101

Lock Conversions

101

Lock Conversions

102

- Lock upgrade: Xact that holds a shared lock can be upgraded to hold an exclusive lock
  - Reduces concurrency
- Lock downgrade: Xact that holds an exclusive lock, downgrades to a shared lock.
  - Improves concurrency (holding locks for writing when not required)
  - Reduces deadlocks

102

2PL with lock conversions

103

- During growing phase:
  - can acquire an S-lock on item
  - can acquire an X-lock on item
  - can convert an S-lock to an X-lock (upgrade)
  - Special case: allow lock downgrades only if Xact did not modify the data object (read only)
- During shrinking phase:
  - can release an S-lock
  - can release an X-lock
  - can convert an X-lock to an S-lock (downgrade)

Credit: Y. Chen, P. Bernstein, R. Ramakrishnan, J. Gehrke

103

Multiple Granularity Locking (MGL)

104

- We've referred to locking 'data objects'
- Sometimes preferable to group several data objects together
  - E.g., locking all records in a file
  - Define multiple levels of locking *granularity*
- Database consists of different data objects:
  - a field (attribute)
  - a database record
  - a page
  - a table
  - a file
  - the entire database

104

MGL (cont'd)

105

- Coarse grained locking → lower degree of concurrency
- Fine grained locking → higher degree of concurrency
  → Increased locking overhead
- What is the best locking granularity?

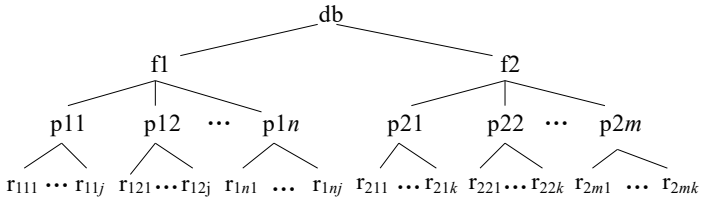
It depends on the transactions and the application

105

Granularity Hierarchy

106

- Most DBMS support multiple levels of locking granularity for different transactions



106

Problem with S and X Locks

107

T1: updates all the records in file f1.
T2: read record r_{1nj}.

Assume that T1 comes before T2:

- T1 locks f1.
- Before T2 is executed, the compatibility of the lock on r_{1nj} with the lock on f1 should be checked.

Assume that T2 comes before T1:

- T2 locks r_{1nj}.
- Before T1 is executed, the compatibility of the lock on f1 with the lock on r_{1nj} should be checked.
- Lock manager must efficiently manage all lock requests across hierarchy
- Each data object (e.g., file or record) has a different id

107

Solution

108

- Exploit the natural hierarchy of data containment
- Before locking fine-grained data, set *intention locks* on coarse grained data that contains it
- E.g., before setting an S-lock on a record, get an intention-shared-lock on the table that contains the record

108

Intention Locks

109

Three types of intention locks:

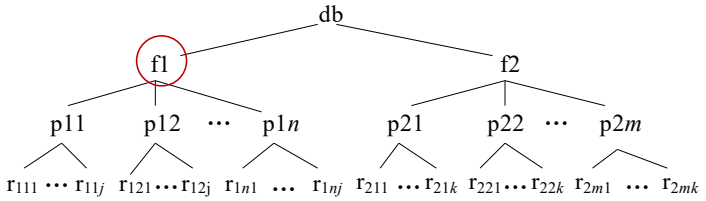
1. Intention-shared (IS) indicates that a shared lock(s) will be requested on some descendant node(s).
2. Intention-exclusive (IX) indicates that an exclusive lock(s) will be requested on some descendant node(s).
3. Shared-intention-exclusive (SIX) indicates that the current node is locked in shared mode but an exclusive lock(s) will be requested on some descendant node(s).

109

Back to our example

110

T1: updates all the records in file f1.
T2: read record r_{1nj}.



110

Compatibility Matrix

111

SIX = read with intent to write, e.g., for a scan that updates some of the records it reads

	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

IS conflicts with X because IS says there's a fine-grained S-lock that conflicts with an X-lock

111

Multiple Granularity Lock Protocol

112

- Each Xact starts from the root of the hierarchy.
- To get S or IS lock on a node, must hold IS or IX on parent node.
- To get X or IX or SIX on a node, must hold IX or SIX on parent node.
- Must release locks in bottom-up (leaf to root) order.

112

Examples

113

- 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 read only part of R:
  - T2 gets an IS lock on R, and repeatedly gets an S lock on tuples of 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 specific tuples to get locks on (rather than locking the entire R).

113