

[返回首页](#)

## 临时沙箱功能

允许基于一个已存在的Cube，创建一个短暂的Cube环境，称为“临时沙箱”，在这个环境中：

1. 允许执行 `新增成员`、`保存`、`查询`、`计算` 或 `动态计算`；
2. 任何修改都不会影响到原始Cube；
3. 允许多次修改迭代，下一次的查询可以基于之前的修改；
4. 基于同一个原始Cube，可以同时创建多个临时沙箱；

## 沙箱基本功能操作

临时沙箱中的操作命令同 `cube` 中的操作命令是一致的。

临时沙箱不支持如下操作：

1. 不支持对cube的新增，修改和删除。
2. 不支持对维度的新增，修改和删除。
3. 不支持对成员的删除。
4. 分区环境下，不支持新增分区维度的成员
5. 沙箱环境下，不支持创建沙箱（沙箱嵌套）

```
public static void sandboxTest(OlapConnection olapConn) {
    Closeable sandbox = olapConn.createSandbox();
    // 在该链接上执行的所有命令都是对沙箱的操作
    try {
        // 对沙箱执行各种操作，像以往代码一样进行保存，查询，计算或动态计算
        ....
    } finally {
        try {
            // 沙箱使用完后，必须要手动关闭沙箱
            sandbox.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

## 判断当前是否是临时沙箱环境

```
public static boolean isSandboxEnv(OlapConnection olapConn){
    return olapConn.isSandboxEnv();
}
```

## 持久化沙箱

允许基于一个已存在的Cube，创建一个永久的Cube环境，称为“持久化沙箱”，在这个环境中：

- 1. 支持查询、保存、计算命令
- 2. 基于同一个原始Cube，可以同时创建多个持久化沙箱，最多支持1000个
- 3. 持久化沙箱不支持元数据命令
- 4. 持久化沙箱不支持嵌套沙箱

内部基于多个度量值实现，故创建沙箱的命令实际内部原理如下：

- 1. 将原始度量值数据格式类型进行修改，因为原有的度量值不支持Undefined类型存储，不能作为持久化沙箱的度量组件。
- 2. 创建两个以沙箱名称拼接的度量值，一个为稀疏类型（ sandboxName-measureName ），一个为差量类型（ sandboxName\_measureName ）。

行号	沙箱		cube
	差量度量值	稀疏度量值	m1
1	指针		100
2	指针		200
3	指针	1	
4	指针	2	
5	指针	3	
6	指针	4	
7	指针	5	
8	指针	6	
9	指针		300
10	指针	7	400
11	指针	8	500
12	指针	9	

对于持久化沙箱而言，优先读取稀疏度量值的数据，如果数据不存在，则读取cube中度量值的数据。

### 创建持久化沙箱

```
/**
 * 创建沙箱
 *
 * @param olapConn 数据库链接
 * @param sandboxName 沙箱名称
 * @param measureName 度量值名称（基于cube哪个度量值进行沙箱创建）
 */
public static void createSandbox(OlapConnection olapConn, String sandboxName, String
measureName) {
    new PersistentSandboxBuilder(sandboxName, olapConn).createSandbox(measureName);
}
```

```

}

public static void main(String[] args) {
    String cubeName = "testCube";
    OlapConnection connection = OlapConnectionUtil.getConnection(cubeName);
    createSandbox(connection, "sandbox", "m1");
    connection.close();
    System.out.println("创建沙箱成功!");
}

```

假设原有的cube 存在多个度量值，可以通过如下方式进行沙箱创建：

```

/**
 * 创建沙箱
 *
 * @param olapConn    数据库链接
 * @param sandboxName 沙箱名称
 * @param measureNames 度量值名称（基于cube哪几个度量值进行沙箱创建）：如果cube 存在多度量值 m1,m2,可以设置多个
 */
public static void createSandbox(OlapConnection olapConn, String sandboxName, String...
measureNames) {
    new PersistentSandboxBuilder(sandboxName, olapConn).createSandbox(measureNames);
}

public static void main(String[] args) {
    String cubeName = "testCube";
    String sandbox = "sandbox";
    OlapConnection connection = OlapConnectionUtil.getConnection(cubeName);
    createSandbox(connection, sandbox, "m1", "m2");
    connection.close();
    System.out.println("创建沙箱成功!");
}

```

## 删除持久化沙箱

```

/**
 * 删除沙箱
 *
 * @param olapConn    数据库链接
 * @param sandboxName 沙箱名称
 */
public static void dropSandbox(OlapConnection olapConn, String sandboxName) {
    new PersistentSandboxBuilder(sandboxName, olapConn).dropSandbox();
}

public static void main(String[] args) {
    String cubeName = "testCube";
    OlapConnection connection = OlapConnectionUtil.getConnection(cubeName);
    dropSandbox(connection, "sandbox");
    connection.close();
    System.out.println("删除沙箱成功!");
}

```

```
}
```

## 操作沙箱

### 1. 创建操作沙箱的数据库连接

与之前获取连接区别：连接字符串 `OlapConnectionStringBuilder` 对象上需要设置 `pSandboxName` 属性值

```
private static OlapConnection getSandboxConnection(String cubeName, String sandboxName) {
    OlapConnectionStringBuilder strBuilder = new OlapConnectionStringBuilder();
    strBuilder.setProvider("kingdee.olap.Shrek");
    // 本用例连接为本地127.0.0.1的服务器，本IP根据实际情况修改
    strBuilder.setDataSource("http://127.0.0.1:8080/bos-olap-webserver/services/httpolap");
    // 设置即将连接或创建的Cube
    strBuilder.setInitialCatalog(cubeName);
    // 设置持久化沙箱名称
    strBuilder.setPersistentSandboxName(sandboxName);
    OlapConnection olapConn = new OlapConnection(strBuilder.toString());
    // 设置用户名及密码，可直接向上级申请
    olapConn.setUserName("admin");
    olapConn.setPassword("Olap@2018");
    return olapConn;
}
```

### 2. 发送查询、计算、保存命令

与之前的命令无任何变化

持久化沙箱保存数据：

```
private static void saveData(OlapConnection olapConn) {
    SaveCommandInfo saveCommandInfo = new SaveCommandInfo();
    saveCommandInfo.setDimensions("year", "period", "entity", "scenario", "account");
    // 此处操作的还是原始度量值
    saveCommandInfo.setMeasures("m1");
    OlapCommand cmd = new OlapCommand(olapConn, saveCommandInfo);
    OlapDataWriter writer = cmd.CreateWriter();

    Object[] newRow = new Object[]{10.1, "2018", "1", "a3", "MRPT", "b3"};
    writer.setValues(newRow);
    newRow = new Object[]{60, "2018", "2", "a3", "MRPT", "b3"};
    writer.setValues(newRow);
    writer.flush();
}

public static void main(String[] args) {
    String cubeName = "testCube";
    String sandbox = "sandbox";
    OlapConnection sandboxConnection = getSandboxConnection(cubeName, sandbox);
    // 执行保存命令
    saveData(sandboxConnection);
    sandboxConnection.close();
}
```

```
        System.out.println("数据保存成功!");
    }
}
```

持久化沙箱计算数据：

```
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");
    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();
}

public static void main(String[] args) {
    String cubeName = "testCube";
    String sandbox = "sandbox";
    OlapConnection sandboxConnection = getSandboxConnection(cubeName, sandbox);
    // 执行计算命令
    compute(sandboxConnection);
    sandboxConnection.close();
    System.out.println("数据计算成功!");
}
```

持久化沙箱查询数据：

```
public static void main(String[] args) {
    String cubeName = "testCube";
    String sandbox = "sandbox";
    OlapConnection sandboxConnection = getSandboxConnection(cubeName, sandbox);
    findData(sandboxConnection);
}

private static void findData(OlapConnection olapConn) {
    SelectCommandInfo selectInfo = new SelectCommandInfo();
    selectInfo.addDims("year", "period", "entity", "scenario", "account")
        .addMeasures("m1")
        .addFilter("year", "2018")
        .addFilter("period", "1", "2", "3")
        .addFilter("entity", "a3")
        .addFilter("scenario", "MRPT")
        .addFilter("account", "b3");
    OlapCommand queryCmd = new OlapCommand(olapConn, selectInfo);
}
```

```
CellSet cellSet = queryCmd.executeCellSet(StringMetadataBuilder.INSTANCE);
System.out.println("-----");
System.out.println("CellSet:");
System.out.println(cellSet); //将结果集打印到控制台
System.out.println("-----");
}
```

- 输出信息：

```
-----
CellSet:
[2018, 3, a3, MRPT, b3, 70.1]
[2018, 1, a3, MRPT, b3, 10.1]
[2018, 2, a3, MRPT, b3, 60]
-----
```

### 3. 断开连接

在使用完毕后，您可以通过调用 `OlapConnection.Close()` 方法断开与shrek服务器的连接

上一篇：[动态计算](#)

下一篇：[cube 分区](#)