TRANSACTION PROCESSING TECHNIQUES

BY
SON NGUYEN
VIJAY RAO
Presentation Overview

_transactions

○ Definition
○ ACIDS Properties
○ Isolation levels
○ Concurrency & failures issues
○ Forms of transaction processing
○ Example: Airline Reservation System
○ Two types of transaction: Simple and Concurrent
○ Concurrency controls: Optimistic, Pessimistic and 2 Phase Locking

 Failures Recovery

○ Types of failures
○ Single site recovery: Abort recovery

 Distributed Transactions

○ Two phase commit

 Summary
What are Transactions?

- Can be defined as an indivisible unit of work comprised of several operations, all or none of which must be performed.
- Has ACID properties:
  - Atomicity
  - Consistency
  - Isolation
  - Durability
ACID Properties (cont)

► Atomicity
  Operations are treated as a single unit (indivisible) one which will either complete fully or not at all

► Consistency
  Must maintain consistency of data before and after a transaction

► Isolation
  Transactions should not affect other transactions

► Durability
  Requires result of a committed transaction be made permanent
Isolation levels of Transaction

- **ReadUncommitted/Dirty Read**
  Updated data not yet committed by a transaction may be read by other transactions

- **ReadCommitted**
  Only committed data can be read by other transactions

- **RepeatableRead**
  Dirty reads and nonrepeatable reads cannot occur

- **Serializable**
  Most restrictive isolation level. Exclusive access to read/write data
Why are Transactions needed?

To resolve:

► Concurrency issues
  ◇ Race condition
  ◇ Inconsistent updates

► Failures issues
  ◇ System failures
  ◇ Communication failures
Forms of transaction processing

► Sequential specification
   ◊ Transactions are executed in order
   ◊ Do not allow concurrency
   ◊ Allow failures

► Concurrent specification
   ◊ Allow transactions to happen concurrently
   ◊ Record the history and the result values of operations executed by each transaction
   ◊ Use transaction identifiers to distinguish concurrent transaction’s operations
   ◊ Allow failures
Applications that use transactions

► Banking systems
► Airline reservation system
► Online shopping
Airline Reservation example (cont)

► What happens if the reservation system tries to book the same seat concurrently?
  ◇ Two passengers might end up with the same seat (concurrency issue)

► Invalid booking also happens if:
  ◇ Reservation system reads data from one of its replicated database that has not been updated due to a network failure
Simple Transaction

- Turn AutoCommit off
- Execute a bunch of inserts and updates
- Commit
- If error then rollback

What happens if multiple threads are running?
Concurrent Transactions

- Need a robust transaction processing application server with services to manage all the issues with concurrent processing
- JTA (Java transaction API) provides the API for managing transactions
- Application servers such as BEA Weblogic and IBM Websphere provide a robust implementation for transaction management
Concurrency Control

- **Optimistic concurrency**
  - Operations are simply carried out
  - Synchronization takes place at the end of a transaction

- **Pessimistic concurrency**
  - Use Murphy’s law “If something can go wrong, it will”
  - Operations are synchronized before they are carried out

- **2 Phase Locking**
  - An algorithm to keep transactions in sync
  - Acquire/Release locks concept
  - Growing phase and Shrinking phase
  - Locks must be strong to ensure only one transaction can write at a time. Read is not allowed while writing.
Concurrency Control – 2 Phase Locking

Diagram:
- X-axis: Time
- Y-axis: Number of locks
- Two phases:
  - Growing phase
  - Shrinking phase
- Lock point

Legend:
- Growing phase: Increase in the number of locks over time.
- Shrinking phase: Decrease in the number of locks over time.
Concurrency Control – 2 Phase Locking (cont)

► Problems with 2 Phase locking

◇ Deadlocks

► Solutions?
Single Site Recovery

 ► Possible types of failures:
   ◦ Transaction abort
   ◦ System crash
   ◦ Media failures
   ◦ Catastrophe

 ► Possible ways of recovery
   ◦ Abort recovery
   ◦ Crash recovery
Single Site Recovery - Abort Recovery

Two methods to recover from transaction abort:

► Update-in-place

◇ Update the database as a transaction runs and undoing updates if aborts.

◇ Actions implementation:

□ Update: record old value in an undo log and update
□ Read: simply just read the desired data
□ Commit: discard transaction’s undo log
□ Abort: use undo log to back out updates
Deferred update

- Saving a transaction’s updates while it runs and using the saved updates to update the database when the transaction commits.

- Actions implementation:
  - Update: record updated value in an redo log
  - Read: combine the redo log and the database
  - Commit: update the database by applying redo log
  - Abort: discard the transaction’s redo log

Which method is better?
Single Site Recovery – Crash Recovery

- Use log to record updates, commits, and abort status
- Use log information to tell which updates belong on disk and which do not
- Abort transaction that doesn’t have a commit record in the log.
Distributed Transactions

Distributed Transactions include one or more statements that individually or as a group update data at two distinct nodes in a distributed environment.

Example of distributed transactions in Oracle

```sql
UPDATE scott.dept@hq.us.acme.com SET loc = 'REDWOOD SHORES' WHERE deptno = 10;
UPDATE scott.emp SET deptno = 11 WHERE deptno = 10;
UPDATE scott.bldg@maint.us.acme.com SET room = 1225 WHERE room = 1163;
COMMIT;
```
Distributed Transactions (Contd.)
Distributed Transactions – 2 Phase Commit

► A distributed transaction involves altering data across multiple databases.

► Databases can be geographically located like database 1 in SFO and 2 in NYC

► This protocol ensures integrity of data

► This consists of 2 phases managed by a global co-ordinator
  ▪ Prepare Phase
  ▪ Commit Phase
The global coordinator sends a message to check for preparedness to commit by all participants.

Once all participants have acknowledged it issues a Commit or rollback.
2 Phase Commit - Failure

- **Failure during Prepare Phase**
  - 1. All databases perform rollback
  - 2. Remote process prepares
  - 3. Crashes before giving prepare response
  - 4. Issues distributed transaction

- **Failure during Commit Phase**
  - 1. All databases commit and return success
  - 2. Remote process prepares
  - 3. Remote process receives prepare message from REMOTE
  - 4. Remote process commits, but remote crashed
  - 5. Issues distributed transaction
Real Life Examples

- Transactions in J2EE
  - Java Transaction API (JTA)
- Enterprise Java Beans
  - Robust and scalable component in J2EE architecture that can participate in transactions.
- Transaction Attributes for EJB
  - Declarative Transaction Management
    - Required, RequiresNew, Mandatory, Supports, NotSupported and Never
Nested Transactions

- Transactions within transactions
- Sub-transactions can commit but outer transaction could rollback...?
- Complex Issues!
- Not supported by J2EE and hence JDBC!
- Only flat transactions are supported
Summary

- Transactions are important
- Transactions in a distributed environment are much more complex compared to a single site transactions
- Must handle transactions appropriately
- Failed to recover a transaction might cause a lot of troubles
Bibliography

► http://www.mssqlcity.com/Articles/General/TIL.html
► http://www.cse.scu.edu/~jholliday/transproc.html
► http://www.ietf.org/rfc/rfc2371.txt
► http://www.cs.umb.edu/cs634/ora9idocs/server.920/a96521/ds_txns.htm
► http://www.cogs.susx.ac.uk/users/ianw/teach/dist-sys/ds-node246.html
► Distributed Systems: Principles and Paradigms by Tanenbaum and Van Steen
Q&A