# Hbase初窥门径

## 需求调研

现在列表页全量搜索，根据商品分库表获取商品id，进行实时计算来把数据同步到ES中。由于没有每天需要更新商品质量分。更新数据量较大所以需要在计算服务和ES中间加一层中间存储层。

## 存储选择

mysql 现在mysql已经使用非常成熟。但是现在列表页数据量和计算量都比较大。后期会经常增加字段。

Hbase 增加和修改列比较简单。数据冗余对性能影响几乎可以忽略不计。比较适合当前场景。所以选型Hbase。

## Hbase

### 使用场景

1．支持海量数据，需要TB/PB级别的在线服务。

2．读写频率高,非完全实时

3．Column动态修改

4．无事物要求

### 弊端

没有二级索引只有主键(rowKey);

没有聚合，排序等等功能。

### Hbase 存储结构

HBase的每个列都属于一个列族，以列族名为前缀，如列e:cid和e:sku属于e列族。访问控制，磁盘和内存使用统计都在列簇层面进行。在实际存储中要控制列簇的数据量。尽量一个表一个列簇。列簇数量会极大影响性能。

|  |  |  |  |
| --- | --- | --- | --- |
| row key | **column Family(e)** | | |
|  | column 1 | column 2 | column 3 |
| cid,sku1 | <cid:1> | sku:1 | price:2 |
| cid,sku2 | <cid:1> | sku:2 | price:10 |

### Row Key 设计

1．尽量短

2. row key 设计决定了你扫描的范围和取出的规则。

Row key使用

|  |  |  |  |
| --- | --- | --- | --- |
| %s,%s |  |  |  |
| 1,2 | <cid:1> | sku:1 | price:2 |
| 1,3 | <cid:1> | sku:2 | price:10 |
| 1,4 | <cid:1> | sku:3 | price:2 |
| 1,5 | <cid:1> | sku:3 | price:10 |

按key取出数据

   Get get = new Get(Bytes.toBytes(row));

Result result = table.get(get);

分页取出数据

String Prefix= "1";

Filter prifixFilter =new PrefixFilter(prefix.getBytes());

Filter pageFilter = new PageFilter(100);

list.add(pageFilter);

list.add(prifixFilter);

Filter all = new FilterList(list);

int totalRows = 0;

String lastRow = null;

Table table = getTable(10");

Scan scan = new Scan();

scan.setFilter(all);

scan.setCaching(100);

scan.setStartRow(Bytes.toBytes(prefix + ",/"));

ResultScanner scanner = table.getScanner(scan);

Scan 使用注意

使用scan 时发现id 不连续是因为Hbase存储是用ASCII排序的。

"/"<"0-9"<":"

## [HBase 二级索引](http://www.cnblogs.com/sixiweb/p/3575132.html)

Hbase 二级索引，是对rowkey做倒排索引。我现在是用ES座二级索引。