T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ
VERİ BİLİMİ VE BÜYÜK VERİ
TEZLİ YÜKSEK LİSANS PROGRAMI

BLM 5117: Veri Tabanı Sistemlerinin Gerçeklenmesi

Homework-2

METİN USLU
235B7014

Ders Sorumlusu
Dr. Öğr. Üyesi Mustafa Utku Kalay

İstanbul
Ocak, 2025

Kavramlar: SQL, NoSQL ve NewSQL

SQL (Structured Query Language)

SQL, yapılandırılmış sorgu dilidir. Bu kavramı biraz daha açıklayacak olursak eğer ilişkisel Veri Tabanı Yönetim Sistemleri (Relational Data Base Managment System) üzerinde standart bir sorgu dilidir. SQL, verileri yönetmek ve tasarlamak için kullanılan bir dildir. Bir programlama dili değildir fakat geleneksel veri tabanları üzerinde defacto ve standartla olduğunu ifade edebiliriz. Tarihçesine biraz değinecek olursak; SEQUEL kelimesi zamanla İngilizce söyleşi ile aynı şekilde ifade edilmiştir. Konunun derinlerine inildiğinde SQL, SEQUAL'ın yani "the Structured English QUERy Language" kısaltılmış halidir.[1] Bu konudaki bir diğer görüş ise orijinal ismin o zamanki tescilli statüsü nedeniyle SEQUEL'in adı SQL olarak değiştirilmek zorunda kaldı.[2,3]

Tarihçesine baktığımızda; 1970'lerde Edgar F. Codd öncülük ettiği, ardından 1974'te IBM System R ile bir veri tabanı yönetim sistemi geliştirdiği, devamında 1979'da Oracle tarafından SQL ticari olarak piyasaya sürüldüğü ve son olarakta 1986'da Amerikan Ulusal Std. Enstitüsü tarafından SQL ilk stardartlarının yayınlandığı (son olarak SQL-2019 [4]) görülmektedir.

Büyük ve bilinen (MySQL, PostgreSQL, Microsoft SQL Server, Oracle.) veri tabanları üzerinde yapılandırılmış tabloları sorgulamak için kullanılan bir dildir. SQL'in anahtar özellikleri olarak Veri Tanımlama Dili (DDL), Veri Manipülasyon Dili (DML), Veri Kontrol Dili (DCL) kavramlarının önemli olduğunu söyleyebiliriz.

NoSQL (Not Only SQL)

NoSQL, bilinen anlamda SQL'in aksine yarı yapılandırılmış (semi structured) ve yapılandırılmamış (unstructered) veriler üzerinde esnek ve ölçeklenebilir veri tabanı türüdür. Tarihçesine baktığımızda 2000'lerin başı olarak gözüküyor olsa da özellikle Web ve Mobil Uygulama Teknolojilerinin ve gelişmesi ve farklı türde verilerin ve ihtiyaçların ortaya çıkması ile popülerliğini kazandığını söyleyebiliriz. SQL ile en temelde farklılaştığı yer; Veri Yapısı (Yapısal vs Yarıyapısal & Yapısal Olmayan) ve Ölçeklenebilirliği olarak ifade edebiliriz. NoSQL veri tabanları kullanım alanlarına göre Column Base(Cassandra, HBase), Document Base(MongoDB), Key&Value(Redis) ve Graph(Neo4j) türlerinde olmak üzere farklı niteliklere sahip veri tabanları bulunmaktadır. [5, 6]

SQL veri tabanlarında olup NoSQL veri tabanları üzerinde (bir tanesi ya da tamamı) olmayan özellikler;

1. Katı Şema (Schema)
2. İlişkiler (Relation)
3. ACID
4. Standart Sorgu Dilinin Olmaması

en temelde bu kavramlar üzerinde farklılaştığı ifade edebiliriz. NoSQL veri tabanlarında genel olarak sosyal medya verileri, kullanıcı mesajlaşmaları yada kullanıcı yorumları, sensör verileri gibi amaçlar için sıklıkla tercih edilmektedir. CAP teorimi açısından baktığımızda;

- NoSQL veri tabanları genellikle yüksek kullanılabilirlik ve ölçeklenebilirlik sağlamak için tasarlandığından, genellikle Erişilebilirlik(Availability) ve Bölünmeye Tolerans(Partition Tolerance) özelliklerine öncelik verirler. Bu da, Tutarlılık(Consistency) özelliğinden ödün verme anlamına (eventual consistency) gelir.
- İlişkisel veri tabanları(RDBMS) ise genellikle merkezi bir sunucu üzerinde çalışır ve tek bir düğümde tutarlılık sağlamaya odaklanır. Bu nedenle, genellikle tutarlılık (consistency) ve erişilebilirlik (availability) özelliklerine öncelik verirler.

NewSQL

NewSQL, SQL ve NoSQL veri tabanlarının en temel güçlü özellikleri üzerine inşa edilmiş yeni nesil bir veri tabanı sistemidir. NewSQL, RDBMS'lerin sağlamlığını ve NoSQL'in esnekliğini bir araya getirmeyi amaçlar. Yukarıda bahsedildiği üzere, SQL ve NoSQL veri tabanları farklı amaçlara sorunlara yönelik çözümler ürettikleri için bazı güçlü özellikleri var. Anahtar Özelliklerine baktığımızda; ACID, Ölçeklenebilirlik, SQL Desteği, Dağıtık Mimari olarak ifade edebiliriz. [5] NewSQL, SQL'in ACID garantilerini NoSQL'in ölçeklenebilirliği ve yüksek performansıyla birleştirmeyi hedefleyen modern bir ilişkisel veri tabanıdır. Google Spanner, VoltDB örnek olarak verilebilir. Aşağıdaki tabloda SQL, NoSQL ve NewSQL veri tabanlarına ait anahtar özellikler yer almaktadır.

SQL vs NoSQL vs NewSQL

FEATURES	SQL	NO SQL	NEW SQL
Schema	It is schema-fix.	It is schema-free.	It is both schema-fix and schema-free.
Base Properties/Theorem	It strictly follows ACID properties.	It follows the CAP theorem.	It takes care of ACID properties.
Security	It is secure.	It is less secure.	It is moderately secure.
Databases	No distributed database.	Distributed database.	Distributed database.
Query Language	It supports SQL as a query language.	It does not support old SQL but supports UQL.	It supports SQL with improved functions and features.
Scalability	It is vertically scalable.	It is only vertically scalable.	It is both vertically and horizontally scalable.
Types of database	Relational database.	Non-relational database.	Relational database but not purely.
Online processing	Online transaction processing but not full functionality.	Online analytical processing.	Online transaction processing with full functionality.
Query Handling	Simple queries can be handled.	Complex queries can be directed better than SQL.	Highly efficient for complex queries.
Example	MySQL	MongoDB	Cockroach DB.

Veri Tabanı Sisteminin Genel Tanıtımı: MongoDB (NoSQL Database)

MongoDB, doküman tabanlı bir NoSQL veritabanıdır. Veriler JSON benzeri bir yapıda BSON (Binary JSON) olarak binary formatta tutulur. Bu verinin modellenmesine esneklik ve depolama da verimlilik sağladığını ifade edebiliriz. Tarihçesine baktığımızda 2007 yılında MongoDB Inc (10Gen) tarafından C++ dili ile geliştirilmeye başlanılıyor ardından 2009 yılında ise açık kaynak olarak sunuluyor. MongoDB hız gerektiren durumlarda ve de dağıtık olması sebebiyle yüksek erişilebilirlik ve yatay ve dikeyde ölçeklenebilirlik sağlanmaktadır.[5, 6, 7, 8]

MongoDB Anahtar Özellikleri

- Belge Tabanlı (BSON formatında)

- Şema Gerekliliğinin Olmaması
- Dağıtık Mimari
- Yüksek Ölçeklenebilirlik & Yüksek Performans

Kullanım Alanları

MongoDB bir NoSQL veri tabanı olduğu için klasik anlamda İlişkisel Veri Tabanlarının (RDBMS)'in std. olarak kullanıldığı ERP, CRM, vd. uygulamalardan ziyade yarı yapısal ve yapısal olmayan veriler için Büyük Veri hacimlerinin tutulması ve analizi, Web ve Mobil uygulamaların bazı servislerinde kullanılmaktadır. Ben ise LCW'de bir Makine Öğrenmesi Modelinin (RestApi) servis bacağında Model Sonuçlarının loglanması amacıyla kullandığımı ifade edebilirim.

Geleneksel Yaklaşımlar: Pipeline ve Materialized Sorgu İşleme Yapıları

Pipeline Sorgu İşleme Yapıları

Pipeline sorgu işleme, veri tabanı sorgularının bir dizi adımda işlendiği bir yapıdır. Bu adımlar, bir boru hattındaki aşamalar gibi sıralı olarak çalışır. Her adım, önceki adımın çıktısını alır ve kendi işlemini uygular. Bu yaklaşım, özellikle karmaşık ve çok adımlı sorgular için performans artışı sağlayabilir. Bu yaklaşım özetleyecek olursak;

1. Adım: Sorgular daha küçük ve yönetilebilir bir hale getirilir.
2. Adım: Sorgular birbiri sıra işlenir burada önceki sorgu sonraki sorguya girdi olacaktır.
3. Adım: Her adım birbirinden bağımsızdır, ayrı ayrı optimize edilebilirler.

Pipeline avantajları olduğu gibi bazı durumlarda dezavantajları da bulunmaktadır. Bunlar karmaşık sorgular ve de bu sorgular üzerinde hata ayıklama aşamasında daha fazla süreye ihtiyaç duyabilmektedir.

Materialized Sorgu İşleme Yapıları

Materialized sorgu işleme ise soru sonuçlarının önceden hesaplanıp saklanması işlemidir. Bu önceden hesaplanmış sonuçlar, bir materialized view olarak adlandırılır. Ve böylece, aynı sorgu tekrar çalıştırıldığında veritabanı, sonucu yeniden hesaplamak yerine materialized view'den doğrudan alabilir. Bu yaklaşım özetleyecek olursak;

1. Adım: Sık kullanılan sorgunun önceden hesaplanır ve çıktısı materialized view olarak saklanır.
2. Adım: Aynı sorgu yeniden çalıştırıldığında materialized üzerinden sonuçlar doğrudan alınır.
3. Adım: Eğer veri üzerinde (verinin kendisi ve yapısı olarak düşünebiliriz) bir güncelleme söz konusu ise materialized view bunu periyodik olarak günceller.

Materialized yaklaşımı performans ve hız olarak avantajları öne çıkıyorken, depolama maliyeti de bir dezavantaj olarak görülebilir. Her iki yaklaşımı da özetlemesi açısından aşağıdaki tabloyu paylaşıyorum. [9, 10]

Özellik	**Pipeline Sorgu İşleme**	**Materialized Sorgu İşleme**
İşleme Zamanı	Sorgu her çalıştırıldığında hesaplanır	Önceden hesaplanır ve saklanır
Esneklik	Yüksek	Düşük (önceden tanımlanmış sorgular için)
Depolama	Genellikle daha az	Daha fazla (materialized view'ler için)
Güncelleme	Her adım bağımsız olarak güncellenebilir	Periyodik olarak güncellenir
Kullanım Alanları	Karmaşık, dinamik sorgular	Sık kullanılan, sabit sorgular

Hangi Yapıyı Ne Zaman Kullanmalıyız?

Pipeline: Karmaşık analizler, gerçek zamanlı işleme ve sürekli değişen veri akışları için idealdir.
Materialized View: Sık kullanılan raporlama sorguları, önceden hesaplanmış metrikler ve veri keşfi için uygundur.

MongoDB üzerine Pipeline ve Materialized Sorgu İşleme Yapıları

Ön Bilgi: Bu başlığa başlamadan evvel ek bir bilgi olarak; farklı veri tabanı sistemleri, bu yapıları farklı şekillerde destekler. Örneğin, PostgreSQL, Oracle ve SQL Server gibi RDBMS'ler her iki sorgu işleme yapılarını desteklerken, MongoDB gibi NoSQL veri tabanları pipeline sorgu işleme için daha uygun olabilmektedir. Yukarıda RDBMS ve NoSQL kullanım amaçlarını ve anahtar özelliklerini de hatırlayacak olursak bunların veri tabanı tipi (SQL, NoSQL) özelinde ayrışması da bir o kadar normaldir. Yine unutulmamalıdır ki bu iki sorgulama yaklaşımı birbirini dışlamaz, yani X veri ambarı üzerinde hem de Pipeline hem de Materialized Sorgu İşleme yapısını kullanabiliriz.

Pipeline Sorgu İşleme

MongoDB'de pipeline, bir dizi işlemden oluşan bir yapıdır. Bu işlemler, verileri filtrelemek, sıralamak, gruplamak ve dönüştürmek gibi çeşitli amaçlar için kullanılır. Pipeline, MongoDB'nin aggregation framework'ünün temelini oluşturur.

MongoDB üzerinden bir pipeline, birçok aşamadan oluşur. Her aşama, önceki aşamanın çıktısını alır ve kendi işlemini uygular. Bu sayede, karmaşık sorguları daha okunaklı ve yönetilebilir hale getirebiliriz. Örnek olarak; bir E-Ticaret Platformu üzerinde bir ürün koleksiyonundaki tüm ürünlerin toplam satış miktarını bulmak için bir pipeline oluşturabiliriz. Bu pipeline, önce ürünlere göre gruplama, ardından her gruptaki satış miktarını toplama gibi aşamalardan oluşacaktır.

Aggregation Framework: MongoDB'nin aggregation framework'ü, pipeline'ları oluşturmak için kullanılan bir dizi işlem ve operatör sağlar.[12]

Materialized Sorgu İşleme

Materialized view, yukarıda da bahsettiğimiz üzere sık kullanılan sorguların hesaplanarak sorgu sonuçlarının koleksiyon(collection) olarak kaydedilerek ardından yeniden aynı sorgunun istenmesi halinde sonucun hızlı bir şekilde getirilmesidir. Örnek kullanım olarak; Bir e-ticaret sitesindeki günlük satış raporları için bir materialized view oluşturabilirsiniz. Bu view, her gün otomatik olarak güncellenerek en güncel satış verilerini sunar.

Change Streams: MongoDB'de gerçek zamanlı veri değişikliklerini izlemek için kullanılan bir mekanizmadır. Change streams ile birlikte materialized view'ler kullanarak gerçek zamanlı analizler yapabilirsiniz.[13]

Yukarıdaki her iki sorgu işleme yapısı için örnekleri aşağıdaki GitHub Repom'da paylaşıyor ve bunları demonstrate ediyor olacağım.

Github Repo: https://github.com/metinuslu/blm5117_dbms_hw2

Kaynaklar

1. https://en.wikipedia.org/wiki/SQL
2. https://en.m.wikipedia.org/wiki/Edgar_F._Codd
3. https://www.reddit.com/r/compsci/comments/e2ip8x/how_do_you_pronounce_sql/
4. https://celerdata.com/glossary/ansi-sql
5. https://www.geeksforgeeks.org/sql-vs-no-sql-vs-new-sql/
6. https://www.mongodb.com/resources/basics/databases/nosql-explained
7. https://tr.wikipedia.org/wiki/MongoDB
8. https://www.geeksforgeeks.org/mongodb-tutorial/
9. https://www.datacamp.com/tutorial/views-in-sql
10. https://www.geeksforgeeks.org/differences-between-views-and-materialized-views-in-sql/
11. https://www.mongodb.com/docs/manual/aggregation/
12. https://www.mongodb.com/developer/products/mongodb/introduction-aggregation-framework/
13. https://www.mongodb.com/resources/products/capabilities/change-streams