Database Designs

Section 1: Introduction to Databases

1.1 Why Do We Need A Database?

When we hear the term database we can think of it as a repository of data from which we can extract information very quickly. Data in the world is ubiquitous i.e. details about everything is stored somewhere especially in the modern connected world. There are a lot of data in the world and data is being constantly generated when we are online which is being stored somewhere. The question we can therefore ask is why do we need a database to store data? Or alternatively, what is it about a database that makes it a good choice for storing data?

As an example, lets say we are planning to setup a large e-commerce site like Amazon or Ebay. There is a lot of technology behind running a good e-commerce system i.e. there are a huge amount of interconnected systems that talk to each other rather than a sing monolithic software. For a e-commerce website this can be a website or a mobile app, a search system, recommendation system, cart, checkout & payment system, order management system, supply chain & logistics system, a catalogue of products and finally a seller system to onboard new merchants.

This provides an idea of of all the systems required to work together for a good working e-commerce site and all the data that these systems would need to hold. The Order Management System (OMS) is the core piece of system the user would interact with each time they purchase something on an e-commerce platform. The user does not need to know an Order Management System exists but it does exist in the background and it keeps track of all the orders that is placed on that website i.e. it keeps track of orders and inventory. With this information it can be used to power personalised searches, recommendations and a whole bunch of other things which can make the user experience more personal and tailored.

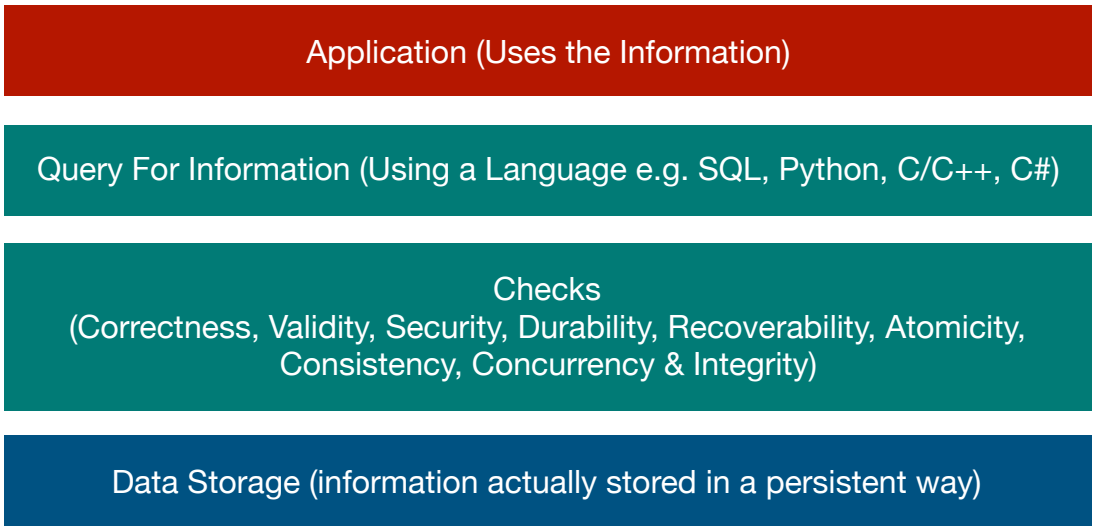
For the e-commerce Order Management System example it must record and store information related to the product such as description, specifications, units available, etc. as well as information about the customer such as name, address and contact details. All of this information needs to be stored and maintained by the OMS. All of this information related to the customer and the product together forms what we would call an order. The order brings together all of the information such as the customer name, product name, quantity, shipping address and a timestamp.

We would want the OMS to work correctly. What we mean by working correctly is that the OMS should allow an order to be placed only when there is available inventory. Each time a product is purchased we would want to reduce the available items when the order was successfully placed. This would also mean to ensure the reduction of inventory is equal to the number of orders placed for that product. There are also other data that needs to be correct such as the status for the order i.e. has it been packed, shipped or returned. We would also want to make sure the correct and valid product is in the order and it is

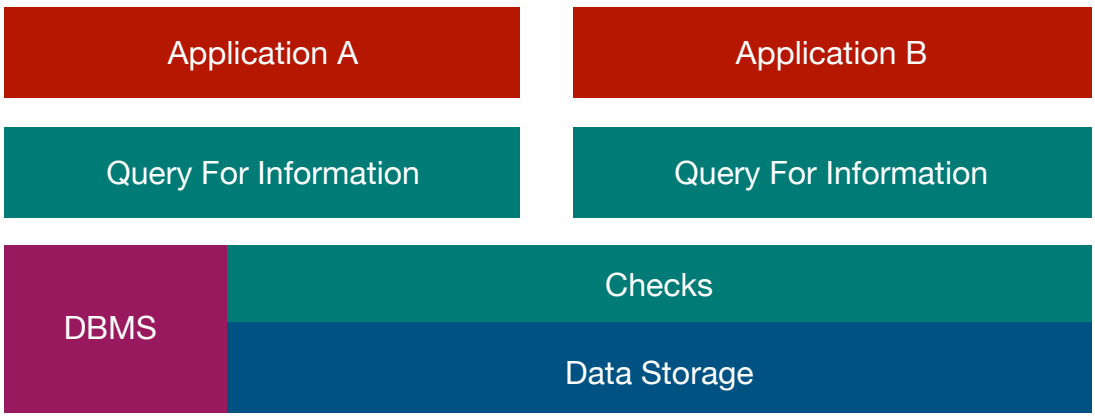
assigned to the correct and valid customer. This is not an exhaustive list of what is required for a OMS system.

The main take away we can see from the above example is that we require a system to implement that does not lose information ever and is backed up in the case of catastrophic failures and can be recovered from failures in order to return back to a sane state. The data must also be secure and not exposed for data protection. Finally, the data must be accessible to multiple authorised people to modify the data at the same time and the data to still make sense to everyone who uses it.

This is a formidable list of requirements for a OMS system. Below is a high level block diagram of how a system would look like:



Suppose we want to create a completely different application but uses the same data i.e. two applications are two different entity but uses the exactly same information. The checks are closely tied with the data itself and should not be a separate system or a separate block and should not be replicated in different systems. The database layer and all the checks should be included within a system and this is known as a Database Management System (DBMS).



DBMS is not purely the storage of data but also incorporates all of the system checks such as security checks, integration checks, consistency checks, etc. Therefore, we can define a DBMS as the following:

A database is a collection of structured data, an abstract representation of some domain.

A Database Manage System is a complex piece of software that sits in front of a collection of data and mediate access to the data guaranteeing many properties of the data and how it is accessed.

The “abstract representation of some domain” simply means a database holds some information in a structure/form that represents something in the real world. While “guaranteeing many properties” means the DBMS carrying out the various checks as seen in the first block diagram above.

To conclude, the above demonstrates why we need a database or a Database Management System to store data for our applications.