# 2024 大数据管理课程任务书

## 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](https://www.yelp.com/dataset/documentation/main)

## 软件安装

请参照华为平台部署手册及实验指导书。

## 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state为加利福尼亚州（CA）的商户的所有信息，结 果按照商户评分（stars）降序排列，限制返回5条。  
   （考察点：JSON提取，难度：★☆☆）
2. 在business表中，查询城市为Edmonton的商户的attributes中的所有键，并返回attributes中键的个数，限制返回5条。  
   （考察点：JSON\_KEYS，JSON\_LENGTH，难度：★☆☆）
3. 在business表中,查询该表business\_info列中name, stars, hours, latitude的内容和JSON类型, 限制返回行数为5.  
   （考察点：JSON\_TYPE，难度：★☆☆）
4. 在business表中，查询位于佛罗里达州（FL）并且店内有电视(attributes列中HasTV为’True’)的商户的name，city，stars，以及WiFi状态（在attributes列中），使用JSON\_UNQUOTE去除name的引号并按照stars降序排序，如果stars相同再按照name进行升序排序，限制返回20条记录。  
   （考察点：JSON\_UNQUOTE，JSON嵌套，难度：★★☆）
5. 使用explain查看select \* from user where user\_info->'$.cool' > 200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).最后, 在MySQL中为user\_info的字段加索引来优化提高查询效率, 对比一下MySQL加索引查询前后的查询效率, 分析加索引前后的执行计划.  
   （考察点：执行计划，难度：★★☆）

**b) . JSON增删改:**

1. 在busines表中,查询id为--eBbs3HpZYIym5pEw8Qdw的商户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attributes中新增"BikeParking":"True"的键值对,并将其评论数量改为42,属性的'WiFi'对应的值置为"Paid",返回其business\_info,同样用JSON\_PRETTY(business\_info)展示在修改前后的差异.  
   （考察点：JSON\_SET，JSON\_PRETTY，难度：★★☆）
2. 向user表插入一个id是'change'的商户,其商户信息与id为'--agAy0vRYwG6WqbInorfg'的商户完全一样,插入完成之后,将这个新记录的user\_info中的fans以及useful键值对删去, 为其增加一个’city’:’New York’的键值对，向其中最后查询'change'的所有信息.

（考察点：JSON插入、删除，难度：★★☆）

**c) . JSON聚合:**

1. 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.  
（考察点：JSON\_OBJECTAGG，难度：★★☆）

1. 从 tip 表中, 选择所有用户的 user\_id, 以及与每个用户关联的 business\_id 和对应的 tip\_info, 使用 JSON\_ARRAYAGG 函数将每个用户的 tips 聚合成一个JSON数组, 限制返回5行.  
   （考察点：JSON\_ARRAYAGG，难度：★★★）

**d) . JSON实用函数的使用:**

1. 在business表中,查询城市在Edmonton的商户,使用JSON\_OVERLAPS()判断商户是否满足WiFi为'u'no', DogsAllowed为'True'以及HasTV为'False'这三个条件中的任意一个, 查询这些城市的name以及以上提及的三个属性, 按名字排序, 限制输出10行.  
   （考察点：JSON\_OVERLAPS，难度：★★☆）
2. 在user表中, 查询useful大于1000的用户, 返回非json数组下他们的name, funny, cool, useful 以及按json数组形式表示的funny, useful, cool, 以及三者的和, 限制10条;  
   （考察点：JSON\_ARRAY，JSON\_ARRAY\_INSERT，难度：★★★）
3. 在tip表中找到business\_id为-1b2kNOowsPrPpBOK4lNkQ的商户和user\_id为--7XOV5T9yZR5w1DIy\_Dog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.  
   （考察点：JSON\_PRESERVE，难度：★★★）
4. 查询被评论数前3的商户,使用JSON\_TABLE()可以将json型数据转换为关系型表格, 请使用JSON\_TABLE()将商户的name, HasTV, 和所有的attributes(不考虑顺序, 一个属性就对应一行, 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.  
   （考察点：JSON\_TABLE导出，难度：★★★）

### 任务2：MongoDB实验

**a) . 条件查询与执行计划:**

1. 查询business集合的5条数据, 跳过前5条.  
   （考察点：skip，limit，难度：★☆☆）
2. 查询business集合中state是CA并且BikeParking为True的5条数据.  
   （考察点：简单的条件查询，难度：★☆☆）
3. 查询review集合中useful大于500的评论, 只需要返回business\_id, user\_id和useful,限制10条数据.  
   （考察点：简单条件查询，返回指定列，难度：★☆☆）
4. 查询user集合中useful属于[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]的user, 返回name和useful, 限制20条数据.  
   （考察点：$in的使用，难度：★☆☆）
5. 查询user集合中100 <= fans < 200且useful >= 1000的user, 返回name, fans, 以及useful, 限制10条.  
   （考察点：与条件查询，难度：★☆☆）
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。  
   （考察点：执行计划，难度：★☆☆）
7. 查询business集合city为Westlake或者Las Vegas的数据, 限制返回5条.  
   （考察点：或条件查询，也可尝试其他方法，难度：★☆☆）
8. 查询business集合中, 找出categories的数量为5的商户, 返回商户的name, categories以及stars, 限制10条.  
   （考察点：size的使用，难度：★★☆）
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。  
   （考察点：查询优化，难度：★★☆）

**b) . 聚合与索引:**

1. 统计各个州的商店数量, 结果按照商店数量降序排序.  
   （考察点：简单聚合，难度：★★☆）
2. 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大于等于50的评价,按照review\_id进行升序排序， 限制返回5条.  
   （考察点：子集合的创建，创建索引，索引查询，难度：★★☆）
3. 在Subreview集合中统计评价中, 找出useful大于50的所有评论, 并返回business\_id, 以及平均打星, 按照商家id排序, 限制返回20条记录.  
   （考察点：聚合运算，难度：★★☆）
4. 在business表中, 查询距离商家smkZUv\_IeYYj\_BA6-Po7oQ(business\_ id) 2公里以内的所有商家, 返回商家名字, 地址和星级, 按照星级降序排序, 限制返回20条.

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  
 （考察点：地图索引，难度：★★★）

1. 在集合Subreview上建立索引, 统计出用户从2000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.  
   （考察点：索引创建，聚合，难度：★★☆）

**c). MapReduce的使用:**

1. 使用map reduce计算Subreview集合中每个商店的平均得分, (不要直接使用聚合函数), 输出为一个集合Map\_Reduce, 其中应该包括business\_id以及values{count(打分次数), sum\_stars(总的打分), avg\_stars(平均打分)}, 最后查询Map\_Reduce, 返回前20条数据.  
   （考察点：Map Reduce，难度：★★★）

### 任务3：Neo4j实验

1. 查询标签是UserNode的节点, 返回name和userid, 限制返回10条记录.   
   （考察点：简单查询，难度：★☆☆）
2. 查询城市是Ambridge的商家节点, 限制返回5条.  
   （考察点：简单条件查询，难度：★☆☆）
3. 查询reviewid是T\_8OnmZbyRhnXutFWGqaRg对应的user信息.  
   （考察点：简单关系查询，难度：★☆☆）
4. 查询userid是d9GraD1OjVyTEd1zPjp7Yg的用户所评价的商家的name, stars, 以及该用户评价的stars, 返回前20条记录.   
   （考察点：简单关系查询，返回特定列，难度：★☆☆）
5. 查询被userid为d9GraD1OjVyTEd1zPjp7Yg的用户评论为5星的商家name, address以及stars.  
   （考察点：多关系查询，返回特定列，难度：★★☆）
6. 查询userid为AWCY8laHjH0-3HMT0LGpUA的用户评价的商家name及用户评价的stars, 按照stars降序排序.  
   （考察点：查询排序，难度：★☆☆）
7. 使用where查询粉丝数大于200的用户的名字和粉丝数, 这里要注意fans的数据类型是字符串, 比较时使用toInteger()将其类型转换为整型, 限制返回20条记录.  
   （考察点：where条件查询，难度：★★☆）
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.  
   （考察点：执行计划查看，难度：★★☆）
9. 查询businessid是KWywu2tTEPWmR9JnBc0WyQ商家包含的种类,以list的形式返回, 同时返回商家的名字.  
   （考察点：collect的使用，难度：★★☆）
10. 查询userid为d7D4dYzF6THtOx9imf-wPw的用户的朋友（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), 返回前20条数据.   
    （考察点：with的使用，难度：★★★）
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.

（考察点：条件查询，count，难度：★★☆）

1. 查询商家名重复次数前10的商家名及其次数。  
   （考察点：条件查询，count，难度：★★☆）
2. 统计评价过商户id为nh\_kQ16QAoXWwqZ05MPfBQ的用户的name以及useful, funny, cool三者的和, 并按照该和降序排列.  
   （考察点：查询时简单运算，难度：★★★）
3. 查询具有评分为5.0的Propane类别的商铺的名字和所在的城市以及地址.  
   （考察点：多关系联合查询，难度：★★★）
4. 统计每个用户评价过多少个不同的商家, 按照此数量降序排列, 返回name, fans, useful以及评价过的不同商家数 结果限制20条记录.  
   （考察点：条件查询，count，难度：★★★）
5. 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）。  
   （考察点：创建索引，难度：★★☆）
6. 查询与用户`user1`（`userid: 4i4lyXBigT2HShIjw7TbDw`) 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, 并尝试根据查询计划优化.  
   （考察点：多关系联合查询，with，count，难度：★★★）
7. 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。  
   （考察点：多数据库的比较，难度：★★☆）

### 任务4：多数据库交互应用实验

1. 使用neo4j查找：找出被超过五个用户评论过的商家, 返回name, stars, city以及address.  
   （考察点：多关系联合查询，返回指定列，count，难度：★★☆）
2. 将1得到的结果导入MongoDB, 并使用该表格数据, 统计每个城市对应的商户数量, 使用aggregate实现.  
   （考察点：数据库交互，聚合，难度：★★★）
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。  
      （考察点：数据库去重，数据库交互，难度：★★★）

### 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）。

（考察点：MVCC实现方式对比，难度：★★☆）