44 | DBMS篇总结和答疑：用SQLite做词云

2019-09-20 陈旸

在认识DBMS篇中，我们讲解了Excel+SQL、WebSQL、SQLite以及Redis的使用，这些DBMS有自己适用的领域，我们可以根据需求选择适合的DBMS。我总结了一些大家常见的问题，希望能对你有所帮助。

关于Excel+SQL

答疑1：关于mysql-for-excel的安装

Excel是我们常用的办公软件，使用SQL做数据分析的同学也可以使用Excel+SQL作为报表工具，通过它们提取一些指定条件的数据，形成数据透视表或者数据透视图。

但是有同学在安装mysql-for-excel-1.3.8.msi 时报错，这里感谢**同学莫弹弹**给出了解答。解决这个问题的办法是在安装时需要Visual Studio 2010 Tools for Office Runtime 才能运行。

它的下载链接在这里： https://www.microsoft.com/zh-CN/download/confirmation.aspx?id=56961

关于WebSQL

我在讲解WebSQL操作本地存储时，可以使用浏览器中的Clear Storage功能。有同学问到：这里只能用户手动删除才可以吗？

事实上，除了在浏览器里手动删除以外，我们完全可以通过程序来控制本地的SQLite。

使用executeSql函数即可，在executeSql函数后面有两个function，分别代表成功之后的调用，

以及执行失败的调用。比如想要删除本地SQLite的heros数据表，可以写成下面这样：

```
tx.executeSql("DROP TABLE heros",[],
function(tx, result) {alert('Drop 成功');},
function(tx, error) {alert('Drop 失败' + error.message);});
```

第二个问题是，Session是什么概念呢？HTTP请求不是无状态的吗？

我在文章中讲到过SessionStorage，这里的Session指的就是一个会话周期的数据，当我们关闭浏览器窗口的时候，SessionStorage存储的数据就会被清空。相比之下localStorage存储的时间没有限制，一年之后数据依然可以存在。

HTTP本身是一个无状态的连接协议，想要保持客户端与服务器之间的交互，可以使用两种交互存储方式，即Cookie和Session。

Cookie是通过客户端保存的数据，也就是可以保存服务器发送给客户端的信息，存储在浏览器中。一般来说，在服务器上也存在一个Session，这个是通过服务器来存储的状态信息，这时会将浏览器与服务器之间的一系列交互称为一个Session。这种情况下，Session会存储在服务器端。

不过我们讲解的sessionStorage是本地存储的解决方式，它存放在浏览器里，借用了session会话的概念，它指的是在本地存储过程中的一种临时存储方案，数据只有在同一个session会话中的页面才能访问，而且当session结束后数据也会释放掉。

关于SQLite

第一个问题关于SQLite查找微信本地的聊天记录，有同学说可以导出聊天记录做个词云。

这是个不错的idea，我们既然有了SQLite，完全可以动手做个数据分析，做个词云展示。

我在《数据分析45讲》里讲到过词云的制作方法，这里使用Python+SQLite查询，将微信的聊天记录做个词云，具体代码如下：

```
import sqlite3
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import jieba
import os
import re


# 去掉停用词
```

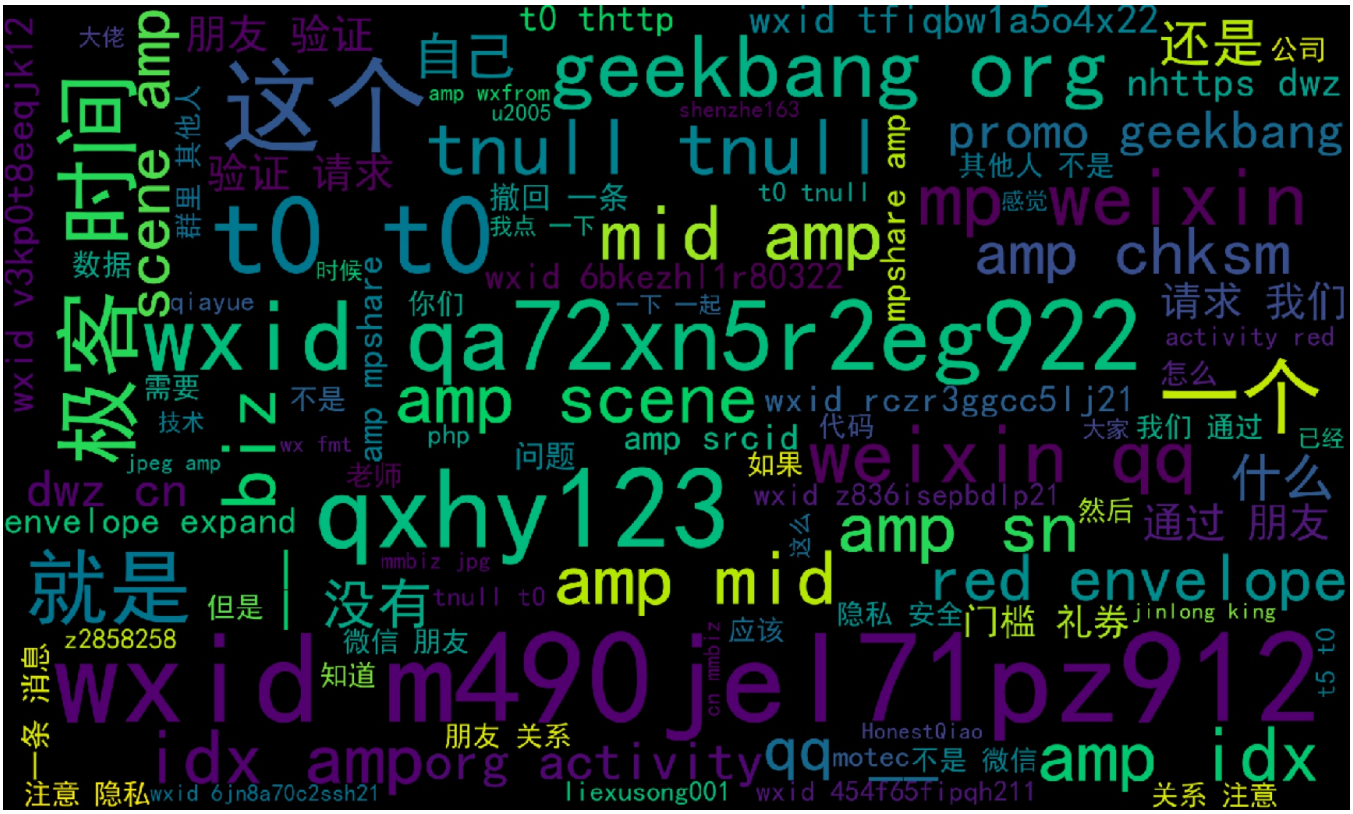
```
# 去掉停用词
def remove_stop_words(f):
    stop_words = ['你好', '已添加', '现在', '可以', '开始', '聊天', '当前', '群聊', '人数', '过多', '显示', '群成员', '昵称', '信息
    for stop_word in stop_words:
        f = f.replace(stop_word, '')
    return f

# 生成词云
def create_word_cloud(f):
    print('根据微信聊天记录，生成词云!')
    # 设置本地的simhei字体文件位置
    FONT_PATH = os.environ.get("FONT_PATH", os.path.join(os.path.dirname(__file__), "simhei.ttf"))
    f = remove_stop_words(f)
    cut_text = " ".join(jieba.cut(f,cut_all=False, HMM=True))
    wc = WordCloud(
        font_path=FONT_PATH,
        max_words=100,
        width=2000,
        height=1200,
    )
    wordcloud = wc.generate(cut_text)
    # 写词云图片
    wordcloud.to_file("wordcloud.jpg")
    # 显示词云文件
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.show()

def get_content_from_weixin():
    # 创建数据库连接
    conn = sqlite3.connect("weixin.db")
    # 获取游标
    cur = conn.cursor()
    # 创建数据表
    # 查询当前数据库中的所有数据表
    sql = "SELECT name FROM sqlite_master WHERE type = 'table' AND name LIKE 'Chat\_%' escape '\\'"
    cur.execute(sql)
    tables = cur.fetchall()
```

```
    content = ''
    for table in tables:
        sql = "SELECT Message FROM " + table[0]
        print(sql)
        cur.execute(sql)
        temp_result = cur.fetchall()
        for temp in temp_result:
            content = content + str(temp)
    # 提交事务
    conn.commit()
    # 关闭游标
    cur.close()
    # 关闭数据库连接
    conn.close()
    return content
content = get_content_from_weixin()
# 去掉HTML标签里的内容
pattern = re.compile(r'<[^>]+>',re.S)
content = pattern.sub('', content)
# 将聊天记录生成词云
create_word_cloud(content)
```

运行结果：



你在Github上也可以找到相应的代码，这个结果图是我运行自己的微信聊天记录得出的。

我来讲解下代码中相关模块的作用。

首先是create_word_cloud函数，通过聊天内容f，展示出词云。这里会用到WordCloud类，通过它配置本地的simhei字体（因为需要显示中文），设置显示的最大词数max_words=100，图片的尺寸width和height。

第二个是remove_stop_words函数，用来设置停用词，也就是不需要统计的单词，这里我设置了一些，不过从结果中，你能看到我们需要更多的停用词，要不会统计出一些没有意义的词汇，比如“撤回”“一条”等。

第三个是get_content_from_weixin函数。这里我们通过访问SQLite来访问微信聊天记录，首先需要查询数据表都有哪些，在微信的本地存储里每个数据表对应着一个聊天对象，然后我们对这些数据表中的message字段进行提取。

最后，因为统计出来的聊天记录会包括大量的HTML标签，这里我们还需要采用正则表达式匹配的方式将content中的HTML标签去掉，然后调用create_word_cloud函数生成词云，结果就是文稿中的图片所示啦。

第二个问题是，Navicat如何导入weixin.db呢？

事实上，使用Navicat导入weixin.db非常简单。首先我们需要创建SQLite连接，然后从本地选择数据库文件，这里选中weixin.db。

然后就导入到Navicat中了，你在左侧可以看到weixin的连接，然后打开main数据库就可以看到聊天记录的数据表了。

我制作了演示视频，可以看下。

关于Redis

第一个问题，MongoDB、Redis之间有什么区别？实际使用时应该怎么选择呢？

Redis是Key-Value数据库，数据存放在内存中，查询和写入都是在内存中进行操作。当然Redis也支持持久化，但持久化只是Redis的功能之一，并不是Redis的强项。通常，你可以把Redis称之为缓存，它支持的数据类型丰富，包括字符串、哈希、列表、集合、有序集合，同时还支持基数统计、地理空间以及索引半径查询、数据流等。

MongoDB面向文档数据库，功能强大，是非关系型数据库中最像RDBMS的，处理增删改查也可以增加条件。

在存储方式上，Redis将数据放在内存中，通过RDB或者AOF方式进行持久化。而MongoDB实际上是将数据存放在磁盘上的，只是通过mmap调用，将数据映射到内存中，你可以将mmap理解为加速的方式。mmap调用可以使得对普通文件的操作像是在内存中进行读写一样，这是因为它将文件映射到调用进程的地址空间中，实现了文件所在的磁盘物理地址与进程空间的虚拟地址一一映射的关系，这样就可以直接在内存中进行操作，然后写完之后同步一下就可以存放到文件中，效率非常高。

不过在使用选择的时候，我们还是将 MongoDB 归为数据库，而将Redis归为缓存。

总的来说，Redis就像一架飞机，查询以及写入性能极佳，但是存储的数据规模有限。MongoDB就像高铁，在处理货物（数据）的功能上强于Redis，同时能承载的数据量远高于Redis，但是查询及写入的效率不及Redis。

第三个问题是，我们能否用Redis中的DECR实现多用户抢票问题？

当然是可以的，在专栏文章中我使用了WATCH+MULTI的乐观锁方式，主要是讲解这种乐观锁的实现方式。我们也可以使用Redis中的DECR命令，对相应的KEY值进行减1，操作是原子性的，然后我们判断下DECR之后的数值即可，当减1之后大于等于0证明抢票成功，否则小于0则说明抢票失败。

这里我给出了相应的代码，你也可以在Github上下载。

```
# 抢票模拟，使用DECR原子操作
import redis
import threading
# 创建连接池
pool = redis.ConnectionPool(host = '127.0.0.1', port=6379, db=0)
# 初始化 redis
r = redis.StrictRedis(connection_pool = pool)

# 设置KEY
KEY="ticket_count"
# 模拟第i个用户进行抢购
def sell(i):
    # 使用decr对KEY减1
    temp = r.decr(KEY)
    if temp >= 0:
        print('用户 {} 抢票成功，当前票数 {}'.format(i, temp))
    else:
        print('用户 {} 抢票失败，票卖完了'.format(i))

if __name__ == "__main__":
    # 初始化5张票
    r.set(KEY, 5)
    # 设置8个人抢票
    for i in range(8):
        t = threading.Thread(target=sell, args=(i,))
        t.start()
```

最后有些同学感觉用Redis，最终还是需要结合程序以及MySQL来处理，因为排行榜展示在前端还是需要用户名的，光给个用户id不知道是谁，除非Redis有序集合的member包含了用户id和name。

这里，排行榜中如果要显示用户名称，需要放到有序集合中，这样就不需要再通过MySQL查询一次。这种需要实时排名计算的，通过Redis解决更适合。如果是排行榜生成之后，用户想看某一个用户具体的信息，比如地区、战绩、使用英雄情况等，可以通过MySQL来进行查询。而对于热点数据使用Redis进行缓存，可以解决高并发情况下的数据库读压力。

所以你能看到Redis通常可以作为MySQL的缓存，它存储的数据量有限，适合存储热点数据，可

以解决读写效率要求很高的请求。而MySQL则作为数据库，提供持久化功能，并通过主从架构提高数据库服务的高可用性。

最后留两个思考题。

我在文稿中，使用SQLite对于微信聊天记录进行查询，使用wordcloud词云工具对聊天记录进行词云展示。同时，我将聊天记录文本保存下来，一共4.82M（不包括HTML标签内容），你可以使用SQLite读取微信聊天记录，然后看下纯文本大小有多少？

第二个问题是，我们使用Redis作为MySQL的缓存，假设MySQL存储了1000万的数据，Redis只保存有限的数据，比如10万数据量，如何保证Redis存储的数据都是热点数据呢？

欢迎你在评论区写下你的思考，也欢迎把这篇文章分享给你的朋友或者同事，一起交流一下。

精选留言

jxs1211 0

有些复杂的sql语句，如何转换成对应的sqlalchemy语句，有什么好的工具和方法吗

2019-09-20

DemonLee 0

1、这里，排行榜中如果要显示用户名称，需要放到有序集合中，这样就不需要再通过 MySQL 查询一次。这种需要实时排名计算的，通过 Redis 解决更适合。

----老师，这里不明白，有序集合里面不是已经存放了userId，如何再存放userName

2、第二个问题是，我们使用 Redis 作为 MySQL 的缓存，假设 MySQL 存储了 1000 万的数据，Redis 只保存有限的数据，比如 10 万数据量，如何保证 Redis 存储的数据都是热点数据呢？
----把查询到的数据保存一份到redis，使用有序集合，每次如果从redis获取到，则score+1，超过10w条数据，则删除。（好像也有问题）

2019-09-20

往事随风，顺其自然

0

通过redis 的得分来进行存储热点数据

2019-09-20

蒙开强

0

老师，你好，用redis做缓存，那么如何保证与MySQL数据库数据一致呢，先存redis和先存mysql都会有问题

2019-09-20