Bölüm

12

Dağıtık Veritabanı Yönetim Sistemleri

Öğrenme Hedefleri

Bu bölümü tamamladıktan sonra şunları yapabileceksiniz:

12-1 Dağıtık veritabanı yönetim sistemlerinin (DDBMS) amacını ve işlevini açıklamak

12-2 DDBMS'lerin avantaj ve dezavantajlarını özetleyin

12-3 DDBMS'lerin özelliklerini ve bileşenlerini tanımlama

12-4 Veritabanı uygulamasının farklı veri ve süreç dağıtım seviyelerinden nasıl etkilendiğini açıklayın

12-5 Dağıtık bir veritabanı ortamında işlemlerin nasıl yönetildiğini tanımlama

12-6 Dağıtık veritabanı tasarımının performans, ölçeklenebilirlik ve kullanılabilirliği nasıl dengelediğini açıklayabilme

12-7 Dağıtık bir veri sisteminin uygulanmasındaki ödünleşimleri açıklamak

ÖN İZLEME

Bu bölümde, tek bir veritabanının coğrafi olarak dağınık bir ağ içindeki farklı bilgisayarlarda depolanan birkaç parçaya bölünebileceğini öğreneceksiniz. İşlem ayrıca birkaç farklı ağ sitesi veya düğüm arasında da dağıtılabilir.

Dağıtık veritabanı sistemlerinin büyümesi, iş operasyonlarının artan küreselleşmesi, devasa kurumsal veri setlerinin birikmesi ve dağıtık ağ tabanlı hizmetleri pratik, daha güvenilir ve uygun maliyetli hale getiren teknolojik değişiklikler tarafından teşvik edilmiştir.

Dağıtık veritabanı yönetim sistemi (DDBMS), dağıtık bir veritabanını tek bir mantıksal veritabanı olarak ele alır; bu nedenle, önceki bölümlerde öğrendiğiniz temel tasarım kavramları geçerlidir. Ancak, verilerin farklı siteler arasında dağıtımı bir bilgisayar ağı sistemin karmaşıklığına katkıda bulunur. Örneğin, dağıtık bir veritabanının tasarımı, verilerin konumunu, verilerin parçalara ayrılmasını ve bu parçaların çoğaltılmasını dikkate almalıdır. Dağıtık bir veritabanı sistemi daha sofistike bir DBMS gerektirse de, dağıtık bir veritabanı sisteminin daha karmaşık olması son kullanıcı için şeffaf olmalıdır.

Günümüzün web merkezli ortamında, herhangi bir dağıtık veri sistemi yüksek oranda ölçeklenebilir olmalıdır; başka bir deyişle, talep arttıkça dinamik olarak büyümelidir. Bu dinamik büyümeyi karşılamak için, bazı arzu edilen özelliklere ulaşmak amacıyla ödünleşimler yapılmalıdır.

Veri Dosyaları ve Mevcut Formatlar

	MS Erişim	Oracle	MS SQL	MySQL
Ch12_Text	Evet	Evet	Evet	Evet

Veri Dosyaları cengage.com adresinde mevcuttur

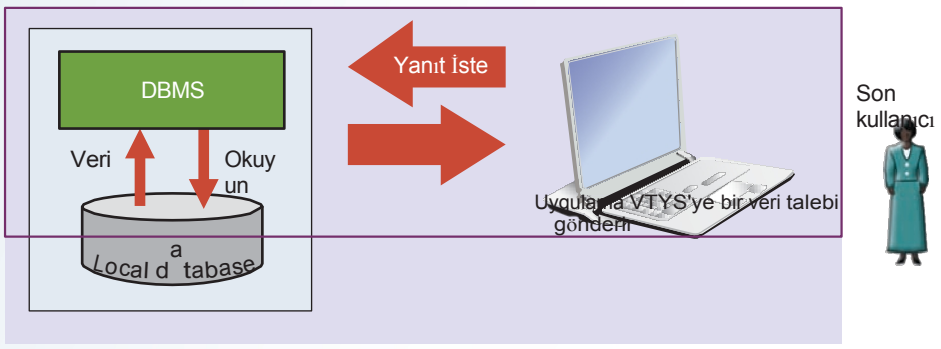
12-1 Dağıtık Veritabanı Yönetim Sistemlerinin Evrimi

dağıtılmış veritabanı yönetim sistemi (DDBMS)
Birkaç farklı veritabanına dağıtılmış bir veritabanını destekleyen bir DBMS Siteler; bir DDBMS, hem verilerin hem de işleme işlevlerinin birkaç arasında dağıtıldığı birbirine bağlı bilgisayar sistemleri üzerinden mantıksal olarak ilişkili verilerin depolanmasını ve işlenmesini yönetir.

Dağıtık veritabanı yönetim sistemi (DDBMS), mantıksal olarak ilişkili verilerin, hem verilerin hem de işlemlerin çeşitli siteler arasında dağıtıldığı birbirine bağlı bilgisayar sistemleri üzerinde depolanmasını ve işlenmesini yönetir. DDBMS'nin DBMS'den nasıl ve neden farklı olduğunu anlamak için DDBMS'nin gelişimine zemin hazırlayan iş ortamındaki değişiklikleri kısaca incelemek faydalı olacaktır.

1970'lerde şirketler, yapılandırılmış bilgi ihtiyaçlarını karşılamak için merkezi veritabanı yönetim sistemleri uyguladılar. Merkezi bir veritabanının kullanılması, kurumsal verilerin tek bir merkezi yerde, genellikle bir ana bilgisayarda depolanmasını gerektiriyordu. Veri erişimi aptal terminaller aracılığıyla sağlanıyordu. Şekil 12.1'de gösterilen merkezi yaklaşım, kurumların yapılandırılmış bilgi ihtiyaçlarını karşılamak için iyi çalıştı, ancak hızlı hareket eden olaylar daha hızlı yanıt süreleri ve bilgiye eşit derecede hızlı erişim gerektirdiğinde yetersiz kaldı. Bilgi talebinden onaya, uzmandan kullanıcıya yavaş ilerleyen süreç, dinamik bir ortamda karar vericilere iyi hizmet etmiyordu. İhtiyaç duyulan şey, yerinde bilgi üretmek için geçici sorgular kullanarak veri tabanlarına hızlı, yapılandırılmamış erişimdi.

Şekil 12.1 Merkezi Veritabanı Yönetim Sistemi



Son yirmi yıl, sistemlerin ve kullandıkları verilerin doğasını etkileyen bir dizi önemli sosyal ve teknolojik değişikliğe yol açmıştır:

- Ticari faaliyetler küreselleşti; bu değişimle birlikte rekabet yan köşedeki dükkandan siber uzaydaki web mağazasına kadar genişledi.
- Müşteri talepleri ve pazar ihtiyaçları, çoğunlukla web tabanlı hizmetlere dayanan talep üzerine işlem tarzını tercih etti.
- Düşük maliyetli, akıllı mobil cihazlarla beslenen hızlı sosyal ve teknolojik değişimler, bunları birbirine bağlayacak karmaşık ve hızlı ağlara olan talebi artırmıştır. Sonuç olarak, kurumlar bilgisayarlı çözümleri için platform olarak gelişmiş ağ teknolojilerini giderek daha fazla benimsemiştir. Bulut tabanlı hizmetlerle ilgili bir tartışma için Bölüm 15, Veritabanı Bağlantısı ve Web Teknolojileri'ne bakınız.
- Veri alanları dijital dünyada daha sık bir araya geliyor. Sonuç olarak, uygulamaların ses, video, müzik ve görüntü gibi birden fazla veri türünü yönetmesi gerekmektedir. Bu tür veriler coğrafi olarak dağıtılma eğilimindedir ve konuma duyarlı mobil cihazlar aracılığıyla farklı konumlardan uzaktan erişilebilir.
- Yeni müşterilere ulaşmanın ve yeni pazarlar açmanın bir yolu olarak sosyal medyanın ortaya çıkışı, büyük miktarlarda dijital verinin depolanması ihtiyacını körükledi ve verilerin yönetilme ve bilgi için çıkarılma biçiminde bir devrim yarattı. İşletmeler, yapılandırılmış ve yapılandırılmamış verilerin geniş depolarının analizi yoluyla iş zekası elde etmenin yeni yollarını arıyor.

Bu faktörler, şirketlerin rekabetçi ve teknolojik baskılara hızla yanıt vermek zorunda olduğu dinamik bir iş ortamı yarattı. Büyük iş birimleri daha yalın, hızlı tepki veren, dağınık operasyonlar oluşturacak şekilde yeniden yapılandırıldıkça, iki veritabanı gereksinimi belirgin hale geldi:

- Hızlı tepki veren karar verme ortamında *hızlı geçici veri erişimi* çok önemli hale geldi.
- Coğrafi olarak dağınık iş birimlerini desteklemek için *dağıtılmış veri erişimine* ihtiyaç vardı.

Son yıllarda bu faktörler daha da yerleşik hale gelmiştir. Ancak, bunların ele alınış biçimi aşağıdaki faktörlerden güçlü bir şekilde etkilenmiştir:

- *İnternetin veri erişimi ve dağıtımı için bir platform olarak giderek daha fazla kabul görmesi*. Web, dağıtılmış veriler için etkin bir depodur.
- *Mobil kablosuz devrim*. Mobil, kablosuz, dijital cihazların yaygın kullanımı akıllı telefonları ve tabletleri içermektedir. Bu cihazlar veri erişimi için yüksek talep yaratmıştır. Verilere coğrafi olarak dağınık konumlardan erişirler ve veri, ses, video, müzik ve resim gibi çoklu formatlarda çeşitli veri alışverişleri gerektirirler. Dağıtık veri erişimi mutlaka dağıtık veri tabanları anlamına de, performans ve hata toleransı gereksinimleri genellikle dağıtık veri tabanlarına benzer veri çoğaltma tekniklerinin kullanılmasına yol açmaktadır.
- *"Hizmet olarak uygulama" kullanan şirketlerin hızla büyümesi.* Bu yeni hizmet türü, uygulama geliştirme, bakım ve operasyonlarını dışarıdan temin etmek isteyen şirketlere uzaktan uygulama sağlamaktadır. Şirket verileri genellikle merkezi sunucularda saklanır ve dağıtılması gerekmez. Tıpkı mobil veri erişiminde olduğu gibi, bu tür bir hizmet tam dağıtık veri işlevselliği gerektirmeyebilir; ancak performans ve hata toleransı gibi diğer faktörler genellikle dağıtık veritabanlarına benzer veri çoğaltma tekniklerinin kullanılmasını gerektirir.

- *Mobil iş zekasına artan odaklanma*. Giderek daha fazla şirket iş planlarında mobil teknolojileri benimsiyor. Şirketler müşterilere yakınlaşmak için sosyal ağları kullandıkça, yerinde karar verme ihtiyacı da artmaktadır. Bir veri ambarı genellikle dağıtık bir veritabanı olmasa da, veri çıkarma ve entegrasyonunu kolaylaştıran veri çoğaltma ve dağıtık sorgular gibi tekniklere dayanır. (Bu konu hakkında daha fazla bilgiyi Bölüm 13, İş Zekası ve Veri Ambarları'nda bulabilirsiniz).
- *Büyük Veri analitiğine vurgu*. Mobil iletişim çağı, pek çok kaynaktan ve pek çok türden veri yığınını ortaya çıkarmıştır. Günümüzün müşterileri, toplulukların harcama alışkanlıkları üzerinde önemli etkiye sahiptir ve kuruluşlar, müşterilere etkili ve verimli bir şekilde ulaşmanın yeni yollarını "keşfetmek" için bu tür verileri toplamanın yollarına yatırım yapmaktadır.

Çevrimiçi İçerik

İnternetin veri erişimi ve dağıtımı üzerindeki etkisi hakkında daha fazla bilgi edinmek için bkz. www.cengage.com adresindeki Ek I, Elektronik Ticarette Veri Tabanları.

Bu noktada, internetin ve mobil devrimin *dağıtık* veritabanı tasarımı ve yönetimi üzerindeki uzun vadeli etkisi henüz hissedilmeye başlanmıştır. Belki de İnternet ve mobil teknolojilerin başarısı, bant genişliğinin daha az sorun yaratan bir darboğaz haline gelmesiyle dağıtık veritabanlarının kullanımını teşvik edecektir. Belki de bant genişliği sorunlarının çözümü, merkezi veritabanı standardını doğrulayacaktır. Her , dağıtık veritabanı kavramları ve bileşenleri, özellikle özel mobil ve konuma duyarlı uygulamalar için gelecekteki veritabanı gelişiminde bir yer bulacaktır.

Dağıtık veritabanı özellikle arzu edilir çünkü merkezi veritabanı yönetimi aşağıdaki gibi sorunlara tabidir:

- Daha uzak mesafelerdeki uzak konumların sayısının artması nedeniyle *performans düşüşü*.
- Büyük merkezi (mainframe) veritabanı sistemlerinin ve fiziksel altyapının bakımı ve işletilmesi ile ilgili *yüksek maliyetler*.
- Merkezi bir siteye bağımlılığın yarattığı *güvenilirlik sorunları* (tek hata noktası sin- dromu) ve veri replikasyonu ihtiyacı.
- Fiziksel alan, sıcaklık koşullandırma ve güç tüketimi gibi tek bir konumun getirdiği fiziksel sınırlarla ilişkili *ölçeklenebilirlik sorunları*.
- Veritabanının dayattığı *organizasyonel katılık*, modern küresel organizasyonların ihtiyaç duyduğu esneklik ve çevikliği desteklemeyebileceği anlamına gelir.

Dinamik iş ortamı ve merkezi veritabanının eksiklikleri, birden fazla farklı kaynaklardan gelen verilere erişmeye dayalı uygulamalara yönelik bir talep doğurmuştur. Böyle bir çoklu kaynak/çoklu konum veritabanı ortamı en iyi şekilde bir DDBMS tarafından yönetilir.

12-2 DDBMS Avantajları ve Dezavantajları

Dağıtık veritabanı yönetim sistemleri geleneksel sistemlere göre çeşitli avantajlar sağlamaktadır. Aynı zamanda bazı sorunlara da tabidirler. Tablo 12.1 bir DDBMS ile ilişkili avantaj ve dezavantajları özetlemektedir.

Dağıtık veritabanları Google ve Amazon gibi pek çok web kuruluşunda başarıyla kullanılıyor, ancak teorik olarak sahip oldukları tam esneklik ve gücü elde etmeleri için daha önlerinde uzun bir yol var.

Bu bölümün geri kalanında dağıtık veritabanının temel bileşenleri ve kavramları incelenmektedir. Dağıtık veritabanı genellikle ilişkisel veritabanı modeline dayandığından, temel kavramları ve bileşenleri açıklamak için ilişkisel terminoloji kullanılmaktadır. En yaygın olarak kullanılan dağıtık veritabanlarından bazıları NoSQL hareketinin bir parçası olsa da (bkz. Bölüm 2, Veri Modelleri), dağıtık verinin temel kavramları ve temelleri hala onlar için geçerlidir.

Tablo 12.1 Dağıtık VTYS Avantajları ve Dezavantajları

Avantajlar	Dezavantajlar
Veriler en çok talep gören yerin yakınında bulunur. Dağıtılmış bir veritabanı sistemindeki veriler, iş gereksinimlerine uyacak şekilde dağıtılır.	***Yönetim ve kontrolün karmaşıklığı.*** Uygulamalar veri konumunu tanımalı ve çeşitli sitelerden gelen verileri bir araya getirebilmelidir. Veritabanı yöneticileri, veri anormallikleri nedeniyle veritabanının bozulmasını önlemek için faaliyetlerini koordine etme becerisine sahip olmalıdır.
Daha hızlı veri erişimi. Son kullanıcılar genellikle verilerin yalnızca en yakın depolanmış alt kümesiyle çalışır.	***Teknolojik zorluklar.*** Veri bütünlüğü, işlem yönetimi, eşzamanlılık kontrolü, güvenlik, yedekleme, kurtarma ve sorgu optimizasyonu gibi konular ele alınmalı ve çözüme kavuşturulmalıdır.
Daha hızlı veri işleme. Dağıtılmış bir veritabanı sistemi, verileri çeşitli sitelerde işleyerek sistemin iş yükünü yayar.	***Güvenlik.*** Veriler birden fazla bulunduğunda güvenlik açıkları olasılığı artar. Veri yönetiminin sorumluluğu çeşitli sahalarda farklı kişiler tarafından paylaşılacaktır.
Büyümeyi kolaylaştırma. Diğer tesislerin faaliyetlerini etkilemeden ağa yeni eklenebilir.	***Standart eksikliği.*** Veritabanı düzeyinde standart iletişim protokolleri yoktur. Örneğin, farklı veritabanı satıcıları, bir DDBMS ortamında veri dağıtımını ve işlemeyi yönetmek için farklı ve genellikle uyumsuz teknikler kullanır.
Geliştirilmiş iletişim. Yerel tesisler daha küçük ve müşterilere daha yakın olduğundan, yerel tesisler departmanlar arasında ve müşteriler ile şirket personeli arasında daha iyi iletişimi teşvik eder.	***Artan depolama ve altyapı gereksinimleri.*** Verilerin farklı yerlerde birden fazla kopyasının olması gerekir, bu da ek depolama alanı gerektirir.
Azaltılmış işletme maliyetleri. Bir ağa düğüm eklemek, bir anabilgisayar sistemini güncellemekten daha uygun maliyetlidir. Geliştirme çalışmaları, düşük maliyetli PC'ler ve dizüstü bilgisayarlarda ana bilgisayarlara göre daha ucuz ve hızlı bir şekilde yapılır.	***Artan eğitim maliyeti.*** Eğitim maliyetleri genellikle dağıtık bir modelde merkezi bir modelde olacağından daha yüksektir, hatta bazen operasyonel ve donanım tasarruflarını dengeleyecek kadar yüksektir.
Kullanıcı dostu arayüz. İstemci cihazlar genellikle kullanımı kolay bir grafik kullanıcı arayüzü (GUI) ile donatılmıştır. GUI, son kullanıcılar için eğitimi ve kullanımı kolaylaştırır.	***Daha yüksek maliyetler.*** Dağıtılmış veritabanlarının çalışması için fiziksel konum, ortam, personel, yazılım ve lisanslama çoğaltılmış altyapı gerekir.
Tek noktadan arıza tehlikesi daha azdır. Bilgisayarlardan biri , iş yükü diğer iş istasyonları tarafından karşılanır. Veriler ayrıca birden fazla siteye dağıtılır.	
İşlemci bağımsızlığı. Son kullanıcı verilerin mevcut herhangi bir kopyasına erişebilir ve son kullanıcının talebi veri konumundaki herhangi bir işlemci tarafından işlenir.	

12-3 Dağıtık İşleme ve Dağıtık Veritabanları

Dağıtık işlemede, bir veritabanının mantıksal işlemi, bir ağ üzerinden bağlı olan iki veya daha fazla fiziksel olarak bağımsız site arasında paylaşılır. Örneğin, veri girişi/çıkışı (I/O), veri seçimi ve veri doğrulaması bir bilgisayarda gerçekleştirilebilir ve bu verilere dayalı bir rapor başka bir bilgisayarda oluşturulabilir.

Şekil 12.2'de temel dağıtık işleme ortamı gösterilmektedir; burada dağıtık bir işleme sistemi veritabanı işleme işlerini bir iletişim ağı üzerinden birbirine bağlı üç site arasında paylaştırmaktadır. Veritabanı yalnızca bir sitede (Miami) bulunmasına rağmen, her site verilere erişebilir ve veritabanını güncelleyebilir. *Veritabanı*, *veritabanı sunucusu* olarak bilinen bir ağ bilgisayarı olan A Bilgisayarında bulunmaktadır.

Öte yandan **dağıtılmış bir veritabanı**, mantıksal olarak ilişkili bir veritabanını fiziksel olarak bağımsız iki veya daha fazla site üzerinde depolar. Siteler bir bilgisayar ağı üzerinden birbirine bağlıdır. Buna karşılık, dağıtılmış işleme sistemi yalnızca tek bir site veritabanı kullanır, ancak işleme işlerini birkaç site arasında paylaştırır. Dağıtık bir veritabanı sisteminde, bir veritabanı, **veritabanı parçaları** olarak bilinen birkaç parçadan oluşur. Veritabanı parçaları farklı sitelerde bulunur ve çeşitli siteler arasında çoğaltılabilir. Her veritabanı parçası kendi yerel veritabanı süreci tarafından yönetilir. Dağıtılmış bir veritabanı ortamının bir örneği Şekil 12.3'te gösterilmektedir.

Şekil 12.3'teki veritabanı farklı bulunan üç veritabanı parçasına (E1, E2 ve E3) bölünmüştür. Bilgisayarlar bir ağ sistemi aracılığıyla birbirine bağlanır. Tamamen dağıtılmış bir

dağıtılmış işleme Bir ağ ile birbirine bağlı iki veya daha fazla site üzerinden bir veritabanının mantıksal olarak paylaşılması.

dağıtılmış veritabanı Fiziksel olarak bağımsız iki veya daha fazla sitede depolanan mantıksal olarak ilişkili bir veritabanı.

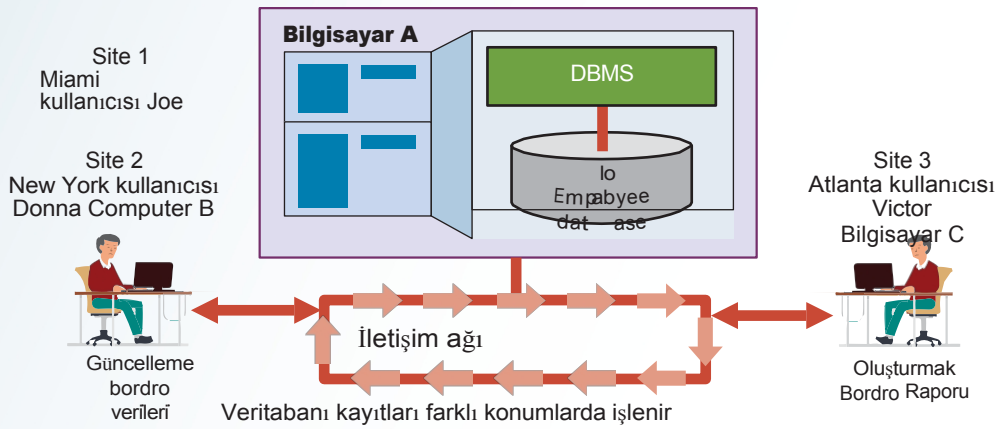
veritabanı parçası Dağıtılmış bir veritabanının alt kümesi. Her ne kadar parçalar bir bilgisayar ağı içinde farklı yerlerde depolanabilir, set tüm parçaların tek bir veritabanı olarak ele alınır. Ayrıca bkz. *yatay parçalanma* ve *dikey parçalanma*.

veritabanına erişmek için Alan, Betty ve Hernando kullanıcılarının her bir veritabanı parçasının adını veya konumunu bilmesine gerek yoktur. Ayrıca, kullanıcılar Miami, New York veya Atlanta dışındaki yerlerde olabilir ve yine de veritabanına tek bir mantıksal birim olarak erişebilirler.

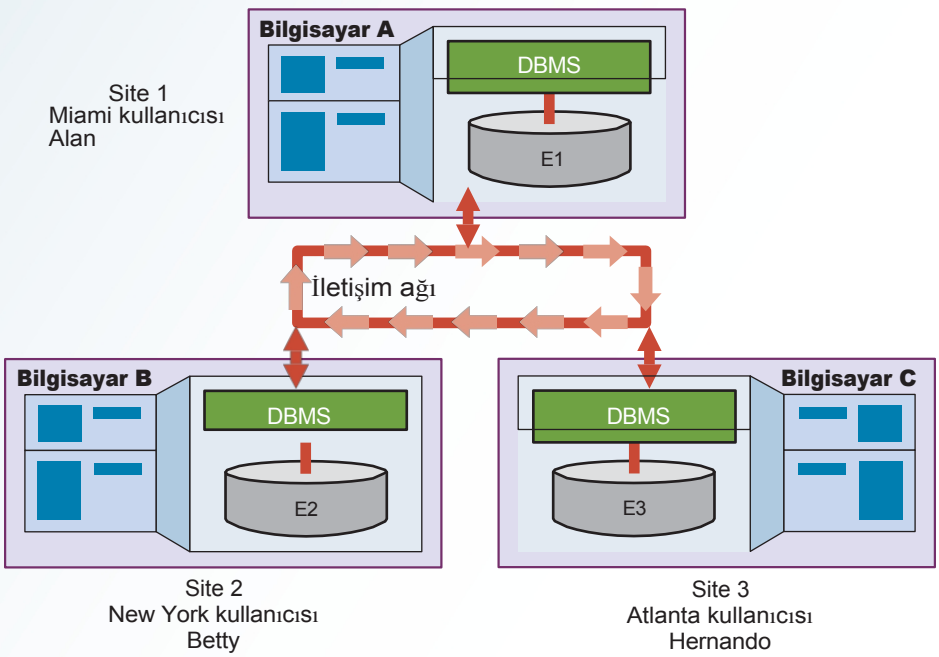
Şekil 12.2 ve 12.3'ü incelerken aşağıdaki noktaları aklınızda tutun:

- Dağıtılmış işleme dağıtılmış bir veritabanı gerektirmez, ancak dağıtılmış bir veritabanı dağıtılmış işleme gerektirir. (Her veritabanı parçası kendi yerel veritabanı süreci tarafından yönetilir).
- Dağıtılmış işleme, tek bir bilgisayarda bulunan tek bir veritabanına dayalı olabilir. Dağıtılmış verilerin yönetiminin gerçekleşmesi için, veritabanı işleme işlevlerinin kopyaları veya bir kısmı tüm veri depolama alanlarına dağıtılmalıdır.
- Hem dağıtılmış işleme hem de dağıtılmış veritabanları, birbirine bağlı bileşenlerden oluşan bir ağ gerektirir.

Şekil 12.2 Dağıtık İşleme Ortamı



Şekil 12.3 Dağıtık Veritabanı Ortamı



12-4 Dağıtık Veritabanı Yönetim Sistemlerinin Özellikleri

Bir DDBMS, hem verilerin hem de işleme işlevlerinin çeşitli siteler arasında dağıtıldığı birbirine bağlı bilgisayar sistemleri üzerinden mantıksal olarak ilişkili verilerin depolanmasını ve işlenmesini yönetir. Bir DBMS'nin dağıtılmış olarak sınıflandırılabilmesi için en azından aşağıdaki işlevlere sahip olması gerekir:

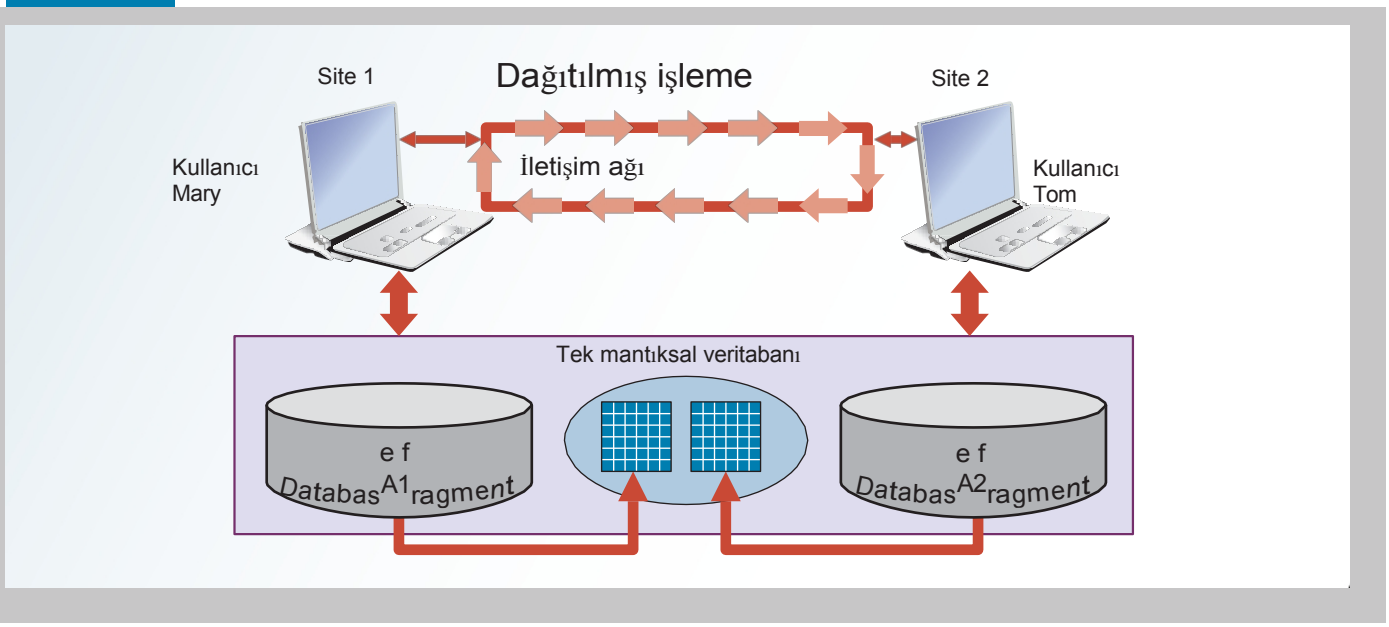
- Son kullanıcı, uygulama programları ve dağıtılmış veritabanı içindeki diğer DBMS'ler ile etkileşim için *uygulama arayüzü*
- Veri taleplerini sözdizimi doğruluğu açısından analiz etmek için *doğrulama*
- Karmaşık talepleri atomik veri talebi bileşenlerine ayrıştırmak için *dönüşüm*
- En iyi erişim stratejisini bulmak için *sorgu optimizasyonu* (sorgu tarafından hangi veritabanı parçalarına erişilmeli ve varsa veri güncellemeleri nasıl senkronize edilmeli?)
- Yerel ve uzak parçaların veri konumunu belirlemek için *haritalama*
- Kalıcı yerel depolama alanından veya bu alana veri okumak veya yazmak için *G/Ç arayüzü*
- Verilerin son kullanıcıya veya bir uygulama programına sunulmak üzere hazırlanması için *biçimlendirme*
- Hem yerel hem de uzak veritabanlarında veri gizliliği sağlamak için *güvenlik*
- Bir arıza durumunda veritabanının kullanılabilirliğini ve kurtarılabilirliğini sağlamak için *yedekleme ve kurtarma*
- Veritabanı yöneticisi için *DB yönetim özellikleri*
- Eşzamanlı veri erişimini yönetmek ve DDBMS'deki veritabanı parçaları arasında veri tutarlılığını sağlamak için eşzamanlılık *kontrolü*
- Verilerin tutarlı bir durumdan diğerine geçmesini sağlamak için *işlem yönetimi*; bu faaliyet yerel ve uzak işlemlerin yanı sıra birden fazla dağıtılmış segmentteki işlemlerin senkronizasyonunu içerir

Tamamen dağıtılmış bir veritabanı yönetim sistemi, merkezi bir DBMS'nin tüm işlevlerini aşağıdaki gibi yerine getirmelidir:

1. Bir uygulamanın veya son kullanıcının talebini alın.
2. Talebi doğrulayın, analiz edin ve ayrıştırın. Talep, aşağıdaki gibi matematiksel ve mantıksal işlemler içerebilir: Bakiyesi 1.000 $'dan büyük olan tüm müşterileri seçin. İstek yalnızca tek bir tablodan veri gerektirebilir veya birkaç tabloya erişim gerektirebilir.
3. Talebin mantıksal-fiziksel veri bileşenlerini eşleştirin.
4. İsteği birkaç disk G/Ç işlemine ayrıştırın.
5. Verileri arayın, bulun, okuyun ve doğrulayın.
6. Veritabanı tutarlılığını, güvenliğini ve bütünlüğünü sağlayın.
7. Varsa, istek tarafından belirtilen koşullar için verileri doğrulayın.
8. Seçilen verileri gerekli formatta sunun.

Buna ek olarak, dağıtık bir DBMS, veri ve işlemenin dağıtımı tarafından dayatılan tüm gerekli işlevleri yerine getirmeli ve bu ek işlevleri son kullanıcıya *şeffaf* bir *şekilde* gerçekleştirmelidir. DDBMS'nin şeffaf veri erişim özellikleri Şekil 12.4'te gösterilmektedir. Şekil 12.4'teki tek mantıksal veritabanı, sırasıyla Site 1 ve 2'de bulunan A1 ve A2 adlı iki veritabanı parçasından oluşur. Mary bu veritabanını yerel bir gibi sorgulayabilir; Tom da öyle. Her iki kullanıcı da yalnızca bir mantıksal veritabanı "görür" ve *parçaların adlarını bilmeleri gerekmez*. Aslında, son kullanıcıların veritabanının aşağıdakilere bölündüğünü bilmelerine bile gerek yoktur *parçalarının nerede olduğunu bilmelerine de gerek yoktur*.

Şekil 12.4 Tamamen Dağıtılmış Bir Veritabanı Yönetim Sistemi



Farklı türdeki dağıtık veritabanı senaryolarını daha iyi anlamak için öncelikle dağıtık veritabanı sisteminin bileşenlerini ele alalım.

12-5 DDBMS Bileşenleri

DDBMS en azından aşağıdaki bileşenleri içermelidir:

- Ağ sistemini oluşturan *bilgisayar iş istasyonları veya uzak cihazlar* (siteler veya düğümler). Dağıtılmış veritabanı sistemi bilgisayar sistemi donanımından bağımsız olmalıdır.
- Her iş istasyonunda veya cihazda bulunan *ağ donanım ve yazılım* bileşenleri. Ağ bileşenleri tüm sitelerin etkileşime girmesini ve veri alışverişinde bulunmasını sağlar. Bileşenlerin (bilgisayarlar, işletim sistemleri, ağ donanımı vb.) farklı satıcılar tarafından tedarik edilmesi muhtemel olduğundan, dağıtılmış veritabanı işlevlerinin birden fazla platformda çalıştırılabilmesini sağlamak en iyisidir.
- Verileri bir düğümden diğerine taşıyan *iletişim ortamı*. DDBMS iletiÇim ortamından bağımsız olmalıdır; yani iletiÇim ortamlarını destekleyebilmelidir.
- **İşlem işlemcisi (TP),** veri talep eden her bilgisayar veya cihazda bulunan yazılım bileşenidir. İşlem işlemcisi uygulamanın uzak ve yerel veri taleplerini alır ve işler. TP aynı zamanda **uygulama işlemcisi (AP)** veya **işlem yöneticisi (TM)** olarak da bilinir.
- **Veri işlemcisi (DP)**, sahada bulunan verileri depolayan ve alan her bilgisayarda veya cihazda bulunan yazılım bileşenidir. DP aynı zamanda **veri yöneticisi (DM)** olarak da bilinir. Bir veri işlemcisi merkezi bir DBMS bile olabilir.

Şekil 12.5 bileşenlerin yerleşimini ve aralarındaki etkileşimi göstermektedir. TP'ler ve DP'ler arasındaki iletişim, DDBMS tarafından kullanılan belirli bir dizi kural veya *protokol* aracılığıyla mümkün kılınmaktadır.

Protokoller, dağıtılmış veritabanı sisteminin nasıl çalışacağını belirler:

- DP'ler ve TP'ler arasında veri ve komutları taşımak için ağ ile arayüz.
- DP'lerden (TP tarafı) alınan tüm verileri senkronize edin ve alınan verileri uygun TP'lere (DP tarafı) yönlendirin.

işlem işlemcisi (TP)
Bir DDBMS'de, her bilgisayarda veri talebinde bulunan yazılım bileşeni. TP, herhangi bir bilgisayardaki verilere erişen yerel bir uygulama tarafından yayınlanan tüm veritabanı isteklerinin yürütülmesinden ve sorumludur.
DP. *İşlem yöneticisi (TM)* veya *uygulama işlemcisi (AP)* olarak da adlandırılır.

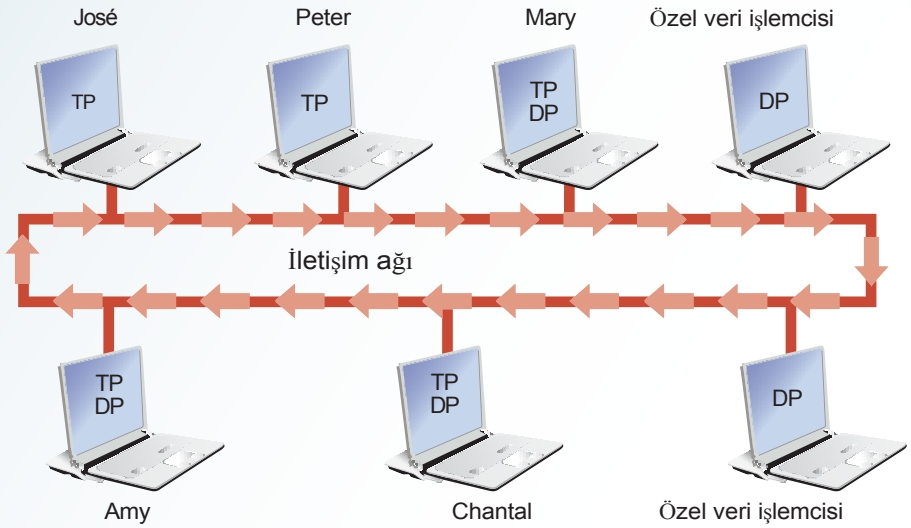
uygulama işlemcisi (AP)
Bkz. *işlem işlemcisi (TP).*

işlem yöneticisi (TM)
Bkz. *işlem işlemcisi (TP).*

veri işlemcisi (DP) Bir DDBMS aracılığıyla verileri depolayan ve alan yerleşik yazılım bileşeni. DP şu şekildedir
Bilgisayardaki yerel verilerin yönetilmesinden ve bu verilere erişimin koordine edilmesinden sorumludur. *Veri yöneticisi (DM)* olarak da bilinir.

veri yöneticisi (DM)
Bkz. *veri işlemcisi (DP).*

Şekil 12.5 Dağıtık Veritabanı Sistemi Bileşenleri



Not: Her TP herhangi bir DP'deki verilere erişebilir ve her DP herhangi bir TP'den gelen tüm yerel veri taleplerini ele alır.

- Dağıtılmış bir sistemde ortak veritabanı işlevlerini sağlamak. Bu işlevler arasında veri güvenliği, işlem yönetimi ve eşzamanlılık kontrolü, veri bölümleme ve senkronizasyon ile veri yedekleme ve kurtarma yer alır.

DP'ler ve TP'ler sisteme, çalışmasını etkilemeden şeffaf bir şekilde eklenmelidir. Bir TP ve bir DP aynı bilgisayarda bulunabilir ve son kullanıcının hem yerel hem de uzak verilere şeffaf bir şekilde erişmesine olanak tanır. Teorik olarak bir DP, ağdaki diğer bağımsız DBMS'lerden uzaktan erişimi desteklemek için uygun arayüzlere sahip bağımsız bir merkezi DBMS olabilir.

12-6 Veri ve Süreç Dağıtım Düzeyleri

Mevcut veritabanı sistemleri, işlem dağıtımı ve veri dağıtımının nasıl desteklendiğine göre sınıflandırılabilir. Örneğin, bir DBMS verileri tek bir sitede (merkezi bir DB kullanarak) veya birden fazla sitede (dağıtılmış bir DB kullanarak) depolayabilir ve bir veya daha fazla sitede veri işlemeyi destekleyebilir. Tablo 12.2, veritabanı sistemlerini veri ve süreç dağılımına göre sınıflandırmak için basit bir matris kullanmaktadır. Bu süreç türleri ilerleyen bölümlerde ele alınmaktadır.

Tablo 12.2 Veritabanı Sistemleri: Veri ve Süreç Dağılımı Düzeyleri

Avantajlar	Tek Yerden Veri	Çoklu-Site Verileri
Tek sahalı süreç	Ev sahibi DBMS	Uygulanamaz (Birden fazla süreç gerektirir)
Çoklu tesis süreci	Dosya sunucusu İstemci/sunucu DBMS (LAN DBMS)	Tamamen dağıtılmış İstemci/sunucu DDBMS

Çevrimiçi İçerik
İstemci/sunucu hakkında daha fazla bilgi için mimari, bkz. Ek F, İstemci/Sunucu Sistemleri, www.cengage.com adresinde mevcuttur.

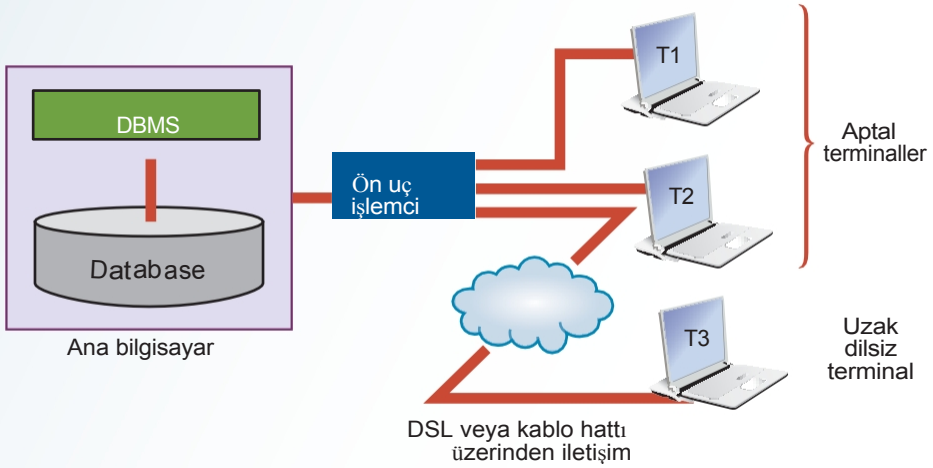
12-6a Tek Yerde İşleme, Tek Yerde Veri

Tek site işleme, tek site veri (SPSD) senaryosunda, tüm işleme tek bir ana bilgisayarda yapılır ve tüm veriler ana bilgisayarın yerel disk sisteminde saklanır. İşlem, sistemin son kullanıcı tarafında yapılamaz. Böyle bir senaryo çoğu ana çerçeve ve orta seviye UNIX/Linux sunucu DBMS'leri için tipiktir. DBMS ana bilgisayarda bulunur ve

tek site işleme, tek site veri (SPSD) Tüm işlemlerin bir ana bilgisayarda yapıldığı ve tüm verilerin ana yerel diskinde depolandığı bir senaryo.

kendisine bağlı terminaller tarafından erişilir (bkz. Şekil 12.6). Bu senaryo aynı zamanda ilk nesil tek kullanıcılı mikrobilgisayar veritabanlarının da tipik bir örneğidir.

Şekil 12.6 Tek Yerde İşleme, Tek Yerde Veri (Merkezi)



Şekil 12.6'yı örnek olarak kullanarak, TP ve DP'nin işlevlerinin ana bilgisayardaki DBMS içine gömülü olduğunu görebilirsiniz. DBMS genellikle zaman paylaşımlı, çok görevli bir işletim sistemi altında çalışır, bu da tek bir DP'ye erişen bir ana birkaç işlemin aynı anda çalışmasına olanak tanır. Tüm veri depolama ve veri işleme tek bir ana bilgisayar tarafından gerçekleştirilir.

çoklu saha işleme, tek saha verileri (MPSD)
Tek bir veri havuzunu paylaşan farklı bilgisayarlarda birden fazla işlemin çalıştığı bir senaryo.

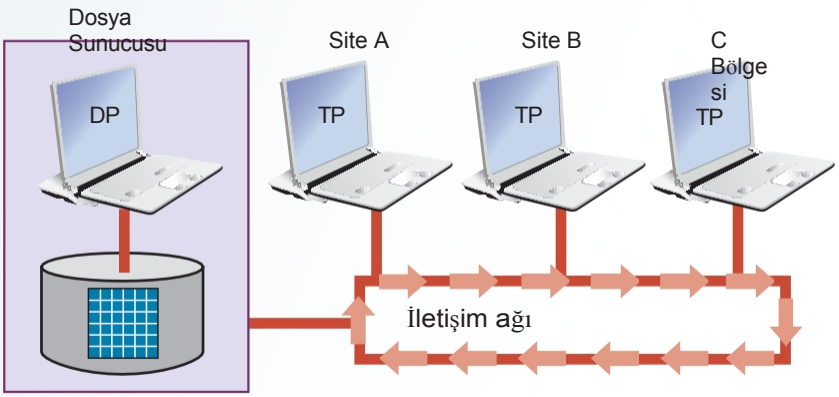
12-6 b Çoklu-Site İşleme, Tek-Site Veri

Çoklu site işleme, tek site veri (MPSD) senaryosunda, tek bir veri havuzunu paylaşan farklı bilgisayarlarda birden fazla işlem çalışır. Tipik olarak, MPSD senaryosu bir ağ üzerinden erişilen geleneksel uygulamaları çalıştıran bir ağ dosya sunucusu gerektirir. Kişisel bilgisayar ağı altında çalışan birçok çok kullanıcılı muhasebe uygulaması böyle bir tanıma uymaktadır (bkz. Şekil 12.7).

Şekil 12.7'yi incelerken şuna dikkat edin:

- Her iş istasyonundaki TP, tüm ağ veri isteklerini dosya sunucusuna yönlendirmek için yalnızca bir yönlendirici görevi görür.

Şekil 12.7 Çoklu-Site İşleme, Tek-Site Veri



- Son kullanıcı dosya sunucusunu sadece başka bir sabit disk olarak görür. Dosya sunucusunun bilgisayarı tarafından yalnızca veri depolama giriş/çıkışı (G/Ç) işlendiğinden, MPSD dağıtılmış işleme için sınırlı yetenekler sunar.
- Son kullanıcı uzaktaki verilere erişmek için dosya sunucusuna doğrudan başvurmalıdır. Tüm kayıt ve dosya kilitleme faaliyetleri son kullanıcı konumunda gerçekleştirilir.
- Tüm veri seçimi, arama ve güncelleme işlevleri iş istasyonunda gerçekleşir, bu nedenle tüm dosyaların iş istasyonunda işlenmek üzere ağ üzerinden seyahat etmesi gerekir. Böyle bir gereklilik ağ trafiğini artırır, yanıt süresini yavaşlatır ve iletişim maliyetlerini artırır.

Son koşulun verimsizliği kolayca gösterilebilir. Örneğin, dosya sunucusu bilgisayarının 100.000 veri satırı içeren bir MÜŞTERİ tablosunu sakladığını ve bunlardan 50'sinin bakiyesinin 1.000 $'dan fazla olduğunu varsayalım. Site A'nın aşağıdaki SQL sorgusunu yayınladığını varsayalım:

```
SEÇİNİZ      *
FROM         MÜŞTERİ
NEREDE       CUS_BALANCE> 1000;
```

Tüm 100.000 MÜŞTERİ satırı, Site A'da değerlendirilmek üzere ağ üzerinden seyahat etmelidir. Çoklu site işleme, tek site veri yaklaşımının bir varyasyonu istemci/sunucu mimarisi olarak bilinir. **İstemci/sunucu mimarisi,** *tüm veritabanı işlemlerinin sunucu sitesinde yapılması ve böylece ağ trafiğinin azaltılması dışında* ağ dosya sunucusuna benzer. Hem ağ dosya sunucusu hem de istemci/sunucu sistemleri çoklu site iGlemi gerçekleGtirse de, istemci/sunucu sisteminin iGlemi dağıtılmıGtır. Ağ dosya sunucusu yaklaşımının veritabanının tek bir sitede bulunmasını gerektirdiğini unutmayın. Buna karşın, istemci/sunucu mimarisi birden fazla sahadaki verileri destekleyebilir.

istemci/sunucu mimarisi
İstemciler, sunucular ve ara yazılımlardan oluşan bir donanım ve yazılım sistemi. Bir kaynak kullanıcısı (istemci) ve bir kaynak sağlayıcısı (sunucu) içerir.

çoklu saha işleme, çoklu saha verileri (MPMD)
Birden fazla veri işlemcisini ve birden fazla işlem işlemcilerini destekleyen tamamen dağıtılmış bir veritabanı yönetim sistemini tanımlayan bir senaryo.

homojen DDBMS Bir ağ üzerinden yalnızca bir tür merkezi veritabanı yönetim sistemini entegre eden bir sistem.

Heterojen DDBMS Farklı türdeki merkezi veritabanı yönetim sistemlerini bir ağ üzerinden entegre eden bir sistem.

tamamen heterojen dağıtık veritabanı sistemi (tamamen heterojen DDBMS) Bir ağ üzerinden farklı türde veritabanı yönetim sistemlerini (hiyerarşik, ağ ve ilişkisel) entegre eden bir sistem. Farklı bilgisayar sistemleri altında çalışan farklı veri modellerini bile destekleyebilen farklı veritabanı yönetim sistemlerini destekler.

12-6c Çoklu-Site İşleme, Çoklu-Site Verileri

Çoklu site işleme, çoklu site verisi (MPMD) senaryosu**,** birden fazla sitede birden fazla veri işlemcisi ve işlem işlemcisi desteği ile tamamen dağıtılmış bir DDBMS'yi tanımlar. Çeşitli veri tabanı türleri için destek seviyesine bağlı olarak, DDBMS'ler ya homojen ya da heterojen olarak sınıflandırılır.

Homojen DDBMS'ler bir ağ üzerinden aynı DBMS'nin birden fazla örneğini entegre eder - örneğin, farklı platformlarda çalışan birden fazla Oracle 21c örneği. Buna karşılık, **heterojen DDBMS**'ler bir ağ üzerinden farklı DBMS türlerini entegre eder, ancak hepsi aynı veri modelini destekler. Örneğin, Tablo 12.3 bir DDBMS içinde entegre edilebilecek çeşitli ilişkisel veritabanı sistemlerini listelemektedir. **Tamamen heterojen** bir **DDBMS**, her biri farklı bir veri modelini destekleyen ve farklı bilgisayar sistemleri altında çalışan farklı DBMS'leri destekleyecektir.

Tablo 12.3 Veritabanı Sistemleri: Veri ve Süreç Dağıtım Düzeyleri

Platform	DBMS	İşletim Sistemi	Ağ İletişim Protokolü
IBM z15	DB2	z/OS	APPC LU 6.2
IBM AS/400	SQL/400	OS/400	3270
RISC bilgisayar	Informix	UNIX	TCP/IP
Intel Xeon CPU	Oracle	Windows Sunucu	TCP/IP

Dağıtık veritabanı uygulamaları, bir DBMS'nin üstünde bir soyutlama katmanı olarak daha iyi anlaşılmaktadır. Bu soyutlama katmanı, doğrudan veri bağlantıları, çoğaltma, gelişmiş veri tabanı özellikleri ve dağıtılmış veri tabanı özellikleri için destek sağlayan ek işlevsellik sağlar.

veri parçalanması, senkronizasyon ve entegrasyon. Aslında, çoğu veritabanı satıcısı artan seviyelerde veri parçalanması, replikasyonu ve entegrasyonu . Bu nedenle, dağıtık veritabanları için destek, homojenden tamamen heterojen dağıtık veri yönetimine giden sürekli bir spektrum olarak daha iyi görülebilir. Sonuç olarak, bu spektrumun herhangi bir noktasında, bir DDBMS belirli kısıtlamalara tabidir. Örneğin:

- Uzaktan erişim salt okunur olarak sağlanır ve yazma ayrıcalıklarını desteklemez.
- Tek bir işlemde erişilebilecek uzak tablo sayısına kısıtlamalar getirilir.
- Erişilebilecek farklı veritabanlarının sayısına kısıtlamalar getirilmiştir.
- Erişilebilecek veritabanı modeline kısıtlamalar getirilmiştir. Böylece, ilişkisel veritabanlarına erişim sağlanabilir ancak ağ veya hiyerarşik veritabanlarına erişim sağlanamaz.

Yukarıdaki kısıtlamalar listesi hiçbir şekilde kapsamlı değildir. DDBMS teknolojisi hızla değişmeye devam etmekte ve sık sık yeni özellikler eklenmektedir. Verilerin birden fazla sitede yönetilmesi, ele alınması ve anlaşılması gereken bir dizi soruna yol açar. Bir sonraki bölümde dağıtık veritabanı yönetim sistemlerinin bazı temel özellikleri incelenmektedir.

dağıtım şeffaflığı
Dağıtık bir veritabanının son kullanıcıya tek bir mantıksal gibi görünmesini sağlayan bir DDBMS özelliği.

işlem şeffaflığı
Veritabanı sağlayan bir DDBMS özelliği işlemlerin dağıtılmış veritabanının bütünlüğünü ve tutarlılığını koruyacağını ve bir işlemin yalnızca ilgili tüm veritabanı siteleri işlemin kendilerine düşen kısmını tamamladığında tamamlanacağını belirtir.

hata şeffaflığı Bir ağ düğümü başarısız olsa bile bir DDBMS'nin sürekli çalışmasına izin veren bir özellik.

performans şeffaflığı
Bir sistemin merkezi bir DBMS çalışmasını sağlayan bir DDBMS özelliği.

heterojenlik şeffaflık
Bir sistemin birkaç merkezi DBMS'yi tek bir mantıksal DDBMS'ye entegre etmesini sağlayan bir özellik.

12-7 Dağıtılmış Veritabanı Şeffaflık Özellikleri

Dağıtık bir veritabanı sistemi, sistemin tüm karmaşıklıklarını son kullanıcıya gizleyen bazı arzu edilen şeffaflık özellikleri sağlamalıdır. Başka bir deyişle, son kullanıcı merkezi bir DDBMS ile çalışıyormuş hissine sahip olmalıdır. Bu nedenle, arzu edilen asgari DDBMS şeffaflık özellikleri şunlardır:

- **Dağıtım şeffaflığı**, dağıtılmış bir veritabanının tek bir mantıksal veritabanı olarak ele alınmasını sağlar. Bir DDBMS dağıtım şeffaflığı sergiliyorsa, kullanıcının bilmesi gerekmez:
  - Veriler bölümlenir; yani tablonun satırları ve sütunları dikey veya yatay olarak bölünür ve birden fazla site arasında depolanır.
  - Veriler coğrafi olarak birden fazla site arasında dağılmıştır.
  - Veriler birden fazla site arasında çoğaltılır.
- **İşlem şeffaflığı,** bir işlemin birden fazla ağ sitesindeki verileri güncellemesine olanak tanır. İşlem şeffaflığı, işlemin ya tamamen tamamlanmasını ya da iptal edilmesini sağlar, böylece veritabanı bütünlüğü korunur.
- **Arıza şeffaflığı**, bir düğüm veya ağ arızası durumunda sistemin çalışmaya devam etmesini sağlar. Arıza nedeniyle kaybedilen işlevler başka bir ağ düğümü tarafından alınacaktır. Bu, özellikle işlerinde güveni sürdürmek için omurga olarak web varlığına bağlı olan kuruluşlarda çok önemli bir özelliktir.
- **Performans şeffaflığı**, sistemin merkezi bir DBMS gibi çalışmasını sağlar. Sistem, bir ağ üzerinde kullanılması veya ağın platform farklılıkları nedeniyle herhangi bir performans düşüşüne maruz kalmayacaktır. Performans şeffaflığı ayrıca sistemin uzaktaki verilere erişmek için en uygun maliyetli yolu bulmasını sağlar. Sistem şeffaf bir şekilde "ölçeklendirilebilmeli" veya sistemin genel performansını etkilemeden daha fazla işlem veya veri işleme düğümü ekleyerek performans kapasitesini artırabilmelidir.
- **Heterojenlik şeffaflığı**, birkaç farklı yerel DBMS'nin (ilişkisel, ağ ve hiyerarşik) ortak veya global bir şema altında entegrasyonuna izin verir. DDBMS, veri taleplerini global şemadan yerel DBMS şemasına çevirmekten sorumludur.

Aşağıdaki bölümlerde bu şeffaflık özelliklerinin her biri daha ayrıntılı olarak ele alınmaktadır.

12-8 Dağıtım Şeffaflığı

parçalanma şeffaflık
Bir sistemin dağıtılmış bir veritabanını bir veritabanı olarak ele almasına izin veren bir DDBMS özelliği
İki veya daha fazla parçaya bölünmüş olsa bile tek bir veritabanı.

konum şeffaflığı
Veritabanı kullanıcının yalnızca veritabanı parçalarının adını bilmesini gerektirdiği bir DDBMS özelliği.
(Parça konumlarının bilinmesi gerekmez.)

Dağıtım şeffaflığı, fiziksel olarak dağınık bir veritabanının merkezi bir gibi yönetilmesini sağlar. DDBMS tarafından desteklenen Çeffaflık düzeyi sistemden sisteme değiÇir. Üç dağıtım şeffaflığı seviyesi tanınmaktadır:

- **Parçalanma şeffaflığı**, dağıtım şeffaflığının en üst düzeyidir. Son kullanıcı ya da programcının bir veritabanının bölümlere ayrıldığını bilmesi gerekmez. Bu nedenle, veri erişiminden önce ne parça adları ne de parça konumları belirtilir.
- **Konum şeffaflığı**, son kullanıcı veya programcının veritabanı parça adlarını belirtmesi gerektiği ancak bu parçaların nerede bulunduğunu belirtmesi gerekmediği durumlarda mevcuttur.
- **Yerel eşleme şeffaflığı**, son kullanıcı veya programcının hem parça adlarını hem de konumlarını belirtmesi gerektiğinde ortaya çıkar.

Şeffaflık özellikleri Tablo 12.4'te özetlenmiştir.

Tablo 12.4 Şeffaflık Özelliklerinin Özeti

SQL Deyimi Gerektiriyorsa:			
Parça Adı?	**Yer Adı?**	**Daha sonra VTYS şunları destekler**	**Dağıtım Şeffaflık Düzeyi**
Evet	Evet	Yerel haritalama şeffaflığı	Düşük
Evet	Hayır	Konum şeffaflığı	Orta
Hayır	Hayır	Parçalanma şeffaflığı	Yüksek

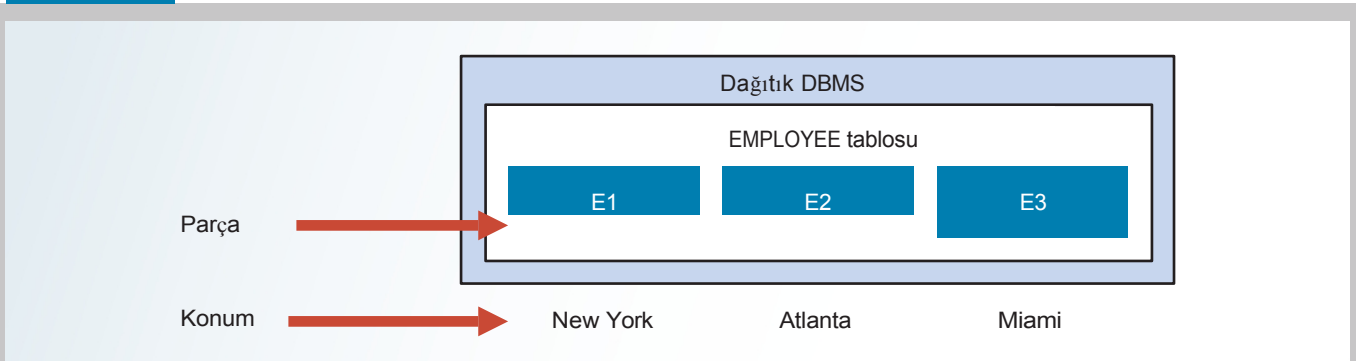
yerel haritalama şeffaflığı
Veritabanı erişiminin kullanıcının aşağıdakileri bilmesini gerektirdiği bir DDBMS özelliği
parçaların hem adı hem de yeri.

Not

Tablo 12.4'ü incelerken, parça adının "Hayır" ve konum adının "Evet" olduğu bir duruma atıfta bulunulmadığına dikkat edin. Bunun nedeni basittir: mevcut bir parçaya referans vermeyen bir konum adına sahip olamazsınız. Bir parça adı belirtmeniz gerekmiyorsa, konumu açıkça önemsizdir.

Çeşitli şeffaflık düzeylerinin kullanımını göstermek için, EMP_NAME, EMP_DOB, EMP_ADDRESS, EMP_DEPARTMENT ve EMP_SALARY niteliklerini içeren bir EMPLOYEE tablonuz olduğunu varsayalım. EMPLOYEE verileri üç farklı konuma dağıtılmıştır: New York, Atlanta ve Miami. Tablo konuma göre bölünmüştür; yani, New York çalışan verileri E1 parçasında, Atlanta çalışan verileri E2 parçasında ve Miami çalışan verileri E3 parçasında saklanır (bkz. Şekil 12.8).

Şekil 12.8 Parça Konumları



benzersiz parça
Bir DDBMS'de, hangi parçada bulunduğuna bakılmaksızın her satırın benzersiz olduğu bir durum.

Şimdi son kullanıcının 1 Ocak 1979 tarihinden önce doğan tüm çalışanları listelemek istediğini varsayalım. Şeffaflık konularına odaklanmak için, EMPLOYEE tablosunun parçalara ayrıldığını ve her bir parçanın benzersiz olduğunu da varsayalım. **Benzersiz parça** koşulu, içinde bulunduğu parçadan bağımsız olarak her satırın benzersiz olduğunu gösterir. Son olarak, veritabanının hiçbir bölümünün ağ üzerindeki başka bir sitede çoğaltılmadığını varsayın.

Dağıtım şeffaflığı desteğinin düzeyine bağlı olarak, üç sorgu durumunu inceleyebilirsiniz.

Vaka 1: Veritabanı Parçalanma Şeffaflığını Destekler

Sorgu, dağıtılmamış bir veritabanı sorgu formatına uygundur; yani, parça adlarını veya konumlarını belirtmez. Sorgu şu şekildedir:

```
SEÇİNİZ     *
FROM        ÇALIŞAN
NEREDE      EMP_DOB< '1979-01-01';
```

Örnek 2: Veritabanı Konum Şeffaflığını Destekliyor

Parça adları sorguda belirtilmelidir, ancak parçanın konumu belirtilmez. Sorgu şu şekildedir:

```
SEÇİNİZ     *
FROM        E1
NEREDE      EMP_DOB< '1979-01-01'
BİRLİK
SEÇİNİZ     *
FROM        E2
NEREDE      EMP_DOB< '1979-01-01'
BİRLİK
SEÇİNİZ     *
FROM        E3
NEREDE      EMP_DOB< '1979-01-01';
```

Örnek 3: Veritabanı Yerel Haritalama Şeffaflığını Destekler

Hem parça adı hem de konumu sorguda belirtilmelidir. Pseudo-SQL kullanma:

```
SEÇİNİZ     *
FROM        El NODE NY
NEREDE      EMP_DOB< '1979-01-01'
UNION
SEÇİNİZ     *
FROM        E2 NODE ATL
NEREDE      EMP_DOB< '1979-01-01'
UNION
SEÇİNİZ     *
FROM        E3 NODE MIA
NEREDE      EMP_DOB< '1979-01-01'
```

Not

NODE, veritabanı parçasının konumunu gösterir. NODE örnekleme amacıyla kullanılır ve standart SQL sözdiziminin bir parçası değildir.

Önceki sorgu biçimlerini incelediğinizde, dağıtım şeffaflığının son kullanıcıların ve programcıların veritabanıyla etkileşim biçimini nasıl etkilediğini görebilirsiniz.

Dağıtım şeffaflığı, dağıtılmış bir **veri sözlüğü (DDD)** veya **dağıtılmış bir veri kataloğu (DDC)** tarafından desteklenir. DDC, veritabanı yöneticisi tarafından görüldüğü şekliyle tüm veritabanının açıklamasını içerir. **Dağıtılmış global şema** olarak bilinen veritabanı açıklaması, kullanıcı isteklerini farklı DP'ler tarafından işlenecek alt sorgulara (uzak istekler) çevirmek için yerel TP'ler tarafından kullanılan ortak veritabanı şemasıdır. DDC'nin kendisi dağıtıktır ve ağ düğümlerinde çoğaltılır. Bu nedenle, DDC tüm sitelerde güncellenerek tutarlılığı korumalıdır.

Mevcut DDBMS uygulamalarından bazılarının şeffaflık desteği düzeyine sınırlamalar getirdiğini unutmayın. Örneğin, bir veritabanını dağıtabilirsiniz, ancak bir tabloyu birden fazla siteye dağıtamazsınız. Böyle bir durum DDBMS'nin konum saydamlığını desteklediğini ancak parçalanma saydamlığını desteklemediğini gösterir.

dağıtılmış veri sözlüğü (DDD)
Bkz. *dağıtılmış veri kataloğu*.

dağıtılmış veri kataloğu (DDC)
Dağıtılmış bir veritabanının açıklamasını (parça adları ve konumları) içeren bir veri sözlüğü.

dağıtılmış küresel şema
Tarafından görüldüğü gibi dağıtılmış bir veritabanının veritabanı şema açıklaması veritabanı yöneticisi.

12-9 İşlem Şeffaflığı

İşlem şeffaflığı, veritabanı işlemlerinin dağıtılmış veritabanının bütünlüğünü ve tutarlılığını korumasını sağlayan bir DDBMS özelliğidir. Bir DDBMS veritabanı işleminin bir ağa bağlı birçok farklı bilgisayarda depolanan verileri güncelleyebileceğini unutmayın. İşlem şeffaflığı, işlemin yalnızca işleme dahil olan tüm veritabanı siteleri işlemin kendilerine düşen kısmını tamamladığında tamamlanmasını sağlar.

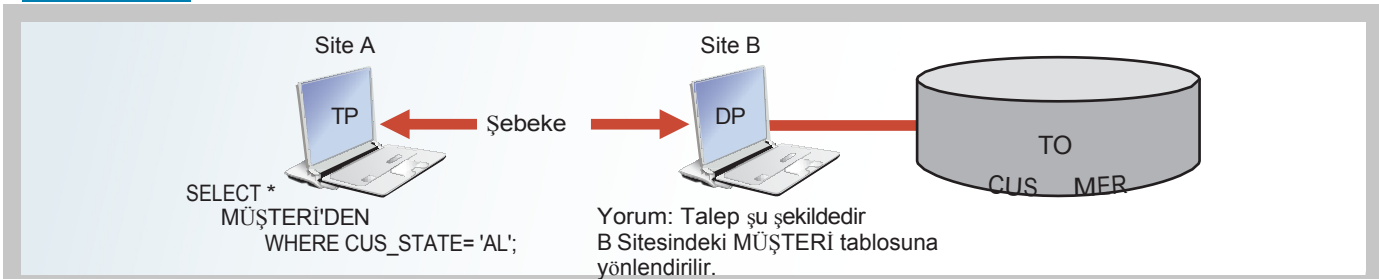
Dağıtılmış veritabanı sistemleri, işlemleri yönetmek ve veritabanının tutarlılığını ve bütünlüğünü sağlamak için karmaşık mekanizmalar gerektirir. İşlemlerin nasıl yönetildiğini anlamak için uzak istekler, uzak işlemler, dağıtılmış işlemler ve dağıtılmış isteklerle ilgili temel kavramları bilmeniz gerekir.

12-9a Dağıtılmış Talepler ve Dağıtılmış İşlemler[1]

Bir işlem dağıtılmış olsun ya da olmasın, bir ya da daha fazla veritabanı isteği tarafından oluşturulur. Dağıtık olmayan bir işlem ile dağıtık bir işlem arasındaki temel fark, dağıtık işlemin bir ağ üzerindeki birkaç farklı uzak siteden veri güncelleyebilmesi veya talep edebilmesidir. Dağıtılmış işlemleri daha iyi anlamak için, BEGIN WORK ve COMMIT WORK işlem biçimini kullanarak uzak ve dağıtılmış işlemler arasındaki farkı öğrenerek başlayın. Veri konumunu belirtmek zorunda kalmamak için konum şeffaflığının var olduğunu varsayın. Şekil 12.9'da gösterilen bir **uzak istek**, tek bir SQL deyiminin tek bir uzak veritabanı işlemcisi tarafından işlenecek verilere erişmesini sağlar. Başka bir deyişle, SQL deyimi (veya isteği) yalnızca bir uzak sitedeki verilere başvurabilir.

uzaktan talep
Tek bir SQL deyimine izin veren bir DDBMS özelliği tek bir uzak DP'deki verilere erişmek için.

Şekil 12.9 Uzak Bir İstek

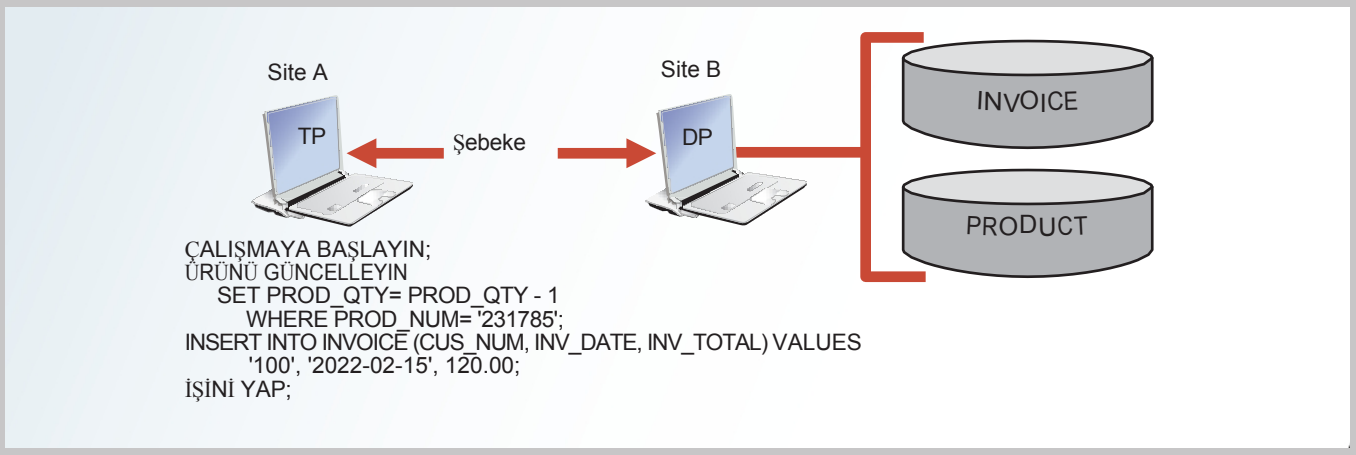


Benzer şekilde, birkaç istekten oluşan bir uzak **işlem**, tek bir uzak sitedeki verilere erişir. Bir uzak işlem Şekil 12.10'da gösterilmiştir.

uzak işlem
Bir işlemin (birkaç istekten oluşan) tek bir uzak DP'deki verilere erişmesini sağlayan bir DDBMS özelliği.

[1]Dağıtılmış isteklerin ve işlemlerin ayrıntıları ilk olarak David McGoveran ve Colin White tarafından açıklanmıştır, "Clarifying client/server," *DBMS 3*(12), Kasım 1990, s. 78-89.

Şekil 12.10 Uzak Bir İşlem



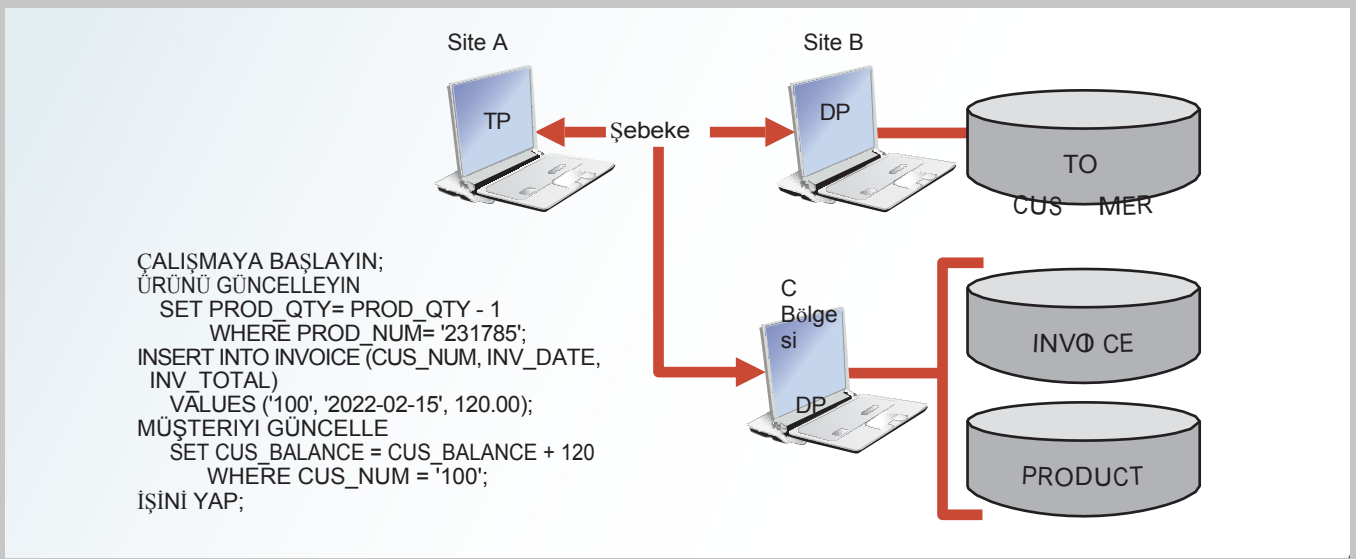
Şekil 12.10'u incelerken aşağıdaki uzak işlem özelliklerine dikkat edin:

- İşlem, PRODUCT ve INVOICE tablolarını günceller (Site B'de bulunur).
- Uzak işlem uzak Site B'ye gönderilir ve orada yürütülür.
- İşlem yalnızca bir uzak DP'ye referans verebilir.
- Her SQL deyimi (veya isteği) bir seferde yalnızca bir (aynı) uzak DP'ye başvurabilir ve tüm işlem yalnızca bir uzak DP'ye başvurabilir ve bu DP'de yürütülebilir.

dağıtılmış **işlem** Dağıtılmış bir ver birkaç uzak veri işlemcisindeki (DP) verilere erişen bir veritabanı işlemi.

Dağıtılmış bir **işlem** birkaç farklı yerel veya uzak DP sitesine başvurabilir. Her bir istek yalnızca bir yerel veya uzak DP sitesine başvurabilse de, her bir istek farklı bir başvurabildiği için işlem bir bütün olarak birden fazla DP sitesine başvurabilir. Dağıtılmış işlem süreci Şekil 12.11'de gösterilmektedir.

Şekil 12.11 Dağıtılmış Bir İşlem



Şekil 12.11'de aşağıdaki özelliklere dikkat edin:

- İşlem, B ve C olmak üzere iki uzak siteye referans vermektedir.
- İlk iki istek, UPDATE PRODUCT ve INSERT INTO INVOICE, uzak Site C'deki DP tarafından işlenir ve son istek (UPDATE CUSTOMER) uzak Site B'deki DP tarafından işlenir.
- Her istek aynı anda yalnızca bir uzak siteye erişebilir.

Üçüncü özellik sorun yaratabilir. Örneğin, PRODUCT tablosunun sırasıyla B ve C Sitelerinde bulunan PRODl ve PROD2 olmak üzere iki parçaya bölündüğünü varsayalım. Bu senaryo göz önüne alındığında, önceki dağıtılmış işlem yürütülemez çünkü aşağıdaki istek birden fazla uzak siteden verilere erişemez:

```
SEÇİNİZ      *
FROM         ÜRÜN
NEREDE       PROD_NUM 5 '231785';
```

Bu nedenle, VTYS dağıtılmış bir destekleyebilmelidir.

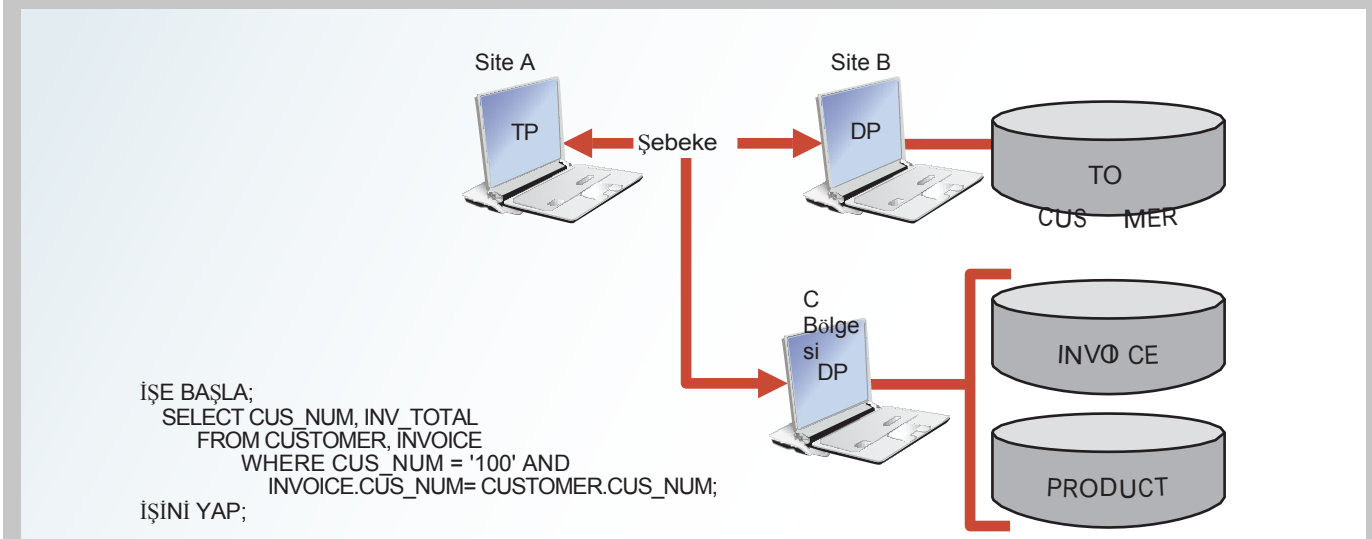
Dağıtılmış bir **istek,** tek bir SQL deyiminin birkaç farklı yerel veya uzak DP sitesinde bulunan verilere başvurmasını sağlar. Her istek (SQL deyimi) birden fazla yerel veya uzak DP sitesindeki verilere erişebildiğinden, bir işlem birden fazla siteye erişebilir. Dağıtılmış bir isteği yürütme yeteneği tamamen dağıtılmış veritabanı işleme sağlar çünkü şunları yapabilirsiniz:

dağıtılmış talep
Tek bir SQL deyiminin dağıtılmış bir veritabanındaki birkaç uzak veri işlemcisindeki (DP) verilere erişmesini sağlayan bir veritabanı isteği.

- Bir veritabanı tablosunu birkaç parçaya böler.
- Bu parçalardan birine veya daha fazlasına yalnızca tek bir istekle başvurabilirsiniz. Başka bir deyişle, parçalanma şeffaflığı vardır.

Verilerin konumu ve bölümü son kullanıcı için şeffaf olmalıdır. Şekil 12.12'de dağıtılmış bir istek gösterilmektedir. Şekli incelerken, işlemin CUSTOMER ve INVOICE adlı iki tabloya başvurmak için tek bir SELECT deyimi kullandığına dikkat edin. Bu iki tablo B ve C olmak üzere iki farklı sitede yer almaktadır.

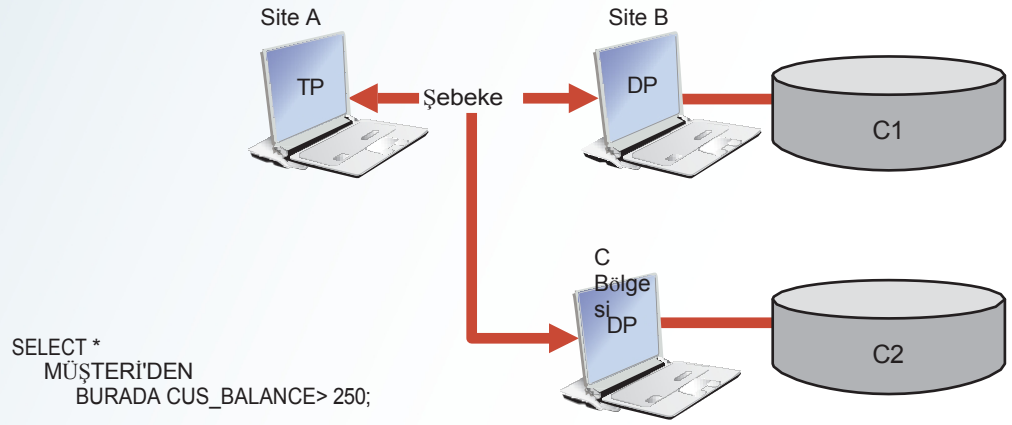
Şekil 12.12 Dağıtılmış Bir Talep



Dağıtılmış istek özelliği, tek bir isteğin fiziksel olarak bölümlenmiş bir tabloya başvurmasına da olanak tanır. Örneğin, bir CUSTOMER tablosunun sırasıyla B ve C Sitelerinde bulunan C1 ve C2 olmak üzere iki parçaya bölündüğünü varsayalım. Ayrıca, son kullanıcının bakiyesi 250 doları aşan tüm müşterilerin bir listesini almak istediğini varsayalım. Bu istek Şekil 12.13'te gösterilmektedir. Tam parçalı şeffaflık desteği yalnızca dağıtılmış istekleri destekleyen bir DDBMS tarafından sağlanır.

Dağıtık veritabanı sistemlerindeki farklı veritabanı istek türlerini anlamak, işlem şeffaflığı sorununu daha etkili bir şekilde ele almanıza yardımcı olur. İşlem şeffaflığı, dağıtık işlemlerin merkezi işlemler gibi ele alınmasını sağlayarak serileştirilebilirliklerini garanti eder. (Gerekirse Bölüm 10, İşlem Yönetimi ve Eşzamanlılık Kontrolü'nü gözden geçirin). Yani, eşzamanlı işlemlerin yürütülmesi, dağıtılmış olsun ya da olmasın, veritabanını bir tutarlı durumdan diğerine götürecektir.

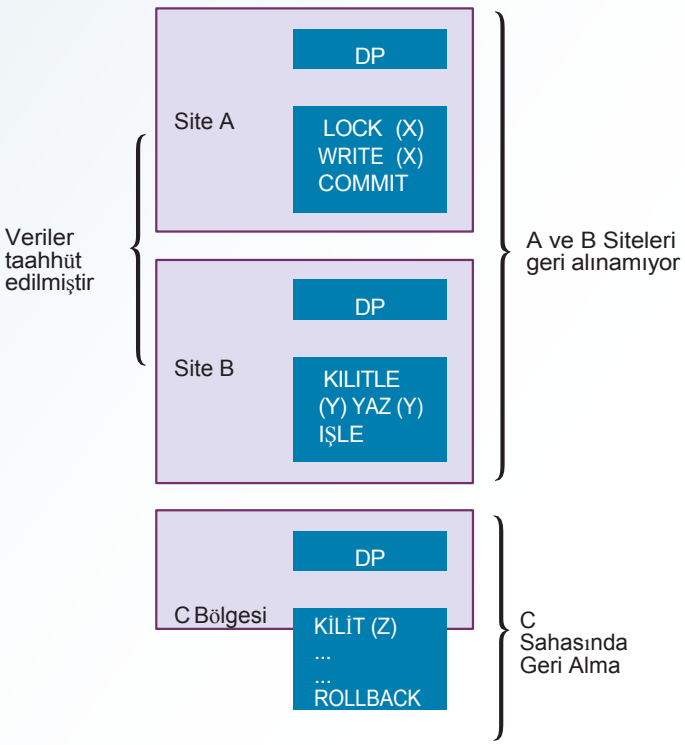
Şekil 12.13 Başka Bir Dağıtılmış Talep



12-9 b Dağıtılmış Eşzamanlılık Kontrolü

Dağıtık veritabanlarında eşzamanlılık kontrolü özellikle önemli hale gelir çünkü çok bölgeli, çok işlemli işlemlerin veri tutarsızlıkları ve kilitlenmiş işlemler yaratma olasılığı tek bölgeli sistemlere göre daha yüksektir. Örneğin, bir DDBMS'nin TP bileşeni, işlemi kaydetmek için son bir COMMIT yayınlanmadan önce işlemin tüm bölümlerinin tüm sitelerde tamamlanmasını sağlamalıdır. Bir işlemin üç DP sitesindeki verileri güncellediğini varsayalım. İlk iki DP sitesi işlemi tamamlar ve her bir yerel DP'de verileri işler; ancak üçüncü DP sitesi işlemi işleyemez. Böyle bir senaryo, kaçınılmaz bütünlük sorunlarıyla birlikte tutarsız bir veritabanı ortaya çıkaracaktır, çünkü işlenen veriler işlenmemiş olamaz! Bu sorun Şekil 12.14'te gösterilmektedir. Bu sorunun çözümü, daha sonra inceleyeceğiniz *iki aşamalı* bir *işleme protokolüdür*.

Şekil 12.14 Erken Bağlılığın Etkisi



12-9c İki Aşamalı Taahhüt Protokolü

Merkezi veritabanları yalnızca bir DP gerektirir. Tüm veritabanı işlemleri yalnızca bir sitede gerçekleşir ve veritabanı işlemlerinin sonuçları DBMS tarafından hemen bilinir. Bunun aksine, dağıtılmış veritabanları bir işlemin birden fazla yerdeki verilere erişmesini mümkün kılar. Tüm siteler işlemin kendi kısımlarını taahhüt edene kadar son bir COMMIT yayınlanmamalıdır. **İki aşamalı taahhüt protokolü (2PC),** bir işlemin bir kısmının taahhüt edilememesi durumunda, işleme katılan diğer sitelerde yapılan tüm değişikliklerin tutarlı bir veritabanı durumunu korumak için geri alınacağını garanti eder.

iki aşamalı taahhüt protokolü (2PC)
Bir DDBMS'de, dağıtılmış bütünlüğün yanı sıra işlemlerin atomikliğini ve veritabanı tutarlılığını sağlamak için kullanılan bir algoritma.

Her DP kendi işlem günlüğünü tutar. İki aşamalı commit protokolü, her DP için işlem günlüğü girdisinin veritabanı parçası gerçekten güncellenmeden önce yazılmasını gerektirir (bkz. Bölüm 10). Bu nedenle, iki aşamalı commit protokolü bir DO-UN- DO-REDO protokolü ve bir write-ahead protokolü gerektirir.

DO-UNDO-REDO protokolü, DP tarafından sistemin işlem günlüğü girdileri yardımıyla işlemleri geri ve ileri sarmak için kullanılır. DO-UNDO-REDO protokolü üç tür işlem tanımlar:

DO-UNDO-REDO protokol
Bir veri işlemcisi (DP) tarafından bir sistemin işlem günlüğü girdileri yardımıyla işlemleri geri almak veya ileri almak için kullanılan bir protokol.

- DO işlemi gerçekleştirir ve "önce" ve "sonra" değerlerini işlem günlüğüne kaydeder.
- UNDO, dizinin DO kısmı tarafından yazılan günlük girdilerini kullanarak bir işlemi tersine çevirir.
- REDO, dizinin DO kısmı tarafından yazılan günlük girdilerini kullanarak bir işlemi yeniden yapar.

DO, UNDO ve REDO işlemlerinin yürütülürken bir sistem çökmesi durumunda hayatta kalabilmelerini sağlamak için bir write-ahead protokolü kullanılır. **Önceden yazma protokolü**, gerçek işlem gerçekleşmeden önce günlük girişinin kalıcı depolama alanına yazılmasını zorlar.

İki aşamalı taahhüt protokolü, iki tür düğüm arasındaki işlemleri tanımlar: **koordinatör** ve bir veya daha fazla **ast** veya *kohort*. Katılımcı düğümler bir koordinatör üzerinde anlaşırlar. Genel olarak, koordinatör rolü işlemi başlatan düğüme atanır. Ancak, farklı sistemler çeşitli, daha sofistike seçim yöntemleri uygular. Pro-tokol, aşağıdaki bölümlerde gösterildiği gibi iki aşamada uygulanmaktadır.

write-ahead protokolü
İşlem günlüklerinin yazılmasını sağlayan bir protokol
herhangi bir veritabanı verisi gerçekten güncellenmeden önce kalıcı depolamaya aktarılır.

KOORDİNATÖR
Bir DDBMS'de iki aşamalı COMMIT'in yürütülmesini koordine eden işlem işlemcisi (TP) düğümü.

ast
Bir DDBMS', dağıtılmış bir işleme katılan bir veri işlemci (DP) düğümü
iki aşamalı COMMIT protokolünü kullanarak.

1. Aşama: Hazırlık

Koordinatör tüm astlara bir TAAHHÜDE HAZIRLAN mesajı gönderir.

1. Astlar mesajı alır, write-ahead protokolünü kullanarak işlem günlüğünü yazar ve koordinatöre bir onay mesajı (YES/PREPARED TO COMMIT veya NO/NOT PREPARED) gönderir.
2. Koordinatör, tüm düğümlerin taahhütte bulunmaya hazır olduğundan emin olur veya eylemi iptal eder.

Eğer tüm düğümler COMMIT'e HAZIR ise, işlem 2. Aşamaya geçer. Bir veya daha fazla düğüm HAYIR veya HAZIR DEĞİL yanıtını verirse, koordinatör tüm astlara bir ABORT mesajı yayınlar.

2Aşama: Son TAAHHÜT

1. Koordinatör tüm astlara bir COMMIT mesajı yayınlar ve yanıtları bekler.
2. Her bir ast COMMIT mesajını alır ve ardından DO protokolünü kullanarak veritabanını günceller.
3. Astlar, koordinatöre bir TAAHHÜT EDİLDİ veya TAAHHÜT EDİLMEDİ mesajı ile yanıt verir.

Eğer bir veya daha fazla ast taahhütte bulunmazsa, koordinatör bir ABORT mesajı göndererek onları tüm değişiklikleri UNDO etmeye zorlar.

İki aşamalı commit'in amacı, her düğümün işlemin kendisine düşen kısmını commit etmesini sağlamaktır; aksi takdirde işlem iptal edilir. Düğümlerden biri işlem yapmazsa, veritabanını kurtarmak için gerekli bilgiler işlem günlüğünde bulunur ve veritabanı DO-UNDO-REDO protokolü ile kurtarılabilir. (Günlük bilgilerinin write-ahead protokolü kullanılarak güncellendiğini unutmayın).

12-10 Performans ve Arıza Şeffaflığı

Bir veritabanının en önemli işlevlerinden biri, verileri kullanılabilir hale getirme yeteneğidir. Web tabanlı dağıtık veri sistemleri yüksek kullanılabilirlik gerektirir; bu da yalnızca verilerin erişilebilir olması değil, aynı zamanda taleplerin zamanında işlenmesi anlamına gelir. Örneğin, ortalama bir Google araması saniyenin altında bir yanıt süresine sahiptir. En son ne zaman bir Google sorgusu girdiniz ve sonuçlar için birkaç saniyeden fazla beklediniz?

Performans şeffaflığı bir DDBMS'nin merkezi bir veritabanı gibi performans göstermesini sağlar. Başka bir deyişle, veri dağıtımı nedeniyle herhangi bir performans düşüşü yaşanmamalıdır. Arıza şeffaflığı, bir düğüm veya ağ arızası durumunda sistemin çalışmaya devam etmesini sağlar. Bunlar iki ayrı konu olmasına rağmen, başarısız bir düğüm veya tıkalı bir ağ yolu performans sorunlarına neden olabileceğinden birbirleriyle ilişkilidirler. Bu nedenle, bu bölümde her iki konu da ele alınmaktadır.

Not

Bölüm 11, Veritabanı Performans Ayarlama ve Sorgu Optimizasyonu, sorgu optimizasyonu hakkında ek ayrıntılar sağlar.

Sorgu optimizasyonunun amacı, bir isteğin yürütülmesiyle ilişkili toplam maliyeti en aza indirmektir. Bir istekle ilişkili maliyetler aşağıdakilerin bir fonksiyonudur:

- Birden fazla uzak siteden verilere erişim için gereken erişim süresi (I/O) maliyeti
- Dağıtık veri tabanı sistemlerinde düğümler arasında veri iletimi ile ilişkili iletişim maliyeti
- Dağıtılmış işlemleri yönetmenin işlem yüküyle ilişkili CPU zaman maliyeti

Maliyetler genellikle iletişim ya da işlem maliyetleri olarak sınıflandırılsa da, bu ikisini birbirinden ayırmak zordur. Tüm sorgu optimizasyon algoritmaları aynı parametreleri kullanmaz ve tüm algoritmalar her bir parametreye aynı ağırlığı atamaz. Örneğin, bazı algoritmalar toplam süreyi en aza indirirken, diğerleri iletişim süresini en aza indirir ve yine de diğerleri, diğer maliyetlere göre maliyetinin önemsiz olduğunu düşünerek CPU süresini hesaba katmaz.

Bölüm 11'de öğrendiğiniz gibi, merkezi bir veritabanı, verilere erişmenin en verimli yolunu bulmak için her veri isteğini değerlendirir. Tüm verilerin yerel olarak depolandığı ve veriler üzerinde çalışan tüm aktif işlemlerin merkezi VTYS tarafından bilindiği bu makul bir gerekliliktir. Buna karşılık, bir DDBMS'de işlemler birden fazla düğüm arasında dağıtılır; bu nedenle, hangi verilerin kullanıldığını belirlemek daha karmaşık hale gelir. Bu nedenle, dağıtık bir veri ortamında veri taleplerinin çözümlenmesinde aşağıdaki hususlar dikkate alınmalıdır:

- *Veri dağıtımı*. Bir DDBMS'de sorgu çevirisi daha karmaşıktır çünkü DDBMS'nin hangi parçaya erişeceğine karar vermesi gerekir. (Dağıtım şeffaflığı bu bölümün önceki kısımlarında açıklanmıştır.) Bu durumda, bir sorguyu yürüten bir TP hangi parçalara erişeceğini seçmeli, seçilen uzak DP'lere birden fazla veri isteği oluşturmalı, DP yanıtlarını birleştirmeli ve verileri uygulamaya sunmalıdır.

- *Veri replikasyonu*. Buna ek olarak, veriler birkaç farklı sitede de çoğaltılabilir. Veri replikasyonu erişim problemini daha da karmaşık hale getirir çünkü veritabanı verilerin tüm kopyalarının tutarlı olmasını sağlamalıdır. Bu nedenle, dağıtık veritabanı sistemlerinde sorgu optimizasyonunun önemli bir özelliği, *replika* şeffaflığı sağlaması gerektiğidir. Çoğaltma **saydamlığı**, DDBMS'nin birden fazla veri kopyasını kullanıcıdan gizleme yeteneğini ifade eder. Bu yetenek özellikle veri güncelleme işlemlerinde önemlidir. Yalnızca okunabilir bir istek işleniyorsa, bu istek mevcut herhangi bir uzak DP'ye erişilerek karşılanabilir. Ancak, bir yazma isteğinin işlenmesi, veri tutarlılığını korumak için mevcut tüm parçaların "senkronize edilmesini" de içerir. Bölüm 12-9c'de öğrendiğiniz iki aşamalı commit protokolü işlemin başarıyla tamamlanmasını sağlar. Ancak, veriler başka sitelerde çoğaltılıyorsa, DDBMS'ler tüm parçaların tutarlılığını da sağlamalıdır; yani, tüm parçalar karşılıklı olarak tutarlı olmalıdır. Bunu başarmak için bir DP tüm değişiklikleri yakalar ve bunları her bir uzak kopyaya gönderir. Bu, sistemde gecikmelere neden olur ve temelde tüm veri değişikliklerinin tüm replikalar tarafından hemen görülmediği anlamına gelir. (Bu konunun sonuçları Bölüm 12-12, CAP Teoremi'nde açıklanmıştır).
- *Ağ ve düğüm kullanılabilirliği*. Uzak sitelerle ilişkili yanıt süresi kolayca önceden belirlenemez çünkü bazı düğümler sorgunun kendilerine düşen kısmını diğerlerinden kısa sürede tamamlar ve ağ yolu performansı bant genişliği ve trafik yükleri nedeniyle değişir. Bu nedenle, performans şeffaflığı elde etmek için, DDBMS **ağ gecikmesi**, bir veri paketinin A noktasından B noktasına gidiş dönüş yapması gereken sürenin getirdiği gecikme veya **ağ bölünmesi**, düğümler bir ağ arızası nedeniyle aniden kullanılamaz hale geldiğinde ortaya çıkan gecikme gibi konuları dikkate almalıdır.

replika şeffaflığı
DDBMS'nin birden fazla veri kopyasının varlığını kullanıcıdan gizleme yeteneği.

ağ gecikmesi
Bir veri paketinin A noktasından B noktasına yapması için gereken süreye bağlı olarak ortaya çıkan gecikme.

ağ **bölümlemesi** Bir ağ arızası nedeniyle düğümler aniden kullanılamaz hale geldiğinde ortaya çıkan gecikme. Dağıtılmış veritabanlarında, sistem bu durumun olasılığını hesaba katmalıdır.

Bir veritabanının nasıl bölümleneceğini ve veritabanı parçalarının nereye yerleştirileceğini dikkatlice planlamak, dağıtılmış bir veritabanının performansını ve tutarlılığını sağlamaya yardımcı olabilir. Aşağıdaki bölümde dağıtık veritabanı tasarımına ilişkin konular ele alınmaktadır.

12-11 Dağıtık Veritabanı Tasarımı

Veritabanı ister merkezi ister dağıtık olsun, önceki bölümlerde açıklanan tasarım ilkeleri ve kavramları hala geçerlidir. Ancak, dağıtık bir veritabanının tasarımı üç yeni konu ortaya çıkarır:

- Veritabanı parçalara nasıl bölünür?
- Hangi parçaların çoğaltılacağı
- Bu parçaların ve kopyaların nerede bulunacağı

Veri parçalanması ve veri çoğaltma ilk iki sorunla, veri dağıtımı ise üçüncü sorunla ilgilenir. İdeal olarak, dağıtık bir veritabanındaki veriler, performansı en üst düzeye çıkarmak, kullanılabilirliği artırmak (darboğazları azaltmak) ve mobil uygulamalar için giderek artan bir gereksinim olan konum farkındalığı sağlamak için eşit olarak dağıtılmalıdır.

12-11a Veri Parçalanması

Veri parçalama, tek bir nesneyi iki veya daha fazla bölüme veya parçaya ayırmanıza olanak tanır. Nesne bir kullanıcı veritabanı, bir sistem veritabanı ya da bir tablo olabilir. Her bir parça bir bilgisayar ağı üzerindeki herhangi bir yerde depolanabilir. Veri parçalanması hakkındaki bilgiler dağıtılmış veri kataloğunda (DDC) saklanır ve kullanıcı isteklerini işlemek için TP tarafından bu bilgilere erişilir.

Burada tartışıldığı gibi veri parçalama stratejileri tablo düzeyine dayanır ve bir tablonun mantıksal parçalara bölünmesinden oluşur. Üç tür veri parçalama stratejisini inceleyeceksiniz: yatay, dikey ve karışık. (Parçalanmış bir tablonun, birleşimler ve birleştirmelerin bir kombinasyonu ile parçalanmış parçalarından her zaman yeniden oluşturulabileceğini unutmayın).

veri parçalanması
Tek bir nesnenin iki veya daha fazla segmente veya parçaya bölünmesine izin veren bir DDBMS özelliği. Nesne bir kullanıcı veritabanı, bir sistem veritabanı ya da bir tablo olabilir. Her parça bir bilgisayar ağı üzerindeki herhangi bir sitede saklanabilir.

yatay parçalanma
Bir tabloyu benzersiz satırlardan oluşan alt kümelere ayıran dağıtılmış veritabanı tasarım süreci.

dikey parçalama Dağıtık veritabanı tasarımında, bir tabloyu orijinal tablodaki sütunların bir alt kümesine ayıran işlem. Parçalar ortak bir birincil anahtarı paylaşmalıdır.

- **Yatay parçalama**, bir ilişkinin tuple'ların (satırların) alt kümelerine (parçalarına) bölünmesi anlamına gelir. Her parça farklı bir düğümde saklanır ve her parçanın benzersiz satırları vardır. Ancak, benzersiz satırların tümü aynı özniteliklere (sütunlara) sahiptir. Kısacası, her bir parça tek bir öznitelik üzerinde WHERE cümlesiyle bir SELECT deyiminin eşdeğerini temsil eder.
- **Dikey parçalama**, bir ilişkinin öznitelik (sütun) alt kümelerine bölünmesi anlamına gelir. Her alt küme (parça) farklı bir düğümde saklanır ve her parçanın benzersiz sütunları vardır - tüm parçalar için ortak olan anahtar sütunu hariç. Bu SQL'deki PROJECT deyiminin eşdeğeridir.
- **Karma parçalama**, yatay ve dikey stratejilerin bir kombinasyonunu ifade eder. Başka bir deyişle, bir tablo, her biri özniteliklerin (sütunların) bir alt kümesine sahip olan birkaç yatay alt kümeye (satırlara) bölünebilir.

Parçalama stratejilerini göstermek için, Şekil 12.15'te gösterildiği gibi XYZ Şirketi için CUSTOMER tablosunu kullanın. Tablo CUS_NUM, CUS_NAME, CUS_ADDRESS, CUS_STATE, CUS_LIMIT, CUS_BAL, CUS_RATING niteliklerini içerir,
ve CUS_DUE.

Şekil 12.15 Örnek Bir Müşteri Tablosu

Tablo adı: CUSTOMER **Veritabanı adı: Ch12_Text**

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
10	Sinex, Inc.	12 Main St.	TN	3500.00	2700.00	3	1245.00
11	Martin Corp.	321 Sunset Blvd.	FL	6000.00	1200.00	1	0.00
12	Mynux Corp.	910 Eagle St.	TN	4000.00	3500.00	3	3400.00
13	BTBC, Inc.	Rue du Monde	FL	6000.00	5890.00	3	1090.00
14	Victory, Inc.	123 Maple St.	FL	1200.00	550.00	1	0.00
15	NBCC Corp.	909 High Awe.	GA	2000.00	350.00	2	50.00

karışık parçalanma
Bir tablonun birkaç satıra bölünebildiği ve her satırın özniteliklerin (sütunların) bir alt kümesine sahip olduğu veri parçalama için yatay ve dikey stratejilerin bir kombinasyonu.

Yatay Parçalanma

Bu durumda, bir tablo birden fazla satır alt kümesine bölünür. Bir tabloyu yatay olarak bölümlemenin çeşitli yolları vardır:

- *Yuvarlak-robin bölümleme*. Satırlar, tüm parçalar arasında eşit bir dağılım sağlamak için belirli bir parçaya yuvarlak robin şeklinde (F1, F2, F3, ..., *Fn*) atanır. Ancak, "konum farkındalığı "na ihtiyacınız varsa bu iyi bir strateji değildir - talep edenin coğrafi konumuna bağlı olarak bir sorguyu hangi DP düğümünün işleyeceğini belirleme yeteneği. Örneğin, Florida müşterilerinden gelen tüm sorguların yalnızca Florida depolayan bir parçadan çözümlenmesini istersiniz. Elbette bu parçanın Florida'ya yakın bir düğümde bulunmasını da istersiniz.

bölme anahtarı
Bölümlenmiş veritabanlarında, satırın depolanacağı parçayı belirleyen bir tablodaki bir veya daha fazla öznitelik.

- *Bir bölüm anahtarına dayalı aralık bölümleme*. **Bölüm anahtarı**, bir tabloda bir satırın depolanacağı parçayı belirleyen bir veya daha fazla özniteliktir. Örneğin, konum farkındalığı sağlamak istiyorsanız, iyi bir bölüm anahtarı müşteri durumu alanı olabilir. Bu, en yaygın ve kullanışlı veri bölümleme stratejisidir.

Bir tabloyu bölümlemek için bir bölüm anahtarının nasıl kullanılacağına daha yakından bakın. XYZ Şirketinin kurumsal yönetiminin üç eyaletteki müşterileri hakkında bilgiye ihtiyaç duyduğunu, ancak her bir eyaletteki (TN, FL ve GA) şirket konumlarının yalnızca yerel müşterilerle ilgili verilere ihtiyaç duyduğunu varsayalım. Bu tür gereksinimlere dayanarak, verileri eyaletlere göre dağıtmaya karar verirsiniz. Bu nedenle, yatay parçaları Tablo 12.5'te gösterilen yapıya uyacak şekilde tanımlarsınız.

Tablo 12.5 Eyalete Göre Müşteri Tablosunun Yatay Olarak Parçalanması

Parça Adı	Konum	Durum	Düğüm Adı	Müşteri Numaraları	Satır Sayısı
CUST_H1	Tennessee	CUS_STATE 5 'TN'	NAS	10, 12	2
CUST_H2	Gürcistan	CUS_STATE 5 'GA'	ATL	15	1
CUST_H3	Florida	CUS_STATE 5 'FL'	TAM	11, 13, 14	3

Bölüm anahtarı CUS_STATE alanı olacaktır. Her yatay parça farklı sayıda satıra sahip olabilir, ancak her parça aynı özniteliklere sahip *olmalıdır*. Ortaya çıkan parçalar Şekil 12.16'da gösterilen üç tabloyu verir.

Şekil 12.16 Üç Lokasyonda Masa Parçaları

Veritabanı adı: Ch12_Text

Tablo adı: CUST_H1 **Konum: Tennessee** **Düğüm: NAS**

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
10	Sinex, Inc.	12 Main St.	TN	3500.00	2700.00	3	1245.00
12	Mynux Corp.	910 Eagle St.	TN	4000.00	3500.00	3	3400.00

Tablo adı: CUST_H2 **Konum: Gürcistan** **Düğüm: ATL**

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
15	NBCC Corp.	909 High Ave.	GA	2000.00	350.00	2	50.00

Tablo adı: CUST_H3 **Konum: Florida** **Düğüm: TAM**

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
11	Martin Corp.	321 Sunset Blvd.	FL	6000.00	1200.00	1	0.00
13	BTBC, Inc.	Rue du Monde	FL	6000.00	5890.00	3	1090.00
14	Victory, Inc.	123 Maple St.	FL	1200.00	550.00	1	0.00

Dikey Parçalanma

MÜŞTERİ ilişkisini, bir nitelikler koleksiyonundan dikey parçalara da bölebilirsiniz. Örneğin, şirketin iki departmana ayrıldığını varsayalım: servis departmanı ve tahsilat . Her departman ayrı bir binada yer almaktadır ve her biri CUSTOMER tablosunun sadece birkaç özniteliğiyle ilgilenmektedir. Bu durumda, parçalar Tablo 12.6'da gösterildiği gibi tanımlanır.

Tablo 12.6 Müşteri Tablosunun Dikey Parçalanması

Parça Adı	Konum	Düğüm Adı	Öznitelik Adları
CUST_V1	Hizmet Binası	SVC	CUS_NUM, CUS_NAME, CUS_ADDRESS, CUS_STATE
CUST_V2	Koleksiyon Bldg	ARC	CUS_NUM, CUS_LIMIT, CUS_BAL, CUS_RATING, CUS_DUE

Her dikey parça aynı sayıda satıra sahip olmalıdır, ancak farklı niteliklerin dahil edilmesi anahtar sütuna bağlıdır. Dikey parçalama sonuçları Şekil 12.17'de gösterilmektedir. Anahtar niteliğin (CUS_NUM) hem CUST_V1 hem de CUST_V2 parçaları için ortak olduğuna dikkat edin.

Şekil 12.17 Dikey Olarak Parçalanmış Tablo İçindekiler

Veritabanı adı: Ch12_Text

Tablo adı: CUST_V1 Konum: Hizmet Binası Düğüm: SVC

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE
10	Sinex, Inc.	12 Main St.	TN
11	Martin Corp.	321 Sunset Blvd.	FL
12	Mynux Corp.	910 Eagle St.	TN
13	BTBC, Inc.	Rue du Monde	FL
14	Victory, Inc.	123 Maple St.	FL
15	NBCC Corp.	909 High Ave.	GA

Tablo adı: CUST_V2 Konum: Koleksiyon Oluşturma Düğümü: ARC

CUS_NUM	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
10	3500.00	2700.00	3	1245.00
11	6000.00	1200.00	1	0.00
12	4000.00	3500.00	3	3400.00
13	6000.00	5890.00	3	1090.00
14	1200.00	550.00	1	0.00
15	2000.00	350.00	2	50.00

Karışık Parçalanma

XYZ Şirketinin yapısı, çeşitli şirket konumlarını barındırmak için MÜŞTERİ verilerinin yatay olarak parçalanmasını gerektirir; konumlar içinde, iki departmanı (hizmet ve tahsilat) barındırmak için verilerin dikey olarak parçalanması gerekir. KısacasıMÜŞTERİ tablosu karışık parçalama gerektirir.

Karma parçalama iki aşamalı bir prosedür gerektirir. İlk olarak, bir eyalet (CUS_STATE) içindeki konuma dayalı olarak her bir site için yatay parçalama yapılır. Yatay parçalama, her bir tesiste bulunan müşteri küplerinin alt kümelerini (yatay parçalar) verir. Departmanlar farklı binalarda bulunduğundan, öznitelikleri bölmek için her bir yatay parça içinde dikey parçalama kullanılır, böylece her bir departmanın her bir alt tesisteki bilgi ihtiyaçları karşılanır. Karma parçalama Tablo 12.7'de gösterilen sonuçları verir.

Tablo 12.7 Müşteri Tablosunun Karışık Parçalanması

Parça Adı	Konum	Yatay Kriterler	Düğüm Adı	Sahada Ortaya Çıkan Satırlar	Her Parçada Dikey Kriter Nitelikleri
CUST_M1	TN-Servis	CUS_STATE 5 TN	NAS-S	10, 12	CUS_NUM, CUS_NAME CUS_ADDRESS, CUS_STATE
CUST_M2	TN-Koleksiyonu	CUS_STATE 5 TN	NAS-C	10, 12	CUS_NUM, CUS_LIMIT, CUS_BAL, CUS_RATING, CUS_DUE
CUST_M3	GA-Service	CUS_STATE 5 GA	ATL-S	15	CUS_NUM, CUS_NAME CUS_ADDRESS, CUS_STATE
CUST_M4	GA-Collection	CUS_STATE 5 GA	ATL-C	15	CUS_NUM, CUS_LIMIT, CUS_BAL, CUS_RATING, CUS_DUE
CUST_M5	FL-Servis	CUS_STATE 5 FL	TAM-S	11, 13, 14	CUS_NUM, CUS_NAME CUS_ADDRESS, CUS_STATE
CUST_M6	FL-Koleksiyonu	CUS_STATE 5 FL	TAM-C	11, 13, 14	CUS_NUM, CUS_LIMIT, CUS_BAL, CUS_RATING, CUS_DUE

Tablo 12.7'de gösterilen her bir parça, müşteri verilerini eyalet bazında ve her bir eyalet içinde, her bir departmanın veri gereksinimlerine uyacak şekilde departman konumuna göre içermektedir. Tablo 12.7'de listelenen parçalara karşılık gelen tablolar Şekil 12.18'de gösterilmektedir.

Şekil 12.18 Karma Parçalanma Sürecinden Sonra Tablo İçeriği

Veritabanı adı: Ch12_Text

Tablo adı: CUST_M1 **Konum: TN-Servis** **Düğüm: NAS-S**

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE
10	Sinex, Inc.	12 Main St.	TN
12	Mynux Corp.	910 Eagle St.	TN

Tablo adı: CUST_M2 **Konum: TN-Koleksiyonu** **Düğüm: NAS-C**

CUS_NUM	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
10	3500.00	2700.00	3	1245.00
12	4000.00	3500.00	3	3400.00

Tablo adı: CUST_M3 **Konum: GA-Servis** **Düğüm: ATL-S**

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE
15	NBCC Corp.	909 High Ave.	GA

Tablo adı: CUST_M4 **Konum: GA-Koleksiyon** **Düğüm: ATL-C**

CUS_NUM	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
15	2000.00	350.00	2	50.00

Tablo adı: CUST_M5 **Konum: FL-Servis** **Düğüm: TAM-S**

CUS_NUM	CUS_NAME	CUS_ADDRESS	CUS_STATE
11	Martin Corp.	321 Sunset Blvd.	FL
13	BTBC, Inc.	Rue du Monde	FL
14	Victory, Inc.	123 Maple St.	FL

Tablo adı: CUST_M6 **Konum: FL-Koleksiyon** **Düğüm: TAM-C**

CUS_NUM	CUS_LIMIT	CUS_BAL	CUS_RATING	CUS_DUE
11	6000.00	1200.00	1	0.00
13	6000.00	5890.00	3	1090.00
14	1200.00	550.00	1	0.00

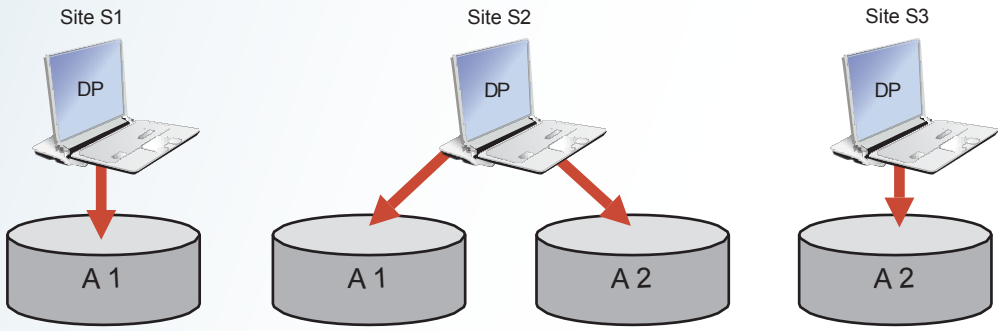
12-11b Veri Çoğaltma

Veri çoğaltma, bir bilgisayar ağı tarafından hizmet verilen birden fazla sitede veri kopyalarının depolanması anlamına gelir. Parça kopyalar, belirli bilgi gereksinimlerini karşılamak için çeşitli yerlerde depolanabilir. Parça kopyaların varlığı veri kullanılabilirliğini ve yanıt süresini artırabileceğinden, veri kopyaları iletişim ve toplam sorgu maliyetlerini azaltmaya yardımcı olabilir.

A veritabanının A1 ve A2 olmak üzere iki parçaya bölündüğünü varsayalım. Çoğaltılmış dağıtılmış bir veritabanında, Şekil 12.19'da gösterilen senaryo mümkündür: A1 parçası S1 ve S2 sitelerinde saklanırken, A2 parçası S2 ve S3 sitelerinde saklanır.

veri çoğaltma
Çoğaltılmış veritabanı parçalarının bir DDBMS üzerinde birden fazla sitede depolanması. Parçaların çoğaltılması son kullanıcı için şeffaftır. Veri replikasyonu hata toleransı ve performans iyileştirmeleri sağlar.

Şekil 12.19 Veri Çoğaltma



Karşılıklı tutarlılık kuralı
Veri parçalarının tüm kopyalarının aynı olmasını gerektiren bir veri çoğaltma kuralı.

Çoğaltılan veriler, veri parçalarının tüm kopyalarının aynı olmasını gerektiren **karşılıklı tutarlılık kuralına** tabidir. Bu nedenle, replikalar arasında veri tutarlılığını korumak için, DDBMS replikaların bulunduğu tüm sitelerde bir veritabanı güncellemesi yapılmasını sağlamalıdır.

Temelde iki tür çoğaltma vardır:

- *İtmeli çoğaltma*. Bir veri güncellemesinden sonra, kaynak DP düğümü, verilerin hemen güncellenmesini sağlamak için değişiklikleri replika düğümlere gönderir. Bu tür çoğaltma, veri tutarlılığını korumaya odaklanır. Ancak, tüm düğümlerde veri tutarlılığının sağlanmasındaki gecikme nedeniyle veri kullanılabilirliğini azaltır.
- *Çekme replikasyonu*. Bir veri güncellemesinden sonra, kaynak DP düğümü replika düğümlere güncellemeyi bildirmek için "mesajlar" gönderir. Çoğaltma düğümleri güncellemeleri kendi yerel parçalarına ne zaman uygulayacaklarına karar verir. Bu tür replikasyonda veri güncellemeleri replikalara daha yavaş yayılır. Odak noktası veri kullanılabilirliğini korumaktır. Ancak bu replikasyon tarzı geçici veri tutarsızlıklarına izin verir.

Çoğaltmanın daha iyi veri kullanılabilirliği, daha iyi yük dağılımı, daha iyi veri hatası toleransı ve daha düşük sorgu maliyetleri gibi bazı faydaları olsa da, her bir veri kopyasının sistem tarafından muhafaza edilmesi gerektiğinden ek DDBMS işlem yükü de getirir. Ayrıca, veriler başka bir sitede çoğaltıldığından, ilişkili depolama maliyetleri ve artan işlem süreleri vardır (karşılıklı tutarlılık kuralına uymak için verilerin birkaç sitede eşzamanlı olarak güncellenmesi gerektiğinden). Bir DDBMS'ye yüklenen replika ek yükünü göstermek için, DDBMS'nin veritabanını kullanmak için gerçekleştirmesi gereken işlemleri göz önünde bulundurun:

- Veritabanı parçalara ayrılmışsa, DDBMS uygun parçalara erişmek için bir sorguyu alt sorgulara ayırmalıdır.
- Veritabanı çoğaltılmışsa, DDBMS hangi kopyaya erişeceğine karar vermelidir. Bir READ işlemi, işlemi karşılamak için *en yakın kopyayı* seçer. Bir WRITE işlemi, karşılıklı tutarlılık kuralını karşılamak için *tüm kopyaların* seçilmesini ve güncellenmesini gerektirir.
- TP, seçilen her DP'ye yürütülmesi için bir veri talebi gönderir.
- DP her bir talebi alır ve yürütür ve verileri TP'ye geri gönderir.
- TP, DP yanıtlarını bir araya getirir.

Ağ topolojisi ve iletişim çıktıları gibi ek faktörleri göz önünde bulundurduğunuzda sorun daha karmaşık hale gelir.

Üç replikasyon senaryosu mevcuttur: bir veritabanı tamamen replike edilebilir, kısmen replike edilebilir veya replike edilmeyebilir.

tamamen çoğaltılmış veritabanı
Bir DDBMS'de, her bir veritabanı parçasının birden fazla kopyasını birden fazla sitede depolayan dağıtılmış veritabanı.

kısmen çoğaltılmış veritabanı
Yalnızca bazı veritabanı parçalarının kopyalarının birden fazla sitede depolandığı dağıtılmış bir veritabanı.

çoğaltılmamış veritabanı
Her bir veritabanının parçası tek bir bölgede depolanır.

- **Tamamen çoğaltılmış** bir **veritabanı**, her veritabanı parçasının birden çok kopyasını birden çok sitede depolar. Bu durumda, tüm veritabanı parçaları çoğaltılır. Tamamen çoğaltılmış bir veritabanı, sisteme yüklediği ek yük miktarı nedeniyle pratik olmayabilir.
- **Kısmen çoğaltılmış** bir **veritabanı**, bazı veritabanı parçalarının birden fazla kopyasını birden fazla sitede depolar. Çoğu DDBMS kısmen çoğaltılmış veritabanını iyi bir şekilde idare edebilir.
- Çoğaltılmamış bir **veritabanı**, her veritabanı parçasını tek bir sitede depolar. Bu nedenle, yinelenen veritabanı parçaları yoktur.

Veri replikasyonu kullanma kararını etkileyen çeşitli faktörler vardır:

- *Veritabanı boyutu*. Çoğaltılan veri miktarı, depolama gereksinimleri ve veri iletim maliyetleri üzerinde etkili olacaktır. Büyük miktarda verinin çoğaltılması, diğer uygulamaları etkileyebilecek bir zaman aralığı ve daha yüksek ağ bant genişliği gerektirir.
- *Kullanım sıklığı*. Veri kullanım sıklığı, verilerin ne sıklıkla güncellenmesi gerektiğini belirler. Sık kullanılan veriler, örneğin sadece üç ayda bir kullanılan büyük veri setlerinden daha sık güncellenmelidir.

- *Maliyetler*. Maliyetler, çoğaltılmış verilerle ilişkilendirilen hata toleransı avantajlarına karşılık işlemlerin ve bileşenlerinin senkronize edilmesiyle ilişkili performans, yazılım ek yükü ve yönetim için olanları içerir.

Uzakta bulunan verilerin kullanım sıklığı yüksek ve veritabanı büyük olduğunda, veri çoğaltma veri isteklerinin maliyetini azaltabilir. Veri çoğaltma bilgileri DDC'de saklanır ve bunların içeriği TP tarafından bir veritabanı parçasının hangi kopyasına erişileceğine karar vermek için kullanılır. Veri çoğaltma, kayıp verilerin geri yüklenmesini mümkün kılar.

12-11c Veri Tahsisi

Veri tahsisi, verilerin nereye yerleştirileceğine karar verme sürecini tanımlar. Veri tahsis stratejileri aşağıdaki gibidir:

- **Merkezi veri tahsisi** ile tüm **veri** tabanı tek bir yerde depolanır.
- **Bölümlere ayrılmış veri tahsisi** ile veritabanı iki veya daha fazla ayrık parçaya (fragman) bölünür ve iki veya daha fazla yerde depolanır.
- **Çoğaltılmış veri tahsisi** ile, bir veya daha fazla veritabanı parçasının kopyaları birkaç sitede saklanır.

Bir bilgisayar ağı üzerinden veri dağıtımı, veri bölümleme, veri çoğaltma veya her ikisinin bir kombinasyonu yoluyla gerçekleştirilir. Veri tahsisi, bir veritabanının bölünme veya parçalanma şekliyle yakından ilgilidir. Çoğu veri dağıtımı çalışması tek bir konuya odaklanır: hangi nereye yerleştirileceği.

Veri tahsis algoritmaları, aşağıdakiler de dahil olmak üzere çeşitli faktörleri göz önünde bulundurur:

- Performans ve veri kullanılabilirliği hedefleri
- Bir varlığın diğer varlıklarla sürdürdüğü ilişkilerin boyutu, satır sayısı ve sayısı
- Veritabanına uygulanacak işlem türleri ve bu işlemlerin her tarafından erişilen öznitelikler
- Mobil kullanıcılar için bağlantısız çalışma

Bazı durumlarda tasarım, özellikle sık güncelleme gerektirmeyen ve replika güncelleme pencerelerinin daha uzun olabileceği salt okunur veriler için mobil kullanıcılar için gevşek şekilde ayrılmış parçaların kullanılmasını düşünebilir. (Çoğaltma güncelleme penceresi, diğer görevlerle eş zamanlı olarak yürütülemeyen bir veri işleme görevini gerçekleştirmek için mevcut olan süredir).

Çoğu algoritma ağ topolojisi, ağ bant genişliği ve verimi, veri boyutu ve konum gibi bilgileri içerir. Henüz en uygun ya da evrensel olarak kabul görmüş bir algoritma mevcut değildir ve her veritabanı satıcısı kendi ürünlerinin güçlü yönlerini sergilemek için kendi versiyonunu uygulamaktadır.

veri tahsisi
Dağıtık bir DBMS'de, veri parçalarının nereye yerleştirileceğine karar verme süreci.

merkezi veri tahsisi
Tüm veritabanının tek bir yerde depolandığı bir veri tahsis stratejisi. *Merkezi veritabanı* olarak da bilinir.

bölümlere ayrılmış veri tahsisi
Bir veritabanını iki veya daha fazla yerde depolanan iki veya daha fazla parçaya bölen bir veri tahsis stratejisi.

çoğaltılmış veri tahsisi
Bir veya daha fazla kopyasının bulunduğu bir veri tahsis stratejisi daha fazla veritabanı parçası çeşitli yerlerde depolanır.

12-12 CAP Teoremi

Dağıtık bilgi işlem üzerine düzenlenen bir sempozyumda Dr. Eric Brewer sunumunda "yüksek oranda dağıtık veri sistemlerinde yaygın olarak istenen üç özellik vardır: tutarlılık, kullanılabilirlik ve bölünme toleransı. Ancak bir sistemin bu üç özelliği aynı sağlaması mümkün değildir."[2] *CAP* kelimelerinin baş harfleri arzu edilen üç özelliği temsil etmektedir. Bu üç özelliği daha ayrıntılı olarak ele alalım:

[2]Eric A. Brewer, "Towards robust distributed systems," Berkeley'deki California Üniversitesi ve Inktomi Corporation, Principles of Distributed Computing, ACM Symposium, Temmuz 2000'deki sunum. Bu teorem daha sonra MIT'den Seth Gilbert ve Nancy Lynch tarafından "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services" başlıklı makalelerinde kanıtlanmıştır, *ACM SIGACT News, 33*(2), 2002, s. 51-59.

- *Tutarlılık*. Dağıtık bir veritabanında tutarlılık daha büyük bir rol oynar. Tüm düğümler aynı anda aynı verileri görmelidir, bu da replikaların hemen güncellenmesi gerektiği anlamına gelir. Ancak bu, Bölüm 12-10'da öğrendiğiniz gibi gecikme ve ağ bölümleme gecikmeleriyle başa çıkmayı gerektirir.
- *Kullanılabilirlik*. Basitçe söylemek gerekirse, bir talep sistem tarafından her zaman yerine getirilir. Alınan hiçbir talep asla kaybolmaz. Online bilet satın alıyorsanız, sistemin işlemin ortasında istemezsiniz. Bu, tüm web merkezli kuruluşların en önemli gereksinimidir.
- *Bölünme toleransı*. Sistem, bir düğüm arızası durumunda bile çalışmaya devam eder. Bu, dağıtılmış veritabanlarındaki hata şeffaflığına eşdeğerdir (bkz. Bölüm 12-7). Sistem yalnızca tüm düğümler başarısız olursa başarısız olur.

İşlem yönetimi tutarlılığını (Bölüm 10'da öğrendiğiniz) CAP tutarlılığı ile karıştırmayın. İşlem yönetimi tutarlılığı, bir işlem yürütüldüğünde ortaya çıkan sonucun tüm bütünlük kısıtlamalarına uyan bir veritabanı verdiğini ifade eder. CAP'de tutarlılık, tüm işlemlerin sanki tek düğümlü bir veritabanında yürütülüyormuş gibi tüm düğümlerde aynı anda gerçekleştiği varsayımına dayanır. ("Tüm düğümler aynı anda aynı verileri görür.")

CAP teoremi yüksek düzeyde dağıtık web tabanlı sistemlere odaklansa da, veritabanları da dahil olmak üzere tüm dağıtık sistemler için yaygındır.

Bölüm 10'da dört veritabanı işlem özelliği olduğunu öğrendiniz: atomiklik, tutarlılık, izolasyon ve dayanıklılık. ACID özellikleri, tüm başarılı işlemlerin tutarlı bir veritabanı durumuyla sonuçlanmasını sağlar - tüm veri işlemlerinin her zaman aynı sonuçları döndürdüğü bir durum. Merkezi ve küçük dağıtılmış veritabanları için gecikme bir sorun değildir. İş büyüdükçe ve kullanılabilirlik ihtiyacı arttıkça, veritabanı gecikmesi daha büyük bir sorun haline gelir. Yüksek oranda dağıtılmış bir veritabanının, ağ gecikmesi veya veri çekişmesi (eşzamanlı veri erişiminin getirdiği gecikmeler) nedeniyle yüksek bir bedel ödemeden ACID işlemlerini sağlaması daha zordur.

Örneğin, Washington, D.C.'de Manchester United-Barcelona futbol maçı için bilet satın almak üzere Amazon.com'u kullandığınızı düşünün. Mevcut biletlere göz atmak ve hangi koltukların en iyi manzaraya sahip olduğunu görmek için stadyum web sitesini kontrol etmek için birkaç dakika harcayabilirsiniz. Aynı anda, dünyanın her yerinden diğer kullanıcılar da aynı şeyi yapıyor olabilir. Ödeme düğmesine tıkladığınızda, seçtiğiniz biletler başka biri tarafından çoktan satın alınmış olabilir! Bu durumda, yeniden başlayacak ve istediğinizi alana kadar diğer biletleri seçeceksiniz. Web sitesi bu şekilde çalışacak şekilde tasarlanmıştır çünkü Amazon, birkaç müşterinin işlemlerini yeniden başlatması gibi küçük bir olasılığı, tutarlılığı sağlamak için veritabanını kilitlemek ve binlerce müşteriyi web sayfalarının yenilenmesini beklemek zorunda bırakmaya tercih eder. Konser bileti satın almak için Ticketmaster'ı kullanırken küçük geri sayım saatini fark ettiyseniz, aynı prensibin işlediğini görmüşsünüzdür.

Bu örnekte de görüldüğü gibi, yüksek oranda dağıtılmış sistemlerle uğraşırken, bazı şirketler daha yüksek kullanılabilirlik elde etmek için ACID özelliklerinin tutarlılık ve izolasyon bileşenlerinden vazgeçme eğilimindedir. Tutarlılık ve kullanılabilirlik arasındaki bu değiş tokuş, verilerin **temelde kullanılabilir, yumuşak durumlu ve nihayetinde tutarlı (BASE)** olduğu yeni bir tür dağıtık veri sistemi ortaya çıkarmıştır. BASE, veri değişikliklerinin anında olmadığı, ancak tüm kopyalar sonunda tutarlı olana kadar sistemde yavaşça yayıldığı bir veri tutarlılığı modelini ifade eder. Örneğin, NoSQL veritabanları nihai tutarlılığa sahip yüksek oranda dağıtılmış bir veritabanı sağlar (bkz. Bölüm 14, Büyük Veri ve NoSQL). Buna karşılık, **NewSQL** veritabanları ilişkisel ve NoSQL veri modellerinin en iyilerini birleştirmeye çalışır. Örneğin, Google Cloud Spanner veri hizmeti, ACID işlemlerini destekleyen yüksek oranda ölçeklenebilir dağıtılmış veritabanları sağlar. Bu yeni veritabanı türü, rahat bölünme toleransı desteğiyle tutarlılık ve yüksek kullanılabilirlik sağlar. Uygulamada, NoSQL ve NewSQL dağıtılmış veritabanlarının ortaya çıkışı, Tablo 12.8'de gösterildiği gibi, yüksek tutarlılıktan (ACID) nihai tutarlılığa (BASE) kadar değişen bir tutarlılık yelpazesi sunmaktadır.

temelde mevcut, yumuşak durum, sonunda tutarlı (BASE)
Veri değişikliklerinin anında olmadığı, ancak tüm kopyalar sonunda tutarlı olana kadar sistemde yavaşça yayıldığı bir veri tutarlılığı modeli.

NewSQL
Oldukça dağıtık bir altyapıda ACID uyumlu işlemler sağlamaya çalışan bir veritabanı modeli.

Tablo 12.8 Dağıtılmış Veritabanı Spektrumu

DBMS Türü	Tutarlılık	Kullanılabilirlik	Bölme Toleransı	İşlem Modeli	Takas
Merkezi DBMS	Yüksek	Yüksek	N/A	ASİT	Dağıtık veri işleme yok
İlişkisel VTYS	Yüksek	Rahat	Yüksek	ASİT (2 ADET)	Tutarlılık ve izolasyon sağlamak için kullanılabilirlikten ödün verir
NoSQL DDBMS	Rahat	Yüksek	Yüksek	BAZ	Kullanılabilirliği sağlamak için tutarlılıktan ödün verir
NewSQL DDBMS	Yüksek	Yüksek	Rahat	ASİT	İşlem tutarlılığını ve kullanılabilirliğini sağlamak için bölüm toleransını feda eder

12-13 C. J. Date'in Dağıtık Veritabanları için 12 Emri

Dağıtık veritabanları kavramı uzun yıllardır var olan bir kavramdır. İlişkisel veritabanlarının yükselişiyle birlikte çoğu satıcı, genellikle kendi ürünlerinin güçlü yanlarını vurgulayarak kendi dağıtık veritabanı sürümlerini uygulamaya koymuştur. Karşılaştırmaları kolaylaştırmak için C. J. Date 12 "emir" ya da dağıtık veritabanlarının temel ilkelerini formüle etmiştir.[3] Mevcut hiçbir DDBMS bunların hepsine uymasa , bunlar faydalı bir hedef oluşturmaktadır. Bu 12 kural Tablo 12.9'da gösterilmektedir.

Tablo 12.9 C. J. Date'in Dağıtık Veritabanları için 12 Emri

Kural Numarası	Kural Adı	Kural Açıklaması
1	*Yerel saha bağımsızlığı*	Her yerel site bağımsız, özerk, merkezi bir DBMS olarak hareket edebilir. Her site güvenlik, eşzamanlılık kontrolü, yedekleme ve kurtarmadan sorumludur.
2	*Merkezi site bağımsızlığı*	Ağdaki hiçbir site merkezi bir siteye veya başka bir siteye bağlı değildir. Tüm siteler aynı yeteneklere sahiptir.
3	*Başarısızlık bağımsızlığı*	Sistem düğüm arızalarından etkilenmez. Bir düğüm arızası veya ağın genişlemesi durumunda bile sistem sürekli çalışır.
4	*Konum şeffaflığı*	Kullanıcının veriyi almak için verinin konumunu bilmesine gerek yoktur.
5	*Parçalanma şeffaflığı*	Veri parçalanması, yalnızca bir mantıksal veritabanı gören kullanıcı için şeffaftır. Kullanıcının bunları almak için veritabanı parçalarının adını bilmesine gerek yoktur.
6	*Çoğaltma şeffaflığı*	Kullanıcı yalnızca bir mantıksal veritabanı görür. DDBMS erişilecek veritabanı parçasını şeffaf bir şekilde seçer. Kullanıcı için DDBMS tüm parçaları şeffaf bir şekilde yönetir.
7	*Dağıtılmış sorgu işleme*	Dağıtılmış bir sorgu birkaç farklı DP sitesinde yürütülebilir. Sorgu optimizasyonu DDBMS tarafından şeffaf bir şekilde gerçekleştirilir.
8	*Dağıtılmış işlem işleme*	Bir işlem birkaç farklı sitedeki verileri güncelleyebilir ve işlem şeffaf bir şekilde yürütülür.
9	*Donanım bağımsızlığı*	Sistem herhangi bir donanım platformunda çalışmalıdır.
10	*İşletim sistemi bağımsızlığı*	Sistem herhangi bir işletim sistemi platformunda çalışmalıdır.
11	*Ağ bağımsızlığı*	Sistem herhangi bir ağ platformunda çalışmalıdır.
12	*Veritabanı bağımsızlığı*	Sistem, herhangi bir satıcının veritabanı ürününü desteklemelidir.

[3]C. J. Date, "Twelve rules for a distributed database," *Computerworld 2*(23), 8 Haziran 1987, s. 77-81.

Özet

- Dağıtılmış bir veritabanı, mantıksal olarak ilişkili verileri bir bilgisayar ağı aracılığıyla birbirine bağlı iki veya daha fazla fiziksel olarak bağımsız sitede depolar. Veritabanı, yatay bir satır kümesi veya dikey bir öznitelik kümesi olabilen parçalara bölünür. Her bir parça farklı bir ağ düğümüne tahsis edilebilir.
- Dağıtılmış işleme, mantıksal veri tabanı işlemenin iki veya daha fazla ağ düğümü arasında bölünmesidir. DağıtılmıĞ veritabanları dağıtılmıĞ iĞlem gerektirir. Dağıtılmış bir veritabanı yönetim sistemi (DDBMS), mantıksal olarak ilişkili verilerin birbirine bağlı bilgisayar sistemleri aracılığıyla işlenmesini ve depolanmasını yönetir.
- Bir DDBMS'nin ana bileşenleri işlem işlemcisi (TP) ve veri işlemcisidir (DP). İşlem işlemcisi bileşeni, veri talep eden her bilgisayar düğümündeki yerleşik yazılımdır. Veri işlemcisi bileşeni, her bilgisayarda bulunan ve verileri depolayan ve alan yerleşik yazılımdır.
- Mevcut veritabanı sistemleri, işleme ve veri dağıtımını ne ölçüde desteklediklerine göre sınıflandırılabilir. Dağıtık veritabanı sistemlerini sınıflandırmak için üç ana kategori kullanılmaktadır: tek site işleme, tek site veri (SPSD); çok site işleme, tek site veri (MPSD); ve çok site işleme, çok site veri (MPMD).
- Homojen dağıtılmış bir veritabanı sistemi, bir bilgisayar ağı üzerinden yalnızca belirli bir DBMS türünü entegre eder. Heterojen dağıtılmış bir veritabanı sistemi, bir bilgisayar ağı üzerinden birkaç farklı DBMS türünü entegre eder.
- DDBMS özellikleri en iyi şekilde bir dizi saydamlık olarak tanımlanır: dağıtım, işlem, performans, başarısızlık ve heterojenlik. Tüm saydamlıkların ortak amacı dağıtılmış veritabanını merkezi bir veritabanı sistemiymiş gibi davranır; yani son kullanıcı verileri tek, mantıksal bir merkezi veritabanının parçası olarak görür ve sistemin karmaşıklıklarından habersizdir.
- Bir işlem, bir veya daha fazla veritabanı isteği tarafından oluşturulur. Dağıtılmamış bir işlem tek bir siteden veri günceller veya talep eder. Dağıtılmış bir işlem birden fazla siteden veri güncelleyebilir veya talep edebilir.
- Dağıtılmış veritabanlarından oluşan bir ağda dağıtılmış eşzamanlılık kontrolü gereklidir. Bir işlemin tüm parçalarının tamamlandığından emin olmak için iki aşamalı bir COMMIT proto- colü kullanılır.
- Dağıtık bir DBMS, dağıtık bir veritabanında optimum erişim yolunu bulmak için her veri isteğini değerlendirir. DDBMS, ilişkili erişim maliyetlerini, iletişim maliyetlerini ve CPU maliyetlerini azaltmak için sorguyu optimize etmelidir.
- Dağıtık bir veritabanının tasarımcısı, verilerin parçalanmasını ve çoğaltılmasını göz önünde bulundurmalıdır. Tasarımcı ayrıca daha iyi genel yanıt süresi elde etmek ve son kullanıcı için veri kullanılabilirliğini sağlamak için her bir parçanın veya kopyanın nasıl tahsis edileceğine karar vermelidir. İdeal olarak, dağıtılmış bir veritabanı performansı, kullanılabilirliği ve konum farkındalığını en üst düzeye çıkarmak için verileri eşit olarak dağıtmalıdır.
- Bir veritabanı, bir bilgisayar ağı üzerindeki birkaç farklı site üzerinde çoğaltılabilir. Veritabanı parçalarının çoğaltılması, veri kullanılabilirliğini artırma ve böylece erişim süresini azaltma amacına sahiptir. Bir veritabanı kısmen, tamamen ya da hiç çoğaltılmayabilir. Veri tahsis stratejileri, veritabanı parçalarının veya kopyalarının konumunu belirlemek için tasarlanmıştır.
- CAP teoremi, yüksek düzeyde dağıtılmış bir veri sisteminin tutarlılık, kullanılabilirlik ve bölünme toleransı gibi bazı arzu edilen özelliklere sahip olduğunu belirtir. Ancak, bir sistem aynı anda bu özelliklerden yalnızca ikisini sağlayabilir.

Anahtar Terimler

uygulama işlemcisi (AP) temel
olarak kullanılabilir, yumuşak
durum,
Sonunda tutarlı (BASE)
merkezi veri tahsisi
istemci/sunucu mimarisi
koordinatörü
Veri tahsisi Veri
parçalanması
veri yöneticisi (DM)
veri işlemcisi (DP)
Veri çoğaltma
veritabanı parçaları
dağıtılmış veritabanı
dağıtılmış veri kataloğu (DDC)
dağıtılmış veri sözlüğü (DDD)
dağıtılmış veritabanı yönetim
sistemi (DDBMS)
dağıtılmış global şema
dağıtılmış işlem dağıtılmış
talep dağıtılmış işlem
Dağıtım şeffaflığı DO-
UNDO-REDO protokolü

Başarısızlık şeffaflığı
Parçalanma şeffaflığı tamamen
heterojen DDBMS tamamen
çoğaltılmış veritabanı
Heterojenlik şeffaflığı heterojen
DDBMS'ler Homojen
DDBMS'ler Yatay parçalanma
Yerel eşleme şeffaflığı Konum
şeffaflığı
Karışık parçalanma
çoklu saha işleme, çoklu saha verileri (MPMD)
çoklu saha işleme, tek saha verileri (MPSD)
karşılıklı tutarlılık kuralı
ağ gecikmesi ağ
bölümleme NewSQL
kısmen çoğaltılmış veritabanı
bölüm anahtarı
bölümlenmiş veri tahsisi
Performans şeffaflığı
uzaktan talep
uzak işlem Çoğaltma
şeffaflığı
çoğaltılmış veri tahsisi
tek sahada işleme, tek sahada veri (SPSD)
astlar
işlem yöneticisi (TM) işlem işlemcisi (TP)
İşlem şeffaflığı
iki aşamalı taahhüt protokolü (2PC)
benzersiz parça
çoğaltılmamış
veritabanı Dikey
parçalama ileriye
yazma protokolü

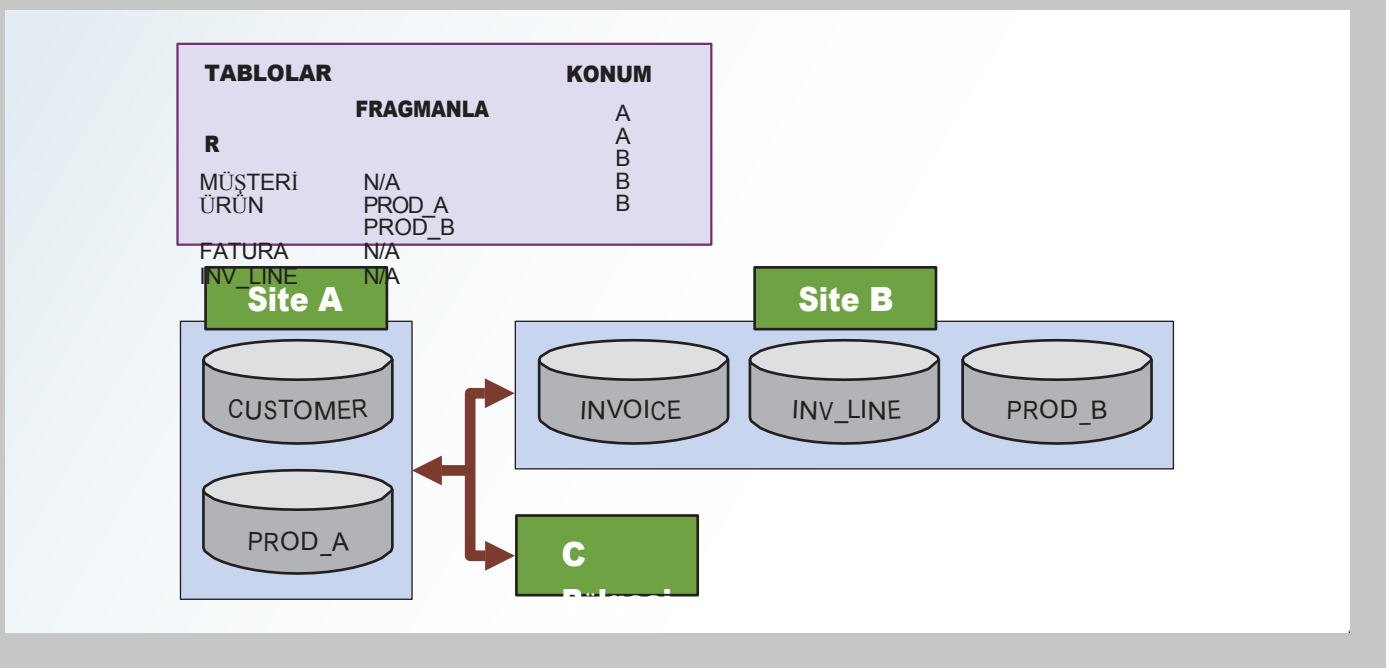
İnceleme Soruları

1. Merkezi VTYS'lerden dağıtık VTYS'lere evrimi açıklayınız.
2. DDBMS'nin evrimini etkileyen bazı faktörleri listeleyiniz ve tartışınız.
3. DDBMS'nin avantajları nelerdir?
4. DDBMS'nin dezavantajları nelerdir?
5. Dağıtık veritabanı ve dağıtık işleme arasındaki farkı açıklayınız.
6. Tamamen dağıtılmış bir veritabanı yönetim sistemi nedir?
7. Bir DDBMS'nin bileşenleri nelerdir?
8. Bir DDBMS'nin şeffaflık özelliklerini listeleyiniz ve açıklayınız.
9. Farklı dağıtım şeffaflığı türlerini tanımlamak ve açıklamak.
10. Farklı veritabanı istek ve işlem türlerini tanımlayabilecektir.
11. İki aşamalı taahhüt protokolüne duyulan ihtiyacı açıklayın. Ardından iki aşamayı açıklayın.
12. Sorgu optimizasyon fonksiyonlarının amacı nedir?
13. Sorgu optimizasyon fonksiyonları hangi şeffaflık özelliği ile ilgilidir?
14. Dağıtılmış bir veri ortamında veri talepleri çözümlenirken hangi hususlar dikkate alınmalıdır?
15. Üç veri parçalama stratejisini tanımlayın. Her birine bazı örnekler verin.
16. Veri replikasyonu nedir ve üç replikasyon stratejisi nelerdir?
17. Veri replikasyonunun iki temel tarzı nedir?
18. Yüksek düzeyde dağıtılmış veri ortamları oluştururken hangi ödünleşmeler söz konusudur?
19. Bir BASE sisteminin geleneksel bir dağıtık veritabanı sisteminden farkı nedir?
20. NewSQL veritabanları tutarlılık, kullanılabilirlik ve bölünme toleransı açısından NoSQL veritabanlarına kıyasla nasıldır?

Problemler

Problem 1, Şekil P12.1'deki DDBMS senaryosuna dayanmaktadır.

Şekil P12.1 Problem 1 için DDBMS Senaryosu



1. Aşağıdaki işlemleri gerçekleştirmek için veritabanının desteklemesi gereken minimum işlem türlerini belirtin. Bu arasında uzak istekler, uzak işlemler, dağıtılmış işlemler ve dağıtılmış istekler yer alır.

   C Sahasında

   a.
   ```
   SEÇİNİZ      *
   FROM         MÜŞTERİ;
   ```

   b.
   ```
   SEÇİNİZ      *
   FROM         FATURA
   NEREDE       INV_TOT< 1000;
   ```

   c.
   ```
   SEÇİNİZ      *
   FROM         ÜRÜN
   NEREDE       PROD_ QOH< 10;
   ```

   d.
   ```
   İŞE BAŞLA;
   GÜNCELLEME   MÜŞTERİ
   SET          CUS_BAL 5 CUS_BAL 1 100
   NEREDE       CUS_NUM 5 '10936';
   INSERT       INTO INVOICE(INV_NUM, CUS_NUM, INV_DATE, INV_TOTAL)
                VALUES ('986391', '10936', '2022-02-15', 100);
   INSERT       INTO LINE(INV_NUM, PROD_NUM, LINE_PRICE)
                VALUES('986391', '1023', 100);
   GÜNCELLEME   ÜRÜN
   SET          PROD_QOH 5 PROD_ QOH -1
   NEREDE       PROD_NUM 5 '1023';
   COMMIT WORK;
   ```

e. İŞE BAŞLA;

```
INSERT      INTO CUSTOMER(CUS_NUM, CUS_NAME, CUS_ADDRESS, CUS_BAL)
            VALUES ('34210', 'Victor Ephanor', '123 Main St.', 0.00);
INSERT      INTO INVOICE(INV_NUM, CUS_NUM, INV_DATE, INV_TOTAL)
            VALUES ('986434', '34210', '2022-08-10', 2.00);
İŞİNİ YAP;
```

A Sahasında

f.
```
SEÇİNİZ     CUS_NUM, CUS_NAME, INV_TOTAL
FROM        MÜŞTERI, FATURA
NEREDE      CUSTOMER.CUS_NUM 5 INVOICE.CUS_NUM;
```

g.
```
SEÇİNİZ     *
FROM        FATURA
NEREDE      INV_TOTAL> 1000;
```

h.
```
SEÇİNİZ     *
FROM        ÜRÜN
NEREDE      PROD_QOH< 10;
```

B Sahasında

i.
```
SEÇİNİZ     *
FROM        MÜŞTERİ;
```

j.
```
SEÇİNİZ     CUS_NAME, INV_TOTAL
FROM        MÜŞTERI, FATURA
NEREDE      INV_TOTAL > 1000 VE CUSTOMER.
            CUS_NUM 5 INVOICE.CUS_NUM;
```

k.
```
SEÇİNİZ     *
FROM        ÜRÜN
NEREDE      PROD_QOH< 10;
```

2. Bir dergi yayıncılık şirketi için aşağıdaki veri yapısı ve kısıtlamalar mevcuttur:

   a. Şirket dört eyaletin her birinde birer bölgesel dergi yayınlamaktadır: Florida (FL), Güney Carolina (SC), Georgia (GA) ve Tennessee (TN).

   b. Şirketin Problem 2a'da listelenen dört eyalete dağılmış 300.000 müşterisi (abonesi) vardır.

   c. Her ayın ilk günü, yıllık abonelik FATURASI basılır ve aboneliğinin yenilenmesi gereken her müşteriye gönderilir. INVOICE varlığı, müşterinin ikamet ettiği eyaleti (FL, SC, GA, TN) belirtmek için bir REGION özniteliği içerir:

   MÜŞTERI (CUS_NUM, CUS_NAME, CUS_ADDRESS, CUS_CITY, CUS_ZIP, CUS_SUBSDATE)

   FATURA (INV_NUM, INV_REGION, CUS_NUM, INV_DATE, INV_TOTAL)

   Şirket, merkezi yönetimle ilgili sorunların farkındadır ve aboneliklerin yönetimini şirketin dört bölgesel iştirakine dağıtmaya karar vermiştir. Her abonelik sitesi kendi müşteri ve fatura verilerini idare edecektir. Ancak şirket merkezindeki yönetim, yıllık raporlar oluşturmak ve aşağıdaki gibi geçici sorgular düzenlemek için müşteri ve fatura verilerine erişebilecektir:

   - Tüm mevcut müşterilerin bölgeye göre listelenmesi
   - Tüm yeni müşterilerin bölgeye göre listelenmesi
   - Tüm faturaların müşteri ve bölge bazında raporlanması

   Bu gereksinimler göz önüne alındığında, veritabanını nasıl bölümlendirmelisiniz?

3. Problem 2'deki senaryo ve gereksinimleri göz önünde bulundurarak aşağıdaki soruları yanıtlayınız:
   a. Gerekli veritabanı sisteminin türü ve özelliklerine ilişkin ne gibi önerilerde bulunacaksınız?
   b. Her tablo için ne tür veri parçalanması gereklidir?
   c. Her bir veritabanını bölümlemek için hangi kriterler kullanılmalıdır?
   d. Veritabanı parçalarını tasarlayın. Düğüm adları, konum, parça , öznitelik adları ve gösterim verilerini içeren bir örnek gösterin.
   e. Her bir uzak sitede ne tür dağıtılmış veritabanı işlemleri desteklenmelidir?
   f. Genel merkez tesisinde ne tür dağıtılmış veritabanı işlemleri desteklenmelidir?