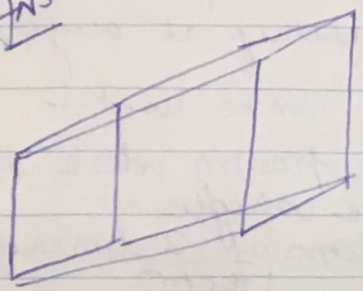
EXCERCISE → 6

GEOMETRIC MODELLING

Ans 1)



How to program it in programming

* can be described using
  - 8 vertexes
  - 12 edges
  - 6 faces
  - 3 dimensional

① 4·3·8 = 96 bytes (24 values)
② 4×3×2×12 = 288 bytes (72 values)
③ 4×6×3×4 = 288 bytes (72 values)
④ ~~3×4×4+3·4=60 bytes~~
9×4 = 36 bytes (9 values)

Solution 1 - 8 vertexes:

Vertex : x, y, z
cuboid : v[8]

```
class cuboid
  { vertex v[8];
  }
class vertex
    { float coordinates [3];
    }
```

→ 4 bytes 4×3 = 12 bytes

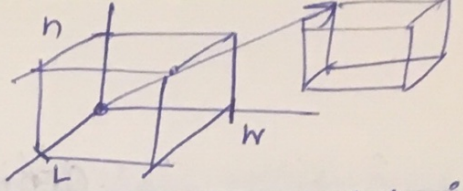
solution 2 - 12 Edges → comprise of 2 vertexes → 3 coordinates

```
class cuboid
  { Edge e [12];      (4×3×2×2)
  }
class edges
  {vertex v[2];       (4×3×2)
  }
class vertex
  { float coordinates [3];   (4×3)
  }
```

solution 3 - 6 faces →

```
class cuboid
  { Face f[6];
  }
class vertex
  { float coordinates[3];
  }
```

```
class face
  { vertex v[4]
  } or list<vertex> v;
```

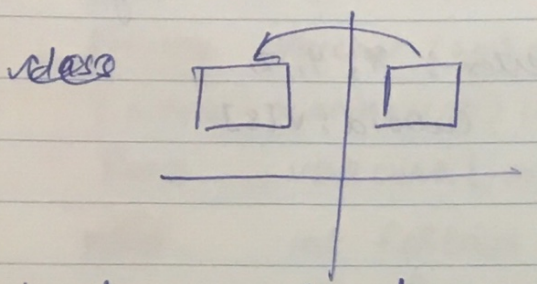


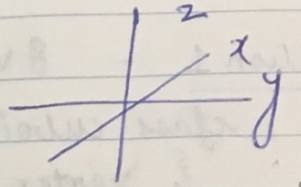
Solution 4 Cuboid :→ length
width
breadth

Here I am just defining something at origin

All cubes cannot be described using only l, b, h and translation vector. Angle is required

Transformation can be used to describe the position where it needs to be shifted
describe it terms of 3 dimensional vector





In two dimension I need only one angle

angle across all axis for 3 dimension x, y, z
We need 3 angles.

Not flexible → I can only define cuboid

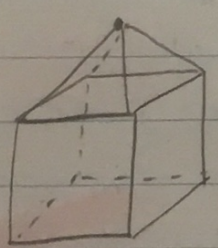
Storage consumption improved
efficiency is also improved
performance

(9 valued)

```
class cuboid
{ float dimension [3];
  float translation vector [3];
  float rotation [3];
}
```

Ans :→2 Ways/Methods and structures to describe geometries/models
Part of 3d designs

(1) Wire Frame Models:-

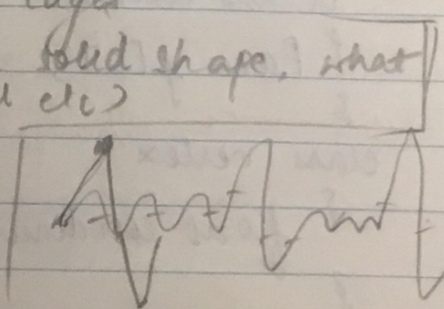


Data structures available for wire frame – edges and vertices
but edges is a topological concept)
(connection between two points — line
— curve B Splines
(specific way to describe boundary curve)

— lot of stuff is missing

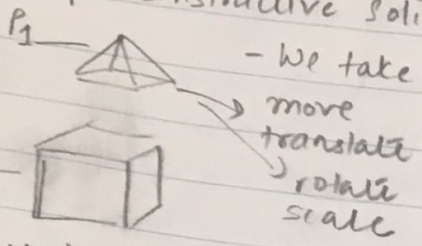
— We store info about 9 vertices, 16 edges
↳ what we don't have is let it a solid shape, what material etc)
↳ No info of face/shape

* Early computer graphics where computation problems was restrictive



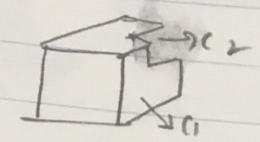
② → exam question

CSG : Constructive Solid Geometry



- We take primitive cuboid, pyramid } basic operations done to primitive shapes

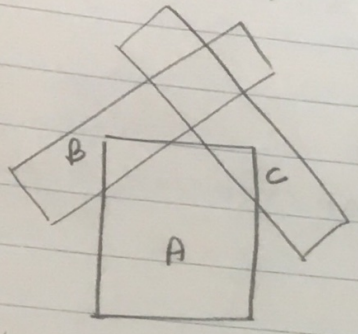
* Union (U) = $P1 \cup C_1$
* ~~Intersection (∩)~~
  difference (−)



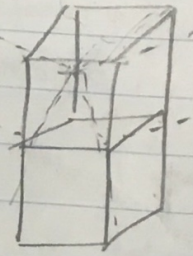
$C_1 - C_2$

* Intersection (∩) common

- We take 1 primitive cuboid



A − B − C

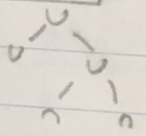


disadvantage → expressing complex geometries becomes difficult

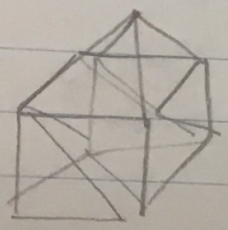
house = $(C_1 \cup C_2) - (C_3 C_4 C_5 C_6)$

advantage → easy to use for building to human

Data structure : Cuboid, Pyramid etc... (position, rotation, scaling
Sphere, encoding of basic operation (union, intersect...)
encoding of complex shapes



③ Triangle Meshes : Data structure (Triangle)



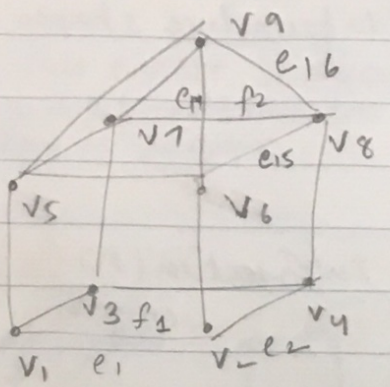
4 triangles + 10 triangles
(save space by doing 1 triangle fence and space)
(define a triangle then the next point)

④ BRep → Boundary representation (describing the volume (which is not defined in Triangle meshes))
~~1~~ Lot more flexible because we can have arbitrary surfaces

Data Structures :- Vertex, Edges, Faces (closed loop of edges), Shells (boundaries of a volume)
defined by faces

Advantage → I get info about volume

1 shell →



$f1 = e_1 e_2 e_3 e_4$

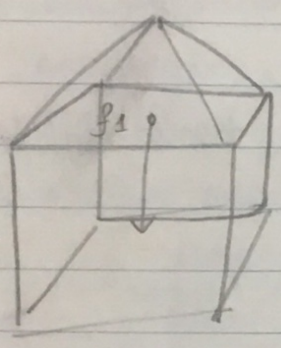
$f2 = e_{14} e_{15} e_{16}$

⋮

$f9 =$

shell = f1 - - - f9 (boundaries of my solid shape)

(5)



(1 face + 1 extrusion)

5 points + 8 edges + 5 faces

for 1 face an extrusion factor

ANS: → 3 Semantically (meaning) Rich model :- lot of modelling constructs, it has a lot of meaning encoded. It becomes easier to use. More primitives, more ways to combine to form complex shapes.

- easily understandable by humans

example file → COLLADA file (closed to BRep) (edges not used here but)

Ans:→4 - Collada files :- bringing together geometry and topological info.

- file format for data description and exchange for automative application . eg:→ Robotics
- low level description of geometries to high level functionalities
- XML file that describes collada data

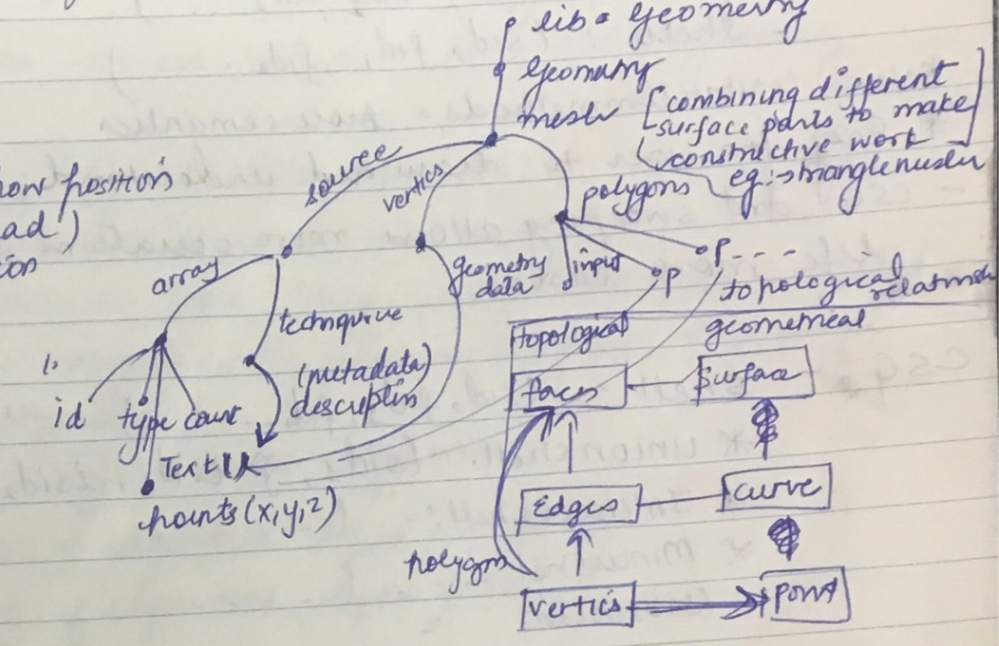
↳ semi-structured [Mixing of description of data with the data] HIERARCHICAL STRUCTURE

- What is described

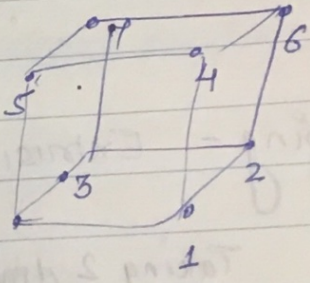
* 8 vertexes

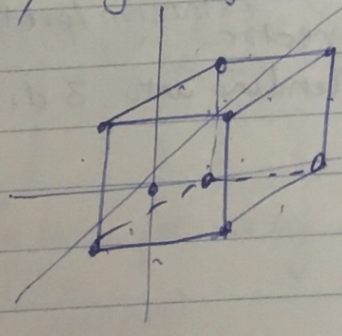
technique (description how position array should be read) interpretation

Metadata



- In <vertices>
  - it is pointing towards source (relationship between branches)
- <polygon> → defines faces (6 polygons)





cube (same length, width and height)

- Cross references are required with an XML document

* Geometrical data is what things are
* Topological is how things are interrelated

EXCERCISE-7

Exam
Imp

Ans:→1 BRep:

Basic Topology:

- vertices : (vid, x, y, z).
- edges : (eid, vid$_1$, vid$_2$)
- faces : (fid, eid$_1$, eid$_2$....)
- shells : (sid, fid$_1$, fid$_2$ ----)

Shells
↑
Faces — surface (type)
↑
edges → curve (type, parameters)
↑
vertexes - points (x, y, z)

*New constructionmethods = more semantics
* Easy for user to draw and understand
- CSG and sweeping allows more operations to change and life more easier

CSG:- shell := (sid, fid$_1$, fid$_2$ ----)
* unionshell: (sid$_1$, sid$_1$, sid$_2$) (union class)
→ inputshells
→ type encodes the operation
* Intersectshell := (- - - -)
* Minusshel - - - (- - - -)
(Substraction :-

Sweeping - Extrusion shell (sid, fid, x, y, z)
shellid ↑ face on which extrusion is performed
↳ vector extrusion/protruding

Taking 2 dimension surface and extending into 3 dimension

→ Extrusion = face + vector
→ Rotation = face + axis + angle

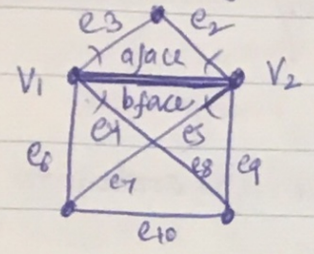
Each edge
- actual data
- defining vertex
- defined faces
- Neighbor edge

vertex
- actual data
- defined edges

faces
defined edges
actual data

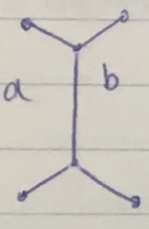
Ans:→2 Winged Edge Data Structure:

→ Explain BRep again — Hierarchical topological concepts (shell)
vertices → edges → faces → volume



→ Winged Edge Data structure :→ classic representation of BRep

* Redundant data
- Defines not only the vertexes but also faces and neighbouring Edges
- For every edge we store more info
* we need direct connection to the faces for some algorithms to work. They always work with connected data. Therefore for the faces we define the neigh
- bounding edges



In programming language objects are the reference

→ List<WeEdge> → list of edges that define vertex
→ List <WE_Edge> → list of edges that define face

* How can the same data be represented in relational database system

WEdge

eid	v1	v2	af	bf	aprev	anext	bprev	bnext	data
e1	v1	v2	fa	fb	e2	e3	e4	e5	

(1) One table

A ~~xxxxx~~ table for every class

(2)

WEVertex

vid	edges	data
v1	e3 e1 e4 e6	

Not normalized → Normalization required

Vertex edge

v'id	e'id
v1	e3
v1	e1
v1	e4
v1	e6

WE face

fid	data
fa	
fb	

Edge face

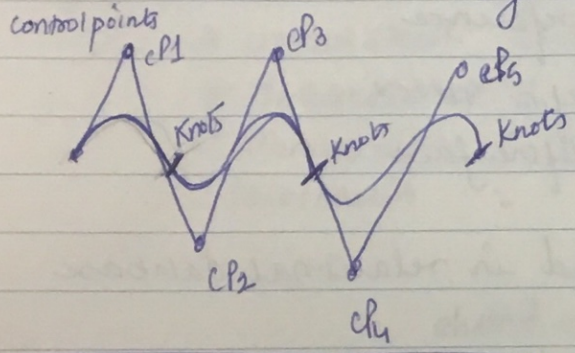
edge	face
e1	fb
e2	fa
e3	fb

Advantages → less joins in the first approach
* could provide performance benefits but lot of inconsistencies

→ ~~At~~ every step we store neighbouring edges / references in order to make algorithms quicker.

Ans:→3 Some part of STEP AP203
example how an edge which is a rounded curve in terms of being Bspline

* SCHEMA definition in terms of Express.
3 Entities — edge curve, bspline curve, b-spline curve with knots



* Have some control points that control the curvature
* Rounded curve is computed based on the control points
* Rounded curve is a computed partial polynom
→ These polynoms are called knots

If their is a certain control point and passing through it 3 times then I am forcing the curve to form the polynom

certain mechanism related to computer graphics !

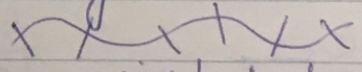
* Knots define how control points should be used

* #1518 = lables and some references to vertex point

edge curve has an edge geometry → difference between topological and geometrical model. where meshs edges are defined by vertexes

#1335 2 → is some kind of curve

degree = (3)



Splines are consisted of partial functions.
The partial function is a polynom of degree

→ (#5569, #5570 . . . . . #5578) → list of control points

→ bspline curve form = unspecified some type of attribute value

→ closed curve = false → self intersect = false

starting point = end point

→ knot multiplicities = Tells how many times the knots is repeated

→ knots = defined by float values

EXERCISE 8

Ans:→1 Storage of CAD data in RDBMS

Product Lifecycle Management [common practice]

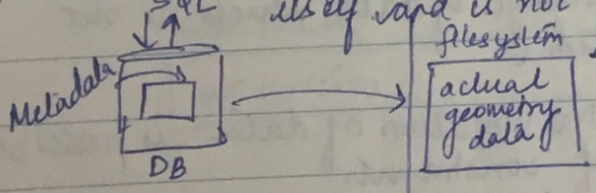
We don't do ✗

	Metadata	BLOB	Database filesystem	Structured Data in tables
Access	↗	→	↗	↘
Concurrency	↓	↓	↓	↑
Consistency	↘	→	↘	↑
Performance	↑	↗	↗ (easier than BLOB)	↓

*use CAD systems to manage/manipulate geometrical data

→ RDBMS - state of the art technology

*) Metadata → info about versioning, states, who created it etc. Not the geometry itself and is not stored in the database but in CAD files maybe



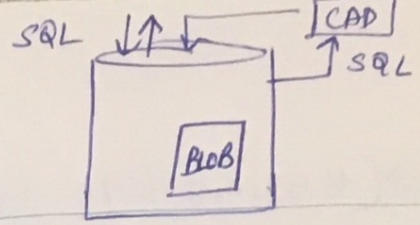
a) If we want to access Metadata → SQL for data in CAD system → use/open files easily

b) Concurrency :→ Can many people access data at the same time Not allowed to do operations in parallel (specifically write operations)

c) Consistency :→ In DB we can put integrity constraints on metadata We can apply application logic in CAD system. Consistency can be achieved till a certain level. But one can also open a file with other editors and manipulate the file.

d) Performance :→ We can quickly access data

DISADVANTAGES :- Some data are stored outside database so we dont achieve concurrency and consistency.



* BLOB - Binary Large object | Huge number of types contents of BLOB will be same as file format

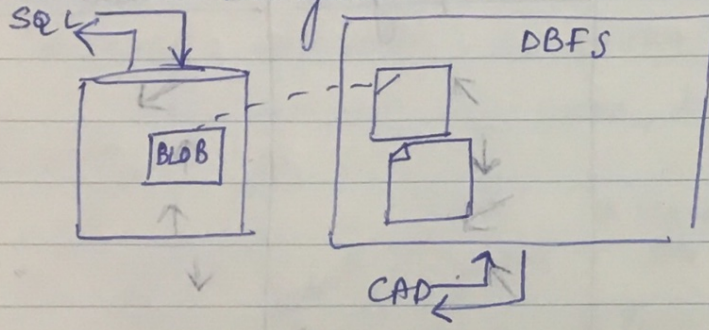
a) Disadvantage :→ Whenever we have to view a file it is difficult as we have to write SQL and access the file

b) Concurrency :- level of control is same as file system because we typically put locks on the file when someone is accessing it

c) Consistency : → We still dont have control on the inside of BLOB. But we can control whose accessing it. Grant access rights, revoke access rights. Opening from other editors is not possible

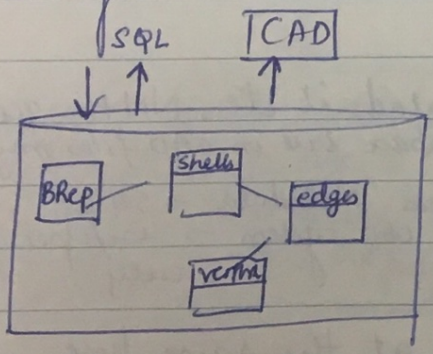
A - d) Performance :→ will take a little while to read data

* Database File System :- Virtual Data Base file system : (interface on top of files.



a) This will be helpful for easy accessible since it no longer uses SQL to fetch the geometrical data.

* Storage as structured data in Tables



a) Access : access is not easy due to complex queries

b) Concurrency: Change of data which is not inter-related.

c) Consistency: No violation of data is possible because of constraints

d) Performance :- complex join operation to find one particular geometry Data is too complex ends up in many relationships

→ extensions to SQL 92 – structured data types

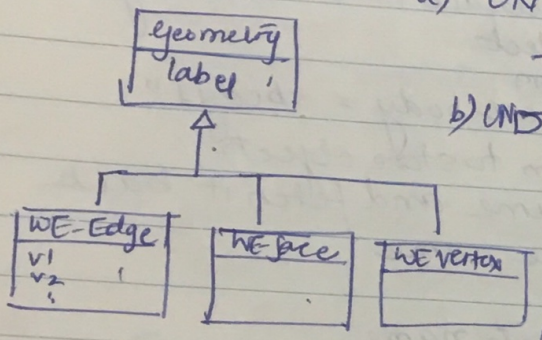
Ans:→2 @ SQL 2003 (CAD is Object relational Database systems)

→ Type geometry with Lable
→ Type hinged edge type as a part of geometry_type

* difference from SQL 92
  → References :- like a pointer replacement of primary key foreign key relationship.
  → Arrays

→ Type Hierarchy
  Table Hierarchy

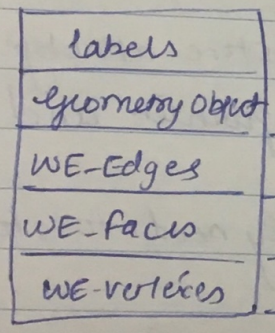
↝ Inheritance = a) UNDER geometry type
– attributs of edge type will include label



b) UNDER geometry object

select * from geometry object → label, geom object, WE-Edges, WE-faces

Table Hierarchy means that we have one table having all sub tables being accessible



→ Performance improves if we use references instead of primary key / foreign key

→ Since we are still using SQL then we have ease of access issue.

(b) Nested Tables : possible within object relational system because of types and tables

Advantages :– just one table (no joins)

disadvantage :→ Redundancy of data. Nested Table models 1:n relationship

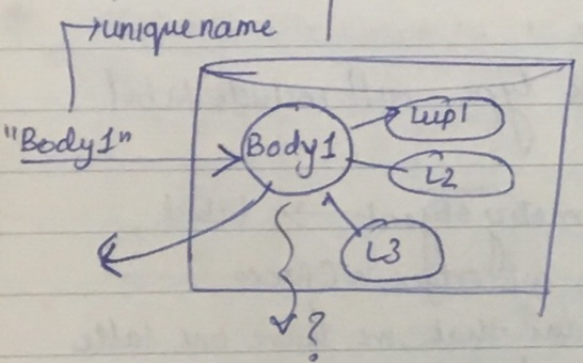
* No discription about geometry types
  eg:→ vertex can be types → splines

Ans!→3 ACIS model :- Its like a library, available software / kernel for modelling geometrical models.

Left side → wire frame Model
Right side → B Rep Model
↳ Topological concept, geometrical concept

→ How do we store objects ??

a) Named objects: How to extract data



Programming interface to extract body will be like
select
from
where body = "body1"

Simple mechanism to store objects
give a unique name and fetch it back using unique name

– Its is a bad idea if I give all objects name

b) Persistence by reachability :- Good if I also store all objects reachable from named object using pointer and references

Advantage → No need to give all objects name, only need to store entry points
→ If I don't store L4 then it will have dangling references

→ Best part for entry object is BODY

EXCERCISE-9 (DMEA) [Electronic engineering data]

Ans:→1) a) Two phases of design for Electronic circuit →
- Schematic design – abstract description of components, connection and functionality (no details of implementating) eg:→class diagram
- design for board layout – sizes, dimensions connection pass. Real description of final product.

b) Requirements to be addressed is both faces:→
- → sizes, real geometrical aspects of the board (single, multilevel), what are the connection pass.
- → 2 dimensional design, size of components.

* - In schematic connections are imp instead of pins
- Behaviour of certain components in terms of logic gates for example, functionalities, properties (charge-discharge of a capacitor)

c) functionality description of all important concepts in terms of simulation is required.

Schematic diagram is a subset of Board Layout

Schematic [- Representing component data and their properties - Electronic circuit
- Connection points (pins), - conduction pass

* EDA (Electronic Design Automation) tools helps moving from schematic to board layout

Board Layout [- size (x, y, z) - locations of pins
- size of components

Ans:→2) DTD → Schemata of XML file [Eagle file format is referring to some standard electronic data]

a) Parts:→ discuss about device = '_ _ _'

name = _

(deviceset / device) attribute which requires some reference to the description file

#IMPLIED:- Optional

DTD describes hierarchical structure of a document

b) What if I have complex components and have more values how can I describe them.

* create subparts and have several attributes of a part

<part> decrease - - - - >
<attributes> . . . . </attributes>
<attribute> - - - - <attributes>

</part>

every part there can be variants and attributes*
-* → 0 . . . ∞ ↳ arbitrary number of values 0 . . . n
-+ → 1 . . . ∞
-? → optional (0 or 1)

Ans:→3) STEP Data → semantics of STEP data
* first part contains the actual data items according to type definitions.

* Component in terms of STep is interpreted as package

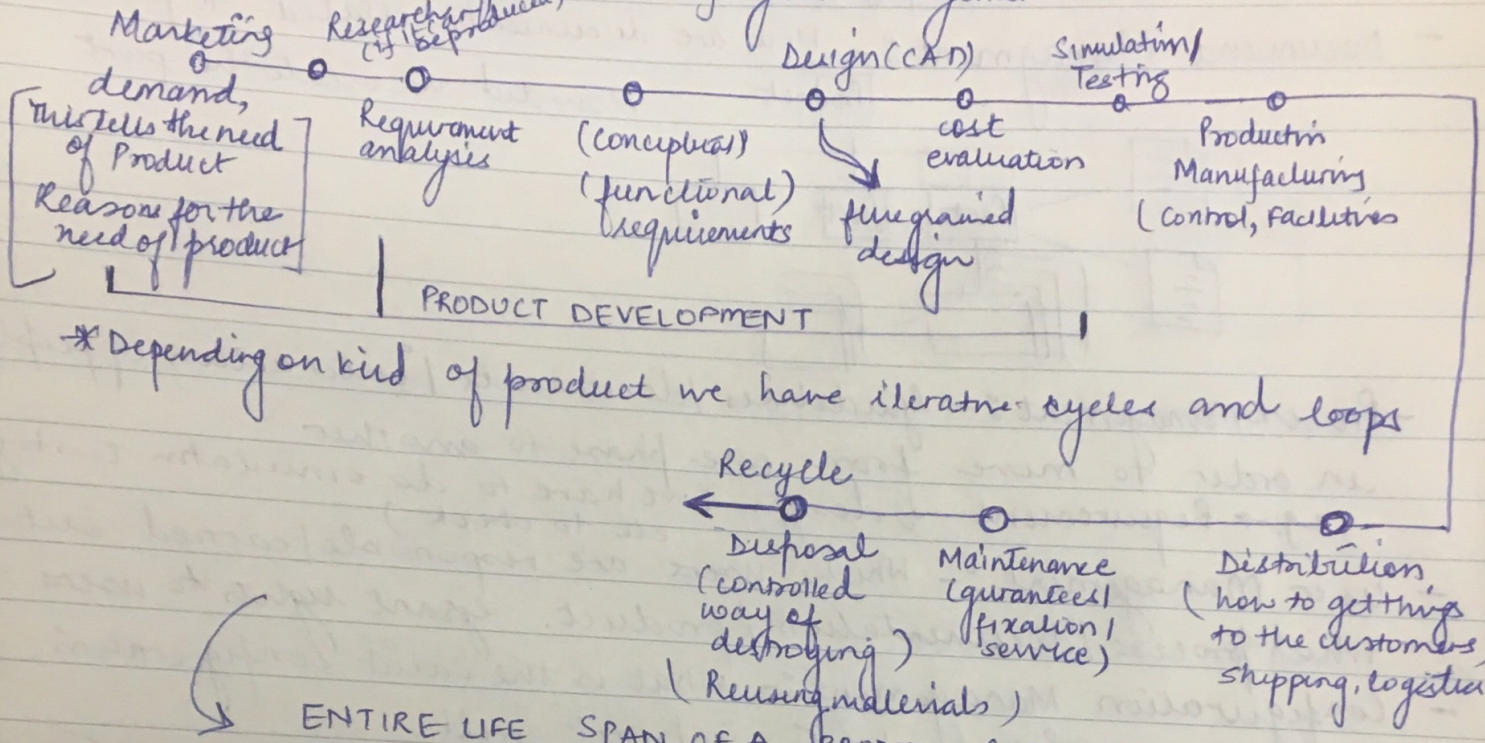
→ Last two lines don't contain any references (Cartesian point, direction)
→ #5716 ⇒ reference to direction, cartesian point
→ Usage concept → establishing relationship between generic description of a package and putting at some positions.

schematic → board

Reference as what kind of package done have and where we place it

EXERCISE 10 (DMEA)

Ans: → 1 a) Foundation of Product Lifecycle Management



* Depending on kind of product we have iterative cycles and loops

ENTIRE LIFE SPAN OF A PRODUCT (PRODUCT LIFECYCLE)

- Activities like Marketing, Research, requirement analysis } works on certain data that support the product indirectly. Product Management deals with all the aspects related to the lifecycle.

b) Database systems that cover entire life span of a product is being dealt by product lifecycle management

Why don't we integrate all data in CAD itself (Design phase)??

↙

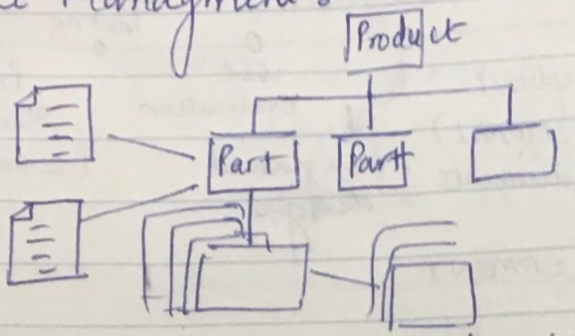
by data we mean, unstructured data, diagrams, market analysis, productions facilities, cost etc.....

* Functionality of CAD system is specific and deep (it describes how the product looks like)

* Lot of functionalities are needed in PLM for all these to be used we need specific tools/solutions eg CAD, EDA

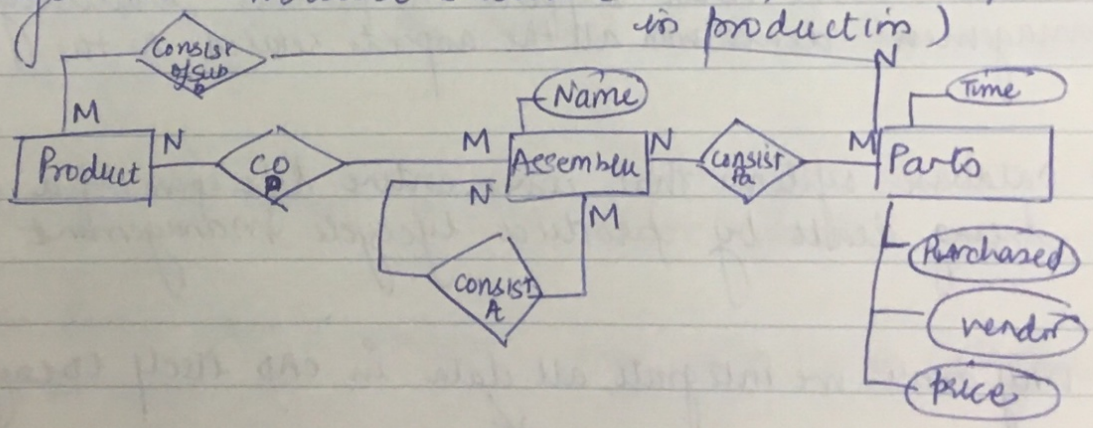
(BOM → Bill of Material - list of ingredients to build complex material

Main functions of PLM:-
- Product Structure Data/ Management: What assemblies parts its made of?? (Hierarchy) Complex parts
  How to combine subparts into the main parts eg:→ cars (components to cars
- Document Managment:- How are documents related to products
  How did we create the part



- Process Management:- Guidelines / documents / interaction of people in order to move from one phase to another
  eg:→ Requirement → design (we have to do simulation testing etc to check)
- User Management:- Which users are responsible / carried out which process / documentation / product. & grant rights to users
- Configuration Management:- what is the valid configuration of a product? What parts interact well.
  It is a very complex task? What kind of version variants could be combined. Specifies dependencies between several parts
- Status Management - Product status (Tested, in development, in production)

Ans:→2 a)

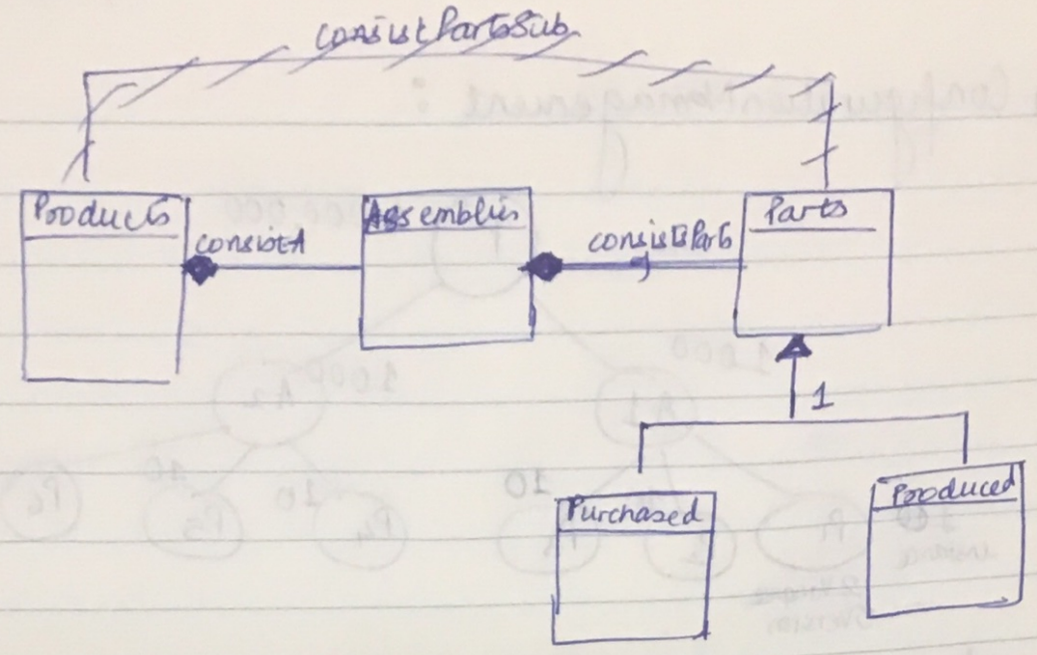


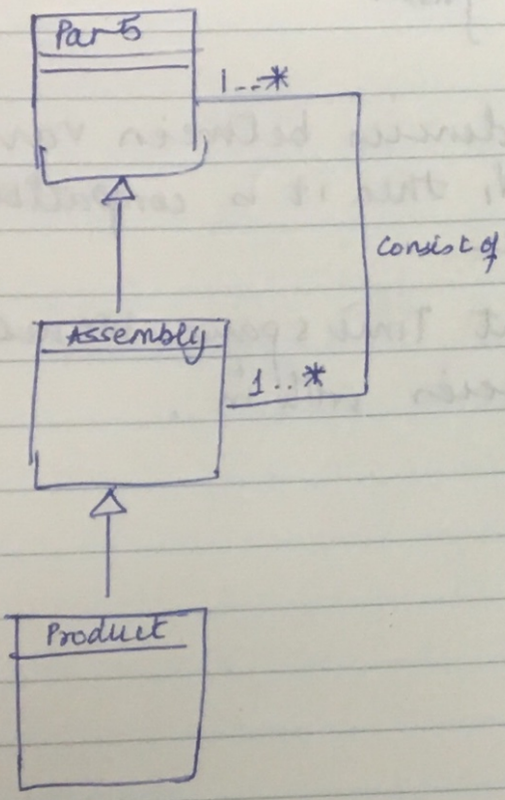
* is a relationship (loss of semantics)

Relational Database

Tables: Product, Assemblies, Parts
COP, Consist A, Consist Pa, Consist of Sub P

b) UML





Ans:→3 Document Management

* Part and document entity type – Why do we have n:m relationship

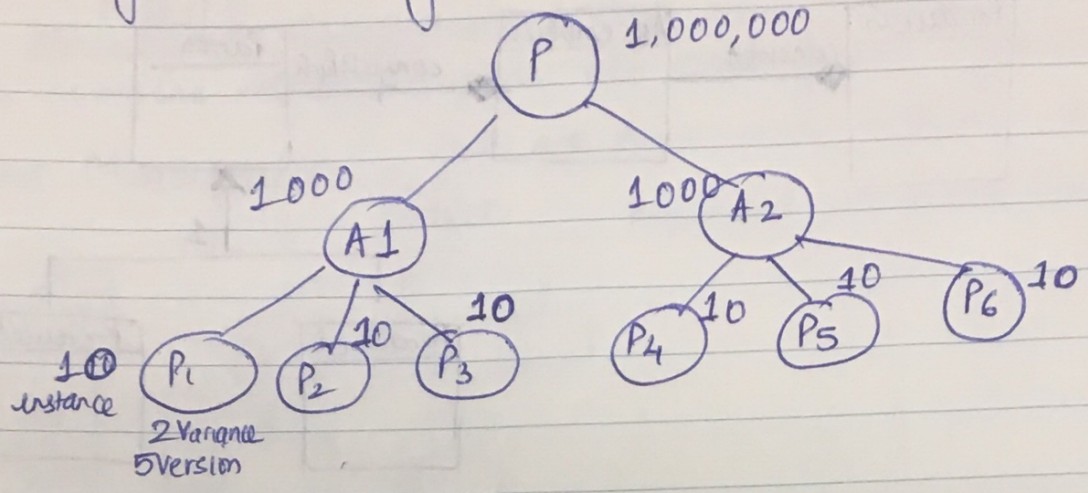
→ one part ~~can~~ there be included in many document showing aspects of that part in different point of view eg requirement phase, conceptual design, cost evaluation

→ Marketing → ~~two~~ two chea products for marketing analysis cheap, expensive. Distribution of different products using same logistic facilities.

- Board Layout / Schematic design.

1,000,000

Ans:→4 a) Configuration Management :



- Lot of products to choose from

b) Variance - describe dependencies between variance and tell if we use V_1 then it is compatible with a particular variance.

Version - built at different Time spans. We need to describe the dependencies with in.