Colorado Technical University

CS 660 – Database Systems

Colorado Technical University
Instructor: Dr. John Conklin

Unit 4 – Database Administration

Security and Administration Transparencies

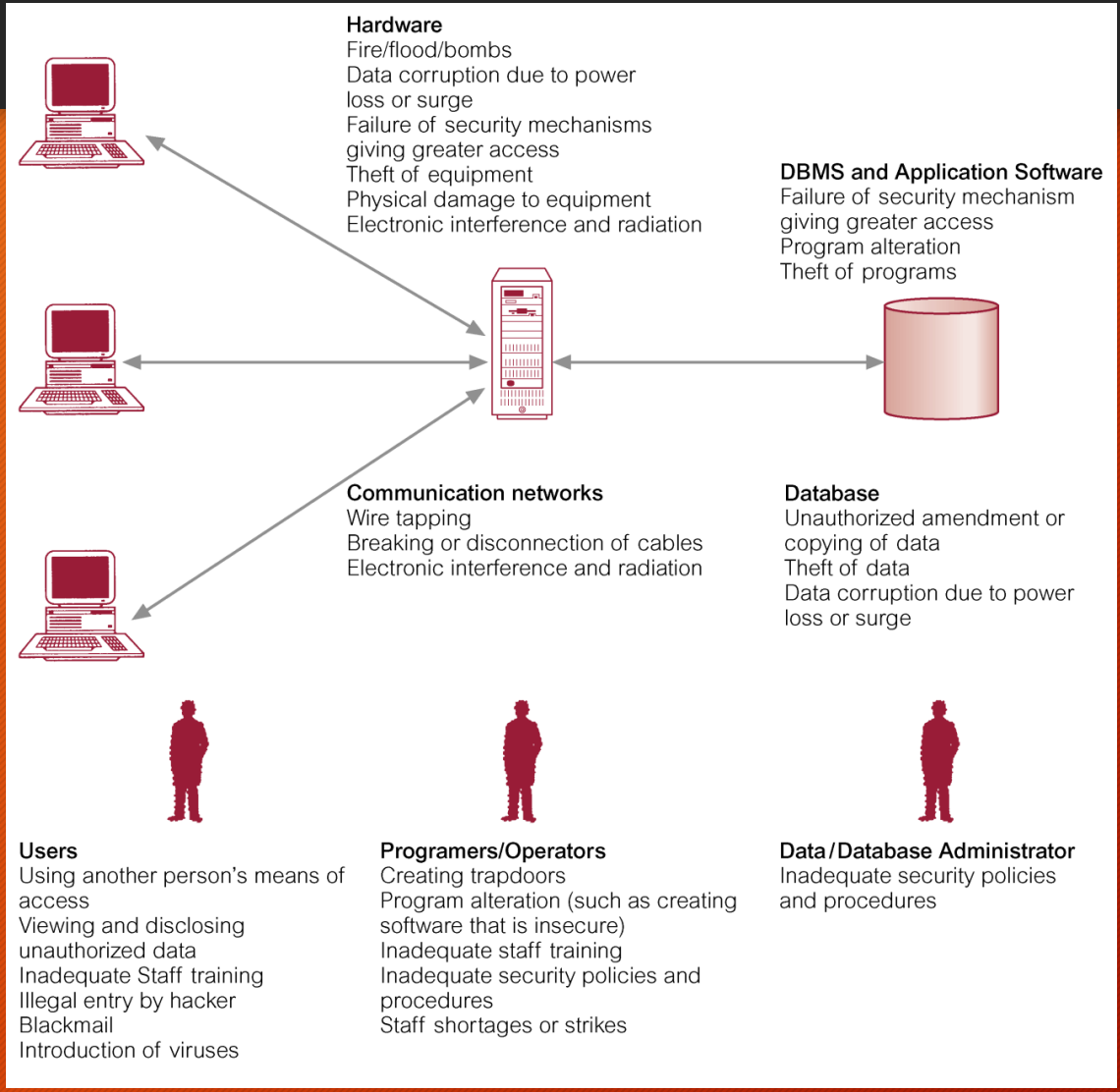
Database Security

- Data is a valuable resource that must be strictly controlled and managed, as with any corporate resource.
- Part or all of the corporate data may have strategic importance and therefore needs to be kept secure and confidential.
- Mechanisms that protect the database against intentional or accidental threats.
- Security considerations do not only apply to the data held in a database. Breaches of security may affect other parts of the system, which may in turn affect the database.

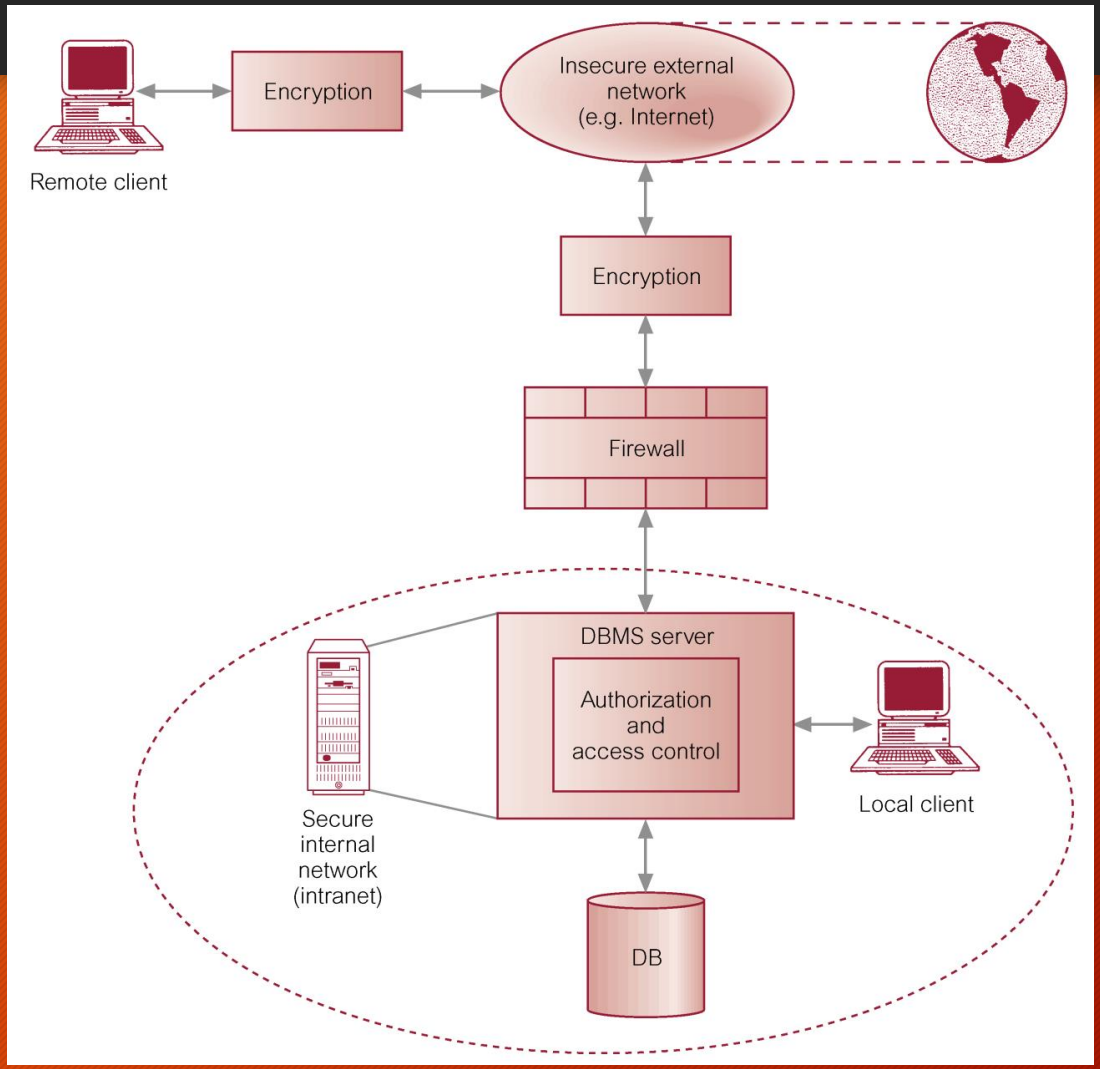
Database Security (cont.)

- Involves measures to avoid:
  - Theft and fraud
  - Loss of confidentiality (secrecy)
  - Loss of privacy
  - Loss of integrity
  - Loss of availability
- Threat
  - Any situation or event, whether intentional or unintentional, that will adversely affect a system and consequently an organization.

Summary of Threats to Computer Systems



Typical Multi-user Computer Environment



Countermeasures - Computer-Based Controls

- Concerned with physical controls to administrative procedures and includes:
  - Authorization
  - Access controls
  - Views
  - Backup and recovery
  - Integrity
  - Encryption
  - RAID technology

We will discuss each on the upcoming slides.

Countermeasures – Computer-Based Controls (cont.)

- Authorization
  - The granting of a right or privilege, which enables a subject to legitimately have access to a system or a system's object.
  - Authorization is a mechanism that determines whether a user is, who he or she claims to be.
- Access control
  - Based on the granting and revoking of privileges.
  - A privilege allows a user to create or access (that is read, write, or modify) some database object (such as a relation, view, and index) or to run certain DBMS utilities.
  - Privileges are granted to users to accomplish the tasks required for their jobs.

Countermeasures – Computer-Based Controls (cont.)

- Most DBMS provide an approach called Discretionary Access Control (DAC).
- SQL standard supports DAC through the GRANT and REVOKE commands.
- The GRANT command gives privileges to users, and the REVOKE command takes away privileges.

Countermeasures – Computer-Based Controls (cont.)

- DAC while effective has certain weaknesses. In particular an unauthorized user can trick an authorized user into disclosing sensitive data.
- An additional approach is required called Mandatory Access Control (MAC).
- DAC based on system-wide policies that cannot be changed by individual users.
- Each database object is assigned a *security class* and each user is assigned a *clearance* for a security class, and *rules* are imposed on reading and writing of database objects by users.

Countermeasures – Computer-Based Controls (cont.)

- DAC determines whether a user can read or write an object based on rules that involve the security level of the object and the clearance of the user. These rules ensure that sensitive data can never be 'passed on' to another user without the necessary clearance.
- The SQL standard does *not* include support for MAC.

Countermeasures – Computer-Based Controls (cont.)

- View
  - Is the dynamic result of one or more relational operations operating on the base relations to produce another relation.
  - A view is a virtual relation that does not actually exist in the database, but is produced upon request by a particular user, at the time of request.
- Backup
  - Process of periodically taking a copy of the database and log file (and possibly programs) to offline storage media.
- Journaling
  - Process of keeping and maintaining a log file (or journal) of all changes made to database to enable effective recovery in event of failure.

Countermeasures – Computer-Based Controls (cont.)

- Integrity
  - Prevents data from becoming invalid, and hence giving misleading or incorrect results.
- Encryption
  - The encoding of the data by a special algorithm that renders the data unreadable by any program without the decryption key.

RAID (Redundant Array of Independent Disks) Technology

- Hardware that the DBMS is running on must be *fault-tolerant*, meaning that the DBMS should continue to operate even if one of the hardware components fails.
- Suggests having redundant components that can be seamlessly integrated into the working system whenever there is one or more component failures.
- The main hardware components that should be fault-tolerant include disk drives, disk controllers, CPU, power supplies, and cooling fans.
- Disk drives are the most vulnerable components with the shortest times between failure of any of the hardware components.

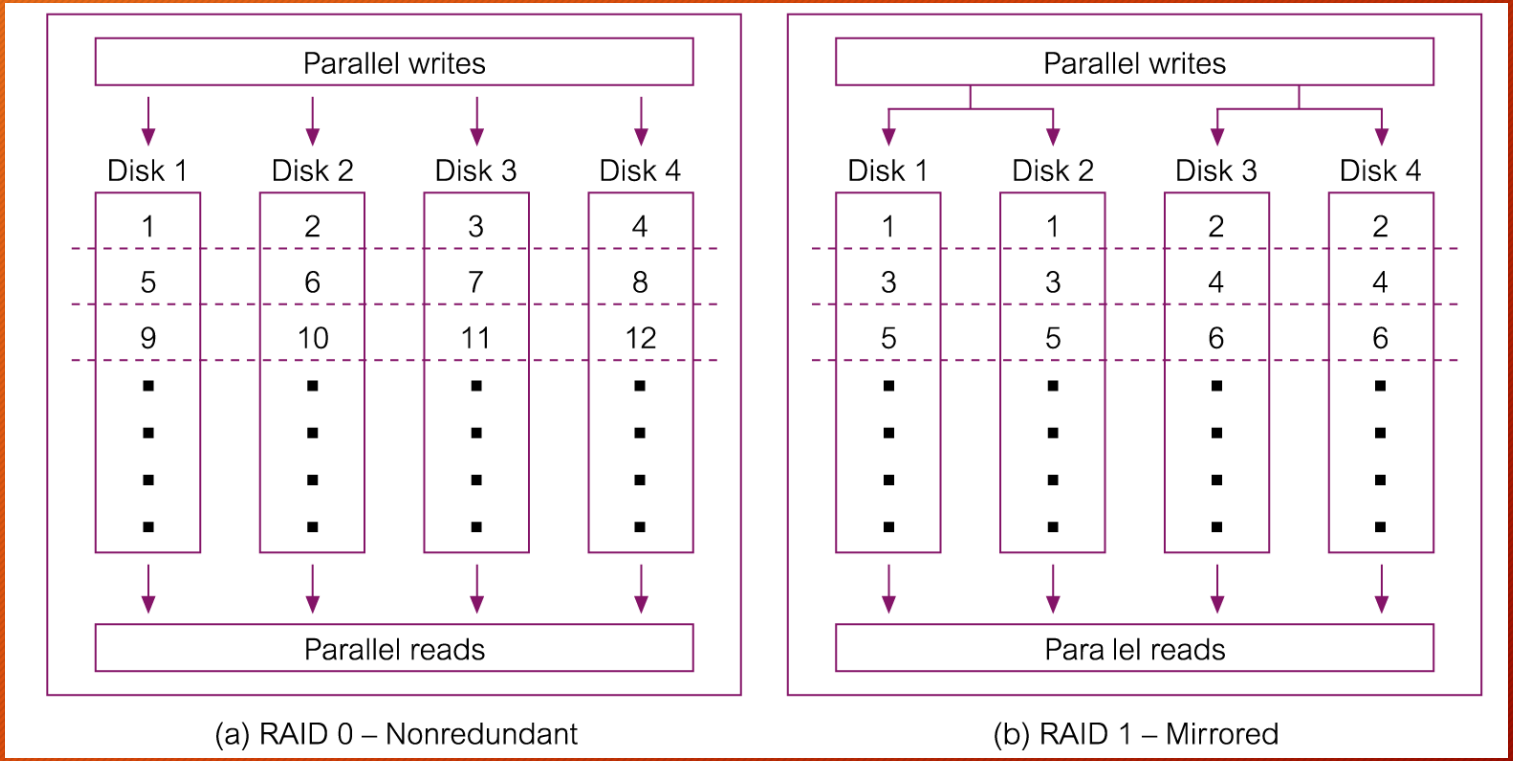
RAID (Redundant Array of Independent Disks) Technology (cont.)

- One solution is to provide a large disk array comprising an arrangement of several independent disks that are organized to improve reliability and at the same time increase performance.
- Performance is increased through *data striping*: the data is segmented into equal-size partitions (the *striping unit*), which are transparently distributed across multiple disks.
- Reliability is improved through storing redundant information across the disks using a *parity* scheme or an *error-correcting* scheme.

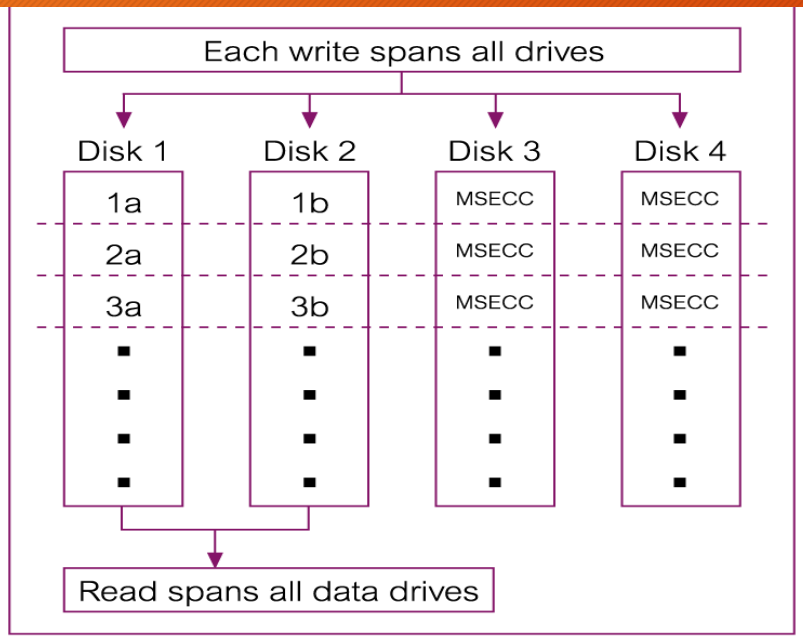
RAID (Redundant Array of Independent Disks) Technology (cont.)

- There are several different disk configurations called RAID levels.
  - RAID 0 Nonredundant
  - RAID 1 Mirrored
  - RAID 0+1 Nonredundant and Mirrored
  - RAID 2 Memory-Style Error-Correcting Codes
  - RAID 3 Bit-Interleaved Parity
  - RAID 4 Block-Interleaved Parity
  - RAID 5 Block-Interleaved Distributed Parity
  - RAID 6 P+Q Redundancy

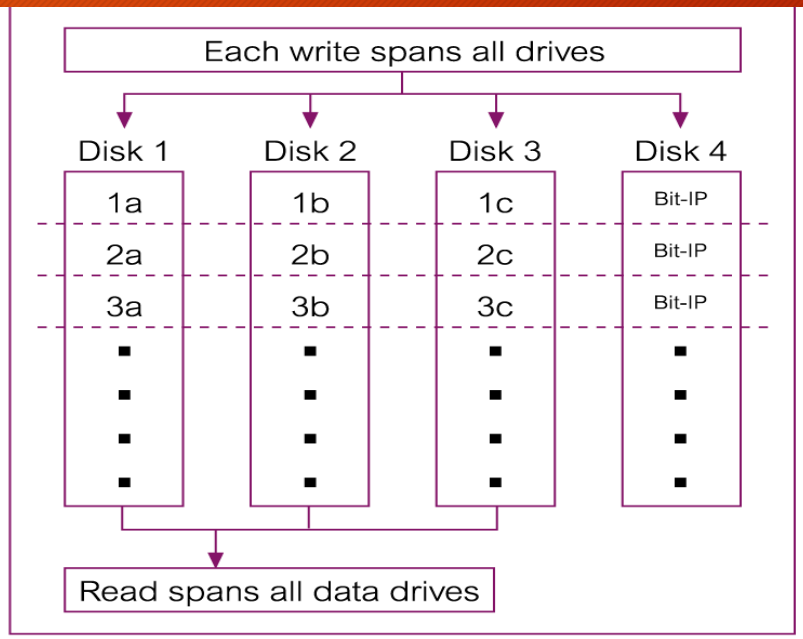
RAID 0 and RAID 1



RAID 2 and RAID 3

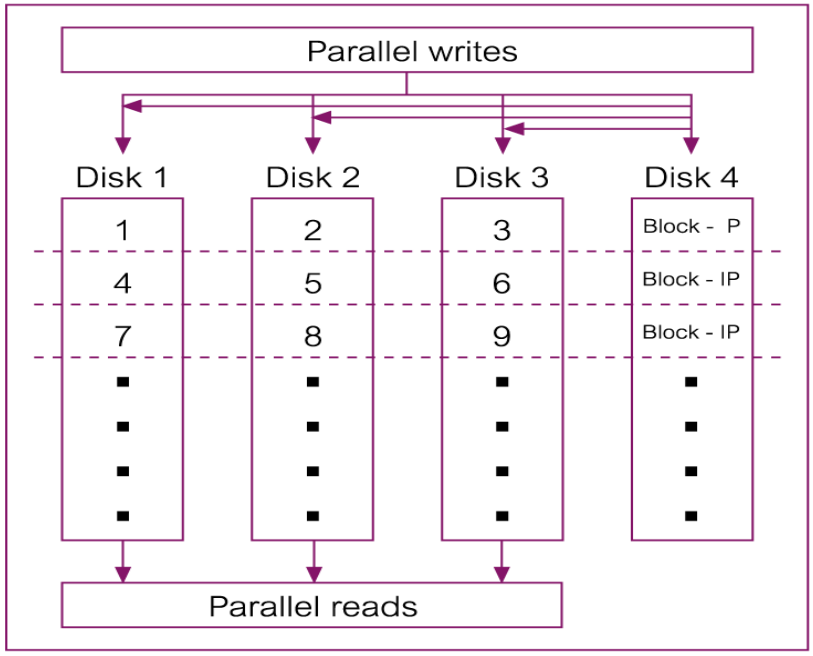


(c) RAID 2 – Memory-Style Error-Correct ng Codes (MSECC)

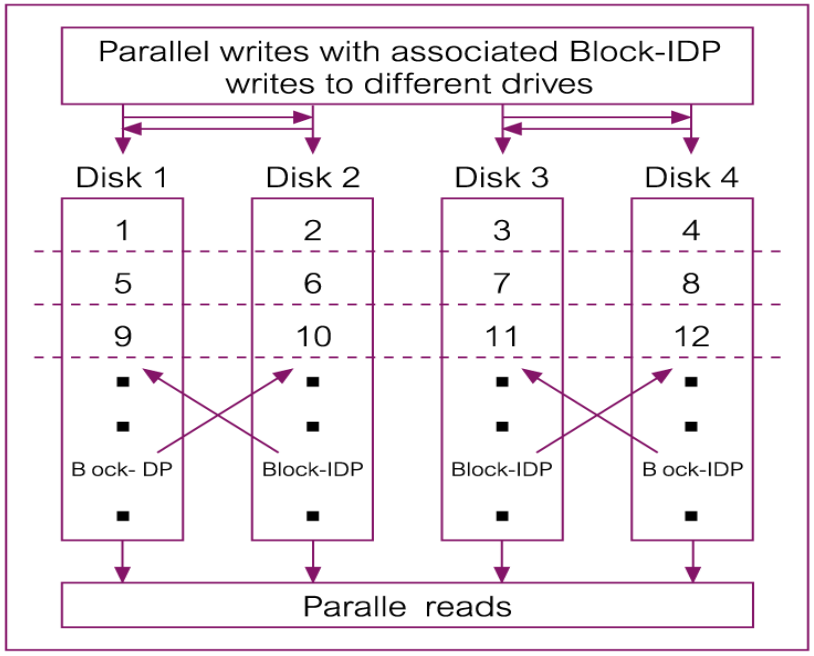


(d) RAID 3 – Bit Interleaved Parity (Bit-IP)

RAID 4 and RAID 5



(e) RAID 4 – Block-Interleaved Parity (Block-IP)



(f) RAID 5 – Block-Interleaved D str buted Parity (Block-IDP)

DBMSs and Web Security

- Internet communication relies on TCP/IP as the underlying protocol. However, TCP/IP and HTTP were not designed with security in mind. Without special software, all Internet traffic travels ‘in the clear’ and anyone who monitors traffic can read it.
- Must ensure while transmitting information over the Internet that:
  - inaccessible to anyone but sender and receiver (privacy);
  - not changed during transmission (integrity);
  - receiver can be sure it came from sender (authenticity);
  - sender can be sure receiver is genuine (non-fabrication);
  - sender cannot deny he or she sent it (non-repudiation).

DBMSs and Web Security (cont.)

- Measures include:
  - Proxy servers
  - Firewalls
  - Message digest algorithms and digital signatures
  - Digital certificates
  - Kerberos
  - Secure sockets layer (SSL) and Secure HTTP (S-HTTP)
  - Secure Electronic Transactions (SET) and Secure Transaction Technology (SST)
  - Java security
  - ActiveX security

Secure Electronic Transactions (SET)

- An open, interoperable standard for processing credit card transactions over the internet.
- Was created jointly by Netscape, Microsoft, Visa, Mastercard, GET, SAIC, Terisa Systems and Verisign.
- Split in a way that the merchant has access to information regarding items purchased but not what payment information the users is using.

Transaction Management

Transaction Support

Transaction

Action, or series of actions, carried out by user or application, which reads or updates contents of database.

- Logical unit of work on the database.
- Application program is series of transactions with non-database processing in between.
- Transforms database from one consistent state to another, although consistency may be violated during transaction.

Example Transaction


```
read(staffNo = x, salary)
salary = salary * 1.1
write(staffNo = x, new_salary)
```

(a)

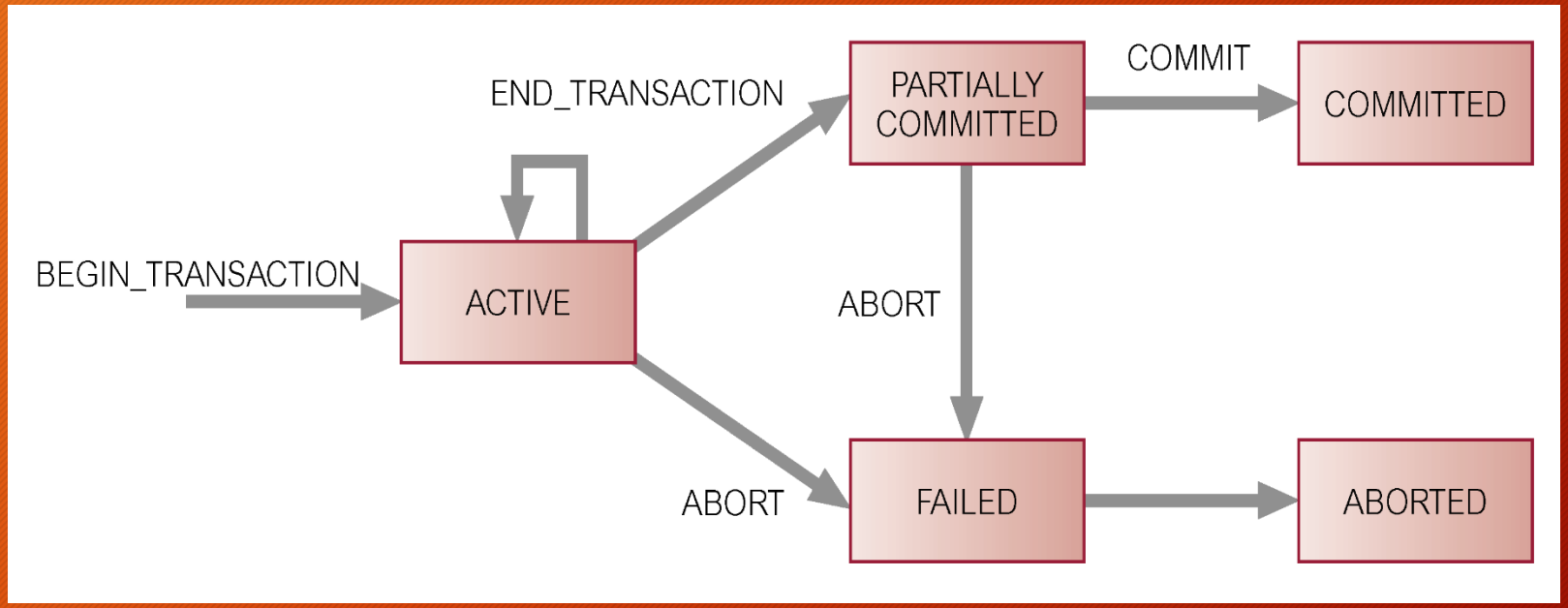
```
delete(staffNo = x)
for all PropertyForRent records, pno
begin
    read(propertyNo = pno, staffNo)
    if (staffNo = x) then
    begin
        staffNo = newStaffNo
        write(propertyNo = pno, staffNo)
    end
end
```

(b)

Transaction Support

- Can have one of two outcomes:
  - Success - transaction *commits* and database reaches a new consistent state.
  - Failure - transaction *aborts*, and database must be restored to consistent state before it started.
  - Such a transaction is *rolled back* or *undone*.
- Committed transaction cannot be aborted.
- Aborted transaction that is rolled back can be restarted later.

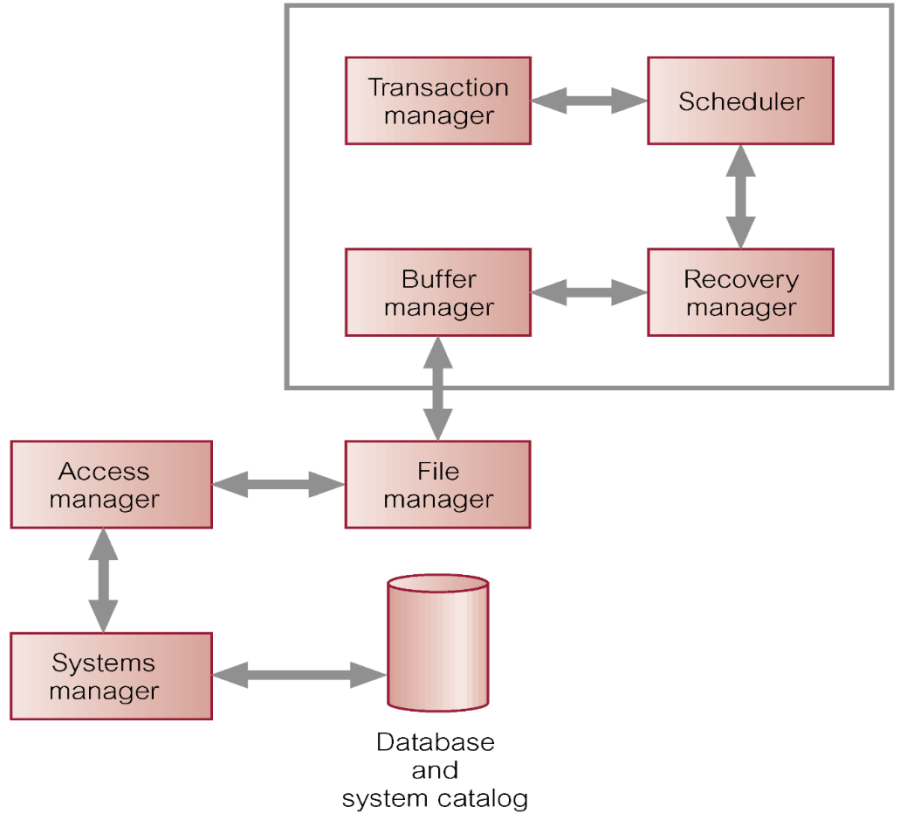
State Transition Diagram for Transaction



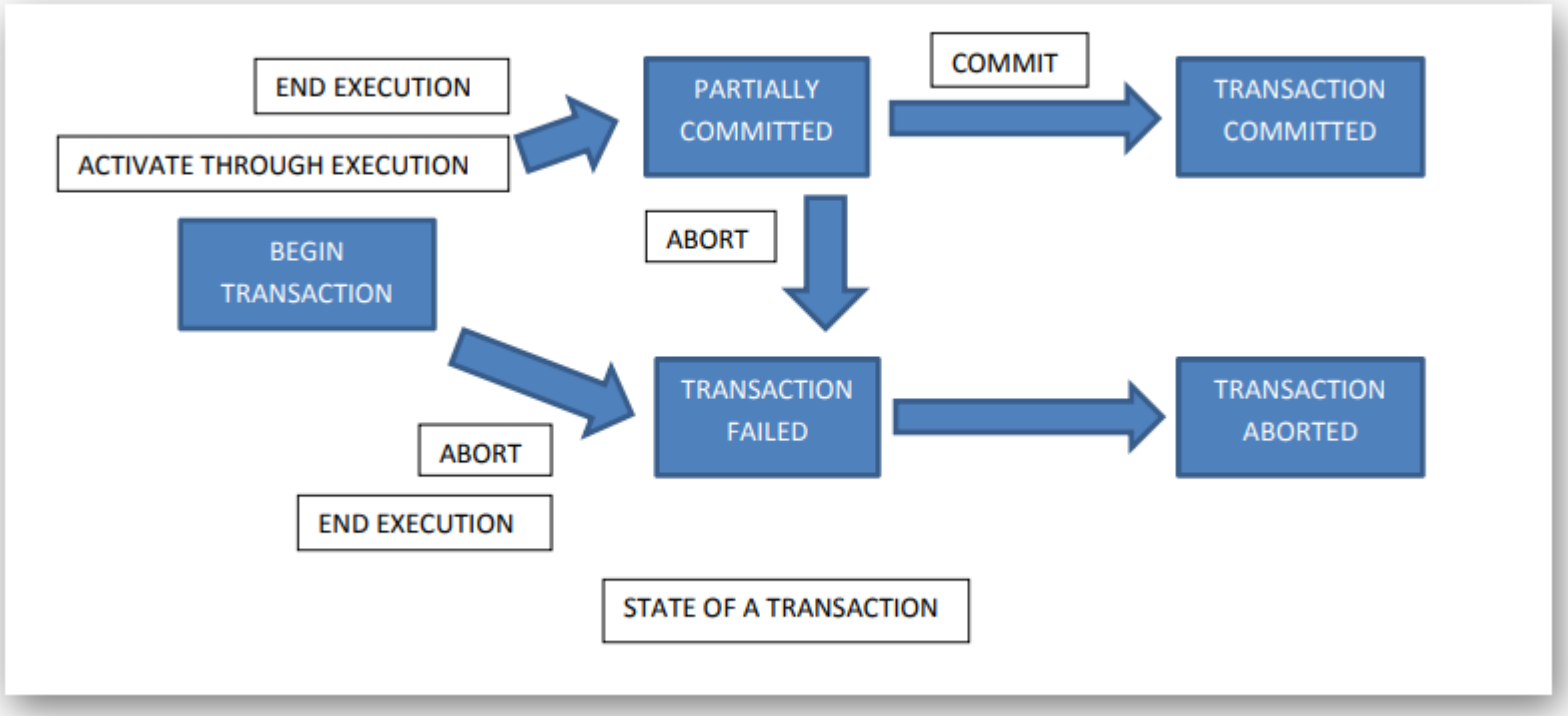
Properties of Transactions

- **Four basic *(ACID)* properties that define a transaction are:**
- **Atomicity** (*a·tuh·**mi**·suh·tee*) **'All or nothing' property.** A guarantee of **atomicity** prevents updates to the **database** occurring only partially, which **can** cause greater problems than rejecting the whole series outright.
- **Consistency: Must transform database from one consistent state to another.**
- **Isolation: Partial effects of incomplete transactions should not be visible to other transactions.**
- **Durability: Effects of a committed transaction are permanent and must not be lost because of later failure.**

DBMS Transaction Subsystem



TRANSACTION FLOW CHART



This is not a good example of a flowchart. It is missing proper flowchart symbols and flow.

Concurrency Control

Process of managing simultaneous operations on the database without having them interfere with one another.

- Prevents interference when two or more users are accessing database simultaneously and at least one is updating data.
- Although two transactions may be correct in themselves, interleaving of operations may produce an incorrect result.

Concurrency Control Techniques

- Two basic concurrency control techniques:
  - Locking,
  - Timestamping.
- Both are conservative approaches: delay transactions in case they conflict with other transactions.
- Optimistic methods assume conflict is rare and only check for conflicts at commit.

Locking

Transaction uses locks to deny access to other transactions and so prevent incorrect updates.

- Most widely used approach to ensure serializability.
- Generally, a transaction must claim a *shared* (*read*) or *exclusive* (*write*) lock on a data item before read or write.
- Lock prevents another transaction from modifying item or even reading it, in the case of a write lock.

Locking - Basic Rules

- If transaction has shared lock on item, can read but not update item.
- If transaction has exclusive lock on item, can both read and update item.
- Reads cannot conflict, so more than one transaction can hold shared locks simultaneously on same item.
- Exclusive lock gives transaction exclusive access to that item.
- Some systems allow transaction to upgrade read lock to an exclusive lock, or downgrade exclusive lock to a shared lock.

Two-Phase Locking (2PL)

Transaction follows 2PL protocol if all locking operations precede first unlock operation in the transaction.

- Two phases for transaction:
  - Growing phase - acquires all locks but cannot release any locks.
  - Shrinking phase - releases locks but cannot acquire any new locks.

Cascading Rollback

- If *every* transaction in a schedule follows 2PL, schedule is serializable.
- However, problems can occur with interpretation of when locks can be released.
- Transactions conform to 2PL.
- T_{14} aborts.
- Since T_{15} is dependent on T_{14}, T_{15} must also be rolled back. Since T_{16} is dependent on T_{15}, it too must be rolled back.
- This is called *cascading rollback*.
- To prevent this with 2PL, leave release of *all* locks until end of transaction.

Deadlock

An impasse that may result when two (or more) transactions are each waiting for locks held by the other to be released.

Time	T_{17}	T_{18}
t_1	begin_transaction	
t_2	write_lock($\mathbf{bal_x}$)	begin_transaction
t_3	read($\mathbf{bal_x}$)	write_lock($\mathbf{bal_y}$)
t_4	$\mathbf{bal_x} = \mathbf{bal_x} - 10$	read($\mathbf{bal_y}$)
t_5	write($\mathbf{bal_x}$)	$\mathbf{bal_y} = \mathbf{bal_y} + 100$
t_6	write_lock($\mathbf{bal_y}$)	write($\mathbf{bal_y}$)
t_7	WAIT	write_lock($\mathbf{bal_x}$)
t_8	WAIT	WAIT
t_9	WAIT	WAIT
t_{10}	⋮	WAIT
t_{11}	⋮	⋮

Deadlock (cont.)

- Only one way to break deadlock: abort one or more of the transactions.
- Deadlock should be transparent to user, so DBMS should restart transaction(s).
- However, in practice DBMS cannot restart aborted transaction since it is unaware of transaction logic even if it was aware of the transaction history (unless there is no user input in the transaction, or the input is not a function of the database state).
- Three general techniques for handling deadlock:
  - Timeouts.
  - Deadlock prevention.
  - Deadlock detection and recovery.

Timeouts

- Transaction that requests lock will only wait for a system-defined period of time.
- If lock has not been granted within this period, lock request times out.
- In this case, DBMS assumes transaction may be deadlocked, even though it may not be, and it aborts and automatically restarts the transaction.

Deadlock Prevention

- DBMS looks ahead to see if transaction would cause deadlock and never allows deadlock to occur.
- Could order transactions using transaction timestamps:
  - Wait-Die - only an older transaction can wait for younger one, otherwise transaction is aborted (*dies*) and restarted with same timestamp.
  - Wound-Wait - only a younger transaction can wait for an older one. If older transaction requests lock held by younger one, younger one is aborted (*wounded*).

Recovery from Deadlock Detection

- Several issues:
  - choice of deadlock victim;
  - how far to roll a transaction back;
  - avoiding starvation.

Timestamping

- Transactions ordered globally so that older transactions, transactions with *smaller* timestamps, get priority in the event of conflict.
- Conflict is resolved by rolling back and restarting transaction.
- No locks so no deadlock.

Timestamp

- A unique identifier created by DBMS that indicates relative starting time of a transaction.
- Can be generated by using system clock at time transaction started, or by incrementing a logical counter every time a new transaction starts.

Timestamping (cont.)

- Read/write proceeds only if *last update on that data item* was carried out by an older transaction.
- Otherwise, transaction requesting read/write is restarted and given a new timestamp.
- Also timestamps for data items:
  - read-timestamp - timestamp of last transaction to read item;
  - write-timestamp - timestamp of last transaction to write item.

Database Recovery

Process of restoring database to a correct state in the event of a failure.

- Need for Recovery Control
  - Two types of storage: volatile (main memory) and nonvolatile.
  - Volatile storage does not survive system crashes.
  - Stable storage represents information that has been replicated in several nonvolatile storage media with independent failure modes.

Types of Failures

- System crashes, resulting in loss of main memory.
- Media failures, resulting in loss of parts of secondary storage.
- Application software errors.
- Natural physical disasters.
- Carelessness or unintentional destruction of data or facilities.
- Sabotage.

Transactions and Recovery

- Transactions represent basic unit of recovery.
- Recovery manager responsible for atomicity and durability.
- If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to *redo* (*rollforward*) transaction's updates.

Transactions and Recovery

- If transaction had not committed at failure time, recovery manager has to *undo* (*rollback*) any effects of that transaction for atomicity.
- Partial undo - only one transaction has to be undone.
- Global undo - all transactions have to be undone.

Recovery Facilities

- DBMS should provide following facilities to assist with recovery:
  - Backup mechanism, which makes periodic backup copies of database.
  - Logging facilities, which keep track of current state of transactions and database changes.
  - Checkpoint facility, which enables updates to database in progress to be made permanent.
  - Recovery manager, which allows DBMS to restore database to consistent state following a failure.

Log File

- Contains information about all updates to database:
  - Transaction records.
  - Checkpoint records.
- Often used for other purposes (for example, auditing).

Log File (cont.)

- Transaction records contain:
  - Transaction identifier.
  - Type of log record, (transaction start, insert, update, delete, abort, commit).
  - Identifier of data item affected by database action (insert, delete, and update operations).
  - Before-image of data item.
  - After-image of data item.
  - Log management information.

Sample Log File

Tid	Time	Operation	Object	Before image	After image	pPtr	nPtr
T1	10:12	START				0	2
T1	10:13	UPDATE	STAFF SL21	(old value)	(new value)	1	8
T2	10:14	START				0	4
T2	10:16	INSERT	STAFF SG37		(new value)	3	5
T2	10:17	DELETE	STAFF SA9	(old value)		4	6
T2	10:17	UPDATE	PROPERTY PG16	(old value)	(new value)	5	9
T3	10:18	START				0	11
T1	10:18	COMMIT				2	0
	10:19	CHECKPOINT	T2, T3				
T2	10:19	COMMIT				6	0
T3	10:20	INSERT	PROPERTY PG4		(new value)	7	12
T3	10:21	COMMIT				11	0

Log File

- Log file may be duplexed or triplexed.
- Log file sometimes split into two separate random-access files.
- Potential bottleneck; critical in determining overall performance.

Checkpointing

Checkpoint

Point of synchronization between database and log file. All buffers are force-written to secondary storage.

- Checkpoint record is created containing identifiers of all active transactions.
- When failure occurs, redo all transactions that committed since the checkpoint and undo all transactions active at time of crash.

Checkpointing (cont.)

- In previous example, with checkpoint at time t_c, changes made by T_2 and T_3 have been written to secondary storage.
- Thus:
  - only redo T_4 and T_5,
  - undo transactions T_1 and T6.

Recovery Techniques

- If database has been damaged:
  - Need to restore last backup copy of database and reapply updates of committed transactions using log file.
- If database is only inconsistent:
  - Need to undo changes that caused inconsistency. May also need to redo some transactions to ensure updates reach secondary storage.
  - Do not need backup, but can restore database using before- and after-images in the log file.

Main Recovery Techniques

- Three main recovery techniques:
  - Deferred Update
  - Immediate Update
  - Shadow Paging

Deferred Update

- Updates are not written to the database until after a transaction has reached its commit point.
- If transaction fails before commit, it will not have modified database and so no undoing of changes required.
- May be necessary to redo updates of committed transactions as their effect may not have reached database.

Immediate Update

- Updates are applied to database as they occur.
- Need to redo updates of committed transactions following a failure.
- May need to undo effects of transactions that had not committed at time of failure.
- Essential that log records are written before write to database. *Write-ahead log protocol*.
- If no "*transaction commit*" record in log, then that transaction was active at failure and must be undone.
- Undo operations are performed *in reverse order in which they were written to log*.

Shadow Paging

- Maintain two-page tables during life of a transaction: *current* page and *shadow* page table.
- When transaction starts, two pages are the same.
- Shadow page table is never changed thereafter and is used to restore database in event of failure.
- During transaction, current page table records all updates to database.
- When transaction completes, current page table becomes shadow page table.

Advanced Transaction Models

- Protocols considered so far are suitable for types of transactions that arise in traditional business applications, characterized by:
  - Data has many types, each with small number of instances.
  - Designs may be very large.
  - Design is not static but evolves through time.
  - Updates are far-reaching.
  - Cooperative engineering.

Advanced Transaction Models (cont.)

- May result in transactions of long duration, giving rise to following problems:
  - More susceptible to failure - need to minimize amount of work lost.
  - May access large number of data items - concurrency limited if data inaccessible for long periods.
  - Deadlock more likely.
  - Cooperation through use of shared data items restricted by traditional concurrency protocols.

Advanced Transaction Models (cont.)

- Look at five advanced transaction models:
  - Nested Transaction Model
  - Sagas
  - Multi-level Transaction Model
  - Dynamic Restructuring
  - Workflow Models

Individual project 4

Key Assignment Draft

The case study retail store is concerned about the possibilities of losing data because of database system malfunctions or downtime. Security is also a major concern to the company because it is common knowledge that engaging in online business can be risky because of known vulnerabilities on the Internet. The company also realizes that its in-store database system is the top priority at this time. What solutions can you propose to effectively manage database transactions, maintain security, and recover the data that are lost from system failure or downtime?

Examples are provided here to help with this assignment.

The assumptions are as follows:

- A high volume of the orders often occurs during the daytime.
- One person will serve the role of database administrator.
- The database administrator account will serve as database owner.
- The transaction log must be backed up.
- Point-in-time recovery is required.
- There is an always-on availability group.
- The ability to purchase products online will be addressed in a future database project.

Individual project 4

The **project deliverables are as follows**:

- What solutions can you propose to effectively manage database transactions, maintain security, and recover the data that are lost from system failure or downtime?
- What is your rationale for the transaction management plan, database security procedure, backup plan, and recovery model that you proposed for the case study organization?
- Database Administration Plan (4-5 pages)
  - Create a database administration plan that is specific to the needs of your retail store.
  - **Include a transaction management plan that includes a flowchart for how each transaction will be handled (including rollback and commit cases).** ***Do not find a flowchart on the internet and put into paper. Must create your own for your specific store.***
  - Include a database security procedure that includes provisions for access control, user authentication, and availability.
  - Include a backup plan and a recovery model
  - Provide your analysis as to how this part of the project fulfills the mission and 1 or more goals of the case study organization.

How to create a flowchart:

Visio: https://support.microsoft.com/en-us/office/create-a-basic-flowchart-in-visio-e207d975-4a51-4bfa-a356-eeec314bd276

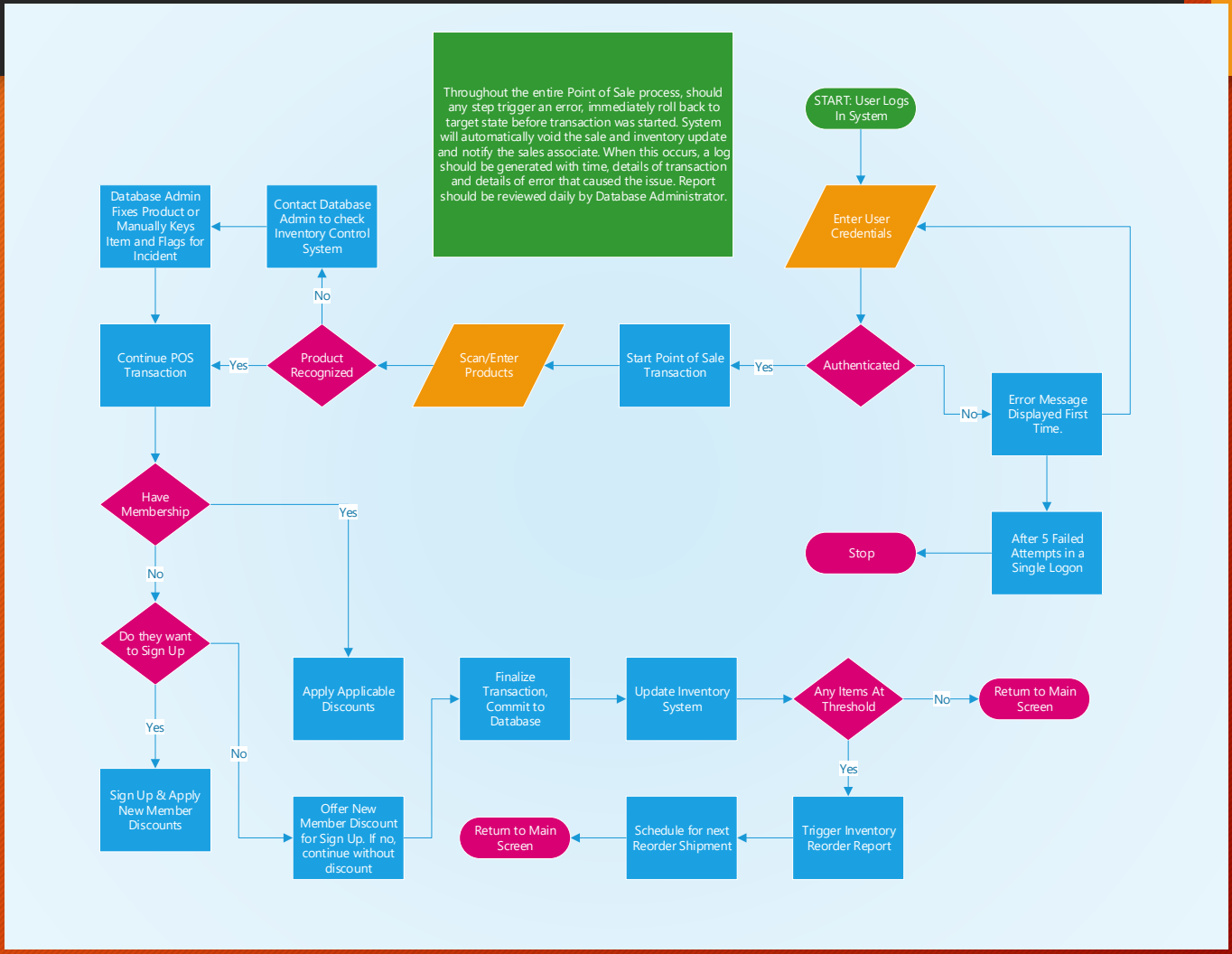
There are literally hundreds of resources available for you to use to learn how to create a flowchart.

To find more resources just Google “How do I create a flowchart”

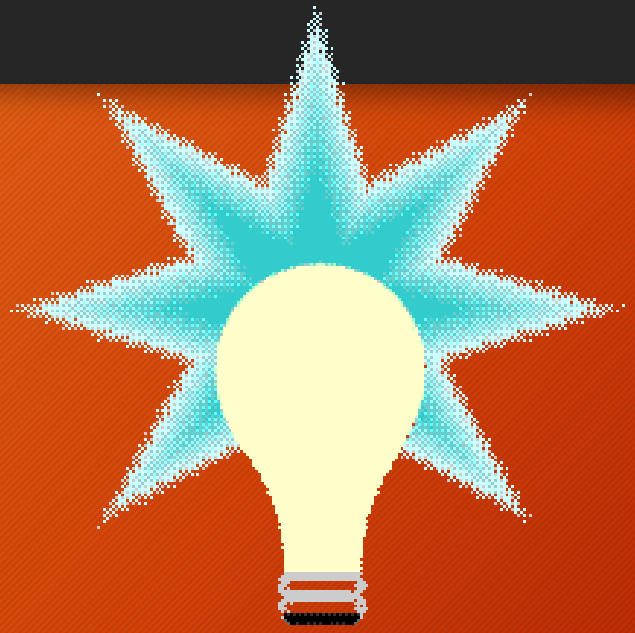
Individual project 4

- All sources should be cited both in-text and in References using APA format.
- Name the document "**yourname**_CS660_IP4.doc."
- **Please submit your assignment.**

Transaction management flow chart



Questions / Comments



(Ideas/Think Web Graphics, 2019).

Contact Information

- My e-mail address- JConklin@coloradotech.edu
- Office Hours - Thursday 6:00 P.M. – 7:00 P.M. CST
  Saturday 11:00 A.M. – 12:00 P.M. CST
- Live Chats - Wednesday 7:00 P.M. – 8:00 P.M. CST

* Please note that only one live chat session per week is required for this course. However, optional live chat sessions may be held sporadically throughout the course.

References

Colorado Technical University. (2019). Instructor's guide for CS 251-1901A-01. Retrieved from Colorado Technical University Online, Virtual Campus, Course Overview: https://campus.ctuonline.edu

Connolly, T. and C. Begg (2015). Database systems; a practical approach to design, implementation, and management, 6th ed. Portland, Pearson Education.

Ideas/Think Web Graphics. (2019). In *Desktop Publishing*. Retrieved from: http://desktoppub.about.com/od/freeclipart/l/blidea1.htm

https://www.pearson.com/us/higher-education/subject-catalog/download-instructor-resources.html?key=23976240871323439795242019