CHTC
中国HBase技术社区

云栖社区

Solr增强HBase检索能力
基础介绍及场景

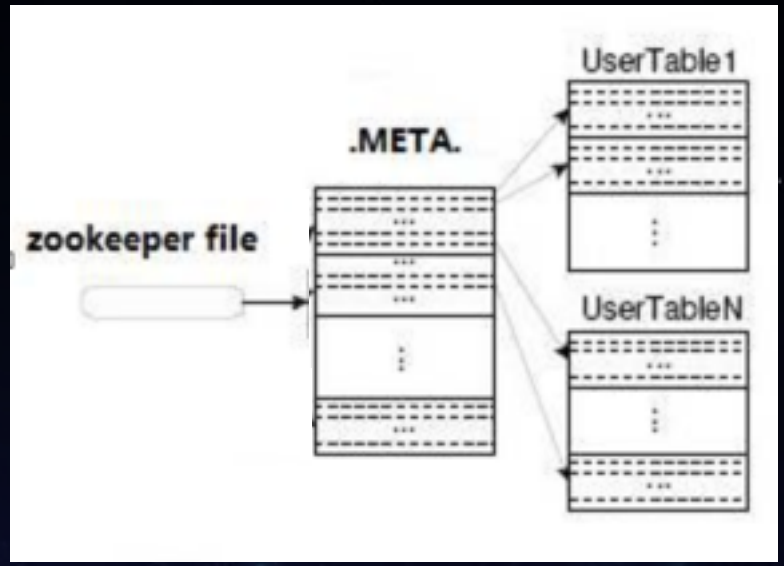
阿里云 天斯

目录 / Contents

01 hbase查询特点

1 hbase查询特点

- HBase中的数据是根据 Rowkey 的字典序来排序的
- 根据Rowkey进行查询，才能快速定位数据所在region
- 没有指定Rowkey的查询，全表扫描



充分利用RowKey

2 查询示例说明

会员信息表

FieldName	Description	FieldType
uid	编号	string
name	姓名	string
age	年龄	int
gender	性别	boolean
phone	电话	string
memberType	会员类型	int
job	职业	string
addr	住址	string
email	邮箱	string
. . . . .		
. . . . .		

业务需求

对uid、name、age、phone、gender、memberType进行任意条件组合查询

3 Rowkey适用前缀查询

业务需求

主要查询业务uid：

- uid = 111 AND phone = 123 AND name = abc
- uid = 111 AND phone = 123
- uid = 123 AND phone = 12?
- uid = 13* order by uid asc
- uid = 332
- uid > 123 AND uid < 224

Rowkey： uid | phone | name

4 新增简单业务

业务需求

新增简单查询业务：

- phone = 123 AND name = abc　Rowkey：phone | name
- phone = 12* order by phone asc　Rowkey：phone
- phone > 123 AND phone < 330　Rowkey：phone
- phone = 123　Rowkey：phone
- name = abc　Rowkey：name
- name = a*　Rowkey：name
- name = abc order by age desc　Rowkey：name | age
- name = abc order by uid asc　Rowkey：name | uid

Rowkey： uid | phone | name

	rowkey	f:name	f:age	f:email	… …
region1	111|123|abc				
	112|163|cba				
	123|123|bba				
region2	134|156|aaa				
	139|158|bbb				
	216|129|bcc				
region3	224|186|caa				
	332|177|cbb				
	……				
…	……				
regionN					

5 新增复杂业务

业务需求

新增复杂查询业务：对uid、name、age、phone、gender、memberType进行任意条件组合查询

- age >23 OR phone > “12” order by uid

  Rowkey：phone | uid ？ Rowkey：age | uid ？ Rowkey：age | phone | uid ？ Rowkey：phone | age | uid ？
- age >23 OR name = 12* order by age desc Rowkey：name | (200 - age) ？ Rowkey：(200 - age) | name ？
- (gender = 1 OR age >23) AND phone > “12” order by uid Rowkey ？
- min/max/count/sum/age(age) where gender = 1 AND merberType in (1,2,3)
- count(*) where gender = 0 AND name = *bc group by memberType
- (age > 12 OR phone in (123,234,156)) AND memberType=1 AND gender = 0 order by uid desc
- … …

6 hbase查询局限性

rowkey查询局限性：

- 一个Rowkey只有一个前缀，也就只能有一个条件做为前缀，查询条件必须给这个条件，否则rowkey无用
- 当有多个 OR条件组合时，一个rowkey无法满足。例如： name = a* OR phone = 12*
  如果建立两张索引表A(rowkey=name)、B(rowkey=phone), 那么需要业务端控制分别索引表A/B来取结果，增加业务复杂度
- Rowkey中即使有多个条件，也必须根据前缀循序依次给出，并且只能最后一个条件为模糊或者范围，否则不能充分发挥Rowkey定位能力。例如： uid|name|phone|age ， 如果只给 name、phone、age，那么还是会全表扫描。
- 无法满足多条件进行任意的各种AND/OR组合，以及order by/group by
- 不能进行高效的统计分析查询，sum/min/max/avg/mean，需要写复杂的endpoint

02 solr增强hbase检索能力

1 solr功能简介

- Solr是在lucene基础上构建的企业级全文搜索平台
- SolrCloud模式高效管理分布式lucene索引，扩展性强
- Solr提供了丰富搜索功能，除了各种类型条件的任意组合查询外，还有如facet、group/cluster、分词、stats、时空等查询，还有一些analytics的查询功能。
- 社区发展成熟，应用广泛

2 hbase查询局限性

HBase

主表一个rowkey，只能设计一个rowkey=XlY …这种场景

合适场景：
X=a
X>=a, X>a
X<=a,或者X<a
X=a and Y = b
X=a and Y <= b
X=a and Y>=b

优点：
高并发、高效快速
缺点：
1.只有一个rowkey设计，后期业务变化不能修改rowkey结构
2.检索场景简单，有局限性，一个rowkey必须由前缀X出现才能快速查找，比如上述 只提供Y=b的话，依然需要全表扫描

HBase二级索引

扩展更多rowkey设计，允许更多
rowkey=XlY
rowkey=YlZ
rowkey=XlZ 。。。这种设计

合适场景：
一个表，允许更多类似左边的场景。
更多X、Y、Z...条件来组合来进行类似左边描述的条件查询

优势：
依然保持高并发、高效rowkey查询。允许有更多rowkey设计，最大化hbase rowkey检索优势
缺点：
1. 表变多，如果业务有U/V/W/X/Y/Z 6个条件两两组合的业务场景，就需要15个表，数据膨胀，例如这个时候，X在索引里被保存了5次。
2. 查询也有局限性，无法满足企业综合的搜索需求

Solr

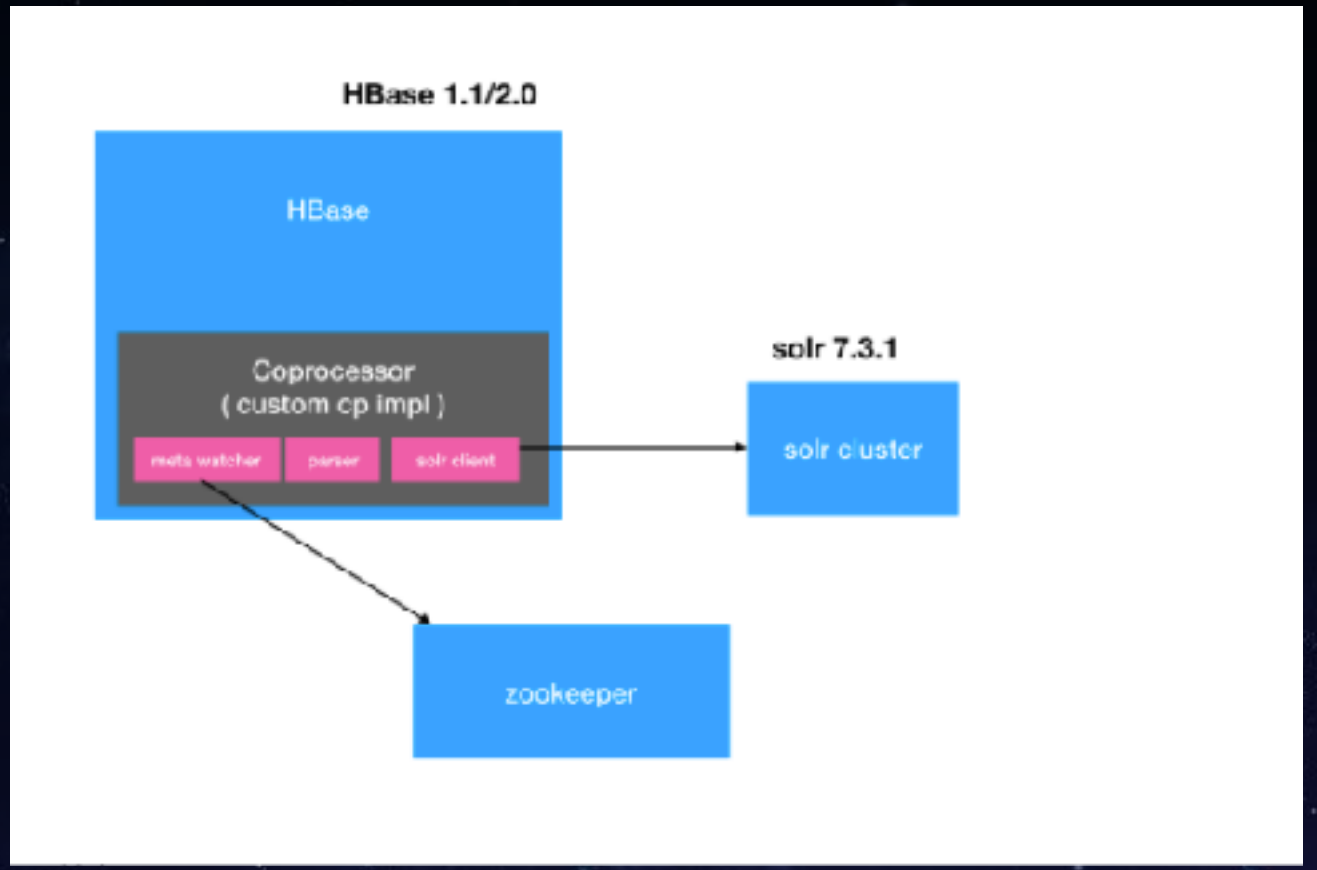
任意条件 X、Y、Z.....

合适场景：
1.X = a or Y= b (rowkey设计不能实现的)
2.X> a and Y < b (rowkey设计不能实现的)
3. X like “%hbase%”
4. Geo地理距离检索
5. 支持分词查询
6. 支持facet/group分类分组查询返回，例如一个关键词搜索新闻网站，它可以分政治、体育、经济等类别返回统计与结果
7. 任意条件组合查询等

优点：
1. 支持检索功能更丰富； OR 组合查询，多个条件范围组合查询，like、分词等全文检索，这些查询使用hbase rowkey设计是难以满足的
2. 对于类似U/V/W/X/Y/Z 6个条件两两组合的业务场景,数据膨胀率远低于 hbase的rowkey方案
缺点：类似hbase这种简单kv查询下，并发不如hbase高效快速

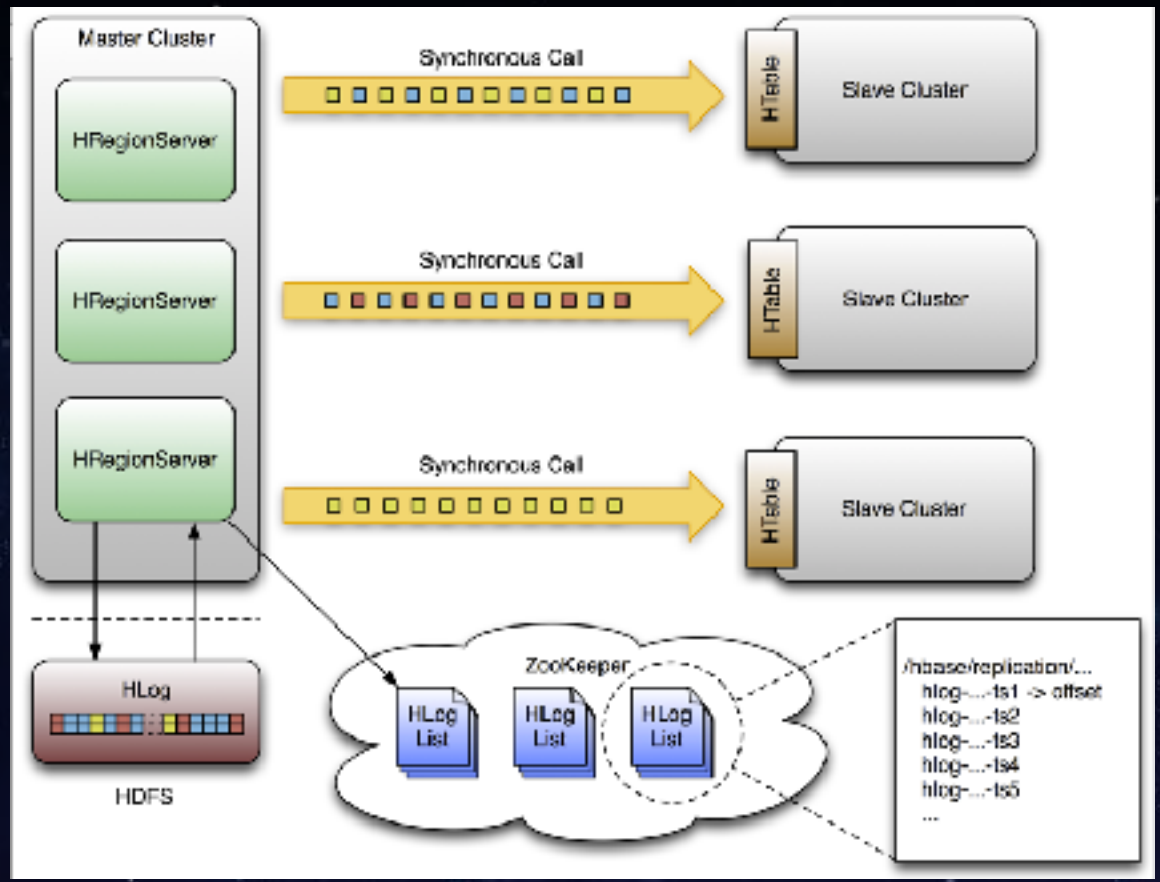
3 solr增强hbase检索能力

hbase 协处理器+solr的架构图



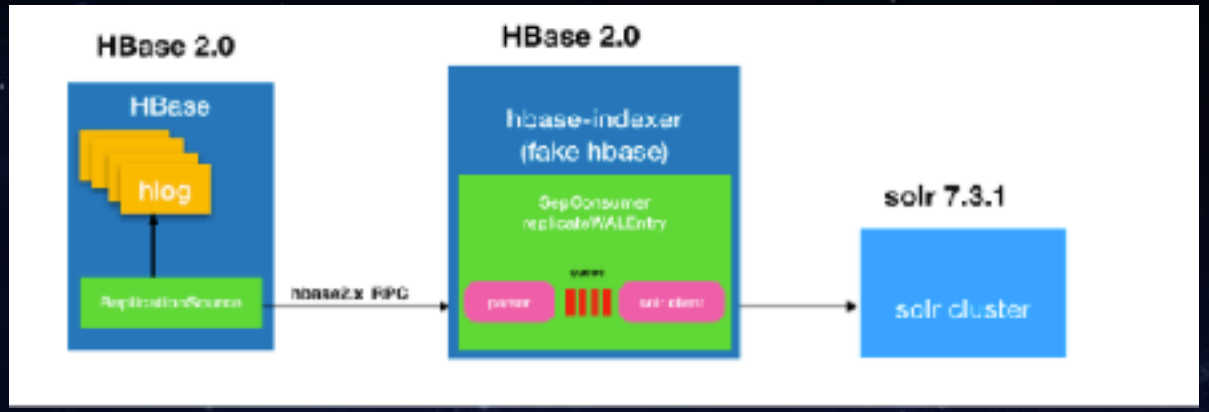
4 solr增强hbase检索能力

hbase replication+solr的架构图



5 架构基础介绍

hbase replication + solr + 开源indexer



6 架构特点

优势：

- 数据的facet、group、order、条件分页、terms计数、常见stats函数、综合复杂的条件组合查询、充分利用cache、支持分词查找

不足：

- a). solr不支持ttl，相应需要建立索引的hbase表也不应该用
- b). solr不支持列级别多版本，相应需要建立索引的hbase表也不应该用多版本
- c). hbase表数据，用户不应该自定义版本

03 场景应用&示例

1 场景应用企业&场景

附录1：PublicServers
附录2：WhoUseSolr

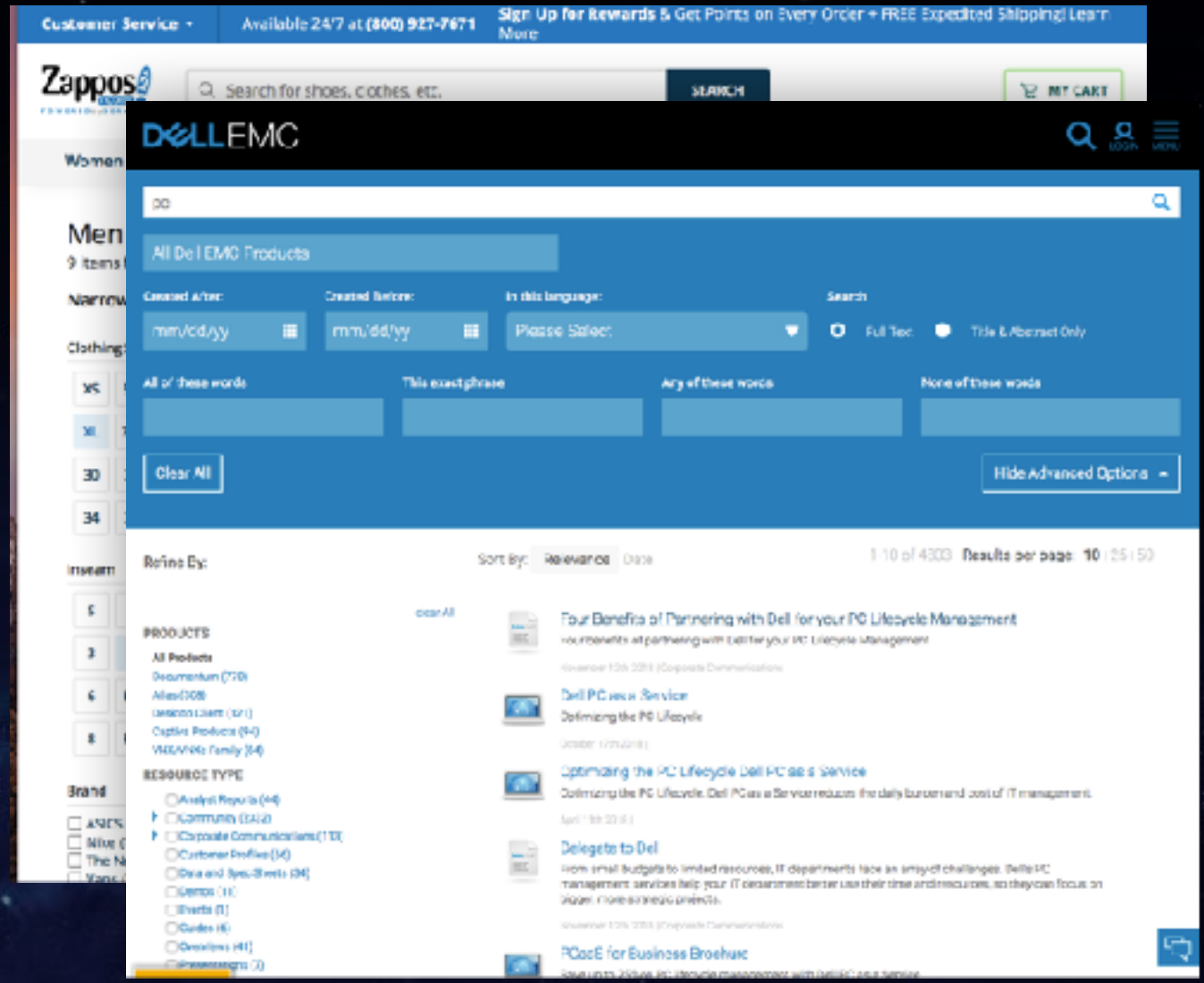
2 具体业务抽象——任意条件

多条件任意组合查询：

各种条件之间进行任意的组合查询
提升用户查询筛选的灵活性

- 例如：购物网站可以按照 裤子、大小、型号、尺码颜色、长短风格等，任意组合查询，并按照各自类型的相关度排序
- 例如：某网站根据时间、关键字、语言类型等条件，随意组合进行查询筛选

扩展思考：我们的系统查询搜索服务是否有类似需求？

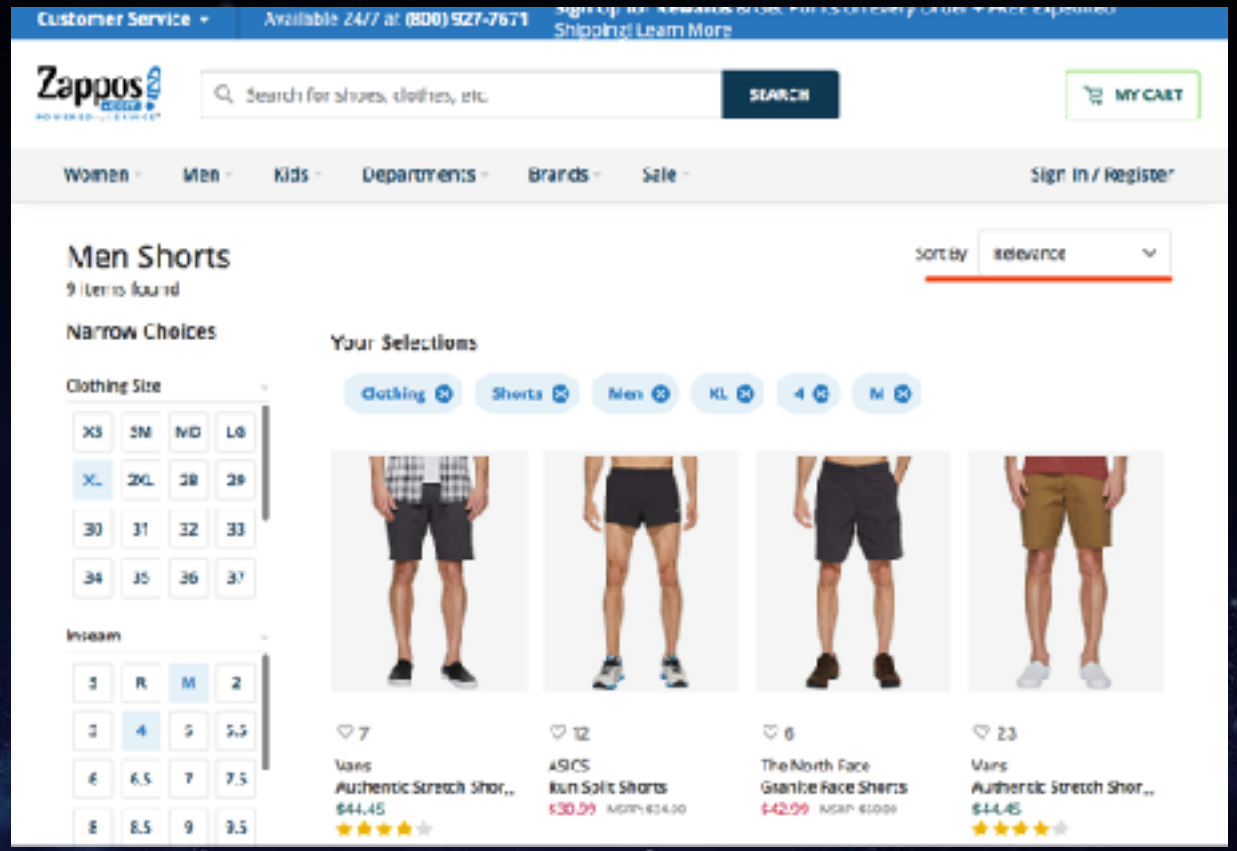


3 具体业务抽象——各种排序

各种排序需求：

一个综合性的查询业务，除了各种条件的任意组合复杂查询外，往往可以允许用户有多种类型的排序需求

- 例如：一个商品搜索，用户可能需要价格、型号、评分、销量等进行排序，以满足不同人参考不同指标进行排序筛选商品



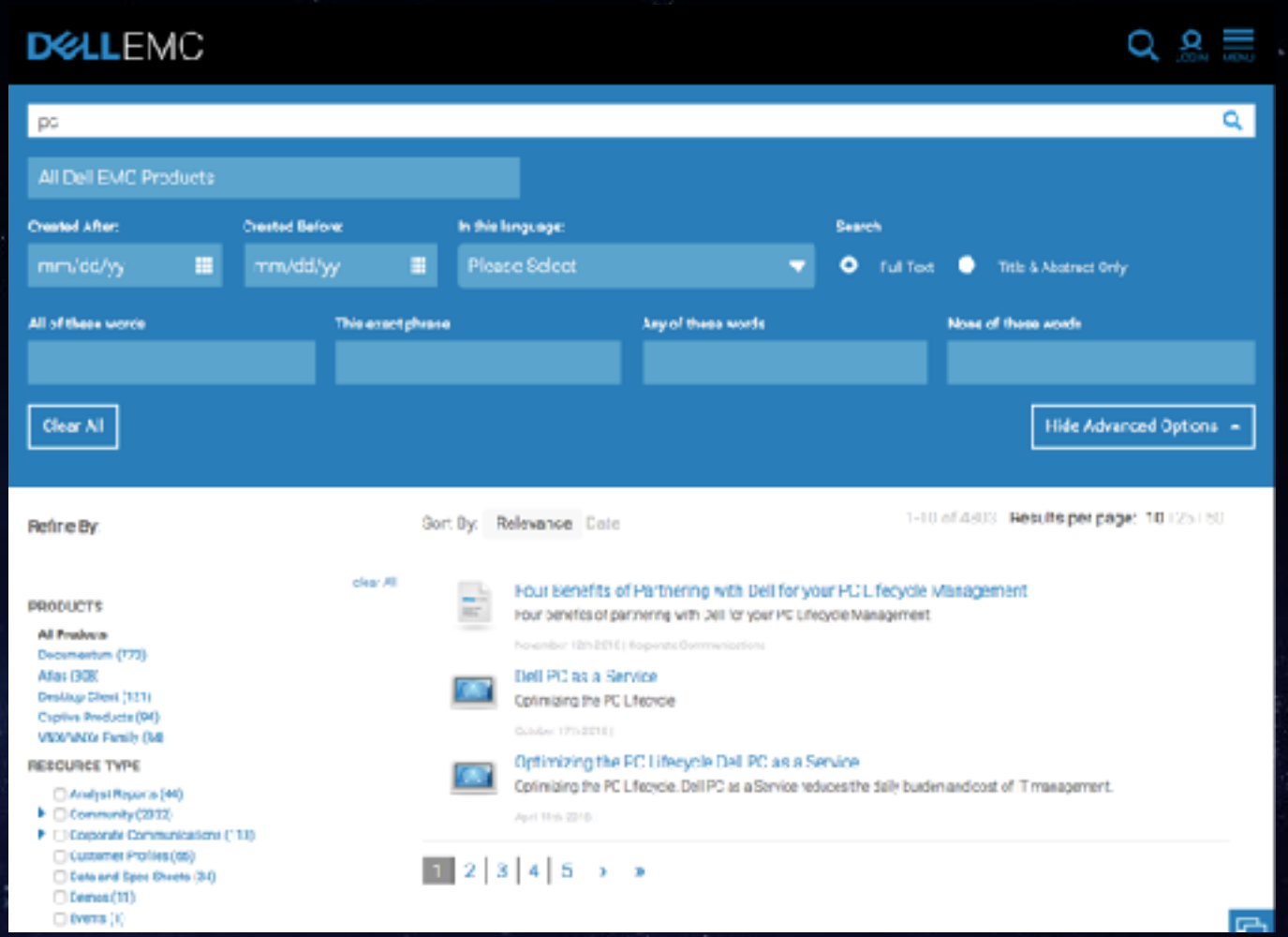
扩展思考：我们的系统会只有多少种排序需求？
时间、用户手机、用户id、正序反序？

4 具体业务抽象——分页

复杂查询查询分页：

如上述的任意条件查询情况下，还要排序并分页

- hbase的一个rowkey只有一个顺序，但是我们可能需要各种类型的排序
- 复杂查询hbase下分页迭代，hbase很难充分利用cache等机制，每次排序任务都是重新进行



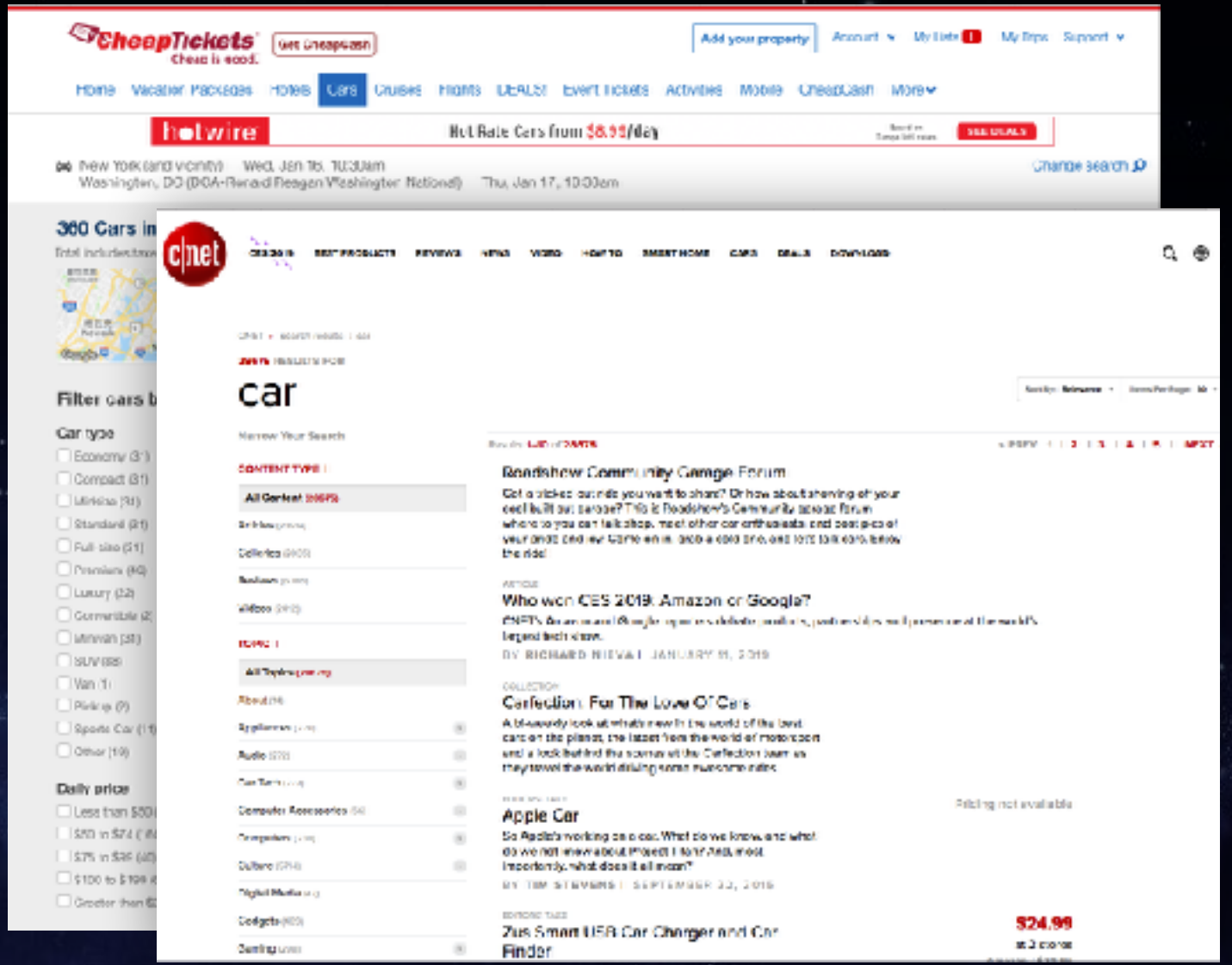
5 具体业务抽象——facet

facet navigation查询需求：

根据查询结果，按照某类型进行统计匹配个数

- 例如：纽约~华盛顿 租车预定
  网站左边提供了满足这个里程需求的情况下，按照Car Type进行分类，每个类别统计有多少辆车符合要求，供用户快速筛选

- 例如：新闻网查询car相关
  左侧显示文章按照类型统计匹配个数



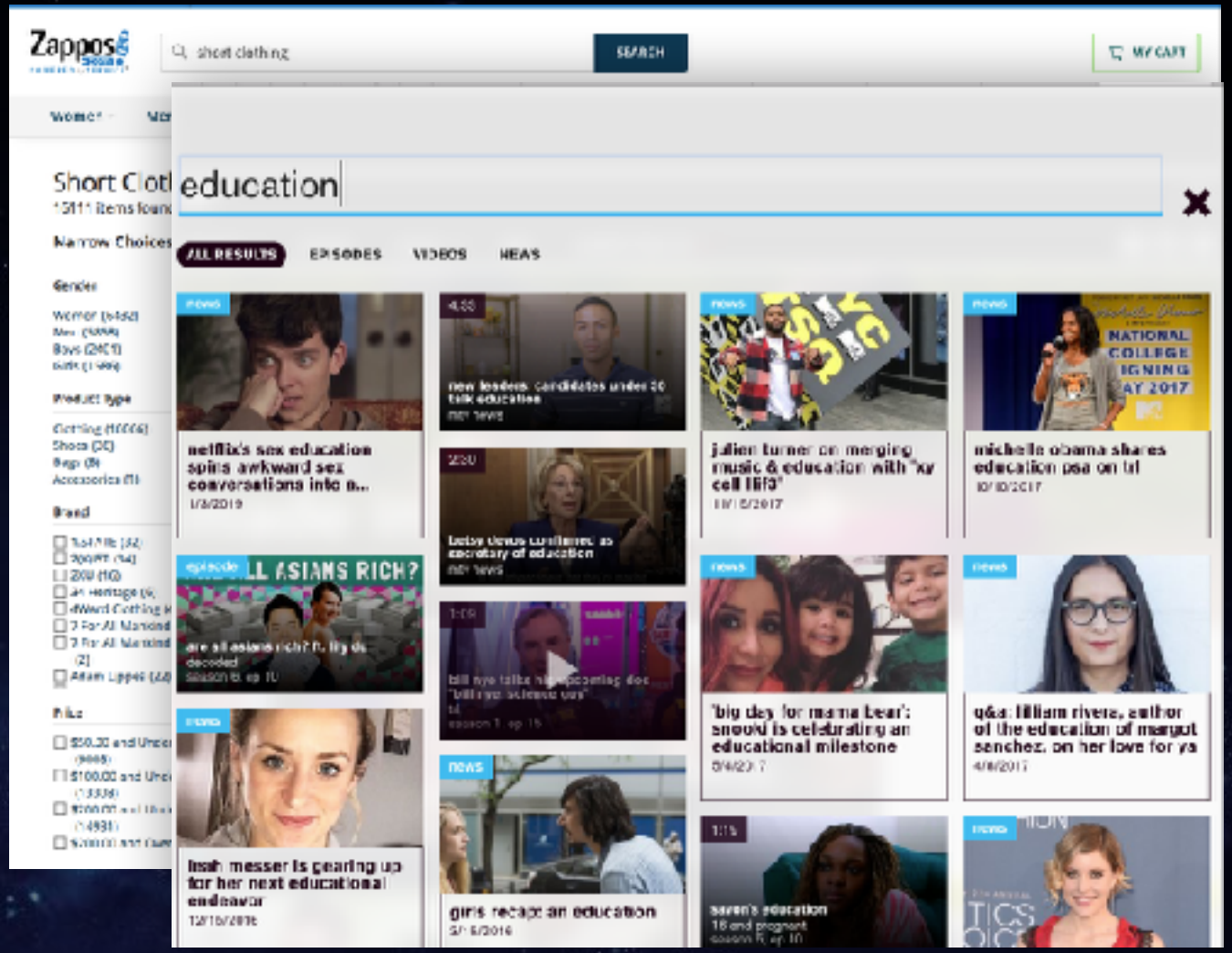
扩展思考：facet在各行各业网站上的应用？
交通、电商、综合信息检索、图书/电影娱乐信息

6 具体业务抽象——分词

分词查询：

输入关键字，匹配文本中带此关键字的字段

- 例如：电商网站搜索商品时，输入的只是关键字，它能匹配商品标题中的关键字，无需匹配整条标题文本
- 例如：某视频娱乐网站，关键字搜索标题带有education的视频



扩展思考：如果没有分词，如何满足这个需求？
传统的是利用模糊查询，但模糊查询往往效率不如意

7 具体业务抽象——统计

常用后台业务统计：

一些后台管理系统，经常会需要做一下常用的统计分析，例如：avg/min/max/mean/sum/count等，再结合一些query条件以及group分组

- 例如：后端会员系统统计平台平均用户年龄，还可能会按照范围 range 统计： <23岁有多少，23~32，>32 岁 有多少，分别按照range进行group统计

会员信息表

FieldName	Description	FieldType
uid	编号	string
name	姓名	string
age	年龄	int
gender	性别	boolean
phone	电话	string
memberType	会员类型	int
job	职业	string
addr	住址	string
email	邮箱	string
.....		
.....		

8 具体业务抽象——func

支持丰富的函数计算下推：

solr的function query支持几十中函数计算，这些计算都发生在server端，有些可以作为条件、也可以作为返回字段，相比function都在客户端计算，性能要高

q=*:*&sort=sum(x,y,z) desc //根据sum(x,y,z) 结果倒序

q=*:*&fl=id,map(x,1,10,0)　// 当x为[1,10]范围时，返回0

q=*:*&fl=id,map(x,0,100,sum(x,599),20) // x为 [0,100]是返回sum(x,599)
//其他值返回 20

q=*:*&fl=id,min(x,y,z),score&wt=xml //返回 x,y,z中的最小值

bs	numdocs	freq
childfield	ord	docFreq
concat	payload	termFreq
def	pow	tf
div	product	top
dist	mul	totaltermfreq
docfreq	query	and
field	recip	or
hsin	rord	xor
idf	scale	not
if	sqedist	exists
linear	sqrt	gt
log	strdist	gte
map	sub	lt
max	sum	lte
min	add	eq
ms	sttf	……
norm	totalTermFreq	

04 后续学习&实践建议

1 后续学习——环境搭建

开源hbase1.1+solr6.3+indexer1.6 学习测试环境搭建

https://github.com/NGDATA/hbase-indexer

https://github.com/apache/lucene-solr

https://github.com/apache/hbase

自行编译所需特定版本

1. 搭建hbase集群
   开启replication功能
   - hbase-site.xml
   - 1.x版本hbase.replication=true
   - 2.x版本默认开启
2. 搭建solr集群
   搭建教程
   - solr.in.sh
   - ZK_HOST=zk1,zk2,zk3:2181/solr
3. 搭建indexer集群
   搭建教程
   - hbase-indexer-site.xml
   - hbaseindexer.zookeeper.connectstring
   - hbase.zookeeper.quorum
4. 构建索引
   插入数据
   查询
   示例demo

2 后续学习——建立索引

1. hbase创建表

```
$ hbase shell
hbase> create 'mytest', { NAME => 'info', REPLICATION_SCOPE => '1' }
```

2. solr创建collection

```
$solr create_collection -c mycollection -shards 1  -replicationFactor 1
```

3. indexer创建映射

```
$HBASE_INDEXER/bin/hbase-indexer  add-indexer -n myindex -c myindex.xml   \
    -cp solr.zk=zk1,zk2,zk3:2181/solr  -cp solr.collection=mycollection
```

myindex.xml文件内容如下：

```
<?xml version="1.0"?>
<indexer table="mytest">
  <field name="firstname_s" value="info:firstname"/>
  <field name="lastname_s" value="info:lastname"/>
  <field name="age_i" value="info:age" type="int"/>
</indexer>
```

HBase → Indexer → Solr

rowkey	info:firstname	info:lastname	info:age
row1	lebron	james	34
row2	kobe	bryant	40


```
{
 "id": "row1",
 "firstname_s": "lebron",
 "lastname_s": "james",
 "age_i":  34,
 "_version_": 1622173949105274880
},
{
 "id": "row2",
 "firstname_s": "kobe",
 "lastname_s": "bryant",
 "age_i":  40,
 "_version_": 1622173949176578048
}
```

3 后续学习——常用实例demo

1. 任意条件组合

```
SolrQuery solrQuery = new SolrQuery("(name:Test AND cat:search) OR cat:electronics”);
QueryResponse response = client.query("techproducts", solrQuery);
```

精确：field:value1
范围：field:[1 TO 10] 、field:[1 TO 11} 、field:{0 TO 11} 、field:[50 TO *]　field:[* TO 100]
wildcard：field:ab*d 、 field:ab?d
fuzzy：field:roam~ 、 field:roam~1 //默认变化距离是2，可以指定为1；例如：roon这种也不能被 roam~1匹配，能被roam~
proximity： “solr apache”~3 // 通过移动单词位置在10次数以内能匹配到的字段, 比如：”Solr is built on Apache Lucene”
phrase：“apache lucene” // 相当于两个词中间不能有其他单词

4 后续学习——常用实例demo

2. 排序&分页

```
SolrQuery solrQuery = new SolrQuery("(name:Test AND cat:search) OR cat:electronics”);
solrQuery.addSort(“id”,SolrQuery.ORDER.asc);
solrQuery.setRows(5);
solrQuery.setStart(0);
QueryResponse response = client.query("techproducts", solrQuery);
```

除了 setRows/setStart这种方式，还可以使用cursorMark来迭代获取

```
String nextCursorMark = CursorMarkParams.CURSOR_MARK_START;
String currentMark = null;
do{
  SolrQuery solrQuery = new SolrQuery("(name:Test AND cat:search) OR cat:electronics”);
  solrQuery.addSort(“id”,SolrQuery.ORDER.asc);
  solrQuery.setRows(5);
  solrQuery.add(CursorMarkParams.CURSOR_MARK_PARAM,currentMark);
  QueryResponse response = client.query("techproducts", solrQuery);
  nextCursorMark = response.getNextCursorMark();
  //do something
}while(!nextCursorMakr.equals(currentMark));
```

cursorMark使用时，sort必须包含uniqueKey 默认也就是id

当nextCursorMark不再变化时，表示此查询数据已迭代完

5 后续学习——常用实例demo

3. facet查询

```
SolrQuery solrQuery = new SolrQuery(“gender:1 AND memberType:2”);
solrQuery.setFacet(true);
solrQuery.addFacetField(“age”);
solrQuery.setFacetLimit(8); //设置返回8个统计数值
QueryResponse response = client.query("userinfo", solrQuery);
SolrDocumentList result = response.getResultes();// 查询匹配doc结果
List<FacetField> facetFields = response.getFacetFields(); //匹配结果中，按照每个年龄统计
```

```
"facet_fields":{
 "age":[
  "18",1,
  "19",6,
  "22",2,
  "24",3,
  "27",9,
  "28",10,
  "29",12,
  “32”,2]}
```

按照每个年龄统计不是很实用，通常都是年龄段范围来统计，修改如下：

```
SolrQuery solrQuery = new SolrQuery(“gender:1 AND memberType:2”);
solrQuery.setFacet(true);
solrQuery.addFacetQuery(“age:[* TO 20]”);
solrQuery.addFacetQuery(“age:[21 TO 30]”);
solrQuery.addFacetQuery(“age:[31 TO 40]”);
solrQuery.addFacetQuery(“age:[41 TO *]”);
QueryResponse response = client.query("userinfo", solrQuery);
SolrDocumentList result = response.getResultes();// 查询匹配doc结果
Map<String,Integer> facetQuery = response.getFacetQuery(); //匹配结果中，按照每年龄范围统计
```

```
"facet_queries":{
  “age:[* TO 20]”:7,
  “age:[21 TO 30]”:36,
  “age:[31 TO 40]”:23,
  “age:[41 TO *]”:4
 }
```

6 后续学习——常用实例demo

4. 分词查询

SolrQuery solrQuery = new SolrQuery(“name:Test”);
QueryResponse response = client.query("techproducts", solrQuery);

techproducts的name字段类型：text_general

```
<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100"
multiValued="true">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
    <!-- in this example, we will only use synonyms at query time
    <filter class="solr.SynonymGraphFilterFactory" synonyms="index_synonyms.txt"
ignoreCase="true" expand="false"/>
    <filter class="solr.FlattenGraphFilterFactory"/>
    -->
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
    <filter class="solr.SynonymGraphFilterFactory" synonyms="synonyms.txt" ignoreCase="true"
expand="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
```

IK中文分词举例：文档参考
结合词典分词和文法分析算法的中文分词组件

SolrQuery solrQuery = new SolrQuery(“description:中国”);
QueryResponse response = client.query("techproducts", solrQuery);

字段类型：text_ik

```
<fieldType name="text_ik" class="solr.TextField">
  <analyzer type="index">
    <tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" useSmart="false" />
  </analyzer>
  <analyzer type="query">
    <tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" useSmart="true" />
  </analyzer>
</fieldType>
```

7 后续学习——常用实例demo

5. function

```
SolrQuery solrQuery = new SolrQuery(“*:*”);
solrQuery.addField(“id”);
solrQuery.addField(“name”);
solrQuery.addField(“age”);
solrQuery.addField(“salary”);
solrQuery.addField(“bonus”);
solrQuery.addField(“map(age,0,17,-1)”); // 年龄在[0,17]的返回-1
solrQuery.addField(“sum(salary,bonus)”);// 工资+奖金
QueryResponse response = client.query("userinfo", solrQuery);
```

function query参考

bs	numdocs	freq
childfield	ord	docFreq
concat	payload	termFreq
def	pow	tf
div	product	top
dist	mul	totaltermfreq
docfreq	query	and
field	recip	or
hsin	rord	xor
idf	scale	not
if	sqedist	exists
linear	sqrt	gt
log	strdist	gte
map	sub	lt
max	sum	lte
min	add	eq
ms	sttf	……
norm	totalTermFreq	

8 后续学习——常用实例demo

6. stats&analytics

min/max/avg/sum统计

```
SolrQuery solrQuery = new SolrQuery(“*:*”);
solrQuery.set(“json.facet”,”{max_age:\”max(age)\”,avg_age:\”avg(age)\”}”); // 查询结果中，所有年龄的最大值和平均值
QueryResponse response = client.query("techproducts", solrQuery);
NestableJsonFacet jsonFacet = response.getJsonFacetingResponse();
System.out.println(jsonFacet.getStatFacetValue("max_age"));
System.out.println(jsonFacet.getStatFacetValue(“avg_age"));
```

analytics 分析

allows users to calculate complex statistical aggregations over result sets

```
curl --data-urlencode 'analytics={
    "expressions" : {
        "revenue" : "sum(mult(price,quantity))"
    }
}'
http://localhost:8983/solr/sales/select?q=*:*&wt=json&rows=0
```

analytics 为contrib模块，默认不打开

使用文档参考链接

9 后续学习——熟悉底层存储

熟悉底层文件格式

借助simpletext codec 明文格式快速了解，跟多实际细节 参考文档
设置简单单个shard，solrconfig.xml中设置： <codecFactory class="solr.SimpleTextCodecFactory"/> 大量明文的格式可以学习
例如：lucene v8.0.0最新的： SimpleTextCodecFactory (SimpleTextCodec) & SchemaCodecFactory (lucene80Codec)
文件对比表见下一页。

借助luke工具查看底层文件保存内容
可以使用独立的luke项目工具，也可以使用solr集成的luke handler进行查看，参考文档
如： curl “http://localhost:8983/solr/techproducts/admin/luke"

10 后续学习——熟悉底层存储

文件	扩展	明文格式扩展	描述
Segments File	segments_N	segments_N	Stores information about a commit point
Lock File	write.lock	write.lock	The Write lock prevents multiple IndexWriters from writing to the same file
Segment Info	.si	.si	Stores metadata about a segment
Term Dictionary	.tim	.pst	The term dictionary, stores term info
Term Index	.tip		The index into the Term Dictionary
Frequencies	.doc		Contains the list of docs which contain each term along with frequency
Positions	.pos		Stores position information about where a term occurs in the index
Payloads	.pay		Stores additional per-position metadata information such as character offsets and user payloads
Norms	.nvd, .nvm	.len	Encodes length and boost factors for docs and fields
Per-Document Values	.dvd, .dvm	.dat	Encodes additional scoring factors or other per-document information.
Term Vector Index	.tvx	.vec	Stores offset into the document data file
Term Vector Data	.tvd		Contains term vector data.
Live Documents	.liv	.liv	Info about what documents are live
Point values	.dii, .dim	.dii, .dim	Holds indexed points, if any
Compound File	.cfs, .cfe	.scf	An optional "virtual" file consisting of all the other index files for systems that frequently run out of file handles

11 实践建议

- 对于时间的范围查询，建议固定好1h/6h/1d这种范围，并利用filterquery进行，能复用cache
- cursor必须sort字段有uniqkey，默认也就是id字段，cursor不能与timeallowed一起使用
- 我们的查询可以都带上一个timeallowed参数，设置60s等，及时中断掉慢查询，防止后台一直跑
- collection的solrconfig.xml中添加slowQueryThreshold, 打印慢查询日志
- collection的solrconfig.xml中，去掉schemaless，效率低，不可确定性。只能用于学习
- filter query用起来，cache配置 ram大小的方式比较好估算，要整体考虑各个collection的cache配置比例
- 前缀模糊如：*abc这种尽量少用，如果是title中需要匹配关键字，可以考虑分词

05 小结&提问

1 小结

1. 为什么增强hbase检索能力？

介绍hbase查询特点

2. solr在哪些方面增强了hbase的检索能力？

任意条件复杂组合、facet、统计&分析、分词、排序分页、func

3. 如何快速上手hbase+solr？

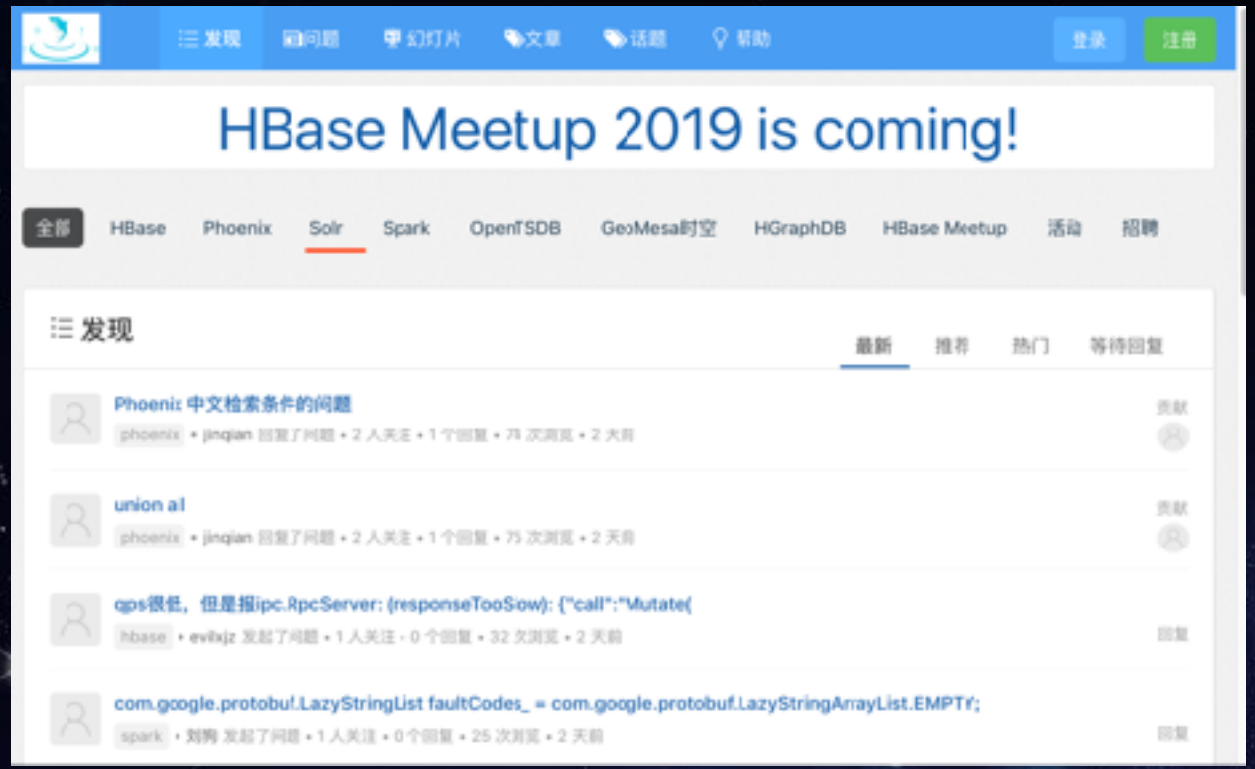
hbase+solr+indexer环境搭建教程，各种业务示例demo，相关的学习材料，实战注意事项

碰到问题 -> 社区提问

课后思考：phoenix + hbase 和 solr + hbase 适用场景思考？对比？替代？

2 提问

可以社区提问：http://hbase.group



HBase生态+Spark社区大群

HBase 技术社区
钉钉交流群

3 加入我们

社区管理员

HBase 技术社区
微信公众号

HBase 技术社区
钉钉交流群

CHTC
中国HBase技术社区

云栖社区

THANK YOU

中国 HBase 技术社区网站：http://hbase.group