使用 MongoDB 整理 OpenStreetMap 数据

杨东

地图位置： 中国上海市

https://www.openstreetmap.org/relation/913067
https://mapzen.com/data/metro-extracts/metro/shanghai_china/

选择原因： 兴趣

1. 在地图中遇到的问题

- name：tag 标签中 name 属性和 official_name 属性有各种语言版本
- postcode： 极少数 postcode 格式不对（不是 6 位数字）
- phone： 只有很少的数据有 phone 属性，并且格式不统一
- 大部分 node 或 way 都没有 name 属性

对于 tag 标签中 name 的处理：

1) 提取 name 属性
2) 对所有 'name:' 属性，添加到集合，转成列表，添加到 names 字段

对 official name 也做同样处理。

对 postcode、phone 未做处理（后面也没有相关查询）

osm 文件分析

经过简单分析发现以下几点：

1. 标签只有 osm/bounds/node/way/relation/tag/nd/member 这几种标签
2. node/way/relation 是主要数据，且每个标签都有 id/uid/user/timestamp/changeset/version 这几个属性，意味着每条数据都是由用户添加更新维护的。

2. 数据概述

文件大小

shanghai_china.osm 717M

shanghai_china.osm.json 1.04G

document 总数

```
> db.shanghai.count()
3993306
```

node 数量

```
> db.shanghai.find({'type': 'node'}).count()
3548777
```

way 数量

```
> db.shanghai.find({'type': 'way'}).count()
444529
```

有 name 的 node 数量

```
> db.shanghai.find({'type': 'node', 'name': {'$exists': 1}}).count()
20108
```

仅占 node 总量的 0.57%

有 name 的 way 数量

```
> db.shanghai.find({'type': 'way', 'name': {'$exists': 1}}).count()
78437
```

占 way 总量的 17.6% 。 way 的名字也比 node 的名字多。

便利设施（amenity）类型和数量

```
> db.shanghai.aggregate([
    {'$match': {'tags.amenity': {'$exists': 1}}},
    {'$group': {
        '_id': '$tags.amenity',
        'count': {'$sum': 1}
    }},
    {'$sort': {'count': -1}},
    {'$limit': 10}
])
```

```
{ "_id" : "bicycle_rental", "count" : 2590 }
{ "_id" : "school", "count" : 1502 }
{ "_id" : "parking", "count" : 1439 }
{ "_id" : "restaurant", "count" : 1393 }
{ "_id" : "bank", "count" : 639 }
{ "_id" : "toilets", "count" : 487 }
{ "_id" : "cafe", "count" : 411 }
```

```
{ "_id" : "fuel", "count" : 399 }
{ "_id" : "fast_food", "count" : 384 }
{ "_id" : "hospital", "count" : 357 }
```

可以看出 bicycle_rental 比较多， 可能是因为共享单车现在比较流行。

用户总数（去重）

```
> db.shanghai.distinct('created.user').length
2340
```

贡献前 5 的用户

```
> db.shanghai.aggregate([
   {'$group': {
      '_id': {'uid': '$created.uid', 'user': '$created.user'},
      'count': {'$sum': 1}
   }},
   {'$sort': {'count': -1}},
   {'$limit': 5}
])

{ "_id" : { "uid" : "288524", "user" : "Chen Jia" }, "count" : 701693 }
{ "_id" : { "uid" : "110639", "user" : "aighes" }, "count" : 181417 }
{ "_id" : { "uid" : "2215592", "user" : "xiaotu" }, "count" : 176522 }
{ "_id" : { "uid" : "17497", "user" : "katpatuka" }, "count" : 137910 }
{ "_id" : { "uid" : "42537", "user" : "XBear" }, "count" : 125762 }
```

只贡献过 1 次的用户数量

```
> db.shanghai.aggregate([
   {'$group': {
      '_id': {'uid': '$created.uid', 'user': '$created.user'},
      'count': {'$sum': 1}
   }},
   {'$match': {'count': 1}},
   {'$group': {
      '_id': 'number_of_users_contribute_once',
      'count': {'$sum': 1}
   }},
])

{ "_id" : "number_of_users_contribute_once", "count" : 486 }
```

3. 其他想法

年度贡献用户分析

基本上每条数据都对应了一个用户。对用户也有很多信息可以挖掘。例如，已经知道贡献最多的用户是 "Chen Jia"，贡献了 701693 条数据。但是每年都是这个用户贡献最多么？年度贡献最多的用户都是谁？占比多少？

```
> db.shanghai.aggregate([
    {'$group': {
        '_id': {
            'year': {'$substr': ['$created.timestamp', 0, 4]},
            'uid': '$created.uid',
            'user': '$created.user'
        },
        'count': {'$sum': 1}
    }},
    {'$group': {
        '_id': '$_id.year',
        'max': {'$max': '$count'},
        'total': {'$sum': '$count'},
        'user': {'$push': {'uid': '$_id.uid', 'user': '$_id.user', 'count':
'$count'}},
    }},
    {'$unwind': '$user'},
    {'$project': {
        '_id': '$_id',
        'count': '$user.count',
        '_diff': {'$subtract': ['$user.count', '$max']},
        'ratio': {'$divide': ['$user.count', '$total']},
        'uid': '$user.uid',
        'total': '$total',
        'user': '$user.user',
    }},
    {'$match': {'_diff': 0}},
    {'$sort': {'_id': -1}}
])
```

分析：

1. group：按创建日期、uid、user 分组计数，得到每年每个用户的贡献数量
2. group：按年分组计数，并将 user 聚合到数组，得到年度贡献总数、用户最多贡献数
3. unwind：将 user 展开
4. project：计算： user 贡献数与最多贡献数的差，与年度贡献总数的比值

5. match：筛选出 差为 0 的用户，即为最多贡献用户（不能直接筛出 $max ）
6. sort： 按年排序

{ "_id" : "2017", "count" : 112202, "_diff" : 0, "ratio" : 0.1647500014683339, "uid" : "288524", "total" : 681044, "user" : "Chen Jia" }
{ "_id" : "2016", "count" : 388062, "_diff" : 0, "ratio" : 0.3748732107011134, "uid" : "288524", "total" : 1035182, "user" : "Chen Jia" }
{ "_id" : "2015", "count" : 123936, "_diff" : 0, "ratio" : 0.16964520367936925, "uid" : "288524", "total" : 730560, "user" : "Chen Jia" }
{ "_id" : "2014", "count" : 90463, "_diff" : 0, "ratio" : 0.17569349629339026, "uid" : "42537", "total" : 514891, "user" : "XBear" }
{ "_id" : "2013", "count" : 59996, "_diff" : 0, "ratio" : 0.1847901414046835, "uid" : "1436097", "total" : 324671, "user" : "Holywindon" }
{ "_id" : "2012", "count" : 39627, "_diff" : 0, "ratio" : 0.1143923582544535, "uid" : "527986", "total" : 346413, "user" : "zzcolin" }
{ "_id" : "2011", "count" : 52390, "_diff" : 0, "ratio" : 0.3285669488867984, "uid" : "304705", "total" : 159450, "user" : "yangfl" }
{ "_id" : "2010", "count" : 33461, "_diff" : 0, "ratio" : 0.4488698101817694, "uid" : "17497", "total" : 74545, "user" : "katpatuka" }
{ "_id" : "2009", "count" : 13121, "_diff" : 0, "ratio" : 0.353579994071519, "uid" : "5553", "total" : 37109, "user" : "dkt" }
{ "_id" : "2008", "count" : 14965, "_diff" : 0, "ratio" : 0.8126086012163336, "uid" : "5553", "total" : 18416, "user" : "dkt" }
{ "_id" : "2007", "count" : 68062, "_diff" : 0, "ratio" : 0.958282294966561, "uid" : "5553", "total" : 71025, "user" : "dkt" }

从上面数据可以得出结论：

1. 2015~2017 年都是 ChenJia 贡献数量最多，但 2015 年之前不是。最早是 dkt 贡献最多（2007 ~ 2009）。
2. 虽然 ChenJia 贡献数量较多，但占比较小（17.0%，37.5%，16.5%），说明其他贡献者也贡献了很多数据。最早的贡献者 dkt 统计占当年相当大的比重（95.8%，81.3%，35.3%）。
3. 2015~2017 年贡献的总量也较多，说明数据贡献者从 2015 年开始就比较活跃了。

解决思路参考：
https://stackoverflow.com/questions/33361697/get-all-the-documents-having-max-value-using-aggregation-in-mongodb

数据改进建议

经过进一步数据探查，可以发现：

- 大部分 node 节点除了基本信息（id/uid/user/timestamp/changeset/version/lat/lon）以外，没有其他任何信息，这个比例是： 3370676 / 3548777 ≈ 95%，只有很少的 node 有其它信息。不太清楚这 95%的数据是如何提交的，但这部分数据的统计分析意义并不大。
- 相反，大部分 way 节点除了基本信息外，都包含了其他信息 （440166/444529 ≈ 99%）。可以说，way 数据的质量较高。

如果将只有基本信息的 node 节点全部过滤掉， 并不会对一些具体的分析产生影响（比如 amenity 公共设施类型和数量）， 仅对贡献用户的统计分析产生影响。 例如，对过滤后的 node 节点（仅 178101 个），统计年度贡献最多用户，会得出不一样的结论：

Year	User	Count	TotalUseful	Total	Ratio (Count/TotalUserful)
2017	Aurimas Fišeras	3054	11803	584556	0.258
2016	yangfl	4174	22880	897966	0.182
2015	42429	5819	18806	641466	0.309
2014	42429	6630	14914	462015	0.444
2013	Toliman	1827	7719	301718	0.236
2012	yangfl	10725	19638	317187	0.546
2011	yangfl	9251	13425	149793	0.689
2010	Xylem	473	1028	71892	0.460
2009	dkt	1003	1374	35320	0.729
2008	dkt	557	1076	16742	0.517
2007	dkt	63594	65438	70122	0.971

大部分有用 node 还是 dkt 最早在 2007 年贡献的。早期数据质量较高，后来的 node 就不知道是怎么提交的了，猜测只是上传了一个 GPS 信息，没有任何附加信息。

所以将这些只有基本信息的 node 节点都删掉，并不会影响到一些具体的分析，并能减少很大一部分数据，减少数据库的负担。

数据改进建议的其他想法

删掉 node 节点会不会影响 way 中的 node_refs，需要进一步研究证明。

4. 总结

mongodb 的 document 没有固定的 schema，相对于 关系型数据库来说，建表、读写、修改更灵活。 但在一些复杂查询上，灵活性却不如关系型数据库查询。例如在聚合操作的 pipeline 中，只是对一个 collection 的单线操作， 就不如 SQL 中各种子查询的组合灵活。

数据中还有很多有趣的信息，因时间和精力原因不再深入挖掘。经过此项目，已掌握数据清洗、审计的基本方法，并学会使用 MongoDB 的一些基本查询、聚合操作。

附：

1. 代码清单

CaseStudy 练习（Mongodb）	quiz_mapparser.py	Quiz3：迭代解析
	quiz_tags.py	Quiz7：标签类型
	quiz_users.py	Quiz8：探索用户
	quiz_audit.py	Quiz11：完善街道名
	quiz_data.py	Quiz12：准备数据库-MongoDB
	example.osm	
OSM 项目	osm_1_analysis.py	对 osm xml 文档的初步分析
	osm_2_process.py	将 osm 清洗，转成 json 格式文件
	osm_3_mongo.py	对 mongodb 的一些查询操作
	shanghai_china.sample.osm	osm 样本文件
	sample.py	osm 采样脚本