### 1、ER分片关系简介

有一类业务，例如订单（ORDER）跟订单明细表（ORDER\_DETAIL）,**明细表**会依赖于**订单表**，就是会存在表的主从关系，类似这种业务的切分可以抽象出合适的切分规则。

比如**根据用户ID切分**,其它相关的表都依赖于用户ID，再或者**根据订单ID进行切分**，总之部分业务总会可以抽象出父子关系的表。**这类表适用于ER分片表，子表的记录与所关联的父表记录存放在同一个数据分片上，避免数据Join跨库操作**。

以order与order\_detail例子为例，schema.xml中定义合适的分片配置,order,order\_detail 根据order\_id进行数据切分，保证相同order\_id的数据分到同一个分片上，在进行数据插入操作时，Mycat会获取order所在的分片，然后将order\_detail也插入到order所在的分片。

### 2、父表按照主键ID分片，子表的分片字段与主表ID关联，配置为ＥＲ分片

#### 2.1 在schema.xml添加如下配置配置文件修改

|  |
| --- |
| **<code class="hljs xml">**  **<childtable joinkey="ORDER\_ID" name="ORDER\_DETAIL" parentkey="ID" primarykey="ID"> </childtable>**  **</code>** |

在rule.xml里面设定分片规则

|  |
| --- |
| **<tableRule name="mod-long">**  **<rule>**  **<columns>id</columns>**  **mod-long</algorithm>**  **</rule>**  **</tableRule>** |

然后重启mycat或者重新加载mycat

#### 2.2 先建表， ORDER 和 ORDER\_DETAIL 表，有主外键关系

|  |
| --- |
| **mysql> CREATE TABLE ORDER1(ID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,SN VARCHAR(64),CREATE\_TIME DATETIME);**  **Query OK, 0 rows affected (0.06 sec)**    **mysql> CREATE TABLE ORDER\_DETAIL(ID INT AUTO\_INCREMENT PRIMARY KEY, ORDER\_ID INT,ORD\_STATUS CHAR(1),ADDRESS VARCHAR(128),CREATE\_TIME DATETIME,CONSTRAINT FK\_ORDID FOREIGN KEY (ORDER\_ID) REFERENCES ORDER1 (ID));**  **Query OK, 0 rows affected (0.02 sec)** |

### 3、录入数据：

录入数据，一组组录入，涉及到外键关系：  
第一组北京的订单

|  |
| --- |
| **mysql> INSERT INTO ORDER1(ID,SN,CREATE\_TIME) VALUES(1,'BJ0001',NOW());**  **mysql>INSERT INTO ORDER\_DETAIL(ID,ORDER\_ID,ORD\_STATUS,ADDRESS,CREATE\_TIME) VALUES (1,1,'1','test data  of ORDER1(ID=1,BJ001) ',NOW());** |

第二组上海的订单：

|  |
| --- |
| **mysql> INSERT INTO ORDER1(ID,SN,CREATE\_TIME) VALUES(3,'SHH001',NOW());**  **mysql> INSERT INTO ORDER\_DETAIL(ID,ORDER\_ID,ORD\_STATUS,ADDRESS,CREATE\_TIME) VALUES (3,3,'1','test data  of ORDER1(ID=3,SHH001) ',NOW());** |

第三组广州的订单：

|  |
| --- |
| **mysql> INSERT INTO ORDER1(ID,SN,CREATE\_TIME) VALUES(4,'GZH004',NOW());**  **mysql>INSERT INTO ORDER\_DETAIL(ID,ORDER\_ID,ORD\_STATUS,ADDRESS,CREATE\_TIME) VALUES (4,4,'1','test data  of ORDER1(ID=4,GZH004) ',NOW());** |

第四组 武汉的订单，这里故意将order\_id设置成4，看看效果，是否随id为4的广州的那组分片：

|  |
| --- |
| **mysql> INSERT INTO ORDER1(ID,SN,CREATE\_TIME) VALUES(6,'WUHAN006',NOW());**  **mysql>INSERT INTO ORDER\_DETAIL(ID,ORDER\_ID,ORD\_STATUS,ADDRESS,CREATE\_TIME) VALUES (6,4,'1','test data  of ORDER1(ID=6,WUHAN005) ',NOW());** |

### 4、验证子表是否随父表分片

#### 果然验证如下，大家看到订单详细表ORDER\_DETAIL的数据的分片都是随ORDER\_ID所在外键的父表的分片：

|  |
| --- |
| **[root@test\_1\_11logs]#/usr/local/mysql56s1/bin/mysql -uroot -p -P3317 --socket=/usr/local/mysql56s1/mysql.sock -e "select @@port;select \* from db3.ORDER1; select \* from db3.ORDER\_DETAIL";**  +--------+      | @@port |      +--------+      |   3327 |      +--------+      +----+--------+---------------------+      | ID | SN     | CREATE\_TIME         |      +----+--------+---------------------+      |  1 | BJ0001 | 2016-02-11 22:54:26 |      |  4 | GZH004 | 2016-02-11 22:57:49 |      +----+--------+---------------------+      +----+----------+------------+--------------------------------------+---------------------+      | ID | ORDER\_ID | ORD\_STATUS | ADDRESS                              | CREATE\_TIME         |      +----+----------+------------+--------------------------------------+---------------------+      |  1 |        1 | 1          | test data  of ORDER1(ID=1,BJ001)     | 2016-02-11 23:07:05 |      |  4 |        4 | 1          | test data  of ORDER1(ID=4,GZH004)    | 2016-02-11 23:09:48 |      |  6 |        4 | 1          | test data  of ORDER1(ID=6,WUHAN005)  | 2016-02-11 23:10:18 |      +----+----------+------------+--------------------------------------+---------------------+ |

### 父表的分片字段为其他字段，以PROVINCE字段分片，字表的分片字段与主表ID关联，配置为ＥＲ分片

#### 3.1 各种配置

在schema.xml里面添加表配置

**<childtable joinkey="ORDER\_ID" name="ORDER\_DETAIL2" parentkey="ID" primarykey="ID"> </childtable>**

|  |
| --- |
| **<tableRule name="province-str">**  **<rule>**  **<columns>PROVINCE</columns>**  **<algorithm>province-str-split</algorithm>**  **</rule>**  **</tableRule>**    **<function class="org.opencloudb.route.function.PartitionByFileMap" name="province-str-split">**  **<property name="type">1</property>**  **<property name="mapFile">partition-hash-str-pro.txt</property>**  **<property name="defaultNode">0</property>**  **</function>** |

新添加partition-hash-str-pro.txt

|  |
| --- |
| **beijing=0**  **shanghai=1**  **tianjing=2** |

#### 建表:

|  |
| --- |
| **mysql>CREATE TABLE ORDER2(ID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,SN VARCHAR(64),CREATE\_TIME DATETIME);**  **mysql>CREATE TABLE ORDER\_DETAIL2(ID INT AUTO\_INCREMENT PRIMARY KEY, ORDER\_ID INT,ORD\_STATUS CHAR(1),ADDRESS VARCHAR(128),CREATE\_TIME DATETIME,CONSTRAINT FK\_ORDID21 FOREIGN KEY (ORDER\_ID) REFERENCES ORDER2 (ID));** |

#### 录入数据，一组组录入，涉及到外键关系：

|  |
| --- |
| **mysql> INSERT INTO ORDER2(ID,PROVINCE,SN,CREATE\_TIME) VALUES(1,'beijing','2BJ0001',NOW());**  **mysql> INSERT INTOORDER\_DETAIL2(ID,ORDER\_ID,ORD\_STATUS,ADDRESS,CREATE\_TIME)**  **VALUES (1,1,'1','test data  of ORDER1(ID=1,2BJ0001) ',NOW());**    **mysql> INSERT INTO ORDER2(ID,PROVINCE,SN,CREATE\_TIME) VALUES(3,'shanghai','2SHH001',NOW());**  **mysql> INSERT INTO ORDER\_DETAIL2(ID,ORDER\_ID,ORD\_STATUS,ADDRESS,CREATE\_TIME) VALUES (3,3,'1','test data  of ORDER1(ID=3,2SHH001) ',NOW());**    **mysql>INSERT INTO ORDER2(ID,PROVINCE,SN,CREATE\_TIME) VALUES(4,'beijing','2GZH004',NOW());**  **mysql>INSERT INTO ORDER\_DETAIL2(ID,ORDER\_ID,ORD\_STATUS,ADDRESS,CREATE\_TIME) VALUES (4,4,'1','test data  of ORDER1(ID=4,2GZH004) ',NOW());**    **mysql> INSERT INTO ORDER2(ID,PROVINCE,SN,CREATE\_TIME) VALUES(6,'shanghai','2WUHAN006',NOW());**  **mysql> INSERT INTO ORDER\_DETAIL2(ID,ORDER\_ID,ORD\_STATUS,ADDRESS,CREATE\_TIME) VALUES (6,4,'1','test data  of ORDER1(ID=6,2WUHAN006) ',NOW());** |

验证：

|  |
| --- |
| **[root@test\_1\_11~]# /usr/local/mysql56s1/bin/mysql -uroot -p --socket=/usr/local/mysql56m1/mysql.sock -e "select @@port;select \* from db3.ORDER2; select \* from db3.ORDER\_DETAIL2";**  **Enter password:**  **+--------+**  **| @@port |**  **+--------+**  **|   3317 |**  **+--------+**  **+----+----------+---------+---------------------+**  **| ID | PROVINCE | SN      | CREATE\_TIME         |**  **+----+----------+---------+---------------------+**  **|  1 | beijing  | 2BJ0001 | 2016-02-12 17:23:46 |**  **|  4 | beijing  | 2GZH004 | 2016-02-12 17:24:07 |**  **+----+----------+---------+---------------------+**  **+----+----------+------------+---------------------------------------+---------------------+**  **| ID | ORDER\_ID | ORD\_STATUS | ADDRESS                               | CREATE\_TIME         |**  **+----+----------+------------+---------------------------------------+---------------------+**  **|  1 |        1 | 1          | test data  of ORDER1(ID=1,2BJ0001)    | 2016-02-12 17:23:50 |**  **|  4 |        4 | 1          | test data  of ORDER1(ID=4,2GZH004)    | 2016-02-12 17:24:11 |**  **|  6 |        4 | 1          | test data  of ORDER1(ID=6,2WUHAN006)  | 2016-02-12 17:24:17 |**  **+----+----------+------------+---------------------------------------+---------------------+**  **[root@test\_1\_11~]#** |

#### 总结：

**1、当子表与父表的关联字段正好是父表的分片字段时，子表直接根据父表规则进行分片，在数据录入的时候子表直接放在父表的分片上面，在进行关联查询join的时候，走的是父表分片规则。**

**2、当子表与父表的关联字段不是父表的分片字段时，必须通过查找对应的父表记录来确认子表所在分片，如果找不到则会抛出错误，在join查询的时候，路由走的是所有分片节点。**