



BHARATHIDASAN UNIVERSITY

Tiruchirappalli- 620024

Tamil Nadu, India.

Programme: M.Sc. Statistics

Course Title: Statistical Methods for Bioinformatics

Course Code: 23ST08DEC

Unit-V

Databases

Dr. T. Jai Sankar
Associate Professor and Head
Department of Statistics

Ms. I. Angel Agnes Mary
Guest Faculty
Department of Statistics

UNIT – V

DATABASES

Data life cycle acquisition

The data life cycle is the sequence of stages that a particular unit of data goes through from its initial generation or capture to its eventual archival and/or deletion at the end of its useful life.

- Generation
- Collection
- Processing
- Storage
- Management
- Analysis
- Visualization
- Interpretation

1. Generation

For the data life cycle to begin, data must first be generated. Otherwise, the following steps can't be initiated.

Data generation occurs regardless of whether you're aware of it, especially in our increasingly online world. Some of this data is generated by your organization, some by your customers, and some by third parties you may or may not be aware of. Every sale, purchase, hires, communication, interaction—everything generates data. Given the proper attention, this data can often lead to powerful insights that allow you to better serve your customers and become more effective in your role.

2. Collection

Not all of the data that's generated every day is collected or used. It's up to your data team to identify what information should be captured and the best means for doing so, and what data is unnecessary or irrelevant to the project at hand. To collect data in a variety of ways, including:

- Forms: Web forms, client or customer intake forms, vendor forms, and human resources applications are some of the most common ways businesses generate data.
- Surveys: Surveys can be an effective way to gather vast amounts of information from a large number of respondents.
- Interviews: Interviews and focus groups conducted with customers, users, or job applicants offer opportunities to gather qualitative and subjective data that may be difficult to capture through other means.

- **Direct Observation:** Observing how a customer interacts with your website, application, or product can be an effective way to gather data that may not be offered through the methods above.

It's important to note that many organizations take a broad approach to data collection, capturing as much data as possible from each interaction and storing it for potential use. While drawing from this supply is certainly an option, it's always important to start by creating a plan to capture the data you know is critical to your project.

3. Processing

Once data has been collected, it must be processed. Data processing can refer to various activities, including:

- **Data wrangling**, in which a data set is cleaned and transformed from its raw form into something more accessible and usable. This is also known as data cleaning, data munging, or data remediation.
- **Data compression**, in which data is transformed into a format that can be more efficiently stored.
- **Data encryption**, in which data is translated into another form of code to protect it from privacy concerns.

Even the simple act of taking a printed form and digitizing it can be considered a form of data processing.

4. Storage

After data has been collected and processed, it must be stored for future use. This is most commonly achieved through the creation of databases or datasets. These datasets may then be stored in the cloud, on servers, or using another form of physical storage like a hard drive, CD, cassette, or floppy disk.

When determining how to best store data for your organization, it's important to build in a certain level of redundancy to ensure that a copy of your data will be protected and accessible, even if the original source becomes corrupted or compromised.

5. Management

Data management, also called database management, involves organizing, storing, and retrieving data as necessary over the life of a data project. While referred to here as a "step," it's an ongoing process that takes place from the beginning through the end of a project. Data management includes everything from storage and encryption to implementing access logs and change logs that track that has accessed data and what changes they may have made.

6. Analysis

Data analysis refers to processes that attempt to glean meaningful insights from raw data. Analysts and data scientists use different tools and strategies to conduct these analyses. Some of the more commonly used methods include statistical modeling, algorithms, artificial intelligence, data mining, and machine learning.

Exactly who performs an analysis depends on the specific challenge being addressed, as well as the size of your organization's data team. Business analysts, data analysts, and data scientists can all play a role.

7. Visualization

Data visualization refers to the process of creating graphical representations of your information, typically through the use of one or more visualization tools. Visualizing data makes it easier to quickly communicate your analysis to a wider audience both inside and outside your organization. The form your visualization takes depends on the data you're working with, as well as the story you want to communicate.

While technically not a required step for all data projects, data visualization has become an increasingly important part of the data life cycle.

8. Interpretation

Finally, the interpretation phase of the data life cycle provides the opportunity to make sense of your analysis and visualization. Beyond simply presenting the data, this is when you investigate it through the lens of your expertise and understanding. Your interpretation may not only include a description or explanation of what the data shows but, more importantly, what the implications may be.

Database Architecture

A **Database Architecture** is a representation of DBMS design. It helps to design, develop, implement, and maintain the database management system. DBMS architecture allows dividing the database system into individual components that can be independently modified, changed, replaced, and altered. It also helps to understand the components of a database.

A Database stores critical information and helps access data quickly and securely. Therefore, selecting the correct Architecture of DBMS helps in easy and efficient data management.

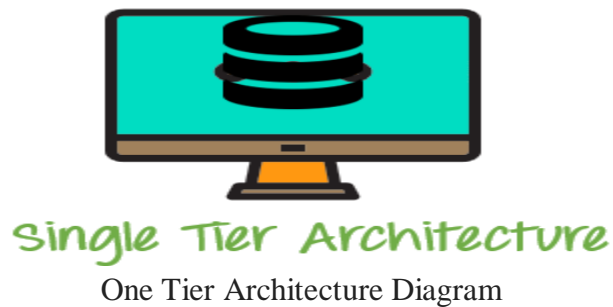
Types of DBMS Architecture

There are mainly three types of DBMS architecture:

- One Tier Architecture (Single Tier Architecture)
- Two Tier Architecture
- Three Tier Architecture

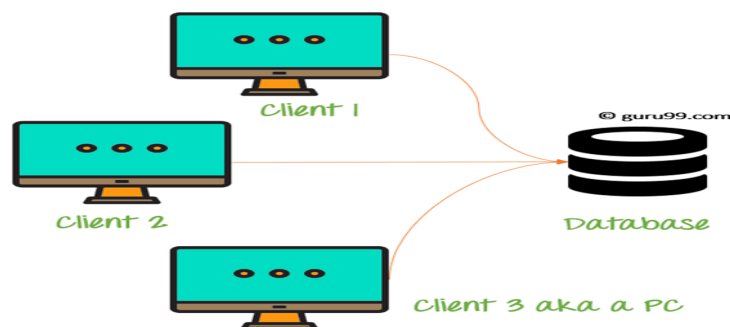
One Tier Architecture

One Tier Architecture in DBMS is the simplest architecture of Database in which the client, server, and Database all reside on the same machine. A simple one tier architecture example would be anytime you install a Database in your system and access it to practice SQL queries. But such architecture is rarely used in production.



Two Tier Architecture

A **Two Tier Architecture** in DBMS is a Database architecture where the presentation layer runs on a client (PC, Mobile, Tablet, etc.), and data is stored on a server called the second tier. Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.



Two Tier Architecture Diagram

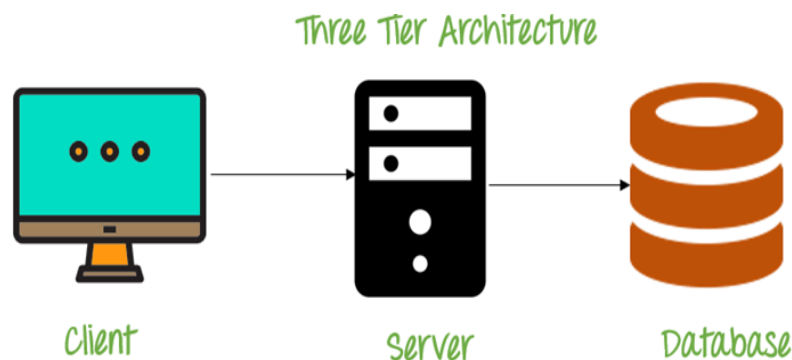
In the above Two Tier client-server architecture of database management system, we can see that one server is connected with clients 1, 2, and 3.

Three Tier Architecture

A **Three Tier Architecture** in DBMS is the most popular client server architecture in DBMS in which the development and maintenance of functional processes, logic and data access, data storage and user interface is done independently as separate modules. Three Tier architecture contains a presentation layer, an application layer, and a database server.

Three Tier database Architecture design is an extension of the 2-tier client-server architecture. Three tier architecture has the following layers:

1. Presentation layer (your PC, Tablet, Mobile, etc.)
2. Application layer (server)
3. Database Server



Three Tier Architecture Diagram

The Application layer resides between the user and the DBMS, which is responsible for communicating the user's request to the DBMS system and send the response from the DBMS to the user. The application layer(business logic layer) also processes functional logic, constraint, and rules before passing data to the user or down to the DBMS.

The goal of Three Tier client-server architecture is:

- To separate the user applications and physical database
- To support DBMS characteristics
- Program-data independence
- Supporting multiple views of the data

Interfaces in DBMS

A database management system (DBMS) interface is a user interface that allows for the ability to input queries to a database without using the query language itself.

User-friendly interfaces provided by DBMS may include the following:

1. Menu-Based Interfaces for Web Clients or Browsing

These interfaces present the user with lists of options (called menus) that lead the user through the formation of a request. Basic advantage of using menus is that they removes the tension of remembering specific commands and syntax of any query language. The query is basically composed step by step by collecting or picking options from a menu that is shown by the system. Pull-down menus are a very popular technique in *Web based interfaces*. They are also often used in *browsing interface* which allow a user to look through the contents of a database in an exploratory and unstructured manner.

Forms-Based Interfaces

A forms-based interface displays a form to each user. Users can fill out all of the form entries to insert new data, or they can fill out only certain entries, in which case the DBMS will redeem same type of data for other remaining entries. These types of forms are usually designed or created and programmed for the users that have no expertise in operating system. Many DBMSs have *forms specification languages* which are special languages that help specify such forms. Example: SQL* Forms is a form-based language that specifies queries using a form designed in conjunction with the relational database schema.

2. Graphical User Interface –

A GUI typically displays a schema to the user in diagrammatic form. The user then can specify a query by manipulating the diagram. In many cases, GUIs utilize both menus and forms. Most GUIs use a pointing device such as mouse, to pick a certain part of the displayed schema diagram.

3. Natural language Interfaces

These interfaces accept request written in English or some other language and attempt to understand them. A Natural language interface has its own schema, which is similar to the database conceptual schema as well as a dictionary of important words.

The natural language interface refers to the words in its schema as well as to the set of standard words in a dictionary to interpret the request. If the interpretation is successful, the interface generates a high-level query corresponding to the natural language and submits it to the DBMS for processing; otherwise a dialogue is started with the user to clarify any provided condition or request. The main disadvantage with this is that the capabilities of this type of interfaces are not that much advance.

4. Speech Input and Output

There is limited use of speech be it for a query or an answer to a question or being a result of a request it is becoming commonplace. Applications with limited vocabularies such as inquiries for telephone directory, flight arrival/departure, and bank account information are allowed speech for input and output to enable ordinary folks to access this information.

The Speech input is detected using predefined words and used to set up the parameters that are supplied to the queries. For output, a similar conversion from text or numbers into speech takes place.

5. Interfaces for DBA

Most database system contains privileged commands that can be used only by the DBA's staff. These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, reorganizing the storage structures of databases.

Use of DBMS in System Software and programming languages

Database Management Systems (DBMS) are software systems used to store, retrieve, and run queries on data. A DBMS serves as an interface between an end-user and a database, allowing users to create, read, update, and delete data in the database. Using a general-purpose programming language, user can write a source program in the normal way. However, instead of writing I/O statements of the form provided by the programming language, the programmer writes commands in a data manipulation language (DML) defined for use with the DBMS. Processor may be used to convert the DML commands into programming language statements that call DBMS routines.

Using the programming language itself some DMLs are defined as a set of CALL statements. Here given are the two principal methods for user interaction with a DBMS.

Interaction with a DBMS using a query language is another approach to DBMS. There is no need for the user to write the programs for accessing a database rather user only needs to enter the commands in a special query language defined by DBMS. These commands are processed by a query-language interpreter, which calls DBMS routines to perform the requested operations.

Each and every approach that leads to user interactivity with a DBMS has its own advantages. Results can be obtain much faster with a help of query language, because there is no need to write and debug programs, which becomes very beneficial for the non-programmers to used it efficiently. Allowing the programmer to use all the flexibility and power of a general-purpose programming language is the big advantage of DML however much effort from the user is required by this approach. Most modern database management systems provide both a query language and a DML so that a user can choose the form of interaction that best meets his or her needs.

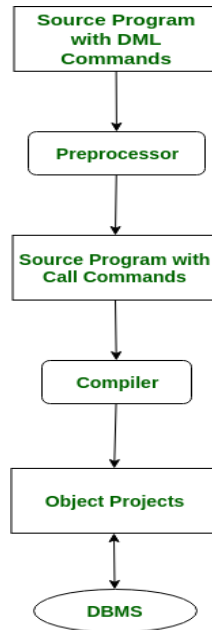


Figure 1 (a): Interaction with a DBMS using a data manipulation language

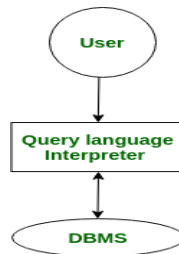


Figure 1 (b): Interaction with a DBMS using a query language

Here are some steps to show how a typical sequence of actions is being performed by a DBMS:

- **Step-1:** The sequence of events begins when the DBMS is entered with the help of a call from application program A. We assume this call is a request to read data from the database. There are similar sequences of events for other types of database operations.
- **Step-2:** The request from program A is stated in terms of the subschema being used by A. To process a request which is being requested from program A and is stated in terms of the subschema which is being used by A, the DBMS must first examine the subschema definition.
- **Step-3:** Relationship between the subschema and the schema must be considered by the DBMS to interpret the request in terms of the overall logical database structure.
- **Step-4:** The DBMS examines the data mapping description, after determining the logical database records that must be read in terms of schema. The information regarding the need of locating the required records in the files of the database is given by this operation.

- **Step-5:** At this point, a logical request for a subschema record has been converted into physical requests by DBMS to read data from one or more files. These requests for file I/O are passed to the operating system using the types of service calls.
- **Step-6:** The operating system then issues channel and device commands to perform the necessary physical I/O operations. These I/O operations read the required records from the database into a DBMS buffer area.
- **Step-7:** All the data requested by the application program is present in central memory after the physical I/O operations have been completed. The DBMS accomplishes this conversion by again comparing the schema and the subschema.
- Finally, the DBMS returns control to the application program and makes available to the program a variety of status information, including any possible error indications.

For clear understanding, here is the diagram given:

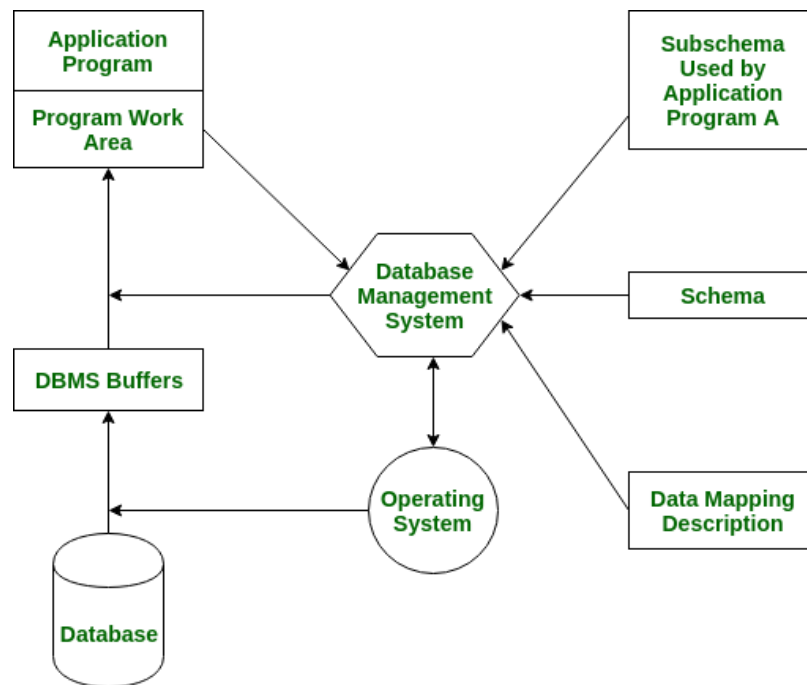


Figure 2: Typical sequence of actions performed by a DBMS

Examples of some bioinformatics database

The four examples of biological databases are:

1. Nucleotide Sequence Databases
2. Protein Sequence Databases
3. Macromolecular Databases and
4. Other Databases

Biological Databases:

The collection of the biological data on a computer which can be manipulated to appear in varying arrangements and subsets is regarded as a database. The biological information can be stored in different databases. Each database has its own website with unique navigation tools. The biological databases are, in general, publicly accessible.

Selected examples of biological databases are briefly described (Table)

<i>Database</i>	<i>Salient features</i>
Primary nucleotide sequence databases	
GenBank (www.ncbi.nih.gov/GeneBank/)	Provides nucleotide sequence databases maintained by the National Center for Biotechnology Information (NCBI), USA.
EMBL (www.ebi.ac.uk/embl/)	European Molecular Biology Laboratory (EMBL) maintains nucleotide sequence databases under the aegis of European Bioinformatics Institute (EBI), UK.
Other nucleotide sequence databases	
UniGene (www.ncbi.nih.gov/UniGene/)	The nucleotide sequences of GenBank in the form of clusters, representing genes are available.
Genome Biology (www.ncbi.nlm.nih.gov/Genomes/)	The information about the completed genomes is available.
EBI Genomes (www.ebi.uk/genomes/)	Provides data and statistics for the completed genomes, besides the information on the ongoing projects.
Protein sequence database	
SWISS-PROT (www.expasy.ch/sport)	Provides the description of the structure of a protein, its domains structure, post-translational modifications, variants etc. It has high level of integration with other databases and minimal level of redundancy.
PIR (pir.georgetown.edu)	Protein information resource (PIR) is a database provided by the National Biomedical Research Foundation (NBRF) in USA.
Protein sequence motif databases	
PROSITE (www.expasy.ch/prosite/)	Provides information on protein families and domains. It also has patterns and profiles for sequences and biological functions.
Pfam (www.cgr.ki.se/Pfam/)	A database of protein families defined in the form of domains. It has multiple alignment of a set of sequences.
Macromolecular databases	
PDB (www.rcsb.org/pdb)	This is the primary database for 3-dimensional (3-D) structures of biological macromolecules (determined by X-ray and NMR studies).
SCOP (www.mrc.lmb.cam.ac.uk/scop/)	Provides information on the structural classification of proteins (SCOP). The proteins are classified on the basis of 3-D structures.
Other databases	
KEGG (www.genome.ad.jp/kegg/)	The Kyoto Encyclopedia of Genes and Genomes (KEGG) is a database with latest computerised information on biomolecules and cell biology. KEGG provides details on information pathways, interacting molecules and the connecting links with genes.

1. Nucleotide Sequence Databases:

The nucleotide sequence data submitted by the scientists and genome sequencing groups is at the databases namely Gen Bank, EMBL (European Molecular Biology Laboratory) and DDBJ (DNA Data Bank of Japan). There is a good coordination between these three databases as they are synchronized on daily basis. Besides the primary nucleotide databases (referred above), there are some other databases also to provide information on genes, genomes and ongoing research projects.

2. Protein Sequence Databases:

Protein sequence databases are usually prepared from the existing literature and/or in consultation with the experts. In fact, these databases represent the translated DNA databases.

3. Macromolecular Databases:

The three dimensional (3-D) structures of macromolecules are determined by X-ray crystallography and nuclear magnetic resonance (NMR). PDB and SCOP are the primary databases of 3-D structures of biological molecules.

4. Other Databases:

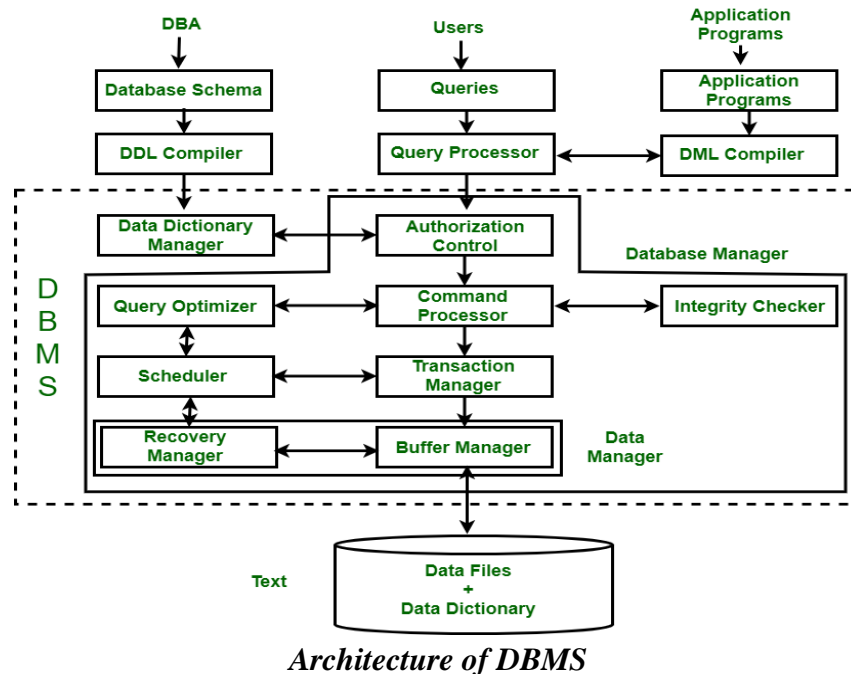
KEGG database is an important one that provides information on the current knowledge of molecular biology and cell biology with special reference to information on metabolic pathways, interacting molecules and genes.

Structure of Database Management System

Database Management System (DBMS) is a software that allows access to data stored in a database and provides an easy and effective method of

- Defining the information.
- Storing the information.
- Manipulating the information.
- Protecting the information from system crashes or data theft.
- Differentiating access permissions for different users.

The database system is divided into three components: Query Processor, Storage Manager, and Disk Storage. These are explained as following below.



1. Query Processor:

It interprets the requests (queries) received from end user via an application program into instructions. It also executes the user request which is received from the DML compiler. Query Processor contains the following components –

- DML Compiler – It processes the DML statements into low level instruction (machine language), so that they can be executed.
- DDL Interpreter – It processes the DDL statements into a set of table containing meta data (data about data).
- Embedded DML Pre-compiler – It processes DML statements embedded in an application program into procedural calls.
- Query Optimizer – It executes the instruction generated by DML Compiler.

2. Storage Manager:

Storage Manager is a program that provides an interface between the data stored in the database and the queries received. It is also known as Database Control System. It maintains the consistency and integrity of the database by applying the constraints and executes the [DCL](#) statements. It is responsible for updating, storing, deleting, and retrieving data in the database. It contains the following components –

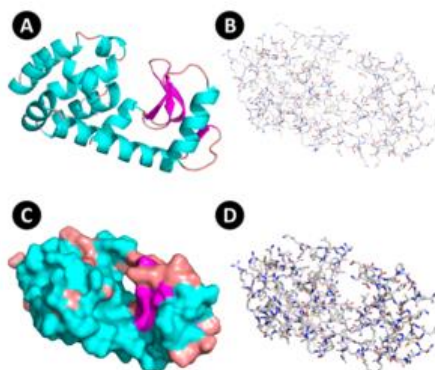
- Authorization Manager – It ensures role-based access control, i.e., checks whether the particular person is privileged to perform the requested operation or not.
- Integrity Manager – It checks the integrity constraints when the database is modified.

- Transaction Manager – It controls concurrent access by performing the operations in a scheduled way that it receives the transaction. Thus, it ensures that the database remains in the consistent state before and after the execution of a transaction.
- File Manager – It manages the file space and the data structure used to represent information in the database.
- Buffer Manager – It is responsible for cache memory and the transfer of data between the secondary storage and main memory.

3. Disk Storage: It contains the following components –

- Data Files – It stores the data.
- Data Dictionary – It contains the information about the structure of any database object. It is the repository of information that governs the metadata.
- Indices – It provides faster retrieval of data item.

Structure visualization



Structural visualization of BACTERIOPHAGE T4 LYSOZYME (PDB ID: 2LZM).

(A) Cartoon; (B) Lines; (C) Surface; (D) Sticks.

Protein structure visualization is an important issue for structural bioinformatics. It allows users to observe static or dynamic representations of the molecules, also allowing the detection of interactions that may be used to make inferences about molecular mechanisms. The most common types of visualization are:

- **Cartoon:** this type of protein visualization highlights the secondary structure differences. In general, α -helix is represented as a type of screw, β -strands as arrows, and loops as lines.
- **Lines:** each amino acid residue is represented by thin lines, which allows a low cost for graphic rendering.
- **Surface:** in this visualization, the external shape of the molecule is shown.
- **Sticks:** each covalent bond between amino acid atoms is represented as a stick. This type of visualization is most used to visualize interactions between amino acids.

Pattern matching

Pattern matching is the process of checking whether a specific sequence of characters/tokens/data exists among the given data.

The Pattern Searching algorithms are sometimes also referred to as String Searching Algorithms and are considered as a part of the String algorithms. These algorithms are useful in the case of searching a string within another string.

Text : A A B A A C A A D A A B A A B A

Pattern : A A B A

```








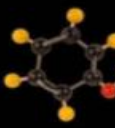
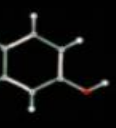
A A B A           A A B A
A A B A A C A A D A A B A A B A
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
                A A B A
```

Pattern Found at 0, 9 and 12

Molecular modelling

Molecular modelling is a collective term that refers to theoretical methods and computational techniques to model or mimic the behaviour of molecules. The techniques are used in the fields of computational chemistry, computational biology and materials science for studying molecular systems ranging from small chemical systems to large biological molecules and material assemblies.

Types of Molecular Models

MOLECULE	MODEL TYPE		
	Space-filling	ball and spoke	skeletal
water			
ethanol (alcohol)			
phenol			

- Ball and spoke models
- Space-filling models
- Crystal lattice models

Ball and spoke models

Ball and spoke models are a common way of representing molecular structures. Each atom is represented by a colored ball that is joined to other atoms using spokes to represent the bonds between them. This type of model emphasises the bonding between atoms.

Space-filling models

Space-filling models give a representation of the size and shape of the whole molecule, showing (relatively) how much space each atom occupies. Space-filling models were first designed by H. A. Stuart in 1934.

Crystal lattice models

Molecular model kits are designed to be re-used; models can be built and then taken apart again, but chemists often make permanent models of molecular structures for demonstrations or teaching. This crystallographic model of the metal beta-manganese was made by Mr C. E. Chapman, Chief technician of the Crystallography Department, part of the Cavendish Laboratory at the University of Cambridge, in about 1952. It shows how the atoms of manganese are arranged at high temperatures.

Mapping databases

- ✓ “Map of Maps”.
- ✓ The different types of markers and methods used for genomic mapping.
- ✓ The inherent complexities in the construction and utilization of genome maps.
- ✓ Several large community databases and method-specific mapping projects.
- ✓ Practical examples of how these tools and resources can be used to aid in specific types of mapping studies.

Genomic Mapping

1. Genetic Mapping

- Crossbreeding and pedigree
- Calculation of recombination frequency by linkage analysis

2. Cytogenetic Mapping

- FISH(Fluorescent In Situ Hybridization)

3. Physical Mapping

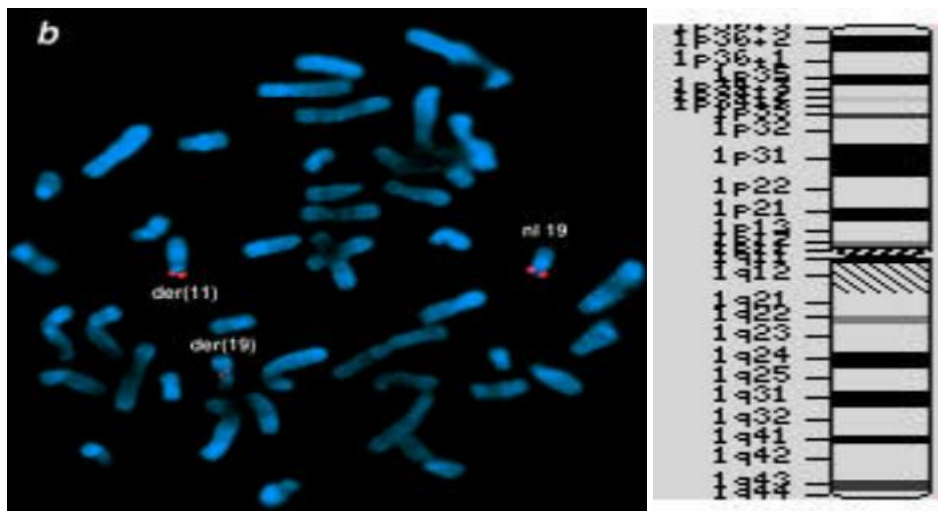
- Molecular biology technique (hybridization, PCR)
- Restriction Mapping
- STS(Sequence Tagged Site) Mapping
- Radiation-hybrid method, Clone library based method.

Type of Maps

- Cytogenetic maps
- Genetic Linkage maps
- Radiation hybrid maps
- Transcript maps
- Physical maps
- Comparative maps
- Integrated maps

Cytogenetic Maps

- ❖ Chromosome staining by Giemsa: G-bands
- ❖ Using FISH method
- ❖ Limited ordering range (<1-2mb)
- ❖ Not well-suited for high-throuput mapping.



Genetic Linkage Maps

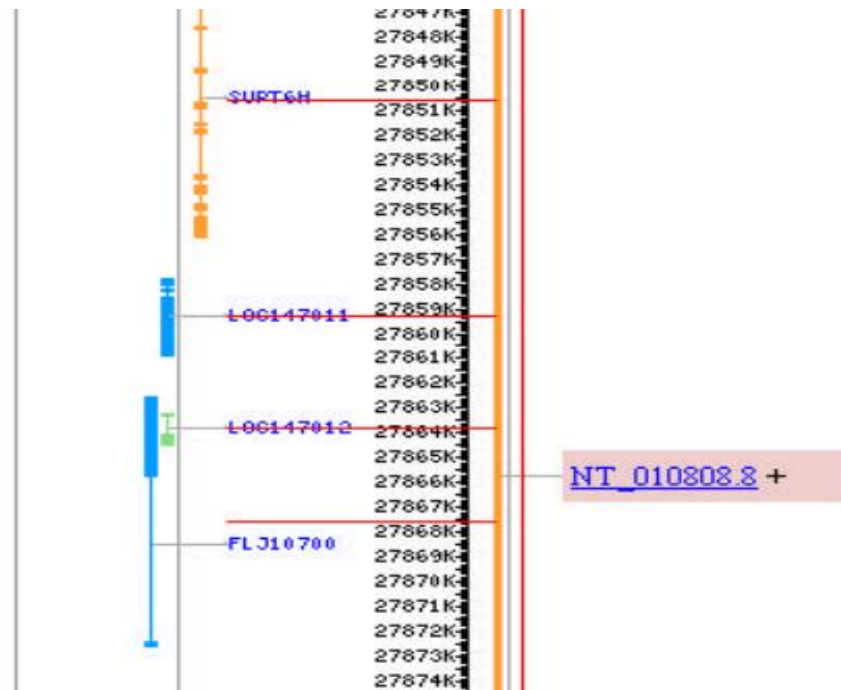
- ❖ 1centiMorgans (cM) represents 1% probability of recombination.
- ❖ Not directly proportional to physical distance.
- ❖ MAP-O-MAT (map distance, statical support for order)
 - <http://compgen.rutgers.edu/mapomat>
- ❖ CEPH(Centre d'Etude du Polymorphism Humain)
 - Human DNAs from a set of reference pedigrees
 - highly polymorphic STR markers , SNP
 - <http://www.cephb.fr/cephdb/>

Radiation Hybrid Maps

- ❖ 1 Centirays (cR) representing a 1% probability of chromosome breakage.
- ❖ Radiation Hybrid Database (RHdb)
- ❖ <http://www.ebi.ac.uk/RHdb/>

Transcript Maps

- ❖ Maps of transcribed sequences.
- ❖ Using Expressed Sequence Tag (EST) and known gene.



Physical Maps

- ❖ STS content mapping (>1Mb)
 - using PCR-based positional markers.
 - Ordering of marker in clone library
 - distance is measured by restriction mapping
- ❖ CEPHYAC map - used a combination of fingerprinting.
- ❖ Optical Mapping.

Comparative Maps

- ❖ Process of identifying conserved chromosome segments across different species.
- ❖ Orthologous genes sharing an identical linear order within a chromosome region.
- ❖ Identify conserved segments and ancient chromosome breakpoints.

Phylogenetic analysis

In Phylogenetic analysis, branching diagrams are made to represent the evolutionary history or relationship between different species, organisms, or characteristics of an organism (genes, proteins, organs, etc.) that are developed from a common ancestor.

The diagram is known as a Phylogenetic tree. Phylogenetic analysis is important for gathering information on biological diversity, genetic classifications, as well as learning developmental events that occur during evolution.

With advancements in genetic sequencing techniques, Phylogenetic analysis now involves the sequence of a gene to understand the evolutionary relationships among species. DNA being the hereditary material can now be sequenced easily, rapidly, and cost-effectively, and the data obtained from genetic sequencing is very informative and specific.

1. Purpose of Phylogenetic:

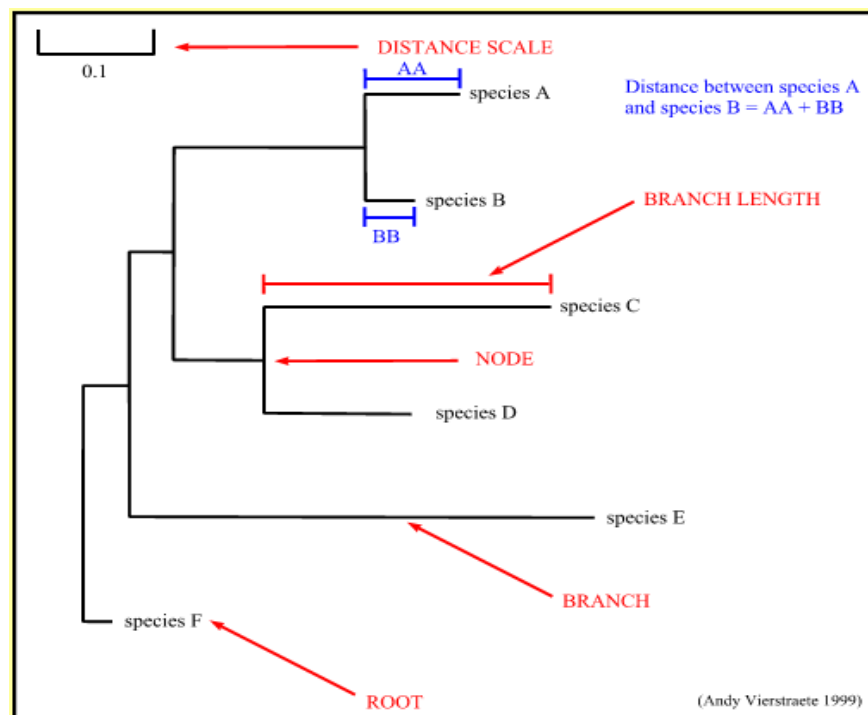
- With the aid of sequences, it should be possible to find the genealogical ties between organisms. Experience learns that closely related organisms have similar sequences; more distantly related organisms have more dissimilar sequences. One objective is to reconstruct the **evolutionary relationship** between species.
- Another objective is to estimate the **time of divergence** between two organisms since they last shared a common ancestor.

2. Disclaimers:

- The theory and practical applications of the different models are not universally accepted.
- With one dataset, different software packages can give different results. Changes in the dataset can also give different results. Therefore it is important to have a good alignment to start with.
- Trees based on an alignment of a gene represent the relationship between genes and this is not necessarily the same relationship as between the whole organisms. If trees are calculated based on different genes from organisms, it is possible that these trees result in different relationships.

3. Terminology:

- **node** : a node represents a taxonomic unit. This can be a taxon (an existing species) or an ancestor (unknown species : represents the ancestor of 2 or more species).
- **branch** : defines the relationship between the taxa in terms of descent and ancestry.
- **topology** : is the branching pattern.
- **branch length** : often represents the number of changes that have occurred in that branch.
- **root** : is the common ancestor of all taxa.
- **distance scale** : scale which represents the number of differences between sequences (e.g. 0.1 means 10 % differences between two sequences)



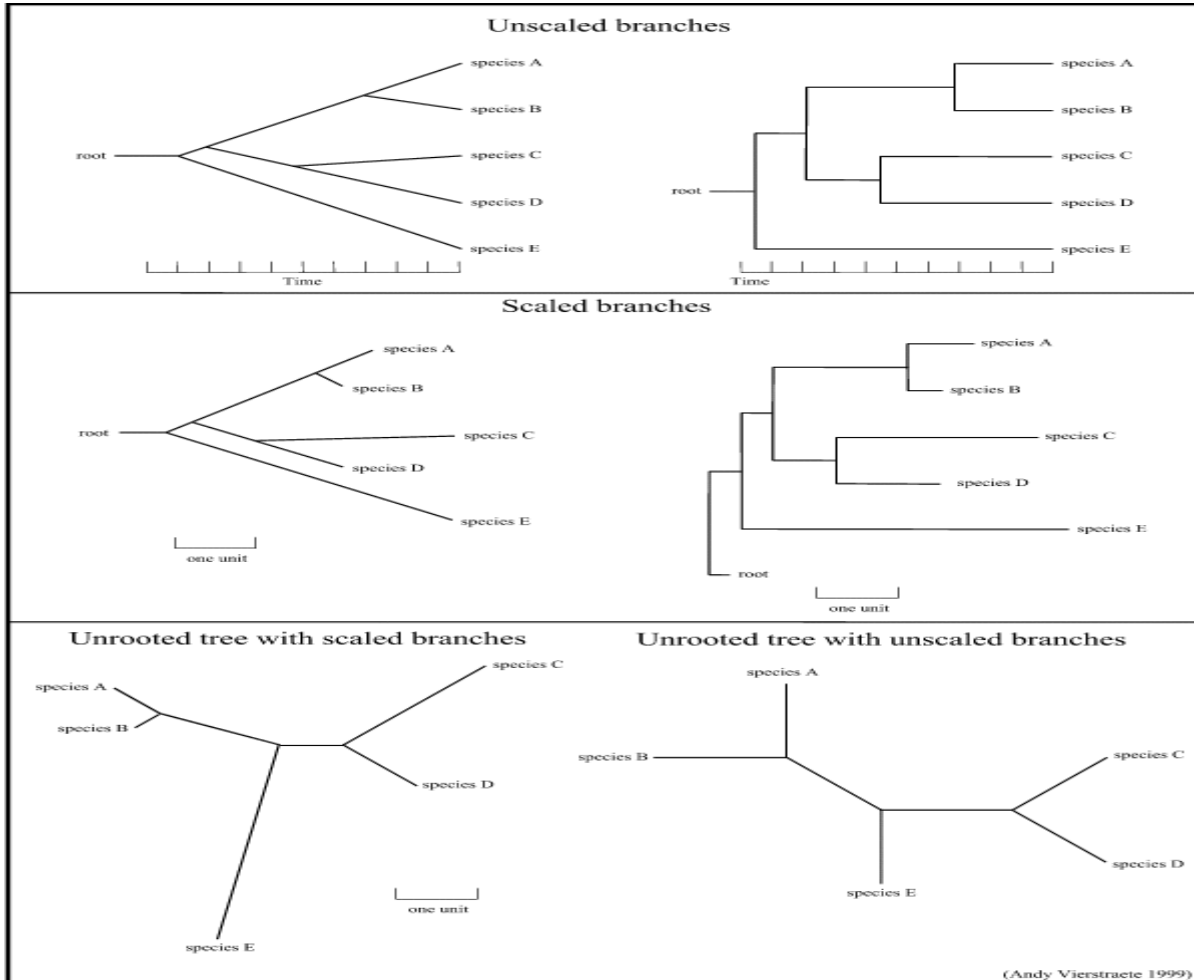
The tree terminology

4. Possible ways of drawing a tree:

Trees can be drawn in different ways. There are trees with **unscaled branches** and with **scaled branches**.

- **Unscaled branches**: the length is not proportional to the number of changes. Sometimes, the numbers of changes are indicated on the branches with numbers. The nodes represent the divergence event on a time scale.
- **Scaled branches**: the length of the branch is proportional to the number of changes. The distance between 2 species is the sum of the length of all branches connecting them.

It is also possible to draw these trees with or without a root. For **rooted trees**, the root is the common ancestor. For each species, there is a unique path that leads from the root to that species. The direction of each path corresponds to evolutionary time. An **unrooted tree** specifies the relationships among species and does not define the evolutionary path.



Some possibilities for drawing a tree. (these are just a few examples, there are a lot of variations possible)

5. Methods of phylogenetic analysis :

There are two major groups of analyses to examine phylogenetic relationships between sequences:

1. Phonetic methods: trees are calculated by *similarities of sequences* and are based on **distance** methods. The resulting tree is called a **dendrogram** and does not necessarily reflect evolutionary relationships. Distance methods compress all of the individual differences between pairs of sequences into a single number.

2. Cladistic methods: trees are calculated by considering the *various possible pathways of evolution* and are based on **parsimony** or **likelihood** methods. The resulting tree is called a **cladogram**. Cladistic methods use each alignment position as evolutionary information to build a tree.

5.1. Phenetic methods based on distances:

1. Starting from an alignment, **pairwise distances** are calculated between DNA sequences as the sum of all base pair differences between two sequences (the most similar sequences are assumed to be closely related). This creates a **distance matrix**.
 - All base changes can be considered equally or a matrix of the possible replacements can be used.
 - Insertions and deletions are given a larger weight than replacements. Insertions or deletions of multiple bases at one position are given less weight than multiple independent insertions or deletions.
 - it is possible to correct for multiple substitutions at a single site.
2. From the obtained **distance matrix**, a phylogenetic tree is calculated with **clustering algorithms**. These cluster methods construct a tree by linking the least distant pair of taxa, followed by successively more distant taxa.
 - **UPGMA clustering** (Unweighted Pair Group Method using Arithmetic averages) : this is the simplest method
 - **Neighbor Joining**: this method tries to correct the UPGMA method for its assumption that the rate of evolution is the same in all taxa.

5.2. Cladistic methods based on Parsimony :

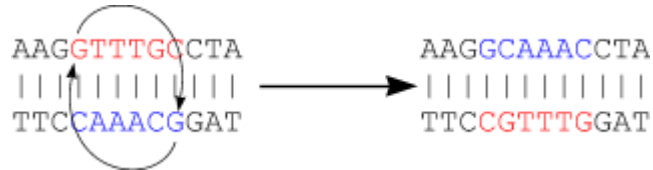
For **each position** in the alignment, **all possible trees** are evaluated and are given a score based on the number of evolutionary changes needed to produce the observed sequence changes. The **most parsimonious tree** is the one with the **fewest evolutionary changes** for all sequences to derive from a common ancestor. This is a more time-consuming method than the distance methods.

5.3. Cladistic methods based on Maximum Likelihood:

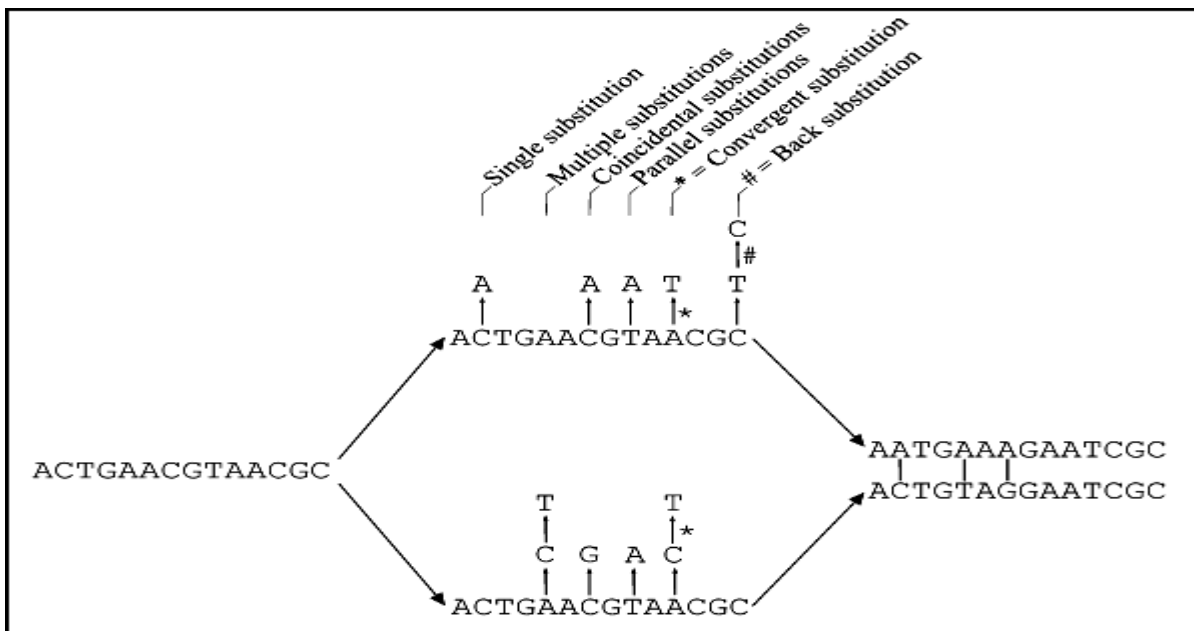
This method also uses **each position** in an alignment, evaluates all possible trees, and calculates the **likelihood for each tree** using an explicit **model of evolution** (<-> Parsimony just looks for the fewest evolutionary changes). The likelihood's for each aligned position are then multiplied to provide a likelihood for each tree. The tree with the maximum likelihood is the most probable tree. This is the slowest method of all but seems to give the best result and the most information about the tree.

6. Theoretical problems with evolutionary changes between sequences

- Transitions: substitutions from A to G ; G to A ; C to T ; T to C.
- Transversions: substitutions from G to C ; C to G ; T to A ; A to T.
- Deletions: removal of one or more nucleotides.
- Insertion: addition of one or more nucleotides.
- Inversion: rotation of 180 °C of a double stranded DNA segment comprised of 2 or more base pairs.



The next figure shows that there is a chance that many more mutations occur than visible at a certain time. Even the best evolutionary models can't solve this problem...



Two homologous DNA sequences which descended from an ancestral sequence and accumulated mutations since their divergence from each other.