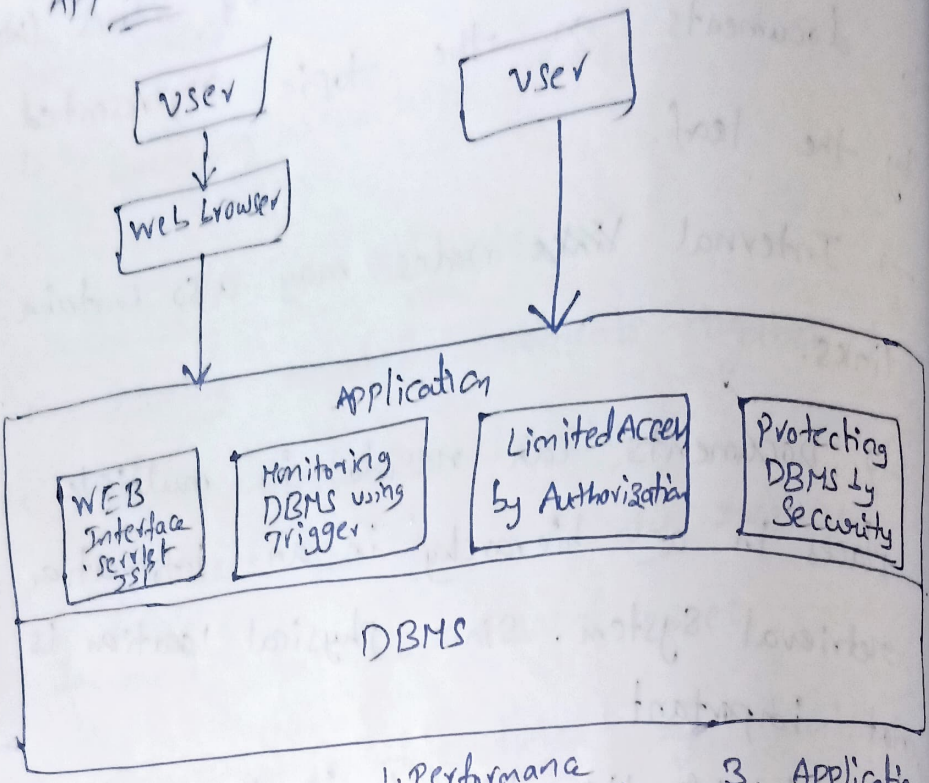
IV. Advanced Application Development

Application Design and Development



1. Performance
2. Standardization
3. Application Migration

Performance Tuning:-

Tuning the performance of a system involves adjusting various parameters and design choices to improve its performance for a specific application.

→ Adjusting various parameters and design choices to improve system performance for a specific application.

→ Tuning is best done by identifying bottlenecks and eliminating them.

→ Can tune a database system at 3 levels. ⟶ Tunable Parameters

1. Hardware 2. Database System Parameters
3. Higher level database design.

→ 1. Hardware : 1. add disks to speed up I/O.

→ add memory to increase buffer hits

→ Move to a faster Processor.

2. DB System Parameters.

1. Set buffer size to avoid paging of buffer
2. Set checkpointing intervals to limit log size.
3. System may have automatic tuning.

3. Higher level database design

1. Such as the schema, indices and transaction.

→ 1. Improving set orientation
2. Tuning of Bulk loads and updates
3. Location of Bottlenecks.

Bottlenecks

→ Performance of most systems (atleast before they are tuned) usually limited by performance of one or a few components these are called bottlenecks.

Eg 80% of the code may take up 20% of time & 20% of code takes up 80% of time.

Worth spending most time on 20% of code that take 80% of time.

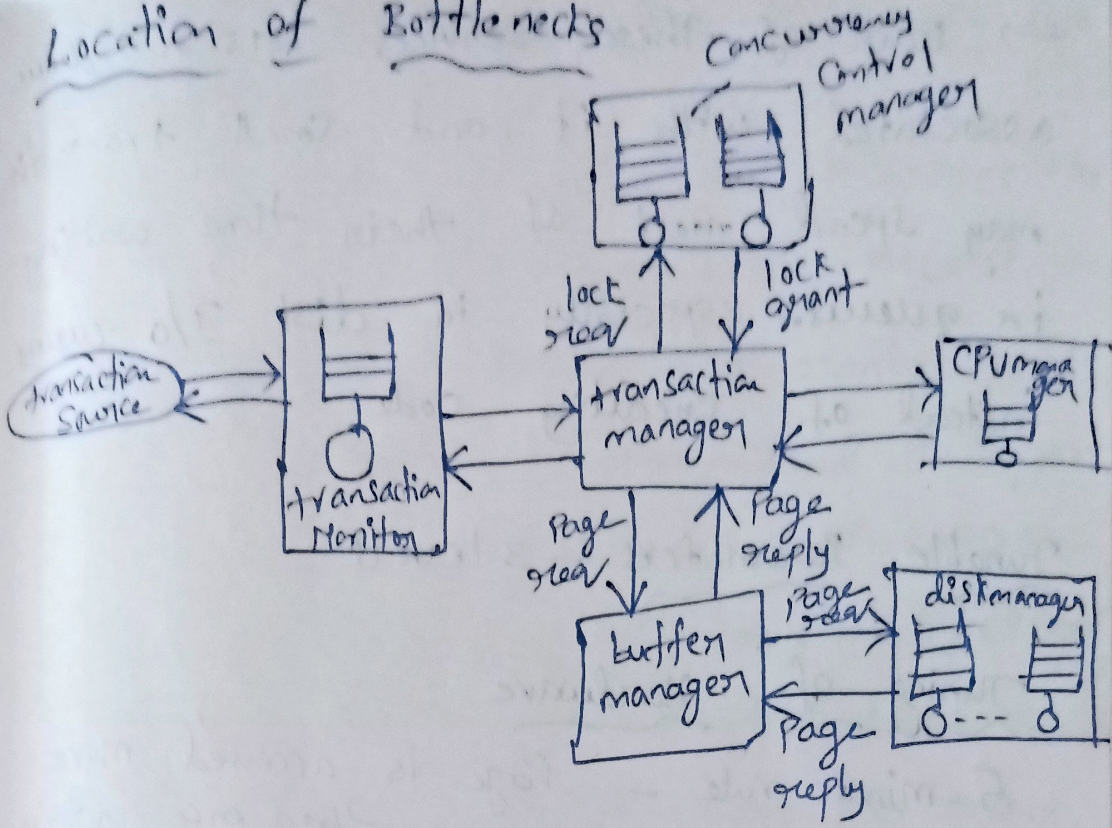
→ Finding bottlenecks is very important.

- Bottlenecks may be in hardware or in software.

  Eg disks are very busy, CPU is idle.

→ A Performance bottleneck occurs when the rate at which data is accessed cannot meet specified system requirements.

Location of Bottlenecks



Queues in database system

For simple programs, the time spent in each region of the code determines the overall execution time. However, database systems are much more complex, and can be modeled as queueing systems.

→ A transaction requests various services from database system, starting from entry into a server process, disk reads during execution, CPU cycles and locks for concurrency control.

→ Each of these services has a queue associated with it, and small transactions may spend most of their time waiting in queues - especially in disk I/O queues instead of executing code.

Tunable Parameters — 3 levels

Tuning of Hardware

5-minute rule — Page is accessed more than once in 5 min or more

1-minute rule.

↳ accessed more than once in 1 minute.

Question: which data to keep in memory

If a page is accessed n times per second, keeping it in memory saves

Break-even point: $n * \frac{\text{Price-Per-disk-drive}}{\text{accesses per second per Disk}} = \frac{\text{Price per megabyte of memory}}{\text{Pages per megabyte of memory.}}$

Another aspect of tuning is whether to use RAID, or RAID 5. The answer depends on how frequently the data are updated.

→ RAID 5 is much slower than RAID 1 on random writes.

→ RAID 5 requires 2 reads and 2 writes to execute a single random write request.

→ If an appln performs r random reads and w random writes per second to support a particular throughput rate, a RAID 5 implementation would require r + 4w I/O operations per second, whereas RAID 1 implementation would require r + 2w I/O operations per second.

→ Thumb rule: RAID 5 is fine when writes are rare and data is very large, but RAID 1 is preferable otherwise.

→ If you need more disks to handle I/O load, just mirror them since disk capacities these days are enormous.

Tuning the database design

Schema tuning:

Vertically partition relations to isolate the data that is accessed most often - only fetch needed information.

E.g. split account into two

(account_number, branch_name) and (account_number, balance)

Branch_name need not be fetched unless needed.

→ Improve performance by storing a denormalized relation, such as a join of instructor and department, where information about dept_name, building and budget is repeated for every instructor.

Tuning of Indices :-

→ Create appropriate indices to speedup slow queries / updates

→ speedup slow updates by removing excess indices (tradeoff between queries and updates)

→ Choose type of index (B-tree / hash) appropriate for most frequent types of queries

→ Choose which index to make clustered

most commercial database systems provide tuning wizards. These tools use the past history of queries and updates (called the workload) to estimate the effect of various on the execution time of the queries updates in workload.

Using Materialized views

→ Materialized views can help speedup certain queries, particularly aggregate queries.

→ In the case of immediate view maintenance if the updates of a transaction affect the materialized view, the materialized view must be updated as part of the same transaction.

→ In case of deferred view maintenance, the materialized view is updated later.

Tuning of Concurrent Transactions

Basic approaches to tuning of transactions

→ Improve set orientation.

→ Reduce lock contention.

→ Performance Simulation:- Using queing model useful to predict bottlenecks as well as the effects of tuning changes, even without access to real system.

→ model service time, but disregard details of service.

E.g: approximate disk read time by using an average disk read time.

→ Performance can be tuned in model and then replicated in real system.

E.g: no of disks, memory, algorithms etc.

Performance Benchmarks

Performance Benchmarks are suites of tasks that are used to quantify the performance of Software Systems.

Suites of tasks

→ Commonly used Performance measures:

Throughput

Response time

Availability or mean time to failure.

Suppose we have 2 tasks T_1 and T_2, that we measure the throughput of a system as the number of transactions of each type that run in a given amount of time - say, 1 sec.

System A runs T_1 at 99 transactions per second

T_2 at 1 transaction per second

Similarly let B runs both T_1 and T_2 at

50 transactions per second.

Work Loads has an equal mixture of two types of transactions.

If we took the average of the two pairs of numbers (that is 99 and 1, versus 50 and 50) it might appear that the two systems have equal performance.

It is wrong to take average in this fashion.

The correct way to average out the throughputs on different transaction types is to take the harmonic mean of the throughputs.

The harmonic mean of n throughputs $t_1, \ldots, t_n$ is defined as:

$$\frac{n}{\frac{1}{t_1} + \frac{1}{t_2} + \cdots + \frac{1}{t_n}}$$

harmonic mean for the throughputs in system A is 1.98. For system B, it is 50.

Database - Application classes

online transaction Processing (OLTP) and decision support including online analytical Processing (OLAP) are two know broad

classes of appl^n's handled by db system.

→ OLTP requires high concurrency and clever techniques to speed up commit processing, to support high rate of update transactions.

→ Decision Support appl^n's including online analytical processing or OLAP appl^n's requires good query evaluation algorithms and query optimization.

Benchmark Suites

Transaction Processing Council (TPC) benchmarks Suites are widely used.

TPC-A, TPC-B : simple OLTP appl^n modeling a bank teller appl^n with and without communication
↳ Not used anymore.

TPC-C : Complex OLTP appl^n modeling an inventory system current standard for OLTP benchmarking.

TPC-D : complex decision support appl^n

TPC-H (H for Ad hoc) based on TPC-D with some extra queries

TPC-R: (R for Reporting) same as TPC-H but without any restriction on materialized views and indices.

TPC-W (W for web) web commerce End-to-end web service benchmark modeling a web bookstore with combination of static and dynamically generated pages.

→ TPC Performance Measures

→ transaction-per-second. with constraints on response time

→ transactions-per-second-per-dollar

Two types of tests for TPC-H and TPC-R

1) Power-test: takes means to find queries per hour.

2) Throughput test: 22 queries.

3) Composite query per hour metric

4) Composite price/Performance metric

→ other Benchmarks — OODB 007 benchmark.

Other Issues in appln Development

Testing Applications

1) Testing of Programs involves designing a test suite – that is a collection of test cases.

→ Testing is not a one time process. since Programs evolve continuously and bugs may appear as an unintended consequence of a change in the Program. Such a bug is referred to as Program regression.

Regression testing involves running the Program on each test case in a test suite, and checking that the Program generates the expected test output.

→ Performance testing: Testing can also be used to ensure that an appln meets Performance requirements.

Appln Migration

Legacy systems are older generation appln systems that are in use by an organization but that the organization wishes to replace with a different appln.

→ Replacing legacy appln's with new appln system is often costly in terms of time and money, since they are often very large, consisting of millions of lines of code developed by teams of programmers over several decades.

→ Newer systems
→ wrapper — b/w relation db and legacy db
→ Reverse engineering → when an organization decide to replace a legacy system with a new system, it may follow a process called reverse Engineering which consists of going over the code of the legacy system to come up with schema design in the required data model.
→ big-bang approach
→ Re engineering.
→ Chicken & - little approach.

replaces the functionality of the legacy system.

Big-Bang approach: abruptly transitioning to a new system, which is called big-bang approach.

Excessive coding is required to support all the functionality that was provided by legacy system, the overall process is called re-engineering.

Standardization

→ Standards define the interface of the software system.

Eg: Standard define the Syntax and Semantics of a Programming language or ~~the~~ functions in an API.

data models (e.g. OODB / Relational DB)
object

→ Formal Standards are standards developed by a Standards organization / ANSI, ISO or by industry groups through Public Process.

ANSI — American National Standards Institute

ISO — International organization for Standardization.

→ De facto standards are generally accepted as standards without any formal process of recognition.

→ Standards defined by dominant vendors (IBM, Microsoft) often become de facto standards.

→ Anticipatory Standards lead the marketplace, defining features that vendors then implement in products (SQL-92 and SQL-1999).

→ Reactionary Standards attempt to standardize features that vendors have already implemented, possibly in different ways.

→ can be hard to convince vendors to change already implemented features
E.g, OODB Systems

SQL Standards

→ Since SQL is the most widely used query language, much work has been done on standardizing it.

→ ANSI and ISO with the various database vendors, have played a leading role.

→ SQL-86 standard was the initial version.

→ The IBM Systems Application Architecture (SAA) standard for sql was released in 19

→ Then SQL-89 & SQL-92.

→ Sql : 1999

→ Sql : 2003 version is the minor extension of the Sql : 1999.

→ Sql : 2003 standard was broken into several Parts

Part 1 : SQL / Framework provides an overview of the standard.

Part 2 : SQL / Foundation defines the basics of the standard types, schemas, tables, views, query and update stmts and so on.

Part 3 : SQL / CLI (Call level Interface) defines appln Program Interface to SQL.

Part 4 : SQL / PSM (Persistent stored Modules) defines extensions to SQL to make it Procedural

Part 9 : SQL / MED (Management of External data) define standards for interfacing an SQL system to external sources.

Part 10 : SQL / OLB (Object Language Bindings) defines standards for embedding SQL in Java.

SQL11: SQL) Schemata (Information and Definition schema) defines a standard catalog interface.

Part13: SQL/JRT (Java Routines and Types) defines standards for accessing routines and types in java.

Part14: SQL/XML defines XML-Related specification.

The missing numbers covers features such as temporal data, distributed transaction Processing and multimedia data for which there is as yet no agreement on the standard.

→ Database Connectivity standards: The ODBC standard is a widely used standard for communication between client appln and db system.

→ ODBC is based on the SQL CLI standard developed by X/Open industry consortium.

→ The X/Open consortium has also developed the X/open XA standards for interoperation of databases.

→ Object Database standards
→ XML-Based Standards } Refer Text Book.

ch
25: Spatial and Temporal data and Mobility

→ For most of the history of database, types of data stored in databases were relatively simple, limited support for datatypes in earlier version of SQL.

→ overtime, they developed increasing need for handling more complex datatypes such as temporal data, spatial data and multimedia data.

Motivation.

Some issues in dealing with each of these types of data.

Temporal data: Most database systems model the current state of the world for instance current customers, current students, current courses currently being offered.

→ In many appl's it is very important to store and retrieve information about past states. Historical information can be incorporated manually into a schema design

Spatial data:- Spatial data include geographic data such as maps and associated information & computer aided design data such as integrated circuit design or building designs.

Multimedia data: video, audio, Image

Mobile databases:- Mobile computing systems, such as laptops, notbook and high end cell phones that are connected to data base stations via wireless communication networks.

→ Time In Databases

→ A database models the state of some aspect of the real world outside itself. Typically databases models only one state - the current state - of the real world, and do not store information about Past states.

→ Databases that store information about states of the real world across time are called temporal databases.

→ When considering the issue of time in db systems, we must distinguish between time as measured by the system and time as observed in the real world.

→ The valid time for a fact is the set of time intervals during which the fact is true in the real world.

→ The transactime time for a fact is the time interval during which the fact is current within the db system.

→ This latter time is based on the transaction serialization order and is generated automatically by the system. Note that valid-time intervals, being a real-world concept, cannot be generated automatically and must be provided to the system.

→ A temporal relation is one where each tuple has an associated time when it is true, the time may be either valid time or transaction time.

→ Both validtime and transaction time can be stored, in which case the relation is said to be a bitemporal relation.

ID	name	dept_name	Salary	from	to
10101	Srinivas	CSE	61000	2007|1|1	2007|12|31
10101	srinivas	CSE	65000	2008|1|1	2008|12|31
12121	Ram	CSE(DS)	70000	2008|1|1	2008|12|31

A Temporal instructor relation

The above figure shows an example of temporal relation.

→ To simplify the representation, each tuple has only one time interval associated with it, thus a tuple is represented once for every disjoint time interval in which it is true.

→ Intervals are shown here as a pair of attributes from and to.

→ Note that some of the tuples have a "*" in the to time column. these asterick indicate that the tuple is true until the value in the to time column is changed, the tuple is true at the current time.

→ Although times are shown in textual form, they are stored internally in a more compact form, such as number of seconds since some fixed time on a fixed date (such as 12:00 AM, January 1, 1900) that can be translated back to the normal textual form.

→ Time specification in SQL

→ The SQL standard defines the types date, time, and time stamp.

→ The type date contains four digits for the year (1-9999), two digits for the month (1-12), and two digits for the date (1-31).

→ The type time contains two digits for the hour, two digits for the minute, and two digits for the second, plus optional fractional digits.

→ The type timestamp contains the of date and time, with six fractional digits for seconds field.

→ Since different places in the world have different local times, there is often a need for specifying the time zone along with the time.

→ The universal Coordinated Time (UTC) is a standard reference point for specifying time, with local times defined as offsets from UTC.

→ The standard abbreviation is UTC, rather than UCT, since it is an abbreviation of "Universal Coordinated Time" written in French as (universal temps coordonne).

→ SQL also supports two types time with time zone, and timestamp with time zone, which specify the time as a local time plus the offset of the local time from UTC.

→ SQL supports a type called interval which allows us to refer to a period of time such as "1 day" or "2 day and 5 hours", without specifying a particular time when this period starts.

Temporal Query Languages.

→ A db relation without temporal information is sometimes called a snapshot relation, since it reflects the state in a snapshot of the real world.

→ A temporal selection is a selection that involves the time attributes; a temporal projection is a projection where the tuples in the projection inherit their times from the tuples in the original relation.

→ A temporal join is a join, with the time of a tuple in the result being the intersection of the times of the times of the tuples from which it is derived.

→ If the times do not intersect, the tuple is removed from the result.

Spatial and Geographic Data

Spatial data support in databases is important for efficiently indexing, and querying of data on the basis of spatial locations.

→ For Example, suppose that we want to store a set of Polygons in a db and to query the database to find all Polygons that intersect a given Polygon. we can't use standard index structures, such as B-trees or hash trees, to answer such a query efficiently.

→ Efficient processing of the above query would require special-purpose index

→ Two Types of spatial data are particularly important

1) Computer-aided-design (CAD) data, which includes spatial information about how objects - such as buildings, cars, or aircraft are constructed.

→ other important examples of computer-aided-design databases are integrated-circuit and electronic-device layouts.

2) Geographic data: Such as road maps, land-usage maps, topographic elevation maps, Political maps showing boundaries, land-ownership maps, and so on.

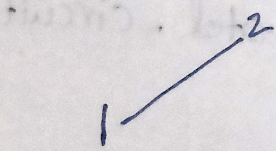
→ Geographic information systems are special purpose databases tailored for storing geographic data.

Representation of Geographic information

→ A line segment can be represented by the coordinates of its endpoints. E.g: in a map database, the two coordinates of point would be its latitude and longitude

→ A Polyline
  ↳ (also called a linestring) consists of connected sequence of line segments and can be represented by a list containing the coordinates of the end points of the segments in sequence.

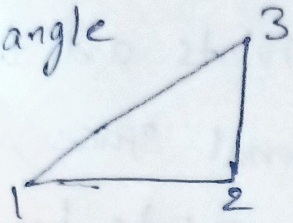
Object



representation

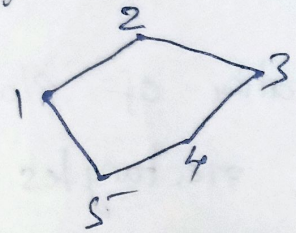
$\{(x_1, y_1), (x_2, y_2)\}$

triangle



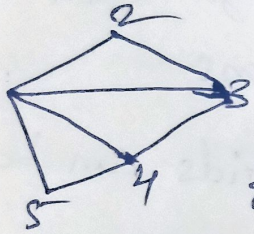
$\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$

Polygon



$\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)\}$

Polygon



$\{(x_1, y_1), (x_2, y_2), (x_3, y_3), ID\}$

$\{(x_1, y_1), (x_3, y_3), (x_4, y_4), ID\}$

$\{(x_1, y_1), (x_4, y_4), (x_5, y_5), ID\}$

Representation of geometric constructs

→ We can represent a polygon by listing its vertices in order.

→ The list of vertices specifies the boundary of a polygon region.

→ A polygon can be divided into set of triangles. This process is called triangulation, and any polygon can be triangulated.

→ The complex polygon can be given an identifier, each of the triangles into which it is divided carries the identifier of the polygon.

→ The representation of points and line segments in three-dimensional space, the only difference being that points have an extra z component.

→ Similarly the representation of planar figures - such as triangles, rectangles, and polygons - does not change much when we move to three dimensions.

→ Tetrahedrons and cuboids can be represented in the same way as triangles and rectangles.

Design Databases

→ Computer-aided-design (CAD) systems traditionally stored data in memory during editing or other processing, and wrote the data back to a file at the end of a session of editing.

→ The drawbacks of such a scheme include the cost of transforming data from one form to another.

→ For large designs, such as the design of a large-scale integrated circuit or the design of an entire airplane, it may be

impossible to hold the complete design in memory.

→ The objects stored in a design db are generally geometric objects.

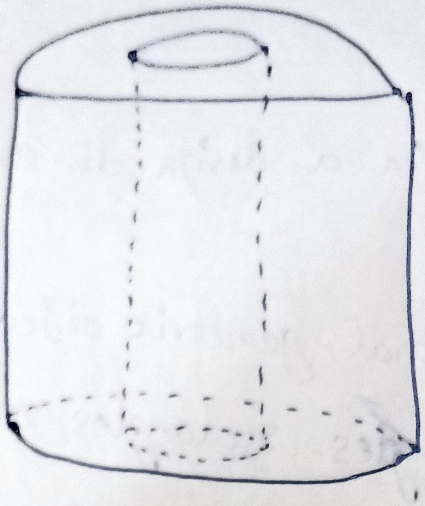
→ Simple, two-dimensional geometric objects include. Points, lines, triangles, rectangles, and, in general Polygons.

→ Complex two-dimensional objects can be formed from simple objects by means of unions, intersection, and difference operations.

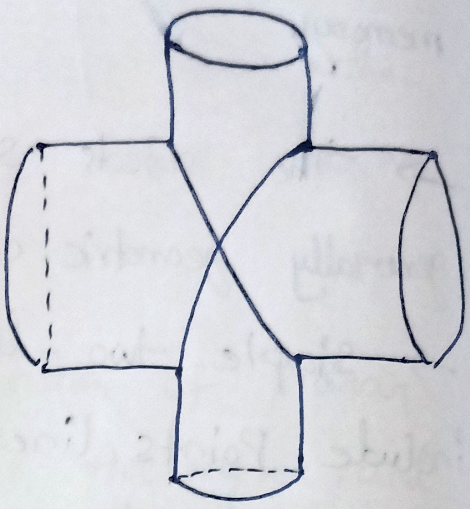
→ Similarly complex three-dimensional objects may be formed from simpler objects such as spheres, cylinders, and cuboids, by union, intersection and difference operations.

→ Three-dimensional surface may also be represented by wireframe models, which essentially model the surface as a set of simpler objects, such as line segments, triangles and rectangles.

→ Spatial-index structures are multidimensional dealing with two- and three dimensional data, rather than dealing with just the simple one-dimensional ordering provided by the B+-trees.



(a) Difference of cylinders



b) ~~Difference~~ Union of cylinders

Complex three-dimensional objects.

Geographic Data

→ Geographic data are spatial in nature, but differ from design data in certain ways.

→ Maps and Satellite images are typical examples of geographic data.

→ Maps may provide not only location information - about boundaries, rivers and roads.

→ But also much more detailed information associated with locations, such as elevation, soil-type, land usage, and annual rainfall

Appl^n's of Geographic Data

Geographic databases have a variety of uses,

including online map services, vehicle navigation systems, distribution-network information for public-service utilities such as telephone, electric-power, and water-supply systems, and land usage information for ecologists and planners.

→ Web based road map services form a very widely used appln of map data, used to generate online road maps of a desired region.

→ Road maps services also store information about roads and services.

→ Vehicle navigation systems are systems that are mounted in automobiles and provide road maps and trip-planning services.

→ Include a Global-Positioning system (GPS) unit, which uses information broadcast from GPS satellite to find the current location with an accuracy of tens of meters.

→ With such a system, a driver can never get lost - the GPS unit finds the location in terms of latitude, longitude and elevation and the navigation system can query the geographic db.

Geographic data can be categorized into two types

1) Raster data.

2) Vector data.

Raster data: Such data consist of bit maps or pixel maps, in two or more dimensions.

→ A typical example of a two-dimensional raster image is a satellite image of an area.

→ Raster data is often represented as tiles, each covering a fixed sized area.

→ A larger area can be displayed by displaying all the tiles that overlap with the area.

→ Raster data can be three-dimensional for eg, the temperature at different altitudes at different regions, again measured with the help of satellite.

Vector data: Vector data are constructed from basic geographic objects such as points, line segments, polylines, triangles and other polygons in two dimensions, and

cylinders, spheres, cuboids and other Polyhedrons in three dimensions.

→ In the context of geographic data, Points are usually represented by latitude and longitude.

→ Map data are often represented in vector format. Roads are often represented as Polylines.

→ Geographic information related to regions, such as annual rainfall, can be represented as an array - that is, in raster form.

→ Topographical information, that is information about the elevation (height) of each Point on a surface, can be represented in raster form.

→ Alternatively it can be represented in vector form by dividing the surface into Polygons covering regions of equal elevation.

→ The latter representation, called the Triangulated Irregular Network (TIN) representation is a compact representation which is particularly for generating three-dimensional views of an area.

Spatial Queries

There are a number of types of queries that involve spatial locations.

1) Nearness queries: request objects that lie near a specified location. A query to find all restaurants that lie within a given distance of a given point is an example of a nearness query.

→ The nearest-neighbour query requests the object that is nearest to a specified point.

Eg: we may want to find the nearest gasoline station. Note that this query does not have to specify a limit on the distance.

2) Region queries: deal with spatial regions. Such a query can ask for objects that lie partially or fully inside a specified region.

Eg: A query to find all retail shops within the geographic boundaries of a given town.

→ queries may also request intersections and unions of regions.

Eg: given region information, such as annual rainfall and population density, a query may request all regions with a low annual rainfall as well as a high population density.

Indexing of Spatial Data

→ Indices are required for efficient access to spatial data.

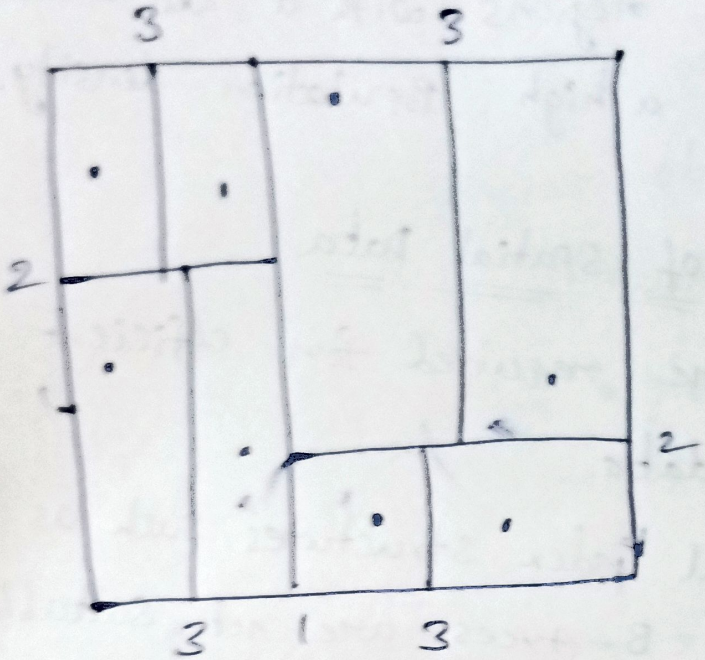
→ Traditional index structures, such as hash indices and B-trees are not suitable, since they deal only with one-dimensional data. Whereas spatial data are typically of two or more dimensions.

K-d trees (K-dimensional)

→ To understand how to index spatial data consisting of two or more dimensions, we consider first the indexing of points in one-dimensional data.

→ Tree structures, such as binary trees and B-trees, operate by successively dividing space into smaller parts.

→ Each internal node of a binary tree partitions a one-dimensional interval in two



Division of space by a k-d trees

→ Points that lie in the left partition go into the left subtree, points that lie in the right partition go into the right subtree.

→ In a balanced binary tree, the partition is chosen so that approximately one-half of the points stored in the subtree fall in each partition.

→ Similarly, each level of a B-tree splits a one-dimensional interval into multiple parts.

→ A tree structure called a k-d tree was one of the early structures used for

indexing in multiple dimensions.

→ Each level of a k-d tree partitions the space into two.

→ The partitioning is done along one dimension at the node at the top level of the tree along another dimension in nodes at the next level and so on, cycling through the dimensions.

→ Partitioning proceeds in such a way that, at each node, approximately one-half of the points stored in the subtree fall on one side and one-half fall on the other.

→ Partitioning stops when a node has less than a given maximum number of points.

→ The figure shows a set of points in two dimensional space, and a k-d tree representation of the set of points.

→ Each line corresponds to a node in the tree, and maximum no. of points in a leaf node has been set at 1.

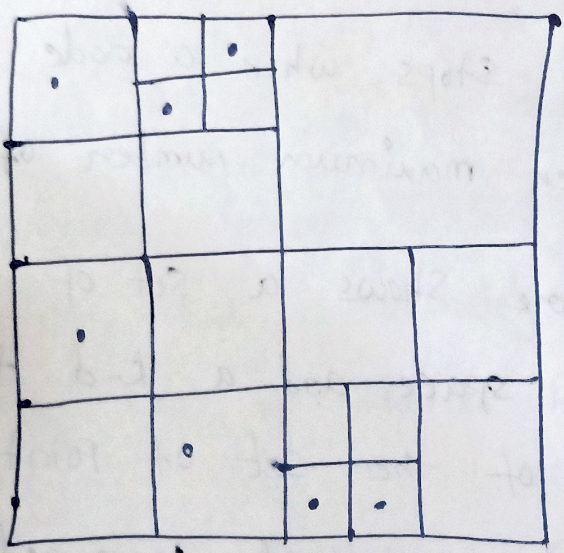
→ Each line in the figure (other than outside box) corresponds to a node in the

K-d tree.

→ The numbering of the lines in the figure indicates the level of the tree at which the corresponding node appears.

→ The K-d-B tree extends the k-d tree to allow multiple child nodes for each internal node, just as a B-tree extends a binary tree, to reduce the height of the tree. K-d-B trees are better suited for secondary storage than k-d trees.

Quadtrees



Division of Space by quadtree

An alternative representation for two-dimensional data is a quadtree.

The figure shows the division of space by a quadtree.

→ Each node of a quadtree is associated with a rectangular region of space.

→ The top node is associated with the entire target space.

→ Each non-leaf node in a quadtree divides its region into four equal-sized quadrants, and corresponding each such node has four child nodes corresponding to the four quadrants.

→ Leaf nodes have between zero and some fixed no of Points.

→ If the region corresponding to a node has more than the maximum no. of Points, child nodes are created for that node. The maximum number of Points in a leaf node is set to 1. This type of quadtree is called a PR quadtree.

→ PR quadtree, to indicate it stores Points, and that the division of space is divided based on regions, rather than the actual set of Points stored.

→ We can use region quadtree to store array (raster) information.

→ A node in a region quadtree is a leaf node if all the array values in the region that it covers are the same. otherwise, it is subdivided further into four children of equal area, and is therefore an internal node.

→ Each node in the region quadtree corresponds to a subarray of values. The subarrays corresponding to leaves either contain a single array element or have multiple array elements, all of which have the same value.

→ Indexing of line segments and polygons presents new problems. There are extensions of K-d trees and quadtrees.

R-Trees

→ A storage structure called an R-tree is useful for indexing of objects such as points, line segments, rectangles and other polygons.

→ An R-tree is a balanced tree structure with the indexed objects stored in leaf node, much like a B+-tree.

→ Instead of a range of values, a rectangular bounding box is associated with each tree node.

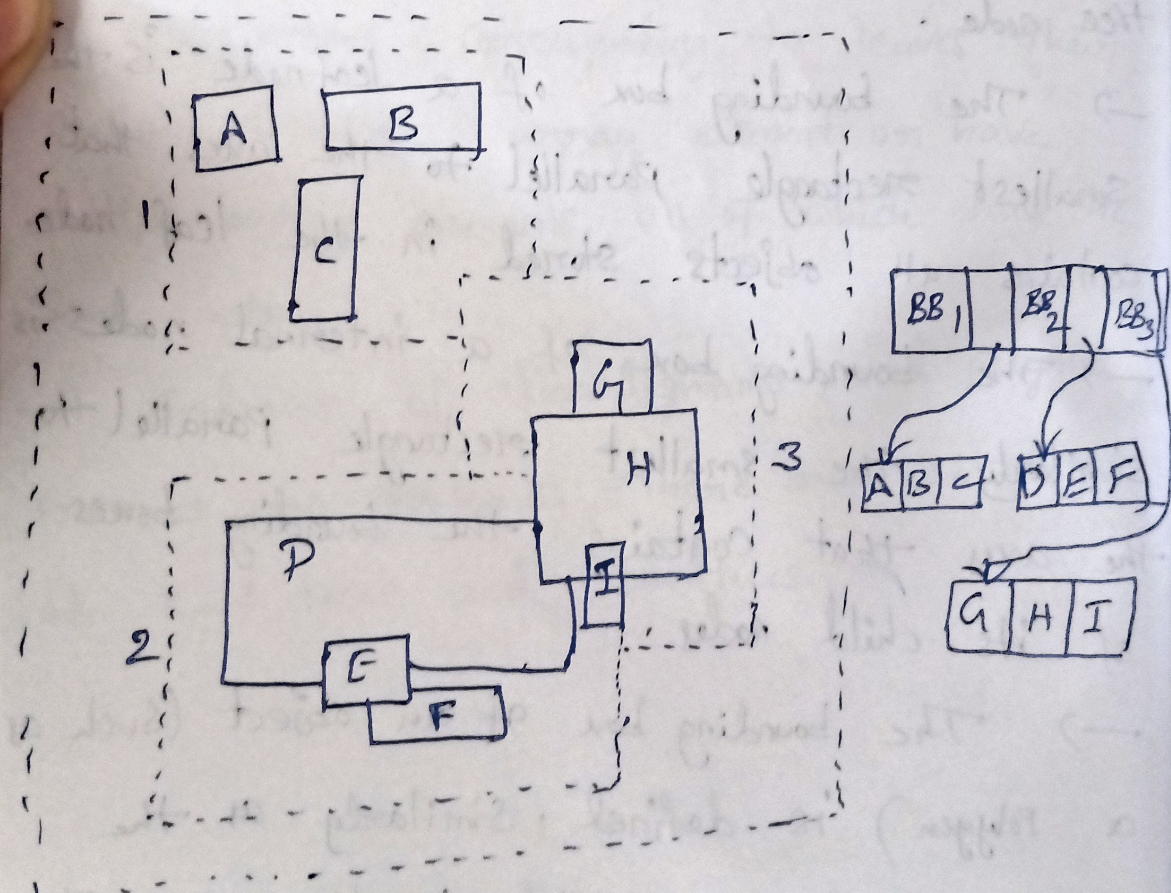
→ The bounding box of a leafnode is the smallest rectangle parallel to the axes that contains all objects stored in the leaf node.

→ The bounding box of a internal nodes is similarly the smallest rectangle parallel to the axes that contains the bounding boxes of its child nodes.

→ The bounding box of an object (such as a polygon) is defined, similarly as the smallest rectangle parallel to the axes that contains the object.

→ Each internode stores the bounding boxes of the child nodes along with the pointers to the child nodes.

→ Each leaf node stores the indexed objects, and may optionally store the bounding boxes of the objects, the bounding boxes help speed up checks for overlaps of the rectangle with the indexed objects. if a query rectangle does not overlap with the bounding box of an object, it cannot overlap with the object, either.



An R-tree

Figure shows an example of a set of rectangles (drawn with a solid line) and the bounding boxes (drawn with a dashed line) of the nodes of an R-tree for the set of rectangles.

→ Note that the bounding boxes are shown with extra space inside them, to make them stand out pictorially. In reality, the boxes would be smaller and fit tightly on the objects that they contain, that is each side of a bounding box B would touch at least one of the objects or bounding boxes that are contained in B.

→ The figure refers to the coordinates of bounding box i as BB_i in the figure.

→ Now we shall see how to implement operations like search, insert and delete on an R-tree.

Search:- As the figure shows, the bounding boxes associated with sibling nodes may overlap; in B+-trees, k-d trees and quadtrees, in contrast, the ranges do not overlap.

→ A search for objects containing a point therefore has to follow all child nodes whose associated bounding boxes contain the point as a result multiple paths may have to be searched.

→ Similarly a query to find all objects that intersect a given object has to go down every node where the associated rectangle intersects the given object.

Insert: 1) When we insert an object into an R-tree, we select a leaf node to hold the object. Ideally we should pick a leaf node that has space to hold a new entry and whose bb contains the bounding box of the object.

→ However, such a node may not exist, even if it did, finding the node may be very expensive, since it is not possible to find it by a single traversal down from the root.

→ At each internal node we may find multiple children whose bounding boxes contain

bounding box of the object, and each of these children needs to be explored.

→ In a traversal from the root, if any of the child nodes has a bounding box containing the bounding box of the object, the R-tree algorithm chooses one of them arbitarily.

→ If none of the children satisfy this condition, the algorithm chooses a child node whose bounding box has the maximum overlap with the bounding box of the object for continuing the traversal.

→ once the leaf node has been reached, if the node is already full, the algorithm performs node splitting in a manner very similar to B^+-tree insertion.

→ Just as with B^+-tree insertion, the R-tree insertion algorithm ensures that the tree remains balanced.

→ The main difference of the insertion procedure from the B^+-tree insertion procedure lies in how the node is split.

→ In a B+-tree, it is possible to find a value such that half the entries are less than midpoint and half are greater than the value.

→ This property does not generalize beyond one dimension, that is for more than one dimension, it is not always possible to split the entries into two sets so that their bounding boxes do not overlap.

→ The set of entries S can be split into two disjoint sets S_1 and S_2 so that the bounding boxes of S_1 and S_2 have the minimum total area, another heuristic ~~the~~ ~~set of entries S can be split into~~ would be to split the entries into two sets S_1 and S_2 in such a way that S_1 and S_2 have minimum overlap.

→ The two nodes resulting from the split would contain the entries in S_1 and S_2 respectively.

→ The cost of finding splits with minimum total area or overlap can itself be large, so cheaper heuristics such as the quadratic split heuristic are used.

→ The quadratic split heuristic works as first, it picks a pair of entries a and b from S such that putting them in the same node would result in a bounding box with the maximum wasted space. i.e the minimum bounding box of a and b minus sum of the areas of a and b is the largest.

→ Deletion: Deletion can be performed like a B+-tree, borrowing entries from sibling nodes, or merging sibling nodes if a node becomes underfull.

→ An alternate approach redistributes all the entries of underfull nodes to sibling nodes, with the aim of improving the clustering of entries in the R-tree.

→ The storage efficiency of R-trees is better than that of K-d trees or quadtrees. Since an object is stored only once, and we can ensure easily that each node is at least half full.

Multimedia Databases

→ Multimedia data, such as images, audio, and video - an increasingly popular form of data are today almost always stored outside the database, in file-system.

→ This kind of storage is not a problem when the number of multimedia objects is relatively small, since features provided by db are usually not important.

→ However, db features become important when the number of multimedia objects storage large. Issues such as transactional updates, querying facilities and indexing then become important.

→ Several issues must be addressed if multimedia data are to be stored in a database

1) The db must support large objects, since multimedia data such as videos can occupy up to a few gigabytes of storage. Many db systems do not support objects larger than a few gigabytes. Larger objects could be split into smaller pieces and stored in the db.

→ Alternatively the multimedia objects may be stored in a file system, but the db may contain a pointer to the object, the pointer would typically the file name.

→ The SQL/MED standard (MED stands for management of External Data) allows external data, such as files, to be treated as if they are part of the db.

→ With SQL/MED, the object would appear to be part of the database but can be stored externally.

2) The retrieval of some types of data, such as audio and video, has the requirement that data delivery must proceed at a guaranteed steady rate. such data are sometimes called isochronous data (or) continuous-media data.

Eg If audio data are not supplied in time, there will be gaps in the sound.

If the data are supplied too fast, system buffers may overflow resulting in loss of data.

3) Similarity-based retrieval is needed in many multimedia database applications.

Eg: In a db that stores fingerprint

images, a query fingerprint image is provided, and fingerprints in the db that are similar to the query fingerprint must be retrieved.

→ Index structures such as B+-trees and R-trees cannot be used for this purpose, special index structures need to be created.

1) Multimedia Data Formats

→ Because of the large number of bytes required to represent multimedia data, it is essential that multimedia data be stored and transmitted form.

→ For image data, the most widely used format is JPEG, named after the standards body that created it, the Joint Picture Experts Group.

→ We can store video data by encoding each frame of video in JPEG formats, but such an encoding is wasteful, since successive frames of a video are often nearly the same.

→ The Moving Picture Experts Group has developed the MPEG series of standards for encoding video and audio data, these encodings exploit commonalities among a sequence of frames to achieve a greater degree of compression.

→ The MPEG-1 standard stores a minute of 30-frame-per-second video and audio in approximately 12.5 megabytes.

→ MPEG-1 encoding introduces some loss of video quality, to a level compared to that of VHS videotape.

→ The MPEG-2 standard is designed for digital broadcast systems and digital video disks (DVDs) it introduces only a negligible loss of video quality.

→ The MPEG-2 compresses 1 minute of video and audio approximately 17 megabytes.

→ MPEG-4 provides techniques for further compression of video, with variable bandwidth to support delivery of video data over networks with a wide range of bandwidths.

→ MP3, which stands for MPEG-1 layer 3, RealAudio, Windows Media Audio and other formats.

→ High-definition video with audio is encoded in several variants of MPEG-4 that include MPEG-4 AVC and AVCHD.

2) Continuous-Media Data

→ The most important types of continuous media data are video and audio data (e.g. a database of movies).

→ Continuous-media systems are characterized by their real-time information-delivery requirements.

1) Data must be delivered sufficiently fast that no gaps in the audio or video result.

2) Data must be delivered at a rate that does not cause overflow of system buffers.

3) Synchronization among distinct data streams must be maintained. The need arises for e.g. when the video of a person speaking must show lips moving synchronously with the audio of the person speaking.

→ Several vendors offer video-on-demand servers.

The basic architecture of a video-on-demand system comprises:

1) video server: Multimedia data are stored on several disks (usually in a RAID configuration).

→ systems containing a large volume of data may use tertiary storage for less frequently accessed data.

2) Terminals: People view multimedia data through various devices, collectively referred to as terminals. E.g Personal Computers and television sets attached to a small, inexpensive computer called a set-top-box.

3) Network: Transmission of multimedia data from a server to multiple terminals requires a high-capacity network.

→ video-on-demand service over cable networks is widely available.

3) Similarity-Based Retrieval:

In many multimedia applications, data are described only approximately in the database.

Eg: Fingerprint data.

other examples are

a) Pictorial data: Two pictures or images that are slightly different as represented in the database may be considered the same by a user.

For Eg: a database may store trademark designs. when a new trademark is to be registered, the system may need first to identify all similar trademarks that were registered previously.

b) Audio data: speech-based user interfaces are being developed that allow the user to give a command identify a data item by speaking. The i/p from the user must then be tested for similarity to those commands or data items stored in the system.

c) Handwritten data: Handwritten i/p can be used to identify a handwritten data item or command stored in the db. Here again, i/p testing is not

Mobility and Personal Databases

→ Large-scale, commercial databases have traditionally been stored in central computing facilities.

→ several technology trends have combined to create applications in which this assumption of central control and administration is not entirely correct.

- The widespread use of laptop, notebook, or network computers.
- The widespread use of cell phones with the capabilities of a computer.
- The development of a relatively low-cost wireless digital communication infrastructure based on wireless local-area-networks, cellular digital packet networks, and other technologies.

→ Mobile computing has proved useful in many appl^ns. Many business travelers use laptop computers so that they can work and access data enroute.

→ Delivery services use mobile computers to assist in package tracking.

→ Emergency response services use mobile computers at the scene of disasters. Medical emergency personnel like to access information and to enter data pertaining to the situation.

→ Wireless computing creates a situation in which machines no longer have fixed locations or n/w addresses. Location-dependent queries are an interesting class of queries that are motivated by mobile computers, in such queries, the location of the user is a parameter of the query.

→ The value of the location parameter is provided by a GPS.

→ Energy (battery power) is a scarce resource for most mobile computers. This limitation influences many aspects of system design.

1) A model of mobile computing

→ The mobile-computing environment consists of mobile computers, referred to as mobile hosts, and a wired n/w of computers.

→ Mobile hosts communicate with the wired n/w via computers referred to as mobile support stations.

→ Each mobile support station manages those mobile hosts within its cell, that is the geographical area that it covers.

→ It is possible for mobile hosts to communicate directly without the intervention of a mobile support station. Such communication can occur only between nearby hosts. Such direct forms of communication often use Bluetooth, a short-range digital radio standard that allows wireless connectivity within a 10 meter range at high speed.

→ The n/w infrastructure for mobile computing consists in large part of two technologies

a) wireless local-area networks

b) Packet-based cellular telephony networks.

→ Wireless appn protocol (WAP) is a standard for wireless Internet access. WAP based browsers access web pages that use wireless markup language (WML), an XML-based language designed for the constraints of mobile and wireless web browsing.

2) Routing and Query Processing:

The route between a pair of hosts may change over time if one of the two hosts is mobile.

Competing notions of cost to consider

a) User time is a highly valuable commodity in many business appln's.

b) Connection time is the unit by which monetary charges are assigned in some cellular systems.

c) No of bytes, or Packets, transferred is the unit by which charges are computed in some digital cellular system

d) Time-of-day-based charges vary, depending on whether communication occurs during Peak or off-Peak Periods.

e) Energy is limited. often, battery Power is scarce whose use must be optimized.

3) Broadcast Data.

→ It is often desirable for frequently requested data to be broadcast in a continuous cycle by mobile support stations, rather than transmitted to mobile hosts on demand.

→ A typical appln of such broadcast data is stock-market price information.

→ There are two reasons for using broadcast data. First the mobile host avoids the energy cost for transmitting data requests. second, the broadcast data can be received by a large no of mobile hosts at once, at no extra cost.

4) Disconnectivity and consistency

Since wireless communication may be paid for on the basis of connection time, there is an incentive for certain mobile hosts to be disconnected for substantial periods.

→ During these periods of disconnection, the mobile host may remain in operation. This situation creates several problems

a) Recoverability: updates entered on a disconnected machine may be lost if the mobile host experiences a catastrophic failure. Since the mobile host represents a single point of failure, stable storage cannot be simulated well.

b) Consistency: Locally cached data may become out-of-date, but the mobile host cannot discover this situation until it is reconnected

Mobile updates

→ Partitioning via disconnection is the normal mode of operation in mobile computing.

→ For data updated by only one mobile host, simple to propagate update when mobile host reconnects.

→ when data are updated by other computers, invalidation reports inform a reconnected mobile host of out-of-date cache entries. However, mobile host may miss a report.

→ version-numbering based schemes also guarantee only that if two hosts independently update the same version of a document.

Detecting inconsistent updates

→ version vector scheme used to detect inconsistent updates to documents at different hosts (sites).

→ copies of document d at hosts i and j are inconsistent if k that have not been propagated to host j (k may be the same as i) and the copy of d at j contains updates performed by host that have not been propagated to host i (i may be the same as j)

Basic idea: each host i stores, with its copy of each document d, a version-vector - a set of version numbers, with an element $v_{d,i}[k]$ for every other host k.

when a host i updates a document d, it increments the version no $v_{d,i}[i]$ by 1.

→ when two hosts i and j connect to each other they check if the copies of all documents d that they share are consistent

→ If version vectors are the same on both hosts

k, $v_{d,i}[k] = v_{d,j}[k]$ then the copies of document d are identical.

→ If, for each k, $v_{d,i}[k] \leq v_{d,j}[k]$ and the version vectors are not identical

3) If there are a pair of hosts k and m such that $v_{d,i}[k] < v_{d,j}[k]$ and $v_{d,i}[m] > v_{d,j}[m]$ - then the copies are inconsistent.

That is two or more updates have been performed on d independently.

Handling Inconsistent updates

→ Dealing with inconsistent updates is hard in general. Manual invention often required to merge the updates.

→ Version vector schemes were developed to deal with failures in a distributed file system, where inconsistencies are rare.

→ used to maintain a unified file system b/w a fixed host and a mobile computer, where updates at the two hosts have to be merge periodically.

→ Inconsistencies must either be very rare, or fall in special cases that are easy to deal with in most cases.

Advanced Transaction Processing

Transaction Processing Monitors (TP Monitors):-

TP Monitors are systems that were developed in the 1970s and 1980s, initially in response to a need to support a large number of remote terminals such as airline-reservation terminals from a single computer.

→ The term TP monitor initially stood for teleprocessing monitor.

→ The CICS TP monitor from IBM was one of the earliest TP monitors, and has been very widely used. other TP monitors include oracle Tuxedo and Microsoft Transaction Server.

→ Provide services such as

a) Presentation facilities to simplify creating user interfaces

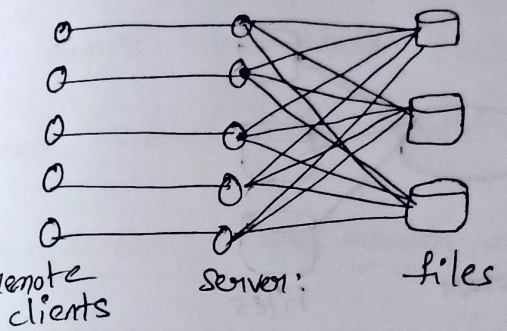
b) Persistent queuing of client requests and server responses

c) Routing of client messages to servers.

d) coordination of two-phase commit when transaction access multiple servers.

TP Monitor Architectures

a) Process-Per-client model



Large scale transaction processing systems are built around a client-server architecture. one way of building such systems is to have a server process for each client; the server performs authentication, and then executes actions requested by the client.

→ This model presents several problems with respect to memory utilization and processing speed.

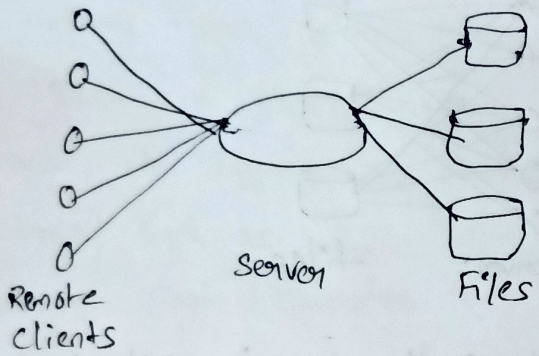
→ Per-Process memory requirements are high.

→ The operating system divides up available CPU time among processes by switching among them, the technique is called multitasking.

→ The above problem can be avoided by having a single-server process which all remote clients connect

this model is called single-server model.

b) Single-server Model



→ All Remote (clients) terminals connect to a single server Process.

→ Remote clients send requests to the server Process, which then executes those requests.

→ This model is also used in client-server environments, where clients send requests to a single server Process.

→ The server process handles tasks, such as user authentication.

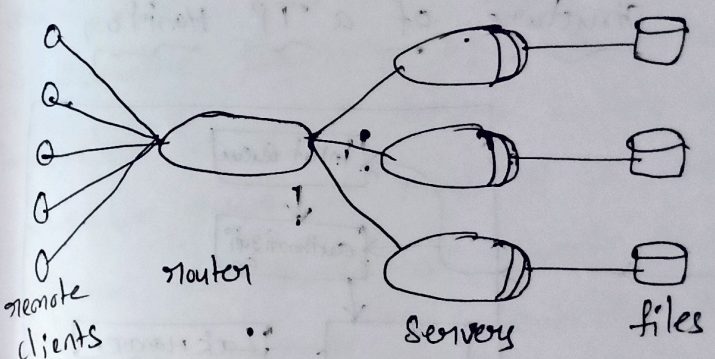
→ To avoid blocking other clients when processing a long request for one client, the server Process is Multithreaded. Low cost for thread switching.

Problems

1) Since all the appl^ns runs as a single process, there is no protection among them. A bug in one appl^n can affect all the other appl^ns as well..

2) Not suited for Parallel or distributed db's

Many-server, single router model



→ Multiple appl^n server processes access a common database.

→ clients communicate with the appl^n through a single communication process that routes requests.

→ This model supports independent server process e.g for multiple appl^ns.

→ Multithreaded server Process.

→ Run on Parallel or distributed database.

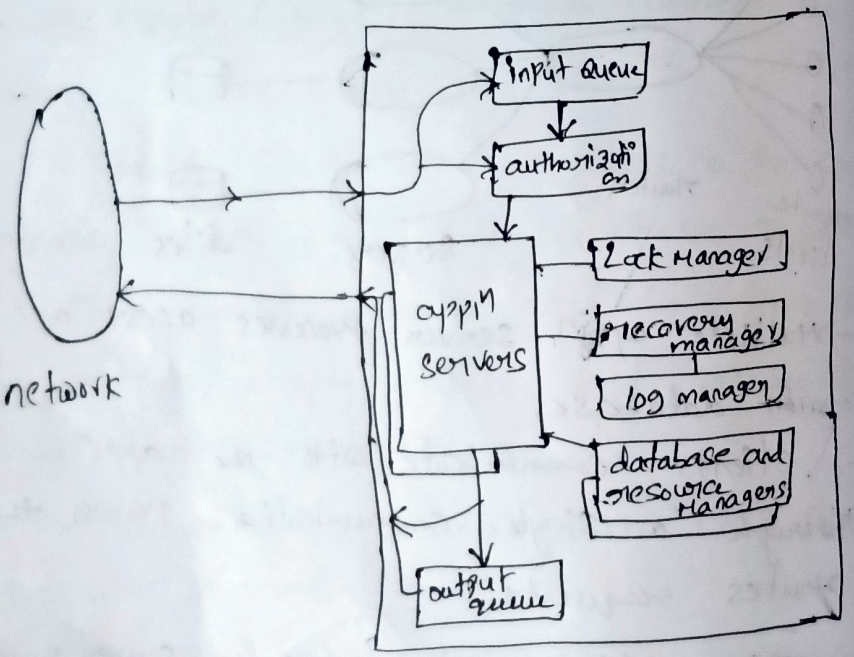
d) Many server many-router model : (Figure) in next page.

Multiple processes communicate with clients.

→ client communication processes interact with router processes that route their requests to the appropriate server.

→ A controller process starts up the other processes and services their functioning.

Detailed structure of a TP Monitor



TP-Monitor components

The above figure shows detailed structure of a TP-monitor.

→ A TP-monitor does more than simply pass messages to appln servers.

→ when messages arrive, they may have to be queued ; thus there is a queue manager for incoming messages.

→ some queue manager provide persistent or durable message queueing ; contents of queue are safe even if system fails.

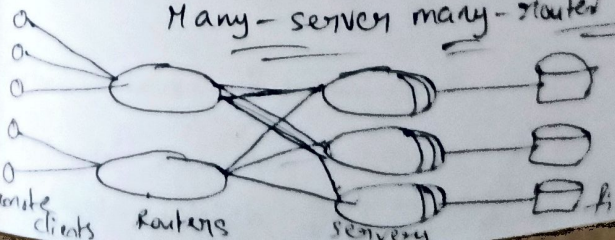
→ Durable queueing of outgoing messages is important. Appln server writes message to durable queue as part of a transaction

→ once the transaction commits, the TP monitor guarantees message is eventually delivered, regardless of crashes.

→ ACID properties are thus provided even for messages sent outside the database.

→ Many TP monitors provide locking, logging and recovery services, to enable appln servers to implement ACID properties by themselves.

Many-server many-router model



Application Coordination using TP Monitors

→ A TP monitor treats each subsystem as a resource manager that provides transactional access to some set of resources.

→ The interface between the TP monitor and the resource manager is defined by a set of transaction primitives. (begin_transaction, commit_transaction, abort_transaction, Prepare_to_commit transaction)

→ The resource manager interface is defined by the XOpen Distributed Transaction Processing standard.

→ TP monitor systems provide a transactional remote procedure call (transactional RPC) interface to their service.

→ Transaction RPC provides calls to enclose a series of RPC calls within a transaction.

→ Updates performed by an RPC are carried out within the scope of the transaction, and can be rolled back if there is any failure.

Transactional Workflows

→ A workflow is an activity in which multiple tasks are executed in coordinated way by different processing entities.

→ A task defines some work to be done and can be specified in a number of ways including a textual description in a file or electronic-mail message, a form, a message or a computer program.

→ The processing entity that performs the tasks may be a person or a software system.

Workflow application	Typical task	Typical processing entity
electronic-mail routing	electronic-mail message	Mailers
loan processing	form processing	humans, application software
purchase-order processing	form processing	humans, application s/w, DBMSs

Example of Workflows

→ The above figure shows examples of workflows. A simple example is that of an electronic mail system which goes through several mails systems to reach destination

→ Each mailer performs a tasks forwarding of the mail to the next mailer.

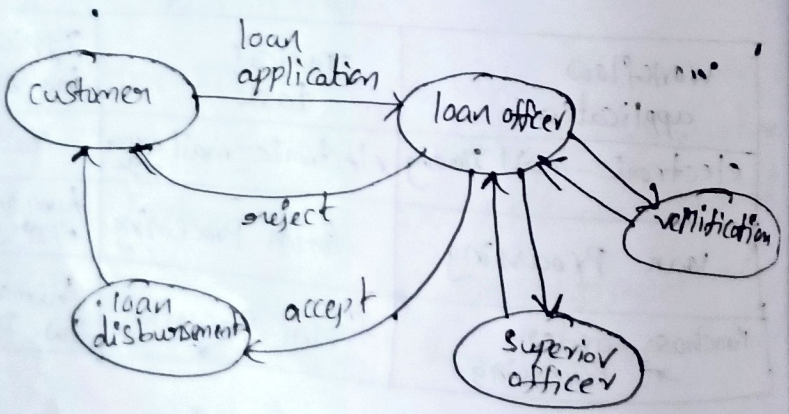
→ If a mailer cannot deliver mail, failure must be handled sematically (delivery failure message)

→ other terms used in the db and related literature to refer to workflows include task flow and multisystem appln's.

→ Workflow tasks are also sometimes called Steps.

→ In general, workflows may involve one or more humans.

→ For example consider the Processing of a loan.



Workflow in loan Processing

The above figure shows workflow in loan Processing. The person who wants a loan fills out a form, which is checked by a loan officer.

→ An employee who processes loan appln verifies the data in the form, using sources such as credit-reference bureaus.

→ When all the required information has been collected, the loan officer may decide to approve the loan, that decision may have to be approved by one or more superior officers, after which loan can be made.

→ Each human here Performs a task.

→ other examples of workflows include Processing of expense vouchers, of Purchase orders and of credit-card transactions.

→ Today all the information related to a workflow is more than likely to be stored in a digital form on one or more computers. and with the growth of networking information can be easily transferred from one computer to another. Hence it is feasible for organizations to automate their workflows

E.g To automate tasks involved in loan Processing. we can store loan appln and associated information in a database.

→ Workflows are becoming increasingly important for multiple reasons within as well as between organizations.

→ organizations are increasingly automating their services. Eg a supplier may provide an automated system for customers to place orders. several tasks may need to be carried out when an order is placed, including reserving production time to create the ordered product and delivery services to deliver the product.

→ We have to address two activities(issues)

a) Workflow specification

b) Workflow execution

Workflow specification: dealing the tasks that must be carried out and defining the execution requirements.

Static specification of task coordination

→ Tasks and dependencies among them are defined before the execution of the workflow Eg: The tasks in expensive voucher workflow may consist Starts. of the approvals of the voucher by a secretary, a manager and an accountant, in that order and finally a delivery of a check.

→ Can establish preconditions for execution of each task, taks are executed only when their preconditions are satisfied.

→ Defined preconditions through dependencies

. Execution states of other tasks, for example, "task t_i cannot start until task t_j has ended", or "task t_i must abort if task t_j has committed".

. output values of other tasks, for example "task t_i can start if task t_j returns a value greater than 25", or "the manager approval task can start if the secretary-approval task returns a value of OK".

. External variables modified by external events - for example, "task t_i cannot be started before 9 A.M." or "task t_i must be started within 24 hours of the completion of task t_j." → we can combine the dependencies by the regular logical connectives (and, or, not).

Dynamic task coordination

Eg Electronic mail routing system in which the text to be schedule for a given mail message depends on the destination address and on which immediate routers are functioning.

b) Workflow execution: Execute transactions specified in the workflow while also providing traditional db safeguards related to the correctness of computation, data integrity and durability.
E.g Loan appln should not get lost even if system fails.

Failure-Atomicity Requirements

→ Usual ACID transactional requirements are too strong / unimplementable for workflow applns. The workflow designer may specify the failure-atomicity requirements of a workflow according to the semantics of the workflow.
→ However, workflows must satisfy some limited transactional properties that guarantee a process is not left in an inconsistent state.
→ By executing a functionally equivalent task at another site.
→ The system must guarantee that every execution of a workflow will terminate in a state that satisfies the failure-atomicity requirements of the workflow by the designer.
We call those states acceptable termination states of a workflow.

→ All other execution states of a workflow constitute a set of nonacceptable termination states, in which the failure-atomicity requirements may be violated.

→ An acceptable termination state can be designated as committed or aborted
→ A committed acceptable termination state is an execution state in which the objectives of a workflow have been achieved
→ An aborted acceptable termination state is a valid termination state in which a workflow has failed to achieve its objectives.
→ A workflow must reach an acceptable termination state even in the presence of system failures.

Execution of Workflows

The execution of the tasks may be controlled by a human coordinator or by a software system called a workflow-management system.
Workflow management systems include
Scheduler:- Program that process workflows by

Submitting various tasks for execution, monitoring various events, and evaluation conditions related to intertask dependencies.

Task agents: control the execution of a task by a processing entity.

Workflow management system Architectures

There are three architectural approaches.

1) Centralized architecture: A centralized architecture has a single scheduler that schedules the tasks for all concurrently executing workflows.

2) The Partially distributed architecture has one scheduler instantiated for each workflow.

3) A fully distributed architecture has no scheduler, but task agents coordinate their execution by communicating with one another to satisfy task dependencies and other workflow execution requirements.

→ The centralized approach is used in workflow systems where the data are stored in a central database. Easier to keep track of the state of a workflow.

Workflow Scheduler

Ideally scheduler should execute a workflow only after ensuring that it will terminate in an acceptable state.

→ Consider a workflow consisting of two tasks S_1 and S_2. Let the failure-atomicity requirement be that either both or neither of the subtransaction should be committed.

→ Suppose systems executing S_1 and S_2 do not provide prepared-to-commit states and S_1 and S_2 do not have compensating transactions.

→ It is then possible to reach where one subtransaction is committed and the other aborted. Both cannot then be brought to same state.

→ Workflow specification is unsafe and should be rejected.

Recovery of a workflow:

The recovery procedures must make sure that, if a failure occurs in any of the workflow processing components (including the scheduler) the workflow will eventually reach an acceptable termination state (whether aborted or committed).

→ Handoff of tasks between agents should occur exactly once in spite of failure

issue: Repeating handoff on recovery may
- lead to duplicate execution of task.
- not repeating handoff may lead to task not being executed.

Soln: Persistent messaging systems.

→ Persistent messages are stored in Permanent message queue and therefore not lost in case of failure.

→ Before an agent commits, it writes to the Persistent message queue whatever messages need to be sent out.

→ The Persistent message System must make sure the messages get delivered eventually if and only if the transaction commits.

E-Commerce

E-commerce is the process of carrying out various activities related to commerce through electronic means, Primarily through internet

activities include:

- Prescale activities, needed to inform the potential buyer about the Product or Service being sold. (catalogs, advertisements, etc)
- The Sale Process, which includes negotiations on Price and quality of service and other contractual matters.
- The marketplace: When there are multiple sellers and buyers for a Product, a market place such as a stock exchange, helps in negotiating the Price to be Paid for the Product.

→ Auctions are used when there is a single seller and multiple buyers, and reverse auctions are used when there is a single buyer and multiple sellers.

- Payment for the sale.
- Activities related to delivery of the Product or service. Some Products and Services can be delivered over the Internet. for others the internet is used only for Providing shipping information and for tracking shipment of Products.

- Customer Support and Postsale Service.

E-catalogs

Any e-commerce site provides users with a catalog of the Products and services the site supplies. The services provided by an e-catalog may vary considerably.

→ At the minimum, an e-catalog must provide browsing and search facilities to help customers find the Product for which they are looking.

→ Customization of catalog for instance a retailer may have an agreement with a large company to supply some products at a discount.

- Negotiated pricing for specific organizations
- Special discounts for customers based on past history.

E.g loyalty discount.

- Legal restrictions on sales.

L certain items not exposed to under-age customers.

- Customization requires extensive customer-specific information.

Marketplaces

when there are multiple sellers or multiple buyers (or both) for a Product, a market place helps in negotiating the price to be paid for the Product.

→ There are several different types of market places:

a) Reverse auction
b) Auction
c) Exchange

→ In a reverse auction system a buyer states requirements, and sellers bid for bidding system, the bids are not made Public, where as in an open bidding system the bids are made public. (open vs closed bidding)
Lowest bidder wins (also known as tongen system)

→ In an auction there are multiple buyers and a single seller.

For eg: Assume that there is only one instance of each item being sold. Buyers bid for the items being sold, and

the highest bidder for an item gets to buy the item at the bid price.

→ when there are multiple copies of an item, things become more complicated.

for eg: Suppose there are four items, and one bidder may want three copies for $10 each, while another wants two copies for $13 each.

→ It is not possible to satisfy both bids. If the items will be of no value if they are not sold (e.g: airline seats), the seller simply picks a set of bids that maximizes the income. otherwise the decision is more complicated.

→ In an exchange, such as a stock exchange, there are multiple sellers and multiple buyers. Buyers can specify the maximum price they are willing to pay, while sellers specify the minimum price they want. There is usually a market maker who matches buy and sell bids, deciding on the price for each trade. Eg: average of buy/sell bids.

There are other more complex types of marketplaces

Among the database issues in handling marketplaces are there:

. Bidders need to be authenticated before they are allowed to bid.

. Bids need to be recorded securely in a database.

. Delays in broadcasting bids can lead to financial losses to some participants.

order settlement

→ After items have been selected and the price determined, the order has to be settled.

→ settlement involves payment for goods and the delivery of the goods.

→ Insecure means for electronic payment: Send credit card number.

There are two major problems

1) First credit-card fraud is possible when a buyer pays for physical goods, companies can ensure that the address for delivery matches the cardholder's address, so no one else can receive the goods, but for goods

delivered electronically no such check is possible.

2) Second, the seller has to be trusted to bill only for the agreed-on item and to not pass on the card number to unauthorized people who may misuse it.

→ Need secure payment systems - Avoid above mentioned problems. Provides greater degree of privacy.

E.g not reveal buyers identity to seller

→ Ensure that anyone monitoring the electronic transmissions cannot access critical information

Secure Payment Systems

→ All information must be encrypted to prevent eavesdropping.

Eg The seller may not be given any unnecessary details about the buyer, and the credit-card company is not provided any unnecessary information about the items purchased.

→ Public/Private key encryption is widely used for this task.

→ Must prevent person-in-the-middle attacks

E.g someone impersonates seller or bank/credit card company to make payment.

→ Impersonation is prevented by a system of digital certificates, whereby public keys are signed by a certification agency, whose public key is well known.

→ Three-way communication between seller, buyer and credit-card company to make payment. Called a secure payment protocol called the Secure Electronic Transaction (SET) Protocol.

Digital Cash

→ credit-card payment does not provide anonymity.

Eg The SET protocol hides buyers identity from seller. But even with SET, buyer can be traced with the help of credit card company.

→ Digital cash systems provide anonymity similar to that provided by physical cash.

Eg DigCash

→ Based on encryption techniques that make it impossible to find out who purchased digital cash from the cashbank.

→ Digital cash can be spent by purchasers in parts. Eg: much like writing a check on an account whose owner is anonymous.

→ Today, many banks provide secure payment gateways, which allow a purchaser a pay online at the banks web site, without exposing credit card or bank account information to the online merchant.

Main-Memory Databases

→ High performance hardware and parallelism help improve the rate of transaction processing but are insufficient to obtain high performance.

→ Since disk I/o remains a bottleneck - about 10 milliseconds are required for each I/o, and this number has not decreased at a rate comparable to the increase in processor speeds.

→ Disk I/o is often the bottleneck for reads, as well as for transaction commits.

→ Commercial 64-bit systems can support main memory database.

→ Memory resident data allows faster processing of transaction.

Disk related limitations:

1) Log records must be written to stable storage before a transaction committed. Logging is a bottleneck when transaction rate is high.

2) use group-commit to reduce number of o/p operations.

3) If the update rate for modified buffer blocks is high, the disk data-transfer rate could became a bottleneck.

4) If the system crashes, all of main memory is lost.

5) To reduce space overheads, main-memory databases can use structures with pointers crossing multiple pages. In disk databases, the I/o cost to traverse multiple pages would be excessively high.

6) No need to pin buffer pages in memory before data are accessed, since buffer pages will never be replaced.

7) Design query-processing techniques to minimize space overhead, avoid exceeding main memory limits during query evaluation.

8) Improve implementation of operations such as locking and latching, so they do not become bottlenecks.

9) optimize recovery algorithms, since pages rarely need to be written out to make space for other pages.

→ The process of committing ~~with~~ ~~T~~ ~~that have not~~ ~~been output to stable storage~~. a transaction T requires these records to be written to stable storage:

a) All log records associated with T that have not been output to stable storage.

b) The <T commit> log record.

→ These output operations frequently require the output of blocks that are only partially filled. To ensure that nearly full blocks are output, we use the group-commit technique

→ Instead of attempting to commit T when T completes, the system waits until several transactions have completed, or a certain period of time has passed since a transaction have completed execution.

→ It then commits the group of transactions that are waiting, together.

→ However, commits are delayed until a sufficiently large group of transaction are ready to commit, or a transaction has been waiting long enough - leads to slightly increased response time.

→ Above delay acceptable in high-performance transaction systems since log buffer blocks will fill up quickly.

→ Real-time Transaction systems

In systems with real-time constraints, correctness of execution involves both database consistency and satisfaction of deadlines.

Deadlines are characterized as follows:

a) Hard deadline: serious problems, such as system crash, may occur if a task is

not completed by its deadline.

b) Firm deadline: The task has zero value if it is completed after the deadline.

c) Soft deadlines: The task has diminishing value if it is completed after the deadline, with the value approaching zero as the degree of lateness increases.

→ Systems with deadlines are called real-time systems.

→ Transaction management in real-time systems must take deadlines into account.

→ A major difficulty in supporting real-time constraints arises from the variance in transaction execution time.

→ If data are resident in main memory, variances in execution time arise from lock waits, transaction aborts, and so on.

→ Design of a real-time system involves ensuring that enough processing power exists to meet deadline without requiring excessive hardware resources.

Long-Duration Transactions

→ The transaction concept developed initially in the context of data-processing applications, in which most transactions are noninteractive and short duration.

→ serious problems arise when this concept is applied to database systems that involve human interaction.

Such transactions have these key properties:

1) Long duration: once a human interacts with an active transaction, that transaction becomes a long-duration transaction from the prespective of the computer, since human response time is slow relative to computer speed.

→ Further-more, in design appl^ns, the human activity may involve hours, days, or an even longer period. Thus, transactions may be of long duration in human terms, as well as in machine terms.

2) Exposure of uncommitted data: Data generated and displayed to a user by a long-duration transaction are uncommitted, since the transaction may abort.

→ Thus users and as a result, other transactions - may be forced to read un-committed data.

→ If several users are cooperating on a project, user transactions may need to exchange data prior to transaction commit

Eg Partial update to a design.

3) Subtasks: An interactive transaction may consist of a set of subtasks initiated by the user. The user may wish to abort a subtask without necessarily causing the entire transaction to abort.

Eg:- Support partial rollback.

4) Recoverability: It is unacceptable to abort a long-duration interactive transaction because of a system crash. The active transaction must be recovered to a state that existed shortly before the crash so that relatively little human work is lost.

4) Performance: Good performance in an interactive transaction system is defined as fast response time.

→ Fast response time is essential so user time is not wasted.

Nonserializable Executions

. Two-phase locking: when a lock cannot be granted, the transaction requesting the lock is forced to wait for the data item in question to be unlocked.

→ The duration of this wait is proportional to the duration of the transaction holding the lock.

→ If the data item is locked by a short-duration transaction, we expect that the waiting time will be short.

→ If the data item is locked by a long-duration transaction, the wait will be of long duration. Long waiting times lead to both longer response time and an increased chance of deadlock.

· Graph-based protocols: Graph-based Protocols allow for locks to be released earlier than under the two-phase locking protocols, and they prevent deadlock.

- Timestamp-based protocols: Timestamp protocols never require a transaction to wait. However, they do require transactions to abort under certain circumstances.

→ If a long-duration transaction is aborted, a substantial amount of work is lost.

→ For noninteractive transactions, this lost work is a performance issue.

- Validation Protocols: Like time-stamp based protocols, validation protocols ensure ~~serializ~~ serializability by means of transaction abort.

Concurrency control:

→ The fundamental goal of database concurrency control is to ensure that concurrent execution of transactions does not result in a loss of database consistency.

→ The concept of serializability can be used to achieve this goal, since all serializable schedules preserve consistency of the database.

→ However, not all schedules that preserve consistency of the db are serializable.

T_1	T_2
read (A)	
A := A − 50	
write (A)	
	read (B)
	B := B − 10
read (B)	
B := B + 50	
write (B)	
	read (A)
	A := A + 10
	write (A)

A non-conflict-serializable schedule.

Above figure shows example of bank db consisting of two accounts A and B, with the consistency requirement that the sum A+B be preserved.

→ It illustrates two important points about the concept of correctness without serializability

- correctness depends on the specific consistency contraints for the database.
- correctness depends on the properties of operations performed by each transaction.

→ Treat some operations besides read and write as fundamental low-level operations and to extend concurrency control to deal with them.

Nested and Multilevel Transactions

A nested or multilevel transaction T is represented by a set $T = \{t_1, t_2, \ldots, t_n\}$ of subtransactions and a partial order P on T.

A subtransaction t_i in T may abort without forcing T to abort

Instead, T may either restart t_i, or simply choose not to run t_i.

If t_i commits, this action does not make t_i permanent. Instead t_i, commits to T, and may still abort if T aborts.

An execution of T must not violate the partial order P, i.e., if an edge $t_i \rightarrow t_j$ appears in the precedence graph, then $t_j \rightarrow t_i$ must not be in the transitive closure of P.

→ Nesting may be several levels deep, representing a subdivision of a transaction into subtasks, subsubtasks and so on.

→ At the lowest level of nesting, we have the standard database operations read and write.

Types of nested / multilevel transactions:

multilevel transaction: If a subtransaction of T is permitted to release locks on completion, T is called multilevel transaction.

→ Saga: when a multilevel transaction represents a long-duration activity, the transaction is sometimes referred to as a saga.

→ Nested transaction: If locks held by a subtransaction t_i of T are automatically assigned to T on completion of t_i, T is called a nested transaction.

Example of Nesting

Rewrite transaction T_1 using subtransaction $T_{1,1}$ and $T_{1,2}$ which performs increment or decrement operations

- T_1 consists of:
  - $T_{1,1}$ which subtracts 50 from A.
  - $T_{1,2}$ which adds 50 to B.

Similarly, rewrite transaction T_2, using subtransactions $T_{2,1}$ and $T_{2,2}$, which also perform increment or decrement operations:

- T_2 consists of
  - $T_{2,1}$, which subtracts 10 from B.
  - $T_{2,2}$, which adds 10 to A.

No ordering is specified on $T_{1,1}$, $T_{1,2}$, $T_{2,1}$, and $T_{2,2}$. Any execution of these subtransactions will generate a correct result.

Compensating Transactions

Alternative to undo operation; compensating transactions deal with the problem of cascading rollbacks.

→ Instead of undoing all changes made by the failed transaction, action is taken to "compensate" for the failure.

→ Consider a long-duration transaction T_i representing a travel reservation, with subtransactions $T_{i,1}$, which makes airline reservations, $T_{i,2}$ which reserves rental cars, and $T_{i,3}$ which reserves a hotel room.

→ Suppose that the hotel cancels the reservation. Instead of undoing all of T_i, we compensate for the failure $T_{i,3}$ by deleting the old hotel reservation and making a new one.

Implementation issues

→ Long-duration transactions must survive system crashes. We can ensure that they will by performing a redo on committed subtransactions and by performing either an undo or compensation for any short-duration subtransactions that were active at the time of the crash.

→ Logging of updates is made more complex by physically large data items (CAD design, document text); undesirable to store both old and new values.

→ There are two approaches to reducing the overhead of ensuring the recoverability of large data items:

- operation logging: only the operation performed on the data item and the data-item name are stored in the log.
→ operation logging is also called logical logging.

- logging and shadow paging: Logging is used for modification to small data items, but large data items are often made recoverable via shadowing, or copy-on-write technique. Only modified pages need to be shadowed or duplicated.

Unit-V

Database Security

1) Introduction to Database Security issues:

Database security is a broad

1.1) Types of Security: are that address many issues.

→ Various legal and ethical issues regarding the right to access certain information.

Eg: Some information may be deemed to be private and cannot be accessed legally by unauthorized organizations or persons. In the united states, there are numerous laws governing privacy of information.

→ Policy issues at the governmental, institutional, or corporate level regarding what kinds of information should not be made publicly available

Eg: credit ratings and personal medical records

→ System related issues such as system levels at which various security functions should be enforced.

Eg: whether a security function should be handled at the physical hardware level, the operating system level or the DBMS level

→ The need in some organizations to identify multiple security levels and to categorize the data and users based on these classifications.

Eg: top secret, secret, confidential and unclassified.

What Does Database Security Mean?

→ Database security refers to the collective measures used to protect and secure a db or database management software from illegitimate use and malicious cyber threats and attacks.

Threats to Databases: Threats to databases can result in the loss or degradation of some or all of the following accepted security goals:

1) Integrity.
2) availability.
3) Confidentiality.

Loss of integrity: Database integrity refers to the requirement that information be protected from improper modification.

(Improper modification of information)