Database Design - A Complete Study
Part 2 - From Theory to Practice

Written by Stéphane Richard (MystikShadows)

INTRODUCTION:

Welcome to this second installement of the Database Design series of articles. As we stand today, we've covered most of all the theory relevant to database design. You are now equipped with the knowledge of terms you're very likely to hear in your programmer's carreer and you should know what those terms mean and a bit of how to use them as well. The best part of this theory is definitaly the Database Normalization section and this is what this second installement will mostly be about.

Indeed, in this installement well describe a typical situation you're likely to face and bring it forward by extracting the information you need from the situation right down to defining the tables and the needed relationships, as per the database normalization theories defined in the first installement of this series. At the end of this installement, you should be able to do the same for any and all database design related situations you'll encounter in your career and in your personal projects as well. So let's get right to it shall we?

THE SITUATION:

In most cases, database design revolves around a company and it's operation. It could be in the goal to automate a specific manual process or to go around billing, accounting or purchasing. The reason why it revolves around that is because in many cases, a business has very specific needs in those areas and often, the commercially available products just don't quite answer the real needs of the companies. Therefore, we will be creating a billing, customer, inventory and purchasing system for the situation.

Now, what would be the first thing to come to mind when you receive this task? This depends on many things actually the first of which is how the problem is described to you. You can receive a complete sentence telling you something like: "I want you to make a billing system wich affects the inventory of the business and something to allow me to purchase items.". you could also receive a complete 500 page book on what exactly the company wants that details absolutely everything you'd need (in a perfect world maybe). The truth of the matter is most companies that would call you or your firm to get a software made don't always know what to ask for or how to ask for what they want. Not eeryone is gifted when it comes to explaining their needs and that's just part of the challenge.

DOMAIN SPECIFIC KNOWLEDGE:

If you remember from the first installement, domain specific knowledge is good to have when you're designing a database for a specific need. And the reason why it's good (read important) to have domain specific knowledge is that in many cases, the company will expect you to know what they want. Don't worry though if you don't have the knowledge you can always learn it pretty quickly in most cases. So let's start by stating the domain specific knowledge for this particular situation.

When you talk about a billing system, usually, you're talking about invoicing. Who do you invoice? Customers, what are they buying? your products, where does the company get their products? From their suppliers (or they made them themselves in which case they have suppliers for the raw materials). Where do they put those products? in their warehouse (which means they have an inventory to keep). Is it just me or do these questions and answers seem somewhat simplistic to you? Well, this first set of questions is pretty much all you need to determine your primary entities (the first set of tables you know you'll need to have). I think it's clearly noticable that there are five distinct elements in this set of questions. You can see that you'll need: Customer management, Invoice Management, Inventory Management, Supplier Management and Purchase Order Management. Let's take each of these and talk about them a bit, see what we can come up with.

- **Customer Management:**
  Customer Management usually involves more than just customers and their information. Sure you'll want to know their name, address, telephone number(s), etc etc...But Customer Management also could imply you want their history (what they bought) their payment history (how well they paid for what they bought) too. It depends on the needs of the company really. But for this situation let's assume that they want it all. There's another concept we'll add here because like the others it can definitaly be asked for. The concept of Accounts Receivables will affect the tables you define your tables and which

table you'll need to define. Although ultimately, Accounts receivables can be managed 100% from a software point of view, many companies like to keep this kind of track record information just to help them make sure that all the invoices they make eventually get paid. Accounts receivable have 6 major cycles each of 30 days 3 Receivable statuses and 3 collection statuses This means that after 180 days (roughly 6 months) that account is considered lost so to speak. The age of the account will be important here

- **Invoice Management:**
  Invoicing generally requires two tables The invoice header which will have such things as the invoice number, the date the invoice was made, to whom and invoice total and taxes. And the invoice detail which gives one line per item purchased. This table details each item that is being sold and should appear on the invoice. If you look at a typical invoice you have had you can distinguish these two elements fairly easily. As far as Invoices go, these tables are all you need. However you'll see that they will relate to other tables in order to perform their designated functionality. Payment Methods are also important to know. Which payment method is allowed depends on the company itself so it's a good idea to keep the payment methods configurable in some way.

- **Inventory Management:**
  In a very broad sense of the word, inventory management implies that you'll be managing the products that you sell and buy. Hence you can conclude that inventory will be affected by what you sell to your customers and by what you buy from your suppliers. It involves a list of product and specific information about products. Some may like to break down their inventory by categories of products to help in managing itself a bit. In essence that's all that's needed but throughout the years inventory has been refined quite a bit. Inventory also plays the role of controling and making sure no products are lost that can't be accounted for. In bigger systems they even have different means of controling inventory depending on the type of customer they deal with. They would control inventory by the item for regular people coming in and buying one or two of a given product and control inventory by the case when they sell to companies. This is why Inventory Management quickly became known as Inventory Control (because now you need to be able to know how many items you bought, how many you sold, how many should be in stock and how many actually is in stock (if these last two aren't the same, then something went wrong in the inventory (maybe internal theft or something) and a good inventory control system can help give many indications as what could have happened to the missing products

- **Supplier Management:**
  Suppliers are of course the companies that you buy your products from. Hence You'll want their information much as you would want information about the customers. Like Accounts receivables help in making sure the company's invoices get paid by the customers, Accounts Payable is a good tool to make sure the company pays it's bill on time. And a history of what you purchased from your supplier is a great way to know which of your suppliers you use the most and how much business you've given them. You might notice some companies care a bit less about supplier management much for the same reason as any of us hate to pay our bills. it's often regarded as something they have to do, not something they want to do.

- **Purchase Order Management:**
  Invoices are the means by which Items are removed from the inventory of the company. Likewise Purchase Orders are the means by which Items are added to your inventory. You buy items from your suppliers to repplenish your inventory and have items ready and available for your customers to be able to by them without havjng to wait for them. However sometimes this is hard to achieve and a system of split purchasing and back order operations is always good to have implemented.

EXTRACTING THE DATABASE RELATED INFORMATION:

So what exactly can we extract from all that's been mentionned above about domain specific knowledge? Well we know we have 5 major players in our design (mentionned above) and we know that each player will need more than one table to perform it's designated task. Let's list the table we'll need here and explain their role briefly. This is where we'll start to name our tables so the clearer the name the better because when we create our queries and scripts these names will play a big role. What I'll do here is create the table structures (in non SQL terms, just to give you an idea of what fields could be needed in this situation and give you a few notes about each table. This should make it clear enough what role the tables play and how we'll later create the database definition for these tables (in the next installement).

Customers Management:

As mentionned, the Customers table definition will need tables for the customer information, List of ways to contact the customer, a history table to keep track of what invoices and how much money the invoices represents as well as an accounts receivable table to organize the way the invoice will be processed. Here's the table definitions for the customers.

Customers		
Field Name	**Type**	**Length**
CustomerID	Numeric	4
CustomerName	Character	50
IsCompany	Boolean	1
CustomerSince	Date	8
Address1	Character	60
Address2	Character	60
City	Character	40
State	Character	25
ZipCode	Character	8

This table is where the customer management system begins. This central table is where all other tables that perform some kind of customer management function will need to relate to.

All field names and lengths I have chosen here are there simply as a sample of what you'd be likely to find in many existing database systems. For the sake of different project types they might change. The naming convention here is simply to have clear names.

CustomerNumbers		
Field Name	**Type**	**Length**
CustomerID	Numeric	4
ContactDescription	Character	30
PhoneNumber	Character	14
ExtensionNumber	Character	6

This table will hold all possible means to contact a given customer. Some typical data that can be found here are Business, (222) 222-2222, Home, (333) 333-3333, etc etc. Note that this means that there will typically be more than one row of data per customer.

CustomerHistory		
Field Name	**Type**	**Length**
HistoryID	Numeric	4
CustomerID	Numeric	4
InvoiceNumber	Numeric	4
InvoiceDate	Date	8
InvoiceTotal	Numeric	8

This table will allow the users to quickly see what the customer bought as well as if all the invoices have been paid. Note here too that there could be more than one record for each customer as each invoice made to that customer will have a record

Balance	Numeric	8

added to this table.

CustomerPaymentHistory		
Field Name	**Type**	**Length**
HistoryID	Numeric	4
PaymentID	Numeric	4
PaymentDate	Date	8
PaymentAmount	Numeric	8
BalanceDue	Numeric	8
PaidInFull	Boolean	1

This table is added to help trace, in detail, what happened throughout the life of a given invoice made to a given customer. Note that more than one payment could potentially be made to an invoice which means that more than one record could be found here for a given history number.

AccountsReceivable		
Field Name	**Type**	**Length**
CustomerID	Numeric	4
InvoiceNumber	Numeric	4
InvoiceDate	Date	8
BalanceDue	Numeric	8
CurrentCycle	Numeric	4
ReceivableOneDateEnd	Date	8
ReceivableTwoDateEnd	Date	8
ReceivableThreeDateEnd	Date	8
CollectableOneDateEnd	Date	8
CollectableTwoDateEnd	Date	8
CollectableThreeDateEnd	Date	8
AccountPaid	Boolean	1

This Table is where Accounts receivables come to play. Typically, the day you create an invoice that won't be paid right then and there, a record will be added here. All the Accounts Receivable cycle dates in this table can be evaluated and save. This will allow for an automatic scheduling of the account for processing when the given cycles reach their ending dates. Only one record here will be present for a given invoice so the relationship here would be one to one.

Typically, the Cycle Date Ends would be, 30, 60, 90 for the ReceivableDateEnds and 120, 150, 180 for the Collectable Date Ends.

Invoice Management:

Like we said earlier, invoices only need 2 tables. However you'll notice that in the table definitions above for the customers, many of the tables have an InvoiceNumber for a field, they will all related back to the invoice to reveal some very detailed information about the invoice and what's been happening to it in it's lifetime. So let's define these invoice tables:

InvoiceHeader		
Field Name	**Type**	**Length**
InvoiceID	Numeric	4
InvoiceDate	Date	8
CustomerID	Numeric	4
InvoiceTotal	Numeric	8
SalesTaxRate	Numeric	8
SalesTaxAmount	Numeric	8
GrandTotal	Numeric	8
IsPaid	Boolean	1

Here is the Header table for the invoicing system. As you can see, this represent esential information you would be likely to find on an invoice. Note here as well that an invoice needs to be made to a customer which means that it needs the CustomerID field to be filled. Needless to say that a customer can have more than one invoice which is yet another relationship to consider.

InvoiceDetails		
Field Name	**Type**	**Length**
InvoiceID	Numeric	4
DetailID	Numeric	4
ItemNumber	Numeric	4
Quantity	Numeric	4
LineTotal	Numeric	8
IsTaxable	Boolean	1

This table holds the details of each invoice. essentially there is one record per each different Item sold as part of this invoice. Note that no product description is specified here because it can be retreived from the the Inventory table. This is one of the normalization forms in action right there. It helps avoid the repetition of information in more than one table wherever possible.

Inventory Management:

And now we're at the part where products are maintained, otherwise known as Inventory Management And Control. The main product table needs to have sufficient information about a product to be able to do everything we need to do with the inventory. In order to truly manage the inventory and everything that relates to it, you'll see here that we will need a few tables to go with the main product table. Let's defined them:

Products		
Field Name	**Type**	**Length**

The Products Table holds information about a product currently present in the company's inventory

ProductID	Numeric	4
SupplierID	Numeric	4
CategoryID	Numeric	4
BrandName	Character	30
ModelNumber	Character	20
Description	Character	50
CurrentPriceBought	Numeric	8
CurrentPriceSold	Numeric	8
QuantityInStock	Numeric	4
MinimumInStock	Numeric	4
MaximumInStock	Numeric	4
IsTaxable	Boolean	1
IsActive	Boolean	1

including where the product was bought (supplier). With this information it's possible to manage the inventory very well. Some other inventory system manage products by unit (depending on the needs) but in this case we'll manage quantities. Not that more than one product can be present that was bought from the same supplier (can you say relationship?).

Note also that Minimum and Maximum quantities are there to help tell is if we're running out of a product or if we'd be in overstock if we bought more.

Categories		
Field Name	**Type**	**Length**
CategoryID	Numeric	4
CategoryDescription	Character	40
ParentCategoryID	Numeric	4

This is a small table that servers to categorize the products. The ParentCategoryID serves to create a hierarchy of categories should the need be. A quick and useful trick in big inventories. More than one product can be in a category.

PriceHistory		
Field Name	**Type**	**Length**
ProductID	Numeric	4
PriceID	Numeric	4
PriceDate	Date	8
PriceBought	Numeric	8
PriceSold	Numeric	8

Another small table to help control the pricing reality of a product. Useful for tracebacks and when establishing special prices. There can be more than one price history per product.

Special Note:

In some inventory systems, quantities and items are not directly taken in and out of the inventory per se. They work with a principle of a transactional table where Items bought and items sold are recorded in a secondary table so that inventory is not touched until such a day where the products actually leave the warehouse or are officially received into the warehouse. A Transaction table would then be defined to accomodate for this functionality.

Supplier Management:

Suppliers are, for the sake of discussion, just like customers. Likewise Accounts payable work pretty much the same way as accounts receivables would but in revers since in this case you're paying your own bills rather than wait on customers to pay their bills to you. You like it when your customers pay their bills on time and it's not far fetched to believe that your suppliers like it when you pay your bills on time too for obvious reasons. Having an Accounts Payable system in place automates the bill paying part for you and can prove a great tool for consistent and good business relations. As such let's define the Supplier Magament Tables. You'll see that they are alot like the customer management tables and rightfully so:

Suppliers

Field Name	Type	Length
SupplierID	Numeric	4
SupplierName	Character	50
SupplierSince	Date	8
Address1	Character	60
Address2	Character	60
City	Character	40
State	Character	25
ZipCode	Character	8
PhoneNumber	Character	14
FaxNumber	Character	14
EMailAddress	Character	70
Website	Character	70

The Supplier Information shown here is typical of what you would need to know about a supplier. Note that here, the phone number related information are stored in this table instead of a seperate table because you don't usually need to track down a supplier, you just need enough information to create your purchase orders.

Note that more than one product can usually be obtained from a supplier. Likewise note that if need be, more than one Purchase Order could be made to a given Supplier for different reasons.

SupplierHistory

Field Name	Type	Length

This table will allow the users to quickly see what was bought from the

HistoryID	Numeric	4
SupplierID	Numeric	4
PurchaseOrderNumber	Numeric	4
PurchaseOrderDate	Date	8
PurchaseOrderTotal	Numeric	8
BalanceDue	Numeric	8

supplier on a given purchase order. Along with if the purchase order was paid or not. Note that more than one supplier history can exist for each supplier.

SupplierPaymentHistory

Field Name	Type	Length
HistoryID	Numeric	4
PaymentID	Numeric	4
PaymentDate	Date	8
ReferenceNumber	Numeric	4
PaymentAmount	Numeric	8
BalanceDue	Numeric	8
PaidInFull	Boolean	1

This table is added to help trace, in detail, when a payment was made to a supplier regarding which purchase order. The reference number here can be the cheque number used, or whatever other means of identifying the payment accurately for easy traceback if needed. Note that more than one payment can be made to a Supplier History for a given purchase order.

AccountsPayable

Field Name	Type	Length
SupplierID	Numeric	4
PurchaseOrderNumber	Numeric	4
PurchaseOrderDate	Date	8
BalanceDue	Numeric	8
CurrentCycle	Numeric	4
ReceivableOneDateEnd	Date	8
ReceivableTwoDateEnd	Date	8
ReceivableThreeDateEnd	Date	8
CollectableOneDateEnd	Date	8
CollectableTwoDateEnd	Date	8
CollectableThreeDateEnd	Date	8
AccountPaid	Boolean	1

The accounts payable table here doesn't quite server the same purpose. A company has no need to collect itself if the bill isn't paid in time. It's up to the supplier to do that. The role this table plays is to make sure that your supplier doesn't need to try to collect the account. Think of it as an agenda where you decide when to be notified that a payment should be made and just a helper tool for the company to maintain good relations with it's suppliers.

One accounts payable per purchase order is all that's needed. The cycles typically are the same as accounts Payable to indicate how quickly purchase orders are paid or "should" be paid.

Purchase Order Management:

Purchase orders are to suppliers what invoices are to customers. What this means is that the structures are quite similar to the invoice tables. THese are kept in different tables however because suppliers are not customers. You can however base your definitions on the invoicing tables. You'll need a purchase order header table and details (for all the products that you'll be buying from your supplier on that particular purchase order. When you receive your order, it will usually come with a packing slip that states exactly what you have received. You can use that to confirm products and quantities against your purchase order to make sure that you received everything that you ordered: Let's then define the Puchase Order tables as per our discussion:

PurchaseOrderHeader		
Field Name	**Type**	**Length**
PurchaseOrderID	Numeric	4
PurchaseOrderDate	Date	8
SupplierID	Numeric	4
PurchaseOrderTotal	Numeric	8
SalesTaxRate	Numeric	8
SalesTaxAmount	Numeric	8
GrandTotal	Numeric	8
IsPaid	Boolean	1

Like the invoice header table this is all the relevant information you need to know about a purchase order. Like invoices, Many tables will relate to this one to give a precise history of what happened with a given purchase order.

PurchaseOrderDetails		
Field Name	**Type**	**Length**
PurchaseOrderID	Numeric	4
DetailID	Numeric	4
ItemNumber	Numeric	4
Quantity	Numeric	4
LineTotal	Numeric	8
IsTaxable	Boolean	1

And that's it, these fields are all that's needed. With these fields you can defined what you are buying and how much of it. Of course more than one detail line can and will exist for each purchase order.

NOTES AND CONSIDERATIONS:

As you can see here, from 5 fairly small paragraphs we defined in the "Domain Specific Knownledge" section we've been able to define no less than 16 specific tables. The art of creating a database definition from simple paragraphs like these is where the domain specific knowledge really comes in handy. Of course, there's a couple more tables that could be defined here to give more functionality to the whole system. We could add PaymentType to the purchase orders and the invoices to help determine the sales / payment types ratios and such. there's plenty more little accessories tables that could be defined just to help bring things together even more than they are now. But that would overcomplicate this document with no particular purpose. It's just important to know that these 16 tables will typically always be what you need and with the proper relationships estables you'll be able to get some very useful information about the company and how it's running itself.

For example, starting from the products you can get a list of who and when the product was sold (hence if the product is popular enough to keep it in the inventory). If you add the dates these products were bought into the equation you can see if there are seasons (times of year) where it's best to make sure you have that product handy to make sure you can supply the demand. These (and many other) database tricks can really help a business know what's going on as well as what is likely to happen. This can greatly help a business be prepared for anything that could happen.

If you don't know what fields to include in the tables or which tables you'd need you can refer back to this document. But another well known method is to sit down with the company and ask the questions you need. No company will not cooperate when it comes to making sure their software does what they need it to do. You just need to ask all the questions you have to make sure that you can provide the best possible solution. Good communication is the key here.

AND TO CONCLUDE THIS SECOND INSTALLEMENT:

And there you have it, the second installement is now completed. There was alot of grounds to cover here so I definitaly hope that I was clear enough. If not, you know what you can do, email me and I can details things alot more than this. The important thing to grasp here is that there is alot of detail and effort to put in a database design. You need to think about what you are designing and what that design needs to allow in terms of information and what will be done with that information. If you stop to think about it you can see that a bit of common sense often helps in determining the information and what to do with it.

In the next installement, I'll teach you SQL itself. We'll learn SQL as it is implemented in the MySQL database system. We'll take the tables we defined here and see how we can create them and how we'd be using SQL to create the tables, add, update and delete records from the table as well as some querying to allow us to find the information we want to display back to the user. With this we'll cover all the SQL you'll need to know to take care of a whole database system of tables. It will be another big one but I'll do my best to be concise and to the point. So until the next installement, happy reading and coding.

MystikShadows
Stéphane Richard
srichard@adaworld.com