

[返回首页](#)

cube 分区

Shrek 多维数据库提供 cube 分区功能，可根据数据分布特点和使用频率，进行分区，将数据存储于不同的区域。适当的分区有助于提高数据库性能，但是分区是一项耗时的操作，不可频繁操作。

一、分区配置方式

Shrek 提供两种不同的分区配置方式

1. 按照维度进行分区。

用户可提供维度清单作为分区维度，系统将分区维度的成员使用笛卡尔积的方式组合出多个分区，数据将存储到对应的分区中。

例如，按照维度 "Year"（成员："FY2023"，"FY2024"）和 "Period"（成员："Q1"，"Q2"，"Q3"，"Q4"）进行分区，

系统将组合出如下分区：FY2023&Q1, FY2023&Q2, FY2023&Q2, FY2023&Q2, FY2023&Q1, FY2023&Q2, FY2023&Q2, FY2023&Q2。

```
// 创建一个元数据命令
MetadataCommandInfo commandInfo = new MetadataCommandInfo();
// 指定元数据命令类型为分区相关的命令
commandInfo.setMetadataType(MetadataTypes.Partition);
// 指定命令类型为修复分区
commandInfo.setAction(CommandTypes.repair);
// 指定进行分区的 cube
commandInfo.setOwnerUniqueName("cube_name");
// 设置分区维度，参数为可变参数
commandInfo.addPartitions("Year", "Period");

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

可以通过 WEB 界面按钮进行分区



2. 简单分层分区。

用户可以提供指定维度的多层分区配置方式，每一层对一个维度进行分区，相比“按照维度进行分区”，有丰富的分区方式，包括自定义分组，哈希等。例如，现有维度

"BudgetPeriod"（成员："FY2022.M01"... "FY2022.M12", "FY2023.M01"... "FY2023.M12",

"FY2024.M01"... "FY2024.M12")

"Entity" (成员 : "001", "001.01", "001.02", "002", "002.01", "002.02"...)

第一层使用分组的方式，将相同历史年份（FY2022, FY2023）的数据存储在相同的分区中，剩下属于 FY2024 的成员则每个成员一个分区。

第二层对于属于 FY2024 的所有分区再进一步分区，先用分组的方式，将 001，001.01，001.02 组成一个分区，002，002.01，002.02 组成一个分区，剩下的成员按照哈希分区，分成 10 个分区。

如下为该分区方案的配置方式示例：

```
[{
  "type": "partition.v2",
  "name": "Year",
  "groups": {
    "FY2022": ["^{FY2022.*}"],
    "FY2023": ["^{FY2023.*}"]
  },
  "hashSize": 0,
  "drillDownFilter": ["^{FY2024.*}"]
},
{
  "type": "partition.v2",
  "name": "Period",
  "groups": {
    "Entity_001": ["001", "001.01", "001.02"],
    "Entity_002": ["002", "002.01", "002.02"]
  },
  "hashSize": 10
}]
```

- **name** 描述了使用哪个维度进行分区。
- **groups** 描述了此维度的所有成员，如何按照自定义的方式分组，同一个组的成员将分配到对应的分区。
 1. 支持直接写成员名称，或者使用正则表达式匹配多个成员（需要加上 {} 进行标识），或者两者混合；
 2. 成员按照组的顺序先匹配的先归类；
- **hashSize** 当执行完 groups 分组后，如果还剩余一些成员没有分配到任何组，这个时候就用到 hashSize 属性。
 1. 如果此值大于 0，表示剩余成员按 $\text{fixedNumber} \% \text{hashSize}$ 为键，分组到不同分区，由于 groups 允许为 null，变相实现此维度完全 hash 分区；
 2. 如果 hashSize 小于等于 0，那么就会每个成员独立一个分区；
 3. groups 和 hashSize 都不填，就是按照每个成员独立一个分区；
- **drillDownFilter** 不为 null 时表示支持下一层的分区。
 1. 符合 drillDownFilter 过滤条件的分区可以进行下一层分区
 2. 支持直接写分区名称，或者使用正则表达式匹配多个分区（需要加上 {} 进行标识），或者两者混合。

如下为示例分区方案的代码形式：

```
// 创建一个元数据命令
```

```

MetadataCommandInfo commandInfo = new MetadataCommandInfo();
// 指定元数据命令类型为分区相关的命令
commandInfo.setMetadataType(MetadataTypes.Partition);
// 指定命令类型为修复分区
commandInfo.setAction(CommandTypes.repair);
// 指定进行分区的 cube
commandInfo.setOwnerUniqueName("cube_name");
// 配置第一层分区
SimpleLevelPartitionMetadataItem level1 = new SimpleLevelPartitionMetadataItem("Year");
level1.addGroup("FY2022", new String[]{"^FY2022.*"});
level1.addGroup("FY2023", new String[]{"^FY2023.*"});
level1.setHashSize(0);
level1.setDrillDownFilter(new String[]{"^FY2024.*"});
// 配置第二层分区
SimpleLevelPartitionMetadataItem level2 = new
SimpleLevelPartitionMetadataItem("Period");
level2.addGroup("Entity_001", new String[]{"001", "001.01", "001.02"});
level2.addGroup("Entity_002", new String[]{"002", "002.01", "002.02"});
level2.setHashSize(10);

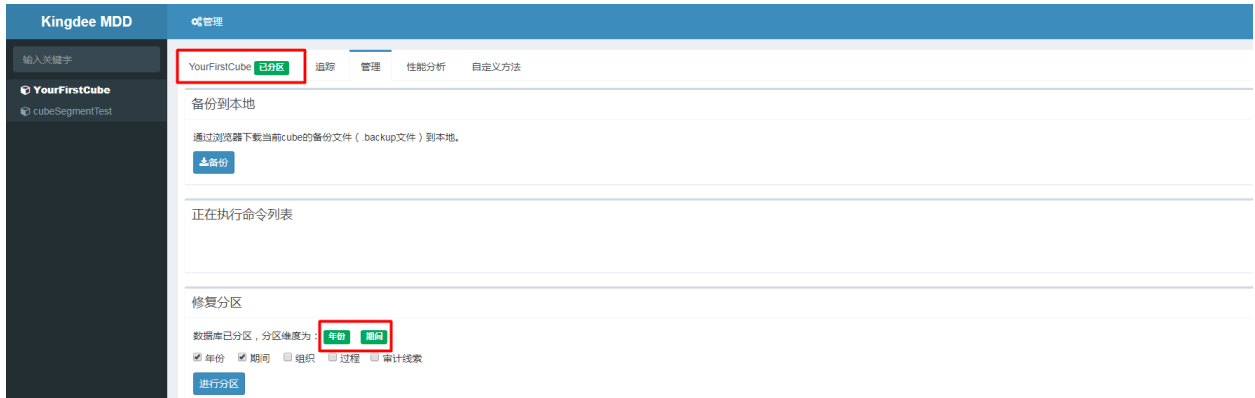
// 按顺序添加到 Items
commandInfo.getItems().add(level1);
commandInfo.getItems().add(level2);

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

```

二、获取 cube 的分区配置方式

1. 如一个 cube 已分区，界面的相关提示如下：



2. 通过自定义方法获取 cube 的分区配置方式：

xyfTest已分派

追踪

管理

性能分析

慢语句查询

自定义方法

异步任务中心

正常

执行自定义方法

getPartitionV2

执行

partition_scheme

```
1 {
2   "type": "singleLevelPartition.v1",
3   "dimension": "Year",
4   "groups": "[\\current\\:\\PF2024\\]\\history\\:\\{.*}\\]",
5   "hashSize": 0,
6   "drillDownFilter": "[\\current\\]",
7   "next": {
8     "type": "singleLevelPartition.v1",
9     "dimension": "Period",
10    "hashSize": 0
11  }
12 }
```

partition_items

```
1 ["currentAM_M06", "currentPeriod", "currentLastPeriod", "currentAM_M05", "currentAM_M08", "currentAM_M07", "currentAM_M09", "currentAM_YearTotal", "currentAM_M11", "currentAM_M10", "currentAM_M02", "currentAM_M01", "currentAM_M12", "currentAM_M04",
2  "currentAM_M03", "currentAMCurrentPeriod", "history"]
```

partition_items_with_data

```
1 {
2   "currentAM_M06": 1,
3   "currentAM_M02": 17149,
4   "currentAM_M01": 152471,
5   "currentAM_M12": 573,
6   "history": 244565
7 }
```

- 上一篇：[沙箱功能](#)
- 下一篇：[有效维度组合规则](#)