Enterprisewide Accounting Information Systems -- I

The course focuses on the business application of information technology, with business coming first.

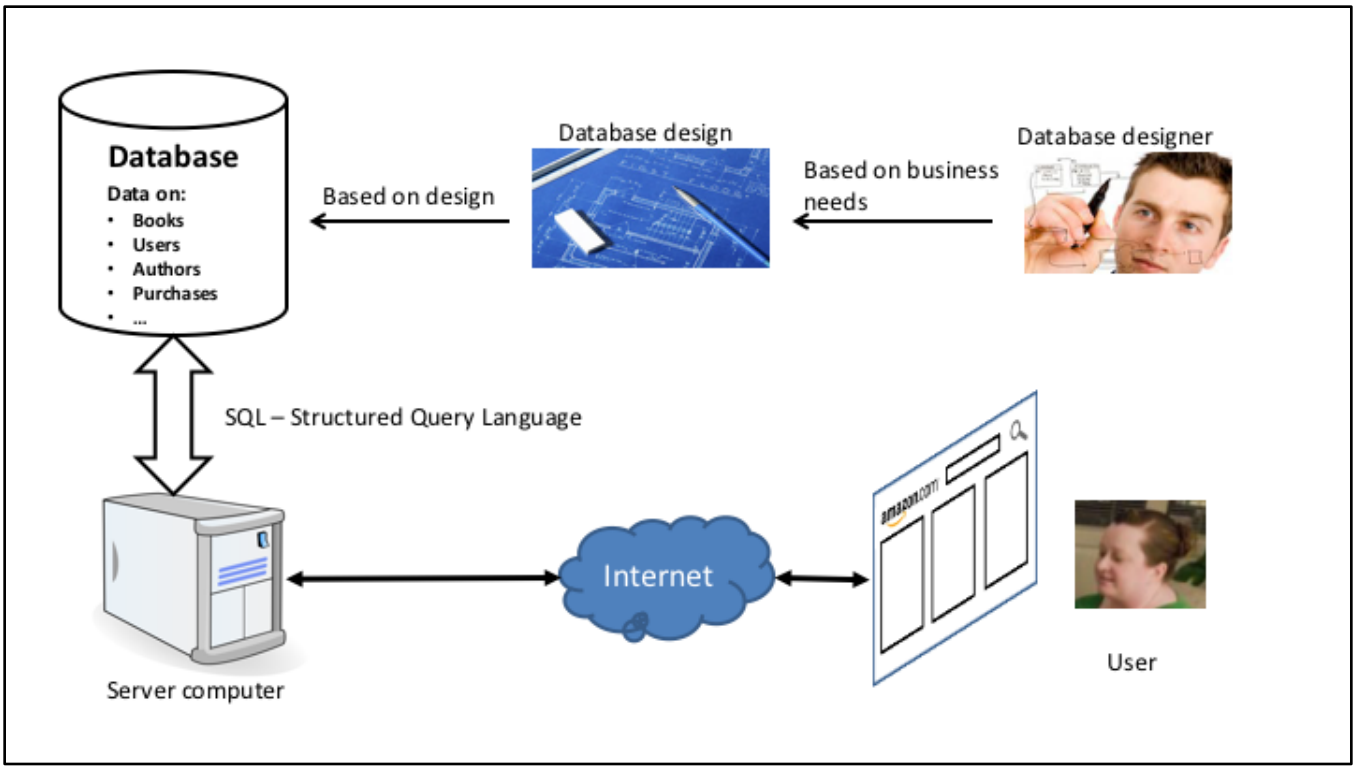
Information systems form the nervous systems of organizations today. Large organizations spend billions on technology infrastructure. People from all functional areas – Finance, Accounting, Marketing, Manufacturing, supply chain, etc., – participate in team efforts to build information systems and hence require a good grasp of the components of a business application and how they fit together.

Week 1

Course overview

Retrieving information from relational tables

Basics of Internet and the web



This diagram captures the essence of the course.

We will tackle various aspects of the diagram at various points in time and return to it for context.

In learning anything, it usually makes a lot of sense to be able to do something with the knowledge. In this course we will learn about enterprise applications:

- What are they?
- How do we design them?
- How do we build them?

Some might say "I only need to be able to use applications, not design and build them and so why should I learn about all this stuff?"

Valid question indeed. After all, all of us drive cars and 99.99% of the people who drive cars do not know or care how the car companies design and build cars – extremely complex systems too. Yet we do not see chaos on the roads with confused people on the roads wondering what to do next.

In fact, many people use this car analogy to argue that we do not need to know too much about several things in order to be able to use them effectively. Today, people use computers for numerous tasks and yet only a very small percentage understands how it all works.

So what is so special about enterprise systems that I am trying to teach you how to design and build them?

Let us first be clear about what the term ***Enterprise Systems*** stands for. We will then be better equipped to consider why you need to learn something about their design and development.

Packaged software

Vs.

Custom–built software

Products like Microsoft Word represent packaged software. The software vendor has developed the product and offers it to us. We play no **direct** role in its design or development.

Custom-built software (or bespoke-software as the Brits call it) represents software that an organization builds for its own needs. The organization might use its own people, or use a vendor, or use a mixed approach to build such systems. Organizations develop these one-off systems for the specific needs of their situation and with no intent to offer it as a product to a wider customer base.

For many activities for which we use software, packaged software serves the purpose rather well. Today, we would hardly need to custom-develop a word processor or a spreadsheet. Almost everyone needs the same features in such applications and no organization should to go through the tremendous time and expense of building its own. The products that we can buy in the market (or even download legally for free) meet our needs; we can buy supremely sophisticated products that would take us years to develop on our own – even if we had the considerable technical skills and sophistication that developing such products would take.

When it comes to software for processing business transactions, we find many significant differences across countries, industries and even individual organizations. Packaged systems that work out of the box for all or most organizations seems inconceivable, at least in today's business and technology landscape.

Software support for business transactions

Every organization performs numerous business activities.

For example, a company

- sells products
- buys raw materials
- buys fixed assets
- moves stock between warehouses
- pays employees
- pays vendors
- processes payments from customers
- …

We will refer to each **occurrence** of any of the above activities as a **transaction**. When a customer orders some products, we say that a **sale transaction** occurs. When a company orders products from a supplier, we say that a **purchase transaction** occurs (which, from the point of view of the supplier, is a sale transaction). When a company receives a shipment from a supplier (possibly against a purchase order that it earlier sent to the supplier), we say that a **goods receipt transaction** occurs and so on. In essence, whenever most business activities occur, information systems record the events as transactions. Of course, organizations cannot record ***everything*** that happens, but aim to record all significant occurrences.

Why do organizations record transactions? Unless they do so, they would be forced to rely on their employees' memory to ensure that necessary work gets done. For example, when a customer places an order, the order would need to be fulfilled (that is, the ordered items should be shipped to the customer). Some department other than sales would likely be responsible for shipping out the goods. It is even possible that a warehouse located halfway around the globe could initiate the process of picking the goods from the shelves and shipping them. The recorded sales transaction could form the basis for the shipping department to initiate the shipment process.

Although all the examples above refer to transactions that an organization conducts with external agencies, organizations certainly need to also record activities that do not involve external entities. For example, in order to manufacture goods, an organization would need to use raw materials. Companies often (but not always) purchase and stock the necessary raw materials. When some of the raw material is issued from stock for manufacturing, recording this event (transaction) will enable the responsible people in the organization to keep track of the stocks of various materials and order more as needed.

In general, organizations aim to record all significant events – no matter whether they deal with entities external to the organization or not – as transactions. They can then use this information to facilitate operations.

Non-business organizations – like universities, libraries, governmental agencies and such – also perform numerous transactions. Some of the transactions involve external entities, but many do not.

Organizations maintain **computer-based transaction processing system**s – information systems that record the essential details of transactions for future use – for many of their day-to-day activities.

As organizations conduct their activities and record significant events using transaction processing systems, they accumulate a huge amount of data over time. They can then generate reports and look for significant patterns in this vast store of data to assist in **planning** and for **strategic decision-making**.

Organizations started using computers for transaction processing as early as the late 1950's. Initially they used computers for automating payroll processing, financial accounting and so on – tasks that required the time of many people, but were easy to automate because the underlying rules and computations were well-defined. In addition to streamlining operations, companies enjoyed direct cost savings from such transaction processing systems.

As the benefits of computers became more and more clear and as computers also became more and more powerful and cheaper, organizations started computerizing many of the other activities as well.

In those days, almost all computer based systems were custom developed – in other words, each organization had to design and develop its own systems; a very expensive activity. Pretty soon however, payroll processing and financial accounting became packaged applications that any organization could buy as a package (much like we can buy MS Office today). They would then **customize** it to account for some of their own unique practices and then put the system into operation. This served to reduce time and cost. Vendors were able to provide packaged (or nearly so) solutions in these two areas because these processes were highly standardized across all organizations and exhibited very few process variations.

Most of the other areas remained in the domain of custom developed applications for a long time.

Building a software application to support organizational processes (like any one mentioned on the slide) requires the involvement of people with very specialized skills – programmers, analysts, designers, system architects and the like). After building and deploying these applications, an organization does not need the services of these people with specialized software development skills. Some companies chose to develop most of their applications software in-house and kept its software development teams as full-time employees and these people would move on to a new project once an existing project finishes. Smaller organizations which could not afford to employ large software development teams and other organizations (large or small) that did not wish to get involved in developing their own software often outsourced software development to a larger or smaller degree.

As a result, some consulting companies (the larger ones being SAP, Oracle, IBM, for example) gained a lot of experience in building such software. Some of these companies developed deep domain expertise in some areas of business systems and started offering packaged solutions that could be customized for use. Although such customization would take time, effort and money, the cost would still be much less than what it would take for companies to build their own systems.

Batch and on-line systems

Given the technology available through the 60's, companies mostly relied on "batch-processing" applications.

In batch processing applications, information on business transactions is accumulated for a period of time and then input into the computer system as a "batch". To take a concrete example, let us suppose that a company uses a "batch" system for processing sales orders.

This would mean that as sales orders are received (perhaps over phone), someone would fill out a form with the details of the order. All of these forms would then be sent to keypunch operators who would key in the details into some machine readable form (magnetic tapes or paper cards). At night, all of the data would be fed into the computer. Thus, during the day, the computer system would not have the most current information. It would become current only after the nightly "batch" process is completed.

This changed with the increasing prevalence of video terminals attached with keyboard, wherein a person could directly key in the details of transactions as they perform them. So a person who receives a sales order would directly enter the details into the terminal and transmit it into the computer system thereby enabling "on-line" systems that captured the transactions as they occurred. This meant that more and more of an organization's data was available "on-line."

What kind of systems do we have today? Think about a company like Amazon …

Clearly we have come a long way. Today, companies that conduct business on the web do not even have to allocate people to capture transactions … customers enter the transactions over the web all by themselves! What could be better?

Despite the availability of technology to support on-line processing, companies that had earlier developed batch systems for many of their applications could not switch overnight to "on-line" systems because of the time, cost and skill requirement. Consulting companies have had a field day ever since … because information technology keeps on changing and organizations find it impossible to keep all the necessary resources in house and have to constantly use the services of consulting companies.

Custom Development: Issues

The above model of custom development necessarily meant that companies developed individual applications piece-meal.

Almost always, these applications ended up not being able to inter-operate very well. We will have no difficulty in seeing that all of the areas mentioned earlier have lots of interconnections with many of the others, but since the systems were developed piecemeal, these interconnections were not accounted for smoothly.

For example, an order processing system needs to interface with the inventory, credit management, accounting and purchasing systems. If these systems were all developed independently, they might not all work together very well – thereby affecting the quality of each of them. The net result would be that a company makes substandard decisions – decisions that do not make use of all available information.

Enterprise Systems

In the meantime, consulting companies had developed much experience with developing systems to support various business processes and some of them started to develop and offer integrated systems that included many different application areas like accounting, order processing, purchasing, human relations management, and manufacturing control. These companies brought their experience with building systems for many different clients in different functional areas and were able to pool all this to develop such integrated systems that solved the interoperability problem to some extent. For an individual company to build such integrated systems would be a gargantuan effort taking many years and distracting the company from its primary business.

After a slow start in the mid-80's these integrated systems kept on increasing in scope and functionality.

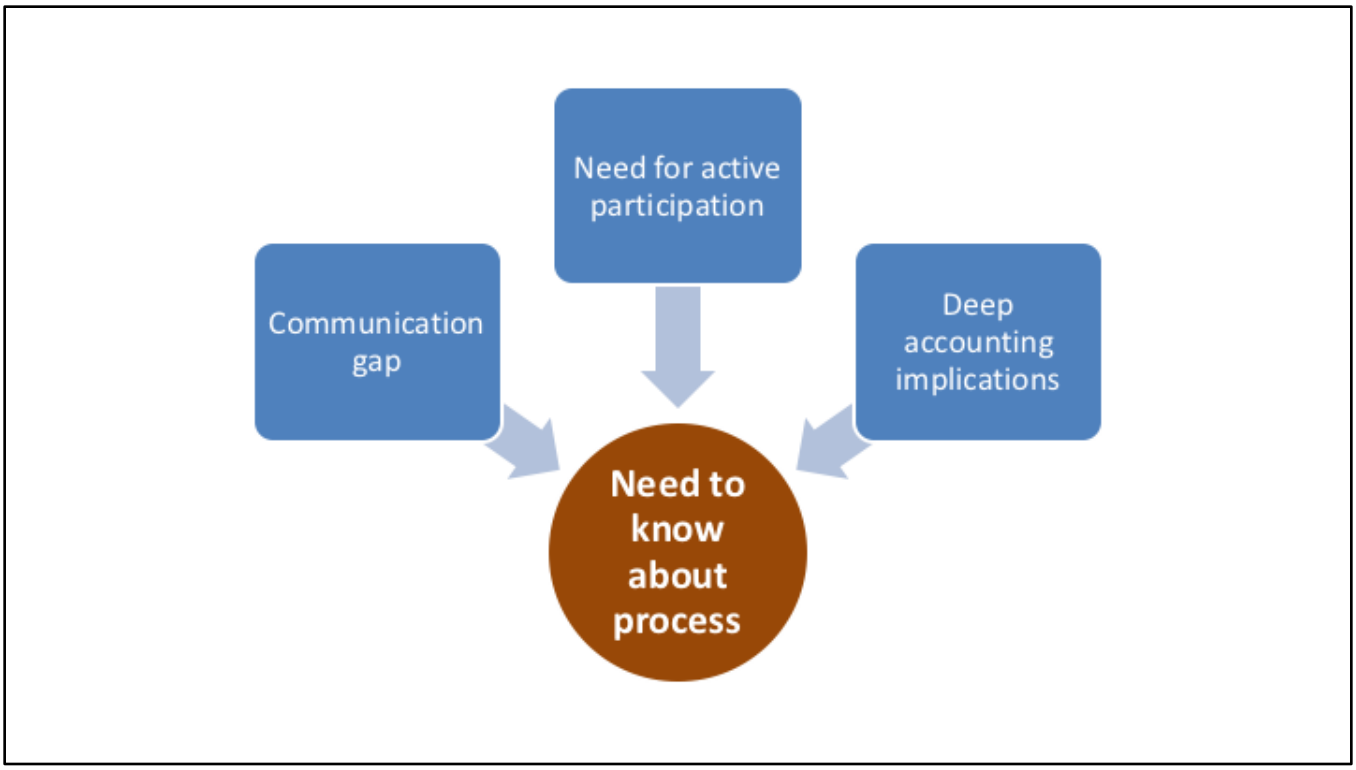
Fast forward to today – companies do not even think of building all of their transaction processing systems in-house. They mostly buy packaged products (like SAP and Oracle Applications) and customize them to their own needs. Of course this does not mean that all companies are using only packaged software to run their businesses. That would be lame – because companies try to use their superior information systems as a competitive weapon. If all the competitors are using the same software, then how can one of them gain any competitive edge from IT?

In reality, companies use packaged software for most of their needs – except for systems that are very closely tied to the core of their business. Here companies tend to invest their resources to build the key components of software systems that truly differentiate their company from others. IT would not make sense for a company to reinvent things like payroll processing and financial accounting – which are extremely standardized and are not expected to give a competitive edge.

On the other hand, a leading-edge investment company would probably invest a lot of effort in some components of its trading systems. A retailer would pay close attention to its supply chain management and order fulfillment systems.

To summarize, Enterprise Systems are integrated computer based information systems that have the functionality to support many functional areas of a business with tight integration across all functional areas. Integration enables companies to take into account a lot of information from diverse areas in order to make better decisions.

All this sounds great … but let us return to the question – "Why should I be teaching you the details of such systems? Would it not suffice for you to know how to "drive the car?" …



… for those concentrating in Information Systems/Information Technology the answer is pretty clear – because you will be playing an important role in decisions about enterprise systems. What about the rest? Why should Accountants, for example, be bothered about this stuff?

Enterprise Systems form the nervous system of almost all organizations today and thus most business executives, even non-IT -- play a role in the process of the design and development of information systems. At the very least, as day-to-day users of these systems, they want to ensure that the system will support the information requirements of their departments.

Many organizations have been burned by assuming that just telling the IT folks what is needed will ensure that the IT folks are then empowered to make the correct decisions. Not that IT folks are incompetent in any way, but business requirements are often very complex and business managers have found it painfully difficult to articulate their requirements up-front. Many requirements come up during the process of evaluating a system or a prototype. These represent things that business managers would have never thought about before, or would never have thought to articulate in writing – "because it is obvious …" Nevertheless, when they see real systems, they quickly realize that what the thought was "obvious" was indeed not at all obvious to people who built the system.

On the other side, business users of information systems often do not realize the ever-growing capabilities of modern information technologies and might express their needs much better if they understood what was possible.

All this points to the need for a highly collaborative process in which functional managers and IT experts need to work together.

Over nearly six decades of building business systems, people have learned that the process has to be collaborative. It requires users and builders of information systems to work very closely together for a good outcome. In fact when information systems implementations fail, they do so much more often because of communication gaps between business users and IT and not because of technology issues.

IT professionals sometimes do not understand business sufficiently and end up delivering systems that do not meet users' needs. Analogously, business managers understand too little about the issues involved in systems development and end up not communicating the proper requirements. So far, I have made the case for all business managers to learn something about enterprise systems.

Accountants are a different ball game …

They play an even larger role than other managers because enterprise systems automate much of the financial and managerial accounting tasks. As the system processes business transactions, it also does the necessary financial and management accounting postings automatically in the background. For this to happen properly, the system has to be set up correctly with the deep involvement of accounting professionals.

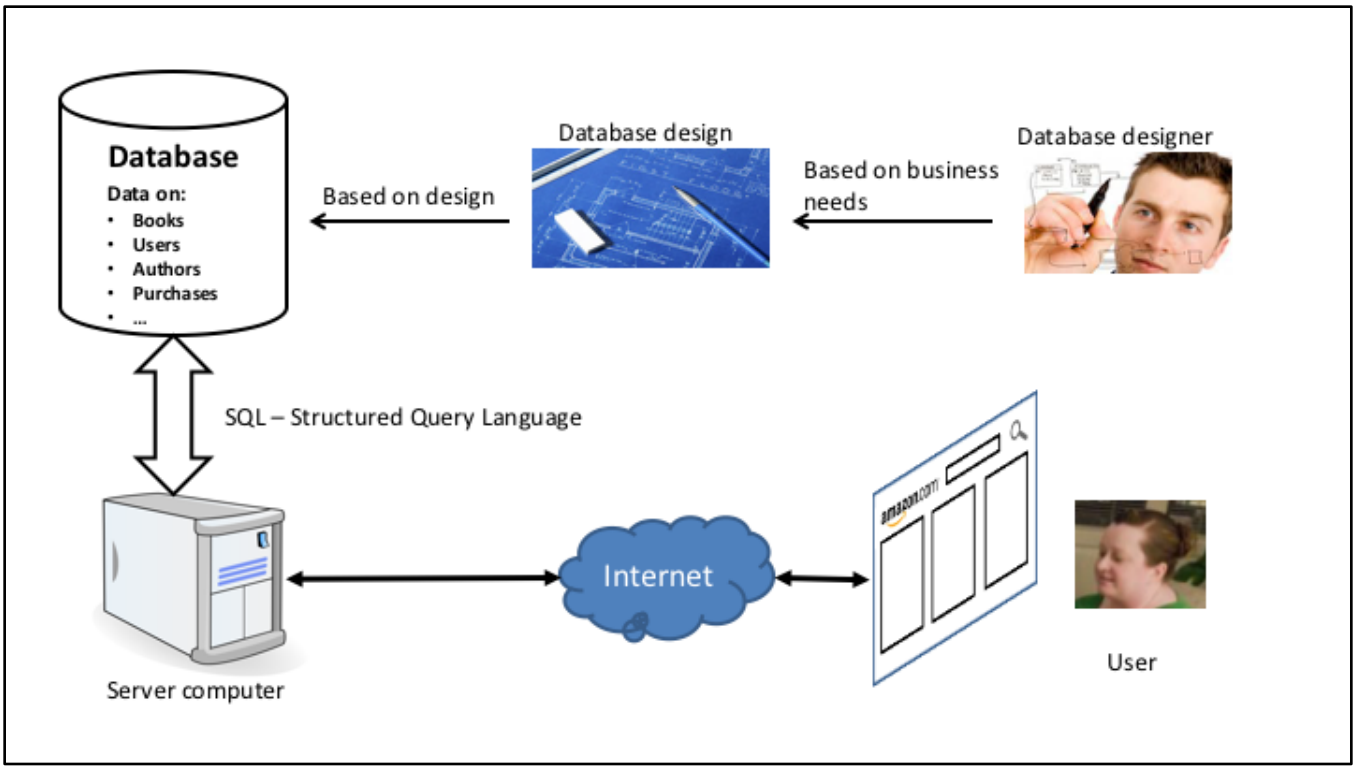
This course will give you a good feel for where these two skill-sets intersect so that you can play a useful role in the process.

Getting back to the car analogy, after a century of making cars, the automobile industry has enhanced the reliability and also understood user needs extremely well. The user interface has become standard – placement of the brake, steering wheel, accelerator, etc., -- as have the driving conditions all over the world. This means that car companies can give us a product that we can use well so long as we understand the universal user interface and know the road rules.

Business systems resist standardization in quite the same way. Organizations differ in their business processes. Rules and regulations differ vastly in different countries and even in different parts of the same country. Business practices and requirements are also often very dynamic.

Although all organizations exhibit many similarities, enterprise systems vendors have thus far not been able to deliver systems that meet all needs out of the box. The process of selecting such systems and then configuring them to meet the specific needs of an organization both require IT and business managers to participate – very actively too – for a successful implementation.

This course aims to equip you with the necessary skills to be able to participate effectively in the process.



Suppose you borrowed a library book and did not return it for a year, because you misplaced the book in your apartment and it got lost among the zillion other things crowded in there. In the meantime the library has been sending you reminders with annoying regularity every month. After a year, you find the book and take it to the library and are not very surprised to learn that the library would charge you a hefty fine. Simple question – how does the library know that you borrowed the book? In fact how does the library even know that the book was issued out in the first place?

You say, "of course they know – I used my library card and they checked out the book." So you did and so they did, but then, how do they know that the library card that was used was connected to you? "Duh, the card has a number and that number is associated with my name and id." OK, where is the library keeping all this "association between your library card number and your id number" in such a way that they can retrieve it whenever they want? Presumably they must be sending such obnoxious reminders to many other students as well – how do they know to find exactly the delinquents like you?

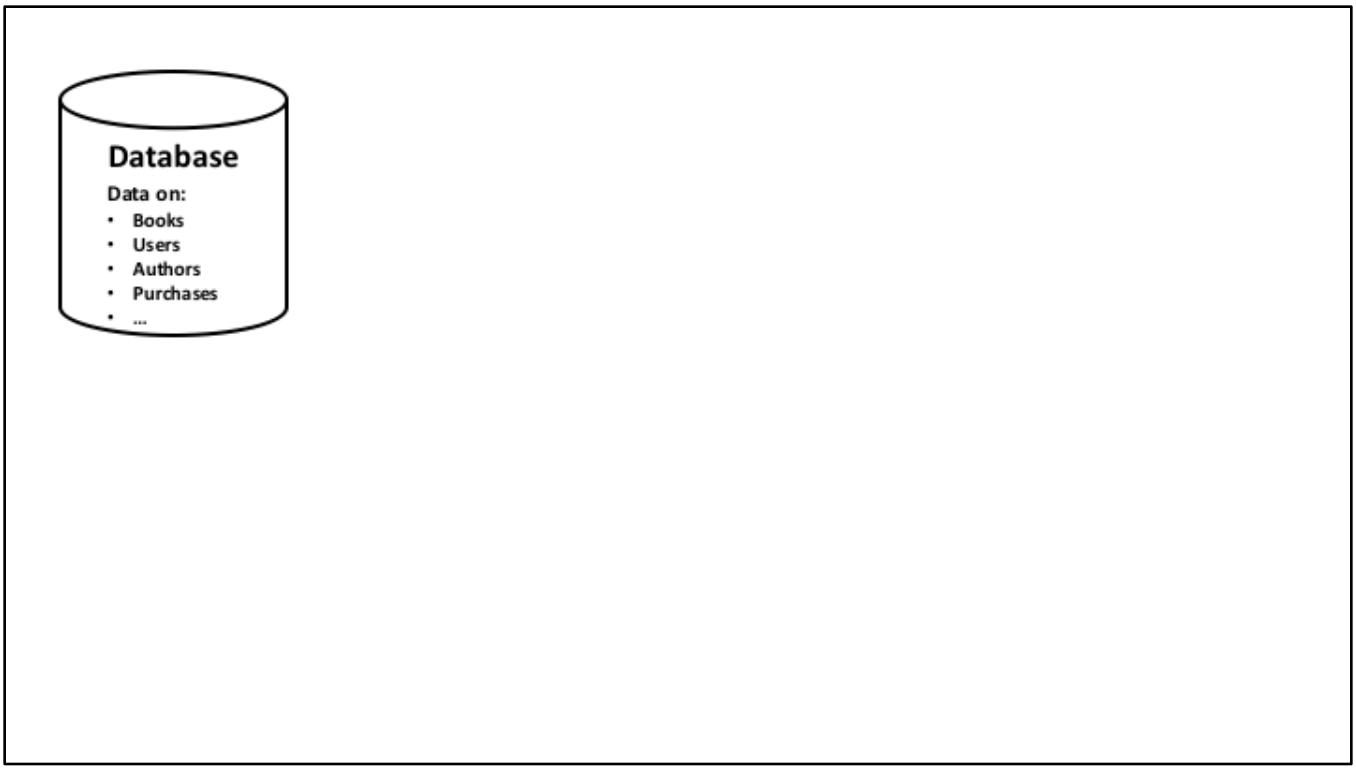
Clearly, in this day and age, we would be very naïve to assume that there is someone – some actual human being – who is doing all of this. We all know that this stuff is all automated and that the "computer" is doing it. We can infer that the library is storing this information and is able to extract the information as needed.

When they scan your library card and pick off the card number by using the barcode scanner, they can immediately retrieve all information about the card, the account and its holder based on the bar code. This might be fine for a university with 10,000 students.

Now imagine Amazon.com … with hundreds of millions of customers (or Facebook with more than a billion users). You enter your username and password and Amazon is able to verify your information in a very small fraction of a second. You then ask for your account details – say purchase history – and again in a fraction of a second you see your purchase history picked out from among the several billions that Amazon has stored.

What makes this all possible? It is not just about the speed of modern computers – it is also about how the information is organized in the computer. It depends critically on how the **information about the business is modeled inside the computer**.

If the system does not have the proper underlying business model, then it cannot provide the results – and this can seriously impede the viability of the company.



Enterprises store their **operational** information in a **database** and refer to this database when they perform various business transactions.

For example, when someone at the library uses a barcode scanner to scan your library card, the system uses your card number to find all other details from the database. Later when you return the book, the system can then use the bar code on the book to find all its details, as well as when it was actually due.

Suppose you try to borrow a book that is kept for reference only, the database contains information about this and using that information the system will prevent you from borrowing the book. So you can see that the information stored also encodes business rules or facilitates applying them

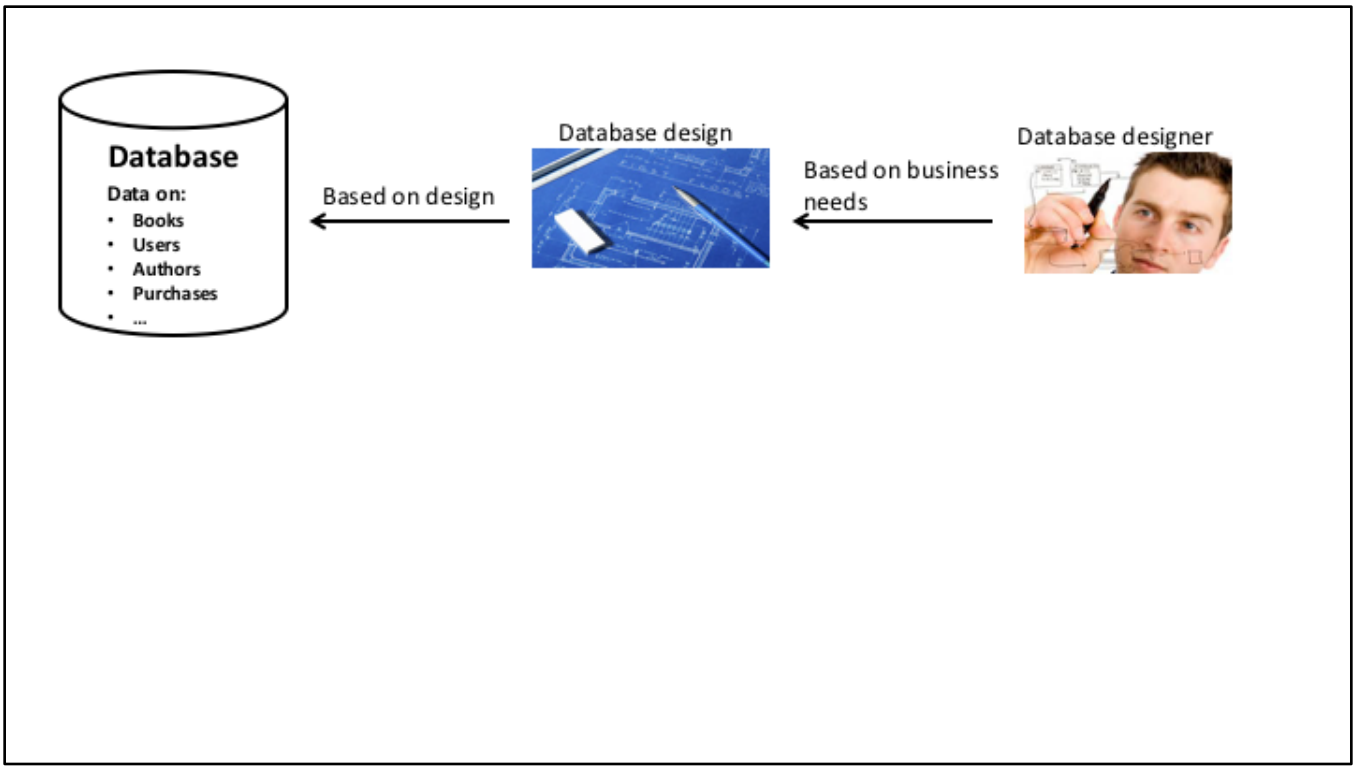
In this sense, the design of an information system is very closely linked to business knowledge and encodes a lot of it. Once business knowledge is encoded in a system, the corresponding rules (like books kept for reference cannot be loaned) are automatically applied over and over again and we deeply minimize the chances of error. On the other hand, if we have business rules not encoded in a system, then a human being has to be present to ensure that the rule is followed and we increase the chances of error. We also burden a human with responsibilities that could be mechanically applied by a computer. Overall, we put an organization at risk when we do not automate routine tasks.

Because business information is so central to an information system, its design and development require the active involvement of people with business knowledge.

The diagram shows that the **database** component stores information. For example, it contains all information about your library account, books, loans, returns and so on. When your card is scanned, the system retrieves your information from the database.

The database is the system's **memory**. If the information in the database is somehow destroyed, then the system loses all its information – that is its memory. A library would lose its record of who has borrowed what and hence risks losing a good chunk of its collection. A business that loses its database risks losing information on who owes the company how much and hence cannot invoice people and can go out of business. In the meantime its vendors will still be invoicing it for purchases. In fact when a vendor send an invoice, the unfortunate company cannot even verify if the invoice is correct. This scenario might scare you, but companies take a lot of care to prevent losing the data in a database.

In an **enterprise system**, a company maintains, in a single database, all information for all of its various applications. This important aspect allows all of its applications to be integrated – that is for applications to refer to information created by other applications.



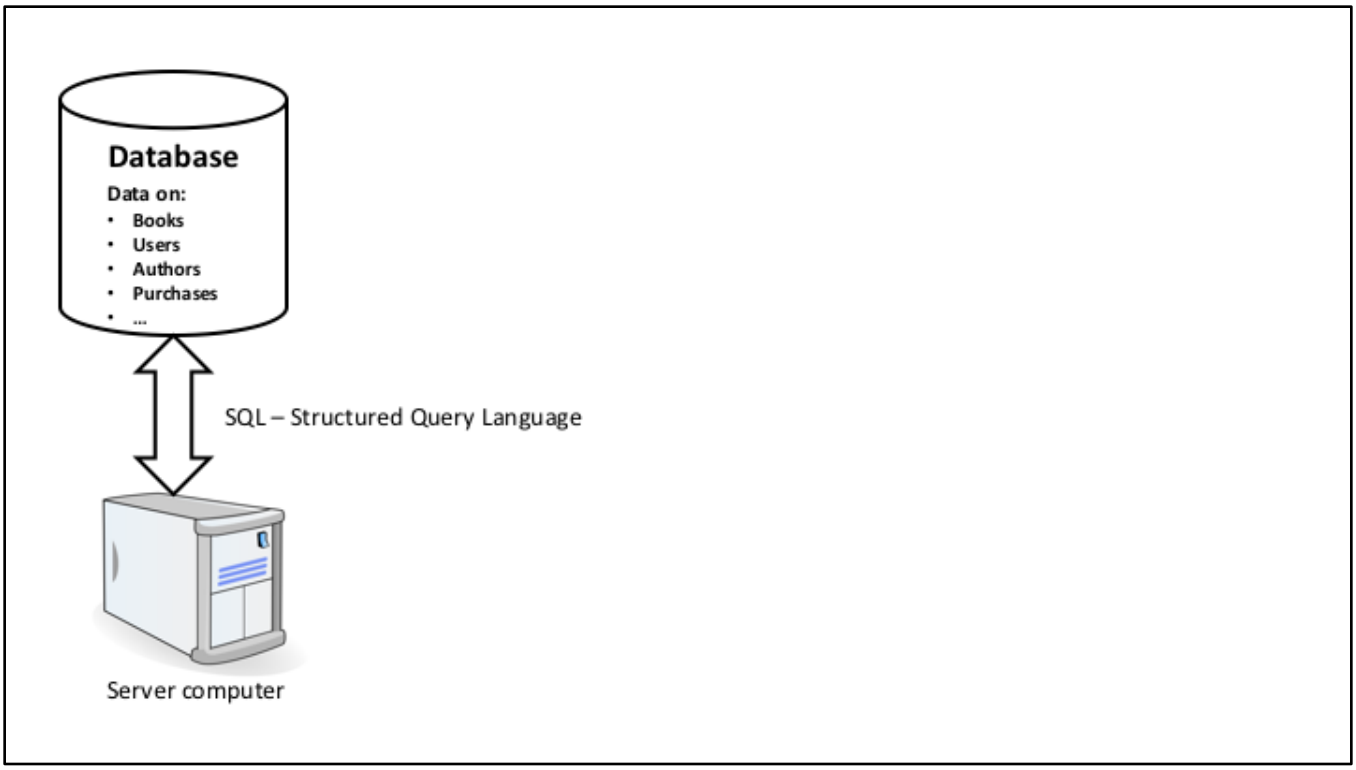
Although we do know that companies store all their information in a **database**, it is not enough to simply throw information into a computer disk. For example, a company cannot just scan millions of invoices and put them on a disk and hope that this information will be useful in the future. It cannot create MS Word documents containing information about thousands of transactions and hope to be able to retrieve information quickly when needed.

We are using the term **database** in a much more technical sense than just as a collection of data. In a very loose sense, any information thrown on to a computer disk can be called a database, but in this course we use it in a much more restricted sense. We will be learning about **relational databases** – more about the details of this shortly. We will learn how to take business requirements and design a suitable **relational database** corresponding to those needs.

As you see from the above figure, database designers **design** databases after **analyzing business needs**. In turn business needs are almost never neatly recorded in a document that someone can just read up. Instead **business needs and requirements** reside in many people's heads. People often even have conflicting ideas about the needs of their businesses. Very often, two parts of an organization inter-operate and each might have very little understanding of what goes on in the other unit. These kinds of common situations necessitate business managers and IT specialists to work together in creating business systems.

In this course you will learn some of the language that occurs at the intersection between business and IT so that you can be a valuable participant in the process.

Based on personal experience, I can say that you are about to learn a powerful language – one that will enable you to get a clear perspective about how a business operates. Often you will be able to derive deep insights and a broad overview that very few others in an organization possess – and this can make you a very valuable member. In fact, people will see your value and you will suddenly find yourself being invited to very high level meetings.

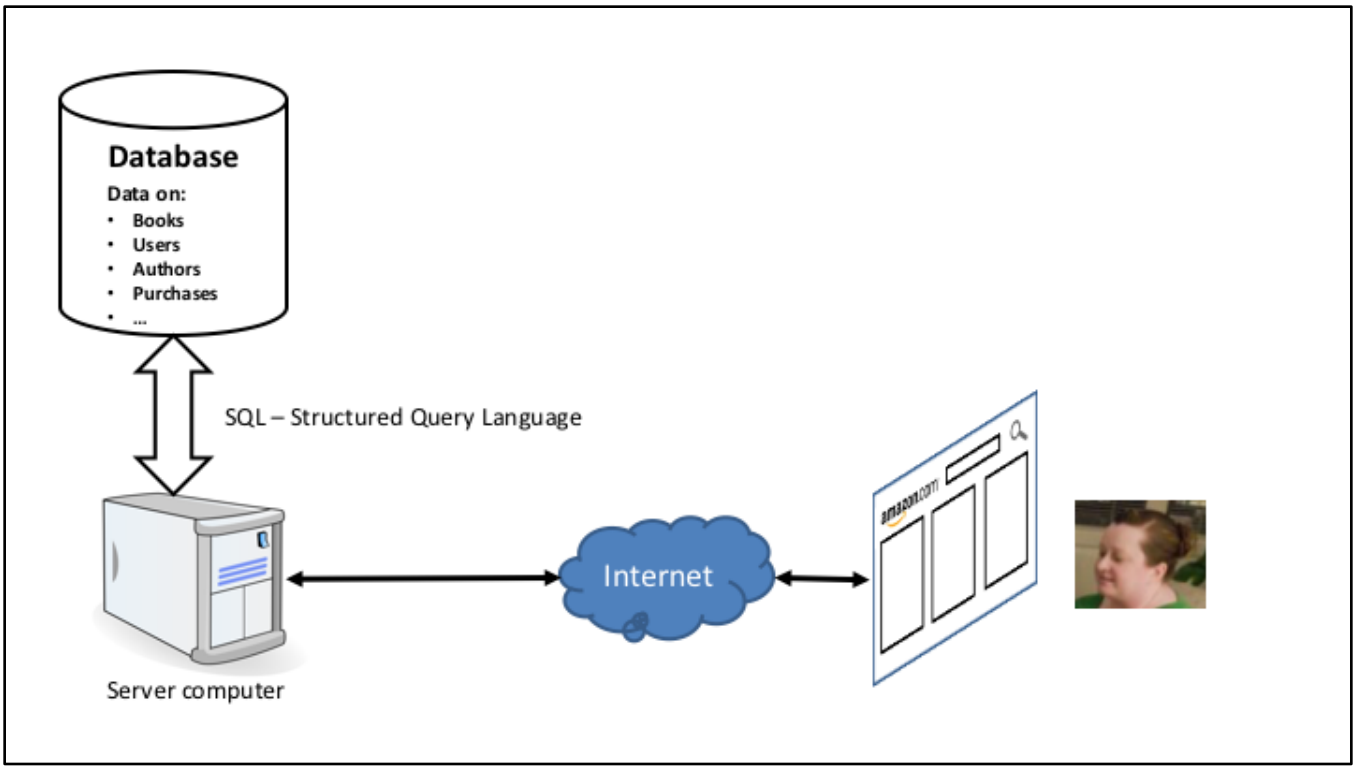


Once we have the data in a database, we can retrieve information from it very flexibly. You will soon see that the database design should be such that we do not need to know up-front what kinds of retrieval requirements exist. Ideally we will encode all important the business rules in the database so that we can pull out whatever information we want from it.

Poor database design can cripple an organization because as the organization grows, its information retrieval requirements will also evolve. With a poorly designed database, future retrieval requirements might not be met and this can be a serious detriment.

The **Structured Query Language** or **SQL** enables us to retrieve information very flexibly from a **relational database**.

You will see that when you design the database, you actually never consider what information retrieval requirements exist. You will just be capturing various business rules – your focus will be on just how the business operates. Magically, you will later see that through fairly simple SQL queries you can extract pretty much any information you want. This is one of the most fascinating aspects – the genius -- of the relational database model that has enabled it to be the engine of computer applications for over three decades when almost everything else about computer applications turned topsy-turvy.

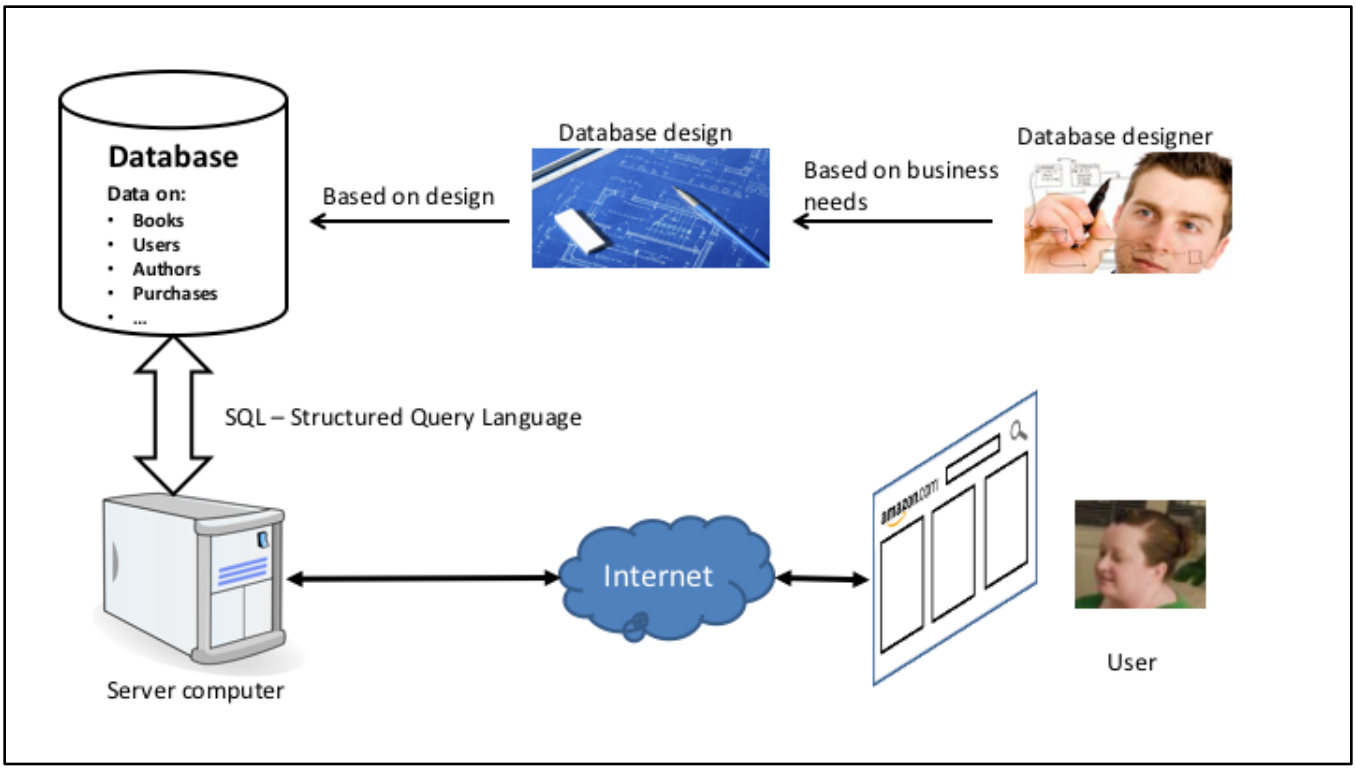


These days, most enterprise systems are deployed as web applications. That is, people either actually use a web browser to access them or access them through some other application which however uses the Internet to connect the user to the system.

In this figure we see how a user action on a web browser (perhaps on a PC or a Mac or an iPhone or Android or Windows phone), goes through the Internet to a server computer. The server computer, in turn executes some code that accesses data from a database and returns the results to be displayed to the user on their device. This explains the core of how web based applications work.

Let us suppose you learn how to design relational databases and how to extract information from them using SQL.

Would you not want to develop at least some part of an enterprise system and actually see it working? What fun would it be if you cannot actually see your application working and I just told you to take it faith that it can be done. In this course we will also learn a little bit of some other technologies to enable you to build an end-to-end application to see the whole picture. The focus of course is on modeling the business.



- Capture business requirements.
- Design a database.
- Create the actual database
- Develop the programs to carry out business transactions
- Write the necessary code to deploy the application over the web.

That is what you will be able to do at the end of the course. It is mostly about "business" and then a little bit to be able to see how it all fits together.