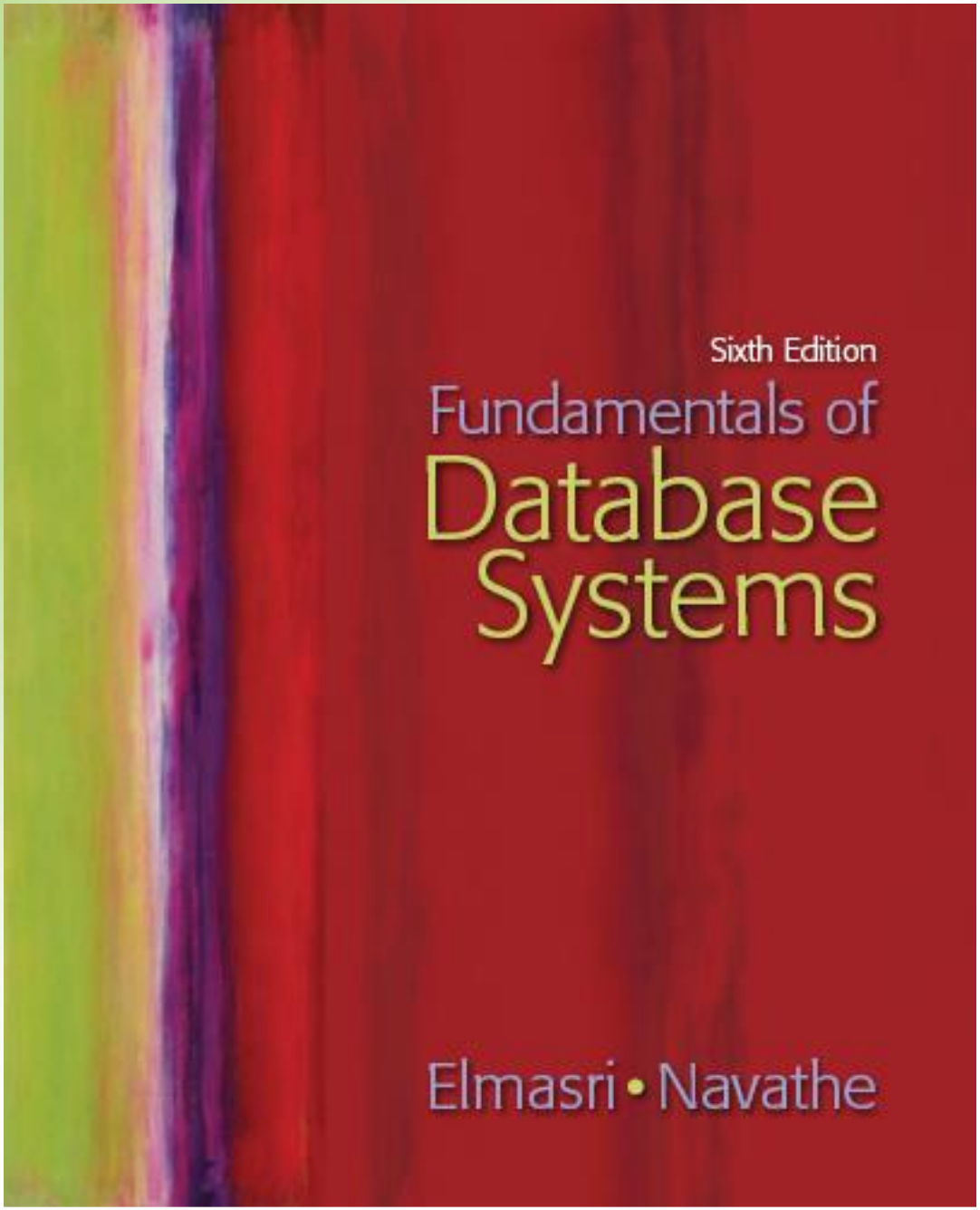
Chapter 22

Concurrency Control Techniques



Addison-Wesley
is an imprint of
PEARSON

Database Concurrency Control

- 1 Purpose of Concurrency Control
- الغرض من التحكم في التزامن
  - To enforce Isolation (through mutual exclusion) among conflicting transactions. لفرض العزلة (من خلال الاستبعاد المتبادل) بين المعاملات المتضاربة
  - To preserve database consistency through consistency preserving execution of transactions.
  - للحفاظ على اتساق قاعدة البيانات من خلال التناسق مع الحفاظ على تنفيذ المعاملات
  - To resolve read-write and write-write conflicts.
  - لحل تعارضات القراءة والكتابة و الكتابة
- 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.

Database Concurrency Control

Two-Phase Locking Techniques تقنيات القفل على مرحلتين

Locking is an operation which secures القفل هو العملية التي تؤمن

- (a) permission to Read (أ) إذن القراءة
- (b) permission to Write a data item for a transaction.
- (ب) إذن لكتابة بند بيانات لمعاملة

- 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. فتح هو عملية تزيل هذه الأذونات من عنصر البيانات
- Example:
  - Unlock (X): Data item X is made available to all other transactions.
- Lock and Unlock are Atomic operations. القفل وفتح القفل هما عمليتان ذريتان

Database Concurrency Control

Two-Phase Locking Techniques: Essential components

تقنيات القفل على مرحلتين: المكونات الأساسية

- Two locks modes: وضعان للقفل
  - (a) shared (read) (أ) مشتركة (قراءة) (b) exclusive (write)..(ب) حصريا (كتابة)
- Shared mode: shared lock (X) الوضع المشترك: القفل المشترك
  - 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. يمكن لأكثر من معاملة تطبيق قفل المشاركة على لقراءة قيمتها ولكن لا يمكن تطبيق قفل الكتابة على بأي معاملة أخرى.
- Exclusive mode: Write lock (X) وضع الحصري: كتابة القفل
  - Only one write lock on X can exist at any time and no shared lock can be applied by any other transaction on X. يمكن أن يوجد قفل كتابة واحد فقط على في أي وقت ولا يمكن تطبيق أيقفل مشترك بواسطة أي معاملة أخرى على
- Conflict matrix مصفوفة الصراع

	Read	Write
Read	Y	N
Write	N	N

Database Concurrency Control

Two-Phase Locking Techniques: Essential components تقنيات القفل على مرحلتين: أساسية عناصر

- Lock Manager: مدير القفل
  - Managing locks on data items. إدارة الأقفال على عناصر البيانات
- Lock table: طاولة القفل
  - 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. يستخدمه مدير القفل لتخزين تعريف معاملة تأمين عنصر البيانات ،وعنصر البيانات ، ووضع القفل ، والمؤشر إلى عنصر البيانات التالي الذي تم قفله. إحدى الطرق البسيطة لتطبيق جدول القفل هي من خلال القائمة المرتبطة

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

Database Concurrency Control

Two-Phase Locking Techniques: Essential components تقنيات القفل على مرحلتين: أساسية عناصر

- Database requires that all transactions should be well-formed. A transaction is well-formed if:
- تتطلب قاعدة البيانات أن تكون جميع المعاملات جيدة التنسيق .تكون المعاملة جيدة إذا
- It must lock the data item before it reads or writes to it. يجب أن يقفل عنصر البيانات قبل قراءته أو الكتابة إليه
  - It must not lock an already locked data items and it must not try to unlock a free data item.
  - يجب ألا يقفل عناصر بيانات مقفلة بالفعل ولا يجب أن يحاول فتح عنصر بيانات مجاني

Database Concurrency Control

Two-Phase Locking Techniques: Essential components

- The following code performs the lock operation:
- يقوم الكود التالي بتنفيذ عملية القفل

```
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;
```

Database Concurrency Control

Two-Phase Locking Techniques: Essential components

- The following code performs the unlock operation:
- قوم الكود التالي بتنفيذ عملية إلغاء القفل

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

Database Concurrency Control

Two-Phase Locking Techniques: Essential components

- The following code performs the read operation:
- الكود التالي ينفذ عملية القراءة

```
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;
```

Database Concurrency Control

Two-Phase Locking Techniques: Essential components

- The following code performs the write lock operation:
- يقوم الكود التالي بتنفيذ عملية قفل الكتابة

```
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;
```

Database Concurrency Control

Two-Phase Locking Techniques: Essential components

تقنياتالقفل على مرحلتين: المكونات الأساسية

- The following code performs the unlock operation:
- يقوم الكود التالي بتنفيذ عملية إلغاء القفل

```
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;
```

Database Concurrency Control

Two-Phase Locking Techniques: Essential components

تقنيات القفل على مرحلتين: المكونات الأساسية

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

    if Ti has a read-lock (X) and Tj has no read-lock (X) ($i \neq j$) 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
  - قفل الرجوع إلى إصدار أقدم: قفل الكتابة الحالي لقراءة القفل

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

Database Concurrency Control

Two-Phase Locking Techniques: The algorithm تقنيات القفل على مرحلتين: الخوارزمية

- Two Phases:
  - (a) Locking (Growing)
  - (b) Unlocking (Shrinking).
  - (أ) قفل (تزايد)
  - (ب) فتح (انكماش)
- **Locking (Growing) Phase:** مرحلة القفل (النمو
- A transaction applies locks (read or write) on desired data items one at a time. تطبق المعاملة الأقفال (القراءة أو الكتابة) على عناصر البيانات المطلوبة واحداً تلو الآخر
- **Unlocking (Shrinking) Phase:** مرحلة الفتح (الانكماش
  - A transaction unlocks its locked data items one at a time. تفتح المعاملة عناصر البيانات المقفلة الخاصة بها واحدة تلو الأخرى
- **Requirement:** المتطلبات
  - 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.
  - بالنسبة للمعاملة ، يجب أن تكون هاتان المرحلتان حصرياً ، أي أثناء مرحلة القفل ، يجب ألا تبدأ مرحلة فتح القفل ويجب ألا تبدأ مرحلة قفل طور الفتح

Database Concurrency Control

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);	

Database Concurrency Control

Two-Phase Locking Techniques: The algorithm

T1	T2
read_lock (Y);	
read_item (Y);	
unlock (Y);	
	read_lock (X);
Time ↓	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);	

Result

X=50; Y=50
Nonserializable because it. violated two-phase policy.

Database Concurrency Control

Two-Phase Locking Techniques: The algorithm

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

Database Concurrency Control

Two-Phase Locking Techniques: The algorithm تقنيات القفل على مرحلتين: الخوارزمية

- Two-phase policy generates two locking algorithms سياسة مرحلتين تولد اثنين من خوارزميات القفل
  - (a) **Basic**
  - (b) **Conservative**
- **Conservative**: محافظ
  - Prevents deadlock by locking all desired data items before transaction begins execution. يمنع الجمود عن طريق قفل جميع عناصر البيانات المطلوبة قبل بدء تنفيذ المعاملة
- **Basic**: الأساسي
  - Transaction locks data items incrementally. This may cause deadlock which is dealt with.
  - تقوم المعاملة بتأمين عناصر البيانات بشكل متزايد. قد يتسبب هذا في طريق مسدود يتم التعامل معه
- **Strict**: صارم
  - 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.
  - إصدار أكثر صرامة من الخوارزمية الأساسية حيث يتم إجراء إلغاء القفل بعد إنهاء المعاملة(الإرتكاب أو الإلغاء والتراجع). هذه هي خوارزمية القفل ثنائية الطور الأكثر استخداما

Database Concurrency Control

Dealing with Deadlock and Starvation التعامل مع الجمود والمجاعة

- **Deadlock**

T'1	T'2
read_lock (Y);	
read_item (Y);	
	read_lock (X);
	read_item (Y);
write_lock (X);	
(waits for X)	
	write_lock (Y);
	(waits for Y)

T1 and T2 did follow two-phase policy but they are deadlock

- Deadlock (T'1 and T'2)

Database Concurrency Control

Dealing with Deadlock and Starvation التعامل مع الجمود والمجاعة

- **Deadlock prevention** الوقاية من الطريق المسدود
  - A transaction locks all data items it refers to before it begins execution.
  - تقوم المعاملة بتأمين جميع عناصر البيانات التي تشير إليها قبل أنتبدأ في التنفيذ
  - This way of locking prevents deadlock since a transaction never waits for a data item.
  - طريقة القفل هذه تمنع الجمود لأن المعاملة لا تنتظر أبداً عنصر البيانات
  - The conservative two-phase locking uses this approach. يستخدم القفل المحافظ على مرحلتين هذا النهج

Database Concurrency Control

Dealing with Deadlock and Starvation التعامل مع الجمود والمجاعة

Deadlock detection and resolution كشف وقرار الجمود

- 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.
- فيهذا النهج ، يسُمح بحدوث الجمود. يحافظ المجدول على انتظار رسم بياني للكشف عن الدورة. في حالة وجود دورة ، يتم تحديد معاملة واحدة متضمنة في الدورة (الضحية) والتراجع
- 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
- يتم إنشاء انتظار الرسم البياني باستخدام جدول القفل. بمجرد حظر المعاملة ،يتم إضافتها إلى الرسم البياني. عندما تنتظر ينتظر وينتظر أو ، فإن هذا يخلق دورة. واحدة من المعاملات سلسلة مثل: ينتظر

Database Concurrency Control

Dealing with Deadlock and Starvation التعامل مع الجمود والمجاعة

- **Deadlock avoidance** تجنب الجمود
  - There are many variations of two-phase locking algorithm.
  - هناك العديد من الاختلافات في خوارزمية القفل على مرحلتين
  - Some avoid deadlock by not letting the cycle to complete.
  - يتجنب البعض الجمود من خلال عدم ترك الدورة تكتمل
  - That is as soon as the algorithm discovers that blocking a transaction is likely to create a cycle, it rolls back the transaction.
  - هذا بمجرد أن تكتشف الخوارزمية أن حظر المعاملة من المحتمل أن يؤدي إلى إنشاء دورة ، فإنها تتراجع عن المعاملة
  - Wound-Wait and Wait-Die algorithms use timestamps to avoid deadlocks by rolling-back victim.
  - تستخدم خوارزميات و طوابع زمنية لتجنب المآزق عن طريق عودة الضحية

Database Concurrency Control

Dealing with Deadlock and Starvation

- **Starvation** مجاعة
  - Starvation occurs when a particular transaction consistently waits or restarted and never gets a chance to proceed further. يحدث الجوع عندما تنتظر معاملة معينة باستمرار أو تعُاد تشغيلها ولا تتاح لها فرصة للمضي قدماً.
  - In a deadlock resolution it is possible that the same transaction may consistently be selected as victim and rolled-back. فيحل الجمود ، من الممكن أن يتم اختيار نفس المعاملة باستمرار كضحية والتراجع عنها
  - This limitation is inherent in all priority based scheduling mechanisms. هذا القيد متأصل في جميع آليات الجدولة القائمة على الأولوية
  - In Wound-Wait scheme a younger transaction may always be wounded (aborted) by a long running older transaction which may create starvation. في مخطط ، قد تتضرر معاملة أصغر (يتم إجهاضها) دائماً بسبب صفقة قديمة طويلة الأمد قد تؤدي إلى المجاعة. حقوق النشر رامز المصري

Database Concurrency Control

Timestamp based concurrency control algorithm خوارزمية التحكم في التزامن القائمة على الطابع الزمني

- **Timestamp** لطابع الزمني
  - 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. متغير متزايد بشكل رتيب (عدد صحيح) يشير إلى عمر عملية أو معاملة. تشير قيمة الطابع الزمني الأكبر إلى حدث أو عملية أحدث
  - Timestamp based algorithm uses timestamp to serialize the execution of concurrent transactions.
  - تستخدم الخوارزمية القائمة على الطابع الزمني الطابع الزمني لتسلسل تنفيذ المعاملات المتزامنة

Database Concurrency Control

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).

Database Concurrency Control

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).

Database Concurrency Control

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).

Database Concurrency Control

Multiversion concurrency control techniques

تقنيات التحكم في التزامن المتعدد

- 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.
- يحافظ هذا الأسلوب على عدد من إصدارات عنصر البيانات ويخصص الإصدار الصحيح لعملية قراءة المعاملة. وبالتالي ، على عكس الآليات الأخرى ، لا يتم رفض عملية القراءة في هذه الآلية أبدا
- Side effect: عراض جانبية
  - 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.
  - *مطلوب مزيد من التخزين (ذاكرة الوصول العشوائي والقرص) بشكل كبير للحفاظ على إصدارات متعددة. للتحقق من النمو غير المحدود للإصدارات ، يتم تشغيل مجموعة البيانات المهملة عند استيفاء بعض المعايير*

Database Concurrency Control

Multiversion technique based on timestamp ordering

على أساس الطابع الزمني يأمر تقنية

- This approach maintains a number of versions of a data item and allocates the right version to a read operation of a transaction.
- يحافظ هذا الأسلوب على عدد من إصدارات عنصر البيانات ويخصص الإصدار الصحيح لعملية قراءة المعاملة
  - Thus unlike other mechanisms a read operation in this mechanism is never rejected. وبالتالي ،على عكس الآليات الأخرى ، لا يتم رفض عملية القراءة في هذه الآلية أبدا
- 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.
- الآثار الجانبية: مطلوب مزيد من التخزين (ذاكرة الوصول العشوائي والقرص) للحفاظ على إصدارات متعددة. للتحقق من النمو غير المحدود للإصدارات ، يتم تشغيل مجموعة البيانات المهملة عند استيفاء بعض المعايير

Database Concurrency Control

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.

Database Concurrency Control

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).

Database Concurrency Control

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.

Database Concurrency Control

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.

Database Concurrency Control

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

(a)

	Read	Write
Read	Yes	No
Write	No	No

read/write locking scheme

(b)

	Read	Write	Certify
Read	Yes	Yes	No
Write	Yes	No	No
Certify	No	No	No

read/write/certify locking scheme

Database Concurrency Control

Multiversion Two-Phase Locking Using Certify تأمين متعدد الطورين باستخدام الشهادة

Locks قفل

- Note:ملحوظة
  - In multiversion 2PL read and write operations from conflicting transactions can be processed concurrently. في التحويل المتعدد ، يمكن معالجة عمليات القراءة والكتابة من المعاملات المتضاربة فينفس الوقت
  - 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. يعمل هذا على تحسين التزامن ولكنه قد يؤخر الالتزام بالمعاملة بسبب الحصول على أقفال تصديق على جميع عمليات الكتابة. إنه يتجنب الإحباط المتتالي ولكن مثل نظام الإغلاق الصارم على مرحلتين ، قد تصل المعاملات المتضاربة إلىطريق مسدود

Database Concurrency Control

Validation (Optimistic) Concurrency Control Schemes

التحقق من صحة (متفائل) مخططات التحكم في التزامن

- In this technique only at the time of commit serializability is checked and transactions are aborted in case of non-serializable schedules.
- فيهذه التقنية فقط في وقت الالتزام بالتسلسل يتم التحقق من قابلية التسلسل وإلغاء المعاملات في حالة الجداول غير القابلة للتسلسل
- Three phases:
  1. **Read phase**
  2. **Validation phase**
  3. **Write phase**
  4. ثلاث مراحل: ١.مرحلة القراءة ٢.مرحلة التحقق ٣.مرحلة الكتابة

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).
- ١.مرحلة القراءة: يمكن للمعاملة قراءة قيم عناصر البيانات الملتزمة. ومع ذلك ، يتم تطبيق التحديثات فقط على النسخ المحلية (الإصدارات) من عناصر البيانات(في ذاكرة التخزين المؤقت لقاعدة البيانات)

Database Concurrency Control

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.

Database Concurrency Control

Validation (Optimistic) Concurrency Control

التحقق من الصحة (متفائل) التحكم في التزامن

Schemes المخططات

3. **Write phase**: On a successful validation transactions' updates are applied to the database; otherwise, transactions are restarted.

- مرحلة الكتابة: في عملية التحقق الناجحة ، يتم تطبيق تحديثات المعاملات على قاعدة البيانات ؛ خلاف ذلك ، يتم إعادة تشغيل المعاملات.

Database Concurrency Control

Granularity of data items and Multiple Granularity Locking

مدى دقة عناصر البيانات وتأمين الدقة المتعددة

- 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).
- تحدد وحدة البيانات القابلة للقفل مدى تفصيلها. يمكن أن تكون الدقة تقريبية (قاعدة بيانات كاملة) أو يمكن أن تكون جيدة (مجموعة أوسمة لعلاقة)
- Data item granularity significantly affects concurrency control performance. Thus, the degree of concurrency is low for coarse granularity and high for fine granularity.
- تؤثر دقة عنصر البيانات بشكل كبير على أداء التحكم في التزامن. وبالتالي ،تكون درجة التزامن منخفضة بالنسبة إلى الدقة الخشنة وعالية الدقة
- 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
  6. مثال على دقة عنصر البيانات: ١.حقل من سجل قاعدة البيانات (سمة من سمات المجموعة) ٢.سجل قاعدة بيانات (مجموعة أو علاقة) ٣.كتلة قرصية ٤.ملف كامل ٥.قاعدة البيانات بأكملها

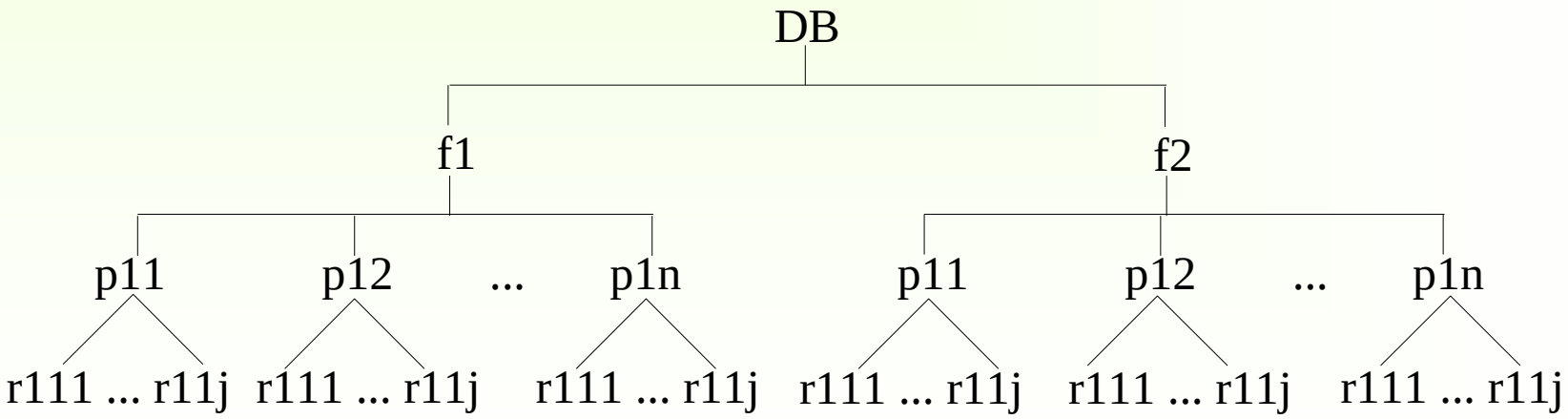
Database Concurrency Control

Granularity of data items and Multiple Granularity

مدى دقة عناصر البيانات وتعددها

Locking قفل

- The following diagram illustrates a hierarchy of granularity from coarse (database) to fine (record).
- يوضح الرسم البياني التالي تسلسلاً هرمياً للدقة من خشن(قاعدة بيانات) إلى دقيق (سجل)



Database Concurrency Control

Granularity of data items and Multiple Granularity Locking

مدى دقة عناصر البيانات وتأمين الدقة المتعددة

- To manage such hierarchy, in addition to read and write, three additional locking modes, called intention lock modes are defined: لإدارة هذا التسلسل الهرمي ، بالإضافة إلى القراءة والكتابة ، يتم تحديد ثلاثة أوضاع قفل إضافية تسمى أوضاع قفل النية
  - **Intention-shared (IS)** النية المشتركة : indicates that a shared lock(s) will be requested on some descendent nodes(s).
  - يشير إلى أنه سيتم طلب قفل (أقفال) مشترك على بعضالعقد (العقد) التابعة
  - **Intention-exclusive :(** النية الحصرية (التاسع)
  - **(IX)**: indicates that an exclusive lock(s) will be requested on some descendent node(s). يشير إلى أنه سيتم طلب قفل (أقفال) خاص علىبعض العقد (العقد) الهابطة
  - **Shared-intention-exclusive (SIX)** النية المشتركة الحصرية
  - : indicates that the current node is locked in shared mode but an exclusive lock(s) will be requested on some descendent nodes(s).

يشير إلى أن العقدة الحالية مؤمنة في الوضع المشترك ولكن سيتم طلب قفل (أقفال) خاص على بعض العقد(العقد) الهابطة

Database Concurrency Control

Granularity of data items and Multiple Granularity

يتم تطبيق هذه الأقفال باستخدام مصفوفة التوافق

Locking قفل

- These locks are applied using the following compatibility matrix: يتم تطبيق هذه الأقفال باستخدام مصفوفة التوافق

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

Database Concurrency Control

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.

Database Concurrency Control

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)

Database Concurrency Control

- Granularity of data items and Multiple Granularity Locking: An example of a serializable execution (continued):
- *مدى دقة عناصر البيانات والتأمين متعدد الجوانب: مثال على تنفيذ قابل للتسلسل*

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)

Summary

- Databases Concurrency Control لتحكم في التزامن في قواعد البيانات

1. Purpose of Concurrency Control الغرض من التحكم في التزامن
2. Two-Phase locking قفل مرحلتين
3. Limitations of CCMs قيود
4. Index Locking قفل الفهرس
5. Lock Compatibility Matrix مصفوفة توافق القفل
6. Lock Granularity قفل حبيبية