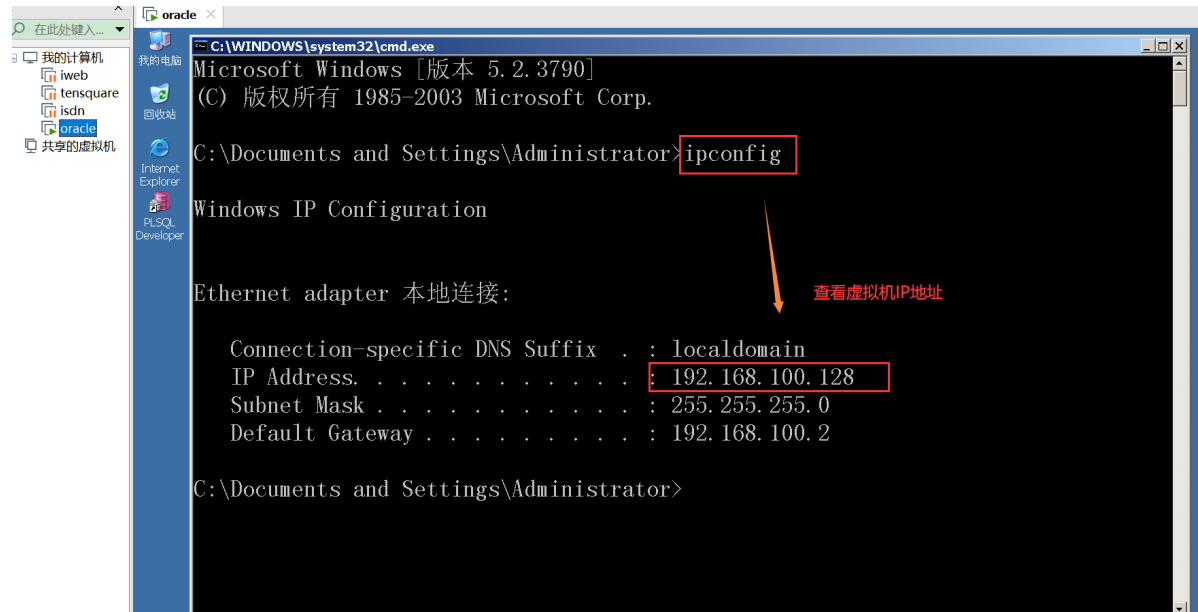


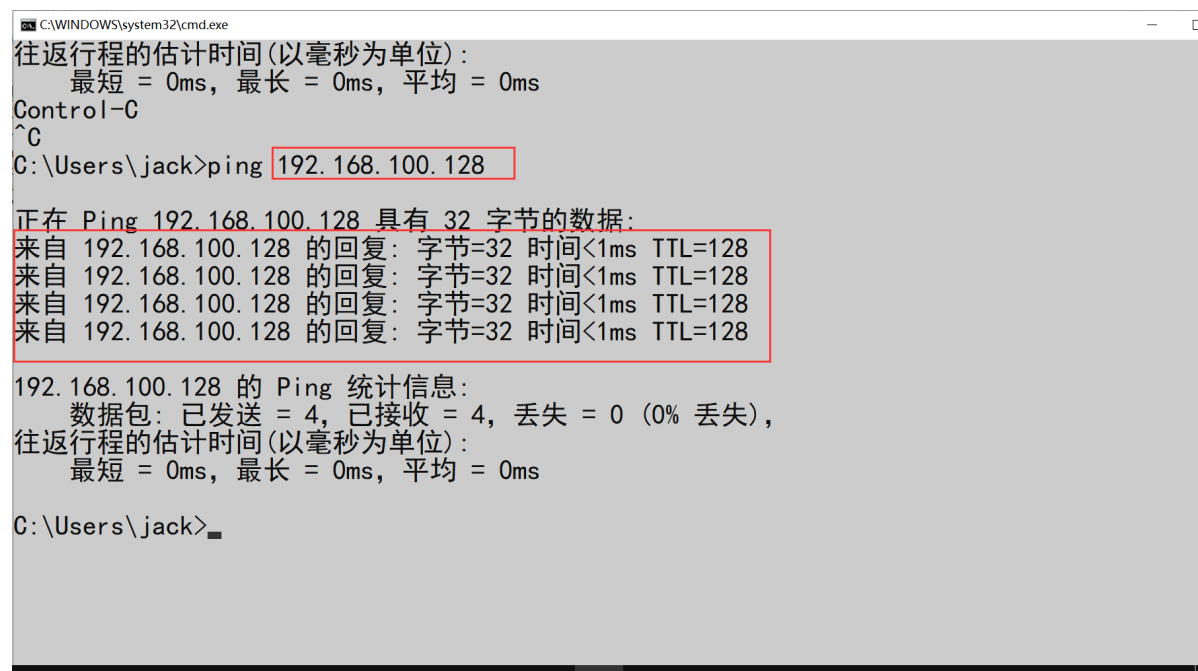
JDBC

准备工作

查看虚拟机的IP地址

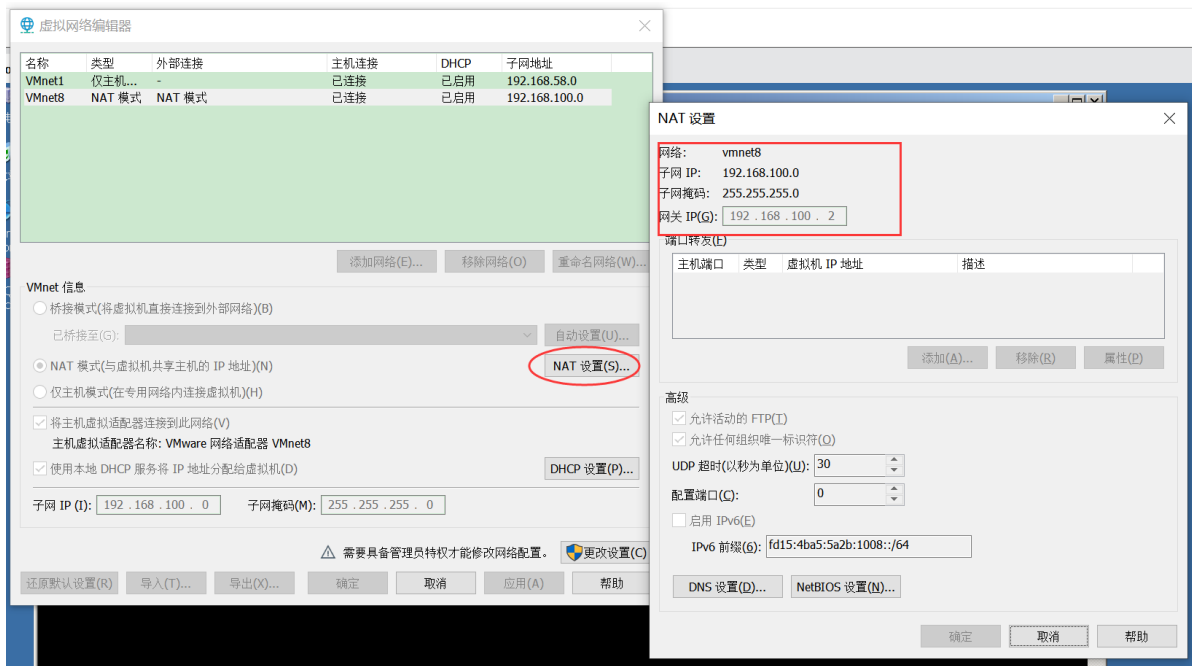
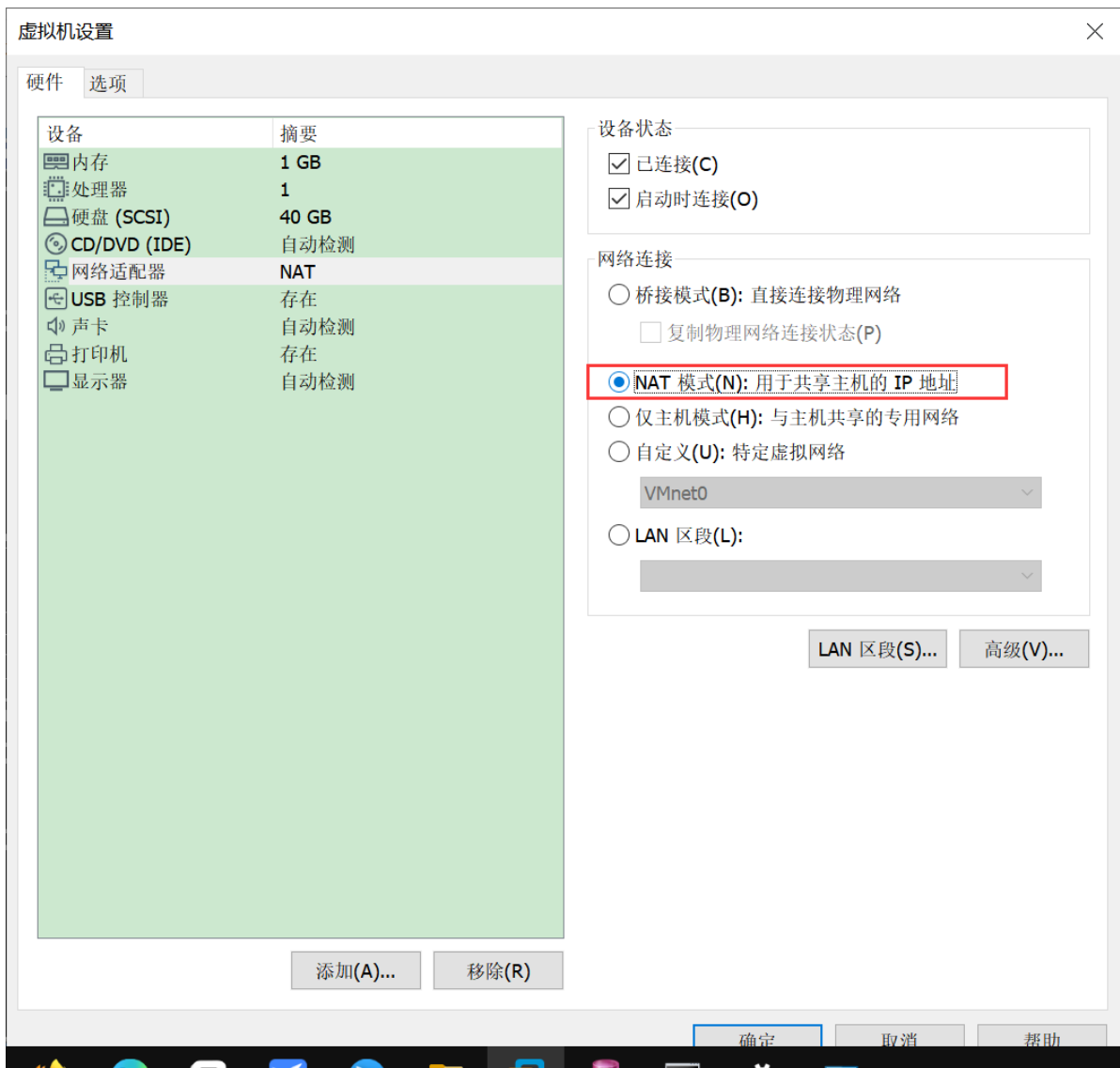


确认网络是否可用



网络不通的解决办法

第一步



第二步：如果不在同一个网段

网络: vmnet8
子网 IP: 192.168.100.0 ✓
子网掩码: 255.255.255.0 ✓
网关 IP(G): 192.168.100.2

真机

IPv4 地址: 192.168.100.1 ✓
IPv4 子网掩码: 255.255.255.0 ✓

Connection-specific DNS Suffix . : localdomain
IP Address. : 192.168.100.128 ✓
Subnet Mask : 255.255.255.0 ✓
Default Gateway : 192.168.100.2

第三步

关闭虚拟主机，还原网络设置

虚拟网络编辑器

名称	类型	外部连接	主机连接	DHCP	子网地址
VMnet0	桥接模式	自动桥接	-	-	-
VMnet1	仅主机...	-	已连接	已启用	192.168.58.0
VMnet8	NAT 模式	NAT 模式	已连接	已启用	192.168.100.0

添加网络(E)...

移除网络(O)

重命名网络(W)...

VMnet 信息

☐ 桥接模式(将虚拟机直接连接到外部网络)(B)

已桥接至(G): 自动

自动设置(U)...

☒ NAT 模式(与虚拟机共享主机的 IP 地址)(N)

NAT 设置(S)...

☐ 仅主机模式(在专用网络内连接虚拟机)(H)

☒ 将主机虚拟适配器连接到此网络(V)

主机虚拟适配器名称: VMware 网络适配器 VMnet8

☒ 使用本地 DHCP 服务将 IP 地址分配给虚拟机(D)

DHCP 设置(P)...

子网 IP (I): 192.168.100.0

子网掩码(M): 255.255.255.0

还原默认设置(R)

导入(I)...

导出(X)...

确定

取消

应用(A)

帮助

还原后网段会重新分配。

虚拟网络编辑器

名称	类型	外部连接	主机连接	DHCP	子网地址
VMnet0	桥接模式	自动桥接	-	-	-
VMnet1	仅主机...	-	已连接	已启用	192.168.58.0
VMnet8	NAT 模式	NAT 模式	已连接	已启用	192.168.100.0

添加网络(E)... 移除网络(O) 重命名网络(W)...

VMnet 信息

☐ 桥接模式(将虚拟机直接连接到外部网络)(B)

已桥接至(G): 自动
 自动设置(U)...

☒ NAT 模式(与虚拟机共享主机的 IP 地址)(N)
 NAT 设置(S)...

☐ 仅主机模式(在专用网络内连接虚拟机)(H)

☒ 将主机虚拟适配器连接到此网络(V)
 主机虚拟适配器名称: VMware 网络适配器 VMnet8

☒ 使用本地 DHCP 服务将 IP 地址分配给虚拟机(D)
 DHCP 设置(P)...

子网 IP (I): 192 . 168 . 100 . 0
 子网掩码(M): 255 . 255 . 255 . 0

还原默认设置(R) 导入(I)... 导出(X)... 确定 取消 应用(A) 帮助

这里会重新分配一个网络

可用手动改回来

再重新确认步骤。

确认数据库是否能够远程客户端操作

确认虚机中数据库的安装路径：C:\oracle\product\10.2.0\db_1\NETWORK\ADMIN

tnsnames.ora 修改 HOST = 你的虚机IP

```
# tnsnames.ora Network Configuration File:
C:\oracle\product\10.2.0\db_1\network\admin\tnsnames.ora
# Generated by Oracle configuration tools.

ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.100.128)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl)
    )
  )

EXTPROC_CONNECTION_DATA =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1))
    )
  )
```

```

)
(CONNECT_DATA =
  (SID = PLSExtProc)
  (PRESENTATION = RO)
)
)

```

listener.ora: 修改 HOST = 你的IP

```

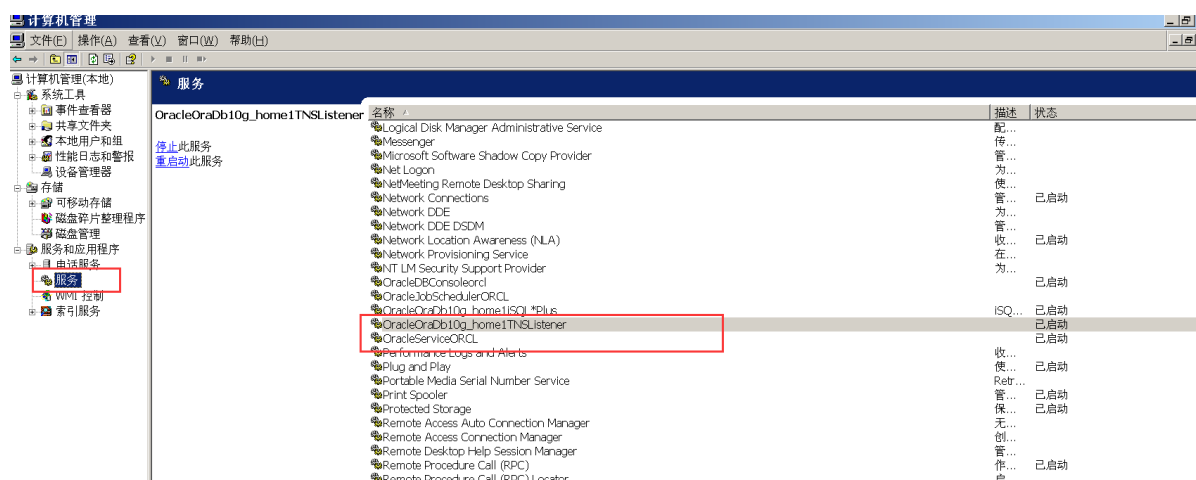
# listener.ora Network Configuration File:
C:\oracle\product\10.2.0\db_1\network\admin\listener.ora
# Generated by oracle configuration tools.

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\product\10.2.0\db_1)
      (PROGRAM = extproc)
    )
  )

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1))
      (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.100.128)(PORT = 1521))
    )
  )

```

在服务重启OracleServiceORCL OracleOraDb10g_home1TNSListener



连接数据库

```
C:\WINDOWS\system32\cmd.exe - sqlplus / as sysdba
C:\Documents and Settings\Administrator>sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on 星期五 5月 7 09:03:31 2021

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

已连接到空闲例程。

SQL> startup
ORACLE 例程已经启动。

Total System Global Area 289406976 bytes
Fixed Size 1248600 bytes
Variable Size 113246888 bytes
Database Buffers 171966464 bytes
Redo Buffers 2945024 bytes
数据库装载完毕。
数据库已经打开。
SQL> ^A_
```

免密、使用DBA的身份登录
只能在Oracle的服务器使用
不能远程

启动数据库

查看用户

```
C:\WINDOWS\system32\cmd.exe - sqlplus / as sysdba
数据库已经打开。
SQL> select * from all_users;

USERNAME                                USER_ID CREATED
-----
XIAOHEI                                  61 11-1月 -21
BI                                        60 08-1月 -21
PM                                        59 08-1月 -21
SH                                        58 08-1月 -21
IX                                        57 08-1月 -21
OE                                        56 08-1月 -21
HR                                        55 08-1月 -21
SCOTT                                     54 30-8月 -05
MGMT_VIEW                               53 30-8月 -05
JACK                                     62 20-3月 -21
MDDATA                                  50 30-8月 -05

USERNAME                                USER_ID CREATED
```

这两个用户都是Oracle自带的

开启用户： 1. 解锁 2. 设置密码

```
SQL> alter user scott account unlock;
用户已更改。

SQL> alter user scott identified by "123456";
用户已更改。
```

测试远程登录

```
SQL> connect scott/123456@orcl;
已连接。
SQL> a_
```

 tnsnames文件中的 service_name

使用java连接数据库

1. 引用jar包：ojdbc14.jar
2. 编写连接数据库代码

```
public class Test {

    public static void main(String[] args) throws Exception {

        /*
        1. 加载驱动 ojdbc14.jar 中
        2. 连接数据库获取会话
        */

        // 数据库连接的 4 个基本参数: url(proto.IP:PORT) user pwd driver
        String user = "scott";
        String pwd = "123456";
        // oracle.jdbc.OracleDriver 11g以上版本
        oracle.jdbc.driver.OracleDriver 11g一下
        String driver = "oracle.jdbc.OracleDriver"; // OracleDriver(记住)
        // jdbc:mysql
        String url = "jdbc:oracle:thin:@192.168.100.128:1521:orcl"; // conn
        scott/123456@orcl

        Connection conn = null; // 数据库会话
        try {
            // 核心代码
            Class.forName(driver); // 加载驱动
            conn = DriverManager.getConnection(url, user, pwd); // 获取会话
            // 测试
            System.out.println(conn);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭数据库
            if (conn != null && !conn.isClosed()) {
                conn.close();
            }
        }

    }

}
```

```

public class DBUtil {

    private String user = "scott";
    private String pwd = "123456";
    private String driver = "oracle.jdbc.OracleDriver"; // OracleDriver(记住)
    private String url = "jdbc:oracle:thin:@192.168.100.128:1521:orcl"; // conn
    scott/123456@orcl

    private Connection conn;

    /**
     * 连接数据库
     *
     * @return
     */
    public Connection getConnection() throws Exception {
        Class.forName(driver); // 加载驱动
        conn = DriverManager.getConnection(url, user, pwd); // 获取会话
        return conn;
    }

    /**
     * 关闭数据库
     */
    public void close() throws Exception {
        if (conn != null && !conn.isClosed()) {
            conn.close();
        }
    }

}

```

简单查询

```

public class Test {
    public static void main(String[] args) throws Exception {

        DBUtil util = new DBUtil();

        try {
            Connection conn = util.getConnection();

            String sql = "select * from emp";
            /**
             * 1. 发送语句
             * 2. 发送执行指令
             * 3. 接收返回结果集
             */
            Statement st = conn.createStatement(); // 这个实例可用发送语句
            ResultSet rs = st.executeQuery(sql); // 发送 并 查询 返回结果集

            // 遍历结果集
            while (rs.next()) {
                Integer empno = rs.getInt("EMPNO");
                String ename = rs.getString("ENAME");
            }
        }
    }
}

```



```

        String job = rs.getString("JOB");
        Integer mgr = rs.getInt("mgr");
        Date hiredate = rs.getDate("HIREDATE");
        Double sal = rs.getDouble("SAL");
        Double comm = rs.getDouble("COMM");
        Integer deptno = rs.getInt("DEPTNO");
        System.out.println(empno + "\t" + ename + "\t" + sal + "\t" +
hiredate);
    }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        util.close();
    }
}
}
}

```

封装结果集

1. 构建Emp的bean
2. 将emp实例保存到List中

```

package com.iweb.lesson02;

import java.util.Date;

/**
 * 作者: jack
 * 时间: 2021-05-07 0007 09:54
 * 描述: Emp
 */
public class Emp {

    private Integer empno;
    private String ename;
    private String job;
    private Integer mgr;
    private Date hiredate;
    private Double sal;
    private Double comm;
    private Integer deptno;

    public Emp() {
    }

    public Emp(Integer empno, String ename, String job, Integer mgr, Date
hiredate, Double sal, Double comm, Integer deptno) {
        this.empno = empno;
        this.ename = ename;
        this.job = job;
        this.mgr = mgr;
        this.hiredate = hiredate;
        this.sal = sal;
        this.comm = comm;
        this.deptno = deptno;
    }
}

```

```
}

public Integer getEmpno() {
    return empno;
}

public void setEmpno(Integer empno) {
    this.empno = empno;
}

public String getEname() {
    return ename;
}

public void setEname(String ename) {
    this.ename = ename;
}

public String getJob() {
    return job;
}

public void setJob(String job) {
    this.job = job;
}

public Integer getMgr() {
    return mgr;
}

public void setMgr(Integer mgr) {
    this.mgr = mgr;
}

public Date getHiredate() {
    return hiredate;
}

public void setHiredate(Date hiredate) {
    this.hiredate = hiredate;
}

public Double getsal() {
    return sal;
}

public void setsal(Double sal) {
    this.sal = sal;
}

public Double getComm() {
    return comm;
}

public void setComm(Double comm) {
    this.comm = comm;
}
```

```

    public Integer getDeptno() {
        return deptno;
    }

    public void setDeptno(Integer deptno) {
        this.deptno = deptno;
    }

    @Override
    public String toString() {
        return "Emp{" +
            "empno=" + empno +
            ", ename='" + ename + '\'' +
            ", job='" + job + '\'' +
            ", mgr=" + mgr +
            ", hiredate=" + hiredate +
            ", sal=" + sal +
            ", comm=" + comm +
            ", deptno=" + deptno +
            '}';
    }
}

```

修改Test的代码

```

List<Emp> emps = new ArrayList<>();
while (rs.next()) {
    Integer empno = rs.getInt("EMPNO");
    String ename = rs.getString("ENAME");
    String job = rs.getString("JOB");
    Integer mgr = rs.getInt("mgr");
    Date hiredate = rs.getDate("HIREDATE");
    Double sal = rs.getDouble("SAL");
    Double comm = rs.getDouble("COMM");
    Integer deptno = rs.getInt("DEPTNO");
    System.out.println(empno + "\t" + ename + "\t" + sal + "\t" + hiredate);
    Emp emp = new Emp(empno, ename, job, mgr, hiredate, sal, comm, deptno);
    emps.add(emp);
}

```

```

// 遍历list
emps.forEach(System.out::println);

```

封装表的操作类

```

public class EmpDao {

    private DBUtil dbUtil;

    public EmpDao(DBUtil dbutil) {
        this.dbutil = dbutil;
    }

    /**
     * 查询所有
     *
     * @return
     */
}

```

```

*/
public List<Emp> selectAll() {
    List<Emp> emps = new ArrayList<>();
    String sql = "select * from emp";
    try {
        Connection conn = dbUtil.getConnection();
        /**
         * 1. 发送语句
         * 2. 发送执行指令
         * 3. 接收返回结果集
         */
        Statement st = conn.createStatement(); // 这个实例可用发送语句
        ResultSet rs = st.executeQuery(sql); // 发送 并 查询 返回结果集
        // 遍历结果集
        while (rs.next()) {
            Emp emp = createEmp(rs);
            emps.add(emp);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            dbUtil.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return emps;
}

/**
 * 条件查询
 *
 * @param dno
 * @return
 */
public List<Emp> selectAllByDeptno(int dno) {
    List<Emp> emps = new ArrayList<>();
    String sql = "select * from emp where deptno = " + dno;
    try {
        Connection conn = dbUtil.getConnection();
        /**
         * 1. 发送语句
         * 2. 发送执行指令
         * 3. 接收返回结果集
         */
        Statement st = conn.createStatement(); // 这个实例可用发送语句
        ResultSet rs = st.executeQuery(sql); // 发送 并 查询 返回结果集

        // 遍历结果集
        while (rs.next()) {
            Emp emp = createEmp(rs);
            emps.add(emp);
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

    } finally {
        try {
            dbUtil.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return emps;
}

/**
 * 推荐使用*****
 * PreparedStatement: Statement的扩展可以执行预编译的查询,将查询分成2步
 * 1. 发送sql语句    sql = select * from emp where deptno = ? and sal > ?
 * ? (1,2,3,4....)
 * 2. 设置查询条件    根据 ? 的位置进行设置
 * 好处: 安全
 */
public List<Emp> selectAllByDeptnoSal(Emp emp) {
    String sql = "select * from emp where deptno = ? and sal > ?";
    List<Emp> emps = new ArrayList<>();

    try {
        Connection conn = dbUtil.getConnection();
        PreparedStatement pst = conn.prepareStatement(sql);
        // 设置参数
        pst.setInt(1, emp.getDeptno());
        pst.setDouble(2, emp.getSal());

        ResultSet rs = pst.executeQuery();

        while (rs.next()) {
            Emp e = createEmp(rs);
            emps.add(e);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            dbUtil.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    return emps;
}

public List<Emp> selectLikeName(String name) {

    String sql = "select * from emp where ename like ?";
    List<Emp> emps = new ArrayList<>();

    try {
        Connection conn = dbUtil.getConnection();
        PreparedStatement pst = conn.prepareStatement(sql);
        pst.setString(1, name);
    }

```

```

        ResultSet rs = pst.executeQuery();

        while (rs.next()) {
            Emp emp = createEmp(rs);
            emps.add(emp);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            dbUtil.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    return emps;
}

/**
 * 主键查询
 *
 * @param eno
 * @return
 */
public Emp selectById(int eno) {
    Emp emp = null;
    String sql = "select * from emp where empno = ?";

    try {
        Connection conn = dbUtil.getConnection();
        PreparedStatement pst = conn.prepareStatement(sql);
        pst.setInt(1, eno);
        ResultSet rs = pst.executeQuery();
        while (rs.next()) {
            emp = createEmp(rs);
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            dbUtil.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    return emp;
}

/**
 * @param current : 当前页
 * @param size    : 当前页的条数
 * @return

```

```

    */
    public List<Emp> selectPage(int current, int size) {

        List<Emp> emps = new ArrayList<>();

        // ? 1 : 开始条 startIndex ? 2: 结束条 endIndex
        String sql = "select * from " +
            "(select a.*,rownum rn from " +
            "(select * from emp order by empno) a where rownum <= ? ) b where
b.rn >?";

        // 由 (int current, int size) 计算得到 (startIndex, endIndex)
        int startIndex = (current - 1) * size;
        int endIndex = current * size;

        try {
            Connection conn = dbutil.getConnection();
            PreparedStatement pst = conn.prepareStatement(sql);

            pst.setInt(1, endIndex);
            pst.setInt(2, startIndex);
            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                Emp emp = createEmp(rs);
                emps.add(emp);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                dbutil.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        return emps;
    }

    private Emp createEmp(ResultSet rs) throws SQLException {
        Integer empno = rs.getInt("EMPNO");
        String ename = rs.getString("ENAME");
        String job = rs.getString("JOB");
        Integer mgr = rs.getInt("mgr");
        Date hiredate = rs.getDate("HIREDATE");
        Double sal = rs.getDouble("SAL");
        Double comm = rs.getDouble("COMM");
        Integer deptno = rs.getInt("DEPTNO");
        return new Emp(empno, ename, job, mgr, hiredate, sal, comm, deptno);
    }
}

```

练习：实现分页条件查询

```
/**
条件是： 指定部门=? 工资=?
*/
public List<Emp> selectPageBySelective(Emp emp, int current, int size){
    // TO DO
}
```