Module 3 (14 hrs)

Transactions :

- A transaction is a *logical unit* of program execution
- It is a combination of database updates which have to be performed together
- It is a logical unit of work.
- It is a unit of work with respect to concurrency and recovery.
- It is a sequence of operations including database operations that is atomic with respect to concurrency and recovery.
- An atomic execution unit that, when applied to a consistent database, generates a consistent but possibly different database.
- A short sequence of operations with the database which represents one meaningful activity in the user's environment

Transaction Processing Systems(TPS):

- Transaction processing systems(TPS) are the systems with large databases and hundreds of concurrent users that are executing database transactions
- Examples: Systems for reservations, banking , credit card processing, stock markets, super markets , super market checkout
- Typical OLTP Environments
    - Airline/ Railway Reservation Systems
    - Banking Systems (ATM)
    - Trading and Brokerage Systems
    - Hotel / Hospital Systems
    - Standard Commercial Systems

Properties (or ACID properties) of Transactions

- **Atomicity**: Either all updates are performed or none
- **Consistency**: If the database state at the start of a transaction is consistent, it will be consistent at the end of the transaction
- **Isolation**: When multiple transactions are executed concurrently, the net effect is as though each transaction has executed in isolation
- **Durability**: After a transaction completes (commits), its changes are persistent

States of transaction

- **Active**: Initial state; when the transaction is executing
- **Partially Committed**: When the last statement has finished execution
- **Failed**: On discovery that normal execution can no longer proceed

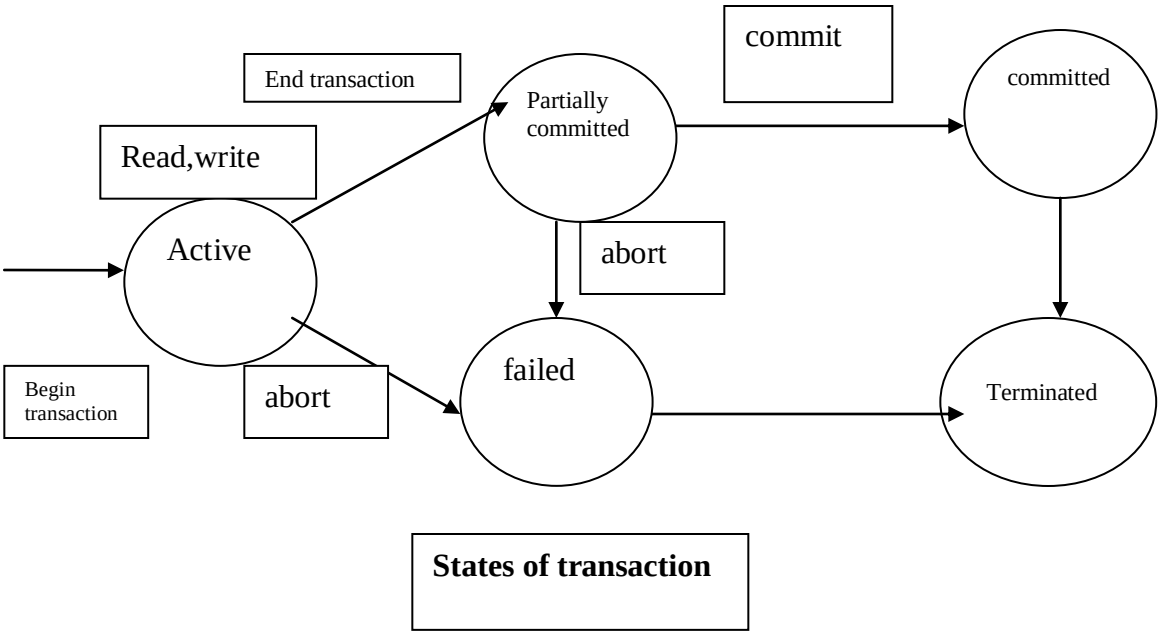
- **Aborted**: After the rollback is performed from a failed transaction
- **Committed**: After successful completion
- **Terminated**: Either committed or aborted

Database model: A database is basically represented as a collection of named data items. The size of data item is called the granularity and it can be a field of some record in the database ,or it may be a larger unit such as a record or even a whole disk block. The basic unit of data transfer from disk to main memory is one block.

Basic database access operations(processes) of a transaction:

Read operation: read_item(X): It reads a database item named X into a program variable named X.

Write operation: write_item(X): It writes the value of program variable X into the database item named X.



States of transaction

System log(DBMS journal): It keeps track of all transaction operations that affect the values of database items in order to be able to recover from failures that affect transactions

Log records: Log records are the types of entries that are written to the log and the action each performs.

In these entries, T refers to a unique transaction_id that is generated automatically by the system and is used to identify each transaction:

1.[start-transaction, T] : Indicates that transaction T has started execution.

2.[write_item, T,X,old_value,new_value]: Indicates that transaction T has changed the value of database item X from an old value to new value.

3.[read_item, T,X]: Indicates that transaction T has read the value of database item X

4.[commit,T]: Indicates that transaction T has completed successfully, and affirms that its effect can be committed (recorded permanently) to the database , it is also known as commit point of a transaction

5.[abort,T]: Indicates that transaction T has been aborted

Concurrency control

- The process of managing simultaneous operations on the database without having them interfere with one another is called concurrency control
- Example: In airline reservation system, where the database containing available seats may accessed by many agents throughout the world
- The need for concurrency control (Why concurrent control is needed?) (Problem of concurrent transactions)
    - The lost update problem
    - The uncommitted dependency or dirty read/Temporary update problem
    - The inconsistent/incorrect summary problem
    - Unrepeatable read

The lost update problem(WW conflicts)

- This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database items incorrect
- An apparently successfully completed update operation by one user can be overridden by another user

- Example: Let transactionsT1 and T2 are concurrently executing, T1 is withdrawing $10 from an account with balance balx, initially $100, and T2 is depositing $100 into the same account. (i.e.Balance=100-10+100=190)

Time	T1	T2	balx
t1		begin_transaction	100
t2	begin_transaction	read(balx)	100
t3	read(balx)	balx=balx+100	100
t4	balx=balx-10	write(balx)	200
t5	write(balx)	commit	90
t6	commit		90

- The loss of T2's update is avoided by preventing T1 from reading the value of balx until after T2' s update has been completed
- T1 overwriting the previous updates(i.e. 200) to write $90 at t5 therby 'losing' the $100 previously added to the balance

Dirty read (WR conflict)

(The uncommitted dependency/temporary update problem)

- This problem occurs when one transaction is allowed to see the intermediate results of another transaction before it has committed

Time	T3	T4	balx
t1		begin_transaction	100
t2		read(balx)	100
t3		balx=balx+100	100
t4	begin_transaction	write(balx)	200
t5	read(balx)	. . .	200
t6	balx=balx-10	rollback	100
t7	write(balx)		190
t8	commit		190

- The value of balx read by T3 at time t5(i.e. $200) is called dirty data, giving rise to alternative name, the dirty read problem ,but actually balx should be 90 instead of 190

The inconsistent analysis (incorrect summary) problem

- This problem occurs when a transaction reads several values from a database but a second transaction updates some of them during the execution of first

Time	T5	T6	balx	baly	balz	sum
t1		begin_transaction	100	50	25	
t2	begin_transaction	sum=0	100	50	25	0
t3	read(balx)	read(balx)	100	50	25	0
t4	balx=balx-10	sum=sum+balx	100	50	25	100
t5	write(balx)	read(baly)	90	50	25	100
t6	read(balz)	sum=sum+baly	90	50	25	150
t7	balz=balz+10		90	50	25	150
t8	write(balz)		90	50	35	150
t9	commit	read(balz)	90	50	35	150
t10		sum=sum+balz	90	50	35	185
t11		commit	90	50	35	185

- Sum should be =175 but it came to be =185, which is inconsistent result

Un(non) repeatable (fuzzy) Reads(RW conflicts)

- This problem occurs when a transaction T1 reads an item twice and the item is changed by another transaction T2 between the two reads
- Thus T1 receives two different values for the same data item
- Example:

T1	T2	X=2000	
		T1	T2
READ X		2000	
	UPDATE X		3000
READ X		3000	

Phantom read

- If a transaction T executes a query that retrieves a set of tuples from a relation satisfying a certain predicate, re-executes the query at a later time but finds that the retrieved set contains an additional (phantom) tuple that has been inserted by another transaction in the mean time
- This is some times referred to as phantom read .

Schedule

- A schedule S is defined as the sequential ordering of the operations of 'n' interleaved transactions
- A schedule is the chronological order in which instructions are executed in the system
- A schedule for a set of transactions must consist of all instructions of those transactions, and must preserve the order in which the instructions appear in each individual transaction

Serial Schedule

- A schedule S of n transactions is said to be a serial schedule if the operations of each transactions are executed consecutively without any interleaved operations from other transactions
- Each serial schedule consists of a sequence of a sequence of instructions from various transactions, where the instructions belonging to one single transaction appear together in that schedule.
- For a set of n transactions , there exist n! different serial schedules
- When several transactions are executed concurrently, the corresponding schedule no longer needs to be serial
- If two transactions are running concurrently, the operating system may execute one transaction for a little while, then perform a context switch, execute the second transaction for some time, and then switch back to the to the first transaction for some time, and so on.
- With multiple transactions , the CPU time is shared among all the transactions

Non-serial schedule

- A non-serial schedule is a schedule where the operations from a set of concurrent transactions are interleaved

Conflict operations

- WR conflict: T2 reads a data object previously written by T1
- RW conflict: T2 writes a data object previously read by T1
- WW conflict: T2 writes a data object previously written by T1

Complete schedule

- A schedule that contains either an abort or a commit as the last operation for each transaction whose actions are listed in it is called a complete schedule
- A complete schedule must contain all the actions of every transaction that appears in it.
- For any two conflicting operations(WR,RW,WW performed by Ti and Tj in order) one of the two must occur before the other in the schedule(Total order)

Serializability (Serialisable schedules)

- Any schedule that produces the same results as a serial schedule is called serial sable schedule
- On executing a set of concurrent transactions on a database the net effect should be as though the transactions were executed in *some* serial order.
- Serialisability theory attempts to determine the correctness of the schedules
- The rule of this theory is: A schedule S of n transactions is serial sable if it is equivalent to some serial schedule of the same 'n' transactions

Example:

Transaction T1:
read (A);
A = A – 50;
write (A);
read (B)
B = B + 50;
write (B);

Transaction T2:
read (A);
t = A * 0.1;
A = A – t;
write (A);
read (B) ;
B = B + t;
write (B);

Serial Schedules (Schedule-1)

(Take: A=1000, B=2000)

T1		
read (A)	A=1000	
A = A – 50;	A=950	Equivalent to T1 followed by T2
write (A)	A=950	
read (B)	B=2000	
B = B + 50;	B=2050	
write (B)	B=2050	

T2		
read (A)	A=950	
t = A * 0.1;	t=95	
A = A – t;	A=855	
write (A)	A=855	
read (B)	B=2050	
B = B + t;	B=2145	
write (B)	B=2145	A+B=855+2145=3000=>consistent state

Serial Schedules (Schedule-2)

T2				
read (A)	1000			
t = A * 0.1;	100		Equivalent to T2	followed by T1
A = A – t;	900			
write (A)	900			
read (B)	2000			
B = B + t;	2100			
write (B)	2100	T1		
		read (A) ;	900	
		A = A – 50;	850	
		write (A);	850	
		read (B);	2100	
		B = B + 50;	2150	Consistent state as
		write (B) ;	2150	A+B=850+2150=3000

Schedule-3:Concurrent executions of serial schedule 1 and 2, producing consistent state as A+B=855+2145=3000

read(A)	1000		
A:=A-50	950		
write(A)	950		
		read(A)	950
		t:=A*0.1;	95
Context switch or interleaved in concurrent transactions T1 and T2 interleaved in concurrent transactions T1 and T2		A:=A-t;	855
		write(A)	855
read(B)	2000		
B:=B+50;	2050		
Write(B)	2050		
		read(B)	2050
		B:=B+t;	2145
		write(B)	2145

Concurrent transactions but inconsistent state

- **Not all concurrent executions result in a correct state**

- Example:Schedule1 and 2 in concurrent transactions T1 and T2

T1		T2	
read(A)	1000	read(A)	1000
A:=A-50;	950	t:=A*0.1;	100
		A:=A-t;	900
		write(A)	900
		read(B)	2000
write(A)	950		
read(B)	2000		
B:=B=50;	2050		
write(B)	2050		
		B:=B+t;	2100
		write(B);	2100

Here A+B=950+2100=3050≠3000 => The schedule-1 and 2 are in inconsistent state.

Serial schedule

- A serial schedule is a schedule in which either transaction T1 is completely done before T2 or transaction T2 is completed done before T1

Non serial schedule

- A non serial schedule is a schedule where the operations form a set of concurrent transactions are interleaved

Resultant equivalent

- Two schedules are called resultant equivalent if they produce the same final state of the database

Conflict Serializability

Let instructions I and J belonging to transactions T1 and T2 be executed consecutively by the DBMS.

1. I and J can be swapped in their execution order if I and J refer to different data elements

2. I and J can be swapped in their execution order **iff** I and J refer to the same data element and both perform a **read** operation only.
3. I and J are said to *conflict* if I and J belong to different transactions and at least one of them is a write operation.

Conflict Serializability and Conflict equivalent

- If a schedule S can be transformed to another S' by swapping non conflicting instructions, then S and S' are said to be *conflict equivalent.*
- A schedule S is said to be *conflict serializable* if it is conflict equivalent to some serial schedule S'.

Example(1) of Conflict equivalent

T1	T2		T1	T2
R(A)			R(A)	
W(A)				R(B)
	R(B)	EQUIVALENT TO	W(A)	
	W(B)			
				W(B)

(W(A) and R(B) are non conflict and swapped) (R(B) and W(A) are conflict and swapped)

EQUIVALENT TO

T1	T2
R(A)	
W(A)	
	R(B)
	W(B)

Example(2) of Conflict equivalent

Schedule S

T1	T2
A=A+100	
B=B-100	
	A=A-7.06
	B=B*7.06

Schedule S'

T1	T2
A=A+100	
	A=A*7.06
B=B-100	
	B=B*7.06

Schedule S is conflict serializable since it it is conflict equivalent to the serial schedule S'

Two schedules produce the same outcome , but that are not conflict equivalent

T1	T2
R(A)	
A:=A-50	
W(A)	
	R(B)
	B:=B-10
	W(B)
R(B)	
B:=B+50	
W(B)	
	R(A)
	A:=A+10
	W(A)

The computation of T1 andT2 in the fig. is same as the computation of serial schedule<T1,T2> but are not conflict equivalent, because write(B) instruction of T2 conflicts with the read(B) instruction of T1

Example of a non-conflict serializable schedule

- T3 T4

read(Q)

write(Q)

write(Q)

This schedule is not conflict- serializable , since it is not equivalent to either the serial schedule<T3,T4> or <T4,T3>.

Serialisability test

Algorithm to determine wheather a schedule is serialisable or not?

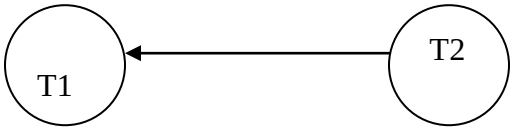
- The steps of constructing a precedence graph are:
  - 1.Create a node for every transaction in the schedule
  - 2.Find the precedence relationships in conflicting operations.Conflicting operations are read-write(WR) or write-read(WR) or write-write(WW) on the same data item in two different transactions
    - (2.1)For a transaction Ti which reads an item A, find a transaction Tj that writes A later in the schedule.If such a transaction is found, draw an edge from Ti to Tj
    - (2.2)For a transaction Ti which has written an item A, find a transaction Tj later in the schedule that reads A. If such a transaction ids found, draw an edge from Ti to Tj
    - (2.3)For a transaction Ti which has written an item A, find a transaction Tj that writes A later than Ti. If such a transaction is found, draw an edge from Ti to Tj
  - 3.If there is any cycle in the graph, the schedule is not serialisable,otherwise, find the equivalent serial schedule of the transaction by traversing the transaction nodes with the node that has input edge
  - Time taken for this test is of order of $n^2 = O(n^2)$, where n is the number of transactions

Example-1 to illustrate serialisability test

An interleaved Schedule		
Schedule	T1	T2
Read X	Read X	

Subtract 100	Subtract 100	
Read X		**Read X**
Write X	**Write X**	
Read Y		**Read Y**
Read y	Read Y	
Add 100	Add 100	
Display x+y		Display x+y
Write Y	**Write Y**	

- As per step1 of the algorithm we draw the two nodes for T1 and T2
- Transaction T2 reads data item X, which is subsequently written by T1, thus there is an edge from T2 to T1(clause 2.1)
- Also T2 reads data items Y, which is subsequently written by T1, thus there is an edge from T2 to T1(clause 2.1)
- However , that edge already exists, so we do not need to redo it.
- There are no cycles in the graph => The given schedule (Previous page) is serialisable
- The equivalent serial schedule(as per step 3) would be T2 followed by T1(the diagram is given in the next slide)
- The schedule is T2-T1
- Precedence graph for the schedule**: Not a cycle**:T2-T1=> The schedule is **serial sable**

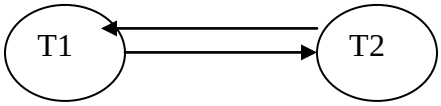


Example-2 to illustrate serialisability test

Schedule	Transaction1 (T1)	Transaction2 (T2)	Example values
Read X	Read X		X=50000
Subtract 100	Subtract 100		49900
Write X	**Write X**		X=49900
Read X		**Read X**	X=49900
Read Y		**Read Y**	Y=100000
Read X+Y		Display X+Y	149900
Read Y	Read Y		Y=100000
Add 100	Add 100		100100

Write Y	**Write Y**		Y=100100

- Schedule in example-2 is not serialisable , because in that schedule , the two edges that exist between nodes T1 and T2 are:
  - T1 writes X which is later read by T2(clause2.2), so there exists an edge from T1 to T2
  - T2 reads Y which is later written by T1(clause2.1), so there exists an edge from T2 to T1
- Thus the graph for the schedule will be : **Cycle**:T1-T2-T1=> Schedule is **not serialisable**



View Equivalent

If S is a schedule, then S' is a *view equivalent* schedule if

- For each data item Q, if a transaction Ti reads the initial value of Q in S, then it should read the initial value in S' also
- In schedule S, for each data item Q, if write(Q) of Tj precedes read(Q) of Ti, it should be the same in S'
- The same transaction that performs the final write(Q) in S, should perform it in S'.

View serializable

- A schedule S is view serializable if it is conflict equivalent to a serial schedule
- Every conflict-serializable schedule is view serializable, but there are view serializable schedules that are not conflict serializable
- The time taken for testing the view serialiability =$O(2^n)$

Example of view serializable:

T1	T2	T3
read(Q)		
	write(Q)	
write(Q)		
		write(Q)

A view equivalent schedule:

T1: Read(Q)
T2: Write(Q)
T1: Write(Q)
T3: Write(Q)

The above Schedule is a view serialisable

Example of View equivalent

Schedule-1

```
T1
read (A)
A = A – 50;
write (A)
read (B)
B = B + 50;
write (B)
```

```
T2
read (A)
t = A * 0.1;
A = A – t;
write (A)
read (B)
B = B + t;
write (B)
```

Schedule2

T2

read (A)
t = A * 0.1;
A = A – t;
write (A)
read (B)
B = B + t;
write (B)

T1

read (A)
A = A – 50;
write (A)
read (B)
B = B + 50;
write (B)

Schedule1 is not view equivalent to Schedule2, since , in schedule 1, the value of account A read by transaction T2 was produced by T1, where as this case doesn't hold in schedule2

Constrained write assumption condition

- The definitions of conflict and view serializability are similar if a condition known as the constrained write assumption holds on all transactions in the schedule
- The constraint write assumption condition is as follows:
- Any write operation in wi(x) in Ti is preceded by a ri(x) in Ti and that the value written by wi(x) in Ti depends only on the value of x read by ri(x)

Blind write

The definitions of view serializability is less restrictive than that of conflict serializability under the unconstrained write assumption , where the value written by an operation Wi(X) in Ti can be independent of its old value from the database. This is called a blind rule.

Example:

T1	T2	T3
read(Q)		
	write(Q)	
write(Q)		
		write(Q)

- Here transactions T2 and T3 perform write(Q) operations without having performed a read(Q) operation
- Writes of this sort are called blind writes
- Blind writes appear in any view-serializable schedule that is not conflict serializable
- Every conflict serializable schedule is also view serializable; however (converse is not true)some view serializable schedules are not conflict serializable
- The problem of testing for view serializability has been shown to be NP-hard
- A schedule that is view serializable but not conflict serializable is characterized by *blind writes*.

Recoverability

- If a transaction Ti fails, for whatever reason, we need to undo the effect of this transaction to ensure the atomicity property of the transaction
- If a system that allows concurrent execution, it is necessary also to ensure that any transaction Tj that is dependent on Ti(i.e. Tj has read data written by Ti) is also aborted
    - Recoverble schedules
    - Cascadeless schedules

- **Recoverble schedules** :A recoverable schedule is one where, for each pair of transactions Ti and Tj such that Tj reads a data item previously written by Ti, the commit operation of Ti appears before the commit operation of Tj
    - Database systems require recoverable schedules.
    - Consider the following schedule

T1	T2
read(A)	
write(A)	
	read(A)
read(B)	

    - Suppose T2commits before T1
    - If T1fails before it commits then T2also has to be aborted.
    - However, T2is already committed.
    - A situation where it is impossible to recover from the failure of T1
    - This is an example of non recoverable schedule
- **Cascade less Schedule**: A cascade less schedule is one where, for each pair of transactions Ti and Tj such that Tj reads a data item previously written by Ti, the commit operation of Ti appears before the read operation of Tj
- The phenomenon , in which a single transaction failure leads to a series of transaction rollbacks, is called cascading rollback
- Cascading rollback is undesirable- leads to undoing a lot of work
- Restrict the schedules to those where cascading rollbacks do not occur.
- It is easy to verify that every cascade less schedule is also recoverable
- Even if a schedule is recoverable, to recover from the failure of a transaction, there is a need to rollback several transactions.

T	T1	T2
read(A)		
read(B)		
write(A)		
	read(A)	
	write(A)	
		read(A)

- if T fails, then it will lead to rolling back T1 and T2.
- This is an example of cascading rollback.

Serializability Test

- It is just a test whether a given interleaved schedule is ok or has a concurrency related problem.
- However, it does not ensure that interleaved concurrent transactions do not have any concurrency related problems.
- This can be done using locks.
- Locking ensures serializability of executing transactions.

Problem of concurrent transactions

- The lost update problem
- The uncommitted dependency or dirty read/Temporary update problem
- The inconsistent/incorrect summary problem
- Unrepeatable read

Concurrency control techniques (algorithms)

- Conservative(or Pessimistic) approach
    - Locking
    - Timestamp
- Optimistic approach

Conservative (or Pessimistic) approach

- This approach causes transactions to be delayed in case they conflict with each other at some time in the future
- Pessimistic execution: If there is a validation according to compatibility of lock then only read, compute and write operations are performed
- The execution sequence is:
  Validate Read Compute Write
- Two pessimistic approaches are:
  - Locking
  - Time stamping

Optimistic approach

- These are based on the premise that conflict is rare so they allow transactions to proceed unsynchronized and only check for conflicts at the end, when transaction commits
- Optimistic execution is :
  Read Compute Validate Write

Types of Conservative/ Pessimistic approach

- Locking/2PL protocol
- Timestamp
  - Timestamp ordering protocol
  - Multi-version concurrency control

Lock-Based Protocols

- A lock/locking is a mechanism to control concurrent access to a data item
- A data can be locked by a transaction in order to prevent this data item from being accessed
- and updated by any other transaction
- The part of the database system which is responsible for locking or unlocking the data items is known as the lock manager
- Data items can be locked in two modes :
  1. *exclusive (X) mode*. Data item can be both read as well as written. X-lock is requested using lock-X instruction. (Exclusive lock)
     (i) It is requested by a transaction on a data item that it needs to update
     (ii) No other transaction can place either a shared lock or an exclusive lock on a data item that has been locked in an exclusive mode
     A data item locked in the exclusive mode can not be locked in the shared mode or in exclusive mode by any other transaction.
  2. *shared (S) mode*. Data item can only be read. S-lock is requested using lock-S instruction. (Shared lock or read lock)
     (i) It is requested by a transaction that wants to just read the value of data item
     (ii) A shared lock on a data item does not allow an exclusive lock to be placed but permits any number of shared locks to be placed on that item.
     A data item locked in shared lock cannot be locked in the exclusive mode by any other transaction but can be locked in the shared lock
- Lock requests are made to concurrency-control manager. Transaction can proceed only after request is granted.

Types of locks

- Binary lock
- Multiple –mode locks
  - Binary lock: This locking mechanism has two states for a data item; locked or unlocked.
  - Disadvantages of binary lock: It can't be used for practical purposes.

- Multiple-mode lock: This locking mechanism has three states for to a data item: read locked or shared locked(read_lock(X)), write locked or exclusive locked. (write_lock(X) and unlocked (unlock(X))

Two Phase Locking(2PL) Protocol

- This is a protocol which ensures conflict-serializable schedules.
- **Definition of 2PL protocol**: A transaction follows the two-phase locking protocol if all locking operations precede the first unlock operation in the transaction.
- Phase 1: Growing Phase
  - transaction may obtain locks
  - transaction may not release locks
- Phase 2: Shrinking Phase
  - transaction may release locks
  - transaction may not obtain locks
- The protocol assures serializability. It can be proved that the transactions can be serialized in the order of their lock points (i.e. the point where a transaction acquired its final lock).
- A transaction must acquire a lock on an item before on operating on the item.
- The lock may be read or write, depending on the type of access needed.
- Once the transaction releases a lock, it can never acquire any new lock.
- By using two phase locking protocol we can prevent the followings:
  - Preventing the lost update problem (WW conflict) using 2PL
  - Preventing the uncommitted dependency(or dirty read or temporary update) (WR conflict) problem using 2PL
  - Preventing the inconsistent analysis (incorrect summary) problem using 2PL
  - Preventing unrepeatable (RW conflict) read using 2PL

Preventing the problems occurred in concurrent transactions using two phase locking(2PL)

Preventing the lost update problem (WW conflict) using 2PL

Example: Let transactionsT1 and T2 are concurrently executing, T1 is withdrawing $10 from an account with balance balx, initially $100, and T2 is depositing $100 into the same account.(i.e.Balance=100-10+100=190)

Time	T1	T2	balx
t1		begin_transaction	100
t2	begin_transaction	write_lock(balx)	100
t3	write_lock(balx)	read(balx)	100
t4	wait	balx=balx+100	100
t5	wait	write(balx)	200
t6	wait	commit /unlock(balx)	200
t7	read(balx)		200
t8	balx=balx-10		200
t9	write(balx)		190
t10	commit/unlock(balx)		

Preventing the uncommited dependency(or dirty read or temporary update) (WR conflict) problem using 2PL

Time	T3	T4	balx
t1		begin_transaction	100
t2		write_lock(balx)	100
t3		read(balx)	100
t4	begin_transaction	balx=balx+100	100
t5	write_lock(balx)	write(balx)	200
t6	wait	rollback/unlock(balx)	100

t7	read(balx)	100
t8	balx=balx-10	100
t9	write(balx)	90
t10	commit/unlock(balx)	90

Preventing the inconsistent analysis(incorrect summary) problem using 2PL

Time	T5	T6	balx	baly	balz	sum
t1		begin_transaction	100	50	25	
t2	begin_transaction	sum=0	100	50	25	0
t3	write_lock(balx)		100	50	25	0
t4	read(balx)	read_lock(balx)	100	50	25	0
t5	balx=balx-10	wait	100	50	25	0
t6	write(balx)	wait	90	50	25	0
t7	write_lock(balz)	wait	90	50	25	0
t8	read(balz)	wait	90	50	25	0
t9	balz=balz+10	wait	90	50	25	0
t10	write(balz)	wait	90	50	35	0
t11	commit/unlock(balx,balz)	wait	90	50	35	0
12		read(balx)	90	50	35	0
t13		sum=sum+balx	90	50	35	90
t14		read_lock(baly)	90	50	35	90
t15		read(baly)	90	50	35	90
t16		sum=sum+baly	90	50	35	140
t17		read_lock(balz)	90	50	35	140
t18		read(balz)	90	50	35	140
t19		sum=sum+balz	90	50	35	175
t20		commit/unlock(balx,baly,balz)	90	50	35	175

Preventing unrepeatable(RW conflict) read using 2PL

T1	T2	X=2000	
		T1	T2
READ X		2000	
	UPDATE X	Write_lock(X)	3000
READ X		2000	

Types of 2PL

- The Basic 2PL
- Strict 2PL
- Rigorous 2PL
- Conservative (Static) 2PL

The basic 2PL

- It allows release of lock at any time after all the locks have been acquired
- Disadvantages of the basic 2PL
    - Basic 2Pl suffers from the problem that it can result into loss of atomic/isolation property of transaction as theoretically speaking once a lock is released on a data item it can be modified by another transaction before the first transaction commits or aborts

Strict 2PL

- Most popular variation of 2PL
- It guarantees strict schedules and avoids the disadvantages of basic 2PL

- In strict 2PL, a transaction T doesn't release any of it's exclusive (write) locks until after it commits or aborts
- No other transaction can read or write an item that is written by T unless T has committed, leading to a strict schedule for recoverability
- Strict 2PL solves the problem of concurrency and atomicity
- Disadvantage is :

  It is not deadlock free i.e. it can introduce "Deadlock".

Rigorous 2PL

- It is a more common restrictive variation of strict 2PL.
- It guarantees strict schedules.
- In this variation, a transaction T doesn't release any of its locks (exclusive or shared) until after it commits or aborts .
- So it is easier to implement than strict 2PL .

Conservative (Static) 2PL

- It is a variation of 2Pl which requires a transaction to lock all the items it accesses before the transaction begins execution, by pre declaring it's read_set and write_set.

Problems or Pitfalls with 2PL protocol

- Cascading rollback
- Deadlock
- Starvation

Cascading rollback

- It is a situation in which a single transaction leads to a series of rollbacks

Time	T1	T2	T3
t1	begin_transaction		
t2	write_lock(balx)		
t3	read(balx)		
t4	read_lock(baly)		
t5	read(baly)		
t6	balx=balx+baly		
t7	write(balx)		
t8	unlock(balx)	begin_transaction	
t9	…..	Write_lock(balx)	
t10	….	read(balx)	
t11	…	balx=balx+100	
t12	…	write(balx)	
t13	…	unlock(balx)	
t14	….	……	
t15	rollback	…….	
t16		……..	Begin_transaction
t17		…. ..	read_lock(balx)
t18		rollback	…….
t19			rollback

- Let us consider ,if transaction T1 fails after the read_lock(balx) operation of transaction T3.
- Then T1 must be rollback, which results into rollback of T2 and T3, so that actual value of
- balx from t1 will be accepted by T2 and T3.

Solutions to avoid cascading of rollbacks

- It can be done in two ways by using
  - Strict 2Pl protocol

- Rigorous 2Pl protocol

Solution to avoid cascading of rollback by using strict 2PL protocol

- The strict 2PL protocol, requires ,that in addition to locking being 2 phase, all exclusive mode(lock_X(A)i.e.write_lock(A)) taken by a transaction must be hold until that transaction commits.
- This requirement ensures that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits, preventing any other transaction from reading the data

Solution to avoid cascading of rollback by using rigorous 2PL protocol

- It requires that all locks to be held until the transaction commits
- It can be easily verified that, with rigorous 2PL, transactions can be serialized in the order in which they commit.

Deadlock

- Definition: It is an impasse that may result when two (or more) transactions are each waiting for locks to be released that are held by the other.
- Deadlock occurs when each transaction T in a set of two or more transactions is waiting for some item that is locked by some other transaction T1 in the set.
- Hence each transaction in the set is waiting on awaiting queue , waiting for one of the other transactions in the set to release the lock on an item.

Example:

Time	T1	T2
t1	BEGIN_TRANSACTION	
t2	WRITE_LOCK(BALX)	BEGIN_TRANSACTION
t3	READ(BALX)	WRITE_LOCK(BALY)
t4	BALX=BALX-10	READ(BALY)
t5	WRITE(BALX)	BALY=BALY+100
t6	WRITE_LOCK(BALY)	WRITE(BALY)
t7	WAIT	WRITE_LOCK(BALX)
t8	WAIT	WAIT
t9	WAIT	WAIT
t10	…	…
t11	…	…

General techniques for handling deadlocks

- Timeouts
- Deadlock prevention
- Deadlock detection and recovery

Time outs (or Lock timeouts)

- A transaction that requests a lock will wait for only a system-defined period of time.
- If the lock has not been granted within this period, the lock request times out .
- The transaction aborts and automatically restarts the transaction.
- This is used by several commercial RDBMS s

Deadlock Prevention Strategies

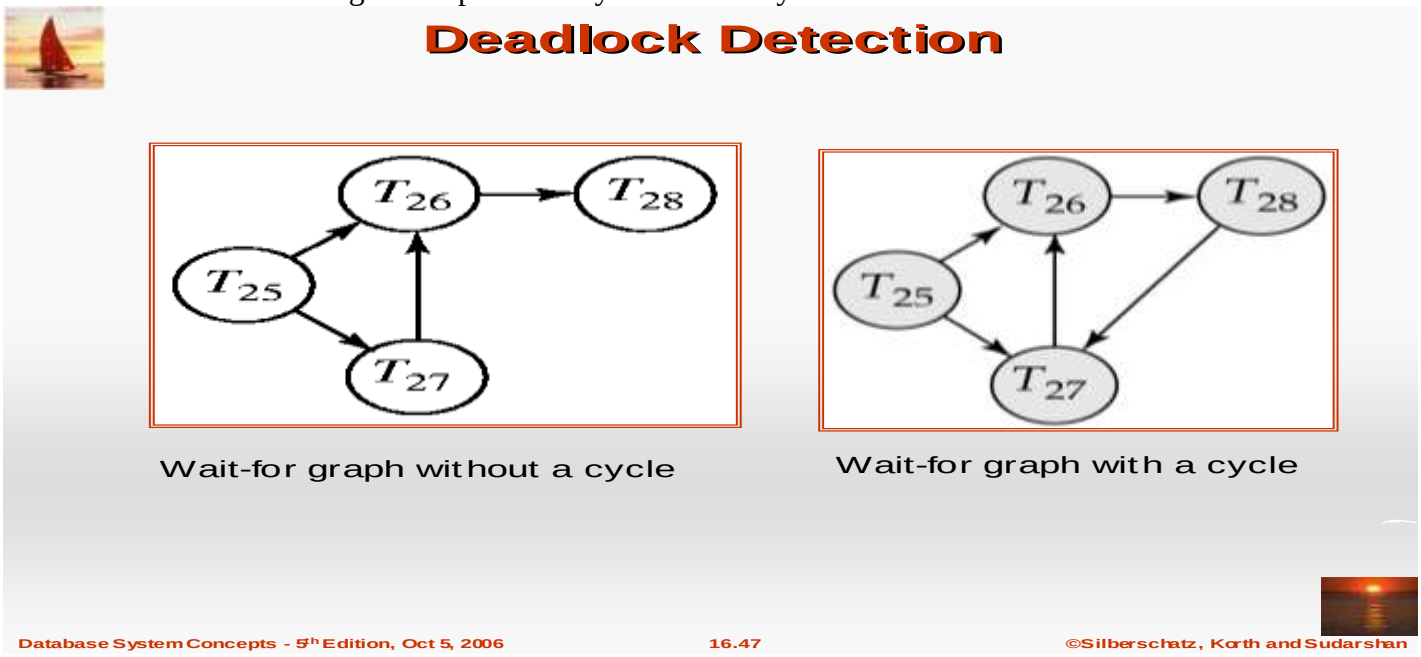
- Following schemes use transaction timestamps for the sake of deadlock prevention alone.
- Transaction timestamp (TS(T)): It is a unique identifier assigned to each transactions .A timestamp is the system generated sequence number that is unique for each transaction. If transaction T1 starts before transaction T2, then TS(T1)<TS(T2).T1 is called **older transaction** whereas T2 is called **younger transaction.**
- There are two algorithms for dead lock prevention:
    - Wait-die and Wound-wait algorithms
    - No waiting(NW) and Cautious waiting(CW) algorithms
- **Wait-die** scheme — non-preemptive

- older transaction may wait for younger one to release data item. Younger transactions never wait for older ones; they are rolled back instead.
- A transaction may die several times before acquiring needed data item.

- **Wound-wait** scheme — preemptive
  - older transaction *wounds* (forces rollback) of younger transaction instead of waiting for it. younger transactions may wait for older ones.
  - May be fewer rollbacks than *wait-die* scheme.
- **No waiting (NW) algorithm**: If a transaction is unable to obtain a lock , it is immediately aborted and then restarted after a certain time delay without checking whether a deadlock will actually occur or not
- **Cautious waiting (CW) algorithm**: If Tj is not blocked (not waiting for some other locked item), then Ti is blocked and allowed to wait; otherwise abort Ti.

Deadlock Detection

- **Deadlock detection algorithm**
  - Here the system checks if a state of deadlock actually exists.
  - Deadlock detection is usually handled by the construction of a wait-for graph(WFG) that shows the transaction dependencies.
  - Transaction Ti is dependent on Tj if transaction Tj holds the lock on a data item that Ti is waiting for.
  - The WFG is a directed graph with Pair $G = (V,E)$,
    - V is a set of vertices (all the transactions in the system)
    - E is a set of edges; each element is an ordered pair $Ti \rightarrow Tj$.
  - If $Ti \rightarrow Tj$ is in E, then there is a directed edge from Ti to Tj, implying that Ti is waiting for Tj to release a data item.
  - When Ti requests a data item currently being held by Tj, then the edge Ti Tj is inserted in the wait-for graph. This edge is removed only when Tj is no longer holding a data item needed by Ti.
  - The system is in a deadlock state if and only if the wait-for graph has a cycle. Must invoke a deadlock-detection algorithm periodically to look for cycles.



- **Invoking the deadlock detection algorithm depends on two** factors:
  - How often does a deadlock occur?

- How many transactions will be affected by the deadlock?

- **Frequency of deadlock detection**:
  - Since a cycle in the WFG is a necessary and sufficient condition for deadlock to exist, the deadlock detection algorithm generates the WFG at regular intervals and examines it for a cycle.
  - The choice of time interval between executions of the algorithm is important.
  - If the interval chosen is too small, deadlock detection will add considerable overhead.
  - If the interval is too large, deadlock may not be detected for a long period.
  - Alternatively a dynamic deadlock detection algorithm could start with an initial interval size, each time no deadlock is detected, the deadlock interval could be increased, for example, to twice the previous interval, and each time deadlock is detected, the interval could be reduced , e.g., to half the previous interval, subject to some upper and lower limits.

Deadlock Recovery

- When a detection algorithm determines that a deadlock exists, the system must recover from the deadlock
- When deadlock is detected :
  - Some transaction will have to rolled back (made a victim) to break deadlock. Select that transaction as victim that will incur minimum cost.
  - Rollback -- determine how far to roll back transaction
    - Total rollback: Abort the transaction and then restart it.
    - More effective to roll back transaction only as far as necessary to break deadlock.
  - Starvation happens if same transaction is always chosen as victim. Include the number of rollbacks in the cost factor to avoid starvation.
  - The following are the several issues:
    - Choice of deadlock victim
    - How far to roll a transaction back
    - Avoiding starvation
  - **Choice of deadlock victim:** Choosing which transaction to abort is known as victim selection. We should rollback those transactions that will incur the minimum cost. When deadlock is detected, the choice of which transaction to abort can be made using the following criteria:
    - The transaction which have the fewest locks
    - The transaction that has done the least work
    - The transaction that is farthest from completion
  - **How far to roll back a transaction back:** Having decided to abort a particular transaction ,we have to decide how far to roll the transaction back. Clearly, undoing all the changes made by a transaction is the simplest solution, although not necessarily the most efficient It may be possible to resolve the deadlock by rolling back only part of the transaction.
  - **Avoiding starvation**: Starvation occurs when a transaction can't proceed for an indefinite period of time while other transactions in the system continue normally. This may occur if the waiting scheme for locked items is unfair, giving priority to some transactions over others. Starvation occurs when the same transaction is always chosen as the victim, and the transaction can never complete. Starvation is very similar to live lock(left in a wait state indefinitely, unable to acquire any new locks, although the DBMS is not in deadlock), which occurs when the concurrency control protocol never selects a particular transaction that is waiting for a lock.

  - **Solutions to starvation:**
    - **1st Scheme :First Come First Served(FCFS) queue:** Transactions are enabled to lock an item in the order in which they originally requested the lock.

- **2nd Scheme:** It allows some transactions to have priority over others but increases the priority of a transaction the no longer it waits, until it eventually gets the highest priority and proceeds.

- **Avoidance of starvation:**
  - The DBMS can avoid starvation by storing a count of the number of times a transaction has been selected as the victim and using a different selection criterion once this count reaches some upper point.
  - The wait-die and wound-wait schemes avoid starvation.

Database recovery:

- It is the process of restoring the database to a correct state in the event of a failure.
- A typical strategy for recovery may be summarized informally as follows:

(i)If there is extensive damage to a wide portion of the database due to catastrophic failures, such as disk crash , the recovery method restores a past copy of the database that was backup to archival storage(tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure

(ii) When the database is not physically damaged , but has become inconsistent due to non catastrophic failures, the strategy is to reverse any changes that caused the inconsistency by undoing some operations. It may be necessary to redo some operations in order to restore a consistent state of the database. In this case , we do not need a complete archival copy of the database. Rather , the entries kept in the online system log are consulted during recovery

Techniques for recovery from non catastrophic transaction failures

- Deferred update(No undo/redo algorithm)
- Immediate update(Undo/redo algorithm)
- Shadow paging
- ARIES recovery algorithm

Deferred update(No undo/redo algorithm)

- This technique does not physically update the database on disk until after a transaction reaches its commit point; then the updates are recorded in the database.
- Before reaching commit, all transaction updates are recorded in the local transaction workspace (or buffers).
- During the commit, the updates are first recorded persistently in the log and then written to the database.
- If a transaction fails before reaching its commit point, it will not have changed the database in any way, so UNDO is needed.
- It may be necessary to REDO the effect of the operations of a committed transaction from the log, because their effect may not yet have been recorded in the database

Immediate update(Undo/redo algorithm)

- In this technique , the database may be updated by some operations of a transaction before the transaction reaches its commit point.
- These operations are typically recorded in the log on disk by force writing before they are applied to the database , making recovery still possible.
- If a transaction fails after recording some changes in the database but before reaching its commit point, the effect of its operations on the database must be undone i.e. the transactions must be rolled back.
- In the general case of immediate update, both undo and redo may be required during recovery

Shadow paging (Lorie, 1977)

- It is an alternative to the log-based recovery schemes (Deferred and Immediate update)

- This scheme maintains two page tables during the life of a transaction:
    - a)A current page table
    - b)A shadow page table
- When the transaction starts, the two-page tables are the same.
- The shadow page table is never changed thereafter, and is used to restore the database in the event of a system failure.
- During the transaction, the current page table is used to record all updates to the database.
- When the transaction completes, the current page table becomes the shadow page table
- Advantages of shadow paging over the log based schemes
    - The overhead of maintaining the log file is eliminated, and recovery is significantly faster since there is no need for undo or redo operations
- Disadvantages
    - Data fragmentation
    - The need for periodic garbage collection to reclaim inaccessible blocks

ARIES recovery algorithm

- ARIES uses a steal/no-force approach for writing , and it is based on three concepts:
    - Write-ahead logging
    - Repeated history during redo
    - Logging changes during undo
- Write-ahead logging

  The recovery mechanism must ensure that the BFIM(Before image-the old value of the data item before updating) of the data item is recorded in the appropriate log entry and that the log entry is flushed to disk before the BFIM is overwritten with the AFIM(after image-the new value after updating) in the database on disk.
- Repeated history during redo

  The ARIES will retrace all actions of te database system prior to the crash to reconstruct the database state when the crash occurred. The transactions that were uncommitted at the time of the crash(active transactions) are undone.
- Logging during undo

  It will prevent ARIES from repeating the completed undo operations if a failure occurs during recovery, which causes a restart of the recovery process.

ARIES recovery procedure: It consists of three main steps :

- Analysis
- REDO
- UNDO

Analysis step: This step identifies the dirty (updated) pages in the buffer and the set of transactions active at the time of the crash.

REDO phase: It actually reapplies updates from the log to the database. The REDO operation is applied only to the committed transaction.

UNDO phase: During this phase, the log is scanned backward and the operations of transactions that were active at the time of the crash are undone in reverse order.

The information's needed for ARIES to accomplish its recovery procedure includes:

- The log
- The transaction table
- The dirty page table
- Check pointing

The log : In ARIES, every log record has an associated log sequence number (LSN) that is monotonically increasing and indicate the address of the log record on disk. Each LSN corresponds to a specific change(action) of some transaction. Each data page will store the LSN of the latest log record corresponding to a change for that page. A log record is written for any of the following actions: updating

a page(write), committing a transaction(commit), aborting a transaction (abort), undoing an update(undo), and ending a transaction(end).

Transaction table :It contains an entry for each active transaction, with information such as the transaction id, transaction status, and the LSN of the most recent log record for the transaction.

The dirty page table: It contains an entry for each dirty page in the buffer, which includes the page id and the LSN corresponding to the earliest update to that page. The transaction table and dirty page table are needed for efficient recovery maintained by the transaction manager. When a crash occurs, these tables are rebuilt in the analysis phase of recovery.

Check pointing:

- It is the point of synchronization between the database and the transaction log file. All buffers are force-written to secondary storage.
- Check points are scheduled at predetermined intervals and involve the following operations:
- Writing all log records in main memory to secondary storage
- Writing a checkpoint record to the log file. This record contains the identifiers of all the transactions that are active at the point of check point
- Check pointing in ARIES consists of the following:
    - Writing a begin_checkpoint record to the log.
    - Writing an end_checkpoint record to log.
    - Writing the LSN of the begin_checkpoint record to a special file.
    - The special file is accessed during recovery to locate the last check point information.
    - With the end_checkpoint record, the contents of both the transaction table and dirty page table are appended to the end of the log. To reduce the cost, fuzzy check pointing is used so that the DBMS can continue to execute transactions during chekpointing

Database Security: It is the mechanism that protect the database against intentional or accidental threats.

Threat: It is a situation or event , whether intentional or accidental , that may adversely affect a system and consequently the organization.

Threats to databases: Threats to databases result in the loss or degradation of some or all of the following commonly accepted security goals: integrity, availability and confidentiality.

Loss of integrity: Integrity is lost if unauthorized changes are made to the data by either intentional or accidental acts. If the loss of system or data integrity is not corrected , continuous use of the contaminated system or corrupted data could result in inaccuracy, fraud , or erroneous decisions .**Database integrity** refers to the requirement that information be protected from improper modification. Modification of data includes creation, insertion, modification, changing the status of data , and deletion.

Loss of availability: Database availability refers to making objects available to human user or a program to which they have a legitimate right.

Loss of confidentiality: Database confidentiality refers to the protection of data from unauthorized disclosure. Unauthorized , unanticipated , or unintentional disclosure could result in loss of public confidence , embarrassment, or legal action against the organization.

Control measures: To protect database against all the thrats , it is common to implement four kinds of control measures:

- Access control
- Inference control
- Flow control
- Encryption

Access control:

- The security mechanism of a DBMS must include provisions for restricting access to the database system as a whole. This function is called access control and is handled by creating user accounts and passwords to control the login process by the DBMS.
- The DBA is the central authority for managing a database system.
- The DBA's responsibilities include granting privileges to users who need to use the system and classifying users and data in accordance with the policy of the organization.

- The DBA has a DBA account in the DBMS, sometimes called a system or supervisor account, which provides powerful capabilities that are not made available to regular database accounts and users.
- DBA-privileged commands include commands for granting and revoking privileges to individual accounts, users, or user groups and for performing the following types of actions:
    - Account creation: This action creates a new account and password for a user or group of users to enable access to the DBMS.
    - Privilege granting: This action permits the DBA to grant certain privileges to certain accounts.
    - Privilege revocation: This action permits the DBA to revoke(cancel) certain privileges that were previously given to certain accounts.
    - Security level assignment: This action consists of assigning user accounts to the appropriate security classification level.
- The DBA is responsible for the overall security of the database system

Inference control

- Security for statistical databases must ensure that information about individuals can't be accessed.
- It is sometimes possible to deduce or infer certain facts concerning individuals from queries that involve only summary statistics on groups; consequently, this must not be permitted either.This problem, called **statistical database security.** The corresponding control measures are called inference control measures.
- The possibility of inferring individual information from statistical queries is reduced if no statistical queries are permitted whenever the number of tuples in the population specified by the selection condition falls below some threshold.
- Another technique for prohibiting retrieval of individual information is to prohibit sequence of queries that refer repeatedly to the same population of tuples.
- Another technique is partitioning the database. Partitioning implies that records are stored in groups of some minimum size; queries can refer to any complete group or set of groups, but never to subsets of records within a group.

Flow control

- It prevents information from flowing in such a way that it reaches unauthorized users.
- Channels that are pathways for information to flow implicitly in ways that violate the security policy of an organization are called **covert channels**

Encryption

- The encoding of the data by special algorithm that renders the data unreadable by any algorithm without the decryption key.
- Encryption can be be used to provide additional protection for sensitive portions of a database as well.
- An unauthorized user who accesses encoded data will have difficulty in deciphering it, but authorized users are given decoding or decryption algorithm(or keys) to decipher the data.

Authorization

- It is the granting of a right or privilege that enables a subject to have legitimate access to a system or a system's object

Authentication

- It is a mechanism that determines whether a user is who he or she claims to be.

Database security and authorization subsystem

- A DBMS typically includes a database security and authorization subsystem that is responsible for ensuring the security of portions of a database against unauthorized access. It is now customary to refer to two types of database security mechanisms:
    - **Discretionary security mechanisms**: These are used to grant privileges to users, including the capability to access specific data files , records, or fields in a specific mode(such as read, insert, delete, or update).

- **Mandatory security mechanisms**: These are used to enforce multilevel security by classifying the data and users into various security classes or levels and then implementing the appropriate security policy of the organization.

An Introduction to Data Mining

Why Data Mining:

- Credit ratings/targeted marketing:
  - Given a database of 100,000 names, which persons are the least likely to default on their credit cards?
  - Identify likely responders to sales promotions
- Fraud detection
  - Which types of transactions are likely to be fraudulent, given the demographics and transactional history of a particular customer?
- Customer relationship management:
  - Which of my customers are likely to be the most loyal, and which are most likely to leave for a competitor?
- Data Mining helps extract such information

Definition of data mining

- Data mining is the extraction or mining of knowledge from large amounts of data
  Example: Mining (extracting) of golds from rocks or sand is known as gold mining rather than rock or sand mining
- It is a step in the knowledge discovery process, which is the process of discovering interesting knowledge from large amounts of data stored either in databases, data ware houses, or other information repositories
- It is the process of semi-automatically analyzing large databases to find patterns that are:
  - valid: hold on new data with some certainty
  - novel: non-obvious to the system
  - useful: should be possible to act on the item
  - understandable: humans should be able to interpret the pattern
- Data mining is also known as Knowledge Discovery in Databases (KDD**)**

Definition of Knowledge Discovery in Databases (KDD): Knowledge Discovery in Data is the *non-trivial* process of identifying *valid, novel,* potentially *useful* and ultimately *understandable patterns* in data.

Knowledge discovery in databases (KDD): KDD is a process of consisting of an iterative sequence of the following steps:

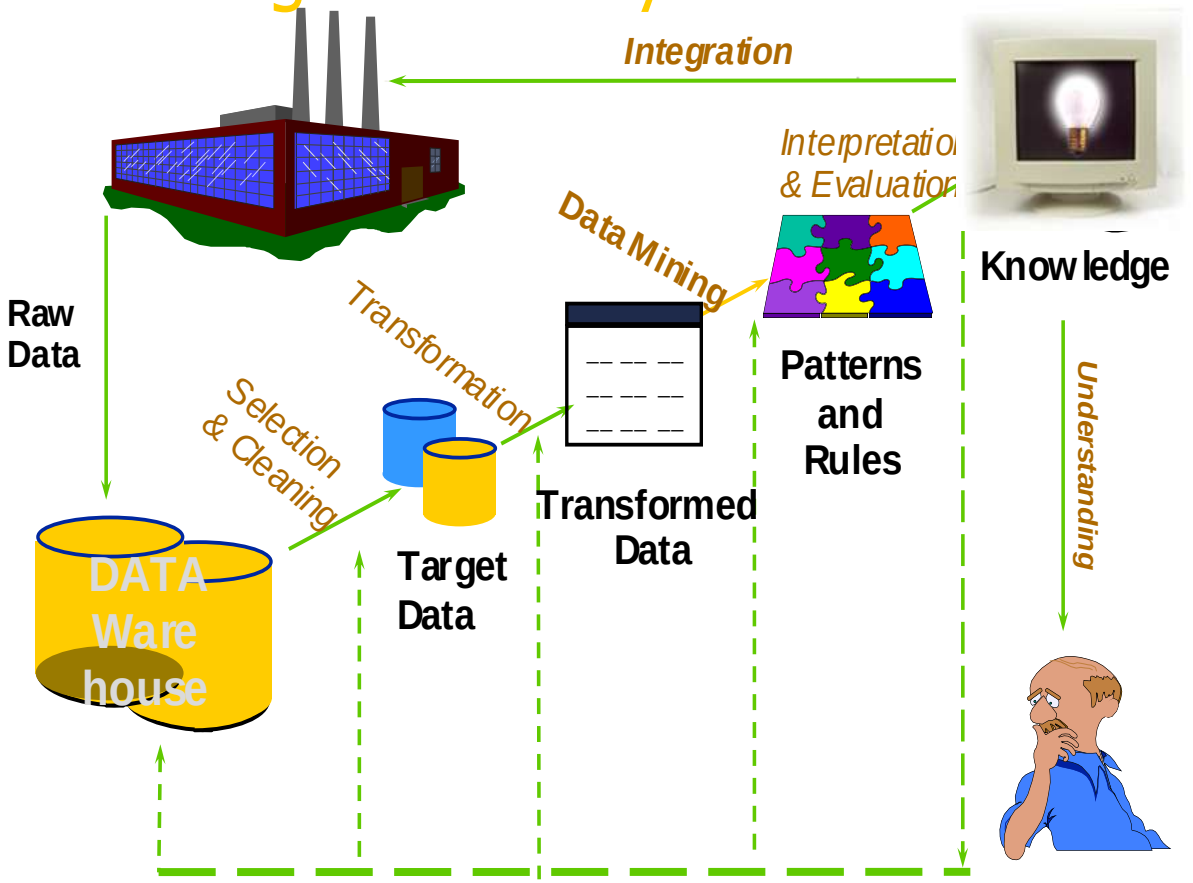
- Data cleaning: Noise and inconsistent data are removed.
- Data integration: Multiple data sources are combined.
- Data selection: Data relevant to the analysis task are retrieved from the database.
- Data transformation: Data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance.
- Data mining: An essential process where intelligent methods are applied in order to extract data patterns.
- Pattern evaluation: Identifies the truly interesting patterns representing knowledge based on some interestingness measures.
- Knowledge presentation: Visualization and knowledge representation techniques are used to present the mined knowledge to the user.

:

Application Areas

Industry	Application
Finance	Credit Card Analysis
Insurance	Claims, Fraud Analysis
Telecommunication	Call record analysis
Transport	Logistics management
Consumer goods	promotion analysis
Data Service providers	Value added data
Utilities	Power usage analysis

Knowledge Discovery Process



Data Mining in Use

- The US Government uses Data Mining to track fraud
- A Supermarket becomes an information broker
- Basketball teams use it to track game strategy
- Cross Selling
- Target Marketing
- Holding on to Good Customers
- Weeding out Bad Customers

Data mining functionalities/techniques

- Concept/Class description
- Association analysis
- Classification and prediction
- Cluster analysis
- Outlier analysis
- Evolution analysis

Concept/Class description

Describes individual classes and concepts in summarized, concise, and yet precise term

Example: Classes of items for sale in all electronics store include Computers & Printers

Association analysis

X=>Y is interpreted as "Database tuples that satisfy the conditions in X are also likely to satisfy the conditions in Y"

Example: Contains (T, "Computer")=>contains (T,"S/W") where T=transaction

Age(X,"20…29") ^income(X,"20K…29K") =>buys (X, "CD player") where X=customer

Single and multidimensional association rule

Association rule that contains more than one attribute or predicate is called multidimensional association rule. Association rule that contains a single attribute or predicate is called single dimensional association rule.

Classification and prediction

Predicting the class level of data objects is known as classification, with known class levels of data objects(training data). Prediction is the data value prediction. Relevance analysis attempts to identify attributes that do not contribute to the classification or prediction process

Cluster analysis

- Analyzes data objects without consulting a known class label.
- The class levels are not present in the training data.
- Used to generate class labels.
- The data objects are clustered or grouped based on the principle of maximizing the intra class similarity and minimizing the interclass similarity.

Outlier analysis

- Outliers: are the data objects that do not comply with the general behavior or model of the data
- Most data mining methods discard outliers as noise or exceptions
- Outlier mining is the analysis of outlier data

Evolution analysis

- Data evolution analysis describes and models regularities or trends for objects whose behavior changes over time
- Distinct features of evolution analysis include time-series data analysis , sequence or periodicity pattern matching, and similarity-based data analysis

Data Warehouse: A data warehouse is a subject-oriented, integrated, time-varying, and non-volatile collection of data that is used primarily in organizational decision making.

Subject -oriented

- The warehouse is organized around the major subjects of the enterprise (such as customers, products, and sales) rather than the major application areas(such as customer invoicing, stock control, and product sales).
- This is reflected in the need to store decision-support data rather than application-oriented data

Integrated nature of data

- Integrated means bringing together data of multiple , dissimilar operational sources
- Identifies and defines the organizational key data items uniquely
- Identifies the logical relationships between them ensuring organization wide consistency in terms of:
  - Data naming and definition
  - Encoding structure
  - Measurement of variables

Time-variant (historical) nature of data

- Because data in the warehouse is only accurate and valid at some point in time or over some time interval
- Historical in nature
- Contains data that is date-stamped
- Operational data changes over a shorter time period

Non-volatile (Static nature) nature of data

- As the data is not updated in real time but refreshed from operational systems on a regular basis
- Data warehouse data is loaded on to the data warehouse database and
- Is subsequently scanned and used , but is not updated in the same classical sense as operational system's data which is updated through the transaction processing cycles

Characteristics of data ware houses:

- Multidimensional conceptual view
- Generic dimensionality
- Unlimited dimensions and aggregation levels
- Unrestricted cross-dimensional operations

- Dynamic spare matrix handling
- Client/Server architecture
- Multi-user support
- Accessibility
- Transparency
- Intuitive data manipulation
- Consistent reporting performance
- Flexible reporting

Advantages (benefits) of data ware house

- Potential high returns on investment
- Competitive advantage
- Increased productivity of corporate decision making

Disadvantages (limitations/ problems) of data ware house

- Underestimation of resources for data loading
- Hidden problems with source systems
- Increased end-user demands
- Data homogenizations
- High demand for resources
- Data ownership
- High maintenance
- Long-duration projects
- Complexity of integration

Typical applications of data ware house

- Online analytical processing(OLAP)
- Decision-support system(DSS)
- Data mining
    - **Online analytical processing(OLAP):** It is used to describe the analysis of complex data from the data ware house. In the hands of skilled knowledge workers, OLAP tools use distributed computing capabilities for analysis that require more storage and processing power than can be economically and efficiently located on an individual desktop.

      Or
    - It is the dynamic synthesis, analysis, and consolidation of large volumes of multi-dimensional data.
        - **Applications of OLAP**
            - Finance
            - Sales
            - Marketing
            - Manufacturing
    - **Online transaction processing(OLTP):**It includes insertions, updates, and deletions, while also supporting information query requirements.
    - **Decision support systems(DSS):** It is also known as Executive Information Systems(EIS) that support an organization's leading decision makers with higher level data for complex and important decisions .
    - **Data mining:** It is used for knowledge discovery, the process of searching data for unanticipated new knowledge
- **OLAP Benefits:**
    - Increased productivity of business end-users, IT developers, and consequently the entire organization.

- Reduced backlog of applications development for IT staff by making end-users self-sufficient enough to make their own schema changes and build their own models.
- Retention of organizational control over the integrity of corporate data as OLAP applications are dependent on data warehouses and OLTP systems to refresh their source level data.
- Reduced query drag and network traffic on OLTP systems or on the data ware house.
- Improved potential revenue and profitability by enabling the organization to respond more quickly to market demands.

Database systems and the internet

- Many electronic commerce(e-commerce)and other internet applications provide web interfaces to access information stored in one or more databases. These databases are often referred to as **data sources.**
- It is common to use two-tier and three-tier client/server architectures for internet applications.
- **Internet**: It is the worldwide collection of interconnected computer networks.
- **Intranet**: It is a website or group of sites belonging to an organization, accessible only by the numbers of the organization.
- **Extranet**: It is an intranet that is potentially accessible to authorized outsiders.
- **E-Commerce**: Customers can place and pay for orders via the business web site.
- **E-business:** Complete integration of internet technology into the economic infrastructure of the business

Search engines

Search engine is a program that searches documents for specified keywords and returns a list of the documents where the keywords were found. Although search engine is really a general class of programs , the term is often used to specifically describe systems like Google, Alta Vista and Excite that enable users to search for documents on the world wide web and USENET newsgroups. Typically, a search engine works by sending out a spider to fetch as many documents as possible. Another program, called an indexer, then reads these documents and create an index based on the words contained in each document. Each search engine uses a proprietary algorithm to create its indices such that, ideally, only meaningful results are returned for each query.

Semi-structured data model:

- **Semi-structured data:** It is the data that may be irregular or incomplete and have a structure that may change rapidly or unpredictably.
- **Structured data:** The information stored in the database is known as structured data because it is represented in a strict format i.e. tabular.
- **Unstructured data:** There is a very limited indication of the type of data.
- **Example:** (1) A text document that contains information within it.
  (2)Web pages in HTML that contains some data

Data model for semi-structured data-Object Exchange Model(OEM)

- Semi-structured data may be displayed as directed graph.
- This model somewhat resembles the object model in its ability to represent complex objects and nested structures.
- The labels or tags on the directed edges represent the schema names: the name of attributes, object types(or entity types or classes), and relationships.
- The internal nodes represent individual objects or composite attributes.
- The leaf nodes represent actual data values of simple(atomic) attributes.
- The figure below illustrates this.

Extensible Markup Language(XML) and Web databases

- It is a meta-language (a language for describing other languages) that enables designer to create their own customized tags to provide functionality not available with HTML.
- XML is a restricted version of standard generalized markup language (SGML) designed especially for web documents.

Document type definitions (DTD): It defines the valid syntax of an XML document.

Advantages of XML

- Simplicity
- Open standard and platform/ vender- independent
- Extensibility
- Reuse
- Separation of content and presentation
- Improved load balancing
- Support for the integration of data from multiple sources
- Ability to describe data from a wide variety of applications
- More advanced search engines
- New opportunities

Object and Object relational databases

Object oriented data model(OODM): It is a logical data model that captures the semantics of objects supported in object oriented programming(OOP).

Object Oriented Database(OODB): It is a persistent and sharable collection of objects defined by an OODM.

Object Oriented DBMS(OODBMS): The manager of an OODB is called as OODBMS.

Object relational DBMS(ORDBMS): ORDBMS consciously try to add OODBMS features to an RDBMS.

Note: OODBMSs and ORDBMSs both support user-defined ADTs, structured types, object identity and reference types **,** and inheritance , both support a query language for manipulating collection types. ORDBMSs support an extended form of SQL, and OODBMSs support ODL/OQL. Both support concurrency and recovery.

Object orientation = Abstract data types(ADT) + Inheritance+ Object identity

OODBMS= Object orientation +Database capabilities

RDBMS traditional business applications:

- Order processing
- Inventory control
- Banking
- Airline reservation

Weakness of RDBMS: leads to development of OODBMS, which are:

- Poor representation of real world entities
- Semantic overloading
- Poor support for integrity and enterprise constraints
- Homogenous data structure.
- Limited operations
- Difficulty handling recursive queries
- Impedance mismatch
- Concurrency, schema changes, and poor navigational access

Advanced database applications: The following new applications can't be easily designed in RDBMS due weaknesses of RDBMS :

- Computer Aided Design(CAD)
- Computer Aided Manufacturing (CAM)
- Computer Aided Software Engineering(CASE)
- Network management system

- Office information system and multimedia system
- Digital publishing
- Geographical information system(GIS)
- Interactive and dynamic web sites

Main concepts used in object oriented databases

- **Object identity**: Objects have unique identities that are independent of their attribute values.
- **Type constructors:** Complex object structures can be constructed by recursively applying a set of basic constructors, such as tuple, set, list, and bag.
- **Encapsulation of operations:** Both the object structures and the operations that can be applied to objects are included in the object class definitions.
- Programming language compatibility: Both persistent and transient objects are handled seamlessly. Objects are made persistent by being attached to a persistent collection or by explicit naming.
- **Type hierarchies and inheritance**: Object types can be specified by using a type hierarchy, which allows the inheritance of both attributes and methods of previously defined types. Multiple inheritance is allowed in some methods.
- **Extents**: All persistent objects of a particular type can be stored in an extent. Extents corresponding to a type hierarchy have set/subset constraints forced on them.
- **Support for complex objects**: Both structured and unstructured complex objects cn be stored and manipulated.

Advantages of OODBMS

- Enriched modeling capabilities
- Extensibility
- Removal of impedance mismatch
- More expressive query language
- Support for schema evolution
- Support for long duration transactions
- Applicability to advanced database applications
- Improved performance

Disadvantages of OODBMS

- Lack of universal data model
- Lack of experience
- Lack of standards
- Competition
- Query optimization compromises encapsulation
- Locking at object level may impact performance
- Complexity
- Lack of support for views and security

Advantages of ORDBMS

- Reuse and sharing
- Increased productivity
- Use of experience in developing RDBMS

Disadvantages of ORDBMS

- **Complexity**: The provision of the functionality that is expected of a good ORDBMS makes the ORDBMS an extremely complex piece of software. Database designers, developers, database administrators and end-users must understand this functionality to take full advantage of it. Failure to understand the system can lead to bad design decisions, which can have serious consequences for an organization.
- **Cost**: The cost of ORDBMS varies significantly, depending on the environment and functionality provided. There is also the recent annual maintenance cost.

- ORDBMS vendors are attempting to portray object models as extensions to the relational model with some additional complexity. This potentially misses the point of object orientation highlighting the large semantic gap between these two technologies.

Distributed Database System?

A distributed database (DDB) is a collection of multiple, *logically interrelated* databases distributed over a *computer network.*

Distributed database management system (DDBMS)

A distributed database management system (D–DBMS) is the software that manages the DDB and provides an access mechanism that makes this distribution transparent to the users.
Distributed database system (DDBS) = DDB + D–DBMS

Distributed Computing

A number of autonomous processing elements (not necessarily homogeneous) that are interconnected by a computer network and that cooperate in performing their assigned tasks.

What is distributed …

- Processing logic/ Processing elements
- Functions
- Data
- Control

Characteristics of DDBMS

- A collection of logically related shared data
- The data is splitted into a number of fragments
- Fragments may be replicated
- Fragments/replicas are allocated to sites
- The sites are linked by a communication network
- The data at each site is under the control of a DBMS
- The DBMS at each site can handle local applications , autonomously
- Each DBMS participates in at least one global application

Local and Global applications

- Users access the distributed database via applications. Applications are classified as those that do not require data from other sites(local applications) and that do require data from other sites(global applications)

Advantages DDBMS

- Reflects organizational structure
- Improved share ability and local autonomy
- Improved availability
- Improved reliability
- Improved performance
- Economics
- Modular growth

Disadvantages of DDBMS

- Complexity
- Cost
- Security
- Integrity control more difficult
- Lack of standards
- Lack of experience
- Database design more complex

What is not a DDBS?

- A timesharing computer system

- A loosely or tightly coupled multiprocessor system
- A database system which resides at one of the nodes of a network of computers - this is a centralized database on a network node

Applications of DDBS

- Manufacturing - especially multi-plant manufacturing
- Military command and control
- EFT
- Corporate MIS
- Airlines
- Hotel chains
- Any organization which has a decentralized organization structure

PARALLEL Vs. DISTRIBUTED TECHNOLOGY

Parallel Database Management Systems: DBMS developed using the following two types of multiprocessor system architectures are called parallel DBMS:

(i) **Shared memory(tightly coupled) architecture**: Multiple processors share secondary(disk) storage and also share primary memory.

(ii)Shared disk(loosely coupled) architecture: Multiple processors share secondary(disk) storage but each has their own primary memory.

Shared nothing architecture:

- In this architecture every processor has its own primary and secondary (disk) memory, no common memory exists, and the processors communicate over a high speed interconnection network(bus or switch).
- In shared nothing architecture, there is symmetry and homogeneity of nodes; this is not true for the distributed database environment where heterogeneity of h/w and OS at each node is very common
- Shared nothing architecture is also considered as an environment for parallel databases

Deductive databases:

- In a deductive database system , we typically specify rules through a declarative language.
- A declarative language is a language in which we specify what to achieve rather than how to achieve it.
- An inference engine (or deductive mechanism) within the system can deduce new facts from the database by interpreting these rules.
- The model used for deductive databases is closely related to the relational data model, and particularly to the domain relational calculus formalism.
- Deductive database is also related to the field of Logic Programming and Prolog language.
- The deductive database work based on logic has used Prolog as a starting point.
- A variation of Prolog called Datalog is used to define rules declaratively in conjunction with an existing set of relations, which are themselves treated as literals in the language.
- A deductive database uses two main types of specifications: facts and rules.
- Facts are specified in a manner similar to the way relations are specified, except that it is not necessary to include the attribute names.
- In a deductive database, the meaning of an attribute value in a tuple is determined solely by its position within the tuple.
- Rules are somewhat similar to relational views. They specify virtual relations that are not actually stored but that can be formed from the facts by applying inference mechanisms based on the rules specifications

Mobile databases:

- Mobile database is a database that is portable and physically separate from a centralized database server but is capable of communicating with that server from remote sites allowing the sharing of corporate data.

- With mobile databases, users have access to corporate data on their laptop, PDA, or other internet access device that is required for applications at remote sites.

The components of a mobile database environment: include:

- Corporate database server and DBMS that manages and stores the corporate data and provides corporate applications.
- Remote database and DBMS that manages and stores the mobile data and provides mobile applications .
- Mobile database platform that include Laptop, PDA, or other internet access devices.
- Two-way communication links between the corporate and mobile DBMS.

The additional functionality required for mobile database include the ability to:

- Communicate with the centralized database server through modes such as wireless or internet access.
- Replicate data on the centralized database server and mobile device.
- Synchronize data on the centralized database server and mobile device.
- Capture data from various sources such as internet.
- Manage data on the mobile device.
- Analyze data on the mobile device.
- Create customized mobile applications.

Multimedia databases:

- Multimedia databases provide features that allow users to store and query different types of multimedia information, which includes images(such as photos or drawings), video clips(such as movies, news reels, or home videos), audio clips(such as songs, phone messages, or speeches), and documents(such as books or articles).
- **Characteristics of multimedia sources:**
    - Image: An image is typically stored either in row form as asset of pixel or cell values, or in compressed form to save space.
    - Video: A video source is typically represented as a sequence of frames , where each frame is a still image.
    - Text/video source: A text/ video source is basically the full text of some article, book, or magazine.
    - Audio sources: It include stored recorded messages, such as speeches , class presentations, or even surveillance recording of phone messages.
- **Content based retrieval queries:** are those in which multimedia source is being retrieved based on its containing certain objects or activities.
- Multimedia database must ensure some model to organize and index the multimedia sources based on their contents.
- Identifying contents of multimedia sources is a difficult and time consuming task.
- **Two main approaches for identifying the contents of multimedia**
    - **Automatic analysis**
    - **Manual identification**
- **Automatic analysis:** Automatic analysis of the multimedia sources to identify certain mathematical characteristics of their contents. It uses different techniques depending on the type of multimedia source(image, text, video, or audio).
- **Manual identification:** Manual identification of the objects and activities of interest in each multimedia source and on using this information to index the sources. This approach can be applied to all the different multimedia sources, but it requires a manual preprocessing phase where a person has to scan each multimedia source to identify and catalog the objects and activities it contains so that they can be used to index these sources

Geographical Information System(GIS):

- GIS are used to collect, model, store, and analyze information describing physical properties of the geographical world.

- The scope of GIS broadly encompasses two types of data:
    - Spatial data
    - Non spatial data
- **Spatial data**: Spatial data originates from maps, digital images, administrative and political boundaries, roads, transportation networks; physical data such as rivers, soil characteristics, climate regions, land elevations
- **Non spatial data**: Non spatial data includes socio-economic data(like census counts), economic data, and sales or marketing information.
- **GIS applications**:
    - Cat graphic applications
    - Digital terrain modeling applications
    - Geographic objects applications
- **Cat graphic applications:**
    - Irrigation
    - Crop yield analysis
    - Land evaluation
    - Planning and facilities management
    - Landscape studies
    - Traffic pattern analysis
- **Digital terrain modeling applications**
    - Earth science studies
    - Civil engg. And military evaluation
    - Soil surveys
    - Air and water pollution studies
    - Flood control
    - Water resource management
- **Geographic objects applications**
    - Car navigation system
    - Geographic market analysis
    - Utility distribution and consumption
    - Consumer product and services economic analysis
- **Data management requirements of GIS:**
    - Data modeling and representation
    - Data analysis
    - Data integration
    - Data capture
- **Specific GIS data operations**
    - Interpolation
    - Interpretation
    - Proximity analysis
    - Raster image processing
    - Analysis of networks
    - Extensibility
    - Data quality control
    - Visualization
- **GIS software**: ARC-INFO