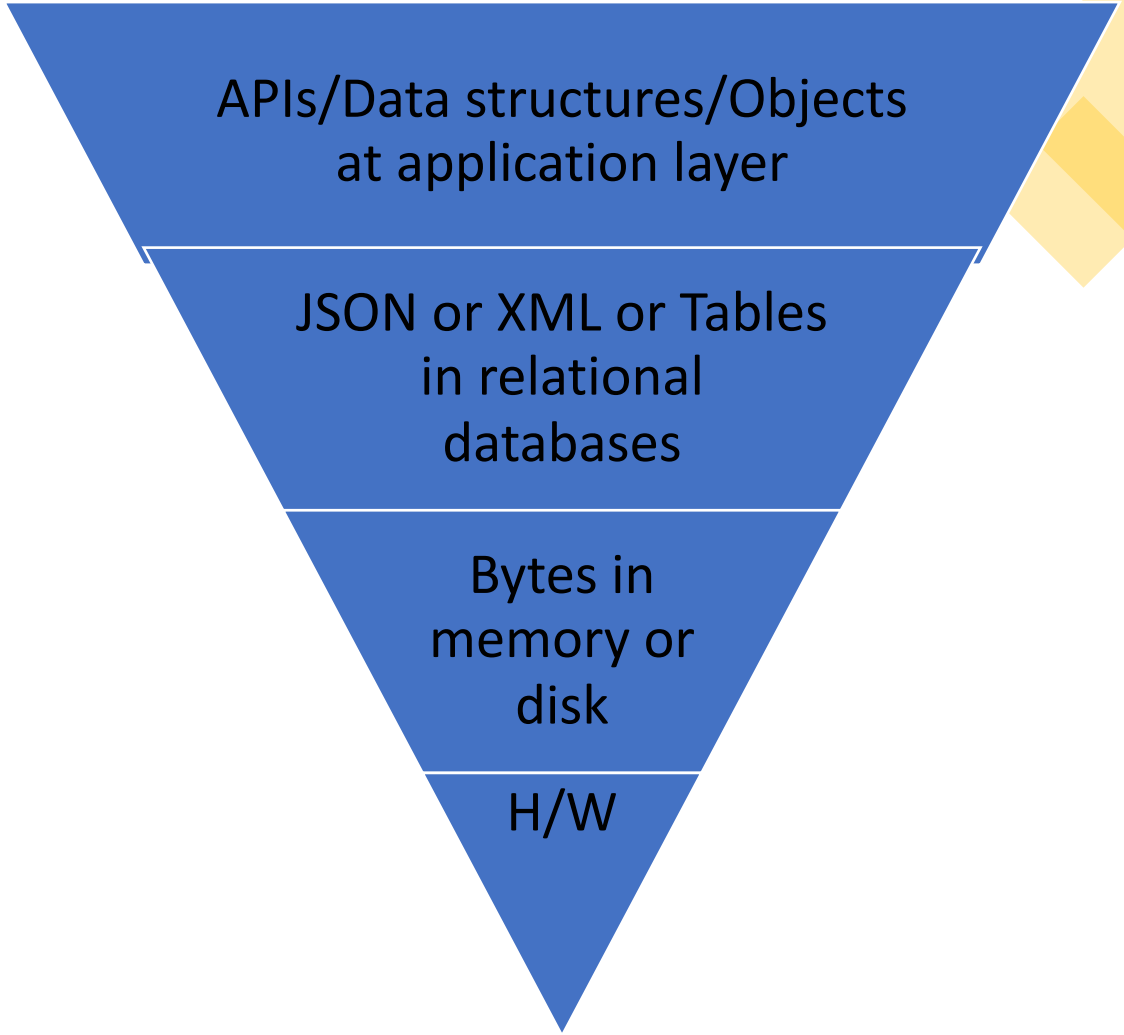
Chapter 2 : Data Models and Query Languages

What is a data model?

- A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities.
- It emphasizes on what data is needed and how it should be organized instead of what operations need to be performed on the data.

How data models and abstraction are related?

- Most applications are built by layering one data model on top of another.
- Look at the real world in which there are people, organizations etc. modeled in terms of objects or data structures or APIs that manipulate these structures.
- How do you store those data structures in DB? – in terms of data model such as JSON or XML or tables in relational database.
- These data structures are represented in terms of bytes in memory on disk or on a network. This might allow data to be queried, processed.
- Then on hardware layer they might be represented in terms of electric currents, magnetic fields etc.



Relational Models

Data models used for data storage and querying

NoSQL aka Not only SQL models

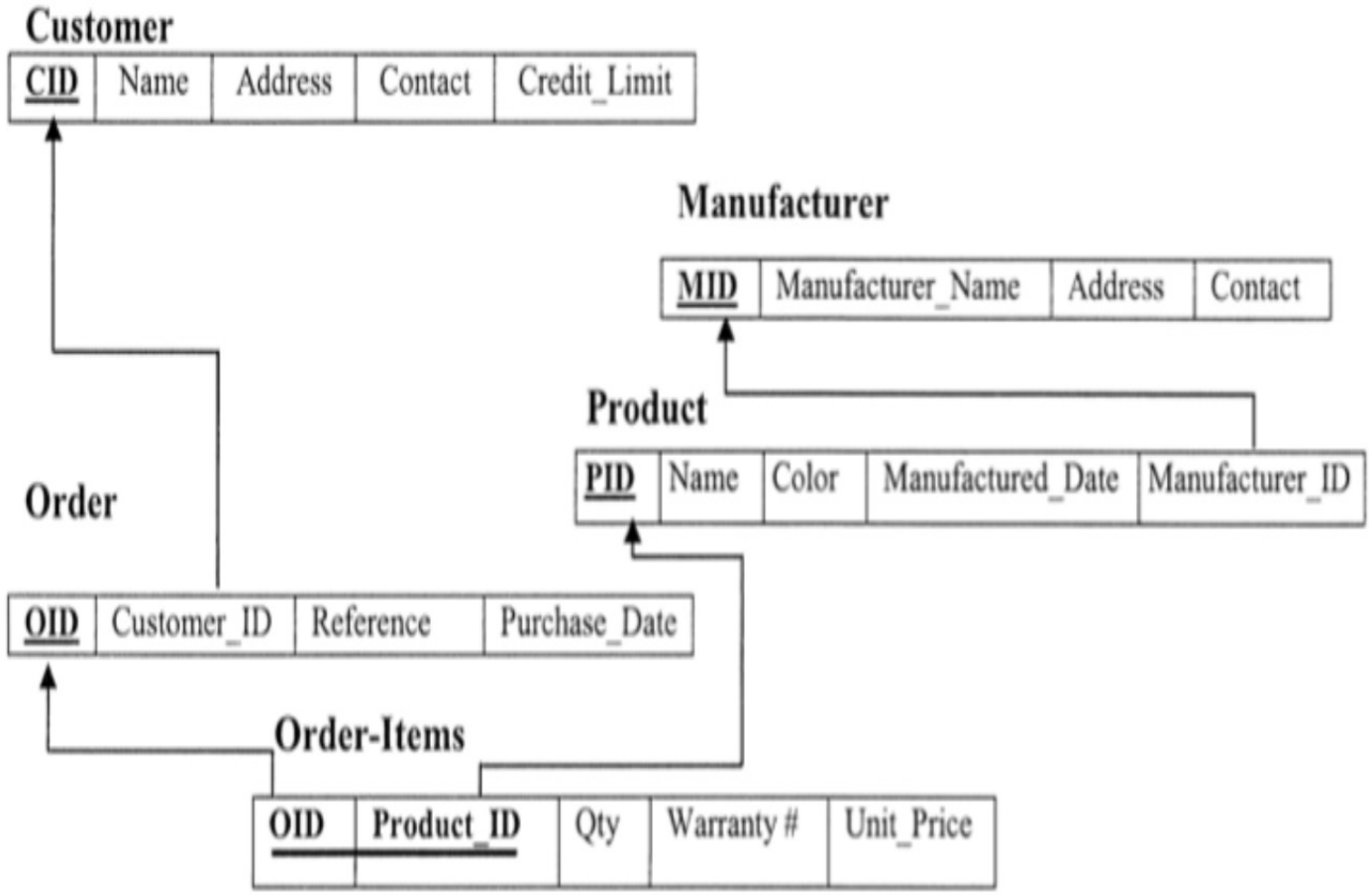


Image Source: stackoverflow

Relational Model

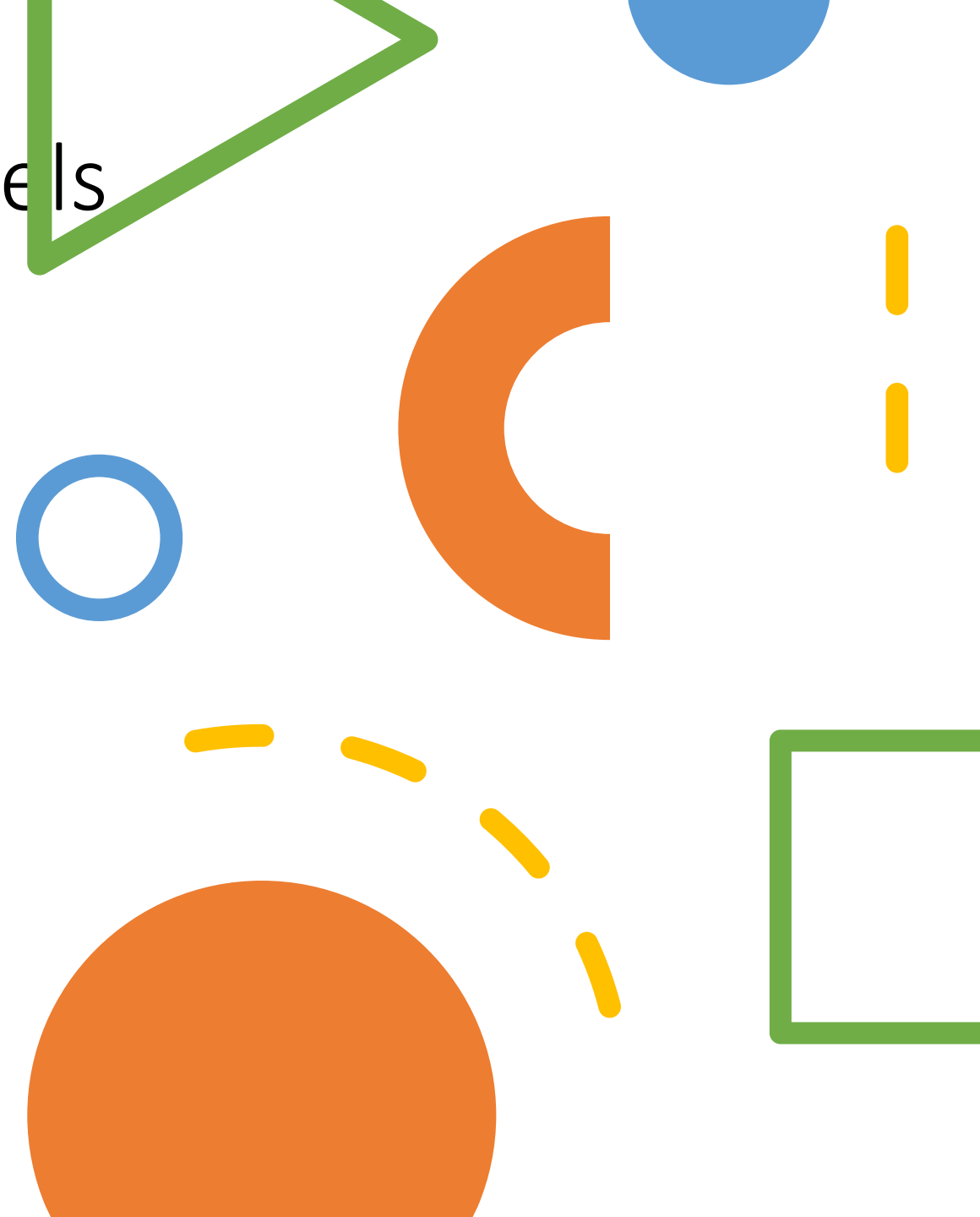
- Data is organized into relations called tables.
- Each relation is an unordered collection of tuples(rows)
- Schema on write – schema is explicit and database ensures all written data conforms to it. – static compile time checking

How to find all the items ordered by a customer?

```
Select
p.Name,oi.quantity
from Customer c
JOIN Order o on c.CID = o.Customer_ID
JOIN Order_items oi on oi.OID = o.OID
JOIN Product p on p.PID=oi.Product_ID;
```

More into Relational models

- Impedance mismatch : The disconnect between objects in application code and database models of tables, rows and columns.
- Lots of joins
- Query optimizer decides which part of query to execute in which order and which indexes to use.



NoSQL aka Not Only SQL

- NoSQL databases allows “unstructured data.”
- Follows schema-on-read(structure of data is implicit and interpreted when data is read) – dynamic type checking
- NoSQL databases can be document based, graph databases, key-value pairs, or wide-column stores.

Document based example

Customers

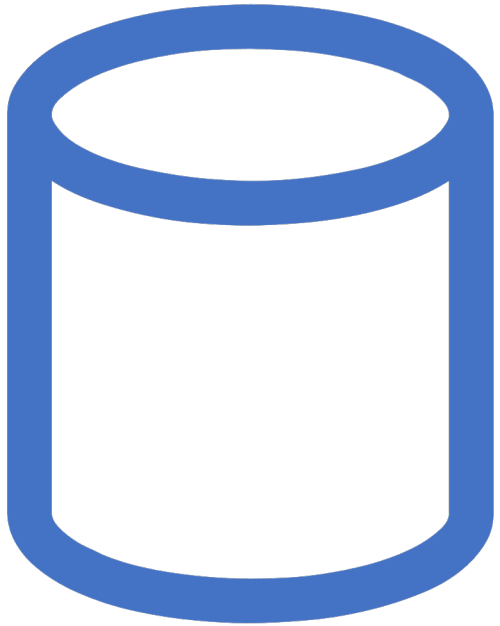
```
{
  "cid": "14545",
  "name": "john",
  "address": "F/56 West avenue",
  "contact": "+2898212(2189)"
}
```

Orders

```
{
  "order": {
    "id": "221782781",
    "cid":"14545",
    "items": [
      {
        "name": "TV",
        "color": "black",
        "quantity": "1",
        "price": "1000$",
        "manufactured_date": "26 Jul 2020"
      },
      {
        "name": "Watch",
        "color": "Red",
        "quantity": "2",
        "price": "500$",
        "manufactured_date": "31 Jul 2020"
      }
    ]
  }
}
```

More into NoSQL

- Better storage locality
- JOINS are weak, emulate joins in application code
- Cannot refer to nested items directly.



Which is better then?

- If data has one to many relationships where a tree can be loaded at once, then document structure is better.
- Relational technique of shredding- creating a document like structure into multiple tables can lead to cumbersome schemas.
- If there comes a situation where a customer first name and last name is currently being stored in one column and we have to separate them.

Query Languages for Data

Declarative vs Imperative

```
Select * from animals where family='sharks';
```

- Specify the pattern of the data – what conditions you want and what transformation – like sorted, aggregated data, but not how to achieve that goal.
- It can often have parallel execute.
- Upto database optimizer to decide which indexes.

```
function getSharks() {
    var sharks = [];
    for(var i=0;i<animals.length;i++){
        if(animals[i].family === "Sharks"){
            sharks.push(animals[i]);
        }
    }
    return sharks;
}
```

- Tells the computer to perform certain operation in a certain order.
- Hard to parallelize because it specifies instructions to be executed in a particular order.

Declarative queries on the Web

```
<ul>
      <li class = “selected”> <p> Sharks</p> </li>
      <li> Whales</li>
      <li> Fish </li
</ul>

li.selected > p {
      background-color:blue;
}
```

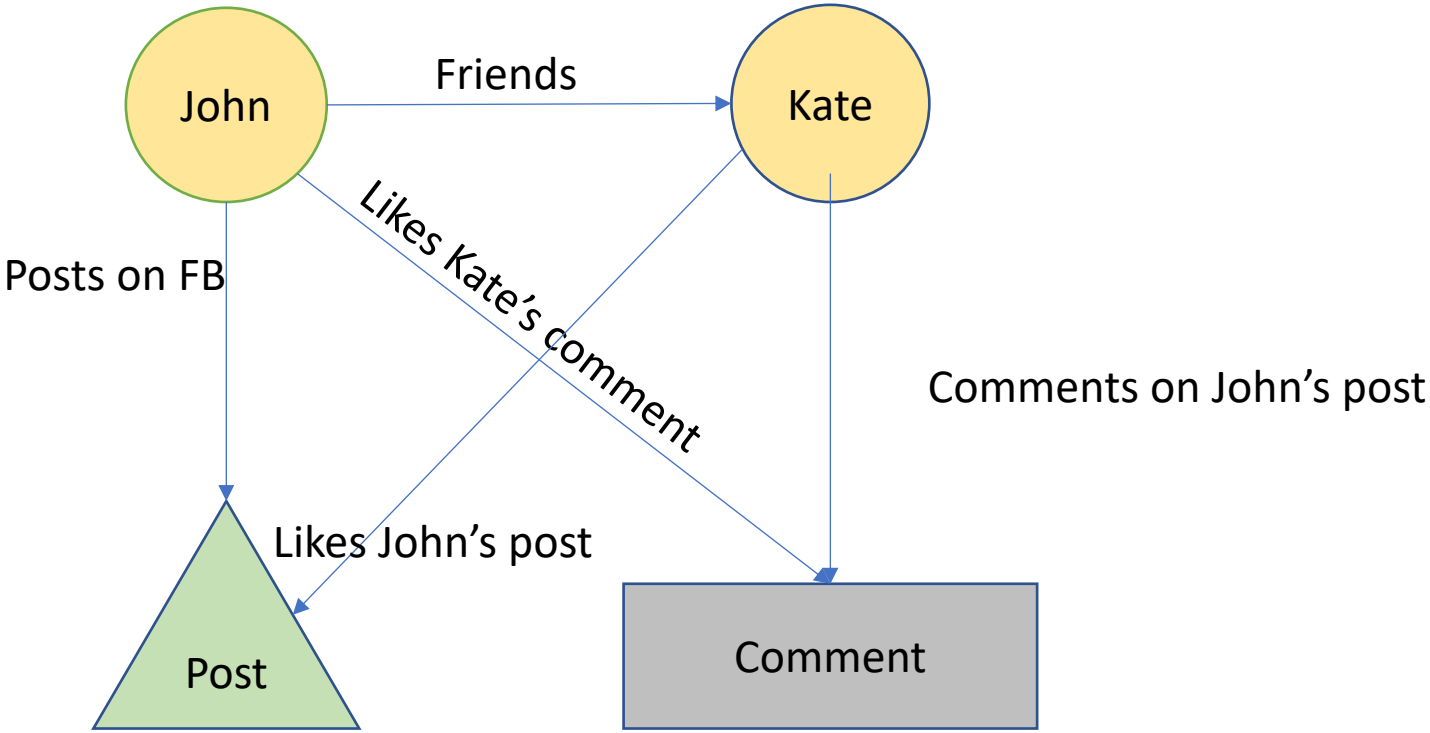
Imperative example

```
var liElements = document.getElementByTagName(“li”);
for(var i=0;i<liElements.length;i++){
        if(liElements[i].className == “selected”{
                var children = liElements[i].childNodes;
                        for(int j=0;j<children.length;j++){
                                if(children[j].nodeType == ELEMENT_NODE &&
                                   children[j].tagName == “P”){
                                        child.setAttribute(“style”,”background-color:blue”);
                                }
                        }
        }
}
```

Graph like data models

- Vertices (nodes or entities)
- Edges (relationships or arcs)
- Examples:
  - Social graphs
  - Web graphs
  - Roads or rail networks

Social graph example



Summary

- Relational Models
- Document based Models
- Graph models
- Declarative languages
- Imperative languages

Thank You!