Relationship 1:1 (Get on both sides)		Relationship 1:1 (Get on one side)	
Tree	Color	Dog	Bone
Use local interfaces /** * @ejb.interface-method view-type = "**local**" * @return */ public abstract Color**Local** getColor(); /** * @ejb.interface-method view-type = "**local**" * * @param colorLocal */ public abstract void setColor(Color**Local** colorLocal);	 /** * @ejb.interface-method view-type = "**local**" * @return */ public abstract Tree**Local** getTree(); /** * @ejb.interface-method view-type = "**local**" * * @param treeLocal */ public abstract void setTree(Tree**Local** treeLocal);	Use local interfaces /** * @ejb.interface-method view-type = "**local**" * @return */ public abstract Bone**Local** getBoneLoca**l**(); /** * @ejb.interface-method view-type = "**local**" * @param boneLocal */ public abstract void setBoneLocal(Bone**Local** boneLocal);	
Define in the javaDoc for the getter the relation with same name on both sides and specify role-name /** * @ejb.interface-method view-type = "local" *** @ejb.relation name = "tree-color"** *** role-name = "tree has color"** * @return */ public abstract ColorLocal getColor();	 /** * @ejb.interface-method view-type = "local" *** @ejb.relation name = "tree-color"** *** role-name = "color of tree"** * @return */ public abstract TreeLocal getTree();	Define in the javaDoc for the getter the relation with same name on one side only /** * @ejb.interface-method view-type = "local" *** @ejb.relation name = "dog-bone"** *** role-name = "dog has bone"** *** target-ejb = "Bone"** *** target-role-name = "bone of dog"** */ public abstract BoneLocal getBoneLocal();	
Add the jboss.relation tag only on **one** side /** * @ejb.interface-method view-type = "local" * @ejb.relation name = "tree-color" * role-name = "tree has color" *** @jboss.relation related-pk-field = "id"** *** fk-column = "house_id"** *** fk-constraint = "true"**	Add the jboss.relation tag only on **one** side /** * @ejb.interface-method view-type = "local" * @ejb.relation name = "tree-color" * role-name = "color of tree" *** @jboss.relation related-pk-field = "id"** *** fk-column = "tree_id"** *** fk-constraint = "true"**	Add the jboss.relation tag /** * @ejb.interface-method view-type = "local" * @ejb.relation name = "dog-bone" * role-name = "dog has bone" * target-ejb = "Bone" * target-role-name = "bone of dog" *** @jboss.relation related-pk-field = "id"** *** fk-column = "bone_id"** *** fk-constraint = "true"** or the other way around /** * @ejb.interface-method view-type = "local" * @ejb.relation name = "dog-bone" * role-name = "dog has bone" * target-ejb = "Bone" * target-role-name = "bone of dog" *** @jboss.target-relation related-pk-field = "id"** *** fk-column = "dog_id"** *** fk-constraint = "true"**	
Cascade delete will delete tree when color is removed * @ejb.relation name = "tree-color" * role-name = "tree has color" *** cascade-delete = "yes"**	Cascade delete will delete color when tree is removed * @ejb.relation name = "tree-color" * role-name = "color of tree" *** cascade-delete = "yes"**	Cascade delete will delete dog when bone is removed * @ejb.relation name = "dog-bone" * role-name = "dog has bone" *** cascade-delete = "yes"** * target-ejb = "Bone" * target-role-name = "bone of dog" Target cascade delete will delete the bone when the dog is removed * @ejb.relation name = "dog-bone" * role-name = "dog has bone" *** cascade-delete = "yes"** * target-ejb = "Bone" * target-role-name = "bone of dog" * target-cascade-delete = "yes"	

Relationship 1:n (Get on both sides)		Relationship 1:n (Get only one the many-side)		Relationship 1:n (Get only one the 1-side)	
Fish	Finger	Bed	Colour	Leaf	Tree
Use a Collection on the many side /** * @ejb.interface-method view-type = "local" * @return */ public abstract Collection getFingers(); /** * @ejb.interface-method view-type = "local" * @param fingers */ public abstract void setFingers(Collection fingers);	Use local interfaces /** * @ejb.interface-method view-type = "local" * @return */ public abstract FishLocal getFish(); /** * @ejb.interface-method view-type = "local" * @param fishLocal */ public abstract void setFish(FishLocal fishLocal);	Use local Interfaces /** * @ejb.interface-method view-type = "**local**" * @return */ public abstract Colour**Local** getColour(); /** * @ejb.interface-method view-type = "**local**" * * @param name */ public abstract void setColour(Colour**Local** colour);			Use local Interfaces /** * @ejb.interface-method view-type = "**local**" * @return */ public abstract Collection get**Leafs**(); /** * @ejb.interface-method view-type = "**local**" * @param Leafs */ public abstract void setLeafs(Collection leafs);
Define in the javaDoc for the getter, the relation with same name on both sides /** * @ejb.interface-method view-type = "local" * @ejb.relation name = "fish-fingers" * role-name = "fish becomes fingers" * @return */ public abstract Collection getFingers();	Define in the javaDoc for the getter, the relation with same name on both sides /** * @ejb.interface-method view-type = "local" * @ejb.relation name = "fish-fingers" * role-name = "fingers from fish" * @return */ public abstract FishLocal getFish();	Relation is defined in the javaDoc of the getter. Put target tags. Put a target **multiple tag=”yes”**! /** * @ejb.interface-method view-type = "local" * @ejb.relation name = "bed-colour" * role-name = "bed has colour" * target-ejb = "Colour" * target-role-name = "colour of bed" * target-multiple = "yes" * @return */ public abstract ColourLocal getColour();			Relation is defined in the javaDoc of the getter. Put target tags. Put a target **multiple tag=”no”**! /** * @ejb.interface-method view-type = "local" * @ejb.relation name = "tree-leafs" * role-name = "tree has leafs" * target-ejb = "Leaf" * target-role-name = "leaf of tree" * target-multiple = "no" * @return */ public abstract Collection getLeafs();
	Add the jboss.relation tag on the many side!!! Jboss target-relation is not working! /** * @ejb.interface-method view-type = "local" * @ejb.relation name = "fish-fingers" * role-name = "fingers from fish" * @jboss.relation related-pk-field = "id" * fk-column = "fish_id" * fk-constraint = "true" * @return */ public abstract FishLocal getFish();	Add the jboss tag. /** * @ejb.interface-method view-type = "local" * @ejb.relation name = "bed-colour" * role-name = "bed has colour" * target-ejb = "Colour" * target-role-name = "colour of bed" * target-multiple = "yes" * @jboss.relation related-pk-field = "id" * fk-column = "farb_id" * fk-constraint = "true" * @return */ public abstract ColourLocal getColour();			Add the jboss tag. /** * @ejb.interface-method view-type = "local" * @ejb.relation name = "tree-leafs" * role-name = "tree has leafs" * target-ejb = "Leaf" * target-role-name = "leaf of tree" * target-multiple = "no" * @jboss.target-relation fk-column = "tree_id" * fk-constraint = "true" * related-pk-field = "id" * @return */ public abstract Collection getLeafs();
No cascade delete here!!! Only possible when the otherside is a 1-side not a many-side.	Cascade delete will delete finger when the fish is removed * @ejb.relation name = "fish-fingers" * role-name = "fingers from fish" * cascade-delete = "yes" * @jboss.relation related-pk-field = "id" * fk-column = "fish_id" * fk-constraint = "true" * @return	No target-cascade delete here!!! Only possible when the otherside is a 1-side not a many-side. * @ejb.relation name = "bed-colour" * role-name = "bed has colour" * cascade-delete = "yes" * target-ejb = "Colour" * target-role-name = "colour of house" * target-multiple = "yes"			No cascade delete here!!! Only possible when the otherside is a 1-side not a many-side. Use target-cascade-delete. * @ejb.relation name = "tree-leafs" * role-name = "tree has leafs" * target-ejb = "Leaf" * target-role-name = "leaf of tree " * target-multiple = "no" * target-cascade-delete = "yes"
Testcode for adding a finger to the collection FishLocalHome fishLocalHome = (FishLocalHome) context.lookup(FishLocalHome.JNDI_NAME); FingerLocalHome fingerLocalHome= (FingerLocalHome)context.lookup(FingerLocalHome.JNDI_NAME); FishLocal fishLocal = fishLocalHome.create(); FingerLocal fingerLocal = fingerLocalHome.create(); fishLocal.getFingers().add(fingerLocal);		Testcode for adding a bed to a Colour InitialContext context = new InitialContext(); ColourLocalHome colourLocalHome = (ColourLocalHome) context.lookup(ColourLocalHome.JNDI_NAME); ColourLocal colourLocal = colourLocalHome.create(); BedLocalHome bed LocalHome = (BedLocalHome) context.lookup(BedLocalHome.JNDI_NAME); BedLocal bed = bed LocalHome.create(); bed.setColour(colourLocal);		Testcode for adding a Leaf to a Tree InitialContext context = new InitialContext(); TreeLocalHome treeLocalHome = (TreeLocalHome) context.lookup (TreeLocalHome.JNDI_NAME); LeafLocalHome leafLocalHome = (LeafLocalHome) context.lookup (LeafLocalHome.JNDI_NAME); TreeLocal tree = treeLocalHome.create(); LeafLocal blatt = leafLocalHome.create(); tree.getLeafs().add(blatt);	

Relationship m:n (Get on both sides)

Pupil

Use local interfaces

```
/**
 * @ejb.interface-method view-type = "local"
 * @return
 */
public abstract Collection getTeachers();

/**
 * @ejb.interface-method view-type = "local"
 * @param teacherLocal
 */
public abstract void setTeachers(Collection teachers);
```

Define in the javaDoc for the getter the relation with same name on both sides and specify role-name

```
/**
 * @ejb.interface-method view-type = "local"
 * @ejb.relation name = "teacher-pupil" role-name = "pupil belongs to teacher"
 */
public abstract Collection getTeachers();
```

Add the jboss.relation tag

```
/**
 * @ejb.interface-method view-type = "local"
 * @ejb.relation name = "teacher-pupil" role-name = "pupil belongs to teacher"
 * @jboss.relation fk-column = "teacher_id"
 * related-pk-field = "id"
 * fk-constraint = "true"
 * @return
 */
```

No cascade delete is possible! Entries in mapping table are delete automatically.

Teacher

Use local interfaces

```
/**
 * @ejb.interface-method view-type = "local"
 * @return
 */
public abstract Collection getPupils();

/**
 * @ejb.interface-method view-type = "local"
 * @param pupil
 */
public abstract void setPupils(Collection pupil);
```

```
/**
 * @ejb.interface-method view-type = "local"
 * @ejb.relation name = "teacher-pupil" role-name = "teacher has pupils"
 */
public abstract Collection getPupils();
```

Add the jboss.relation tag

```
/**
 * @ejb.interface-method view-type = "local"
 * @ejb.relation name = "teacher-pupil" role-name = "teacher has pupils"
 *
 * @jboss.relation related-pk-field = "id"
 * fk-column = "pupil_id"
 * fk-constraint = "true"
 * @return
 */
```

No cascade delete is possible! Entries in mapping table are delete automatically.

Testcode

```
InitialContext context = new InitialContext();
TeacherLocalHome teacherLocalHome = (TeacherLocalHome) context.lookup(TeacherLocalHome.JNDI_NAME);
PupilLocalHome pupilLocalHome = (PupilLocalHome) context.lookup(PupilLocalHome.JNDI_NAME);
TeacherLocal teacherLocal = teacherLocalHome.create();
PupilLocal pupilLocal = pupilLocalHome.create();
teacherLocal.getPupils().add(pupilLocal);
```

Relationship m:n (Unidirectional)

Question

Use local interfaces

```
/**
 * @ejb.interface-method view-type = "both"
 * @return
 */
public abstract Collection getIdiots();

/**
 * @ejb.interface-method view-type = "both"
 * @param idiots
 */
public abstract void setIdiots(Collection idiots);
```

```
/**
 * @ejb.interface-method view-type = "both"
 * @ejb.relation name = "idiots-questions"
 * role-name = "question of idiot"
 * target-role-name = "idiot has question"
 * target-ejb = "Idiot"
 * target-multiple = "yes"
 * @return
 */
public abstract Collection getIdiots();
```

Add the jboss.relation tag

```
/**
* @ejb.interface-method view-type = "both"
* @ejb.relation name = "idiots-questions"
* role-name = "question of idiot"
* target-role-name = "idiot has question"
* target-ejb = "Idiot"
* target-multiple = "yes"
* @jboss.relation fk-column = "idiot_id"
* fk-constraint = "true"
*related-pk-field = "id"
* @jboss.target-relation fk-column = "question_id"
* fk-constraint = "true"
* related-pk-field = "id"
* @return
*/
public abstract Collection getIdiots();

/**
* @ejb.interface-method view-type = "both"
* @param idiots
*/
public abstract void setIdiots(Collection idiots);
```

No cascade delete is possible! Entries in mapping table are delete automatically.

Testcode

```
InitialContext context = new InitialContext();
IdiotLocalHome idiotLocalHome = (IdiotLocalHome) context.lookup(IdiotLocalHome.JNDI_NAME);
QuestionLocalHome questionLocalHome = (QuestionLocalHome) context.lookup(QuestionLocalHome.JNDI_NAME);
IdiotLocal idiotLocal = idiotLocalHome.create();
QuestionLocal questionLocal = questionLocalHome.create();
questionLocal.getIdiots().add(idiotLocal);
```

Idiots