Recommending Database Architectures for Social Queries: A Twitter Case Study

Michael Marountas[1], Georgios Drakopoulos[2](✉), Phivos Mylonas[2], and Spyros Sioutas[1]

[1] University of Patras, CEID, Patras 26504, Hellas
{marounta,sioutas}@ceid.upatras.gr
[2] Department of Informatics, Ionian University, Kerkyra 49100, Hellas
{c16drak,fmylonas}@ionio.gr

Abstract. Database deployment is a complex task depending on a multitude of operational parameters such as anticipated data scaling trends, expected type and volume of queries, uptime requirements, replication policies, available budget, and personnel training and experience. Thus, enterprise database administrators eventually rely on various performance metrics in conjunction to existing company policies in order to determine the best possible solution under these constraints. The recent advent of NoSQL databases, including graph databases such as Neo4j and document stores like MongoDB, added another degree of freedom in database selection since for a number of years relational databases such as PostgreSQL were the only available technology. In this work the scaling characteristics of a representative set of social queries executed on virtual machine installations of PostgreSQL and MongoDB are evaluated on a large volume of political tweets regarding Brexit. Moreover, Wiener filters for predicting the execution time of social query windows of fixed length over both databases are designed.

Keywords: Database administration · Database evaluation · Signal processing for databases · Information engineering · Social network analysis

1 Introduction

Database deployment is by no means a trivial engineering task since it entails the fine tuning of a plethora of high- and low-level operational parameters including hardware management, data transformations and cleansing mechanisms, integrated access policy, communication with other production-grade systems, and query optimization techniques. Additionally, the complexity as well as the performance of each database installation with respect to at least a representative set of anticipated queries should be assessed [41]. With the advent of social media, databases have been widely adopted as a reliable means for storing large volumes of (semi)structured social data in enterprise environments.

Published by Springer Nature Switzerland AG 2021
I. Maglogiannis et al. (Eds.): AIAI 2021, IFIP AICT 627, pp. 715–728, 2021.
https://doi.org/10.1007/978-3-030-79150-6_56

Twitter is perhaps the most popular microblogging platform, suitable for online interplay mainly in the form of conversations [21]. The topic of the latter may well range from smart cities [34] and public health and the ongoing COVID-19 pandemic [22] to education [23] and sustainable development [31]. Nevertheless, it was the political Twitter which during the recent past years has gained considerable popularity. In part this can be attributed to the fact that Twitter was the primary media vehicle of the 45th US President [35,40], with the 2016 US presidential elections being an important milestone [10,45]. As a result, Twitter is now widely considered as a central political stage.

The primary research contribution of this conference paper is the assessment of two standalone instances of PostgreSQL and MongoDB to handle queries closely related to social analytics. First, the scaling dynamics of four representative queries forming the basis for three Twitter analytics are evaluated based on a large collection of English tweets regarding Brexit. Second, Wiener filters of various lengths are designed for the prediction of the execution time of fixed length non-overlapping query windows over these two installations. This point differentiates this work from previous approaches.

The remainder of this work is structured as follows. Section 2 summarizes the recent scientific literature regarding Twitter analytics, relational database management systems (RDBMSs), and document databases. The experimental setup and the proposed methodology are described in Sect. 3. In Sect. 4 possible future research directions are given. Random variables (r.vs) are represented with capital calligraphic letters. Technical acronyms are explained the first time they are met in the text. Table 1 summarizes notation.

Table 1. Notation of this conference paper.

Symbol	Notation	First in
$\triangleq$	Definition or equality by definition	Eq. (1)
$(t_1, \ldots, t_n)$	Tuple with elements $t_1, \ldots, t_n$	Eq. (9)
$\lvert S\rvert$	Set or tuple cardinality	Eq. (1)
$\mathrm{E}\left[\mathcal{X}\right]$	Mean value of random variable $\mathcal{X}$	Eq. (11)
ρ_j	j-th autocorrelation coefficient of a random variable	Eq. (10)
$\varPhi\left[k\right]$	Set of followers of the k-th account	Eq. (1)
$\varPsi\left[k\right]$	Set of followees by the k-th account	Eq. (1)

2 Previous Work

Relational databases have been a productivity mainstay almost since their introduction [27]. They rely on a tabular data format as well as on a number of integrity constraints to ensure both high performance and data consistency [16].

Their performance has been evaluated in multiple ways, including query scalability [42], indexing [30], and fault tolerance [17]. From an operational complexity view normalized forms [29] and user rights assignment [24] play a central role.

Document databases typically operate on structured documents, usually in extensible markup language (XML) or JavaScript object notation (JSON) format [4]. The prospect of developing normalization and embedding operations in MongoDB is explored in [20], while the expressivity of MongoDB queries is the subject of [8]. Neural networks in keras for predicting mentions to Twitter verified accounts driven by data stored in MongoDB are described in [26]. Finally, a persistent data structure with rollback capabilities which can represent graphs but also structured documents is presented in [25].

There is a plethora of social analytics for Twitter. Digital influence as inferred through numerous attributes is paramount [36], with affective influence being also taken into consideration [7]. Also community discovery can take many forms such as spatio-linguistic [14], semantic based on hashtags [12], or hashtag similarity [44]. One-dimensional topological correlation for Twitter graphs based on structural and functional attributes is proposed in [13]. Recently data-driven approaches have been developed for examining Twitter graphs [6]. In [18] a socio-technical analysis of tweets is presented. Twitter as a vehicle for political campaigns is examined in [3]. Surveys covering the topic are [33] and [2].

3 Methodology

3.1 ACID vs BASE

During the 2010s the advent of NoSQL databases marked an addition of variation of fundamental database models. Along with the existing tabular data format of relational databases special infrastructure has been added for graphs, documents, associative arrays, and column families with the corresponding database type. Table 2 summarizes the main architectural differences between an RDBMS such as PostgreSQL and a document database like MongoDB.

Table 2. PostgreSQL vs MongoDB.

Feature	PostgresSQL	MongoDB
Data type	Table	Document collection
Data unit	Table row	JSON document
Feature	Table column	Document field
Operations	Relational model	CRUD (Create, read, update, delete)
Join	Relational joins	Embedded documents
Keys	Explicit	Implicit
Schema	Static	Dynamic

3.2 Dataset

The two databases were installed on two sepearate virtual machines (VMs) running over a public cloud. Table 3 has the technical specifications. In either case no indexing was activated. In the case of MongoDB the Python driver was used to write code for the Twitter analytics examined here.

In order to test the query dynamics of each database, a Twitter dataset containing a large number of tweets pertaining to Brexit, a term encompassing major political events since 2018, has been collected by a social crawler utilizing a topic sampling approach. Given that the actual Brexit took effect on February 1st of 2020, there was considerable interest in it during the few preceding months. Because of the paramount geopolitical importance of the event, a considerable number of English tweets was eventually collected over four months preceding it. The main properties of the dataset are shown in Table 3.

Table 3. Experimental setup.

Property	Value	Property	Value
VM operating system	Ubuntu	Number of tweets	500k
Number of processors	2	Collection interval	Nov.2019–Jan.2020
Processor type	Intel Core i7-10700	Accounts	102.3k
Level 1 cache size	16 MB	*follow* relationships	5.62 m
Available GPU	No	Hashtags	15.1k
Available memory	8 GB	Number of repetitions r_0	10000
Disk size	2TB	Number of equations N_0	10

3.3 Twitter Analytics

As stated earlier, social analytics abound for both the general case and for Twitter alone. In this work the following analytics will be used as a benchmark. More details about their respective implementations will be given later in the text.

One way to assess the impact of the k-th Twitter account is to compute the respective followers-to-followees logarithmic ratio defined as in Eq. (1):

$$J_r\left[k\right] \triangleq \log\left(1 + \frac{\left|\Phi\left[k\right]\right|}{\min\left\{1, \Psi\left[k\right]\right\}}\right) \tag{1}$$

From another perspective, the digital influence of the l-th tweet, regardless of the influence of the respective posting account, can be measured by the number of hashtags H_l it contains. This is shown in Eq. (2):

$$J_h\left[l\right] \triangleq H_l \tag{2}$$

An alternative for accounts requiring combined information is to count the total number of tweets W_k each account has posted multipied by the number of the respective followers as a visibility metric. This is shown in Eq. (3):

$$J_t\left[k\right] \triangleq W_k|\Phi\left[k\right]| \tag{3}$$

Observe that each of the above metrics captures a different aspect of Twitter activity, whether it is account- or tweet-oriented. In general, so far and to the best of the knowledge of the authors there is no single metric describing Twitter influence or account online behavior. From a database standpoint this translates to the need for quantifying the performance of individual queries as well as of query sequences. This can be done in terms of total execution time, system resources such as memory utilization or disk usage, or of operational complexity and personnel training costs. For the purposes of this work the total execution time has been selected as the single database performance criterion.

3.4 Response to Individual Queries

In order to execute the four queries described below, in the PostgreSQL case two tables were created, one containing information about accounts and one about tweets. After that, a set Q of reference queries, called *social queries*, was created. The wallclock time is a reliable indicator of the system execution time since these queries are strongly CPU bound [38]. Specifically, Q consists of the following queries, which are cast in SQL in addition to being verbally explained:

- q_1: This query selects rows based on the values of a given column. In social media it is frequently used to find accounts with a certain attribute and the value of the attribute in question satisfies certain numerical constraints. For instance, q_1 can be a filter for accounts having more than one follower.

  ```
  SELECT account FROM accounts WHERE followers >= 1;
  ```

- q_2: This query is similar to q_1 but the results are additionally sorted in descending order based on the values of a given column. Continuing the previous example, q_2 returns accounts as q_1 in descending order of followers. This is a popularity metric as well as a crude measure of digital influence.

  ```
  SELECT account FROM accounts WHERE followers >= 1
  ORDER BY followers DESC;
  ```

- q_3: The next step in query complexity is aggregation, in this case by the field preceding the GROUP BY clause. The following returns the total hashtags for each Twitter account.

  ```
  SELECT hastags, account FROM tweets
  GROUP BY account;
  ```

- q_4: Finally, the most time-consuming query is join. There are no such operations in the NoSQL world where each of the four primary technologies deals differently with it. The following query returns the number of tweets and followers for each account, provided the latter has more than one follower.

```
SELECT accounts.account, accounts.followers
FROM accounts INNER JOIN tweets
ON accounts.account = tweets.account
WHERE accounts.followers >= 1;
```

There is a correspondence between the above queries and the Twitter analytics presented earlier. Specifically, J_a can be implemented with two q_2 to find followers and followees respectively and one q_1, J_h with q_3 and a q_1, and J_t with q_4 and a q_1. In each case the last q_1 is used to select the top accounts or tweets.

In order to accurately determine total response times, each query for each database was run r_0 times, each time on a randomly selected subset of n tweets. The sample mean $T(n)$ was recorded as follows, where t_j is the j-th measurement:

$$T(n) \triangleq \frac{1}{r_0} \sum_{j=1}^{r_0} t_j \tag{4}$$

Equation (4) can be thought of the sample mean approximation of the true stochastic mean of the random variable counting the total execution time. Under mild conditions of ergodicity, the former converges to the latter as r_0 grows.

A metric of how much are the samples concentrated around the respective mean value is the variance. Along a similar line of reasoning, the sample variance $\sigma^2(n)$ approximates the true one and it is computed as follows:

$$\sigma^2(n) \triangleq \frac{1}{r_0 - 1} \sum_{j=1}^{r_0} \left(t_j - T(n)\right)^2 \tag{5}$$

In Fig. 1 the mean times for q_1 and q_2 for both databases are displayed.

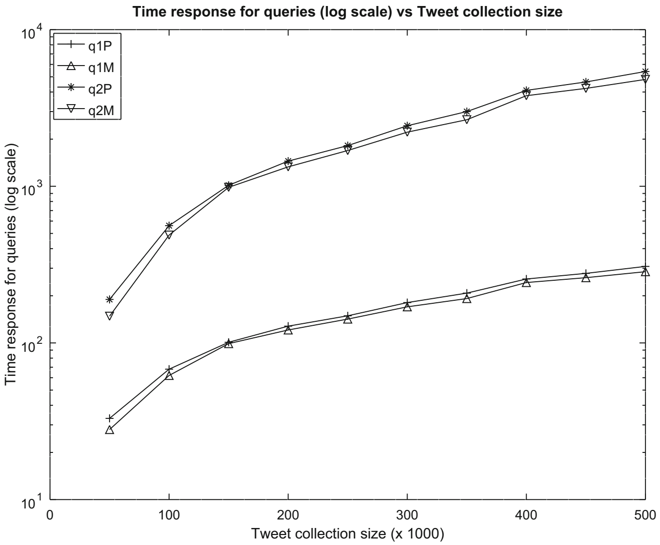

Fig. 1. Execution time for q_1 and q_2 vs query size (Source: Authors).

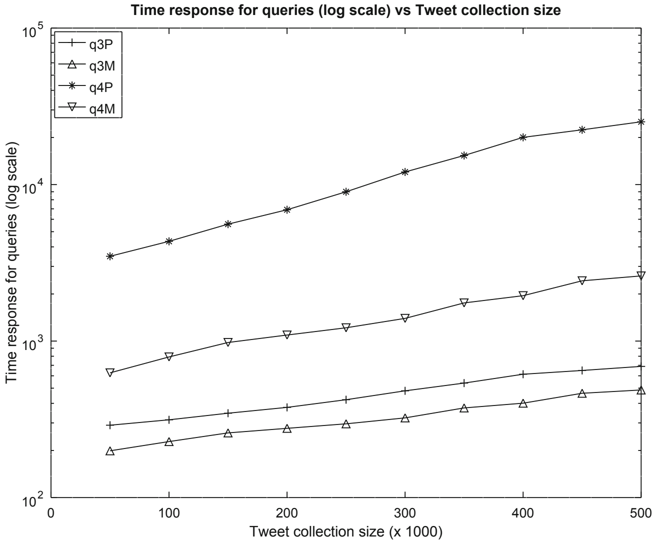

Fig. 2. Execution time for q_3 and q_4 vs query size (Source: Authors).

Likewise, in Fig. 2 the response times for queries q_3 and q_4 is shown.

The values of the sample variance are shown in Table 4. Observe that the sample variance grows very slowly compared with the growth of the respective query size. This is an indicator that the estimates obtained by the sample mean are reliable, which can be partly attributed to the selection of r_0 (for its actual value see Table 3). Note that the PostgreSQL installation tends to yield more consistent estimators, as denoted by the systematically lower sample variance.

Table 4. Sample variance vs query size and database technology (msec).

SQL	50k	100k	150k	200k	250k	300k	350k	400k	450k	500k
q_1	16.24	16.57	17.34	18.54	20.20	22.02	23.60	23.36	23.35	23.08
q_2	18.09	18.47	19.04	19.02	21.41	22.93	24.84	24.98	24.34	24.45
q_3	18.23	19.07	19.49	20.59	23.46	25.49	27.37	27.57	27.74	27.34
q_4	22.33	23.11	24.36	25.47	27.73	28.11	28.46	28.55	28.12	28.76
MDB	50k	100k	150k	200k	250k	300k	350k	400k	450k	500k
q_1	17.33	17.45	18.74	19.69	21.58	22.67	23.45	24.35	24.45	24.48
q_2	23.02	23.81	23.70	24.35	25.89	26.85	26.94	26.56	26.33	26.35
q_3	24.22	24.38	23.54	25.96	26.44	27.20	27.68	28.73	28.67	28.68
q_4	25.11	25.58	25.36	26.56	27.76	27.15	28.48	28.24	29.92	30.57

The semilogarithmic scale can reveal certain scaling patterns. In particular, the almost straight lines of time curves for both databases for q_4 suggests a power law scaling with n of the total response time $T(n)$ of the form [32]:

$$T(n; \alpha_0, \gamma_0, n_0) \triangleq \alpha_0 (n + n_0)^{\gamma_0}, \qquad \alpha_0, \gamma_0, n_0 > 0 \tag{6}$$

In models like that of Eq. (6) the most important parameter is the scaling exponent γ_0 which determines the growth rate, whereas parameters α_0 and n_0 represent initial conditions and have minor effect on growth.

Taking the natural logarithm of Eq. (6) yields the equivalent (7). Notice there are N_0 equations, one for each of the possible values of n as shown in Table 4. The same value is repeated in Table 3 for convenience.

$$\ln T(n; \alpha_0, \gamma_0, n_0) \; = \; \ln \alpha_0 + \gamma_0 \ln (n + n_0) \tag{7}$$

Stacking the N_0 Eqs. (7) for the different values of n in increasing order of the latter yields the non-linear system of Eq. (8):

$$\begin{bmatrix} \ln T(n_1) \\ \vdots \\ \ln T(n_{N_0}) \end{bmatrix} = \begin{bmatrix} 1 & \ln (n_1 + n_0) \\ \vdots & \vdots \\ 1 & \ln (n_{N_0} + n_0) \end{bmatrix} \begin{bmatrix} \ln \alpha_0 \\ \gamma_0 \end{bmatrix} \tag{8}$$

Observe that (8) is non-linear and moreover the parameter n_0 is difficult to be separated from the data values n. To this end, instead of the standard least squares (LS) estimator, the following iterative scheme was used:

- The initial estimate for n_0 is used to obtain an LS solution for α_0 and γ_0.
- Using a line search with step $\mu_0 = \ln n_0$, a new estimate for n_0 is obtained.
- A new LS solution for α_0 and γ_0 is computed.
- If this solution increases residual error, then the search reverses direction.
- If two successive reverses occur, then the search terminates.
- The values of α_0, γ_0, n_0 with the least residual error are returned.

From the semilogarithmic plot of (7) these initial estimates can be deduced:

- The average arctangent of the plot of $T(\cdot)$ approximates the exponent γ_0.
- The rise of the plot of $T(\cdot)$ at the beginning of the latter is $\ln \alpha_0$.
- The shift of the plot of $T(\cdot)$ at the beginning of the latter is n_0.

From the definition of model (6) it is clear that it is the exponent γ_0 which dominates the time growth. The values obtained by the process described above for γ_0 for each query are given in Table 5. Notice that the values for q_1 are very close, but for the remaining queries MongoDB scales with a much lower rate.

Table 5. Values for the exponent γ_0.

SQL	q_1	q_2	q_3	q_4	MDB	q_1	q_2	q_3	q_4
γ_0	0.8273	1.1355	2.2148	2.8914	γ_0	0.8541	1.0534	2.0993	2.5231

3.5 Response to Query Sequences

In a typical enterprise environment even a standalone database installation is bound to serve query sequences with varying characteristics and of different sizes. The prediction of the execution time of each window allows the preemptive allocation of resources least under a certain set of normal operating conditions.

In this case the following scenario was simulated. The same batch of L_1 tweets, in the order they were created, are serially inserted to the two databases. This simulates the constant flow of tweets under normal operational conditions with no special events sparking intense activity happening. At the end of each such window the top ten accounts and tweets as determined by the three metrics of Eqs. (1), (2), and (3) are computed. The sum of the individual times of these analytics is defined as the reponse time for the respective window.

Notice that even in a simple scenario like this there is no direct correspondence between the execution time of each window and the queries it contains. This can be attributed to the following reasons:

- Although a connection between analytics and queries has been derived, the relationship between queries and response time is complicated.
- There is a non-linear connection between the time reponse of a query and its respective input size. Therefore, the reponse of an entire query sequence is more difficult to model and analyze.
- The query input size keeps growing, which translates not only to non-constant total response times but also to an increased variations thereof. Therefore, the prediction complexity is non-trivial.
- The response time depends not only on the query but also on factors such as the available memory, disk and processor utilization, disk controller performance, and the load factor which have not been modeled.

One way for predicting the execution time of a query window is a linear filter which takes as input the execution times of L_0 past windows and yields as output an estimate of the execution time of the $(L_0 + 1)$-st window. Depending on the filter as well as on the stochastic properties of the windows the coefficients of a filter may be updated periodically, updated after certain events, or remain fixed.

In this work the Wiener filter of length L_0 will be used, which has numerous applications in signal processing. It relies on approximation of input autocorrelation coefficients as computed by their respective sample counterparts. For instance, given a tuple of past window execution times as in Eq. (9):

$$X \triangleq (x_0, \ldots, x_{L_0-1}) \tag{9}$$

Observe that tuple X essentially contains L_0 samples of the r.v $\mathcal{X}_k$ which models the response time of the k-th execution window. Since these times were generated by the same database installation under the same conditions, they can be assumed to be modeled by identically distributed r.vs. As the queries of each window do not interact with those of other windows, then $\mathcal{X}_k$ can be also assumed to be independent. This probabilistic assumption greatly facilitates the approximate computation of the autocorrelation sequence, which is necessary in order to compute the filter coefficients. Assuming a stationary distribution for the execution times, the j-th autocorrelation coefficient ρ_j be defined as in (10):

$$\rho_j \triangleq \mathrm{E}\left[\mathcal{X}_k \mathcal{X}_{k-j}\right] \tag{10}$$

For example, the first two coefficients of the autocorrelation sequence $\mathrm{E}\left[\mathcal{X}_k^2\right]$ and $\mathrm{E}\left[\mathcal{X}_k \mathcal{X}_{k-1}\right]$ are approximated as follows in Eq. (11):

$$\begin{aligned} \rho_0 \triangleq \mathrm{E}\left[\mathcal{X}_k^2\right] &\approx \frac{1}{L_0} \sum_{k=0}^{L_0-1} x_k^2 \\ \rho_1 \triangleq \mathrm{E}\left[\mathcal{X}_k \mathcal{X}_{k-1}\right] &\approx \frac{1}{L_0-1} \sum_{k=0}^{L_0-2} x_k x_{k-1} \end{aligned} \tag{11}$$

In general, the coefficients w_k of a Wiener filter of length L_0 are the solution of the following linear system (12). Observe that the coefficient matrix is diagonally dominant as determined by the properties of the autocorrelation sequence, symmetric, and Toeplitz. These lead to efficient solutions for long filters.

$$\begin{bmatrix} \rho_0 & \rho_1 & \cdots & \rho_{L_0-1} \\ \rho_{L_0-1} & \rho_0 & \cdots & \rho_{L_0-2} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{L_0-1} & \rho_{L_0-2} & \cdots & \rho_0 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{L_0-1} \end{bmatrix} = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_{L_0} \end{bmatrix} \tag{12}$$

Once the Wiener filter has been computed for both database architectures and for various values of the filter length L_0, the mean square error between the actual and the predicted execution times was computed. The normalized results are shown in Table 6. The latter was selected in order for the differences in errors of the various combinations to be easier to understand.

Table 6. Normalized mean square error for database architectures.

L_0	11	21	31	41	51
PostgreSQL	1.7645	1.5656	1.4456	1.3412	1.142
MongoDB	1.616	1.5671	1.4563	1.3733	1

From the values of Table 6 it follows that MongoDB achieves a lower mean square error and, thus, the time responses of its query sequences are easier to be modeled and predicted. This in turn translates to the database administrators having a better view about the scaling dynamics of incoming tweet flows. The ability for quicker responses may be vital in extraordinary events.

3.6 Remarks

The findings of this previous are consistent, to the extent where a comparison can be made, with those reported in the recent literature examining the performance of MongoDB against that of the PostgreSQL in various engineering cases. Specifically, the scaling dynamics were similar to those in [39] and in [28], although refers to spatial data. Other tests, again on spatial data, in [5] showed the strengths and weaknesses of both databases. In [19] big data considerations were examined and constraints were derived again for both databases. The perspective of queries about line intersection and spatial containment problems was given in [1] where MongoDB was better. Queries on unstructured data were run on the same pair of databases in [9] with MongoDB being the winner because of its enhanced flexibility. When it comes to queries for semistructured, high velocity sensor measurements as explored in [43], again MongoDB seems to be the choice at the cost of extensive memory use. In [11] the performance of k-nearest neighbors is explored with the relational solution offering more advantages. MongoDB was the better for handling a stock market dataset over Hadoop [37]. Finally, the potential of MongoDB for mobile applications was explored with encouraging results [15].

Based on the experiments, the following remarks can be made:

- Concerning isolated queries, MongoDB achieves lower execution times but PostgreSQL has lower variance. The latter implies that PostgreSQL performance is easier to predict. This can be attributed to the structured data format and the greater degree of automation it offers.
- Concerning query windows, the total execution time is easier to predict in the case of MongoDB with Wiener filters of lower order. Since Wiener filters are efficiently implemented in most if not all scientific computation tools and are easy to understand intuitively, database administrators can easily obtain an understanding of MongoDB scaling dynamics and, based on their experience, of the major drivers behind them.

4 Future Research Directions

This conference paper focuses on the performance comparison between MongoDB and PostgreSQL, a special case of the general BASE vs ACID question, when handling Twitter analytics. As is the case in most realistic engineering scenarios, the answer is not simple. Rather, it depends on the evaluation of the current state as well as the near-term trend of a plethora of operational variables including memory and disk utilization, system load, and input query size. In this conference paper the scaling dynamics of both individual social queries, namely queries forming the base for Twitter analytics, and query sequences were evaluated on a large number of English tweets about Brexit. Although PostgeSQL achieved consistently lower variance of query execution times, MongoDB had lower mean times. Moreover, the execution times of query sequences were easier to predict in the MongoDB case, meaning that database administrators had a much better view of the query scaling trends.

Concerning possible future research directions, the use of larger datasets with a higher variability of attributes as performance benchmarks should be examined. Moreover, the scaling dynamics of database server clusters should be investigated. Additionally, other query sequence execution time predictors, whether fixed or adaptive, can be the focus or future research.

Acknowledgments. This research has been funded by the Ionian University.

References

1. Agarwal, S., Rajan, K.: Performance analysis of MongoDB versus PostGIS/PostgreSQL databases for line intersection and point containment spatial queries. Spat. Inf. Res. **24**(6), 671–677 (2016)
2. Antonakaki, D., Fragopoulou, P., Ioannidis, S.: A survey of Twitter research: Data model, graph structure, sentiment analysis and attacks. Expert Syst. Appl. **164**, (2021)
3. Badawy, A., Ferrara, E., Lerman, K.: Analyzing the digital traces of political manipulation: The 2016 Russian interference Twitter campaign. In: ASONAM, pp. 258–265. IEEE (2018)
4. Bagga, S., Sharma, A.: A comparative study of NoSQL databases. In: Singh, P.K., Singh, Y., Kolekar, M.H., Kar, A.K., Chhabra, J.K., Sen, A. (eds.) ICRIC 2020. LNEE, vol. 701, pp. 51–61. Springer, Singapore (2021). https://doi.org/10.1007/978-981-15-8297-4_5
5. Bartoszewski, D., Piorkowski, A., Lupa, M.: The comparison of processing efficiency of spatial data for PostGIS and MongoDB databases. In: Kozielski, S., Mrozek, D., Kasprowski, P., Małysiak-Mrozek, B., Kostrzewa, D. (eds.) BDAS 2019. CCIS, vol. 1018, pp. 291–302. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19093-4_22
6. Belhadi, A., Djenouri, Y., Lin, J.C.W., Cano, A.: A data-driven approach for Twitter hashtag recommendation. IEEE Access **8**, 79182–79191 (2020)
7. Bibi, M., Aziz, W., Almaraashi, M., Khan, I.H., Nadeem, M.S.A., Habib, N.: A cooperative binary-clustering framework based on majority voting for Twitter sentiment analysis. IEEE Access **8**, 68580–68592 (2020)
8. Botoeva, E., Calvanese, D., Cogrel, B., Xiao, G.: Expressivity and complexity of MongoDB queries. In: ICDT. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2018)
9. Cheng, Y., Zhou, K., Wang, J.: Performance analysis of PostgreSQL and MongoDB databases for unstructured data. In: MBDASM. Atlantis Press (2019)
10. Clarke, I., Grieve, J.: Stylistic variation on the Donald Trump Twitter account: a linguistic analysis of tweets posted between 2009 and 2018. PLoS One **14**(9), (2019)

11. Coşkun, İ., Sertok, S., Anbaroğlu, B.: k-nearest neighbour query performance analysis on a large scale taxi dataset: PostgreSQL vs MongoDB. International archives of the photogrammetry, remote sensing, and spatial information sciences (2019)
12. Drakopoulos, D., Giotopoulos, K.C., Giannoukou, I., Sioutas, S.: Unsupervised discovery of semantically aware communities with tensor Kruskal decomposition: a case study in Twitter. SMAP. IEEE (2020). https://doi.org/10.1109/SMAP49528.2020.9248469
13. Drakopoulos, G., Kafeza, E.: One dimensional cross-correlation methods for deterministic and stochastic graph signals with a Twitter application in Julia. SEEDA-CECNSM. IEEE (2020). https://doi.org/10.1109/SEEDA-CECNSM49515.2020.9221815
14. Drakopoulos, G., et al.: A genetic algorithm for spatiosocial tensor clustering. Evol. Syst. **11**(3), 491–501 (2019). https://doi.org/10.1007/s12530-019-09274-9
15. Fotache, M., Cogean, D.: NoSQL and SQL databases for mobile applications. Case study: MongoDB versus PostgreSQL. Informatica Economica **17**(2), 41–58 (2013)
16. Freitag, M., Bandle, M., Schmidt, T., Kemper, A., Neumann, T.: Adopting worst-case optimal joins in relational database systems. PVLDB **13**(12), 1891–1904 (2020)
17. Gorbenko, A., Karpenko, A., Tarasyuk, O.: Analysis of trade-offs in fault-tolerant distributed computing and replicated databases. In: DESSERT, pp. 1–6. IEEE (2020)
18. Grover, P., Kar, A.K., Davies, G.: Technology enabled health - Insights from Twitter analytics with a socio-technical perspective. Int. J. Inf. Manage. **43**, 85–97 (2018)
19. Jung, M.G., Youn, S.A., Bae, J., Choi, Y.L.: A study on data input and output performance comparison of MongoDB and PostgreSQL in the big data environment. In: DTA. IEEE (2015)
20. Kanade, A., Gopal, A., Kanade, S.: A study of normalization and embedding in MongoDB. In: IACC. IEEE (2014)
21. Karami, A., Lundy, M., Webb, F., Dwivedi, Y.K.: Twitter and research: a systematic literature review through text mining. IEEE Access **8**, 67698–67717 (2020)
22. Kearney, M.W.: rtweet: collecting and analyzing Twitter data. J. Open Source Softw. **4**(42), 1829 (2019)
23. Khan, H.U., Nasir, S., Nasim, K., Shabbir, D., Mahmood, A.: Twitter trends: S ranking algorithm analysis on real time data. Expert Syst. Appl. **164**, 45–67 (2021)
24. Khan, M.I., O'Sullivan, B., Foley, S.N.: Towards modelling insiders behaviour as rare behaviour to detect malicious RDBMS access. In: Big Data, pp. 3094–3099. IEEE (2018)
25. Kontopoulos, S., Drakopoulos, G.: A space efficient scheme for graph representation. ICTAI. IEEE (2014). https://doi.org/10.1109/ICTAI.2014.52
26. Kyriazidou, I., Drakopoulos, G., Kanavos, A., Makris, C., Mylonas, P.: Towards predicting mentions to verified Twitter accounts: building prediction models over MongoDB with Keras. WEBIST. SCITEPRESS (2019). https://doi.org/10.5220/0007810200250033
27. Luo, S., Gao, Z.J., Gubanov, M., Perez, L.L., Jermaine, C.: Scalable linear algebra on a relational database system. TKDE **31**(7), 1224–1238 (2018)
28. Makris, A., Tserpes, K., Spiliopoulos, G., Anagnostopoulos, D.: Performance evaluation of MongoDB and PostgreSQL for spatio-temporal data. In: EDBT/ICDT Workshops (2019)

29. Masri, D.: Relational databases and normalization. Developing Data Migrations and Integrations with Salesforce, pp. 1–11. Apress, Berkeley, CA (2019). https://doi.org/10.1007/978-1-4842-4209-4_1
30. Medina, J.M., Barranco, C.D., Pons, O.: Indexing techniques to improve the performance of necessity-based fuzzy queries using classical indexing of RDBMS. Fuzzy Sets Syst. **351**, 90–107 (2018)
31. Mills, J., Reed, M., Skaalsveen, K., Ingram, J.: The use of Twitter for knowledge exchange on sustainable soil management. Soil Use Manag. **35**(1), 195–203 (2019)
32. Newman, M.E.: Network structure from rich but noisy data. Nat. Phys. **14**(6), 542–545 (2018)
33. Nugroho, R., Paris, C., Nepal, S., Yang, J., Zhao, W.: A survey of recent methods on deriving topics from Twitter: Algorithm to evaluation. Knowl. Inf. Syst. **62**(7), 2485–2519 (2020)
34. Osorio-Arjona, J., Horak, J., Svoboda, R., García-Ruíz, Y.: Social media semantic perceptions on Madrid Metro system: using Twitter data to link complaints to space. Sustainable Cities Society **64**, (2021)
35. Ott, B.L.: The age of Twitter: Donald J. Trump and the politics of debasement. Critical Studi. Media Commun. **34**(1), 59–68 (2017)
36. Rezaie, B., Zahedi, M., Mashayekhi, H.: Measuring time-sensitive user influence in Twitter. Knowl. Inf. Syst. **62**(9), 3481–3508 (2020). https://doi.org/10.1007/s10115-020-01459-y
37. Rutishauser, N., Noureldin, A.: TPC-H applied to MongoDB: How a NoSQL database performs. Department of Informatik Vertiefung, University Zurich, Technical report (2012)
38. Shanbhag, A., Madden, S., Yu, X.: A study of the fundamental performance characteristics of GPUs and CPUs for database analytics. In: SIGMOD, pp. 1617–1632 (2020)
39. Sharma, M., Sharma, V.D., Bundele, M.M.: Performance analysis of RDBMS and NoSQL databases: PostgreSQL, MongoDB, and Neo4j. In: ICRAIE, pp. 1–5. IEEE (2018)
40. Stolee, G., Caton, S.: Twitter, trump, and the base: a shift to a new form of presidential talk? Signs Soc. **6**(1), 147–165 (2018)
41. Taipalus, T.: The effects of database complexity on SQL query formulation. J. Syst. Softw. **165**, 110576 (2020)
42. Thomas, A., Kumar, A.: A comparative evaluation of systems for scalable linear algebra-based analytics. PVLDB **11**(13), 2168–2182 (2018)
43. Van der Veen, J.S., Van der Waaij, B., Meijer, R.J.: Sensor data storage performance: SQL or NoSQL, physical or virtual. In: International Conference on Cloud Computing. IEEE (2012)
44. Xu, S., Zhou, A.: Hashtag homophily in Twitter network: examining a controversial cause-related marketing campaign. Comput. Hum. Behav. **102**, 87–96 (2020)
45. Yaqub, U., Chun, S.A., Atluri, V., Vaidya, J.: Analysis of political discourse on Twitter in the context of the 2016 US presidential elections. Gov. Inf. Q. **34**(4), 613–626 (2017)