Chapter 25: Advanced Data Types and New Applications

Database System Concepts, 6th Ed.

Chapter 25: Advanced Data Types and New Applications

- Temporal Data
- Spatial and Geographic Databases
- Multimedia Databases
- Mobility and Personal Databases

Time In Databases

- While most databases tend to model reality at a point in time (at the "current" time), temporal databases model the states of the real world across time.
- Facts in temporal relations have associated times when they are *valid*, which can be represented as a union of intervals.
- The **transaction time** for a fact is the time interval during which the fact is current within the database system.
- In a **temporal relation**, each tuple has an associated time when it is true; the time may be either valid time or transaction time.
- A **bi-temporal relation** stores both valid and transaction time.

Time In Databases (Cont.)

- Example of a temporal relation:

ID	*name*	*dept_name*	*salary*	*from*	*to*
10101	Srinivasan	Comp. Sci.	61000	2007/1/1	2007/12/31
10101	Srinivasan	Comp. Sci.	65000	2008/1/1	2008/12/31
12121	Wu	Finance	82000	2005/1/1	2006/12/31
12121	Wu	Finance	87000	2007/1/1	2007/12/31
12121	Wu	Finance	90000	2008/1/1	2008/12/31
98345	Kim	Elec. Eng.	80000	2005/1/1	2008/12/31

- Temporal query languages have been proposed to simplify modeling of time as well as time related queries.

Time Specification in SQL-92

- **date**: four digits for the year (1--9999), two digits for the month (1--12), and two digits for the date (1--31).
- **time**: two digits for the hour, two digits for the minute, and two digits for the second, plus optional fractional digits.
- **timestamp**: the fields of **date** and **time**, with six fractional digits for the seconds field.
- Times are specified in the *Universal Coordinated Time*, abbreviated UTC (from the French); supports **time with time zone**.
- **interval**: refers to a period of time (e.g., 2 days and 5 hours), without specifying a particular time when this period starts; could more accurately be termed a *span*.

Temporal Query Languages

- Predicates *precedes, overlaps,* and *contains* on time intervals.
- *Intersect* can be applied on two intervals, to give a single (possibly empty) interval; the union of two intervals may or may not be a single interval.
- A **snapshot** of a temporal relation at time t consists of the tuples that are valid at time t, with the time-interval attributes projected out.
- **Temporal selection**: involves time attributes
- **Temporal projection**: the tuples in the projection inherit their time-intervals from the tuples in the original relation.
- **Temporal join**: the time-interval of a tuple in the result is the intersection of the time-intervals of the tuples from which it is derived. It intersection is empty, tuple is discarded from join.

Temporal Query Languages (Cont.)

- Functional dependencies must be used with care: adding a time field may invalidate functional dependency
- A **temporal functional dependency** $\mathbf{X} \xrightarrow{\tau} \mathbf{Y}$ holds on a relation schema R if, for all legal instances r of R, all snapshots of r satisfy the functional dependency $X \rightarrow Y$.
- SQL:1999 Part 7 (SQL/Temporal) is a proposed extension to SQL:1999 to improve support of temporal data.

Spatial and Geographic Databases

Database System Concepts, 6th Ed.

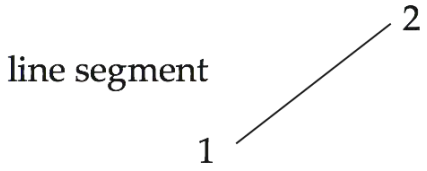
Spatial and Geographic Databases

- Spatial databases store information related to spatial locations, and support efficient storage, indexing and querying of spatial data.
- Special purpose index structures are important for accessing spatial data, and for processing spatial join queries.
- **Computer Aided Design (CAD)** databases store design information about how objects are constructed E.g.: designs of buildings, aircraft, layouts of integrated-circuits
- Geographic databases store geographic information (e.g., maps): often called **geographic information systems or GIS**.

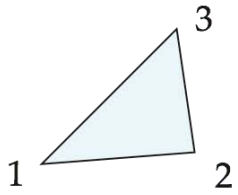
Represented of Geometric Information

- Various geometric constructs can be represented in a database in a normalized fashion.
- Represent a line segment by the coordinates of its endpoints.
- Approximate a curve by partitioning it into a sequence of segments
  - Create a list of vertices in order, or
  - Represent each segment as a separate tuple that also carries with it the identifier of the curve (2D features such as roads).
- Closed polygons
  - List of vertices in order, starting vertex is the same as the ending vertex, or
  - Represent boundary edges as separate tuples, with each containing identifier of the polygon, or
  - Use **triangulation** — divide polygon into triangles
    - Note the polygon identifier with each of its triangles.

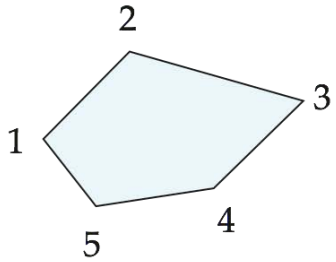
Representation of Geometric Constructs



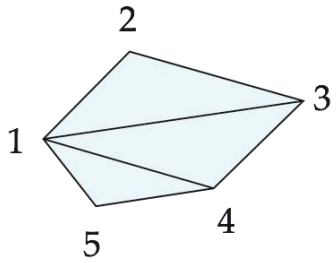
{(x1,y1), (x2,y2)}



{(x1,y1), (x2,y2), (x3,y3)}



{(x1,y1), (x2,y2), (x3,y3), (x4,y4), (x5,y5)}



{(x1,y1), (x2,y2), (x3,y3), ID1}
{(x1,y1), (x3,y3), (x4,y4), ID1}
{(x1,y1), (x4,y4), (x5,y5), ID1}

object **representation**

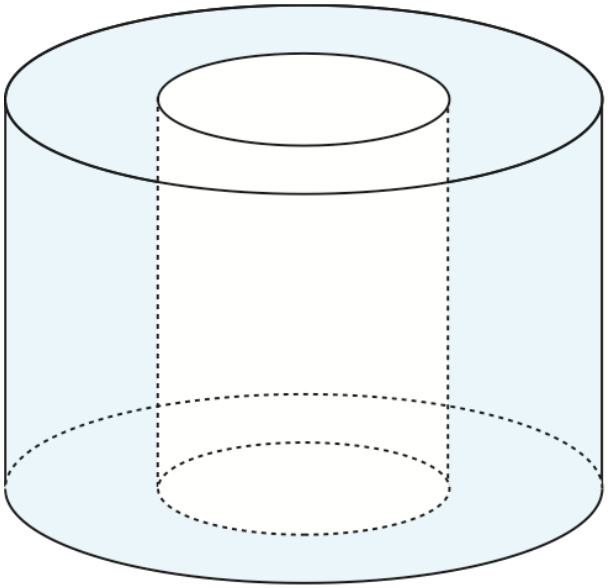
Representation of Geometric Information (Cont.)

- Representation of points and line segment in 3-D similar to 2-D, except that points have an extra z component
- Represent arbitrary polyhedra by dividing them into tetrahedrons, like triangulating polygons.
- Alternative: List their faces, each of which is a polygon, along with an indication of which side of the face is inside the polyhedron.

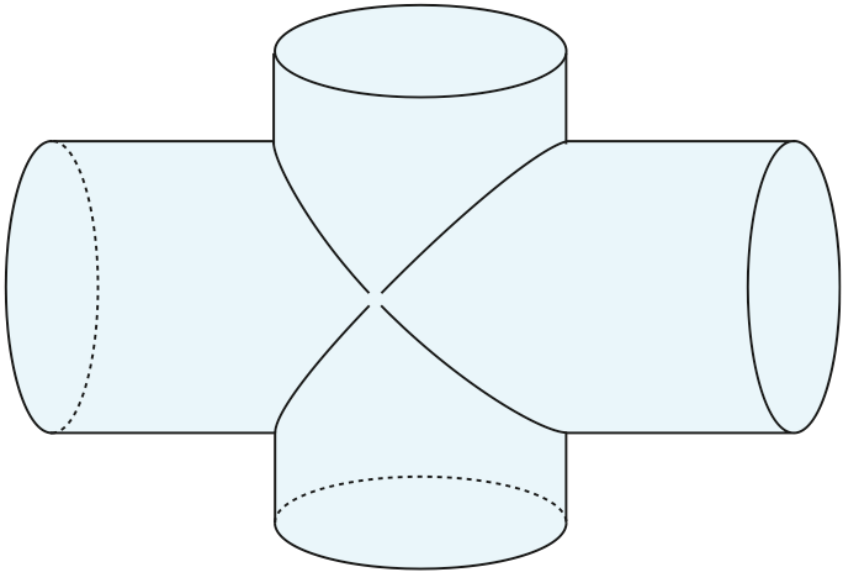
Design Databases

- Represent design components as objects (generally geometric objects); the connections between the objects indicate how the design is structured.
- Simple two-dimensional objects: points, lines, triangles, rectangles, polygons.
- Complex two-dimensional objects: formed from simple objects via union, intersection, and difference operations.
- Complex three-dimensional objects: formed from simpler objects such as spheres, cylinders, and cuboids, by union, intersection, and difference operations.
- Wireframe models represent three-dimensional surfaces as a set of simpler objects.

Representation of Geometric Constructs



(a) Difference of cylinders



(b) Union of cylinders

- Design databases also store non-spatial information about objects (e.g., construction material, color, etc.)
- Spatial integrity constraints are important.
  - E.g., pipes should not intersect, wires should not be too close to each other, etc.

Geographic Data

- **Raster data** consist of bit maps or pixel maps, in two or more dimensions.
  - Example 2-D raster image: satellite image of cloud cover, where each pixel stores the cloud visibility in a particular area.
  - Additional dimensions might include the temperature at different altitudes at different regions, or measurements taken at different points in time.
- Design databases generally do not store raster data.

Geographic Data (Cont.)

- **Vector data** are constructed from basic geometric objects: points, line segments, triangles, and other polygons in two dimensions, and cylinders, spheres, cuboids, and other polyhedrons in three dimensions.
- Vector format often used to represent map data.
  - Roads can be considered as two-dimensional and represented by lines and curves.
  - Some features, such as rivers, may be represented either as complex curves or as complex polygons, depending on whether their width is relevant.
  - Features such as regions and lakes can be depicted as polygons.

Applications of Geographic Data

- Examples of geographic data
  - map data for vehicle navigation
  - distribution network information for power, telephones, water supply, and sewage
- Vehicle navigation systems store information about roads and services for the use of drivers:
  - **Spatial data**: e.g., road/restaurant/gas-station coordinates
  - **Non-spatial data**: e.g., one-way streets, speed limits, traffic congestion
- **Global Positioning System (GPS)** unit - utilizes information broadcast from GPS satellites to find the current location of user with an accuracy of tens of meters.
  - increasingly used in vehicle navigation systems as well as utility maintenance applications.

Spatial Queries

- **Nearness queries** request objects that lie near a specified location.
- **Nearest neighbor queries**, given a point or an object, find the nearest object that satisfies given conditions.
- **Region queries** deal with spatial regions. e.g., ask for objects that lie partially or fully inside a specified region.
- Queries that compute intersections or **unions** of regions.
- **Spatial join** of two spatial relations with the location playing the role of join attribute.

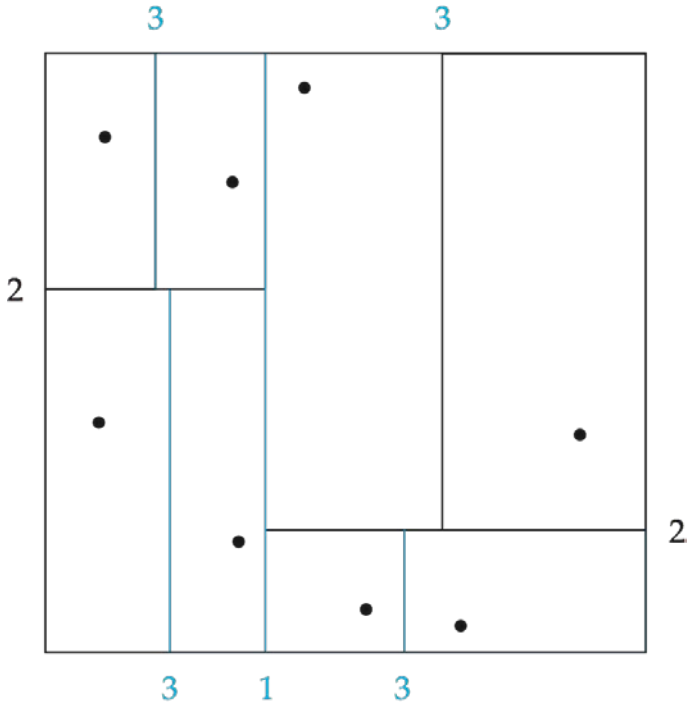
Spatial Queries (Cont.)

- Spatial data is typically queried using a graphical query language; results are also displayed in a graphical manner.
- Graphical interface constitutes the front-end
- Extensions of SQL with abstract data types, such as lines, polygons and bit maps, have been proposed to interface with back-end.
  - allows relational databases to store and retrieve spatial information
  - Queries can use spatial conditions (e.g., contains or overlaps).
  - queries can mix spatial and nonspatial conditions

Indexing of Spatial Data

- **k-d tree** - early structure used for indexing in multiple dimensions.
- Each level of a *k-d* tree partitions the space into two.
  - choose one dimension for partitioning at the root level of the tree.
  - choose another dimensions for partitioning in nodes at the next level and so on, cycling through the dimensions.
- In each node, approximately half of the points stored in the sub-tree fall on one side and half on the other.
- Partitioning stops when a node has less than a given maximum number of points.
- The **k-d-B tree** extends the *k-d* tree to allow multiple child nodes for each internal node; well-suited for secondary storage.

Division of Space by a k-d Tree

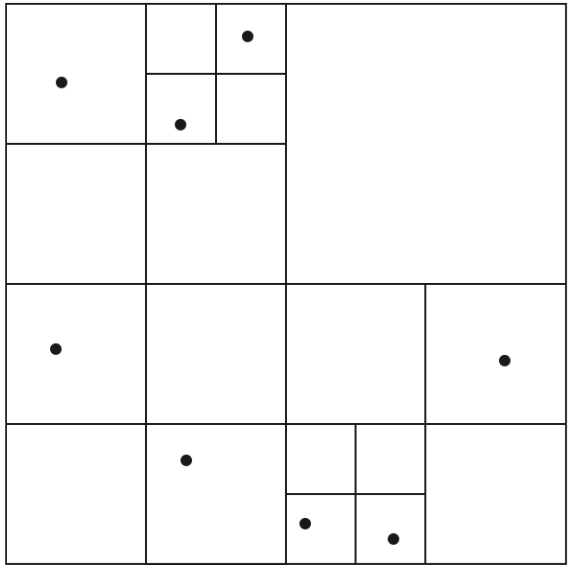


- Each line in the figure (other than the outside box) corresponds to a node in the *k-d* tree.
  - The maximum number of points in a leaf node has been set to 1.
- The numbering of the lines in the figure indicates the level of the tree at which the corresponding node appears.

Division of Space by Quadtrees

Quadtrees

- 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 nodes divides its region into four equal sized quadrants
  - Correspondingly each such node has four child nodes corresponding to the four quadrants and so on
- Leaf nodes have between zero and some fixed maximum number of points (set to 1 in example).



Quadtrees (Cont.)

- **PR quadtree**: stores points; space is divided based on regions, rather than on the actual set of points stored.
- **Region quadtrees** store array (raster) information.
  - A node is a leaf node is 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 corresponds to a sub-array of values.
  - The sub-arrays corresponding to leaves either contain just a single array element, or have multiple array elements, all of which have the same value.
- Extensions of *k-d* trees and PR quadtrees have been proposed to index line segments and polygons
  - Require splitting segments/polygons into pieces at partitioning boundaries
    - Same segment/polygon may be represented at several leaf nodes

R-Trees

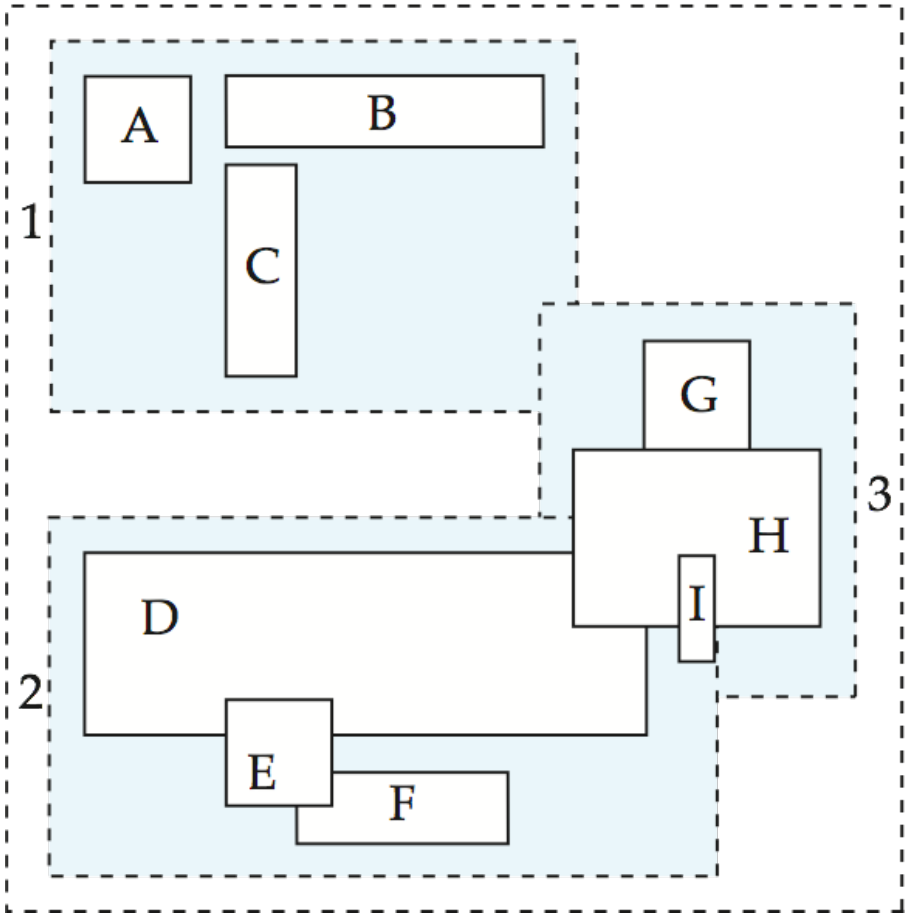
- **R-trees** are a N-dimensional extension of B^+-trees, useful for indexing sets of rectangles and other polygons.
- Supported in many modern database systems, along with variants like R^+ -trees and R*-trees.
- Basic idea: generalize the notion of a one-dimensional interval associated with each B+ -tree node to an N-dimensional interval, that is, an N-dimensional rectangle.
- Will consider only the two-dimensional case ($N = 2$)
  - generalization for $N > 2$ is straightforward, although R-trees work well only for relatively small N

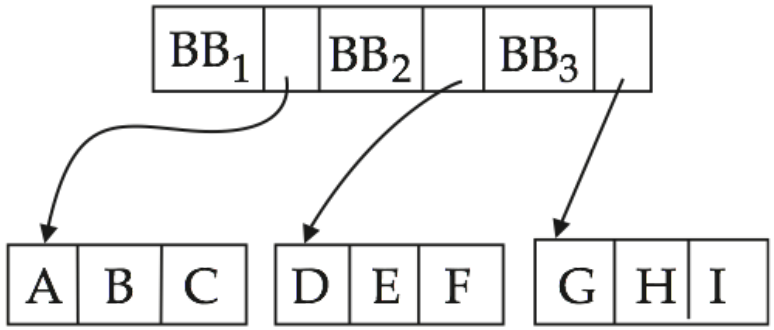
R Trees (Cont.)

- A rectangular **bounding box** is associated with each tree node.
  - Bounding box of a leaf node is a minimum sized rectangle that contains all the rectangles/polygons associated with the leaf node.
  - The bounding box associated with a non-leaf node contains the bounding box associated with all its children.
  - Bounding box of a node serves as its key in its parent node (if any)
  - *Bounding boxes of children of a node are allowed to overlap*
- A polygon is stored only in one node, and the bounding box of the node must contain the polygon.
  - The storage efficiency or R-trees is better than that of k-d trees or quadtrees since a polygon is stored only once.

Example R-Tree

- A set of rectangles (solid line) and the bounding boxes (dashed line) of the nodes of an R-tree for the rectangles. The R-tree is shown on the right.





Search in R-Trees

- To find data items (rectangles/polygons) intersecting (overlaps) a given query point/region, do the following, starting from the root node:
  - If the node is a leaf node, output the data items whose keys intersect the given query point/region.
  - Else, for each child of the current node whose bounding box overlaps the query point/region, recursively search the child
- Can be very inefficient in worst case since multiple paths may need to be searched
  - but works acceptably in practice.
- Simple extensions of search procedure to handle predicates *contained-in* and *contains*

Insertion in R-Trees

- To insert a data item:
  - Find a leaf to store it, and add it to the leaf
    - To find leaf, follow a child (if any) whose bounding box contains bounding box of data item, else child whose overlap with data item bounding box is maximum
  - Handle overflows by splits (as in B+-trees)
    - Split procedure is different though (see below)
  - Adjust bounding boxes starting from the leaf upwards
- Split procedure:
  - Goal: divide entries of an overfull node into two sets such that the bounding boxes have minimum total area
    - This is a heuristic. Alternatives like minimum overlap are possible
  - Finding the “best” split is expensive, use heuristics instead
    - See next slide

Splitting an R-Tree Node

- **Quadratic split** divides the entries in a node into two new nodes as follows
  1. Find pair of entries with “maximum separation”
     - that is, the pair such that the bounding box of the two would has the maximum wasted space (area of bounding box – sum of areas of two entries)
  2. Place these entries in two new nodes
  3. Repeatedly find the entry with “maximum preference” for one of the two new nodes, and assign the entry to that node
     - Preference of an entry to a node is the increase in area of bounding box if the entry is added to the *other* node
  4. Stop when half the entries have been added to one node
     - Then assign remaining entries to the other node
- Cheaper **linear split** heuristic works in time linear in number of entries,
  - Cheaper but generates slightly worse splits.

Deleting in R-Trees

- Deletion of an entry in an R-tree done much like a B^+-tree deletion.
  - In case of underfull node, borrow entries from a sibling if possible, else merging sibling nodes
  - Alternative approach removes all entries from the underfull node, deletes the node, then reinserts all entries

Multimedia Databases

Database System Concepts, 6th Ed.

Multimedia Databases

- To provide such database functions as indexing and consistency, it is desirable to store multimedia data in a database
  - rather than storing them outside the database, in a file system
- The database must handle large object representation.
- Similarity-based retrieval must be provided by special index structures.
- Must provide guaranteed steady retrieval rates for continuous-media data.

Multimedia Data Formats

- Store and transmit multimedia data in compressed form
  - JPEG and GIF the most widely used formats for image data.
  - MPEG standard for video data use commonalties among a sequence of frames to achieve a greater degree of compression.
- MPEG-1 quality comparable to VHS video tape.
  - stores a minute of 30-frame-per-second video and audio in approximately 12.5 MB
- MPEG-2 designed for digital broadcast systems and digital video disks; negligible loss of video quality.
  - Compresses 1 minute of audio-video to approximately 17 MB.
- Several alternatives of audio encoding
  - MPEG-1 Layer 3 (MP3), RealAudio, WindowsMedia format, etc.

Continuous-Media Data

- Most important types are video and audio data.
- Characterized by high data volumes and real-time information-delivery requirements.
  - Data must be delivered sufficiently fast that there are no gaps in the audio or video.
  - Data must be delivered at a rate that does not cause overflow of system buffers.
  - Synchronization among distinct data streams must be maintained
    - Video of a person speaking must show lips moving synchronously with the audio

Video Servers

- **Video-on-demand** systems deliver video from central video servers, across a network, to terminals
  - Must guarantee end-to-end delivery rates
- Current video-on-demand servers are based on file systems; existing database systems do not meet real-time response requirements.
- Multimedia data are stored on several disks (RAID configuration), or on tertiary storage for less frequently accessed data.
- Head-end terminals - used to view multimedia data
  - PCs or TVs attached to a small, inexpensive computer called a set-top box.

Similarity-Based Retrieval

Examples of similarity based retrieval

- Pictorial data: Two pictures or images that are slightly different as represented in the database may be considered the same by a user.
  - E.g., identify similar designs for registering a new trademark.
- Audio data: Speech-based user interfaces allow the user to give a command or identify a data item by speaking.
  - E.g., test user input against stored commands.
- Handwritten data: Identify a handwritten data item or command stored in the database

Mobility

Database System Concepts, 6th Ed.

Mobile Computing Environments

- A mobile computing environment consists of mobile computers, referred to as **mobile hosts**, and a wired network of computers.
- Mobile host may be able to communicate with wired network through a *wireless digital communication network*
  - Wireless local-area networks (within a building)
    - E.g., Avaya's Orinico Wireless LAN
  - Wide areas networks
    - Cellular digital packet networks
      - 3 G and 2.5 G cellular networks

Mobile Computing Environments (Cont.)

- A model for mobile communication
  - Mobile hosts communicate to the wired network via computers referred to as **mobile support (or base) stations**.
  - Each mobile support station manages those mobile hosts within its **cell**.
  - When mobile hosts move between cells, there is a **handoff** of control from one mobile support station to another.
- Direct communication, without going through a mobile support station is also possible between nearby mobile hosts
  - Supported, for e.g., by the Bluetooth standard (up to 10 meters, atup to 721 kbps)

Database Issues in Mobile Computing

- New issues for query optimization.
  - Connection time charges and number of bytes transmitted
  - Energy (battery power) is a scarce resource and its usage must be minimized
- Mobile user's locations may be a parameter of the query
  - GIS queries
  - Techniques to track locations of large numbers of mobile hosts
- Broadcast data can enable any number of clients to receive the same data at no extra cost
  - leads to interesting querying and data caching issues.
- Users may need to be able to perform database updates even while the mobile computer is disconnected.
  - E.g., mobile salesman records sale of products on (local copy of) database.
  - Can result in conflicts detected on reconnection, which may need to be resolved manually.

Routing and Query Processing

- Must consider these competing costs:
  - User time.
  - Communication cost
    - Connection time - used to assign monetary charges in some cellular systems.
    - Number of bytes, or packets, transferred - used to compute charges in digital cellular systems
    - Time-of-day based charges - vary based on peak or off-peak periods
  - Energy - optimize use of battery power by minimizing reception and transmission of data.
    - Receiving radio signals requires much less energy than transmitting radio signals.

Broadcast Data

- Mobile support stations can broadcast frequently-requested data
  - Allows mobile hosts to wait for needed data, rather than having to consume energy transmitting a request
  - Supports mobile hosts without transmission capability
- A mobile host may optimize energy costs by determining if a query can be answered using only cached data
  - If not then must either;
    - Wait for the data to be broadcast
    - Transmit a request for data and must know when the relevant data will be broadcast.
- Broadcast data may be transmitted according to a fixed schedule or a changeable schedule.
  - For changeable schedule: the broadcast schedule must itself be broadcast at a well-known radio frequency and at well-known time intervals
- Data reception may be interrupted by noise
  - Use techniques similar to RAID to transmit redundant data (parity)

Disconnectivity and Consistency

- A mobile host may remain in operation during periods of disconnection.
- Problems created if the user of the mobile host issues queries and updates on data that resides or is cached locally:
  - **Recoverability**: Updates entered on a disconnected machine may be lost if the mobile host fails. Since the mobile host represents a single point of failure, stable storage cannot be simulated well.
  - **Consistency**: Cached data may become out of date, but the mobile host cannot discover this 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
  - In other cases data may become invalid and updates may conflict.
- 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 guarantee only that if two hosts independently update the same version of a document, the clash will be detected eventually, when the hosts exchange information either directly or through a common host.
  - More on this shortly
- Automatic reconciliation of inconsistent copies of data is difficult
  - Manual intervention may be needed

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
  1. the copy of document *d* at *i* contains updates performed by host *k* that have not been propagated to host *j* (*k* may be the same as *i*), and
  2. the copy of *d* at *j* contains updates performed by host *l* that have not been propagated to host *i* (*l* 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 number $V_{d,i}[i]$ by 1

Detecting Inconsistent Updates (Cont.)

- When two hosts *i* and *j* connect to each other they check if the copies of all documents d that they share are consistent:
  1. If the version vectors are the same on both hosts (that is, for each *k*, $V_{d,i}[k] = V_{d,j}[k]$) then the copies of *d* are identical.
  2. If, for each *k*, $V_{d,i}[k] \leq V_{d,j}[k]$, and the version vectors are not identical, then the copy of document *d* at host *i* is older than the one at host *j*
     - That is, the copy of document *d* at host *j* was obtained by one or more modifications of the copy of *d* at host *i*.
     - Host *i* replaces its copy of *d*, as well as its copy of the version vector for *d*, with the copies from host *j*.
  3. If there is 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 intervention often required to merge the updates.
- Version vector schemes
  - were developed to deal with failures in a distributed file system, where inconsistencies are rare.
  - are used to maintain a unified file system between a fixed host and a mobile computer, where updates at the two hosts have to be merged periodically.
    - Also used for similar purposes in groupware systems.
  - are used in database systems where mobile users may need to perform transactions.
    - In this case, a “document” may be a single record.
- Inconsistencies must either be very rare, or fall in special cases that are easy to deal with in most cases

End of Chapter

Figure 25.02

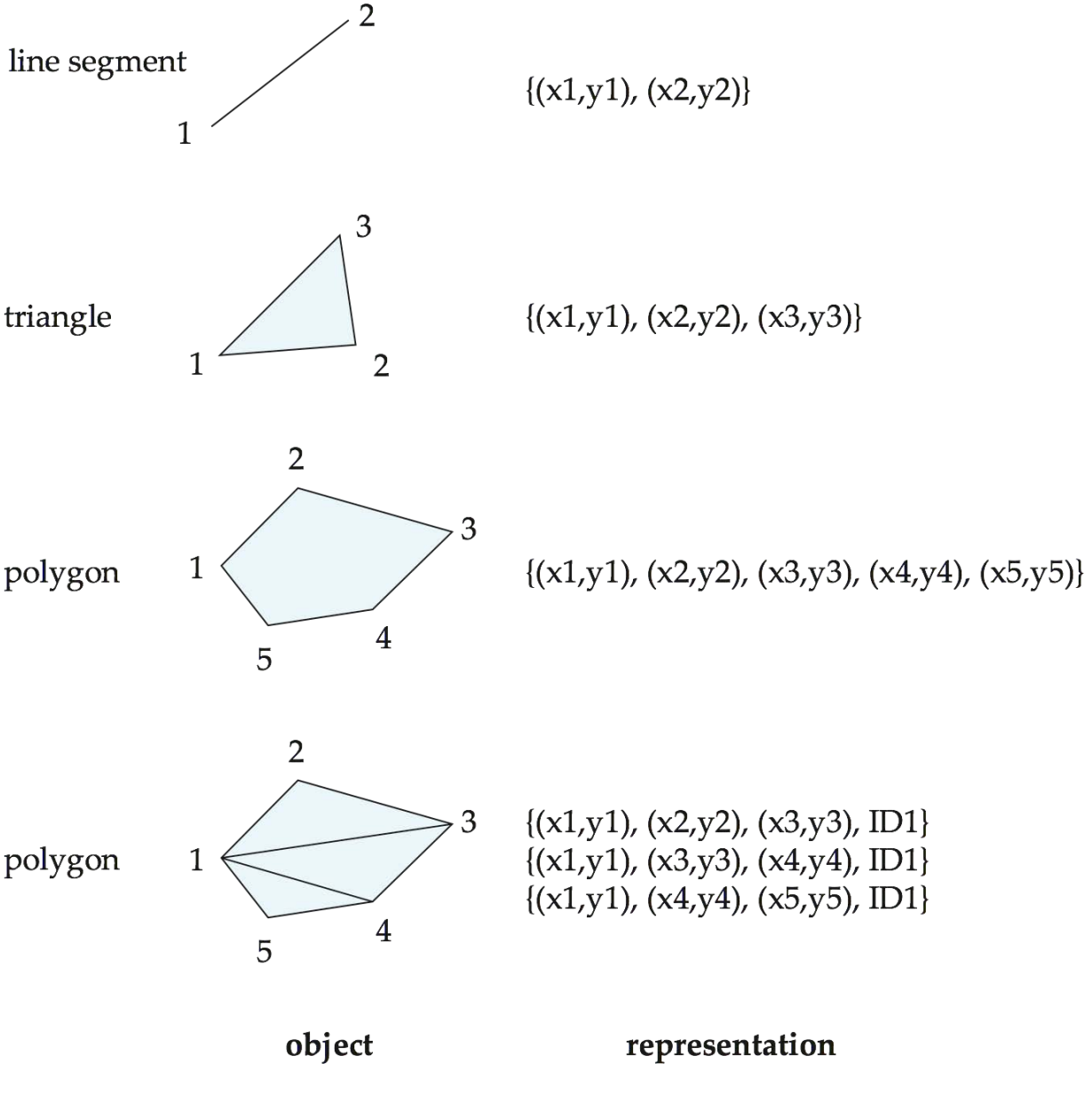
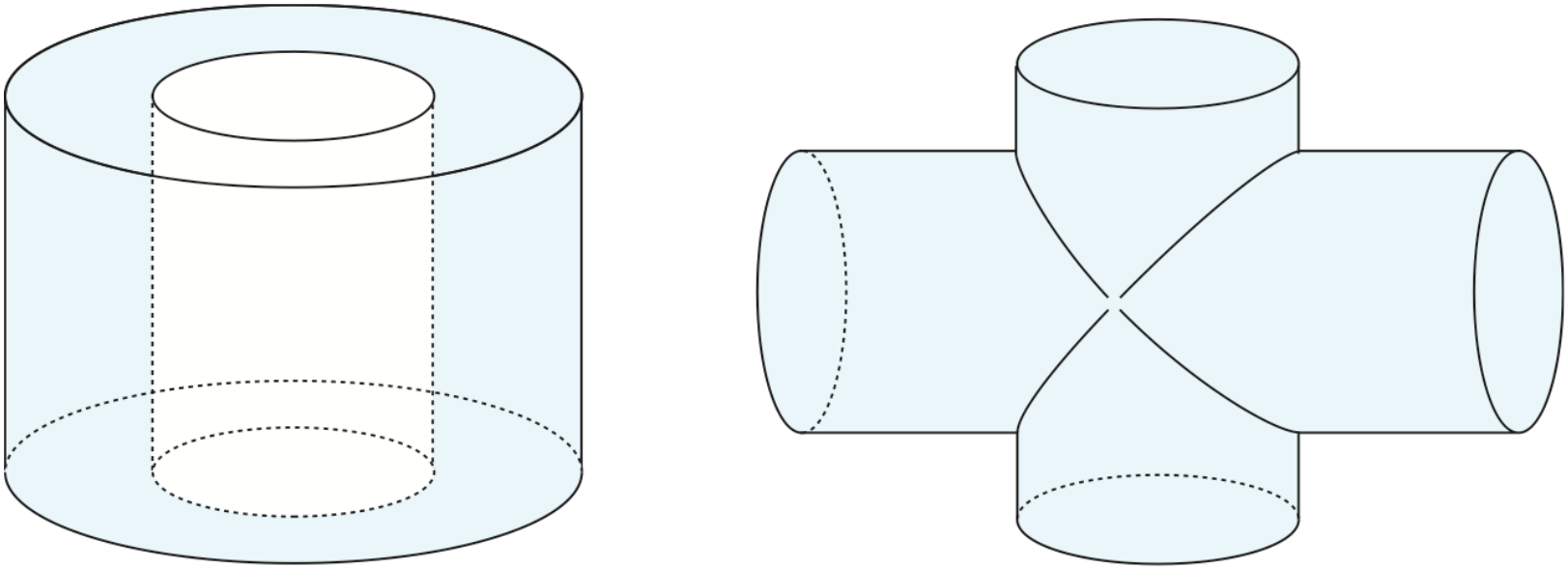


Figure 25.03



(a) Difference of cylinders

(b) Union of cylinders

Figure 25.04

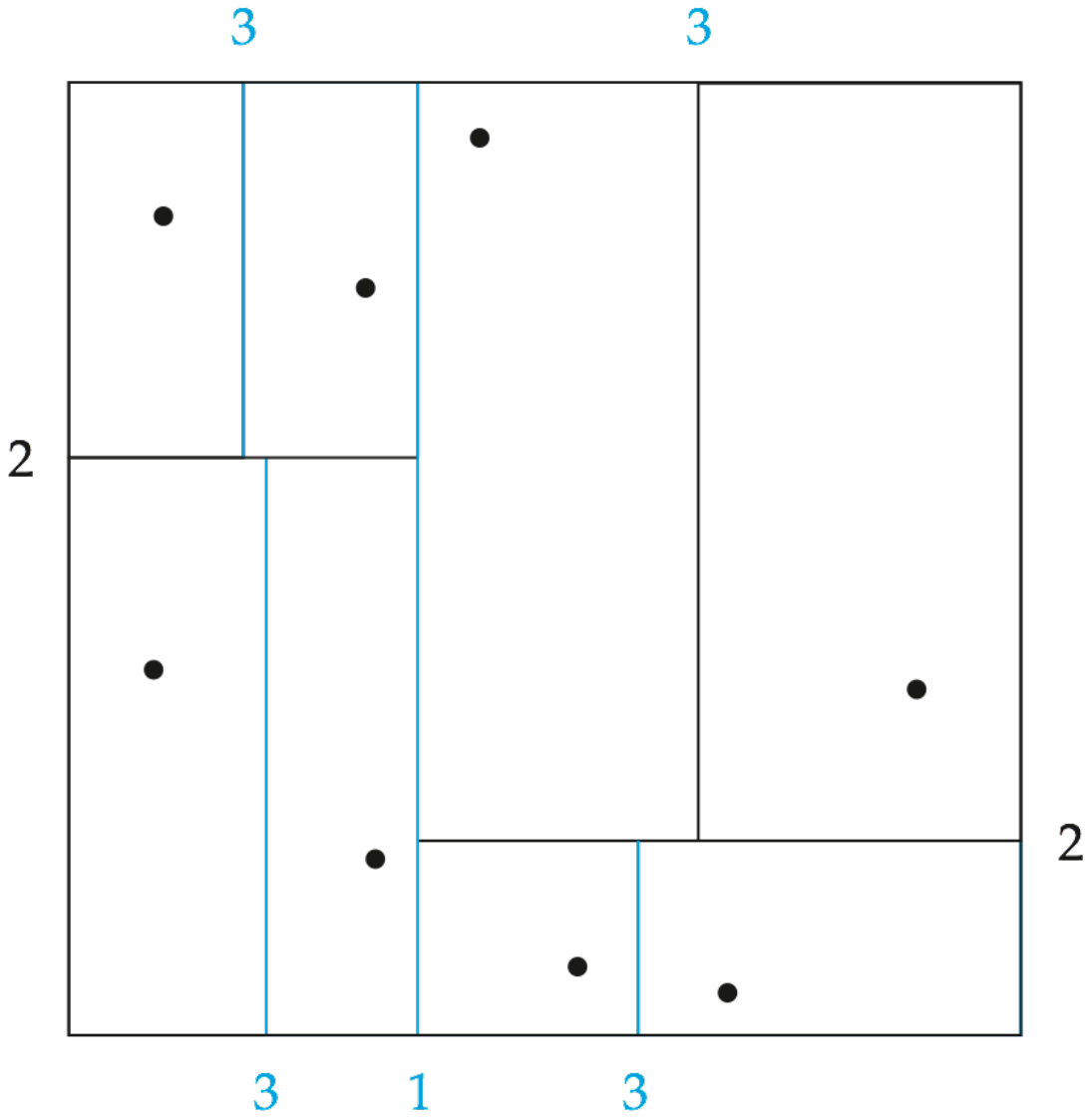


Figure 25.05

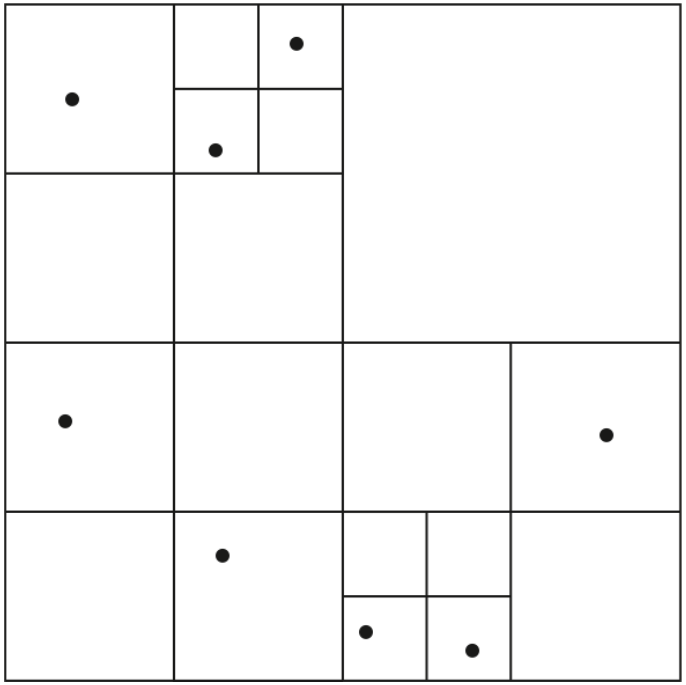


Figure 25.06