14th Feb. 2022
Monday.

Page 63

## Unit - 2.

## Transaction management & concurrency control

* Transaction.

Defi.
- A transaction is any action that reads from all writes to a database.
- A transaction can be consist of SELECT, UPDATE and INSERT statement.

A.
Transaction property. [Acid property]
- A transaction consist of following properties:
  - Atomicity
  - Consistency
  - Isolation
  - Durability
  - Serializability.

1) Atomicity.
- It requires that all operations of a transaction must be completed.
- If a transaction $T_1$ has 4 SQL request, all 4 request must be successfully completed otherwise the entire transaction is aborted.

2) Consistency.
- It indicates the permanence of database consistant state.
- When a tram. is completed, the database must be in a consistant state.

3) Isolation.
It means that the data use during the execution of a tram. can not be used by a second tram. until the first one is completed.

4) Durability.
It ensures that once tram. changes are done and commited, they can not be undone or lost. even in the event of a system failure.

5) Serializability
It ensures that the schedule for the concurrent execution of a tram. displays consistant result.

Q. 2 Transaction log.
- DBMS uses a transaction log to keep track of all tram. that update the database.
- The DBMS uses the information stored in this log for a recovery requirement.

- The tram. log consist of following information:
  - A record for beginning of the tram.
  - SQL statement
  - Name of the object affected by the tram. [tablename]
  - The before and after value
  - The pointers to the previous and next tram. log entries
  - The ending [commit] of the tram.

| Tram ID | Tram no. | PREV PTR | NEXT PTR | Operation | Table | Attribute | Before value | After value |
|---|---|---|---|---|---|---|---|---|
| 1 | 101 | null | 2 | START | --- | | | |
| 2 | 101 | 1 | 3 | UPDATE | PRODUCT | Qty | 35 | 135 |
| 3 | 101 | 2 | 4 | UPDATE | CUSTOMER | Balance | 1000 | 90 |
| 4 | 101 | 3 | null | COMMIT | --- | | | |

* Concurrency control.

- The parallel execution of a tram. in multiuser database system is called concurrency control.
- Its objective is to ensure the serializability of tram. in a multi-user database environment.
- Concurrency control has 3 main problems:
  - Lost update
  - Uncommited data
  - Incomsistant retrieval

A) Lost update

- The lost update problem occurs when 2 tram. $T_1$ and $T_2$ are updating the same data element and one of the update is lost.

e.g.,
Two concurrent tram. $T_1$ and $T_2$ update QTY value for some item in the PRODUCT table.
$T_1$ → Purchase 100 unit
$T_2$ → Sell 30 unit

Following table shows serial execution of tram. and provides correct result:

| Time | Tram. | Step | Stored value |
|---|---|---|---|
| 1 | $T_1$ | Read Qty | 35 |
| 2 | $T_1$ | Qty → Qty + 100 | |
| 3 | $T_1$ | Write Qty | 135 |
| 4 | $T_2$ | Read Qty | 135 |
| 5 | $T_2$ | Qty → Qty - 30 | |
| 6 | $T_2$ | Write Qty | 105 |

Following table shows the problem of lost update:

- The first tram. $T_1$ has not commited when tram. $T_2$ is executed.
- The value of tram. $T_1$ is overwritten by tram. $T_2$.

| Time | Tram. | Step | Stored value |
|---|---|---|---|
| 1 | $T_1$ | Read QTY | 35 |
| 2 | $T_2$ | Read QTY | 35 |
| 3 | $T_1$ | QTY = 35 + 100 | |
| 4 | $T_2$ | QTY = 35 - 30 | |
| 5 | $T_1$ | Write QTY | 135 |
| 6 | $T_2$ | Write QTY | 5 |

B) Uncommited data.

- The problem of uncommited data occurs when a tram. $T_1$ and $T_2$ are executed concurrently and $T_1$ is rollback after $T_2$ has already accesse the uncommited data.
- It violets the isolation property of tram.

e.g.,
The 2 concurrent tram. $T_1$ and $T_2$ updates QTY in the PRODUCT table.
$T_1$ → Purchase 100 unit
$T_2$ → Sell 30 unit

Following table shows serial execution of this tram. and displays correct answer:

| Time | Tram. | Step | Stored value |
|---|---|---|---|
| 1 | $T_1$ | Read Qty | 35 |
| 2 | $T_1$ | Qty → Qty + 100 | |
| 3 | $T_1$ | Write Qty | 135 |
| 4 | $T_1$ | Roll back | 35 |
| 5 | $T_2$ | Read Qty | 35 |
| 6 | $T_2$ | Qty = Qty - 30 | |
| 7 | $T_2$ | Write Qty | 5 |

Following table shows the problem of uncommited data when rollback is completed:

| Time | Tram. | Step | Stored value |
|---|---|---|---|
| 1 | $T_1$ | Read Qty | 35 |
| 2 | $T_1$ | Qty = Qty + 100 | |
| 3 | $T_1$ | Write Qty | 135 |
| 4 | $T_2$ | Read Qty | 135 |
| 5 | $T_2$ | Qty = Qty - 30 | |
| 6 | $T_1$ | Rollback | 35 |
| 7 | $T_2$ | Write Qty | 105 |

15th Feb. 2022
Tuesday.



c) Incomsistant retrievals.

- It occur when a tram. accesses data before and after one or more other tram. finish working with such data.
- The problem is that the tram. might read some data before they are changed and other data after they are changed which results incomsistant retrievals.

D) The scheduler.

- It is a special DBMS process that establish the order in which the operations are executed within concurrent tram.
- The scheduler provides execution of database operations to ensure isolation and serializability of tram.
- Its main job is to create a serializable schedule of a tram. operation.
- It also make sure that the computer's CPU and storage system are used efficiently.
- The problem with that approach is that the processing time is wasted when CPU wait for read or write operation to finish.
- In the following table, 2 operations are incomflict when they access the same data and atleast one of them is write operation.

| Operations | Transaction $T_1$ | Transaction $T_2$ | Result |
|---|---|---|---|
| | Read | Read | No conflict |
| | Read | Write | Conflict |
| | Write | Read | Conflict |
| | Write | Write | Conflict |

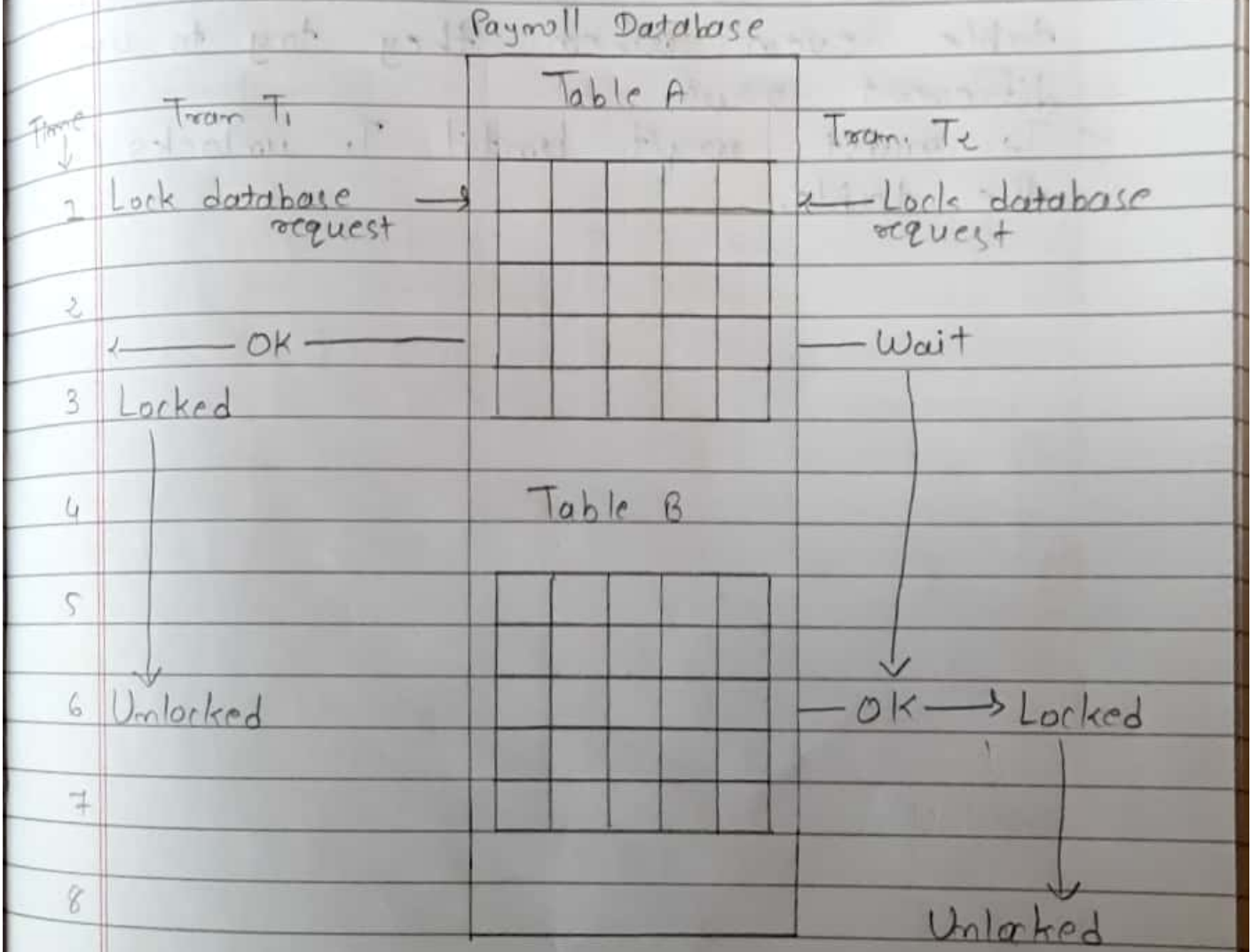
- Several methods are used like locking, time stamping, optimistic methods.

* Concurrency control with locking methods.
- A lock guarantees execusive use of data item to a current tram.
- It means that tram. $T_2$ does not have access to a data item that is currently used by tram. $T_1$.
- The use of lock based on the assumption that conflict between tram. is likely which is called pessimistic methods [locking].
- All lock information is handled by the lock manager which is responsible for assigning the locks used by the tram.

A) Lock granularity / level.
- It indicates the level of lock used.
- Locking can be take place at the database, table, page, row or even field [attribute].

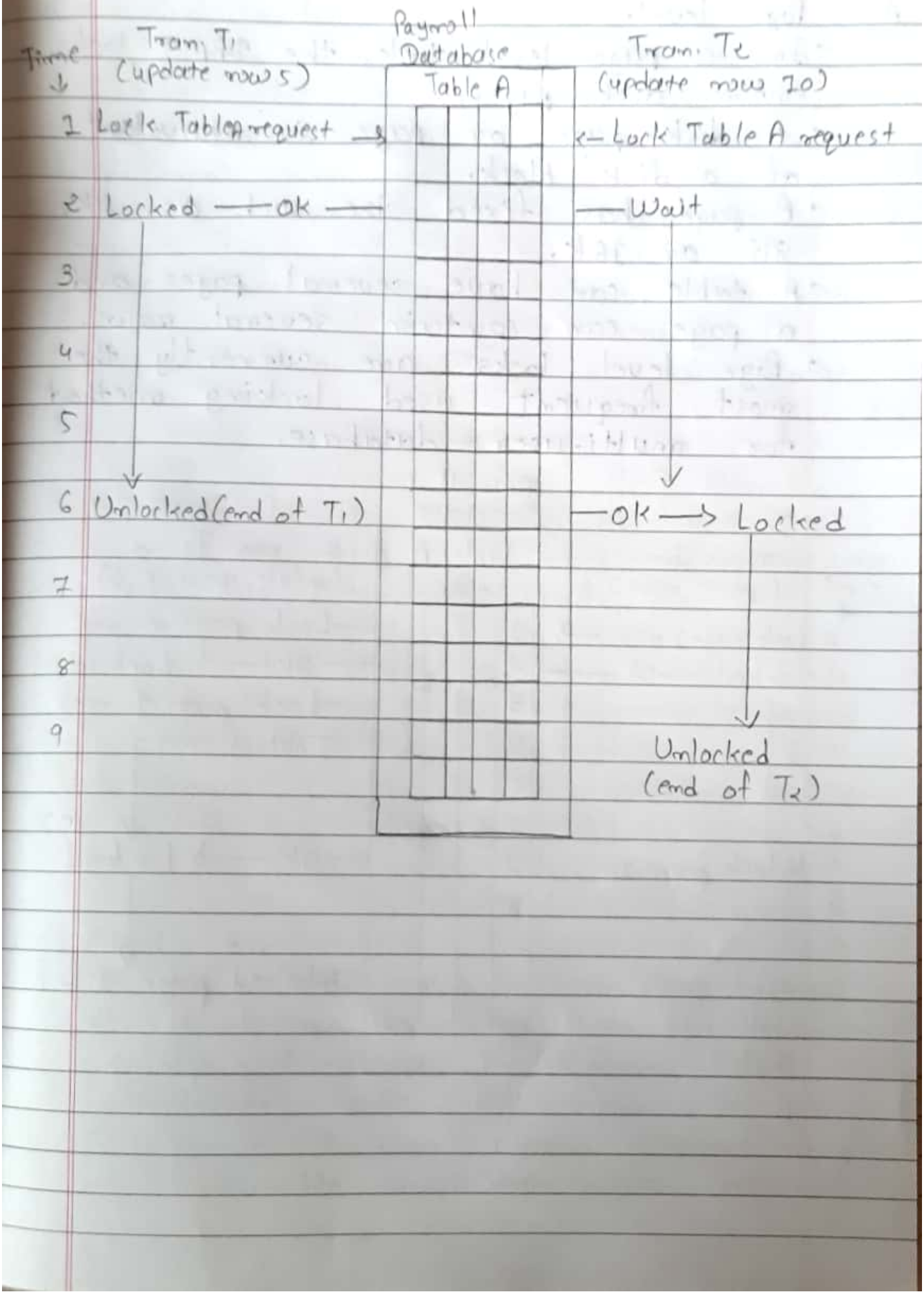
(A.1.) Database level.

- In a database level lock, the entire database is locked.
- It prevents the use of any tables in the database by tram. $T_2$ while tram. $T_1$ is executed.
- The following figure shows database level lock in which tram. $T_1$ and $T_2$ can not access the same database concurrently even when they use different tables.



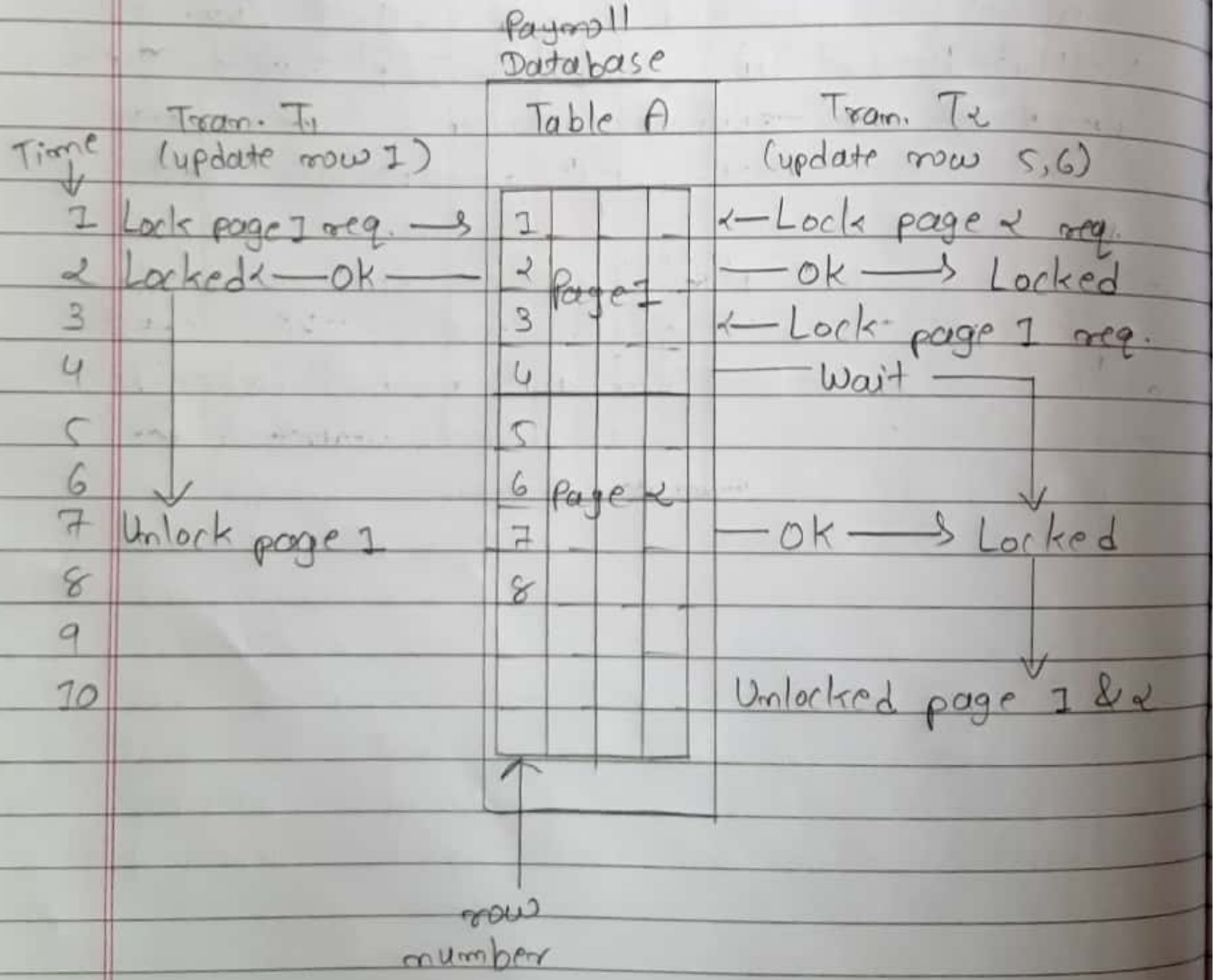
A.2.) Table level.

- In a table level lock, the entire table is locked.
- It prevents the access of any row by tran. $T_2$ while tran. $T_1$ is using the table.
- If a tran. requires access to several tables, each table maybe locked.
- Table level locks are not suitable for multi-user database.
- In the following figure tran. $T_1$ and $T_2$ can not access the same table even when they try to use different rows.
- $T_2$ must wait until $T_1$ unlocks the table.



A.3.) Page level.

- In a page level lock, the DBMS locks entire disk page.
- A disk page or page is equivalent of a disk block.
- A page has fixed size such as 4K, 8K or 16K.
- A table can have several pages and a page can contain several rows.
- Page level locks are currently the most frequent used locking method for multi-user database.

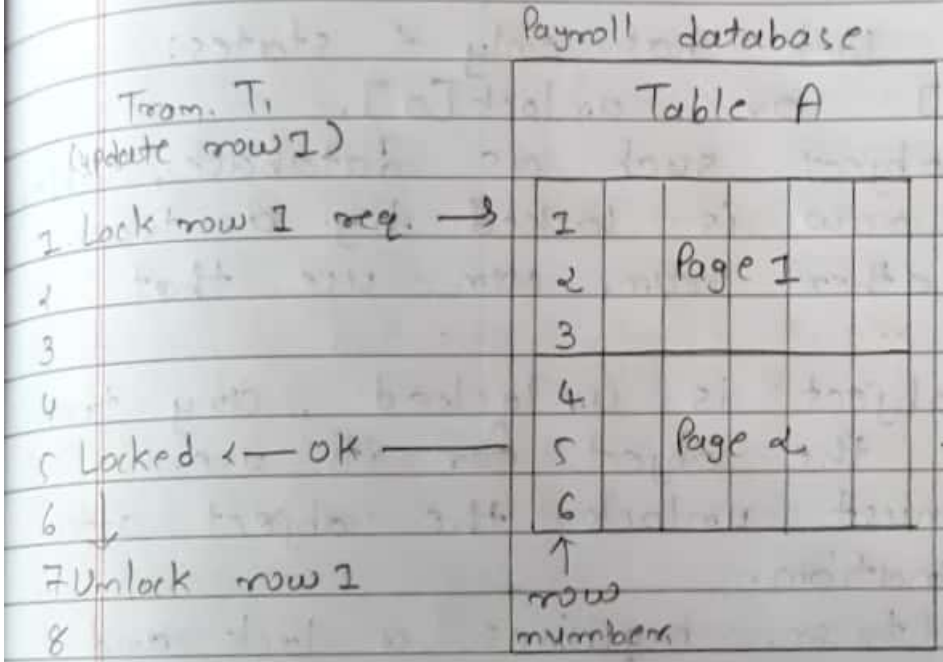


21st Feb. 2022
Monday.

Page 75

Q.No.) Rolling level.

- It is much less restrictive than the above locks.
- The dbms allows tran. to access different rows of the same table even when the rows are locked on the same page.
- The following example shows both tran. can execute certainly even when the requested rows are on the same page.




A.S.) Field level

- It allows concurrent tran. to access the same ~~row~~ row as long as they required the use of different field [attributes] within the row.
- Field level locking clearly displays the most flexible multi-user data access.

- The lock used by specified field can not be accessed by other tran. until the first one unlocks the field [attribute].

B) Lock types.

- DBMS uses 2 different types of lock:
  A) . Binary lock
  B) . Shared / exclusive lock.

B.1.) Binary lock.

- A binary lock has only 2 states: locked [1] or unlock [0].
- If an object such as database, table, page or row is locked by a tran. then no other tran. can use that object.
- If an object is unlocked, any tran. can lock the object for its use.
- A tran. must unlock the object after its termination.
- So, every tran. requires a lock and unlock operation for each accessed data items.
- The lock, unlock features eleminates the problem of lost update because lock is not realised until the WRITE statement is not completed.
- Binary locks are too restrictive.

Ex. In the following example tram. $T_1$ locks the PRODUCT table. So, tram. $T_2$ can not access the same table until tram. $T_1$ unlocks PRODUCT table.

| Time | Tram. | Step | Stored value |
|---|---|---|---|
| 1 | $T_1$ | Lock PRODUCT | |
| 2 | $T_1$ | Read QTY | 15 |
| 3 | $T_1$ | QTY = QTY + 10 | |
| 4 | $T_1$ | Write QTY | 25 |
| 5 | $T_1$ | Unlock PRODUCT | |
| 6 | $T_2$ | Lock PRODUCT | |
| 7 | $T_2$ | Read QTY | 25 |
| 8 | $T_2$ | QTY = QTY - 10 | |
| 9 | $T_2$ | Write QTY | 15 |
| 10 | $T_2$ | Unlock PRODUCT | |

B.2.) Shared / exclusive lock.

- The shared lock exists when concurrent tram. are granted read access on the bases of a common lock.
- A shared lock produces no conflict because the concurrent tram. are read only.
- A shared lock issued when a tram. want to read data from a database and no exclusive lock is held on that data item.

Ex. If tram. $T_1$ has a shared lock on data item X and tram. $T_2$ wants to read data item X then $T_2$ may also obtain a shared lock on data item X.

Shared lock

| Tram. | Operation | Shared lock |
|---|---|---|
| $T_1$ | Read X | Allow |
| $T_2$ | Read X | Allow |

Exclusive lock.

- An exclusive lock is issued when a tram. want to update [write] a data item and no locks are currently held on that data item by any other tram.
- The exclusive lock is granted if and only if no other locks are held on the data item.
- This condition is called exclusive rule:- only one tram. at a time can own an exclusive lock on the object.

Ex. 2 If tram. $T_1$ updates the data item X, an exclusive lock is required by $T_1$ over data item X.

Exclusive lock

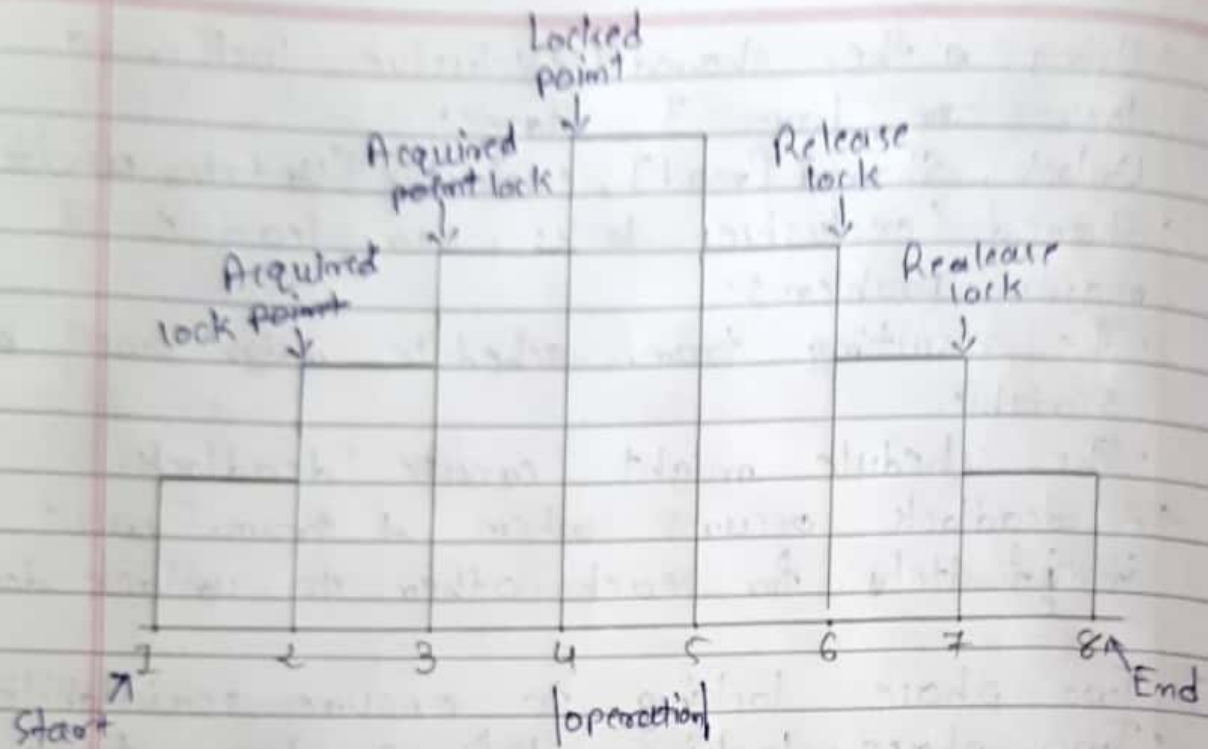
| Tram. | Operation | Exclusive lock |
|---|---|---|
| $T_1$ | Update X | Allow |
| $T_2$ | Update X | Wait |

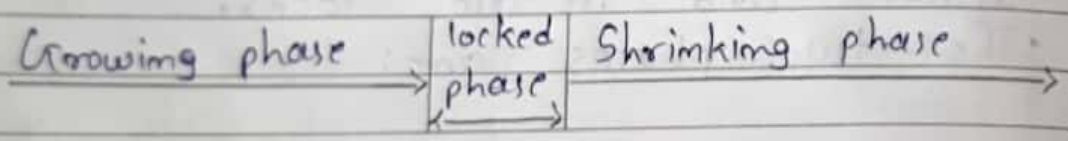
- Using the shared / exclusive lock, a lock can have 3 states:
Unlock, Shared [read], exclusive [Update, write]
- Shared / exclusive locks can lead to 2 major problems:
  - The resulting tran. schedule might not be seriable.
  - The schedule might create deadlock.
- A deadlock occurs when 2 tran. wait indefinitely for each other to unlock data.

(*) Two phase locking to ensure serializability.
- Two phase locking defines how tran. aquires and release locks.
- The two phases are:

(i) Growing phase.
- In this phase, a tran. acquire all required locks without ~~on~~ unlocking data.
- Once all locks have been acquired, the tran. is in its locked point.

(ii) Shrinking phase.
- In this phase, a tran. release all locks and can not obtain a new lock.




- The two phase locking protocol has following rules:
  1) Two tram. can not have comflicting locks.
  2) No: unlock operation can preside a lock operation.
  3) No data are affected until the tram. is in ~~th~~ its locked point.

- Two phase locking increase the tram. processing cost and might cause undescribable effects like deadlock.

22nd Feb. 2022
Tuesday.

Page 81

D) Deadlock.

- A deadlock occurs when 2 tram. wait indefinitely for each other to unlock data.
- For e.g., a deadlock occurs when 2 tram. $T_1$ and $T_2$ exist in the following mode:

$T_1$ = Access data item X & Y
$T_2$ = Access data item Y & X

If, $T_1$ has not unlocked data item Y, $T_2$ cam not begin.
If $T_2$ has not unlocked data item X, $T_1$ cam not begin.
$T_1$ and $T_2$ both wait for the other to unlock the required data item.

- There are 3 basic techniques to control deadlock:

A) Deadlock prevention.

- A tram. requesting a lock is aborted when there is a possibility that a deadlock cam occur.
- If the tram. is aborted, all changes made by the tram. are rollback.

B) Deadlock detection

- The DBMS periodically test the database for deadlock.
- If a deadlock is found, the victim tram. is aborted and other tram. continues.

c) Deadlock avoidance.
- The tram. must obtain all the locks it needs before it can be executed. This technique avoids conflicting tram.

* Concurrency control with timestamping methods.
- The timestamping approach to scheduling concurrence tram. assigns a global, unique timestamp to each tram.
- Timestamps must have 2 properties:
  - Uniqueness
    It ensures that no equal timestamp value can exist.
  - Monotonicity.
    It ensures that timestamp value always increase.
- All database operations within the same tram. must have same timestamp.
- If 2 tram. comflict then one is stopped, rollback, reschedule and assign a new timestamp value.

Method:
Wait / Die & Wound / Wait scheme

1) Wait / Die

- Using wait / die scheme, older tram. waits for the younger one to complete and release its lock.
- If the tram. requesting the lock is older of 2 tram., it will wait until other tram. is completed and locks are released.
- If the tram. requesting the lock is younger of 2 tram., it will die [rollback] and is rescheduled using the same timestamp.

| Tram. requesting lock | Tram. owning lock | Wait / Die scheme |
|---|---|---|
| $T_1$ (101) | $T_2$ (102) | $T_1$ waits until $T_2$ is completed & $T_2$ releases its locks. |
| $T_2$ (102) | $T_1$ (101) | $T_2$ dies [rollback]. $T_2$ is rescheduled using same timestamp. |

2) Wound / Wait

- In wound / wait scheme the older tram. rollbacks the younger tram. and reschedule it.
- If the tram. requesting the lock is older of 2 tram., it will preempt [rollback / wound] the younger tram. and the younger tram. is rescheduled using the same timestamp.
- If the tram. requesting the lock is younger of 2 tram., it will wait until other tram. is completed and its locks are released.

| Tram. requesting lock | Tram. owning lock | Wound / Wait scheme |
|---|---|---|
| $T_1$ (101) | $T_2$ (102) | $T_1$ preempts [rollback] $T_2$. $T_2$ is rescheduled using same timestamp. |
| $T_2$ (102) | $T_1$ (101) | $T_2$ waits until $T_1$ is completed & $T_1$ releases its locks. |

## Concurrency control with optimistic methods.

- The optimistic approach is based on the assumption that the database operation do not conflict.
- It requires neither locking nor timestamping methods.
- A tram. is executed without restrictions until it is commited.
- Using optimistic approach, each tram. has 3 phases: read, validation, write.

A) Read phase.
- During read phase, the tram. reads the database, executes needed computation and makes the update to a private copy of the database value.
- All update operations of the tram. are recorded in a temporary update file.

B) Validation phase.
- During this phase, the tram. is validated to ensure that the changes made will not affect integrity and consistancy of the database.
- If the validation test is positive, the tram. goes to the write phase.
- If the validation test is negative, the tram. is restarted and the changes are discarded.

c) Write phase.
- During this phase, the changes are permamently applied to the database.

2nd Feb. 2022
Wednesday.

* Database recovery management.
- Database recovery restores a database from a inconsistant state to the consistant state.
- If a tran. operation can not be completed for some reason, the tran. must be aborted and the changes are rollback.
- Critical events makes a database to stop working.
- This events can be
  - Hardware / software failure
    A failure of this type could be a hard disk failure, motherboard failure or application program and operating system failure.
  - Human caused insident
    This type of events can be categorised as unintentional or intentional.
    An unintentional failure is caused by a careless end user.
    Intentional events are more severe mature and indicate that the company data are at high risk.

- Natural disaster.
  This category includes earthquake, fire, fluad and power failure.

→ Transaction recovery.
- Tran. recovery uses data in the tran. log to recover a database. from inconsistant state to consistant state.
- 4 important concepts that affect the tran. recovery are

A) Write - ahead - log protocol.
It ensures that tran. logs are always written before any database data are actually updated.
It also ensures that incase of a failure, the database can be recovered to a consistant state.

B) Redundant - tran. - log
It ensures that physical disk failure will not impact the DBMS ability to recover data.

c) Database buffer.
They are the temporary storage area in primary memory used to speed up disk operations.

D) Database checkpoint
They are the operations in which DBMS write all the updated buffer to disk.

It is also registered in the tram: log.

→ Tram. recovery techmiques.
- Tram. recovery procedures make use of following 2 methods:
A) Deferred-write techmique
B) Write-through techmique

A) Deferred-write techmique
- When the recovery procedures uses ~~deff~~ deferred-write techmique, the tram. operations do not immediately update the physical database.
- Only tram. log is updated.
- The database is physically updated only after the tram. reaches its commit point using information from the tram. log.
- It is also called ~~deffe~~ deferred update.

B) Write-through techmique
- When the recovery procedure uses a write-through techmique, the database is immediately updated by tram. operation during execution, even before the commit point.
- If the tram. aborts before its commit point, a ROLLBACK operation needs to be done to restore the database to a consistant state.
- It is also called immediate update.

Unit-2. Questions.

1. What is tram.? Explain tram. property. [ACIDS property]
2. What is concurrency control? List problems associated with it. Explain any 1 in detail.
3. Explain lock granularity. [Levels of lock]
4. List and explain types of locks.
5. Explain 2 phase locking.
6. What is a deadlock? Explain its methods. to control deadlock.
7. Explain concurrency control with timestamping method.

   or

   Explain wait/die and wound/wait method.
8. List and explain optimistic methods of concurrency control.
9. Explain different tram. recovery concepts with recovery techniques.