

## Chapter 9 Managing Multi-User Databases



**DATABASE PROCESSING**  
Fundamentals, Design,  
and Implementation, 9/e

## Database Administration

- All large and small databases need database administration
- Data administration refers to a function concerning all of an organization's data assets
- Database administration (DBA) refers to a person or office specific to a single database and its applications

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 9/2

## DBA Tasks

- Managing database structure
- Controlling concurrent processing
- Managing processing rights and responsibilities
- Developing database security
- Providing for database recovery
- Managing the DBMS
- Maintaining the data repository

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 9/3

## Managing Database Structure

- DBA's tasks:
  - Participate in database and application development
    - Assist in requirements stage and data model creation
    - Play an active role in database design and creation
  - Facilitate changes to database structure
    - Seek community-wide solutions
    - Assess impact on all users
    - Provide configuration control forum
    - Be prepared for problems after changes are made
    - Maintain documentation

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 9/4

## Concurrency Control

- Concurrency control ensures that one user's work does not inappropriately influence another user's work
  - No single concurrency control technique is ideal for all circumstances
  - Trade-offs need to be made between level of protection and throughput

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 9/5

## Atomic Transactions

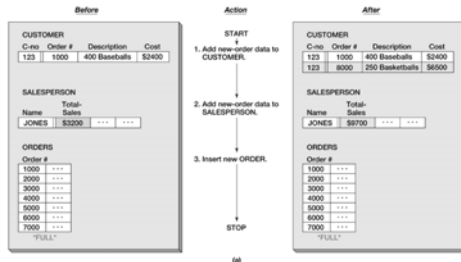
- A transaction, or logical unit of work (LUW), is a series of actions taken against the database that occurs as an atomic unit
  - Either all actions in a transaction occur or none of them do

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 9/6

## Example: Atomic Transaction

Figure 9.3a Need for Transaction Processing — Errors Introduced without Transaction

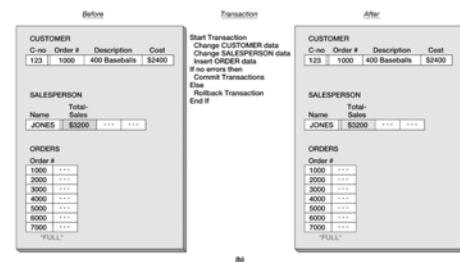


Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/7

## Example: Atomic Transaction

Figure 9.3b Need for Transaction Processing — Atomic Transaction Prevents Errors



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/8

## Concurrent Transaction

- Concurrent transactions refer to two or more transactions that appear to users as they are being processed against a database at the same time
- In reality, CPU can execute only one instruction at a time
  - Transactions are interleaved meaning that the operating system quickly switches CPU services among tasks so that some portion of each of them is carried out in a given interval
- Concurrency problems: lost update and inconsistent reads

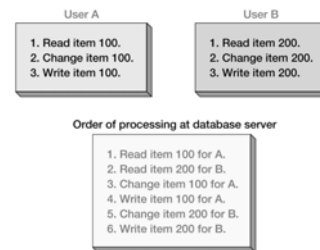
Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/9

## Example: Concurrent Transactions

No problems due to concurrency

Figure 9.4 Concurrent Processing Example

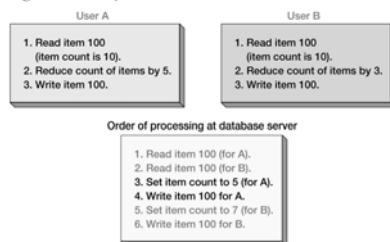


Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/10

## Example: Lost Update Problem

Figure 9.5 Lost Update Problem



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/11

## Resource Locking

- Resource locking prevents multiple applications from obtaining copies of the same record when the record is about to be changed

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/12

## Lock Terminology

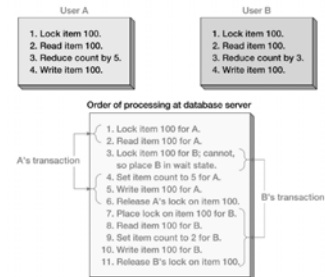
- Implicit locks are locks placed by the DBMS
- Explicit locks are issued by the application program
- Lock granularity refers to size of a locked resource
  - Rows, page, table, and database level
  - Large granularity is easy to manage but frequently causes conflicts
- Types of lock
  - An exclusive lock prohibits other users from reading the locked resource
  - A shared lock allows other users to read the locked resource, but they cannot update it

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/13

## Example: Explicit Locks

Figure 9.6 Concurrent Processing with Explicit Locks



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/14

## Serializable Transactions

- Serializable transactions refer to two transactions that run concurrently and generate results that are consistent with the results that would have occurred if they had run separately
- Two-phased locking is one of the techniques used to achieve serializability

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/15

## Two-phased Locking

- Two-phased locking
  - Transactions are allowed to obtain locks as necessary (growing phase)
  - Once the first lock is released (shrinking phase), no other lock can be obtained
- A special case of two-phased locking
  - Locks are obtained throughout the transaction
  - No lock is released until the COMMIT or ROLLBACK command is issued
  - This strategy is more restrictive but easier to implement than two-phase locking

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/16

## Deadlock

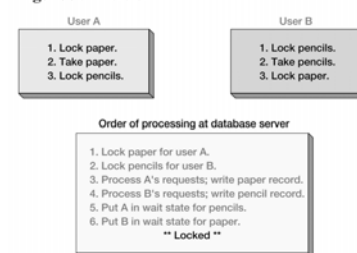
- Deadlock, or the deadly embrace, occurs when two transactions are each waiting on a resource that the other transaction holds
- Preventing deadlock
  - Allow users to issue all lock requests at one time
  - Require all application programs to lock resources in the same order
- Breaking deadlock
  - Almost every DBMS has algorithms for detecting deadlock
  - When deadlock occurs, DBMS aborts one of the transactions and rollbacks partially completed work

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/17

## Example: Deadlock

Figure 9.7 Deadlock



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/18

## Optimistic/Pessimistic Locking

- Optimistic locking assumes that no transaction conflict will occur
  - DBMS processes a transaction; checks whether conflict occurred
    - If not, the transaction is finished
    - If so, the transaction is repeated until there is no conflict
- Pessimistic locking assumes that conflict will occur
  - Locks are issued before transaction is processed, and then the locks are released
- Optimistic locking is preferred for the Internet and for many intranet applications (better performance)

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/19

## Example: Optimistic Locking

**Figure 9.8a** Optimistic and Pessimistic Locking — Optimistic Locking

```

SELECT  PRODUCTName, PRODUCT.Quantity
FROM    PRODUCT
WHERE   PRODUCTName = 'Pencil'

OldQuantity = PRODUCT.Quantity
Set NewQuantity = PRODUCT.Quantity - 5
(process transaction - take exception action if NewQuantity < 0, etc.
Assuming all is OK: )
Lock → LOCK PRODUCT

UPDATE  PRODUCT
SET     PRODUCT.Quantity = NewQuantity
WHERE   PRODUCTName = 'Pencil'
AND     PRODUCT.Quantity = OldQuantity

Unlock → UNLOCK PRODUCT
(check to see if update was successful;
if not, repeat transaction)
(a)
  
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/20

## Example: Pessimistic Locking

**Figure 9.8b** Optimistic and Pessimistic Locking — Pessimistic Locking

```

Lock → LOCK PRODUCT

SELECT  PRODUCTName, PRODUCT.Quantity
FROM    PRODUCT
WHERE   PRODUCTName = 'Pencil'

Set NewQuantity = PRODUCT.Quantity - 5
(process transaction - take exception action if NewQuantity < 0, etc.
Assuming all is OK: )

UPDATE  PRODUCT
SET     PRODUCT.Quantity = NewQuantity
WHERE   PRODUCTName = 'Pencil'

Unlock → UNLOCK PRODUCT
(no need to check if update was successful)
(b)
  
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/21

## Declaring Lock Characteristics

- Most application programs do not explicitly declare locks due to its complications (deadlocks, etc.)
- Instead, they mark transaction boundaries and declare locking behavior they want the DBMS to use
  - Transaction boundary markers: BEGIN, COMMIT, and ROLLBACK TRANSACTION
- Advantage
  - If the locking behavior needs to be changed, only the lock declaration need be changed, not the application program

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/22

## Example: Marking Transaction Boundaries

**Figure 9.9** Marking Transaction Boundaries

```

BEGIN TRANSACTION;

SELECT  PRODUCTName, PRODUCT.Quantity
FROM    PRODUCT
WHERE   PRODUCTName = 'Pencil'

OldQuantity = PRODUCT.Quantity
Set NewQuantity = PRODUCT.Quantity - 5
(process transaction - take exception action if NewQuantity < 0, etc.)

UPDATE  PRODUCT
SET     PRODUCT.Quantity = NewQuantity
WHERE   PRODUCTName = 'Pencil'
(continue processing transaction) ...

IF transaction has completed normally THEN
  COMMIT TRANSACTION
ELSE
  ROLLBACK TRANSACTION
END IF
(Continue processing other actions not part of this transaction ... )
  
```

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/23

## ACID Transactions

- Acronym ACID transaction is one that is Atomic, Consistent, Isolated, and Durable
- Atomic means either all or none of the database actions occur
- Durable means database committed changes are permanent

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/24

## ACID Transactions (cont.)

- Consistency means either statement level or transaction level consistency
  - Statement level consistency: each statement independently processes rows consistently
  - Transaction level consistency: all rows impacted by any of the SQL statements are protected from changes during the entire transaction
  - See following examples:

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/25

## Statement-level Consistency

- Consider the following action on 1M customer rows:
  - Update Customer  
Set AreaCode = '425'  
Where ZipCode = '98050';
- Should other transactions be allowed during this update?
  - No = statement-level consistency

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/26

## Transaction-level Consistency

- Two queries in the same transaction:
  - Update Customer Set AreaCode = '425'  
where ZipCode = '98050';
  - {other transaction work here}
  - Update Customer Set Discount = 0.05  
where AreaCode = '425';

Stmnt-level consistency would allow changes to AreaCode or Discount between updates above.  
Trans-level consistency does not allow this!

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/27

## ACID Transactions (cont.)

- Isolation means application programmers are able to declare the type of isolation level and to have the DBMS manage locks so as to achieve that level of isolation
- SQL-92 defines four transaction isolation levels:
  - Read uncommitted
  - Read committed
  - Repeatable read
  - Serializable

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/28

## Transaction Isolation Levels

- Dirty reads
  - Occur when one transaction reads a changed record that has not been committed to the database
- Non-repeatable reads
  - Occur when a transaction re-reads data and finds modifications or deletions caused by a committed transaction

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/29

## Transaction Isolation Levels

- Phantom reads
  - Occur when a transaction re-reads data and finds new rows that were inserted by a committed transaction since the prior read

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/30



## DBMS Security Guidelines (cont.)

- Manage accounts and passwords
  - Use a low privilege user account for the DBMS service
  - Protect database accounts with strong passwords
  - Monitor failed login attempts
  - Frequently check group and role memberships
  - Audit accounts with null passwords
  - Assign accounts the lowest privileges possible
  - Limit DBA account privileges
- Planning
  - Develop a security plan for preventing and detecting security problems
  - Create procedures for security emergencies and practice them

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 9/37

## Application Security

- If DBMS security features are inadequate, additional security code could be written in application program
  - Application security in Internet applications is often provided on the Web server computer
- However, you should use the DBMS security features first
  - The closer the security enforcement is to the data, the less chance there is for infiltration
  - DBMS security features are faster, cheaper, and probably result in higher quality results than developing your own

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 9/38

## SQL Injection Attack

- SQL injection attack occurs when data from the user is used to modify a SQL statement
- User input that can modify a SQL statement must be carefully edited to ensure that only valid input has been received and that no additional SQL syntax has been entered
- Example: users are asked to enter their names into a Web form textbox
  - User input: Benjamin Franklin ' OR TRUE '  
SELECT \* FROM EMPLOYEE  
WHERE EMPLOYEE.Name = 'Benjamin Franklin' OR TRUE;
  - Result: every row of the EMPLOYEE table will be returned

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 9/39

## Database Recovery

- In the event of system failure, that database must be restored to a usable state as soon as possible
- Two recovery techniques:
  - Recovery via reprocessing
  - Recovery via rollback/rollforward

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 9/40

## Recovery via Reprocessing

- Recovery via reprocessing: the database goes back to a known point (database save) and reprocesses the workload from there
- Unfeasible strategy because
  - The recovered system may never catch up if the computer is heavily scheduled
  - Asynchronous events, although concurrent transactions, may cause different results

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 9/41

## Rollback/Rollforward

- Recovery via rollback/rollforward:
  - Periodically save the database and keep a database change log since the save
    - Database log contains records of the data changes in chronological order
- When there is a failure, either rollback or rollforward is applied
  - Rollback: undo the erroneous changes made to the database and reprocess valid transactions
  - Rollforward: restored database using saved data and valid transactions since the last save

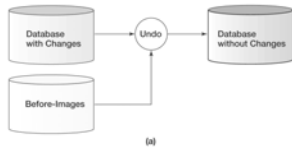
Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 9/42

## Example: Rollback

- Before-images: a copy of every database record (or page) before it was changed

Figure 9.15a Undo and Redo/Transactions — Rollback



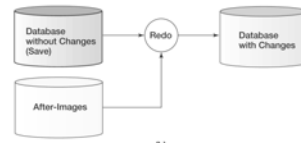
Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/43

## Example: Rollforward

- After-images: a copy of every database record (or page) after it was changed

Figure 9.15b Undo and Redo/Transactions — Rollforward



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/44

## Example: Transaction Log

Figure 9.16 Example Transaction Log

Relative Record Number	Transaction ID	Reverse Pointer	Forward Pointer	Time	Type of Operation	Object	Before-Image	After-Image
1	OT1	0	2	11:42	START			
2	OT1	1	4	11:43	MODIFY	CUST 100	(old value)	(new value)
3	OT2	0	8	11:46	START			
4	OT1	2	5	11:47	MODIFY	SP AA	(old value)	(new value)
5	OT1	4	7	11:47	INSERT	ORDER 11		(value)
6	CT1	0	9	11:48	START			
7	OT1	5	0	11:49	COMMIT			
8	OT2	3	0	11:50	COMMIT			
9	CT1	6	10	11:51	MODIFY	SP BB	(old value)	(new value)
10	CT1	9	0	11:51	COMMIT			

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/45

## Example: Database Recovery

Figure 9.17a Recovery Example — Processing with Problem

```

Accept order data from browser.
Read CUSTOMER and SALESPERSON records.
Change CUSTOMER and SALESPERSON records.
Rewrite CUSTOMER record.
Rewrite SALESPERSON record.
Insert new ORDER record.
    (Log records written here)
****CRASH****
    
```

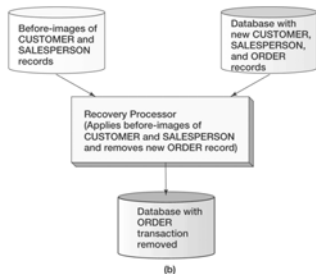
(a)

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 9/46

## Example: Database Recovery

Figure 9.17b Recovery Example — Recovery Processing



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

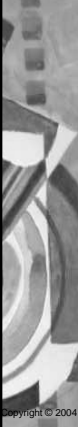
Chapter 9/47

## Checkpoint

- A checkpoint is a point of synchronization between the database and the transaction log
  - DBMS refuses new requests, finishes processing outstanding requests, and writes its buffers to disk
  - The DBMS waits until the writing is successfully completed → the log and the database are synchronized
- Checkpoints speed up database recovery process
  - Database can be recovered using after-images since the last checkpoint
  - Checkpoint can be done several times per hour
- Most DBMS products automatically checkpoint themselves

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

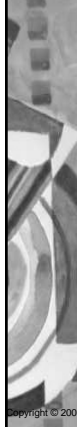
Chapter 9/48



## Managing the DBMS

- DBA's Responsibilities
  - Generate database application performance reports
  - Investigate user performance complaints
  - Assess need for changes in database structure or application design
  - Modify database structure
  - Evaluate and implement new DBMS features
  - Tune the DBMS

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/49




## Maintaining the Data Repository

- DBA is responsible for maintaining the data repository
- Data repositories are collections of metadata about users, databases, and its applications
- The repository may be
  - Virtual as it is composed of metadata from many different sources: DBMS, code libraries, Web page generation and editing tools, etc.
  - An integrated product from a CASE tool vendor or from other companies
- The best repositories are active and they are part of the system development process

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 9/50

## Chapter 9 Managing Multi-User Databases

---



**DATABASE PROCESSING**  
Fundamentals, Design,  
and Implementation, 9/e