

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.



S. Adams

THE PROBLEM
IS THAT WE
HAVE POOR
PROCESSES.

THAT COULD
BE THE
SLOGAN ON
OUR MUGS!



4/4/94, © 1998 United Feature Syndicate, Inc. (NYC)

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

Checking (C)
Balance Before
\$100

Checking (C)
Balance After
\$200

	<u>ATM \$100 Deposit</u> <u>Transaction</u>	<u>ATM \$20 Withdrawal</u> <u>Transaction</u>
T1	Read C = 100	Read C = 100
T2		Write C $100 - 20 = 80$
T3		Commit
T4	Write C $100 + 100 = 200$	
T5	Commit	

DATABASE ADMINISTRATION

Concurrency Problem 2

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

Checking (C)
Balance Before
\$200

Checking (C)
Balance After
\$280

	<u>ATM \$100 Deposit</u> <u>Transaction</u>	<u>ATM \$20 Withdrawal</u> <u>Transaction</u>
T1		Read = 200
T2		Write 200 - 20 = 180
T3	Read C = 180	Commit
T4	Write 180 + 100 = 280	
T5	Commit	

DATABASE ADMINISTRATION

Concurrency Problem 3

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

BEFORE
<u>Checking (C)</u>
\$10
<u>Savings (S)</u>
\$500

AFTER
<u>Checking (C)</u>
\$110
<u>Savings (S)</u>
\$400

	<u>View S/C Account Balance Transaction</u>	<u>Transfer \$100 from S to C Transaction</u>
T1	Read C = 10	Read S = 500
T2		Write 500-100 = 400
T3		Read C = 10
T4		Write 10+100 = 110
T5		Commit
T6	Read S = 400	
T7	Commit	

DATABASE ADMINISTRATION

Concurrency Problem 4

- Phantom Read (Row)

- Transaction rereads data and finds rows deleted/inserted by different transaction from a previous read

<u>Deposit History</u>	
<u>(H) Before</u>	
01/02/07	\$100
02/05/07	\$20

<u>Deposit History</u>	
<u>(H) After</u>	
01/02/07	\$100
02/05/07	\$20
02/31/07	\$1000

	<u>ATM \$1000 Deposit Transaction</u>	<u>Fraud Detection \geq\$1000 Transaction</u>
T1	Read C = 200	
T2	Write C 200+100 = 300	Read H(amt) \geq \$1000
T3	Write H 02/31/07 \$1000	Write acct#
T4	Commit	Commit
T5		

DATABASE ADMINISTRATION

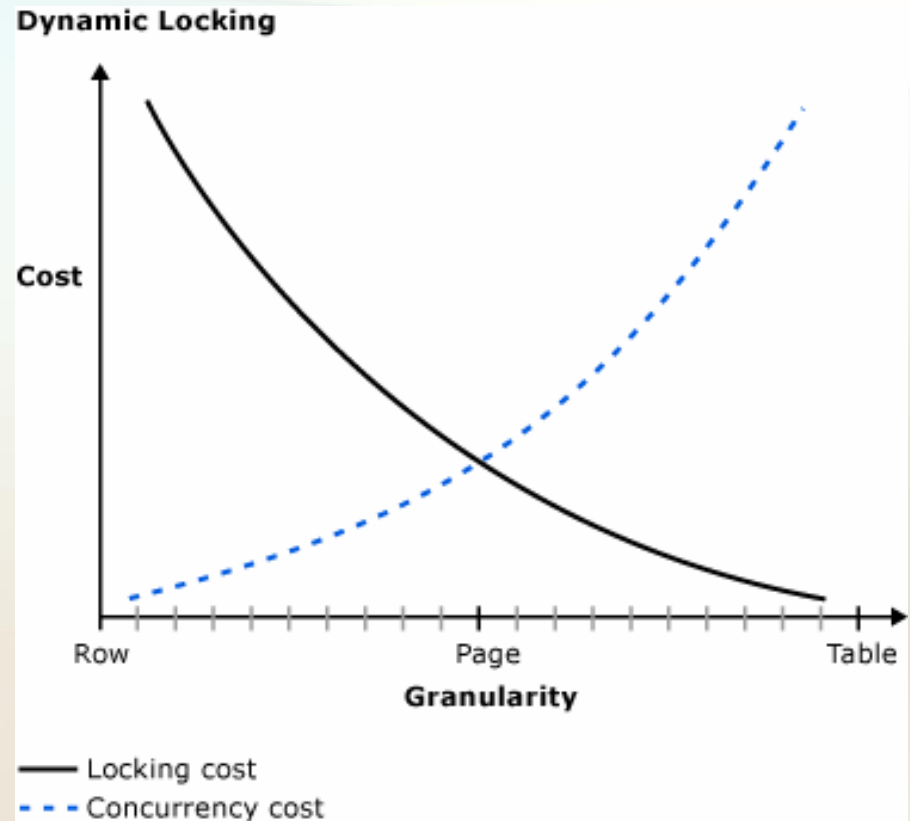
Resource Locking - Intro

- 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



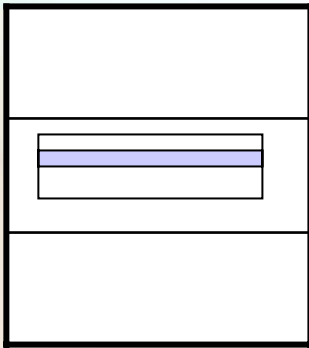
Source: <http://msdn2.microsoft.com/en-us/library/ms189286.aspx>

DATABASE ADMINISTRATION

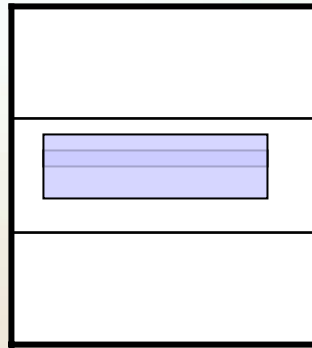
Resource Locking - Lock Granularity

- Granularity Options (most common)

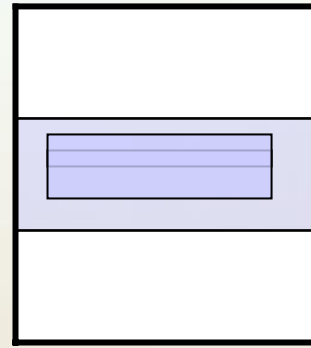
Row/Key



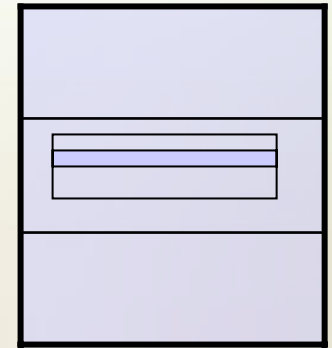
Page



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

		Isolation Level			
		Read Uncommitted	Read Committed	Repeatable Read	Serializable
Problem	Dirty Read	✓	✗	✗	✗
	Nonrepeatable Read	✓	✓	✗	✗
	Phantom Read	✓	✓	✓	✗

Source: Kroenke, D.M., & Auer, D.J. (2008). *Database Concepts*. 3rd ed. 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

DATABASE ADMINISTRATION

Resource Locking - Locking Strategies

Type of Data	Examples	Suggested Strategy
Live High-Volume	<ul style="list-style-type: none">• Financial Accounts	<ol style="list-style-type: none">1. Optimistic2. Pessimistic
Live Low-Volume	<ul style="list-style-type: none">• Personal Information• Insurance Policies	<ol style="list-style-type: none">1. Pessimistic2. Optimistic
Log (Append only)	<ul style="list-style-type: none">• Access Logs• Account Histories• Transaction Records	Overly Optimistic
Lookup/Reference (Read Only)	<ul style="list-style-type: none">• State• Payment Type	Overly Optimistic

Source: Ambler, S. (2003). Agile Database Techniques: Effective Strategies for the Agile Software Developer. Indianapolis, IN: Wiley Publishing, Inc.

DATABASE ADMINISTRATION

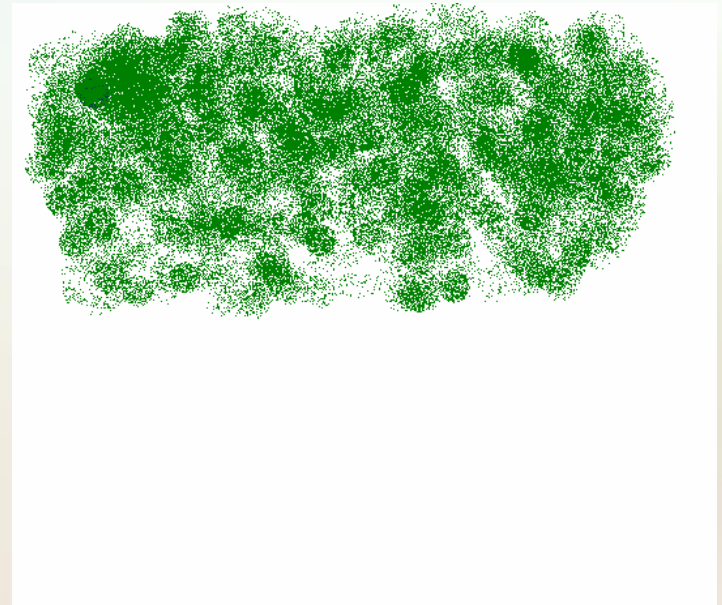
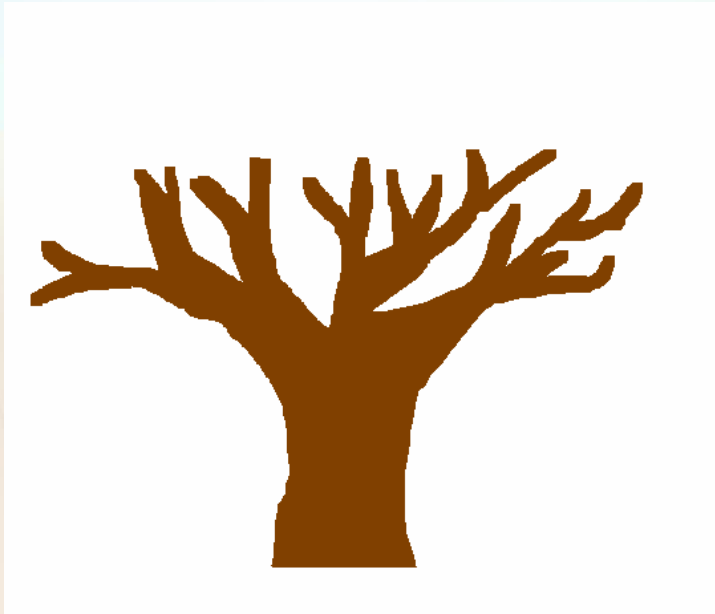
Concurrency Problem 5

- 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

DATABASE ADMINISTRATION

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

- Ambler, S. (2003). *Agile Database Techniques: Effective Strategies for the Agile Software Developer*. Indianapolis, IN: Wiley Publishing, Inc.
- Kroenke, D.M., & Auer, D.J. (2008). *Database Concepts*. 3rd ed. Upper Saddle River, NJ: Pearson Education Inc.
- The Database Journal, <http://www.databasejournal.com>
- ASA SQL User's Guide, http://www.ianywhere.com/developer/product_manuals/sqlanywhere/0901/en/html/dbugen9/dbugen9.htm