Enhanced Data Models

As the use of database systems has grown, users have demanded additional functionality from these software packages, with the purpose of making it easier to implement more advanced and complex user applications. Object-oriented databases and object relational systems do provide features that allow users to extend their systems by specifying additional abstract data types for each application. However, it is quite useful to identify certain common features for some of these advanced applications and to create models that can represent them.

- Active database & triggers
- Temporal databases
- Spatial and Multimedia databases
- Deductive databasesν

Active database & triggers :

An active database system is a database management system endowed with active rules, i.e. stored procedures activated by the system when specific events occur.

- Typically an active rule consists of three parts: event, condition, and action.
- Normally written as: WHEN event IF condition THEN action.
  "event" specifies a list of events,
  "Condition" is a query on the database,
  "action" generally consists of one or more updates or queries on the database.

- Automated bill payments are an example of an active database. A bank customer may instruct his institution to pay a payee a specific amount on a certain date each month. When the specified date is reached, the electronic payments are automatically sent to the payees indicated by the information in the database

Temporal Database Concepts

A **temporal database** stores data relating to time instances. It offers temporal data types and stores information relating to past, present and future time. Temporal databases can be uni-temporal, bi-temporal or tri-temporal.

More specifically the temporal aspects usually include valid time, transaction time or decision time.

- **Valid time** is the time period during or event time at which a fact is true in the real world.
- **Transaction time** is the time at which a fact was recorded in the database.
- **Decision time** is the time at which the decision was made about the fact.

Uni-temporal

A uni-temporal database has one axis of time, either the validity range or the system time range.

Bi-temporal

A bi-temporal database has two axes of time:

- valid time
- transaction time or decision time

Tri-temporal

A tri-temporal database has three axes of time:

- valid time
- transaction time
- decision time

This approach introduces additional complexities.

Temporal databases are in contrast to current databases (not to be confused with currently available databases), which store only facts which are believed to be true at the current time.

Using a non-temporal database

Date	Real world event	Database action	What the database shows

April 3, 1975	John is born	Nothing	There is no person called John Doe
April 4, 1975	John's father officially reports John's birth	Inserted:Person(John Doe, Smallville)	John Doe lives in Smallville
August 26, 1994	After graduation, John moves to Bigtown, but forgets to register his new address	Nothing	John Doe lives in Smallville
December 26, 1994	Nothing	Nothing	John Doe lives in Smallville
December 27, 1994	John registers his new address	Updated:Person(John Doe, Bigtown)	John Doe lives in Bigtown
April 1, 2001	John dies	Deleted:Person(John Doe)	There is no person called John Doe

Using a single axis: valid time or transaction time

For the example above, to record valid time, the *Person* table has two fields added, *Valid-From* and *Valid-To*. These specify the period when a person's address is valid in the real world. On April 4, 1975, John's father registered his son's birth. An official then inserts a new entry into the database stating that John lives in Smallville from April 3. Note that although the data was inserted on the fourth, the database states that the information is valid since the third. The official does not yet know if or when John will move to another place, so the *Valid-To* field is set to infinity (∞). The entry in the database is:

```
Person(John Doe, Smallville, 3-Apr-1975, ∞).
```

On December 27, 1994, John reports his new address in Bigtown where he has been living since August 26, 1994. A new database entry is made to record this fact:

```
Person(John Doe, Bigtown, 26-Aug-1994, ∞).
```

The original entry `Person (John Doe, Smallville, 3-Apr-1975, ∞)` is not deleted, but has the *Valid-To* attribute updated to reflect that it is now known that John stopped living in Smallville on August 26, 1994. The database now contains two entries for John Doe

```
Person(John Doe, Smallville, 3-Apr-1975, 26-Aug-1994).
Person(John Doe, Bigtown, 26-Aug-1994, ∞).
```

When John dies his current entry in the database is updated stating that John does not live in Bigtown any longer. The database now looks like this

```
Person(John Doe, Smallville, 3-Apr-1975, 26-Aug-1994).
Person(John Doe, Bigtown, 26-Aug-1994, 1-Apr-2001).
```

Spatial database:

A **spatial database** is a general-purpose database (usually a relational database) that has been enhanced to include spatial data that represents objects defined in a geometric space, along with tools for querying and analyzing such data. Most spatial databases allow the representation of 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 triangulated irregular networks (TINs). While typical databases have developed to manage various numeric and character types of data, such databases require additional functionality to process spatial data types efficiently, and developers have often added *geometry* or *feature* data types. The Open Geospatial Consortium (OGC) developed the Simple Features specification (first released in 1997) and sets standards for adding spatial functionality to database systems.[2] The *SQL/MM Spatial* ISO/IEC standard is a part of the SQL/MM multimedia standard and extends the Simple Features standard with data types that support circular interpolations.[3] Almost all current relational and object-relational database management systems now have spatial extensions, and some GIS software vendors have developed their own spatial extensions to database management systems.

Geographic database (or **geodatabase**) is a georeferenced spatial database, used for storing and manipulating geographic data (or geodata), i.e., data associated with a location on Earth. The term "geodatabase" may also refer specifically to a set of proprietary spatial database formats, Geodatabase

Multimedia Databases

Multimedia data is an interactive way to represent information to a user. It includes several categories of data like textual data, audio data, video data, etc.
The **database** which is used to hold these different kinds of multimedia data is known as a **multimedia database**.

Multimedia provides us with an interactive way to display information to a user. Hence, managing and storing these different kinds of multimedia data is essential. This is done using a database known as a multimedia database.

Multimedia database is a collection of multimedia data which includes text, images, graphics (drawings, sketches), animations, audio, video, among others. These databases have extensive amounts of data which can be multimedia and multisource.

The multimedia database can be classified into three types. These types are:

1. Static media
2. Dynamic media
3. Dimensional media

This still leads to a number of challenges to multimedia databases. These are:

1. **Modeling:** Work in this area can improve the database versus information retrieval techniques.
2. Design: The physical, conceptual, and logical design of multimedia databases is not addressed entirely leading to performance and tuning issues.
3. **Storage:** The storage of databases on a standard disc can lead to problems like representation, mapping to disc hierarchies, compression, etc.
4. **Performance:** Audio-video synchronization and audio playback applications are where physical limitations dominate. Parallel processing can reduce these problems, but these techniques have not been completely developed yet. Multimedia databases also consume a lot of processing power and bandwidth.
5. **Queries and Retrieval:** Multimedia such as images, audio, video lead to retrieval and queries issues such as efficient query formation, query execution, etc.

Multimedia Database Applications:

1. **Documents and record management:** Industries which keep a lot of documentation and records. Ex: Insurance claim industry.
2. **Knowledge dissemination:** Multimedia database is an extremely efficient tool for knowledge dissemination and providing several resources. Ex: electronic books
3. **Education and training:** Multimedia sources can be used to create resources useful in education and training. These are popular sources of learning in recent days. Ex: Digital libraries.
4. **Real-time monitoring and control:** Multimedia presentation when coupled with active database technology can be an effective means for controlling and monitoring complex tasks. Ex: Manufacture control
5. Marketing
6. Advertisement
7. Retailing
8. Entertainment
9. Travel

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 its 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 have found new application in data integration, information extraction, networking, program analysis, security, and cloud computing.

Deductive databases reuse many concepts from logic programming; rules and facts specified in Datalog look very similar to those written in Prolog, but there are some important differences:

- Order sensitivity and procedurality: In Prolog, program execution depends on the order of rules in the program and on the order of parts of rules; these properties are used by programmers to build efficient programs. In database languages (like SQL or Datalog), however, program execution is independent of the order of rules and facts.
- Special predicates: In Prolog, programmers can directly influence the procedural evaluation of the program with special predicates such as the cut. This has no correspondence in deductive databases.
- Function symbols: Logic Programming languages allow function symbols to build up complex symbols. This is not allowed in deductive databases.
- Tuple-oriented processing: Deductive databases use set-oriented processing while logic programming languages concentrate on one tuple at a time.

XML and Internet Databases

An **XML database** is a data persistence software system that allows data to be specified, and sometimes stored, in XML format. This data can be queried, transformed, exported and returned to a calling system. XML databases are a flavor of document-oriented databases which are in turn a category of NoSQL database.

A XML database is a database that can be used to store large amounts of data or information in the XML format. They can handle data which is of any size or format.

XML is a markup language that uses “tags” or specially formatted text labels, to identify the function of varied data elements within a document.

There are two major types of XML databases:

- **XML- enabled** – A XML enabled databases is the extension provided for the conversion of an XML document. This database is a relational database, in which data is stored in tables consisting of rows and columns.
- **Native XML (NXD)** – A Native XML database is based on the container rather than table format. This type of database can store large amounts of XML documents and data. Native XML databases have an advantage over the XML-enabled database, as it is easier to store, query and maintain the XML document in a native database than in a XML-enabled database.

The most active XML database available today include:

- MarkLogic
- Oracle Berkeley DB
- Virtuoso

Example of XML Type Query in IBM DB2 SQL[edit]

```
select
  id, vol, xmlquery('$j/name', passing journal as "j") as name
from
  journals
where
  xmlexists('$j[licence="CreativeCommons"]', passing journal as "j")
```

Internet Databases/ online database

An **online database** is a database accessible from a local network or the Internet, as opposed to one that is stored locally on an individual computer or its attached storage (such as a CD). Online databases are hosted on websites, made available as software as a service products accessible via a web browser. They may be free or require payment, such as by a monthly subscription. Some have enhanced features such as collaborative editing and email notification.

Cloud database

A cloud database is a database that is run on and accessed via the Internet, rather than locally. So, rather than keep a customer information database at one location, a business may choose to have it hosted on the Internet so that all its departments or divisions can access and update it. Most database services offer web-based consoles, which the end user can use to provision and configure database instances.

Mobile Database

A mobile database is a database that resides on a mobile device such as a PDA, a smart phone, or a laptop. Such devices are often limited in resources such as memory, computing power, and battery power.

A Mobile Database is a type of database that can be accessed by a mobile network and connected to a mobile computing device (or wireless network). Here, there is a wireless connection between the client and the server. In the modern world, **Mobile Cloud Computing** is expanding quickly and has enormous potential for the database industry. It will work with a variety of various devices, including Mobile Databases powered by iOS and Android, among others. Couchbase Lite, Object Box, and other popular databases are examples of databases.

Mobile Database Consists of Three Parties Which are Described Below:

- **Fixed Hosts:**
  With the aid of database servers, it handles transactions and manages data.
- **Mobile Units:**
  These are mobile, transportable computers, and the cell tower they utilize to connect to base stations is a part of that geographical area.
- **Base Stations:**
  These two-way radios, which are installed in fixed places, allow communication between the stationary hosts and the mobile units.

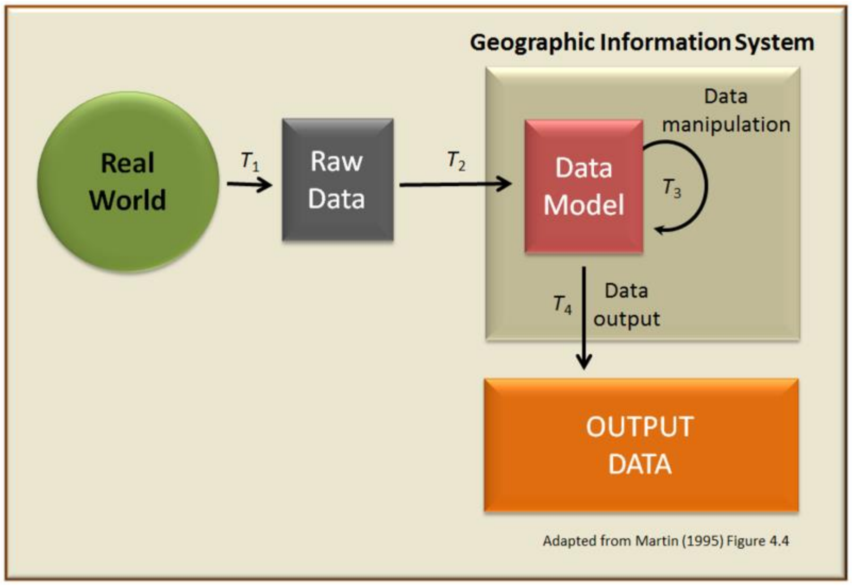
Geographic Information Systems

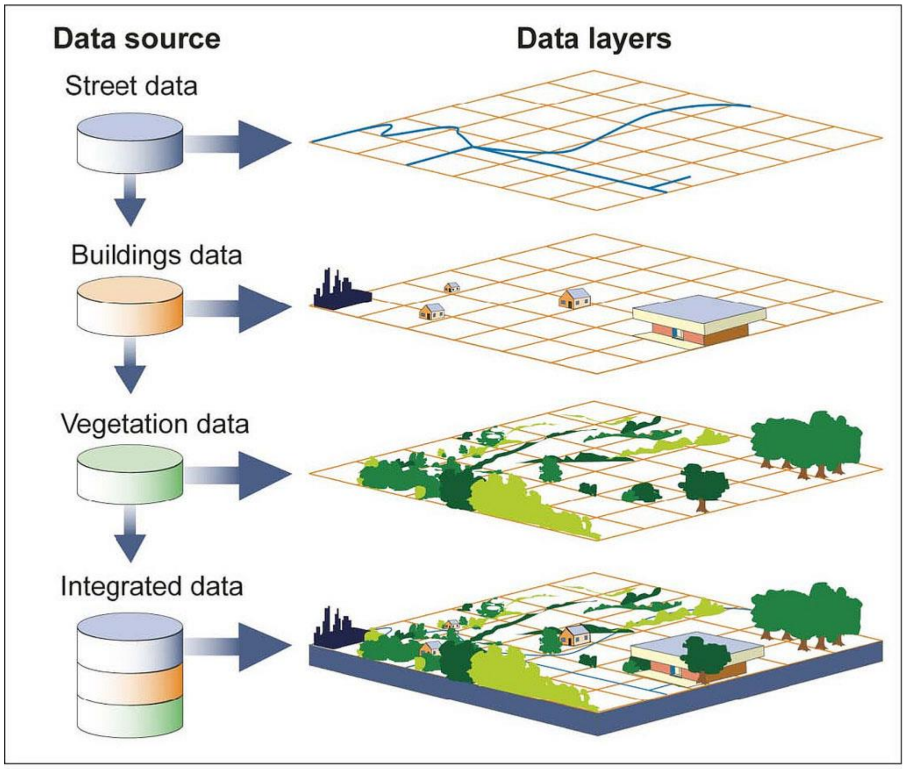
A **geographic information system** (**GIS**) consists of integrated computer hardware and software that store, manage, analyze, edit, output, and visualize geographic data.[1][2] Much of this often happens within a spatial database, however, this is not essential to meet the definition of a GIS. In a broader sense, one may consider such a system also to include human users and support staff, procedures and workflows, the body of knowledge of relevant concepts and methods, and institutional organizations.

The uncounted plural, *geographic information systems*, also abbreviated GIS, is the most common term for the industry and profession concerned with these systems. It is roughly synonymous with geoinformatics. The academic discipline that studies these systems and their underlying geographic principles, may also be abbreviated as GIS, but the unambiguous GIScience is more common. GIScience is often considered a subdiscipline of geography within the branch of technical geography.

Geographic information systems are utilized in multiple technologies, processes, techniques and methods. They are attached to various operations and numerous applications, that relate to: engineering, planning, management, transport/logistics, insurance, telecommunications, and business.[4] For this reason, GIS and location intelligence applications are at the foundation of location-enabled services, which rely on geographic analysis and visualization.

GIS provides the capability to relate previously unrelated information, through the use of location as the "key index variable". Locations and extents that are found in the Earth's spacetime are able to be recorded through the date and time of occurrence, along with x, y, and z coordinates; representing, longitude (*x*), latitude (*y*), and elevation (*z*). All Earth-based, spatial–temporal, location and extent references should be relatable to one another, and ultimately, to a "real" physical location or extent. This key characteristic of GIS has begun to open new avenues of scientific inquiry and studies.




Data source
Data layers
Street data
Buildings data
Vegetation data
Integrated data

Source: GAO.

Genome Data Management

Genome data management allows for the identification of genetic mutations and variants associated with various diseases, which can lead to improved diagnosis, treatment, and prevention strategies.

Genome Data Management

Data Storage − Storing large volumes of genome data requires a combination of scalable storage solutions and efficient data compression methods. Popular storage solutions include cloud storage, distributed file systems, and relational databases.

Data Quality Control − Quality control is essential for ensuring the accuracy and reliability of genome data. This includes checking for errors in sequencing, contamination, and data integrity.

Data Analysis − The complexity and diversity of genome data require a wide range of analytical tools and methods. These include alignment tools, variant calling, annotation, functional analysis, and visualization tools.

Data Integration − Integrating data from different sources and in different formats is a major challenge in genome data management. This requires the use of standard data formats, ontologies, and data integration tools.

Data Security − The sensitive nature of genome data requires strict security measures to protect the privacy of research participants and to comply with regulations. This includes data encryption, access controls, and data-sharing policies.

Real-world Examples

The National Center for Biotechnology Information (NCBI) is a well-known repository for a wide variety of biological data, including genome data. It provides a range of tools and resources for data storage, analysis, and visualization.

The European Bioinformatics Institute (EBI) is another major repository for biological data, including genome data. It offers a wide range of data storage, analysis, and visualization tools, as well as access to a large number of public datasets.

The Genomic Data Commons (GDC) is a platform for storing, sharing, and analyzing cancer genomic data. It provides a centralized repository for cancer genomics data, as well as a wide range of analytical tools.

Example

In this example we will use python and the Biopython library to extract information from a GenBank file, which is a common file format for storing genome data.

```
from Bio import SeqIO

#parse the GenBank file
for record in SeqIO.parse("example.gb", "genbank"):
```

```
•
•     #print the record's ID
•     print(record.id)
•
•     #print the record's annotation
•     print(record.annotations)
•
•     #print the record's sequence
•     print(record.seq)
```

In this example, we are using the Bio.SeqIO module from the Biopython library to parse the GenBank file "example.gb". The SeqIO.parse() function returns an iterator that yields SeqRecord objects, which contain the record's ID, annotation, and sequence. We can then access these attributes and print them out. This is just a simple example of how the Biopython library can be used to extract information from genome data files.

Distributed Databases and ClientServer Architectures

Distributed Databases:

A **distributed database** is a database in which data is stored across different physical locations.[1] It may be stored in multiple computers located in the same physical location (e.g. a data centre); or maybe 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 (e.g. in a database) across multiple physical locations. A distributed database can reside on organised network servers or decentralised independent computers on the Internet, on corporate intranets or extranets, or on other organisation networks. Because distributed databases store data across multiple computers, distributed databases may improve performance at end-user worksites by allowing transactions to be processed on many machines, instead of being limited to one.[2]

A distributed database is basically a database that is not limited to one system, it is spread over different sites, i.e, on multiple computers or over a network of computers. A distributed database system is located on various sites that don't share physical components. This may be required when a particular database needs to be accessed by various users globally. It needs to be managed such that for the users it looks like one single database.

Types:
1. Homogeneous Database:
In a homogeneous database, all different sites store database identically. The operating system, database management system, and the data structures used – all are the same at all sites. Hence, they're easy to manage.
2. Heterogeneous Database:
In a heterogeneous distributed database, different sites can use different schema and software that can lead to problems in query processing and transactions. Also, a particular site might be completely unaware of the other sites. Different computers may use a different operating system, different database application. They may even use different data models for the database. Hence, translations are required for different sites to communicate.

Two processes ensure that the distributed databases remain up-to-date and current: replication[3] and duplication.

1. 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 much time and computer resources.
2. Duplication, on the other hand, 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.

Distributed Data Storage :
There are 2 ways in which data can be stored on different sites. These are:
1. Replication –
In this approach, the entire relationship is stored redundantly at 2 or more sites. If the entire database is available at all sites, it is a fully redundant database. Hence, in replication, systems maintain copies of data.
This is advantageous as it increases the availability of data at different sites. Also, now query requests can be processed in parallel.
However, it has certain disadvantages as well. Data needs to be constantly updated. Any change made at one site needs to be recorded at every site that relation is stored or else it may lead to inconsistency. This is a lot of overhead. Also, concurrency control becomes way more complex as concurrent access now needs to be checked over a number of sites.

2. Fragmentation –
In this approach, the relations are fragmented (i.e., they're divided into smaller parts) and each of the fragments is stored in different sites where they're required. It must be made sure that the fragments are such that they can be used to reconstruct the original relation (i.e, there isn't any loss of data).
Fragmentation is advantageous as it doesn't create copies of data, consistency is not a problem.

Fragmentation of relations can be done in two ways:

- **Horizontal fragmentation – Splitting by rows –**
  The relation is fragmented into groups of tuples so that each tuple is assigned to at least one fragment.
- **Vertical fragmentation – Splitting by columns –**
  The schema of the relation is divided into smaller schemas. Each fragment must contain a common candidate key so as to ensure a lossless join.

In certain cases, an approach that is hybrid of fragmentation and replication is used.

Applications of Distributed Database:
- It is used in Corporate Management Information System.
- It is used in multimedia applications.

- Used in Military's control system, Hotel chains etc

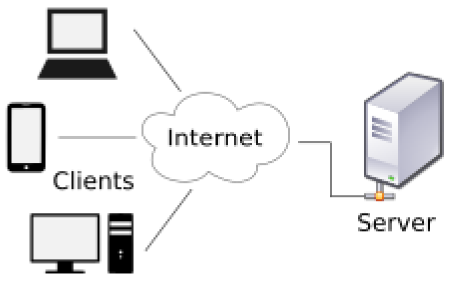
client-server architecture:

The **client–server model** is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.[1] Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs, which share their resources with clients. A client usually does not share any of its resources, but it requests content or service from a server. Clients, therefore, initiate communication sessions with servers, which await incoming requests. Examples of computer applications that use the client–server model are email, network printing, and the World Wide Web.

In client-server architecture many clients connected with one server. The server is centerlines.it provides services to all clients. All clients request to the server for different Service. The server displays the results according to the client's request.

Client/server architecture is a computing model in which the server hosts (computer), send and manages most of the resources and works to be required by the client. In this type of architecture has one or more client computers attached to a central server over a network. This system shares different resources.

Client/server architecture is also called as a networking computing model and client-server network because all the requests and demands are sent over a network.



A computer network diagram of clients communicating with a server via the Internet