/\* TestDeptOrEmp.java \*/

/\*\*

\* 关联查询、分组查询、子查询

\* @author admin

\*

\*/

public class TestDeptOrEmp {

/\*\*

\* 关联查询

\*/

Session session = null;

Transaction tx = null;

/\*\*

\* 查询员工信息，以及部门名称

\*/

@Test

public void test01() {

String hql = "select e.empNo, e.empName, e.dept.deptName from Emp e";

// object[] 构造方法 Map

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Object[]> list = q.list();

for (Object[] o : list) {

System.out.println(o[0]+"---"+o[1]+"---"+o[2]);

}

}

/\*\*

\* inner join

\*/

@Test

public void test02() {

String hql = "select e.empNo, e.empName, d.deptName"

+ " from Emp e inner join e.dept d";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Object[]> list = q.list();

for (Object[] o : list) {

System.out.println(o[0]+"---"+o[1]+"---"+o[2]);

}

}

/\*\*

\* left join

\*/

@Test

public void test03() {

String hql = "select e.empNo, e.empName, d.deptName"

+ " from Emp e left join e.dept d";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Object[]> list = q.list();

for (Object[] o : list) {

System.out.println(o[0]+"---"+o[1]+"---"+o[2]);

}

}

/\*\*

\* 先查询部门名称

\* 再查询部门中的员工信息

\*/

@Test

public void test04() {

String hql = "from Dept";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Dept> list = q.list();

for (Dept d : list) {

System.out.println(d.getDeptName());

Set<Emp> sets = d.getEmps();

Iterator<Emp> it = sets.iterator();

while(it.hasNext()) {

Emp e = it.next();

System.out.println(e.getEmpNo()+"---"+e.getEmpName());

}

}

}

/\*\*

\* 查询人员个数

\*/

@Test

public void test05() {

String hql = "select count(e.empNo) from Emp e";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

long count = (long) q.uniqueResult();

System.out.println("人员个数："+count);

}

/\*\*

\* 查看每个部门的部门名称，员工人数

\*/

@Test

public void test06() {

String hql = "from Dept";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Dept> list = q.list();

for (Dept d : list) {

System.out.println("部门名称："+d.getDeptName()+"---员工人数："+d.getEmps().size());

}

}

/\*\*

\* 聚合/分组查询

\*/

/\*\*

\* 每个部门的部门名称和部门人数

\*/

@Test

public void test07() {

String hql = "select e.dept.deptName, count(e.dept)"

+ " from Emp e group by e.dept";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Object[]> list = q.list();

for (Object[] o : list) {

System.out.println(o[0]+"\t"+o[1]);

}

}

/\*\*

\* 查询部门人数大于1人的部门名称和部门人数

\*/

@Test

public void test08() {

String hql = "select e.dept.deptName, count(e.dept) "

+ "from Emp e group by e.dept "

+ " having count(e.dept)>1";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Object[]> list = q.list();

for (Object[] o : list) {

System.out.println(o[0]+"\t"+o[1]);

}

}

/\*\*

\* 子查询，where子句中关键字：

\* all：返回的所有记录

\* any：返回的任意一条记录

\* some：和"any"意思相同

\* in：和"=any"意思相同

\* exists：至少返回一条记录

\*/

/\*\*

\* 查询所有员工id小于4的部门

\*/

@Test

public void test09() {

String hql = "from Dept d "

+ "where 4> all(select e.empNo from d.emps e)";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Dept> list = q.list();

for (Dept d : list) {

System.out.println(d.getDeptId()+"---"+d.getDeptName());

}

}

/\*\*

\* 查询至少有一位员工的id低于4的部门

\*/

@Test

public void test10() {

String hql = "from Dept d "

+ "where 4>any(select e.empNo from d.emps e)";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Dept> list = q.list();

for (Dept d : list) {

System.out.println(d.getDeptId()+"---"+d.getDeptName());

}

}

/\*\*

\* 查询有员工id正好是4的部门

\*/

@Test

public void test11() {

String hql = "from Dept d "

+ "where 4 = any(select e.empNo from d.emps e)";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Dept> list = q.list();

for (Dept d : list) {

System.out.println(d.getDeptId()+"---"+d.getDeptName());

}

}

/\*\*

\* 查询员工id小于等于4的部门信息

\*/

@Test

public void test12() {

String hql = "from Dept d where d.deptId in "

+ "(select e.dept.deptId from d.emps e where e.empNo<=4) ";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Dept> list = q.list();

for (Dept d : list) {

System.out.println(d.getDeptId()+"---"+d.getDeptName());

}

}

/\*\*

\* 查询至少有一位员工的部门信息

\*/

@Test

public void test13() {

String hql = "from Dept d where exists (from d.emps e)";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Dept> list = q.list();

for (Dept d : list) {

System.out.println(d.getDeptId()+"---"+d.getDeptName());

}

}

}

/\* TestCache.java \*/

public class TestCache {

Session session = null;

Transaction tx = null;

/\*\*

\* 测试一级缓存

\*/

@Test

public void test1() {

// 同一个session下来，执行相同的查询操作，第一次查询数据库，

// 放入一级缓存中（session 缓存）,执行二次相同操作的时候，就会从

// 缓存中，直接获取数据

session = HB.getCurrentSession();

tx = session.beginTransaction();

Dept d = (Dept) session.load(Dept.class, 2);

System.out.println(d.getDeptName());

Dept d1 = (Dept) session.get(Dept.class, 2);

System.out.println(d1.getDeptName());

tx.commit();

}

/\*\*

\* 会放入缓存，不会从缓存中取数据

\*/

@Test

public void test2() {

String hql = "from Dept ";

session = HB.getCurrentSession();

tx = session.beginTransaction();

Query q = session.createQuery(hql);

List<Dept> ds = q.list();

for (Dept d : ds) {

System.out.println(d.getDeptId()+"---"+d.getDeptName());

}

List<Dept> ds1 = q.list();

for (Dept d : ds1) {

System.out.println(d.getDeptId()+"---"+d.getDeptName());

}

tx.commit();

}

/\*\*

\* 懒加载 get load

\*/

@Test

public void test3() {

session = HB.getCurrentSession();

tx = session.beginTransaction();

//session.clear();

Dept d = (Dept) session.load(Dept.class, 2);

System.out.println(d.getDeptName());

tx.commit();

}

}