Journal of Computer Science
https://dergipark.org.tr/en/pub/bbd
https://doi.org/10.53070/bbd.990869

Journal of Computer Science
ISSN: 2548-1304
Volume: IDAP-2021, Issue: Special, pp: 329-334
@Anatolian Science

Veri Göçü Sonrasında Çizge Veri Tabanları Üzerinde Desenlerin İncelenmesi

Investigation of Patterns on Graph Databases After Data Migration

Ramazan Altın[1], Ahmet Cumhur Kınacı[2]

[1]Department of Computer Engineering, Çanakkale Onsekiz Mart University, Çanakkale, Turkey

[2] Department of Computer Engineering, Çanakkale Onsekiz Mart University, Çanakkale, Turkey

(ramazan@comu.edu.tr, cumhur.kinaci@comu.edu.tr)

Received: Sep.3, 2021 Accepted: Sep.16, 2021 Published: Oct.20, 2021

Özetçe — İlişkisel veri tabanları uzun yıllardır verileri saklamak ve saklanan veriler üzerinde işlem yapmak amacıyla kullanılmaktadırlar. Büyük veri kavramının ortaya çıkması ile birlikte depolama ve performans sorunları ilişkisel veri tabanlarında çözüm aranan konular haline gelmiştir. Çok fazla tabloyu ilişkilendirmenin getirdiği maliyetin artması son yıllarda NoSQL veri tabanlarına ilgiyi arttırmıştır. NoSQL veri tabanı türü olan çizge veri tabanları düzensiz verileri iyi yönetmesi ve ölçeklenebilirliğinin kolay olması nedeniyle yaygın hale gelmektedir. Veri paylaşımının çok fazla olduğu sosyal medya uygulamaları başta olmak üzere pek çok yazılım, kullanmakta olduğu ilişkisel veri tabanlarından çizge veri tabanlarına geçmek için veri göçüne ihtiyaç duymaktadır. Bu çalışmada ilişkisel veri tabanından çizge veri tabanına iki farklı yöntem kullanarak veri göçü gerçekleştirebilen yazılım geliştirilmiştir. Her iki yöntemde herhangi bir veri kaybı olmaksızın veri göçü gerçekleştirilmiştir. Veri göçü sonucunda çizge veri tabanları üzerinde düğümler ve bağlantılar farklı şekillerde oluşturulabilmektedir. Bu sayede farklı ihtiyaçlara yönelik olan yazılımların kullanımına uygun şekilde çizge veri tabanları tasarlanabilmektedir. Kullanım alanına göre desen oluşturmanın sorgu süresine ve depolamaya olan etkileri çalışma kapsamında ele alınmıştır.

Anahtar Kelimeler: *İlişkisel veri tabanı, Çizge veri tabanı, Büyük veri, NoSQL*

Abstract — Relational databases have been used for many years to store data and to operate on stored data. With the emergence of big data, storage and performance problems have become issues that are sought for solutions in relational databases. Interest has been given to NoSQL databases in recent years due to the increasing cost of associating too many tables. Graph databases, a type of NoSQL database, are becoming common because they manage irregular data well and are easy to scale. Many software, especially social media applications where data sharing is very high, needs data migration from relational databases to graph databases. This study has developed software that can perform data migration from relational database to graph database using two different methods. In both methods, data migration was carried out without any data loss. As a result of data migration, nodes and connections can be created in different ways on graph databases. In this way, graph databases can be designed following the use of software for different needs. The effects of creating patterns according to usage area on query time and storage are discussed within the scope of the study.

Keywords: *Relational database, Graph database, Big data, NoSQL*

1. Giriş

İyi tasarlanmamış ilişkisel veri tabanlarında veri tutarsızlığı ortaya çıkmaktadır. Bir ilişkisel veri tabanında veri kayıplarını önlemek ve yapılacak sorgularda doğru çıktılar elde etmek için standartlar tanımlanmıştır (Haerder & Reuter, 1983). Bu standartlar veri güvenliği üzerinde yoğunlaşmış olup hız ile ilgili herhangi bir taahhütte bulunmazlar. NoSQL veri tabanlarında ise performans bakımından öne çıkan yeni tanımlamalar getirilmiştir. BASE teoremi olarak bilinen bu tanımlama verilerin daha hızlı bir şekilde kullanılmasını hedeflemektedir (Tang, E. & Fan, Y, 2016). Bir NoSQL veri tabanının sahip olduğu en önemli özelliklerden birisi ise yatay ölçeklenebilmesidir. İlişkisel veri tabanlarında veri miktarı büyüdükçe sistem dikey ölçeklenme ile büyüyecektir. Dikey ölçeklenme yatay ölçeklenmeye göre daha maliyetli bir yöntem olup sistem arızalarında yaşanacak sorunlar çok büyük olabilmektedir. Bu sebepler ya da daha farklı ihtiyaçlar sonucunda bir ilişkisel veri tabanından NoSQL veri tabanına verilerin taşınması gerekebilmektedir.

Çalışma kapsamında veri göçü gerçekleştirilecek olan ilişkisel veri tabanı olarak MySQL veri tabanı seçilmiş olup aktarım için iki yöntem sunulmuştur. MySQL veri tabanında veriler satır ve sütunların kesişiminden oluşan hücreler üzerinde tutulmaktadır. Eğer ilişkili veriler farklı tablolar üzerinde yer alıyor ise yabancı anahtarlar ile gerekli ilişki kurulabilmektedir.

Çizge veri tabanı olarak seçtiğimiz Neo4j ise günümüzde en çok tercih edilen açık kaynak kodlu bir yazılımdır. Neo4j ilişkisel veri tabanlarının sahip olduğu ACID özelliklerine sahip olmasının yanında bir NoSQL veri tabanın hızına ayrıca sahiptir (Miller, J. J. ,2013). Neo4j veri tabanı, Cypher adında kendisine ait sade ve anlaşılır bir sorgu diline sahiptir.

Çalışmada Java programlama dili ile bir yazılım geliştirilmiş olup veri göçü işlemleri Cypher ve SQL sorgu dillerinden de faydalanılarak gerçekleştirilmiştir. Uygulamada ilişkisel veri tabanı olarak veri göçü çalışmalarında sıkça kullanılan Sakila[1] veri tabanı kullanılmıştır (Jia T. et al, 2016). Tüm verilerin düzenli bir şekilde aktarılmasının sonunda farklı görünümlere sahip desenlerin elde edilebildiği gösterilmiştir.

2. Materyal ve Yöntem

2.1. Veri Göçü Yöntemleri

Veri göçü için geliştirdiğimiz birinci yöntemde sadece veri tabanı kullanıcı adı ve parola girişi yapılacak şekilde JDBC üzerinden MySQL veri tabanına bağlanılarak tüm tablolar aktarılabilmektedir. Yazılan sorgular ile ilişkisel veri tabanında tanımlanmış olan kriterler var ise bunların kontrolü gerçekleştirilmiştir. Veri göçü gerçekleştirilmeden önce boş veri içeren sütunlar tespit edilerek aktarım sürecine dahil edilmemiştir. SQL sorguları ile birincil anahtar olarak tanımlanmış tüm sütunlar çizge veri tabanına aktarılırken kontrol edilmektedir. İlişkisel veri tabanı tasarlanırken benzersiz olması istenilen alanlar tanımlanmış ise bunların tespiti yapılmaktadır. Bu alanların bulunması durumunda oluşturulacak düğümler çizge veri tabanı içerisinde yine benzersiz olarak tanımlanmaktadır.

Bir diğer veri göçü yöntemi ise daha kullanışlı olacağını düşündüğümüz ve farklı ilişkisel veri tabanları için genel bir çözüm sunan JSON dosyaları üzerinde gerçekleştirilmektedir. İlişkisel verileri JSON formatında saklamak ve yönetmek yaygınlaşmakta olup üzerinde yapılan veri okuma süreleri oldukça hızlıdır (Petković, D. ,2017). Veri tiplerinin çizge veri tabanı üzerinde tek bir tipte tutulduğu bu yöntemde ilişkisel veri tabanına ait tüm veriler JSON dosya formatında dışa aktarılmış şekilde girdi olacaktır. Her bir tablo için girdi olarak bir JSON dosyası okunarak veri göçü gerçekleştirilmekte olup daha kısa sürede işlem gerçekleştirilebilmektedir.

İlişkisel veri tabanı olsun ya da olmasın JSON dosya formatında verilecek bir veri tabanı bu yöntem sayesinde çizge veri tabanı üzerinde veri kayıpları yaşanmadan taşınabilmektedir. İhtiyaç duyulması

[1] https://dev.mysql.com/doc/sakila/en

durumunda NoSQL veri tabanlarından çizge veri tabanlarına veri göçü yapılmasına olanak sağlaması adına yöntem önemli bir avantaj sağlamaktadır. Tablo içerikleri ile birlikte yabancı anahtar tanımlamaları da ilişkisel veri tabanına bağlandıktan sonra JSON formatında dışa aktarılabilmektedir (Şekil 1).


```
{"fks":
[["film_actor","actor_id","actor","actor_id"],
["customer","address_id","address","address_id"],
["staff","address_id","address","address_id"],
["store","address_id","address","address_id"],
["film_category","category_id","category","category_id"],
["address","city_id","city","city_id"],
["city","country_id","country","country_id"],
["payment","customer_id","customer","customer_id"],
["rental","customer_id","customer","customer_id"],
["film_actor","film_id","film","film_id"],
["film_category","film_id","film","film_id"],
["inventory","film_id","film","film_id"],
["rental","inventory_id","inventory","inventory_id"],
["film","language_id","language","language_id"],
["film","original_language_id","language","language_id"],
["payment","rental_id","rental","rental_id"],
["payment","staff_id","staff","staff_id"],
["rental","staff_id","staff","staff_id"],
["store","manager_staff_id","staff","staff_id"],
["customer","store_id","store","store_id"],
["inventory","store_id","store","store_id"],
["staff","store_id","store","store_id"]]}
```

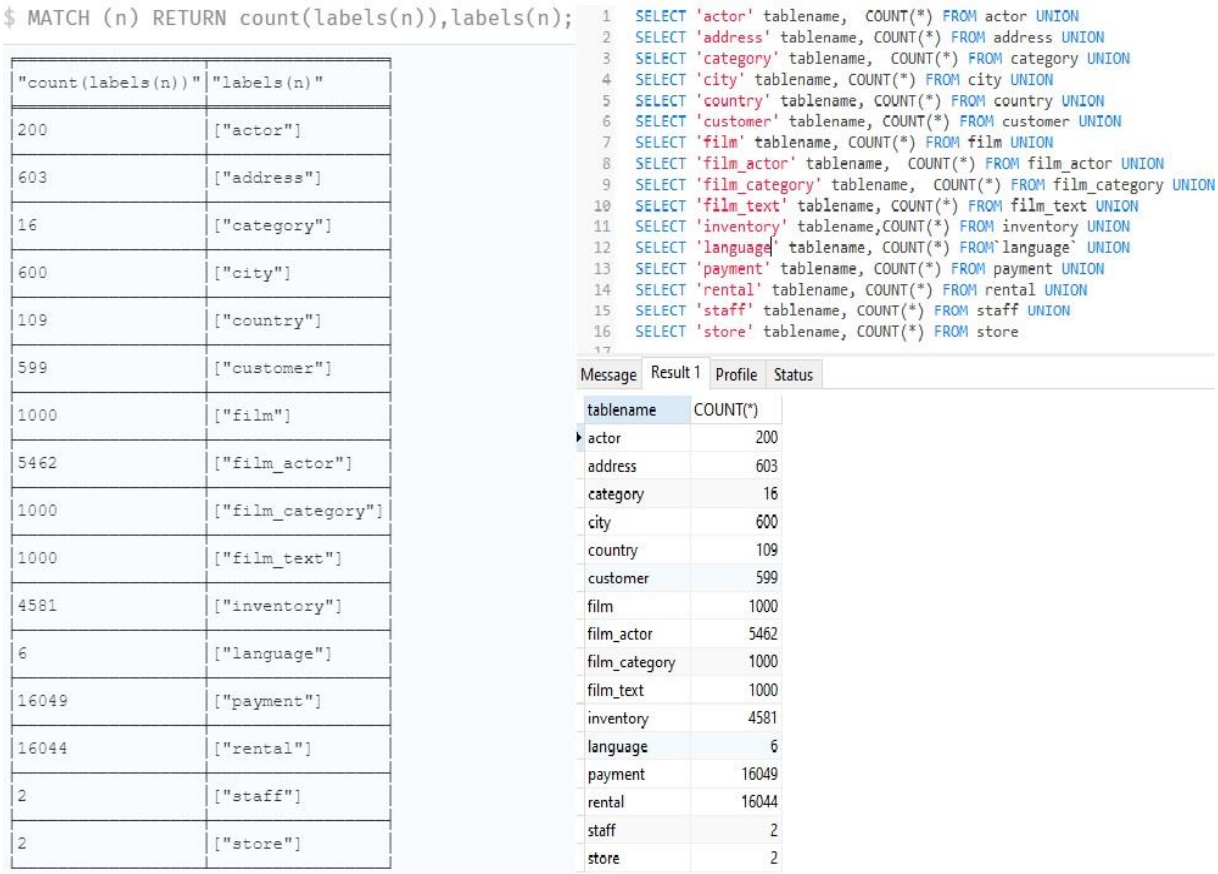

Şekil 1: Yabancı anahtar bilgilerinin Json formatında gösterimi

Her iki veri göçü yöntemi sonrasında ilişkisel veri tabanı ve çizge veri tabanı karşılaştırılmıştır. Cypher sorgusunda "COUNT" fonksiyonu ile tüm etiketler ve sayıları çıktı alınmıştır. MySQL üzerinde ise tablo adları ve her tablonun satır sayısı sorgulanmış "UNION" komutu ile birleştirilerek tek bir çıktı elde edilmiştir. İlişkisel veri tabanında yer alan tüm tabloların satır sayıları ile çizge veri tabanındaki etiket sayıları eşit olacak şekilde verilerin tamamı eksiksiz aktarılmıştır (Şekil 2).

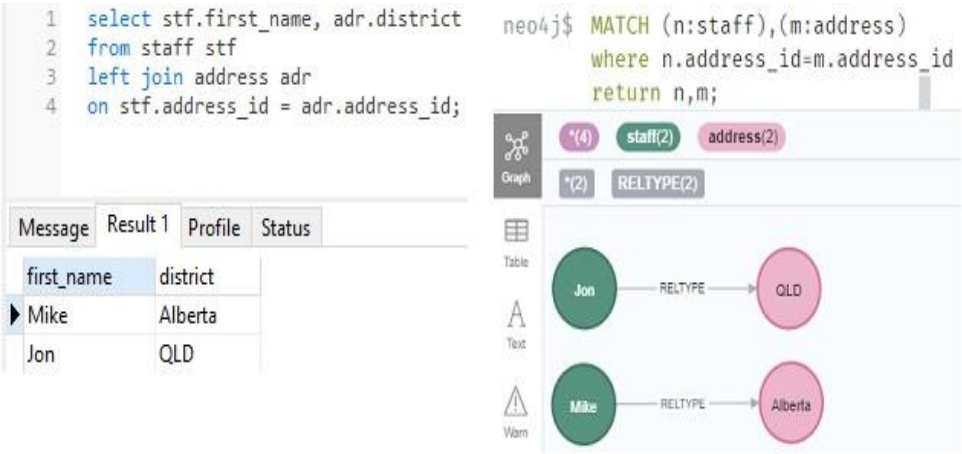
Veri göçü işleminde son adım olarak bağlantı bulunan tüm alanların çizge veri tabanında ilişkilendirilmesi yapılmıştır. Şekil 1'de yer alan yabancı anahtar bilgileri kullanılarak ilişkiler kurulmaktadır. Bu işlem göç sonrasında şu şekilde yapılmaktadır:

- Yabancı anahtar bilgilerini içeren Json dosyası satır satır okunur.
- Json dosyasında yer alan satırlarda yabancı anahtar tanımlamaları fk_tablo_adı, fk_sütun_adı, referans_tablo_adı, referans_sütun_adı şeklinde oluşturulmuştur.
- fk_tablo_adı altında bulunan fk_sütun_adı ile referans_tablo_adı altında bulunan referans_sütun_adının ilişkili olduğu bilgisi alınır.
- Çizge veri tabanında fk_tablo_adı etiketine sahip düğümler bulunur.
- Bulunan düğümlerin fk_sütun_adı alanı ile referans_tablo_adı etiketine sahip olan düğümlerin referans_sütun_adı alanı eşleşenler tespit edilir.
- Eşleşen yabancı anahtarlar için iki düğüm arasında RELTYPE adında bir ilişki oluşturulur.
- Tüm satırlar için bu işlem uygulanır.

Veriler çizge veri tabanına aktarıldıktan sonra bu işlemler sonucunda tüm ilişkili düğümler birbirleri ile bağlanmaktadır. Şekil 3'te yer alan örnek SQL sorgusunda iki tablo arasında oluşturulmuş yabancı anahtar bilgisi ile çalışanların adresleri bulunmuştur. Veri göçü sonrasında aynı çıktıya kurulan ilişkiler kullanılarak Cypher sorgusu ile erişilebildiği görülmektedir.



Şekil 2: İlişkisel ve çizge veri tabanlarında veri göçü sonrası karşılaştırma



Şekil 3: Veri göçü sonrası ilişkilerin sorgulanması

Çizge veri tabanında ilişkilere isim vermek ve ilişki üzerinde veri tutmak mümkün olmaktadır. İlişki adı için veri göçü sırasında ek bir tanımlama yapılmamıştır. İhtiyaç olması durumunda erişimin az olduğu veriler ilişki üzerinde tutulabilir ve ilişki adlarına isim verilerek çizge veri tabanında veriler arasındaki ilişkinin daha anlaşılır olması sağlanabilir.

2.2. Veri Göçü Sonrası Oluşabilecek Desenler

İlişkisel veri tabanları oluşturulurken tek bir tasarıma bağlı kalınarak tüm veriler satır ve sütunların kesiştiği hücreler üzerinde tutulmaktadırlar. Tabloların ilişkilerini görüntülemek için ER diyagramlar kullanılsa bile çizge veri tabanlarına göre anlaşılırlık daha zor olmaktadır. Neo4j veri tabanında verileri kullanım amacına göre farklı desenler oluşturacak şekilde tasarlamak mümkün olmaktadır. Bu sayede

veri tabanı tasarımcısı yazılımın gereksinimlerini göz önüne alarak ilişki ve düğümleri bu doğrultuda oluşturabilmektedir. Ayrıca oluşan desenler üzerinde sorgu yaparken indeksleme yaparak daha kısa sorgu süreleri elde edilebilmektedir (Pokorný et al, 2018). Çalışmada en çok ihtiyaç duyulacak verileri analiz ederek bu verilere erişimin daha hızlı olacağı tasarımlar oluşturulabilmektedir. Farklı desenler için farklı sorgu süreleri ve farklı düğüm-ilişki sayıları elde edilmektedir. Veri göçü gerçekleştirilecek veri tabanlarından iki farklı desen oluşturulmuştur.

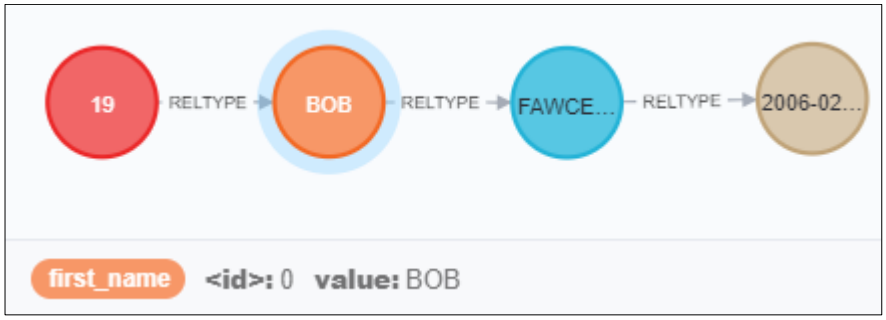
2.2.1.Satır-Düğüm Deseni (Row-to-Node)

İlişkisel veri tabanı üzerinde yer alan bir tablo üzerinde bulunan satırdaki verilerin tamamı bir bütün olarak ele alınacak olursa bu verileri tek bir düğüm üzerinde temsil etmek kullanışlı olacaktır. Bir kişiye ait isim, soy isim bilgileri genelde ilişkisel veri tabanlarında aynı tablo üzerinde tutulmaktadır. Kişiye ait bilgiler görüntülenmek istendiğinde ise bu iki bilgi aynı sorgu üzerinden tek bir seferde çekilmektedir. Bu gibi durumlarda isim ve soy isim bilgilerini tek bir düğüm üzerinde tutmak çizge veri tabanlarında kolaylık sağlamaktadır. Bu amaçla yapılacak olan veri göçü işleminde düğüm adı olarak tablo adı tanımlanmış olup isim ve soy isim bilgileri etiket olarak bu düğümün üzerinde tutulmuştur (Şekil 4).

Şekil 4: İlişkisel veri tabanında bulunan bir satırın düğüm olarak oluşturulması

2.2.2.Hücre-Düğüm Deseni (Cell-to-Node)

Çalışmada oluşturulan bir diğer desen ise her düğümün içerisinde yalnızca tek bir veri tutulacak şekilde oluşturulan veri göçü yöntemidir (Şekil 5). İlişkisel veri tabanında satır ve sütunların kesiştiği alanlara hücre denilmektedir. Çizge veri tabanına veri göçü gerçekleştirilirken bir hücre bir düğüme karşılık gelecek şekilde aktarım yapılmıştır. Özellikle doğal dil işleme çalışmalarında kelime etiketleme yapılacağında bu desenin kullanılması uygun olacaktır. Cümleyi oluşturan her bir kelime farklı düğümler üzerinde tutulacak şekilde çizge veri tabanında oluşturulduğunda bir sonraki kelimenin ne olabileceği daha kolay bulunabilecektir. Yine kelimenin sıfat, zarf, isim, edat, bağlaç mı olduğunu anlamaya çalışan kelime türü tespiti çalışmalarında bu desen kullanılabilecektir.



Şekil 5: Her hücrenin tek bir düğüm olarak oluşturulması

Oluşturulan Satır-Düğüm deseninde her düğüm bir nesnenin özelliklerini tümüyle taşıyacak şekilde tasarlanmıştır. Çalışmada kullanılan Sakila veri tabanı için Satır-Düğüm desenine göre Hücre-Düğüm deseninde daha fazla ilişki ve düğüm oluşturulmaktadır. Ancak tekrarlı verilerin fazla olduğu veri tabanlarında Hücre-Düğüm deseni kullanıldığında tekrar eden hücreler için yeniden düğüm

oluşturulmamaktadır. Bunun için düğümde oluşturulmuş olan “value” alanının kontrol edilmektedir. Var olan düğüme yeni bir ilişki eklenerek bir sonraki desen oluşturulmaya başlanmaktadır.

3. Sonuç

Çizge veri tabanı oluşturulurken farklı ihtiyaçlara göre farklı desenler elde edilebileceği gösterilmiş olup çalışma alanlarına göre hangi desenin kullanılmasının doğru olabileceğine dair öneriler sunulmuştur. İlişkisel veri tabanlarından çizge veri tabanlarına veri göçü gerçekleştirmek için iki farklı yöntem ile nasıl aktarılacağı hakkında bilgi verilmiştir. Aktarım sonucu oluşan çizge veri tabanının ilişkisel veri tabanıyla tutarlılığını kontrol etmek için yapılacak işlemler önerilmiştir. İlişkisel veri tabanlarındaki verilerin çizge veri tabanlarına aktarımı sırasında oluşturulabilecek iki temel dönüşüm deseni incelenmiş olup birbirlerine göre avantajları ve dezavantajları belirtilmiştir. Daha karmaşık dönüşüm desenlerinin incelenmesi ve bunların karşılaştırılması için metriklerin geliştirilmesi ilerleyen çalışmalarda hedeflenmiştir.

4. Kaynaklar

Haerder, T., & Reuter, A. (1983). Principles of transaction-oriented database recovery. ACM Computing Surveys (CSUR), 15(4), 287–317. https://doi.org/10.1145/289.291

Tang, E., & Fan, Y. (2016, November). Performance comparison between five NoSQL databases. In 2016 7th International Conference on Cloud Computing and Big Data (CCBD) (pp. 105-109). IEEE.

Miller, J. J. (2013, March). Graph database applications and concepts with Neo4j. In Proceedings of the southern association for information systems conference, Atlanta, GA, USA (Vol. 2324, No. 36).

Jia, T., Zhao, X., Wang, Z., Gong, D., & Ding, G. (2016, June). Model transformation and data migration from relational database to MongoDB. In 2016 IEEE International Congress on Big Data (BigData Congress) (pp. 60-67). IEEE.

Petković, D. (2017). JSON integration in relational database systems. Int J Comput Appl, 168(5), 14-19.

Constantinov, C. Iordache, L., Georgescu, A., Popescu, P. Ş., & Mocanu, M. (2018, October). Performing social data analysis with neo4j: Workforce trends & corporate information leakage. In 2018 22nd International Conference on System Theory, Control and Computing (ICSTCC) (pp. 403-406). IEEE.

Pokorný, J., Valenta, M., & Troup, M. (2018, July). Graph Pattern Index for Neo4j Graph Databases. In International Conference on Data Management Technologies and Applications (pp. 69-90). Springer, Cham.