系统表介绍

1 ClassTableItem

为一个类表表项，其包含属性如下：

```
public class ClassTableItem implements Serializable {
    public String classname = "";       // 类名
    public int classid = 0;             // 类id
    public int attrnum = 0;             // 类属性总个数
    public int attrid = 0;              // 当前属性id
    public String attrname = "";        // 当前属性名
    public String attrtype = "";        // 当前属性类型
    public String classtype = "";       // ori为源类的属性
    public String alias = "";
}
```

当我们创建一个类时，该类存在多少个属性，数据库就会对应地为其生成多少个ClassTableItem。例如，我们在系统中使用如下语句新建一个test类：

```
CREATE CLASS test (id int,value1 int,value2 int);
```

相应地，系统会生成如下三个ClassTableItem：

```
tmdb> show ClassTable
|class name          |class id           |attribute name      |attribute id        |attribute type      |
|test                |1                  |id                  |0                   |int                 |
|test                |1                  |value1              |1                   |int                 |
|test                |1                  |value2              |2                   |int                 |
```

2 DeputyTableItem

记录**源类和代理类之间关系**的系统表表项（可理解为双向指针列表），用于指明类（对应一个class）与类之间的代理关系，其包含属性如下：

```
public class DeputyTableItem implements Serializable {
    public int originid = 0;            // 源类id（从哪里代理而来）
    public int deputyid = 0;            // 代理类id
    public String[] deputyrule = new String[0];    // 代理规则
}
```

当我们使用代理类创建语句为一个已经存在的类创建代理时，DeputyTableItem会帮助我们记录源类和代理类之间的对应关系。例如，我们在系统中使用如下语句为test类创建一个代理时：

```
CREATE SELECTDEPUTY deputyTest1 AS (SELECT id, value1, value2 FROM test);
```

系统会相应地生成一个DeputyTableItem来记录源类和代理类之间的对应关系：

```
tmdb> show DeputyTable
|origin class id      |deputy class id      |
|1                    |2                    |
```

这里源类为test，其class id为1；代理类为deputyTest1，其class id为2。

3 SwitchingTableItem

记录**源类属性和代理类属性之间关系**的系统表表项（可理解为双向指针列表），用于指明属性（对应一个ClassTableItem的表项）与属性之间的代理关系，其包含属性如下：

```
public class SwitchingTableItem implements Serializable {
    public int oriId;           // 源类id
    public int oriAttrid;       // 源类属性id
    public String oriAttr;      // 源类属性名称
    public int deputyId;        // 代理类id
    public int deputyAttrId;    // 代理类属性id
    public String deputyAttr;   // 代理类属性名称
    public String rule = "";    // 代理规则
}
```

当我们创建代理类时也会同时更新并创建SwitchingTableItem。例如，针对上面例子，系统生成结果如下：

```
tmdb> show SwitchingTable
|origin class id     |origin attribute id |origin attribute name|deputy class id     |deputy attribute id |deputy attribute name|
|1                   |0                   |id                   |2                   |0                   |id                   |
|1                   |1                   |value1               |2                   |1                   |value1               |
|1                   |2                   |value2               |2                   |2                   |value2               |
```

4 ObjectTableItem

存储对象的索引列表表项，该表记录类的id和类所包含对象的id，表中属性如下：

```
public class ObjectTableItem implements Serializable{
    public int classid = 0;    //类id
    public int tupleid = 0;    //元组id
}
```

5 BiPointerTableItem

记录**源类对象和代理类对象之间关系**的系统表表项（可理解为双向指针列表），用于指明对象（对应一个object）与对象之间的代理关系，其包含属性如下：

```
public class BiPointerTableItem implements Serializable {
    public int classid = 0;          // 源类id
    public int objectid = 0;         // 源类对象id
    public int deputyid = 0;         // 代理类id
    public int deputyobjectid = 0;   // 代理类对象id
}
```

针对上面样例，在test类（class id为1）中插入4个对象（对应的object id为0~3）后，其代理类deputyTest1（class id为2）也会包含四个对象（对应的object id为4~7）。因此，BiPointerTableI所含表项如下：

```
tmdb> show BiPointerTable
|class id          |object id         |deputy id         |deputy object id   |
|1                 |0                 |2                 |4                  |
|1                 |1                 |2                 |5                  |
|1                 |2                 |2                 |6                  |
|1                 |3                 |2                 |7                  |
```

6 表间关系示意图