==========

CONCURRENCY CONTROL

The coordination of the simultaneous execution of transactions in a multi-user database system is **concurrency control**. The **objective** of concurrency control is to ensure the serializability of transaction multi-user database environment. Concurrency control is **important** because the simultaneous execution of transactions over a shared database can create several data integrity and consistency problems. The three main problems are:

التحكم التزامني

يعتبر تنسيق التنفيذ المتزامن للمعاملات في نظام قاعدة بيانات متعدد المستخدمين هو التحكم في التزامن. الهدف من التحكم في التزامن هو ضمان إمكانية إجراء تسلسل لبيئة قاعدة بيانات المعاملات متعددة المستخدمين. يعد التحكم في التزامن أمرًا مهمًا لأن التنفيذ المتزامن للمعاملات عبر قاعدة بيانات مشتركة يمكن أن يؤدي إلى العديد من مشاكل تكامل البيانات والاتساق. المشاكل الرئيسية الثلاث هي:

1. Lost Updates

The lost update problem occurs when two concurrent transactions, T1 and T2, are updating the same data element and one of the updates is lost (overwritten by the other transaction).

1. التحديثات المفقودة

تحدث مشكلة التحديث المفقود عندما تقوم معاملتان متزامنتان ، T1 و T2 ، بتحديث نفس عنصر البيانات وفقد أحد التحديثات (تمت الكتابة فوقه بواسطة المعاملة الأخرى).

The figure bellow explain the serial execution of two transactions

يوضح الشكل أدناه التنفيذ التسلسلي لعمليتين

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	PROD_QOH = 35 + 100	
3	T1	Write PROD_QOH	135
4	T2	Read PROD_QOH	135
5	T2	PROD_QOH = 135 − 30	
6	T2	Write PROD_QOH	105

The following figure show lost updates يوضح الشكل التالي التحديثات المفقودة

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T2	Read PROD_QOH	35
3	T1	PROD_QOH = 35 + 100	
4	T2	PROD_QOH = 35 − 30	
5	T1	Write PROD_QOH (**Lost update**)	135
6	T2	Write PROD_QOH	5

2. Uncommitted data

The phenomenon of uncommitted data occurs when two transactions, T1 and T2, are executed concurrently and the first transaction (T1) is rolled back after the second transaction (T2) has already accessed the uncommitted data—thus violating the isolation property of transactions.

The figure bellow explain the correct execution of two transactions

2. البيانات غير الملتزم بها

تحدث ظاهرة البيانات غير الملتزم بها عندما يتم تنفيذ معاملتين ، T1 و T2 ، بشكل متزامن ويتم إرجاع المعاملة الأولى (T1) بعد أن وصلت المعاملة الثانية (T2) بالفعل إلى البيانات غير الملتزم بها - وبالتالي تنتهك خاصية عزل المعاملات.

يوضح الشكل أدناه التنفيذ الصحيح لعمليتين

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	PROD_QOH = 35 + 100	
3	T1	Write PROD_QOH	135
4	T1	*****ROLLBACK *****	35
5	T2	Read PROD_QOH	35
6	T2	PROD_QOH = 35 – 30	
7	T2	Write PROD_QOH	5

The following figure show an uncommitted data problem يوضح الشكل التالي مشكلة بيانات غير ملتزم بها

TIME	TRANSACTION	STEP	STORED VALUE
1	T1	Read PROD_QOH	35
2	T1	PROD_QOH = 35 + 100	
3	T1	Write PROD_QOH	135
4	T2	Read PROD_QOH (**Read uncommitted data**)	135
5	T2	PROD_QOH = 135 – 30	
6	T1	***** **ROLLBACK** *****	35
7	T2	Write PROD_QOH	105

3. Inconsistent retrievals

Inconsistent retrievals occur when a transaction accesses data before and after another transaction(s) finish working with such data. For example, an inconsistent retrieval would occur if transaction T1 calculates some summary (aggregate) function over a set of data while another transaction (T2) is updating the same data, The problem is that the transaction might read some data before they are changed and other data after they are changed, thereby yielding inconsistent results.

3. عمليات الاسترجاع غير المتسقة

تحدث عمليات الاسترداد غير المتسقة عندما تصل معاملة ما إلى البيانات قبل وبعد انتهاء معاملة (معاملات) أخرى من العمل بهذه البيانات. على سبيل المثال ، قد يحدث استرداد غير متسق إذا قامت المعاملة T1 بحساب بعض الوظائف الموجزة (التجميعية) على مجموعة من البيانات بينما تقوم معاملة أخرى (T2) بتحديث نفس البيانات ، والمشكلة هي أن المعاملة قد تقرأ بعض البيانات قبل تغييرها و البيانات الأخرى بعد تغييرها ، مما يؤدي إلى نتائج غير متسقة.

Retrieval During Update

TRANSACTION T1	TRANSACTION T2
SELECT SUM(PROD_QOH) FROM PRODUCT	
	UPDATE PRODUCT SET PROD_QOH = PROD_QOH + 10 WHERE PROD_CODE = '1546-QQ2'
	UPDATE PRODUCT SET PROD_QOH = PROD_QOH – 10 WHERE PROD_CODE = '1558-QW1'
	COMMIT;

==========

Transaction Results: Data Entry Correction

	BEFORE	AFTER
PROD_CODE	PROD_QOH	PROD_QOH
11QER/31	8	8
13-Q2/P2	32	32
1546-QQ2	15	(15 + 10) → 25
1558-QW1	23	(23 − 10) → 13
2232-QTY	8	8
2232-QWE	6	6
Total	92	92

Inconsistent Retrieval

TIME	TRANSACTION	ACTION	VALUE	TOTAL
1	T1	Read PROD_QOH for PROD_CODE = '11QER/31'	8	8
2	T1	Read PROD_QOH for PROD_CODE = '13-Q2/P2'	32	40
3	T2	Read PROD_QOH for PROD_CODE = '1546-QQ2'	15	
4	T2	PROD_QOH = 15 + 10		
5	T2	Write PROD_QOH for PROD_CODE = '1546-QQ2'	25	
6	T1	Read PROD_QOH for PROD_CODE = '1546-QQ2'	25	(After) 65
7	T1	Read PROD_QOH for PROD_CODE = '1558-QW1'	23	(Before) 88
8	T2	Read PROD_QOH for PROD_CODE = '1558-QW1'	23	
9	T2	PROD_QOH = 23 − 10		
10	T2	Write PROD_QOH for PROD_CODE = '1558-QW1'	13	
11	T2	***** COMMIT *****		
12	T1	Read PROD_QOH for PROD_CODE = '2232-QTY'	8	96
13	T1	Read PROD_QOH for PROD_CODE = '2232-QWE'	6	102

THE SCHEDUULER

The scheduler is a special DBMS process that establishes the order in which the operations within concurrent transactions are executed. The scheduler interleaves the execution of database operations to ensure serializability isolation of transactions. To determine the appropriate order, the scheduler bases its actions on concurrency control algorithms, such as **locking** or **time stamping** methods, However. Important to understand that not all transactions are serializable. The DBMS determines what transactions serializable and proceeds to interleave the execution of the transaction's operations. Generally, transactions that are not serializable are executed on a first-come, first-served basis by the DBMS.

المجدول

المجدول هو عملية DBMS خاصة تحدد الترتيب الذي يتم به تنفيذ العمليات ضمن المعاملات المتزامنة. يقوم المجدول بتشذير تنفيذ عمليات قاعدة البيانات لضمان قابلية التسلسل لعزل المعاملات. لتحديد الترتيب المناسب ، يبني المجدول إجراءاته على خوارزميات التحكم في التزامن ، مثل طرق القفل أو ختم الوقت ، ومع ذلك. من المهم أن نفهم أنه ليست كل المعاملات قابلة للتسلسل. يحدد نظام إدارة قواعد البيانات (DBMS) المعاملات القابلة للتسلسل ويستمر في تشذير تنفيذ عمليات المعاملة. بشكل عام ، يتم تنفيذ المعاملات غير القابلة للتسلسل على أساس أسبقية الحضور من خلال نظام إدارة قواعد البيانات.

The scheduler's **main job** is to create serializable schedule of a transaction's operations. **A serializable schedule** is a schedule of a transaction's operations in which the interleaved execution of the transactions (T1, T2, T3, etc.) yields the same results as if the transactions were executed in serial order (one after another).

تتمثل المهمة الرئيسية للمجدول في إنشاء جدول قابل للتسلسل لعمليات المعاملة. الجدول القابل للتسلسل هو جدول لعمليات إحدى المعاملات حيث يؤدي التنفيذ المتداخل للمعاملات (T1، T2، T3 ، إلخ) إلى نفس النتائج كما لو تم تنفيذ المعاملات بترتيب تسلسلي (واحدًا تلو الآخر).

==========

The scheduler also makes sure that the computer's central processing unit (CPU) and storage systems are efficiently used. If there were no way to schedule the execution of transactions, all transactions would be executed on a first-come, first-served basis. The problem with that approach is that processing time is wasted when the CPU waits for a READ or WRITE operation to finish, thereby losing several CPU cycles. In short , first-come. first-served. scheduling tends to yield unacceptable response times within the multi-user DBMS environment. Therefore, some other scheduling method is needed to improve the efficiency of the overall system.

يتأكد المجدول أيضًا من استخدام وحدة المعالجة المركزية للكمبيوتر (CPU) وأنظمة التخزين بكفاءة. إذا لم تكن هناك طريقة لجدولة تنفيذ المعاملات ، فسيتم تنفيذ جميع المعاملات على أساس أسبقية الحضور. تكمن المشكلة في هذا النهج في إهدار وقت المعالجة عندما تنتظر وحدة المعالجة المركزية انتهاء عملية القراءة أو الكتابة ، وبالتالي تفقد العديد من دورات وحدة المعالجة المركزية. باختصار ، من يأتي أولاً. يخدم أولا. تميل الجدولة إلى إنتاج أوقات استجابة غير مقبولة في بيئة نظم إدارة قواعد البيانات متعددة المستخدمين. لذلك ، هناك حاجة إلى بعض طرق الجدولة الأخرى لتحسين كفاءة النظام العام.

Additionally, the scheduler facilitates data isolation to ensure that two transactions do not update the same data element at the same time. Database operations might require READ and/or WRITE actions that produce conflicts. For example, The Table bellow shows the possible conflict scenarios when two transactions, T1 and T2, are executed concurrently over the same data. Note that in the Table two operations are in conflict when they access the same data and at least one of them is a WRITE operation.

بالإضافة إلى ذلك ، يسهل المجدول عزل البيانات لضمان عدم قيام معاملتين بتحديث نفس عنصر البيانات في نفس الوقت. قد تتطلب عمليات قاعدة البيانات إجراءات القراءة و / أو الكتابة التي تؤدي إلى حدوث تعارضات. على سبيل المثال ، يعرض الجدول أدناه سيناريوهات التعارض المحتملة عند تنفيذ معاملتين ، T1 و T2 ، بشكل متزامن على نفس البيانات. لاحظ أنه في الجدول تتعارض عمليتان عند وصولهما إلى نفس البيانات وأن إحداهما على الأقل عملية الكتابة.

Read/write Conflict Scenario: Conflicting Database Operations Matrix

	TRANSACTIONS		
	T1	T2	RESULT
Operations	Read	Read	No conflict
	Read	Write	Conflict
	Write	Read	Conflict
	Write	Write	Conflict

Several methods have been proposed to schedule the execution of conflicting operations in concurrent transactions. Those methods have been classified as **locking**, **time stamping**, and **optimistic**.

تم اقتراح عدة طرق لجدولة تنفيذ العمليات المتضاربة في المعاملات المتزامنة. تم تصنيف هذه الأساليب على أنها قفل وختم زمني ومتفائلة.

CONCURRENCY CONTROL WITH LOCKING METHODS

A **lock** guarantees exclusive use of a data item to a current transaction. In other words, transaction T2 does not have access to a data item that is currently being used by transaction T1. A transaction acquires a lock prior to data access; the lock is released (unlocked) when the transaction is completed so that another transaction can lock the data item for its exclusive use.

Most multiuser DBMSs automatically initiate and enforce locking procedures. All lock information is managed by a **lock manager**.

التحكم في التناقض باستخدام طرق القفل

يضمن القفل الاستخدام الحصري لعنصر بيانات في معاملة حالية. بمعنى آخر ، لا تتمتع المعاملة T2 بإمكانية الوصول إلى عنصر بيانات يتم استخدامه حاليًا بواسطة المعاملة T1. تحصل المعاملة على قفل قبل الوصول إلى البيانات ؛ يتم تحرير القفل (غير مؤمن) عند اكتمال المعاملة بحيث يمكن لمعاملة أخرى قفل عنصر البيانات لاستخدامه الحصري.

تبدأ معظم أنظمة إدارة قواعد البيانات متعددة المستخدمين إجراءات القفل وتنفذها تلقائيًا. تتم إدارة جميع معلومات القفل من قبل مدير القفل.

==========

Lock Granularity

Indicates the level of lock use. Locking can take place at the following levels: database, table, page, row or even field.

قفل حبيبية
يشير إلى مستوى استخدام القفل. يمكن أن يحدث القفل على المستويات التالية: قاعدة بيانات أو جدول أو صفحة أو صف أو حتى حقل.

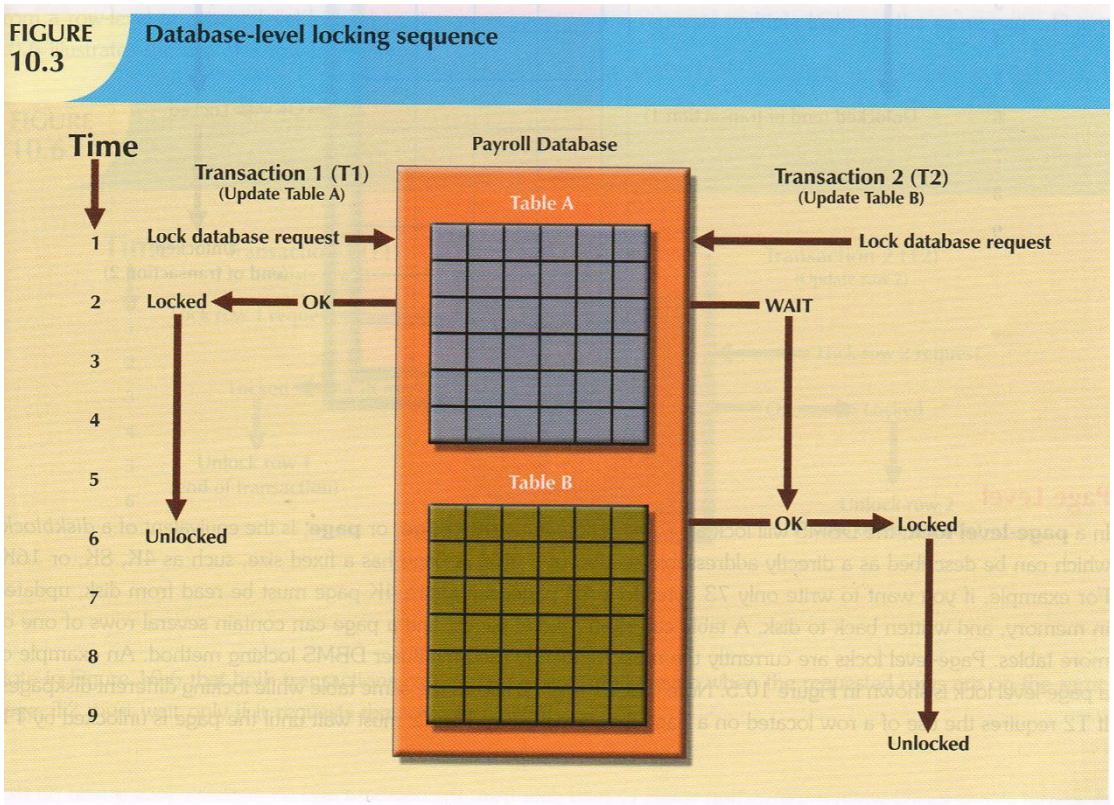
Database Level

In a database-level lock, the entire database is locked, thus preventing the use of any tables in the database by transaction T2 while transaction Tl is being executed. This level of locking is good for batch processes, but it is unsuitable for multiuser DBMSs. You can imagine how slow the data access would be if thousands of transactions had to wait for the previous transaction to be completed before the next one could reserve the entire database.

مستوى قاعدة البيانات
في القفل على مستوى قاعدة البيانات ، يتم تأمين قاعدة البيانات بأكملها ، وبالتالي منع استخدام أي جداول في قاعدة البيانات من خلال المعاملة T2 أثناء تنفيذ المعاملة Tl. يعد هذا المستوى من القفل جيدًا لعمليات الدُفعات ، ولكنه غير مناسب لنظم إدارة قواعد البيانات متعددة المستخدمين. يمكنك أن تتخيل مدى بطء الوصول إلى البيانات إذا اضطرت آلاف المعاملات إلى الانتظار حتى تكتمل المعاملة السابقة قبل أن تتمكن المعاملة التالية من حجز قاعدة البيانات بأكملها.

Figure 10.3 illustrates the database-level lock. Note that because of the database-level lock, transactions T1 and T2 cannot access the same database concurrently even when they use different tables.

يوضح الشكل 10.3 القفل على مستوى قاعدة البيانات. لاحظ أنه بسبب تأمين مستوى قاعدة البيانات ، لا يمكن للمعاملات T1 و T2 الوصول إلى نفس قاعدة البيانات بشكل متزامن حتى عند استخدام جداول مختلفة.



==========

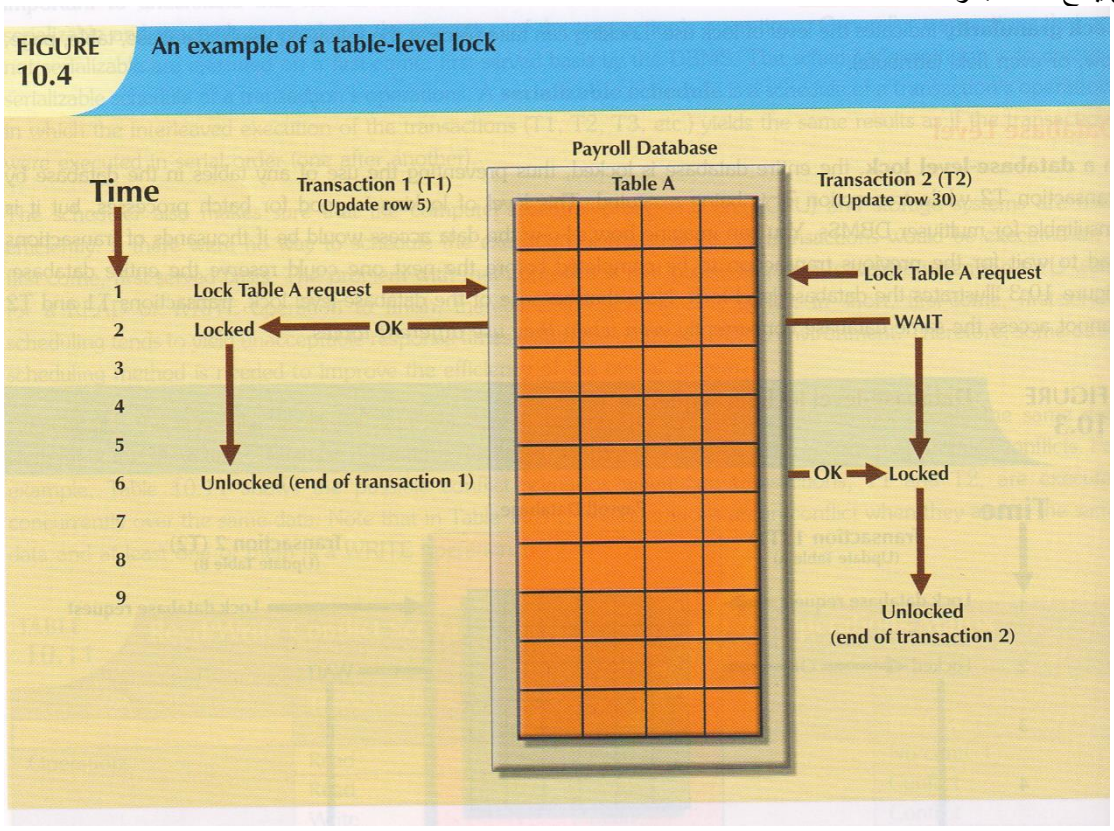
Table Level

In a table-level lock, the entire table is locked, preventing access to any row by transaction T2 while transaction T1 is using the table. If a transaction requires access to several tables, each table may be locked. However, two transactions can access the same database as long as they access different tables.
Table—level locks, while less restrictive than database-level locks, cause **traffic jams** when many transactions are waiting to access the same table.

مستوى الجدول
في حالة القفل على مستوى الجدول ، يتم تأمين الجدول بأكمله ، مما يمنع الوصول إلى أي صف عن طريق المعاملة T2 أثناء استخدام المعاملة T1 الجدول. إذا كانت المعاملة تتطلب الوصول إلى عدة جداول ، فقد يتم تأمين كل جدول. ومع ذلك ، يمكن لمعاملتين الوصول إلى نفس قاعدة البيانات طالما أنهما يصلان إلى جداول مختلفة.
تتسبب أقفال مستوى الجدول ، في حين أنها أقل تقييدًا من أقفال مستوى قاعدة البيانات ، في حدوث اختناقات مرورية عند انتظار العديد من المعاملات للوصول إلى نفس الجدول.

Consequently, table-level locks are not suitable for multiuser DBMSs. Figure 10.4 illustrates the effect of a table-level lock. Note that in Figure 10.4, transactions T1 and T2 cannot access the same table even when they try to use different rows; T2 must wait until T1 unlocks the table.

وبالتالي ، فإن الأقفال على مستوى الجدول ليست مناسبة لنظم إدارة قواعد البيانات متعددة المستخدمين. يوضح الشكل 10.4 تأثير القفل على مستوى الطاولة. لاحظ أنه في الشكل 10.4 ، لا يمكن للمعاملات T1 و T2 الوصول إلى نفس الجدول حتى عند محاولتهما استخدام صفوف مختلفة ؛ يجب أن ينتظر T2 حتى يفتح T1 الجدول.



==========

Page Level

In a page-level lock, the DBMS will lock an entire diskpage. A diskpage, or page, is the equivalent of a disk block, which can be described as a directly addressable section of a disk. A page has a fixed size, such as 4K, 8K, or 16K.

For example, if you want to write only 73 bytes to a 4K page, the entire 4K page must be read from disk, updated in memory, and written back to disk. A table can span several pages, and a page can contain several rows of one or more tables. Page—level locks are currently the most frequently used multiuser DBMS locking method. An example of a page—level lock is shown in Figure 10.5. Note that T1 and T2 access the same table while locking different diskpages.

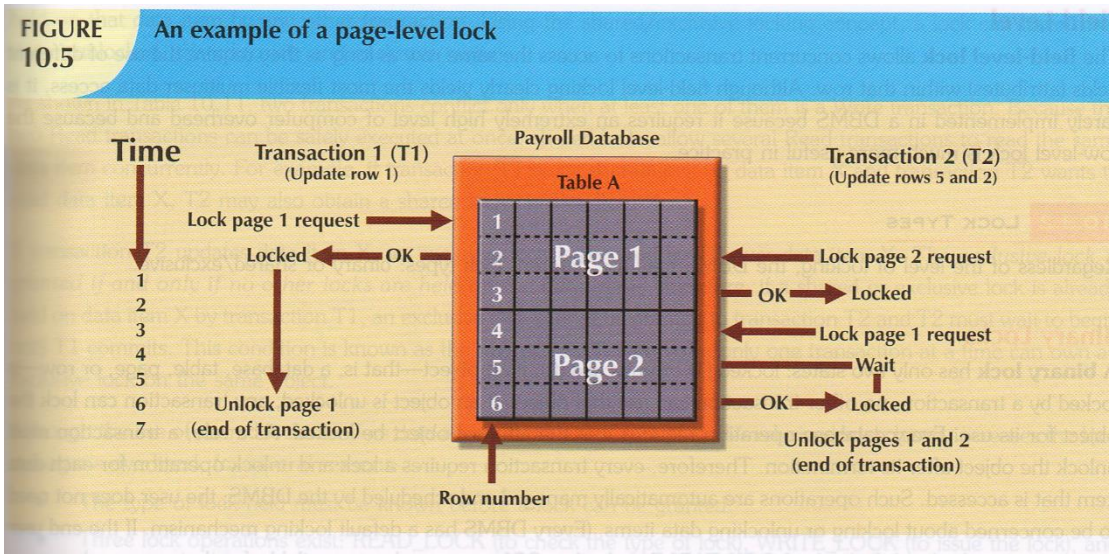
If T2 requires the use of a row located on a page that is locked by T1, T2 must wait until the page is unlocked by T1.

مستوى الصفحة

في قفل مستوى الصفحة ، سيقوم نظام إدارة قواعد البيانات (DBMS) بقفل صفحة قرص بأكملها. صفحة القرص ، أو الصفحة ، هي ما يعادل كتلة القرص ، والتي يمكن وصفها على أنها قسم قابل للعنونة مباشرة من القرص. الصفحة ذات حجم ثابت ، مثل K4 أو K8 أو K16.

على سبيل المثال ، إذا كنت تريد كتابة 73 بايت فقط على صفحة K4 ، فيجب قراءة صفحة K4 بالكامل من القرص وتحديثها في الذاكرة وإعادة كتابتها على القرص. يمكن أن يمتد الجدول على عدة صفحات ، ويمكن أن تحتوي الصفحة على عدة صفوف من جدول واحد أو أكثر. تعد أقفال مستوى الصفحة حاليًا أكثر طرق تأمين نظام إدارة قواعد البيانات متعددة المستخدمين استخدامًا. يظهر مثال لقفل مستوى الصفحة في الشكل 10.5. لاحظ أن T1 و T2 يصلان إلى نفس الجدول أثناء قفل صفحات القرص المختلفة.

إذا تطلب T2 استخدام صف موجود على صفحة مؤمنة بواسطة T1 ، يجب أن ينتظر T2 حتى يتم إلغاء تأمين الصفحة بواسطة T1.



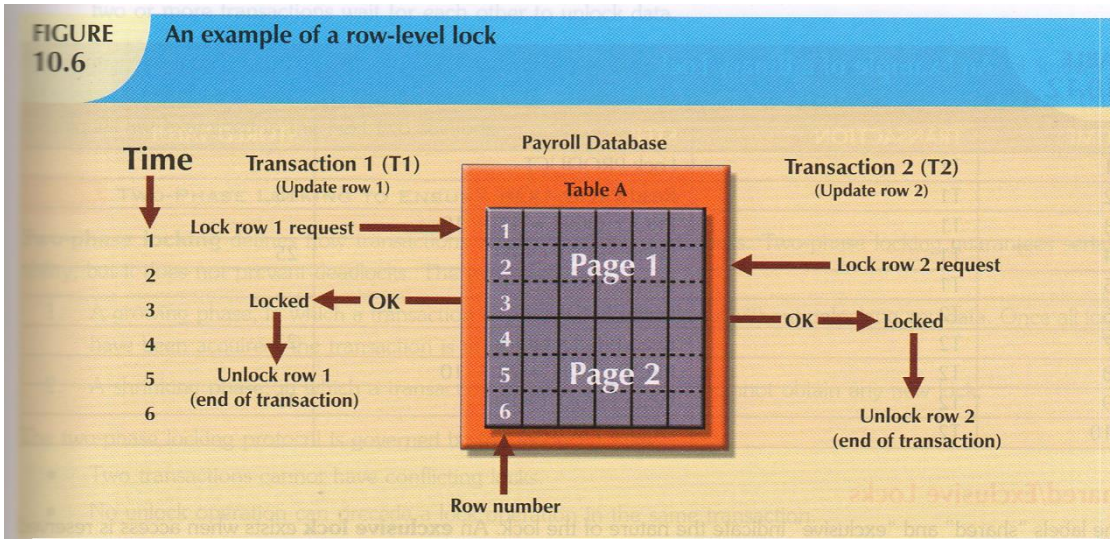
==========

ROW LEVEL

A row-level lock is much less restrictive than the locks discussed earlier. The DBMS allows concurrent transactions to access different rows of the same table even when the rows are located on the same page. Although the row—level locking approach improves the availability of data, its management requires high overhead because a lock exists for each row in a table of the database involved in a conflicting transaction. **Modern DBMSs automatically escalate a lock from a row-level to a page**—level lock when the application session requests multiple locks on the same page. Figure 10.6 illustrates the use of a row-level lock.

مستوى الصف

قفل مستوى الصف أقل تقييدًا بكثير من الأقفال التي تمت مناقشتها سابقًا. يسمح DBMS للمعاملات المتزامنة بالوصول إلى صفوف مختلفة من نفس الجدول حتى عندما تكون الصفوف موجودة في نفس الصفحة. على الرغم من أن أسلوب قفل مستوى الصفوف يحسِّن من توافر البيانات ، فإن إدارتها تتطلب عبءًا مرتفعًا نظرًا لوجود قفل لكل صف في جدول قاعدة البيانات المتضمن في معاملة متضاربة. تقوم أنظمة DBMS الحديثة تلقائيًا بتصعيد قفل من مستوى صف إلى صفحة - قفل مستوى عندما تطلب جلسة التطبيق أقفالًا متعددة على نفس الصفحة. يوضح الشكل 10.6 استخدام قفل مستوى الصف.



Note in Figure 10.6 that both transactions can execute concurrently, even when the requested rows are on the same page. T2 must wait only if it requests the same row as T1.

لاحظ في الشكل 10.6 أنه يمكن تنفيذ كلتا العمليتين بشكل متزامن ، حتى عندما تكون الصفوف المطلوبة في نفس الصفحة. يجب أن تنتظر T2 فقط إذا طلبت نفس الصف مثل T1.

Field Level

The field-level lock allows concurrent transactions to access the same row as long as they require the use of different fields (attributes) within that row. Although field-level locking clearly yields the **most flexible** multiuser data access, it is rarely implemented in a DBMS because it requires an extremely high level of **computer overhead** and because the row—level lock is much more useful in practice.

المستوى الميداني

يسمح القفل على مستوى الحقل للمعاملات المتزامنة بالوصول إلى نفس الصف طالما أنها تتطلب استخدام حقول (سمات) مختلفة داخل هذا الصف. على الرغم من أن القفل على مستوى الحقل يؤدي بوضوح إلى الوصول إلى البيانات متعددة المستخدمين الأكثر مرونة ، إلا أنه نادرًا ما يتم تنفيذه في نظام إدارة قواعد البيانات (DBMS) لأنه يتطلب مستوى عالٍ للغاية من عبء الكمبيوتر ولأن قفل مستوى الصف أكثر فائدة من الناحية العملية.

==========

LOCK TYPES:

Regardless of the level of locking, the DBMS may use different lock types:

1. Binary Locks

Have only two states: locked (1) or unlocked (0).

2. Shared/Exclusive Locks

An **exclusive lock** exists when access is reserved specifically for the transaction that locked the object .The exclusive lock must be used when the potential for conflict exists. A **shared lock** exists when concurrent transactions are granted read access on the basis of common lock. A shared lock produces no conflict as long as all the concurrent transactions are read only.

أنواع القفل:

بغض النظر عن مستوى القفل ، قد يستخدم نظام إدارة قواعد البيانات أنواع قفل مختلفة:

1. أقفال ثنائية

لديك حالتان فقط: مقفل (1) أو غير مؤمن (0).

2. أقفال مشتركة / حصرية

يوجد قفل حصري عندما يكون الوصول محجوزًا خصيصًا للمعاملة التي أغلقت الكائن. يجب استخدام القفل الحصري عند وجود احتمال حدوث تعارض. يوجد قفل مشترك عندما يتم منح المعاملات المتزامنة حق الوصول للقراءة على أساس القفل المشترك. لا ينتج عن القفل المشترك أي تعارض طالما أن جميع المعاملات المتزامنة تتم قراءتها فقط.

DEADLOCKS

A deadlock occurs when two transactions wait indefinitely for each other to unlock data.

The three basic techniques to control deadlocks are:

- Deadlock prevention . A transaction requesting a new lock is aborted when there is the possibility that a deadlock can occur. if the transaction is aborted , all changes made by this transaction are rolled back and all locks obtained by the transaction are released .The transaction is then rescheduled for execution.
- Deadlock detection. The DBMS periodically tests the database for deadlocks. if a deadlock is found one of the transactions is aborted (rolled back and restarted) and the other transaction are continues.
- Deadlock avoidance. The transaction must obtain all of the locks it needs before it can be executed .This technique avoids the rollback of conflicting transactions by requiring that locks be obtained in succession.

DEADLOCKS

يحدث طريق مسدود عندما تنتظر معاملتان إلى أجل غير مسمى حتى تقوم كل منهما الأخرى بإلغاء تأمين البيانات.

الأساليب الثلاثة الأساسية للسيطرة على الجمود هي:

- منع الجمود. يتم إلغاء المعاملة التي تطلب قفلًا جديدًا عندما يكون هناك احتمال بحدوث طريق مسدود. إذا تم إحباط المعاملة ، يتم إرجاع جميع التغييرات التي تم إجراؤها بواسطة هذه المعاملة ، ويتم تحرير جميع الأقفال التي تم الحصول عليها من خلال المعاملة ، ثم تتم إعادة جدولة المعاملة للتنفيذ.
- كشف الجمود. يقوم نظام إدارة قواعد البيانات (DBMS) باختبار قاعدة البيانات بشكل دوري بحثًا عن حالات الجمود. إذا تم العثور على طريق مسدود ، تم إحباط إحدى المعاملات (التراجع وإعادة التشغيل) وتستمر المعاملة الأخرى.
- تجنب الجمود. يجب أن تحصل المعاملة على جميع الأقفال التي تحتاجها قبل أن يتم تنفيذها. تتجنب هذه التقنية التراجع عن المعاملات المتضاربة من خلال طلب الحصول على الأقفال على التوالي.

Concurrency control with time stamping methods

The **time stamping** approach to scheduling concurrent transactions assigns a global, unique transaction. The time stamp value produces an explicit order in which transactions are submitted to the DBMS. Time stamps must have two properties: uniqueness and monotonicity. **Uniqueness** ensures that no equal time stamp values can exist; **Monotonicity** ensures that time stamp values always increase.

التحكم في التزامن مع طرق ختم الوقت

نهج الطابع الزمني لجدولة المعاملات المتزامنة يعيّن معاملة عالمية فريدة. تنتج قيمة الطابع الزمني ترتيبًا صريحًا يتم فيه إرسال المعاملات إلى نظام إدارة قواعد البيانات. يجب أن تحتوي الطوابع الزمنية على خاصيتين: التفرّد والرتابة. يضمن التفرد عدم وجود قيم طابع زمني متساوية ؛ تضمن الرتابة زيادة قيم الطابع الزمني دائمًا.

The disadvantage of this approach is that each value stored in the database requires two additional time stamp fields: one for the last time the field was read and one for the last update .Time stamping thus increases memory needs and the database's processing overhead.

عيب هذا الأسلوب هو أن كل قيمة مخزنة في قاعدة البيانات تتطلب حقلين إضافيين للطابع الزمني: أحدهما لآخر مرة تمت فيها قراءة الحقل والآخر لآخر تحديث ، وبالتالي يزيد ختم الوقت من احتياجات الذاكرة ونفقات معالجة قاعدة البيانات.

Concurrency control with optimistic methods

Each transaction moves through two or three phases are:

التحكم في التزامن بأساليب متفائلة

تنتقل كل معاملة خلال مرحلتين أو ثلاث مراحل هي:

- During the read phase, the transaction reads the database, executes the needed computations, and makes the updates to private copy of the database values. All update operations of the transaction are recorded in a temporary update file, which is not accessed by the remaining transactions.
- During the validation phase, the transaction is validated to ensure that the changes made will not affect the integrity and consistency of the database. if the validation test is positive , the transaction goes to the write phase .if the validation test is negative, the transaction is restarted and the changes are discarded.
- During the write phase, the changes are permanently applied to the database.

- أثناء مرحلة القراءة ، تقوم المعاملة بقراءة قاعدة البيانات وتنفيذ الحسابات المطلوبة وإجراء التحديثات على نسخة خاصة من قيم قاعدة البيانات. يتم تسجيل جميع عمليات التحديث للمعاملة في ملف تحديث مؤقت ، والذي لا يتم الوصول إليه بواسطة المعاملات المتبقية.
- أثناء مرحلة التحقق من الصحة ، يتم التحقق من صحة المعاملة للتأكد من أن التغييرات التي تم إجراؤها لن تؤثر على تكامل قاعدة البيانات واتساقها. إذا كان اختبار التحقق موجبًا ، تنتقل المعاملة إلى مرحلة الكتابة ، وإذا كان اختبار التحقق سالبًا ، يُعاد تشغيل المعاملة ويتم تجاهل التغييرات.
- أثناء مرحلة الكتابة ، يتم تطبيق التغييرات بشكل دائم على قاعدة البيانات.

The optimistic approach is acceptable for most read or query database systems that require few update transactions.

النهج المتفائل مقبول لمعظم أنظمة قواعد البيانات للقراءة أو الاستعلام التي تتطلب القليل من معاملات التحديث.

TWO-PHASE COMMIT PROTOCOL

The two-phase commit protocol guarantees that if a portion of a transaction operation cannot be committed, all changes made at the other sites participating in the transaction will be undone to maintain a consistent database state.

بروتوكول الالتزام من مرحلتين

يضمن بروتوكول الالتزام المكون من مرحلتين أنه إذا تعذر الالتزام بجزء من عملية المعاملة ، فسيتم التراجع عن جميع التغييرات التي تم إجراؤها في المواقع الأخرى المشاركة في المعاملة للحفاظ على حالة قاعدة بيانات متسقة.

Each DP maintains its own transaction log. The two-phase commit protocol requires that the transaction entry log for each DP be written before the database fragment is actually updated. Therefore, the two-phase commit protocol requires a DO-UNDO-REDO protocol and a write-ahead protocol.

يحتفظ كل موانئ دبي بسجل المعاملات الخاص به. يتطلب بروتوكول الالتزام على مرحلتين كتابة سجل إدخال المعاملة لكل DP قبل تحديث جزء قاعدة البيانات بالفعل. لذلك ، يتطلب بروتوكول الالتزام ذي المرحلتين بروتوكول DO-UNDO-REDO وبروتوكول الكتابة المسبقة.

==========

The DO-UNDO-REDO protocol is used by the DP to roll back and/or roll forward transactions with the help of the systems transaction log entries. The DO-UNDO-REDO protocol defines three types of operations:
يتم استخدام بروتوكول DO-UNDO-REDO بواسطة DP للتراجع و / أو إعادة المعاملات إلى الأمام بمساعدة إدخالات سجل معاملات الأنظمة. يحدد بروتوكول DO-UNDO-REDO ثلاثة أنواع من العمليات:

• **DO** performs the operation and records the "before" and "after" values in the transaction log.
• **UNDO** reverses an operation, using the log entries written by the DO portion of the sequence.
• **REDO** redoes an operation, using the log entries written by the DO portion of the sequence.

• يقوم بالعملية ويسجل القيم "قبل" و "بعد" في سجل المعاملات.
• يعكس UNDO إحدى العمليات ، باستخدام إدخالات السجل المكتوبة بواسطة جزء DO من التسلسل.
• يعيد REDO العملية باستخدام إدخالات السجل المكتوبة بواسطة جزء DO من التسلسل.

To ensure that the DO, UNDO, and REDO operations can survive a system crash while they are being executed, a write-ahead protocol is used .The **write-ahead protocol** ensures that transaction logs are always written before any database data are actually updated and forces the log entry to be written to permanent storage before the actual operation takes place.

لضمان أن عمليات DO و UNDO و REDO يمكن أن تنجو من تعطل النظام أثناء تنفيذها ، يتم استخدام بروتوكول الكتابة المسبقة. يضمن بروتوكول الكتابة المسبقة كتابة سجلات المعاملات دائمًا قبل تحديث أي بيانات قاعدة بيانات فعليًا و يفرض كتابة إدخال السجل إلى التخزين الدائم قبل حدوث العملية الفعلية.

The two-phase commit protocol defines the operations between two types of nodes: the coordinator and one or more subordinates. The participating nodes agree on a coordinator. Generally, the coordinator role is assigned to the node that initiates the transaction. However, different systems implement various, more sophisticated election methods. The protocol is implemented in two phases:
يحدد بروتوكول الالتزام المكون من مرحلتين العمليات بين نوعين من العقد: المنسق ومرؤوس واحد أو أكثر. العقد المشاركة توافق على منسق. بشكل عام ، يتم تعيين دور المنسق للعقدة التي تبدأ المعاملة. ومع ذلك ، فإن الأنظمة المختلفة تطبق أساليب انتخابية متنوعة وأكثر تعقيدًا. يتم تنفيذ البروتوكول على مرحلتين:

Phase 1: Preparation

The coordinator sends a PREPARE TO COMMIT message to all subordinates.
1. The subordinates receive the message; write the transaction log, using the write-ahead protocol; and send an acknowledgment (YES/PREPARED TO COMMIT or NO/NOT PREPARED) message to the coordinator.
2. The coordinator makes sure that all nodes are ready to commit, or it aborts the action.

If all nodes are PREPARED TO COMMIT, the transaction goes to phase 2. If one or more nodes reply NO or NOT PREPARED, the coordinator broadcasts an ABORT message to all subordinates.

المرحلة الأولى: التحضير
يرسل المنسق رسالة التحضير للالتزام إلى جميع المرؤوسين.
1. المرؤوسين تلقي الرسالة. كتابة سجل المعاملات باستخدام بروتوكول الكتابة المسبقة ؛ وإرسال رسالة إقرار (نعم / جاهز للالتزام أو لا / لم يتم الاستعداد) إلى المنسق.
2. يتأكد المنسق من أن جميع العقد جاهزة للالتزام ، أو أنه يجهض الإجراء.

إذا كانت جميع العقد معدة للالتزام ، تنتقل المعاملة إلى المرحلة 2. إذا ردت واحدة أو أكثر من العقد بـ "لا" أو "لم يتم إعدادها" ، يرسل المنسق رسالة "إحباط" إلى جميع المرؤوسين.

==========

Phase 2: The Final COMMIT

1. The coordinator broadcasts a COMMIT message to all subordinates and waits for the replies.
2. Each subordinate receives the COMMIT message, and then updates the database using the DO protocol.
3. The subordinates reply with a COMMITTED or NOT COMMITTED message to the coordinator.

If one or more subordinates did not commit, the coordinator sends an ABORT message, thereby forcing them to UNDO all changes.

المرحلة الثانية: الالتزام النهائي

1. يبث المنسق رسالة COMMIT إلى جميع المرؤوسين وينتظر الردود.
2. يتلقى كل مرؤوس رسالة COMMIT ، ثم يقوم بتحديث قاعدة البيانات باستخدام بروتوكول DO.
3. رد المرؤوسين برسالة تم الالتزام بها أو عدم الالتزام بها إلى المنسق.

إذا لم يلتزم مرؤوس واحد أو أكثر ، يرسل المنسق رسالة إحباط ، وبالتالي يجبرهم على إلغاء جميع التغييرات.