YBS Ansiklopedi

www.YBSAnsiklopedi.com

Cilt 4, Sayı 4, Aralık 2017

Büyük Veri, TUB Teoremi, ACID ve BASE Yaklaşımları

(Big Data, CAP Theorem, ACID and BASE Approaches)

Şadi Evren ŞEKER

1. *İstanbul Şehir Üniversitesi, Yönetim Bilişim Sistemleri Bölümü*

Özet

Bu makalenin amacı, büyük veri çalışmalarında önemli bir rol oynayan ve büyük veri saklama sistemlerinin tasarım temellerini oluşturan tutarlılık, ulaşılabilirlik ve bölünmüş ağ dayanıklılığı kavramlarının baş harflerinden oluşan TUB teoremini açıklamaktır. Teoremin tanımı yapıldıktan sonra basitçe günümüz veri tabanı teknolojileri için önemini, günümüzde değişen veri saklama beklentileri ve pratiklerini açıklanacaktır. Ayrıca veri tabanı teorisinde bulunan ve Türkçe'ye asit ve baz olarak kelime tercümesi yapılabilecek olan ACID ve BASE kavramlarını açıklayarak büyük veri dünyasında değişen yaklaşımlar açıklanacaktır.

Anahtar Kelimeler:*ayrıştırma,dönüştürme,yükleme ,veri tabanı , veri madenciliği, veri ambarı*

Abstract

The aim of this study is explaining CAP theorem, which is short cut of consistency, availability and partition tolerance and has a major role for understanding the contemporary database technologies. After the definition of theorem, the importance of theorem for current database technologies, changing expectations and practices in data storage is discussed. Also the ACID and BASE concepts in big data world is explained in this paper.

Keywords:*extract,transform,load,database,data mining, data warehouse*

1. Giriş

Büyük veri kavramı, tarihsel süreçte farklı ihtiyaçların motor güç oynaması ile şekillenmiştir. Örneğin arama motoru teknolojisindeki gelişme ve ihtiyaçlara paralel olarak doğan veri saklama pratikleri ve paralel işleme teknolojisindeki gelişmeler, büyük veri dünyasının şekillenmesinde önemli rol oynamıştır. Paralel işleme için geliştirilen mesaj geçiş ara yüzü (message passing interface, MPI) teknolojisi daha sonraları büyük veri için kritik önem taşıyan haritalama ve indirgeme (map-reduce, MR) teknolojisi için öncü rol oynamış, yine büyük veri teknolojisi için çıkış dönemlerinde öncü rol oynayan Apache Hadoop teknolojisi için belirleyici olmuştur [1]. İşlem katmanında paralel işlemeden edinilen tecrübeler kullanılırken verinin saklanması için yine bu alanda öncü işletmeler ve teknolojiler Google arama motoru ve amazon'un veri saklama problemleri [2] belirleyici rol

oynamıştır. Örneğin Google tarafından verinin hızlı erişilmesi ve işlenmesine olanak sağlayan hafıza içi (in-memory) teknolojisi yine amazonun veri tutmada kullandığı dynamo-db gibi gelişmeler sektörü belirlemiştir.

Büyük veri dünyasındaki en önemli gelişmelerden birisi, verinin tek bir bilgisayarda veya veri merkezinde (data center) tutulup işlenemeyecek kadar büyük ve hızlı değişen özelliklerde olmasıdır. Bu sebeple verinin dağıtılması ve yerelleştirilmesi gündeme gelmiştir. Örneğin coğrafi olarak Çin'e yakın bir veri merkezindeki bilgisayarlar, Çin'den gelen taleplere cevap verirken, Türkiye'ye yakın bir veri merkezi Türkiye'ye hizmet verebilmektedir. Bu sayede ağ gecikmeleri veya ağda kaynaklanan problemlerin önüne geçmek mümkün olmaktadır [3].

Yine büyük veri teknolojilerindeki önemli bir motivasyon da, hizmetlerin sürekli erişilebilir olmasıdır. Yani facebook, twitter, linked-in gibi sosyal ağların amacı, sürekli hizmet veren ve erişime açık hizmetler sunmaktır [4]. Bu hizmetler aynı zamanda %100 içerik ihtiyacı olmayan hizmetlerdir. Mesela, facebook'a bağlanan birisinin, kendi hesabı ile ilgili bütün detayları görmesi gerekmekle birlikte, facebook'taki bütün kullanıcıların bütün detaylarını görmesi gerekmez. Dolayısıyla sosyal ağlar gibi büyük yapıların kısmi bölünebilirlik özelliği bulunur. Bu özellik, verilerin de tek merkezde toplanması ihtiyacını ortadan kaldırmaktadır. Hem ihtiyaç açısından hem teknoloji açısından yaşanan bu gelişmeler büyük veri teknolojilerinin daha hızlı yayılmasını sağlamıştır.

2. TUB Teoremi (CAP Theorem)

TUB teoremi, tutarlılık (consistency), ulaşılabilirlik (availability) ve bölünmüş ağ dayanıklılığı (partition tolerance) kavramlarının kısaltmasıdır ve literatüre ilk kez kazandıran Eric Brewer'a atfen, Brewer's teorem veya İngilizce baş harflerini birleştirerek oluşturulan CAP teorem olarak da geçmektedir. Teorem basitçe bu üç kavramın en fazla iki tanesinin aynı anda sağlanabileceğini ve üçünün birden sağlanmasının imkansız olduğunu söyler [5].

Buna göre bir veri tabanı teknolojisindeki üç özellik aşağıdaki şekilde sıralanabilir:

Tutarlılık (Consistency): Veri tabanının birden fazla kopyası olması veya birden fazla veri merkezine dağıtılmış olması durumunda, bu kopyalar arasındaki tutarlılığı ifade eder. Örneğin Çin'de ve Türkiye'de iki ayrı veri merkezi bulunan bir projede, veri merkezleri birbirini sürekli güncellemekte veya kendi üzerindeki verilerle ilgili haberdar etmektedir. Hatta verinin nasıl dağıtılacağı ile ilgili belirlenen politikaları iki tarafında biliyor olması beklenir. Birden fazla kopyanın tutulduğu bu gibi projelerde, iki sunucu üzerindeki verilerin tutarlı olması, hangi sunucudan veri sorgulanırsa sorgulansın aynı sonucun (en son güncellenen sonucun) alınması anlamına gelir.

Ulaşılabilirlik (Availability): Veri merkezlerinin sürekli ulaşılabilir olması anlamına gelir. Yani sunucuların çeşitli sebeplerle erişilemez olması durumunda sunucu kendisine erişmek isteyenlerle bağlantıyı kesebilir ve ulaşılamaz olabilir. Veya, üzerindeki mevcut imkanlarla hizmet sunmaya devam eder.

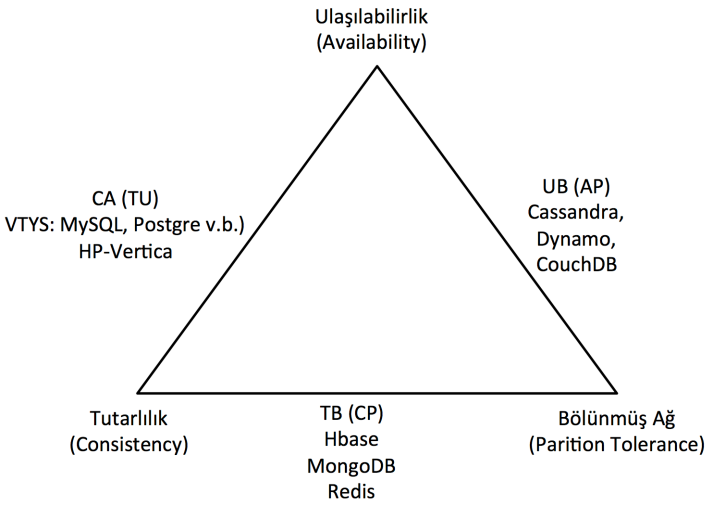
Bölünmüş ağ dayanıklılığı (partitioned network tolerance): Çeşitli sebeplerle ağda yaşanan kopmalar ve dolayısıyla veri merkezleri arasındaki iletişim problemlerinin teknoloji tarafından nasıl karşılanacağını anlatır. Yani ağda bir kopma olması ve iletişimin mümkün olmaması durumunu ifade eder.

TUB Teoremi, yukarıda geçen 3 durum için teknolojinin nasıl davranacağını açıklar. Teknolojiler farklı önceliklere göre tasarlanır. Mesela Apache Cassandra, tasarımı itibariyle bölünmüş ağa dayanıklıdır ve ulaşılabilirliği öneler, bununla birlikte tutarlılığı feda eder. Yani ağda bir kopma olması ve veri merkezlerinin birbiri ile senkronizasyon yapamaması durumunda, iki veri merkezi de hizmet vermeye devam eder (ulaşılabilirdir) ancak iki veri merkezindeki veriler de farklı şekilde güncellendiği için tutarlı değildir. Mesela facebook gibi bir sosyal ağı düşünecek olursak, bir kullanıcının verilerinin birden fazla kopyasının ağda birden fazla veri merkezinde durduğunu düşünelim. Bu verilerden birisi güncellendiğinde, diğer sunuculara bu bilgi gitmeyebilir (ağda bölünme yaşandığını

kabul edersek) ve dolayısıyla, aynı kullanıcı için iki ayrı sunucuda iki ayrı bilgi bulunacak bu da tutarlılığı feda edecektir.

Diğer bir örnek olarak klasik, işlem temelli (OLTP) veri tabanı yönetim sistemleri (vtys, relational database management systems, RDBMS) olabilir. Klasik veri tabanı teknolojileri tutarlılığı önceler ve ağda herhangi bir problem olması durumunda diğer kopyalarla erişim probleminden dolayı hizmet vermeyi durdurur. Örneğin klasik bir yazma işleminde şayet veri kopyaları (replication) bulunuyorsa, bu kopyaların birbirini senkronize ederek hep birlikte yazması gerekir (3 aşamalı yazma- 3 Phase commit işleminde olduğu gibi). Bu sırada yaşanan bağlantı problemleri, hizmetin durdurulması ve netice olarak tutarlılık sağlamak için bölünmüş ağ toleransının feda edilmesi anlamına gelir. Bu sırada verilere erişilebilir ve dolayısıyla veriler için ulaşılabilir diyebiliriz. Yani klasik bir veri tabanı için tutarlı ve ulaşılabilir ama bölünmüş ağ toleransı yoktur demek mümkündür.

Diğer bir örnek ise MongoDB teknolojisi olabilir. MongoDB, yapısı itibariyle, tutarlı ve ağ kopmalarına dayanıklıdır. Ancak bu sefer de ulaşılabilirliği feda etmiştir. MongoDB, çok farklı şekillerde ayarlanabilir ancak varsayılan ayarlarında, yönetici / takipçi (master/slave) yapısı bulunmaktadır ve bütün okuma işleri bir yöneticiden geçer. Bu yönetici okuma işlerini sunuculara dağıtır. Bu ayarlarla MongoDB'nin tutarlı ve ağ bölünmelerine dayanıklı olduğu söylenebilir. Çünkü ağdaki bir bağlantı probleminde, okuma işlemi yöneticiye gidecek, yönetici ise veriyi okuma işlemini ilgili sunucuya yönlendirecektir. Şayet ilgili sunucuya erişilebiliyorsa okuma işlemi gerçekleşecektir (ulaşılabilir), şayet erişilemiyorsa o zaman ulaşılabilirlik feda edilmiş olacak ve okuma işlemi gerçekleşemeyecektir. Ayarları ile oynayarak MongoDB bütün sunucuların okuma işlemini yöneteceği şekilde ayarlanabilir. Bu ayar, MongoDB yapısını tutarlı olmaktan çıkaracak ve bölünmüş ağa duyarlı ve ulaşılabilir yapacaktır (Bu açıdan, Apache Cassandra gibi davranacaktır).



Şekil 1: TUB Teoremi ve Teknolojilerin Konumları

Şekil 1'de görselleştirilen TUB teoremine göre, farklı teknolojilerin teoremdeki konumları bulunmaktadır. Örneğin MySQL, Postgres gibi klasik veri tabanları, SAP HANA veya HP-Vertica gibi teknolojiler, ulaşılabilirlik ve tutarlılığı öncelerken, Cassandra, DynamoDB veya CounchDB gibi teknolojiler, ulaşılabilirlik ve bölünüş ağ toleransını öncelemekte, HBase, MongoDB veya Redis gibi teknolojiler ise tutarlılık ve bölünmüş ağ toleransını öncelemektedir [6].

Şekildeki teknolojiler, varsayılan ayarları ile kabul edilmiştir. Çoğu teknoloji, farklı ayarlamalarla TUB üçgeninde farklı noktalara kayabilmektedir. Ayrıca şekil, teknolojilerin öncelikleri üzerine kuruludur, yani teknolojiler aslında TUB teoreminde bulunan 3 özelliği de destekleyebilir. Nitekim literatürde, örneğin Apache Cassandra gibi teknolojiler için sonunda tutarlı (eventually consistent) gibi ifadelere rastlanabilir, ki bu ifade üçüncü bölümde baz kavramının bir parçası olarak da anlatılacaktır. Bu ifadelerin amacı sisteme vakit verilmesi ve her şeyin yolunda gitmesi durumunda, sistem kopya veriler (replica) üzerinde kendisini tutarlı hale getirebilmektedir. Ancak, örneğin ağ bölünmesi gibi bir durumda, tutarlılığı ikinci plana itmektedir. Tam bu noktada, TUB teoreminin en çok yanlış anlaşılan yönünden bahsetmekte fayda var. TUB teoremi, veri tabanı teknolojilerinin üç özellikten hangi iki tanesine öncelik verdiğini belirtir. Yani aslında bütün veri tabanları normal şartlar altında üç özelliği de destekleyebilir ancak TUB teoremi, örneğin bir ağ bölünmesi olduğunda hangi özelliğin daha öncelikli olduğunu belirlemek için kullanışlıdır.

TUB teoremi, veri tabanı teknolojilerini, özellikle de büyük veri perspektifinden kategorize etmek için oldukça kullanışlı bir sınıflandırma sistemi kurmasına karşılık, bu sistemin yetersiz kaldığı ile ilgili de eleştiriler vardır hatta Eric Brewer 12 yıl sonra Türkçe'ye "kurallar nasıl değişti" şeklinde başlığı çevrilebilecek makalesinde bu eleştirilerden bazılarına cevap vermiştir [7].

3. Asit veya Baz

Veri tabanı teknolojileri asidik olmaları veya bazik olmalarına göre de ikiye ayrılabilir. Buradaki asitlerin ve bazların kimyadaki asit ve bazlarla ilgili olmadığını söyleyerek başlayalım. Bu kavramlar İngilizcedeki ACID ve BASE kelimelerinin baş harflerini oluşturan kavramlardan gelmektedir. Buna göre ACID aşağıdaki kavramlardan oluşur:

Atomluk (Atomicity): bir işlemin alt işlemlerinin tamamının birden yapılması veya hiç yapılmaması. Örneğin bir banka havalesi sırasında, gönderen hesaptaki para azalmakta ve alıcı hesaptaki para artmaktadır. Dolayısıyla banka havalesi alt bazı adımlardan oluşur. Bir banka havalesinin atomik olması demek, bu alt adımlara bölünememesi yani bir işlem ya tamamen gerçekleşecek ya da hiç gerçekleşmeyecek demektir. Yani bir hesaptan paranın azalıp diğer hesapta para artmadan bu işlem kesilemez veya benzer şekilde bir hesapta para artıp diğer hesaptan para azalmadan da bu işlem kesilemez demektir. Atom kelimesinin antik yunanda bölünemeyen anlamına geldiğini hatırlayınız.

Tutarlılık (Consistency): Veri tabanları üzerinde kural ve protokol tanımları yapılabilir. Örneğin veri tabanındaki bir kural gereği, silme veya güncelleme işlemi, veri tabanındaki başka bir yerler tutarlılık kontrolü gerektiriyorsa, ACID özelliğinin bir parçası olarak tutarlılık sağlanmalıdır. Güncel bir örnek olarak çalışanların bağlı oldukları departmanların farklı bir tabloda durduğunu düşünelim. Bu durumda her çalışanın bağlı olduğu departmanı, çalışan tablosunda bir yabancı anahtar (foreign key) ile tutması beklenir. Şayet bir departman silinirse, bu departmanda çalışanların yabancı anahtarlarına karşılık gelen bir departman bilgisi bulunmayacak ve bu durumda bir tutarsızlık (inconsistency) oluşacaktır. Bu durumda iki tablo arasında silme tutarlılığı (delete consistency) kuralı tanımlanabilir ve departman tablosundaki birincil anahtarın silinmesi, diğer bağlı olduğu (yabancı anahtar olarak tutulduğu) tablolarda bir dizi kontroller gerektirebilir. İşte bu tutarlılık durumu tamamen sağlanmadan işlem gerçekleştirilemez ve veri tabanı üzerinde yarım-tutarlı, veya yarım-tamamlanmış, veya kısmen sağlanan kurallar olamaz.

İzolasyon (Isolation): Veri tabanı üzerindeki hiçbir işlem, diğer işlemlere erişemez (birbirlerinden izole çalışırlar). Bir diğer işlemin çalışması için o anda çalışan bütün işlemler, farklı kaynaklara erişmelidir veya aynı kaynakları kullanıyorlarsa da işlem bekletilerek kaynaklar serbest bırakıldıktan sonra çalışmalıdır. Çoğu veri tabanı

bu özelliği sağlamak için veri tabanı kaynakları üzerine kilitler (lock) koymakta ve işlem bitene kadar diğer işlemlerin erişimini engellemektedir.

Sağlamlık (Durability): Bir veri tabanının üzerinde çalıştırmakta olduğu işlemleri, sistemsel problemler kadar donanımsal problemlere karşı da koruyor olmasıdır. Yani sistem çökmeleri, ani kapanmalar, ağ kopmaları gibi çok sayıda donanımsal veya veri tabanı sisteminin dışında cereyan eden problemlerde, veri tabanı sisteminin verilerin ve işlemlerin (transactions) sağlığını garanti ediyor olması gerekir.

Benzer şekilde baz (BASE) kavramı ise aşağıdaki kavramların baş harflerinden oluşturulmuştur [8]:

Temel olarak Ulaşılabilir (Basically Available): TUB teoreminde göre, sistemin ulaşılabilir (available) olmasını garanti etme durumudur. Buradaki ulaşılabilme durumu, sunucunun veya sistemin cevap vermesini kapsar ama bu cevap verme bir hata mesajı veya sistemin şu anda istenen veriyi döndürememesi şeklinde de olabilir. Örneğin banka sisteminde, şu anda bakım yapılıyor olması yüzünden hesaplara ulaşılamaması, ve sistemin bu şekilde bir hata vermesi sistemin ulaşılabilir olduğu anlamına gelir.

Yumuşak Durum (Soft State): Sistemin veri güncelleme veya geçiş durumlarında takındığı haldir. Buna göre sistemde bulunan verinin birden fazla kopyası arasında farklılıklar olabilir ve sistem nihai olarak tutarlılık garanti edebilir. Yani anlık farklılıklardan doğan ve sistemin anlık resmi çekilse farklı olan veriler, sistemin yumuşak durumunu oluşturur. Bu verilerin sonunda eşlenip senkronize edilmesi durumu, yumuşak durum oluşumunu engellemez.

Nihai Tutarlı (Eventual Consistency): Sistem daha fazla değişiklik ve yeni veri almadığı durumlarda, yeterli zaman tanınırsa, verinin farklı kopyalarını eşleştirerek bütün sunucularda tutarlı bir duruma erişebilir ve yumuşak durumdan çıkmış olur. Bu tutarlılık durumuna ulaşılabilecek olması, nihai tutarlılık özelliğidir. Ancak, çoğu sistem çalışma sürecinde böyle bir tutarlılığı aramaz ve sürekli akan işlemlere cevap vermeyi önceler, dolayısıyla tutarlılık ikinci planda kalabilir.

Kaynakça

[1] Katal, Avita, Mohammad Wazid, and R. H. Goudar. "Big data: issues, challenges, tools and good practices." Contemporary Computing (IC3), 2013 Sixth International Conference on. IEEE, 2013.

[2] DeCandia, Giuseppe, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. "Dynamo: amazon's highly available key-value store." ACM SIGOPS operating systems review 41, no. 6 (2007): 205-220.

[3] Lin, Jimmy, and Dmitriy Ryaboy. "Scaling big data mining infrastructure: the twitter experience." ACM SIGKDD Explorations Newsletter 14, no. 2 (2013): 6-19.

[4] John Walker, S., 2014. Big data: A revolution that will transform how we live, work, and think.

[5] Brewer, Eric A. "Towards robust distributed systems." PODC. Vol. 7. 2000.

[6] Han, Jing, E. Haihong, Guan Le, and Jian Du. "Survey on NoSQL database." In Pervasive computing and applications (ICPCA), 2011 6th international conference on, pp. 363-366. IEEE, 2011.

[7] Brewer, Eric. "CAP twelve years later: How the" rules" have changed." Computer 45.2 (2012): 23-29.

[8] Pritchett, Dan. "Base: An acid alternative." Queue 6, no. 3 (2008): 48-55.