Transactions, Recovery and Concurrency (II)

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Concurrenc rol

Concurrency Control Methods

- **Locking Mechanism**

The idea of locking some data item X is to:

- give a tran a item X,
- do not restr s.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

This prevents one transaction from changing a data item currently being used in another transaction.

- We will discuss a simple locking scheme which locks individual items, using read and write locks

Locking Rules

- In this schema, every transaction T must obey the following rules.

- 1) If T has only one operation (read/write) manipulating an item X:
Assignment Project Exam Help
  - obtain a read lo
  - obtain a write l
https://eduassistpro.github.io/
  - unlock X when done with it.
Add WeChat edu_assist_pro

- 2) If T has several operations manipulating X:
  - obtain one proper lock only on X:

    a read lock if all operations on X are reads;

    a write lock if one of these operations on X is a write.
  - unlock X after the last operation on X in T has been executed.

Locking Rules (cont.)

- In this scheme,
  - Several read locks can be issued on the same data item at the same time.

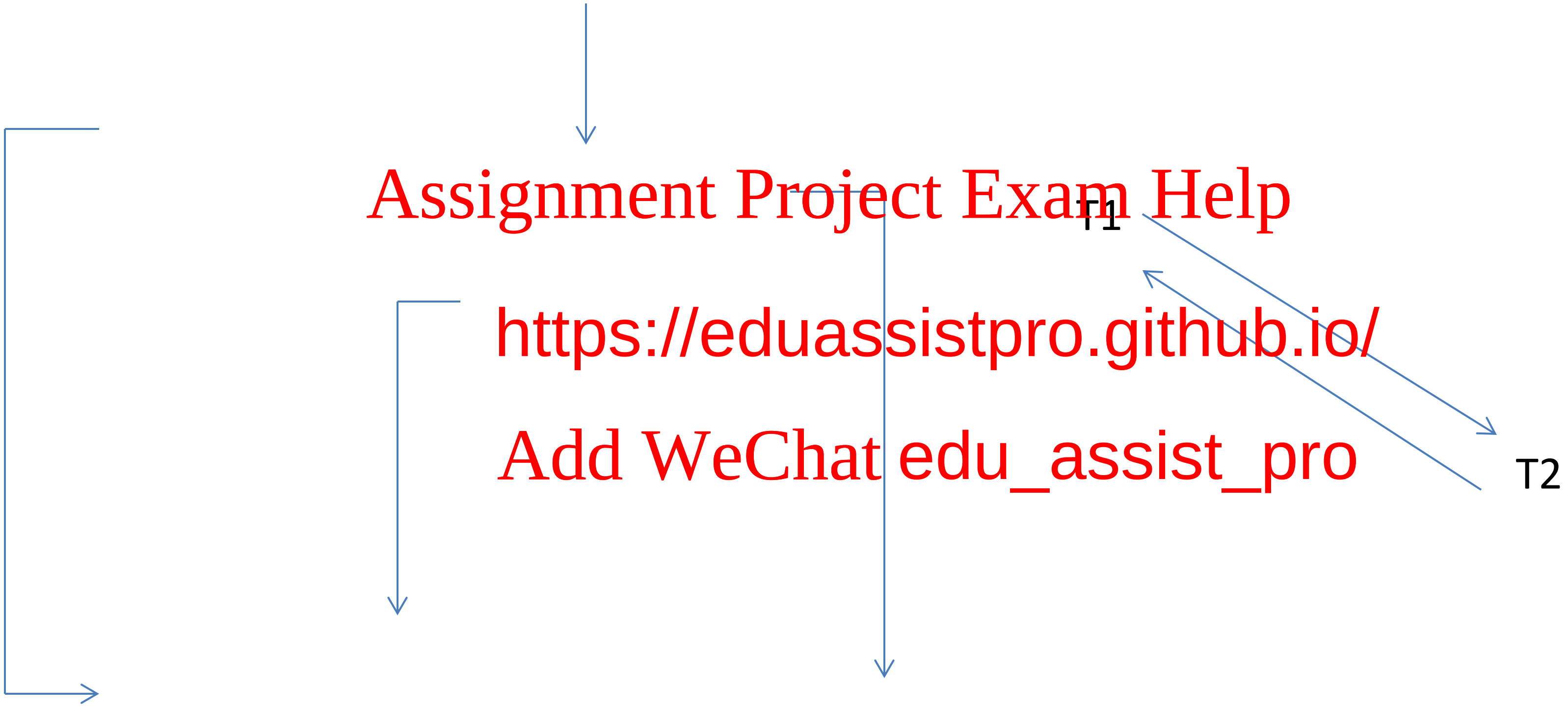
Assignment Project Exam Help

  - A read lock and a same data item at the same time, neither two

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- This still does not guarantee seriali



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Two Phase Locking (2PL)

- To guarantee serializability, transactions must also obey the *two-phase locking protocol*:

Assignment Project Exam Help

  - *Growing Phase* t be obtained before any locks are releas

https://eduassistpro.github.io/

  - *Shrinking Phase*: gradually release a lock is released no new locks may be requested).

Add WeChat edu_assist_pro

Two Phase Locking (2PL) (Cont.)

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- Locking thus provides a solution to the problem of correctness of schedules.

Two phase locking ensures conflict serializability

Deadlock

- A problem that arises with locking is **deadlock**.
- Deadlock occurs when two transactions are each waiting for a lock on an item held by the other.

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Deadlock Check

- Create the *wait-for graph* for currently active transactions:
  - create a vertex for each transaction; and
  - an arc from T_i to T_j if T_i is waiting for an item locked by T_j.

Assignment Project Exam Help

- If the graph has https://eduassistpro.github.io/ occurred.

Add WeChat edu_assist_pro

Several methods to deal with deadlocks

- *deadlock detection*
  - periodically check for deadlocks, abort and rollback some transactions (restart them later). This is a good choice if transactions pendent.

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Several methods to deal with deadlocks (Cont.)

- *deadlock prevention* - Assign priorities based on timestamps. Assume Ti wants a lock that Tj holds. Two policies are possible:

Assignment Project Exam Help

  - Wait-Die: If Ti Tj; otherwise Ti aborts

https://eduassistpro.github.io/

  - Wound-wait: If Ti has higher prior otherwise Ti waits

Add WeChat edu_assist_pro

- If a transaction re-starts, make sure it has its original timestamp

Timestamp ordering

- The idea here is:

  – to assign each transaction a timestamp (e.g. start time of transaction), and

  – to ensure that the schedule used is xecuting the transactions in timestamp order.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- Each data item, X, is assigned
  - a read timestamp, *read_TS*(X) - the latest timestamp of a transaction that read X, and
  - a write timesta estamp of a transaction that write X.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- These are used in read and write operations as follows. Suppose the transaction timestamp is T.

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- **Thomas' write rule:**

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

T_1	T_2	T_3
read (x)		
	read (y)	
	write (y)	
read (z)		
		read (z)
Write (z)		

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

r_TS (x) = 0 → 1
w_TS(x) = 0
r_TS (y) = 0 → 2
w_TS(y) = 0 → 2
r_TS (z) = 0 → 1 → 3
w_TS(z) = 0 → 3

- Some problems:
  - Cyclic restart: There is no deadlock, but a kind of livelock can occur - some transactions may be constantly aborted and restarted.

Assignment Project Exam Help

https://eduassistpro.github.io/

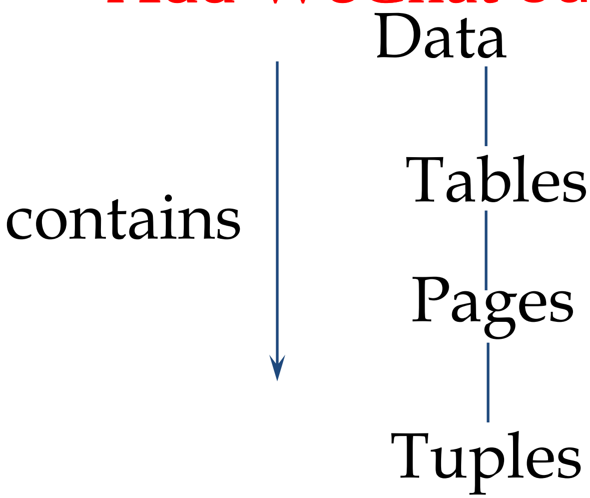
Add WeChat edu_assist_pro

  - Cascading roll led back, so are any transactions w transactions which read a value written by the can be avoided by not allowing transactions to read valu ncommitted transactions (make them wait).

Multiple-Granularity Locks

- Hard to decide what granularity to lock (tuples vs. pages vs. tables).
- Shouldn't
- Data “cont

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro



Solution: New Lock Modes, Protocol

- Allow Xacts to lock at each level, but with a special protocol using new “intention” locks:

Assignment Project Exam Help

- Before locki
must set “int
https://eduassistpro.github.io/
its ancestors.
Add WeChat edu_assist_pro
- For unlock, go from specifi
general (i.e., bottom-up).
- SIX mode: Like S & IX at the same time.

	--	IS	IX	S	X
	√	√	√	√	√
IS	√	√	√	√	
IX	√	√	√		
S	√	√		√	
X	√				

Multiple Granularity Lock Protocol

- Each Xact starts from the root of the hierarchy.
- To get S or X lock on a node, must hold IS or IX on parent node.
Assignment Project Exam Help
  - What if X t? S on parent?
https://eduassistpro.github.io/
- To get X or IX or SIX on a nod IX or SIX on parent node.
Add WeChat edu_assist_pro
- Must release locks in bottom-up order.

Protocol is correct in that it is equivalent to directly setting locks at the leaf levels of the hierarchy.

Examples

- T1 scans R, and updates a few tuples:
  - T1 gets an SIX lock on R, then repeatedly gets an S lock on tuples of R, and occasionally upgrades to X on the tuples.
- T2 uses an index to
  - T2 gets an IS lock on R, and repeatedly gets an S lock on R.
- T3 reads all of R:
  - T3 gets an S lock on R.
  - OR, T3 could behave like T2; can use lock escalation to decide which.

	--	IS	IX	S	X
--	√	√	√	√	√
IS	√	√	√	√	
IX	√	√	√		
S	√	√		√	
X	√				

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Dynamic Databases

Sailors (*sid*: integer, *sname*: string, *rating*: integer, *age*: real)
Reserves (*sid*: integer, *bid*: integer, *day*: dates, *rname*: string)

- If we relax the assumption that the DB is a fixed collection of objects, even Strict 2PL will not assure serializability:
  - T1 locks all pages containing sailor records with *rating* = 1, say, *age* = 71).
  - Next, T2 i = 1, *age* = 96.
  - T2 also deletes oldest sail ng = 2 (and, say, *age* = 80), and commits.
  - T1 now locks all pages containing sailor records with *rating* = 2, and finds oldest (say, *age* = 63).
- No consistent DB state; however T1 "correctly" gets through!

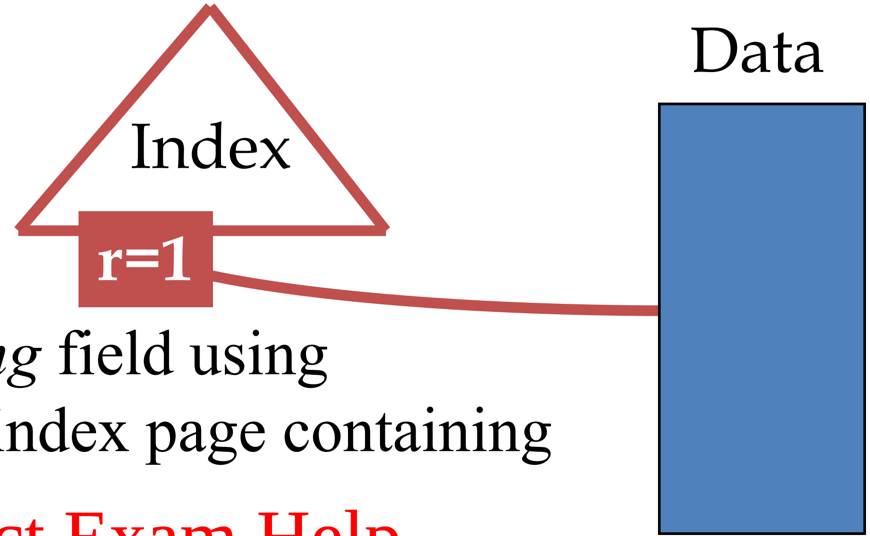
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

The Problem

- T1 implicitly assumes that it has locked the set of all sailor records with *rating* = 1.
  - Assumption only holds if no sailor records are added while T1 is executing!
  - Need so rce this assumption. (Index lo predicate locking.)
- Example shows that conflict serializability guarantees serializability only if the set of objects is fixed!

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Index Locking



- If there is a dense index on the *rating* field using Alternative (2), T1 should lock the index page containing the data entries with *rating* = 1.

Assignment Project Exam Help

  - If there are no 1, T1 must lock the index

https://eduassistpro.github.io/

    ta entry *would* be, if it existed!

Add WeChat edu_assist_pro

- If there is no suitable index, T1 must lock all pages, and lock the file/table to prevent new pages from being added, to ensure that no new records with *rating* = 1 are added.

Predicate Locking

- Grant lock on all records that satisfy some logical predicate, e.g. *age > 2*salary*.

Assignment Project Exam Help

- Index lockin f predicate locking for orts efficient implementation of the p ock.

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- What is the predicate in the sailor example?
- In general, predicate locking has a lot of locking overhead.

Locking in B+ Trees

- How can we efficiently lock a particular leaf node?
Assignment Project Exam Help
  - Btw, don't confuse this with multiple granularity locking!
- One solution https://eduassistpro.github.io/ cture, just lock pages while traversing th Add WeChat edu_assist_pro owing 2PL.
- This has terrible performance!
  - Root node (and many higher level nodes) become bottlenecks because every tree access begins at the root.

Two Useful Observations

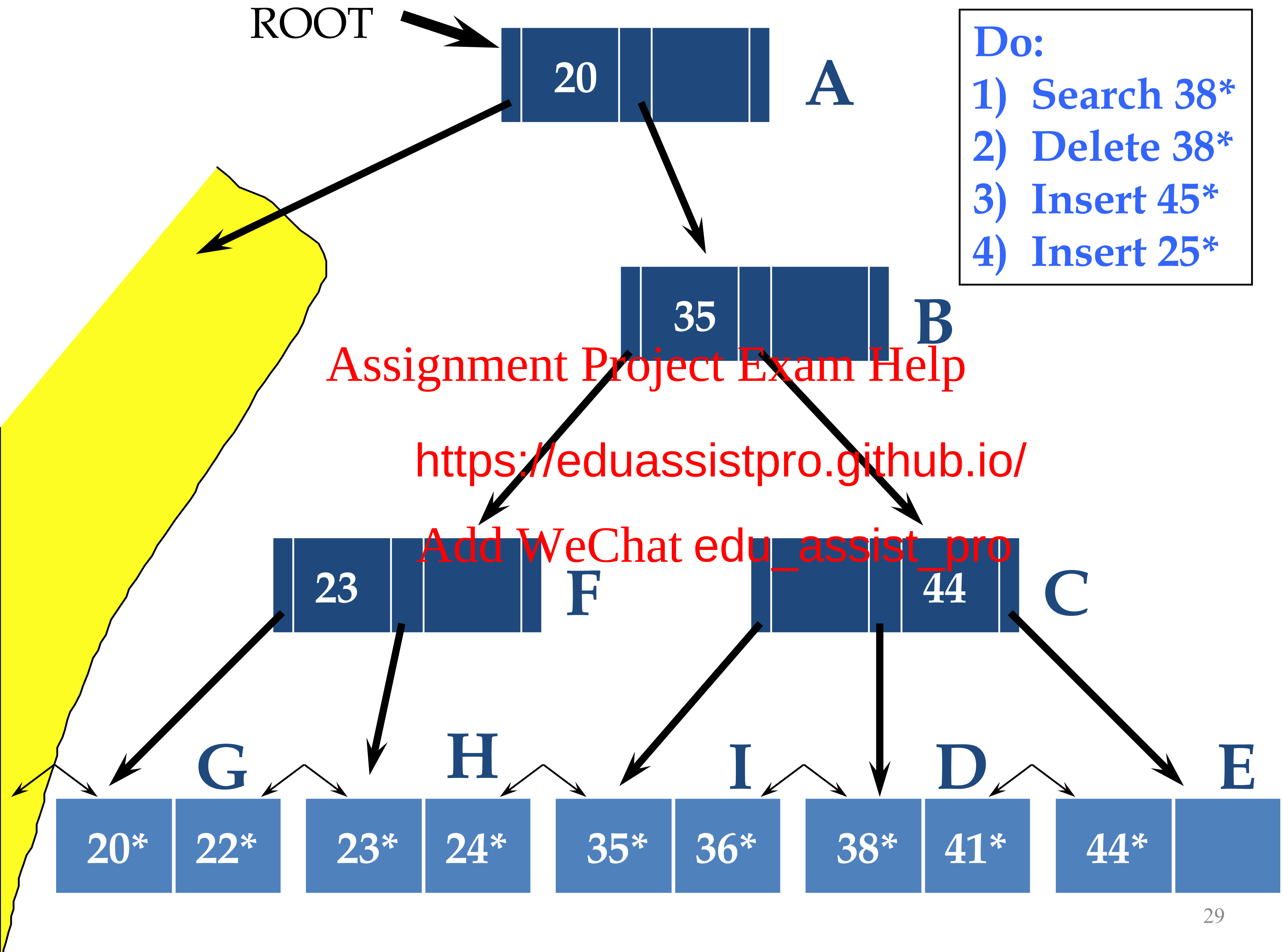
- Higher levels of the tree only direct searches for leaf pages.
- For inserts, a node on a path from root to modified leaf must be l course), only if a split can pr e modified leaf. (Similar point holds es.)
- We can exploit these observations to design efficient locking protocols that guarantee serializability *even though they violate 2PL.*

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

A Simple Tree Locking Algorithm

- Search: Start at root and go down; repeatedly, S lock child then unlock parent.
- Insert/Delete: Start at root and go down, obtaining X locks as needed. O if it is safe:
  - If child i s on ancestors.
- Safe node: Node such that c ot propagate up beyond this node.
  - Inserts: Node is not full.
  - Deletes: Node is not half-empty.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro



Do:
1) Search 38*
2) Delete 38*
3) Insert 45*
4) Insert 25*

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

A Better Tree Locking Algorithm (See Bayer-Schkolnick paper)

- Search: As before.
- Insert/Delete:

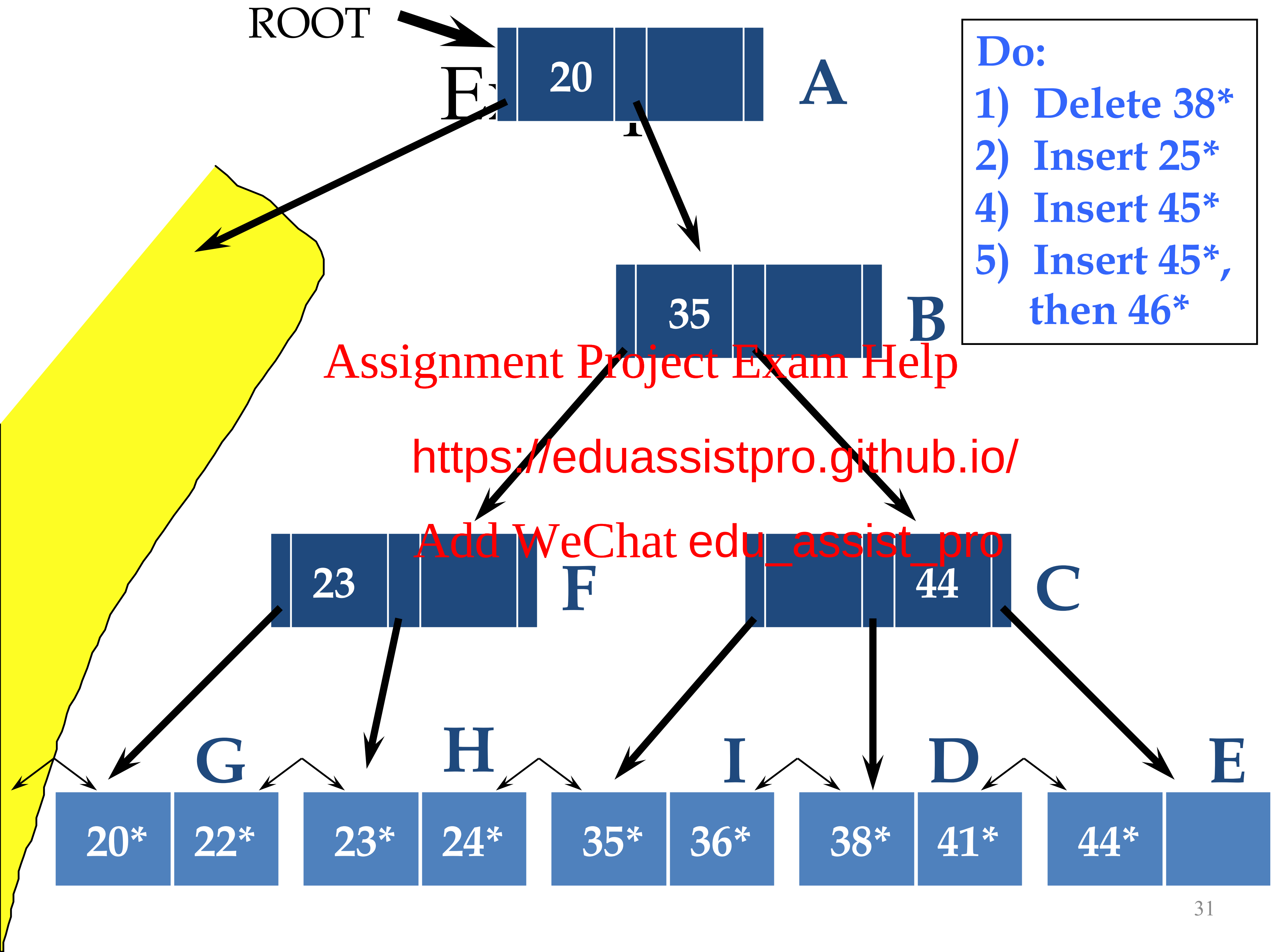
Assignment Project Exam Help

  - Set lock leaf, and set X lock on l

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

  - If leaf is not safe, rele s, and restart Xact using previous Insert/Delete protocol.
- Gambles that only leaf node will be modified; if not, S locks set on the first pass to leaf are wasteful. In practice, better than previous alg.



Do:

1) Delete 38*
2) Insert 25*
4) Insert 45*
5) Insert 45*, then 46*

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Even Better Algorithm

- Search: As before.
- Insert/Delete:
  - Use original Insert/Delete protocol, but set IX locks ins odes.
  - Once leaf is locked, co locks to X locks top-down: i.e., st node nearest to root. (Top-down reduces chances of deadlock.)

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

(Contrast use of IX locks here with their use in multiple-granularity locking.)

Hybrid Algorithm

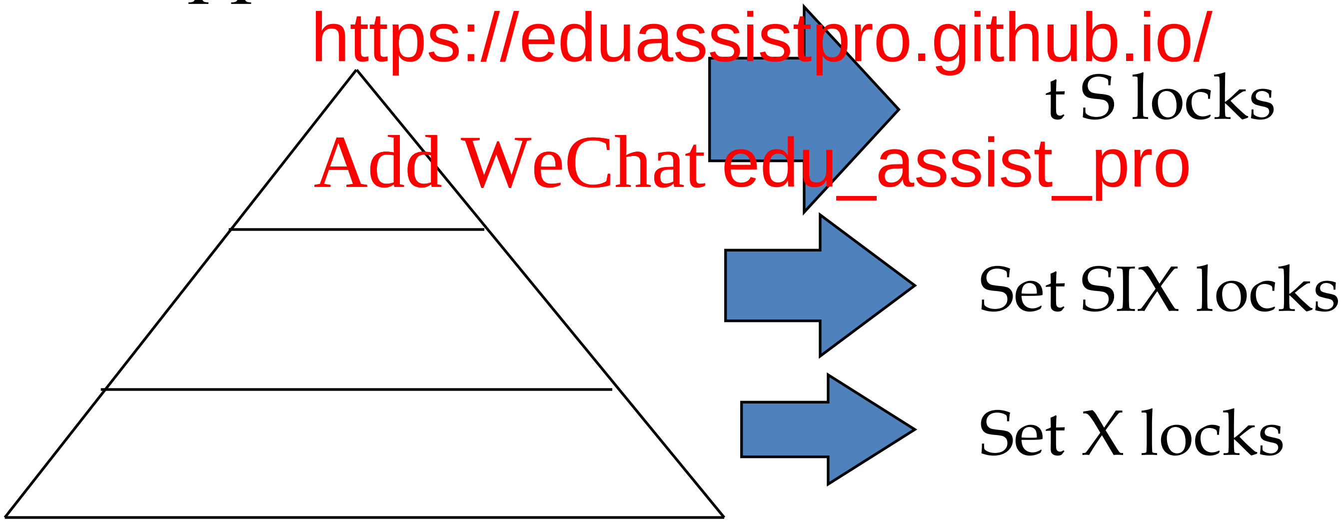
- The likelihood that we really need an X lock decreases as we move up the tree.

Assignment Project Exam Help

- Hybrid appr

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



Multiversioning

- Similar to the timestamp ordering approach; but is allowed to access “old” versions of a table.

Assignment Project Exam Help

- A history of the ersions) of each item is kept.

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- When the value of an item is needed, the system chooses a **proper** version of the item that maintains serializability.

- This results in fewer aborted transactions at the cost of greater complexity to maintain more versions of each item.

- We will look at a scheme, several versions $X_1, \ldots, X_k$ of each data item are kept. For each X_i we also keep

Assignment Project Exam Help

  – *read* $TS(X_i)$ - a

https://eduassistpro.github.io/

  – *write* $TS(X_i)$ - as for timestamp or

Add WeChat edu_assist_pro

- Read and write are done as follows for a transaction *P* with timestamp T.

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- *Note:* Cascading rollback and cyclic restart problems can still occur, but should be reduced.

- However, there is an increased overhead in maintaining multiple versions of items.

Optimistic scheduling

- In two-phase locking, timestamp ordering, and multiversioning concurrency control techniques, a certain degree of checking is done **before** a database operation can be executed.

Assignment Project Exam Help

- The idea here is t https://eduassistpro.github.io/ est!

Add WeChat edu_assist_pro

- No checking is done while the tran cuting.

- The protocol has three phases.

  – *read phase* - A transaction can read data items from the database into local variables. However, updates are applied only to local copies of the data items kept in the transaction workspace.

Assignment Project Exam Help

https://eduassistpro.github.io/

  – *validation pha* hat serializability is not violated,

Add WeChat edu_assist_pro

  – *write phase* -if validation succeeds, updates are applied and the transaction is committed. Otherwise, the updates are discarded and the transaction is restarted.

- A scheme uses timestamps and keeps each transaction's
  - read-set - the set of items read by the transaction,
  - write-set - the set of items written by the transaction.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

- During validation, we check that t does not interfere with any transaction that is committed or currently validating.

- Each transaction T is assigned 3 timestamps:

  Start(T), *Validation*(T), *Finish*(T).

- To pass the validation test for T, one of the following must be true:

Assignment Project Exam Help

  – 1. *Finish*(S) $<$

https://eduassistpro.github.io/

  – 2. for S s.t. *Start*(T) $<$ *Finish*(S), t

Add WeChat edu_assist_pro

    a) write set of S is disjoint from the nd

    b) *Finish*(S) $<$ *Validation*(T).

- Optimistic control is a good option if there is not much interaction between transactions.

- **Note:** Our earlier treatment of recovery methods largely ignored concurr

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

2PL vs. TSO vs. MV vs. OP

- A Comparison among two-phase locking (2PL), timestamp ordering (TSO), multiversioning (MV), optimistic (OP) concurrency control techniques.

Assignment Project Exam Help

- MV should provi https://eduassistpro.github.io/ egree (in average). However, we nee ach data item.

Add WeChat edu_assist_pro

- 2PL can offer the second greatest concurrency degree (in average); but will result in deadlocks. To resolve the deadlocks, either
  - need additional computation to detect deadlocks and to resolve the deadlocks, or
  - reduce the concurrency degree to prevent deadlocks by adding other restrictions.

2PL vs. TSO vs. MV vs. OP (cont.)

- If most transactions are very short, we can use 2PL + deadlock detection and resolution.

Assignment Project Exam Help

- TSO has a less co f 2PL if a proper deadlock resoluti oes not cause deadlocks. Other art and cascading rollback, will appear in TSO.

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- If there are not much interaction between transactions, OP is a very good choice. Otherwise, OP is a bad choice.