Isolation Levels (cont'd)

65

- SQL-92 provides different isolation levels that control the degree of concurrency

Isolation Level	In DB2	Dirty Read	Unrepeatable Read
Read Uncommitted	Uncommitted Read	Maybe	Maybe
Read Committed	Cursor Stability (default)	No	Maybe
Repeatable Reads	Repeatable Reads	No	No
Serializable	Read Stability	No	No

Phantoms possible

Degrees of Isolation in SQL

66

- **Four levels of isolation**
  - READ UNCOMMITTED: no read locks
  - READ COMMITTED: short duration read locks
  - REPEATABLE READ:
    - Long duration read locks on individual items
  - SERIALIZABLE:
    - All locks long duration
- **Trade-off: consistency vs concurrency**
- Commercial systems give choice of level

Isolation Levels in SQL

67

1. "Dirty reads"
   SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
2. "Committed reads"
   SET TRANSACTION ISOLATION LEVEL READ COMMITTED
3. "Repeatable reads"
   SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
4. Serializable transactions
   SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

ACID

Choosing Isolation Level

68

- Trade-off: efficiency vs correctness
- DBMSs give user choice of level

Beware!!
- Default level is often NOT serializable
- Default level differs between DBMSs
- Serializable may not be exactly ACID

Always read docs!

Performance of Locking

69

- Locking aims to resolve conflicts among transactions by:
  - Blocking
  - Aborting

  } Performance penalty

- Blocked Xacts hold locks other Xacts may want
- Aborting Xact wastes work done thus far
- Deadlock: Xact is blocked indefinitely until one of the Xacts is aborted

Locking Performance

70

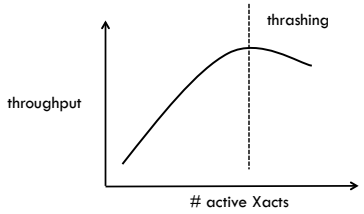
- Locking performance problems are *common!*
- The problem is too much blocking.
- The solution is to reduce the "locking load"
- Good heuristic – If more than 30% of transactions are blocked, then reduce the number of concurrent transactions

Credit: Phil Bernstein

Performance

71

- Locking overhead is primarily due to delays from blocking; minimizes throughout
- What happens to throughput as you increase the # of Xacts?



Improving Performance

72

- Lock the smallest sized object
  - Reduce likelihood that two Xacts need the same lock
- Reduce the time Xacts hold locks
- Reduce hot spots. A hot spot is an object that is frequently accessed, and causes blocking delays

Locking Granularity

73

- Granularity - size of data items to lock
  - e.g., files, pages, records, fields
- Coarse granularity implies
  - very few locks, so little locking overhead
  - must lock large chunks of data, so high chance of conflict, so concurrency may be low
- Fine granularity implies
  - many locks, so high locking overhead
  - locking conflict occurs only when two transactions try to access the exact same data concurrently

Deadlocks

74

- Deadlock: Cycle of transactions waiting for locks to be released by each other.
- Two ways of dealing with deadlocks:
  - Deadlock detection
  - Deadlock prevention

R. Ramakrishnan and J. Gehrke

Deadlocks

75

T1	T2
X(A)	
	X(B)
X(B)	
	X(A)

What happens?

Queued

Queued

T1 is waiting for T2 to release its lock
T2 is waiting for T1 to release its lock
→ Such a cycle of transactions is a *deadlock*

Implications:
•T1 and T2 will make no further progress
•They may hold locks needed by other Xacts
•DBMS tries to prevent or detect (and resolve) deadlocks

Deadlock Detection

76

- Detect deadlocks automatically, and abort a deadlocked transaction (the *victim*).
- Preferred approach, because it allows higher resource utilization
- Timeout-based deadlock detection - If a transaction is blocked for too long, then abort it.
  - Simple and easy to implement
  - But aborts unnecessarily (pessimistic) and
  - some deadlocks persist for too long

Deadlock Detection

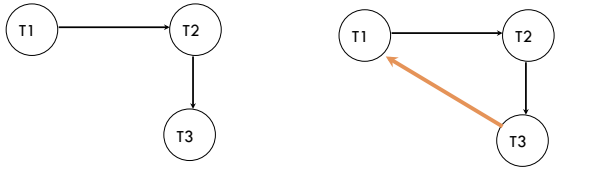
77

- Create a waits-for graph:
  - Nodes are transactions
  - There is an edge from Ti to Tj if Ti is waiting for Tj to release a lock
  - Lock mgr adds edge when lock request is queued
  - Remove edge when lock request granted
- A deadlock exists if there is a cycle in the waits-for graph
- Periodically check for cycles

Waits-For Graph: Example

78

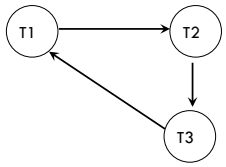
T1: S(A), R(A), S(B)
T2: X(B),W(B) X(C)
T3: S(C), R(C)



Example

79

T1: X(A), W(A), S(B)
T2: X(B),W(B) X(C)
T3: S(C), R(C), X(A)



Selecting a Victim

80

- Deadlock is resolved by aborting a Xact in the cycle, and releasing its locks
- Different criteria may be used:

a) Xact with the fewest/most locks
b) Xact that has done the least work
c) Xact that is farthest from completion
d) If a Xact is repeatedly restarted, increase its priority to allow it to complete

Commercial DBMS Approaches

81

- **MS SQL Server**: Aborts the transaction that is "cheapest" to roll back.
  - "Cheapest" is determined by the amount of log generated.
  - Allows transactions that you've invested a lot in, to complete.
- **Oracle:** The transaction that detects the deadlock is the victim.
- **DB2:** the deadlock detector arbitrarily selects one deadlocked process as the victim to roll back.