T.C.

MALTEPE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ÇOĞALTILMIŞ BİRİNCİL VERİ TABANI KULLANAN SİSTEMLERDE KİLİTLENMELERİ AZALTMA

Şemseddin AKSOY

Yüksek Lisans Tezi

Tez Danışmanı

Prof. Dr. Murat TAYLI

İSTANBUL – 2014

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ÇOĞALTILMIŞ BİRİNCİL VERİ TABANI KULLANAN SİSTEMLERDE KİLİTLENMELERİ AZALTMA

Şemseddin AKSOY

Yüksek Lisans Tezi

Tez Danışmanı

Prof. Dr. Murat TAYLI

İSTANBUL-2014

Bu tez çalışması, Maltepe Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... / ..... / ..... tarih ve ..... / ..... sayılı kararıyla oluşturulan jüri tarafından ……. ***……. Yüksek Lisansı Tezi*** olarak kabul edilmiştir.

JÜRİ

Prof. Dr. Murat TAYLI

(Danışman)

Prof. Dr. Müslim BOZYİĞİT

Üye

Yrd. Doç. Dr. Fatih YÜCALAR

Üye

ÖZET

Yüksek Lisans Tezi, Çoğaltılmış birincil veri tabanı kullanan sistemlerde kilitlenmeleri azaltma, T.C. Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı.

Tezde veri tabanı sistemlerinin ölçeklenmesi ve çoğaltılmış veri tabanı sistemlerindeki kilitlenme sorunu araştırılmıştır. Çoğaltılmış birincil veri tabanı sistemi kullanan, ölçeklenir bir mimaride sınamalar yapılmıştır. Farklı yük ve veri tabanı sunucu sayılarıyla, başarım ve kilitlenme oranları tespit edilmiştir. Sistemin beklendiği gibi ölçeklendiği, okuma-yazma işlemleri kurallara bağlı olarak dağıtıldığında veri tabanındaki kilitlenmelerin azaldığı saptanmıştır.

Bu tez 2014 yılında tamamlanıp ve 44 sayfadan oluşmaktadır.

Anahtar kelimeler: Ölçekleme, Çoğaltılmış Birincil Veri Tabanı, Kilitlenme, Çakışma

ABSTRACT

Master Thesis, Minimizing Deadlocks in Multi-master Database Systems. Maltepe University, Institute of Natural Sciences, Department of Computer Engineering

In this thesis scalability of database systems and deadlock problems of replicated database systems are investigated. Tests have been done in a scalable architecture that uses the primary replicated database system. The rates of performance and deadlocks are determined by using different loads and database servers. It is observed that deadlocks in database system is decreased when the system is scaled as expected and read-write transactions are distributed properly.

This thesis has been completed in 2014 and consists of 44 pages.

Keywords: Scaling, Multi-master replicated database, Deadlock, Conflict

TEŞEKKÜR

Başta değerli hocam Prof. Dr. Murat TAYLI olmak üzere, meslektaşlarım Selim BAYRAKLI, Murat CANBAZ, Şükrü GÜLLÜOĞLU, Nurettin Onur TUĞCU' ya, desteklerini esirgemeyen ailem ve Maltepe Üniversitesi Bilgi İşlem Daire Başkanlığı çalışanlarına teşekkürlerimi sunarım.

İÇİNDEKİLER

KISALTMALAR

Kısaltma	İngilizcesi	Türkçesi
JVM	Java Virtual Machine	Java Sanal Makinesi
JDBC	Java Database Connectivity	Java Veri Tabanı Bağlantısı
GPL	General Public License	Genel Kamu Lisansı
DB	Database	Veri Tabanı
VTYS	Database Management System	Veri Tabanı Yönetim Sistemi

ÇİZİM LİSTESİ

1. GİRİŞ

Teknolojideki ilerlemelerin sonucunda bireylerin hayatı, kişisel iş istasyonları, dizüstü bilgisayarlar, tablet ve akıllı telefonlardan oluşan geniş bir bilgisayar yelpazesi sayesinde, bilgi ve hizmetlere kolayca erişim olanaklarıyla zenginleşmiştir. Bu teknolojilerin gelişmesinin temelinde bilgisayar ağlarındaki ve ağ tabanlı hizmet alanındaki ilerlemeler yatmaktadır. Bu sayede kullanıcılar çeşitli ve zengin veri hizmetlerine zaman ve coğrafya sınırlamaları olmadan erişebilmektedir.

Başlangıcından bu yana Internet etkileşimli düz yardımlı metinden (hypertext), dinamik çoklu ortamlı bilgi sistemlerine doğru her gün artan kullanıcı sayısı ile gelişmektedir. Geliştirilen çeşitli platform, programlama dili ve araçları günümüzde yüksek sığalı uygulamaların büyük ölçekli işlerin ihtiyaçlarını karşılamakta ve genişleyen kullanıcı kitlelerine hizmet vermektedir.

Uygulamaların kullanımının yaygınlaşması ölçeklenme sorununu da birlikte getirmektedir, çünkü uygulamanın ölçeklenmesi, kullanıcı sayısının artış ivmesini genellikle karşılamamaktadır. Sorun sadece artan kullanıcı sayısından değil, bu kullanıcıların uygulamayı daha ayrıntılı ve etkin kullanmasından da kaynaklanır. Örnek olarak kullanıcılarının sadece resim paylaşabildiği bir milyon aktif kullanıcısı olan bir sitenin, hizmetlerine video paylaşımı eklemesi, sitenin isteklere geç cevap vermesine ya da çökmesine yol açması beklenir. Bu oluşumun sebebi, video paylaşım özelliğinin sadece bir kaç kullanıcı ile tasarlanıp test edilmesi ve çok sayıda kullanıcıya sunulduğunda sistemin ağır yük altında istenilen başarımda çalışmaması olarak açıklanabilir. Küçük ölçekte çalışabilen teknolojiler, akış ve veri hacmi olarak büyük ölçeğe geçiş yapıldığında istenilen başarım ile çalışmaz hale gelebilir.

Benzer bir başarım sorunuyla karşılaşıldığında yapılaması gereken ilk şey, darboğazın ayrıntılarıyla araştırılarak, doğru gerekçelerinin bulunmasıdır. Bu iş

kolay gözükse de karmaşık sistemlerde zaman, iş gücü ve uzmanlık gerektirdiği açıktır. Zamanla kapsamı genişleyen yazılımlar farklı diller kullanan yazılımcılardan katkılar içerebilir. Bunun yanı sıra sorunun kaynağı sistemin veri tabanı veya uygulama sunucusu gibi farklı bileşenlerinde ortaya çıkıyor olabilir. Darboğaz bulunsa bile çözüm, uygulamayı olması gereken hale getirmek için aksayan kod kümesinin düzeltilmesinden, belirli bir parçanın tekrardan yazılmasına kadar uzanabilir. Uzun vadeli çözüm, uygulamanın doğru şekilde tasarlandığı ve kodlandığı varsayıldığında, zaman kaybını ve sorunları önlemek için, başlangıçta küçük ölçekte çalışacak uygulamanın artan yüke göre ileride büyük değişimlere ihtiyaç duymadan kolayca ölçeklenmesidir.

Bir sistem doğru şekilde çalışıyor mu sorusu ancak iki ölçüt ile değerlendirilerek cevaplanabilir. Birinci ölçüt; sistem kullanıcıları için doğru sonuç üretiyor mu? İkincisi de bu sonucu kabul edilebilir bir sürede mi üretebiliyor? Artan kullanıcı veya veri yükünde istenilen başarımda sonuç üretmek için ölçeklenir sistemlerin tasarlanıp, uygulanması bir zorunluluktur. Dikey ölçeklemenin sınırlarına ulaşıldığında ise yatay ölçekleme kaçınılmaz olmaktadır. Veri tabanı sunucularında, yatay ölçekleme çıkabilecek veri tutarsızlığı sorunları yüzünden, uygulama sunucularından daha karmaşıktır.

Tezde veri tabanı yatay ölçeklemesi için çoğaltma yöntemi kullanılmıştır. Kilitlenmeleri incelemek için uygulama sunucusu üzerinde sorguları, okuma ve yazma olarak türlerine göre farklı sunuculara bölüştüren bir sınama sistemi kurulmuştur. Daha sonra okuma-yazma sorgularını karışık işleyen bir mimaride sınanmıştır. Bu çoğaltma yönteminde oluşan kilitlenmeler ve başarım incelenmiştir.

Yapılan sınamalarda elde edilen bulgular incelendiğinde, veri tabanı kopyaları arttıkça, yatay ölçekleme kuramına göre beklendiği gibi, sistemin hizmet sığasının arttırdığı görülmektedir. Bunun yanı sıra sunucu sayısının artması, yazma işlemleri sırasında, veri tabanlarında kilitlenmeleri de arttırmaktadır.

1.1. Sistem Ölçekleme

Bilgisayar sistemlerinde ölçekleme**,** donanımın veya yazılımın ilerideki gereksinimlerini karşılamak üzere büyüyebilme yeteneği olarak tanımlanmaktadır. Kullanıcı sayısı ve nitelikleri büyüyebilecek sistemlerde olması gereken bir özelliktir. Ölçeklenirlik, sistemlerde büyümenin kontrol altında yapılabilme becerisi veya yeni eklenen kaynaklarla (donanımlarla) işlem hacminin arttırabilmesi olarak da tanımlanabilir. Sığası arttırılarak başarımı orantılı olarak geliştirilen sistemler ölçeklenir sistem olarak adlandırılır. Aynı şekilde verimli ve pratik bir şekilde büyük sistemlere uygulanabilir algoritmalara da ölçeklenir denilebilir. Eğer kaynaklar geliştirildiğinde, büyütüldüğünde (yeni donanım eklendiğinde) kullanılan algoritma çöküyor veya başarımı artmıyorsa bu algoritma ölçeklenir değildir [1] [2].

Başarım sınırları belirlenmiş bir sistem için ölçeklemenin önemi, başarıma olan katkısı ile doğru orantılıdır. Ölçeklemenin olası kullanıcıları kuramsal araştırmacılar, ar-ge çalışanları, endüstriyel sistem tasarımcıları ve bilgisayar sistem uzmanları gibi pek çok değişik iş alanından olabilir [3] .

Algoritmaların ölçeklemeye uygun şekilde geliştirilmesi, çok izlekli (multi-thread), koşut programlama temelinde veya kuyruklama alt yapısını kullanılması gerekmektedir.

Ölçekleme ayrıca, bir sistemin ya da parçasının öngörülmemiş değişimlere uyması için düzenlenebilir olma kolaylığı olarak da tanımlanabilir. Ölçeklenir bir sistemin üç temel özelliği vardır [4].

- Sistem artan kullanıcı yüküne hizmet düzeyini düşürmeden cevap verebilmelidir.
- Sistem büyüyen veri kümesine hizmet düzeyini düşürmeden cevap verebilmelidir.

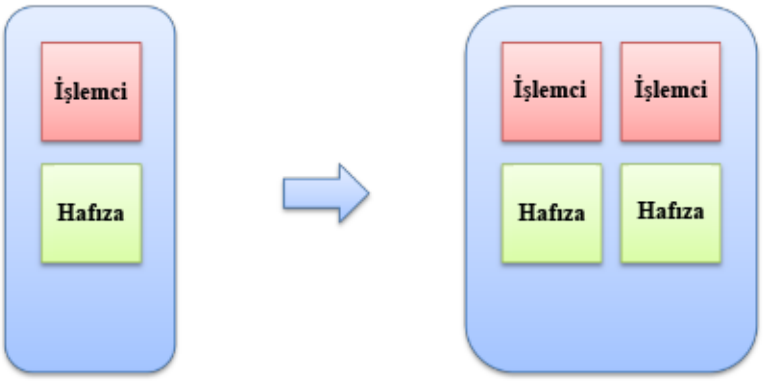
- Sistem ölçeği büyüse bile sürdürülebilir ve tanımlanmış başarımda hizmet verebilmelidir.

Ölçekleme sadece işleme hızı ve veri depolama sığası ile ilgili değildir, başarım ve ölçekleme birbirinden farklı kavramlar olsalar da aralarında bağımlı bir ilişki mevcuttur. Başarım, sistemin belirli bir işi ne kadar hızlı ve verimli bir şekilde yaptığını ölçer iken, ölçekleme artan iş yükü ile birlikte başarımda olagelen değişiklikleri ölçer. Bir yazılım sisteminin başarımı istenilen iş yükü düzeyine ulaşılmadan önce, artan yükle birlikte hızla düşüyorsa bu sistemin ölçeklenir olmadığı söylenebilir [4].

Ölçekleme temel olarak, istenilen hizmeti saptanmış başarım sınırlarında kalacak şekilde daha büyük veya küçük boyutta yapma anlamına gelir. Bilgisayar sistemlerinde ölçeklenir mimari yatay ve dikey olmak üzere iki sınıfa ayrılır.

1.1.1. Dikey ölçekleme

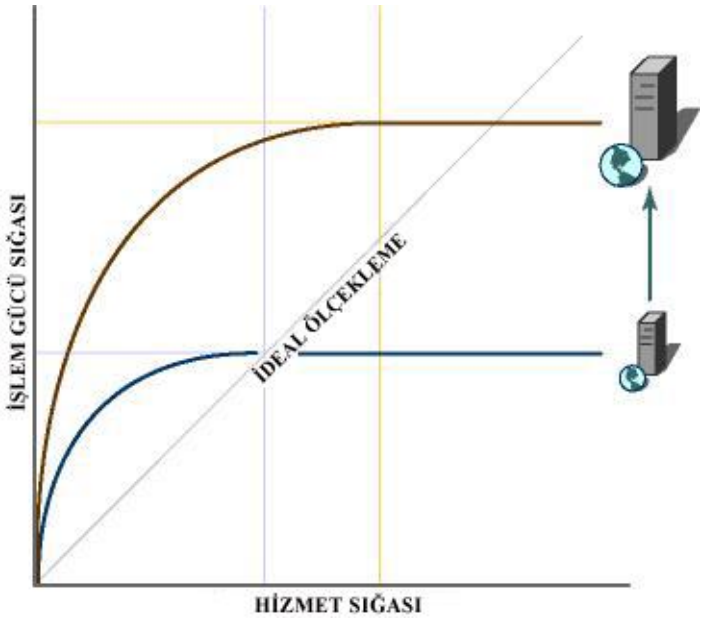
Dikey ölçekleme bir donanım biriminin kaynaklarının sayılarının artırımı veya daha yüksek sığalı donanımlarla değiştirilmesi yoluyla artan iş yükünü kaldırabilecek duruma getirilmesidir. Çizim 1.1'de gösterildiği gibi bir sunucu, işlemci ve hafıza artırımı ile desteklenebilir. Yazılım için de algoritmaları ve uygulamayı iyileştirmek gerekebilir. Donanım kaynaklarının koşut olarak çalıştırılmaları ya da çalışmayan işlemlerin sayısının işlemciye göre düzenlenmesi de bir dikey ölçekleme yöntemi olarak kullanılabilir.



Çizim 1.1 Dikey Ölçekleme Mimarisi [4]

Dikey ölçeklemede sisteme kaynak ekledikçe, eklenen her birime karşılık başarım getirisi düşer ve doğrusal bir artış göstermez. Donanımda büyüme maliyete üstel artış getirir.

Çizim 1.2'de gösterildiği gibi işlem sığası ile hizmet sığası, ideal ölçekleme çizgisinden uzaklaşma eğilimindedir.

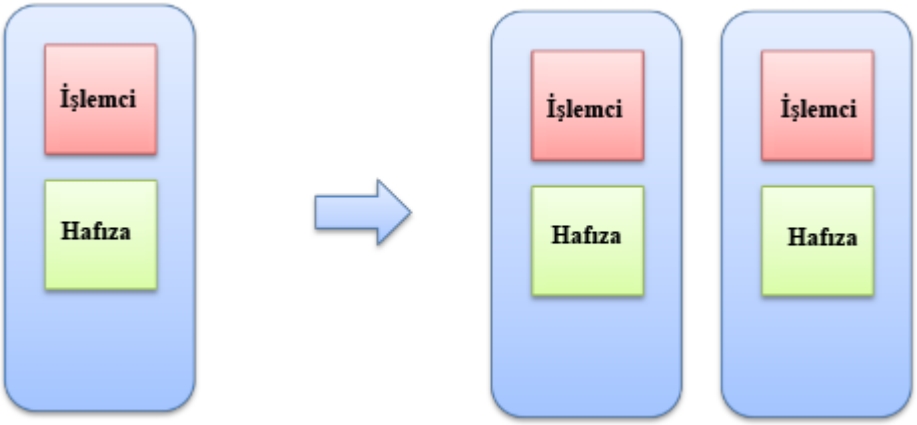


Çizim 1.2 Dikey Ölçeklemede İşleme Sığası-Hizmet Sığası Çizimi [5]

Amaçlanan işlem sığasına ulaşabilmek için gereken donanım yatırımı başarım ile doğrusal ilişkili değildir. Yatırım arttıkça elde edilen başarım azalacaktır. Bu şekilde yapılan yatırımın karşılığı tam olarak alınamadığından, dikey ölçeklemenin verimli bir yaklaşım olmadığı söylenebilir.

1.1.2. Yatay ölçekleme

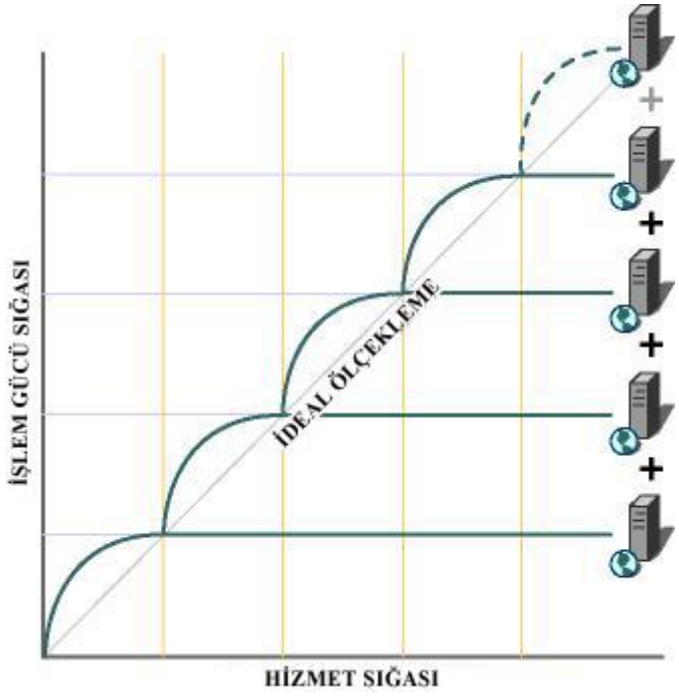
Kaynakların bağımsız yeni birimler ekleyerek kademeli olarak arttırılmasına yatay ölçekleme adı verilir. Bu işlem Çizim 1.3'de gösterildiği gibi benzer sığalı bir veya daha fazla sistemin eklenmesi ile oluşturulur. Bu yapıda kaynak talepleri birçok birim arasında dağıtılır ve tek bir sisteme aşırı yük binmesi engellenir.



Çizim 1.3 Yatay Ölçekleme Mimarisi [4]

Sistemdeki bileşenleri çoğaltmak, kimi bileşenler çalışmasa bile, tüm sistemin işler halde kalmasını sağlar ve sistemin kesintisiz hizmet verme süresini uzatır. Genellikte daha küçük birkaç sunucunun oluşturduğu maliyet yükü, eşdeğer başarımdaki tek bir büyük sığalı sunucunun toplam maliyetinden daha az olur. Bu yüzden belirli bir başarım beklentisinin üzerine çıkıldığında, dikey ölçeklemeye göre daha ekonomiktir.

Yatay ölçekleme Çizim 1.4'de gösterildiği gibi ideal ölçekleme değerlerine daha yakın olmasının yanı sıra, birim sayısının artması bakım giderlerini arttırır. Uygulamanın koşut işletim yönünde ve iş yükünün faklı birimler arasında dağıtılması düzenlemelerini gerektirir. Bu yük dengeleme yeni yazılım veya donanım gerektirebilir.



Çizim 1.4 Yatay Ölçeklemede İşleme Sığası-Hizmet Sığası Çizimi [5]

2. VERİ TABANI ÖLÇEKLEME MİMARİLERİ

Veri tabanı yönetim sistemi, veri tabanı işlemlerini gerçekleştirmek üzere bütünleşen yazılım ve donanımdan meydana gelir. Veri tabanı sistemleri veri kümelerini işlemek için, verinin depolanacağı alanı tanımlar. Depolanan veri üzerinde arama ve değiştirme için gerekli mekanizmalar ile sistemin yönetimini

sağlar. Veri tabanı yönetim sistemi, ek olarak sistem çökmelerine ve veriye yetkisiz erişim girişimlerine karşı koyacak şekilde bilginin güvenilir ve güvenli depolanmasından sorumludur. Veri tabanı nesneleri birden fazla kullanıcı arasında paylaşıldığında, veri tabanı yönetim sistemi kullanıcılar arasında oluşabilecek veri tutarsızlıklarına da engel olur [5] [6].

Veri tabanı, "veri nesneleri" adı verilen, veri kümelerinden oluşur. Her veri nesnesi belirli bir anda atanmış bir değere sahiptir. Veri tabanı durumu ise her zaman dilimindeki veri nesnelerinin değerlerinden oluşur [6].

Günümüzde, veri aktarma işlemleri için donanımların fiber kanallar ile birbirine bağlaması oldukça yaygın bir teknoloji haline gelmiştir. Veri tabanı sunucularında ise koşut işlem gücünün artması ve donanımların önemli şekilde gelişmesi sayesinde, veri tabanı ölçeklemesi üzerindeki tartışmalar artık ölçeklemenin sınırları değil, en uygun şekle sokma üzerine yapılmaktadır [7].

Günümüzdeki gelişmiş işlemcili donanımlar üretilmeden çok önce, daha yüksek veri tabanı başarımı gerektiren projelerde kullanmak üzere bazı ölçekleme teknikleri geliştirilmiştir. Bu tekniklerden biri, eğer mümkünse büyük tabloların farklı sunuculara dağıtılmasıdır. Bu dağıtım yapılırken büyük tablolar, ana anahtar aralıklarına göre dilimlenir. Bu işlem başarıyla gerçekleştirildiğinde işlem gücü düşük sunucu sayısı arttırılarak, toplamda daha yüksek veri tabanı işlem gücü elde edilebilir. Bir başka yöntem ise, aralarında hızlı bağlantı olan işlem gücü düşük sistemlerin ortak paylaşılan bir önbelleği güncelleyerek sık kullanılan sorgulara aynı noktadan hızlı erişim sağlamaya çalışmalarıdır [7].

Yazılımların geliştirilmesinde çıkardıkları karmaşıklık, kullanılması gerekli olan orta katman yazılım ve donanımları, veri kaybı tehditleri gibi sebeplerle birlikte, sistemlerdeki veri depolama kısıtının aşılmış olduğu günümüzde, bu yaklaşımlar yerini birebir kopyalanarak çoğaltılan veri tabanlarına bırakmıştır.

Çoğaltma sistemlerinin oluşturulmasındaki ana nedenlerden biri verinin kopyalarını farklı donanımlar üzerinde tutarak olası bir çökme durumunda hizmet kesintisine veya veri kaybına engel olmak, diğeri ise ölçeklenirliği arttırmaktır. Bilgiyi bir küme bilgisayarda bulundurma ana çalışma ilkesi verinin kopyalarını küme içindeki bilgisayarlara çoğaltmak ve aralarında yayınlamaktır. Ölçekleme kuramına göre, sisteme daha fazla bilgisayar eklenmesi işlem hacminin artmasını sağlamalıdır [8].

Ölçeklemenin etkin çalışabilmesi için işlemler birçok sunucuya dağıtılmalıdır. Okuma işlemlerinin sunucular arasında dağıtmasıyla, daha verimli ve ölçeklenir sistemler oluşturulabilir [8].

Çoğaltılmış veri tabanı sistemlerinde güncelleme işlemleri daha duyarlı ve tehlikeye açıktır. Güncelleme işleminin etkilediği her sunucuda güncellenmiş bilginin bir kopyası bulunmalıdır. Bu da çoğaltma yöntemlerinin geliştirilmesine sebep olmuştur [8].

2.1. Bütünlük Sağlama Yöntemine Göre Çoğaltma

Veri tabanı sistemlerinde çoğaltılan veriler arasında bütünlük, güncellemeleri zaman uyumlu ve zaman uyumsuz olarak yürüten iki ayrı yöntemle sağlanır.

Zaman uyumlu ve zaman uyumsuz güncelleme teknikleri arasındaki temel fark, zaman uyumlu güncelleme mimarisinde değişikliklerin kümedeki diğer sunuculara anında yansıtılması ve veriler güncellenene kadar istemciyi bekleterek yapılan işlemin sonucunun döndürülmesidir.

2.1.1. Zaman uyumsuz güncelleme

Zaman uyumsuz güncelleme yönteminde, veri tabanı sunucularının birinde yapılan değişiklikler diğer veri tabanı sunucularına anında ve tümüyle yansıtılmaz. İşlemlerin diğer veri tabanlarına yansıması için geçen sürenin uzun veya kısa olması sistemlerin yükü ve aralarındaki iletişim bağlantısının başarımına göre değişir. Sistem kurulumunda sunucular arasındaki bağlantı zaman kısıtına bağımlı olmadığı için sunucular arasındaki ağ bağlantısında yüksek başarım zorunluluğu aranmaz, bu sebeple geniş alan ağ bağlantıları için uygun bir çözümdür. Bu özelliği sayesinde felaket kurtarma kurguları için oldukça tercih edilir hale gelmektedir. Öte yandan sunucular arasındaki gecikmenin uzaması, sunucuların birinin çökmesi halinde, güncellemenin diğer sunuculara yansımayan kısımlarında veri kaybına sebep olabilir [9].

2.1.2. Zaman uyumlu güncelleme

Zaman uyumlu güncelleme yönteminde, veri tabanı yönetim sistemine bağlanan istemcinin yaptığı bir değişikliğin, işlem tamamlanmadan diğer sistemlere de işlenmesi sağlanır. Aksi halde uygulamaya işlemin başarısız olduğu ve yapılmadığı bilgisi verilir. Bu yöntemin zaman uyumsuz yönteme göre üstünlüğü, sistemlerden birinin çökmesi halinde yapılan tüm değişiklikler sistemler arasında koşut olarak işletildiği için veri kaybı olmamasıdır [9].

Sunucular arasındaki bağlantı kesilecek olursa, veri değiştirme işlemleri diğer sunuculara yayılamadığı için işlemler başarısız olacaktır. Bu durumda çalışmayan sunucunun kümeden çıkarılması gerekir.

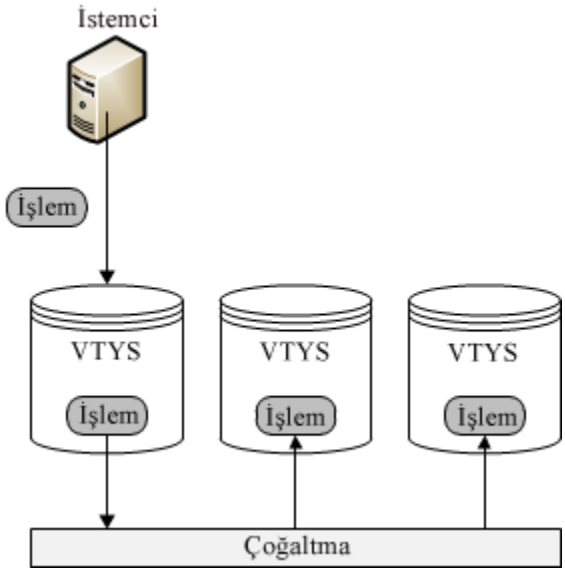
2.2. Kümeleme Yöntemine Göre Çoğaltma

Çoğaltılan veri tabanlarında, verilerin bire bir kopyaları bulunduğundan, uygulamalar verileri herhangi bir sistemden okuyabilir. Yazma işlemi gerçekleştiği anda tutarlılığın sağlanması amacıyla güncellenmiş veri tüm sistemlere birden gönderilir. Bu çoğaltılmış veri tabanı sisteminde verilerin bulunduğu ana sunucuya birincil (master), bu veri kopyalarını birincil sistemden alanlar ise ikincil (slave) veri tabanı olarak adlandırılır.

Birincil ve ikincil veri tabanı sistemlerinin kümelenmesine göre, iki mimari model ortaya çıkmaktadır. Bir birincil ve en az bir ikincil veri tabanı sisteminden oluşan mimari birincil-ikincil sistem modeli olarak, en az iki birincil veri tabanı sisteminden oluşan kümeler çoğaltılmış birincil sistem modeli olarak adlandırılır.

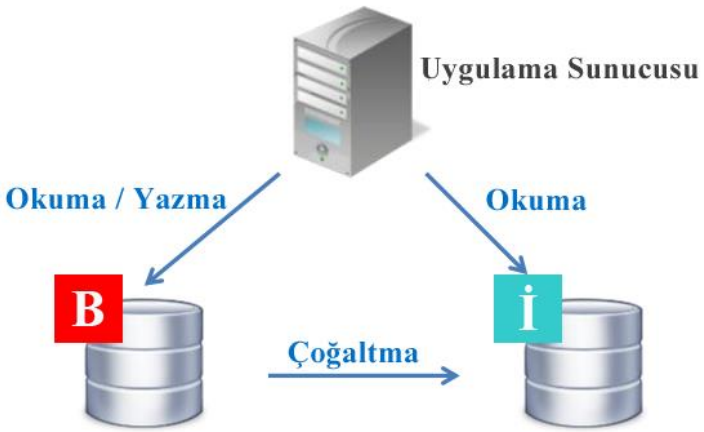
2.2.1. Birincil-ikincil sistem modeli

Tek veri tabanı sunucusu kullanan sistemlerde, çoğaltmanın en yalın yolu, verileri fiziki olarak bir birincil ve bir ikincil sunucuda tutmaktır. İkincil sunucu sayısı gereksinimlere göre artabilir. Böylece sistemler veri kaybı tehlikesini azaltır ve çöken bir veri tabanı sunucusundan daha az etkilenirler. Birincil veri tabanı sunucusunun çökmesi halinde, ikincil veri tabanı sunucusu, birincil sunucu olarak hizmete devam etmek üzere düzenlenebilir.



Çizim 2.1 Birincil-İkincil Veri Tabanı Sistemi Çalışma İlkesi [11]

Kullanıcı, uygulama sunucusundan veya doğrudan istemci bilgisayarlardaki uygulamadan bağlantı kurarak, birincil veri tabanında istediği değişikliği yapabilir. Çoğaltma işlemi tamamlandıktan sonra veriye birincil ya da ikincil sunucuların herhangi birinden erişebilirler. Ancak tüm yazma işlemleri doğrudan yalnız birincil sunucuya yapılır. Bunun nedeni ise birincil sunucunun veriyi daha sonra tek yönlü olarak ikincil sistemlere kopyalamasıdır.

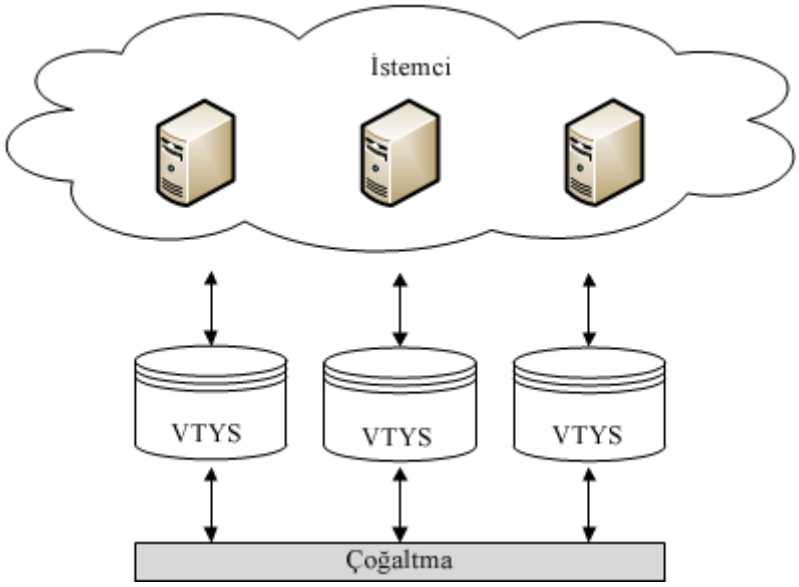


Çizim 2.2 Birincil-İkinci Veri Tabanı Sistemli Uygulama [4]

Veri tabanı çoğaltma işlemi ile düzenlenmiş bir yük dengeleyici, okuma taleplerini birden fazla sisteme gönderebilir. Bunun yanında tek birincil sisteme sahip olmak birincil sunucunun çökmesi halinde ikincil bir sunucunun birinci sunucu olarak atanmasına veya birincil sunucunun yerinden çalışır hale getirilmesine kadar sistemin veri tabanındaki değiştirme işlemlerine kapalı olacağı anlamına gelir. Tek birincil sunucunun çalışır durumda olması güncelleme işlemleri için zorunluluk olduğu için tüm sistemin hizmet verememe riski, çoğaltılmış birincil veri tabanı sistemlerine göre daha fazla olacaktır.

2.2.2. Çoğaltılmış birincil sistem modeli

Çoğaltılmış birincil sistem modelinde, her birincil sunucu güncellenen veriyi küme içindeki diğer sunuculara yayarak, verinin kümedeki tüm sunucularda güncel kopyalarının tutulmasını sağlar.



Çizim 2.3 Çoğaltılmış Birincil Veri Tabanı Sistemi Çalışma İlkesi [11]

Veri güncelleme işlemi için birden fazla bilgisayar sorumlu olması nedeniyle, birincil-ikincil veri tabanı çoğaltma modelinde karşılaşılan, tek sunucuya bağımlı olarak hizmet verebilme kısıtı ortadan kalkmaktadır. Ancak bu modelle birlikte de aynı verinin farklı bilgisayarlarda güncellenmesiyle ortaya çıkan, veri tutarsızlığı sorunuyla karşı karşıya kalınmaktadır.

Bu sorunun oluşumu basitçe kurgulanacak olursa; eğer bir veri tabanı tablosunda satır kimlik numarasının otomatik artım özelliği kullanılıyorsa ve iki farklı kayıt aynı zaman diliminde, farklı birincil sunuculara eklenmeye çalışılırsa tutarsızlık sorunu ile karşılaşılır. Bu durum eş sayılı satırların aynı zaman diliminde güncellemeye çalışılmasında da oluşacaktır.

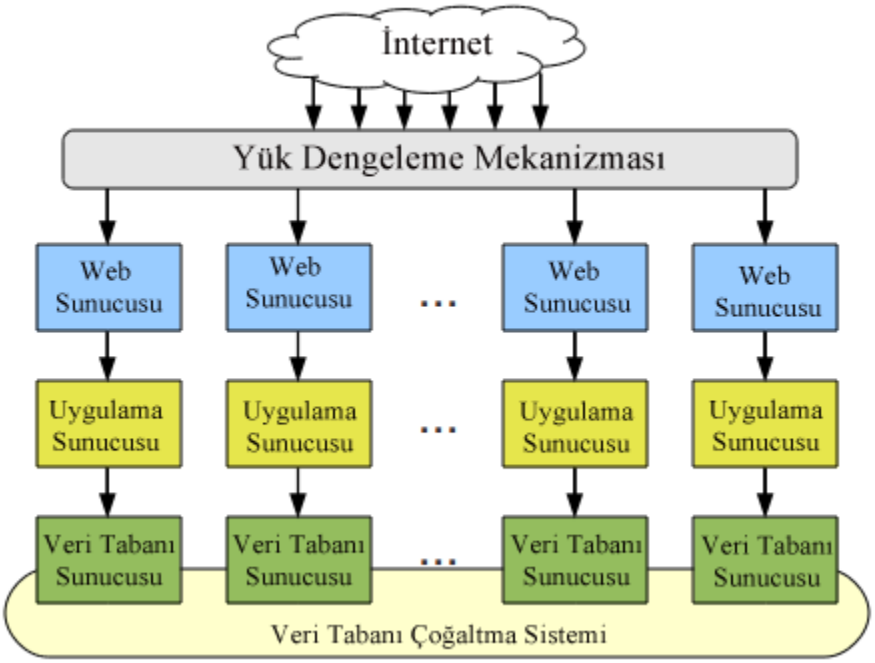
Veri tutarsızlığını engellemede birden fazla yöntem kullanılmaktadır. Bunlardan biri yazılan uygulamanın kodunda belirli bir tablonun sadece bir birincil sunucuya yazılmasını ve bu birincil veri tabanı sisteminden diğer birincil sunuculara verinin dağıtılmasıdır. Bu yöntem geleneksel veri tabanı parçalama yöntemine (database sharding) benzemektedir. Bir diğer yöntem ise veri tabanı işlemlerini okuma ve yazma olarak ikiye ayırarak bağlantıları farklı sunucularla kurarak yapılır.

Çizim 2.4 Çoğaltılmış Birincil Veri Tabanı Sistemli Uygulama [4]

3. ÇOĞALTILMIŞ BİRİNCİL SİSTEM KULLANILAN ÖLÇEKLEME MİMARİLERİ

Uygulamalar, çoğaltılmış birincil veri tabanı sistemini kullanılarak, birçok şekilde ölçeklenebilir. İhtiyaçları karşılamaya en uygun mimari, maliyet, yüksek başarım ve kurulum kolaylığı ölçütlerine göre belirlenir.

3.1. Veri Tabanı Yükünü Dengelemeyen Mimariler

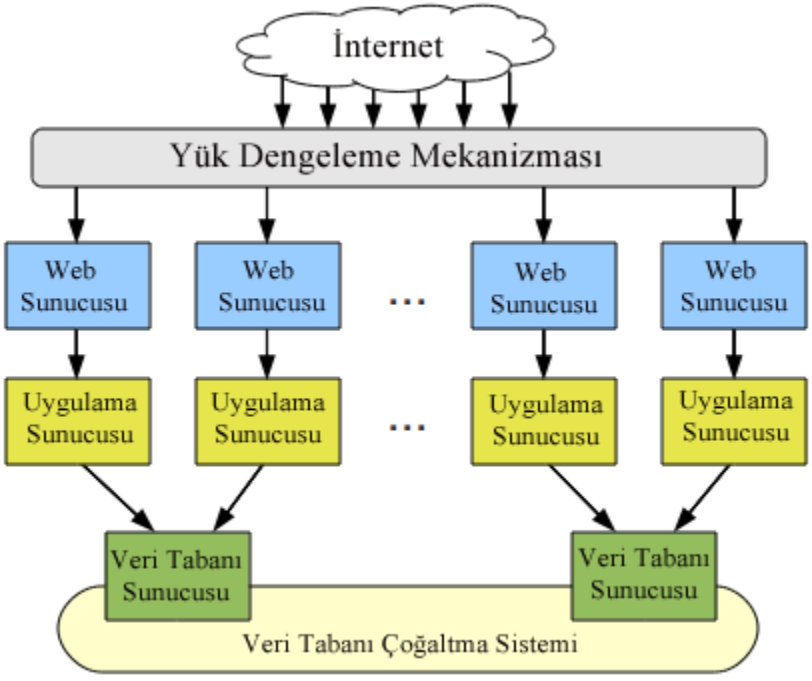


Çizim 3.1 Yük Dengeleme Yapılmayan Mimari [12]

Bu mimaride, sistem bileşenleri web, uygulama ve veri tabanı sunucuları Çizim 3.1 de gösterildiği gibi veri tabanı katmanına kadar, dikey eksende ardıl olarak

bağlanır. Sistem katmanları arasında yatay ilişki bulunmaz. Dikey olarak sıralanmış her sistemin kendi veri tabanı sunusu bulunur. Bu mimari, kurulumu ve yönetimi basit bir çözümdür. Özellikle tüm ardıl bileşenler tek bir fiziksel sunucuda çalışacaklarsa kurulum işleri diğer mimarilere göre daha kolay olur.

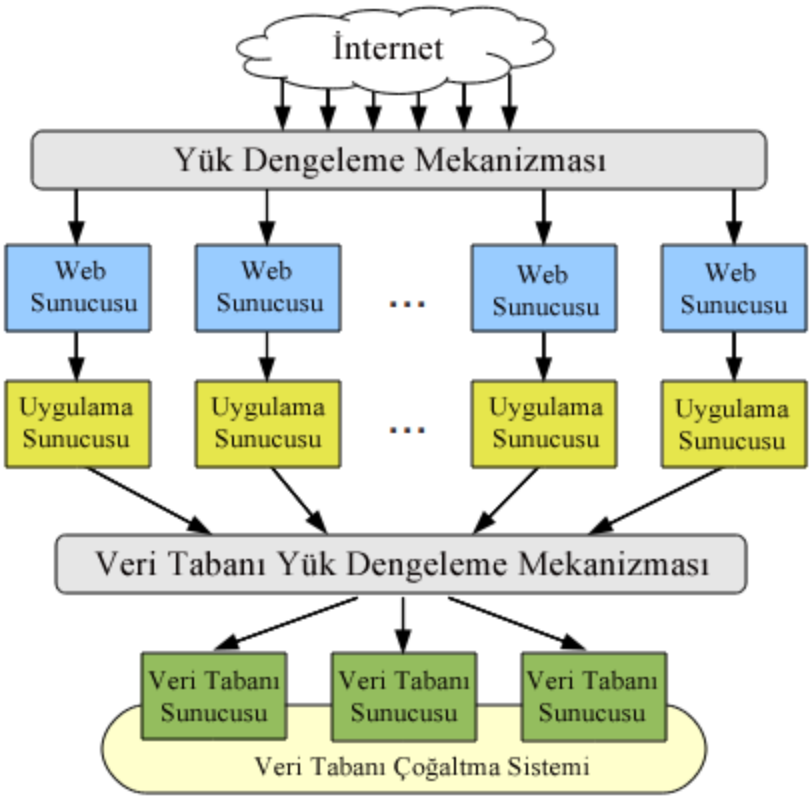
Bu mimaride dikey eksende oluşan bir sorun, o dikey eksen üzerindeki tüm katmanların hizmet vermesini engelleyebilir. Her uygulama sunucusu için ihtiyaç olmasa dahi mutlaka bir veri tabanı sunucusu bulundurma zorunluluğu vardır. Bunun yanında, tek bir büyük ön bellek yerine, eş boyutta yatay olarak bölünmüş birçok küçük önbellek alanı kullanılması, okuma işlemlerinin ön bellekten karşılanma şansını düşüreceğinden, kaynakların verimli kullanılmadığı söylenebilir. Her ardıl sistem, diğerinin yaptığı işi bire bir taklit eder, veri tabanı yükünü dinamik şekilde dağıtmaya olanak yoktur.



Çizim 3.2 Yük Dengeleme Yapılmayan Mimari [12]

Çizim 3.2' de gösterilen mimari, birkaç uygulama sunucusunu, sabit olarak bir veri tabanına bağlayan melez bir yöntemdir. Böylece yatay olarak veri tabanı ve uygulama sunucu kümesi bağımsız olarak genişletilip daraltılabilir. Bu yöntem dikey olarak tüm katmanların ardıl bağlandığı kümeleme yöntemindeki kaynak yönetiminin, ihtiyaca göre uyarlanabilecek şekilde düzenlenmesi ile ortaya çıkmıştır. Kurulumu ve yönetimi basit olup, dinamik bir yük dengelemesi yapılamaz.

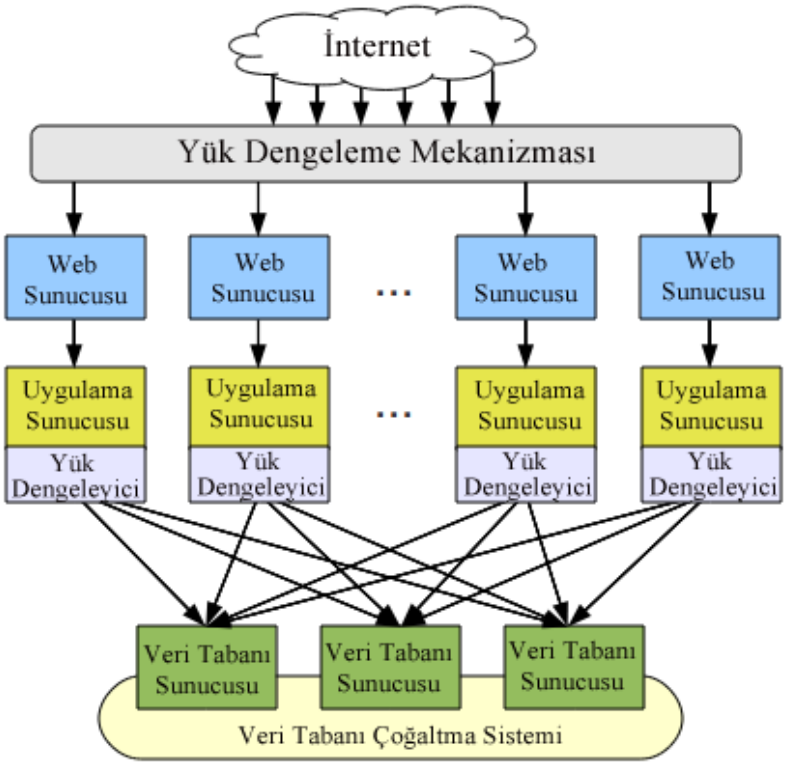
3.2. Veri Tabanı Yükünü Dengeleyen Mimariler



Çizim 3.3 Yük Dengeleme Yapılan Mimari [12]

Bileşenleri ve aralarındaki ilişki çizim 3.3'de gösterilen bu mimaride, dikey eksendeki bir bileşenin sorun çıkarması tüm dikey ekseni etkilemez. Uygulama sunucuları veya veri tabanı sunucularından birinin çalışmayı durdurması halinde diğer sunucular hizmet vermeye devam edecektir. Mimari esnek bir kurulum yapısına sahiptir. Uygulama ve veri tabanı sunucu sayıları ihtiyaca göre birbirinden bağımsız olarak değiştirilebilir. İhtiyaç hangi bölümdeyse oraya takviye yapılır. Veri tabanı sunucuları arasında dinamik yük dengelemesi yapılabilir.

Bu mimarideki yapı karışık olduğu için kurulumu ve yönetimi diğer mimarilere göre daha çok zaman alacaktır. Mimari, veri tabanı kümesi üzerinde bir yük dengeleyici kurulmasını gerektirir. Uygulama sunucuları veri tabanı isteklerini bu sunucuya gönderir, yük dengeleyici de veri tabanı kümesinden çalıştığı yük dengeleme yöntemine göre bir veri tabanı sunucusu seçerek isteği iletir. Özetle, fazladan bir sunucu gerektirdiği gibi, tüm veri tabanı sunucularına erişim bu yük dengeleyici üzerinden geçtiği için çökmesi halinde tüm sistem hizmet veremez hale gelir.



Çizim 3.4 Yük Dengeleme Yapılan Mimari [12]

Çizim 3.4'de gösterilen, dağıtık yük dengelemeli veri tabanı üstü kümeleme mimarisi, veri tabanı üstü kümeleme mimarisinin risklere karşı geliştirilmiş halidir. Sistem tek bir yük dengeleyiciye bağımlı kalmak yerine, her uygulama sunucusu üzerinde kendi yük dengeleyicisi olacak şekilde düzenlenmiştir. Böylece yük dengeleyici üzerindeki iş yükü de dengelenmektedir.

Veri tabanı kümesinde, sunucu ekleme veya eksiltme yapıldığında, her uygulama sunucusunda yeni kümeye göre düzenleme yapmak gerekecektir.

4. VERİ TUTARLILIĞI

Çoğaltılmış bir veri tabanı sisteminde, her biri veri tabanın bir kopyasını yürüten birden fazla veri tabanı sunucusu bulunur. Bu yüzden veri tabanında her bir mantıksal nesnenin birden fazla fiziksel kopyası vardır. Veri tabanı işlemleri, okuma ve yazma işlerini, mantıksal nesnelere gönderdikleri zaman çoğaltılmış sistemin, çoğaltma denetim bileşeni bu işlemleri fiziksel veri kopyalarındaki kayıtlara dönüştürür. Çoğaltılmış sistemin tutarlılık denetim bileşeni bu işlerin yürütülme sırasını denetler. Tutarlılık denetimi için kötümser, yarı iyimser ve iyimser yaklaşımlar kullanılmaktadır [10].

4.1. İyimser Tutarlılık Denetimi

İyimser tutarlılık denetimi, ya da bilinen diğer isimleriyle, doğrulama veya belgeleme tabanlı tutarlılık denetimi sistem üzerinde bir çakışma olmayacağını varsayar ve bir çakışma olana kadar da etki etmez.

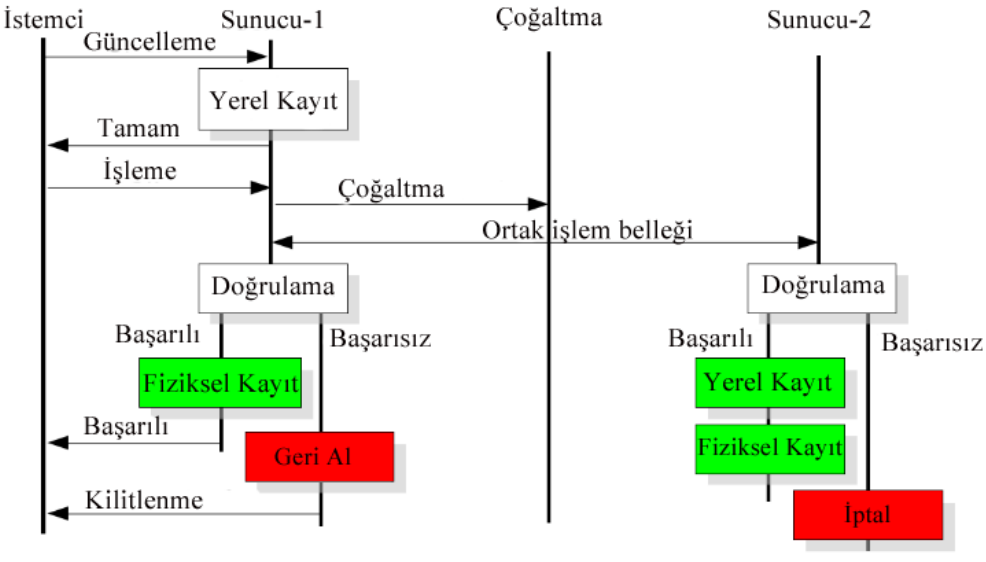
Bu yöntemde veri tabanı işlemi sonuçlanana kadar güncelleme içeren veri tabanı öğeleri veri tabanına kaydedilmez. Veri tabanı işleminin çalıştırılması sırasında tüm güncellenecek veriler, işlemin sonunda fiziksel alanlarına kaydedilmek üzere yerel değişkenlerde saklanır. Veri tabanı işleminin sonuçlandığında, işlemin tutarlılığını (güncelleme sırasını ihlal edip etmediğini) denetleyen bir doğrulama aşamasından geçer. Doğrulama aşaması için gerekli zaman damgası (timestamp) bilgisi, veri tabanı yönetim sisteminin tutarlılık mekanizması tarafından tutulur. Zaman damgasına göre, güncelleme sırasına uyulmuşsa, yerel değişkende tutulan veri, veri tabanı üzerinde kalıcı hale getirilir, aksi halde işlem yarıda kesilir kilitlenme bilgisi verilir.

Bu tutarlılık denetim algoritması sırası ile üç aşamadan meydana gelir;

1. Okuma aşaması: İşleme alınacak verileri okunur ve yerel değişkenlere atanır. Tüm yazma işlemleri geçici yerel değişkenlerde tutulur ve bu sırada gerçek veri kaynağına yazma olmaz.

2. Doğrulama aşaması: yazma işleminin, yerel değişkenlerde bulunan verilere bakılarak işlem sırasına uygunluğu denetlenir.

3. Yazma aşaması: Doğrulama işlemi başarılı olursa, geçici değişkenlerde tutulan yazma işlemleri veri tabanına kalıcı olarak kaydedilir. Doğrulama işleminin başarılı olmadığı durumda ise yapılan işlemim tümü iptal edilir [11].



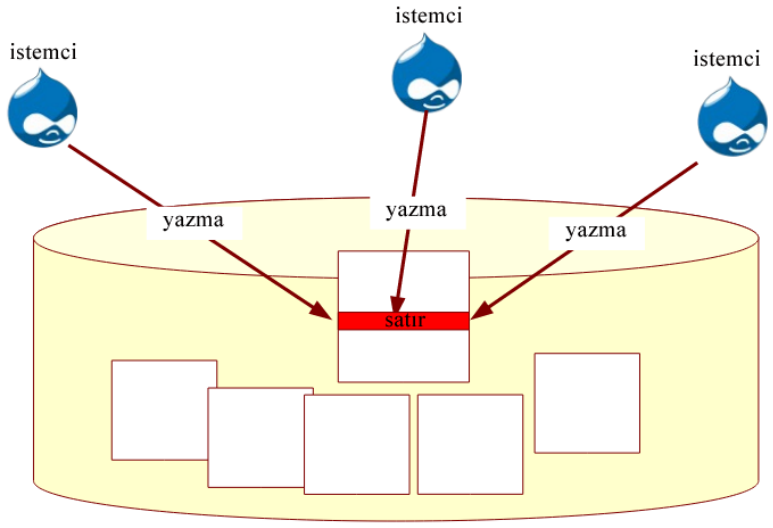
Çizim 4.1 İyimser Tutarlılık Denetimi [15]

4.2. Kilitlenme

Tekil çalışan veri tabanı sisteminde farklı nesneler tutarlı olmak zorundadır. Çoğaltılmış veri tabanında ise bu tutarlılığı sağlamak tekil veri tabanı sistemine göre daha karmaşıktır. İdeal durumda çoğaltılmış bütün veri tabanı sistemleri arasında ortak tutarlılık sağlanmalıdır. Ortak tutarlılık derken şu kastedilmektedir, veri nesnelerinin kopyaları kümedeki tüm çoğaltılmış veri tabanı sistemlerinde aynı değerlere sahip olmalıdır.

Aynı zaman diliminde ve aynı veri kümesi üzerinde işlem yapılması halinde 3 durum gözlenir. [12].

- Okuma-okuma: Kilitlenme oluşmaz. Belirli bir veri kümesi üzerinde aynı anda birçok okuma işlemi çakışma olmadan yapılabilir.
- Yazma-okuma: Kilitlenme Oluşmaz. Yazma işlemi tamamlamadan gelen okuma işlemleri günceliğini yitirmiş olabilir.
- Yazma-yazma: Kilitlenme oluşur. İşlemlerden biri iptal edilecektir.



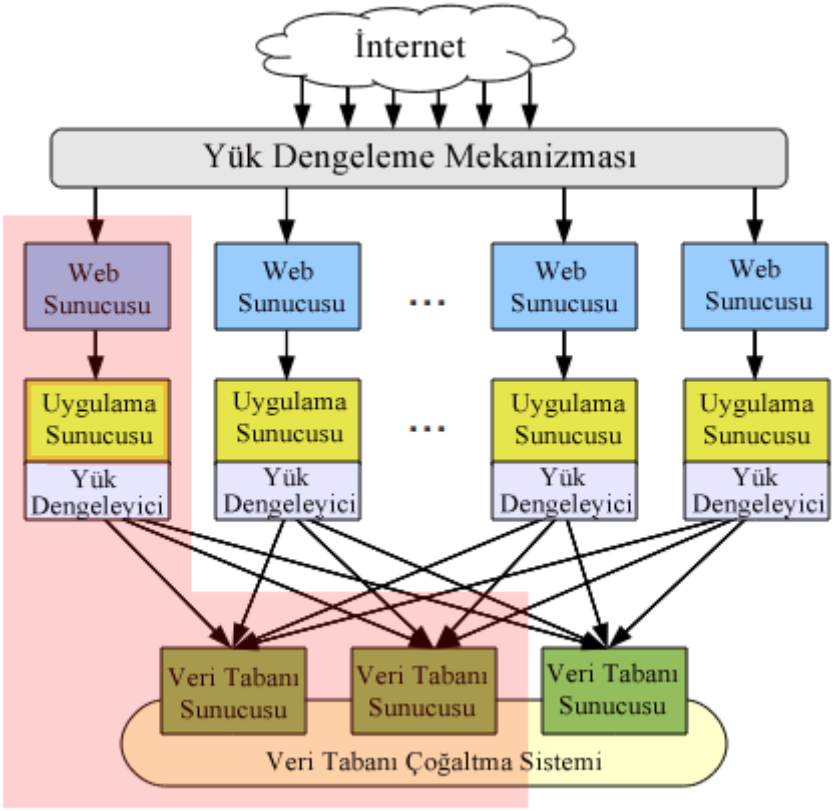
Çizim 4.2 Kilitlenme Oluşumu [17]

Aynı zaman diliminde, farklı istemciler, aynı veri kümesine yazma girişimlerine, veri çekişmesi adı verilir. Paylaşım gerektiren ve ortak verileri güncelleyen istemcilerin bulunması veri çekişmesini doğurur. Veri çekişmesinin yoğun olduğu satırlarda, aynı zaman diliminde gelen yazma-yazma işlemleri veri tabanında kilitlenmelere sebep olur. Çizim 4.2'de veri çekişmesi gösterilmektedir.

İyimser tutarlılık kontrolü, kilitlenmeleri tespit ederek, zaman damgasına göre, kurban olarak seçilen işlemi (son gelen işlem) iptal etmektedir. Yazma işlemlerinin tamamını tek bir sunucuya gönderilmesi kilitlenmelerden kaçınma yöntemi olarak önerilmektedir.

5. ÇOĞALTILMIŞ BİRİNCİL SİSTEMLİ MİMARİDE KİLİTLENMELERİ AZALTMA

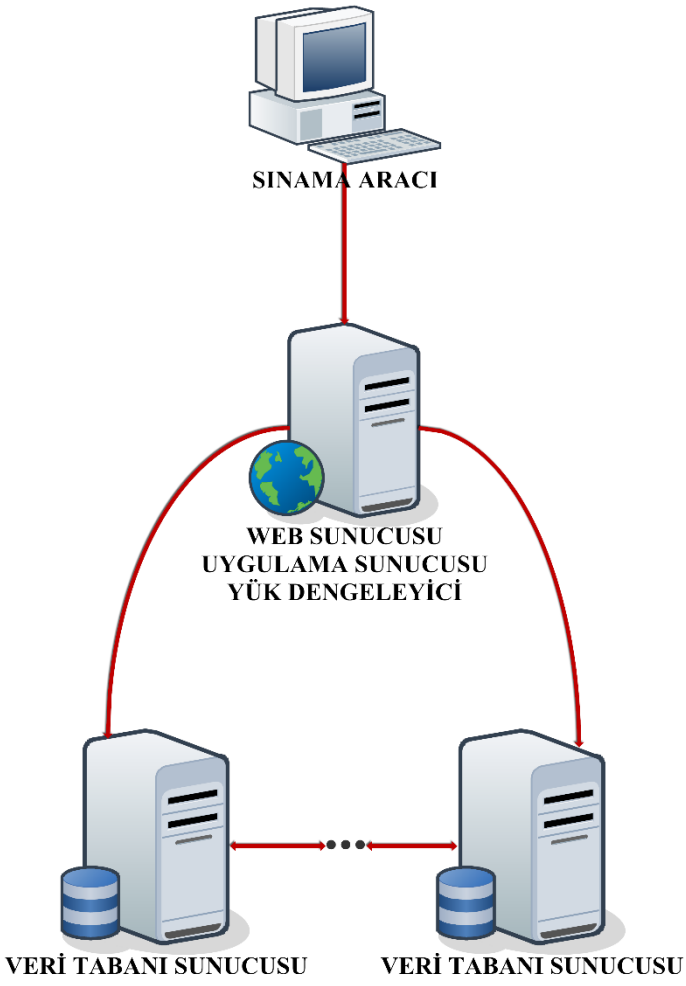
Veri tabanı katmanında yük dengeleme yapan, yatay ölçekleme mimarisi Çizim 5.1'de gösterilmiştir. Tezde kullanılan sınama düzeneği, taralı alan üzerinde gösterilen, bir web sunucusu, bir uygulama sunucusu, bir yük dengeleyici ve değişken sayıda veri tabanı sunucusundan oluşmaktadır. Veri tabanı sunucularının, çoğaltılmış birincil sistem yapısında çalışması için en alt katmanda veri tabanı çoğaltma sistemi kurulmuştur.



Çizim 5.1 Sınanan Mimari [12]

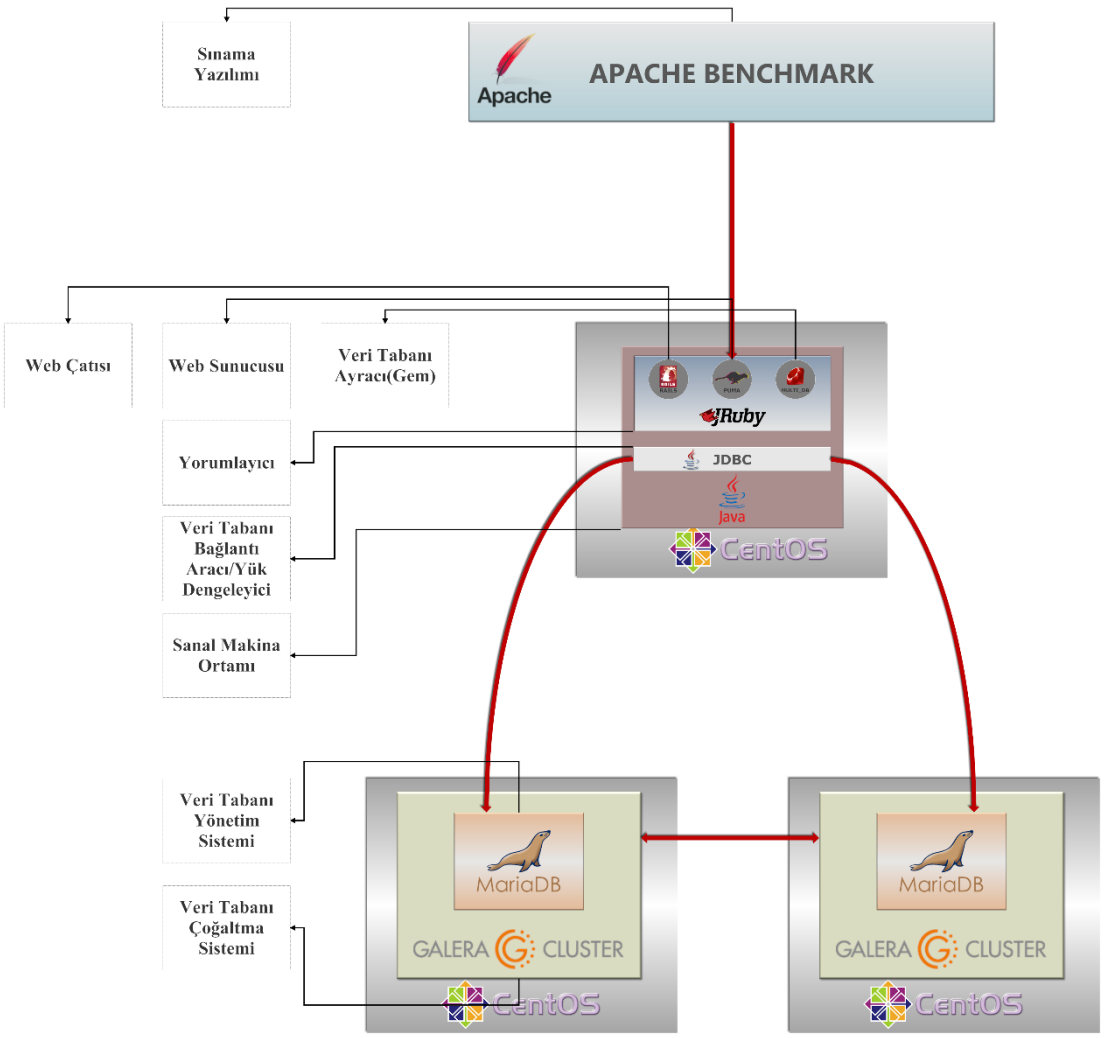
Sınama düzeneğinde yazılım bileşenlerinin fiziksel sunuculara dağılımı Çizim 5.2'de gösterilmektedir. Web sunucusu, uygulama sunucusu ve yük

dengeleyici aynı sunucu üzerinde tutulurken, sistemin veri tabanı değişken sayıda farklı fiziksel sunucu üzerinde çalıştırılmıştır. Bunların dışında bir sunucu da sistemin başarımını ölçmek amacıyla sınama yazılımını işletmektedir.



Çizim 5.2 Sınama Düzeneği Fiziksel Kurulumu

Zaman uyumlu, çoğaltılmış birincil veri tabanı kullanılan sistem, veri tabanı katmanının üzerinde yük dengelenecek şekilde kurulmuştur. Sınama düzeneğini kurmak için kullanılan yazılım bileşenleri Çizim 5.3'de gösterilmiştir.



Çizim 5.3 Sınama Düzeneği Yazılım Bileşenleri

Sınama yazılımı için kurulan sistem Apache Benchmark isimli açık kaynak kodlu uygulama ile sağlanmaktadır. Web sunucusuna istenilen sayıda istek göndererek başarımlarını ölçen bu araç, Linux işletim sisteminde çalışan web sunucusu Apache'yi test etmek üzere hazırlanıp daha sonra bağımsız kullanıma sunulmuştur.

Uygulama sunucusu oluşturan her bileşen Java (JVM) üzerinde çalışmaktadır. Ruby dilinin, Java üzerinde çalışmasını sağlayan yorumlayıcı (JRUBY) aynı zamanda web sunucusunu da çalıştırmaktadır. İsteklerin gönderildiği web sunucusu katmanı Puma adı verilen Ruby için hazırlanmış web sunucusu ile oluşturulmuştur.

Web sunucusuna gelen istekler Jruby yorumlayıcısı ile Rails adlı web çatısına gelmektedir. Rails çatısı, gelen http isteklerini algılayıp web ortamı için hazırlanmış kodu çalıştırmaktadır. Rails çatısı altında çalışabilen, Multi_db adlı Ruby bileşeni ile okuma ve yazma sorguları tespit edilip, istenilen veri tabanlarına dağıtılabilmektedir.

Uygulama sunucusu üzerindeki tüm bileşenlerin Java sanal makinesi üzerinde çalışması sistemin hem platform bağımsız çalışabilmesine hem de veri tabanı yük dengelemesi için JDBC(Java tabanlı veri tabanı bağlantı aracı)'nin kullanılmasına olanak verir.

Veri tabanı yönetim sistemi olarak kullanılan MariaDB, Linux işletim sistemlerinde yaygın olarak kullanılan MySQL'in kaynak kodundan türetilmiş bir veri tabanı yönetim sistemidir. Çoğaltılmış birincil veri tabanı mimarisinde çalıştırılmak üzere özelleşmiş sürümü bulunmaktadır. GNU Genel Kamu Lisansı (GPL) altında dağıtılarak ücretsiz olarak kullanılabilen sistemin geliştirilmesi, ilgili topluluk tarafından sürdürülmektedir.

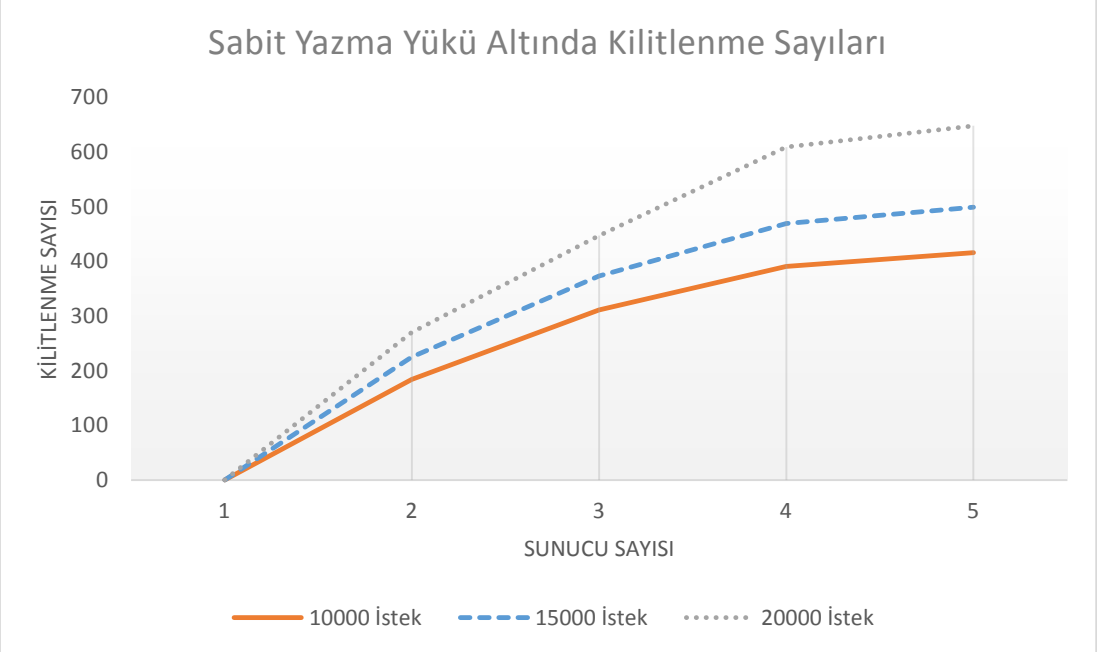
MariaDB'nin zaman uyumlu ve çoğaltılmış birincil veri tabanı ilkesiyle iki sunucu üzerinde koşut olarak çalıştırılmasını sağlayan, Galera Cluster kümelemesi, birden fazla işletim sistemi üzerine kurulmuştur. Veri tabanı sunucuları arasındaki veri aktarımı, yerel ağ bağlantısı üzerinden gerçekleştirilmektedir.

Kilitlenmelerin azaltılması için, uygulama sunusundan veri tabanına giden sorguların türünün tespit edilip, önceden seçilmiş veri tabanı sunucusuna gönderilmektedir.

6. SINAMA VE BULGULAR

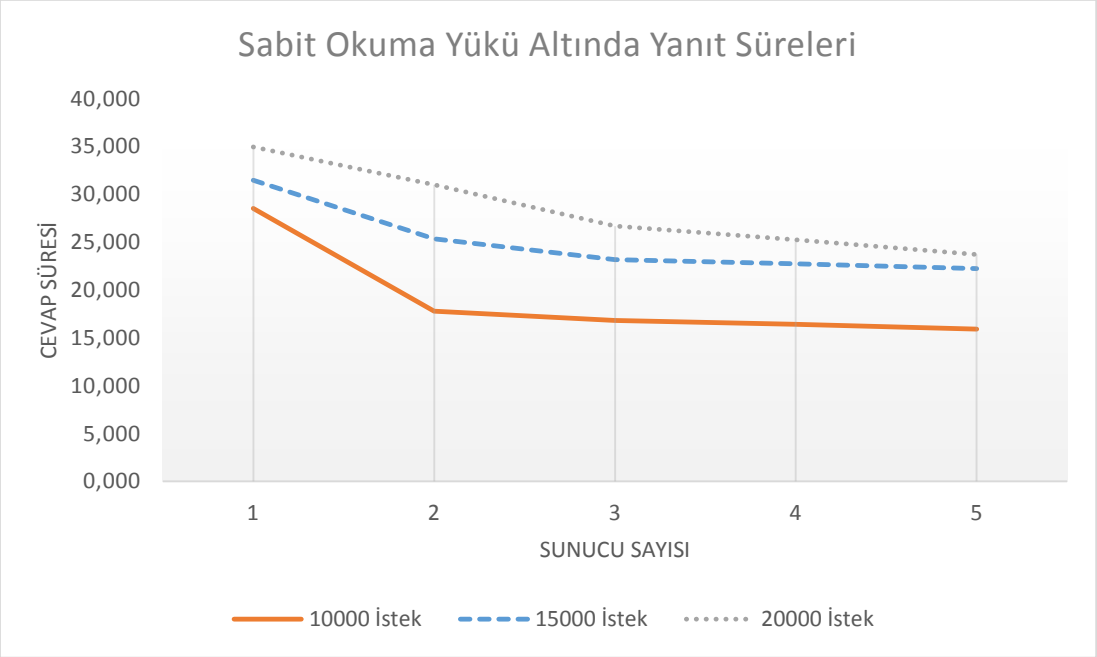
Veri tabanı çoğaltma, ölçeklemede yaygın olarak kullanılmaktadır. Çoğaltılmış birincil veri tabanı kullanılarak, değişken sayıda veri tabanı sisteminin oluşturduğu bir uygulama düzeneği kurulmuştur. Bu düzenekte, yöntemin başlıca sorunu olan kilitlemeler ele alınarak, kilitlenmeleri azaltmak için okuma ve yazma isteklerinin farklı sunuculara dağıtılması ve dağıtım yapılmadan işlenme durumları incelenmiştir.

Çoğaltılmış birincil veri tabanı sisteminde, yazma işlemlerinin tek bir sunucuda yapılması ve diğer sunuculara yayılmasıyla kilitlenmelerin önlendiği görülmüştür. Öte yandan, uygulama sunucusunda okuma-yazma işlemlerinin türlerine göre ayrılmadan, faklı veri tabanı sunucularına dağıtıldığı durumda, veri tabanı sunucularının artması ile kilitlenmelerin de arttığı görülmüştür. Çizim 6.1'de, artan sunucu sayısı ile birlikte kilitlenmeler neredeyse doğrusal olarak artmaktadır.



Çizim 6.1 Sabit Yazma Yükü Altında Kilitlenme Sayıları

Sistemdeki ölçeklenirlik incelediğinde ise, okuma işlemlerinde sunucu sayısının başarıma katkısı, uygulama sunucusundan gelen yanıt süreleri incelenerek yorumlanmıştır. Sabit yük altında, veri tabanı sunucu sayısı 1'den 2'ye çıkarıldığında elde edilen yanıt süresindeki kısalma oranı, daha fazla sunucu eklemesiyle elde edilen sürelere oranla fazladır. Yani sabit yük altında sunucu sayılarını arttırmak Çizim 6.2'de gösterildiği gibi azalan verimlilik göstermektedir.



Çizim 6.2 Sabit Okuma Yükü Altında Yanıt Süreleri

İncelenen uygulama mimarisinin beklendiği gibi ölçeklendiği, okuma-yazma işlemleri kurallara bağlı olarak dağıtıldığında veri tabanındaki kilitlenmelerin azaldığı saptanmıştır.

Bundan sonraki çalışmalarda, bir sistem için kilitlenme ve başarım oranlarını gözlemleyerek, en yüksek başarım ve en az kilitlenmenin oluşacağı veri tabanı sunucu sayısını belirlemek üzere bir yöntem geliştirilebilir.

KAYNAKLAR

1. D. Agrawal, A. El Abbadi, S. Das ve A. J. Elmore, "Database Scalability, Elasticity, and Autonomy in the Cloud," içinde *Database Systems for Advanced Applications*, Springer Berlin Heidelberg, 2011, pp. 2-15.

2. Microsoft, "Technet," [Çevrimiçi]. Available: http://technet.microsoft.com/tr-tr/library/cc776523%28v=ws.10%29.aspx. [Erişildi: 20 3 2014].

3. M. D. Hill, "What is scalability?," *ACM SIGARCH Computer Architecture News*, no. 18.4, pp. 18-21, 1990.

4. A. Khare, Y. Huang, H. Doan ve M. S. Kanwal, "Scalability," içinde *A Fresh Graduate's Guide to Software Development Tools and Technologies*, National University of Singapore, 2012, pp. 2-24.

5. "Scalability," [Çevrimiçi]. Available: http://www.diranieh.com/DistributedDesign_1/Scalability.htm. [Erişildi: 25 4 2014].

6. A. Silberschatz, H. F. Korth ve S. Sudarshan, Database system concepts, New York: McGraw-Hill, 2002.

7. V. Hadzilacos, P. A. Bernsetein ve N. Goodman, "Concurrency Control and Recovery in Database Systems", Massachusetts: Addison-Wesley Publishing Company, 1987.

8. Base On Internationlar Corp., [Çevrimiçi]. Available: http://www.boic.com/scalability.htm. [Erişildi: 25 03 2014].

9. R. Jiménez-Peris ve M. Patiño-Martínez, "Replication for Scalability," içinde *Encyclopedia of Database Systems*, Springer US, 2009, pp. 2403-2408.

10. Codership, "codership.com," [Çevrimiçi]. Available: http://codership.com/products/galera_replication. [Erişildi: 31 3 2014].

11. "Database Replication," codership, [Çevrimiçi]. Available: http://galeracluster.com/documentation-webpages/introduction.html. [Erişildi: 22 5 2014].

12. "Cluster Deployment Variants," codership, [Çevrimiçi]. Available: http://galeracluster.com/documentation-webpages/deploymentvariants.html. [Erişildi: 22 5 2014].

13. B. Kemme, "Replicated Database Concurrency Control," içinde *Encyclopedia of Database Systems*, Springer US, 2009, pp. 2390-2391.

14. R. Elmasri ve S. B. Navathe, "Validation (Optimistic Concurrency Control Techniques)," içinde *Fundamentals Of Database Systems*, Addison-Wesley, 2011, pp. 794-795.

15. "Certification Based Replication," [Çevrimiçi]. Available: http://galeracluster.com/documentation-webpages/certificationbasedreplication.html. [Erişildi: 20 03 2014].

16. H. Stockinger, "Data Replication in Distributed Database Systems," *CMS-NOTE*, no. 046, 1999.

17. S. Jaakola, "Galera Cluster Best Practices," 2013.

18. R. Cattell, “Scalable SQL and NoSQL data stores,” *ACM SIGMOD Record*, no. 39.4, pp. 12-27, 2010.

19. T. Atwood, “Vertical and Horizontal Computing Architectures - Trends and Attributes,” içinde *SUPerG*, Berlin, 2003.

ÖZGEÇMİŞ

Şemseddin AKSOY, 1984 yılında Şanlıurfa'da doğdu. Öğrenimlerini sırasıyla Bahçelievler İlkokulu ve Şanlıurfa Anadolu Lisesi'nde tamamladı. 2002 yılında Maltepe Üniversitesi Bilgisayar Mühendisliği Bölümünde eğitim görmeye hak kazandı. Öğrenciliği esnasında Maltepe Üniversitesi Uzaktan Eğitim projesinde yazılım geliştirme görevi aldı. Mezun olduktan sonra üniversite içinde ve dışında, çeşitli eğitim yazılımlarında geliştirici görev aldı. 2010 yılında yeniden başladığı Maltepe Üniversitesi Bilgi İşlem Dairesi'nde görevlerini sürdürmektedir.