1.Phoenix介绍

- Apache Phoenix是一款可以通过sql操作hbase的工具，只不过这个对hbase做了更深层次的优化，使得hbase可以更加高效，底层大量用到协处理器
- HBase+Phoenix无法取代hadoop+hive，前者只有查询引擎，后者还有计算

2.基本入门操作

1.创建表

```
格式

create table [if not exists] 表名(
rowkey名称 数据类型 primary key,
列族.列名1 数据类型,
列族.列名2 数据类型,
列族.列名3 数据类型,
...
);
```

```
示例

create table order_dtl(
id varchar primary key,
C1.status varchar,
C1.money integer,
C1.pay_way integer,
C1.user_id varchar,
C1.operation varchar,
C1.category varchar
);
```

```
0: jdbc:phoenix:> create table order_dtl(
. . . . . . . . > id varchar primary key,
. . . . . . . . > C1.status varchar,
. . . . . . . . > C1.money integer,
. . . . . . . . > C1.pay_way integer,
. . . . . . . . > C1.user_id varchar,
. . . . . . . . > C1.operation varchar,
. . . . . . . . > C1.category varchar
. . . . . . . . > );
No rows affected (1.396 seconds)
0: jdbc:phoenix:> !table
+------------+--------------+-------------+---------------+
| TABLE_CAT  | TABLE_SCHEM  | TABLE_NAME  |  TABLE_TYPE   |
+------------+--------------+-------------+---------------+
|            | SYSTEM       | CATALOG     | SYSTEM TABLE  |
|            | SYSTEM       | FUNCTION    | SYSTEM TABLE  |
|            | SYSTEM       | LOG         | SYSTEM TABLE  |
|            | SYSTEM       | SEQUENCE    | SYSTEM TABLE  |
|            | SYSTEM       | STATS       | SYSTEM TABLE  |
|            |              | ORDER_DTL   | TABLE         |
+------------+--------------+-------------+---------------+
0: jdbc:phoenix:>
```

- 执行创建表完成后，hbase webui查看，多了一张表，该表region默认是1，有很多协处理器
- 注意：Phoenix会自动把小写转成大写，表名 列族 列名

```
create table "order_dtl1"(
"id" varchar primary key,
"c1"."status" varchar,
"c1"."money" integer,
"c1"."pay_way" integer,
C1.user_id varchar,
C1.operation varchar,
C1.category varchar
); -- 创建的表有两个列族
```

- 如果想要使用小写，需要在小写的内容两端加双引号（必须是双引号），单引号表示普通字符串
- 建议直接使用大写，如果使用了双引号的小写，后续所有操作都不需带双引号

2.查看所有表

```
!table
```

```
0: jdbc:phoenix:> !table
+------------+--------------+-------------+---------------+
| TABLE_CAT  | TABLE_SCHEM  | TABLE_NAME  |  TABLE_TYPE   |
+------------+--------------+-------------+---------------+
|            | SYSTEM       | CATALOG     | SYSTEM TABLE  |
|            | SYSTEM       | FUNCTION    | SYSTEM TABLE  |
|            | SYSTEM       | LOG         | SYSTEM TABLE  |
|            | SYSTEM       | SEQUENCE    | SYSTEM TABLE  |
|            | SYSTEM       | STATS       | SYSTEM TABLE  |
|            |              | ORDER_DTL   | TABLE         |
|            |              | order_dtl   | TABLE         |
|            |              | order_dtl1  | TABLE         |
+------------+--------------+-------------+---------------+
```

3.查看表结构

```
!desc 表名
```

```
0: jdbc:phoenix:> !desc order_dtl
+------------+--------------+-------------+--------------+------------+
| TABLE_CAT  | TABLE_SCHEM  | TABLE_NAME  | COLUMN_NAME  | DATA_TYPE  |
+------------+--------------+-------------+--------------+------------+
|            |              | ORDER_DTL   | ID           | 12         |
|            |              | ORDER_DTL   | STATUS       | 12         |
|            |              | ORDER_DTL   | MONEY        | 4          |
|            |              | ORDER_DTL   | PAY_WAY      | 4          |
|            |              | ORDER_DTL   | USER_ID      | 12         |
|            |              | ORDER_DTL   | OPERATION    | 12         |
|            |              | ORDER_DTL   | CATEGORY     | 12         |
+------------+--------------+-------------+--------------+------------+
0: jdbc:phoenix:> !desc "order_dtl1"
+------------+--------------+-------------+--------------+------------+
| TABLE_CAT  | TABLE_SCHEM  | TABLE_NAME  | COLUMN_NAME  | DATA_TYPE  |
+------------+--------------+-------------+--------------+------------+
|            |              | order_dtl1  | id           | 12         |
|            |              | order_dtl1  | status       | 12         |
|            |              | order_dtl1  | money        | 4          |
|            |              | order_dtl1  | pay_way      | 4          |
|            |              | order_dtl1  | USER_ID      | 12         |
|            |              | order_dtl1  | OPERATION    | 12         |
|            |              | order_dtl1  | CATEGORY     | 12         |
+------------+--------------+-------------+--------------+------------+
```

4.退出

```
!q
!quit
```

5.插入数据

- Phoenix中插入数据，插入和更新都是upsert

```
格式
upsert into 表名(列族.列名1, 列族.列名2, 列族.列名3, ...) values(值1,值2,值3,...)

示例
upsert into order_dtl values('000001', '已提交', 4070, 1 , '4944191', '2020-04-25 12:09:16', '手机');

upsert into order_dtl values('000001', '未提交', 4070, 1 , '4944191', '2020-04-25 12:09:16', '手机');


简写
upsert into order_dtl (id, status) values('000001', '未提交1');
upsert into order_dtl (id, C1.status) values('000001', '未提交2');

```

6.查询操作

```
select 语句
和sql一致
仅支持单表查询，不支持多表关联查询
```

7.删除数据

```
格式
delete from 表名 where

delete from order_dtl where id = '000001'; -- 这里一定是单引号
```

8.分页查询

```
格式
select * from 表名 limit 每页几条数据 offset 从第几条开始查询;

示例
select * from order_dtl limit 5 offset 10;
```

```
0: jdbc:phoenix:> select * from order_dtl limit 5 offset 10;
+---------+---------+--------+---------+---------+---------------------+-----------+
|   ID    | STATUS  | MONEY  | PAY_WAY | USER_ID |      OPERATION      | CATEGORY  |
+---------+---------+--------+---------+---------+---------------------+-----------+
| 000012  | 已提交     | 8340   | 2       | 2948003 | 2020-04-25 12:09:26 | 男装;男鞋;    |
| 000013  | 已付款     | 8340   | 2       | 2948003 | 2020-04-25 12:09:30 | 男装;男鞋;    |
| 000014  | 已提交     | 7060   | 2       | 2092774 | 2020-04-25 12:09:38 | 酒店;旅游;    |
| 000015  | 已提交     | 640    | 3       | 7152356 | 2020-04-25 12:09:49 | 维修;手机;    |
| 000016  | 已付款     | 9410   | 3       | 7152356 | 2020-04-25 12:10:01 | 维修;手机;    |
+---------+---------+--------+---------+---------+---------------------+-----------+
5 rows selected (0.228 seconds)
0: jdbc:phoenix:> select * from order_dtl limit 5 offset 0;
+---------+---------+--------+---------+---------+---------------------+-------------+
|   ID    | STATUS  | MONEY  | PAY_WAY | USER_ID |      OPERATION      |  CATEGORY   |
+---------+---------+--------+---------+---------+---------------------+-------------+
| 000002  | 已提交     | 4070   | 1       | 4944191 | 2020-04-25 12:09:16 | 手机;         |
| 000003  | 已完成     | 4350   | 1       | 1625615 | 2020-04-25 12:09:37 | 家用电器;;电脑;  |
| 000004  | 已提交     | 6370   | 3       | 3919700 | 2020-04-25 12:09:39 | 男装;男鞋;      |
| 000005  | 已付款     | 6370   | 3       | 3919700 | 2020-04-25 12:09:44 | 男装;男鞋;      |
| 000006  | 已提交     | 9380   | 1       | 2993700 | 2020-04-25 12:09:41 | 维修;手机;      |
+---------+---------+--------+---------+---------+---------------------+-------------+
5 rows selected (0.158 seconds)
0: jdbc:phoenix:>
```

3.预分区操作

- 通过Phoenix创建的表默认也是一个region
- Phoenix设置预分区
  - **手动设置**

```
格式

create table [if not exists] 表名(
rowkey名称 数据类型 primary key,
列族.列名1 数据类型,
列族.列名2 数据类型,
列族.列名3 数据类型,
...
)
compression='GZ' --指定GZ的压缩方式
split on (分区边界值) --指定分区边界
;

示例
drop table if exists order_dtl;

create table order_dtl(
id varchar primary key,
C1.status varchar,
C1.money integer,
```

```
C1.pay_way integer,
C1.user_id varchar,
C1.operation varchar,
C1.category varchar
)
compression='GZ'
split on('3', '5', '7')
;

把 测试预分区数据.txt 数据插入到表中，观察每个分区数据量
如果手动设定分区边界，有可能每个分区内数据不均衡
```

Table Regions

Sort As RegionName Ascending ShowDetailName&Start/End Key Reorder

Name(4)	Region Server	ReadRequests (0)	WriteRequests (97)	StorefileSize (0 B)	Num.Storefiles (0)	MemSize (0 B)	Locality	Start Key	End Key
ORDER_DTL,,1646967359079.1b94ff7594369b903f145a3544d7117a.	node3:16030	0	23	0 B	0	0 B	0.0		3
ORDER_DTL,3,1646967359079.c49a8e30ea9fa24bef3503acb99320c1.	node1:16030	0	10	0 B	0	0 B	0.0	3	5
ORDER_DTL,5,1646967359079.335bca0fed6deb31fba397e8f80982fb.	node2:16030	0	10	0 B	0	0 B	0.0	5	7
ORDER_DTL,7,1646967359079.8041302f75ea7025b518126f11081f78.	node3:16030	0	54	0 B	0	0 B	0.0	7	

- **hash预分区**

```
格式

create table [if not exists] 表名(
rowkey名称 数据类型 primary key,
列族.列名1 数据类型,
列族.列名2 数据类型,
列族.列名3 数据类型,
...
)
compression='GZ', --指定GZ的压缩方式
SALT_BUCKETS=N --指定hash分区数量，加盐预分区
;

示例
drop table if exists order_dtl;

create table order_dtl(
id varchar primary key,
C1.status varchar,
C1.money integer,
C1.pay_way integer,
```

```
C1.user_id varchar,
C1.operation varchar,
C1.category varchar
)
compression='GZ',
SALT_BUCKETS=10
;

把 测试预分区数据.txt 数据插入到表中，观察每个分区数据量
使用hash加盐预分区数据会更加均衡一些
```

Table Regions

Sort As RegionName Ascending ShowDetailName&Start/End Key Reorder

Name(10)	Region Server	ReadRequests (0)	WriteRequests (97)	StorefileSize (0 B)	Num.Storefiles (0)	MemSize (0 B)	Locality	Start Key	End Key
ORDER_DTL,,1646967722508.770bb21e4587cdd48dc35c88402aa50e.	node3:16030	0	15	0 B	0	0 B	0.0		\x01
ORDER_DTL,\x01,1646967722508.e53c24e8e7611aa54483994e1a12d78b.	node3:16030	0	19	0 B	0	0 B	0.0	\x01	\x02
ORDER_DTL,\x02,1646967722508.b5fb8e7e463395ca35b5b223aa3b183c.	node2:16030	0	7	0 B	0	0 B	0.0	\x02	\x03
ORDER_DTL,\x03,1646967722508.bf7a08a584070d4e6aa3f4abed75d687.	node1:16030	0	8	0 B	0	0 B	0.0	\x03	\x04
ORDER_DTL,\x04,1646967722508.63e232d1432e993b18d760ea17e0c0dd.	node1:16030	0	12	0 B	0	0 B	0.0	\x04	\x05
ORDER_DTL,\x05,1646967722508.24b00b7ce9cdc0befdeb6bceb93468b8.	node3:16030	0	8	0 B	0	0 B	0.0	\x05	\x06
ORDER_DTL,\x06,1646967722508.abfffaaffb52aa99d364aa8724f0327e.	node2:16030	0	4	0 B	0	0 B	0.0	\x06	\x07
ORDER_DTL,\x07,1646967722508.dae25927fba95f27e7665ef5cbb16ad4.	node2:16030	0	12	0 B	0	0 B	0.0	\x07	\x08
ORDER_DTL,\x08,1646967722508.8898eac36dd89804aa6071e8b0a3383f.	node3:16030	0	3	0 B	0	0 B	0.0	\x08	\x09
ORDER_DTL,\x09,1646967722508.c0e000422c93a7de8d68d74ced3e1dbd.	node1:16030	0	9	0 B	0	0 B	0.0	\x09	

- 如果使用了hash加盐预分区，会自动在rowkey前面进行加盐处理，用户在Phoenix中操作是无感，但是回到hbase中无法查询看到原始数据，无法通过hbase的原生Java api 操作数据。此时只能通过jdbc连接到Phoenix使用sql语句进行操作

4.视图

- 默认情况Phoenix中只能看到自己创建的表，如果表是在hbase中创建的，此时Phoenix中无法直接查看。也就是说Phoenix无法操作hbase的原始表
- 希望通过Phoenix操作hbase原始表，需要使用Phoenix视图

```
格式
create view "名称空间"."hbsase中对应的表名" (    --表名必须是hbase中的表，不能随便起名
key varchar primary key, -- rowkey
"列族"."列名1" 类型,
"列族"."列名2" 类型,
"列族"."列名3" 类型,
...
)[default_columns_family="列族名"]; 如果不在后面设置列族名，就在前面设定

视图名称必须和hbase中表名一致
```

```
key名称任意，必须加primary key
普通列名必须和hbase中列名保持一致
```

- 案例，针对water_bill创建一个视图

```
hbase(main):020:0> scan "WATER_BILL", {LIMIT=>1, FORMATTER=>"toString"}
ROW                                   COLUMN+CELL
 0000132                              column=C1:ADDRESS, timestamp=1588911408895, value=山西省忻州市偏关县新关镇7单元124室

 0000132                              column=C1:LATEST_DATE, timestamp=1588911408895, value=2020-08-02
 0000132                              column=C1:NAME, timestamp=1588911408895, value=方浩轩

 0000132                              column=C1:NUM_CURRENT, timestamp=1588911408895, value=@p@3333
 0000132                              column=C1:NUM_PREVIOUS, timestamp=1588911408895, value=@}@3333
 0000132                              column=C1:NUM_USAGE, timestamp=1588911408895, value=@5
 0000132                              column=C1:PAY_DATE, timestamp=1588911408895, value=2020-09-09
 0000132                              column=C1:RECORD_DATE, timestamp=1588911408895, value=2020-10-07
 0000132                              column=C1:SEX, timestamp=1588911408895, value=女

 0000132                              column=C1:TOTAL_MONEY, timestamp=1588911408895, value=@\@
1 row(s)
Took 0.0359 seconds
hbase(main):021:0>
```

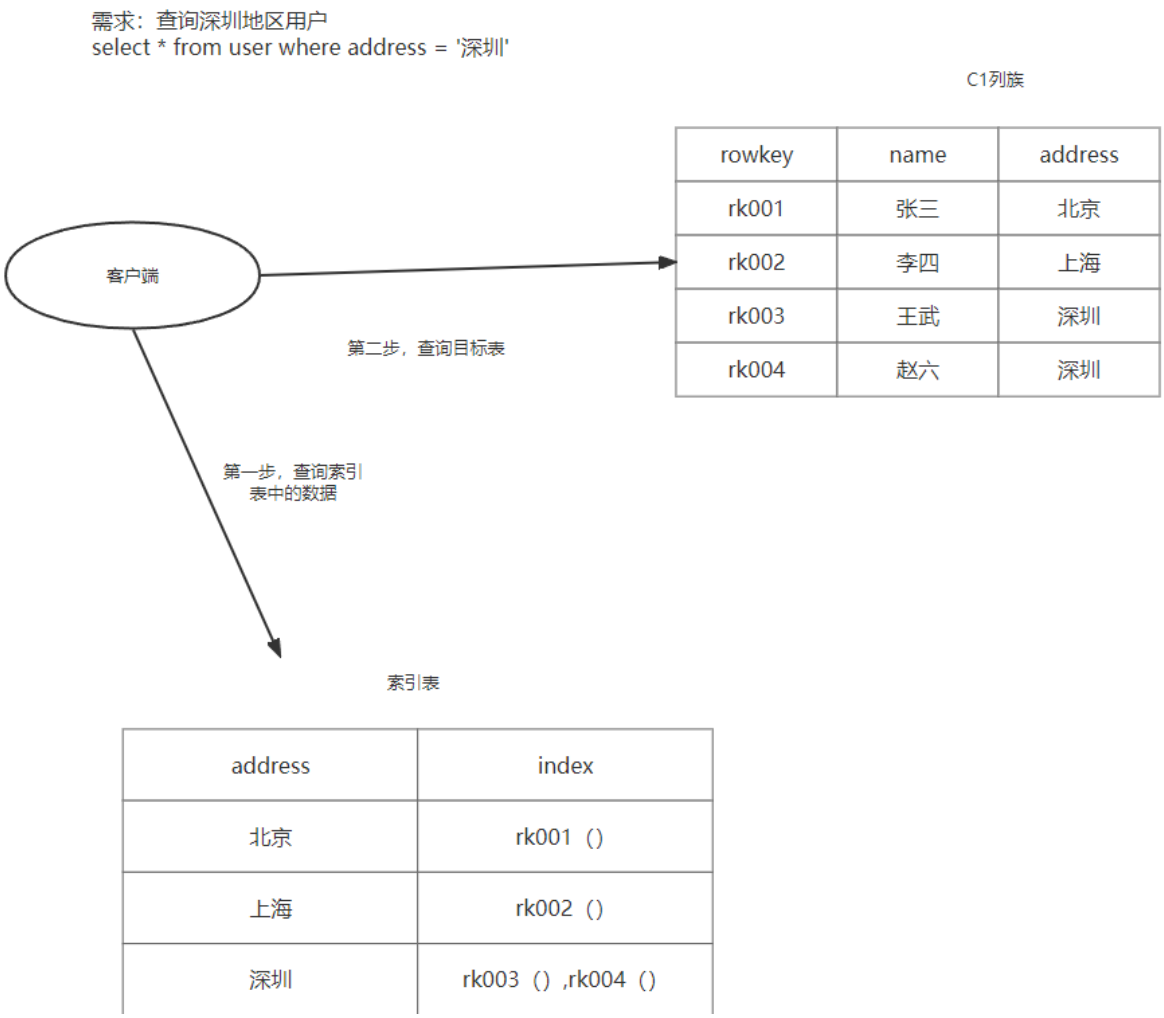
```
create view "WATER_BILL" (
ID varchar primary key,
C1.ADDRESS varchar,
C1.LATEST_DATE varchar,
C1.NAME varchar,
C1.NUM_CURRENT UNSIGNED_DOUBLE,
C1.NUM_PREVIOUS UNSIGNED_DOUBLE,
C1.NUM_USAGE UNSIGNED_DOUBLE,
C1.PAY_DATE varchar,
C1.RECORD_DATE varchar,
C1.SEX varchar,
C1.TOTAL_MONEY UNSIGNED_DOUBLE
);

UNSIGNED_DOUBLE 无符号双精度浮点类型
```

- 现在通过创建视图，关联到hbase原始的water_bill表，接下进行sql操作

```
查询2020年6月的用户用水的数据
select name, num_usage from water_bill where record_date between '2020-06-01' and '2020-06-30';
```

5.二级索引

- 建立索引目的：提供高效的查询操作（空间换时间）



1.索引的分类

- Phoenix可以创建四种索引
  - ## 全局索引
    - 适合于读多写少
    - 特点

```
1.构建全局索引后会生成一张单独的索引表，查询数据时，先到索引表进行查询，根据结果再到目标表进行查询
2.如果目标表进行修改，则全局索引表也要进行修改
3.修改索引表时需要对全局的索引都进行更新操作，代价较大，效率较低
4.在查询过程中，如果sql语句中使用了非索引字段，全局索引无法生效
5.单独构建的索引表和目标表有相同的region数量
```

    - 如何创建

```
格式

create index 索引名称 on 表名(列名1, 列名2, 列名3 ...);
```

    - 如何删除

```
格式

drop index 索引名称 on 表名;
```

- ## 本地索引
  - 适合于写多
  - 特点

```
1.本地索引不会单独创建索引表，索引数据直接附在目录表对应的字段后面，修改时直接在目标表对应的索引数据中进行一次性处理
2.执行查询操作时，Phoenix自动选择是否适用于本地索引，即使查询字段有非索引字段，本地索引仍然适用
3.如果构建表时采用的是加盐分区，则不要适用本地索引，因为一部分操作不支持
```

  - 创建本地索引

```
create local index 索引名称 on 表名(列名1, 列名2, 列名3 ...);
```

  - 如何删除

```
格式

drop index 索引名称 on 表名;
```

- ## 覆盖索引
  - 适用于一些不参与条件的字段，但是会参与展示
  - 特点

```
1.覆盖索引无法单独存在，必须和全局/本地索引配合使用
2.架构覆盖索引之后，对应的索引字段会直接放在全局/本地索引之后，再去查询时不用查询目标表，直接在索引表可以把需要的字段获取到
```

  - 如何创建

```
格式

create [local] index 索引名称 on 表名(列名1, 列名2, 列名3...) include (列名4, 列名5...)
```

  - 如何删除

```
覆盖索引无法单独删除，随着本地/全局索引一起删除
```

- ## 函数索引
  - 适用于有些函数在某些列上频繁被调用

- 热点

```
1. 不是针对某个函数去建立索引
def abc(a, b):
        return a+b

针对abc建立索引是没有意义
针对abc(10000, 10000)建立索引就是有意义的

2 针对某个函数在某些特定列上的结构构建索引，后续可以直接使用这个结果，提高效率

3 无法单独存在，依赖于全局/本地索引
```

- 如何创建

```
创建索引
create [local] index 索引名称 on 表名(函数名称(列名1, 列名2, 列名3...))
create index upper_name_idx on (emp(upper(first_name ||' '|| last_name))) -- 示例
upper_name_idx 全局索引名称
emp表名
upper 函数名
first_name 列名
last_name 列名

使用时
select emp_id from emp where (upper(first_name ||' '|| last_name)) = "JOHN DOE"

```

- 如何删除

```
函数索引无法单独删除，随着本地/全局索引一起删除
```

2.案例一：创建全局索引+覆盖索引

- 需求：查询用户id 8237476 订单数据，订单id 金额

```
explain select user_id, id, money from order_dtl where user_id = '8237476';
```

- full scan:扫描全表

```
0: jdbc:phoenix:> select user_id, id, money from order_dtl where user_id = '8237476';
+----------+--------------------------------------+--------+
| USER_ID  |                  ID                  | MONEY  |
+----------+--------------------------------------+--------+
| 8237476  | 2909b28a-5085-4f1d-b01e-a34fbaf6ce37 | 9390   |
+----------+--------------------------------------+--------+
1 row selected (0.062 seconds)
0: jdbc:phoenix:> explain select user_id, id, money from order_dtl where user_id = '8237476';
+-----------------------------------------------------------------------+-----------------+-----------------+----------------+
|                                 PLAN                                  | EST_BYTES_READ  | EST_ROWS_READ   | EST_INFO_TS    |
+-----------------------------------------------------------------------+-----------------+-----------------+----------------+
| CLIENT 10-CHUNK PARALLEL 10-WAY ROUND ROBIN FULL SCAN OVER ORDER_DTL  | null            | null            | null           |
|     SERVER FILTER BY C1.USER_ID = '8237476'                           | null            | null            | null           |
+-----------------------------------------------------------------------+-----------------+-----------------+----------------+
2 rows selected (0.031 seconds)
```

- 创建索引

```
1 create index global_index_order_dtl on order_dtl(user_id) include (id, money);
2
3 执行
4 explain select user_id, id, money from order_dtl where user_id = '8237476';
```

```
0: jdbc:phoenix:> create index global_index_order_dtl on order_dtl(user_id) include (id, money);
66 rows affected (7.408 seconds)
0: jdbc:phoenix:> explain select user_id, id, money from order_dtl where user_id = '8237476';
+-----------------------------------------------------------------------------------------------------------+-----------+
|                                                   PLAN                                                    | EST_BYTE  |
+-----------------------------------------------------------------------------------------------------------+-----------+
| CLIENT 10-CHUNK PARALLEL 10-WAY ROUND ROBIN RANGE SCAN OVER GLOBAL_INDEX_ORDER_DTL [0,'8237476'] - [9,'8237476'] | null      |
+-----------------------------------------------------------------------------------------------------------+-----------+
1 row selected (0.054 seconds)
0: jdbc:phoenix:>
```

- 如果查询语句中出现了非索引字段，则全局索引无法生效

```
1 STATUS没有构建索引的
2
3 执行
4 explain select user_id, id, money, status from order_dtl where user_id = '8237476';
```

- 如果查询语句中出现非索引字段，全局索引无法生效，但是可以让其强制生效

```
1 通过强制的方式使用全局索引 /*+index(表名 全局索引名)*/
2 explain select /*+index(order_dtl global_index_order_dtl)*/ user_id, id, money, status
  from order_dtl where user_id = '8237476';
```

```
0: jdbc:phoenix:> explain select /*+index(order_dtl global_index_order_dtl)*/ user_id, id, money, status from order_dtl where us
er_id = '8237476';
+-----------------------------------------------------------------------------------------------------------+---+
|                                                   PLAN                                                    |   |
+-----------------------------------------------------------------------------------------------------------+---+
| CLIENT 10-CHUNK PARALLEL 10-WAY ROUND ROBIN FULL SCAN OVER ORDER_DTL                                       |   |
|     SKIP-SCAN-JOIN TABLE 0                                                                                |   |
|         CLIENT 10-CHUNK PARALLEL 10-WAY ROUND ROBIN RANGE SCAN OVER GLOBAL_INDEX_ORDER_DTL [0,'8237476'] - [9,'8237476'] |   |
|             SERVER FILTER BY FIRST KEY ONLY                                                               |   |
|     DYNAMIC SERVER FILTER BY "ORDER_DTL.ID" IN ($22.$24)                                                  |   |
+-----------------------------------------------------------------------------------------------------------+---+
5 rows selected (0.092 seconds)
0: jdbc:phoenix:>
```

没有索引的字段，进行全表扫描

有索引的字段，从索引表中得到

- 删除前查看webui
- 全局索引表是一个单独表，它的分区数量和目标表一致

Namespace	Table Name	Online Regions	Offline Regions	Failed Regions	Split Regions	Other Regions	Description
default	GLOBAL_INDEX_ORDER_DTL	10	0	0	0	0	'GLOBAL_INDEX_ORDER_DTL', {TABLE_ATTRIBUTES => {PRIORITY => '1000', coprocessor$1 => '|org.apache.phoen '|org.apache.phoenix.coprocessor.UngroupedAggregateRegionObserver|805306366|', coprocessor$3 => '|org.apache.pho '|org.apache.phoenix.coprocessor.ServerCachingEndpointImpl|805306366|', METADATA => {'DATA_TABLE_NAME' => 'O BLOOMFILTER => 'NONE'}
default	ORDER_DTL	10	0	0	0	0	'ORDER_DTL', {TABLE_ATTRIBUTES => {coprocessor$1 => '|org.apache.phoenix.coprocessor.ScanRegionObserver|80 '|org.apache.phoenix.coprocessor.UngroupedAggregateRegionObserver|805306366|', coprocessor$3 => '|org.apache.pho '|org.apache.phoenix.coprocessor.ServerCachingEndpointImpl|805306366|', coprocessor$5 => '|org.apache.phoenix.hbase.index.Indexer|805306366|index.builder=org.apache.phoenix.index.PhoenixIndexBuilder,org.a {NAME => 'C1', DATA_BLOCK_ENCODING => 'FAST_DIFF', BLOOMFILTER => 'NONE', COMPRESSION => 'GZ'}

- 删除索引

```
drop index global_index_order_dtl on order_dtl;
```

3.案例二：本地索引

- 需求：根据user_id查看支付状态

```
创建本地索引
create local index local_index_order_dtl on order_dtl (id, status, money, pay_way, user_id);
```

```
hbase(main):022:0> scan "ORDER_DTL", {LIMIT=>1, FORMATTER=>"toString"}
ROW                                   COLUMN+CELL
 0f46d542-34cb-4ef4-b7fe-6dcfa5f column=C1:, timestamp=1646967737313, value=x
 14751
 0f46d542-34cb-4ef4-b7fe-6dcfa5f column=C1:
                                       , timestamp=1646967737313, value=已付款

 14751
 0f46d542-34cb-4ef4-b7fe-6dcfa5f column=C1:
                                       , timestamp=1646967737313, value=
 14751
, timestamp=1646967737313, value=
 14751
 0f46d542-34cb-4ef4-b7fe-6dcfa5f column=C1: timestamp=1646967737313, value=2993700
 14751
 0f46d542-34cb-4ef4-b7fe-6dcfa5f column=C1: timestamp=1646967737313, value=2020-04-25 12:09:46
 14751
 0f46d542-34cb-4ef4-b7fe-6dcfa5f column=C1: timestamp=1646967737313, value=维修;手机;

 14751
1 row(s)
Took 0.0313 seconds
hbase(main):023:0>
```

因为在创建order_dtl时，进行了hash加盐预分区，所以这里无法查看到详细信息

- 测试1 查询索引字段都有本地索引的

```
explain select user_id, id, money, status from order_dtl where user_id = '8237476';
```

```
0: jdbc:phoenix:> explain select user_id, id, money, status from order_dtl where user_id = '8237476';
+-----------------------------------------------------------------------------+-----------------+----------------+---------------+
|                                    PLAN                                     | EST_BYTES_READ  | EST_ROWS_READ  | EST_INFO_TS   |
+-----------------------------------------------------------------------------+-----------------+----------------+---------------+
| CLIENT 10-CHUNK PARALLEL 10-WAY ROUND ROBIN RANGE SCAN OVER ORDER_DTL [1]    | null            | null           | null          |
|     SERVER FILTER BY FIRST KEY ONLY AND "USER_ID" = '8237476'                | null            | null           | null          |
+-----------------------------------------------------------------------------+-----------------+----------------+---------------+
2 rows selected (0.043 seconds)
0: jdbc:phoenix:>
```

- 测试2 查询字段一部分有本地索引，一部分没有

```
explain select user_id, id, money, status, operation from order_dtl where user_id = '8237476';
```

```
0: jdbc:phoenix:> explain select user_id, id, money, status, operation from order_dtl where user_id = '8237476';
+-----------------------------------------------------------------------------+-----------------+----------------+---------------+
|                                    PLAN                                     | EST_BYTES_READ  | EST_ROWS_READ  | EST_INFO_TS   |
+-----------------------------------------------------------------------------+-----------------+----------------+---------------+
| CLIENT 10-CHUNK PARALLEL 10-WAY ROUND ROBIN RANGE SCAN OVER ORDER_DTL [1]    | null            | null           | null          |
|     SERVER FILTER BY FIRST KEY ONLY AND "USER_ID" = '8237476'                | null            | null           | null          |
+-----------------------------------------------------------------------------+-----------------+----------------+---------------+
2 rows selected (0.043 seconds)
0: jdbc:phoenix:>
```

- 如果有本地索引，一部分字段没有，本地索引仍然生效
- 测试3 查询所有字段

```
explain select * from order_dtl where user_id = '8237476';
```

```
0: jdbc:phoenix:> explain select * from order_dtl where user_id = '8237476';
+-----------------------------------------------------------------------+-----------------+-----------------+----------------+
|                                  PLAN                                 | EST_BYTES_READ  | EST_ROWS_READ   | EST_INFO_TS    |
+-----------------------------------------------------------------------+-----------------+-----------------+----------------+
| CLIENT 10-CHUNK PARALLEL 10-WAY ROUND ROBIN FULL SCAN OVER ORDER_DTL   | null            | null            | null           |
|     SERVER FILTER BY C1.USER_ID = '8237476'                           | null            | null            | null           |
+-----------------------------------------------------------------------+-----------------+-----------------+----------------+
2 rows selected (0.041 seconds)
0: jdbc:phoenix:>
```

- 测试4，range scan

```
explain select user_id, id, money, status, operation, CATEGORY from order_dtl where user_id = '8237476';
```

```
0: jdbc:phoenix:> explain select user_id, id, money, status, operation, CATEGORY from order_dtl where user_id = '8237476';
+-----------------------------------------------------------------------+-----------------+-----------------+----------------+
|                                  PLAN                                 | EST_BYTES_READ  | EST_ROWS_READ   | EST_INFO_TS    |
+-----------------------------------------------------------------------+-----------------+-----------------+----------------+
| CLIENT 10-CHUNK PARALLEL 10-WAY ROUND ROBIN RANGE SCAN OVER ORDER_DTL [1]  | null            | null            | null           |
|     SERVER FILTER BY FIRST KEY ONLY AND "USER_ID" = '8237476'          | null            | null            | null           |
+-----------------------------------------------------------------------+-----------------+-----------------+----------------+
2 rows selected (0.051 seconds)
0: jdbc:phoenix:>
```

- 删除索引之前，创建本地索引不会多一张表

```
hbase(main):022:0> scan "ORDER_DTL", {LIMIT=>1, FORMATTER=>"toString"}
ROW                                   COLUMN+CELL
 0f46d542-34cb-4ef4-b7fe-6dcfa5f      column=C1:, timestamp=1646967737313, value=x
 14751
 0f46d542-34cb-4ef4-b7fe-6dcfa5f      column=C1:
                                      , timestamp=1646967737313, value=已付款
 14751
 0f46d542-34cb-4ef4-b7fe-6dcfa5f      column=C1:
                                      , timestamp=1646967737313, value=
 14751
, timestamp=1646967737313, value=
 14751
 0f46d542-34cb-4ef4-b7fe-6dcfa5f      column=C1:, timestamp=1646967737313, value=2993700
 14751
 0f46d542-34cb-4ef4-b7fe-6dcfa5f      column=C1:, timestamp=1646967737313, value=2020-04-25 12:09:46
 14751
 0f46d542-34cb-4ef4-b7fe-6dcfa5f      column=C1:, timestamp=1646967737313, value=维修;手机;
 14751
1 row(s)
Took 0.0313 seconds
hbase(main):023:0> scan "ORDER_DTL", {LIMIT=>1, FORMATTER=>"toString"}
ROW                                   COLUMN+CELL
 0f46d542-34cb-4ef4-b7fe-6dcfa column=L#0:, timestamp=1646967737313, value=
 5f14751已付款Q2993700
1 row(s)
Took 0.0138 seconds
hbase(main):024:0>
```

进行本地索引的表，索引的内容直接写在目标表中，所以这里看起来是乱码

- 删除索引表

```
drop index local_index_order_dtl on order_dtl;
```

4.案例三：实现water_bill查询操作

- 需求：通过sql语句查询6月份用户用水量

```
explain select name, num_usage, record_date from water_bill where record_date between '2020-06-01' and '2020-06-30'
```

```
0: jdbc:phoenix:> explain select name, num_usage, record_date from water_bill where record_date between '2020-06-01' and '2020-06-30';
+-----------------------------------------------------------------------+-----------------+-----------------+
|                                  PLAN                                 | EST_BYTES_READ  | EST_ROWS_READ   |
+-----------------------------------------------------------------------+-----------------+-----------------+
| CLIENT 1-CHUNK PARALLEL 1-WAY ROUND ROBIN FULL SCAN OVER WATER_BILL    | null            | null            |
|     SERVER FILTER BY (C1.RECORD_DATE >= '2020-06-01' AND C1.RECORD_DATE <= '2020-06-30') | null            | null            |
+-----------------------------------------------------------------------+-----------------+-----------------+
2 rows selected (0.045 seconds)
```

- 优化

```
1 创建全局索引+覆盖索引
```

```
2 create index global_index_water_bill on water_bill (record_date) include (name,
  num_usage);
```

```
0: jdbc:phoenix:> explain select name, num_usage, record_date from water_bill where record_date between '2020-06-01' and '2020-0
6-30';
+-----------------------------------------------------------------------------------------------------------+----------+
|                                                   PLAN                                                    | EST_BYT  |
+-----------------------------------------------------------------------------------------------------------+----------+
| CLIENT 1-CHUNK PARALLEL 1-WAY ROUND ROBIN RANGE SCAN OVER GLOBAL_INDEX_WATER_BILL ['2020-06-01'] - ['2020-06-30']  | null     |
+-----------------------------------------------------------------------------------------------------------+----------+
1 row selected (0.038 seconds)
0: jdbc:phoenix:>
```

- # 索引小结
  - 1.思想：空间换时间
  - 2.前期不常用，后期经常用的字段，在使用前创建索引
  - 3.前期经常用，后期不常用的字段，使用后删除
  - 4.创建索引优点提升查询效率，缺点：数据冗余，占用磁盘空间