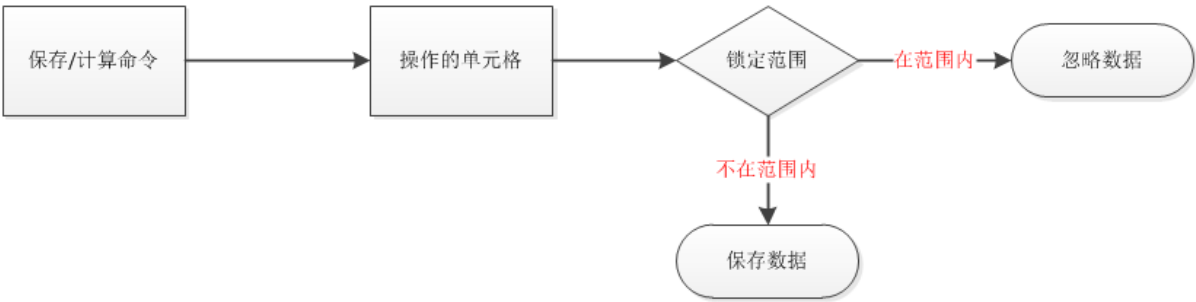


数据锁定规则

多维库提供一套对锁定范围进行创建、修改和删除的元数据命令。

用户发送保存和计算命令时，最终保存到数据库的实际单元格会通过锁定范围进行一次过滤，如果单元格被锁定，则忽略，不存储到多维库。



数据锁定规则元数据操作命令

范围创建

```
/**
 * 操作数据范围
 *
 * @param scopeItems 具体的数据范围元数据信息列表
 * @param type       可执行 create,alter,drop,repair
 */
public static void execCubeDataScope(CommandTypes type, CubeDataScopeItem... scopeItems) {
    OlapConnection olapConn = OlapConnectionUtil.getConnection(cubeName);
    MetadataCommandInfo commandInfo = new MetadataCommandInfo();
    commandInfo.setMetadataType(MetadataTypes.CubeLockRule);
    commandInfo.setOwnerUniqueName(cubeName); // cubeName
    commandInfo.setAction(type); // create,drop,repair
    Arrays.stream(scopeItems).forEach(item -> {
        commandInfo.getItems().add(item);
    });

    OlapCommand command = new OlapCommand(olapConn, commandInfo);
    command.executeNonQuery();
}
```

创建一个数据锁定范围，范围名称为 `test1`，锁定的范围空间为：`year in (2020,2021,2022) and period in (1,2,3) and entity in (a1,a2,a3)`

```

/**
 * 创建范围
 */
public void createScope() {
    SliceScopeItemV2 test1 = new SliceScopeItemV2("test1");
    test1.addFilter("year", "2020", "2021", "2022");
    test1.addFilter("period", "1", "2", "3");
    test1.addFilter("entity", "a1", "a2", "a3");
    execCubeDataScope(CommandTypes.create, test1);
}

```

注意：一旦创建了范围，后续新增的范围必须要包含与之前相同的维度，且需要顺序一致

批量创建范围：test2, test3

```

/**
 * 批量创建范围
 */
public void createScopes() {
    SliceScopeItemV2 test2 = new SliceScopeItemV2("test2");
    test2.addFilter("year", "2023");
    test2.addFilter("period", "4", "5");
    test2.addFilter("entity", "a4");
    SliceScopeItemV2 test3 = new SliceScopeItemV2("test3");
    test3.addFilter("year", "2023");
    test3.addFilter("period", "1", "2");
    test3.addFilter("entity", "a2");
    execCubeDataScope(CommandTypes.create, test2, test3);
}

```

创建范围时，也支持* 符号，表示当前维度下的所有成员:

范围 test4 表示 2020年的所有期间所有组织

```

public void createScope2() {
    SliceScopeItemV2 test1 = new SliceScopeItemV2("test4");
    test1.addFilter("year", "2020");
    test1.addFilter("period", "*");
    test1.addFilter("entity", "*");
    execCubeDataScope(CommandTypes.create, test1);
}

```

范围查询

通过函数接口 getCubeDataLockRule ，可基于名称进行查询

执行自定义方法

getCubeDataLockRule

cubeName ?

testCube

name ?

test3

执行

scope

```
1 [{
2     "dimensions": ["year", "period", "entity"],
3     "name": "test3",
4     "members": [
5         ["2023"],
6         ["1", "2"],
7         ["a2"]
8     ]
9 }]
```

```
/**
 * 根据scopeName 获取范围信息
 *
 * @param connection
 * @param cubeName
 * @param scopeName
 * @return
 */
private static String getCubeDataLockRule(OlapConnection connection, String cubeName, String
scopeName) {
    PropertyBag propertyBag = new PropertyBag();
    propertyBag.set("cubeName", cubeName);
    propertyBag.set("name", scopeName);
    FunctionCommandInfo functionCommandInfo = new FunctionCommandInfo("getCubeDataLockRule",
propertyBag);
    OlapCommand cmd = new OlapCommand(connection, functionCommandInfo);
    PropertyBag retPropertyBag = cmd.executeFunction();
    return retPropertyBag.get("scope");
}
public static void main(String[] args) {
    OlapConnection olapConn = OlapConnectionUtil.getConnection(cubeName);
    String scopeInfo = getCubeDataLockRule(olapConn, cubeName, "test1");
    System.out.println(scopeInfo);
}
```

输出结果：

```
[{
  "dimensions": ["year", "period", "entity"],
  "name": "test1",
  "members": [["2020", "2021", "2022"], ["1", "2", "3"], ["a1", "a2", "a3"]]
}]
```

执行保存操作：

```
public void saveData() {
    OlapConnection olapConn = OlapConnectionUtil.getConnection(cubeName);
    SaveCommandInfo saveCommandInfo = new SaveCommandInfo();
    saveCommandInfo.setDimensions("year", "period", "entity", "scenario", "account");
    saveCommandInfo.setMeasures("m1");
    OlapCommand cmd = new OlapCommand(olapConn, saveCommandInfo);
    OlapDataWriter writer = cmd.CreateWriter();

    Object[] newRow = new Object[]{100, "2023", "1", "a2", "MRPT", "b2"};
    writer.setValues(newRow);
    newRow = new Object[]{101, "2023", "1", "a3", "MRPT", "b2"};
    writer.setValues(newRow);
    writer.flush();
}
```

结果如下：{"year":"2023","period":"1","entity":"a2","account":"b2","scenario":"NRPT"} 单元格被锁定范围 test3 锁定了，无法保存成功。

如果希望保存数据的时候，一旦出现保存单元格被锁定就报错提示，保存命令可调用方法

```
setDisallowLockData(true)。
```

```
// 一旦出现保存单元格被锁定就报错提示
saveCommandInfo.setDisallowLockData(true);
```

范围删除

```
/**
 * 执行批量删除操作
 */
public void dropScope() {
    execCubeDataScope(CommandTypes.drop, new SliceScopeItemV2("test2"), new
    SliceScopeItemV2("test3"));
}
```

再次执行保存，结果如下：{"year":"2023","period":"1","entity":"a2","account":"b2","scenario":"NRPT"} 单元格保存成功。

修改范围

```

/**
 * 修改数据范围
 */
public static void alterCubeDataScope(String scopeName, CubeDataScopeItem... scopeItems) {
    OlapConnection olapConn = OlapConnectionUtil.getConnection(cubeName);
    MetadataCommandInfo commandInfo = new MetadataCommandInfo();
    commandInfo.setMetadataType(MetadataTypes.CubeLockRule);
    commandInfo.setOwnerUniqueName(cubeName); // cubeName
    commandInfo.setName(scopeName); // scopeName
    commandInfo.setAction(CommandTypes.alter);
    Arrays.stream(scopeItems).forEach(item -> {
        commandInfo.getItems().add(item);
    });
    OlapCommand command = new OlapCommand(olapConn, commandInfo);
    command.executeNonQuery();
}

/**
 * 修改scope: 可修改名称
 */
@Test
public void alterScope() {
    SliceScopeItemV2 test1 = new SliceScopeItemV2("new_test");
    test1.addFilter("year", "2023");
    test1.addFilter("period", "1", "2", "3");
    test1.addFilter("entity", "a1", "a2", "a3");
    alterCubeDataScope("test1", test1);
}

```

将原 `test1` 范围 `year in (2020,2021,2022) and period in (1,2,3) and entity in (a1,a2,a3)` 修改为 `year in (2023) and period in (1,2,3) and entity in (a1,a2,a3)` , 同时名称修改为 `new_test`

批量修改

支持批量修改, 但是 `alterCubeDataScope` 方法中 `scopeName` 为空。

```

@Test
public void alterScopes() {
    SliceScopeItemV2 new_test = new SliceScopeItemV2("new_test");
    new_test.addFilter("year", "2022");
    new_test.addFilter("period", "1", "2", "3");
    new_test.addFilter("entity", "a1", "a2", "a3");

    SliceScopeItemV2 test4 = new SliceScopeItemV2("test4");
    test4.addFilter("year", "2022");
    test4.addFilter("period", "1");
    test4.addFilter("entity", "a1", "a2", "a3");
    alterCubeDataScope("", test4, new_test);
}

```

重置范围

删除之前的范围，新增 `scope1` 和 `scope2` 范围。（重置后，新的范围维度可与之前的可以不一致）

```
public void repairScope() {
    SliceScopeItemV2 scope1 = new SliceScopeItemV2("scope1");
    scope1.addFilter("year", "2023");
    scope1.addFilter("period", "1");
    scope1.addFilter("entity", "a2");
    scope1.addFilter("scenario", "MRPT");
    SliceScopeItemV2 scope2 = new SliceScopeItemV2("scope2");
    scope2.addFilter("year", "2023");
    scope2.addFilter("period", "1");
    scope2.addFilter("entity", "a3");
    scope2.addFilter("scenario", "MRPT");
    execCubeDataScope(CommandTypes.repair, scope1, scope2);
}
```

范围内容修改

新增内容

现有范围 `scope1` 内容为 `year in (2023) and period in (1) and entity in (a2) and scenario in (MRPT)`，需要新增增加维度成员 `period in (2,3)`

```
/**
 * 执行范围成员操作
 *
 * @param type
 * @param scopeName
 * @param item
 */
public static void execCubeDataScopeMembers(CommandTypes type, String scopeName,
CubeDataScopeItem item) {
    OlapConnection olapConn = OlapConnectionUtil.getConnection(cubeName);
    MetadataCommandInfo commandInfo = new MetadataCommandInfo();
    commandInfo.setMetadataType(MetadataTypes.CubeLockRule);
    commandInfo.setOwnerUniqueName(cubeName + "." + scopeName); // cubeName.scopeName
    commandInfo.setAction(type); // create or drop
    commandInfo.getItems().add(item);
    OlapCommand command = new OlapCommand(olapConn, commandInfo);
    command.executeNonQuery();
}

/**
 * 新增范围内的维度成员
 */
public void createScopeMembers() {
    SliceScopeItemV2 scope1 = new SliceScopeItemV2("scope1");
    scope1.addFilter("year");
    scope1.addFilter("period", "2", "3");
    scope1.addFilter("entity");
    scope1.addFilter("scenario");
}
```

```
    execCubeDataScopeMembers(CommandTypes.create, "scope1", scope1);
}
```

修改范围时，必须要包含范围所需要的所有维度，且需要顺序一致

支持 * 符号，表示当前维度下的所有成员：

```
/**
 * 新增范围内的维度成员
 */
public void createScopeMembers2() {
    SliceScopeItemV2 scope1 = new SliceScopeItemV2("scope1");
    scope1.addFilter("year");
    scope1.addFilter("period");
    scope1.addFilter("entity", "*");
    scope1.addFilter("scenario");
    execCubeDataScopeMembers(CommandTypes.create, "scope1", scope1);
}
```

注意：* 号对后续本维度新增的成员也是有效的。

比如组织维度(entity) 增加一个成员 a_new，此成员自动归并到 scope1 范围内

```
public void createMember() {
    OlapConnection olapConn = OlapConnectionUtil.getConnection(cubeName);
    MetadataCommandInfo commandInfo = new MetadataCommandInfo();
    commandInfo.setAction(CommandTypes.create);
    commandInfo.setMetadataType(MetadataTypes.Member);
    commandInfo.setName("a_new");
    commandInfo.setOwnerUniqueName(cubeName + "." + "entity");
    OlapCommand command = new OlapCommand(olapConn, commandInfo);
    command.executeNonQuery();
}
```

查看范围：

执行自定义方法

getCubeDataLockRule

▼

cubeName ?

testCube

name ?

scope1

执行

scope

```
1 [{
2     "dimensions": ["year", "period", "entity", "scenario"],
3     "name": "scope1",
4     "members": [
5         ["2023"],
6         ["2", "3", "1"],
7         ["*"],
8         ["MRPT"]
9     ]
10  }]
```

注意，如果对范围 `scope1` 中的组织维度(`entity`) 删除一个成员，然后再新增这个成员，将不再拥有 `*` 的特性。

```
public void repairScopeMembers() {
    // entity 原成员为 * , 现在删除 a2
    SliceScopeItemV2 dropScopeMember = new SliceScopeItemV2("scope1");
    dropScopeMember.addFilter("year");
    dropScopeMember.addFilter("period");
    dropScopeMember.addFilter("entity", "a2");
    dropScopeMember.addFilter("scenario");
    execCubeDataScopeMembers(CommandTypes.drop, "scope1", dropScopeMember);
    // 然后新增 a2
    SliceScopeItemV2 createScopeMember = new SliceScopeItemV2("scope1");
    createScopeMember.addFilter("year");
    createScopeMember.addFilter("period");
    createScopeMember.addFilter("entity", "a2");
    createScopeMember.addFilter("scenario");
    execCubeDataScopeMembers(CommandTypes.create, "scope1", createScopeMember);
}
```

查看范围：

执行自定义方法

getCubeDataLockRule

cubeName ?

testCube

name ?

scope1

执行

scope

```
1  [{
2      "dimensions": ["year", "period", "entity", "scenario"],
3      "name": "scope1",
4      "members": [
5          ["2023"],
6          ["2", "3", "1"],
7          ["a2", "a1", "a3", "a4", "a5", "a_new"],
8          ["MRPT"]
9      ]
10 }]
```

删除内容

现有范围 `scope1` 需要删除维度成员 `period in (2,3)`

```
/**
 * 删除范围内的维度成员
 */
public void dropScopeMembers() {
    SliceScopeItemV2 scope1 = new SliceScopeItemV2("scope1");
    scope1.addFilter("year");
    scope1.addFilter("period", "2", "3");
    scope1.addFilter("entity");
    scope1.addFilter("scenario");
    execCubeDataScopeMembers(CommandTypes.drop, "scope1", scope1);
}
```

验证单元格是否锁定

通过函数接口 `getCubeDataLockRule`，可基于单元格进行查询

执行自定义方法

getCubeDataLockRule

cubeName ?

testCube

name ?

{ "year": "2023", "period": "1", "entity": "a1", "account": "b1", "scenario": "MRPT" }

执行

isLock



```
/**
 * 根据单元格查询是否处于锁定范围
 *
 * @param connection
 * @param cubeName
 * @param cellKey
 * @return
 */
private static Boolean isLock(OlapConnection connection, String cubeName, String cellKey) {
    PropertyBag propertyBag = new PropertyBag();
    propertyBag.set("cubeName", cubeName);
    propertyBag.set("cellKey", cellKey);
    FunctionCommandInfo functionCommandInfo = new FunctionCommandInfo("getCellLockStatus",
propertyBag);
    OlapCommand cmd = new OlapCommand(connection, functionCommandInfo);
    PropertyBag retPropertyBag = cmd.executeFunction();
    return Boolean.valueOf(retPropertyBag.get("isLock"));
}
public static void main(String[] args) {
    OlapConnection olapConn = OlapConnectionUtil.getConnection(cubeName);
    Boolean isLock = isLock(olapConn, cubeName, "
{"year": "2023", "period": "1", "entity": "a1", "account": "b1", "scenario": "MRPT"}"
);
    System.out.println(isLock);
}
```

结果如下：

删除内容特性说明

现有范围 `scope1` 中 `period` 维度只有一个成员 `1`，假设删除内容为 `period in (1)`，则会自动将该范围 `scope1` 删除。

```
/**
 * 删除范围内的维度成员，使该维度下成员为空
 */
public void dropScopeMembers2() {
    SliceScopeItemV2 scope1 = new SliceScopeItemV2("scope1");
    scope1.addFilter("year");
    scope1.addFilter("period", "1");
    scope1.addFilter("entity");
    scope1.addFilter("scenario");
    execCubeDataScopeMembers(CommandTypes.drop, "scope1", scope1);
}
```

查看所有的范围名称

通过函数接口 `getCubeDataLockRuleNames`，可查看所有的范围名称

执行自定义方法

getCubeDataLockRuleNames

x

cubeName ?

testCube

执行

scopeNames

1	["scope2"]
---	------------

实际 `scope1` 已经被自动删除了

支持批量新增范围中的成员

比如当前已存在 `1000` 个范围，用户希望批量的增加成员，对每个范围的 `year` 维度增加 `2022`

元数据命令中将 `dataLockAppendMember` 属性设置为 `true` 表明是新增内容

```
public static void appendScopeMembers(CubeDataScopeItem... scopeItems) {
    OlapConnection olapConn = OlapConnectionUtil.getConnection(cubeName);
    MetadataCommandInfo commandInfo = new MetadataCommandInfo();
    commandInfo.setMetadataType(MetadataTypes.CubeLockRule);
}
```

```

commandInfo.setOwnerUniqueName(cubeName);
// 追加成员
commandInfo.getProperties().set("dataLockAppendMember", "true");
commandInfo.setAction(CommandTypes.create);
Arrays.stream(scopeItems).forEach(item -> {
    commandInfo.getItems().add(item);
});
OlapCommand command = new OlapCommand(olapConn, commandInfo);
command.executeNonQuery();
}

public void alterScopeAppendMembers() {
    // "*" 代表全部
    SliceScopeItemV2 scope = new SliceScopeItemV2("*");
    scope.addFilter("year", "2022");
    scope.addFilter("period");
    scope.addFilter("entity");
    scope.addFilter("scenario");
    appendScopeMembers(scope);
}

```

用户希望对其中某几个scope 进行批量的增加成员，比如 scope1 的 year 维度添加 2021，scope2 的 period 维度添加 5

```

public void alterScopeAppendMembers2() {
    SliceScopeItemV2 scope1 = new SliceScopeItemV2("scope1");
    scope1.addFilter("year", "2021");
    scope1.addFilter("period");
    scope1.addFilter("entity");
    scope1.addFilter("scenario");

    SliceScopeItemV2 scope2 = new SliceScopeItemV2("scope2");
    scope2.addFilter("year");
    scope2.addFilter("period", "5");
    scope2.addFilter("entity");
    scope2.addFilter("scenario");
    appendScopeMembers(scope1, scope2);
}

```

支持批量删除范围中的成员

比如当前已存在 1000 个范围，用户希望批量的删除成员，对每个范围的 year 维度删除 2022

元数据命令中将 dataLockRemoveMember 属性设置为 true 表明是删除内容

```

public static void removeScopeMembers(CubeDataScopeItem... scopeItems) {
    OlapConnection olapConn = OlapConnectionUtil.getConnection(cubeName);
    MetadataCommandInfo commandInfo = new MetadataCommandInfo();
    commandInfo.setMetadataType(MetadataTypes.CubeLockRule);
    commandInfo.setOwnerUniqueName(cubeName);
    // 移除成员
}

```

```

        commandInfo.getProperties().set("dataLockRemoveMember", "true");
        commandInfo.setAction(CommandTypes.drop);
        Arrays.stream(scopeItems).forEach(item -> {
            commandInfo.getItems().add(item);
        });
        OlapCommand command = new OlapCommand(olapConn, commandInfo);
        command.executeNonQuery();
    }

    public void removeScopeAppendMembers() {
        // "*" 代表全部
        SliceScopeItemV2 scope = new SliceScopeItemV2("*");
        scope.addFilter("year", "2022");
        scope.addFilter("period");
        scope.addFilter("entity");
        scope.addFilter("scenario");
        removeScopeMembers(scope);
    }

```

用户希望对其中某几个scope 进行批量的删除成员，比如 scope1 的 year 维度删除 2021，scope2 的 period 维度删除 5

```

    public void removeScopeAppendMembers2() {
        SliceScopeItemV2 scope1 = new SliceScopeItemV2("scope1");
        scope1.addFilter("year", "2021");
        scope1.addFilter("period");
        scope1.addFilter("entity");
        scope1.addFilter("scenario");

        SliceScopeItemV2 scope2 = new SliceScopeItemV2("scope2");
        scope2.addFilter("year");
        scope2.addFilter("period", "5");
        scope2.addFilter("entity");
        scope2.addFilter("scenario");
        removeScopeMembers(scope1, scope2);
    }

```

命令中明确不锁定

在某些场景下，用户可能希望跳过锁定，可以在保存命令或计算命令上设置属性 ignoreDataLock 值为 true，命令执行时自动忽略锁定规则。

```

    public void saveData() {
        OlapConnection olapConn = OlapConnectionUtil.getConnection(cubeName);
        SaveCommandInfo saveCommandInfo = new SaveCommandInfo();
        saveCommandInfo.setDimensions("year", "period", "entity", "scenario", "account");
        saveCommandInfo.setMeasures("m1");
        // 忽略数据锁定
        saveCommandInfo.setIgnoreDataLock(true);
        OlapCommand cmd = new OlapCommand(olapConn, saveCommandInfo);
        OlapDataWriter writer = cmd.CreateWriter();
    }

```

```

Object[] newRow = new Object[]{100, "2023", "1", "a2", "MRPT", "b2"};
writer.setValues(newRow);
newRow = new Object[]{101, "2023", "1", "a3", "MRPT", "b2"};
writer.setValues(newRow);
writer.flush();
}

private static void compute(OlapConnection olapConn) {
    ComputingCommandInfo commandInfo = new ComputingCommandInfo();
    commandInfo.setMainMeaName("m1");
    commandInfo.setMainDimName("period");
    commandInfo.addFilter("year", "2018");
    commandInfo.addFilter("entity", "a3");
    commandInfo.addFilter("scenario", "MRPT");
    commandInfo.addFilter("account", "b3");
    // 忽略数据锁定
    commandInfo.setIgnoreDataLock(true);
    FellambdaExpressionItem item = new FellambdaExpressionItem();
    item.setExpressLeft("period@3");
    item.setExpression("value('period@2') +value('period@1') ");
    commandInfo.addExpression(item);
    OlapCommand cmd = new OlapCommand(olapConn, commandInfo);
    cmd.executeCompute();
}

```

上一篇：[分段公式](#)

下一篇：[简单成员](#)