Veritabanı Yönetim Sistemleri

Dr. Öğr. Üyesi Ahmet Arif AYDIN

Hareket Yöetimi (Transaction Management)

Özet

- Sorgu Optimizasyonu
- Sorgunun değerlendirilmesinde hangi işlemler gerçekleştirilir?
- Sorguların oluşturulmasında selection, projection, join işlemleri neden önemlidir?
- partitioning, iteration, indexing
- tree and hash based indexing
- explain analyze

Hareket Yönetimi (Transaction Management)

Kullanıcı programın veritabanı içerisinde gerçekleştirmiş olduğu **CRUD işlemlerinin (execution)** her biri **hareket (transaction)** olarak isimlendirilir.

Para Transfer işlemi :

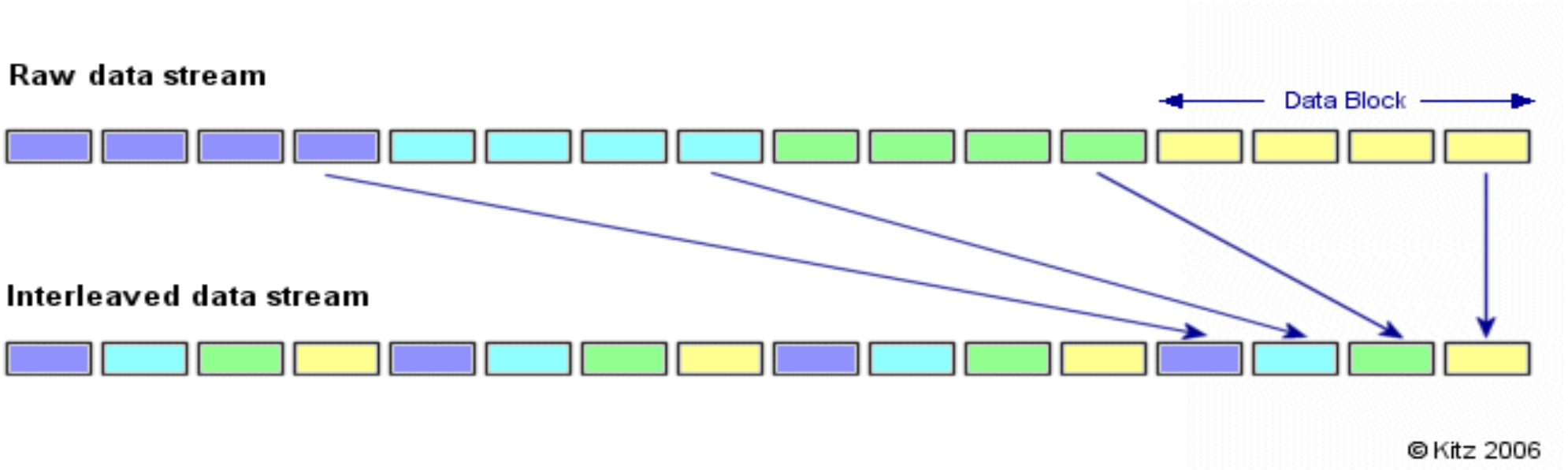
1. Banka Hesabını control
2. istenen miktar yeterli ise hesaptan düş
3. Banka hesabını güncelle (yeni miktar ile)
4. Arkadaşının hesabını kontrol et
5. Arkadaşının hesabına ıstenilen miktarı gönder
6. Arkadaşının hesabını güncelle

Hareket Yönetimi (Transaction Management)

Concurrency (eşzamanlılık) veritabanı yönetim sistemlerinin veriyi **sistem hatalarından korunmasının** (recovery from failure) temelini oluşturmaktadır.

Hareket Yönetimi (Transaction Management)

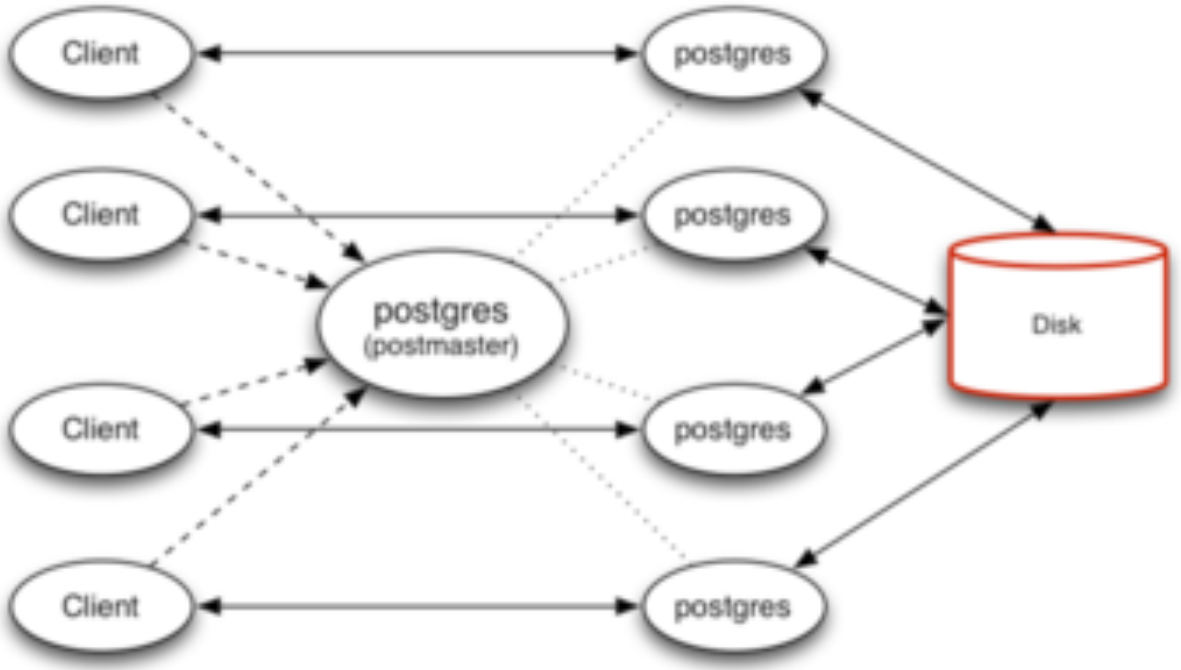
İstenilen performası gerçekleştirmek için VTYS hareketler arasında **interlaving** (dönüşümlü çalıştırmak) tekniğini kullanmaktadır.



https://kitz.co.uk/adsl/interleaving.htm

Hareket Yönetimi (Transaction Management)

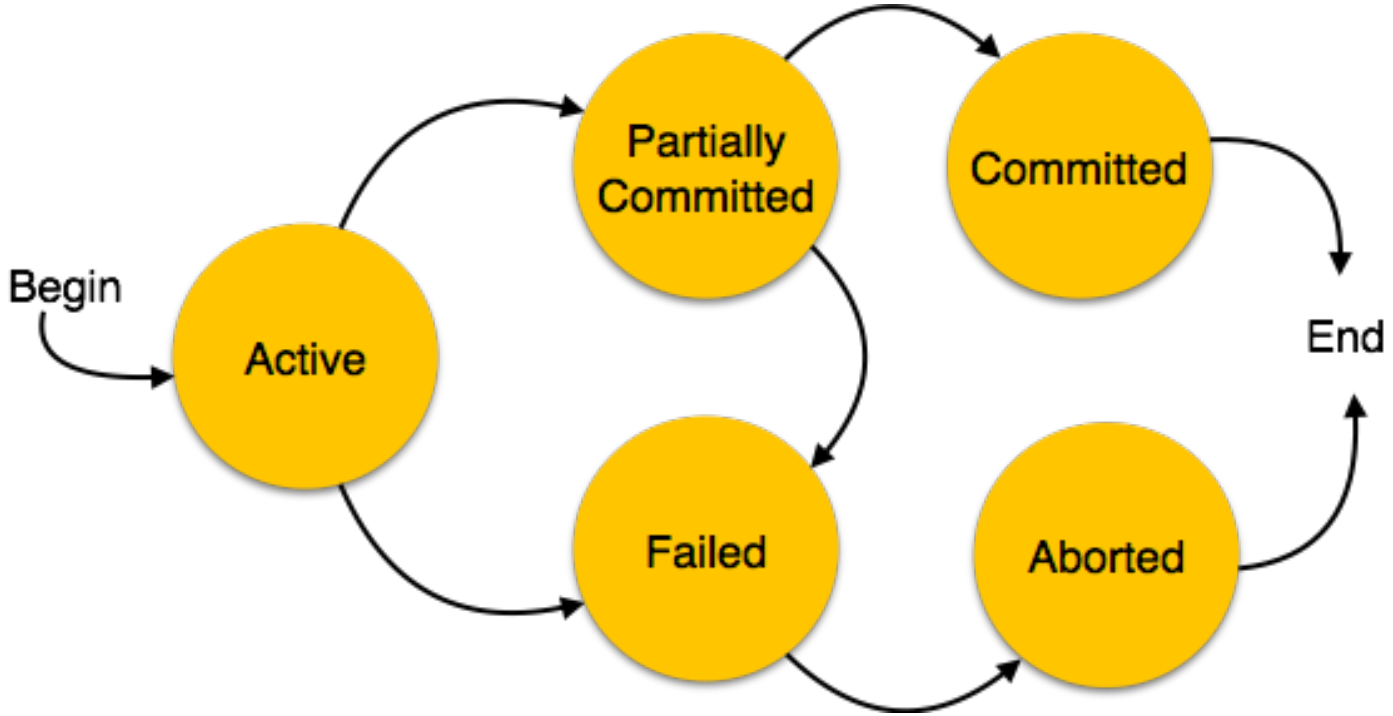
Veritabanı Yönetim Sistemleri eş zamanlı olarak gerçekleştirilen hareketlerin yönetimini ve kontrolünü sağlamaktadır. (concurrency control)



https://webcms3.cse.unsw.edu.au/COMP9315/16s1/resources/2352

Hareket Yönetimi (Transaction Management)

- Veritabanı yönetim sistemleri tamamlanamayan (partial- incomplete transactions) hareketleri de yönetmektedir
- Hareketlerin yönetimi gerçekleştirilirken aşağıdaki durumlar (states of transactions) gerçekleşebilir.

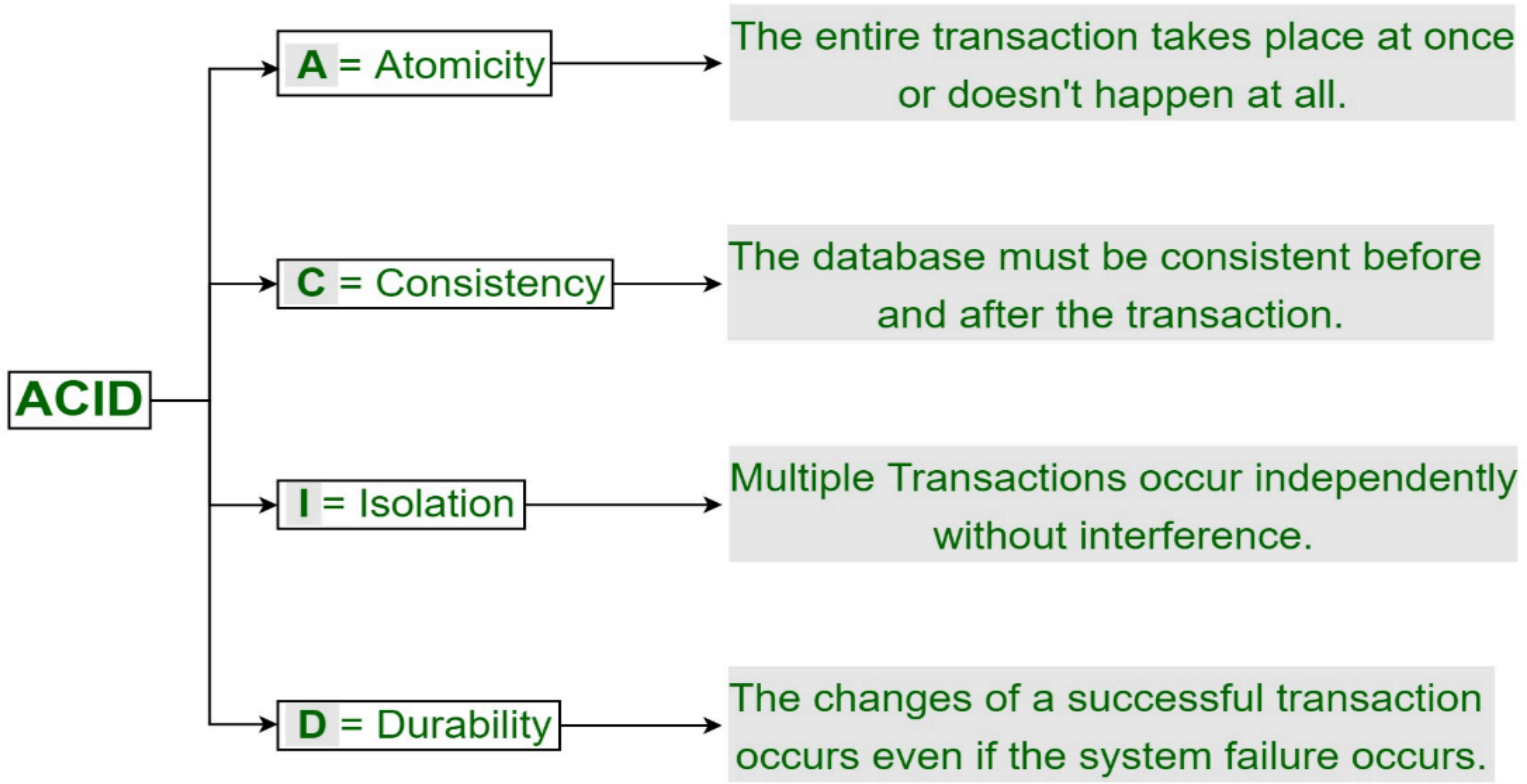


https://www.tutorialspoint.com/dbms/dbms_transaction.htm

Hareket Yönetimi (Transaction Management)

Veritabanı konseptinde hareketlerin dört temel özelliği bulunmaktadır.

Atomicity (bölünemezlik) , Consistency (tutarlılık) Isolation (yalıtım) , Durability (dayanıklılık)



https://www.geeksforgeeks.org/acid-properties-in-dbms/

ACID: Atomicity (bölünemezlik)

- Bir hareketin tamamlanabilmesi için kümelenmiş sıralı işlemlerin bölünmezliğidir.
- İşleminlerin sadece bir kısmı gerçekleştirilmez yani hepsi başlar biter veya işlem gerçekleştirilmez.
- Tamamlanamayan veya yarıda kalan hareketlerin işlemleri veritabanı tarafından geri alınır.

ACID: Atomicity (bölünemezlik)

- Bir hareketin tamamlanabilmesi için kümelenmiş sıralı işlemlerin bölünmezliğidir.
- İşleminlerin sadece bir kısmı gerçekleştirilmez yani hepsi başlar biter veya işlem gerçekleştirilmez.
- Tamamlanamayan veya yarıda kalan hareketlerin işlemleri veritabanı tarafından geri alınır.

Before: X : 500	Y: 200
Transaction T	
T1	**T2**
Read (X) X: = X – 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

ACID: Atomicity (bölünemezlik)

Mehmet'in hesabında 400tl bulunmaktadır

Kamil'in hesabında 700tl bulunmaktadır.

- Kamil'in Mehmet'e 200 tl gönderilmesi gerekmektedir.
- Bu transfer iki aşamada gerçekleşmektedir.
  1. Kamil'in hesabından 200tl düşülecek
  2. Mehmet'in hesabına 200 tl eklenecek

1. Başarılı

2. başarısız olursa

Mehmet'in hesabında (400tl) ve Kamil'in hesabında ise (500tl) olacak

ACID: Consistency (tutarlılık)

- Veritabanı üzerinde gerçekleştirilen hareketlerde veritabanında bulunan verinin tutarlılığının korunmasına **consistency** denir.
- Veritabanı gerçekleştirilen her işlemde tutarlılığın sağlandığını varsaymaktadır
- İlişkisel bütünlük (relational integrity) korunmaktadır.

Hesaplar arasında transfer yapılırken toplam miktar aynı kalmalıdır

Hareketten önce Kamil (700)+ Mehmet (400) = 1100
Hareket tamamlandıktan sonra Kamil (500)+ Mehmet (600) = 1100

ACID: İsolation (yalıtım)

- Veritabanı yönetim sistemlerinde birden fazla hareket yönetilirken bir hareketin diğer bir hareketten izole edilmesine **yalıtım** denir.

T	T''
Read (X) X: = X*100 Write (X) Read (Y) Y: = Y – 50 Write	Read (X) Read (Y) Z: = X + Y Write (Z)

Bir hesaptan aynı anda (T1 ve T2) para çekildiğinde önce başlanılan işlem (T1) tamamlanmadan diğerinin (T2) tamamlanması beklenir.

ACID: Durability (dayanıklılık)

- Sistem hataları ortaya çıktığında veya hareketlerde problemler ile karşılaşıldığında veritabanın hareket bilgilerini ve veritabanında kayıtlı olan veriyi kaybetmeden işleme devam edebilmesi **durability** olarak tanımlanır.

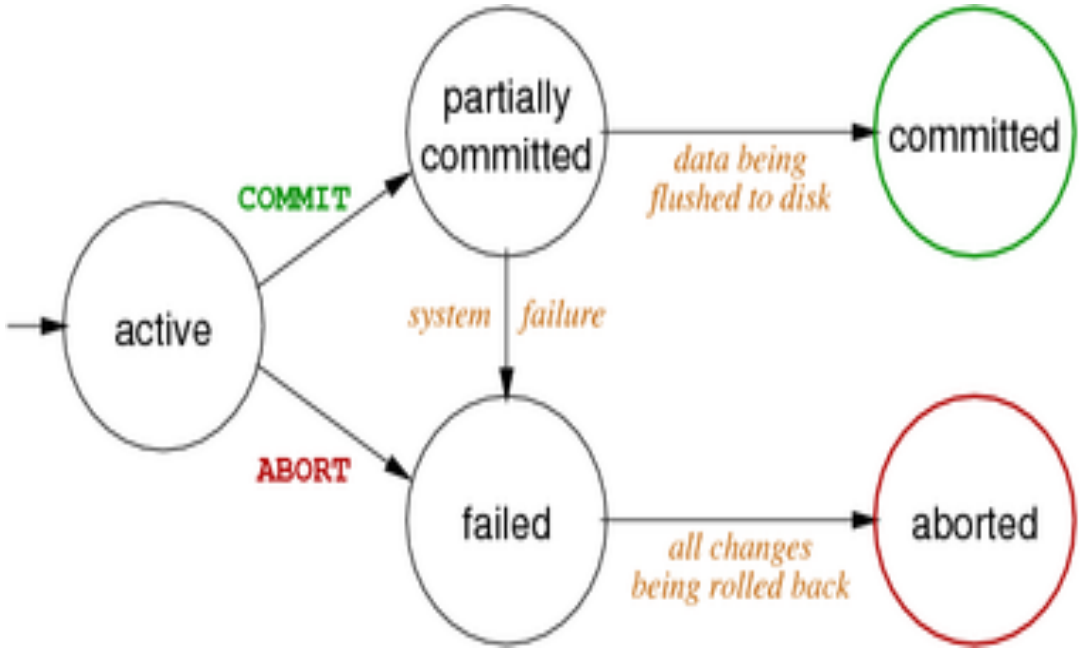
Hareket Yönetimi (Transaction Management)

- Bir banka hesabından geröekleştirilmek istenen transferrin tammalanmadan ortaya çıkan sistem hatasını **veritabanı yönetim sistemleri düzeltmek zorundadır**.

- VTYS gerçekleştirilen her işlemi log dosyalarına (hard diske) **WAL (Write-Ahead Log)** prensibini kullanarak yazmaktadır.

- Hataların düzeltilmesi işlemi VTYS tarafından periyodik olarak disk üzerindeki log dosyaları okunarak yapılır. Bu işlem **kontrol noktası (checkpoint)** olarak adlandırılır

Hareket Yönetimi (Transaction Management)



- Bir hareket içerisindeki bütün işlemler başarıyla tamamlanmışsa bu işlem **commited (teslim etmek)** olarak adlandırılır ve yapılan bütün değişiklikler diske kaydedilir.
- Bir hareket içerisinde herhangi bir hata oluştuğunda yapılan bütün değişiklikler geri alınarak hareket başlamadan önceki duruma geri dönülmesine **rollback (geri dönüş)** denir.

https://webcms3.cse.unsw.edu.au/COMP9315/16s1/resources/2352

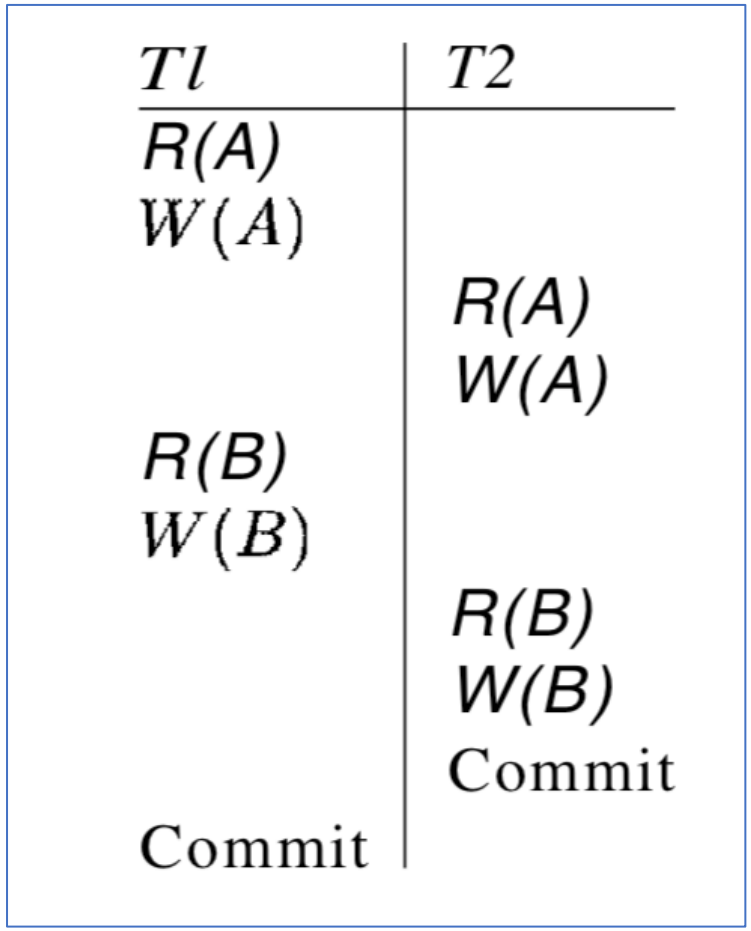
Hareket Planı (Transaction Plan)

- Birden fazla hareketin gerçekleştireceği işlemler (read, write, commit, abort) dizisine **schedule** (plan) denir
- Schedule veritabanı yönetim sistemleri tarafından yönetilir.

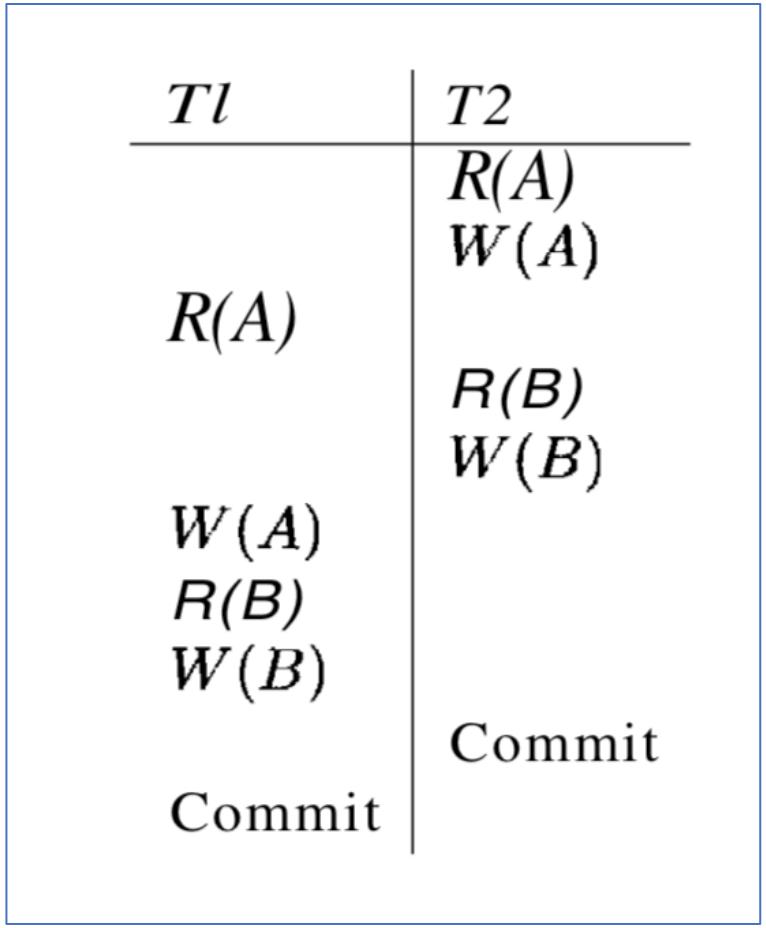
Hareket Yönetimi (Transaction Management)

- Birden fazla işlemin aynı zaman dilimi içerisinde gerçekleştirilmesine **concurrent execution (eşzamanlı çalışma)** denir.
- Belirlenen bir zaman dilimi içerisinde gerçekleştirilen ortalama işlem sayısına **throughput** (üretilen iş) denir
- Aynı zaman dilimi içerisinde gerçekleştirilen işlem performansını arttırmak için **concurrency** kullanılır.
- Sıralı işlemlerde (concurrent olmayan) kısa süreli bir hareket uzun zaman alan bir transaction ın ardında ise ne zaman biteceği tam olarak öngörülemeyebilir.

Hareket Yönetimi (Transaction Management)



T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	
W(B)	
	R(B)
	W(B)
	Commit
Commit	



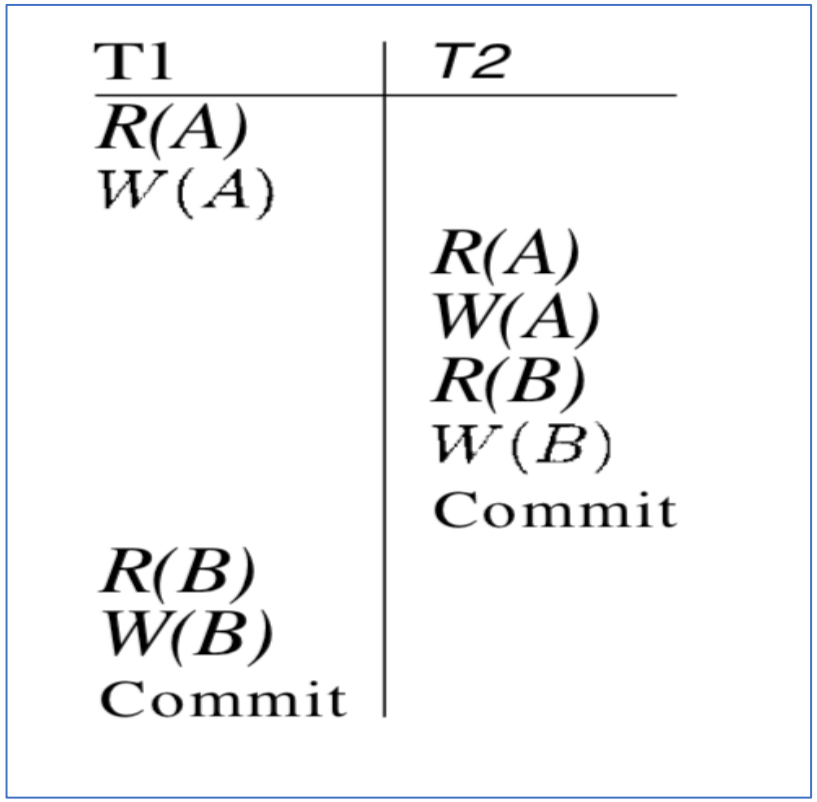
T1	T2
	R(A)
	W(A)
R(A)	
	R(B)
	W(B)
W(A)	
R(B)	
W(B)	
	Commit
Commit	

Serializable Schedule (sıra ile gerçekleştirilebilen plan)

Hareket Yönetimi Problemler (Transaction Management Problems)

Reading uncommitted data (WR Conflict)

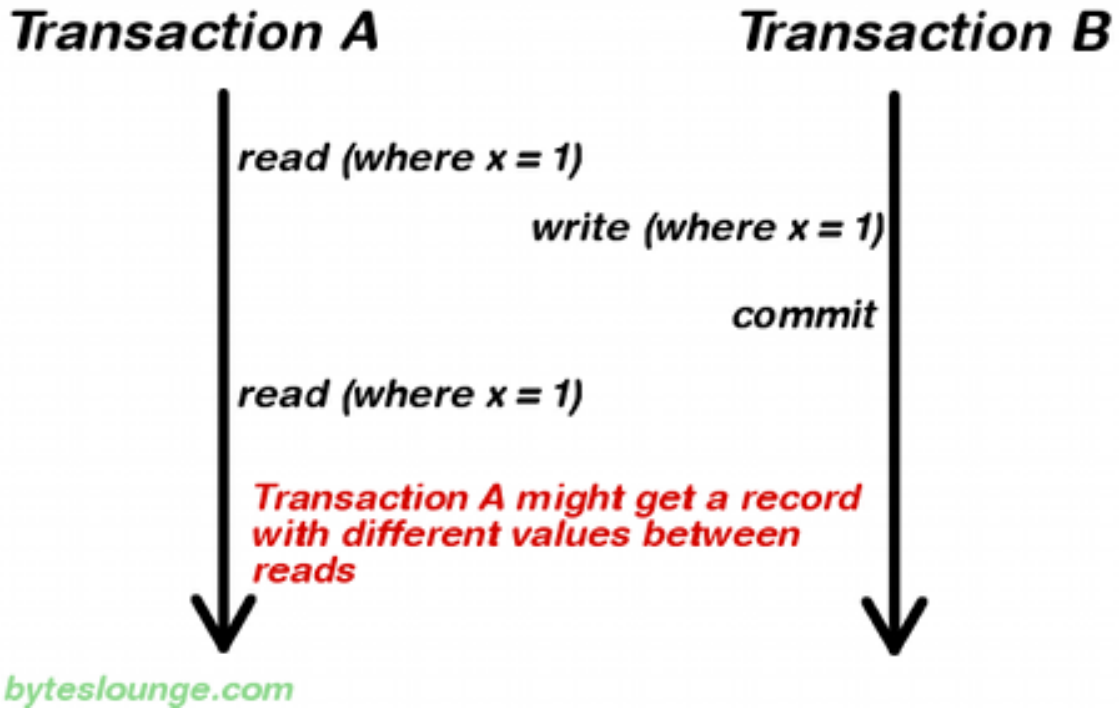
Bir hareket tarafından işlem yapılıp içeriği değiştirilen bir veriyi **commit** işlemi tamamlanmadan başka bir hareketin okuması



Hareket Yönetimi Problemler (Transaction Management Problems)

Unrepeatable Reads (RW Conflicts)

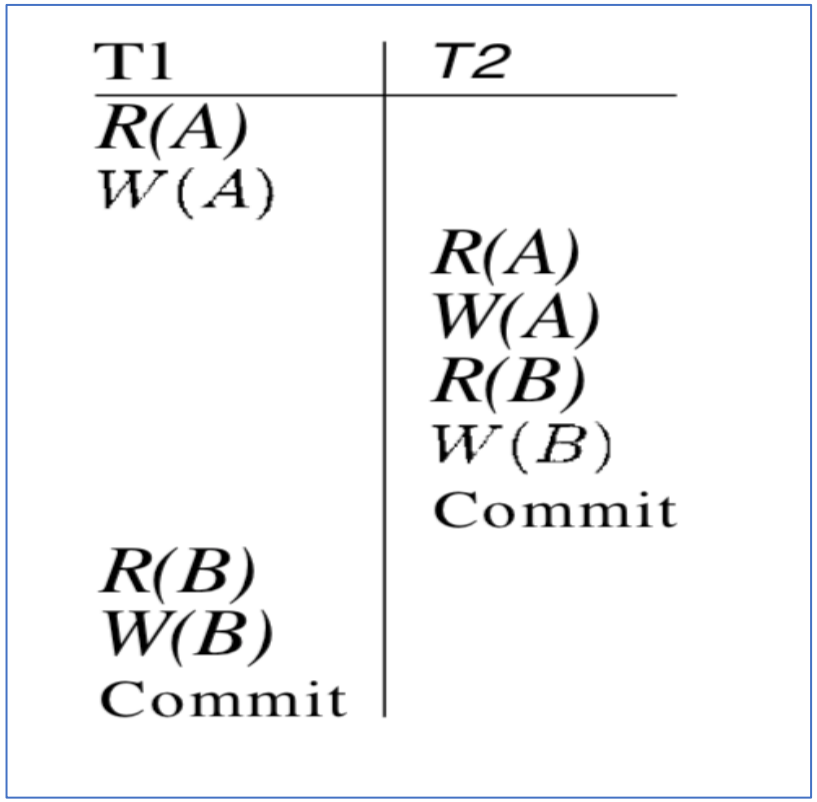
- TA'nın x değerini okuyup işlemi bitirmeden TB' nin x değerini değiştirmesi ve TA'nın tekrar x değerine erişim anında bir öncekinden faklı değer alması.
- Serial işlemlerde bu hata ile karşılaşılmaz.



Hareket Yönetimi Problemler (Transaction Management Problems)

Overwriting Uncommitted Data (WW Conflicts)

- T1 tarafından değiştirilen ve commit işlemi gerçekleştirlemeyen A değerinin , T2 tarafından tekrar değiştirilmesi



Hareket Yönetimi Problemler (Transaction Management Problems)

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
	R(B)
	W(B)
	Commit
Abort	

Unrecoverable schedule
(telafi edilemeyen plan)

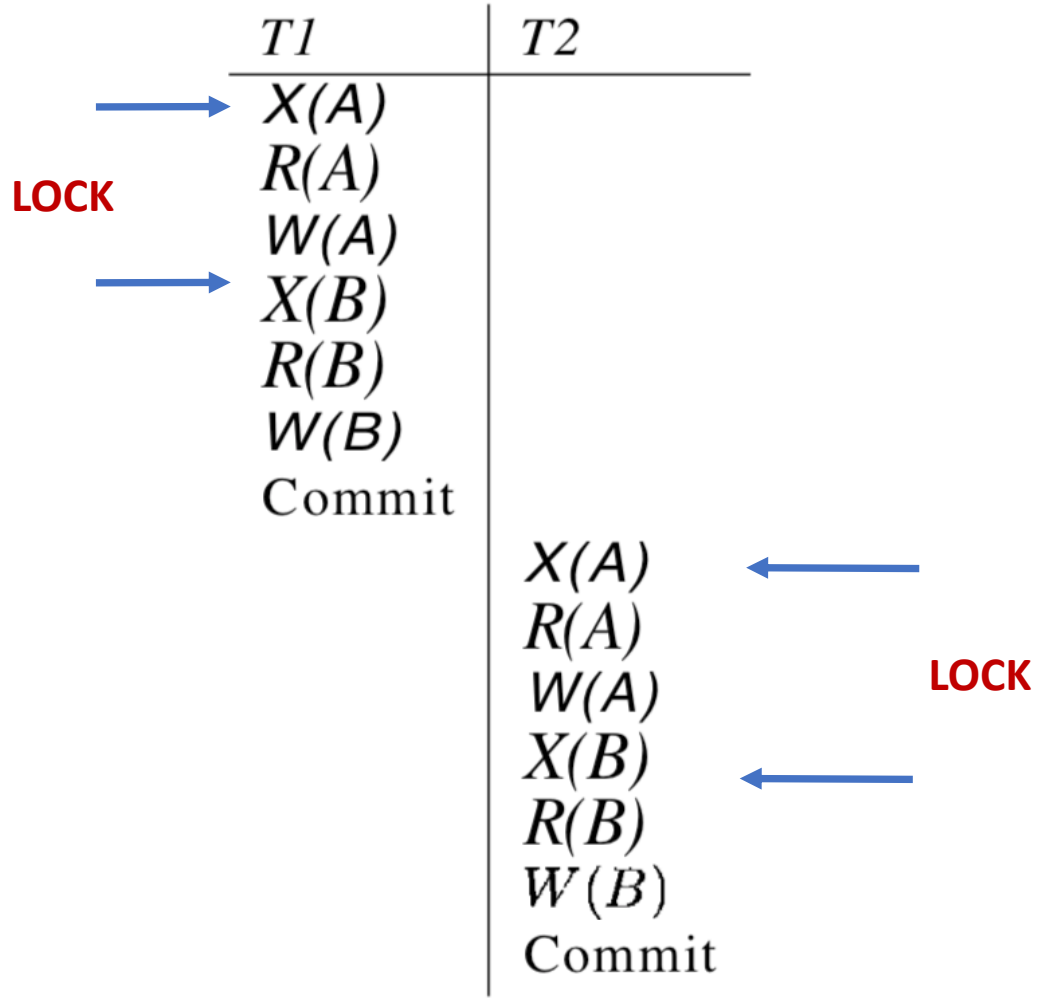
Anahtarlama Protokolleri (locking protocols)

- Bahsedilen problemlerin ortadan kaldırılması için VTYS tarafınddan **locking protocol** (anahtarlama protokolleri) kullanılmaktadır.
- Locking protocol her bir transaction tarafından uyulması gereken kurallardır.
- Farklı anahtarlama protokolleri farklı anahtar kullanabilirler.

Anahtarlama Protokolleri (locking protocols)

Strict Two-Phase Locking (Strict 2PL)

1. Bir nesne üzerinde okuma veya yazma işlemi gerçekleştirecek olan bir hareket VTYS den bir ortak anahtar (shared lock) istemektedir.
2. Nesne üzerinde işlem tamamlandığında bütün anahtarlar sisteme geri verilecektir.



PostgreSQL Transaction

```
BEGIN;

UPDATE accounts SET balance = balance - 100.00 WHERE name = 'Alice';

SAVEPOINT my_savepoint;

UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Bob';
-- oops ... forget that and use Wally's account

ROLLBACK TO my_savepoint;

UPDATE accounts SET balance = balance + 100.00 WHERE name = 'Wally';

COMMIT;
```