

[返回首页](#)

成员的存储类型

成员的存储类型可以在创建成员时指定也可以创建成员后进行设置，创建时如果没有指定成员的存储类型时则默认为存储型；

目前存储类型(MemberStorageTypes)有：

- Stored : 存储型
- DynamicCalc : 动态计算型
- DynamicCalcAndStored: 动态计算且存储(暂未实现)

1、创建成员时设置成员的存储类型

1.1、创建单个成员时指定成员的存储类型

```
// 创建一个元数据命令
MetadataCommandInfo memberCreateCmd=new MetadataCommandInfo();
// 指定元数据命令为操作成员的命令
memberCreateCmd.setMetadataType(MetadataTypes.Member);
// 命令类型为创建成员
memberCreateCmd.setAction(CommandTypes.create);
// 该成员所属cube及维度名,中间用'.'连接
memberCreateCmd.setOwnerUniqueName("cube_name.dimension_name");

// 新建的成员名
memberCreateCmd.setName("member_name");
// 设置成员存储类型（如设置为动态计算类型）
memberCreateCmd.setStorageType(MemberStorageTypes.DynamicCalc);
```

1.2、批量创建成员时指定成员的存储类型

```
// 创建一个元数据命令
MetadataCommandInfo memberCreateCmd=new MetadataCommandInfo();
// 指定元数据命令为操作成员的命令
memberCreateCmd.setMetadataType(MetadataTypes.Member);
// 命令类型为创建成员
memberCreateCmd.setAction(CommandTypes.create);
// 该成员所属cube及维度名,中间用'.'连接
memberCreateCmd.setOwnerUniqueName("cube_name.dimension_name");

List<MetadataItem> items = memberCreateCmd.getItems();
// 第一个新增成员
MemberMetadataItem item1 = new MemberMetadataItem("member1_name");
item1.setStorageType(MemberStorageTypes.DynamicCalc);
items.add(item1);
```

```
// 第二个新增成员
MemberMetadataItem item2 = new MemberMetadataItem("member2_name");
item2.setStorageType(MemberStorageTypes.DynamicCalc);
items.add(item2);
```

2、修改成员存储类型

```
// 创建一个元数据命令
MetadataCommandInfo memberAlterCmd=new MetadataCommandInfo();
// 指定元数据命令为操作成员的命令
memberAlterCmd.setMetadataType(MetadataTypes.Member);
// 命令类型为修改成员属性
memberAlterCmd.setAction(CommandTypes.alter);
// 该成员所属cube及维度名,中间用'.'连接
memberAlterCmd.setOwnerUniqueName("cube_name.dimension_name");
// 要修改的成员名
memberAlterCmd.setName("member_name");

// 设置成员存储类型（如设置为动态计算类型）
memberAlterCmd.setStorageType(MemberStorageTypes.DynamicCalc);
```

成员的动态计算表达式

成员的动态计算表达式可以在批量创建成员时指定也可以在成员创建后进行设置，支持对一个成员进行表达式创建、表达式修复、修改表达式因子的操作符和删除表达式因子。注意表达式不能出现循环引用。

1、创建成员时指定表达式，需注意表达式因子必须是已创建的成员或者在同一命令中即将创建的成员

```
// 假设 cube1 下维度 dimension1 现有成员 member1、member2
// 现新增成员 member3、member4
// 其中成员 member3 指定其表达式为 member3 = member2 - member4

// 创建一个元数据命令
MetadataCommandInfo memberCreateCmd = new MetadataCommandInfo();
// 指定元数据命令为操作成员的命令
memberCreateCmd.setMetadataType(MetadataTypes.Member);
// 命令类型为创建成员
memberCreateCmd.setAction(CommandTypes.create);
// 该成员所属cube及维度名,中间用'.'连接
memberCreateCmd.setOwnerUniqueName("cube1.dimension1");

List<MetadataItem> items = memberCreateCmd.getItems();
// 第一个新增成员 member3，并指定其表达式为 member3 = -member2 + member4
MemberMetadataItem item1 = new MemberMetadataItem("member3");
List<AggFactorMetadataItem> factorItems = new ArrayList<>();
factorItems.add(new AggFactorMetadataItem("member2", AggOperators.SUBTRACT));
factorItems.add(new AggFactorMetadataItem("member4", AggOperators.PLUS));
item1.setFactors(factorItems);
items.add(item1);
```

```
// 第二个新增成员 member4
MemberMetadataItem item2 = new MemberMetadataItem("member4");
items.add(item2);
```

2、删除成员时，默认将其他表达式中引用被删除成员的因子删除

假设现有成员及其表达式：

```
member1
  |__ -member2
  |__ member3
```

```
member2
  |__ member4
```

```
member3
```

```
member4
```

如删除成员 member2，则现有成员及表达式变更为：

```
member1
  |__ member3
```

```
member3
```

```
member4
```

3、创建成员的表达式

```
// 设置 cube1.dimension1 的成员 member1 的表达式为 member1 = member2 + member3 - member4
```

```
// 创建一个元数据命令
```

```
MetadataCommandInfo factorCreateCmd=new MetadataCommandInfo();
```

```
// 命令类型为创建成员计算表达式
```

```
factorCreateCmd.setMetadataType(MetadataTypes.AggregFactor);
```

```
// 命令操作类型为创建
```

```
factorCreateCmd.setAction(CommandTypes.create);
```

```
// 该表达式所属cube及维度名、成员名，中间用'.'连接
```

```
factorCreateCmd.setOwnerUniqueName("cube1.dimension1.member1");
```

```
List<MetadataItem> items = factorCreateCmd.getItems();
```

```
// 设置成员 member1 的计算表达式为 member2 + member3 - member4
```

```
items.add(new AggregFactorMetadataItem("member2", AggregOperators.PLUS));
```

```
items.add(new AggregFactorMetadataItem("member3", AggregOperators.PLUS));
```

```
items.add(new AggregFactorMetadataItem("member4", AggregOperators.SUBTRACT));
```

4、修复成员的表达式，此操作将新的表达式替换原来的表达式

```
// 假设 cube1 下的 dimension1 的成员 member1 原表达式为 member1 = member2 + member3 - member4 ,
```

```
// 现修改为 member1 = -member2 + member5
```

```

// 创建一个元数据命令
MetadataCommandInfo factorRepairCmd=new MetadataCommandInfo();
// 命令类型为创建成员计算表达式
factorRepairCmd.setMetadataType(MetadataTypes.AggregateFactor);
// 命令操作类型为修复
factorRepairCmd.setAction(CommandTypes.repair);
// 该表达式所属cube及维度名、成员名，中间用'.'连接
factorRepairCmd.setOwnerUniqueName("cube1.dimension1.member1");

List<MetadataItem> items = factorRepairCmd.getItems();
// 修改成员 member1 的计算表达式为 - member2 + member5
items.add(new AggregateFactorMetadataItem("member2", AggregateOperators.SUBTRACT));
items.add(new AggregateFactorMetadataItem("member5", AggregateOperators.PLUS));

```

5、修改表达式因子的操作符

```

// 假设 cube1 下的 dimension1 的成员 member1 原表达式为 member1 = member2 - member3 ,
// 现将因子 [member2] 修改 [-member2]，将因子 [-member3] 修改 [member3]，则新的表达式为
member1 = -member2 + member3

// 创建一个元数据命令
MetadataCommandInfo factorAlterCmd = new MetadataCommandInfo();
// 命令类型为创建成员计算表达式
factorAlterCmd.setMetadataType(MetadataTypes.AggregateFactor);
// 命令操作类型为修改
factorAlterCmd.setAction(CommandTypes.alter);
// 该表达式所属cube及维度名、成员名，中间用'.'连接
factorAlterCmd.setOwnerUniqueName("cube1.dimension1.member1");

List<MetadataItem> items = factorAlterCmd.getItems();
// 将因子 [member2] 修改 [-member2]
items.add(new AggregateFactorMetadataItem("member2", AggregateOperators.SUBTRACT));
// 将因子 [-member3] 修改 [member3]
items.add(new AggregateFactorMetadataItem("member3", AggregateOperators.PLUS));

```

6、删除表达式因子

```

// 假设 cube1 下的 dimension1 的成员 member1 原表达式为 member1 = member2 - member3 ,
// 现将因子 [member2] 删除，则新的表达式为 member1 = member3

// 创建一个元数据命令
MetadataCommandInfo factorDropCmd = new MetadataCommandInfo();
// 命令类型为创建成员计算表达式
factorDropCmd.setMetadataType(MetadataTypes.AggregateFactor);
// 命令操作类型为删除
factorDropCmd.setAction(CommandTypes.drop);
// 该表达式所属cube及维度名、成员名，中间用'.'连接
factorDropCmd.setOwnerUniqueName("cube1.dimension1.member1");

List<MetadataItem> items = factorDropCmd.getItems();

```

```
// 将因子 [member2] 删除
items.add(new AggFactorMetadataItem("memner2", AggOperators.PLUS));
```

动态计算屏蔽规则

多维数据库成员默认有三种内置动态计算屏蔽规则

- all：成员默认采用的屏蔽规则，不屏蔽任何维度
- none：屏蔽任何维度，包括自身维度
- other：屏蔽除自身维度以外的所有维度

动态计算屏蔽规则支持操作：create、drop、repair

- 均采用批量操作模式
- OwnerUniqueName 为 cube 名
- 规则名不能是 null, "", all, none, other, 数字或字母开头；只能包含字母，下划线，横线，数字；最长50字符，不能小于1个字符；且不能创建同名规则；
- 不允许操作all、none、other三个内置规则
- 不能删除已经被某些成员引用的规则

1、新增屏蔽规则

```
// 新增 cube1 下的一个黑名单屏蔽规则，名称为 timePoint，屏蔽维度为 Period 和 Year

// 创建一个元数据命令
MetadataCommandInfo createAggShieldRuleCmd = new MetadataCommandInfo();
// 命令类型为创建成员计算表达式
createAggShieldRuleCmd.setMetadataType(MetadataTypes.AggShieldRule);
// 命令操作类型为新增
createAggShieldRuleCmd.setAction(CommandTypes.create);
// 该屏蔽规则所属cube
createAggShieldRuleCmd.setOwnerUniqueName("cube1");

List<MetadataItem> items = createAggShieldRuleCmd.getItems();
// 新增黑名单屏蔽规则 timePoint，并指定其屏蔽维度列表
AggShieldRuleBlackListMetadataItem item1 = new
AggShieldRuleBlackListMetadataItem("timePoint");
item1.setDimensionList(Arrays.asList("Period", "Year"));
items.add(item1);
```

2、修复屏蔽规则，该操作用于将新的屏蔽维度列表替换旧的屏蔽规则列表

```
// 假设 cube1 下现有黑名单屏蔽规则 timePoint，屏蔽维度为 Period 和 Year
// 现将其屏蔽维度修复为 Period

// 创建一个元数据命令
MetadataCommandInfo repairAggShieldRuleCmd = new MetadataCommandInfo();
// 命令类型为创建成员计算表达式
repairAggShieldRuleCmd.setMetadataType(MetadataTypes.AggShieldRule);
// 命令操作类型为修复
repairAggShieldRuleCmd.setAction(CommandTypes.repair);
```

```
// 该屏蔽规则所属cube
repairAggShieldRuleCmd.setOwnerUniqueName("cube1");

List<MetadataItem> items = repairAggShieldRuleCmd.getItems();
// 修复黑名单屏蔽规则 timePoint 的屏蔽维度列表
AggShieldRuleBlackListMetadataItem item1 = new
AggShieldRuleBlackListMetadataItem("timePoint");
item1.setDimensionList(Arrays.asList("Period"));
items.add(item1);
```

3、删除屏蔽规则，注意不能删除已经被某些成员引用的规则

```
// 假设 cube1 下现有黑名单屏蔽规则 timePoint，屏蔽维度为 Period 和 Year
// 现将其删除

// 创建一个元数据命令
MetadataCommandInfo dropAggShieldRuleCmd = new MetadataCommandInfo();
// 命令类型为创建成员计算表达式
dropAggShieldRuleCmd.setMetadataType(MetadataTypes.AggShieldRule);
// 命令操作类型为删除
dropAggShieldRuleCmd.setAction(CommandTypes.drop);
// 该屏蔽规则所属cube
dropAggShieldRuleCmd.setOwnerUniqueName("cube1");

List<MetadataItem> items = dropAggShieldRuleCmd.getItems();
// 删除黑名单屏蔽规则 timePoint
AggShieldRuleBlackListMetadataItem item1 = new
AggShieldRuleBlackListMetadataItem("timePoint");
// 此处无需指定屏蔽维度列表
// item1.setDimensionList(Arrays.asList("Period", "Year"));
items.add(item1);
```

设置成员动态计算屏蔽规则

成员的动态计算屏蔽规则可以在创建成员时指定也可以创建后进行设置，创建时没有指定动态计算屏蔽规则时则默认为 all；

1、创建成员时设置成员的动态计算屏蔽规则

1.1、创建单个成员时指定动态计算屏蔽规则

```

// 创建一个元数据命令
MetadataCommandInfo memberCreateCmd=new MetadataCommandInfo();
// 指定元数据命令为操作成员的命令
memberCreateCmd.setMetadataType(MetadataTypes.Member);
// 命令类型为创建成员
memberCreateCmd.setAction(CommandTypes.create);
// 该成员所属cube及维度名,中间用'.'连接
memberCreateCmd.setOwnerUniqueName("cube_name.dimension_name");

// 新建的成员名
memberCreateCmd.setName("member_name");
// 设置成员的动态计算屏蔽规则(如设置为 timePoint)
memberCreateCmd.setMemberAggShieldRule("timePoint");

```

1.2、批量创建成员时指定动态计算屏蔽规则

```

// 创建一个元数据命令
MetadataCommandInfo memberCreateCmd=new MetadataCommandInfo();
// 指定元数据命令为操作成员的命令
memberCreateCmd.setMetadataType(MetadataTypes.Member);
// 命令类型为创建成员
memberCreateCmd.setAction(CommandTypes.create);
// 该成员所属cube及维度名,中间用'.'连接
memberCreateCmd.setOwnerUniqueName("cube_name.dimension_name");

List<MetadataItem> items = memberCreateCmd.getItems();
// 第一个新增成员
MemberMetadataItem item1 = new MemberMetadataItem("member1_name");
item1.setMemberAggShieldRule("timePoint");
items.add(item1);

// 第二个新增成员
MemberMetadataItem item2 = new MemberMetadataItem("member2_name");
item1.setMemberAggShieldRule("timePoint");
items.add(item2);

```

2、修改成员存储类型

```
// 创建一个元数据命令
MetadataCommandInfo memberAlterCmd=new MetadataCommandInfo();
// 指定元数据命令为操作成员的命令
memberAlterCmd.setMetadataType(MetadataTypes.Member);
// 命令类型为修改成员属性
memberAlterCmd.setAction(CommandTypes.alter);
// 该成员所属cube及维度名,中间用'.'连接
memberAlterCmd.setOwnerUniqueName("cube_name.dimension_name");
// 要修改的成员名
memberAlterCmd.setName("member_name");

// 设置成员的动态计算屏蔽规则 (如设置为 all)
memberAlterCmd.setMemberAggShieldRule("all");
```

上一篇：[计算引擎](#)

下一篇：[沙箱功能](#)