第二次上机实验报告

MongoDB

`MongoDB` 中的数据存储方式如下。将 `S` `P` `O` 作为一组键值同时存入。

```
def mdb_store(file, names, db, sep):
    with open(file) as f:
        for line in f:
            line = line.strip('\n')
            db.insert_one(make_dict(names, line.split(sep)))
```

查询一 给定一个si，给出它所有的P和O

```
rs = lord.find({'S': S})
    for tmp in rs:
        print((tmp['P'], tmp['O']))
```

查询二 给定一个oi, 给出它所有的S和P

```
rs = lord.find({'O': O})
   for tmp in rs:
       print((tmp['P'], tmp['O']))
```

查询三 给定两个p1,p2, 给出同时拥有它们的S

```
rs1 = lord.find({'P': p1})
rs2 = lord.find({'P': p2})
result = set([i['S'] for i in rs1]).
        intersection(set([i['S'] for i in rs2]))
print(result)
```

这里的处理方式是先将包含 `p1` `p2` 元组分别查，再进行集合求交操作

查询四 给定一个oi, 给出拥有这样oi最多的S

```
rs = lord.find({'O': O})
```

```
d = defaultdict(int)
for tmp in rs:
    d[tmp['S']] += 1
print(sorted(d.keys(), key=lambda obj:d[obj], reverse=True)[0]
if len(d) > 0 else None)
```

这里的处理方式是，先将包含 `O` 的元组全部找出，再利用字典计数后排序找出包含 `O` 最多的 `S`

Redis

`Redis` 的存储方式如下，为了方便查询，对于每一个元组 `<s, p, o>`

- 在以 `s` 为键值的集合中，将 `<p, o>` 存入 `s` 集合
- 在以 `o` 为键值的集合中，将 `<s, p>` 存入 `o` 集合
- 在以 `p` 为键值的集合中，将 `<s>` 存入 `p` 集合
- 在以 `o` 为键值的哈希表中，对哈希表的 `s` 位置进行递增计数

```
def redis_store(file, r, sep):
    with open(file, 'r') as f:
        for line in f:
            line = line.strip('\n')
            s, p, o = line.split(' ')
            r.sadd(s, '{} {}'.format(p,o))
            r.sadd(o, '{} {}'.format(s,p))
            r.sadd(p, s)
            r.hsetnx(o+'hash', s, 0) # key-hash表，键值key不能重复
            r.hincrby(o+'hash', s, 1) # 递增计数
```

查询一 给定一个si，给出它所有的P和O

```
rs = r.smembers(arg_list[0])
print('有S，查所有P，O：')
for i in rs:
    print(i)
```

查询二 给定一个oi, 给出它所有的S和P

```
rs = r.smembers(arg_list[1])
print('有O，查S，P：')
```

```
for i in rs:
    print(i)
```

查询三 给定两个p1,p2, 给出同时拥有它们的S

利用 `redis` 的 `sinter()` 操作求交

```
p1, p2 = arg_list[2], arg_list[3]
rs = r.sinter(p1, p2)
print('给出p1，p2，查同时拥有它们的S：')
for i in rs:
    print(i)
```

查询四 给定一个oi, 给出拥有这样oi最多的S

```
rs = r.hgetall(arg_list[4]+'hash')
print('给定O，查拥有O最多的S：')
result = sorted(rs.keys(), key=lambda obj:rs[obj], reverse=True)[0]
        if len(rs)>0 else None
print(result)
print(rs.get(result, None))
```

Cassandra

`Cassandra` 中的数据存储方式如下，将 `<id, s, p ,o>` 作为一个记录存入数据库， `id` 为主键

```
def cassandra_store(file, session, sep):
    key_string = '(id,S1,P1,O1)'
    id = 0
    with open(file) as f:
        for line in f:
            line = line.strip('\n')
            id += 1
            s, p, o = line.split(' ')
            values_string =
                "({},'{}','{}','{}')".format(id,s,p,o)
            insert = "insert into bigdata.lord" + key_string +
                "values" + values_string
            session.execute(insert)
```

查询一 给定一个si，给出它所有的P和O

需要 `allow filtering` 选项否则查询无法进行

```
rs = session.execute("select * from bigdata.lord
        where S1='{}' allow filtering".format(arg_list[0]))
print('查询1：')
for i in rs:
    print('{},{}'.format(i.s1, i.o1))
```

查询二 给定一个oi, 给出它所有的S和P

```
rs = session.execute("select * from bigdata.lord
        where O1='{}' allow filtering".format(arg_list[1]))
print('查询2：')
for i in rs:
    print('{},{}'.format(i.s1, i.p1))
```

查询三 给定两个p1,p2, 给出同时拥有它们的S

```
rs1 = session.execute("select * from bigdata.lord
        where P1='{}' allow filtering".format(arg_list[2]))
rs2 = session.execute("select * from bigdata.lord
        where P1='{}' allow filtering".format(arg_list[3]))
result_set = set(rs1).intersection(set(rs2))
print('查询3：')
for i in result_set:
    print(i.s1)
```

查询四 给定一个oi, 给出拥有这样oi最多的S

```
rs = session.execute("select * from bigdata.lord
        where O1='{}' allow filtering".format(arg_list[4]))
d = defaultdict(int)
for i in rs:
    d[i.s1] += 1
result = sorted(d.keys(), key=lambda obj:d[obj], reverse=True)[0]
        if len(d)>0 else None
print('查拥有 {} 最多的S：{},拥有:{}'.
        format(arg_list[4], result, d.get(result, None)))
```