DMEA - Exercise 1

Memory Hierarchy:

- CPU:
  - Registers
  - CPU Cache: KB - MB
  - Random Access Memory: GB

- Persistent:
  - Solid State Disk (SSD)
  - Hard Disk

- Backup Media (Tapes, DVDs, Network)

---

**Memory Hierarchy** - Part of Information Technology that tells about data storage.

1. **Storage duration** → least in registers and cache, going down it data can be kept for longer:
   - RAM → as long as we keep the program open/data is there. If computer shuts down, data will be gone.
   - Solid State Disk/Hard Disk → non-volatile and can be kept for long time.

2. **Levels discussed in lecture** → focusing on SSD, HD, Backup Media.
   - We want data stored. We want to focus on persistent storage, other half is transient. Suitable for long-term storage.

3. **Capacity** → storage size
   - Classical Backup Media → disks, optical media, CD/DVD → backup copy of your data

4. **Access Performance** → CPU, CPU cache, RAM → very fast
5. **Price per unit** → going down is cheaper
6. **Granularity of access** → what kind of data can I access at one time?

- e.g., for CPU: bits/bytes
  - CPU cache → KB - MB
  - and so on

---

Going down for persistent storage we transfer large data.

Even if we are interested in one value, we have to transfer the whole block of data.
File System

- Consistency: Data is same for all users in case of changes. Contradictions within the database or real world by violating some rules, Incorrect states of data.

++
- Several rules already defined in database
  - E.g., Data Types (numeric, string)

---
- Software that controls the changes in the database

---
- Integration: Avoid redundancies by storing data again and again, every single record is stored only once and at one place.

++
- Concurrent access: Several access at the same time

(-)
- It's hard but can cause consistency scenarios.
- Temporary lock is applied on the file. Once work is complete then it is available for access.

++
- Data Exchange: Data exchange between new systems and legacy systems
  - Exchange of data between development and manufacturing.
  - 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 - consistency
- License fees
  Increase of file we also have to pay consultancy in order to achieve
  Hardware functionality (e.g. development)
  Oracle - pay license fees
  We pay for support

AutoCAD

- Mechanical Engineering CAD systems

  Data = geometrical models, materials, dimensions, edges, vertices, faces, shells, body

Exam - Persistence, basic file system - concurrency, dooms
DMEA Exercise 2 Product Data Requirements

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

```
Bicycle
| Frame 2 | Wheel 3 | Pedal 10 |
```

Typical Engineering application - objects consist of other objects

How can we encode hierarchical data?

Numbering format representing positions in a tree. Numbering scheme end a tree.
Issue: Suppose I want to find parent and child nodes for part 2, 1, 3 then I need substring query.

b) Add one more table with part 20 and Subpart 20

<table>
<thead>
<tr>
<th>Part ID</th>
<th>Sub Part ID</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

This is how we deal with hierarchical data in relational model.

ER Diagram:

- If a certain part has only one part (1:N) relationship we can just put it in one table.
- But if we have a mass like screws it be used in several contained part 20 eg: 5/3 violates the 1st normal form atomicity, Redundant data.

Advantage:
- 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 joins.
It is not only one object that we need to store data for, we need

1. Store versions and variants

Ans: → 2 \( \text{Versions} \rightarrow \text{Temporal sequence of development stage of an object. It is important to store previous versions, removing the old data as important.} \) eg: → bike incident customer feedback: often broke

\[ \text{Frame: Version 1.0} \rightarrow 0 \rightarrow 0 \rightarrow 0 \]

- Improved strength of material
- Improved performance

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

What's the problem:
Everything is interdepend. Pedal system dia must match all the frame and also wheel dia.

Versioning can be done to improve it.

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

Variants:

CI

Frame

Eng 1
Eng 2
Eng 3

CI

Bicycle 2.0

X

How to synchronize the work:

- Check in / Check out mechanism = Check our frame into our own workspace, generates there our versions and works on it
- They get a copy of it in their own workspace

* Design conflicts as the moment when check in is done by Eng 2 because of conflicts in frame and wheels, we have to perform a merge now operation to check constraints and whether the frame and wheels fit any more.
Ans: +3 - Components which have different values
- Certain conduction/pins for each component

Java:
```java
class component {
    String name;
    ... set <component> connected;
}
```

```java
class diode extends component {
    String colour;
}
class capacitor {
}
class conduction
}
```

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

Relational:
Some of relational we have to add new columns every time we need to add a new value

```xml
text file:
<component>
    <name> cl</name>
    <type> capacitor <type>
    <value> Red</value>
</component>
```

```xml
xml file
text format
<component>
    <name>
        <type> capacitor <type>
    <value> Red</value>
</component>
```

Entity Attribute Value Table
- There are no semantics in the schema

Entity | Attribute | Value  
-------|-----------|--------
Diode  | Color     | Red    
Student| Name      | Muller 

How much semantics you can put in the database to make it faster to use.
Databases describe in an abstract way how data is being represented in a db of an information system.

Exercise 3: Data Management foundations

Ans: 1) STEP file - that is text files, we can store engineering data in some file.

* Data - structured representation, something with meaningful info, taking some real-life facts and processing in several ways.

* Metadata - Data about data.

  * Metadata - data about metadata (does not describe the schema but the data model).
  * Things needed to describe the file that be explored by relational schema.

ISO-10303-21

STEP standard

General structure of file (data section, separate lines, etc.)

METADATA

- Number of file formats, and have certain things common.
- Before actual data starts there is a certain description about the file.
  
  HEADER
  
  contains description of schema (what is the structure of schema) by doing a reference to a standard electronic data
  
  Creator Info: Author, Timestamp, What tool was used, version, what is transformer.

It describes the process, who did it, when and how.

# 5 Product ('UM-PR-149-E-04', 'product', & (#4290)); primary key is italic.

Metadata

what is the data referring to (we can look it up at #P2400)
In the first exercise we discussed persistence (storing data for long) important concepts for engineering applications. We can use them to point out certain aspects requirements of engineers.

Ans: (2) 1) Data integration: data can be acquired from several sources and stored in a standard format.
   * Avoid redundancy (redundancy is not always bad) eg cars
     - Bad because performance is lower, inconsistent data
     - Lot of effort are required to avoid inconsistencies
     - Different teams are working on different data

   Legacy
   - Waste required for storage
   - Ideal solution is to store data only once.

   In a distributed scenario, eg XYZ servers, mechanisms are there where fast track of updates is kept.

b) Data Security: Security during to avoid that the data is being stolen, modified, corrupted.
   - Authentication - who users it (discretionary, password)
   - Authorization - grants privileges to access data
   - Typically 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: factors for use.
   - Accessibility of data
   - Consistency - No contradiction amongst data itself.
   - Representational accuracy: eg 1.45 -- 1.45
   - Completeness: all columns contain valid values, no NULL values.
   - Accuracy: constant \( \pi = 3.14 \) - But in database we have just 2 decimal values

   \[ \pi = 3.14 \]

Precision could go wrong in data management.

Calculations can go wrong time to keep data very precise

Poor data quality -> y claim that we put into calculations some inexplicable data.

Inconsistent data across companies (demographics made in several departments) regard data values that one differ.

Completeness testing on data which is incomplete.

Simulation for certain test cases not possible.
Semantics - how we say things and what does it say
EXPRESS G - graphical representation similar to ER, UML

a) entity (not predefined)

b) data types below
Real-floating point number

name: str

d) cardinalities

Express diagram only has one part of cardinality eg: drawing can have many shapes. How many shapes are contained in a drawing? [1, ∞]

ER diagram -> bidirectional
Express $\textit{G}$ -> one direction $\textit{H}$. If I need another direction I have to build relationship

e) $x, y$ are attributes of each point (real value)
f) line $\rightarrow$ end1 and end2 are relationships

g) Inheritance / specialization $\rightarrow$ drawing consist of several lines

```
#513 = Point ('second', 5.3, 4.7)
#512 = Point ('first', 4.7, 5.1)
#417 = Line ('myline', #512, #513)
#700 = Drawing ('mydrawing', #417, #512, #513)
```
Concept of polymorphism: the same function but different parameters or return types. For example:

```java
for all sh in dr.elements
    sh.draw();
end;
```

Just because they belong to a common class doesn't mean they behave in the same way.

Loss of meaning:

- It says that line and point are different, though related to shape. Disjointing is not shown.
- A point may be related to many lines or no lines.

Aggregation:

Inheritance can be established.

Triangle order doesn't matter for polygon. I need a defined order.

If I draw three lines, we can have parallel lines. Also, end points and start points need to be same.

This can be done but not in graphical representation. In textual notation, it can be done by defining constraints.
EXERCISE 4

How many attributes does a CAD drawing have? 4

Product "Tindia" + 4 n:m relation

Table relationship
2 + 3

Schema is simple
- Schema implies
  - Performance good
Discuss again how tables can be created?

Ans: 2 Express \rightarrow Textual notation (logical), more comprehensive\& detailed
Express \rightarrow Graphical model description of engineering data
(Data model) \{conceputal\} part of STEP
(Standard for the exchange of product data)

* Defined type = predefinition of a complex data type
* Attribute = relationship between entity and pre-defined or defined
  relationship = relationship between two entities
* Set = without duplicates \& not ordered
  Bag = with duplicates
  List = not fixed ordered
  Array = fixed number, ordered
* Contains relation: $431 = MAP(\ldots , \#50, \#31)$
  $451 = CAD(\ldots )$
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 mechanism, taking complex info and breaking into more details.
What is really specific to the application?
* early binding creates a schema from the step file
* late binding = mapped all schema concepts to data level
  on schema level I am completely independent \( \Rightarrow \) schema is independent
  on what I want to do with the data
  \( \Rightarrow \) only when the application is running the schema is associated with the data

ANS: 2

Product Description \( \Rightarrow \) Product Lifecycle Management

How this schema can be mapped in relational database

* Logical to conceptual schema

Certain violations (not expressing real-world facts)

VERSION \( \rightarrow \) PRODUCT

What is a version?
For one product, there can be many versions
Can be true if we are keeping only one current version

Discussion about Set, List, Array, Bag
-Incorporated order
-Duplicates
-Order the version by creating a specific attribute like version or creation date

**In ER we have straight forward rules**

Product

- Product Definition + FK PV + FK DO
- Work Order + FK PV
- Product Version + FK P
- Design Order + S[1..2]

Table A \( \leftrightarrow \) Table B

Relationships:
- 1:N, N:1, 1:1
- Optional
- Not optional
- For every work order + each product
- 1-1, 1-N

\( A \rightarrow_{S[0..2]} B \)

N:M
discuss about primary key (we need it to have relationship) used for identification of an object

e.g.: pd-id

NOT NULL primary

CREATE TABLE DESIGN OWNER (
  id INT PRIMARYKEY,
  FNAME VARCHAR(20) NOTNULL,
  LNAME VARCHAR(20) NOTNULL,
  ROLE VARCHAR(100) NOTNULL,
  pdid INT FOREIGN KEY REFERENCES ProductDescription (Pd-id)
);

The data is pretty straightforward 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 condition