resultMap

resultMap is a mapping element defined in outside of SQL. The usage of the MappingCommAreaOperation will be shown: accessing a CICS with ECI mode with the IBM CICS ECI connector. resultMap is the important mapping element most frequently used and allows column type instruction, replacement value of null value, typeHandler processing, complex property mapping(complex object including other JavaBean, Collections) in comparison with auto mapping access method using resultClass property.

Using basic resultMap

Refer to below sample resultMap.

Sample resultMap

```
..
<typeAlias alias="empVO" type="egovframework.rte.psl.dataaccess.vo.EmpVO" />

<resultMap id="empResult" class="empVO" >
        <result property="empNo" column="EMP_NO" columnIndex="1"
                javaType="decimal" jdbcType="NUMERIC" />
        <result property="empName" column="EMP_NAME" columnIndex="2"
                javaType="string" jdbcType="VARCHAR" />
        <result property="job" column="JOB" columnIndex="3" javaType="string"
                jdbcType="VARCHAR" />
        <result property="mgr" column="MGR" columnIndex="4" javaType="decimal"
                jdbcType="NUMERIC" />
        <result property="hireDate" column="HIRE_DATE" columnIndex="5"
                javaType="date" jdbcType="DATE" />
        <result property="sal" column="SAL" columnIndex="6" javaType="decimal"
                jdbcType="NUMERIC" />
        <result property="comm" column="COMM" columnIndex="7" javaType="decimal"
                jdbcType="NUMERIC" nullValue="0" />
        <result property="deptNo" column="DEPT_NO" columnIndex="8"
                javaType="decimal" jdbcType="NUMERIC" />
</resultMap>

<select id="selectEmpUsingResultMap" parameterClass="empVO" resultMap="empResult">
        <![CDATA[
                select EMP_NO,
                        EMP_NAME,
                        JOB,
                        MGR,
                        HIRE_DATE,
                        SAL,
                        COMM,
                        DEPT_NO
                from    EMP
                where   EMP_NO = #empNo#
        ]]>
</select>
```

In the above sql mapping file, id called empResult is given as resultMap element and the target result object designates EmpVO. Mapping definition is performed as result lower element for detailed attribute(property) for EmpVO. Map the select target column(name of relevant alias if column alias is used) that can be obtained in result set as column property. In above example, it was described for columnIndex, javaType, jdbcType additionally. If type is indicated clearly as shown above, there is more advantage in terms of performance than getting the type for individual property of target class using reflection technology of java. If designating columnIndex, there is small advantage in terms of performance processed as rs.getString("EMP_NAME") → rs.getString(2), but is not recommended due to inconvenience of setting itself or designation of order. In addition, for the property designating nullValue property, it is replaced with the value designated in nullValue, when relevant value is read as

null in database and set in JavaBeans property. Besides, it can refer to external resultMap mapping element for the internal object for processing of complex property or results of other query sentence through select, resultMap as a lower element process of result. In addition, custom typeHandler implement can be indicated, rather than default processing of iBATIS through typeHandler property.

- resultMap structure

```
<resultMap id="resultMapName" class="some.domain.Class"
        [extends="parent-resultMap"] [groupBy="some property list"]>
        <result property="propertyName" column="COLUMN_NAME"
                [columnIndex="1"] [javaType="int"] [jdbcType="NUMERIC"] [nullValue="-
999999"]
                [select="someOtherStatement"] [resultMap="someOtherResultMap"]
                [typeHandler="com.mydomain.MyTypehandler"] />
        <result … />
        <result … />
        <result … />
</resultMap>
```

In above, indicating the extends property of resultMap tag can inherit other resultMap defined externally(without defining related property - column mapping in current resultMap). Create the rows with same value of property list indicated through relevant property to solve N+1 query in nested resultMap using groupBy property, as one result object.

Sample TestCase

```
..
    @Test
    public void testResultMapSelect() throws Exception {
        EmpVO vo = new EmpVO();
        // 7369,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20
        vo.setEmpNo(new BigDecimal(7369));

        // select
        EmpVO resultVO = empDAO.selectEmp("selectEmpUsingResultMap", vo);

        // check
        assertNotNull(resultVO);
        assertEquals(new BigDecimal(7369), resultVO.getEmpNo());
        assertEquals("SMITH", resultVO.getEmpName());
        assertEquals("CLERK", resultVO.getJob());
        assertEquals(new BigDecimal(7902), resultVO.getMgr());
        SimpleDateFormat sdf =
            new SimpleDateFormat("yyyy-MM-dd", java.util.Locale.getDefault());
        assertEquals(sdf.parse("1980-12-17"), resultVO.getHireDate());
        assertEquals(new BigDecimal(800), resultVO.getSal());

        // nullValue test - <result property="comm" column="COMM" .. nullValue="0" />
        assertEquals(new BigDecimal(0), resultVO.getComm());
        assertEquals(new BigDecimal(20), resultVO.getDeptNo());
    }
```

It is the test case that inquires and processes the input object of setting the inquiry conditions(pk) for the selectEmpUsingResultMap query sentence that processes result object mapping using resultMap in above. In case of using resultMap in mapping file actually and the case of using resultClass as it is, application area can obtain the result object of same type, but is recommended since use and definition of resultMap in sql mapping file is advantageous. Through above test validation, it was confirmed that the inquiry result is well reflected as result object of EmpVO and for the comm attribute designated as nullValue property, database is null but replaced with numeric value corresponding to 0.

Inheriting resultMap

Refer to the definition of sample resultMap.

Sample VO

```
..
public class EmpExtendsDeptVO extends DeptVO {

    private static final long serialVersionUID = -4653117983538108612L;

    private BigDecimal empNo;

    private String empName;

    private String job;

    private BigDecimal mgr;

    private Date hireDate;

    private BigDecimal sal;

    private BigDecimal comm;

    public BigDecimal getEmpNo() {
        return empNo;
    }

    public void setEmpNo(BigDecimal empNo) {
        this.empNo = empNo;
    }
..
```

Sample resultMap (extends)

```
        ..
        <typeAlias alias="empExtendsDeptVO"
type="egovframework.rte.psl.dataaccess.vo.EmpExtendsDeptVO" />

        <!--
                cf.) Inheritance relation of VO and that of resultMap do not need to be same.
Following empExtendsDeptResult extends
                empResult (above resultMap with Emp property), but the child has Emp property
while EmpExtendsDeptVO extends DeptVO actually.
        -->
        <resultMap id="empExtendsDeptResult" class="empExtendsDeptVO" extends="empResult">
                <!--<result property="deptNo" column="DEPT_NO"/>-->
                <result property="deptName" column="DEPT_NAME"/>
                <result property="loc" column="LOC"/>
        </resultMap>

        <select id="selectEmpExtendsDeptUsingResultMap" parameterClass="empVO"
resultMap="empExtendsDeptResult">
                <![CDATA[
                        select EMP_NO,
                                EMP_NAME,
                                JOB,
                                MGR,
                                HIRE_DATE,
                                SAL,
                                COMM,
                                A.DEPT_NO,
                                B.DEPT_NAME,
```

```
                B.LOC
        from    EMP A, DEPT B
        where   A.DEPT_NO = B.DEPT_NO
        and     EMP_NO = #empNo#
    ]]>
</select>
```

empExtendsDeptResult resultMap extends empResult resultMap defined in how to use resultMap by default, and adds property mapping for additional deptName, loc, and can be confirmed that it can be processed automatically depending on mapping definition. Inheritance of mapping definition for resultMap is performed regardless of inheritance of result object JavaBeans. (in above, VO extends code and resultMap extends are opposite)

Sample TestCase

```
..
    @Test
    public void testExtendsResultMapSelect() throws Exception {
        EmpVO vo = new EmpVO();
        // 7369,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20
        vo.setEmpNo(new BigDecimal(7369));

        // select
        EmpExtendsDeptVO resultVO =
            empDAO.selectEmpExtendsDept("selectEmpExtendsDeptUsingResultMap",
                vo);

        // check
        assertNotNull(resultVO);
        // resultMap extends test (extends empResult)
        assertEquals(new BigDecimal(7369), resultVO.getEmpNo());
        assertEquals("SMITH", resultVO.getEmpName());
        assertEquals("CLERK", resultVO.getJob());
        assertEquals(new BigDecimal(7902), resultVO.getMgr());
        SimpleDateFormat sdf =
            new SimpleDateFormat("yyyy-MM-dd", java.util.Locale.getDefault());
        assertEquals(sdf.parse("1980-12-17"), resultVO.getHireDate());
        assertEquals(new BigDecimal(800), resultVO.getSal());
        // nullValue test - <result property="comm" column="COMM" .. nullValue="0" />
        assertEquals(new BigDecimal(0), resultVO.getComm());
        assertEquals(new BigDecimal(20), resultVO.getDeptNo());

        assertEquals("RESEARCH", resultVO.getDeptName());
        assertEquals("DALLAS", resultVO.getLoc());

    }
```

The test case for inquiring and processing for selectEmpExtendsDeptUsingResultMap processing the result object mapping that uses empExtendsDeptResult resultMap inheriting upper resultMap through extends property. Through above test validation, the inquiry result is well reflected as a result value of EmpExtendsDeptVO through above test validation. nullValue replacement processing that exists in parent resultMap is well reflected.

Simple Composite resultMap

Refer to sample resultMap definition.

Sample VO

```
..
public class EmpDeptSimpleCompositeVO implements Serializable {
```

```
    private static final long serialVersionUID = -8049578957221741495L;

    private BigDecimal empNo;

    private String empName;

    private String job;

    private BigDecimal mgr;

    private Date hireDate;

    private BigDecimal sal;

    private BigDecimal comm;

    private BigDecimal deptNo;

    private String deptName;

    private String loc;

    public BigDecimal getEmpNo() {
        return empNo;
    }

    public void setEmpNo(BigDecimal empNo) {
        this.empNo = empNo;
    }
..
```

The abstract EciMappingOperation class can then be subclassed to specify mappings between custom objects and Records.

Sample resultMap

```
        ..
        <typeAlias alias="empDeptSimpleCompositeVO"
type="egovframework.rte.psl.dataaccess.vo.EmpDeptSimpleCompositeVO" />

        <resultMap id="empDeptSimpleComposite" class="empDeptSimpleCompositeVO" >
                <result property="empNo" column="EMP_NO"/>
                <result property="empName" column="EMP_NAME"/>
                <result property="job" column="JOB"/>
                <result property="mgr" column="MGR"/>
                <result property="hireDate" column="HIRE_DATE"/>
                <result property="sal" column="SAL"/>
                <result property="comm" column="COMM" nullValue="0"/>
                <result property="deptNo" column="DEPT_NO"/>
                <result property="deptName" column="DEPT_NAME"/>
                <result property="loc" column="LOC"/>
        </resultMap>

        <select id="selectEmpDeptSimpleCompositeUsingResultMap" parameterClass="empVO"
resultMap="empDeptSimpleComposite">
                <![CDATA[
                        select EMP_NO,
                                EMP_NAME,
                                JOB,
                                MGR,
```

```
                    HIRE_DATE,
                    SAL,
                    COMM,
                    A.DEPT_NO,
                    B.DEPT_NAME,
                    B.LOC
            from    EMP A, DEPT B
            where   A.DEPT_NO = B.DEPT_NO
            and     EMP_NO = #empNo#
        ]]>
    </select>
```

Query sentence inquiring the result row containing DEPT information through join query for EMP and DEPT is the same as above and in comparison with empExtendsDeptResult resultMap that uses extends, resultMap definition contains all elements. Since the property of result object is same as inquiry field, it can be changed simply by matching resultClass only as resultMap that uses extends.

Sample TestCase

```
..
    @Test
    public void testSimpleCompositeResultMapSelect() throws Exception {
        EmpVO vo = new EmpVO();
        // 7369,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20
        vo.setEmpNo(new BigDecimal(7369));

        // select
        EmpDeptSimpleCompositeVO resultVO =
            empDAO.selectEmpDeptSimpleComposite(
                "selectEmpDeptSimpleCompositeUsingResultMap", vo);

        // check
        assertNotNull(resultVO);
        assertEquals(new BigDecimal(7369), resultVO.getEmpNo());
        assertEquals("SMITH", resultVO.getEmpName());
        assertEquals("CLERK", resultVO.getJob());
        assertEquals(new BigDecimal(7902), resultVO.getMgr());
        SimpleDateFormat sdf =
            new SimpleDateFormat("yyyy-MM-dd", java.util.Locale.getDefault());
        assertEquals(sdf.parse("1980-12-17"), resultVO.getHireDate());
        assertEquals(new BigDecimal(800), resultVO.getSal());
        // nullValue test - <result property="comm" column="COMM" .. nullValue="0" />
        assertEquals(new BigDecimal(0), resultVO.getComm());
        assertEquals(new BigDecimal(20), resultVO.getDeptNo());
        assertEquals("RESEARCH", resultVO.getDeptName());
        assertEquals("DALLAS", resultVO.getLoc());

    }
```

Test case of inquiring and processing selectEmpDeptSimpleCompositeUsingResultMap that processes result object mappoing using resultMap defining all mapping elements simply without extends. It is confirmed that inquiry results are well reflected in result object of EmpDeptSimpleCompositeV through above test validation.

If result object obtained through join inquiry through implementation of above comparison is simple composite VO type, use resultMap extends for easy mapping.

Using Complex Properties resultMap

In managed mode (that is, in a J2EE environment), the configuration could look as follows.

Complext Properties - (1:1 relationship) resultMap

```
..
public class EmpIncludesDeptVO implements Serializable {

    private static final long serialVersionUID = -4113989804152701350L;

    private BigDecimal empNo;

    private String empName;

    private String job;

    private BigDecimal mgr;

    private Date hireDate;

    private BigDecimal sal;

    private BigDecimal comm;

    private BigDecimal deptNo;

    // EMP - DEPT 1:1 relation
    private DeptVO deptVO;

    public BigDecimal getEmpNo() {
        return empNo;
    }

    public void setEmpNo(BigDecimal empNo) {
        this.empNo = empNo;
    }
..
    public DeptVO getDeptVO() {
        return deptVO;
    }

    public void setDeptVO(DeptVO deptVO) {
        this.deptVO = deptVO;
    }
}
```

Above EmpIncludesDeptVO contains DeptVO as 1:1 relationship member attribute.

```
<sqlMap namespace="EmpComplexResult">
          ..
          <typeAlias alias="empIncludesDeptVO"
type="egovframework.rte.psl.dataaccess.vo.EmpIncludesDeptVO" />

          <resultMap id="empIncludesDeptResult" class="empIncludesDeptVO">
                    <result property="empNo" column="EMP_NO" />
                    <result property="empName" column="EMP_NAME" />
                    <result property="job" column="JOB" />
                    <result property="mgr" column="MGR" />
                    <result property="hireDate" column="HIRE_DATE" />
                    <result property="sal" column="SAL" />
                    <result property="comm" column="COMM" nullValue="0" />
                    <result property="deptNo" column="DEPT_NO" />

                    <!--
                              Emp-Dept 1:1 relation
                              As a result of test, when referring resultMap, regardless of
                              useStatementNamespaces="false" of sql-map-config.xml, namespace prefix
should be used
```

```
                -->
                <result property="deptVO" resultMap="EmpComplexResult.getDeptResult" />
        </resultMap>

        <resultMap id="getDeptResult" class="deptVO">
                <result property="deptNo" column="DEPT_NO" />
                <result property="deptName" column="DEPT_NAME" />
                <result property="loc" column="LOC" />
        </resultMap>

        <select id="selectEmpIncludesDeptResultUsingResultMap" parameterClass="empVO"
resultMap="empIncludesDeptResult">
                <![CDATA[
                        select EMP_NO,
                                EMP_NAME,
                                JOB,
                                MGR,
                                HIRE_DATE,
                                SAL,
                                COMM,
                                A.DEPT_NO as DEPT_NO,
                                B.DEPT_NAME,
                                B.LOC
                        from    EMP A, DEPT B
                        where   A.DEPT_NO = B.DEPT_NO
                        and     EMP_NO = #empNo#
                ]]>
        </select>
```

At the time of resultMap mapping processing for EmpIncludesDeptVO containing Complex Properties that expresses 1:1 relationship in above sql mapping file, external resultMap for DeptVO is reused and referred as shown in resultMap="EmpComplexResult.getDeptResult" for mapping definition for deptVO member attribute of relevant objects. iBATIS maps the result column by join queries automatically to complex object(in particular, in DeptVO related member object) in reference to mapping definition for nested resultMap.

Sample TestCase

```
..
    @Test
    public void testComplexPropertiesOneToOneResultMapSelect() throws Exception {
        EmpVO vo = new EmpVO();
        // 7369,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20
        vo.setEmpNo(new BigDecimal(7369));

        // select
        EmpIncludesDeptVO resultVO =
            empDAO.selectEmpDeptComplexProperties(
                "selectEmpIncludesDeptResultUsingResultMap", vo);

        // check
        assertNotNull(resultVO);
        assertEquals(new BigDecimal(7369), resultVO.getEmpNo());
        assertEquals("SMITH", resultVO.getEmpName());
        assertEquals("CLERK", resultVO.getJob());
        assertEquals(new BigDecimal(7902), resultVO.getMgr());
        SimpleDateFormat sdf =
            new SimpleDateFormat("yyyy-MM-dd", java.util.Locale.getDefault());
        assertEquals(sdf.parse("1980-12-17"), resultVO.getHireDate());
        assertEquals(new BigDecimal(800), resultVO.getSal());
        assertEquals(new BigDecimal(0), resultVO.getComm());
        assertEquals(new BigDecimal(20), resultVO.getDeptNo());
```

```
        // 1:1 relation included DeptVO
        assertEquals(new BigDecimal(20), resultVO.getDeptVO().getDeptNo());
        assertEquals("RESEARCH", resultVO.getDeptVO().getDeptName());
        assertEquals("DALLAS", resultVO.getDeptVO().getLoc());
    }
```

In above test case validation code, each attribute of internal object is well set as shown in resultVO.getDeptVO().getXXX.

Complext Properties - (1:N relationship) resultMap

```
..
public class DeptIncludesEmpListVO implements Serializable {

    private static final long serialVersionUID = -3369530755443065377L;

    private BigDecimal deptNo;

    private String deptName;

    private String loc;

    private List<EmpVO> empVOList;

    public BigDecimal getDeptNo() {
        return deptNo;
    }

    public void setDeptNo(BigDecimal deptNo) {
        this.deptNo = deptNo;
    }
..
    public List<EmpVO> getEmpVOList() {
        return empVOList;
    }

    public void setEmpVOList(List<EmpVO> empVOList) {
        this.empVOList = empVOList;
    }
}
```

Above DeptIncludesEmpListVO contains list of EmpVO as member attribute of 1:N relationship.

```
<sqlMap namespace="EmpComplexResult">
        ..
        <typeAlias alias="deptIncludesEmpListVO"
type="egovframework.rte.psl.dataaccess.vo.DeptIncludesEmpListVO" />

        <!-- indicate groupBy property in case of 1:N  -->
        <resultMap id="deptIncludesEmpListResult" class="deptIncludesEmpListVO"
groupBy="deptNo">
                <result property="deptNo" column="DEPT_NO" />
                <result property="deptName" column="DEPT_NAME" />
                <result property="loc" column="LOC" />

                <!-- Dept-EmpList 1:N relation -->
                <result property="empVOList" resultMap="EmpComplexResult.getEmpResult" />
        </resultMap>

        <resultMap id="getEmpResult" class="empVO">
                <result property="empNo" column="EMP_NO" />
                <result property="empName" column="EMP_NAME" />
```

```
                <result property="job" column="JOB" />
                <result property="mgr" column="MGR" />
                <result property="hireDate" column="HIRE_DATE" />
                <result property="sal" column="SAL" />
                <result property="comm" column="COMM" nullValue="0" />
                <result property="deptNo" column="DEPT_NO" />
        </resultMap>

        <select id="selectDeptIncludesEmpListResultUsingResultMap" parameterClass="deptVO"
resultMap="deptIncludesEmpListResult">
                <![CDATA[
                        select    A.DEPT_NO as DEPT_NO,
                                  DEPT_NAME,
                                  LOC,
                                  EMP_NO,
                                  EMP_NAME,
                                  JOB,
                                  MGR,
                                  HIRE_DATE,
                                  SAL,
                                  COMM
                        from       DEPT A,
                                  EMP B
                        where      A.DEPT_NO = B.DEPT_NO
                                  and A.DEPT_NO = #deptNo#
                        order by B.EMP_NO
                ]]>
        </select>
```

When mapping resultMap for DeptIncludesEmpListVO that contains Complex Properties that expresses 1:N relation in above sql mapping file, external resultMap for EmpVO is used and referred as shown in resultMap="EmpComplexResult.getEmpResult" for mapping definition for empVOList member attribute of relevant object. In above queries, the number of row inquired by join is not simple like that of row, 1:1 relationship, but several results are obtained for DEPT_NO. Since groupBy="deptNo" was designated when defining resultMap for this, List for EmpVo, the same deptNo is set in list member attribute of multiple object. iBATIS automatically maps the result value by join query to multiple object in reference to mapping definition for nested resultMap and lower element can be automatically to list type by grouping it as property designated as groupBy.

Sample TestCase

```
..
    @Test
    public void testComplexPropertiesOneToManyResultMapSelect()
            throws Exception {
        DeptVO vo = new DeptVO();
        // 20,'RESEARCH','DALLAS'
        vo.setDeptNo(new BigDecimal(20));

        // select
        DeptIncludesEmpListVO resultVO =
            empDAO.selectDeptEmpListComplexProperties(
                "selectDeptIncludesEmpListResultUsingResultMap", vo);

        // check
        assertNotNull(resultVO);
        assertEquals(new BigDecimal(20), resultVO.getDeptNo());
        assertEquals("RESEARCH", resultVO.getDeptName());
        assertEquals("DALLAS", resultVO.getLoc());

        assertTrue(0 < resultVO.getEmpVOList().size());
```

```
        /*
         * EmpList of deptNo 20 is, according to initial data, 7369,'SMITH','CLERK',7902,'1980-12-
17',800,NULL,20
         * 7566,'JONES','MANAGER',7839,'1981-04-02',2975,NULL,20
7788,'SCOTT','ANALYST',7566,'1987-04-19',3000,NULL,20
         * 7876,'ADAMS','CLERK',7788,'1987-05-23',1100,NULL,20
7902,'FORD','ANALYST',7566,'1981-12-03',3000,NULL,20
         */
        assertEquals(5, resultVO.getEmpVOList().size());

        assertEquals(new BigDecimal(7369), resultVO.getEmpVOList().get(0)
            .getEmpNo());
        assertEquals("SMITH", resultVO.getEmpVOList().get(0).getEmpName());
        assertEquals("CLERK", resultVO.getEmpVOList().get(0).getJob());
        assertEquals(new BigDecimal(7902), resultVO.getEmpVOList().get(0)
            .getMgr());
        SimpleDateFormat sdf =
            new SimpleDateFormat("yyyy-MM-dd", java.util.Locale.getDefault());
        assertEquals(sdf.parse("1980-12-17"), resultVO.getEmpVOList().get(0)
            .getHireDate());
        assertEquals(new BigDecimal(800), resultVO.getEmpVOList().get(0)
            .getSal());
        assertEquals(new BigDecimal(0), resultVO.getEmpVOList().get(0)
            .getComm());
        assertEquals(new BigDecimal(20), resultVO.getEmpVOList().get(0)
            .getDeptNo());

        assertEquals(new BigDecimal(7566), resultVO.getEmpVOList().get(1)
            .getEmpNo());
        assertEquals(new BigDecimal(7788), resultVO.getEmpVOList().get(2)
            .getEmpNo());
        assertEquals(new BigDecimal(7876), resultVO.getEmpVOList().get(3)
            .getEmpNo());
        assertEquals(new BigDecimal(7902), resultVO.getEmpVOList().get(4)
            .getEmpNo());
    }
```

In above test case validation code, each EmpVo of internal object(List<EmpVO>) such as resultVO.getEmpVOList().get(X).getXXX and relevant attributes are set.

Complext Properties - (1:N relationship - outer join case) resultMap

For global transactions, you can use Spring's generic transaction infrastructure to demarcate transactions, with JtaTransactionManager as backend (delegating to the J2EE server's distributed transaction coordinator underneath).

```
<sqlMap namespace="EmpComplexResult">
        ..
        <select id="selectDeptIncludesEmpListResultListUsingResultMap" parameterClass="deptVO"
resultMap="deptIncludesEmpListResult">
                <![CDATA[
                        select    A.DEPT_NO as DEPT_NO,
                                  DEPT_NAME,
                                  LOC,
                                  EMP_NO,
                                  EMP_NAME,
                                  JOB,
                                  MGR,
                                  HIRE_DATE,
                                  SAL,
                                  COMM
                        from       DEPT A
```

```
                                left outer join EMP B
                                  on (A.DEPT_NO = B.DEPT_NO)
                        where     A.DEPT_NAME like '%'||#deptName#||'%'
                        order by A.DEPT_NO,
                                B.EMP_NO
                ]]>
        </select>
```

Sample TestCase

```
..
    @Test
    public void testComplexPropertiesOneToManyVOListResultMapSelect()
            throws Exception {
        DeptVO vo = new DeptVO();
        // like search test by deptName   '%'|| 'E' ||'%' --> R'E'S'E'ARCH, SAL'E'S, OP'E'RATIONS
        // 20,'RESEARCH','DALLAS'
        // 30,'SALES','CHICAGO'
        // 40,'OPERATIONS','BOSTON'
        vo.setDeptName("E");

        // select
        List<DeptIncludesEmpListVO> resultList =
            empDAO.selectDeptEmpListComplexPropertiesList(isMysql
                ? "selectDeptIncludesEmpListResultListUsingResultMapMysql"
                : "selectDeptIncludesEmpListResultListUsingResultMap", vo);

        // check
        assertNotNull(resultList);
        assertEquals(3, resultList.size());

        assertEquals(new BigDecimal(20), resultList.get(0).getDeptNo());
        assertEquals(new BigDecimal(30), resultList.get(1).getDeptNo());
        assertEquals(new BigDecimal(40), resultList.get(2).getDeptNo());

        /*
         * EmpList of deptNo 20 is set to 5 persons according to initial data; EmpList of deptNo 30
EmpList is 6 persons according to initial data, and EmpList of deptNo 40 is 0 according to initial data.
         * --> cf.) there is only 1 case of EmpVo with depNo only according to outer join
         */
        assertEquals(5, resultList.get(0).getEmpVOList().size());
        assertEquals(6, resultList.get(1).getEmpVOList().size());
        // cf.)check occurrence of only 1 case of EmpVO with deptNo only according to
outer join. Careful!
        assertEquals(1, resultList.get(2).getEmpVOList().size());
        assertNull(resultList.get(2).getEmpVOList().get(0).getEmpNo());

        /*
         * According to initial data, EmpList of deptNo 20 is 7369,'SMITH','CLERK',7902,'1980-12-
17',800,NULL,20
         * 7566,'JONES','MANAGER',7839,'1981-04-02',2975,NULL,20
7788,'SCOTT','ANALYST',7566,'1987-04-19',3000,NULL,20
         * 7876,'ADAMS','CLERK',7788,'1987-05-23',1100,NULL,20
7902,'FORD','ANALYST',7566,'1981-12-03',3000,NULL,20
         */
        assertEquals(new BigDecimal(7566), resultList.get(0).getEmpVOList()
            .get(1).getEmpNo());
        assertEquals(new BigDecimal(7788), resultList.get(0).getEmpVOList()
            .get(2).getEmpNo());
        assertEquals(new BigDecimal(7876), resultList.get(0).getEmpVOList()
            .get(3).getEmpNo());
        assertEquals(new BigDecimal(7902), resultList.get(0).getEmpVOList()
            .get(4).getEmpNo());
```

```
    }
```

If mapping the result of outer join using resultMap of iBATIS in the above test case validation code, (type of using groupBy property), since the lower object with values corresponding to join key may occur unintentionally. Be careful in using!

Complext Properties - (1:N relationship - N+1 select form) resultMap

For global transactions, you can use Spring's generic transaction infrastructure to demarcate transactions, with JtaTransactionManager as backend (delegating to the J2EE server's distributed transaction coordinator underneath).

```
<sqlMap namespace="EmpComplexResult">
        ..
        <!-- 1:N case N+1 select form – not recommended -->
        <resultMap id="deptIncludesEmpListUsingSelectAttrResult" class="deptIncludesEmpListVO">
                <result property="deptNo" column="DEPT_NO" />
                <result property="deptName" column="DEPT_NAME" />
                <result property="loc" column="LOC" />

                <!-- Dept-EmpList 1:N relation using select attribute -->
                <result property="empVOList" column="DEPT_NO" select="selectEmpList" />
        </resultMap>

        <select id="selectDeptIncludesEmpListResultListUsingRepetitionSelect"
parameterClass="deptVO"
                        resultMap="deptIncludesEmpListUsingSelectAttrResult">
                <![CDATA[
                        select    DEPT_NO,
                                  DEPT_NAME,
                                  LOC
                        from       DEPT
                        where      DEPT_NAME like '%'||#deptName#||'%'
                        order by DEPT_NO
                ]]>
        </select>

        <select id="selectEmpList" parameterClass="decimal" resultMap="getEmpResult">
                <![CDATA[
                        select    EMP_NO,
                                  EMP_NAME,
                                  JOB,
                                  MGR,
                                  HIRE_DATE,
                                  SAL,
                                  COMM,
                                  DEPT_NO
                        from       EMP
                        where      DEPT_NO = #deptNo#
                        order by EMP_NO
                ]]>
        </select>
```

Example of calling separate query sentence(processed as resultMap for EmpVO) as shown in select="selectEmpList" for mapping definition for empVOList member attribute of relevant object when processing the resultMap mapping for DeptIncludesEmpListVO containing Complex Properties expressing 1:N relationship in above sql mapping file. In query sentence, join is not used and calling of main query sentence once (no. 1) may result in execution of separate query (N) designated as select property as many as rows, so that it is not undesirable in terms of performance.

Sample TestCase

..

```
    @Test
    public void testComplexPropertiesOneToManyVOListRepetitionSelect()
            throws Exception {
        DeptVO vo = new DeptVO();
        // like search test by deptName '%'|| 'E' ||'%' --> R'E'S'E'ARCH, SAL'E'S, OP'E'RATIONS
        // 20,'RESEARCH','DALLAS'
        // 30,'SALES','CHICAGO'
        // 40,'OPERATIONS','BOSTON'
        vo.setDeptName("E");

        // select
        List<DeptIncludesEmpListVO> resultList =
            empDAO
                .selectDeptEmpListComplexPropertiesList(
                    isMysql
                        ? "selectDeptIncludesEmpListResultListUsingRepetitionSelectMysql"
                        : "selectDeptIncludesEmpListResultListUsingRepetitionSelect",
                    vo);

        // check
        assertNotNull(resultList);
        assertEquals(3, resultList.size());

        assertEquals(new BigDecimal(20), resultList.get(0).getDeptNo());
        assertEquals(new BigDecimal(30), resultList.get(1).getDeptNo());
        assertEquals(new BigDecimal(40), resultList.get(2).getDeptNo());

        /*
         * According to initial data, deptNo 20 of EmpList is set to 5 persons, deptNo 30 of EmpList is 6
persons, deptNo 40 of EmpList is 0.
         * --> Different from above outer join case, there is no number of EmpList
         */
        assertEquals(5, resultList.get(0).getEmpVOList().size());
        assertEquals(6, resultList.get(1).getEmpVOList().size());
        assertEquals(0, resultList.get(2).getEmpVOList().size());

        /*
         * According to initial data, deptNo 20 of EmpList is 7369,'SMITH','CLERK',7902,'1980-12-
17',800,NULL,20
         * 7566,'JONES','MANAGER',7839,'1981-04-02',2975,NULL,20
7788,'SCOTT','ANALYST',7566,'1987-04-19',3000,NULL,20
         * 7876,'ADAMS','CLERK',7788,'1987-05-23',1100,NULL,20
7902,'FORD','ANALYST',7566,'1981-12-03',3000,NULL,20
         */
        assertEquals(new BigDecimal(7566), resultList.get(0).getEmpVOList()
            .get(1).getEmpNo());
        assertEquals(new BigDecimal(7788), resultList.get(0).getEmpVOList()
            .get(2).getEmpNo());
        assertEquals(new BigDecimal(7876), resultList.get(0).getEmpVOList()
            .get(3).getEmpNo());
        assertEquals(new BigDecimal(7902), resultList.get(0).getEmpVOList()
            .get(4).getEmpNo());
    }
```

From the above test case validation code, like resultList.get(X).getEmpVOList().get(X).getXXX, it can be confirmed that relevant attribute and each EmpVo of internal object (List<EmpVO>) of inquiry result (List) is set well. However, it may cause performance problem of N+1 inquiry, it is desirable to process with 1:N relationship processing through join query and groupBy property when defining sql mapping.

Complext Properties - (Hierarchy relations) resultMap

```
..
public class EmpIncludesMgrVO implements Serializable {

    private static final long serialVersionUID = 5695339933191681519L;

    private BigDecimal empNo;

    private String empName;

    private String job;

    private BigDecimal mgr;

    private Date hireDate;

    private BigDecimal sal;

    private BigDecimal comm;

    private BigDecimal deptNo;

    // Hierarchy relations
    private EmpIncludesMgrVO mgrVO;

    public BigDecimal getEmpNo() {
        return empNo;
    }

    public void setEmpNo(BigDecimal empNo) {
        this.empNo = empNo;
    }
..
    public EmpIncludesMgrVO getMgrVO() {
        return mgrVO;
    }

    public void setMgrVO(EmpIncludesMgrVO mgrVO) {
        this.mgrVO = mgrVO;
    }
}
```

Above EmpIncludesMgrVO includes EmpIncludesMgrVO same as itself, as the member attribute of Hierarchy relations. In above, it is contained for manager of Emp. When it goes up MgrVO, it is the object that contains information for all employees in the administrator path.

```
<sqlMap namespace="EmpComplexResult">
        ..
        <typeAlias alias="empIncludesMgrVO"
type="egovframework.rte.psl.dataaccess.vo.EmpIncludesMgrVO" />

        <!—Example of process by ibatis resultMap select in case of Hierarchical relation -->
        <resultMap id="empIncludesMgrResult" class="empIncludesMgrVO"
extends="getEmpResult">
                <result property="mgrVO" column="MGR" select="selectMgrHierarchy" />
        </resultMap>

        <!—change to enable to use with repeated inquiry of Hierarchy according to existence of
empNo, mgr property using one query
                 Since parameterClass differs with empVO and decimal for initial inquiry and select
inquiry of resultMap, it is not separately indicated but processed by automatic reflection
         -->
        <select id="selectMgrHierarchy" resultMap="empIncludesMgrResult">
                <![CDATA[
```

```
                    select    EMP_NO,
                              EMP_NAME,
                              JOB,
                              MGR,
                              HIRE_DATE,
                              SAL,
                              COMM,
                              DEPT_NO
                    from       EMP
                    where      1=1
                ]]>
                <!-- Initial - empNo is the property of parameter bean-->
                <isPropertyAvailable property="empNo" prepend="and">
                        EMP_NO = #empNo#
                </isPropertyAvailable>
                <!-- Repetition – due to connection by column="MGR", empNo is not property -->
                <isNotPropertyAvailable property="empNo" prepend="and">
                        EMP_NO = #mgr#
                </isNotPropertyAvailable>
        </select>
```

The type of recalling its own sql sentence like select="selectMgrHierarchy" for mapping definition for mgrVO member attribute of relevant objects when processing resultMap mapping for EmpIncludesMgrVO containing Complex Properties expressing the Hierarchy relationship in above sql mapping file.

Sample TestCase

```
..
    @Test
    public void testComplexPropertiesHierarcyRepetitionSelect()
            throws Exception {
        EmpVO vo = new EmpVO();
        // 7369,'SMITH','CLERK',7902
        // --> 7902,'FORD','ANALYST',7566
        // --> 7566,'JONES','MANAGER',7839
        // --> 7839,'KING','PRESIDENT',NULL
        vo.setEmpNo(new BigDecimal(7369));

        try {

            // select
            // EmpIncludesMgrVO resultVO =
            // empDAO.selectEmpMgrHierarchy("selectEmpWithMgr", vo);
            EmpIncludesMgrVO resultVO =
                empDAO.selectEmpMgrHierarchy("selectMgrHierarchy", vo);

            // check
            assertNotNull(resultVO);
            assertEquals(new BigDecimal(7369), resultVO.getEmpNo());
            assertEquals("SMITH", resultVO.getEmpName());
            assertEquals("CLERK", resultVO.getJob());
            assertEquals(new BigDecimal(7902), resultVO.getMgr());
            SimpleDateFormat sdf =
                new SimpleDateFormat("yyyy-MM-dd", java.util.Locale
                    .getDefault());
            assertEquals(sdf.parse("1980-12-17"), resultVO.getHireDate());
            assertEquals(new BigDecimal(800), resultVO.getSal());
            assertEquals(new BigDecimal(0), resultVO.getComm());
            assertEquals(new BigDecimal(20), resultVO.getDeptNo());

            assertTrue(resultVO.getMgrVO() instanceof EmpIncludesMgrVO);
```

```
            assertEquals(new BigDecimal(7902), resultVO.getMgrVO().getEmpNo());
            assertEquals(new BigDecimal(7566), resultVO.getMgrVO().getMgrVO()
                .getEmpNo());
            assertEquals(new BigDecimal(7839), resultVO.getMgrVO().getMgrVO()
                .getMgrVO().getEmpNo());
            assertNull(resultVO.getMgrVO().getMgrVO().getMgrVO().getMgrVO());

        } catch (UncategorizedSQLException ue) {
            // In case of tibero, mapping sub object of self query form of ibatis
com.tmax.tibero.jdbc.TbSQLException: TJDBC-90646:Resultset
            // is already closed error generation checked!
            assertTrue(isTibero);
            assertTrue(ue.getCause() instanceof NestedSQLException);
            assertTrue(ue.getCause().getCause().getCause().getCause() instanceof TbSQLException);
            assertTrue(((TbSQLException) ue.getCause().getCause().getCause()
                .getCause()).getMessage().contains(
                "TJDBC-90646:Resultset is already closed"));
        }
    }
```

In above test case validation code, each attribute can be checked to set containing all connected administrator information connected in the form of SMITH → FORD → JONES → KING. Be careful in using since a problem may occur in error.