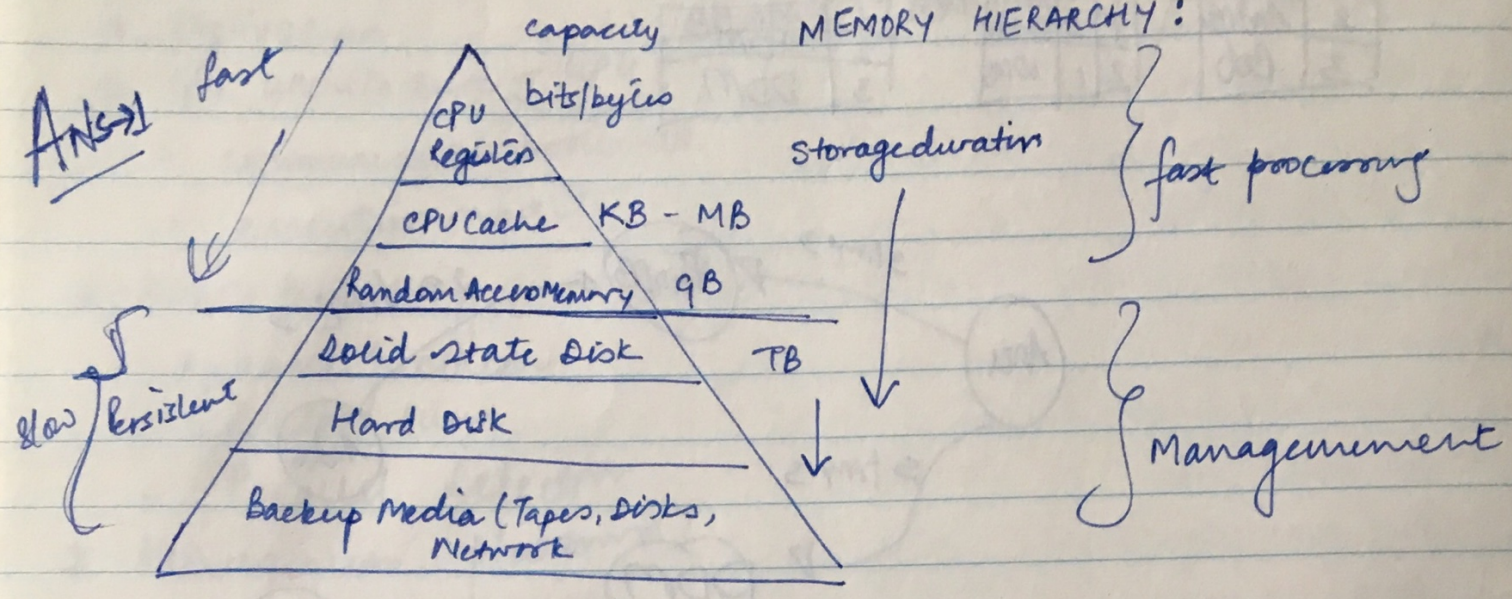
DMEA - Excercise 1

Ans→



Memory Hierarchy - Part of information technology that tells about data storag

* storage duration → least in Registers and cache, going down it is below me
data can be kept for long time:
RAM → as long as we keep the program open data is there. If computer is shut down. data will go away.
Solid State Disk/Hard Disk — comparable and can be kept for long time

* Levels focussed in lecture => focussing on SSD, HD, Backup Media we want data stored. We want to focus on persistent storage other half is transient. Suitable for long time storage

* Capacity :- Storage size
* classical back up Media → Disks - optical media, CD DVD
  ↳ back up copy of your Data
* Access Performance - CPU, CPU cache, RAM → very fast
* Price per unit → going down is cheaper
* granularity of access → What kind of data can I access at one time

eg:→ for CPU → bits/bytes
CPU cache → KB MB
and so on

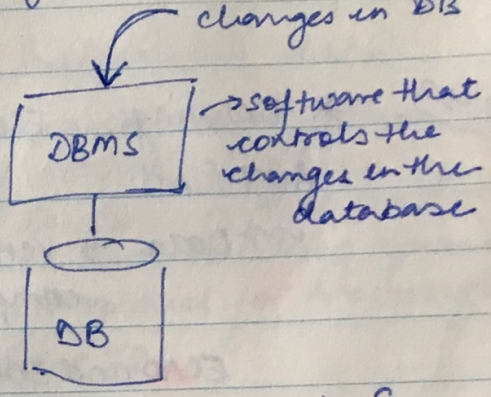
going down for persistente storage we transfer large data even if we are interested in one value we have to transfer the whole block of data

Ans:→2 File System Database Management System

→ Consistency :→ Data is same for all users in case of changes contradictions with in the database or real world by violating some rules. Incorrect states of data

(- -)

(++)
Several rules already defined in database
eg:→ Data Types (numeric, string)



- Integration:→ avoid redundancies by storing data again and again, every real world fact is stored only once and at one place.

→ Concurrent access → Several access at the same time

(-+)

- Its hard but can cause consistency scenarios.
- Temporary lock is applied on the file. Once work is complete then its available for access.

(++)

DBMS gets request and will kind of synchronised

→ DataExchange - Data exchange between new systems and legacy systems Exchange of data between development and manufacturing dept team

- Exchange is easy and we can easily define relationships between data
(++)

- I would not want to share my DB because of security reasons
- Can share relevant data
- lot of vulnerabilities when exchanging data
(-)

- Fine grained access - Granularity of data
(-) (++)
important for number of reasons to consistency

- License fees
In case of file we also have to pay something in order to achieve the same functionality (eg: → development)

Oracle - pay license fees we pay for support

Ans: → 3. * Drawings - Electronic Engineering (ECad)
EDA
Electronic design automate
AUTOCAD
↝ Data ⇒ version, Name, Geometrical data, Circuit, Electronic components, connectivis

ECAD and EDA have different ~~to~~ level of abstractions schematic level and PCB level.

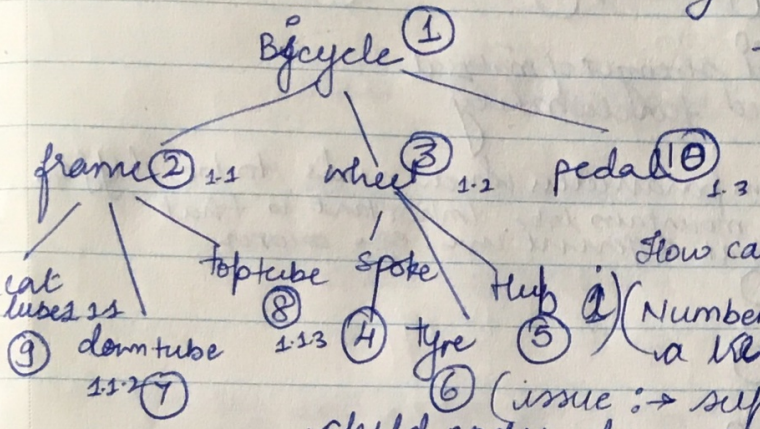
* Mechanical Engineering CAD systems

→ Data ⇒ Geometrical Models, Materials, Dimensions, Edges, Vertices, faces, shells, body

Exam - Persistence, Basic file system - concurrency, DBMS.

DMEA Excercise 2 Product Data / Requirements

Ans:→ 1 Data Management Solutions → eg:- Hierarchical structure
* Engineering data is complex unlike a flat data structure
* Consider the whole bicycle which consist of several parts



Typical Engineering application – objects consist of other objects

How can we encode hierarchical data

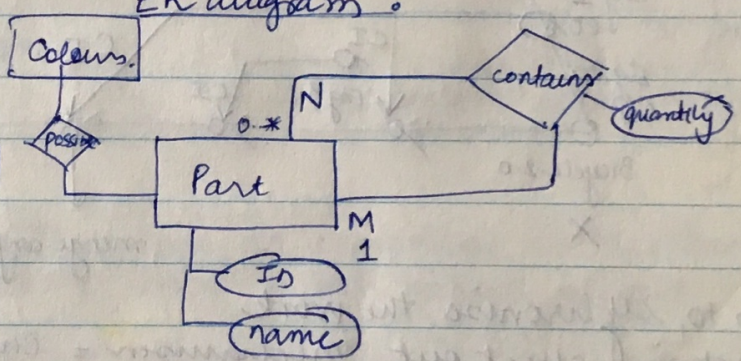
i) Numbering position representing positions in a level / Numbering scheme in a Tree
(issue :→ suppose I want to find parent and child nodes for part 1.2.1.3 then I need substring query)
example :→ chemical industry / Not optimal for processing

ii) Add one more table with part ID and subpart ID

Part ID	Contained Sub Part ID	Quantity
1	null	null
2	1	1
3	1	2
4	3	40
5	3	
6	3	
7	2	
8	2	
9	2	
10	1	

This is how we deal with hierarchical data in relational model

ER diagram :



- If a certain part has only one parent part (1-N) relationship we can just put it in one table
- But if we have a case like screws it is used in several contained part ID eg:

5	3
5	12

violates the 1st normal form atomicity
Redundant data

Disadvantages:
- Return all parts of a bicycle: We have to go through the whole hierarchy. complex data and more number of joins
  in SQL – recursion (until 1999)
- Less number of join

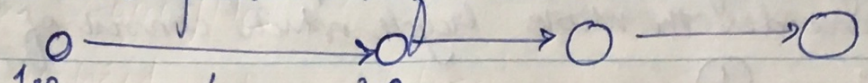
It is not only one object that we need to store data for, we need to store versions and variants

Exam ques

Ans:→2 Versions :→ Temporal sequence of development stage of an object. It is important to store previous versions. Tracking the old data is important eg :→ bike accident for legal reas

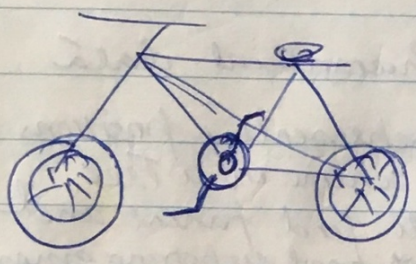
eg :→ Frame : Version 1.0

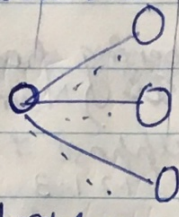
customer feedback : often broke



↳ improved strength of material
improved flexibility

Variants : certain parameters which needs to be differ
eg :→ sports like mountain bike. Important is that these state exist at different time. eg :→ colour

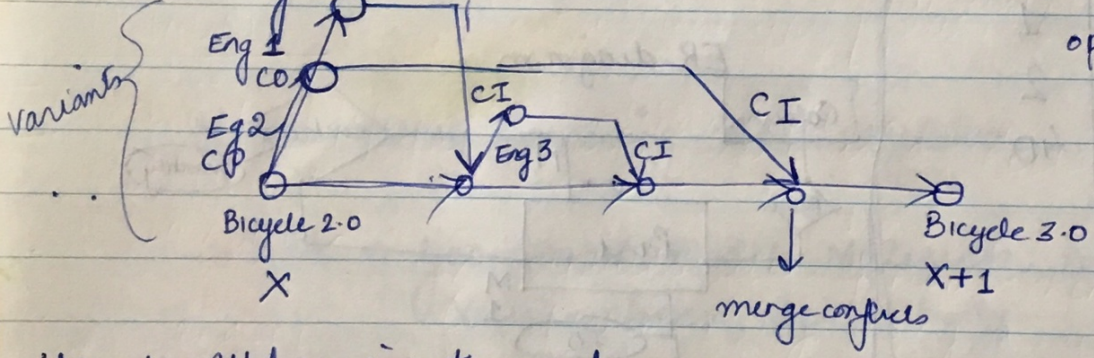




What's the problem
Everything is interdependent. Pedal system dia must match to the frame and also wheel dia

Versioning can be done to improve it

frame - Eng 1
Pedal system - Eng 3
Wheels - Eng 2



optimistic synchronisation
=> we hope their is no conflict and only known at the end.

How to synchronise the work
- Check in & check out mechanism = Check out frame into our own workspace. Generates their own versions and works on it
  ⇝ They get a copy of the it in their own workspace

* design conflicts at the moment when check in is done by Eng 2 because of conflicts in frame and wheels. We have to perform a merge operation to check constraints and whether the frame and wheel fit any more

Ans:→3 - Components which have different values
- Certain conduction/pins for each component

Java:- class component { string name;
.... set<component> connected;
}

class diode extends component { string colour;
}

class capacitor { }

class conduction {
}

every connect is modeled as object

How many sub classes do we need? unlimited
If I need to add more data (I have to rewrite the whole program

Relational → Same of relational we have to add new columns every time we need to add a new value

Text file:

<component>
<name> C1 </name>
<Type> capacitor </Type>
<value>
</component>

] XML files
text format
can define any kind of data
as structure is not fixed

Entity Attribute value Table → There are no semantics in the schema
We don't know what value means

Entity	Attribute	Value
Diode1	Color	Red.
student1	Name	Miller

How much semantics you can put in the database to make it easier to use.

Datamodel describes in an abstract way how data is being represented in a db of an information system.

Excercise 3 Data Management foundations

Exam

Ans:→ 1) STEP file - Just a text files, we can store engineering data in some file

* data - structured representation, something with meaningful info. Taking some real life facts and preserving in several ways. encoded information. | data about universe of discourse application field real world.

* Metadata → Data about data
eg:→ How the table is, what are the datatypes, where is it stored, constraints, relations ~~between data~~, structure of data — about schema
map of concepts

* Meta-Metadata
- data about metadata (does not describe the schema but the data model)
- data about schemas (eg:→ concept of inheritance in JAVA can be expressed by relational schema)
- Things we can do to describe the whole file
ISO - 10303 - 21
step standard — ↳ several packages general structure of file [file must be structured like header data section, separate lines etc)

METADATA

- Number of fileformats and have certain things common.
- Before actual data starts there is a certain description about the file

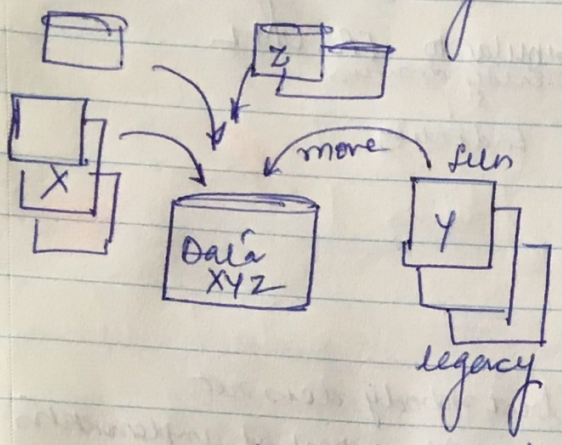
HEADER
;
Endsec
} contains description schema (what is the structure of schema) by doing a reference to a standardised electronic data. eg:→ AP210 → application protocol

- Creator Info: Author, Timestamp. What tool was used, versions, was it transformed
It describes the process, who did it, when and how

- #5 Product ('UM-PR-149-E-04', 'product', $ (#4290));
data (pk) — (data acts like a primary key in a table)
↳ data
Just a number of line → Metadata
what is the data refering to
↓ reference to line 4290
(we can look it up at AP210)

In the first exercise we discribed persistence (storing data for long) important concepts for engineering applications. We can use them to point out certain aspects, requirements of engineering data

Ans:→2) a) Data integration :→ Data can be acquired from several sources and storing in one standard format



* avoid redundancy (redundancy is not always bad) eg cars
  - bad because performance issues, inconsistent data
  - lot of efforts are required to avoid inconsistencies (different teams are working on ~~differ~~ same data
  - Waste resources for storage
  - Ideal solution is to store data only once

In a distributed scenario eg XYZ :- several mechanisms are there where ~~fact~~ fact of updates is kept

b) Data Security: Security trying to avoid that the data is being stolen, misused, corrupted.
- authentication - who uses it (username, password)
- authorization - grants privilages to access data

typical way to grant security. This person is allowed to read write and modify the data
eg: client data, prototype data (during test).

c) Data Quality :→ If data coming from sources can be used for certain data.

Aspects of data quality filters for use.
- accessible data
- consistency :- No contradiction within data itself
- Representational problem :→ eg 1,45 -- 1.45
- Completeness :- all columns must have values. No NULL values
- Accuracy : constant $\pi = 3.14...$ but in database we have just 2 decimal so value $= \pi = 3.14$
  precision

⇝ Precision could go wrong in data management
Calculations can go wrong (Imp to keep data very precise)
⇝ poor data quality → if data that we put into calculation and reconstruct data
⇝ inconsistent data across companies (decisions made in several departments ~~not~~ regarding data values that are diff)
⇝ completeness → testing on a data which is incomplete
simulation for certain test cases not possible

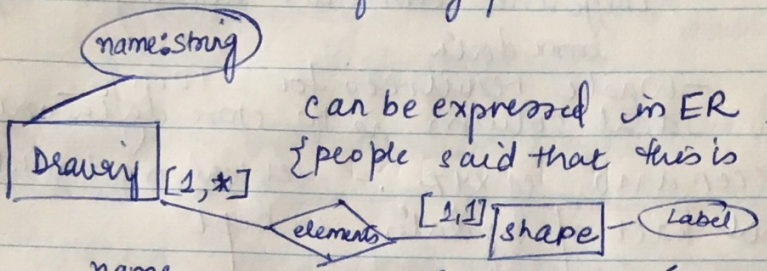
conceptual → high level
tables/connections
(name, relationships)
logical → columns in each table
(name, relationships, attributes)
PK FK

Ans:-3) (part of the step standards)
semantics - how we say things and what does the say

EXPRESS G - graphical representation similar to ER, UML
Entity drawing
;
End Entity

a) ▭ → entity (not predefine)

b) ▭| → (predefined types)
↳ data types Bool
Real - floating point number



can be expressed in ER but nobody does not
[people said that this is already a part of implementation]

c) ○ name → Modelling attributes / relationships

There is no clear distinction between a relationship and attribute eg:→ complex elements

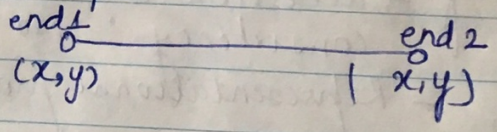
d) S [1,?] → cardinalities | Aggregation Data Types (Set and Bag unordered list and Array ordered) | unique

Express G diagram only has one part of cardinality eg:→ drawing can have many shapes. How many shapes are contained in a drawing? ? ?

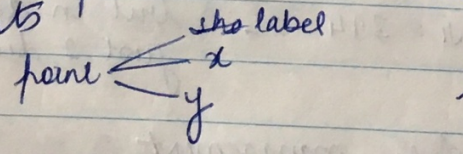
ER diagram → bidirectional.
Express G → one direction [if I need another direction I have to build relationship]

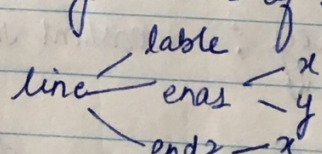
e) X, Y are attributes of each point (real value)

f) line → end1 and end2 are relationships



g) Inheritance / specialization :→ drawing consist of several lines or points





#513 = Point ('second', 5.3, 4.7)
#512 = Point ('first', 4.7, 5.1)
#477 = Line ('myline', #512, #513)
#700 = Drawing ('my drawing', #477, #512, #513)

#1 := POINT ('first', 5.3, ?
#2 := POINT ('secon', ?, 4.5)
#3 := LINE ('myli', #1, #2
#4 := Draw ('Draw1', #3)

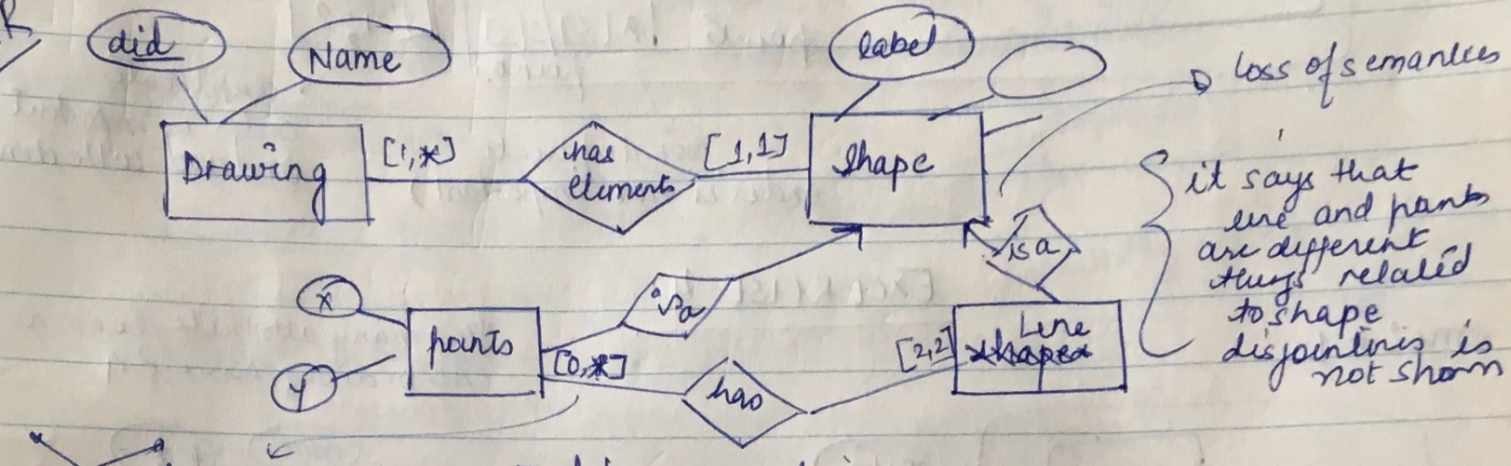
polygon → triangle

Concept of polymorphism:- same functions but different parameters or return types.

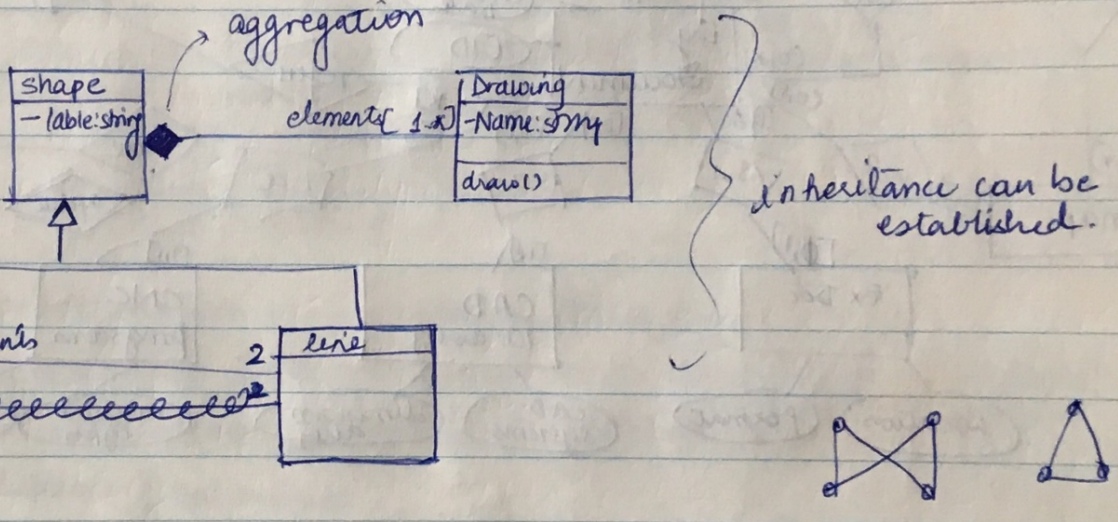
eg:→ for all sh in dr.elements
sh.draw();
end;

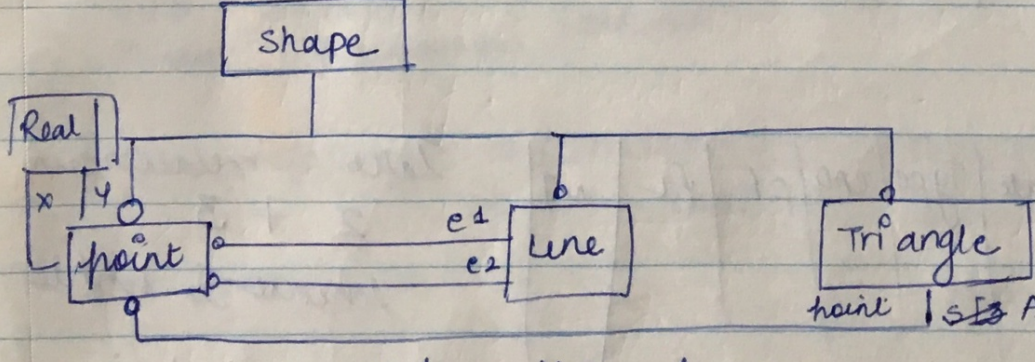
just because they belong to a common class I can treat them in a similar way

ER



point may be related to many lines or no lines



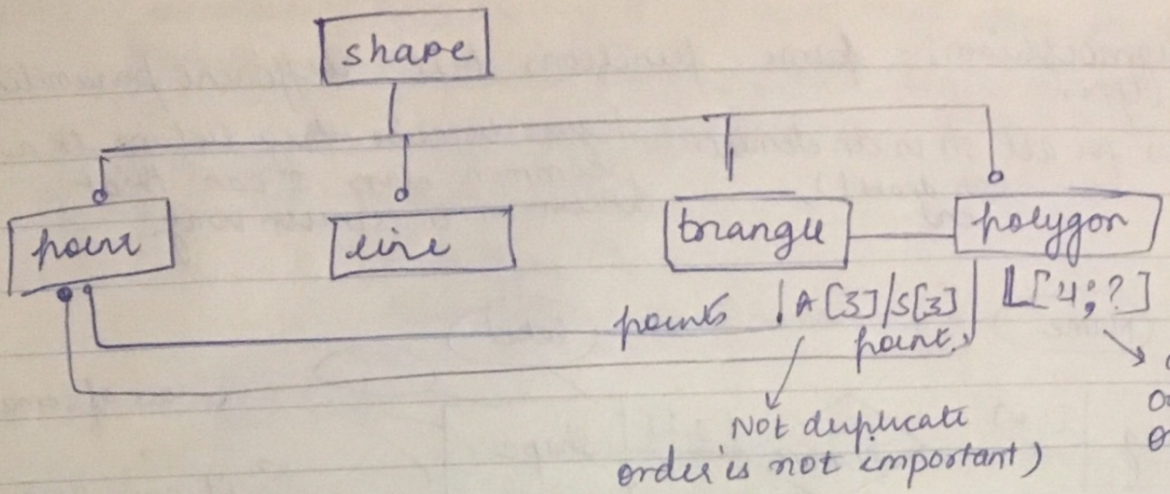


Triangle order does not matter for polygon I need an order

if I draw three lines
problem is we can have parallel lines
also end points and start points need to be same

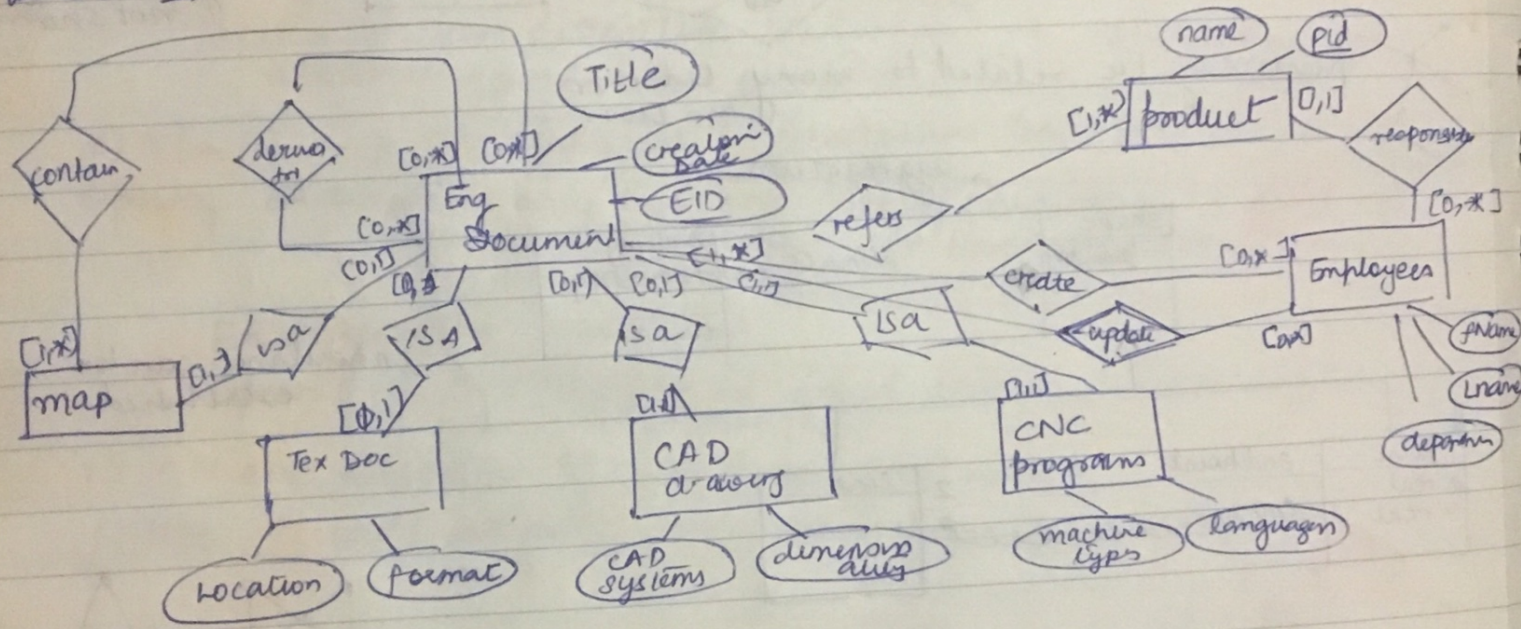
—This can be done but not in graphical representation is textual notation can be done by defining constraints

Conceptual → hint..



EXCERCISE 4

How many attribute does a CAD drawing have ? 4



Product 7 Entities + 4 n:m relation

Document

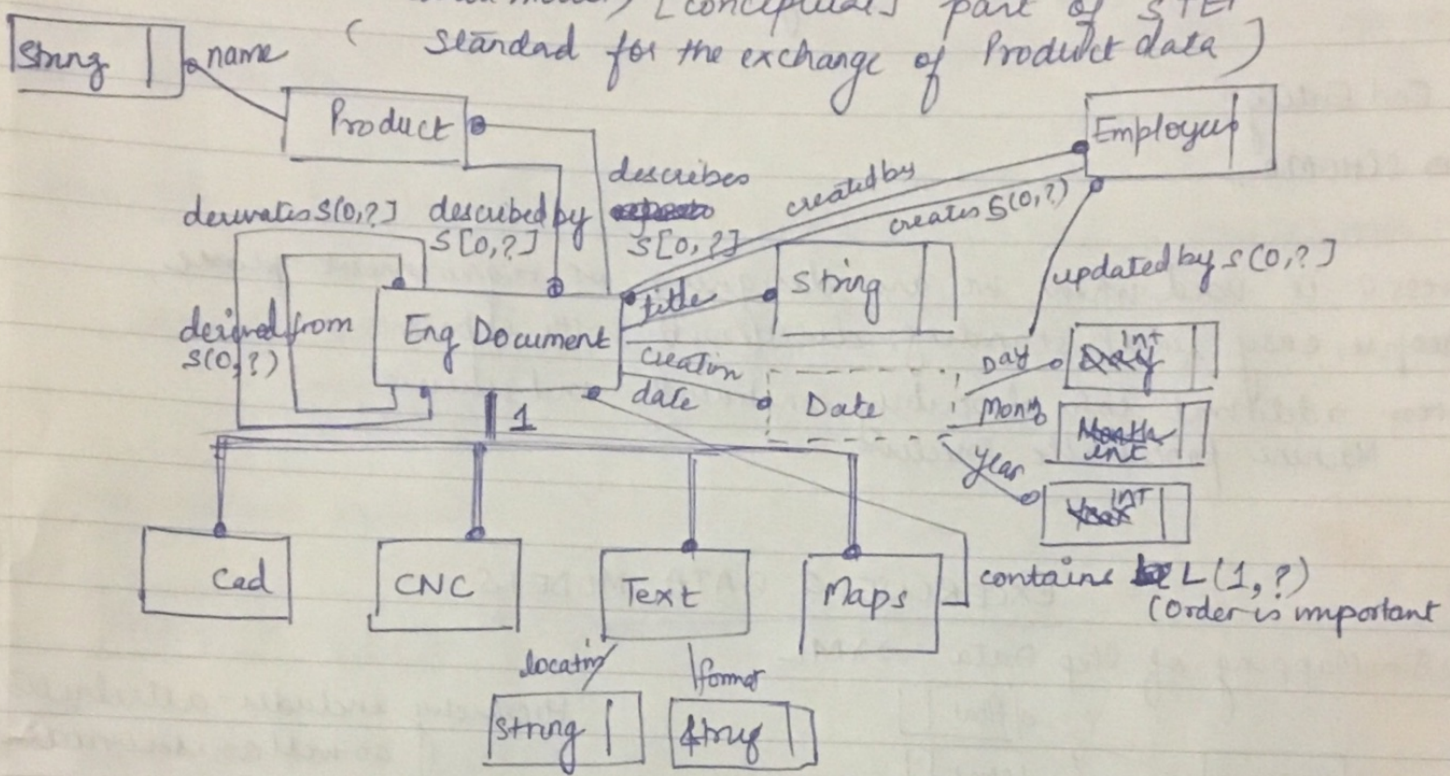
DID	TITle	create	Type	CAD	FF	FL	NT

Tables relationship
2 + 3
Schema is simple

- Schema is simpli
- performance is good

Discuss again how tables can be created?

Ans:→2 Express → Textual notation (logical), more comprehensive/detailed
Express G → Graphical model description of engineering data (data model) [conceptual] part of STEP
(standard for the exchange of Product data)



* Defined type = predefination of a complex data typ.
* attributes = relationship between entity and predefined or defined data typ
  relationship = relationship between two entities
* Set - without duplicate } not ordered
  Bag - with duplicates
  list - no fixed] ordered
  Array - fixed number

* contains relation- #437 = MAP(" .... (#50, #51)
  #451 = CAD(...)

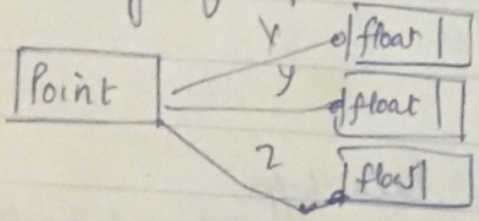
```
SchemaDocs;
  |
Entity Product;
 name: string;
 describedBy: set [1,?] of EngDocs;

End Entity;
END SCHEMA;
```

→ Express G is used when we are designing or requirement phase, people, easy understanding; discussions with experts
→ Express additional info describing constraints and schema
Machine processable structure

Ans:→1

EXCERCISE 5: DATA MODELS

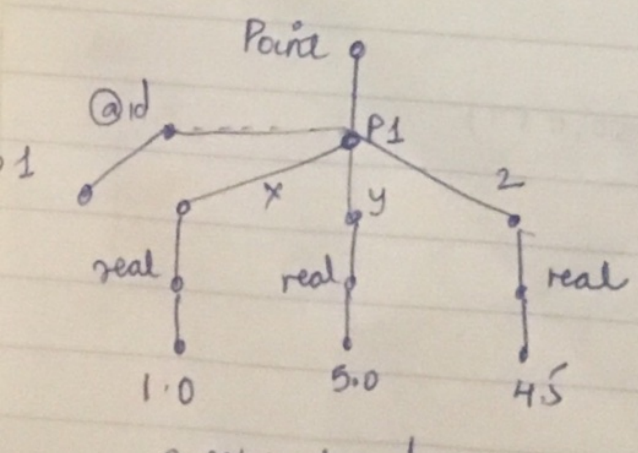
28- Mapping of step Data → XML



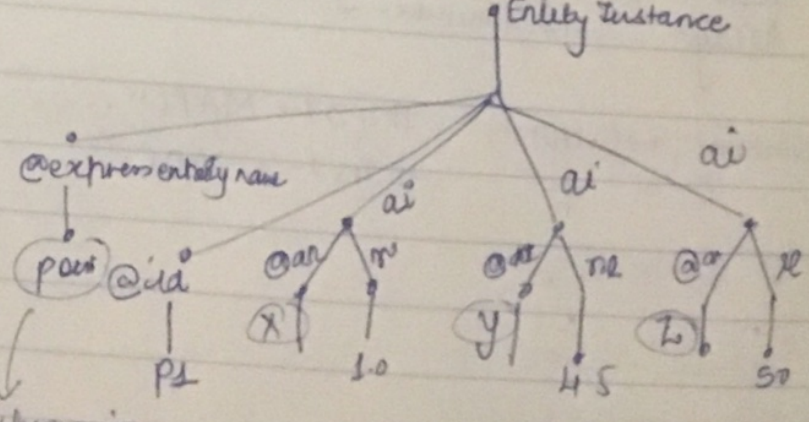
Hierarchy includes - attributes as well as subnodes

early binding (element type early binding)
late binding

→ XML is a hierarchical structure (taking complex info and breaking into more details



early binding



Schema.ipo

late binding
flexible
(hard to read, complex

→What is really specific to the application?

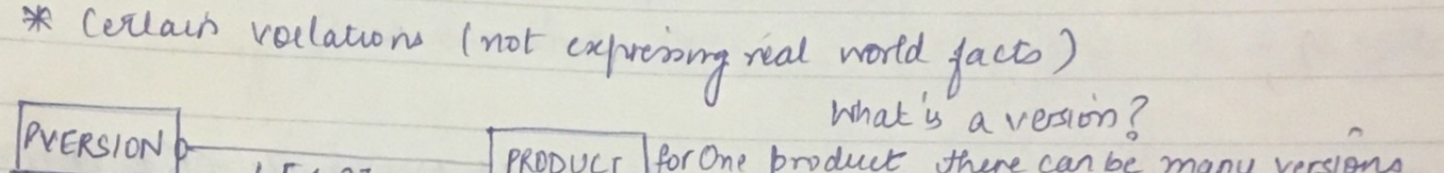
* early binding creates a schema from the step file
* late bindig - mapped all schema concepts to the data level
  on schema level I am completely independend → schema is independent on what I want to do with the data
→ ~~I am~~ only when the application is running the schema is associated with the data

ANS:→2 Product description } PRODUCT LIFECYCLE MANAGEMENT
Product schema

How this schema can we mapped in relational database

Logical schema —map→ conceptual schema

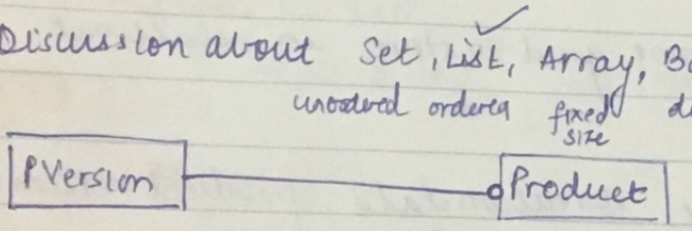
* Certain relations (not expressing real world facts)



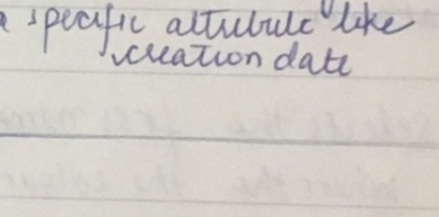
What's a version?

Discussion about Set, List, Array, Bag
unordered ordered fixed size duplicates

[ordered the version by creating a specific attribute like creation date]



* In ER we have straight forward rules



non optional → 1:N, N:1, 1:1 → foreign key merge
(lecturer for lecture)

optional → another table
eg:→ being married

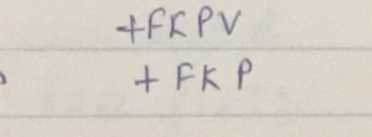
Product

Product Definitn +FK PV +FK DO

Work order +FK PV

Product Versin +FK P

Design Order



N:M
↳ Table

Relationship :-

* for every work order → each product versin
  1-1 / 1-N
  not optional.
* Product - Product versin - not optimal 1-N relationship

→ discuss about primary key (we need it to have relationship)
used for identification of an object
eg:→ pd_id
→ Not null primary

CREATE TABLE DESIGN OWNER (
did INT PRIMARYKEY,
FName varchar(20) NOTNULL,
LNAME varchar(20) NOTNULL,
ROLE VARCHAR(100) NOTNULL,
pdid int FOREIGN KEY REFERENCES
ProductDescriptrin (Pd-id)

);

- The data is pretty straight forward and flat so it's OK to store in relational database
- Hierarchical or tree structure is not good to store in relational database

selects the first name, last name, creation date, product version id where the the colour of the product is red.
* Join conditing