Professional Development Program
Database Concepts

DATABASE ADMINISTRATION

Damira Pon
College of Computing and Information
University at Albany, SUNY
DATABASE ADMINISTRATION

Outline

- Transaction Processing
- Concurrency Problems
- Resource Locking
- Backup and Recovery
DATABASE ADMINISTRATION

Why?

- Programming with other applications
- Enforcement of referential integrity
- Many people using a system 24/7/365
- Information is important!
- CIA (Confidentiality Integrity Availability)
YOU HAVEN'T HEARD WHAT THE PROBLEM IS YET; HOW CAN YOU RECOMMEND BUILDING A DATABASE TO SOLVE IT??

WE ALWAYS BUILD A DATABASE.
AND WE'LL NEED COFFEE MUGS FOR THE PROJECT TEAM.

THE PROBLEM IS THAT WE HAVE POOR PROCESSES.

THAT COULD BE THE SLOGAN ON OUR MUGS!
DATABASE ADMINISTRATION

Transaction Processing - Intro

• What is a *transaction*?
  – Unit of interaction with a database system
  – Single logical operation on data (SQL)
  – Logical unit of work (LUW)

• Transaction Steps
  – Begin the transaction
  – Associated SQL queries execute
  – Commit the transaction
DATABASE ADMINISTRATION
Transaction Processing - ACID

- Atomicity
  - guarantees all parts of a transaction are complete

- Consistency
  - ensures that integrity constraints (rules) are maintained

- Isolation
  - transactions may not be seen in an intermediate state (by other operations)

- Durability
  - Once transaction has been verified by user (cannot be undone)
DATABASE ADMINISTRATION
Concurrency Problem 1

- Lost (Concurrent) Update
  - Row updates based on original value read

<table>
<thead>
<tr>
<th>Checking (C) Balance Before</th>
<th>ATM $100 Deposit Transaction</th>
<th>ATM $20 Withdrawal Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$100</td>
<td>Read C = 100</td>
<td>Read C = 100</td>
</tr>
<tr>
<td>T1</td>
<td></td>
<td>Write C 100–20 = 80</td>
</tr>
<tr>
<td>T2</td>
<td></td>
<td>Commit</td>
</tr>
<tr>
<td>T3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4 Write C 100+100 = 200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5 Commit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checking (C) Balance After</td>
<td>$200</td>
<td></td>
</tr>
<tr>
<td>$200</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DATABASE ADMINISTRATION
Concurrency Problem 2

- Uncommitted Dependency (Dirty Read)
  - Transaction uses value based on another non-committed transaction

<table>
<thead>
<tr>
<th>Checking (C) Balance Before</th>
<th>ATM $100 Deposit Transaction</th>
<th>ATM $20 Withdrawal Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$200</td>
<td>T1 Read C = 180</td>
<td>Read = 200</td>
</tr>
<tr>
<td></td>
<td>T2 Write 200 – 20 = 180</td>
<td>Write 200 – 20 = 180</td>
</tr>
<tr>
<td></td>
<td>T3 Commit</td>
<td>Commit</td>
</tr>
<tr>
<td>$280</td>
<td>T4 Write 180 + 100 = 280</td>
<td></td>
</tr>
</tbody>
</table>

Damira Pon, University at Albany, SUNY
**DATABASE ADMINISTRATION**

Concurrency Problem 3

- **Inconsistent Analysis (Non-Repeatable Read)**
  - Data read by a transaction is different during multiple times based on another transaction

<table>
<thead>
<tr>
<th>BEFORE</th>
<th>View S/C Account Balance Transaction</th>
<th>Transfer $100 from S to C Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checking (C)</td>
<td>Read C = 10</td>
<td>Read S = 500</td>
</tr>
<tr>
<td>$10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Savings (S)</td>
<td></td>
<td>Write 500-100 = 400</td>
</tr>
<tr>
<td>$500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| AFTER | | |
|-------| | |
| Checking (C) | | |
| $110 | | |
| Savings (S) | | |
| $400 | | |

T1: Read C = 10
T2: Read C = 10
T3: Read S = 400
T4: Write 10+100 = 110
T5: Commit
T6: Write 500-100 = 400
T7: Commit
Concurrency Problem 4

- **Phantom Read (Row)**
  - Transaction rereads data and finds rows deleted/inserted by different transaction from a previous read

<table>
<thead>
<tr>
<th>Deposit History (H) Before</th>
<th>ATM $1000 Deposit Transaction</th>
<th>Fraud Detection ≥$1000 Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/02/07  $100</td>
<td>T1 Read C = 200</td>
<td>T5 Commit</td>
</tr>
<tr>
<td>02/05/07  $20</td>
<td>T2 Write C 200+100 = 300</td>
<td>T4 Commit</td>
</tr>
<tr>
<td>Deposit History (H) After</td>
<td>T3 Write H 02/31/07 $1000</td>
<td>T5 Commit</td>
</tr>
<tr>
<td>01/02/07  $100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02/05/07  $20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02/31/07  $1000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What is resource locking?
- Prevents different transactions from obtaining copies of the same rows/table when being modified.

Locks can be implemented by:
- DBMS (Implicit Lock)
- Program/Application (Explicit Lock)
DATABASE ADMINISTRATION
Resource Locking - Lock Granularity

• Larger
  – administration easy
  – more conflicts

• Smaller
  – administration hard
  – less conflicts

DATABASE ADMINISTRATION
Resource Locking - Lock Granularity

• Granularity Options (most common)

Row/Key
Page
Table
Database
DATABASE ADMINISTRATION
Resource Locking - Lock Types

- **Exclusive**
  - No other transactions can read/change data

- **Shared**
  - Data cannot be changed, but can be read

- And others depending on specific DBMS
DATABASE ADMINISTRATION
Transaction Processing - Isolation Levels

1. **Read Uncommitted (Least Restrictive)**
   - Shared locks are not issued when reading data
   - Physically corrupt data is not read

2. **Read Committed**
   - Use shared locks when reading data
   - Will not allow reading of uncommitted data

3. **Repeatable Read**
   - Locks placed on all data used in query (other transactions cannot update data)

4. **Serializable (Most Restrictive)**
   - Prevents updates or appending of new rows until transaction is complete.
### Database Administration

#### Transaction Processing - Isolation Levels

<table>
<thead>
<tr>
<th>Problem</th>
<th>Isolation Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Read Uncommitted</td>
</tr>
<tr>
<td>Dirty Read</td>
<td>✔️</td>
</tr>
<tr>
<td>Nonrepeatable Read</td>
<td>✔️</td>
</tr>
<tr>
<td>Phantom Read</td>
<td>✔️</td>
</tr>
</tbody>
</table>

Upper Saddle River, NJ: Pearson Education Inc.
DATABASE ADMINISTRATION
Resource Locking - Locking Strategies

• “Overly Optimistic”
  - Assumes conflicts WILL NEVER occur
  - For single-user systems, read-only tables, or where records are guaranteed to only be accessed by one person at a time.

• Optimistic
  - Assumes conflict WILL GENERALLY NOT occur
  - Lock obtained after transaction processed
  - Better when lock granularity is large

• Pessimistic
  - Assumes conflict WILL GENERALLY occur
  - Lock obtained before transaction processing and released afterward
  - Better when lock granularity is small
<table>
<thead>
<tr>
<th>Type of Data</th>
<th>Examples</th>
<th>Suggested Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Live High-Volume</td>
<td>• Financial Accounts</td>
<td>1. Optimistic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Pessimistic</td>
</tr>
<tr>
<td>Live Low-Volume</td>
<td>• Personal Information</td>
<td>1. Pessimistic</td>
</tr>
<tr>
<td></td>
<td>• Insurance Policies</td>
<td>2. Optimistic</td>
</tr>
<tr>
<td>Log (Append only)</td>
<td>• Access Logs</td>
<td>Overly Optimistic</td>
</tr>
<tr>
<td></td>
<td>• Account Histories</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Transaction Records</td>
<td></td>
</tr>
<tr>
<td>Lookup/Reference (Read Only)</td>
<td>• State</td>
<td>Overly Optimistic</td>
</tr>
<tr>
<td></td>
<td>• Payment Type</td>
<td></td>
</tr>
</tbody>
</table>


Damira Pon, University at Albany, SUNY
DATABASE ADMINISTRATION
Concurrency Problem 5

- Deadlock “Deadly Embrace” Problem

- Four “Coffman” Conditions

1. Mutual Exclusion
   - Resource assigned to one process or available

2. Hold-and-Wait
   - Processes already holding resources may request new resources

3. No Preemption
   - Only process holding resource can release it

4. Circular Wait
   - Two or more processes in a chain wait for resources that next process in chain has a lock
DATABASE ADMINISTRATION
Concurrency Problem 5 - Deadlock

- Crayon Example
DATABASE ADMINISTRATION
Concurrency Problem 5 - Deadlock

• Prevention
  – Only issue one lock request at a time (all resources needed should be locked prior)
  – Issue locks in same order
  – Avoid user interaction in transactions
  – User lower isolation levels

• Detection
  – DBMS Monitoring Tools
DATABASE ADMINISTRATION
Concurrency Problem 5 - Deadlock

- To deal with deadlock situation, need to stop at least one transaction.

- “Victim Selection”
  - Priority
  - Amount of locks held
  - Run time length
  - Time of transaction start
Concurrency Problem 5 - Deadlock

- In instances of a distributed database, need *two-phase locking*.

- **Phase 1**
  - Each database will vote to commit or abort the transaction.

- **Phase 2**
  - Unanimous commit vote → Commit Transaction
  - Else → Rollback Transaction
DATABASE ADMINISTRATION
Backup and Recovery - Types of Failures

- System crashes
- Application errors
- Corruption of database
- Database (full/part) deletion
- Hardware failure
- Natural disasters, etc.
DATABASE ADMINISTRATION
Backup and Recovery - Transactions

• Reprocessing
  – Goes back to a known point and reprocesses workload after that point
  – Cons: Takes time and infeasible for high-volume systems & asynchronous

• Rollforward
  – Database restored using saved data
  – Valid transactions since save reapplied

• Rollback
  – Partially processed or bad transactions are undone
DATABASE ADMINISTRATION
Backup and Recovery - Log File

- Log File
  - Rollforward/Rollback use this function

- Contains:
  - Before-images
  - After-images
  - Time of actions
  - Operation types: begin, abort, commit, queries, database shutdown
  - Objected being acted upon (record type / identifier)

- When full should be saved on eternal storage media
DATABASE ADMINISTRATION
Backup and Recovery - Backups

• Backups
  – Should be on-site and on-site backups of important information
  – Should be routinely tested
  – May also be needed for audits/forensic investigations

• Generally, databases should be repaired ONLY when infeasible to restore it from a previous backup
DATABASE ADMINISTRATION
Backup and Recovery - Hybrid Strategy

- **Restore, Repair and Merge**
  - Used when have good backup but not all transaction logs created after backup

- **Steps**
  - Restore backup
  - Repair damaged copy of database (separately)
  - Merge data from repaired to restored database
DATABASE ADMINISTRATION

Summary

- Transaction Processing
- Concurrency Problems
- Resource Locking
- Backup and Recovery
DATABASE ADMINISTRATION

References


• The Database Journal, http://www.databasejournal.com