NOSQL Sitemlerine Giriş / Elmasri
24.1 Introduction to NOSQL Systems

3.
24.1.1 Emergence of NOSQL Systems
Çoğu NOSQL sistemi, anında veri tutarlılığına, güçlü sorgu dillerine ve yapılandırılmış veri depolamasına vurgu yapmak yerine yarı yapılandırılmış veri depolama, yüksek performans, kullanılabilirlik, veri çoğaltma ve ölçeklenebilirliğe odaklanan dağıtılmış veritabanları veya dağıtılmış depolama sistemleridir.

4…….
Tüm bu e-postaları yönetebilecek bir depolama sistemine ihtiyaç vardır; Yapılandırılmış bir ilişkisel SQL sistemi uygun olmayabilir çünkü (1) SQL sistemleri bu uygulamanın ihtiyaç duymayabileceği çok fazla hizmet (güçlü sorgulama dili, eşzamanlılık kontrolü vb.) sunar; ve (2) geleneksel ilişkisel model gibi yapılandırılmış bir veri modeli çok kısıtlayıcı olabilir.

5…….
■ Google, Gmail, Google Haritalar ve Web sitesi indeksleme gibi büyük miktarda veri depolaması gerektiren Google uygulamalarında kullanılan BigTable olarak bilinen tescilli bir NOSQL sistemi geliştirdi. Apache Hbase, benzer kavramlara dayanan açık kaynaklı bir NOSQL sistemidir. Google'ın yeniliği, sütun tabanlı veya geniş sütunlu mağazalar olarak bilinen NOSQL sistemleri kategorisine yol açtı; bazen sütun ailesi mağazaları olarak da adlandırılırlar.
■ Amazon, Amazon'un bulut hizmetleri aracılığıyla kullanılabilen DynamoDB adlı bir NOSQL sistemi geliştirdi. Bu yenilik, anahtar-değer veri depoları veya bazen anahtar-tuple veya anahtar-nesne veri depoları olarak bilinen kategoriye yol açtı.
■ Facebook, şu anda açık kaynak olan ve Apache Cassandra olarak bilinen Cassandra adlı bir NOSQL sistemi geliştirdi. Bu NOSQL sistemi, hem anahtar-değer depolarından hem de sütun tabanlı sistemlerden kavramları kullanır.

6……
■ Diğer yazılım şirketleri kendi çözümlerini geliştirmeye ve bunları, örneğin belge tabanlı NOSQL sistemleri veya belge depoları olarak sınıflandırılan MongoDB ve CouchDB gibi bu yeteneklere ihtiyaç duyan kullanıcılara sunmaya başladı.
■ NOSQL sistemlerinin başka bir kategorisi, grafik tabanlı NOSQL sistemleri veya grafik veritabanlarıdır; Bunlar, diğerleri arasında Neo4J ve GraphBase'i içerir.
■ OrientDB gibi bazı NOSQL sistemleri, yukarıda tartışılan birçok kategoriden kavramları birleştirir.
■ Yukarıda listelenen daha yeni NOSQL sistemleri türlerine ek olarak, veritabanı sistemlerini nesne modeline (bkz. Bölüm 12) veya yerel XML modeline (bkz. Bölüm 13) dayalı olarak NOSQL sistemleri olarak sınıflandırmak da mümkündür, ancak bunlar böyle olmayabilir. diğer NOSQL sistemlerinin yüksek performans ve replikasyon özelliklerine sahiptir.

7……
24.1.2 NOSQL Sistemlerinin Özellikleri
Dağıtılmış veritabanları ve dağıtılmış sistemlerle ilgili NOSQL özellikleri.
1. Ölçeklenebilirlik:
Yatay ve dikey. NOSQL sistemlerinde, genellikle, veri hacmi büyüdükçe veri depolama ve işleme için daha fazla düğüm eklenerek dağıtılmış sistemin genişletildiği yatay ölçeklenebilirlik kullanılır. Dikey ölçeklenebilirlik ise mevcut düğümlerin depolama ve bilgi işlem gücünü genişletmeyi ifade eder. NOSQL sistemlerinde, sistem çalışır durumdayken yatay ölçeklenebilirlik kullanılır, bu nedenle mevcut verileri yeni düğümler arasında sistem çalışmasını kesintiye uğratmadan dağıtmak için teknikler gereklidir.
2. Kullanılabilirlik, Çoğaltma ve Nihai Tutarlılık:
NOSQL sistemlerini kullanan birçok uygulama, sürekli sistem kullanılabilirliği gerektirir. Bunu başarmak için, veriler iki veya daha fazla düğüm üzerinde şeffaf bir şekilde çoğaltılır, böylece bir düğüm başarısız olursa,

veriler diğer düğümlerde hala kullanılabilir durumda olur. Çoğaltma, veri kullanılabilirliğini iyileştirir ve okuma performansını da iyileştirebilir, çünkü okuma isteklerine genellikle çoğaltılan veri düğümlerinden herhangi birinden hizmet verilebilir. Birçok NOSQL uygulaması serileştirilebilir gerektirmez tutarlılık, dolayısıyla nihai tutarlılık olarak bilinen daha gevşek tutarlılık biçimleri kullanılır.

3. Replikasyon Modelleri: NOSQL sistemlerinde iki ana replikasyon modeli kullanılmaktadır:
- master-slave ve master-master replikasyonu.

- master/slave çoğaltma, bir kopyanın ana kopya olmasını gerektirir; tüm yazma işlemleri ana kopyaya uygulanmalı ve daha sonra genellikle nihai tutarlılık kullanılarak slave kopyalara yayılmalıdır (slave kopyalar sonunda ana kopya ile aynı olacaktır). Okumak için master-slave paradigması çeşitli şekillerde konfigüre edilebilir.

- Bir yapılandırma, tüm okumaların ana kopyada olmasını gerektirir, bu nedenle bu, benzer avantaj ve dezavantajlara sahip, birincil taraf veya dağıtılmış eşzamanlılık denetiminin birincil kopya yöntemlerine benzer olacaktır. - Başka bir konfigürasyon, slave kopyalarda okumalara izin verir, ancak slave düğümlere yazmalar, ana kopyaya uygulandıktan sonra yapılabileceğinden, değerlerin en son yazılanlar olduğunu garanti etmez. Master-master çoğaltma, çoğaltmaların herhangi birinde okuma ve yazma işlemlerine izin verir, ancak farklı kopyaları depolayan düğümlerdeki okumaların aynı değerleri göreceğini garanti etmeyebilir. Farklı kullanıcılar aynı veri öğesini sistemin farklı düğümlerine aynı anda yazabilir, bu nedenle öğenin değerleri geçici olarak tutarsız olacaktır. Farklı düğümlerde aynı veri öğesinin çakışan yazma işlemlerini çözmek için bir mutabakat yöntemi, - master-master çoğaltma şemasının bir parçası olarak uygulanmalıdır.

- master-master çoğaltma

kopyalardan herhangi birinde okuma ve yazma işlemlerine izin verir, ancak farklı kopyaları depolayan düğümlerdeki okumaların aynı değerleri göreceğini garanti edemez. Farklı kullanıcılar aynı veri öğesini sistemin farklı düğümlerine aynı anda yazabilir, bu nedenle öğenin değerleri geçici olarak tutarsız olacaktır. Farklı düğümlerde aynı veri öğesinin çakışan yazma işlemlerini çözmek için bir mutabakat yöntemi, ana-ana çoğaltma şemasının bir parçası olarak uygulanmalıdır.

4. Dosyaların Parçalanması: Birçok NOSQL uygulamasında, dosyalar (veya veri nesneleri koleksiyonları) milyonlarca kayda (veya belge veya nesne) sahip olabilir ve bu kayıtlara aynı anda binlerce kullanıcı tarafından erişilebilir. Bu nedenle, tüm dosyayı tek bir düğümde depolamak pratik değildir. Dosya kayıtlarının parçalanması (yatay bölümleme olarak da bilinir), NOSQL sistemlerinde sıklıkla kullanılır. Bu, dosya kayıtlarına erişim yükünü birden çok düğüme dağıtmaya yarar. Dosya kayıtlarını parçalama ve parçaları çoğaltma kombinasyonu, veri kullanılabilirliğinin yanı sıra yük dengelemeyi iyileştirmek için birlikte çalışır.

5. Yüksek Performanslı Veri Erişimi: Birçok NOSQL uygulamasında, bir dosyadaki milyonlarca veri kaydı veya nesne arasından tek tek kayıtları veya nesneleri (veri öğelerini) bulmak gerekir. Bunu başarmak için çoğu sistem iki teknikten birini kullanır: hashing veya nesne anahtarlarında aralık bölümleme . Bir nesneye erişimlerin çoğu, karmaşık sorgu koşulları kullanmak yerine anahtar değeri sağlayarak olacaktır. Nesne anahtarı, nesne kimliği kavramına benzer. Hash işleminde, K anahtarına bir hash işlevi h(K) uygulanır ve nesnenin K anahtarıyla konumu h(K) değeriyle belirlenir. Aralık bölümlemede konum, bir dizi anahtar değer aracılığıyla belirlenir; örneğin, i konumu, K anahtar değerleri Ki-min ≤ K ≤ Ki-max aralığında olan nesneleri tutar. Bir dizi anahtar değeri içindeki birden çok nesnenin alındığı aralık sorguları gerektiren uygulamalarda, aralık bölümlü tercih edilir. Diğer dizinler, K anahtarından farklı nitelik koşullarına dayalı olarak nesnelerin yerini belirlemek için de kullanılabilir.

8…….

Veri modelleri ve sorgu dilleriyle ilgili NOSQL özellikleri.

NOSQL sistemleri, modelleme gücü ve karmaşık sorgulama üzerinde performans ve esnekliği vurgular.

1. Şema Gerektirmeme: Şema gerektirmeme esnekliği, birçok NOSQL sisteminde yarı yapılandırılmış, kendi kendini tanımlayan verilere izin verilerek elde edilir. Kullanıcılar, depolama verimliliğini artırmak için bazı sistemlerde kısmi bir şema belirleyebilir, ancak çoğu NOSQL sisteminde şema olması gerekli değildir.

Kısıtlamaları belirtmek için bir şema olmayabileceğinden, veriler üzerindeki herhangi bir kısıtlamanın, veri öğelerine erişen uygulama programlarında programlanması gerekecektir. JSON (JavaScript Nesne Gösterimi) ve XML (Genişletilebilir İşaretleme Dili) gibi yarı yapılandırılmış verileri tanımlamak için çeşitli diller vardır. JSON, birkaç NOSQL sisteminde kullanılır, ancak yarı yapılandırılmış verileri tanımlamak için başka yöntemler de kullanılabilir.

2. Daha Az Güçlü Sorgu Dilleri: NOSQL sistemlerini kullanan birçok uygulama, SQL gibi güçlü bir sorgu dili gerektirmeyebilir, çünkü bu sistemlerdeki arama (okuma) sorguları genellikle tek nesneleri, nesne anahtarlarına dayalı olarak tek bir dosyada bulur. NOSQL sistemleri tipik olarak bir programlama API'si (uygulama programlama arabirimi) olarak bir dizi işlev ve işlem sağlar, bu nedenle veri nesnelerinin okunması ve yazılması, programcı tarafından uygun işlemler çağrılarak gerçekleştirilir. Bazı NOSQL sistemleri ayrıca üst düzey bir sorgu dili sağlar, ancak SQL'in tam gücüne sahip olmayabilir; SQL sorgulama yeteneklerinin yalnızca bir alt kümesi sağlanacaktır. Özellikle, birçok NOSQL sistemi sorgu dilinin bir parçası olarak birleştirme işlemleri sağlar; birleştirmelerin uygulama programlarında uygulanması gerekir.

3. Sürüm Oluşturma: Bazı NOSQL sistemleri, veri sürümünün oluşturulduğu zamanın zaman damgalarıyla birlikte veri öğelerinin birden çok sürümünün depolanmasını sağlar.

9……

24.1.3 NOSQL Sistemlerinin Kategorileri

NOSQL sistemleri, diğer sistem türlerini kapsayan bazı ek kategorilerle birlikte dört ana kategoride karakterize edilmiştir. En yaygın sınıflandırma, aşağıdaki dört ana kategoriyi listeler:

1. Belge tabanlı NOSQL sistemleri: Bu sistemler, verileri JSON (JavaScript Object Notation) gibi iyi bilinen biçimleri kullanan belgeler biçiminde depolar. Belgelere belge kimlikleri üzerinden erişilebilir, ancak aynı zamanda
diğer dizinler kullanılarak hızla erişilir.

2. NOSQL anahtar-değer depoları: Bu sistemler, anahtarla ilişkili değere anahtarın hızlı erişimini temel alan basit bir veri modeline sahiptir; değer bir kayıt, bir nesne veya bir belge olabilir veya daha karmaşık bir veri yapısına sahip olabilir.

3. Sütun tabanlı veya geniş sütunlu NOSQL sistemleri: Bu sistemler, bir tabloyu sütun ailelerine böler (bir tür dikey bölümleme, burada her sütun ailesi kendi dosyalarında saklanır. Ayrıca veri değerlerinin versiyonlanmasına da izin verirler.

4. Grafik tabanlı NOSQL sistemleri: Veriler grafikler olarak temsil edilir ve yol ifadeleri kullanılarak kenarlardan geçilerek ilgili düğümler bulunabilir.

Yukarıdaki dört kategoriye kolayca sınıflandırılamayan bazı sistemlerin yanı sıra NOSQL terimi yaygın olarak kullanılmaya başlamadan önce bile mevcut olan bazı diğer sistem türlerini içerecek şekilde ek kategoriler eklenebilir.

5. Hibrit NOSQL sistemleri: Bu sistemler, yukarıdaki dört kategoriden iki veya daha fazlasına ait özelliklere sahiptir.

6. Nesne veritabanları

7. XML veritabanları

10…….

Anahtar kelime tabanlı arama motorları bile büyük miktarda veriyi hızlı arama erişimiyle depolar, bu nedenle depolanan veriler büyük NOSQL büyük veri depoları olarak kabul edilebilir.

24.2 CAP Teoremi

Veri replikasyonu olan bir sistemde, her bir veri öğesinin birden çok kopyası olabileceğinden, eşzamanlılık kontrolü daha karmaşık hale gelir. Bu nedenle, bir öğenin bir kopyasına bir güncelleme uygulanırsa, diğer tüm kopyalara tutarlı bir şekilde uygulanmalıdır.

Bir X öğesinin bir kopyasının bir T1 işlemi tarafından güncellenmesi, diğer bir kopyanın ise bir T2 işlemi tarafından güncellenmesi olasılığı vardır, bu nedenle aynı öğenin iki tutarsız kopyası dağıtılmış sistemdeki iki farklı düğümde bulunur.

Diğer iki işlem T3 ve T4 X'i okumak isterse, her biri X öğesinin farklı bir kopyasını okuyabilir.

CAP'deki üç harf, çoğaltılmış verilere sahip dağıtılmış sistemlerin istenen üç özelliğine atıfta bulunur: tutarlılık (çoğaltılmış kopyalar arasında), kullanılabilirlik (okuma ve yazma işlemleri için sistemin) ve bölüm toleransı (bölümlenen sistemdeki düğümler karşısında bir ağ hatası nedeniyle).

CAP teoremi, veri replikasyonu ile dağıtılmış bir sistemde istenen özelliklerin üçünün de (tutarlılık, erişilebilirlik ve bölüm toleransı- consistency, availability, and partition tolerance) aynı anda garanti edilmesinin mümkün olmadığını belirtir. Bu durumda, dağıtılmış sistem tasarımcısının garanti etmek için üç özellikten ikisini seçmesi gerekir. Genellikle birçok geleneksel (SQL) uygulamasında, ACID özellikleri aracılığıyla tutarlılığın garanti edilmesinin önemli olduğu varsayılır. Öte yandan, bir NOSQL dağıtılmış veri deposunda, genellikle daha zayıf bir tutarlılık düzeyi kabul edilebilir ve diğer iki özelliğin (kullanılabilirlik, bölüm toleransı) garanti edilmesi önemlidir. Bu nedenle, NOSQL sisteminde serileştirilebilirliği garanti etmek yerine genellikle daha zayıf tutarlılık seviyeleri kullanılır. Özellikle, nihai tutarlılık olarak bilinen bir tutarlılık biçimi, NOSQL sistemlerinde sıklıkla benimsenir.