e-PG Pathshala

Subject : Introduction to Visualization

Paper: Visualization Techniques

Module 28 : Abstractions in Visualization

Module No: -------28------

Quadrant 1 – Visualization

Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. We start with an overview of Visualization Definition and History of Visualization in this module. The learning objectives for this module are to explore the following:

Learning Objectives:

- To introduce the types of abstractions
- To learn how abstractions are applied in visualization
- To learn about the design principles of abstraction
- To know about the examples of abstractions in Visualization

1.1 Introduction of Abstractions

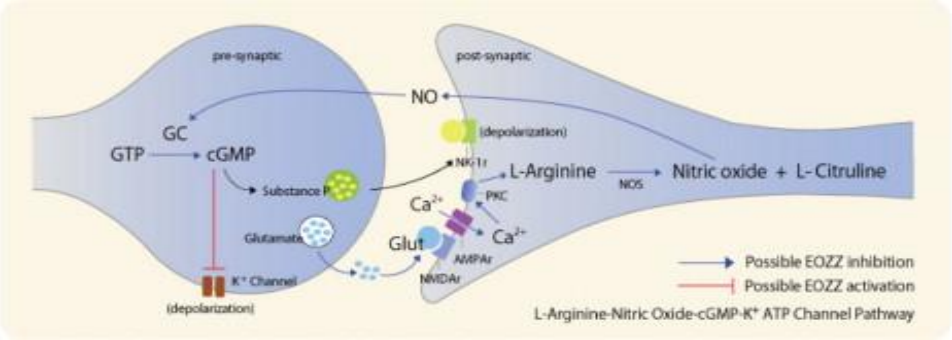
Abstraction is the process of hiding details while maintaining the essential characteristics. Various types of abstractions are possible in real world scenario such as, Graphic abstraction, Layout abstraction, Event abstraction, Data abstraction

1.1.1 Graphics abstractions

Graphics abstractions, allow us to draw a line, fill a shape or rotate a drawing, are an essential part of programming interactive visualizations. A new declarative approach to resolution-independent 2D graphics that generalizes and simplifies the functionality of traditional frameworks, while preserving their efficiency. A framework which makes it

easier to produce high image quality and makes non-affine transformations more efficient. As a real-world example, the implementation of focus context lenses gives higher image quality and better performance.

The graphical abstract will be one image file and that will visualize the one process or it will make one point clear. Graphical abstraction must have a clear form of start and the end for ease of use by the user, and also the user can prefer the reading from the left to right or from top to bottom. Here the user must try to decrease the cluttering elements and distraction as much as possible.



1.1.2 Layout abstractions

Interactive visualizations often use abstractions that produce some kind of layout, such as a force directed graph layout or a tree map. When laying out trees in a node-link diagram, a classical algorithm exists that produces the layout in linear time, but the resulting layout takes more space than necessary. Layout abstractions present a novel algorithm that also runs in linear time, but produces more compact drawings.

1.1.3 Event abstractions

Interactive visualizations need to deal with events such as mouse clicks and touch commands. Dealing with such events is traditionally done with either blocking I/O or callbacks. The former requires concurrency to compose reactive parts, which leads to non-determinism. The later leads to inversion of control: the control-flow of the program is dictated by the events that occur, not by the programmer.

1.1.4 Data abstractions

Data abstraction techniques are widely used in multi resolution visualization systems to reduce visual clutter and facilitate analysis from overview to detail. The analysts

are usually unaware of how well the abstracted data represent the original dataset, which can impact the reliability of results gleaned from the abstractions. There are three types of data abstraction quality measures for computing the degree to which the abstraction conveys the original dataset: the Histogram Difference Measure, the Nearest Neighbor Measure and Statistical Measure.

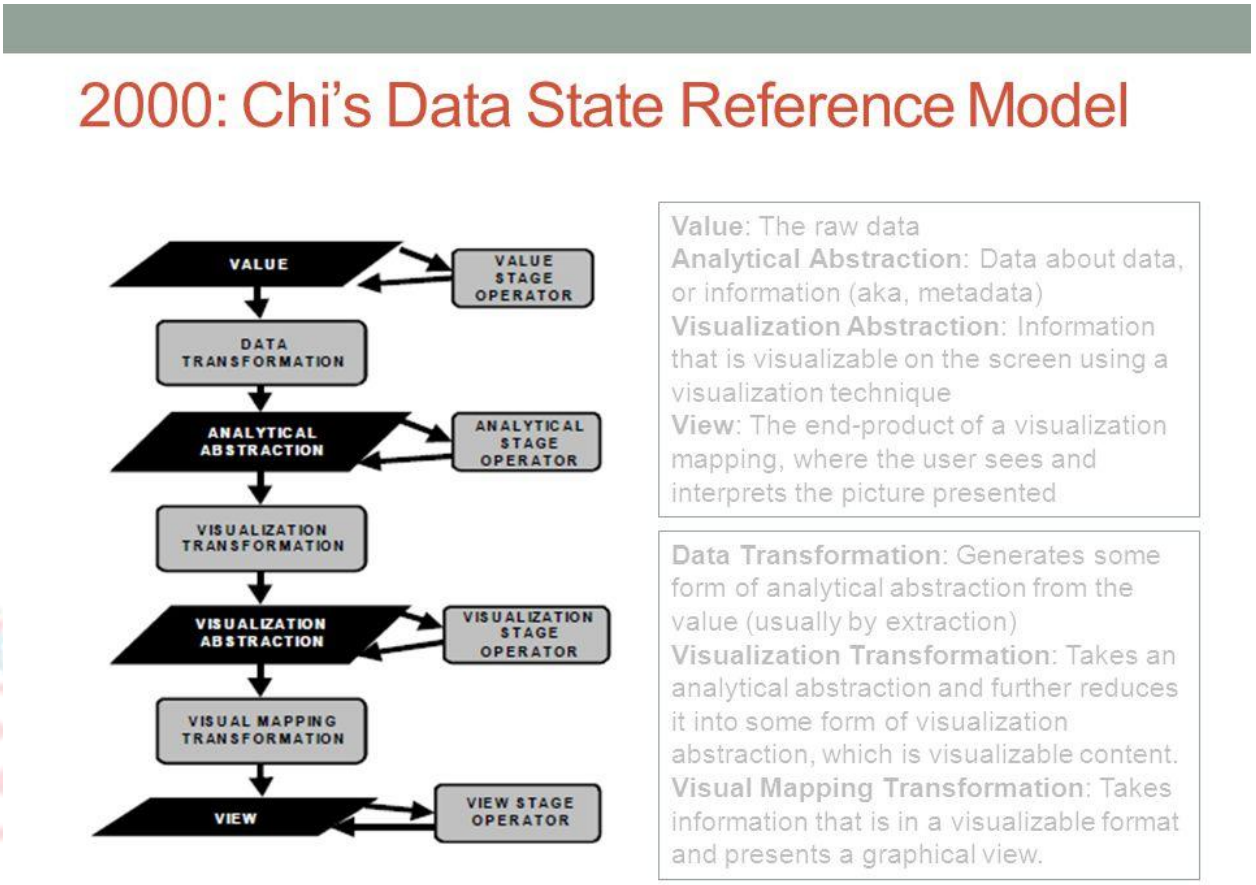


Fig 1 : Abstraction Layers

1.2 How Abstractions are applied in visualization

1.2.1 Multi scale Visualization

Visual representation changes as user pans and zooms. lots of data is highly abstracted, Zoom, data is having density decreases detailed information.

Visual and data abstraction

Visual abstraction refers to different representation/same data. Data abstraction refers to Transformations to reduce data set size.

1.2.2 Existing Multi scale Visualizations

There are some of the existing multi scale visualizations are available namely, Cartography, Multi scale information visualization, Pad++: Alternate desktops, Data Splash, XmdvTool, ADVIZOR. Main limitations are, One zoom path, Primarily visual abstraction. Multi scale visualization with both visual and data abstraction using generalized mechanisms, Data Abstraction is done by using Data Cubes and Visual Abstraction is done using Polaris .

1.2.3 Specifying Multi scale Visualizations

Specify multi scale visualization using a graph of Polaris specifications is done by Zoom Graph.

1.2.4 Exploring Data Cubes using Polaris

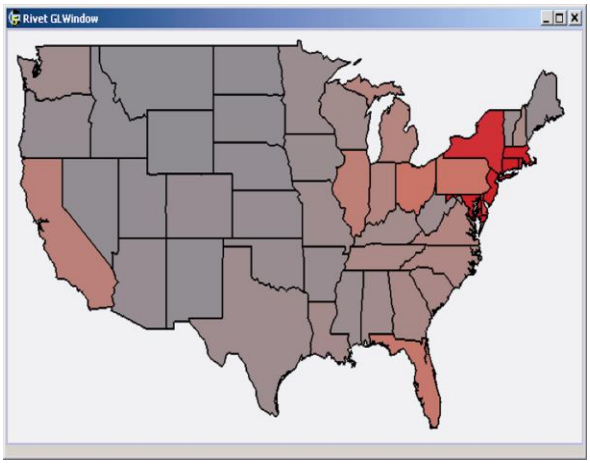
Polaris contains A UI for exploration, analysis of data warehouses, A formal language for specifying queries & visualizations and An interpreter for compiling specification into queries/drawing commands.

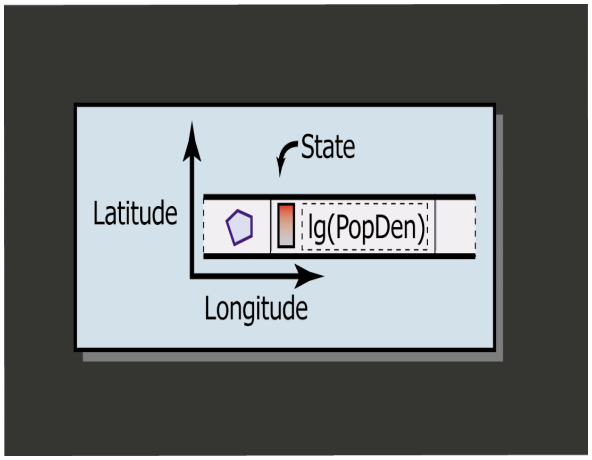
1.2.5 Polaris Formalism

Visualization described using *visual specifications* that defined by, Table configuration (algebra), Type of graphic in each pane, Encoding of data as visual properties of marks (encoding system), Data transformations and queries, Each specification corresponds to a projection of the data cube.

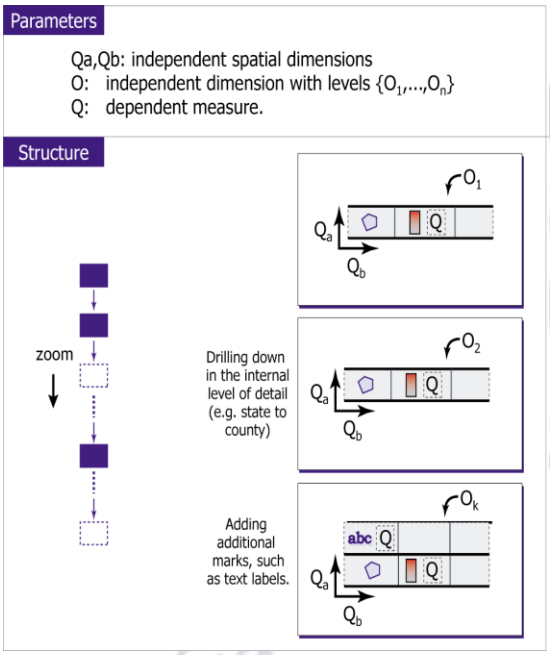
1.3 The Design principles of abstraction

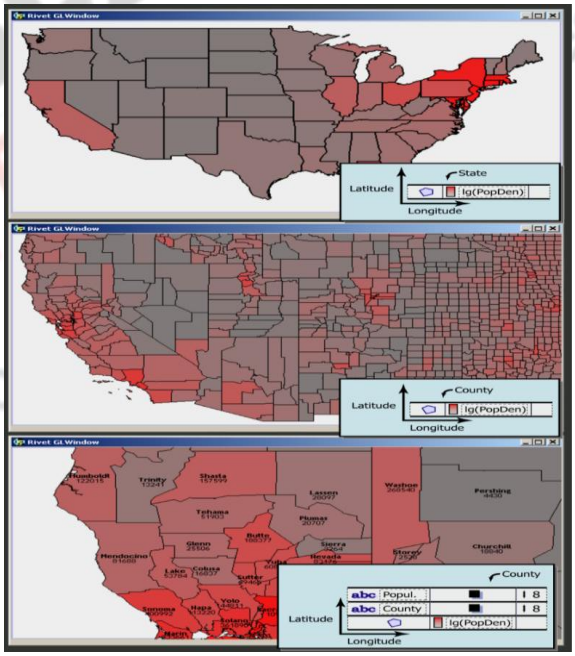
Zoom graphs simplify specifying and implementing multiscale visualizations such as Design is still very hard. "*Design patterns*" (a la Gamma et al.), Capture zoom structures that have been used effectively & reuse in new designs.

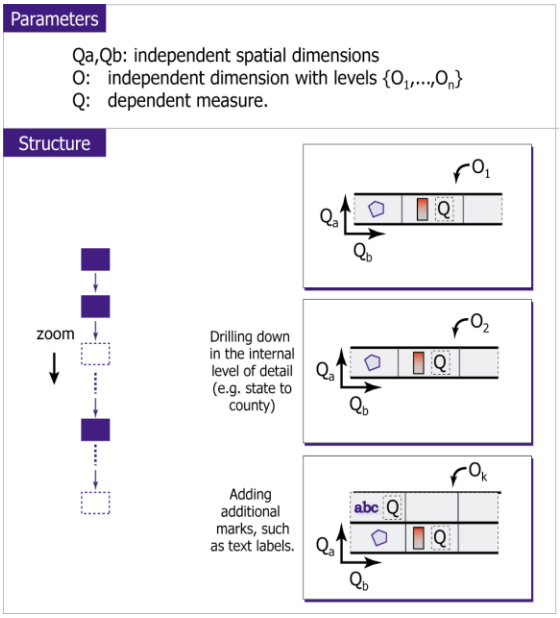


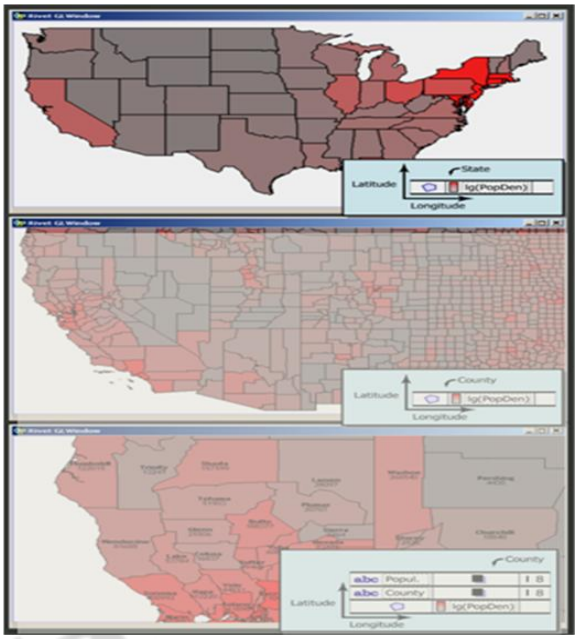


1.3.1 Thematic Maps

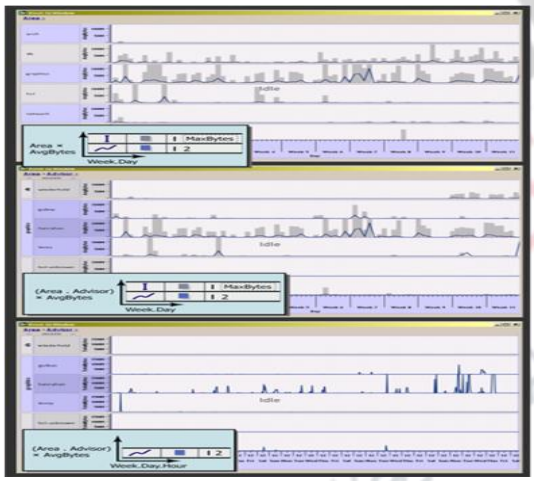


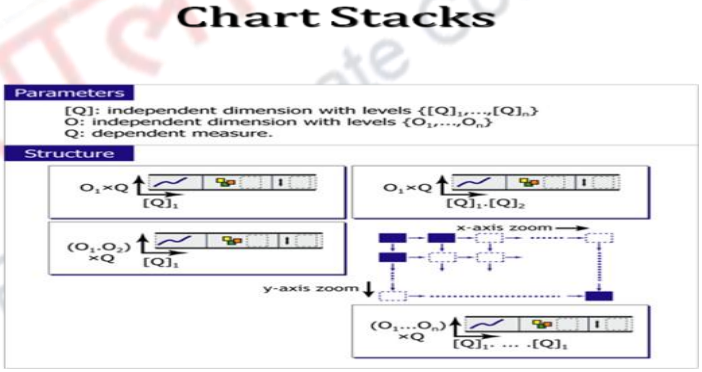




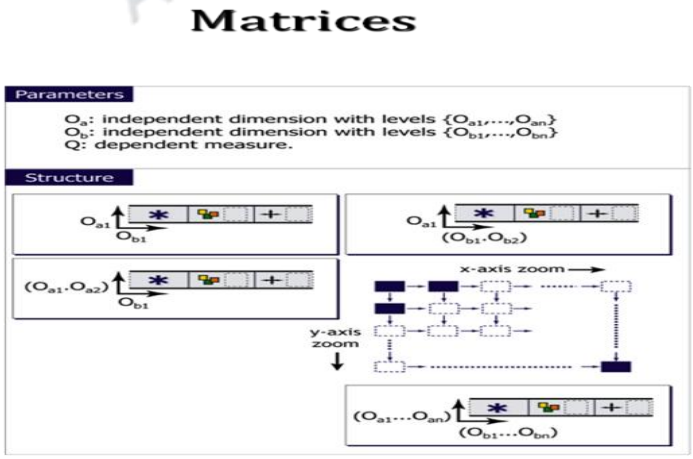


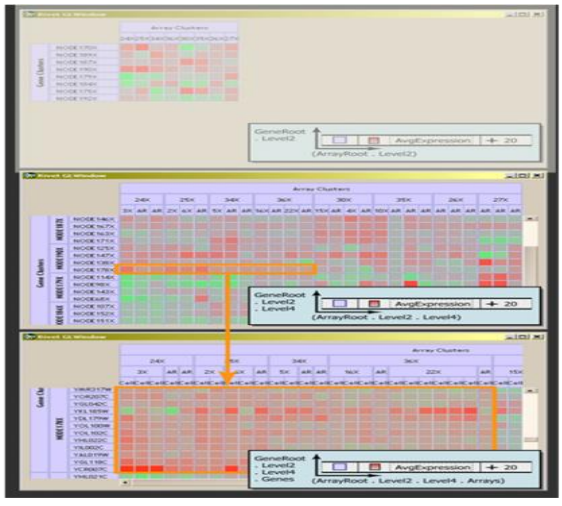
1.3.2 Chart Stacks





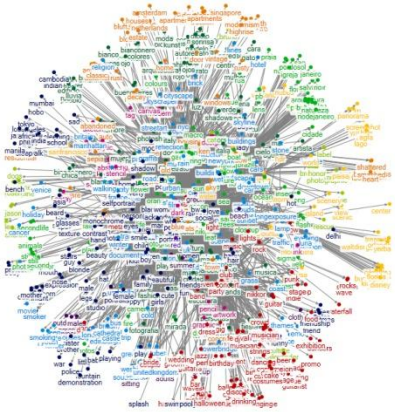
1.3.3 Matrices



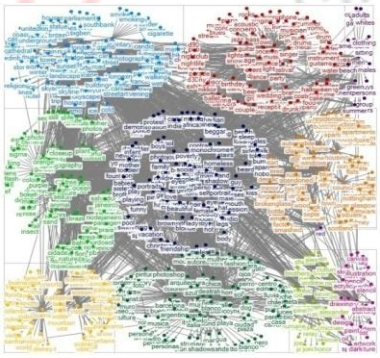


1.3.4 Visual Abstractions

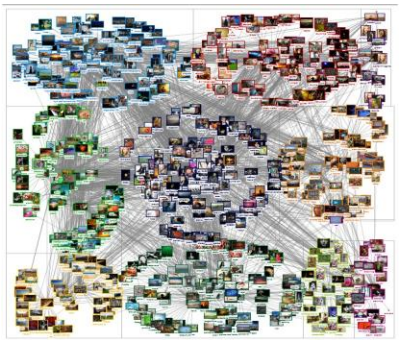
To visualize the respective related tags, there are two algorithms that seem the most effective: Fruchterman-Reingold force-based algorithm and Harel-Koren Fast Multiscale. For related tags networks, both include default vertex labels. The following ("A 'Social' Related Tags Network…") was laid out using the Harel-Koren Fast Multiscale layout algorithm.



The most visually clear way to access this data is to partition the groups. The partitioning may be set up in the Graph Options area with the direction to lay out the graph's groups in their own boxes in "packed rectangles."



To make the visualizations more understandable, it may help to "Scale" the visualization in the Graph Pane, so that the images and the text labels may be easily read and so there is less overlap.



1.3.5 Abstractions using Programming languages

Different programming languages provide different types of abstraction, depending on the intended applications for the language. For example: In object-oriented programming languages such as C++, Object Pascal, or Java, the concept of **abstraction** has itself become a declarative statement - using the keywords virtual (in C++) or abstract and interface. Functional programming languages commonly exhibit abstractions related to functions, such as lambda abstractions (making a term into a function of some variable), higher-order functions (parameters are functions), bracket abstraction (making a term into a function of a variable).

1.3.6 Control abstraction

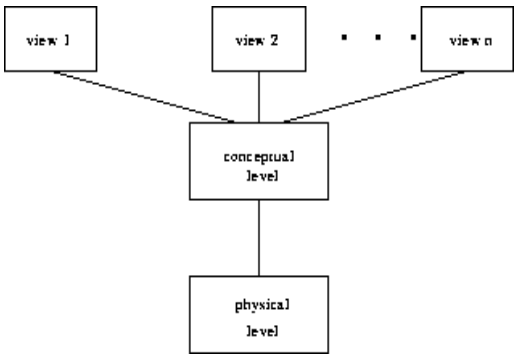
Programming languages offer control abstraction as one of the main purposes of their use. Computer machines understand operations at the very low level such as moving some bits from one location of the memory to another location and producing the sum of two sequences of bits. Programming languages allow this to be done in the higher level.

1.3.7 Abstraction in object oriented programming

Various object-oriented programming languages offer similar facilities for abstraction, all to support a general strategy of polymorphism in object-oriented programming, which includes the substitution of one type for another in the same or similar role. Although not as generally supported, a configuration or image or package may predetermine a great many of these bindings at compile-time, link-time, or load time. This would leave only a minimum of such bindings to change at run-time.

1.3.8 Database Abstraction

The main process of a database is to provide the user with the view as an abstract for the system. The systems may hides some details that how the data was created, stored and maintained. The complexity will be hidden from the users. There are several levels of abstraction:



There are three levels of abstractions namely, Physical Level, Conceptual Level, View Level

Physical Level

In Physical level How the data are stored. E.g. index, B-tree, hashing, It is Lowest level of abstraction.

Conceptual Level

Next highest level of abstraction is known as conceptual level.. In Conceptual Level Describes what data are stored. And also describes the relationships among the data, Database administrator level.

View Level

The view level will describe the part of database to a particular groups of user. It will contain more different views in database. For example, the teller in a banks will gets a view of the customer account, but they not get the payroll data's.

1.3.9. Data Models

The Data model is refer as a collection of the conceptual tool used for describing data's, data constraint, data semantic and the data relationship. The data model contain three different group there are, Record-based Logical Models, Object-based Logical Model and Physical Data Models.

1.3.9.1 Object-based logical models

The object based logical model will describes the data in conceptual and in view level. It provides the fairly flexible structured capability. And also it allows the one to describe the data constraints by explicitly. Here describe the some of the model of based logic are, Binary model, Entity relationship model, semantic data model, functional data model, object oriented model and Info logical model.

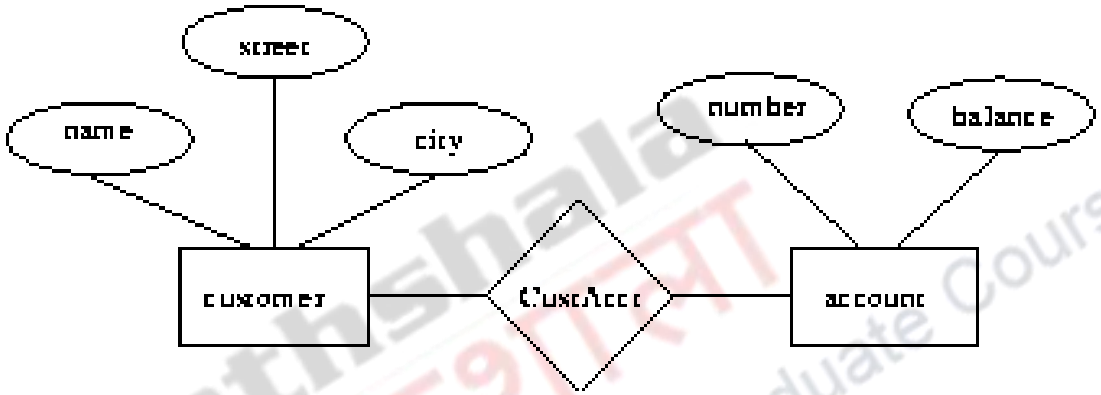
1.3.9.2 Record based Logical Model

Entities in this model will be distinguishable object which exists. Every entity will be associated with the set of attribute describing on it. For example balance and the number will used for an account entities. The relationship will be association among several entities. For example Cust_acct relationship will associates with a customer to each account she or he has. The set of every relationships or an entities of the similar type will be known as the relationship set or an entity set. Other important elements of an E-R diagram will be the

mapping cardinality; it will express the no. of entity in which another entity will be associate through the relationship sets.

1.3.9.2.1 E-R Model

The full logical structures of the databases will be expressed in a graph form by an E-R diagram: Here the entity sets will be represented in rectangles. The attributes will be represented in Ellipses; the relationships among the entity sets will be represented in Diamond's and finaly the link between the entity sets to relationships and attributes to entity sets will be represented in lines.



1.3.9.2.2 The Object-Oriented Model

The value stored in an instance variable with in the object. It is unlike as record oriented model, the values are referred as themselves object. The objects contain deep level of the nesting. Here an object contains the body of the code it will operate on the objects. The body of the code will also call as method. The object that has the same type of the method and same type of value are grouped in to a single class.

The class will be view as a definition of type for an object. Analogy: it refer as the abstract data types is provide through the programming language. The invoking method is used for the access the data of one object by another object. This is referred as the sending the message between the objects. The object has some internal parts namely method code and the instance variable will not be visible externally. The results are the two level of the data abstraction.

1.3.9.2.3 The Relational Model

In relational model data and relationships are represented by a collection of tables. Each table has a number of columns with unique names, e.g. *customer, account*.

name	street	city	number
Lowery	Maple	Queens	900
Shiver	North	Bronx	556
Shiver	North	Bronx	617
Hodges	Sidehill	Brooklyn	801
Hodges	Sidehill	Brooklyn	617

name	balance
900	55
556	100000
617	105366
801	10533

1.3.9.3 Physical Data Models

The physical data model will be used to describe the data's at the low levels. It has very few models which are, frame memory and unifying model.

Benefits of Data Abstraction:

First, the interface protects the implementer from incorrect use of the data structure by the client. An interface creates an abstraction barrier that protects the implementation. A client that uses knowledge of the implementation not contained in the interface is violating the abstraction barrier. Second, when something goes wrong in a program, the presence of an interface makes it easier to assign blame to either the client or the implementer. Either the client is using the interface incorrectly (possibly violating the abstraction barrier), or the implementer is implementing it incorrectly. A clear interface makes it possible to argue that one of the two needs to fix their code.

Benefits of abstraction:

Third, the interface also gives the implementer flexibility to change the implementation, as long as the client is only using the implementation through the defined operations, and those operations are still provided by the new implementation. This flexibility results in a loosely coupled system. The interface creates a contract between the client and implementer, which frees them up to work more independently as long as they stick to the contract. In a tightly coupled system, the client and the implementation are not as free to change because changes to either are more likely to change the interface.

Procedural Abstraction:

Procedural abstraction provides mechanisms for abstracting well defined procedures or operations as entities. The implementation of the procedure requires a number of steps to be performed. A simple example is a *debit* operation which performs various steps to debit certain amount from the bank account. Hence at the banking level, *credit* and *debit* become well defined procedural abstractions. Procedural abstractions are used extensively by requirements analysts, as well as designers and programmers. Procedural abstractions are normally characterized in a programming language as "function/sub-function" or "procedure" abstraction.

The Temporal-Abstraction Task:

Time-stamped clinical data and relevant events (interventions). This is an interval-based abstraction. It identifies past and present trends and states. Supports decisions based on temporal patterns.

Uses of Temporal Abstractions:

Therapy planning and monitoring (e.g., to support guideline-based care). Creating high-level summaries of time-oriented medical records. Supporting explanation modules for a medical DSS. Visualization and exploration of time-oriented clinical data.

1.4 Examples of abstractions in Visualization