POI 结构化和非结构化数据存储检索策略

吴 姗，赵 健，刘丹丹，余江顺，伍思睿

（中国电建集团 贵州电力设计研究院有限公司，贵阳 550002）

摘要：POI（兴趣点）数据具有结构化和非结构化属性，使用关系型数据库存储时，非结构化占用内存空间大，检索时直接从数据库读取，且 POI 空间信息需要通过 HBase 的 rowkey 进行多条件检索，效率低下。通过研究 POI 数据特点，构建数据模型，针对 POI 结构化属性数据和非结构化数据提出不同的存储和检索策略并创建索引。最后使用 Sqoop 完成 POI 属性数据和非结构化数据的交互并嵌入 Solr 技术进行检索。经实验验证，应用该方法针对 POI 非结构化数据检索效率有所提高。

关键词：POI；HBase；Solr；POI 存储；POI 检索

中图分类号：TP311 **文献标志码**：A **文章编号**：1671－1807(2023)03－0267－06

随着社会日趋智能化，地图逐渐进入人们的生活，而地图与之相关的专业名词就是兴趣点（point of interest，POI）。POI 广义可以理解为点的地理对象，主要用来对事物地址进行描述，能较大程度上增强对事物位置的描述和查询能力。POI 数据既包含诸如名称、电话、经纬度、地址等属性数据，也包含视频、音频等一系列非结构化数据，而且 POI 数据日益增长，能够达到 T-B 级甚至 P-B 级。

对于 POI 的研究越来越广泛，但并没有很好地对 POI 进行高效管理的方式。本文旨在 POI 数据在爆炸性增长的问题上，针对海量 POI 数据存储问题，采用混合存储策略；针对 POI 数据检索，使用关系型数据库的检索方案和 HBase 检索数据的相关方法，且对 HBase 建立二级索引，同时嵌入 Solr 全文的搜索引擎，Solr 可实现全量更新与自动缓存，与 Oracle 形成互补的关系，增强 Oracle 的查询，提高检索效率。

1 POI 存储策略

1.1 POI 数据分析

通过分析，POI 数据包含了名称、类别和地址 3 个属性。POI 的名字解决了“是什么”，POI 的地址回答了“在哪里”，除此之外，由于 POI 的复杂性，还包括其他的一些属性信息，如电话、地址编码、描述信息等。还包括 POI 的非结构化的数据，如空间坐标、POI 点的视频、音频等信息[1]，非结构化的数据关系并不紧密。数据结构对比见表 1。

表 1 数据结构对比

数据结构	数据模型	表现形式	其他
结构化数据	二维表（关系型）	行数据	先有结构，再有数据
半结构化数据	树、图	HTML 文档	先有数据，再有结构
非结构化数据	无	文档、图片、视频等	—

由于每个 POI 都有属于自己的名称、电话、经纬度、地址等信息，在 POI 建模时，考虑 POI 通用的数据模型，将其提取出来，方便 POI 的管理。

1.2 结构化属性数据存储策略

POI 属性数据可使用关系型数据库 MySQL 或者 Oracle 存储。关系数据库存储的基本形式是数据表，每个表只能属于一种模式，但可以被多个用户访问[2-3]。表中的记录由众多字段组成，用户可以对某些字段定义规则，这类规则叫作完整性约束，它对于表中这个字段的所有记录都是有效的[4-5]。

针对本文的 POI 属性样本数据，在 Oracle 中建立一张通用的名为 POI 的表，其结构见表 2，并在关系型数据库中建立用户名，进行权限分配、表空间分配，最后借助 Oracle 操作软件 PL/SQL，将数据导入 Oracle 数据库，完成属性数据存储。

收稿日期：2022-09-01

作者简介：吴姗（1993—），女，贵州毕节人，中国电建集团贵州电力设计研究院有限公司，工程师，地图制图学与地理信息工程专业硕士，研究方向为地理信息基础理论研究与软件研发。

表 2　POI 属性表结构

序号	字段代码	类型	字段名称
1	F_ID	String	元素代码
2	F_NAME	String	POI 名称
3	F_LON	Double	经度
4	F_LATI	Double	纬度
5	F_ADDRESS	String	POI 地址
6	F_PHONE	String	POI 联系方式
7	F_TYPE	String	POI 所属类型
……	……	……	……

1.3　非结构化数据存储策略

关系数据库存储 POI 数据时，信息不存在的字段会自动存为 null，即便为 null，仍然占用关系数据库内存，且增加 POI 字段时，需要停止服务，耗时较大。而 HBase 对非结构化数据处理性能远远高于关系型数据库，且 HBase 能动态增加列，可以很好地解决停止服务方可增加字段的缺陷。故针对 POI 非结构化数据，本文采用 Hbase 数据库进行存储。

1.3.1　HBase 存储 POI 非结构化数据逻辑

HBase 可以动态增加，可理解成一张偌大的表，列数一般可以增加到超 10 亿列，列宽足以对动态增加的数据列进行管理，可随时进行数据扩展，但增加的列并不是全都用来存储数据。本文研究数据存储时，主要是要对行键和列簇进行设计。表 3为 HBase 逻辑视图。

表 3　HBase 逻辑视图

Row Key	Time Stamp	POI Inform				
		qualifier	name	latlon	picture	describe
1001	t_4	point	大厦	22.809,108.36	BLOB	…6 号
1002	t_2	point	广场	22.816,108.32	BLOB	…大道

Row Key 为行键，POI Inform 是列簇，示例数据（1001，POIInform：name，t_4）和（1002，POIInform:name,t_2）分别存储大厦和广场 POI 所在非结构化数据，HBase 中存储 POI 的非结构化信息，主要通过使用二进制文件 BLOB。

1.3.2　Hbase 存储 POI 非结构化数据策略

二进制文件主要是对数据类型为 blob 的文件进行存储。POI 的 blob 文件主要有对象名字、大小、类型、内容等字段。如果对应的存储文件较小，则可以直接将内容存储在 HBase 中，如果内容较大，需考虑存储文件的所在路径，HBase 通过寻找路径来对路径对应的数据进行存储。

以非结构化数据（图片）存储为例，首先在 HBase 里新建一张大表，用一个单独的列簇存储 POI 的图片内容，其他列簇存储图片信息的大小、名称、类型等其他信息，结构见表 4，图片内容与名称、大小等不属于同一属性，这样存储可以帮助图片的多属性查询操作。Hbase 列簇的动态增加，可以进行扩展，进而存储图片的更多信息，能实现图片相关信息的扩展查询效果。

表 4　Hbase 的 POI 图片存储

行键 Rowkey	时间戳 Timestamp	列簇 Column family	列簇 Column family		
		内容	大小	名称	类型
Rowkey	T1	Content	3M	Name	type

使用 HBase 对 POI 图片存储具有以下优点：①满足 HBase 的列簇设计要求，能够较好地进行列簇的设计，即将图片内容存在一个列簇，将图片大小、内容等信息保存在另外一个列簇，而图片的内容跟大小、名称等不属于同一属性，这样存储能够帮助图片的多属性查询操作；②HBase 列簇能够动态增加，可以对列簇扩展，来保存跟图片相关的信息，能够达到对图片相关消息的扩展查询的效果。

1.4　属性数据与非结构化数据交互

采用关系型数据库 Oracle 和分布式数据库 HBase 这种混合存储策略实现 POI 的结构化属性数据和非结构化数据交互，在实际应用中需进行交互，本文采用 Sqoop 实现交互操作，交互架构如图 1 所示。

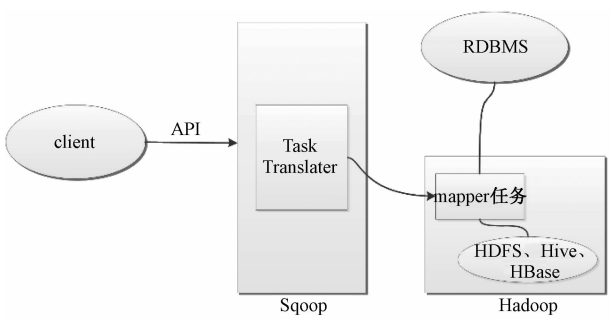


图 1　Sqoop 交互架构

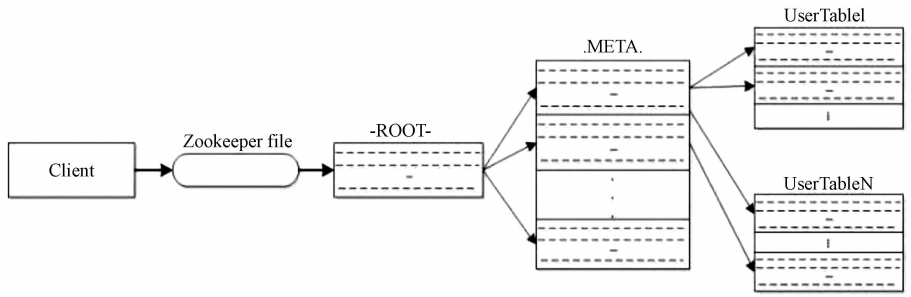
客户端发送请求到 Sqoop，Sqoop 通过 Java 代码调用与请求相关的 API[6-7]，通过任务转换器，将

请求发送到Hadoop框架,并将任务转换成mapreduce命令,通过mapreduce将关系数据库和HBase数据库的信息进行交互。

2　POI检索优化策略

2.1　Hbase的POI检索机制

HBase数据检索是在数据的存储块region所对应的regionserver下完成,用户发出搜索命令到regionserver,regionserver找到对应的region,即找到用户查找的存储信息,这是HBase进行数据检索的核心。而检索到regionserver,需对HBase中的-ROOT-和.META.两张核心表进行研究,其检索策略如图2所示。



META.:记录用户表的存储块的信息,它有多个存储块,以及存储块所对应的RegionServer的服务器地址;
-ROOT-:记录.META.表的存储块信息。

图2　Hbase检索策略

从图1中的表-ROOT-和.META.能在HBase中发现用户进行数据访问的核心流程。首先用户访问zookpeer协调服务,协调服务接收请求后去访问-ROOT-,得到-ROOT-的数据之后,再对.META.表进行访问,最后通过访问-ROOT-和.META.表数据,才能检索到用户需要的数据,找到该数据所对应的用户表[8]。-ROOT-和.META.表结构见表5。

表5　-ROOT-和.META.表结构

行键 rowkey	family:info		
	regioninfor	server	Serverstarcode
rowkey	Startkey,endke,famliy list(family,bloomfilter,compress,blocksize,blockcache)	address	

在表5中,行键rowkey包含表名、startkey和时间戳,3部分用逗号隔开,可表示为tablename,starkey,timestamp;列簇family:info包含基本列regioninfor、sever、severcode,其中regioninfor是存储块的基本信息,包括startkey、endkey和每个family的信息[9-10],Sever存储块对应的RegionServer地址,故存储信息区域操作即是对这两张表操作。用POI样本数据进行实验,先构建.META.表和-ROOT-表,表结构分别见表6和表7。

针对表6,假设HBase中仅有table1和table2,table1较大,分成很多存储块,在.META.表中有需使用很多行来记录信息。而table2较小,仅被分成两个存储块,因此在.META.中只有两行用来存储信息。若从table1里面查询一条rowkey=PK3000的数据,需遵循如下步骤:①从.META里面查询包含PK3000的存储块;②获取存储块的服务地址;③连接服务地址,获取数据[11-12]。

表6　Hbase存储POI之.META.结构

行键 rowkey	family:info			
	regioninfor	video	server	serverstarcode
table1 RK1000,t1	XX市信息化大楼	blob	RS1	
table1 RK2000,t1	XX旅行社	blob	RS2	
table1 RK3000,t1	XX市政府	blob	RS3	
…	…	…	…	…
table2 RK0000,t1		blob	RS1	
table2 RK1234,t1		blob	RS2	

表7　Hbase存储POI之-ROOT-表结构

行键 rowkey	family:info			
	regioninfor	video	server	serverstarcode
.META.,table1 RK3000,t1,t2	XX市政府	blob	RS1	
.META.,table2 RK1234,t1,t2		blob	RS2	

针对表7,同样假设Hbase只有两种表table1和table2,由于-ROOT-是记录.META.的信息,其结构设计为表6。假设在table1里查找rowkey=PK3000的数据,需要进行递归调用:获取table1中行键为RK30000的存储块所对应的服务=>获取.META.表,行键为table1 RK3000,t1的存储块所

对应的服务＝＞获取-ROOT-，行键为 table1 RK3000，t1，t2 的存储块所对应的服务＝＞Zoo-Keeper 得到-ROOT-的存储块所对应的服务，并查找行键为. META.，table1，RK3000 的一条 Row，得到. META. 的存储块所对应的服务 ＝＞从 .META. 表中查到行键为 table1 的 RK3000 的存储块，得到 table1 存储块所对应的服务＝＞从 table1 中查到 RK3000 的存储记录，即得到结果。

2.2 POI 数据检索策略

为实现 POI 数据的准确快速检索，本文使用 Solr 进行优化，同时针对非结构化数据的检索创建二级索引实现优化。Solr 可定义接口，实现全量更新和增量更新，且 Solr 自带缓存功能，在服务器存储经常搜索的 POI，下一次触发时，无须再进入数据库查询，其次，Solr 是基于 Lucene 操作的，检索效率较高。因本文 POI 存储涉及到关系型数据库 Oracle 和分布式数据库 Hbase，需分别对两者创建索引。

2.2.1 关系型数据库索引创建

本文研究的 POI 数据除经纬度坐标之外，其他均为属性信息，因此对于样本数据可以考虑使用 Solr 对 Oracle 数据库建立索引。使用 Solr 相关文件配置，将保存在 Oracle 的数据库导入 Solr，完成 POI 全文索引的创建，然后使用 Java 语言，查询 Solr 中的数据库。

2.2.2 非结构化数据索引创建

针对单条件搜索，只需使用行键 rowkey 进行即可，较为简单，本文主要针对多条件搜索进行优化。Hbase 数据库能同时存储 POI 的非结构化数据和属性数据，针对多条件搜索，需建立二级索引。因此使用 HBase 自带的协处理器生成索引，基于 Redis 的 HBase 的二级索引的设计，通过协处理器，将索引存为一张表[13-14]。

2.2.2.1 基于 Redis 的 HBase 的二级索引

该二级索引结构主要包括数据处理、二级索引和客户端三大模块。当用户发出数据的增删改查操作时，HBase 数据处理模块接受用户请求，分别对请求进行处理，同样，二级索引模块根据客户端请求进行二级索引的增删改查。基于 Redis 实现 HBase 二级索引，协处理器相当于一个回调函数，协处理器通过配置可以实现作用到 HBase 中所有表，也可以单独指定作用到一张表。当用户操作 get/put/delete 数据时，HBase 中的 regionserver 会在 get/put/delete 回调一个实现协处理器类中的方法，方法中可以获得 get/put/delete 的数据，从而实现构建二级索引的操作[15]。

2.2.2.2 HBase 二级索引数据结构研究

POI 多条件搜索，即 POI 非 rowke 搜索，包括 POI 精确查询和范围查询，通过 Redis set 和 sorted set 实现二级索引的创建，以达到较高的搜索效率。以该索引创建思路，以表 8 POI 样本数据为例，对 POI 的 tel 创建索引。

表 8 POI 样本数据

rowkey	POI:qualifier	value
rowkey1	POI1:name	范记饼屋
rowkey2	POI1:tel	0771－3074225
rowkey3	POI2:name	玲莉快餐总店
rowkey4	POI2:tel	0771－5883775

POI 精确查询，使用 Redis 的 set 类型建立二级索引，结构为 key-value，key 为列名_值，value 为 rowkey，为(key，value)添加二级索引。生成的二级索引见表 9。

表 9 Redis 中的 set 建立二级索引

key	value
tel_0771－3074225	rowkey2
tel_0771－5883775	rowkey4

POI 范围查询，使用 Redis 的 sorted set 类型进行建立二级索引，结构为 key-score-value，key 为列名，score 为列值，value 为列值_rowkey。为(key，score，value)添加二级索引。生成的二级索引，见表 10。

表 10 Redis 中的 sorted set 建立二级索引

key	score	value
Tel	0771－3074225	0771－5883775_ rowkey2
Tel	0771－5883775	0771－5883775_ rowkey4

2.2.2.3 HBase 二级索引优化实验

在 HBase 和 Redis 系统、JDK1.7.0_55、Hadoop 1.1.2、HBase 0.96.2、Redis 3.2.1 的实验环境下，对 POI 样本数据建立 Hbase 二级索引，用 Hbase 系统查询 POI 数据，使用查询语句对 Hbase 查询进行性能测试，结果见表 11。

表 11 Hbase 二级索引测试结果

查询条数/万条	5	50	100
无二级索引时间/ms	4 768.3	5 546.6	6 023.7
有二级索引时间/ms	356.1	476.6	537.6

根据测试结果可知,在实现二级索引的情况下,HBase查询性能极大提升。这是因为HBase在执行基于rowkey的查询时,查询速率很高,而在建立二级索引的情况,系统会将对非rowkey的查询,通过查询Redis中的二级索引获得符合条件的rowkey,再通rowkey在HBase中获得数据集,间接地将基于非rowkey的查询转化为基于rowkey的查询,获得较高的查询效率。

2.3　Solr检索优化

通过建立二级索引可以明显提升检索效率,同时在POI搜索引擎上,嵌入Solr的全文的搜索引擎,增强检索优势。Solr可实现全量更新,与Oracle形成互补关系。把数据导入到Solr,比如将多个表的数据全部导入Solr,由于Solr被理解成一张很大的表,故多个表的数据相当于保存在一张表,一张表建立索引,查询数据效率就会高很多;且Solr自带缓存功能,某些POI被搜索的频率较高,Solr会自动将这些数据进行缓存,当用户对该POI二次搜索时,可直接从缓存库中将搜索结果展现出来,不再需要对数据库进行遍历,从而提升检索效率,且Solr索引和全文搜索可以实现更多的搜索功能,增强Oracle的查询。

Solr增量导入Oracle数据技术,首先需要配置数据结构配置文件schema.xml、数据库配置文件oracle-poi.xml和数据库实体配置文件solrconfig.xml,然后进行dataimport操作,将oracle数据导入Solr中。通过java调用solr接口,将保存在oracle的POI样本数据保存在服务器上,来完成POI的搜索功能。在搜索引擎中对POI非结构化数据检索,主要代码实现类如图3所示。

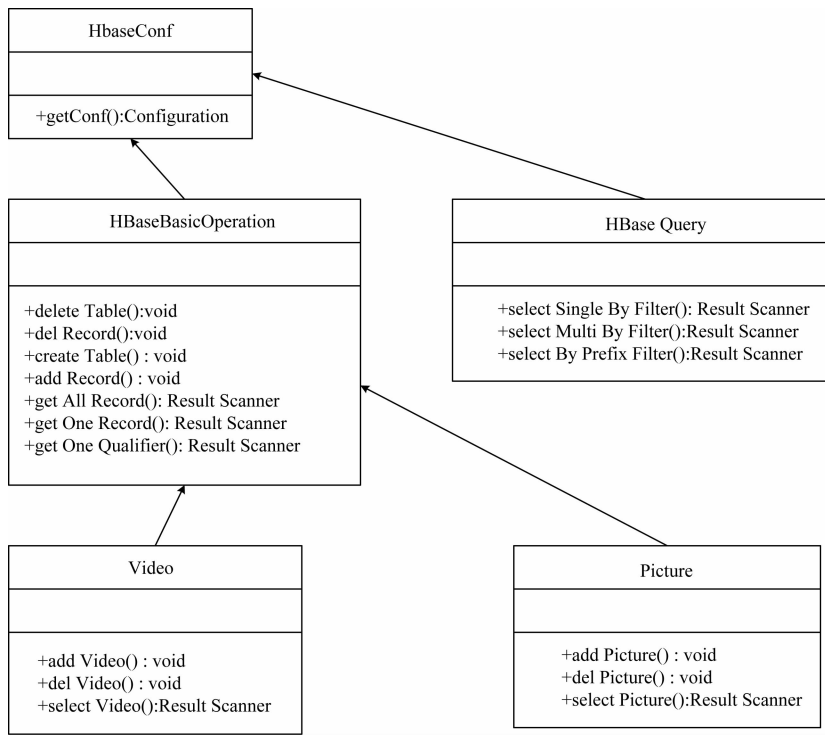


图3　POI操作类图

3　POI非结构化数据检索实验分析

本文采用1 000～100 000个POI非结构化文件测试数据,文件大小从20.5 G到3 T,数据大小见表12。

表12　POI非结构化数据测试数据大小

POI视频文件量/个	POI视频文件大小
1 000	20.5G
5 000	105.75G
10 000	223.54G
50 000	1.21T
100 000	3T

将其POI信息分别存储在Oracle集群中和HBase分布式数据库中,然后输入相同的搜索条件,进行POI搜索,得到具体搜索条件见表13。

表13　检索POI非结构化数据时间对比

视频文件量/个	时间/ms	
	Oracle	HBase
1 000	895	976
5 000	4 566	4 773
10 000	11 851	9 679
50 000	91 590	48 990
100 000	246 498	99 120

在 POI 数据量不大的情况下，采用关系型数据库与分布式数据库 HBase 进行数据检索差别不大，查询时间在同一量级上。但是随着 POI 数据量的增多，采用 HBase 进行 POI 非结构化数据查询时，检索效率明显比关系型数据库快得多，为直观对比，制作折线图(图 4)。经过实验验证，可以得到使用 HBase 进行 POI 非结构化数据的存储管理的方法是可行的且是正确高效的。

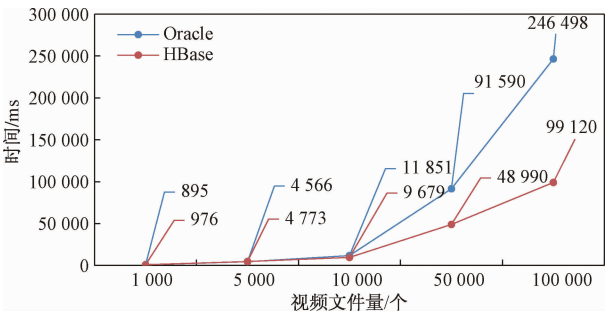


图 4　Oracle 与 HBase 检索耗时对比

4　结论

通过研究 POI 数据特点，建立 POI 数据模型，提出 POI 结构化属性数据和非结构化数据的混合存储策略；针对不同的存储方式，采用关系型数据库和 HBase 检索数据的方法，对 HBase 建立二级索引，得出建立二级索引的优点。最后嵌入 Solr 的全文搜索引擎，使“全量更新”与 Oracle 形成互补的关系；Solr 索引和全文搜索可以实现更多的搜索功能，增强 Oracle 的查询。实验结果表明，在使用本文的存储和检索方案之后，对 POI 数据的检索效率有所提高。

参考文献

[1]　辜寄蓉，吴修月，黄志勤，等. POI 数据辅助下的泸州市城市地块用地类型现状快速判断与验证[J]. 中国农业资源与区划，2019，40(11)：72-79.

[2]　蒋奎，陈亮. 一种基于关系数据库管理系统的图计算平台[J]. 华东师范大学学报(自然科学版)，2016(5)：103-111.

[3]　缪燕，孙燕，陈晓娟，等. 多源对象关系数据库细粒度强制访问控制机制实现方法[J]. 计算机应用与软件，2021，38(12)：14-21，55.

[4]　庞云璇. 空间数据管理面向对象数据库技术探究[J]. 电脑编程技巧与维护，2019(3)：108-109，138.

[5]　黄建伟. 可扩展数据库系统数据完整性约束的实现与优化[D]. 上海：华东师范大学，2019.

[6]　CARSTOIU D，LEPADATU E，GASPAR M. HBase-non sql database，performances evaluation[J]. International Journal of Advancements in Computing Technology，2010，2(5)：42-52.

[7]　LIZHI C，SHIDONG H，LEILEI C，et al. Performance analysis and testing of HBase based on its architecture[C]// Computer and Information Science (ICIS)，2013IEEE/ACIS 12th International Conference on Niigata. NewJersey：IEEE，2013：353-358.

[8]　GEORGE L. HBase：the definitive guide[M]. O，Reily Media，Inc.，2011.

[9]　瞿龙俊. 基于 HBase 的交通流数据实时存储与查询优化方案的设计与实现[D]. 镇江：江苏大学，2017.

[10]　邹敏昊. 基于 Lucene 的 HBase 全文检索功能的设计与实现[D]. 南京：南京大学，2013.

[11]　伍祈应. 云环境中可搜索加密关键技术研究[D]. 西安：西安电子科技大学，2018.

[12]　张榆，马友忠，孟小峰. 一种基于 HBase 的高效空间关键字查询策略[J] 小型微型计算机系统，2012(10)2141-2146.

[13]　王飞，张明，晏学义. 基于位图技术创建 HBASE 二级索引的实践[J]. 信息通信技术，2021，15(3)：78-82.

[14]　崔丹，史金鑫. 基于 Redis 实现 HBase 二级索引的方法[J]. 软件，2016，37(11)：64-67.

[15]　ZENG C Y，LI J X. Application of Redis in caching system[J]. Microcomputer and Its Applications，2013，32(12)：11-13.

The Storage and Retrieval Strategy Structured and Unstructured Data of POI

WU Shan，ZHAO Jian，LIU Dandan，YU Jiangshun，WU Sirui

(POWERCHINA Guizhou Electric Power Design & Research Institute Co.，ltd.，Guiyang 550002，China)

Abstract：POI data has both structured and unstructured attributes. When relational database storage is used，unstructured storage occupies large memory space，The data is directly retrieved from the database. In addition，the HBase rowkey is used for multi-condition retrieval of POI space information，which is inefficient. By studying the characteristics of POI data and building a data model，Different storage and retrieval strategies are proposed for structured attribute data and unstructured data of POI and create index. Finally，Sqoop is used to complete the interaction between POI attribute data and unstructured data，and Solr technology is embedded for retrieval. Experimental results show that the proposed method can improve the efficiency of POI unstructured data retrieval.

Keywords：POI；HBase；Solr；POI storage；POI search