2023 大数据管理课程任务书

一、实验软件和数据

1. 数据库：图数据库 Neo4j，关系数据库 MySQL，文档数据库 MongoDB。
2. 编程语言：java
3. 数据集：Yelp Dataset

二、软件安装

请参照华为平台部署手册及实验指导书。

三、实验任务

任务 1：MySQL for JSON 实验

a) . JSON 基本查询:

1. 在 business 表中,查询 city 位于 Tampa 的商户所有信息,按被评论数降序排序,限制返回 10 条.
2. 在 business 表中,查询前五条记录的 business_info 列和 business_info 中 attributes 的所有键,以 Json 数组形式返回,同时返回对应键的数量.
3. 在 business 表中,查询该表 business_info 列中 name, stars, attributes 的内容和 JSON 类型,限制返回行数为 5.
4. 在 business 表中,查询拥有电视("HasTV"是"True")且星期天不营业(Sunday 键不存在,当然也可以 hours 键不存在,即 is null)的商户,返回它的名字(不带双引号),属性,营业时间,并按名字升序排序,限制 10 条记录.
5. 使用 explain 查看 select * from user where user_info->'$.name'='Wanda'的执行计划,其中执行计划按 JSON 格式输出;并且实际执行一次该查询,请注意观察语句消耗的时间并与 MongoDB 的查询方式进行对比(MongoDB 要执行此查询要求,相应的语句是什么?执行计划是怎样的?并给出查询效率对比).

b) . JSON 增删改:

6. 在 busines 表中,查询 id 为 4r3Ck65DCG1T6gpWodPyrg 的商户的 business_info, 这里对 info 列的显示需要使用 JSON_PRETTY(business_info)让可读性更高, 然后在它的 hours 中新增"Tuesday":"16:0-23:0"的键值对,并将其评分改为 4.5, 属性的'WiFi'对应的值置为"Free", 返回其 business_info, 也用 JSON_PRETTY(business_info)展示在修改前后的差异.
7. 向 business 表插入一个 id 是'aaaaaabbbbbbccccccc2023'的商户,其商户信息与 id 为'5d-fkQteaqO6CSCqS5q4rw'的商户完全一样,插入完成之后,将这个新记录的 info 中的 name 键值对删去,最后查询'aaaaaabbbbbbccccccc2023'的所有信息.

c) . JSON 聚合:

8. 在 business 表的所有商户中,按所在州(state)进行聚合,对于每个州返还一个 JSON 对象,这个对象的每一个键值对中,key 是城市,value 是城市总共出现的次数,结果按照州名升序排序.

   提示:这里需要去掉引号让 group by 的 key 更好一些.
9. 查询 user_id 是'__1cb6cwl3uAbMTK3xaGbg'的用户的所有朋友的建议,并按用户进行分组聚合,对每一个用户,返还用户的 id,用户的名字,由他/她的所有建议构成的字符串数组,最后按名字升序排序输出.

   提示:由于 user 表的 friends 存的是由 id 组成的字符串,而不是字符串数组,这里可以使用 REGEXP_LIKE()函数进行模式匹配.

d) . JSON 实用函数的使用:

10. 在 business 表中,分别查询城市在 EdMonton 和 Elsmere 的商铺,并使用 JSON_OVERLAPS()判断在这两个城市的商铺之间是否在一周中至少有一天营业时间完全重合(即 hours 对象中至少有一个键值对相同,不考虑键不存在的情况),是返回 1,不是返回 0.
11. 在 user 表中,查询 funny 大于 2000 且平均评分大于 4.0 的用户,返回他们的名字,平均评分,以及按 json 数组形式表示的 funny, useful, cool,三者的和,限制 10 条;尝试按平均评分降序排序,并使用 explain 查看排序的开销,与第一题的排序情况做对比,主要关注"rows_examined_per_scan"和"cost_info".
12. 在 tip 表中找到被提建议最多的商户和提出建议最多的用户,合并二者的 info

列的 JSON 文档为一个文档显示,对于 JSON 文档中相同的 key 值,应该保留二者的 value 值.

13. 查询被评论数前三的商户,使用 JSON_TABLE()导出他们的名字,被评论数,是否在星期二营业(即"hours"中有"Tuesday"的键值对,是返回 1,不是返回 0),和一周所有的营业时段(不考虑顺序,一个时段就对应一行,对每个商户,从 1 开始对这些时段递增编号), 最后按商户名字升序排序.

任务 2：MongoDB 实验

a)．条件查询与执行计划:

1. 查询 review 集合的 2 条数据，跳过前 6 条。
2. 查询 business 集合中 city 是 Las Vegas 的 5 条数据。
3. 查询 user 集合中 name 是 Steve 的 user，只需要返回 useful 和 cool,限制 10 条数据。
4. 查询 user 集合中 funny 位于[66，67，68]的 user，只需返回 name 和 funny，限制 20 条数据。
5. 查询 user 集合中 15≤cool<20 且 useful≥50 的 user，限制 10 条。
6. 统计 business 一共有多少条数据，并使用 explain 查询执行计划，了解 MongoDB 对集函数的执行方式。
7. 查询 business 集合 city 为 Westlake 或者 Calgary 的数据。
8. 查询 business 集合中，categories 为 6 种的商户`信息，显示这 6 种类别，限制 10 条。
9. 使用 explain 看 db.business.find({business_id: "5JucpCfHZltJh5r1JabjDg"})的执行计划，了解该查询的执行计划及查询执行时间，并给出物理优化手段，以提高查询性能，通过优化前后的性能对比展现优化程度。

b)．聚合与索引:

10. 统计各个星级的商店的个数，返回星级数和商家总数，按照星级降序排列。
11. 创建一个 review 的子集合 Subreview(取 review 的前五十万条数据)，分别对评论的内容建立全文索引，对 useful 建立升序索引，然后查询评价的内容中包含关键词 delicious 且 useful 大于 9 的评价。插入数据过程耗时约 150s，建

索引耗时约 60s。

12. 在 Subreview 集合中统计评价中 useful、funny 和 cool 都大于 6 的商家，返回商家 id 及平均打星，并按商家 id 降序排列。
13. 查询距离商家 xvX2CttrVhyG2z1dFg_0xw(business_ id) 100 米以内的商家，只需要返回商家名字，地址和星级。

    提示：使用 2dsphere 建立索引、获取商家地理坐标、使用坐标进行查询
14. 在集合 Subreview 上建立索引，统计出用户从 2017 年开始发出的评价有多少，按照评价次数降序排序，需要返回用户 id 和评价总次数，只显示前 20 条结果。

c). MapReduce 的使用:

15. 使用 map reduce 计算每个商家的评价的平均分（建议在 Subreview 集合上做，review 过于大）,不要直接使用聚合函数。

任务 3：Neo4j 实验

1. 查询标签是 CityNode 的节点，限制 10 个。
2. 查询城市是 Ambridge 的商家节点。
3. 查询 reviewid 是 rEITo90tpyKmEfNDp3Ou3A 对应的 bussiness 信息。
4. 查询评价过 businessid 是 fyJAqmweGm8VXnpU4CWGNw 商家的用户的名字和粉丝数。
5. 查询被 userid 为 TEtzbpgA2BFBrC0y0sCbfw 的用户评论为 5 星的商家名称和地址。
6. 查询商家名及对应的星级和地址，按照星级降序排序（限制 15 条）。
7. 使用 where 查询粉丝数大于 200 的用户的名字和粉丝数（限制 10 条）。
8. 查询 businessid 是 tyjquHslrAuF5EUejbPfrw 商家包含的种类数,并使用 PROFILE 查看执行计划，进行说明。
9. 查询 businessid 是 tyjquHslrAuF5EUejbPfrw 商家包含的种类,以 list 的形式返回。
10. 查询 Allison 的朋友（直接相邻）分别有多少位朋友。(考察：使用 with 传递查询结果到后续的处理)

11. 查询拥有类别为 Salad 的商家数量前 5 的城市，返回城市名称和商家数量。
12. 查询商家名重复次数前 10 的商家名及其次数。
13. 统计评价数大于 5000 的商家名热度（名字的重复的次数在所有的商家名中的占比），按照评价数量排序，返回热度和商家名和评价数。
14. 查询具有评分为 5.0 的 Zoos 类别的商铺所在城市。
15. 统计每个商家被多少个不同用户评论过，按照此数量降序排列，返回商家 id，商家名和此商家被多少个不同用户评论过，结果限制 10 条记录。
16. 体会建立索引对查询带来的性能提升，但会导致插入，删除等操作变慢（需要额外维护索引代价）。
17. 查询与用户 user1（userid: tvZKPah2u9G9dFBg5GT0eg) 不是朋友关系的用户中和 user1 评价过相同的商家的用户，返回用户名、共同评价的商家的数量，按照评价数量降序排序（查看该查询计划，了解该查询的执行计划及查询执行时间，并给出物理优化手段，以提高查询性能，通过优化前后的性能对比展现优化程度。）。
18. 分别使用 Neo4j 和 MongoDB 查询 review_id 为 TIYgnDzezfeEnVeu9jHeEw 对应的 business 信息，比较两者查询时间，指出 Neo4j 和 MongoDB 主要的适用场景。

任务 4：多数据库交互应用实验

1. 使用 Neo4j 查找：找出评论过超过 5 家不同商户的用户，并在 Neo4j 以表格形式输出满足以上条件的每个用户的信息：name, funny, fans。
2. 将 1 得到的结果导入 MongoDB，并使用该表格数据，统计其中所有出现的用户名及该用户名对应的出现次数，并按照出现次数降序排序,使用 aggregate 实现.
3. 在 Neo4j 中查找所有商家，要求返回商家的名字，所在城市、商铺类。
   (1) 将查找结果导入 MongoDB 中实现对数据的去重（提示：使用 aggregate，仅保留城市、商铺类型即可）
   (2) 将去重后的结果导入 Neo4j 中的新库 result 中，完成（City-[Has]->Category）图谱的构建。

任务 5：不同类型数据库 MVCC 多版本并发控制对比实验

内容：请同学们自行构造多用户同时对同一数据库对象的增删改查案例，实验对比 MySQL 和 MongoDB 数据库对 MVCC 多版本并发控制的支持。

1. 体验 MySQL 在 InnoDB 存储引擎下的 MVCC 多版本并发控制，实现的事务 ACID 特性。请注意 Mysql 需要选用什么事务隔离级来支持 MVCC？请构造多用户多写多读案例来展现 MVCC 并发控制特性，解释各种结果产生的原因。
2. 体验 MongoDB 的 MVCC，数据集可自建或选用 yelp 数据集中的 test 集合中进行测试，测试方法同 MySQL。请对测试结果进行说明，并与 MySQL 的 MVCC 实验结果进行对比分析。建议创建 MongoDB 副本或分片集群，体验 MVCC 的不同效果（可选做其一）。