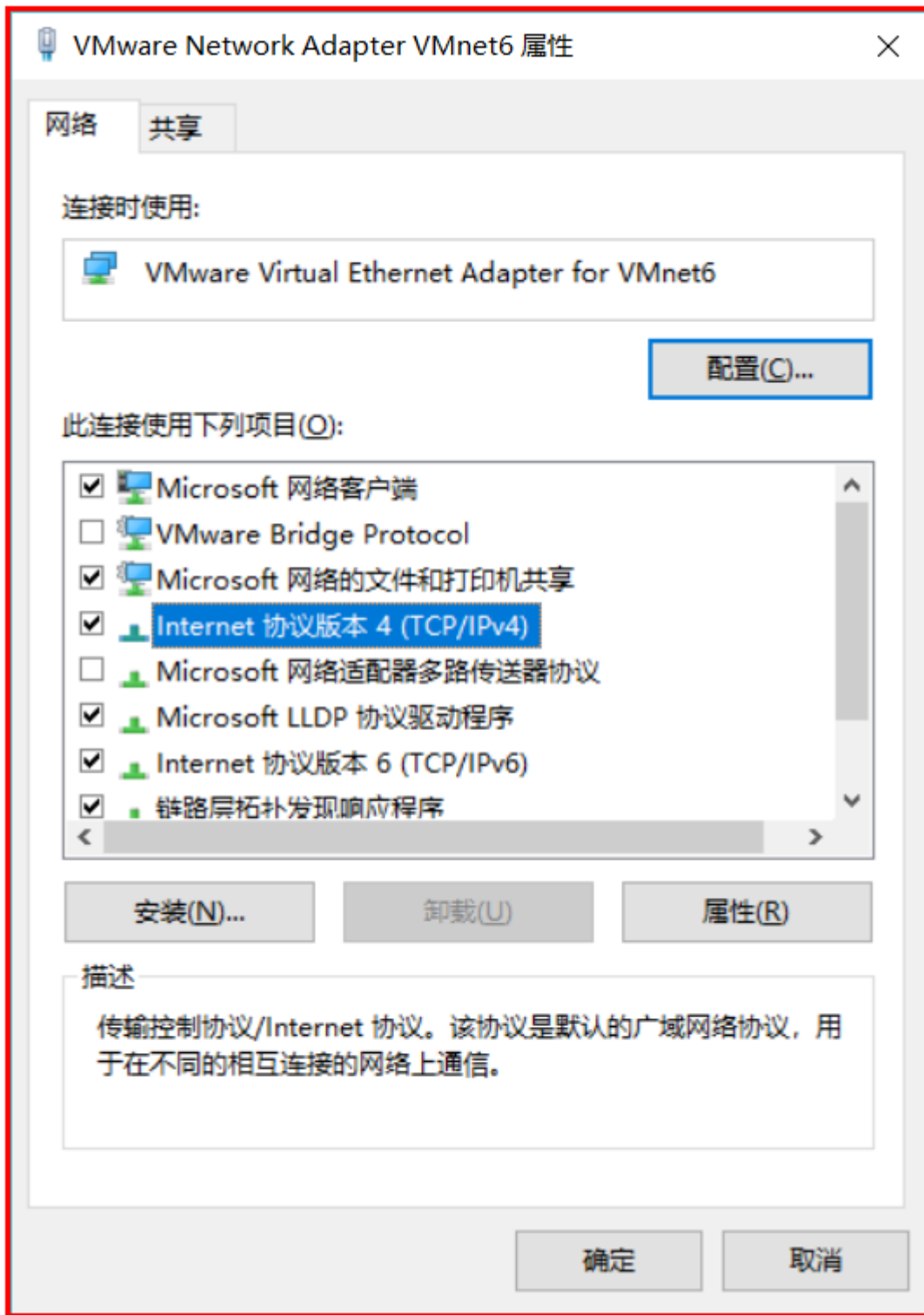
选择更改适配器设置



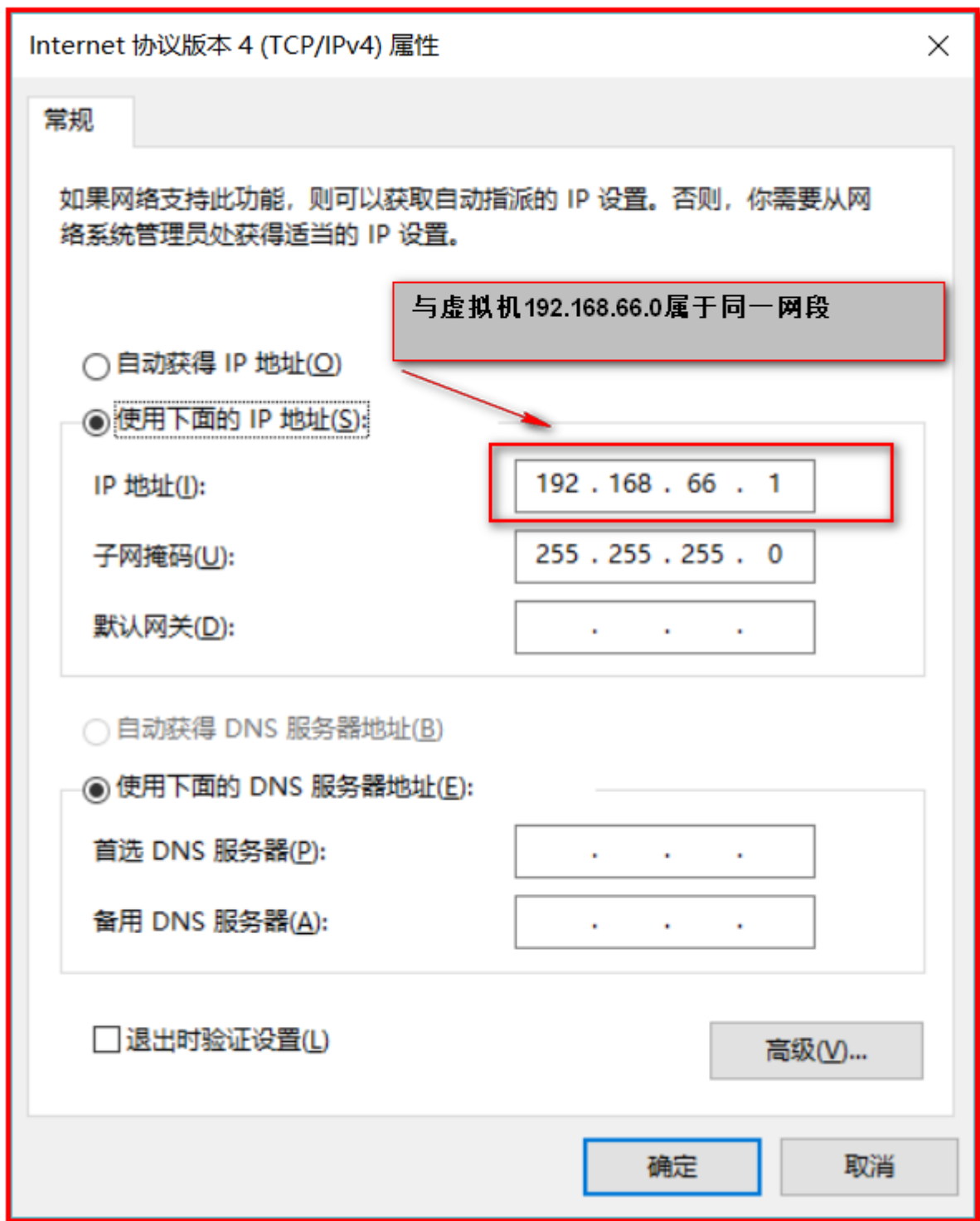
右键选择属性



选择TCP/IPv4，然后左键双击



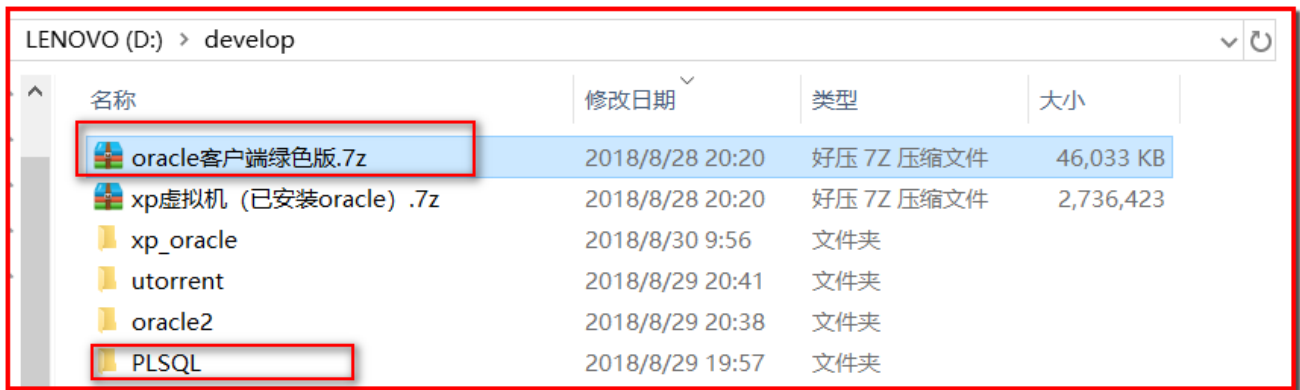
查看IP地址是否和虚拟机是属于同一网段



2.安装和使用pl/sql客户端

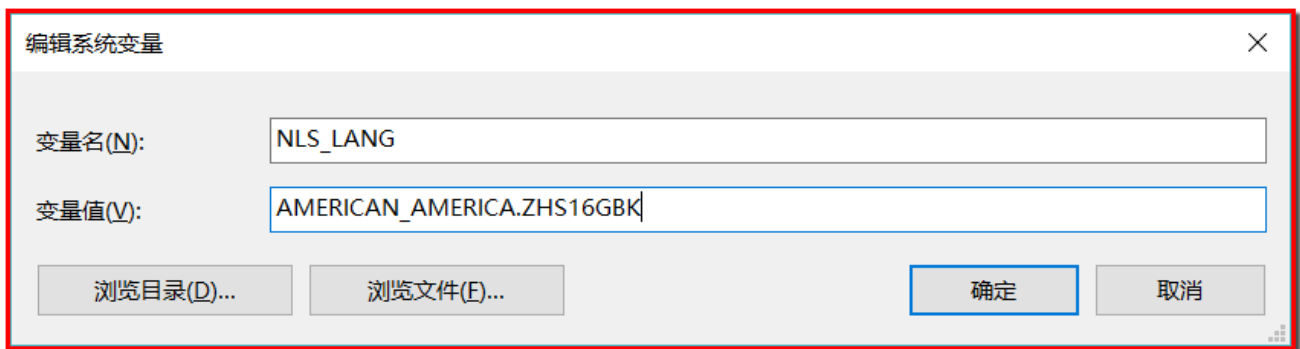
2-1.解压oracle客户端绿色版

最好是解压到一个不含中文名称的目录

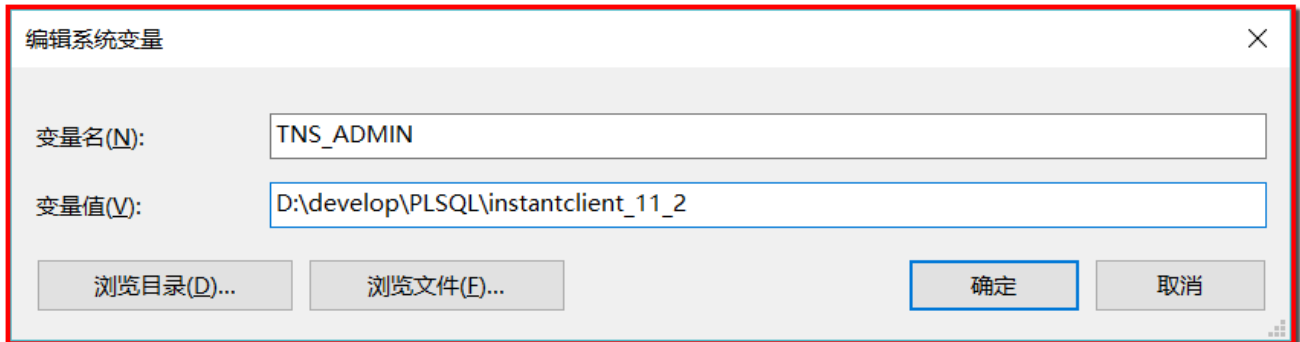


2-2.配置环境变量

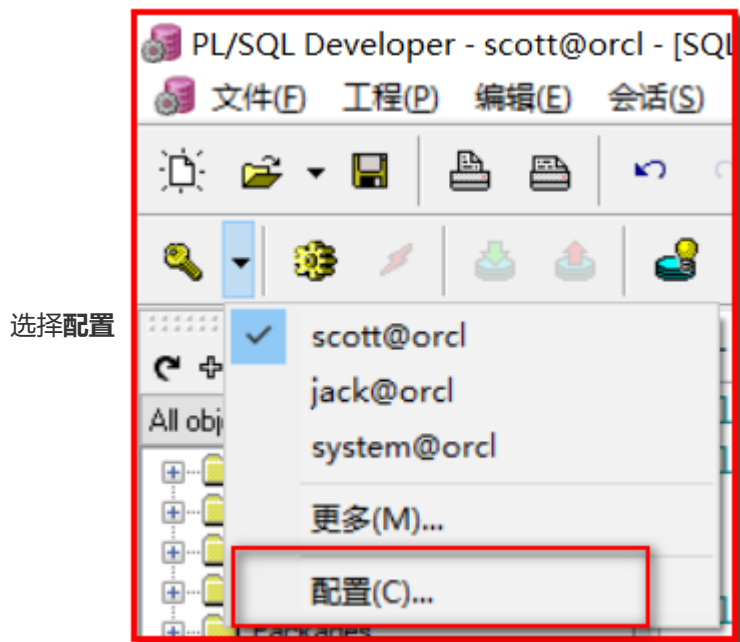
解决客户端乱码问题 key: **NLS_LANG** value: **AMERICAN_AMERICA.ZHS16GBK**



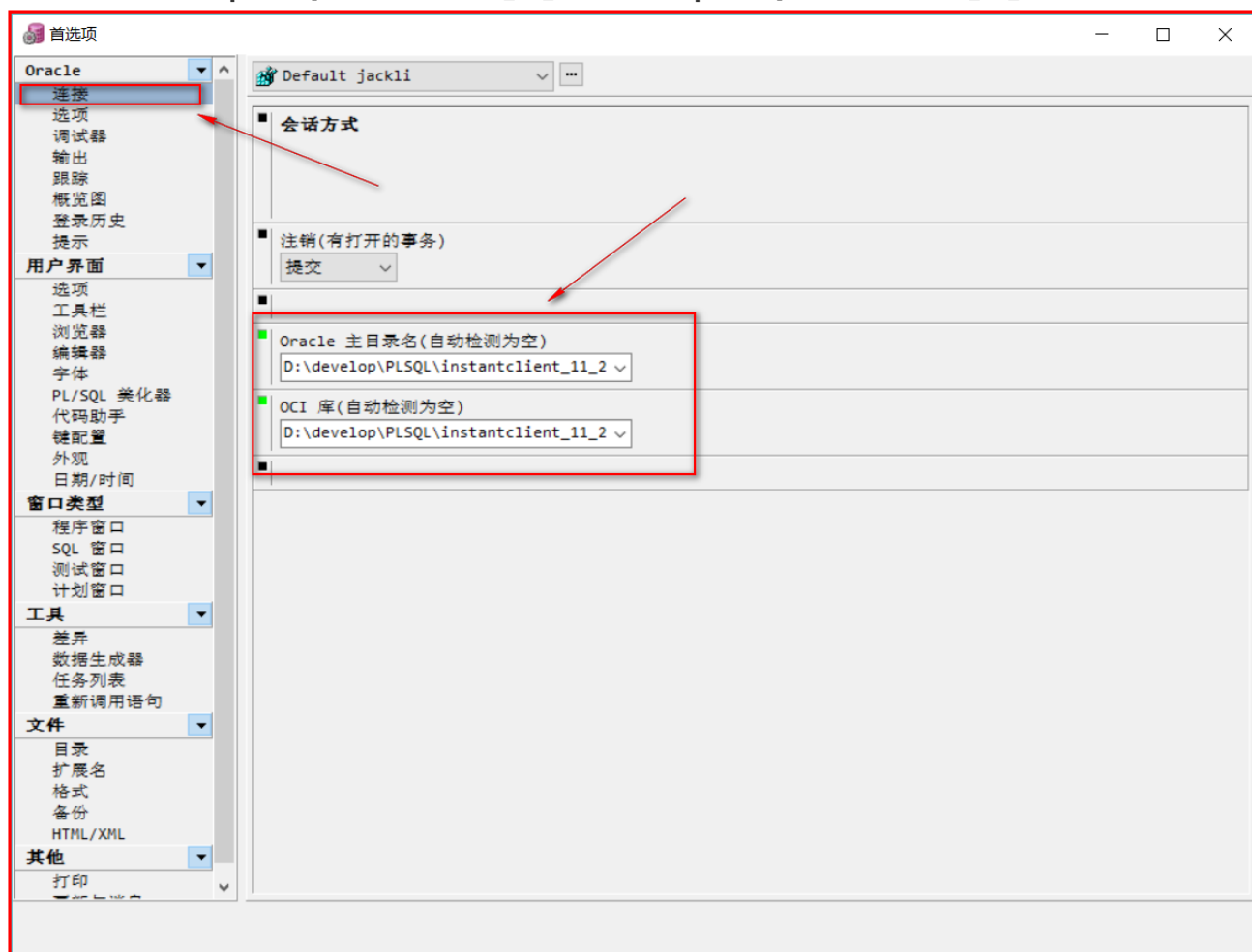
设置连接Linux的oracle的 key: **TNS_ADMIN** value: **D:\develop\PLSQL\instantclient_11_2**



2-3.在plsql中设置oracle客户端



设置值: D:\develop\PLSQL\instantclient_11_2 D:\develop\PLSQL\instantclient_11_2\oci.dll



2-4.查看tnsnames.ora配置是否正确

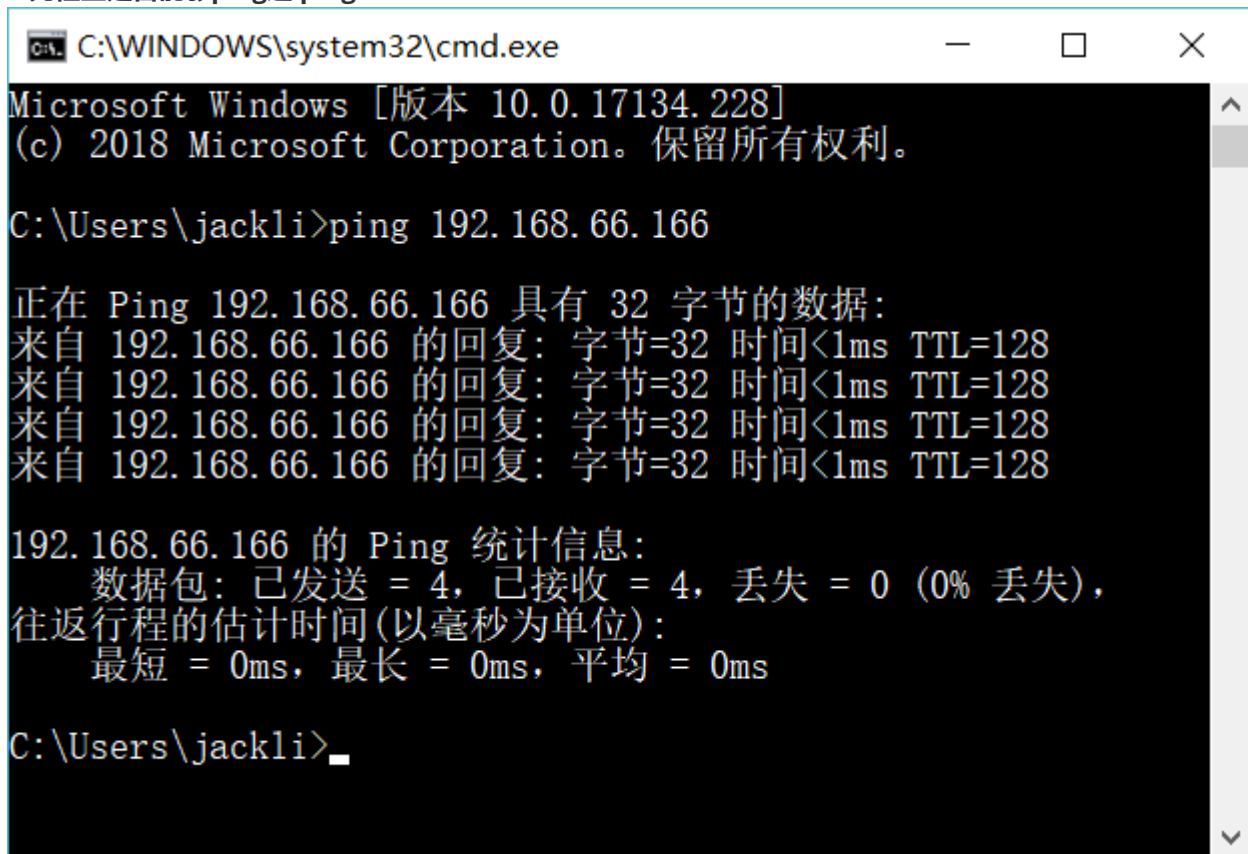
D:\develop\PLSQL\instantclient_11_2\tnsnames.ora

```
#system/system sys/sys orcl/orcl

orcl =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = 192.168.66.166) (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORCL)
    )
  )
```

2-5.测试本机是否能够连接上虚拟机上的oracle数据库

1.先检查是否能够ping通 ping 192.168.66.166



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The text inside the window is as follows:

```
Microsoft Windows [版本 10.0.17134.228]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\jackli>ping 192.168.66.166

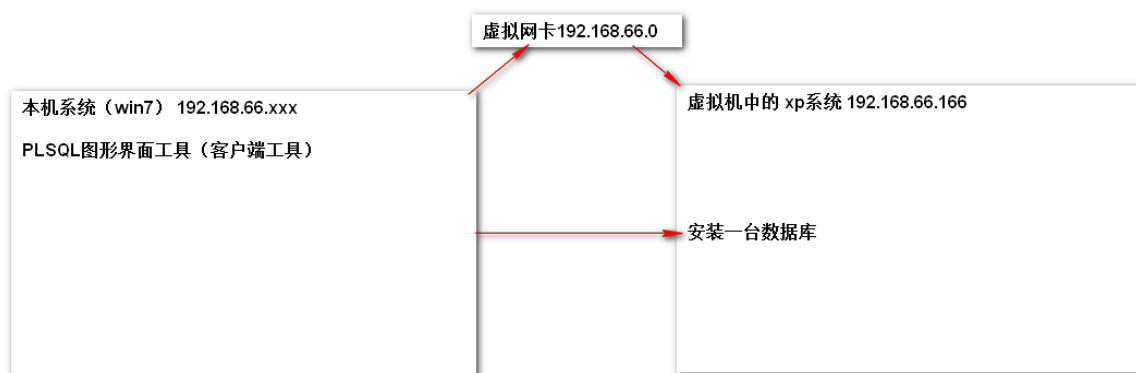
正在 Ping 192.168.66.166 具有 32 字节的数据:
来自 192.168.66.166 的回复: 字节=32 时间<1ms TTL=128
来自 192.168.66.166 的回复: 字节=32 时间<1ms TTL=128
来自 192.168.66.166 的回复: 字节=32 时间<1ms TTL=128
来自 192.168.66.166 的回复: 字节=32 时间<1ms TTL=128

192.168.66.166 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 0ms, 平均 = 0ms

C:\Users\jackli>_
```

2.再检查能够连上虚拟机上的oracle数据库 sqlplus system/orcl@192.168.66.166:1521/orcl

2-6.plsql连接虚拟机上的oracle数据库



2-7.oracle最重要的两个服务

OracleOraDb11g_home1TNSListener: 监听服务

- 1 主要是留给客户端访问本机时所使用的，例如：在程序开发的过程之中，需要连接数据库，那么如果此服务没有启动或者是错误，那么将导致程序无法连接。

OracleServiceFZTOMASTER: Oracle数据库的实例服务

- 1 在Oracle平台上可以同时配置有多个数据库。使用“DatabaseConfigurationAssistant”，这个工具可以建立更多的数据库，每一个数据库建立完成之后都会按照“OracleServiceSID”这样的服务，如果要想使用MLDN数据库进行数据操作，那么此服务必须打开。

3.oracle的基本使用

3-1.mysql和oracle使用的比较

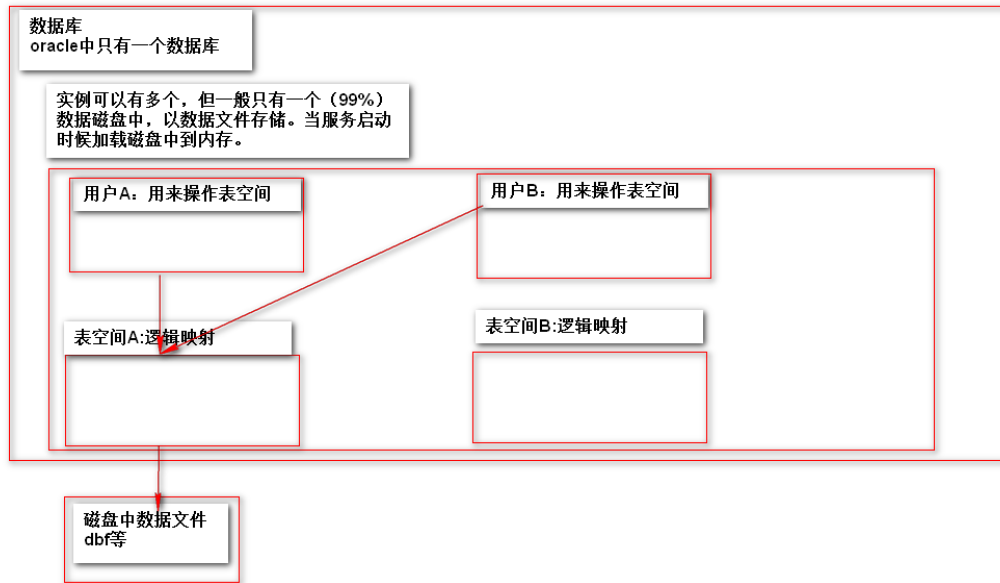
使用mysql数据库

- 创建数据库
- 创建表
- 操作表

使用oracle数据库

- 创建表空间
- 创建用户来操作表空间
- 授权
- 创建表

- 操作表



3-2.oracle的基本查询

```

1  -----使用system账号操作-----
2  -- 1.创建表空间
3  create tablespace jack --表空间名称
4  datafile 'c:\jack.dbf' --指定表空间对应数据文件名称
5  size 10M --初始化大小
6  autoextend on --自动扩展
7  next 10M --每次添加10M
8  -- 2.删除表空间
9  drop tablespace jack including contents and datafiles;
10
11 -- 3.创建用户
12 create user jack identified by jack default tablespace jack;
13 -- 4.删除用户
14 drop user jack cascade;
15 -- 5.授权
16 sys :超级管理员 创建数据库设置密码: orcl
17 system :管理员 创建数据库设置密码: orcl
18 scott :普通用户 默认密码: tiger
19 -- 6.oracle数据库自带角色
20 connect: 连接角色
21 resource: 普通开发角色
22 dba: 超级管理员角色
23 -- 7.授权dba权限
24 grant dba to jack;
25 -----使用jack账号来操作-----
26 -- 1.数据类型介绍和创建表
27 varchar --可变长度 name varchar(10) --'jack' 4
28 varchar2 --可变长度 name varchar2(10) --'jack' 4(推荐使用)
29 char --固定长度 name char(10) --'jack' 10
30 date --mysql date 'yyyy-MM-dd' mysql中的datetime和oracle中date 'yyyy-MM-dd hh24:mi:ss'
31 -- 2.删除表

```

```

32 drop table person
33 -- 3.创建表
34 create table person (
35     pid number(10) primary key, --主键id, oracle没有自增, 使用序列来完成
36     pname varchar2(30) not null,
37     gender char(1) --gender 1 0
38 )
39
40 -- 4.修改表结构
41 -- 4.1添加字段
42 alter table person add address varchar2(50);
43 -- 4.2修改字段
44 alter table person modify address varchar2(100);
45 -- 4.3删除字段
46 alter table person drop column address;
47 -- 5.数据的增删改
48 -- 5.1增加一条数据
49 insert into person values(1, '杰克', 1);
50 commit; -- 事务提交, 增、删、改都要commit
51 rollback; -- 事务回滚
52 select * from person;
53 -- 5.2修改数据, 一定记住要加条件, 不然就是全部更新
54 update person set pname = '肉丝' where pid = 1;
55 commit; -- alt+左右键: 窗口切换
56 -- 5.3删除数据, 一定记住要加条件
57 delete from person where pid = 1;
58 commit;
59 truncate table person;
60 -- delete和truncate的区别
61 delete -- 可以有条件, 但是效率低, 可以反悔
62 truncate -- 没有条件, 但是效率高, 不能反悔
63 -- 6.事务隔离级别
64 -- oracle的事务隔离级别: 以提交的数据为主
65 -- mysql的事务隔离级别: 可重复读取
66 -- 7.序列(重要)
67 -- 7.1序列解决oracle没有自增的问题
68 create sequence seq_person; -- 创建序列
69 drop sequence seq_person; -- 删除序列
70 -- 7.2序列的使用, dual伪表主要就是为了拼接sql语句
71 select seq_person.currval from dual;
72 select seq_person.nextval from dual;
73 -- 7.3在person表中测试
74 select * from person;
75 insert into person values(seq_person.nextval, '杰克', 1);
76 commit;
77
78 -----使用scott用户来操作-----
79 -- 默认是锁定的, 默认密码是: tiger
80 -- 为scott用户解锁
81 alter user scott account unlock;
82 -- 为scott用户设置密码
83 alter user scott identified by tiger;
84 -- 函数分为两类: 单行函数和多行函数

```

```

85 -- 单行函数: 操作10条数据, 使用单行函数, 操作数据的结果也是10条数据
86 -- 多行函数(聚合函数): 操作10条数据, 返回一条数据
87
88 -- 1.单行函数
89 -- 1.1字符串函数: upper、lower
90 select * from emp;
91 select lower(e.ename), e.* from emp e; --lower
92 select upper('jack') from dual; --upper
93 -- 1.2数值函数: round、trunc、mod
94 -- round: 返回四舍五入后的值
95 select round(66.6666, 2), round(1.23, 1), round(15000.22) from dual;
96 -- trunc: 返回x按精度y截取后的值
97 select trunc(45.57, 1.1) from dual;
98 -- mod: 返回x%y的值, 求余
99 select mod(19, 5) from dual;
100 select userenv('language') from dual; -- 查看当前数据库编码
101 -- 1.3日期函数
102 -- sysdate, 无参函数
103 select sysdate from dual;
104 select sysdate+1 from dual; -- 明天此时
105 -- 例子: 查询出emp表中所有员工入职距离现在几周。
106 select round(months_between(sysdate, hiredate)), e.* from emp e;
107 -- 例子: 查询出emp表中所有员工入职距离现在几年。
108 select round(months_between(sysdate, hiredate)/12), e.* from emp e;
109 -- 例子: 查询出emp表中所有员工入职距离现在几周
110 select round((sysdate - hiredate)/7), e.* from emp e;
111 -- 1.4转换函数
112 -- to_char: 日期/数据转换字符串类型
113 select
114 to_char(sysdate, 'd'), -- 每周第几天
115 to_char(sysdate, 'dd'), -- 每月第几天
116 to_char(sysdate, 'ddd'), -- 每年第几天
117 to_char(sysdate, 'ww'), -- 每年第几周
118 to_char(sysdate, 'mm'), -- 每年第几个月
119 to_char(sysdate, 'q'), -- 每年第几季
120 to_char(sysdate, 'yyyy'), -- 年
121 to_char(sysdate, 'yyyy/mm/dd hh24:mi:ss')
122 from dual;
123 -- to_date: 字符串转日期
124 select to_date('2017/10/1', 'yyyy-MM-dd') from dual;
125 -- 1.5通用函数
126 select sal*12+comm, e.* from emp e; -- 由于comm有null值, 参与运算为空, 所以查询不准确
127 select sal*12+nvl(comm, 0), e.* from emp e;
128 -- 1.6条件表达式
129 -- 第一种写法
130 select
131 case e.ename
132 when 'SMITH' then '史密斯' --这个后面不能有逗号
133 when 'ALLEN' then '艾伦'
134 else '杰克'
135 end
136 from emp e;
137 -- 第二种写法

```

```

138 select
139     case
140     when e.ename = 'SMITH' then '史密斯'
141     when e.ename = 'WARD' then '韦德'
142     else '肉丝'
143     end
144 from emp e;
145 -- 例子：判断emp表中员工工资，如果高于3000显示高收入，如果高于1500低于3000显示中等收入
146 select
147     case
148     when e.sal > 3000 then '高收入人群'
149     when e.sal > 1500 then '中等收入人群'
150     else '低收入人群'
151     end
152 from emp e;
153
154 -- 2.多行函数
155 -- 2.1count、max、min、avg、sum
156 select count(*) from emp; -- 查询总数量
157 select sum(sal) from emp; -- 查询总工资
158 select max(sal) from emp; -- 查询最高工资
159 select min(sal) from emp; -- 查询最低工资
160 select avg(sal) from emp; -- 查询平均工资
161 -- 2.1 分组查询
162 -- 例子：查询出每个部门的平均工资
163 -- 思路：
164 -- 1.确定要使用的数据表；
165 -- 2.确定已知的关联字段
166 -- 3.确定要查询哪些值
167 select avg(sal) from emp group by deptno;
168 -- 例子：查询出平均工资高于2000的部门信息
169 select avg(sal), e.deptno from emp e group by e.deptno having avg(sal) > 2000 order by
e.deptno;
170 -- oracle的sql执行顺序
171 [确定要显示的数据列] SELECT [DISTINCT] * | 列[别名], 列[别名]...
172 [确定数据来源(行与列的集合)] FROM 表名称[别名], 表名称[别名], ...
173 [针对于数据行进行筛选] [WHERE 限定条件(s)]
174 [针对于筛选的行分组] [GROUP BY 分组字段, 分组字段, 分组字段, ...]
175 [针对于筛选大的行分组] [HAVING 分组过滤]
176 [对选定数据的行与列进行排序] [ORDER BY 排序字段[ASC|DESC], 排序字段[ASC|DESC], ...]
177 -- HAVING是在GROUP BY分组之后才执行的筛选，在HAVING里面可以直接使用统计函数(count、sum、min、
max、avg)
178 -- 例子：查询出每个部门工资高于800的员工的平均工资,然后再查询出平均工资高于2000的部门
179 -- 1.查询出每个部门工资高于800的员工的平均工资
180 select deptno, avg(sal) from emp where sal > 800 group by deptno;
181 -- 2.查询出平均工资高于2000的部门
182 select deptno, avg(sal) from emp where sal > 800 group by deptno having avg(sal) > 2000;
183
184 -- 3.多表查询
185 -- 多表查询分为：内连接、外连接、子查询
186 -- 3.1内连接使用
187 -- 例子：查询员工表和部门表
188 select * from emp e, dept d; -- 如果没有关联条件，那么会出现笛卡尔积

```



```

189 select * from emp e, dept d where e.deptno = d.deptno;
190 -- 例子: 查询出所有部门, 以及部门下的员工信息。
191 -- +号的使用, emp表: 非全量表, dept表: 全量表
192 select * from emp e, dept d where e.deptno(+) = d.deptno;
193 -- 例子: 查询所有员工信息, 以及员工所属部门
194 -- 显式内连接
195 select * from emp e, dept d where e.deptno = d.deptno;
196 -- 隐式内连接
197 select * from emp e inner join dept d on e.deptno = d.deptno;
198
199 -- 3.2外连接使用
200 -- 例子: 查询出员工姓名, 员工领导姓名
201 select e1.ename "员工姓名", e2.ename "领导姓名" from emp e1, emp e2 where e1.mgr = e2.empno;
202 -- 注意使用双引号
203 -- 例子: 查询出员工姓名, 员工部门名称, 员工领导姓名, 员工领导部门名称
204 select e.ename "员工姓名", d.dname "员工部门名称", e2.ename "员工领导姓名", d2.dname "员工领导
205 部门名称"
206 from emp e, emp e2, dept d, dept d2
207 where e.mgr = e2.empno and e.deptno = d.deptno and e2.deptno = d2.deptno;
208
209 -- 4.子查询
210 -- 子查询 主要是子句查询结果 提供主句使用
211 -- 4.1 单行单列子查询
212 -- 例子: 查询出工资和SCOTT一样的员工信息
213 select * from emp where sal = (select sal from emp where ename = 'SCOTT');
214 -- 4.2 多行单列子查询
215 -- 例子: 查询出工资和10号部门任意员工一样的员工信息
216 select * from emp where sal in (select sal from emp where deptno = 10);
217 -- 例子: 查询出每个部门最低工资, 和最低工资员工姓名, 和该员工所在部门名称
218 select e.ename, e.sal, d.dname from emp e, dept d
219 where e.sal in(select min(sal) from emp group by deptno) and e.deptno = d.deptno;
220
221 -- 5.oracle中的分页
222 -- mysql中的分页是limit, 而oracle中的分页需要使用rownum伪列来完成, 伪列是随着select查询而产生的
223 -- rownum: 1.取得第一行数据, 2.取得前n行的数据
224 -- 分页公式
225 -- currentPage, 表示的是当前所在页;
226 -- lineSize, 表示每页显示的数据行;
227 SELECT * FROM (
228 SELECT ROWNUM rn, 列, ... FROM 表名称 WHERE ROWNUM <= currentPage * lineSize) temp
229 WHERE temp.rn > (currentPage - 1) * lineSize;
230
231 select rownum, e.* from emp e;
232 -- 例子: 查询工资最高的5条记录
233 select rownum, temp.sal from (select sal from emp order by sal desc) temp where rownum < 6;
234 select * from (select rownum rn, sal from emp where rownum <= 2*5) temp where temp.rn > 5;
235 -- 使用公式
236
237 -- 例子: emp表工资倒叙排列后, 每页五条记录, 查询第二页
238 select eee.* from (
239 select rownum rn, ee.* from (
240 select e.* from emp e order by sal desc) ee
241 ) eee where eee.rn > 5 and eee.rn <= 10

```

```

239
240 select * from (select rownum rn, e.* from emp e where rownum <= 2*5) temp where temp.rn>5;
241
242 ----重点
243 -- 1.oracle了解
244 -- 2.oracle会使用
245 -- 3.oracle数据类型 操作表 操作数据 (重点)
246 -- 4.多行函数 多行 (重点) 分组统计
247 -- 5.多表关联查询 子查询 分页

```

4.oracle的高级应用

4-1.视图

```

1  -- 1.视图
2  -- 查询scott用户的emp表的数据放入myemp表中(dba权限)
3  -- create table 表名 as 查询语句;
4  -- is、as、for
5  create table myemp as select * from scott.emp;
6  select * from myemp;
7
8  -- 视图: 1.封装复杂sql语句, 2.隐藏敏感信息
9  -- 两种创建方式
10 create view view_myemp as select * from myemp;
11 create or replace view view_myemp as select e.empno, e.ename, e.mgr, e.deptno from myemp e;
12 -- 创建只读视图, 就是只能查询, 不能增、删、改
13 create view view_myemp as select * from myemp with read only;
14 -- 封装一个分页查询的视图
15 -- select * from (select rownum rn, m.* from myemp m where rownum <= currentPage*pageSize)
16 -- temp where temp.rn > (currentPage-1)*pageSize;
17 create or replace view view_queryPage as select * from (select rownum rn, m.* from myemp m
18 where rownum <= 2 * 5) temp
19 where temp.rn > 5 with read only;
20 -- 查询视图
21 select * from view_myemp;
22 select * from view_queryPage;
23 -- 更新数据, 一般视图只用作查询, 不推荐使用更新操作
24 update view_myemp set sal=800 where ename='SMITH';
25 -- 删除视图
26 drop view view_myemp;

```

4-2.索引

```

1  -- 2.索引
2  -- 索引是可以优化查询速度的, 特别是当数据量很大的时候
3  -- 单列索引
4  create index index_pname on person(pname);
5  -- 复合索引
6  create index index_pname_gender on person(pname, gender);
7  -- 测试单列索引

```

```

8 select * from person where pname='xxx'; --使用了index_pname索引
9 select * from person where upper('pname')='xxx'; --未用index_pname索引
10 select * from person where pname=upper('xxx'); --使用了index_pname索引
11 select * from person where pname like '%xxx%'; --未用index_pname索引
12 -- 总结: 单列索引, 查询条件列 如果有使用到单行函数、模糊查询, 不走索引
13 -- 测试复合索引
14 select * from person where pname='xxx' and gender='xxx'; --使用了index_pname_gender索引
15 select * from person where upper('pname')='xxx' and gender='xxx'; --未使用index_pname_gender索引
16 select * from person where pname=upper('xxx') and gender='xxx'; --使用了index_pname_gender索引
17 select * from person where pname like '%xxx%' and gender='xxx'; --未使用index_pname_gender索引
18 select * from person where pname='xxx' and upper('gender')='xxx'; --使用了index_pname_gender索引
19 select * from person where pname='xxx' and gender=upper('xxx'); --使用了index_pname_gender索引
20 select * from person where pname='xxx' and gender like '%xxx%'; --使用了index_pname_gender索引
21 -- 总结: 复合索引 查询条件列 (第二个列) 如果有使用单行函数、模糊查询 不走索引
22 select * from person;
23 -- 删除索引
24 drop index index_pname_gender;

```

4-3.plsql编程

4-3-1.声明变量

```

1 -- 3.pl/sql数据库编程语言
2 -- plsql: Procedure language sql: 过程化语言, 数据库端开发语言
3 -- 将以前业务逻辑中复杂业务代码, 通过这种过程化语言来实现。可以在程序中只需要调用一次, 既可以完成业务功能。
4 -- 语法
5 declare
6     -- 声明变量(普通变量、引用型变量、记录型变量)
7 begin
8     -- DML语句(insert、update、select、delete)
9 end;
10
11 -- 3.1声明变量
12 declare
13     myname varchar2(20):='杰克'; -- 普通变量
14     myage constant number(10):=12; -- 常量
15     myname2 myemp.ename%type; -- 引用型变量, 存放查询出来的 一个数据
16     v_row myemp%rowtype; -- 记录型变量, 存放一行数据, 实体对象.列名
17 begin
18     myname := '杰克';
19     -- myage := 22; 常量不能再次赋值
20     -- myname2 := '肉丝';
21     select ename into myname2 from myemp where empno = 7369;
22     select * into v_row from myemp where empno = 7369;
23     -- v_row不能直接输出, 需要输出里面的值, v_row.ename等
24     dbms_output.put_line(myname || '-' || myage || '-' || myname2 || '-' || v_row.deptno);
25 end;

```

4-3-2.plsql中的if判断

```
1  -- 3.2plsql中的if判断
2  -- 输入小于18的数字，输出未成年 输入大于18小于40的数字，输出中年人 输入大于40的数字，输出老年人
3  declare
4      age number(2) := &age;--定义变量并提示输入信息
5  begin
6      if age<18 then dbms_output.put_line('未成年人');
7      elsif age<40 then dbms_output.put_line('中年人'); -- 注意是elsif而不是elseif
8      else dbms_output.put_line('老年人');
9      end if; -- 结束if判断
10 end;
11
12 -- plsql中的loop循环
13 -- 用三种方式输出1到10是个数字
14 -- 方式一
15 declare
16 begin
17     for i in 1..10
18     loop
19         dbms_output.put_line(i);
20     end loop;
21 end;
22 -- 方式二
23 declare
24     num number(10) := 1;
25 begin
26     loop
27         exit when num > 10;
28         dbms_output.put_line(num);
29         num := num+1;
30     end loop;
31 end;
32 -- 方式三
33 declare
34     num number(10) := 1;
35 begin
36     while num <= 10
37     loop
38         dbms_output.put_line(num);
39         num := num + 1;
40     end loop;
41 end;
```

4-3-3.游标

```
1  -- 3.3游标cursor
2  -- 游标作用：可以存放多行数据
3  -- 定义游标可以有参数也可以没有参数
4  -- 应用：循环游标把每一行数据放入记录型变量中
5  -- 语法
```

```

6 declare
7     cursor cursor_myemp[(参数名称 参数数据类型)] is 查询语句;
8 begin
9     open cursor_myemp[(参数名称 参数数据类型)];
10    loop
11        exit when cursor_myemp%notfound; -- 没有数据则退出循环
12        fetch cursor_myemp into 记录性变量;
13        -- 打印数据
14    end loop;
15    close cursor_myemp;
16 end;
17 -- 输出emp表中所有员工的姓名
18 declare
19     v_row myemp%rowtype; -- 记录型变量
20     cursor cursor_myemp is select * from myemp;
21 begin
22     open cursor_myemp;
23     loop
24         exit when cursor_myemp%notfound; -- 没有数据则退出循环
25         fetch cursor_myemp into v_row;
26         dbms_output.put_line(v_row.empno || '-' || v_row.ename);
27     end loop;
28     close cursor_myemp;
29 end;
30 -- 给指定部门员工涨工资
31 declare
32     v_row myemp%rowtype;
33     cursor cursor_myemp(v_deptno myemp.deptno%type) is select * from myemp where deptno =
v_deptno;
34 begin
35     open cursor_myemp(&mydeptno);
36     loop
37         exit when cursor_myemp%notfound; -- 没有数据则退出循环
38         fetch cursor_myemp into v_row;
39         -- dbms_output.put_line(v_row.empno || '-' || v_row.ename || '-' || v_row.sal || '-'
|| v_row.deptno);
40         update myemp set sal = sal + 1 where empno = v_row.empno;
41     end loop;
42     close cursor_myemp;
43 end;
44 -- 测试
45 select * from myemp;

```

4-4.存储过程

```

1  -- 4.存储过程
2  -- 将复杂业务逻辑写到这个过程语言中，通过java代码调用这个存储过程
3  -- 存储过程有名称 可以有输入参数 也可以没有输入参数 （java输入参数）
4  -- 输入参数的in关键字可以省略
5  -- 输出参数一定要out关键字
6  -- 存储过程可以有输出参数 也可以没有输出参数(处理结果)
7  -- 语法

```

```

8  create [or replace] procedure 存储过程名称[(输入参数名称 参数类型, 输出参数名称 out 参数类型)]
9  is|as
10  --声明变量
11  begin
12  --DML语句
13  end;
14
15  -- 例子: 给指定员工涨100块钱
16  create or replace procedure pro_addsal(v_empno myemp.empno%type)
17  is
18  begin
19      update myemp set sal = sal +100 where empno = v_empno;
20      commit;
21  end;
22  -- 测试存储过程
23  declare
24  begin
25      pro_addsal(7369);
26  end;
27  -- 查询数据
28  select * from myemp;
29
30  -- 例子: 使用存储过程来算年薪 (使用out类型参数)
31  create or replace procedure pro_calcsal(v_empno myemp.empno%type, v_yearsal out
myemp.sal%type)
32  is
33  begin
34      select sal*12+nvl(comm, 0) into v_yearsal from myemp where empno = v_empno;
35  end;
36  -- 测试存储过程
37  -- 方式一
38  declare
39      mysal myemp.sal%type;
40  begin
41      pro_calcsal(7369, mysal);
42      dbms_output.put_line(mysal);
43  end;
44  -- 方式二: 通过右击存储过程, 进行测试

```

4-5.函数

```

1  -- 5.函数
2  -- 存储过程和函数的区别
3  -- 存储过程有名称 可以有输入参数 也可以没有输入参数 (java输入参数)
4  -- 存储过程java程序需要得到结果, 必须要out参数
5  -- 函数有名称 可以有输入参数 也可以没有输入参数 (java输入参数)
6  -- 函数, 必须return返回值 可以有输出参数 也可以没有参数
7
8  -- 一般情况都是存储过程中调用函数
9  -- 语法
10 create [or replace] function 函数名称[(输入参数名称 参数数据类型, 输出参数名称 out 参数数据类型)]
11 return 数据类型

```

```

12  is|as
13      -- 声明变量
14  begin
15      -- DML语句
16      return 结果;
17  end;
18  -- 例子: 通过函数(自定义)实现计算指定员工的年薪
19  create or replace function func_myemp(v_empno myemp.empno%type)
20  return number
21  as
22      mysal myemp.sal%type;
23  begin
24      select sal*12+nvl(comm, 0) into mysal from myemp where empno = v_empno;
25      return mysal;
26  end;
27  -- 测试函数
28  -- 方式一
29  declare
30      mysal myemp.sal%type;
31  begin
32      mysal := func_myemp(7369);
33      dbms_output.put_line(mysal);
34  end;
35  -- 方式二
36  select func_myemp(7369) from dual;

```

4-6.触发器

```

1  -- 6.触发器
2  -- 在数据库上建立触发器(基于表), 编写plsql过程语言, 当操作某一个表的时候(改变此表数据的时候oracle自动执行触发器)
3  -- insert、update、delete
4  -- 最新登录时间, 登录日志表记录登录时间
5  -- 语法
6  create or replace trigger 触发器名称
7  before|after
8  insert | update of 字段名 | delete
9  on 表名
10 [for each row]
11 declare
12     -- 声明变量
13 begin
14     -- 执行具体业务
15 end;
16
17 -- 例子: 插入一条记录, 输出一个新员工入职
18 create or replace trigger myperson
19 after
20 insert
21 on person
22 declare
23 begin

```

```

24     dbms_output.put_line('新员工入职了...');
25 end;
26 -- 测试
27 select * from person;
28 insert into person values(3, '肉丝', 1);
29 commit;
30 -- 例子：不能给员工降薪
31 -- 删除触发器
32 drop trigger myperson;
33 create or replace trigger myperson
34 before
35 update of sal
36 on myemp
37 for each row
38 declare
39 begin
40     -- 更新前的薪水，更新后的薪水
41     -- :old更新前的数据，:new更新后的数据
42     if :old.sal >= :new.sal then
43         --错误提示框来提示不能给员工降薪
44         --错误编号 -20000 - -20999
45         raise_application_error('-20000', '该员工不能降薪');
46     end if;
47 end;
48 -- 测试
49 select * from myemp;
50 update myemp set sal = sal -100 where empno = 7369;
51 commit;
52
53 -- 触发器实现主键自增
54 -- 跟mysql的主键自增一样的效果
55 drop trigger myperson;
56 create trigger myperson
57 before
58 insert
59 on person
60 for each row
61 declare
62 begin
63     --在插入表之前 为:new新记录 主键id赋值
64     select seq_person.nextval into :new.pid from dual;
65 end;
66 -- 测试
67 insert into person(pname) values('杰瑞');
68 commit;
69 select * from person;

```

5.Java调用存储过程和函数

创建工程 com.jack.oralce_jdbc **添加maven依赖**

```
1 <?xml version="1.0" encoding="UTF-8"?>
```



```

2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
6
7     <groupId>com.jack</groupId>
8     <artifactId>oracle_jdbc</artifactId>
9     <version>1.0-SNAPSHOT</version>
10    <packaging>jar</packaging>
11
12    <!--oracle驱动包 ojdbc14: oracle10g   ojdbc6: oracle11g-->
13    <dependencies>
14        <dependency>
15            <groupId>com.oracle</groupId>
16            <artifactId>ojdbc14</artifactId>
17            <version>10.2.0.4.0</version>
18        </dependency>
19        <dependency>
20            <groupId>junit</groupId>
21            <artifactId>junit</artifactId>
22            <version>4.12</version>
23        </dependency>
24    </dependencies>
25 </project>

```

创建包com.jack.oracle, 创建类OracleJdbc.java

```

1 public class OracleJdbc {
2
3     /**
4      * 测试连接oracle jdbc
5      * @throws Exception
6      */
7     @Test
8     public void javaCallOracle() throws Exception {
9         // 加载数据库驱动
10        Class.forName("oracle.jdbc.driver.OracleDriver");
11        // 得到Connection连接
12        Connection connection =
DriverManager.getConnection("jdbc:oracle:thin:@192.168.66.166:1521:orcl", "jack", "jack");
13        // 得到预编译的Statement对象
14        PreparedStatement pstmt = connection.prepareStatement("select * from myemp where
empno = ?");
15        // 给参数赋值
16        pstmt.setInt(1, 7369);
17        // 执行数据库查询操作
18        ResultSet rs = pstmt.executeQuery();
19        // 输出结果
20        while(rs.next()) {
21            System.out.println(rs.getString("ename"));
22        }
23    }
24 }

```

```

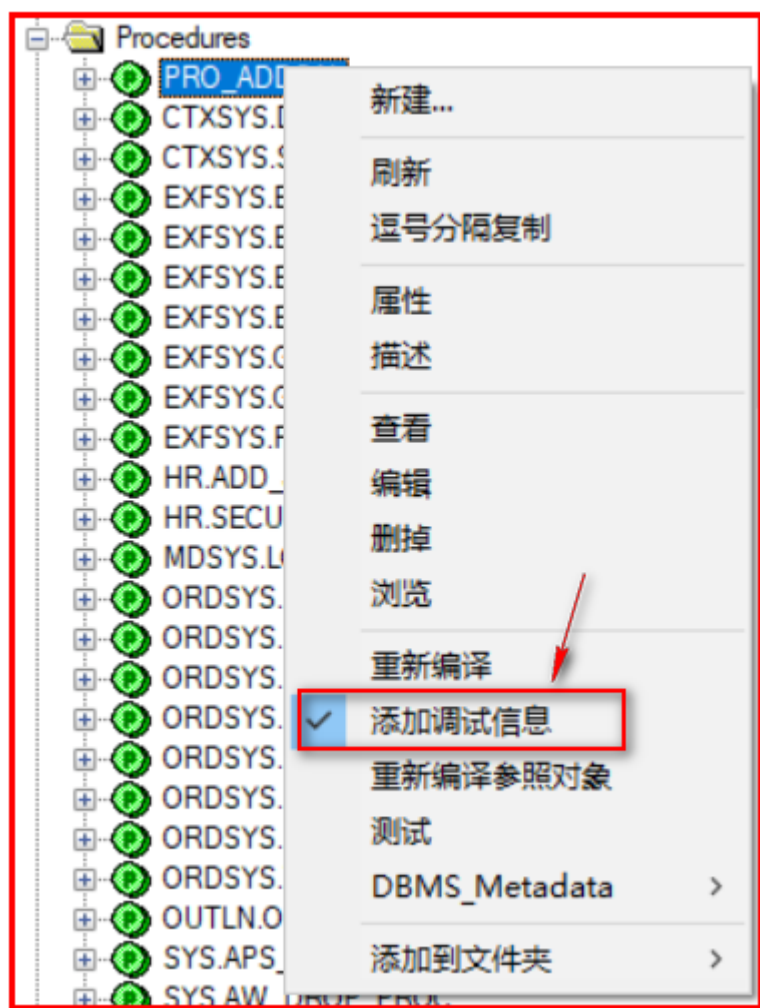
23         // 释放资源
24         rs.close();
25         pstmt.close();
26         connection.close();
27     }
28
29     /**
30     * 测试连接oracle 存储过程pro_calcsal(v_empno myemp.empno%type, v_yearsal out
myemp.sal%type)
31     * @throws Exception
32     */
33     @Test
34     public void javaCallProcedure() throws Exception {
35         // 加载数据库驱动
36         Class.forName("oracle.jdbc.driver.OracleDriver");
37         // 得到Connection连接
38         Connection connection =
DriverManager.getConnection("jdbc:oracle:thin:@192.168.66.166:1521:orcl", "jack", "jack");
39         // 得到预编译的Statement对象
40         /**
41         *      {?= call <procedure-name>[(<arg1>,<arg2>, ...)]}
42         *      {call <procedure-name>[(<arg1>,<arg2>, ...)]}
43         */
44         CallableStatement callableStatement = connection.prepareCall("{call
pro_calcsal(?,?)}");
45         // 给参数赋值
46         callableStatement.setInt(1, 7369);
47         // 输出参数 注册数据类型
48         callableStatement.registerOutParameter(2, OracleTypes.NUMBER);
49         // 执行数据库查询操作
50         callableStatement.execute();
51         // 输出结果
52         Object object = callableStatement.getObject(2);
53         System.out.println(object);
54         // 释放资源
55         callableStatement.close();
56         connection.close();
57     }
58
59     /**
60     * 测试连接oracle 函数func_myemp(v_empno myemp.empno%type)
61     */
62     @Test
63     public void javaCallFun() throws Exception {
64         // 加载数据库驱动
65         Class.forName("oracle.jdbc.driver.OracleDriver");
66         // 得到Connection连接
67         Connection connection =
DriverManager.getConnection("jdbc:oracle:thin:@192.168.66.166:1521:orcl", "jack", "jack");
68         // 得到预编译的Statement对象
69         /**
70         *      {?= call <procedure-name>[(<arg1>,<arg2>, ...)]}
71
72         *      {call <procedure-name>[(<arg1>,<arg2>, ...)]}

```

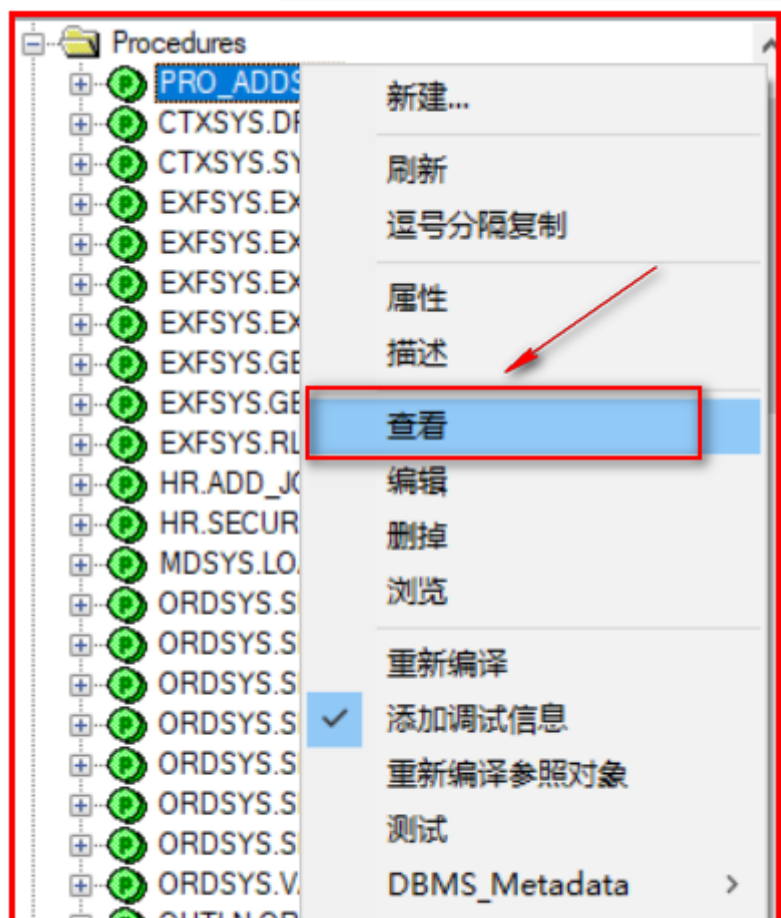
```
72         */
73
74         CallableStatement callableStatement = connection.prepareCall("{?= call
func_myemp(?)}");//
75         //输出参数 注册数据类型
76         callableStatement.registerOutParameter(1, OracleTypes.NUMBER);
77         //给参数赋值
78         callableStatement.setObject(2, 7369);
79         //执行数据库查询操作
80         callableStatement.execute();
81         //输出结果
82         Object object = callableStatement.getObject(1);
83         System.out.println(object);
84
85         //释放资源
86         callableStatement.close();
87         connection.close();
88     }
89 }
```

6.oracle调试

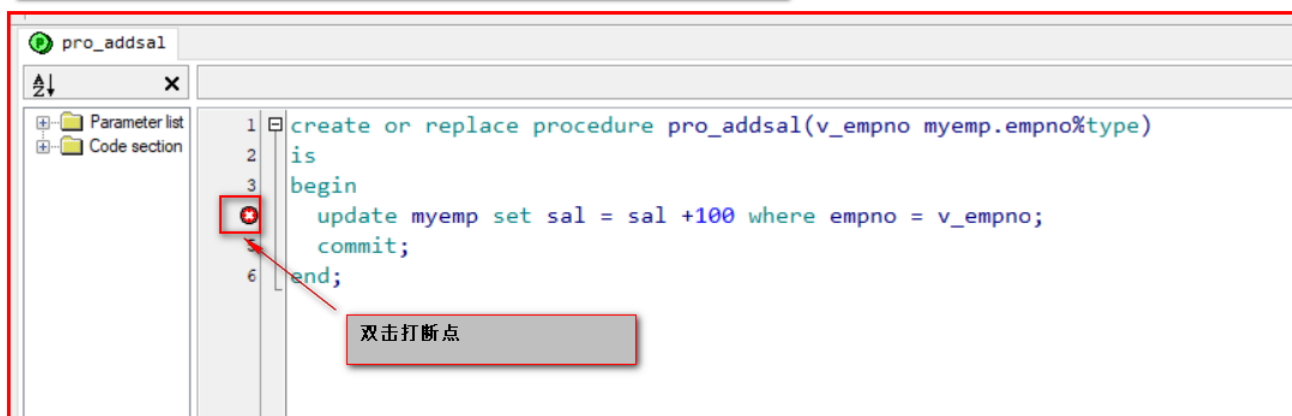
右键选择添加调试信息



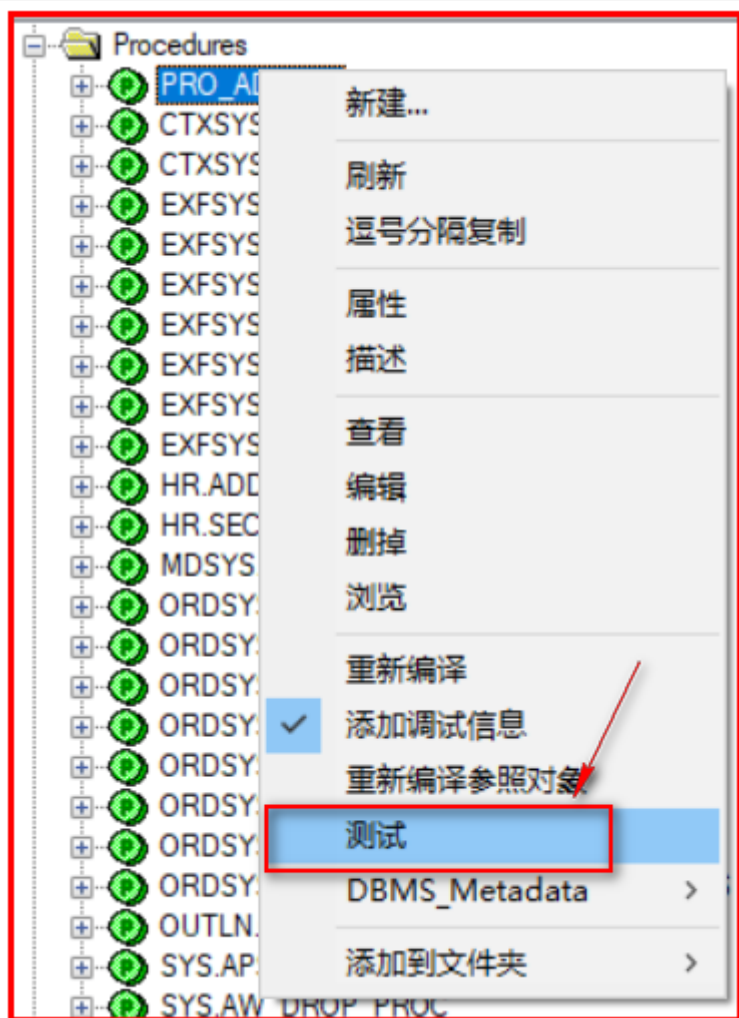
右键选择查看



打断点



测试



输入测试值，然后点击开始测试

测试脚本 DBMS 输出 统计表 概览图 跟踪



```
1 begin
2   -- Call the procedure
3   pro_addsal(v_empno => :v_empno);
4 end;
```

点击开始测试

输入测试值

	变量	类型	值
▶	✓ v_empno	Integer	7369
*	✓		