---

0. Database Concurrency Control

- **1 Purpose of Concurrency Control**
  - **To enforce Isolation (through mutual exclusion) among conflicting transactions.**
  - **To preserve database consistency through consistency preserving execution of transactions.**
  - **To resolve read-write and write-write conflicts.**
  - **Example:**
    In concurrent execution environment if T1 conflicts with T2 over a data item A, then the existing concurrency control decides if T1 or T2 should get the A and if the other transaction is rolled-back or waits

- 1غرض التحكم في التزامن
- لفرض العزلة (من خلال الاستبعاد المتبادل) بين المعاملات المتعارضة.
- للمحافظة علي تناسق قاعده البيانات من خلال التناسق مع المحافظة علي تنفيذ الحركات.
- لحل تعارضات القراءة والكتابة وكتابه الكتابة.

مثال:

- في بيئة التنفيذ المتزامنة إذا تعارض t1 مع T2 علي عنصر بيانات a ، ثم التحكم التزامن الموجودة يقرر إذا t1 أو T2 يجب الحصول علي a وإذا كانت المعاملة الأخرى التي يتم إرجاع أو ينتظر

1. Database Concurrency Control

Two-Phase Locking Techniques

- **Locking is an operation which secures**
  - **(a) permission to Read**
  - **(b) permission to Write a data item for a transaction.**
- **Example:**
  - **Lock (X). Data item X is locked in behalf of the requesting transaction.**
- **Unlocking is an operation which removes these permissions from the data item.**
- **Example:**
  - **Unlock (X): Data item X is made available to all other transactions.**
- **Lock and Unlock are Atomic operations.**

- تقنيات القفل علي مرحلتين
- التامين هو عمليه التي تؤمن
- (ا) الاذن بالقراءة
- (ب) الاذن بكتابه عنصر بيانات للحركة.
- مثال:
- Lock (X).يتم تامين عنصر البيانات X بالنيابة عن الحركة الطالبة.
- إلغاء التامين هو عمليه تزيل هذه الأذونات من عنصر البيانات.
- مثال:
- Unlock (x):يتم توفير عنصر البيانات X لكافة المعاملات الأخرى.
- قفل وفتح هي العمليات الذرية.

2. Database Concurrency Control

Two-Phase Locking Techniques: Essential components

- Two locks modes:
  - (a) shared (read) (b) exclusive (write).
- Shared mode: shared lock (X)
  - More than one transaction can apply share lock on X for reading its value but no write lock can be applied on X by any other transaction.
- Exclusive mode: Write lock (X)
  - Only one write lock on X can exist at any time and no shared lock can be applied by any other transaction on X.
- Conflict matrix

	Read	Write
Read	Y	N
Write	N	N

- تقنيات الاقفال علي مرحلتين: المكونات الاساسيه
- أوضاع تامين اثنين:
- (ا) مشتركه (قراءه) (ب) حصريه (كتابه).
- الوضع المشترك: القفل المشترك (X)
- يمكن لأكثر من معامله واحده تطبيق تامين المشاركة علي x لقراءه قيمتها ولكن لا يمكن تطبيق تامين الكتابة علي x بواسطة إيه معامله أخرى.
- الوضع الحصري: قفل الكتابة (س)
- يمكن ان يوجد قفل كتابه واحد فقط علي x في اي وقت ولا يمكن تطبيق اي تامين مشترك بواسطة اي معامله أخرى علي x.
- مصفوفة الصراع

3. Database Concurrency Control

Two-Phase Locking Techniques: Essential components

- Lock Manager:
  - Managing locks on data items.
- Lock table:
  - Lock manager uses it to store the identify of transaction locking a data item, the data item, lock mode and pointer to the next data item locked. One simple way to implement a lock table is through linked list.

Transaction ID	Data item id	lock mode	Ptr to next data item
T1	X1	Read	Next

- تقنيات الاقفال علي مرحلتين: المكونات الاساسيه
- أداره التامين:
- أداره التامين علي عناصر البيانات.
- تامين الجدول:
- يستخدم "أداره التامين" لتخزين تعريف المعاملات التي تقوم بتامين عنصر بيانات وعنصر البيانات ووضع التامين والمؤشر إلى عنصر البيانات التالي مؤمن. طريقه واحده بسيطه لتنفيذ جدول تامين هو من خلال قائمه مرتبطة.

Two-Phase Locking Techniques: Essential components

- Database requires that all transactions should be well-formed. A transaction is well-formed if:
  - It must lock the data item before it reads or writes to it.
  - It must not lock an already locked data items and it must not try to unlock a free data item.

- تقنيات الاقفال علي مرحلتين: المكونات الاساسيه
- تتطلب قاعده البيانات ان تكون كافة المعاملات مكونه بشكل جيد .يتم تكوين المعاملة بشكل جيد إذا:
- يجب تامين عنصر البيانات قبل قراءته أو كتابته.
- فانه يجب عدم تامين عناصر البيانات المؤمنة بالفعل ولا يجب محاولة إلغاء تامين عنصر بيانات حره.

Two-Phase Locking Techniques: Essential components

- The following code performs the lock operation:

```
B: if LOCK (X) = 0 (*item is unlocked*)
        then LOCK (X) ← 1 (*lock the item*)
        else begin
                wait (until lock (X) = 0) and
                the lock manager wakes up the transaction);
        goto B
        end;
```

- تقنيات الاقفال علي مرحلتين: المكونات الاساسيه
- التعليمه البرمجية التالية تنفذ عمليه التامين:

Two-Phase Locking Techniques: Essential components

- The following code performs the unlock operation:

```
        LOCK (X) ← 0 (*unlock the item*)
        if any transactions are waiting then
                wake up one of the waiting the transactions;
```

Two-Phase Locking Techniques: Essential components

- The following code performs the read operation:

```
        B: if LOCK (X) = "unlocked" then
begin LOCK (X) ← "read-locked";
        no_of_reads (X) ← 1;
end
else if LOCK (X) ← "read-locked" then
            no_of_reads (X) ← no_of_reads (X) +1
          else begin wait (until LOCK (X) = "unlocked" and
                    the lock manager wakes up the transaction);
                    go to B
                  end;
```

Two-Phase Locking Techniques: Essential components

- The following code performs the write lock operation:

```
B: if LOCK (X) = "unlocked" then
begin LOCK (X) ← "read-locked";
        no_of_reads (X) ← 1;
end
else if LOCK (X) ← "read-locked" then
            no_of_reads (X) ← no_of_reads (X) +1
         else begin wait (until LOCK (X) = "unlocked" and
                    the lock manager wakes up the transaction);
                    go to B
                  end;
```

Two-Phase Locking Techniques: Essential components

- The following code performs the unlock operation:

```
        if LOCK (X) = "write-locked" then
begin LOCK (X) ← "unlocked";
         wakes up one of the transactions, if any
end
else if LOCK (X) ← "read-locked" then
        begin
            no_of_reads (X) ← no_of_reads (X) -1
            if  no_of_reads (X) = 0 then
            begin
                  LOCK (X) = "unlocked";
                 wake up one of the transactions, if any
            end
        end;
```

Two-Phase Locking Techniques: Essential components

- Lock conversion
  - Lock upgrade: existing read lock to write lock

```
        if Ti has a read-lock (X) and Tj has no read-lock (X) (i ≠ j) then
    convert read-lock (X) to write-lock (X)
        else
    force Ti to wait until Tj unlocks X
```

  - Lock downgrade: existing write lock to read lock

```
        Ti has a write-lock (X)    (*no transaction can have any lock on X*)
        convert write-lock (X) to read-lock (X)
```

- قنيات الاقفال علي مرحلتين: المكونات الاساسيه
- تحويل القفل
- تامين الترقية: قراءه الموجودة قفل لكتابه قفل

Two-Phase Locking Techniques: The algorithm

- **Two Phases:**
  - **(a) Locking (Growing)**
  - **(b) Unlocking (Shrinking).**
- **Locking (Growing) Phase:**
  - **A transaction applies locks (read or write) on desired data items one at a time.**
- **Unlocking (Shrinking) Phase:**
  - **A transaction unlocks its locked data items one at a time.**
- **Requirement:**
  - **For a transaction these two phases must be mutually exclusively, that is, during locking phase unlocking phase must not start and during unlocking phase locking phase must not begin.**

- تقنيات القفل علي مرحلتين: الخوارزميه
- مرحلتين:
- (ا) التامين (النمو)
- (ب) الفتح (الانكماش).
- تامين (النمو) المرحلة:
- تطبق المعاملة الاقفال (القراءة أو الكتابة) علي عناصر البيانات المطلوبة واحده في كل مره.
- فتح (الانكماش) المرحلة:
- يقوم المعامل بإلغاء تامين عناصر البيانات المؤمنة الخاصة به في كل مره.
- شرط:
- النسبة للمعاملة ، يجب ان تكون هاتان المرحلتان حصريين ، اي انه اثناء مرحله اقفال المرحلة يجب الا يبدا التشغيل ولا يجب ان تبدا مرحله اقفال المرحلة.

Two-Phase Locking Techniques: The algorithm

T1	T2	Result
read_lock (Y);	read_lock (X);	Initial values: X=20; Y=30
read_item (Y);	read_item (X);	Result of serial execution
unlock (Y);	unlock (X);	T1 followed by T2
write_lock (X);	Write_lock (Y);	X=50, Y=80.
read_item (X);	read_item (Y);	Result of serial execution
X:=X+Y;	Y:=X+Y;	T2 followed by T1
write_item (X);	write_item (Y);	X=70, Y=50
unlock (X);	unlock (Y);	

Two-Phase Locking Techniques: The algorithm

T1	T2	Result
read_lock (Y);		X=50; Y=50
read_item (Y);		Nonserializable because it.
unlock (Y);		violated two-phase policy.
Time ↓	read_lock (X);	
	read_item (X);	
	unlock (X);	
	write_lock (Y);	
	read_item (Y);	
	Y:=X+Y;	
	write_item (Y);	
	unlock (Y);	
write_lock (X);		
read_item (X);		
X:=X+Y;		
write_item (X);		
unlock (X);		

Two-Phase Locking Techniques: The algorithm

T'1	T'2	
read_lock (Y);	read_lock (X);	T1 and T2 follow two-phase
read_item (Y);	read_item (X);	policy but they are subject to
write_lock (X);	Write_lock (Y);	deadlock, which must be
unlock (Y);	unlock (X);	dealt with.
read_item (X);	read_item (Y);	
X:=X+Y;	Y:=X+Y;	
write_item (X);	write_item (Y);	
unlock (X);	unlock (Y);	

Two-Phase Locking Techniques: The algorithm

- Two-phase policy generates two locking algorithms
  - (a) Basic
  - (b) Conservative
- Conservative:
  - Prevents deadlock by locking all desired data items before transaction begins execution.
- Basic:
  - Transaction locks data items incrementally. This may cause deadlock which is dealt with.
- Strict:
  - A more stricter version of Basic algorithm where unlocking is performed after a transaction terminates (commits or aborts and rolled-back). This is the most commonly used two-phase locking algorithm.

- تقنيات القفل علي مرحلتين: الخوارزميه
- يقوم نهج المرحلتين بإنشاء خوارزميات تامين اثنين
  - (ا) الاساسيه
  - (ب) المحافظة
- محافظ:
  - يمنع حاله التوقف التام بواسطة تامين كافة عناصر البيانات المطلوبة قبل بدء العملية التنفيذ.
- أساسي:
  - يقوم المعامل بتامين عناصر البيانات بشكل تزايدي .قد يؤدي ذلك إلى حاله توقف تام تم التعامل معها.
- صارم:
  - إصدار أكثر صرامة من الخوارزميه الاساسيه حيث يتم تنفيذ إلغاء التامين بعد إنهاء المعاملة (التزام أو الإجهاض والتراجع) .هذه هي خوارزميه القفل الأكثر شيوعا المستخدمة علي مرحلتين.

Dealing with Deadlock and Starvation

- **Deadlock**

T'1	T'2	
read_lock (Y);		T1 and T2 did follow two-phase policy but they are deadlock
read_item (Y);		
	read_lock (X);	
	read_item (Y);	
write_lock (X);		
(waits for X)		
	write_lock (Y);	
	(waits for Y)	

- Deadlock (T'1 and T'2)

Dealing with Deadlock and Starvation

- Deadlock prevention
  - A transaction locks all data items it refers to before it begins execution.
  - This way of locking prevents deadlock since a transaction never waits for a data item.
  - The conservative two-phase locking uses this approach.

- التعامل مع الجمود والمجاعة
- منع الجمود
  - يقوم المعامل بتامين كافة عناصر البيانات التي يشير اليها قبل بدء التنفيذ.
  - تمنع طريقه التامين هذه حاله التوقف التام لان المعاملة لا تنتظر أبدا لعنصر بيانات.
  - المحافظة علي مرحلتين التامين يستخدم هذا النهج.

- Dealing with Deadlock and Starvation
- Deadlock detection and resolution
  - **In this approach, deadlocks are allowed to happen. The scheduler maintains a wait-for-graph for detecting cycle. If a cycle exists, then one transaction involved in the cycle is selected (victim) and rolled-back.**
  - **A wait-for-graph is created using the lock table. As soon as a transaction is blocked, it is added to the graph. When a chain like: Ti waits for Tj waits for Tk waits for Ti or Tj occurs, then this creates a cycle. One of the transaction o**

- التعامل مع الجمود والمجاعة
- الكشف عن حاله توقف تام وحلها
  - وفي هذا النهج ، يسمح بحدوث الجمود. يحتفظ برنامج جدوله الانتظار للرسم البياني للكشف عن دوره. إذا كانت الدورة موجودة ، يتم تحديد أحدي الحركات المتضمنة في الدورة (الضحية) والتي يتم إرجاعها.
  - يتم إنشاء الانتظار للرسم البياني باستخدام جدول التامين. حالما يتم حظر المعاملة ، تتم اضافتها إلى الرسم البياني. عندما سلسله مثل: تي تنتظر تي جي ينتظر المعارف التقليدية لانتظار تي أو جي تي يحدث ، ثم هذا يخلق دوره. واحده من المعاملات o

- Dealing with Deadlock and Starvation
  - Deadlock avoidance
  - **There are many variations of two-phase locking algorithm.**
  - **Some avoid deadlock by not letting the cycle to complete.**
  - **That is as soon as the algorithm discovers that blocking a transaction is likely to create a cycle, it rolls back the transaction.**
  - **Wound-Wait and Wait-Die algorithms use timestamps to avoid deadlocks by rolling-back victim.**

- التعامل مع الجمود والمجاعة
- تجنب التوقف التام
- هناك العديد من الاختلافات من خوارزميه القفل علي مرحلتين.
- بعض تجنب الجمود من خلال عدم السماح للدورة لإكمال.
- وهذا هو بمجرد ان يكتشف الخوارزميه ان حظر المعاملة من المحتمل ان تخلق دوره ، فانه يعيد المعاملة.
- الجرح--الانتظار والانتظار--الموت الخوارزميات استخدام الطوابع الزمنيه لتجنب المازق من قبل الضحية المتداول الظهر.

- Dealing with Deadlock and Starvation
- Starvation
  - **Starvation occurs when a particular transaction consistently waits or restarted and never gets a chance to proceed further.**
  - **In a deadlock resolution it is possible that the same transaction may consistently be selected as victim and rolled-back.**
  - **This limitation is inherent in all priority based scheduling mechanisms.**
  - **In Wound-Wait scheme a younger transaction may always be wounded (aborted) by a long running older transaction which may create starvation.**

- التعامل مع الجمود والمجاعة
- مجاعة
- المجاعة يحدث عندما معامله معينه باستمرار ينتظر أو أعاده تشغيل وأبدا يحصل علي فرصه للمضي قدما.
- وفي حاله الجمود ، من الممكن ان يتم تحديد المعاملة نفسها باستمرار كضحية وان يتم إرجاعها.
- وهذا القيد متاصل في جميع أليات الجدولة القائمة علي الاولويه.
- في مخطط الجرح الانتظار قد تكون المعاملة الأصغر سنا دائما مجروحة (أجهضت) بواسطة معامله قديمه طويلة الأمد قد تتسبب في المجاعة.

- Timestamp based concurrency control algorithm
- Timestamp
  - **A monotonically increasing variable (integer) indicating the age of an operation or a transaction. A larger timestamp value indicates a more recent event or operation.**
  - **Timestamp based algorithm uses timestamp to serialize the execution of concurrent transactions.**

- خوارزميه التحكم بالتزامن المستندة إلى الطابع الزمني
- طابع
- متغير المتزايد بشكل رتيب (عدد صحيح) يشير إلى عمر العملية أو الحركة .تشير قيمه الطابع الزمني الأكبر إلى حدث أو عمليه أحدث.
- خوارزميه تستند الطابع الزمني يستخدم الطابع الزمني لتسلسل تنفيذ المعاملات المتزامنة.

- Timestamp based concurrency control algorithm
- Basic Timestamp Ordering
  - **1. Transaction T issues a write_item(X) operation:**
  - **If read_TS(X) > TS(T) or if write_TS(X) > TS(T), then an younger transaction has already read the data item so abort and roll-back T and reject the operation.**
  - **If the condition in part (a) does not exist, then execute write_item(X) of T and set write_TS(X) to TS(T).**
  - **2. Transaction T issues a read_item(X) operation:**
  - **If write_TS(X) > TS(T), then an younger transaction has already written to the data item so abort and roll-back T and reject the operation.**
  - **If write_TS(X) ≤ TS(T), then execute read_item(X) of T and set read_TS(X) to the larger of TS(T) and the current read_TS(X).**

- خوارزميه التحكم بالتزامن المستندة إلى الطابع الزمني
- ترتيب الطابع الزمني الأساسي
- 1-المعاملات T تصدر عمليه:write_item (X)
- إذا (read_TS س > (اتس (t) أو إذا write_TS س > (اتس (t) ، ثم المعاملات الأصغر سنا بالفعل قراءه عنصر البيانات حتى إحباط ولفه العودة T ورفض العملية.
- إذا كان الشرط في جزء (ا) غير موجود ، ثم تنفيذ write_item (x) من T و set write_TS (x) إلى TS (t).
- 2-المعاملة T تصدر عمليه:read_item (X)
- إذا) write_TS س > (اتس (T) ، ثم المعاملات الصغيرة قد كتب بالفعل إلى عنصر البيانات بحيث إحباط ولفه عوده T ورفض العملية.
- إذا كان) write_TS س (t) ≤ (، ثم تنفيذ read_item (س) من T وتعيين read_TS (س) إلى أكبر من اتس (T) والحالي) read_TS س.)

- Timestamp based concurrency control algorithm
- Strict Timestamp Ordering
  - **1. Transaction T issues a write_item(X) operation:**
  - **If TS(T) > read_TS(X), then delay T until the transaction T' that wrote or read X has terminated (committed or aborted).**
  - **2. Transaction T issues a read_item(X) operation:**

If TS(T) > write_TS(X), then delay T until the transaction T' that wrote or read X has terminated (committed or aborted).

- خوارزميه التحكم بالتزامن المستندة إلى الطابع الزمني
- صارمة ترتيب الطابع الزمني
- 1-المعاملات T تصدر عمليه:write_item (X)
- إذا اتس read_TS < (t) (س) ، ثم تاخير t حتى المعاملة ' t التي كتبت أو قراءه س قد انهي (الملتزمة أو أجهضت).
- 2-المعاملة T تصدر عمليه:read_item (X)
- إذا اتس write_TS < (t) (س) ، ثم تاخير t حتى المعاملة ' t التي كتبت أو قراءه س قد انهي (الملتزمة أو أجهضت).

- **Timestamp based concurrency control algorithm**
- **Thomas's Write Rule**
  - **If read_TS(X) > TS(T) then abort and roll-back T and reject the operation.**
  - **If write_TS(X) > TS(T), then just ignore the write operation and continue execution. This is because the most recent writes counts in case of two consecutive writes.**
  - **If the conditions given in 1 and 2 above do not occur, then execute write_item(X) of T and set write_TS(X) to TS(T).**

- خوارزميه التحكم بالتزامن المستندة إلى الطابع الزمني
- توماس كتابه القاعدة
- إذا read_TS س < (اتس (ر) ثم إحباط ولفه عوده T ورفض العملية.
- إذا) write_TSس < (اتس (T) ، ثم تجاهل فقط عمليه الكتابة ومتابعه التنفيذ .ويرجع ذلك إلى ان أحدث الكتابات تعول في حاله الكتابتين المتتاليتين.
- إذا كانت الشروط المعطية في 1 و 2 أعلاه لا تحدث, ثم تنفيذ write_item (x) من T و set write_TS (x) إلى TS (t).

- **Multiversion concurrency control techniques**
  - **This approach maintains a number of versions of a data item and allocates the right version to a read operation of a transaction. Thus unlike other mechanisms a read operation in this mechanism is never rejected.**
- **Side effect:**
  - **Significantly more storage (RAM and disk) is required to maintain multiple versions. To check unlimited growth of versions, a garbage collection is run when some criteria is satisfied.**

- تقنيات التحكم بالتزامن المتعدد الإصدارات
- يحافظ هذا النهج علي عدد من إصدارات عنصر البيانات ويخصص الإصدار الصحيح لعمليه قراءه لأحدي الحركات. وعلي عكس أليات الأخرى ، فان عمليه القراءة في هذه اليه لم ترفض أبدا.
- التاثير الجانبي:
- مطلوب تخزين أكثر بشكل ملحوظ (RAM وقرص) للاحتفاظ إصدارات متعددة. للتحقق من النمو غير المحدود للإصدارات ، يتم تشغيل مجموعه البيانات المهملة عند استيفاء بعض المعايير.

Multiversion technique based on timestamp ordering

- This approach maintains a number of versions of a data item and allocates the right version to a read operation of a transaction.
  - Thus unlike other mechanisms a read operation in this mechanism is never rejected.
- Side effects: Significantly more storage (RAM and disk) is required to maintain multiple versions. To check unlimited growth of versions, a garbage collection is run when some criteria is satisfied.

Multiversion technique based on timestamp ordering

- Assume X1, X2, …, Xn are the version of a data item X created by a write operation of transactions. With each Xi a read_TS (read timestamp) and a write_TS (write timestamp) are associated.
- read_TS(Xi): The read timestamp of Xi is the largest of all the timestamps of transactions that have successfully read version Xi.
- write_TS(Xi): The write timestamp of Xi that wrote the value of version Xi.
- A new version of Xi is created only by a write operation.

Multiversion technique based on timestamp ordering

- **To ensure serializability, the following two rules are used.**
- **If transaction T issues write_item (X) and version i of X has the highest write_TS(Xi) of all versions of X that is also less than or equal to TS(T), and read _TS(Xi) > TS(T), then abort and roll-back T; otherwise create a new version Xi and read_TS(X) = write_TS(Xj) = TS(T).**
- **If transaction T issues read_item (X), find the version i of X that has the highest write_TS(Xi) of all versions of X that is also less than or equal to TS(T), then return the value of Xi to T, and set the value of read _TS(Xi) to the largest of TS(T) and the current read_TS(Xi).**

- تقنيه الإصدار المتعدد استنادا إلى ترتيب الطابع الزمني
- لضمان امكانيه التسلسل ، يتم استخدام القاعدتين التاليتين.
- إذا كانت المعاملة t المسائل write_item (س) والإصدار الأول من x لديه اعلي write_TS (xi)من كافة إصدارات x التي هي أيضا اقل من أو يساوي اتس (t)، وقراءه ts _ الحادي عشر > (اتس (t) ، ثم إحباط ولفه الظهر ر ؛ والا إنشاء الإصدار الجديد الحادي عشر و read_TS س = write_TS (xsi)) اتس .(T).
- إذا المعاملات T المسائل ,read_item (X) العثور علي الإصدار الأول من x التي لديها اعلي write_TS (xi) من كافة إصدارات x التي هي أيضا اقل من أو يساوي اتس (t) ، ثم إرجاع قيمه الحادي عشر إلى T ، وتعيين قيمه القراءة (xi) إلى أكبر من اتس (T) والحالي.read_TS (Xi)

Multiversion technique based on timestamp ordering

- **To ensure serializability, the following two rules are used.**
  - **If transaction T issues write_item (X) and version i of X has the highest write_TS(Xi) of all versions of X that is also less than or equal to TS(T), and read _TS(Xi) > TS(T), then abort and roll-back T; otherwise create a new version Xi and read_TS(X) = write_TS(Xj) = TS(T).**
  - **If transaction T issues read_item (X), find the version i of X that has the highest write_TS(Xi) of all versions of X that is also less than or equal to TS(T), then return the value of Xi to T, and set the value of read _TS(Xi) to the largest of TS(T) and the current read_TS(Xi).**
- **Rule 2 guarantees that a read will never be rejected.**

- تقنيه الإصدار المتعدد استنادا إلى ترتيب الطابع الزمني
- لضمان امكانيه التسلسل ، يتم استخدام القاعدتين التاليتين.
- إذا كانت المعاملة t المسائل write_item (س) والإصدار الأول من x لديه اعلي write_TS (xi) من كافة إصدارات x التي هي أيضا اقل من أو يساوي اتس (t) ، وقراءه _ ts (الحادي عشر) > اتس (t) ، ثم إحباط ولفه الظهر ر ؛ والا إنشاء الإصدار الجديد الحادي عشر و read_TS (س) = write_TS (xsi) = اتس (T).
-
- إذا المعاملات T المسائل read_item (X) ,العثور علي الإصدار الأول من x التي لديها اعلي write_TS (xi) من كافة إصدارات x التي هي أيضا اقل من أو يساوي اتس (t) ، ثم إرجاع قيمه الحادي عشر إلى T ، وتعيين قيمه القراءة (xi) إلى أكبر من اتس (T) والحالي read_TS (Xi).
- وتكفل القاعدة 2 عدم رفض القراءة أبدا.

Multiversion Two-Phase Locking Using Certify Locks

- **Concept**
  - **Allow a transaction T' to read a data item X while it is write locked by a conflicting transaction T.**
  - **This is accomplished by maintaining two versions of each data item X where one version must always have been written by some committed transaction. This means a write operation always creates a new version of X.**

- متعدد الإصدارات التامين المرحلة الثانية باستخدام تامين الشهادات
- مفهوم
  - السماح بمعامله ' t لقراءه عنصر بيانات X اثناء الكتابة مؤمنه بواسطة معامله متعارضة.t
  - يتم إنجاز ذلك بالاحتفاظ بإصدارين من كل عنصر بيانات X حيث يجب ان يكون الإصدار الواحد مكتوبا دائما من قبل بعض المعاملات الملتزم بها .وهذا يعني عمليه الكتابة دائما بإنشاء إصدار جديد من.X

Multiversion Two-Phase Locking Using Certify Locks

- Steps
  1. X is the committed version of a data item.
  2. T creates a second version X' after obtaining a write lock on X.
  3. Other transactions continue to read X.
  4. T is ready to commit so it obtains a certify lock on X'.
  5. The committed version X becomes X'.
  6. T releases its certify lock on X', which is X now.

Compatibility tables for

	Read	Write
Read	yes	no
Write	no	no

read/write locking scheme

	Read	Write	Certify
Read	yes	no	no
Write	no	no	no
Certify	no	no	no

read/write/certify locking scheme

- **Multiversion Two-Phase Locking Using Certify Locks**
- **Note:**
  - **In multiversion 2PL read and write operations from conflicting transactions can be processed concurrently.**
  - **This improves concurrency but it may delay transaction commit because of obtaining certify locks on all its writes. It avoids cascading abort but like strict two phase locking scheme conflicting transactions may get deadlocked.**

- متعدد الإصدارات التامين المرحلة الثانية باستخدام تامين الشهادات
- مذكره:
- يمكن معالجه عمليات القراءة والكتابة المتعددة الإصدارات من المعاملات المتعارضة بشكل متزامن.
- وهذا يحسن التزامن لكنه قد يؤخر المعاملة التزام بسبب الحصول علي شهادات تامين علي كافة الكتابات الخاصة به .فانه يتجنب المتتالية إحباط ولكن مثل الصارمة مرحلتين تامين نظام المعاملات المتضاربة قد الحصول علي طريق مسدود.

- **Validation (Optimistic) Concurrency Control Schemes**
- **In this technique only at the time of commit serializability is checked and transactions are aborted in case of non-serializable schedules.**
- **Three phases:**

1. **Read phase**
2. **Validation phase**
3. **Write phase**

- **1. Read phase:**
  - **A transaction can read values of committed data items. However, updates are applied only to local copies (versions) of the data items (in database cache).**

- مخططات التحكم في التزامن (المتفائلة)
- في هذه التقنية فقط في وقت التزام التسلسل يتم التحقق ويتم إحباط الحركات في حاله الجداول غير القابلة للتسلسل.
- ثلاث مراحل:
  1. مرحله القراءة
  2. مرحله التحقق من الصحة
  3. مرحله الكتابة
- 1. قراءه المرحلة:
- يمكن للحركة قراءه قيم عناصر البيانات الملتزم بها. ومع ذلك ، يتم تطبيق التحديثات فقط علي النسخ المحلية (الإصدارات) من عناصر البيانات (في ذاكره التخزين المؤقت لقاعده البيانات).

- Validation (Optimistic) Concurrency Control Schemes
- 2. Validation phase: Serializability is checked before transactions write their updates to the database.
- This phase for Ti checks that, for each transaction Tj that is either committed or is in its validation phase, one of the following conditions holds:
  - **Tj completes its write phase before Ti starts its read phase.**
  - **Ti starts its write phase after Tj completes its write phase, and the read_set of Ti has no items in common with the write_set of Tj**
  - **Both the read_set and write_set of Ti have no items in common with the write_set of Tj, and Tj completes its read phase.**
  - **When validating Ti, the first condition is checked first for each transaction Tj, since (1) is the simplest condition to check. If (1) is false then (2) is checked and if (2) is false then (3) is checked. If none of these conditions holds, the validation fails and Ti is aborted.**

- مخططات التحكم في التزامن (المتفائلة)
- -2مرحله التحقق من الصحة: يتم التحقق من امكانيه التسلسل قبل ان تقوم الحركات بكتابه التحديثات الخاصة بها إلى قاعده البيانات.
- هذه المرحلة ل Ti يتحقق من انه بالنسبة لكل المعاملات التي يتم التزام بها أو التي هي في مرحله التحقق من الصحة الخاصة به ، فان أحد الشروط التالية يحمل:
- تي جي يكمل مرحله الكتابة قبل Ti يبدا مرحله القراءة.
- tiيبدا مرحله الكتابة بعد تي جيه يكمل مرحله الكتابة ، و read_set من تي ليس لديه عناصر مشتركه مع write_set من تي جي
- كل من read_set و write_set من Ti ليس لديهم العناصر المشتركة مع write_set من تي جي ، و تي جي يكمل مرحله القراءة.
- عند التحقق من صحة Ti ، يتم فحص الشرط الأول أولا لكل معامله تي جي ، منذ (1) هو ابسط شرط للتحقق .إذا كان (1) هو كاذبه ثم (2) يتم التحقق وإذا كان (2)هو كاذبه ثم (3) يتم التحقق .في حاله عدم وجود اي من هذه الشروط ، يفشل التحقق من الصحة ويتم إحباط.Ti

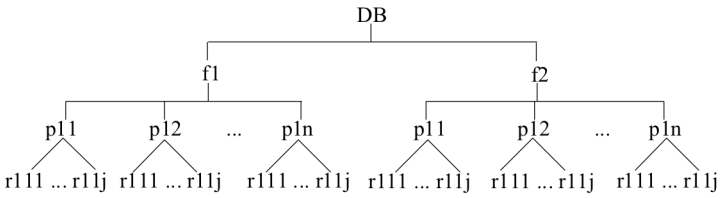
- Validation (Optimistic) Concurrency Control Schemes
- 3. Write phase: On a successful validation transactions' updates are applied to the database; otherwise, transactions are restarted.

- مخططات التحكم في التزامن (المتفائلة)
- .3مرحله الكتابة: يتم تطبيق التحديثات الناجحة لعمليات التحقق من الصحة علي قاعده البيانات ؛ والا ، يتم أعاده تشغيل الحركات.

- Granularity of data items and Multiple Granularity Locking
- A lockable unit of data defines its granularity. Granularity can be coarse (entire database) or it can be fine (a tuple or an attribute of a relation).
- Data item granularity significantly affects concurrency control performance. Thus, the degree of concurrency is low for coarse granularity and high for fine granularity.
- Example of data item granularity:

1. **A field of a database record (an attribute of a tuple)**
2. **A database record (a tuple or a relation)**
3. **A disk block**
4. **An entire file**
5. **The entire database**

- Granularity of data items and Multiple Granularity Locking
- The following diagram illustrates a hierarchy of granularity from coarse (database) to fine (record).



- التفاصيل الخاصة بعناصر البيانات وتامين العديد من الخصائص الحبيبية
- يوضح الرسم التخطيطي التالي تسلسل هرمي لمجموعه التفرعات من الخشن (قاعده البيانات) إلى الغرامة (السجل).

Granularity of data items and Multiple Granularity Locking

- To manage such hierarchy, in addition to read and write, three additional locking modes, called intention lock modes are defined:
  - **Intention-shared (IS): indicates that a shared lock(s) will be requested on some descendent nodes(s).**
  - **Intention-exclusive (IX): indicates that an exclusive lock(s) will be requested on some descendent node(s).**
  - **Shared-intention-exclusive (SIX): indicates that the current node is locked in shared mode but an exclusive lock(s) will be requested on some descendent nodes(s).**

- التفاصيل الخاصة بعناصر البيانات وتامين العديد من الخصائص الحبيبية
- لأداره مثل هذا التسلسل الهرمي ، بالاضافه إلى القراءة والكتابة ، يتم تعريف ثلاثه أوضاع تامين اضافيه ، تسمي أوضاع تامين النية:
- النية المشتركة (هو): يشير إلى انه سيتم طلب تامين مشترك (s) علي بعض العقد التابعة.
- النية الحصرية :(IX) تشير إلى انه سيتم طلب تامين (وحدات) خاصه علي بعض العقدات التابعة.
- النية المشتركة-الحصرية (سته): تشير إلى ان العقدة الحالية مؤمنه في الوضع المشترك ولكن سيتم طلب تامين (s) خاص علي بعض العقد التابعة.

Granularity of data items and Multiple Granularity Locking

- These locks are applied using the following compatibility matrix:

Intention-shared (IS
Intention-exclusive (IX)
Shared-intention-exclusive (SIX)

	IS	IX	S	SIX	X
IS	yes	yes	yes	yes	no
IX	yes	yes	no	no	no
S	yes	no	yes	no	no
SIX	yes	no	no	no	no
X	no	no	no	no	no

Granularity of data items and Multiple Granularity Locking

- The set of rules which must be followed for producing serializable schedule are

1. **The lock compatibility must adhered to.**
2. **The root of the tree must be locked first, in any mode..**
3. **A node N can be locked by a transaction T in S or IX mode only if the parent node is already locked by T in either IS or IX mode.**
4. **A node N can be locked by T in X, IX, or SIX mode only if the parent of N is already locked by T in either IX or SIX mode.**
5. **T can lock a node only if it has not unlocked any node (to enforce 2PL policy).**
6. **T can unlock a node, N, only if none of the children of N are currently locked by T.**

- التفاصيل الخاصة بعناصر البيانات وتامين العديد من الخصائص الحبيبية
- مجموعه القواعد التي يجب اتباعها لإنتاج جدول قابل للتسلسل هي
- يجب التزام بتوافق التامين.
- يجب تامين جذر الشجرة أولا ، في اي وضع..
- يمكن تامين العقدة N بواسطة معامله t في الوضع S أو ix فقط إذا كانت العقدة الأصل مؤمنه مسبقا بواسطة t في وضع اما أو التاسع.
- يمكن تامين n عقده بواسطة t في X أو التاسع أو وضع الستة فقط إذا كان الأصل من n مؤمنا بواسطة t في الوضع التاسع أو الستة بالفعل.
- لا يمكن ل T تامين عقده الا إذا لم تقم بإلغاء تامين اي عقده (لفرض النهج الثاني).
- يمكن إلغاء تامين عقده ، n، فقط إذا لم يتم تامين اي من الأطفال من N حاليا بواسطة t.

- Granularity of data items and Multiple Granularity Locking: An example of a serializable execution (continued):

T1	T2	T3
	unlock(p12)	
	unlock(f1)	
	unlock(db)	
unlock(r111)		
unlock(p11)		
unlock(f1)		
unlock(db)		
		unlock (r111j)
		unlock (p11)
		unlock (f1)
		unlock(f2)
		unlock(db)

Granularity of data items and Multiple Granularity Locking: An example of a serializable execution:

T1	T2	T3
IX(db)		
IX(f1)		
	IX(db)	
		IS(db)
		IS(f1)
		IS(p11)
IX(p11)		
X(r111)		
	IX(f1)	
	X(p12)	
		S(r11j)
IX(f2)		
IX(p21)		
IX(r211)		
Unlock (r211)		
Unlock (p21)		
Unlock (f2)		
		S(f2)