

Application Controls

CSH5 Chapter 52
 “Application Controls”
 Myles Walsh

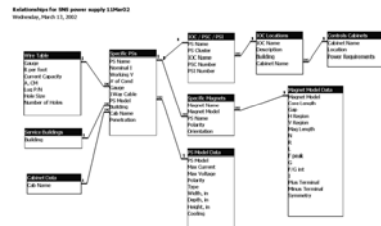
Topics

- Protection in Development
- Protecting Databases
- Protecting Batch Files
- Assuring that Information in the System is Valid



Protection in Development

- Software Quality Assurance (QA)
 - ❑ Focuses on methods for preventing, catching and correcting errors in source code and other operational instructions
- Application Controls
 - ❑ Specific subset of methods for preventing data corruption in production systems
- Databases
 - ❑ Primary mechanism for storing and manipulating data in today’s systems
- Batch vs Online
 - ❑ Live interaction (online)
 - ❑ Automated processing (batch)



Types of Data Corruption

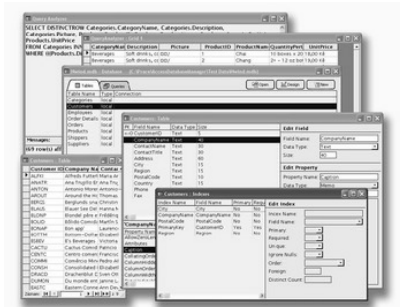
- Physical
 - ❑ Caused by hardware failures
 - ❑ Errors do not correlate with (respect)
 - ✓ Applications
 - ✓ Files / datasets / databases
 - ✓ Why not?
- Logical
 - ❑ Caused by software failures
 - ❑ Errors may correlate with
 - ✓ Applications
 - ✓ Files / datasets / databases
 - ❑ Why only “may”?



DBMS Controls



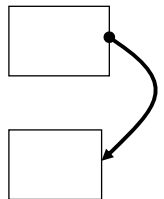
- Referential Integrity
- Uniqueness Constraints
- Locking
- Transaction Controls
- Database Recovery



Referential Integrity



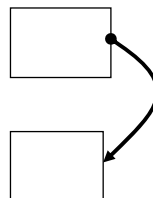
- In designing databases, may stipulate that certain records may not exist without pre-existing indexes
- E.g., cannot normally enter
 - Order-detail* without entering *order-header*
 - Prescription data* without entering
 - ✓ Patient data
 - ✓ Doctor data
 - ✓ Pharmacist data
 - ✓ Drug data
 - ✓ Admission data



Referential Integrity (cont'd)



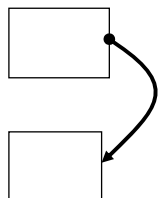
- DBMS will prevent deletion of records when others are *dependent* on them
- E.g.,
 - Cannot delete *order-header* if there are orders with the order-number of that header
 - Cannot delete *patient-master* record if there are admission records for that patient



DBMS Controls

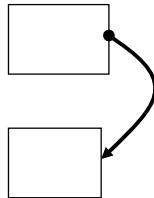


- Referential Integrity
- Uniqueness Constraints
- Locking
- Transaction Controls
- Database Recovery



Uniqueness Constraints

- In relational DBMS (RDBMS), every record in a table (dataset) must be unique
 - ❑ If there is no natural *key* or *index* field (or combination of fields) that guarantees uniqueness, can create one automatically
- DBMS can enforce uniqueness of specific fields in records using the *unique key* characteristic
- E.g.,
 - ❑ Order-number
 - ❑ Patient-ID
 - ❑ Student-ID....



9

Copyright © 2011 M. E. Kabay. All rights reserved.

DBMS Controls

- Referential Integrity
- Uniqueness Constraints
- Locking
- Transaction Controls
- Database Recovery



10

Copyright © 2011 M. E. Kabay. All rights reserved.

Locking

- Concurrency Control
- Basic Concepts of Locking
- Serializable Transactions
- Deadlock (Deadly Embrace)
- Locking Strategies



11

Copyright © 2011 M. E. Kabay. All rights reserved.

Concurrency Control

- Multi-Step Transactions
- Resource Locking
- Consistent Transactions
- Transaction Isolation Level
- Cursor Type



12

Copyright © 2011 M. E. Kabay. All rights reserved.

Multi-Step Transactions Are Fragile



- Think about order-entry system
- Create order-header
 - ❑ Includes *total* of cost of line-items (details)
 - ✓ Updated at END of detail data entry
- Begin entering line-items
 - ❑ Enter 3 records ... *have not yet finished*
 - ❑ System crashes
- What is the value in the order-header's *total* field?

13

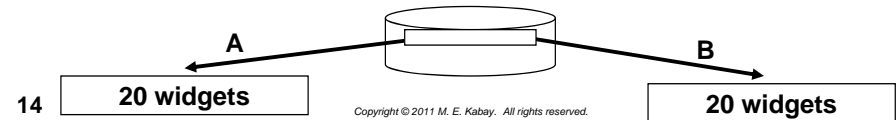
Copyright © 2011 M. E. Kabay. All rights reserved.

Concurrency Causes New Problems



E.g., The Lost Update Problem:

- User A reads inventory: finds 20 widgets.
- User B reads inventory: also finds 20 widgets.
- User A subtracts 10 widgets from 20, writes total ____ widgets back into inventory
- User B subtracts 5 widgets from ____, writes total ____ widgets back into inventory
- But actually, there are only ____ widgets left in the real inventory



14

Copyright © 2011 M. E. Kabay. All rights reserved.

Atomic Transactions



- We want to complete
 - ❑ All the steps of a transaction or
 - ❑ None of the steps
- ATOMIC
 - ❑ Greek “a” for “not” & “tomos” for “cut”
 - ❑ Thus “atomic” means “can not be cut.”
- We mark *atomic transactions* with boundaries
 - ❑ Start transaction
 - ❑ Commit transaction
- If necessary, can reverse steps taken
 - ❑ Rollback transaction



15

Copyright © 2011 M. E. Kabay. All rights reserved.

Resource Locking



- Basic Concepts of Locking
- Lock Terminology
- Serializable Transactions
- Deadlocks
- Optimistic vs Pessimistic Locking
- Declaring Lock Characteristics

16

Copyright © 2011 M. E. Kabay. All rights reserved.

Basic Concepts of Locking



- Locking is used in *inter-process communication* (IPC)
- A lock is a form of *semaphore* (signal)
- Locks allow processes to
 - ❑ Coordinate their access to resources
 - ❑ Prevent inconsistencies
- In DBMS, primarily used to *serialize* data access
 - ❑ One process gets control of data at a time

17

Copyright © 2011 M. E. Kabay. All rights reserved.

Lock Terminology



- Implicit vs explicit
 - ❑ Automatic locks placed by DBMS: implicit
 - ❑ Programmatically ordered: explicit
- Lock *granularity*
 - ❑ Large: database, dataset
 - ❑ Fine: records
- *Exclusive* vs *shared* locks
 - ❑ Exclusive:
 - ✓ One process READ/WRITE
 - ✓ No other processes allowed at all
 - ❑ Shared:
 - ✓ One process has R/W
 - ✓ Other processes can only READ

18

Copyright © 2011 M. E. Kabay. All rights reserved.

Conditional vs Unconditional Locking



- *Conditional* locking
 - ❑ Process 1 locks resource A
 - ❑ Process 2 locks resource A
 - ✓ Receives error condition
 - ✓ Lock fails and process 2 continues
- *Unconditional* locking
 - ❑ Process 1 locks resource A
 - ❑ Process 2 locks resource A
 - ✓ Does not receive a condition report
 - ✓ Process 2 waits in suspense (hangs) until lock is granted

19

Copyright © 2011 M. E. Kabay. All rights reserved.

Serializable Transactions

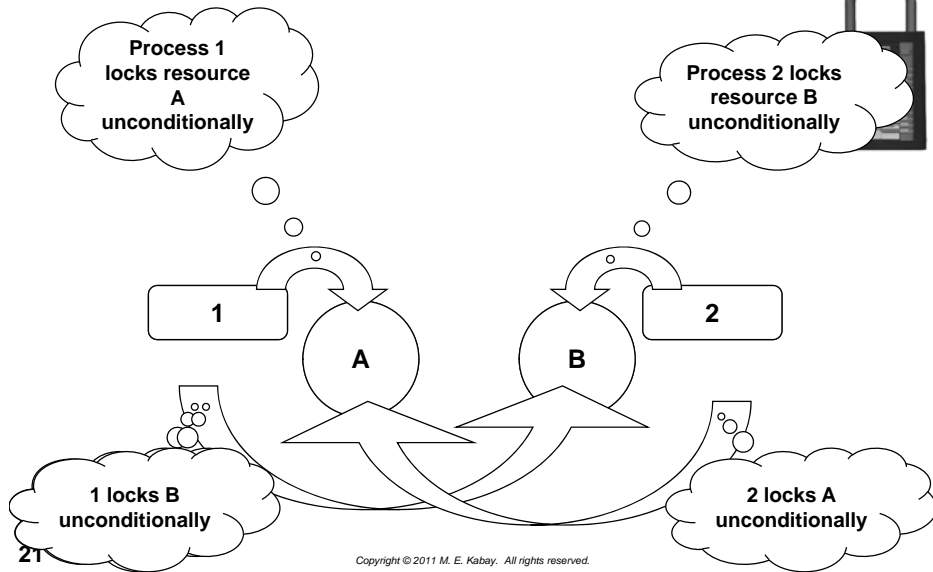


- Prevent transactions affecting same records from overlapping
- Two-phase locking
 - ❑ Can accumulate locks
 - ✓ But once any lock is released, cannot get more until all are released
 - ❑ Defines growing phase and shrinking phase
- More restrictive (and more common) strategy
 - ❑ No locks released until COMMIT or ROLLBACK

20

Copyright © 2011 M. E. Kabay. All rights reserved.

Deadlock (Deadly Embrace)



Preventing Deadlocks



- Deadlock is example of a *race condition*
 - ❑ Will not necessarily occur
 - ❑ Occurs by chance when specific events happen at specific time
- Always ensure that processes in applications
 - ❑ LOCK RESOURCES IN SAME ORDER
 - ❑ UNLOCK RESOURCES IN REVERSE ORDER
- Apply these principles to example on previous slide to see how they absolutely prevent deadlock

22

Copyright © 2011 M. E. Kabay. All rights reserved.

Pessimistic Locking Strategy



- Assume collisions will occur and prevent conflicts
 - ❑ Lock records
 - ❑ Process transaction
 - ❑ Release locks
- But very dangerous for performance if processing involves human interaction
 - ❑ Not controllable
 - ❑ Operator can leave resources locked and hang system
 - ✓ Operator could go to lunch!

DO NOT LOCK AROUND HUMAN INTERVENTION!

23

Copyright © 2011 M. E. Kabay. All rights reserved.

Optimistic Locking Strategy

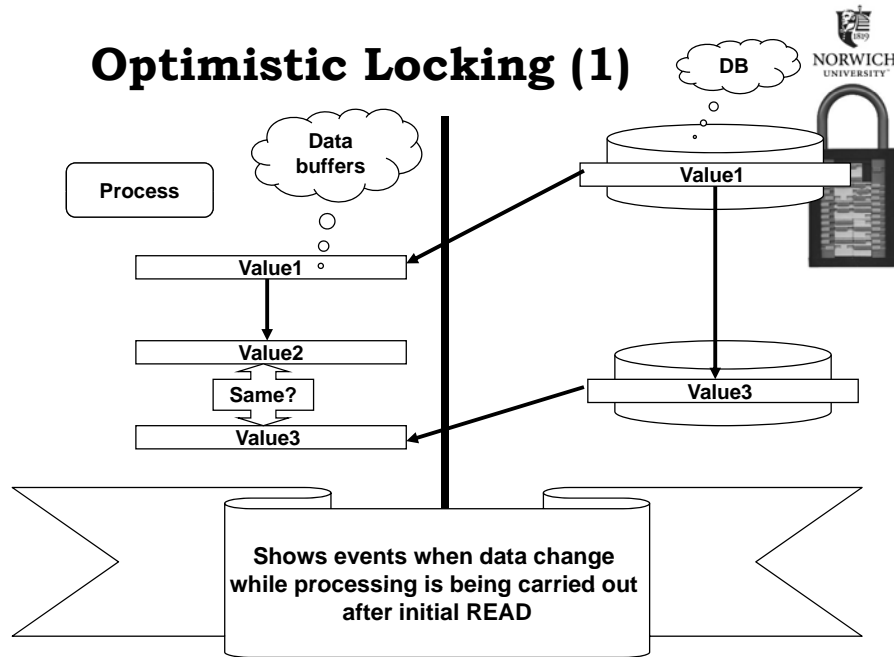


- Assume collisions will be rare and recover if they happen
 - ❑ Read original data records
 - ❑ Process transaction using buffers
 - ❑ Lock original data records
 - ❑ Check to see if original data have changed
 - ✓ If no change, commit transaction & unlock
 - ✓ If change, unlock & start over

24

Copyright © 2011 M. E. Kabay. All rights reserved.

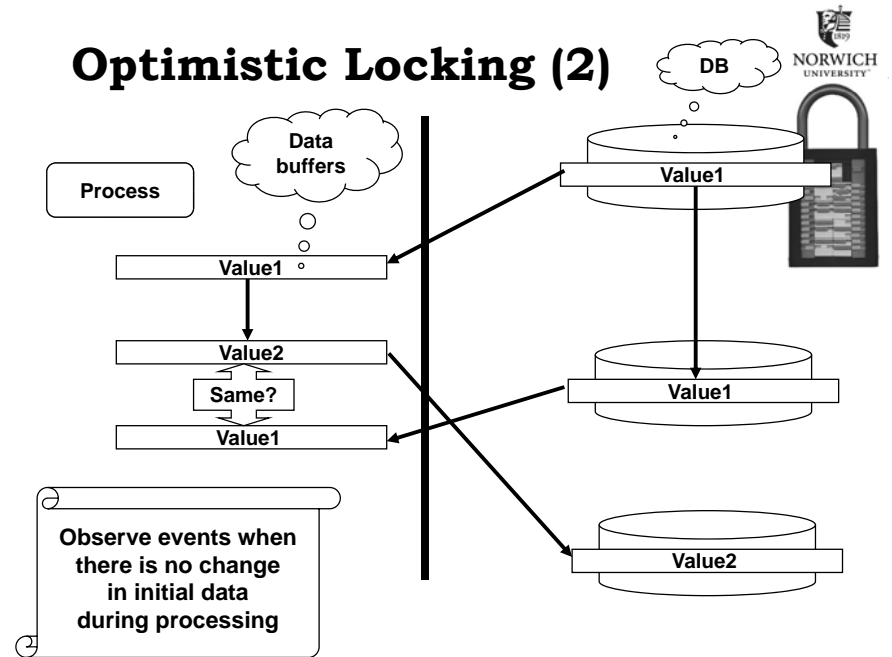
Optimistic Locking (1)



25

Copyright © 2011 M. E. Kabay. All rights reserved.

Optimistic Locking (2)



26

Copyright © 2011 M. E. Kabay. All rights reserved.

Optimistic vs Pessimistic Strategies

- Optimistic locking *advantages*
 - ❑ Does not lock resources around human intervention
 - ❑ Appropriate for Web / Internet transactions
 - ❑ Especially important if lock granularity is large (e.g., entire DB or entire tables)
- Optimistic locking *disadvantages*
 - ❑ If specific resource is in high demand (much *contention* for specific records) then can cause repeated access (*thrashing*)
 - ❑ Can degrade individual and system performance



27

Copyright © 2011 M. E. Kabay. All rights reserved.

Declaring Lock Characteristics

- Older programs often used specific calls to locking routines
 - ❑ E.g., "DBLOCK"
 - ❑ Passed parameters to set exact type of lock
 - ✓ Conditional or not, granularity etc.
- Modern programming using DBMS uses transaction markers
 - ❑ BEGIN, COMMIT, ROLLBACK
 - ❑ Allows global definition of locking strategy
 - ❑ DBMS handles details
 - ❑ Can thus change locking globally without reprogramming



28

Copyright © 2011 M. E. Kabay. All rights reserved.

DBMS Controls



- Referential Integrity
- Uniqueness Constraints
- Locking
- Transaction Controls
- Database Recovery

29

Copyright © 2011 M. E. Kabay. All rights reserved.

ACID Transactions



- Transactions sometimes described as ideally ACID
 - Atomic: all changes committed or none
 - Consistent: all records changed or none
 - Isolated: concurrency does not harm integrity
 - Durable: not reversible once committed

30

Copyright © 2011 M. E. Kabay. All rights reserved.

Consistency



- Statement-level consistency
 - If change is supposed to apply to group of records, then no changes to any of those records will be permitted until all records have been changed
- Transaction-level consistency
 - Same principle applied to multiple steps
 - Not always easy to achieve
 - If locking applied around very long processes, will see performance / throughput degradation for other users
 - May want to limit long updates to batch processing during off-hours

31

Copyright © 2011 M. E. Kabay. All rights reserved.

Transaction Isolation Level



- Can have difficulties / inconsistencies when concurrent processes access intermediate results during transactions
- *Dirty read*: access a record changed by another process but not yet committed
- *Nonrepeatable* read: some other process has altered the original record (e.g., during optimistic locking)
- *Phantom* read: a new movie by George Lucas – NO NO – means new records inserted or deleted since last read

32

Copyright © 2011 M. E. Kabay. All rights reserved.

ANSI SQL Isolation Levels



➤ Can specify degree of protection desired

ANSI SQL		Isolation Level			
		Read Uncommitted	Read Committed	Repeatable Read	Serializable
Problem Type	Dirty Read	Y	N	N	N
	Nonrepeatable Read	Y	Y	N	N
	Phantom Read	Y	Y	Y	N

DBMS Controls



- Referential Integrity
- Uniqueness Constraints
- Locking
- Transaction Controls
- Database Recovery

Database Recovery



- Transactions
- Application Logging
- Transactions and Log Files
- Backups & Log Files
- Recovery from Backups
- Recovery from Log Files

Transactions



- What are *transactions*?
- Why would we care if a transaction were interrupted by a DBMS failure or a system failure?

Application Logging



- Benefits of logging
 - Audit trail for security / investigations
 - Performance data
 - Debugging
- What might a logging process write into the log file when a process is
 - Adding a record?
 - Changing a record?
 - Deleting a record?

Transactions and Log Files



- Why would it matter to anyone that a log file keep a distinction among different transactions?
- How does a log file mark an atomic transaction?

Backups & Log Files



Distinguish among the following types of backups:

- System vs application
- Full (everything)
- Differential (aka "Partial") (everything changed since last full)
- Incremental (everything changed since last incremental) (aka "Partial")
- Delta (only changed data) (aka "Partial")
- Log files (only the *information about the changes*)

Backup Types



File	SUN	MON	TUE	WED	THU	FRI	SAT
A							
B							
C							
D							
E							

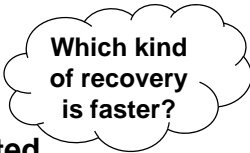
Backup Type	SUN	MON	TUE	WED	THU	FRI	SAT
FULL	ABCDE	ABCDE	ABCDE	ABCDE	ABCDE	ABCDE	ABCDE
DIFFERENTIAL		A	AB	ABD	ABCD	ABCDE	ABCDE
INCREMENTAL		A	B	AD	ABCD	CDE	ABC
DELTA (records)		A'	B'	A'D'	A'B'C'D'	C'D'E'	A'B'C'

Do not use the term "partial backup."

Recovery from Log Files



- Roll-backward recovery
 - Use log file to identify interrupted (incomplete) transactions using *checkpoints*
 - ✓ Look for *start marker* without *end marker*
 - Remove all changes that are part of those incomplete transactions
- Roll-forward recovery
 - Start with valid backup
 - Use log file to re-apply all completed transactions
 - Leave out the incomplete transactions



Assuring that Information in the System is Valid



- Validation Controls: catching data input errors
 - Check digits in input stream
 - Hash totals
 - Digital signatures
 - Range checks
 - Table lookups (including combinations)
- Diagnostic Utilities: catching data corruption or tampering
 - Edit checks
 - Business rules
 - Exception reports
 - Statistical Quality Control (SQC) methods (anomaly detection)



Review Questions (1)



1. Distinguish between SQA and application controls.
2. Why should we pay attention to applications when planning our security procedures?
3. Why are databases of such concern in application security discussions?
4. Name and distinguish between the two fundamental types of data corruption (by cause).
5. Explain the concept of *referential integrity* using examples.
6. How do DBMSs enforce uniqueness constraints?

Review Questions (2)



7. Why do concurrently-accessed databases require *locking strategies*?
8. What's a *transaction*?
9. What is meant by *atomic* transactions?
10. How is a transaction marked in a log file?
11. Which has finer granularity, locking an entire dataset or locking a set of records?
12. Distinguish between *exclusive* and *shared* locks.
13. Distinguish between *conditional* and *unconditional* locks.
14. What's a deadlock and how can you prevent it?

Review Questions (3)

15. Distinguish between *pessimistic* and *optimistic* locking strategies.
16. What does *ACID* mean in discussions of transactions? Explain each of the components.
17. Why do production applications normally include log files as part of their design?
18. Explain how *roll-backward* recovery works.
19. Explain how *roll-forward* recovery works.
20. Explain how applying each of the validation controls described in slide 42 could help check the validity of stored information in a database.
21. Explain how applying each of the diagnostic approaches described in slide 52 could help check the validity of stored information in a database.

Homework

- Study CSH5 Ch 52 using SQ3R
- **OPTIONAL** by 23:55 Sun 20 Mar 2011
 - For 1 extra point per question
 - Answer discussion questions online in NUoodle
 - Post ~250-500 word report on how an industry-standard RDBMS such as ORACLE, MySQL, Focus, or MS-ACCESS handles concurrency controls
 - ✓ Up to 0.02 pt / word depending on quality of contribution

DISCUSSION