Concurrency Control 4

Instructor: Beom Heyn Kim

beomheynkim@hanyang.ac.kr

Department of Computer Science

Overview

- Insert Operations, Delete Operations, and Predicate Reads
- Assignments

Insert/Delete Operations and Predicate Reads

- Locking rules for insert/delete operations
  - An exclusive lock must be obtained on an item before it is deleted
  - A transaction that inserts a new tuple into the database is automatically given an X-mode lock on the tuple
- Ensures that
  - reads/writes conflict with deletes
  - Inserted tuple is not accessible by other transactions until the transaction that inserts the tuple commits

Phantom Phenomenon

- Example of **phantom phenomenon**.
  - A transaction T1 that performs **predicate read** (or scan) of a relation
    - **select count**(*)
      from *instructor*
      where *dept_name* = 'Physics'
  - and a transaction T2 that inserts a tuple while T1 is active but after predicate read
    - **insert into** *instructor* **values** ('11111', 'Feynman', 'Physics', 94000)
      (conceptually) conflict in spite of not accessing any tuple in common.
- If only tuple locks are used, non-serializable schedules can result
  - E.g. the scan transaction does not see the new instructor, but may read some other tuple written by the update transaction
- Can also occur with updates
  - E.g. update Wu's department from Finance to Physics

Insert/Delete Operations and Predicate Reads

- **Another Example**: T1 and T2 both find maximum instructor ID in parallel, and create new instructors with ID = maximum ID + 1
  - Both instructors get same ID, not possible in serializable schedule
- Schedule

T1	T2
Read(instructor where dept_name='Physics')	
	Insert Instructor in Physics
	Insert Instructor in Comp. Sci.
	Commit
Read(instructor where dept_name='Comp. Sci.')	

Handling Phantoms

- There is a conflict at the data level
  - The transaction performing predicate read or scanning the relation is reading information that indicates what tuples the relation contains
  - The transaction inserting/deleting/updating a tuple updates the same information.
  - The conflict should be detected, e.g. by locking the information.
- One solution:
  - Associate a data item with the relation, to represent the information about what tuples the relation contains.
  - Transactions scanning the relation acquire a shared lock in the data item,
  - Transactions inserting or deleting a tuple acquire an exclusive lock on the data item. (Note: locks on the data item do not conflict with locks on individual tuples.)
- Above protocol provides very low concurrency for insertions/deletions.

Index Locking To Prevent Phantoms

- **Index locking protocol** to prevent phantoms
  - Every relation must have at least one index.
  - A transaction can access tuples only after finding them through one or more indices on the relation
  - A transaction T_i that performs a lookup must lock all the index leaf nodes that it accesses, in S-mode
    - Even if the leaf node does not contain any tuple satisfying the index lookup (e.g. for a range query, no tuple in a leaf is in the range)
  - A transaction T_i that inserts, updates or deletes a tuple t_i in a relation r
    - Must update all indices to r
    - Must obtain exclusive locks on all index leaf nodes affected by the insert/update/delete
  - The rules of the two-phase locking protocol must be observed
- Guarantees that phantom phenomenon won't occur

Overview

- Insert Operations, Delete Operations, and Predicate Reads
- Assignments

Assignments

- Reading: Ch18.4
- Practice Excercises: n/a

Solutions to the Practice Excercises:
https://www.db-book.com/Practice-Exercises/index-solu.html

The End