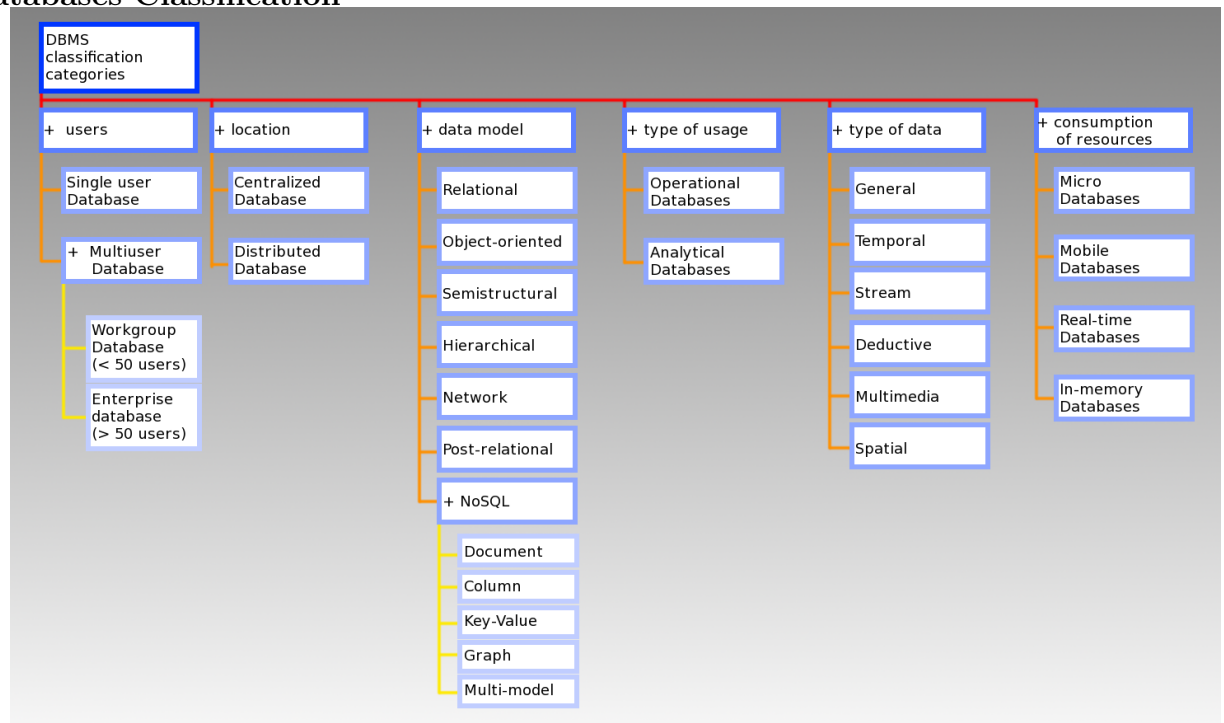
Table of Contents

Spis treści

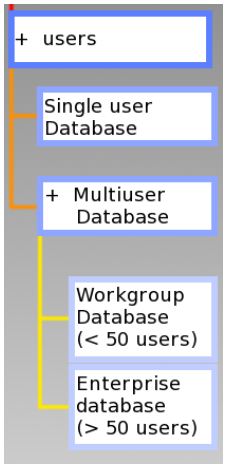
1 Classification of Database Systems because of:

Databases Classification



1.1 users

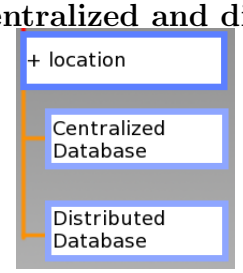
Single and multi user databases



- Single-user database allows only one connection to a database at a given moment in time.
- Multi-user database provides simultaneous access for multiple users.

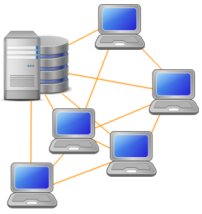
1.2 location

centralized and distributed databases



- Centralized Databases are located in a single location.
- Distributed Databases are located in many locations.

Centralized Databases



A centralized database
is a database that is located, stored, and maintained in a single location. This location is most often a central computer or database system, for example a desktop or server CPU, or a mainframe computer. In most cases, a centralised database would be used by an organisation (e.g. a business company) or an institution (e.g. a university.) Users access a centralised database through a computer network which is able to give them access to the central CPU, which in turn maintains to the database itself.

Advantages

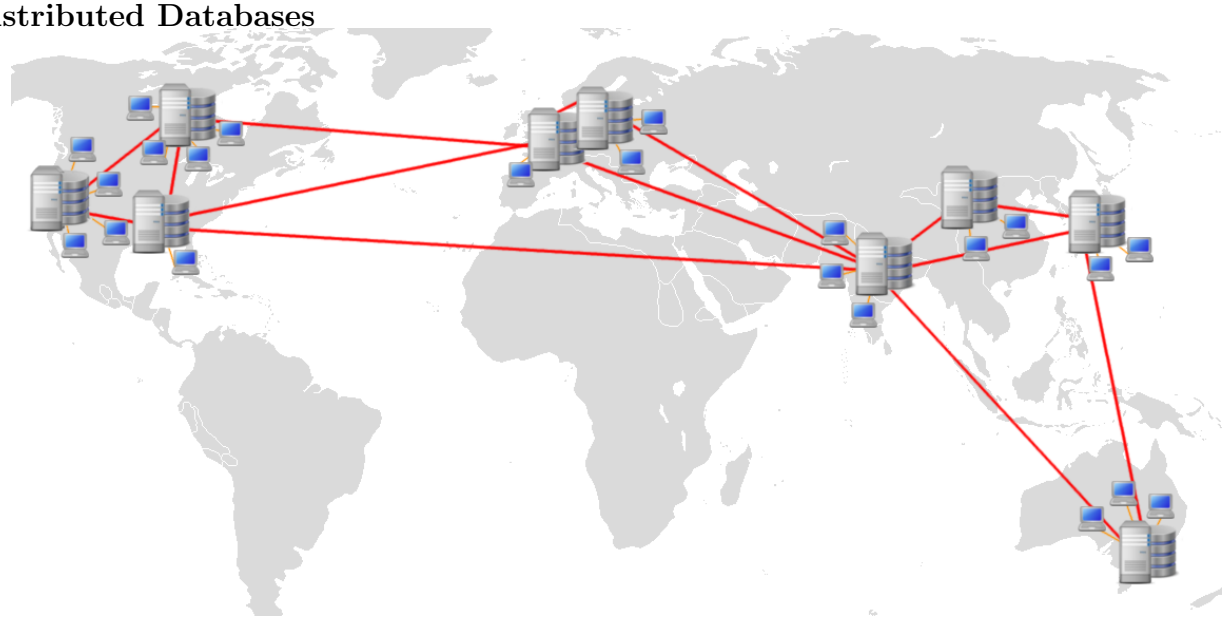
- Data integrity is maximised and data redundancy is minimised, as the single storing place of all the data also implies that a given set of data only has one primary record. This aids in the maintaining of data as accurate and as consistent as possible and enhances data reliability.
- Generally bigger data security, as the single data storage location implies only a one possible place from which the database can be attacked and sets of data can be stolen or tampered with.
- Better data preservation than other types of databases due to often-included fault-tolerant setup.

- Easier for use by the end-user due to the simplicity of having a single database design.
- Generally easier data portability and database administration.
- More cost effective than other types of database systems as labour, power supply and maintenance costs are all minimised.
- Data kept in the same location is easier to be changed, re-organised, mirrored, or analysed.
- All the information can be accessed at the same time from the same location.
- Updates to any given set of data are immediately received by every end-user.

Disadvantages

- Centralised databases are highly dependent on network connectivity. The slower the internet connection is, the longer the database access time needed will be.
- Bottlenecks can occur as a result of high traffic.
- Limited access by more than one person to the same set of data as there is only one copy of it and it is maintained in a single location. This can lead to major decreases in the general efficiency of the system.
- If there is no fault-tolerant setup and hardware failure occurs, all the data within the database will be lost.
- Since there minimal to none data redundancy, if a set of data is unexpectedly lost it is very hard to retrieve it back, and in most cases it would have to be done manually.

Distributed Databases



A distributed database
is a database that may be stored in multiple computers, located in the same physical location; or may be dispersed over a network of interconnected computers. Unlike parallel systems, in which the processors are tightly coupled and constitute a single database system, a distributed database system consists of loosely-coupled sites that share no physical components.

System administrators can distribute collections of data across multiple physical locations. A distributed database can reside on network servers on the Internet, on corporate intranets or extranets, or on other company networks. They can improve performance at end-user worksites by allowing transactions to be processed on many machines, instead of being limited to one.

Replication and Duplication

Two processes ensures that the distributed databases remain up-to-date and current:

- Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be complex and time-consuming depending on the size and number of the distributed databases. This process can also require a lot of time and computer resources.
- Duplication, has less complexity. It identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours. This is to ensure that each distributed location has the same data. In the duplication process, users may change only the master database. This ensures that local data will not be overwritten.

Architecture

- A **Homogeneous Distributed DBMS** has identical software and hardware running all databases instances, and may appear through a single interface as if it were a single database.
- A **Heterogeneous DDBMS** may have different hardware, operating systems, database management systems, and even data models for different databases.

Advantages

- Management of distributed data with different levels of transparency like network transparency, fragmentation transparency, replication transparency, etc.
- Increase reliability and availability
- Easier expansion
- Reflects organizational structure - database fragments potentially stored within the departments they relate to
- Local autonomy or site autonomy - a department can control the data about them (as they are the ones familiar with it)
- Protection of valuable data - if there were ever a catastrophic event such as a fire, all of the data would not be in one place, but distributed in multiple locations
- Improved performance - data is located near the site of greatest demand, and the database systems themselves are parallelized, allowing load on the databases to be balanced among servers. (A high load on one module of the database won't affect other modules of the database in a distributed database)
- Economics - it may cost less to create a network of smaller computers with the power of a single large computer

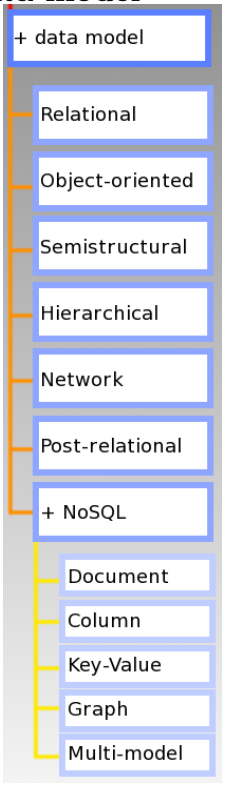
- Modularity - systems can be modified, added and removed from the distributed database without affecting other modules (systems)
- Reliable transactions - due to replication of the database
- Hardware, operating-system, network, fragmentation, DBMS, replication and location independence
- Continuous operation, even if some nodes go offline (depending on design)
- Distributed query processing can improve performance
- Single-site failure does not affect performance of system.

Disadvantages

- Complexity - DBAs may have to do extra work to ensure that the distributed nature of the system is transparent. Extra work must also be done to maintain multiple disparate systems, instead of one big one. Extra database design work must also be done to account for the disconnected nature of the database - for example, joins become prohibitively expensive when performed across multiple systems.
- Economics - increased complexity and a more extensive infrastructure means extra labour costs
- Security - remote database fragments must be secured, and they are not centralized so the remote sites must be secured as well. The infrastructure must also be secured (for example, by encrypting the network links between remote sites).
- Difficult to maintain integrity - but in a distributed database, enforcing integrity over a network may require too much of the network's resources to be feasible
- Inexperience - distributed databases are difficult to work with, and in such a young field there is not much readily available experience in "proper" practice
- Lack of standards - there are no tools or methodologies yet to help users convert a centralized DBMS into a distributed DBMS
- Database design more complex - In addition to traditional database design challenges, the design of a distributed database has to consider fragmentation of data, allocation of fragments to specific sites and data replication
- Additional software is required
- Operating system should support distributed environment
- Concurrency control poses a major issue. It can be solved by locking and timestamping.
- Distributed access to data
- Analysis of distributed data

1.3 data model

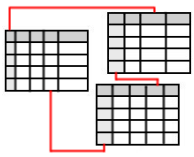
Data model



Logical data model
represents the abstract structure of a domain of information. Is independent of technology.

Physical data model
is a representation of a data design which takes into account the facilities and constraints of a given database management system. In the lifecycle of a project it derives from a logical data model.

Relational databases



Relational database
is based on the relational model of data, as proposed by E.F. Codd in 1970. This model organizes data into one or more tables (or "relations") of rows and columns, with a unique key for each row.

A relation is defined as a set of tuples that have the same attributes. A tuple usually represents an object and information about that object. Objects are typically physical objects or concepts. A relation is usually described as a table, which is organized into rows and columns. All the data referenced by an attribute are in the same domain and conform to the same constraints.

RDBMS

Relational database management system
is based on the relational model.

According to research company Gartner, the five leading commercial relational database vendors by revenue in 2011 were Oracle (48.8%), IBM (20.2%), Microsoft (17.0%), SAP including Sybase (4.6%), and Teradata (3.7%).

The three leading open source implementations are MySQL, PostgreSQL, and SQLite. MariaDB is a prominent fork of MySQL prompted by Oracle's acquisition of MySQL AB.

Advantages

- Ease of use: The revision of any information as tables consisting of rows and columns is much easier to understand .

- Flexibility: Different tables from which information has to be linked and extracted can be easily manipulated by operators such as project and join to give information in the form in which it is desired.

- Precision: The usage of relational algebra and relational calculus in the manipulation of he relations between the tables ensures that there is no ambiguity, which may otherwise arise in establishing the linkages in a complicated network type database.

- Security: Security control and authorization can also be implemented more easily by moving sensitive attributes in a given table into a separate relation with its own authorization controls. If authorization requirement permits, a particular attribute could be joined back with others to enable full information retrieval.

- Data Independence: Data independence is achieved more easily with normalization structure used in a relational database than in the more complicated tree or network structure.

- Data Manipulation Language: The possibility of responding to query by means of a language based on relational algebra and relational calculus e.g SQL is easy in the relational database approach. For data organized in other structure the query language either becomes complex or extremely limited in its capabilities.

- The ability to use modern techniques: multiprocessing, cross-platform, parallelism, advanced methods of optimization, advanced security methods, an extended system of transactions.

Disadvantages

- Performance: A major constraint and therefore disadvantage in the use of relational database system is machine performance. If the number of tables between which relationships to be established are large and the tables themselves effect the performance in responding to the sql queries.

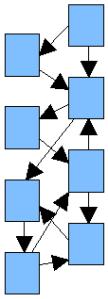
- Physical Storage Consumption: With an interactive system, for example an operation like join would depend upon the physical storage also. It is, therefore common in relational databases to tune the databases and in such a case the physical data layout would be chosen so as to give good performance in the most frequently run operations. It therefore would naturally result in the fact that the lays frequently run operations would tend to become even more shared.

- Slow extraction of meaning from data: if the data is naturally organized in a hierarchical manner and stored as such, the hierarchical approach may give quick meaning for that data.

- Weak set of types and structures: lack of user functions, procedures, types, the problem of hierarchisation, no data nested, lack of inheritance, does not include the timing and versioning.

- Mismatch between the data manipulation language and application programming languages.

Object-oriented databases



Object database
is a database management system in which information is represented in the form of objects as used in object-oriented programming.

OODBMSs allow object-oriented programmers to develop the product, store them as objects, and replicate or modify existing objects to make new objects within the OODBMS. Because the database is integrated with the programming language, the programmer can maintain consistency within one environment, in that both the OODBMS and the programming language will use the same model of representation. Relational DBMS projects, by way of contrast, maintain a clearer division between the database model and the application. Examples: Caché, ConceptBase, Db4o, GemStone/S, NeoDatis ODB, ObjectDatabase++, ObjectDB, Objectivity/DB, ObjectStore, Perst, VelocityDB, WakandaDB, ZooDB.

Technical features

- Most object databases also offer some kind of query language, allowing objects to be found using a declarative programming approach. It is in the area of object query languages, and the integration of the query and navigational interfaces, that the biggest differences between products are found. An attempt at standardization was made by the ODMG with the Object Query Language, OQL.
- Access to data can be faster because joins are often not needed (as in a tabular implementation of a relational database). This is because an object can be retrieved directly without a search, by following pointers.
- Many object databases, offer support for versioning. An object can be viewed as the set of all its versions. Also, object versions can be treated as objects in their own right. Some object databases also provide systematic support for triggers and constraints which are the basis of active databases.
- The efficiency of such a database is also greatly improved in areas which demand massive amounts of data about one item. x

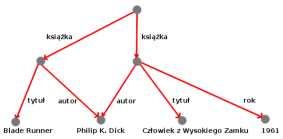
Advantages

- Objects don't require assembly and disassembly saving coding time and execution time to assemble or disassemble objects.
- Reduced paging
- Easier navigation
- Better concurrency control - A hierarchy of objects may be locked.
- Data model is based on the real world.
- Works well for distributed architectures.
- Less code required when applications are object oriented.

Disadvantages

- Lower efficiency when data is simple and relationships are simple.
- Relational tables are simpler.
- Late binding may slow access speed.
- More user tools exist for RDBMS.
- Standards for RDBMS are more stable.
- Support for RDBMS is more certain and change is less likely to be required.

Semistructured databases



Semistructured database
is a database in which there is no separation between the data and the schema, and the amount of structure used depends on the purpose.

The advantages of this model are the following:

- It can represent the information of some data sources that cannot be constrained by schema.
- It provides a flexible format for data exchange between different types of databases.
- It can be helpful to view structured data as semi-structured (for browsing purposes).
- The schema can easily be changed.
- The data transfer format may be portable.

XML databases

XML databases
allows data to be specified, and sometimes stored, in XML format. These data can then be queried, transformed, exported and returned to a calling system.

In different publications they are treated as a semistructural or NoSQL (document oriented) types of databases. Examples of native XML databases:

- dbXML
- eXist
- Sedna
- BaseX

XML databases

- Data-centric XML retrieval:
  - XML is used as a transport medium
  - the regular structure of documents
  - data on the level with limited detail
  - data comes from a database or will be entered into the database
- Document-centric:
  - documents mostly are created manually
  - irregular structure of documents
  - relatively large granularity of data

Post-relational databases

Post-relational databases
a conventional name for database offering a hybrid approach: a combination of relational, object-oriented, semi-structured databases and so on.

Some of these extensions to the relational model integrate concepts from technologies that pre-date the relational model. For example, they allow representation of a directed graph with trees on the nodes.

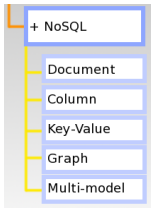
Object-relational database

ORDBMS
is a database management system (DBMS) similar to a relational database, but with an object-oriented database model: objects, classes and inheritance are directly supported in database schemas and in the query language. In addition, just as with pure relational systems, it supports extension of the data model with custom data-types and methods.

ORDBMS offers:

- multimedia and spatial data,
- complex, abstract and reference types, collections, records, sets, lists, object-oriented perspectives, object tables, nested tables, arrays of variable length, etc.
- classes, methods, functions and procedures defined in other languages, inheritance, aggregation, overloading,
- SQL3 query language also called ObjectSQL.

NoSQL databases



NoSQL database
(often interpreted as Not only SQL) database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Motivations for this approach include simplicity of design, horizontal scaling, and finer control over availability. The data structures used by NoSQL databases (e.g. key-value, graph, or document) differ from those used in relational databases, making some operations faster in NoSQL and others faster in relational databases.

Types of NoSQL databases

- Column (two dimensional arrays whereby each key (i.e. row / record) has one or more key / value pairs attached to it and these management systems allow very large and unstructured data to be kept and used). Example: Accumulo, Cassandra, Druid, HBase, Vertica
- Document (documents encapsulate and encode data (or information) in some standard formats or encodings. Encodings in use include XML, YAML, and JSON as well as binary forms like BSON). Examples: Lotus Notes, Clusterpoint, Apache CouchDB, Couchbase, MarkLogic, MongoDB, OrientDB, Qizx

- Key-value (data is represented as a collection of key-value pairs, such that each possible key appears at most once in the collection). Examples: CouchDB, Dynamo, FoundationDB, MemcacheDB, Redis, Riak, FairCom c-treeACE, Aerospike, OrientDB, MUMPS
- Graph (for data whose relations are well represented as a graph - elements interconnected with an undetermined number of relations between them; this kind of data could be social relations, public transport links, road maps or network topologies). Examples: Allegro, Neo4J, InfiniteGraph, OrientDB, Virtuoso, Stardog
- Multi-model: OrientDB, FoundationDB, ArangoDB, Alchemy Database, CortexDB

Advantages

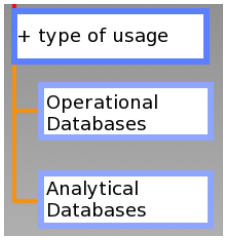
- If will be working with very large sets of data, consistently scaling is easier to achieve with many of the DBMS from NoSQL family.
- NoSQL databases are usually faster - and sometimes extremely speedier - when it comes to writes. Reads can also be very fast depending on the type of NoSQL database and data being queried.
- Relational DBMSs require structure from the beginning. NoSQL solutions offer a large amount of flexibility.
- NoSQL databases are growing rapidly and they are being actively built today - vendors are trying to tackle common issues and one of them clearly is replication and scaling.
- When it comes to choosing a NoSQL data store, there are a variety of models, as we have discussed, that you can choose from to get the most out of the database management system - depending on your data type.

Disadvantages

- Most NoSQL systems are in pre-production versions with many key features yet to be implemented.
- Low level of enterprise support.
- NoSQL databases offer few facilities for queries and analysis since they do not work with SQL. Things that would otherwise require simple queries in RDBMS require significant programming expertise when using NoSQL databases.
- Require a significant level of skill and effort to install and maintain.
- Almost every NoSQL developer is in learning mode.

1.4 type of usage

Operational vs Analytical Databases



- **Operational databases** (also referred to as OLTP On Line Transaction Processing databases), are used to manage dynamic data in real-time, emphasis on transaction efficiency, supports day to day operations.
- **Analytical databases** (also referred to as OLAP On Line Analytical Processing databases) enable users to analyze multidimensional data interactively from multiple perspectives.

Operational databases

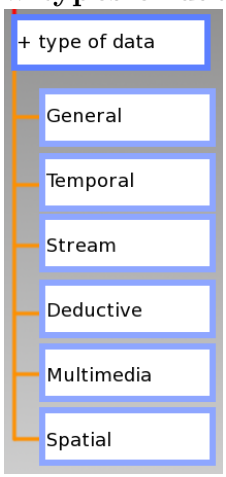
- Operational databasses are used to store, manage and track real-time business information.
- Operational databases allow a business to enter, gather, and retrieve large quantities of specific information, such as company legal data, financial data, call data records, personal employee information, sales data, customer data, data on assets and many other information.
- Operational databases can be used to manage mission-critical business data, to monitor activities, to audit suspicious transactions, or to review the history of dealings with a particular customer.
- They can also be part of the actual process of making and fulfilling a purchase, for example in e-commerce.
- They are optimized for faster transaction processing (create, read, update and delete operations).

Analytical databases

- Supports tactical or strategic decision making.
- They consist of OLAP tools, and data warehouse system.
- They emphasis on integration of data elements to provide o coherent picture.
- Databases configured for OLAP use a multidimensional data model, allowing for complex analytical and ad hoc queries with a rapid execution time.
- They borrow aspects of navigational databases, hierarchical databases and relational databases.
- At the core of any OLAP system is an OLAP cube (also called a 'multidimensional cube' or a hypercube). It consists of numeric facts called measures which are categorized by dimensions. The measures are placed at the intersections of the hypercube, which is spanned by the dimensions as a vector space.

1.5 type of data

New types of data - new challenges



Standard data types and standard database can not cope with new types that are coming to us from various sources. We have continuous data from CCTV cameras, multimedia data, the data that must take into account the time or logical inference.

Temporal databases

Temporal databases

- is a database with built-in support for handling data involving time, being related to the slowly changing dimension concept, for example a temporal data model and a temporal version of Structured Query Language (SQL).

More specifically the temporal aspects usually include valid time and transaction time. These attributes can be combined to form bitemporal data:

- Valid time is the time period during which a fact is true with respect to the real world.
- Transaction time is the time period during which a fact stored in the database is considered to be true.
- Bitemporal data combines both Valid and Transaction Time.

Examples: TimeDB, Tiger.

Features

- A time period datatype, including the ability to represent time periods with no end (infinity or forever)
- The ability to define valid and transaction time period attributes and bitemporal relations
- System-maintained transaction time
- Temporal primary keys, including non-overlapping period constraints
- Temporal constraints, including non-overlapping uniqueness and referential integrity
- Update and deletion of temporal records with automatic splitting and coalescing of time periods
- Temporal queries at current time, time points in the past or future, or over durations
- Predicates for querying time periods, often based on Allen's interval relations

Data stream management system

A Data stream management system (DSMS)
is a computer program to manage continuous data streams. It is similar to a database management system (DBMS), which is, however, designed for static data in conventional databases. A DSMS also offers a flexible query processing so that the information need can be expressed using queries. However, in contrast to a DBMS, a DSMS executes a continuous query that is not only performed once, but is permanently installed. Therefore, the query is continuously executed until it is explicitly uninstalled.

Features

- continuous query (an implementation plan is locked in a dead loop),
- data stream is an infinite (specific operators of join and aggregate functions),
- queries predefined and ad-hoc asked,
- data streams algebra.

Application

- medical monitoring systems,
- control of sensor networks,
- monitoring data traffic over networks,
- support of current financial analysis,
- ongoing analysis of the transaction,
- military applications, etc

Examples: Stream, Borealis, TelegraphCQ, NiagaraCQ, Cougar.

Deductive databases

A Deductive database
is a database system that can make deductions (i.e., conclude additional facts) based on rules and facts stored in the (deductive) database. Datalog is the language typically used to specify facts, rules and queries in deductive databases.

Deductive databases have grown out of the desire to combine logic programming with relational databases to construct systems that support a powerful formalism and are still fast and able to deal with very large datasets. Deductive databases are more expressive than relational databases but less expressive than logic programming systems. In recent years, deductive databases such as Datalog have found new application in data integration, information extraction, networking, program analysis, security, and cloud computing.

Multimedia databases

A Multimedia databases
(MMDB) is a collection of related multimedia data. The multimedia data include one or more primary media data types such as text, images, graphic objects (including drawings, sketches and illustrations) animation sequences, audio and video.

Data classification:

- continuous (audio, speech, animation, video) - taking into account the time dimension
- discrete (text, image, computer graphics, conventional types) - without taking into account the time.

Challenges

- The large size of the data,
- the lack of a standard for multimedia information storage,
- multimedia data transmission method is not standardized,
- problems of compression,
- synchronization during playback of various media elements,
- query language adapted to multimedia data.

Spatial databases

A Spatial databases
or geodatabase is a database that is optimized to store and query data that represents objects defined in a geometric space. Most spatial databases allow representing simple geometric objects such as points, lines and polygons. Some spatial databases handle more complex structures such as 3D objects, topological coverages, linear networks, and TINs.

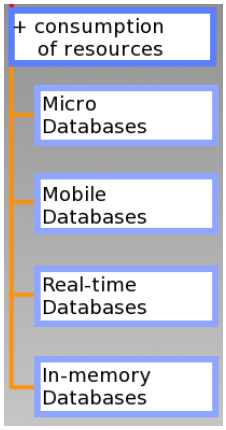
While typical databases are designed to manage various numeric and character types of data, additional functionality needs to be added for databases to process spatial data types efficiently. These are typically called geometry or feature. The Open Geospatial Consortium created the Simple Features specification and sets standards for adding spatial functionality to database systems.

Features

- Spatial Measurements: Computes line length, polygon area, the distance between geometries, etc.
- Spatial Functions: Modify existing features to create new ones, for example by providing a buffer around them, intersecting features, etc.sposób przesyłania danych multimedialnych nie jest ujednolicony
- Spatial Predicates: Allows true/false queries about spatial relationships between geometries. Examples include "do two polygons overlap" or 'is there a residence located within a mile of the area we are planning to build the landfill?'
- Geometry Constructors: Creates new geometries, usually by specifying the vertices (points or nodes) which define the shape.
- Observer Functions: Queries which return specific information about a feature such as the location of the center of a circle

1.6 consumption of resources

Differences in consumption of resources



In the case of modern databases, we must also take into account the nature of the memory resources used, for example, for small mobile devices.

Micro databases

Micro databases
- stripped down, simplified database system.

Features:

- simplicity,
- functionality satisfying the needs of many applications,
- transferability,
- database often in a single file,
- micro-size,
- compile SQL commands to the virtual machine code (SQLLite).

Examples: SQLite, HSQLDB, tinySQL, picoSQL, Mckoi SQL, Axion etc...

Mobile databases

Mobile databases
- is either a stationary database that can be connected to by a mobile computing device (e.g., smartphones and PDAs) over a mobile network, or a database which is actually stored by the mobile device. This could be a list of contacts, price information, distance travelled, or any other information.

Examples: SQL Anywhere, DB2 Everyplace, SQL Server Compact, SQL Server Express, Oracle Database Lite, SQLite, SQLBase, Sparksee 5 mobile, Couchbase Lite, iBoxDB, Realm, CryptonorDB

Considerations

- Mobile users must be able to work without a network connection due to poor or even non-existent connections. A cache could be maintained to hold recently accessed data and transactions so that they are not lost due to connection failure. Users might not require access to truly live data, only recently modified data, and uploading of changing might be deferred until reconnected.
- Bandwidth must be conserved (a common requirement on wireless networks that charge per megabyte or data transferred).
- Mobile computing devices tend to have slower CPUs and limited battery life.

- Users with multiple devices (e.g. smartphone and tablet) need to synchronize their devices to a centralized data store. This may require application-specific automation features.
- Users may change location geographically and on the network. Usually dealing with this is left to the operating system, which is responsible for maintaining the wireless network connection.

Real-time Database System

Real-time Database System
- is a database system which uses real-time processing to handle workloads whose state is constantly changing. This differs from traditional databases containing persistent data, mostly unaffected by time.

Real-time processing means that a transaction is processed fast enough for the result to come back and be acted on right away. Real-time databases are useful for accounting, banking, law, medical records, multi-media, process control, reservation systems, and scientific data analysis.

Main Memory Database System

In-memory database
is a database management system that primarily relies on main memory for computer data storage. It is contrasted with database management systems that employ a disk storage mechanism. Main memory databases are faster than disk-optimized databases since the internal optimization algorithms are simpler and execute fewer CPU instructions. Accessing data in memory eliminates seek time when querying the data, which provides faster and more predictable performance than disk.

Examples: DataBlitz, FastDB, Oracle TimesTen, Polyhedra, eXtremeDB, Monet, Tachyon.

Features

- very fast execution of queries (using of specific algorithms)
- vertical fragmentation of columns
- immediate recovery after server error
- pre-downloading of data,
- the need of using traditional databases as a back-office,
- sorting of join indexes.

2 Resources

Resources

- `http://en.wikipedia.org`
- S. Kozielski, B. Małysiak, P. Kasprowski, D. Mrożek, Bazy Danych: Modele, Technologie, Narzędzia, WKŁ 2005
- `http://www.cs.put.poznan.pl/mmorzy/`

- http://www.ipipan.eu/staff/k.subieta/prezentacje/Obiektowe\%20bazy\%20danych\%20kontra\%20relacyjne\%201997.ppt
- C.Zaniolo, S.Ceri, Ch.Faloutsos, R.T. Snodgrass, V. S. Subrahmanian, R.Zicari, Advanced Database Systems, Morgan Kaufmann, 1997
- http://putwiki.informatyka.org/wiki/Kategoria:Zaawansowane_systemy_baz_danych
- http://wazniak.mimuw.edu.pl/index.php?title=Zaawansowane_systemy_baz_danych
- Materiały do wykładów Pani mgr. wandy Kik
- "O modelach danych i innych rzeczach, które utrudniają życie informatykom" Krzysztof Goczyła, VI Konferencja PLOUG 2000
- M. Lentner, Oracle 9i Kompletny podręcznik użytkownika, PJWSTK - W-wa, 2003
- http://www.odbms.org/
- http://www.objectivity.com/pages/objectivity/faq.asp
- http://fatcat.ftj.agh.edu.pl/~wojdyla/anno/art.php.htm
- http://maxlogix.blogspot.com/2009/09/advantages-and-disadvantages-of-using.html