

Bölüm 4

**İleri Veritabanı Kavramları**

1. İşlem Yönetimi ve Eşzamanlılık Kontrolü
2. Veritabanı Performans Ayarlama ve Sorgu Optimizasyonu
3. Dağıtık Veritabanı Yönetim Sistemleri
4. İş Zekası ve Veri Ambarları
5. Büyük Veri ve NoSQL

**Bölüm**

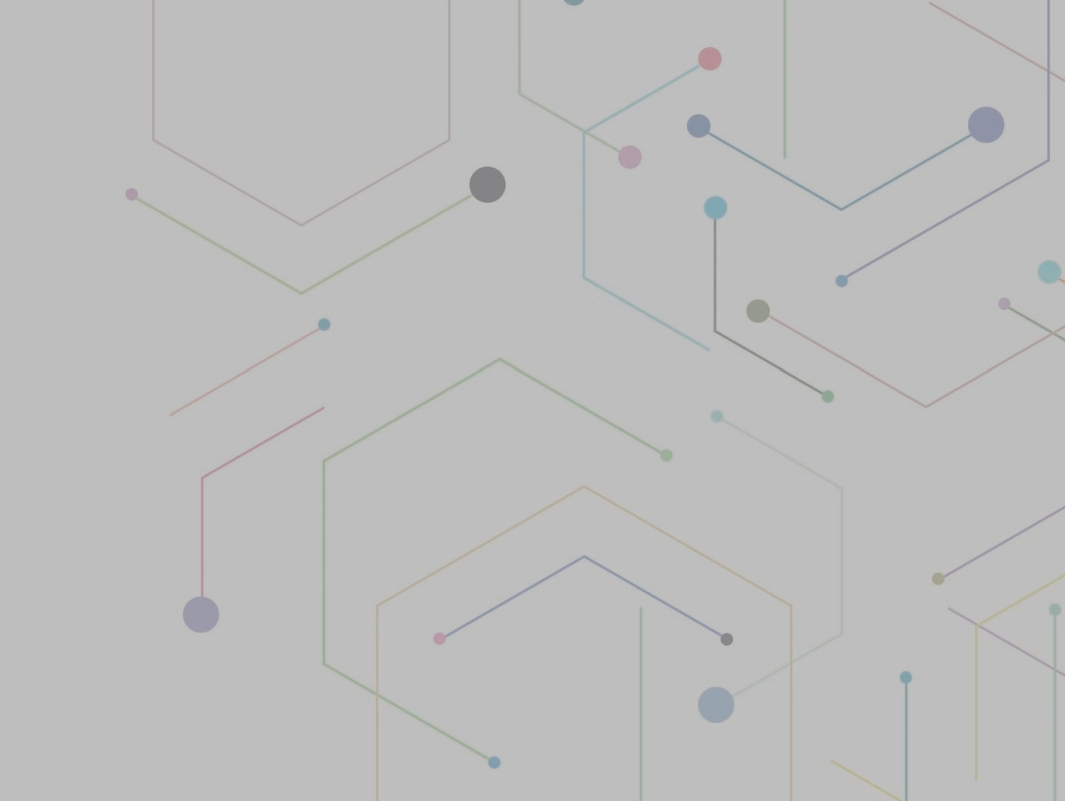
10

İşlem Yönetimi ve Eşzamanlılık Kontrolü

**Öğrenme Hedefleri**

Bu bölümü tamamladıktan sonra şunları yapabileceksiniz:

* 1. **Veritabanı işlem yönetimi sürecini tanımlama**
  2. **Bir veritabanı işleminin dört özelliğini tanımlama**
  3. **Eşzamanlılık kontrolünü ve veritabanı bütünlüğünü korumadaki rolünü açıklama**
  4. **Kilitleme yöntemlerinin eşzamanlılık kontrolü için nasıl kullanıldığını açıklayın**
  5. **Damgalama yöntemlerinin eşzamanlılık kontrolü için nasıl kullanıldığını açıklayın**
  6. **İyimser yöntemlerin eşzamanlılık kontrolü için nasıl kullanıldığını açıklayın**
  7. **ANSI işlem yalıtımı seviyelerini listeleyin açıklayın**
  8. **Veritabanı bütünlüğünün korunmasında veritabanı kurtarma yönetiminin rolünü tanımlama**



**Önizleme**

Veritabanı işlemleri, bir ürün satın almak, bir kursa kaydolmak veya bir çek hesabına para yatırmak gibi olaylar tarafından tetiklenen gerçek dünya işlemlerini yansıtır. İşlemlerin, bir müşterinin hesabının güncellenmesi, ürün envanterinin ayarlanması ve satıcının alacak hesaplarının güncellenmesi gibi birçok adım içermesi muhtemeldir.

Veri bütünlüğü sorunlarını önlemek için bir işlemin tüm parçaları başarıyla tamamlanmalıdır. Bu nedenle, işlemlerin yürütülmesi ve yönetilmesi önemli veritabanı sistemi faaliyetleridir.

Bu bölümde, veritabanı işlemlerinin temel özellikleri hakkında bilgi edineceksiniz: atomiklik, tutarlılık, izolasyon ve dayanıklılık. İşlem özelliklerini tanımladıktan sonra, eş zamanlı işlemlerin veritabanı tutarlılığını ve bütünlüğünü korumasını sağlamak için serileştirilebilirliğin önemini öğreneceksiniz. Ardından, SQL'in işlemleri temsil etmek için nasıl kullanılabileceğini ve işlem günlüklerinin VTYS'nin işlemleri kurtarma yeteneğini nasıl sağlayabileceğini öğreneceksiniz.

Birçok işlem aynı anda gerçekleştiğinde, bunlara *eşzamanlı* işlemler denir. Bu tür işlemlerin yürütülmesini yönetmeye *eşzamanlılık kontrolü* denir. Bu bölümde, eşzamanlı işlemlerde ortaya çıkabilecek bazı sorunlar (kayıp güncellemeler, taahhüt edilmemiş veriler ve tutarsız geri alımlar) ve

eşzamanlılık kontrolü için en yaygın algoritmalar: kilitler, zaman damgası ve iyimser yöntemler. Son olarak, veritabanı kurtarma yönetiminin bir donanım veya yazılım arızası durumunda veritabanının içeriğinin geçerli bir tutarlı duruma geri yüklenmesini nasıl sağlayabileceğini göreceksiniz.

**Veri Dosyaları ve Mevcut Formatlar**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MS Erişim** | | **Oracle** | **MS SQL** | **MySQL** |
| Ch10\_SaleCo | Evet | Evet | Evet | Evet |
| Ch10\_ABC\_Markets | Evet | Evet | Evet | Evet |

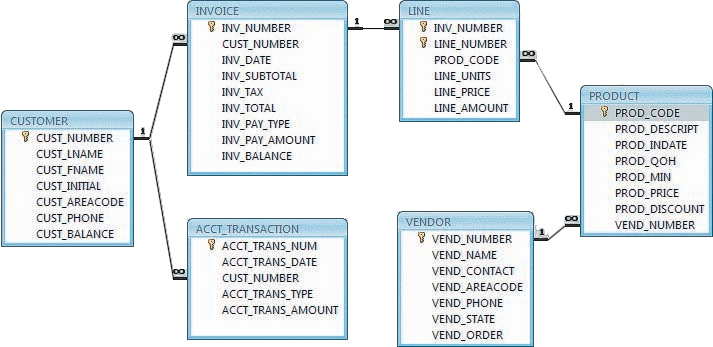
Veri Dosyaları cengage.com adresinde mevcuttur

# İşlem Nedir?

İşlemlerin ne olduğunu ve nasıl çalıştığını göstermek için Ch10\_SaleCo veritabanını kullanın. Veritabanı için

ilişkisel diyagram Şekil 10.1'de gösterilmektedir.

**Şekil 10.1 Ch10\_SaleCo Veritabanı İlişkisel Diyagramı**



**Not**

SQL komutları çeşitli işlem ve eşzamanlılık kontrolü konularını açıklasa da, Bölüm 7, Yapısal Sorgu Diline (SQL) Giriş ve Bölüm 8, Gelişmiş SQL'i çalışmamış olsanız bile tartışmaları takip edebilmeniz gerekir. SQL bilmiyorsanız, SQL komutlarını görmezden gelin ve tartışmalara odaklanın. SQL bilginiz varsa, Ch10\_SaleCo veritabanını kullanarak kendi SELECT ve UPDATE örneklerinizi oluşturabilir ve kendi tetikleyicilerinizi ve saklı yordamlarınızı yazarak Bölüm 7 ve 8'deki materyali geliştirebilirsiniz.

Şekil 10.1'deki ilişkisel diyagramı incelerken aşağıdaki özelliklere dikkat edin:

* Tasarım, müşterinin borçlu olduğu toplam tutarı göstermek için müşteri bakiyesi (CUST\_BALANCE) değerini CUSTOMER tablosunda saklar (bunun türetilmiş bir nitelik olduğunu Bölüm 4'ten hatırlayın). CUST\_BALANCE özniteliği, müşteri kredili bir satın alma yaptığında artar ve müşteri bir ödeme yaptığında azalır. Mevcut müşteri hesap bakiyesinin CUSTOMER tablosuna dahil edilmesi, herhangi bir müşterinin mevcut bakiyesini belirlemek ve toplam, ortalama, minimum ve maksimum bakiyeler gibi önemli özetler oluşturmak için bir sorgu yazmayı kolaylaştırır.
* ACCT\_TRANSACTION tablosu, müşteri hesabı faaliyetlerinin ayrıntılarını izlemek için tüm

müşteri satın alımlarını ve ödemelerini kaydeder.

Muhasebe uygulamalarını daha doğru yansıtmak için Ch10\_SaleCo veritabanının tasarımını değiştirebilirsiniz, ancak burada sağlanan uygulama, bölümün tartışmalarını anlamak için işlemleri yeterince iyi izlemenizi sağlayacaktır.

İşlem kavramını anlamak için, bir müşteriye bir ürün sattığınızı varsayalım. Ayrıca, müşterinin satın aldığı ürünü kendi hesabından tahsil edebileceğini varsayalım. Bu senaryo göz önüne alındığında, satış işleminiz en azından aşağıdaki bölümlerden oluşur:

* Yeni bir müşteri faturası yazmalısınız.
* Ürünün envanterindeki eldeki miktarı azaltmanız gerekir.
* Hesap işlemlerini güncellemeniz gerekir.
* Müşteri bakiyesini güncellemeniz gerekir.

Önceki satış işlemi veritabanına yansıtılmalıdır. Veritabanı açısından bir işlem, bir veritabanından okuyan veya veritabanına yazan herhangi bir eylemdir. Bir işlem aşağıdakilerden oluşabilir:

* Tablo içeriklerinin bir listesini oluşturmak için basit bir SELECT deyimi.
* Çeşitli tablolardaki özniteliklerin değerlerini değiştirmek için bir dizi ilgili UPDATE deyimi.
* Bir veya daha fazla tabloya satır eklemek için bir dizi INSERT deyimi.
* SELECT, UPDATE ve INSERT deyimlerinin bir kombinasyonu.

Satış işlemi örneği INSERT ve UPDATE deyimlerinin bir kombinasyonunu içerir. Önceki

tartışma göz önüne alındığında, bir işlemin tanımını artırabilirsiniz.

Bir **işlem**, tamamen tamamlanması veya tamamen iptal edilmesi gereken *mantıksal* bir iş birimidir; hiçbir ara durum kabul edilemez. Başka bir deyişle, daha önce bahsedilen satış gibi çok bileşenli bir işlem kısmen . Yalnızca envanterin veya yalnızca alacak hesaplarının güncellenmesi kabul edilemez. İşlemdeki tüm SQL deyimleri başarıyla tamamlanmalıdır. SQL deyimlerinden herhangi biri başarısız olursa, tüm işlem, işlem başlamadan önce var olan orijinal veritabanı durumuna geri döndürülür. Başarılı bir işlem, veritabanını tutarlı bir durumdan diğerine değiştirir. **Tutarlı** bir **veritabanı durumu**, tüm veri bütünlüğü kısıtlamalarının karşılandığı bir durumdur.

#### işlem

Veritabanına erişen bir dizi veritabanı isteği. Bir işlem mantıksal bir iş birimidir; yani, *tamamen* tamamlanmalı veya iptal edilmelidir - hiçbir ara bitiş durumu kabul edilmez. Tüm işlemler atomiklik, tutarlılık, izolasyon ve dayanıklılık özelliklerine sahip olmalıdır.

#### tutarlı veritabanı durumu

Tüm veri bütünlüğü kısıtlamalarının karşılandığı bir veritabanı durumu.

#### veritabanı talebi

Bir uygulama programındaki veya bir işlemdeki tek bir SQL deyiminin eşdeğeri.

Veritabanının tutarlılığını sağlamak için, her işlem veritabanı bilinen tutarlı bir durumdayken başlamalıdır. Veritabanı tutarlı bir durumda değilse, işlem bütünlüğünü ve iş kurallarını ihlal eden tutarsız bir veritabanı ortaya çıkaracaktır. Bu nedenle, daha sonra ele alınacak sınırlamalara tabi olarak, veritabanı bütünlüğünü garanti etmek için tüm işlemler VTYS tarafından kontrol edilir ve yürütülür.

Gerçek dünyadaki veritabanı işlemlerinin çoğu iki veya daha fazla veritabanı isteğinden oluşur. Bir **veritabanı isteği**, bir uygulama programındaki veya işlemdeki tek bir SQL deyimine eşdeğerdir. Örneğin, bir işlem iki UPDATE deyimi ve bir INSERT deyiminden oluşuyorsa, işlem üç veritabanı isteği kullanır. Buna karşılık, her veritabanı isteği fiziksel depolama ortamından okuyan veya fiziksel depolama ortamına yazan birkaç giriş/çıkış (I/O) işlemi oluşturur.

## a İşlem Sonuçlarının Değerlendirilmesi

Tüm işlemler veritabanını güncellemez. Diyelim ki 10016 numaralı müşterinin mevcut bakiyesini belirlemek için CUSTOMER tablosunu incelemek istiyorsunuz. Böyle bir işlem aşağıdaki SQL kodu kullanılarak tamamlanabilir:

SEÇİNİZ CUST\_NUMBER, CUST\_BALANCE FROM MÜŞTERİ

NEREDE CUST\_NUMBER 5 10016;

Sorgu CUSTOMER tablosunda herhangi bir değişiklik yapmamasına rağmen, SQL kodu veritabanına *eriştiği* için bir işlemi temsil eder. Veritabanı erişimden önce tutarlı bir durumdaysa, işlem veritabanını değiştirmediği için veritabanı erişimden sonra tutarlı bir durumda kalır.

Bir işlemin tek bir SQL deyiminden veya ilgili SQL deyimleri koleksiyonundan oluşabileceğini unutmayın. Ch10\_SaleCo veritabanını kullanarak daha karmaşık bir işlemi göstermek için önceki satış örneğini tekrar gözden geçirin. Diyelim ki 18 Ocak 2022 tarihinde, 89-WRE-Q ürününün bir biriminin 10016 numaralı müşteriye 277,55 $ karşılığında kredili satışını kaydediyorsunuz. Gerekli işlem INVOICE, LINE, PRODUCT, CUSTOMER ve ACCT\_TRANSACTION tablolarını etkiler. Bu işlemi temsil eden SQL deyimleri aşağıdaki gibidir:

FATURAYA EKLE

VALUES (1009, 10016, '18-Jan-2022', 256.99, 20.56, 277.55, 'cred', 0.00, 277.55); INSERT

INTO LINE

DEĞERLER (1009, 1, '89-WRE-Q', 1, 256,99, 256,99);

GÜNCELLEME ÜRÜN

SET PROD\_QOH 5 PROD\_QOH 2 1 BURADA PROD\_CODE 5 '89-WRE-Q';

GÜNCELLEME MÜŞTERİ

SET CUST\_BALANCE 5 CUST\_BALANCE 1 277,55

NEREDE CUST\_NUMBER 5 10016;

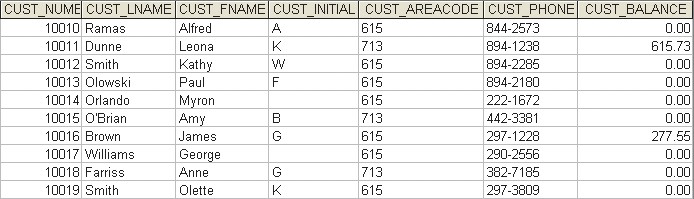
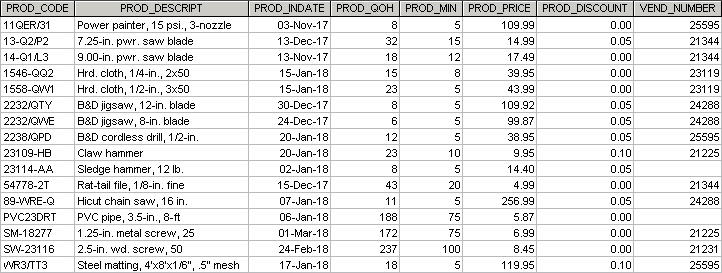
INSERT INTO ACCT\_TRANSACTION

VALUES (10007, '18-Jan-22', 10016, 'charge', 277.55);

TAAHHÜT;

Başarıyla tamamlanan işlemin sonuçları Şekil 10.2'de gösterilmektedir (İşleme dahil olan tüm kayıtlar kırmızı renkle belirtilmiştir).

**Şekil 10.2 Ch10\_SaleCo Veritabanındaki İşlemin İzlenmesi**



Tablo adı: INVOICE

Veritabanı adı: Ch10\_SaleCo Tablo adı: LINE

Tablo adı: ÜRÜN

Tablo adı: CUSTOMER

Tablo adı: ACCT\_TRANSACTION

İşlem sonuçlarını daha iyi anlamak için aşağıdakilere dikkat edin:

* INVOICE tablosuna yeni bir 1009 satırı eklendi. Bu satırda, fatura ara toplamı, vergi, fatura

toplamı ve fatura bakiyesi için türetilmiş öznitelik değerleri saklandı.

* 1009 numaralı fatura için SATIR satırı, bir birim 89-WRE-Q ürününün 256,99 $ fiyatla satın alındığını yansıtacak şekilde eklenmiştir. Bu satırda, satır tutarı için türetilmiş öznitelik değerleri saklandı.
* ÜRÜN tablosundaki 89-WRE-Q ürününün eldeki miktarı (PROD\_QOH) bir azaltılarak

12'den 11'e düşürüldü.

* Müşteri 10016'ya ait müşteri bakiyesi (CUST\_BALANCE) eklenerek güncellendi Mevcut bakiyeye 277,55 $ eklenmiştir (başlangıç değeri 0,00 $ idi).
* Yeni hesap işlem numarası 10007'yi yansıtmak için ACCT\_TRANSACTION tablosuna yeni bir

satır eklendi.

* COMMIT deyimi başarılı bir işlemi sonlandırmak için kullanılmıştır. (Bkz. Bölüm 10-1c.)

Şimdi VTYS'nin ilk üç SQL deyimini tamamladığını varsayalım. Ayrıca, dördüncü deyimin (10016 numaralı müşteri için CUSTOMER tablosunun CUST\_BALANCE değerinin GÜNCELLENMESİ) yürütülmesi sırasında bilgisayar sisteminin elektrik gücünün kesildiğini varsayın. Bilgisayarın yedek bir güç kaynağı yoksa işlem tamamlanamaz. Bu nedenle, FATURA ve SATIR satırları eklendi ve ÜRÜN tablosu 89-WRE-Q ürününün satışını temsil edecek şekilde güncellendi, ancak 10016 numaralı müşteriden ücret alınmadı ve ACCT\_TRANSACTION tablosuna gerekli kayıt yazılmadı. Veritabanı artık tutarsız bir durumdadır ve sonraki işlemler için kullanılamaz. DBMS'nin işlem yönetimini desteklediğini varsayarsak, *DBMS veritabanını önceki tutarlı duruma geri döndürecektir*.

DBMS, bir kesinti bir işlemin tamamlanmasını engellediğinde veritabanını önceki tutarlı durumuna geri döndürmek üzere tasarlanmış olsa da, işlemin kendisi son kullanıcı veya programcı tarafından tanımlanır ve anlamsal olarak doğru olmalıdır. *VTYS, işlemin semantik anlamının gerçek dünyadaki olayı gerçekten temsil ettiğini garanti edemez*. Örneğin, 10 birim 89-WRE-Q ürününün satışının ardından envanter GÜNCELLEME komutlarının şu şekilde yazıldığını varsayalım:

**Not**

Varsayılan olarak, MS Access burada tartışıldığı gibi işlem yönetimini destekler. Oracle, SQL Server ve DB2 gibi daha sofistike DBMS'ler de bu bölümde tartışılan işlem yönetimi bileşenlerini destekler. MS Access, işlem yönetimini kendi yerel JET motoru, harici bir DBMS'ye ODBC arabirimi veya Access Data Objects (ADO) bileşenleri aracılığıyla destekler (daha fazla bilgi için bkz. Bölüm 15, Veritabanı Bağlantısı ve Web Teknolojileri).

#### atomiklik

Bir tüm parçalarının tek, bölünmez, mantıksal bir iş birimi olarak ele alınmasını gerektiren işlem özelliğiBir işlemin tüm parçaları tamamlanmalıdır, aksi takdirde işlemin tamamı iptal edilir.

#### tutarlılık

Tüm veri bütünlüğü kısıtlamalarının karşılandığı veritabanı durumu. Bir veritabanının sağlamak için her

işlem, veritabanı bilinen tutarlı bir durumdayken başlamalıdır. Aksi takdirde, işlem bütünlüğünü ve iş kurallarını ihlal eden tutarsız bir veritabanı ortaya çıkaracaktır.

GÜNCELLEME ÜRÜN

SET PROD\_QOH 5 PROD\_QOH 1 10 BURADA PROD\_CODE 5 '89-WRE-Q';

Satış, 89-WRE-Q ürünü için PROD\_QOH değerini 10 *azaltmalıydı*.

Bunun yerine, UPDATE 89-WRE-Q ürününün PROD\_QOH değerine 10 *eklemiştir*.

UPDATE komutunun sözdizimi doğru olmasına rağmen, kullanımı yanlış sonuçlar, yani gerçek dünyadaki olayla tutarsız bir veritabanı verir. Yine de, DBMS işlemi yine de yürütecektir. VTYS, işlemin gerçek dünya olayını doğru bir şekilde temsil edip etmediğini değerlendiremez; bu son kullanıcının sorumluluğundadır. Son kullanıcılar ve programcılar bu şekilde pek çok hataya neden olabilirler. 89-WRE-Q ürünü yerine 1546-QQ2 ürünü için eldeki miktarı azaltmanın veya 10016 müşterisi yerine 10012 müşterisi için CUST\_BALANCE değerini alacaklandırmanın sonuçlarını düşünün.

Uygunsuz veya eksik işlemlerin veritabanı bütünlüğü üzerinde yıkıcı bir etkisi olabileceği açıktır. Bazı VTYS'ler, *özellikle* de ilişkisel türdekiler, kullanıcının iş kurallarına dayalı olarak uygulanabilir kısıtlamalar tanımlayabileceği araçlar sağlar. Referans ve varlık bütünlüğünü yönetenler gibi diğer bütünlük kuralları, tablo yapıları uygun şekilde tanımlandığında DBMS tarafından otomatik olarak uygulanır ve böylece DBMS'nin bazı işlemleri doğrulamasına izin verir. Örneğin, bir işlem müşteri tablosuna yeni bir müşteri numarası eklerse ve numara zaten mevcutsa, DBMS birincil anahtar bütünlüğü kuralının ihlal belirtmek için işlemi bir hata koduyla sonlandıracaktır.

## b İşlem Özellikleri

Her bir işlem *atomiklik*, *tutarlılık*, *izolasyon* ve *dayanıklılık* göstermelidir. Bu dört özellik bazen ACID testi olarak da adlandırılır. Şimdi her bir özelliğe kısaca göz atalım.

* **Atomiklik**, bir işlemin *tüm* işlemlerinin (SQL istekleri) tamamlanmasını gerektirir; işlem iptal edilir. Bir T1 işleminin dört SQL isteği varsa, dört isteğin de başarıyla tamamlanması gerekir; aksi takdirde, işlemin tamamı iptal edilir. Başka bir deyişle, bir işlem tek, bölünmez, mantıksal bir iş birimi olarak ele alınır.
* **Tutarlılık**, veritabanının tutarlı durumunun kalıcılığını gösterir. Bir işlem, bir veritabanını bir tutarlı durumdan diğerine götürür. Bir işlem tamamlandığında, veritabanı tutarlı bir durumda olmalıdır. İşlem parçalarından herhangi biri bir bütünlük kısıtlamasını ihlal ederse, işlemin tamamı iptal edilir.
  + **İzolasyon**, bir işlemin yürütülmesi sırasında kullanılan verilerin, ilk işlem tamamlanana kadar ikinci bir işlem tarafından kullanılanlamına gelir. Başka bir deyişle, T1 işlemi yürütülüyorsa ve X veri öğesini kullanıyorsa, bu veri öğesine T1 sona erene kadar başka bir işlem (T2 ... *Tn*) tarafından erişilemez. Bu özellik özellikle çok kullanıcılı veritabanı ortamlarında kullanışlıdır çünkü birden fazla kullanıcı aynı anda veritabanına erişebilir ve veritabanını güncelleyebilir.
  + **Dayanıklılık**, işlem değişiklikleri yapıldıktan ve işlendikten sonra, bir sistem arızası

durumunda bile geri alınamamasını veya kaybolmamasını sağlar.

Yukarıda belirtilen bireysel işlem özelliklerine ek olarak, aynı anda birden fazla işlem yürütülürken bir başka önemli özellik daha geçerlidir. Örneğin, VTYS'de aynı anda üç işlemin (T1, T2 ve T3) yürütüldüğünü varsayalım. İşlemleri düzgün bir şekilde yürütmek için, DBMS'nin işlemlerin eşzamanlı yürütülmesini planlaması gerekir. Bu durumda, her bir işlem ACID özelliklerine uygun olmalı ve aynı zamanda bu tür çoklu işlemlerin çizelgesi serileştirilebilirlik özelliği göstermelidir. **Serileştirilebilirlik**, işlemlerin eşzamanlı yürütülmesine ilişkin çizelgenin tutarlı sonuçlar vermesini sağlar. Bu özellik, birden fazla işlemin eşzamanlı olarak yürütülmesinin muhtemel olduğu çok kullanıcılı ve dağıtık veritabanlarında önemlidir. Doğal olarak, yalnızca tek bir işlem yürütülüyorsa, serileştirilebilirlik bir sorun değildir.

Tek kullanıcılı bir veritabanı sistemi otomatik olarak veri tabanının serileştirilebilirliğini ve izolasyonunu sağlar çünkü bir seferde yalnızca bir işlem yürütülür. İşlemlerin atomikliği, tutarlılığı ve sürekliliği tek kullanıcılı VTYS'ler tarafından garanti edilmelidir. (Tek kullanıcılı bir VTYS bile işletim sistemi kaynaklı kesintiler, güç kesintileri ve anormal uygulama sonlandırmaları veya çökmeleri nedeniyle oluşan hatalardan kurtarmayı yönetmelidir).

Çok kullanıcılı veritabanları tipik olarak birden fazla eş zamanlı işleme tabidir. Bu nedenle, çok VTYS, veritabanının tutarlılığını ve bütünlüğünü korumak için atomiklik ve dayanıklılığa ek olarak işlemlerin serileştirilebilirliğini ve izolasyonunu sağlamak için kontroller uygulamalıdır. Örneğin, aynı veri kümesi üzerinde birden fazla eşzamanlı işlem yürütülürse ve ikinci işlem ilk işlem tamamlanmadan önce veritabanını güncellerse, izolasyon özelliği ihlal edilmiş olur ve veritabanı artık tutarlı değildir. DBMS, istenmeyen durumlardan kaçınmak için eşzamanlılık kontrol tekniklerini kullanarak işlemleri yönetmelidir.

## c SQL ile İşlem Yönetimi

Amerikan Ulusal Standartlar Enstitüsü (ANSI) SQL veritabanı işlemlerini yöneten standartları tanımlamıştır. İşlem desteği iki SQL deyimi tarafından sağlanır: COMMIT ve ROLLBACK. ANSI standartları, bir kullanıcı veya uygulama programı tarafından bir işlem dizisi başlatıldığında, dizinin aşağıdaki dört olaydan biri gerçekleşene kadar birbirini izleyen tüm SQL deyimleri aracılığıyla devam etmesini gerektirir:

* + Bir COMMIT deyimine ulaşıldığında, tüm değişiklikler veritabanına kalıcı olarak kaydedilir. COMMIT deyimi SQL işlemini otomatik olarak sonlandırır.
  + Bir ROLLBACK ifadesine ulaşıldığında tüm değişiklikler iptal edilir ve veri tabanı önceki

tutarlı durumuna geri döndürülür.

* + Bir programın sonuna başarıyla ulaşıldığında, bu durumda tüm değişiklikler veritabanına kalıcı olarak kaydedilir. Bu eylem COMMIT ile eşdeğerdir.
  + Program anormal bir şekilde sonlandırılır, bu durumda veritabanı değişiklikleri iptal edilir ve

veritabanı önceki tutarlı durumuna geri döndürülür. Bu eylem ROLLBACK ile eşdeğerdir.

COMMIT'in kullanımı aşağıdaki basitleştirilmiş satış örneğinde gösterilmiştir; bu örnekte bir ürünün eldeki miktarı (PROD\_QOH) ve müşterinin bakiyesi müşteri

#### İZOLASYON

Bir işlem tarafından kullanılan bir veri öğesinin, ilki sona erene kadar diğer işlemler tarafından kullanılamadığı bir veritabanı işlem özelliği.

#### dayanıklılık

İşlem değişiklikleri yapıldıktan ve sonra, bir sistem arızası durumunda bile geri alınamamasını veya kaybolmamasını sağlayan işlem özelliği.

**serileştirilebilirlik** Eşzamanlı işlem operasyonlarının seçilen sırasının, aşağıdaki durumlarda üretilecek aynı nihai veritabanı durumunu oluşturduğu bir özellik işlemler seri bir şekilde yürütülmüştür.

birim fiyatı 43,99 $ (toplam 87,98 $) olan 1558-QW1 ürününden iki birim satın alır ve satın alma

işlemini müşterinin hesabına yansıtır:

GÜNCELLEME ÜRÜN

SET PROD\_QOH 5 PROD\_QOH 2 2 BURADA PROD\_CODE 5 '1558-QW1'; GÜNCELLE MÜŞTERİ

SET CUST\_BALANCE 5 CUST\_BALANCE 1 87,98

NEREDE CUST\_NUMBER 5 '10011'; COMMIT;

(İşlemin izlenmesini kolaylaştırmak için örneğin basitleştirildiğine dikkat edin. Ch10\_ SaleCo veritabanında, işlem birkaç ek tablo güncellemesi içerecektir).

UPDATE deyimi uygulamanın son eylemiyse ve uygulama normal şekilde sonlandırılıyorsa, önceki örnekte kullanılan COMMIT deyimi gerekli değildir. Ancak, iyi programlama uygulamaları COMMIT deyimini bir işlem bildiriminin sonuna eklemenizi gerektirir.

Bir işlem, ilk SQL deyimiyle karşılaşıldığında dolaylı olarak başlar. Tüm SQL uygulamaları ANSI standardını takip etmez; bazıları (SQL Server gibi) yeni bir işlemin başladığını belirtmek için aşağıdaki gibi işlem yönetimi ifadelerini kullanır:

İŞLEMI BAŞLAT;

Diğer SQL uygulamaları, BEGIN deyimine parametre olarak işlemler için özellikler atamanıza izin verir. Örneğin, Oracle RDBMS yeni bir işlemin başlangıcını ve özelliklerini bildirmek için SET TRANSACTION deyimini kullanır.

**işlem günlüğü** Veritabanını güncelleyen tüm işlemlerin kaydını tutmak için DBMS tarafından kullanılan bir özellik. Bu günlükte saklanan bilgiler DBMS tarafından kurtarma amacıyla kullanılır.

## d İşlem Günlüğü

Bir DBMS, veritabanını güncelleyen tüm işlemlerin kaydını tutmak için bir **işlem günlüğü** kullanır. DBMS bu günlükte saklanan bilgileri bir ROLL- BACK deyimi, bir programın anormal şekilde sonlandırılması veya bir ağ tutarsızlığı ya da bir disk çökmesi gibi bir sistem arızası tarafından tetiklenen bir kurtarma gereksinimi için kullanır. Bazı RDBMS'ler işlem günlüğünü bir veritabanı *for- ward'*ını o anda tutarlı bir duruma kurtarmak *için* kullanır. Örneğin bir sunucu arızasından sonra Oracle otomatik olarak işlenmemiş işlemleri geri alır ve işlenmiş ancak henüz fiziksel veritabanına yazılmamış işlemleri ileri alır. Bu davranış işlemsel doğruluk için gereklidir ve tüm işlemsel VTYS'ler için tipiktir.

DBMS veritabanını değiştiren işlemleri yürütürken, işlem günlüğünü de otomatik olarak günceller. İşlem günlüğü aşağıdakileri saklar:

* İşlemin başlangıcı için bir kayıt.
* Her bir işlem bileşeni (SQL ifadesi) için:

Ď Gerçekleştirilmekte olan işlem türü INSERT, UPDATE, DELETE).

Ď İşlemden etkilenen nesnelerin adları (tablonun adı).

Ď Güncellenmekte olan alanlar için "önce" ve "sonra" değerleri.

Ď Aynı işlem için önceki ve sonraki günlüğü girdilerine işaretçiler.

* İşlemin sona ermesi (COMMIT).

Bir işlem günlüğü kullanmak bir DBMS'nin işlem yükünü artırsa da, bozulmuş bir

veritabanını geri yükleme yeteneği bu bedele değer.

Tablo 10.1, iki SQL UPDATE deyiminden oluşan temel bir işlemi yansıtan basitleştirilmiş bir işlem günlüğünü göstermektedir. Bir sistem arızası meydana gelirse, DBMS işlem günlüğünü işlenmemiş veya tamamlanmamış tüm işlemler için inceleyecek ve bu bilgilere dayanarak veritabanını önceki durumuna geri yükleyecektir (ROLLBACK). Kurtarma işlemi tamamlandığında

tamamlandığında, DBMS, hata oluşmadan önce veritabanına fiziksel olarak yazılmamış olan tüm

işlenmiş işlemleri günlüğe yazacaktır.

Bir işlemin sonlandırılmasından önce bir ROLLBACK yayınlanırsa, DBMS önceki işlemlerin *dayanıklılığını* korumak için veritabanını için değil, yalnızca söz konusu işlem için geri yükleyecektir. Başka bir deyişle, taahhüt edilen işlemler geri alınmaz.

İşlem günlüğü veritabanının kritik bir parçasıdır ve genellikle gerçek veritabanı dosyalarından ayrı olarak yönetilen bir veya daha fazla dosya olarak uygulanır. İşlem günlüğü, disk dolu koşulları ve disk çökmeleri gibi yaygın tehlikelere maruz kalır. İşlem günlüğü bir VTYS'deki en kritik verilerden bazılarını içerdiğinden, bazı uygulamalar bir sistem arızasının sonuçlarını azaltmak için birkaç farklı diskteki günlükleri destekler.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tablo 10.1** | | **Bir İşlem Günlüğü** | | | | | | | | |
| **TRL\_ID** | **TRX\_NUM** | | **ÖNCEKİ PTR** | **SONRAKİ PTR** | **OPERASYON** | **TABLO** | **SIRA KİMLİĞİ** | **ATIF** | **ÖNCEKİ**  **DEĞER** | **DEĞER SONRA SI** |
| 341 | 101 | | Null | 352 | BAŞLA | \*\*\*\*İşlem Başlat |  |  |  |  |
| 352 | 101 | | 341 | 363 | GÜNCELLEME | ÜRÜN | 1558-QW1 | PROD\_QOH | 25 | 23 |
| 363 | 101 | | 352 | 365 | GÜNCELLEME | MÜŞTERİ | 10011 | CUST\_ BALANCE | 525.75 | 615.73 |
| 365 | 101 | | 363 | Null | TAAHHÜT | \*\*\*\* İşlem Sonu |  |  |  |  |

TRL\_ID 5 İşlem günlüğü kayıt kimliği TRX\_NUM 5 İşlem numarası



PTR 5 İşlem günlüğü kayıt kimliğine işaretçi (N*ot*: İşlem numarası DBMS tarafından otomatik olarak atanır).

# Eşzamanlılık Kontrolü

Çok kullanıcılı bir veritabanı sisteminde işlemlerin **eşzamanlı** olarak yürütülmesini koordine etmek eşzamanlılık **kontrolü** olarak bilinir. Eşzamanlılık amacı, çok kullanıcılı bir veritabanı ortamında işlemlerin serileştirilebilirliğini sağlamaktır. Bu hedefe ulaşmak için çoğu eşzamanlılık kontrol tekniği, eşzamanlı olarak yürütülen işlemlerin izolasyon özelliğini korumaya yöneliktir. Eşzamanlılık kontrolü önemlidir çünkü paylaşılan bir veritabanı üzerinde işlemlerin aynı anda yürütülmesi çeşitli veri bütünlüğü ve tutarlılık sorunları yaratabilir. Üç ana sorun kayıp güncellemeler, taahhüt edilmemiş veriler ve tutarsız geri alımlardır.

## a Kayıp Güncellemeler

**Kayıp güncelleme** sorunu, T1 ve T2 gibi iki eşzamanlı işlem aynı veri öğesini güncellediğinde ve güncellemelerden biri kaybolduğunda (diğer işlem tarafından üzerine yazıldığında) ortaya çıkar. Kayıp güncellemelerin bir örneğini görmek için basit bir PRODUCT tablosunu inceleyin. Tablonun veri öğelerinden biri ürünün eldeki miktarıdır (PROD\_QOH). Mevcut PROD\_QOH değeri 35 olan bir ürününüz olduğunu varsayın. Ayrıca, T1 ve T2 adlı iki eşzamanlı işlemin gerçekleştiğini ve PRODUCT tablosundaki bazı öğeler için PROD\_QOH değerini güncellediğini varsayın. İşlemler Tablo 10.2'de gösterilmektedir.

**eş zamanlılık kontrolü** İşlemlerin aynı anda yürütülmesini koordine eden bir DBMS özelliği

Veri bütünlüğünü korurken çok işlemli bir veritabanı sistemi.

**kayıp güncelleme** İşlemlerin eş zamanlı yürütülmesi sırasında bir veri güncellemesinin kaybolduğu bir eşzamanlılık kontrol problemi.

|  |  |  |
| --- | --- | --- |
| **Tablo 10.2** | **PROD\_QOH'u Güncellemek için İki Eşzamanlı İşlem** | |
| **İşlem** | | **Hesaplama** |
| T1: 100 birim satın alın | | PROD\_QOH 5 PROD\_QOH 1 100 |
| T2: 30 birim satmak | | PROD\_QOH 5 PROD\_QOH 2 30 |

Tablo 10.3, işlemlerin normal koşullar altında seri olarak yürütülmesini göstermekte ve PROD\_QOH 5 105 doğru yanıtını vermektedir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tablo 10.3** | **İki İşlemin Seri Olarak Yürütülmesi** | | | |
| **Zaman** | | **İşlem** | **Adım** | **Saklanan Değer** |
| 1 | | T1 | PROD\_QOH'u okuyun | 35 |
| 2 | | T1 | PROD\_QOH 5 35 1 100 |  |
| 3 | | T1 | PROD\_QOH yazın | 135 |
| 4 | | T2 | PROD\_QOH'u okuyun | 135 |
| 5 | | T2 | PROD\_QOH 5 135 2 30 |  |
| 6 | | T2 | PROD\_QOH yazın | 105 |

Ancak, bir işlemin, *aynı* ürünü kullanan önceki bir işlem gerçekleştirilmeden *önce* tablodan bir ürünün PROD\_QOH değerini okuyabildiğini varsayalım. Tablo 10.4'te gösterilen sıra, kayıp güncelleme sorununun nasıl ortaya çıkabileceğini göstermektedir. İkinci işlem (T2) yürütüldüğünde ilk işlemin (T1) henüz işlenmemiş olduğuna dikkat edin. Bu nedenle, T2 hala 35 değeri üzerinde çalışmaktadır ve bunun çıkarılması bellekte 5 değerini verir. Bu arada, T1 diske 135 değerini yazar ve bu değer hemen T2 tarafından üzerine yazılır. Kısacası, 100 birimlik toplama işlemi işlem sırasında "kaybolur".

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tablo 10.4** | **Kayıp Güncellemeler** | | | |
| **Zaman** | | **İşlem** | **Adım** | **Saklanan Değer** |
| 1 | | T1 | PROD\_QOH'u okuyun | 35 |
| 2 | | T2 | PROD\_QOH'u okuyun | 35 |
| 3 | | T1 | PROD\_QOH 5 35 1 100 |  |
| 4 | | T2 | PROD\_QOH 5 35 2 30 |  |
| 5 | | T1 | PROD\_QOH yaz (kayıp güncelleme) | 135 |
| 6 | | T2 | PROD\_QOH yazın | 5 |

taahhüt edilmemiş **veriler** Bir işlemin başka bir işlemden taahhüt edilmemiş verilere eriştiği bir eşzamanlılık kontrolü sorunu.

## b Taahhüt Edilmemiş Veriler

**Taahhütsüz veri** olgusu, iki işlem (T1 ve T2) aynı anda yürütüldüğünde ve ilk işlem (T1), ikinci işlem (T2) taahhütsüz veriye eriştikten sonra geri alındığında, dolayısıyla işlemlerin izolasyon özelliği ihlal edildiğinde ortaya çıkar. Bu olasılığı göstermek için, kayıp güncellemeler tartışması sırasında açıklanan işlemlerin aynısını kullanın. T1'in iki atomik parçası vardır, bunlardan biri envanterin güncellenmesidir; diğer olası parça ise fatura toplamının güncellenmesidir (gösterilmemiştir). T1, fatura toplamının güncellenmesi sırasında oluşan bir hata nedeniyle geri dönmeye zorlanır; sonuna kadar geri dönerek envanter güncellemesini de geri alır. Bu kez T1 işlemi

100 birimin eklenmesini ortadan kaldırmak için geri alınır. (Bkz. Tablo 10.5.) T2 orijinal 35 birimden 30 birim çıkardığı için doğru cevap 5 olmalıdır.

|  |  |  |
| --- | --- | --- |
| **Tablo 10.5** | **Taahhüt Edilmemiş Veri Sorunu Yaratan İşlemler** | |
| **İşlem** | | **Hesaplama** |
| T1: 100 birim satın alın | | PROD\_QOH 5 PROD\_QOH 1 100 (Geri alındı) |
| T2: 30 birim satmak | | PROD\_QOH 5 PROD\_QOH− 30 |

Tablo 10.6, bu işlemlerin seri olarak yürütülmesinin normal şartlar altında doğru cevabı nasıl verdiğini göstermektedir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tablo 10.6** | **İki İşlemin Doğru Yürütülmesi** | | | |
| **Zaman** | | **İşlem** | **Adım** | **Saklanan Değer** |
| 1 | | T1 | PROD\_QOH'u okuyun | 35 |
| 2 | | T1 | PROD\_QOH 5 35 1 100 |  |
| 3 | | T1 | PROD\_QOH yazın | 135 |
| 4 | | T1 | \*\*\*\*\*ROLLBACK \*\*\*\*\* | 35 |
| 5 | | T2 | PROD\_QOH'u okuyun | 35 |
| 6 | | T2 | PROD\_QOH 5 35 2 30 |  |
| 7 | | T2 | PROD\_QOH yazın | 5 |

Tablo 10.7, T2 yürütülmeye başladıktan sonra ROLLBACK tamamlandığında işlenmemiş veri sorununun nasıl ortaya çıkabileceğini göstermektedir.

**Taahhütsüz Veri Sorunu**

**Tablo 10.7**

|  |  |  |  |
| --- | --- | --- | --- |
| **Zaman** | **İşlem** | **Adım** | **Saklanan Değer** |
| 1 | T1 | PROD\_QOH'u okuyun | 35 |
| 2 | T1 | PROD\_QOH 5 35 1 100 |  |
| 3 | T1 | PROD\_QOH yazın | 135 |
| 4 | T2 | PROD\_QOH Oku (Taahhüt edilmemiş verileri oku) | 135 |
| 5 | T2 | PROD\_QOH 5 135 2 30 |  |
| 6 | T1 | \*\*\*\*\* ROLLBACK \*\*\*\*\* | 35 |
| 7 | T2 | PROD\_QOH yazın | 105 |

## c Tutarsız Geri Alımlar

**Tutarsız** erişimler, bir işlem verilere bir veya daha fazla diğer işlemin bu verilerle çalışmayı bitirmesinden önce veya sonra eriştiğinde meydana gelir. Örneğin, başka bir işlem (T2) aynı verileri güncellerken T1 işlemi bir veri kümesi üzerinde bazı özet (toplama) işlevleri hesaplarsa tutarsız bir geri alma meydana gelebilir. Sorun, işlemin bazı verileri değiştirilmeden önce, diğer verileri ise değiştirildikten *sonra* okuması ve böylece tutarsız sonuçlar elde etmesidir.

Problemi açıklamak için aşağıdaki koşulları varsayalım:

* + 1. T1, PRODUCT tablosunda depolanan ürünlerin eldeki toplam miktarını hesaplar.
    2. Aynı zamanda T2, PRODUCT tablosundaki iki ürün için eldeki miktarı (PROD\_QOH) günceller.

Bu iki işlem Tablo 10.8'de gösterilmektedir.

**tutarsız erişimler** Bir veri kümesi üzerinde özet (toplu) işlevleri hesaplayan bir işlem olduğunda ortaya çıkan bir eşzamanlılık kontrolü sorunu

Diğer işlemler verileri güncellerken hatalı sonuçlar ortaya çıkabilir.

|  |  |  |
| --- | --- | --- |
| **Tablo 10.8** | **Güncelleme sırasında geri alma** | |
| **İşlem T1** | | **İşlem T2** |
| SELECT SUM(PROD\_QOH) FROM PRODUCT | | GÜNCEL ÜRÜN  SET PROD\_QOH 5 PROD\_QOH 1 10 WHERE  PROD\_CODE 5 1546-QQ2 |
|  | | GÜNCEL ÜRÜN  SET PROD\_QOH 5 PROD\_QOH 2 10 WHERE  PROD\_CODE 5 1558-QW1 |
|  | | TAAHHÜT; |

T1 tüm kalemler için eldeki toplam miktarı (PROD\_QOH) hesaplarken, T2 bir yazım hatasının düzeltilmesini temsil eder: kullanıcı 1558-QW1 ürününün PROD\_QOH'sine 10 birim eklemiş ancak 10 birimi 1546-QQ2 ürününün PROD\_QOH'sine eklemek istemiştir. Sorunu düzeltmek için kullanıcı 1546-QQ2 ürününün PROD\_QOH değerine 10 ekler ve 1558- QW1 ürününün PROD\_QOH değerinden 10 çıkarır. (Tablo 10.8'deki iki UPDATE deyimine bakın.) İlk ve son PROD\_QOH değerleri Tablo 10.9'da yansıtılır. (PRODUCT tablosu için yalnızca birkaç PROD\_CODE değeri gösterilmiştir. Konuyu açıklamak için, PROD\_QOH değerlerinin toplamı bu birkaç ürün için gösterilmiştir).

**İşlem Sonuçları: Veri Girişi Düzeltmesi**

**Tablo 10.9**

|  |  |  |
| --- | --- | --- |
|  | **Önce** | **Sonra** |
| **PROD\_CODE** | **PROD\_QOH** | **PROD\_QOH** |
| 11QER/31 | 8 | 8 |
| 13-Q2/P2 | 32 | 32 |
| 1546-QQ2 | 15 | (15 1 10) → 25 |
| 1558-QW1 | 23 | (23 2 10) → 13 |
| 2232-QTY | 8 | 8 |
| 2232-QWE | 6 | 6 |
| **Toplam** | **92** | **92** |

Tablo 10.9'da gösterilen nihai sonuçlar ayarlamadan sonra doğru olsa da, Tablo 10.10 işlemin yürütülmesi sırasında tutarsız alımların mümkün olduğunu ve T1'in yürütülmesinin sonucunu yanlış hale getirdiğini göstermektedir. Tablo 10.10'da gösterilen "Sonra" toplamı, 1546-QQ2 ürünü için 25 değerinin WRITE deyimi tamamlandıktan *sonra* okunduğunu yansıtmaktadır. Bu nedenle, "Sonra" toplamı 40 1 25 5 65'tir. "Önce" , 1558-QW1 ürünü için 23 değerinin, 13'ün düzeltilmiş güncellemesini yansıtmak için bir sonraki WRITE deyimi tamamlanmadan *önce* okunduğunu yansıtır. Bu nedenle, "Önce" toplamı 65 1 23 5 88'dir.

Hesaplanan 102 cevabı yanlıştır çünkü Tablo 10.9'dan doğru cevabın 92 olduğunu biliyorsunuz. DBMS eşzamanlılık kontrolü uygulamadığı sürece, çok kullanıcılı bir veritabanı ortamı bilgi sistemi içinde tahribat yaratabilir.

## d Zamanlayıcı

Artık iki veya daha fazla eşzamanlı işlem yürütüldüğünde ciddi sorunların ortaya çıkabileceğini biliyorsunuz. Ayrıca bir veritabanı işleminin, veritabanını tutarlı bir durumdan diğerine götüren bir dizi veritabanı I/O işlemini içerdiğini de biliyorsunuz. Son olarak, veritabanı tutarlılığının yalnızca işlemlerin yürütülmesinden önce ve sonra sağlanabileceğini biliyorsunuz. Bir işlem birden fazla tablo ve satırı güncelliyorsa, bir veritabanı her zaman bir işlemin yürütülmesi sırasında kaçınılmaz bir geçici tutarsızlık durumundan geçer. (İşlem yalnızca bir güncelleme içeriyorsa, geçici tutarsızlık söz konusu değildir). Geçici tutarsızlık vardır çünkü bilgisayar işlemleri birbiri ardına seri olarak yürütür. Bu seri işlem sırasında, işlemlerin izolasyon özelliği, henüz diğer işlemler tarafından serbest bırakılmamış verilere erişmelerini engeller. Bu husus, aynı anda birden fazla talimatı yürütebilen çok çekirdekli işlemcilerin kullanıldığı günümüzde daha da önemlidir. İki işlem eşzamanlı olarak yürütülürse ve aynı verilere erişirlerse ne olur?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tablo 10.10** | **Tutarsız Alımlar** | | | | |
| **Zaman** | | **İşlem** | **Eylem** | **Değer** | **Toplam** |
| 1 | | T1 | PROD\_CODE 5 '11QER/31' için PROD\_QOH dosyasını okuyun | 8 | 8 |
| 2 | | T1 | PROD\_CODE 5 '13-Q2/P2' için PROD\_QOH dosyasını okuyun | 32 | 40 |
| 3 | | T2 | PROD\_CODE 5 '1546-QQ2' için PROD\_QOH dosyasını okuyun | 15 |  |
| 4 | | T2 | PROD\_QOH 5 15 1 10 |  |  |
| 5 | | T2 | PROD\_CODE 5 '1546-QQ2' için PROD\_QOH yazın | 25 |  |
| 6 | | T1 | PROD\_CODE 5 '1546-QQ2' için PROD\_QOH dosyasını okuyun | 25 | (Sonra) 65 |
| 7 | | T1 | PROD\_CODE 5 '1558-QW1' için PROD\_QOH dosyasını okuyun | 23 | (Önce) 88 |
| 8 | | T2 | PROD\_CODE 5 '1558-QW1' için PROD\_QOH dosyasını okuyun | 23 |  |
| 9 | | T2 | PROD\_QOH 5 23 2 10 |  |  |
| 10 | | T2 | PROD\_CODE 5 '1558-QW1' için PROD\_QOH yazın | 13 |  |
| 11 | | T2 | \*\*\*\*\* COMMIT \*\*\*\*\* |  |  |
| 12 | | T1 | PROD\_CODE 5 '2232-QTY' için PROD\_QOH'u okuyun | 8 | 96 |
| 13 | | T1 | PROD\_CODE 5 '2232-QWE' için PROD\_QOH dosyasını okuyun | 6 | 102 |

Önceki örneklerde, bir işlem içindeki işlemler rastgele bir sırada yürütülmüştür. İki işlem, T1 ve T2, *ilgisiz* verilere eriştiği sürece herhangi bir çakışma olmaz ve yürütme sırasının nihai sonuçla ilgisi yoktur. Ancak, işlemler birbiriyle ilişkili veriler veya aynı veriler üzerinde çalışıyorsa, işlem bileşenleri arasında çakışma olabilir ve bir yürütme sırasının diğerine göre seçilmesi bazı istenmeyen sonuçlara yol açabilir. Peki, doğru sıra nasıl belirlenir ve bu sırayı kim belirler? Neyse ki, DBMS bu zor görevi yerleşik bir zamanlayıcı kullanarak halleder.

Zamanlayıcı, işlemlerin eşzamanlı işlemler içinde yürütülme sırasını belirleyen özel bir DBMS işlemidir. Zamanlayıcı, işlemlerin serileştirilebilirliğini ve izolasyonunu sağlamak için veritabanı işlemlerinin yürütülmesini *birbirine bağlar*. Uygun sırayı belirlemek için zamanlayıcı, eylemlerini sonraki bölümlerde açıklanan kilitleme veya zaman damgası yöntemleri eşzamanlılık kontrol algoritmalarına dayandırır. Ancak, tüm işlemlerin serileştirilebilir olmadığını anlamak önemlidir. DBMS hangi işlemlerin serileştirilebilir olduğunu belirler ve işlemlerin yürütülmesini serileştirmeye devam eder. Genel olarak, serileştirilemeyen işlemler VTYS tarafından ilk gelene ilk hizmet esasına göre yürütülür. Zamanlayıcının ana görevi, işlemlerin (T1, T2, T3, vb.) serileştirilmiş yürütülmesinin, işlemlerin seri sırayla (birbiri ardına) yürütülmesiyle aynı sonuçları verdiği bir işlemin işlemlerinin **serileştirilebilir** bir **programını** oluşturmaktır. Zamanlayıcı ayrıca bilgisayarın merkezi işlem biriminin (CPU) ve depolama sistemlerinin verimli bir şekilde kullanılmasını sağlar. Eğer işlemlerin yürütülmesini zamanlamanın bir yolu olmasaydı, tüm işlemler ilk gelen ilk hizmet alır esasına göre yürütülürdü.

Bu yaklaşımla ilgili sorun, CPU bir OKU veya YAZ işleminin bitmesini beklerken işlem süresinin boşa harcanması ve böylece birkaç CPU döngüsünün kaybedilmesidir. Kısacası, ilk gelene ilk hizmet çizelgelemesi, çok kullanıcılı VTYS ortamında kabul edilemez yanıt süreleri verme eğilimindedir. Bu nedenle, genel sistemin verimliliğini artırmak için başka bir zamanlama yöntemine ihtiyaç vardır. Ayrıca, zamanlayıcı iki işlemin aynı veri öğesini aynı anda güncellememesini sağlamak için veri izolasyonunu kolaylaştırır. Veritabanı işlemleri, çakışmalara neden olan OKU ve/veya YAZ eylemleri gerektirebilir. Örneğin, Tablo 10.11, T1 ve T2 adlı iki işlemin aynı veriler üzerinde eşzamanlı olarak yürütülmesi durumunda olası çakışma senaryolarını göstermektedir. Tablo 10.11'de, aynı verilere eriştiklerinde iki işlemin çakışma

içinde olduğuna dikkat edin

**zamanlayıcı** Eşzamanlı işlemlerin yürütülme sırasını

belirleyen DBMS bileşeni. Zamanlayıcı yürütmeyi *birbirine bağlar Serileştirilebilirliği* sağlamak için veritabanı işlemlerinin belirli bir sırayla gerçekleştirilmesi.

**serileştirilebilir program** İşlem yönetiminde, bir işlem programı

İşlemlerin serpiştirilmiş olarak yürütülmesi, seri sırayla yürütülmüş aynı sonucu verir.

ve bunlardan en az biri bir YAZMA işlemidir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tablo 10.11** | **Okuma/Yazma Çakışma Senaryoları: Çakışan Veritabanı İşlemleri Matrisi** | | | |
|  | | **İşlemler** | |  |
|  | | **T1** | **T2** | **Sonuç** |
| Operasyonlar | | Okuyun | Okuyun | Çatışma yok |
|  | | Okuyun | Yazmak | Çatışma |
|  | | Yazmak | Okuyun | Çatışma |
|  | | Yazmak | Yazmak | Çatışma |

Eşzamanlı çakışan işlemlerin yürütülmesini planlamak için çeşitli yöntemler önerilmiştir. Bu yöntemler kilitleme, zaman damgası ve iyimser olarak sınıflandırılmıştır. Daha sonra ele alınacak olan kilitleme yöntemleri en sık kullanılan yöntemlerdir.

#### kilit

Belirli bir işlemde bir veri öğesinin benzersiz kullanımını garanti eden bir cihaz. Bir işlem, veri erişiminden önce bir kilit gerektirir; kilit

diğer işlemlerin veri öğesini kendi kullanımları için kilitleyebilmelerini sağlamak için işlemin yürütülmesinden sonra serbest bırakılır.

**kötümser kilitleme** Kilitlerin kullanımı, işlemler arasında çakışma varsayımına dayanır.

#### kilit yöneticisi

Kilitlerin atanması ve serbest bırakılmasından sorumlu olan bir DBMS bileşeni.

#### kilit granülerliği

Kilit kullanım seviyesi. Kilitleme şu düzeylerde gerçekleşebilir: veritabanı, tablo, sayfa, satır ve alan (öznitelik).

**veritabanı düzeyinde kilit** Veritabanı erişimini kilidin sahibiyle kısıtlayan ve aynı anda yalnızca bir kullanıcının veritabanına izin veren bir kilit türüdür. Bu kilit

toplu işlemler için çalışır ancak çevrimiçi çok kullanıcılı VTYS'ler için uygun değildir.

# Kilitleme Yöntemleri ile Eşzamanlılık Kontrolü

Kilitleme yöntemleri eşzamanlılık kontrolünde kullanılan en yaygın tekniklerden biridir çünkü eşzamanlı olarak yürütülen işlemlerde kullanılan veri öğelerinin izolasyonunu kolaylaştırırlar. Bir **kilit**, bir veri öğesinin mevcut bir işleme özel kullanımını garanti eder. Başka bir deyişle, T2 işlemi o anda T1 işlemi tarafından kullanılmakta olan bir veri öğesine erişemez. Bir işlem veri erişiminden önce bir kilit edinir; işlem tamamlandığında kilit serbest bırakılır (kilidi açılır), böylece başka bir işlem veri öğesini kendi özel kullanımı için kilitleyebilir. Bu kilitleme eylemleri serisi, eşzamanlı işlemlerin aynı veri öğesini aynı anda manipüle etmeye çalışabileceğini varsayar. İşlemler arasında çakışmanın muhtemel olduğu varsayımına dayanan kilitlerin kullanımı genellikle **kötümser** kilitleme olarak adlandırılır.

Bölüm 10-1a ve 10-1b'den, bir işlem *sırasında* veri tutarlılığının garanti edilemeyeceğini hatırlayın; birkaç güncelleme yürütüldüğünde veritabanı geçici olarak tutarsız bir durumda olabilir. Bu nedenle, başka bir işlemin tutarsız verileri okumasını önlemek için kilitler gereklidir.

Çoğu çok kullanıcılı DBMS kilitleme prosedürlerini otomatik olarak başlatır ve uygular. Tüm kilit bilgileri, işlemler tarafından kullanılan kilitlerin atanması ve denetlenmesinden sorumlu olan bir **kilit yöneticisi** tarafından ele alınır.

## a Kilit Granülerliği

Kilit **ayrıntı** düzeyi, **kilit** kullanım düzeyini gösterir. Kilitleme şu seviyelerde gerçekleşebilir: veritabanı, tablo, sayfa, satır ve hatta alan (öznitelik).

### Veritabanı Seviyesi

**Veritabanı düzeyinde** bir **kilitlemede**, tüm veritabanı kilitlenir, böylece T1 işlemi yürütülürken T2 işlemi tarafından veritabanındaki herhangi bir tablonun kullanılması engellenir. Bu kilitleme seviyesi toplu işlemler için iyidir, ancak çok kullanıcılı VTYS'ler için uygun değildir. Binlerce işlemin bir sonraki işlemin tüm veritabanını rezerve edebilmesi için bir önceki işlemin tamamlanmasını beklemek zorunda kalması durumunda veri erişiminin ne kadar s-l-o-w olacağını hayal edebilirsiniz. Şekil 10.3'te veritabanı düzeyinde kilit gösterilmektedir; bu kilit nedeniyle, T1 ve T2 işlemleri *farklı tabloları kullansalar bile* aynı veritabanına eşzamanlı olarak erişemezler.

**Şekil 10.3 Veritabanı Düzeyinde Kilitleme Sırası**



Zam an

İşlem 1 (T1)

(Tablo A'yı güncelleyin)

Bordro Veritabanı

Tablo A

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

İşlem 2 (T2)

(Tablo B'yi güncelleyin)

* + 1. Veritabanını kilitleme isteği
    2. Kilitli

TAM

BEKL E

Veritabanını kilitleme isteği

AM.

3

4

5

6 Kilidi Açıldı

7

8

9

Tablo B

TAM AM.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Kilitli

Kilitli değil

### Tablo Seviyesi

**Tablo düzeyinde** bir kilitte, tüm **tablo** kilitlenir ve T1 işlemi tabloyu kullanırken T2 işleminin herhangi bir satıra erişimi engellenir. Bir işlem birden fazla tabloya erişim gerektiriyorsa, her tablo kilitlenebilir. Ancak, farklı tablolara eriştikleri sürece iki işlem aynı veritabanına erişebilir. Tablo düzeyindeki kilitler, veritabanı düzeyindeki kilitlere göre daha az kısıtlayıcı olsa da, aynı tabloya erişmek için bekleyen çok sayıda işlem olduğunda trafik sıkışıklığına neden olur. Böyle bir durum, özellikle farklı işlemler aynı tablonun farklı bölümlerine erişim gerektirdiğinde kilit bir gecikmeye zorlarsa, yani işlemler birbirini engellemeyecekse can sıkıcıdır. Sonuç olarak, tablo düzeyindeki kilitler çok kullanıcılı VTYS'ler için uygun değildir. Şekil 10.4, tablo düzeyinde bir kilidin etkisini göstermektedir. T1 ve T2 işlemlerinin aynı tabloya erişemeyeceklerine dikkat edin.

farklı satırları kullanmaya çalışırsa; T2, T1 tablonun kilidini açana kadar beklemelidir.

**tablo düzeyinde kilit** Bir seferde yalnızca bir işlemin bir kilitleme şemasına erişmesine izin veren bir kilitleme şeması tablo. Tablo düzeyinde bir

kilit tüm tabloyu kilitler ve T1 tabloyu kullanırken T2 işlemi tarafından herhangi bir satıra erişimi engeller.

**Şekil 10.4 Tablo Düzeyinde Kilit Örneği**



Zam an

1

2

3

4

5

6

7

8

9

İşlem 1 (T1)

(Satır 5'i güncelleyin)

Bordro Veritabanı

Tablo A

İşlem 2 (T2) (Güncelleme satırı 30)

Kilit Tablosu A talebi

Kilitli

Kilit Tablosu A talebi

TA MA M.

BEKLE

Kilit açıldı (işlem 1'in sonu)

TAMAM. Kilitli

Kilitli değil (işlem 2'nin sonu)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**sayfa düzeyinde kilit** Bu kilit türünde, veritabanı yönetim sistemi bir disk

sayfasının tamamını veya bir bölümünü kilitler. Bir disk sayfası, bir veya daha fazla satır için ve bir veya daha fazla tablodan veri içerebilir.

#### diskpage (sayfa)

Kalıcı depolamada, bir diskin doğrudan adreslenebilir bir bölümü olarak tanımlanabilen bir disk eşdeğeri. Bir disk sayfasının bir

4K, 8K veya 16K gibi sabit

### Sayfa Seviyesi

**Sayfa düzeyinde** bir **kilitte**, DBMS tüm bir disk sayfasını kilitler. Bir **disk sayfası** ya da **sayfa**, bir diskin doğrudan adreslenebilir bir bölümü olarak tanımlanabilen bir *disk* bloğunun eşdeğeridir. Bir sayfanın 4K, 8K ya da 16K gibi sabit bir boyutu vardır. Örneğin, 4K sayfaya yalnızca 73 bayt yazmak istiyorsanız, 4K sayfanın tamamı diskten okunmalı, bellekte güncellenmeli ve diske geri yazılmalıdır. Bir tablo birkaç sayfaya yayılabilir ve bir sayfa bir veya daha fazla birkaç satırını içerebilir. Sayfa seviyesi kilitler şu anda çok kullanıcılı VTYS'ler için en sık kullanılan kilitleme yöntemidir. Şekil 10.5'te sayfa düzeyinde bir kilit örneği gösterilmektedir. T1 ve T2'nin farklı disk sayfalarını kilitlerken aynı tabloya eriştiğine dikkat edin. Eğer T2, T1 tarafından kilitlenmiş bir sayfada bulunan bir satırın kullanımına ihtiyaç duyarsa, T2, T1 sayfanın kilidini açana kadar beklemelidir.



**Şekil 10.5 Sayfa Düzeyinde Kilit Örneği**

Bordro Veritabanı

boyutlar.

Zam an

1

2

3

4

5

6

7

İşlem 1 (T1)

(Satır 1'i güncelleyin)

Sayfa 1 isteğini kilitle

Kilitli TAMAM.

Sayfa 1'in kilidini aç (işlem sonu)

Tablo A

1

2 Sayfa

1

3

4

5 Sayfa

2

6

İşlem 2 (T2)

(Satır 5 ve 2'yi güncelleyin)

Sayfa 2 isteğini kilitle TAMAM. Kilitli

Sayfa 1 isteğini kilitle BEKLE

TAMAM. Kilitli

Sayfa 1 ve 2'nin kilidini açın (işlem sonu)

Satır numarası

**satır düzeyinde kilit** DBMS'nin, satırlar aynı sayfada bile eşzamanlı işlemlerin aynı tablonun farklı satırlarına erişmesine izin verdiği daha az kısıtlayıcı bir veritabanı kilidi.

### Sıra Seviyesi

**Satır düzeyinde** bir kilit, daha önce tartışılan kilitlerden çok daha az kısıtlayıcıdır. DBMS, satırlar aynı sayfada bulunsa bile eşzamanlı işlemlerin aynı tablonun farklı satırlarına erişmesine izin verir. Satır düzeyinde kilitleme yaklaşımı verilerin kullanılabilirliğini artırsa da, çakışan bir işlemde yer alan veritabanının bir tablosundaki her satır için bir kilit bulunduğundan, yönetimi yüksek ek yük gerektirir. Modern DBMS'ler, uygulama oturumu aynı sayfa üzerinde birden fazla kilit talep ettiğinde kilidi otomatik olarak satır seviyesinden sayfa seviyesine yükseltir. Şekil 10.6'da satır düzeyinde kilit kullanımı gösterilmektedir.

**Şekil 10.6 Satır Düzeyinde Kilit Örneği**



Satır numarası

Satır 2'nin kilidini aç (işlem sonu)

2. sırayı kilitleme isteği

Kilitli

TA MA M.

Sayfa 2

Sayfa 1

1

2

3

4

5

6

Satır 1'in kilidini aç (işlem sonu)

TA MA M.

Kilitli

Satır 1 isteğini kilitle

İşlem 2 (T2)

(Satır 2'yi güncelleyin)

Bordro Veritabanı

Tablo A

İşlem 1 (T1)

(Satır 1'i güncelleyin)

Zam an

1

2

3

4

5

6

**Bölüm 10: İşlem Yönetimi ve Eşzamanlılık Kontrolü 489**

Şekil 10.6'da, istenen satırlar aynı sayfada olsa bile her iki işlemin de eşzamanlı olarak

yürütülebileceğine dikkat edin. T2 yalnızca T1 ile aynı satırı talep ederse beklemelidir.

### Saha Seviyesi

**Alan düzeyinde kilit**, eşzamanlı işlemlerin aynı satıra, bu satır içindeki farklı alanların (niteliklerin) kullanılmasını gerektirdiği sürece erişmesine olanak tanır. Alan düzeyinde kilitleme açıkça en esnek çok kullanıcılı veri erişimini sağlamasına rağmen, son derece yüksek düzeyde bilgisayar ek yükü gerektirdiği ve satır düzeyinde kilit pratikte çok daha kullanışlı olduğu için bir VTYS'de nadiren uygulanır.

## b Kilit Tipleri

Kilidin ayrıntı düzeyine bakılmaksızın, DBMS farklı kilit türleri veya modları kullanabilir: ikili veya

paylaşımlı/özel.

### İkili

Bir **ikili kilidin** yalnızca iki durumu vardır: kilitli (1) veya kilitsiz (0). Veritabanı, tablo, sayfa veya satır gibi bir nesne bir işlem tarafından kilitlenirse, başka hiçbir işlem bu nesneyi kullanamaz. Bir nesnenin kilidi açılmışsa, herhangi bir işlem nesneyi kullanmak için kilitleyebilir. Her veritabanı işlemi, etkilenen nesnenin kilitlenmesini gerektirir. Kural olarak, bir işlem sonlandırıldıktan sonra nesnenin kilidini açmalıdır. Bu nedenle, her işlem erişilen her veri öğesi için bir kilitleme ve kilit açma işlemi gerektirir. Bu işlemler DBMS tarafından otomatik olarak yönetilir ve zamanlanır; kullanıcı veri öğelerini kilitlemez veya kilidini açmaz. (Her VTYS'nin varsayılan bir kilitleme mekanizması vardır. Son kullanıcı varsayılan ayarları geçersiz kılmak isterse, LOCK TABLE komutu ve diğer SQL komutları bu amaç için kullanılabilir).

İkili kilitleme tekniği, Tablo 10.4'te karşılaştığınız kayıp güncelleme problemi kullanılarak Tablo 10.12'de gösterilmiştir. Kilitleme ve kilit açma özelliklerinin kayıp güncelleme sorununu ortadan kaldırdığına dikkat edin çünkü WRITE deyimi tamamlanana kadar kilit serbest bırakılmaz. Bu nedenle, bir PROD\_QOH değeri uygun şekilde güncellenene kadar kullanılamaz. Ancak, ikili kilitlerin artık optimum eşzamanlılık koşullarını sağlamak için çok kısıtlayıcı olduğu düşünülmektedir. Örneğin, iki işlem de veritabanını güncellemese ve bu nedenle eşzamanlılık sorunları oluşmasa bile DBMS iki işlemin aynı veritabanı nesnesini okumasına izin vermeyecektir. Tablo 10.11'den eşzamanlılık çakışmalarının yalnızca iki işlem eşzamanlı olarak yürütüldüğünde ve bunlardan biri veritabanını güncellediğinde ortaya çıktığını hatırlayın.

**saha düzeyinde kilit** Eşzamanlı işlemlerin, bu farklı alanların (niteliklerin) kullanılmasını gerektirdiği sürece aynı satıra erişmesine izin veren bir kilit. Bu kilit türü en esnek çok kullanıcılı veri erişimini sağlar ancak

yüksek düzeyde bilgisayar yükü.

#### ikili kilit

Yalnızca iki durumu olan bir kilit: *kilitli* (1) ve *kilitsiz* (0). Bir veri öğesi bir işlem tarafından , başka hiçbir işlem bu veri öğesini kullanamaz.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tablo 10.12** | **İkili Kilit Örneği** | | | |
| **Zaman** | | **İşlem** | **Adım** | **Saklanan Değer** |
| 1 | | T1 | Kilit ÜRÜN |  |
| 2 | | T1 | PROD\_QOH'u okuyun | 15 |
| 3 | | T1 | PROD\_QOH 5 15 1 10 |  |
| 4 | | T1 | PROD\_QOH yazın | 25 |
| 5 | | T1 | Ürün Kilidini Aç |  |
| 6 | | T2 | Kilit ÜRÜN |  |
| 7 | | T2 | PROD\_QOH'u okuyun | 23 |
| 8 | | T2 | PROD\_QOH 5 23 2 10 |  |
| 9 | | T2 | PROD\_QOH yazın | 13 |
| 10 | | T2 | Ürün Kilidini Aç |  |

#### özel ki̇li̇t

Bir işlem bir veri öğesini güncellemek için izin istediğinde ve bu veri öğesi üzerinde başka hiçbir işlem tarafından kilit tutulmadığında özel bir kilit verilir. Özel bir kilit kilidi diğer işlemlerin veritabanına erişmesine

izin vermez.

#### paylaşılan kilit

Bir işlem talep ettiğinde verilen bir kilit

Bir veritabanından veri okuma izni ve başka bir işlem tarafından veriler üzerinde hiçbir özel kilit tutulmaması. Paylaşılan kilit, diğer salt okunur işlemlerin veritabanına erişmesine izin verir.

#### karşılıklı münhasırlık

**kuralı**

Aynı anda yalnızca bir işlemin aynı nesne üzerinde özel bir kilide sahip olabileceği bir durum.

#### kilitlenme

İki veya daha fazla işlemin, diğerinin bir kilit üzerindeki kilidi serbest bırakması süresiz olarak beklediği bir durum

önceden kilitlenmiş veri öğesi. *Ölümcül kucaklama* olarak da adlandırılır.

**iki aşamalı** kilitleme **(2PL)** İşlemlerin kilitleri nasıl alacağını ve bırakacağını yöneten bir dizi kural. İki aşamalı kilitleme serileştirilebilirliği garanti eder, ancak

kilitlenmeleri engellemez. İki aşamalı kilitleme protokolü iki aşamaya ayrılmıştır:

1. *Büyüme aşaması*, işlem *mevcut* veri kilitlerini açmadan ihtiyaç duyduğu kilitleri aldgerçekleşir. Tüm elde edildikten sonra, işlem *kilitli* noktasındadır. (2) İşlem tüm kilitleri serbest bıraktığında ve yeni bir kilit al*küçülme aşaması* gerçekleşir.

### Paylaşımlı / Özel

Erişim, nesneyi kilitleyen işlem için **özel** olarak ayrıldığında **özel** bir kilit mevcuttur. Çakışma olasılığı olduğunda özel kilit kullanılmalıdır (bkz. Tablo 10.11). **Paylaşılan** kilit, eşzamanlı işlemlere ortak bir kilit temelinde okuma erişimi verildiğinde ortaya çıkar. Paylaşılan kilit, tüm eşzamanlı işlemler salt okunur olduğu sürece çakışma yaratmaz.

Bir işlem veritabanından veri okumak istediğinde ve söz konusu veri öğesi üzerinde özel bir kilit bulunmadığında paylaşılan kilit verilir. Bir işlem bir veri öğesini güncellemek (yazmak) istediğinde ve o anda başka bir işlem tarafından o veri öğesi üzerinde hiçbir kilit tutulmadığında özel bir kilit verilir. Paylaşımlı/özel kilitleme konseptini kullanarak, bir kilit üç duruma sahip olabilir: kilitsiz, paylaşımlı (okuma) ve özel (yazma).

Tablo 10.11'de gösterildiği gibi, iki işlem yalnızca en az biri yazma işlemi olduğunda çakışır. İki okuma işlemi aynı anda güvenli bir şekilde yürütülebildiğinden, paylaşılan kilitler birkaç okuma işleminin aynı veri öğesini eşzamanlı olarak okumasına olanak tanır. Örneğin, T1 işlemi X veri öğesi üzerinde paylaşılan bir kilide sahipse ve T2 işlemi X veri öğesini okumak istiyorsa, T2 de X veri öğesi üzerinde paylaşılan bir kilit elde edebilir.

T2 işlemi X veri öğesini güncellerse, T2 tarafından veri öğesi üzerinde özel bir kilit gerekir

X. *Özel kilit, yalnızca ve yalnızca veri öğesi üzerinde başka hiçbir kilit yoksa verilir* (bu koşul **karşılıklı** *özel* **kural olarak** bilinir: bir seferde yalnızca bir işlem bir nesne üzerinde **özel** bir kilide sahip olabilir). Bu nedenle, T1 işlemi tarafından X veri öğesi üzerinde paylaşılan (veya özel) bir kilit zaten tutulmuşsa, T2 işlemine özel bir kilit verilemez ve T2, T1 işlemi kadar başlamak için beklemelidir. Başka bir deyişle, paylaşılan bir kilit her zaman özel (yazma) bir kilidi engeller; dolayısıyla işlem eşzamanlılığını azaltır.

Paylaşılan kilitlerin kullanımı veri erişimini daha verimli hale getirse de, paylaşılan/özel kilit

şeması kilit yöneticisinin ek yükünü çeşitli nedenlerle artırır:

* + Bir kilit verilmeden önce sahip olunan kilidin türü bilinmelidir.
  + Üç kilit işlemi mevcuttur: Kilit türünü kontrol etmek için READ\_LOCK, kilidi vermek için WRITE\_LOCK ve kilidi serbest bırakmak için UNLOCK.
  + Şema, paylaşımlıdan özele yükseltilmesine ve özelden paylaşımlıya düşürülmesine izin verecek

şekilde geliştirilmiştir.

Kilitler ciddi veri tutarsızlıklarını önlese de iki önemli soruna yol açabilir:

* + Ortaya çıkan işlem çizelgesi serileştirilebilir olmayabilir.
  + Program kilitlenmelere yol açabilir. İki işlem verinin kilidini açmak için süresiz olarak birbirini beklediğinde kilitlenme meydana gelir. Büyük bir şehirdeki trafik tıkanıklığına benzeyen bir veritabanı kilitlenmesi, iki veya daha fazla işlem verinin kilidini açmak için birbirini beklediğinde ortaya çıkar.

Neyse ki, her iki sorun da yönetilebilir: serileştirilebilirlik iki aşamalı kilitleme olarak bilinen bir kilitleme protokolü ile elde edilir ve kilitlenmeler kilitlenme tespit ve önleme teknikleri kullanılarak yönetilebilir. Bu teknikler sonraki iki bölümde incelenmektedir.

## 10-3c Serileştirilebilirliği Sağlamak için İki Aşamalı Kilitleme

**İki aşamalı kilitleme (2PL),** işlemlerin kilitleri nasıl edindiğini ve bıraktığını tanımlar. İki aşamalı

kilitleme serileştirilebilirliği garanti eder, ancak kilitlenmeleri engellemez. İki aşama şunlardır:

1. Bir işlemin herhangi bir verinin kilidini açmadan gerekli tüm kilitleri aldığı bir büyüme

aşaması. Tüm kilitler elde edildikten sonra, işlem kilitli noktasındadır.

1. Bir işlemin tüm kilitleri serbest bıraktığı ve yeni bir kilit elde edemediği bir küçülme aşaması.

İki aşamalı kilitleme protokolü aşağıdaki kurallara tabidir:

* + İki işlem çakışan kilitlere sahip olamaz.
  + Hiçbir kilit açma işlemi aynı işlemdeki bir kilitleme işleminden önce gelemez.
  + Tüm kilitler elde edilene kadar, yani işlem kilitli noktasına gelene kadar hiçbir veri etkilenmez. Şekil

10.7'de iki aşamalı kilitleme protokolü gösterilmektedir.

Bu örnekte, işlem önce ihtiyaç duyduğu iki kilidi alır. İki kilide sahip olduğunda, kilitli noktasına ulaşır. Ardından, veriler işlemin gereksinimlerine uyacak şekilde değiştirilir. Son olarak, işlem ilk aşamada edindiği tüm kilitleri serbest bırakarak tamamlanır. İki aşamalı kilitleme, işlem işleme maliyetini artırır ve kilitlenmeler gibi istenmeyen ek etkilere neden olabilir.

## d Kilitlenmeler

İki işlem birbirinin veri kilidini açmasını süresiz olarak beklediğinde kilitlenme meydana gelir.

Örneğin, iki işlem, T1 ve T2, aşağıdaki modda mevcut olduğunda bir kilitlenme meydana gelir:

T1 5 X ve Y veri öğelerine erişim T2 5 Y ve Xveri öğelerine erişim

T1 Y veri öğesinin kilidini açmamışsa T2 başlayamaz; T2 X veri öğesinin kilidini açmamışsa T1 devam edemez. Sonuç olarak, T1 ve T2'nin her biri diğerinin gerekli veri öğesinin kilidini açmasını bekler. Böyle bir kilitlenme **ölümcül kucaklaşma** olarak da bilinir. Tablo 10.13 bir kilitlenme durumunun nasıl yaratıldığını göstermektedir.

#### ölümcül kucaklaşma

*Çıkmaza* bakınız.

**Şekil 10.7**  **İki Fazlı Kilitleme Protokolü**

Kilit

edinm e

Kilit

edinm e

Kilitli nokta

Serbest bırak ma kilidi

Serbest bırak ma kilidi



Zaman 1 2 3 4 5 6 7 8

Başlangıç Operasyonlar Uç Kilitli

Büyüme aşaması faz Küçülme aşaması

Bir önceki örnekte kilitlenme durumunu göstermek için yalnızca iki eşzamanlı işlem kullanılmıştır. Gerçek bir VTYS'de aynı anda çok daha fazla işlem yürütülebilir ve böylece kilitlenme olasılığı artar. Kilitlenmelerin yalnızca işlemlerden biri bir veri öğesi üzerinde özel bir kilit elde etmek istediğinde mümkün olduğunu unutmayın; *paylaşılan* kilitler arasında kilitlenme durumu söz konusu olamaz.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tablo 10.13** | **Çıkmaz Durum Nasıl Oluşturulur?** | | | | |
| **Zaman** | | **İşlem** | **Yanıtla** | **Kilit Durumu** | |
|  | |  |  | **Veri X** | **Veri Y** |
| 0 | |  |  | Kilitli değil | Kilitli değil |
| 1 | | T1:KİLİT(X) | TAMAM. | Kilitli | Kilitli değil |
| 2 | | T2:KİLİT(Y) | TAMAM. | Kilitli | Kilitli |
| 3 | | T1:KİLİT(Y) | BEKLE | Kilitli | Kilitli |
| 4 | | T2:KİLİT(X) | BEKLE | Kilitli D | Kilitli |
| 5 | | T1:KİLİT(Y) | BEKLE | Kilitli e  a | Kilitli |
| 6 | | T2:KİLİT(X) | BEKLE | Kilitli d  l | Kilitli |
| 7 | | T1:KİLİT(Y) | BEKLE | Kilitli o | Kilitli |
| 8 | | T2:KİLİT(X) | BEKLE | c  Kilitli k | Kilitli |
| 9 | | T1:KİLİT(Y) | BEKLE | Kilitli | Kilitli |
| ... | | .............. | ........ | ......... | .......... |
| ... | | .............. | ........ | ......... | .......... |
| ... | | .............. | ........ | ......... | .......... |
| ... | | .............. | ........ | ......... | ......... |

#### zaman damgalama

İşlem yönetiminde, eşzamanlı işlemlerin zamanlanmasında kullanılan ve her işleme küresel bir benzersiz zaman damgası atayan bir teknik.

#### benzersizlik

Eşzamanlılık kontrolünde, eşit zaman damgası değerlerinin bulunmamasını sağlayan bir zaman damgası özelliği.

#### monotonluk

Zaman damgası değerlerinin her zaman artmasını sağlayan bir nitelik. (Eşzamanlı işlemlerin zamanlanmasına yönelik zaman damgası yaklaşımı, her işleme global ve benzersiz bir zaman damgası atar. Zaman damgası değeri açık bir sipariş üretir işlemlerin VTYS'ye gönderildiği yer).

Kilitlenmeleri kontrol etmek için üç temel teknik vardır:

* *Kilitlenme önleme*. Yeni bir kilit talep eden bir işlem, kilitlenme olasılığı olduğunda iptal . İşlem iptal edilirse, bu işlem tarafından yapılan tüm değişiklikler geri alınır ve işlem tarafından elde edilen tüm kilitler serbest bırakılır. İşlem daha sonra yürütme için yeniden planlanır. Kilitlenme önleme, kilitlenmeye yol açan koşulları önlediği için işe yarar.
* *Kilitlenme tespiti*. DBMS veritabanını periyodik olarak kilitlenmelere karşı test eder. Bir kilitlenme bulunursa, "kurban" işlem iptal edilir (geri alınır ve yeniden başlatılır) ve diğer işlem devam eder.
* *Kilitlenme önleme*. İşlem, yürütülmeden önce ihtiyaç duyduğu tüm kilitleri elde etmelidir. Bu teknik, kilitlerin art arda elde edilmesini gerektirerek çakışan işlemlerin geri alınmasını önler. Ancak, ölü kilitten kaçınmada gerekli olan seri kilit ataması işlem yanıt sürelerini artırır.

Hangi kilitlenme kontrol yönteminin kullanılacağının seçimi veritabanı ortamına bağlıdır. Örneğin, kilitlenme olasılığı düşükse kilitlenme tespiti önerilir. Ancak kilitlenme olasılığı yüksekse kilitlenmenin önlenmesi önerilir. Eğer yanıt süresi sistemin öncelik listesinde üst sıralarda yer almıyorsa, kilitlenmelerden kaçınma yöntemi kullanılabilir. Mevcut tüm DBMS'ler işlemsel veritabanlarında kilitlenme tespitini desteklerken, bazı DBMS'ler veri ambarları veya XML verileri gibi diğer veri türleri için önleme ve kaçınma tekniklerinin bir karışımını kullanır.

# Zaman Damgası Yöntemleri ile Eşzamanlılık Kontrolü

Eşzamanlı işlemlerin **zamanlanmasına** yönelik **zaman damgası** yaklaşımı, her bir işleme global ve benzersiz bir zaman damgası atar. Zaman damgası değeri, işlemlerin VTYS'ye gönderildiği açık bir sıra oluşturur. Zaman damgaları iki özelliğe sahip olmalıdır: teklik ve monotonluk. **Teklik**, eşit zaman damgası değerlerinin bulunmamasını sağlar ve **monotoniklik**1 zaman damgası değerlerinin her zaman artmasını sağlar.

1 *Monotonisite* terimi standart eşzamanlılık kontrolü sözlüğünün bir parçasıdır. Yazarların bu terimle ve doğru kullanımıyla ilk tanışması W. H. Kohler tarafından yazılan "A survey of techniques for synchronization and recovery in decentralized computer systems," *Computer Surveys* 3(2), June 1981, pp. 149-183 makalesinde olmuştur.

Aynı işlem içindeki tüm veritabanı işlemleri (okuma ve yazma) aynı zaman damgasına sahip olmalıdır. DBMS çakışan işlemleri zaman damgası sırasına göre yürütür ve böylece işlemlerin serileştirilebilirliğini sağlar. İki işlem çakışırsa, biri durdurulur, geri alınır, yeniden planlanır ve yeni bir zaman damgası değeri atanır.

Zaman damgası yaklaşımının dezavantajı, veritabanında depolanan her değerin iki ek zaman damgası alanı : biri alanın son okunduğu zaman ve diğeri son güncelleme için. Bu nedenle zaman damgası bellek ihtiyacını ve veritabanının işlem yükünü artırır. Zaman damgası çok fazla sistem kaynağı gerektirir çünkü birçok işlemin durdurulması, yeniden programlanması ve yeniden damgalanması gerekebilir.

## a Bekle/Öl ve Yara/Bekle Şemaları

Zaman damgası yöntemleri, eşzamanlı işlem yürütmeyi yönetmek için kullanılır. Bu bölümde, hangi işlemin geri alınacağına ve hangisinin yürütülmeye devam edeceğine karar vermek için kullanılan iki şema hakkında bilgi edineceksiniz: wait/die şeması ve wound/wait şeması.2 Bir örnek farkı göstermektedir. İki çakışan işleminiz olduğunu varsayın: T1 ve T2, her biri benzersiz bir zaman damgasına sahip. T1'in zaman damgasının 11548789 ve T2'nin zaman damgasının 19562545 olduğunu varsayalım. Zaman damgalarından T1'in daha eski işlem (daha düşük damgası değeri) ve T2'nin daha yeni işlem olduğu sonucunu çıkarabilirsiniz. Bu senaryo göz önüne alındığında, dört olası sonuç Tablo 10.14'te gösterilmektedir.

**Wait/Die ve Wound/Wait Eşzamanlılık Kontrol Şemaları**

**Tablo 10.14**

|  |  |  |  |
| --- | --- | --- | --- |
| **Kilit Talep Eden**  **İşlem** | **İşlem Sahibi Kilidi** | **Bekle/Öl Şeması** | **Yara/Bekleme Şeması** |
| T1 (11548789) | T2 (19562545) | * T1, T2 tamamlanana ve T2 kilitlerini serbest   bırakana kadar bekler. | * T1, T2'yi önler (geri alır). * T2 aynı zaman damgası kullanılarak yeniden   planlanır. |
| T2 (19562545) | T1 (11548789) | * T2 ölür (geri döner). * T2 aynı zaman damgası kullanılarak yeniden   planlanır. | * T2, T1 tamamlanana ve T1 kilitlerini serbest   bırakana kadar bekler. |

Bekle/öl şemasını kullanarak:

* Kilidi talep eden işlem iki işlemden daha eskiyse, diğer işlem tamamlanana ve kilitler serbest

bırakılana kadar *bekleyecektir*.

* Kilidi talep eden işlem iki işlemden daha küçükse, *ölür* (geri döner) ve aynı zaman damgası kullanılarak yeniden planlanır.

Kısacası, **bekle/öl** şemasında, daha yaşlı işlem daha genç olanın tamamlanmasını ve kilitlerini serbest bırakmasını bekler.

Yara/bekleme şemasında:

* Kilidi talep eden işlem iki işlemden daha eskiyse, daha genç olan işlemi geri döndürerek önceleyecektir (*yaralayacaktır*). T1, T2'yi geri aldığında T1, T2'yi öncelemiş olur. Öncelenen daha genç işlem aynı zaman damgası kullanılarak yeniden planlanır.
* Kilidi talep eden işlem iki işlemden daha küçükse, diğer işlem tamamlanana ve kilitler serbest

bırakılana kadar bekleyecektir.

Kısacası, **yara/bekleme** şemasında, daha eski işlem daha genç işlemi geri alır ve yeniden planlar.

2 Prosedür ilk olarak R. E. Stearnes ve P. M. Lewis II tarafından "System-level concurrency control for distributed database systems," *ACM Transactions on Database Systems*, No. 2, Haziran 1978, s. 178-198'de tanımlanmıştır.

#### bekle/öl

Daha eski bir işlemin, kilitleri kendisi talep etmeden önce daha genç işlemin tamamlanmasını ve kilitleri serbest bırakmasını beklemesi gereken bir eşzamanlılık kontrol şeması.

Aksi takdirde

daha yeni işlem ölür ve yeniden planlanır.

#### yara/bekle

Daha eski bir işlemin kilit talep edebileceği, daha genç işlemin önüne geçebileceği ve onu yeniden planlayabileceği bir eşzamanlılık kontrol şeması. Aksi takdirde, yeni işlem eski işlem bitene kadar bekler.

Her iki şemada da işlemlerden biri diğer işlemin bitirmesini ve kilitleri serbest bırakmasını bekler. Ancak birçok durumda bir işlem birden fazla kilit talebinde bulunur. Bir işlemin her kilit isteği için ne kadar beklemesi gerekir? Açıkçası, bu senaryo bazı işlemlerin süresiz olarak beklemesine ve kilitlenmeye neden olabilir. Kilitlenmeyi önlemek için her kilit isteğinin ilişkili bir zaman aşımı değeri vardır. Zaman aşımı süresi dolmadan kilit verilmezse, işlem geri alınır.

# İyimser Yöntemlerle Eşzamanlılık Kontrolü

**iyimser yaklaşım** İşlem yönetiminde, çoğu veritabanı işleminin çakışmadığı varsayımına dayanan bir eşzamanlılık kontrol tekniği.

#### kirli okuma

İşlem yönetiminde, bir işlem henüz verileri okur.

**tekrarlanamaz okuma** İşlem yönetiminde, bir işlem t1 zamanında belirli bir satırı okuduğunda, daha sonra t2 zamanında aynı satırı ve orijinal satır güncellenmiş veya silinmiş olabileceğinden farklı sonuçlar verir.

#### hayalet okuma

İşlem yönetiminde, bir işlem t1 zamanında bir sorgu yürüttüğünde, daha sonra t2 zamanında aynı sorguyu çalıştırarak sorguyu karşılayan ek satırlar elde eder.

**İyimser yaklaşım**, veritabanı işlemlerinin çoğunun çakışmadığı varsayımına dayanır. İyimser yaklaşım ne kilitleme ne de zaman damgası teknikleri gerektirir. Bunun yerine, bir işlem işlenene kadar kısıtlama olmaksızın yürütülür. İyimser bir yaklaşım kullanıldığında, her işlem *okuma*, *doğrulama* ve *yazma* olarak adlandırılan iki veya üç aşamadan geçer.(3)

* *Okuma aşamasında*, işlem veritabanını okur, gerekli işlemleri yürütür ve veritabanı değerlerinin özel bir kopyasında güncellemeler yapar. İşlemin tüm güncelleme işlemleri, diğer işlemler tarafından erişilmeyen geçici bir güncelleme dosyasına kaydedilir.
* Doğrulama *aşamasında*, yapılan değişikliklerin veritabanının bütünlüğünü ve tutarlılığını etkilemeyeceğinden emin olmak için işlem doğrulanır. Doğrulama testi olumlu sonuçlanırsa, işlem yazma aşamasına geçer. Doğrulama testi olumsuzsa, işlem yeniden başlatılır ve değişiklikler atılır.
* *Yazma aşamasında*, değişiklikler kalıcı olarak veritabanına uygulanır.

İyimser yaklaşım, az sayıda güncelleme işlemi gerektiren çoğu okuma veya sorgulama veritabanı sistemi için kabul edilebilir. Yoğun olarak kullanılan bir VTYS ortamında, kilitlenmelerin yönetimi - bunların önlenmesi ve tespiti - önemli bir VTYS işlevi oluşturur. VTYS, burada tartışılan tekniklerden birini veya daha fazlasını ve bu tekniklerin varyasyonlarını kullanacaktır. İşlem yönetiminin bir veritabanında nasıl uygulandığını daha iyi anlamak için, ANSI SQL 1992 standardında tanımlanan işlem izolasyon seviyeleri hakkında bilgi edinmeniz önemlidir.(4)

# ANSI İşlem Yalıtımı Düzeyleri

ANSI SQL standardı (1992) işlem yönetimini işlem izolasyon seviyelerine dayalı olarak tanımlar. İşlem izolasyon seviyeleri, işlem verilerinin diğer eşzamanlı işlemlerden "korunma veya izole edilme" derecesini ifade eder. İzolasyon seviyeleri, yürütme sırasında diğer işlemlerin hangi verileri görebileceğine (okuyabileceğine) bağlı olarak tanımlanır. Daha açık bir ifadeyle, işlem izolasyon seviyeleri bir işlemin izin verdiği veya vermediği "okuma" türlerine göre tanımlanır. Okuma işlemlerinin türleri şunlardır:

* **Kirli okuma**: bir işlem henüz verileri okuyabilir.
* **Tekrarlanamayan okuma**: Bir işlem t1 zamanında belirli bir satırı okur ve ardından t2 zamanında aynı satırı okuyarak farklı sonuçlar verir. Orijinal satır güncellenmiş veya silinmiş olabilir.
* **Hayali okuma**: bir işlem t1 zamanında bir sorgu yürütür ve ardından t2 zamanında aynı sorguyu çalıştırarak sorguyu karşılayan ek satırlar elde eder.

3 Eşzamanlılık kontrolüne iyimser yaklaşım H. T. King ve J. T. Robinson tarafından yazılan bir makalede açıklanmıştır, "Optimistic methods for concurrency control," *ACM Transactions on Database Systems* 6(2), Haziran 1981, s. 213-226. En güncel yazılımlar bile yirmi yıldan daha uzun bir süre önce geliştirilen kavramsal standartlar üzerine inşa edilmiştir.

4 "SQL, 1986 yılında ANSI X3.135'te standartlaştırıldı ve birkaç ay içinde ISO 9075-1987 tarafından benimsendi. SQL'in

uluslararası standart (şimdi ISO/IEC 9075) en son 2016 yılında olmak üzere periyodik olarak revize edilmiştir." See https://blog. ansi.org/2018/10/sql-standard-iso-iec-9075-2016-ansi-x3-135/

Yukarıdaki işlemlere dayanarak, ANSI dört işlem yalıtımı seviyesi tanımlamıştır: Read Uncommitted, Read Committed, Repeatable Read ve Serializable. Tablo 10.15 dört ANSI işlem yalıtımı düzeyini göstermektedir. Tabloda ayrıca Oracle ve MS SQL Server veritabanları tarafından sağlanan ek bir izolasyon düzeyi de gösterilmektedir.

**Read Uncommitted**, diğer işlemlerden işlenmemiş verileri okuyacaktır. Bu izolasyon seviyesinde, veritabanı verilere herhangi bir kilit koymaz, bu da işlem performansını artırır, ancak veri tutarlılığı pahasına. **Read Committed**, işlemleri yalnızca taahhüt edilen verileri okumaya zorlar. Bu, çoğu veritabanı için (Oracle ve SQL Server dahil) varsayılan çalışma modudur. Bu seviyede, veritabanı veriler üzerinde özel kilitler kullanır ve diğer işlemlerin orijinal işlem tamamlanana kadar beklemesine neden olur. **Tekrarlanabilir Okuma** izolasyon seviyesi, sorguların tutarlı sonuçlar döndürmesini sağlar. Bu tür izolasyon seviyesi, orijinal sorgu bir satırı okuduktan sonra diğer işlemlerin o satırı güncellememesini sağlamak için paylaşılan kilitler kullanır. Ancak, ilk sorgu çalıştığında bu satırlar mevcut olmadığından yeni satırlar okunur (hayalet okuma). **Serializable** izolasyon seviyesi ANSI SQL standardı tarafından tanımlanan en kısıtlayıcı seviyedir. Ancak, Serializable izolasyon seviyesinde bile kilitlenmelerin her zaman mümkün olduğunu unutmamak önemlidir. Çoğu veri tabanı işlem yönetimi için kilitlenme tespit yaklaşımı kullanır ve bu nedenle işlem doğrulama aşamasında "kilitlenmeleri" tespit eder ve işlemi yeniden planlar.

Taahhüt **Edilmemiş** Verileri **Oku** İşlemlerin taahhüt edilmemiş verileri okumasına izin veren bir ANSI SQL işlem izolasyon düzeyi

diğer işlemler ve tekrarlanamayan okumalara ve hayali okumalara izin verir. ANSI SQL tarafından tanımlanan en az kısıtlayıcı düzey.

**Read Committed** İşlemlerin yalnızca işlenen verileri okumasına izin veren bir ANSI SQL işlem izolasyon düzeyi. Bu, çoğu veritabanı için varsayılan işlem modudur.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tablo 10.15** | **İşlem Yalıtım Düzeyleri** | | | | | |
|  | | **İzolasyon Seviyesi** | **İzin verildi** | | | **Yorum** |
|  | |  | **Kirli Okuma** | **Tekrarlanamaz Okuma** | **Phantom Read** |
| Daha az kısıtlayıcı  Daha kısıtlayıcı | | Taahhütsüz Okuyun | Y | Y | Y | Taahhüt edilmemiş verileri okur ve tekrarlanamayan okumalara ve hayali okumalara izin verir. |
| Read Committed | N | Y | Y | Taahhütsüz veri okumalarına izin vermez ancak tekrarlanamayan okumalara ve hayali okumalara izin verir. |
| Tekrarlanabilir Okuma | N | N | Y | Sadece hayali okumalara izin verir. |
| Serializable | N | N | N | Kirli okumalara, tekrarlanamayan okumalara veya hayali okumalara izin vermez. |
| Yalnızca Oracle/SQL Server | | Salt Okunur / Anlık Görüntü | N | N | N | Oracle ve SQL Server tarafından desteklenir. İşlem yalnızca işlemin başladığı sırada işlenmiş olan değişiklikleri görebilir. |

Farklı izolasyon seviyelerinin nedeni işlem eşzamanlılığını artırmaktır. İzolasyon seviyeleri en az kısıtlayıcı olandan (Read Uncommitted) daha kısıtlayıcı olana (Seri- alizable) doğru gider. İzolasyon seviyesi yükseldikçe, işlem eşzamanlılığı performansı pahasına veri tutarlılığını artırmak için daha fazla kilit (paylaşılan ve özel) gerekir. Bir işlemin izolasyon seviyesi, örneğin genel ANSI SQL sözdizimi kullanılarak işlem deyiminde tanımlanır:

IŞLEM IZOLASYON SEVIYESINE BAŞLA OKUMA TAAHHÜT EDILDI

... SQL IFADELERI. COMMIT

IŞLEMI;

MySQL, Oracle ve MS SQL Server, belirli bir işlemin izolasyon düzeyini tanımlamak için SET TRANSACTION ISOLATION LEVEL deyimini kullanır. MySQL ve SQL Server dört ANSI izolasyon seviyesinin tamamını destekler. Oracle READ COMMITED ve SERIALIZABLE'ı destekler

**Tekrarlanabilir Okuma** Orijinal sorgu bir satırı güncelledikten sonra diğer işlemlerin o satırı güncellememesini sağlamak için paylaşılan kilitleri kullanan bir ANSI SQL işlem izolasyon düzeyi. Ancak, hayali okumalara izin verilir.

#### Serializable

Kirli okumalara, tekrarlanamayan okumalara veya hayali okumalara izin vermeyen bir ANSI SQL işlem izolasyon seviyesi; ANSI SQL standardı tarafından tanımlanan en kısıtlayıcı seviye.

ANSI standardı tarafından tanımlandığı gibi. Oracle, diğer iki izolasyon seviyesinin üstün uygulamasına sahip olduğunu öne sürer. Standardın izolasyon seviyelerini ulaşılması gereken hedefe göre değil, hangi sorunlara izin verildiğine göre tanımladığına dikkat edin. Oracle, standartta belirtilen sorunlara izin vermeden diğer izolasyon seviyelerinin hedeflerine ulaşan eşzamanlılık teknikleri kullanmaktadır. Örneğin, READ UNCOMMITED seviyesinin amacının herhangi bir işlem tarafından herhangi bir okuma erişiminin engellenmesini önlemek olduğu iddia edilebilir. Bunu yapmanın bir yolu, işlemlerin diğer işlemler tarafından yapılan taahhüt edilmemiş değişiklikleri okumasına izin vermektir (yani kirli okuma), ANSI standardı da bunu belirtir. Ancak Oracle'ın yaklaşımı, herhangi bir işlemin okuma erişimini asla engellemeyecek, ancak kirli okuma sorununa izin vermeyecek bir çoklu sürümleme sistemi kullanmaktır.

Önceki tartışmadan da görebileceğiniz gibi, işlem yönetimi karmaşık bir konudur ve veritabanları işlemlerin eş zamanlı yürütülmesini yönetmek için çeşitli tekniklerden yararlanır. Ancak, bazen veritabanını tutarlı bir duruma geri getirmek için veritabanı kurtarma tekniklerini kullanmak gerekebilir.

**veritabanı kurtarma** Bir veritabanını önceki tutarlı durumuna geri yükleme işlemi.

#### atomik işlem

**özelliği**

Bir işlemin tüm bölümlerinin tek bir mantıksal iş birimi olarak ele alınmasını gerektiren ve tüm işlemlerin

tutarlı bir veritabanı oluşturmak için tamamlanmalıdır (taahhüt edilmelidir).

# Veritabanı Kurtarma Yönetimi

**Veritabanı kurtarma**, bir veritabanını belirli bir durumdan (genellikle tutarsız) daha önceki tutarlı bir duruma geri yükler. Kurtarma teknikleri **atomik işlem özelliğine** dayanır: işlemin tüm bölümleri, tutarlı bir veritabanı üretmek için tüm işlemlerin uygulandığı ve tamamlandığı tek bir mantıksal iş birimi olarak ele alınmalıdır. Bir işlem herhangi bir nedenle tamamlanamazsa, işlem iptal edilmeli ve veritabanındaki tüm değişiklikler geri alınmalıdır (undone). Kısacası, işlem kurtarma işlemi, işlem iptal edilmeden önce işlemin veritabanında yaptığı tüm değişiklikleri tersine çevirir.

Bu bölümde *işlemlerin* kurtarılması üzerinde durulmuş olsa da, kurtarma teknikleri *veritabanına* ve bir tür kritik hata meydana geldikten sonra *sisteme* de uygulanır. Kritik olaylar bir veritabanının çalışmayı durdurmasına ve verilerin bütünlüğünün tehlikeye girmesine neden olabilir. Kritik olaylara örnekler şunlardır:

* *Donanım/yazılım arızaları*. Bu tür bir arıza sabit disk ortamındaki bir arıza, anakarttaki bozuk bir kapasitör ya da arızalı bir bellek bankası olabilir. Bu kategori altındaki diğer hata nedenleri arasında verilerin üzerine yazılmasına, silinmesine veya kaybolmasına neden olan uygulama programı veya işletim sistemi hataları yer alır. Bazı veritabanı yöneticileri bunun veritabanı sorunlarının en yaygın kaynaklarından biri olduğunu savunur.
* *İnsan kaynaklı olaylar*. Bu tür olaylar kasıtsız veya kasıtlı olarak kategorize edilebilir.

Kasıtsız bir hata, dikkatsiz bir son kullanıcıdan kaynaklanır. Bu tür hatalar arasında bir tablodan yanlış satırların silinmesi, klavyede yanlış tuşa basılması veya ana veritabanı sunucusunun yanlışlıkla kapatılması yer alır.

Ď

Kasıtlı olaylar daha ciddi niteliktedir ve normalde şirket verilerinin ciddi risk altında olduğunu gösterir. Bu kategori altında, veri kaynaklarına yetkisiz erişim sağlamaya çalışan bilgisayar korsanlarının neden olduğu güvenlik tehditleri ve veri tabanı işleyişini tehlikeye atmaya ve şirkete zarar vermeye çalışan hoşnutsuz çalışanların neden olduğu virüs saldırıları yer alır.

Ď

* + *Doğal afetler*. Bu kategori yangınları, depremleri, selleri ve elektrik kesintilerini içerir.

Nedeni ne olursa olsun, kritik bir hata veritabanını tutarsız bir duruma getirebilir. Aşağıdaki bölümde, veritabanını tutarsız bir durumdan tutarlı bir duruma kurtarmak için kullanılan çeşitli teknikler tanıtılmaktadır.

## 10-7a İşlem Kurtarma

Bölüm 10-1d'de, işlem günlüğünü ve veritabanı kurtarma amaçları için nasıl veri içerdiğini öğrendiniz. Veritabanı işlem kurtarma, bir veritabanını tutarsız bir durumdan tutarlı bir duruma kurtarmak için işlem günlüğündeki verileri kullanır.

Devam etmeden önce, iyileşme sürecini etkileyen dört önemli kavramı inceleyin:

* **write-ahead-log protokolü**, işlem günlüklerinin her zaman herhangi bir veritabanı verisi gerçekten güncellenmeden *önce* yazılmasını sağlar. Bu protokol, bir arıza durumunda veri tabanının daha sonra işlem günlüğündeki veriler kullanılarak tutarlı bir duruma getirilebilmesini sağlar.
* **Yedekli işlem günlükleri** (işlem birkaç kopyası), fiziksel bir disk arızasının DBMS'nin verileri kurtarma yeteneğini bozmamasını sağlar.
* Veritabanı **tamponları**, disk işlemlerini hızlandırmak için kullanılan birincil bellekteki geçici depolama alanlarıdır. İşlem süresini iyileştirmek için DBMS yazılımı verileri fiziksel diskten okur ve bir kopyasını birincil bellekteki bir "tamponda" saklar. Bir işlem verileri güncellediğinde, aslında verilerin tampondaki kopyasını günceller çünkü bu işlem her seferinde fiziksel diske erişmekten çok daha hızlıdır. Daha sonra, güncellenmiş verileri içeren tüm tamponlar tek bir işlem sırasında fiziksel diske yazılır ve böylece işlem süresinden önemli ölçüde tasarruf edilir.
* Veritabanı **kontrol noktaları**, DBMS'nin bellekteki tüm güncellenmiş tamponlarını (*kirli tamponlar* olarak da bilinir) diske yazdığı işlemlerdir. Bu işlem gerçekleşirken, DBMS başka herhangi bir istek yürütmez. Bir kontrol noktası işlemi de işlem günlüğüne kaydedilir. Bu işlemin sonucunda fiziksel veritabanı ve işlem günlüğü senkronize olur. Bu senkronizasyon gereklidir çünkü güncelleme işlemleri verilerin fiziksel veritabanındaki değil tamponlardaki kopyasını günceller. Kontrol noktaları, belirli operasyonel parametrelere göre (işlem günlüğü boyutu veya bekleyen işlemlerin hacmi için yüksek bir filigran gibi) DBMS tarafından otomatik olarak ve periyodik olarak yürütülür, ancak açık bir şekilde (bir veritabanı işlem deyiminin parçası olarak) veya dolaylı olarak (bir veritabanı yedekleme işleminin parçası olarak) da yürütülebilir. Elbette, çok sık kontrol noktaları işlem performansını etkileyecektir; çok seyrek kontrol noktaları ise veritabanı kurtarma performansını etkileyecektir. Her durumda, kontrol noktaları çok pratik bir işleve sahiptir. İleride göreceğiniz gibi, kontrol noktaları işlem kurtarmada da önemli bir rol oynar.

Veritabanı kurtarma işlemi, bir hatadan sonra veritabanının tutarlı bir duruma getirilmesini içerir. İşlem kurtarma prosedürleri genellikle ertelenmiş-yazma ve write- through tekniklerini kullanır.

Kurtarma prosedürü **ertelenmiş-yazma tekniği** (**ertelenmiş güncelleme** olarak da adlandırılır) kullandığında, işlem işlemleri fiziksel veritabanını hemen **güncellemez**. Bunun yerine, yalnızca işlem günlüğü güncellenir. Veritabanı, işlem günlüğündeki bilgiler kullanılarak yalnızca işlenen işlemlerden gelen verilerle fiziksel olarak güncellenir. İşlem, taahhüt noktasına ulaşmadan önce iptal edilirse, veritabanı hiç güncellenmediği için veritabanında hiçbir değişiklik (ROLLBACK veya undo) yapılması gerekmez. Bu örneği daha iyi görselleştirmek için Şekil 10.8'e bakın. Başlatılan ve işlenen tüm işlemler için kurtarma süreci ( önce) şu adımları izler:

1. İşlem günlüğündeki son kontrol noktasını tanımlar. Bu, işlem verilerinin diske fiziksel olarak en son kaydedildiği zamandır.
2. Son kontrol noktasından önce başlatılan ve işlenen bir işlem için, veriler zaten

kaydedilmiş olduğundan hiçbir şey yapılmasına gerek yoktur.

**write-ahead-log protokolü** Eşzamanlılık kontrolünde, aşağıdakileri sağlayan bir süreç

İşlem günlükleri, herhangi bir veritabanı verisi gerçekten güncellenmeden önce kalıcı depolama alanına yazılır.

İleriye yazma protokolü olarak da adlandırılır.

#### gereksiz işlem

**günlükleri**

Bir diskin fiziksel arızasının DBMS'nin verileri kurtarma yeteneğini bozmamasını sağlamak veritabanı yönetim sistemleri tarafından tutulan işlem günlüğünün birden fazla kopyası.

#### tampon

Disk işlemlerini hızlandırmak için kullanılan birincil bellekteki geçici depolama alanı.

#### kontrol noktası

İşlem yönetiminde, veritabanı yönetim sisteminin tüm güncellenmiş tamponlarını diske yazdığı bir işlem.

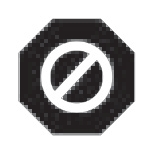
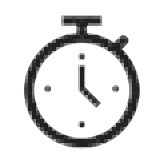
#### ertelenmiş-yazma tekniği

*Ertelenmiş güncellemeye* bakınız.

**ertelenmiş güncelleme** İşlem yönetiminde, işlem işlemlerinin fiziksel bir veritabanını hemen güncellemediği bir durum. *Ertelenmiş yazma tekniği* olarak da adlandırılır.

**Şekil 10.8**  **İşlemlerin Geri Kazanımı**

**Bölüm 10: İşlem Yönetimi ve Eşzamanlılık Kontrolü 499**



Zaman

Tüm taahhütler Kurtarma tüm işlemleri yeniden yapacaktır taahhütlü işlemler

kararlı (T2 ve T3) kalıcı

depolamaya kalıcı depolama. (T1).

8

7

6

5

4

3

2

1

T1

T2

Kurtarma, taahhüt edilmemiş veya iptal edilmiş tüm işlemleri geri alacaktır (T4).

T3

T4

Çarpış

ma

Kontrol Noktası

Eşzamanlı İşlemler

#### write-through tekniği

Eşzamanlılık kontrolünde, bir veritabanının, işlemin yürütülmesi sırasında, işlemden önce bile işlemler tarafından hemen güncellenmesini sağlayan bir işlem

işlem taahhüt noktasına ulaşır. *Anında güncelleme* olarak da adlandırılır.

#### acil güncelleme

Bkz. *write-through tekniği*.

1. Son kontrol noktasından sonra bir commit işlemi gerçekleştiren bir işlem için, DBMS işlemi yinelemek ve işlem günlüğündeki "sonra" değerlerini kullanarak veritabanını güncellemek için işlem kayıtlarını kullanır. Değişiklikler en eskiden en yeniye doğru artan sırada yapılır.
2. Son kontrol noktasından sonra bir ROLLBACK işlemi olan veya hata oluşmadan önce etkin bırakılan (ne COMMIT ne de ROLLBACK ile) herhangi bir işlem için, veritabanı hiç güncellenmediği için hiçbir şey yapılması gerekmez.

Kurtarma yordamı bir **write-through tekniği** (**anında güncelleme** olarak da adlandırılır) kullandığında, veritabanı işlemin yürütülmesi sırasında, işlem commit noktasına ulaşmadan önce bile işlem işlemleri tarafından hemen güncellenir. İşlem, taahhüt noktasına ulaşmadan önce iptal edilirse, veritabanını tutarlı bir duruma geri getirmek için bir ROLLBACK veya geri alma işleminin yapılması gerekir. Bu durumda, ROLLBACK işlemi işlem günlüğünün "önceki" değerlerini kullanacaktır. Kurtarma işlemi şu adımları izler:

1. İşlem günlüğündeki son kontrol noktasını tanımlar. Bu, işlem verilerinin diske fiziksel olarak en son kaydedildiği zamandır.
2. Son kontrol noktasından önce başlatılan ve işlenen bir işlem için, veriler zaten

kaydedilmiş olduğundan hiçbir şey yapılmasına gerek yoktur.

1. Son kontrol noktasından sonra işlenen bir işlem için DBMS, işlem günlüğünün "sonra" değerlerini kullanarak işlemi yeniden yapar. Değişiklikler en eskiden en yeniye doğru artan sırada uygulanır.
2. Son kontrol noktasından sonra bir ROLLBACK işlemi gerçekleştirilmiş veya hata oluşmadan önce etkin bırakılmış (ne COMMIT ne de ROLLBACK ile) herhangi bir işlem için DBMS, işlem günlüğündeki "önceki" değerleri kullanarak işlemleri ROLLBACK etmek veya geri almak için işlem günlüğü kayıtlarını kullanır. Değişiklikler en yeniden en eskiye doğru ters sırada uygulanır.

Basit bir veritabanı kurtarma işlemini izlemek için Tablo 10.16'daki işlem günlüğünü kullanın. Kurtarma sürecini anladığınızdan emin olmak için, basit işlem günlüğü üç işlem ve bir kontrol noktası içerir. Bu işlem günlüğü, bölümün önceki kullanılan işlem bileşenlerini içerir, bu nedenle temel sürece zaten aşina olmanız gerekir. İşlem göz önüne alındığında, işlem günlüğü aşağıdaki özelliklere sahiptir:

* İşlem 101, iki birim 54778-2T ürününün kredili satışı için 54778-2T ürününün eldeki miktarını

azaltan ve 10011 müşterisinin müşteri artıran iki GÜNCELLEME tablosundan oluşur.

* İşlem 106, Bölüm 10-1a'da gördüğünüz kredili satış olayının aynısıdır. Bu işlem, 89-WRE-Q ürününün bir biriminin 10016 numaralı müşteriye 277,55 $ karşılığında kredili satışını temsil eder. Bu işlem beş SQL DML deyiminden oluşur: üç INSERT deyimi ve iki UPDATE deyimi.
* İşlem 155 basit bir envanter güncellemesini temsil eder. Bu işlem, 2232/QWE ürününün eldeki miktarını 6 birimden 26 birime çıkaran bir UPDATE deyiminden oluşur.
* Bir veritabanı kontrol noktası, tüm güncellenmiş veritabanı arabelleklerini diske yazar. Kontrol noktası olayı yalnızca önceden işlenmiş tüm işlemlerin değişikliklerini yazar. Bu durumda, kontrol noktası 101 numaralı işlem tarafından yapılan tüm değişiklikleri veritabanı veri dosyalarına uygular.

Tablo 10.16'yı kullanarak, ertelenmiş güncelleme yöntemini kullanan bir VTYS için

veritabanı kurtarma sürecini aşağıdaki gibi izleyebilirsiniz:

1. Son kontrol noktasını (bu durumda TRL ID 423) tanımlayın. Bu, veritabanı tamponlarının

diske fiziksel olarak en son yazıldığı zamandır.

1. İşlem 101'in son kontrol noktasından önce başladığını ve bittiğini unutmayın. Bu nedenle, tüm değişiklikler zaten diske yazılmıştır ve ek bir işlem yapılmasına gerek yoktur.
2. Son kontrol noktasından (TRL ID 423) sonra işlenen her işlem için, DBMS değişiklikleri diske yazmak için "sonra" değerlerini kullanarak işlem günlüğü verilerini kullanacaktır. Örneğin, 106 numaralı işlem için:
   1. COMMIT'i bulun (TRL ID 457).
   2. İşlemin başlangıcını bulmak için önceki işaretçi değerlerini kullanın (TRL ID 397).
   3. Her DML deyimini bulmak için sonraki işaretçi değerlerini kullanın ve "sonra" değerlerini kullanarak değişiklikleri diske uygulayın. (TRL ID 405 ile başlayın, ardından 415, 419, 427 ve 431.) TRL ID 457'nin bu işlem için COMMIT deyimi olduğunu unutmayın.
   4. İşlem 155 için işlemi tekrarlayın.
3. Diğer tüm işlemler göz ardı edilir. Bu nedenle, ROLL- BACK ile biten veya aktif bırakılan işlemler için (COMMIT veya ROLLBACK ile bitmeyenler), diske hiçbir değişiklik yazılmadığı için hiçbir şey yapılmaz.

**Bölüm 10: İşlem Yönetimi ve Eşzamanlılık Kontrolü 501**

**İşlem Kurtarma Örnekleri için Bir İşlem Günlüğü**

**Bölüm 4:**

**500**

**İleri Veritabanı Kavramları**

Telif Hakkı 2023 Cengage Learning. Tüm hakları saklıdır. Tamamen veya kısmen kopyalanamaz, taranamaz veya çoğaltılamaz. Elektronik haklar nedeniyle, bazı üçüncü taraf içerikleri eKitap ve/veya eBölüm(ler)den çıkarılmış olabilir.

Editoryal inceleme, bastırılan herhangi bir içeriğin genel öğrenme deneyimini önemli ölçüde etkilemediğine karar vermiştir. Cengage Learning, sonraki hak kısıtlamalarının gerektirmesi halinde ek içeriği istediği zaman kaldırma hakkını saklı tutar.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tablo 10.16** | |  | | | | | | | | |
| **TL KIMLIĞI** | **TRX NUM** | | **ÖNCEKİ**  **PTR** | **SONRAKİ PTR** | **OPERASYON** | **TABLO** | **SIRA KİMLİĞİ** | **ATIF** | **ÖNCEKİ DEĞER** | **DEĞER SONRASI** |
| 341 | 101 | | Null | 352 | BAŞLA | \*\*\*\*İşlem Başlat |  |  |  |  |
| 352 | 101 | | 341 | 363 | GÜNCELLEME | ÜRÜN | 54778-2T | PROD\_QOH | 45 | 43 |
| 363 | 101 | | 352 | 365 | GÜNCELLEME | MÜŞTERİ | 10011 | CUST\_BALANCE | 615.73 | 675.62 |
| 365 | 101 | | 363 | Null | TAAHHÜT | \*\*\*\* İşlem Sonu |  |  |  |  |
| 397 | 106 | | Null | 405 | BAŞLA | \*\*\*\*İşlem Başlat |  |  |  |  |
| 405 | 106 | | 397 | 415 | INSERT | FATURA | 1009 |  |  | 1009,10016, ... |
| 415 | 106 | | 405 | 419 | INSERT | HAT | 1009,1 |  |  | 1009,1, 89-WRE-Q,1, ... |
| 419 | 106 | | 415 | 427 | GÜNCELLEME | ÜRÜN | 89-WRE-Q | PROD\_QOH | 12 | 11 |
| 423 |  | |  |  | KONTROL NOKTASI |  |  |  |  |  |
| 427 | 106 | | 419 | 431 | GÜNCELLEME | MÜŞTERİ | 10016 | CUST\_BALANCE | 0.00 | 277.55 |
| 431 | 106 | | 427 | 457 | INSERT | ACCT\_TRANSACTION | 10007 |  |  | 1007, 18-JAN-2022, ... |
| 457 | 106 | | 431 | Null | TAAHHÜT | \*\*\*\* İşlem Sonu |  |  |  |  |
| 521 | 155 | | Null | 525 | BAŞLA | \*\*\*\*İşlem Başlat |  |  |  |  |
| 525 | 155 | | 521 | 528 | GÜNCELLEME | ÜRÜN | 2232/QWE | PROD\_QOH | 6 | 26 |
| 528 | 155 | | 525 | Null | TAAHHÜT | \*\*\*\* İşlem Sonu |  |  |  |  |
| \* \* \* \* C \*R\*A\* S\* H \* \* \* \* | | | | | | | | | | |

# Özet

* Bir işlem, veritabanına erişen bir veritabanı işlemleri dizisidir. Bir işlem mantıksal bir çalışma birimidir; yani, tüm parçalar yürütülür veya işlem iptal edilir. Bir işlem, bir veritabanını bir tutarlı durumdan diğerine götürür. Tutarlı bir veritabanı durumu, tüm veri bütünlüğü kısıtlamalarının karşılandığı bir durumdur.
* İşlemlerin dört ana özelliği vardır: atomiklik, tutarlılık, izolasyon ve dayanıklılık. Atomiklik, işlemin tüm parçalarının yürütülmesi gerektiği anlamına gelir; aksi takdirde işlem iptal edilir. Tutarlılık, veritabanının tutarlı durumunun korunduğu anlamına gelir. İzolasyon, bir işlem tarafından kullanılan verilere, ilki tamamlanana kadar başka bir işlem tarafından erişilemeyeceği anlamına gelir. Dayanıklılık, bir işlem tarafından yapılan değişikliklerin işlem tamamlandıktan sonra geri alınamayacağı anlamına gelir. Buna ek olarak, işlem çizelgeleri serileştirilebilirlik özelliğine sahiptir - işlemlerin eşzamanlı yürütülmesinin sonucu, işlemlerin seri sırayla yürütülmesinin sonucuyla aynıdır.
* SQL, iki deyim kullanarak işlemler için destek sağlar: COMMIT, değişiklikleri diske kaydeder ve ROLLBACK, önceki veri tabanı durumunu geri yükler. SQL işlemleri birkaç SQL deyimi veya veritabanı isteği tarafından oluşturulur. Her veritabanı isteği birkaç G/Ç veritabanı işlemine yol açar. İşlem günlüğü, veritabanını değiştiren tüm işlemlerin kaydını tutar. İşlem günlüğünde saklanan bilgiler kurtarma (ROLLBACK) amacıyla kullanılır.
* Eşzamanlılık kontrolü, işlemlerin eşzamanlı yürütülmesini koordine eder. İşlemlerin eşzamanlı olarak üç ana soruna yol açabilir: kayıp güncellemeler, taahhüt edilmemiş veriler ve tutarsız geri alımlar. Zamanlayıcı, eşzamanlı işlemlerin sırasını belirlemekten sorumludur. İşlem yürütme sırası kritiktir ve çok kullanıcılı veritabanı sistemlerinde veritabanı bütünlüğünü sağlar. Zamanlayıcı, işlemlerin serileştirilebilirliğini sağlamak için kilitleme, zaman damgası ve optimizasyon yöntemlerini kullanır.
* Kilit, bir işlem tarafından bir veri öğesine benzersiz

erişimi garanti eder. Kilit, bir işlemin bir veri öğesine

# Anahtar Terimler

başka bir işlem kullanırken veri öğesini kullanmak. Çeşitli kilit seviyeleri vardır: veritabanı, tablo, sayfa, satır ve alan. Veri tabanı sistemlerinde iki tür kilit kullanılabilir: ikili kilitler ve paylaşımlı/özel kilitler. Bir ikili kilidin yalnızca iki durumu olabilir: kilitli (1) veya kilitsiz (0). Paylaşımlı kilit, bir işlem veritabanından veri okumak istediğinde ve başka hiçbir işlem aynı veriyi güncellemediğinde kullanılır. Belirli bir öğe için birden fazla paylaşımlı veya "okuma" kilidi bulunabilir. Bir işlem veritabanını güncellemek (yazmak) istediğinde ve veriler üzerinde başka hiçbir kilit (paylaşılan veya özel) özel kilit verilir.

* Programların serileştirilebilirliği iki aşamalı kilitleme kullanılarak garanti edilir. İki aşamalı kilitleme şeması, işlemin herhangi bir verinin kilidini açmadan ihtiyaç duyduğu tüm kilitleri edindiği bir büyüme aşamasına ve işlemin yeni kilitler edinmeden tüm kilitleri serbest bıraktığı bir küçülme aşamasına sahiptir. İki veya daha fazla işlem birbirlerinin bir kilidi serbest bırakmasını süresiz olarak beklerse, ölümcül kucaklaşma olarak da adlandırılan bir kilitlenme içindedirler. Üç kilitlenme kontrol tekniği vardır: önleme, tespit etme ve kaçınma.
* Zaman damgası yöntemleriyle eşzamanlılık kontrolü, her işleme benzersiz bir zaman damgası atar ve çakışan işlemlerin zaman damgası sırasına göre yürütülmesini planlar. Hangi işlemin geri alınacağına ve hangisinin yürütülmeye devam edeceğine karar vermek için iki şema kullanılır: bekle/öl şeması ve yara/bekle şeması.
* İyimser yöntemlerle eşzamanlılık kontrolü, veritabanı işlemlerinin çoğunun çakışmadığını ve işlemlerin verilerin özel, geçici kopyalarını kullanarak eşzamanlı olarak yürütüldüğünü varsayar. İşlem zamanında, özel kopyalar veritabanına güncellenir. ANSI standardı dört işlem izolasyon seviyesi tanımlar: Read Uncommitted, Read Committed, Repeatable Read ve Serializable.
* Veritabanı kurtarma, veritabanını belirli bir durumdan önceki tutarlı bir duruma geri yükler. Veritabanı kurtarma, donanım hatası veya uygulama hatası gibi kritik bir olay meydana geldiğinde tetiklenir.

atomik işlem özelliği

atomiklik ikili kilit tamponu

kontrol noktası eşzamanlılık kontrolü tutarlılık

tutarlı veritabanı durumu

veritabanı kurtarma veritabanı isteği veritabanı düzeyinde kilit kilitlenme

ölümcül kucaklaşma ertelenmi̇ş güncelleme

ertelenmiş-yazma tekniği

kirli okuma disk sayfası

dayanıklılığı özel kilit alan düzeyinde kilit anında güncelleme

tutarsız geri alma izolasyonu

kilit

kilit ayrıntı düzeyi kilit yöneticisi kayıp güncelleme

monotonluk

karşılıklı özel kural tekrarlanamaz okuma iyimser yaklaşım sayfa sayfa düzeyinde

kilit kötümser kilitleme hayalet okuma Okuma Taahhüt Edildi

Okuma İşlenmemiş yedek işlem günlüğü Tekrarlanabilir Okuma satır düzeyinde

kilit zamanlayıcı serileştirilebilir liği

Serializable serializable zamanlama paylaşılan kilit

tablo düzeyinde kilit zaman damgalama işlem işlem günlüğü

iki aşamalı kilitleme (2PL) taahhütsüz veri benzersizliği bekle/öl yara/bekle

write-ahead-log protokolü write-through tekniği

# İnceleme Soruları

1. Aşağıdaki ifadeyi açıklayınız: Bir işlem mantıksal bir iş

birimidir.

1. Tutarlı bir veritabanı durumu nedir ve nasıl elde edilir?
2. DBMS, işlemin semantik anlamının gerçek dünya olayını gerçekten temsil ettiğini garanti etmez. Bu sınırlamanın olası sonuçları nelerdir? Bir örnek veriniz.
3. Dört ayrı işlem özelliğini listeleyiniz ve tartışınız.
4. İşlemlerin serileştirilebilirliği ne anlama geliyor?
5. İşlem günlüğü nedir ve işlevi nedir?
6. Zamanlayıcı nedir, ne yapar ve eşzamanlılık kontrolü için

etkinliği neden önemlidir?

1. Kilit nedir ve genel olarak nasıl çalışır?
2. Farklı kilit ayrıntı düzeyleri nelerdir?
3. Sayfa düzeyinde bir kilit neden alan düzeyinde bir kilide tercih edilebilir?
4. Eşzamanlılık kontrolü nedir ve amacı nedir?
5. Münhasır kilit nedir ve hangi koşullar altında verilir?
6. Kilitlenme nedir ve nasıl önlenebilir? Kilitlenmelerle başa çıkmak için çeşitli stratejileri tartışın.
7. Eşzamanlılık kontrolü için zaman damgalama yöntemlerinin bazı dezavantajları nelerdir?
8. Eşzamanlılık kontrolü için iyimser bir yaklaşım kullanıldığında işlemlerin tamamlanması neden uzun sürebilir?
9. Veritabanı kurtarma sürecini tetikleyebilecek üç tür veritabanı açısından kritik olay nedir? Her biri için bazı örnekler verin.
10. Dört ANSI işlem izolasyon seviyesi nedir? Her seviye ne tür okumalara izin verir?

# Problemler

1. A, B ve C parçalarından oluşan bir ABC ürünü üreticisi olduğunuzu varsayalım. Her yeni ABC ürünü oluşturulduğunda, PRODUCT adlı bir tablodaki PROD\_QOH kullanılarak ürün envanterine eklenmelidir. Ayrıca, ürün her oluşturulduğunda, PART adlı bir tablodaki PART\_QOH kullanılarak parça envanteri A, B ve C parçalarının her biri kadar azaltılmalıdır.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tablo P10.1** |  | | | | |
| **Tablo Adı: Ürün** | | |  | **Masa Adı: Parça** | |
| PROD\_CODE | | PROD\_QOH | PART\_CODE | PART\_QOH |
| ABC | | 1,205 | A | 567 |
|  | |  | B | 98 |
|  | |  | C | 549 |

Yukarıdaki bilgileri göz önünde bulundurarak Problem 1a'dan 1e'ye kadar olan problemleri tamamlayınız.

* 1. Hem ÜRÜN hem de PARÇA için bir envanter güncellemesi için kaç veritabanı talebi tanımlayabilirsiniz?
  2. SQL kullanarak, Problem 1a'da tanımladığınız her bir veritabanı isteğini yazın.
  3. İşlem(ler)in tamamını yazın.
  4. Tablo 10.1'i şablon olarak kullanarak işlem günlüğünü yazın.
  5. Problem 1d'de oluşturduğunuz işlem günlüğünü kullanarak, veritabanı kurtarmadaki kullanımını izleyin.

1. Eşzamanlı işlem yürütme ile ilgili en yaygın üç sorunu tanımlayınız. Bu sorunlardan kaçınmak için eşzamanlılık

kontrolünün nasıl kullanılabileceğini açıklayın.

1. Eşzamanlılık kontrolünden hangi DBMS bileşeni sorumludur? Bu özellik çakışmaları

**Çevrimiçi İçerik**

Ch10\_ABC\_ Piyasalar veritabanı

[www.cengage.com](http://www.cengage.com/) adresinde mevcuttur. Problem 6-11'e çözüm bulmak bu veri tabanını kullanınız.

çözmek için nasıl kullanılır?

1. Basit bir örnek kullanarak, bir VTYS'de ikili ve paylaşımlı/özel kilitlerin kullanımını açıklayınız.
2. Veritabanı sisteminizin arızalandığını varsayalım. Veritabanı kurtarma sürecini ve

ertelenmiş-yazma ve write-through tekniklerinin kullanımını açıklayın.

1. ABC Marketleri müşterilere ürün satmaktadır. Şekil P10.6'da gösterilen ilişkisel diyagram ABC'nin veritabanı için ana varlıkları temsil etmektedir. Aşağıdaki önemli özelliklere dikkat ediniz:

* Bir müşteri, her biri bir fatura ile temsil edilen çok sayıda satın alma işlemi gerçekleştirebilir.

Ď CUS\_BALANCE her kredili satın alma veya ödeme ile güncellenir ve müşterinin borçlu olduğu tutarı temsil eder.

Ď CUS\_BALANCE her kredi alımında artar (1) ve her müşteri ödemesinde azalır (-).

Ď Son satın alma tarihi, müşteri tarafından yapılan her yeni satın alma ile güncellenir.

Ď Son ödeme tarihi, müşteri tarafından yapılan her yeni ödeme ile güncellenir.

* Fatura, bir müşteri tarafından satın alınan bir ürünü temsil eder.

Ď Bir FATURA, satın alınan her ürün için bir tane olmak üzere birçok fatura SATIRINA sahip olabilir.

Ď INV\_TOTAL, vergiler dahil olmak üzere faturanın toplam maliyetini temsil eder.

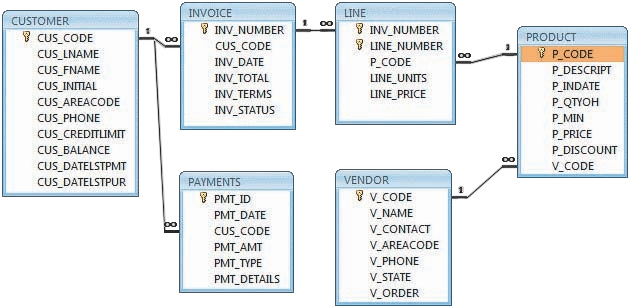
Ď INV\_TERMS "30," "60," veya "90" (kredi gün sayısını temsil eder) veya "CASH," "CHECK," veya "CC" olabilir.

Ď Fatura durumu "AÇIK, "ÖDENDİ" veya "İPTAL" olabilir.

* Bir ürünün eldeki miktarı (P\_QTYOH) her ürün satışında güncellenir (azalır).

**ABC Piyasaları İlişkisel Diyagramı**

**Şekil P10.6**



* Bir müşteri birçok ödeme yapabilir. Ödeme türü (PMT\_TYPE) aşağıdakilerden biri olabilir:

Ď Nakit ödemeler için "NAKİT".

Ď Çek ödemeleri için "ÇEK".

Ď Kredi kartı ödemeleri için "CC".

* Ödeme detayları (PMT\_DETAILS) çek veya kredi kartı ödemeleri ile ilgili verileri kaydetmek için kullanılır:

Ď Çek ödemeleri için banka, hesap numarası ve çek numarası.

Ď Kredi kartı ödemeleri için kartı veren kuruluş, kredi kartı numarası ve son kullanma tarihi.

*Not*: Bu örnekte tüm varlıklar ve öznitelikler temsil edilmemiştir. Yalnızca belirtilen öznitelikleri kullanın.

Bu veritabanını kullanarak, aşağıdaki işlemlerin her birini temsil edecek SQL kodunu yazın. SQL deyimlerini mantıksal işlemlerde gruplamak için BEGIN TRANSACTION ve COMMIT kullanın.

* 1. 11 Mayıs 2022 tarihinde, 10010 numaralı müşteri, birim fiyatı 110,00 $ olan 11QER/31 numaralı üründen bir birim kredili (30 gün) alım yapar; vergi oranı yüzde 8'dir. Fatura numarası 10983'tür ve bu faturada yalnızca bir ürün satırı vardır.
  2. 3 Haziran 2022'de, 10010 numaralı müşteri nakit olarak 100$'lık bir ödeme yapar. Ödeme kimliği 3428'dir.

1. Problem 6a ve 6b'deki işlemlerin eylemlerini temsil etmek için basit bir işlem günlüğü oluşturun (Tablo 10.14'te gösterilen

formatı kullanarak).

1. Kötümser kilitlemenin kullanıldığını ancak iki aşamalı kilitleme protokolünün kullanılmadığını varsayarak, Problem 6a'da açıklanan aktarımın tam olarak işlenmesi sırasında gerçekleşecek kilitleme, kilit açma ve veri işleme faaliyetlerinin kronolojik bir listesini oluşturun.
2. Kötümser kilitlemenin iki aşamalı kilitleme protokolü ile kullanıldığını varsayarak, Problem 6a'da açıklanan işlemin tam olarak

işlenmesi sırasında gerçekleşecek kilitleme, kilit açma ve veri manipülasyonu faaliyetlerinin kronolojik bir listesini oluşturun.

1. Kötümser kilitlemenin kullanıldığını ancak iki aşamalı kilitleme protokolünün kullanılmadığını varsayarak, Problem 6b'de açıklanan aktarımın tam olarak işlenmesi sırasında gerçekleşecek kilitleme, kilit açma ve veri işleme faaliyetlerinin kronolojik bir listesini oluşturun.
2. İki aşamalı kilitleme protokolü ile kötümser kilitlemenin satır düzeyinde kilit ayrıntı düzeyi ile kullanıldığını varsayarak, Problem 6b'de açıklanan işlemin tam olarak işlenmesi sırasında gerçekleşecek kilitleme, kilit açma ve veri manipülasyonu faaliyetlerinin kronolojik bir listesini oluşturun.