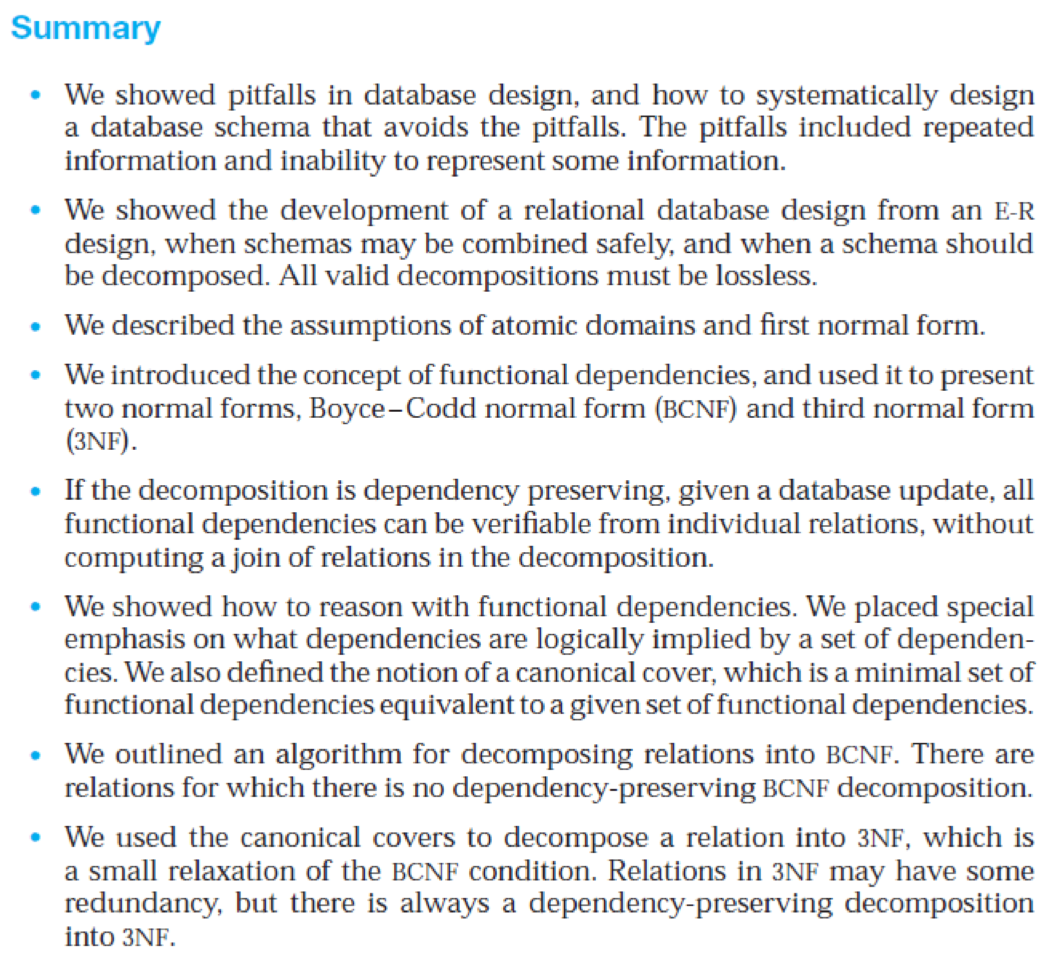
Fitxers i bases de dades
Resums dels principals conceptes

Model entitat relació (cap. 7)

Summary

- Database design mainly involves the design of the database schema. The entity-relationship (E-R) data model is a widely used data model for database design. It provides a convenient graphical representation to view data, relationships, and constraints.
- The E-R model is intended primarily for the database-design process. It was developed to facilitate database design by allowing the specification of an enterprise schema. Such a schema represents the overall logical structure of the database. This overall structure can be expressed graphically by an E-R diagram.
- An entity is an object that exists in the real world and is distinguishable from other objects. We express the distinction by associating with each entity a set of attributes that describes the object.
- A relationship is an association among several entities. A relationship set is a collection of relationships of the same type, and an entity set is a collection of entities of the same type.
- The terms superkey, candidate key, and primary key apply to entity and relationship sets as they do for relation schemas. Identifying the primary key of a relationship set requires some care, since it is composed of attributes from one or more of the related entity sets.
- Mapping cardinalities express the number of entities to which another entity can be associated via a relationship set.
- An entity set that does not have sufficient attributes to form a primary key is termed a weak entity set. An entity set that has a primary key is termed a strong entity set.
- The various features of the E-R model offer the database designer numerous choices in how to best represent the enterprise being modeled. Concepts and objects may, in certain cases, be represented by entities, relationships, or attributes. Aspects of the overall structure of the enterprise may be best described by using weak entity sets, generalization, specialization, or aggregation. Often, the designer must weigh the merits of a simple, compact model versus those of a more precise, but more complex, one.
- A database design specified by an E-R diagram can be represented by a collection of relation schemas. For each entity set and for each relationship set in the database, there is a unique relation schema that is assigned the name of the corresponding entity set or relationship set. This forms the basis for deriving a relational database design from an E-R diagram.

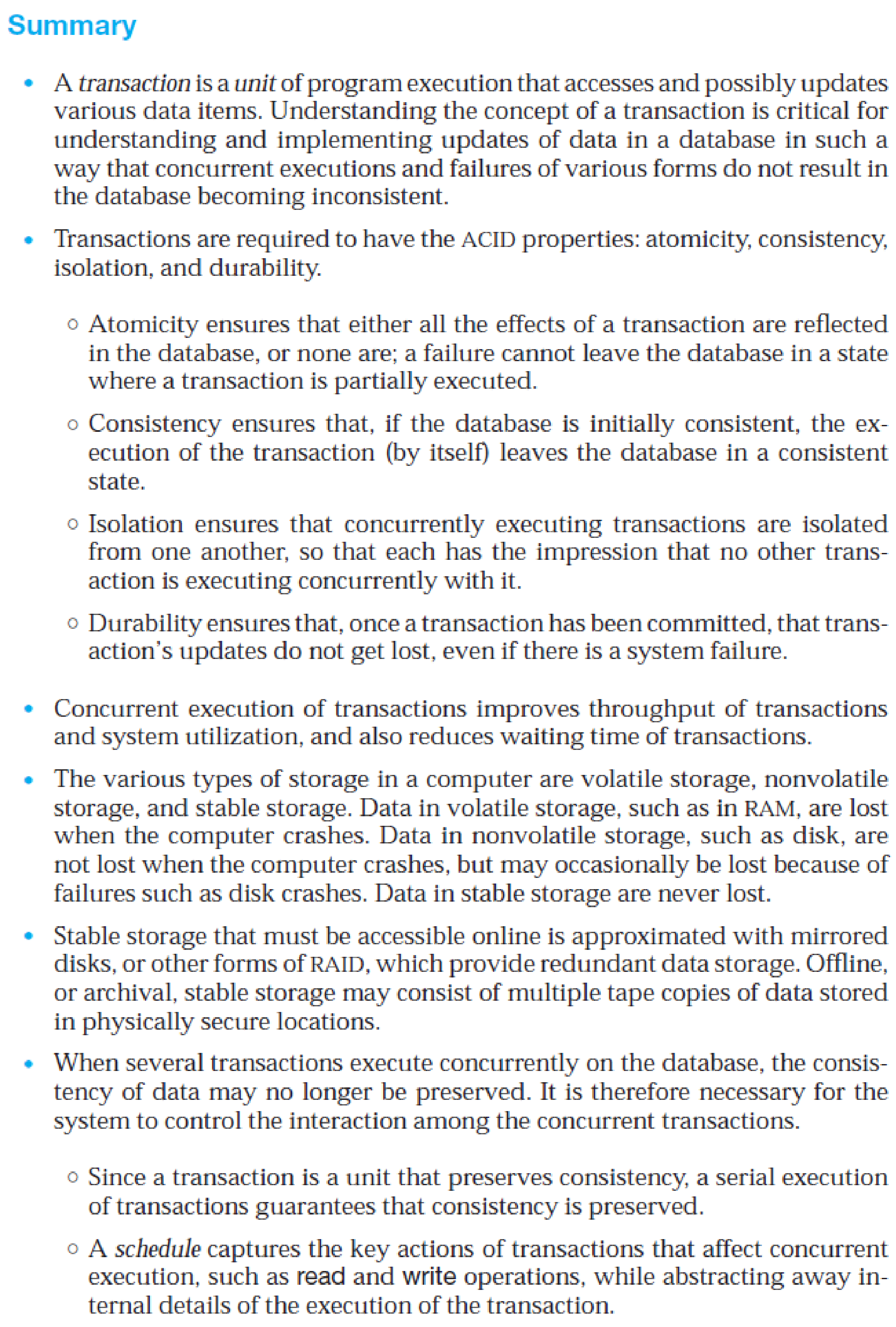
Disseny de bases de dades i normalització (cap. 8)



Summary

- We showed pitfalls in database design, and how to systematically design a database schema that avoids the pitfalls. The pitfalls included repeated information and inability to represent some information.
- We showed the development of a relational database design from an E-R design, when schemas may be combined safely, and when a schema should be decomposed. All valid decompositions must be lossless.
- We described the assumptions of atomic domains and first normal form.
- We introduced the concept of functional dependencies, and used it to present two normal forms, Boyce–Codd normal form (BCNF) and third normal form (3NF).
- If the decomposition is dependency preserving, given a database update, all functional dependencies can be verifiable from individual relations, without computing a join of relations in the decomposition.
- We showed how to reason with functional dependencies. We placed special emphasis on what dependencies are logically implied by a set of dependencies. We also defined the notion of a canonical cover, which is a minimal set of functional dependencies equivalent to a given set of functional dependencies.
- We outlined an algorithm for decomposing relations into BCNF. There are relations for which there is no dependency-preserving BCNF decomposition.
- We used the canonical covers to decompose a relation into 3NF, which is a small relaxation of the BCNF condition. Relations in 3NF may have some redundancy, but there is always a dependency-preserving decomposition into 3NF.

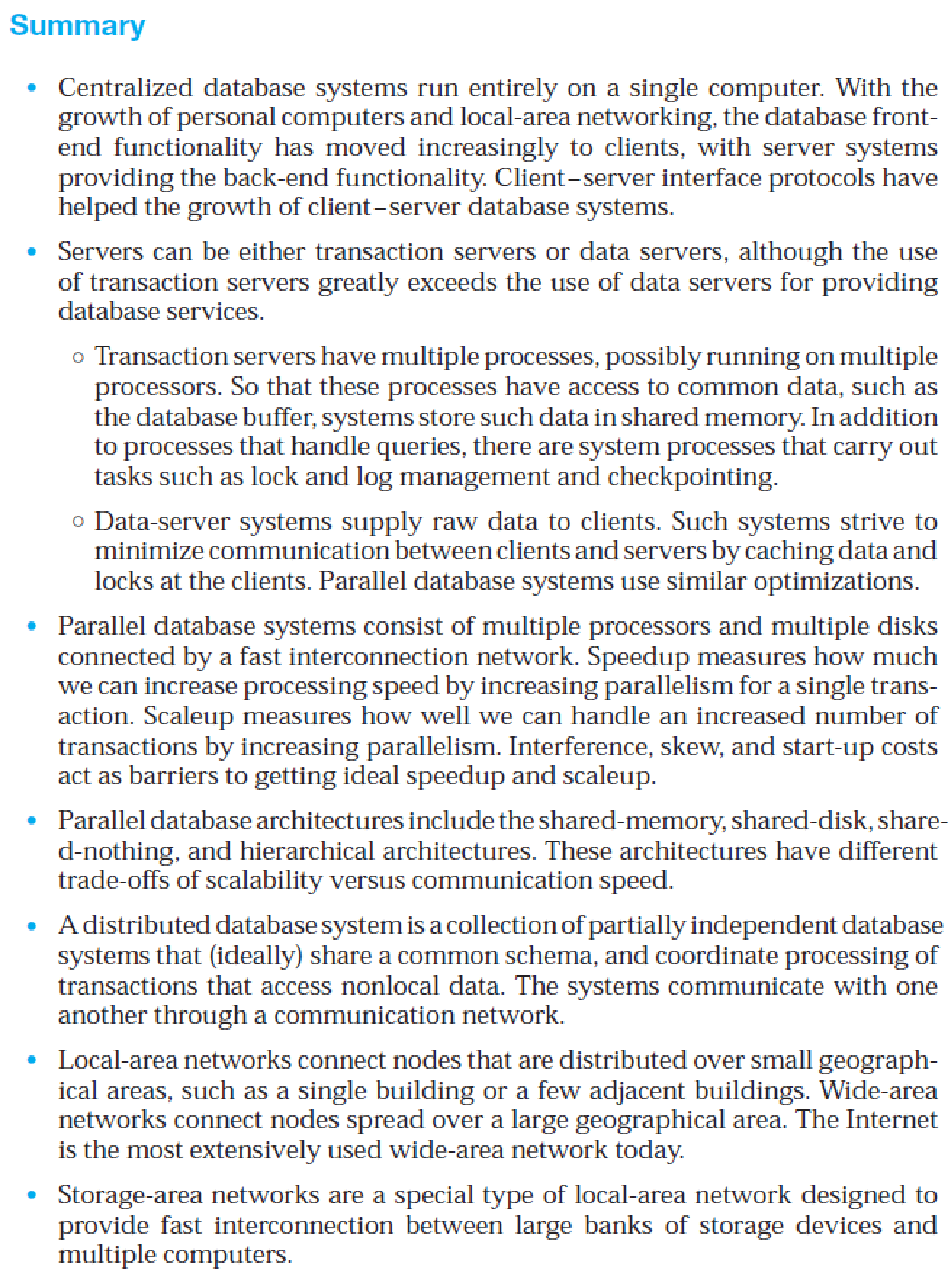
Transaccions (cap 14)



Summary

- A *transaction* is a *unit* of program execution that accesses and possibly updates various data items. Understanding the concept of a transaction is critical for understanding and implementing updates of data in a database in such a way that concurrent executions and failures of various forms do not result in the database becoming inconsistent.
- Transactions are required to have the ACID properties: atomicity, consistency, isolation, and durability.
  - Atomicity ensures that either all the effects of a transaction are reflected in the database, or none are; a failure cannot leave the database in a state where a transaction is partially executed.
  - Consistency ensures that, if the database is initially consistent, the execution of the transaction (by itself) leaves the database in a consistent state.
  - Isolation ensures that concurrently executing transactions are isolated from one another, so that each has the impression that no other transaction is executing concurrently with it.
  - Durability ensures that, once a transaction has been committed, that transaction's updates do not get lost, even if there is a system failure.
- Concurrent execution of transactions improves throughput of transactions and system utilization, and also reduces waiting time of transactions.
- The various types of storage in a computer are volatile storage, nonvolatile storage, and stable storage. Data in volatile storage, such as in RAM, are lost when the computer crashes. Data in nonvolatile storage, such as disk, are not lost when the computer crashes, but may occasionally be lost because of failures such as disk crashes. Data in stable storage are never lost.
- Stable storage that must be accessible online is approximated with mirrored disks, or other forms of RAID, which provide redundant data storage. Offline, or archival, stable storage may consist of multiple tape copies of data stored in physically secure locations.
- When several transactions execute concurrently on the database, the consistency of data may no longer be preserved. It is therefore necessary for the system to control the interaction among the concurrent transactions.
  - Since a transaction is a unit that preserves consistency, a serial execution of transactions guarantees that consistency is preserved.
  - A *schedule* captures the key actions of transactions that affect concurrent execution, such as read and write operations, while abstracting away internal details of the execution of the transaction.

Arquitectura de sistemes de bases de dades (cap. 17)



Summary

- Centralized database systems run entirely on a single computer. With the growth of personal computers and local-area networking, the database front-end functionality has moved increasingly to clients, with server systems providing the back-end functionality. Client–server interface protocols have helped the growth of client–server database systems.

- Servers can be either transaction servers or data servers, although the use of transaction servers greatly exceeds the use of data servers for providing database services.

  - Transaction servers have multiple processes, possibly running on multiple processors. So that these processes have access to common data, such as the database buffer, systems store such data in shared memory. In addition to processes that handle queries, there are system processes that carry out tasks such as lock and log management and checkpointing.

  - Data-server systems supply raw data to clients. Such systems strive to minimize communication between clients and servers by caching data and locks at the clients. Parallel database systems use similar optimizations.

- Parallel database systems consist of multiple processors and multiple disks connected by a fast interconnection network. Speedup measures how much we can increase processing speed by increasing parallelism for a single transaction. Scaleup measures how well we can handle an increased number of transactions by increasing parallelism. Interference, skew, and start-up costs act as barriers to getting ideal speedup and scaleup.

- Parallel database architectures include the shared-memory, shared-disk, share-d-nothing, and hierarchical architectures. These architectures have different trade-offs of scalability versus communication speed.

- A distributed database system is a collection of partially independent database systems that (ideally) share a common schema, and coordinate processing of transactions that access nonlocal data. The systems communicate with one another through a communication network.

- Local-area networks connect nodes that are distributed over small geographical areas, such as a single building or a few adjacent buildings. Wide-area networks connect nodes spread over a large geographical area. The Internet is the most extensively used wide-area network today.

- Storage-area networks are a special type of local-area network designed to provide fast interconnection between large banks of storage devices and multiple computers.

Desenvolupament d’aplicacions avançades (cap. 24)

Summary

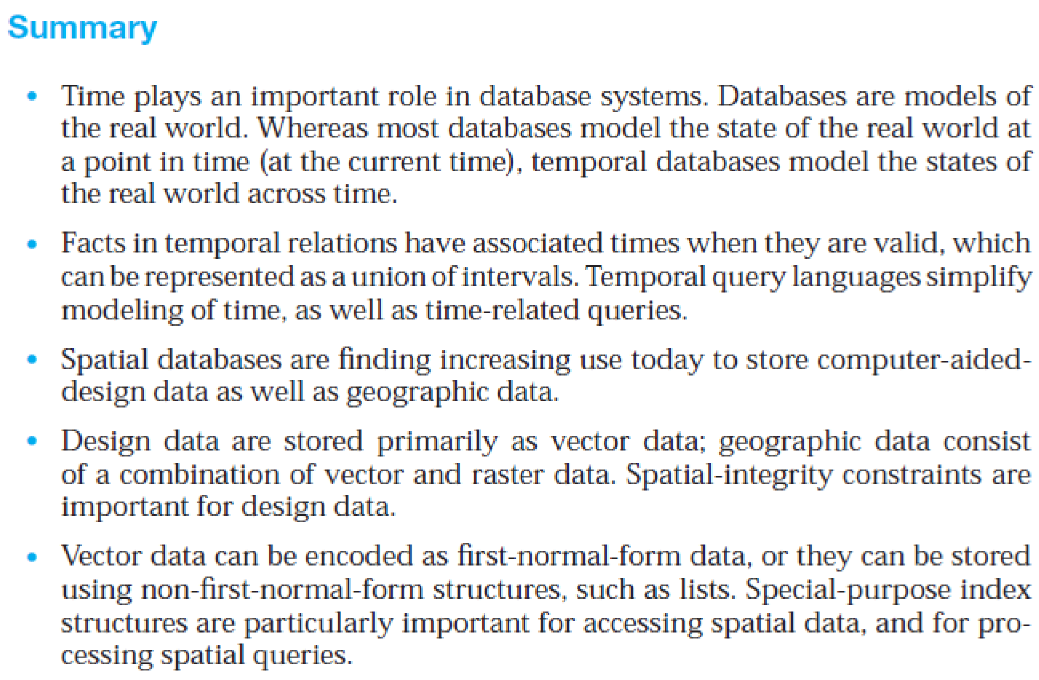
- Tuning of the database-system parameters, as well as the higher-level database design—such as the schema, indices, and transactions—is important for good performance. Queries can be tuned to improve set-orientation, while bulk-loading utilities can greatly speed up data import into a database.

  Tuning is best done by identifying bottlenecks and eliminating them. Database systems usually have a variety of tunable parameters, such as buffer sizes, memory size, and number of disks. The set of indices and materialized views can be appropriately chosen to minimize overall cost. Transactions can be tuned to minimize lock contention; snapshot isolation, and sequence numbering facilities supporting early lock release are useful tools for reducing read-write and write-write contention.

- Performance benchmarks play an important role in comparisons of database systems, especially as systems become more standards compliant. The TPC benchmark suites are widely used, and the different TPC benchmarks are useful for comparing the performance of databases under different workloads.

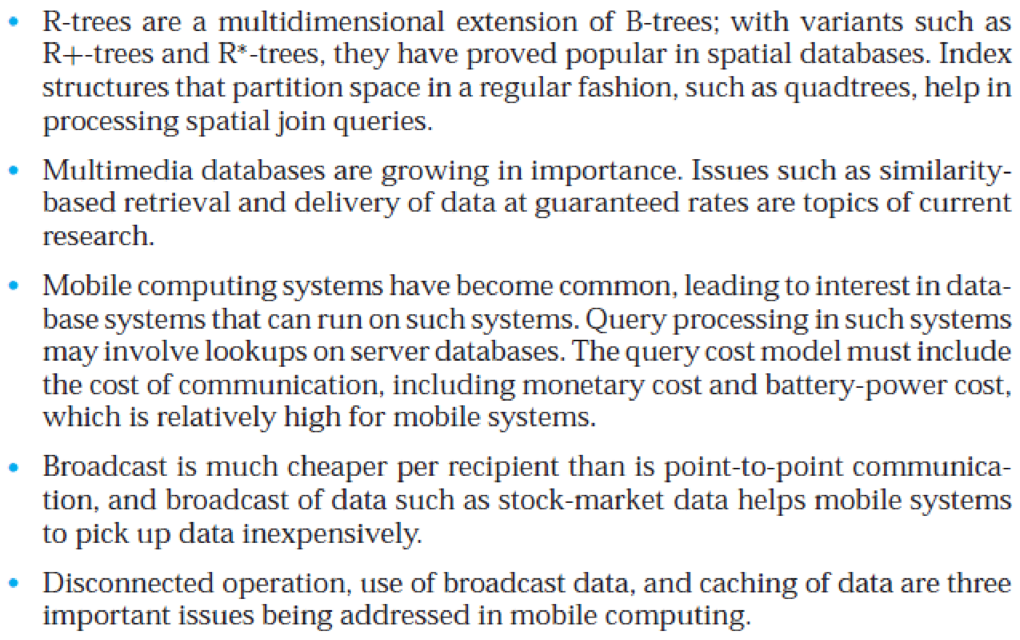
- Applications need to be tested extensively as they are developed, and before they are deployed. Testing is used to catch errors, as well as to ensure that performance goals are met.

Dades espacio-temporals i mobilitat (cap 25)



Summary

- Time plays an important role in database systems. Databases are models of the real world. Whereas most databases model the state of the real world 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. Temporal query languages simplify modeling of time, as well as time-related queries.
- Spatial databases are finding increasing use today to store computer-aided-design data as well as geographic data.
- Design data are stored primarily as vector data; geographic data consist of a combination of vector and raster data. Spatial-integrity constraints are important for design data.
- Vector data can be encoded as first-normal-form data, or they can be stored using non-first-normal-form structures, such as lists. Special-purpose index structures are particularly important for accessing spatial data, and for processing spatial queries.



- R-trees are a multidimensional extension of B-trees; with variants such as R+-trees and R*-trees, they have proved popular in spatial databases. Index structures that partition space in a regular fashion, such as quadtrees, help in processing spatial join queries.
- Multimedia databases are growing in importance. Issues such as similarity-based retrieval and delivery of data at guaranteed rates are topics of current research.
- Mobile computing systems have become common, leading to interest in database systems that can run on such systems. Query processing in such systems may involve lookups on server databases. The query cost model must include the cost of communication, including monetary cost and battery-power cost, which is relatively high for mobile systems.
- Broadcast is much cheaper per recipient than is point-to-point communication, and broadcast of data such as stock-market data helps mobile systems to pick up data inexpensively.
- Disconnected operation, use of broadcast data, and caching of data are three important issues being addressed in mobile computing.