

**Accounting Database Design
(Limited Version)
Derek Liew Lei Mun**

**Published by Derek Liew at Smashwords
Copyright 2010**

Copyright © 2010 by Derek Liew Lei Mun. All rights reserved.

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner.

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, the names are used only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The information in this book is on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, the author is not liable to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

About the Author

The author is an ACCA qualified Accountant. He has vast technical knowledge in database design and development, with more than 5 years research experience in database design, especially in the area of accounting system design. The author is an experienced user of various accounting application and ERP system.

Comments may be directed to the author at: derek@accountingdes.com
Complete version can be downloaded at : <http://www.accountingdes.com>
Register for a copy of an Accounting Software now for free at :
<http://www.accountingdes.com>

Table of Contents

Introduction
What Is This Book Is About?
Who This Book Is For?

What You Need To Use This Book
Conventions
How It Works

Chapter 1 Database Design

Database
Relational Database
Primary Key (PK)
Foreign Key (FK)
Normalization Concept

- First Normal Form
- Second Normal Form
- Third Normal Form

What is SQL?
Transact-SQL
Installing Microsoft SQL Server (Personal Edition)
Creating a Database

Chapter 2 Developing the Journals Table

Normalization Journal Table

- First Normal Form
- Second Normal Form
- Third Normal Form

Designing Chart of Account Table
Designing Journal Table
Designing Sales Table
Designing Product Table

Chapter 3 Developing the Inventory Table

Normalizing Inventory Table

- First Normal Form
- Second Normal Form
- Third Normal Form

Designing Stock Movement Table
Designing Stock Balance Table
Designing Product Account Set Table
Designing Product Category Table

Chapter 4 Developing the Purchase Table

Normalizing Purchase Table

- First Normal Form
- Second Normal Form
- Third Normal Form

Designing Creditor Table

Chapter 5 Developing the Sales Table

Normalizing Sale Table

- First Normal Form
- Second Normal Form
- Third Normal Form

Designing Customer Table

Chapter 6 Developing the Cash Table

Normalizing Cash Table

- First Normal Form
- Second Normal Form
- Third Normal Form

Designing Bank Table

Chapter 7 Developing the Asset Table

Normalizing Asset Table

- First Normal Form
- Second Normal Form
- Third Normal Form

Chapter 8 Creating Reports from Journals Table

Using SQL to produce Trial Balance

How It Works – Select Query for Trial Balance Listing

Using SQL to produce Income Statement

How it Works – Select Query for Income Statement

Using SQL to produce Income Statement by Segment

How it Works – Select Query for Income Statement by Segment

Using SQL to produce Balance Sheet

How it Works – Select Query for Balance Sheet

Using SQL to produce Transaction Listing
How it Works – Select Query for Transaction Listing

Chapter 9 Creating Reports from Inventory Table

Using SQL to produce Stock Movement Report
How It Works – Select Query for Stock Movement Report
Using SQL to produce Stock Ageing & Balance Report
How It Works – Select Query for Stock Ageing & Balance Report

Chapter 10 Creating Reports from Purchase Table

Using SQL to produce Accounts Payable Ageing Report
How It Works – Select Query for Accounts Payable Ageing Report
Using SQL to produce Accounts Payable Payment Status Report
How It Works – Select Query for Payable Payment Status Report

Chapter 11 Creating Reports from Sales Table

Using SQL to produce Accounts Receivable Ageing Report
How It Works – Select Query for Accounts Receivable Ageing Report
Using SQL to produce Accounts Receivable Collection Status Report
How It Works – Select Query for Accounts Receivable Collection
Status Report
Using SQL to produce Sales Analysis Report
How It Works – Select Query for Sales Analysis Report

Chapter 12 Creating Reports from Cash Table

Using SQL to produce Cash Flow Forecast
How It Works – Select Query for Cash Flow Forecast Report
Using SQL to produce Cash Flow Summary Statement
How It Works – Select Query for Cash Flow Summary
Statement Report
Using SQL to produce Cash Flow Periodic Statement
How It Works – Select Query for Cash Flow Periodic Statement Report
Using SQL to produce Bank Reconciliation Statement
How It Works – Select Query for Bank Reconciliation Statement

Chapter 13 Creating Reports from Asset Table

Using SQL to produce Asset Summary
How It Works – Select Query for Asset Summary Report
Using SQL to produce Asset Movement Report
How It Works – Select Query for Asset Movement Report

Introduction

In our modern world today, it is undisputable fact, that most of the corporate world has and is continuously changing and adapting to new technology, especially in the area of computerization, in order to remain competitive in the business world. One of the greatest importances in any corporate industry is adopting a robust and powerful accounting application, that are not just user-friendly, but capable of providing the flexibility and scalability needed in a rapid changing environment.

A powerful accounting application depends fundamentally on a well structured and designed database. The traditional method of designing and creating a flat-file database is no longer viable and economical, as it has numerous flaw and limitation comparing to a relational database. Most of the existing database today, are developed using the relational database management system (RDBMS) approach, of which it is capable of enforcing greater data integrity and consistency, maximizing storage space efficiency and eliminating redundant data.

What Is This Book Is About?

This book will introduce the concept of normalization, adopting the first normal form to third normal form approach in designing and developing an accounting database. We begin to learn how to design and build a group of fundamental tables, representative of each accounting modules that forms the foundation of an accounting database. We learn how to normalize tables, by continuously adding and changing key fields, as we progress from one chapter to the next.

We'll then discuss the function of primary key (PK) and foreign key (FK) in each tables, and the use of building relationship in the Database Diagram. Finally, we'll walk you through creating query to produce report using the SQL Query Analyzer.

Who This Book Is For?

This book is targeted for database developer, database administrator, accountant and university students, who wants to increase their knowledge and skill set in designing and developing a relational accounting database, and have interest in writing SQL query for accounting reports.

This book assumes you are an inexperienced user of Microsoft SQL Server, and will guide you how to install Microsoft SQL Server and how to use SQL Query Analyzer to create query to generate accounting reports.

A basic understanding of relational database concepts will be advantageous, but is not assumed, as it is covered in this book. It is also not assumed that the reader of this book

has any experience working with SQL, but will be helpful if you already have the knowledge.

What You Need To Use This Book

You will need a copy of Microsoft SQL Server (at least version 7.0 and above), depending on the type of operating system installed in your workstation. In our exercise, Microsoft SQL Server 2000 for Personal Edition is used. Your workstation can be Windows 98, Window NT 4.0, Windows 2000 and Windows XP if you wish to install Microsoft SQL Server 2000 for Personal Edition.

All code and samples in this book were developed and tested on workstation running Windows XP Professional Edition (SP2).

Conventions

To help you in better understanding this book, different typefaces is used to differentiate between SQL code and regular English, and also help you to identify key concepts.

Text that you will type on your screen should appear in courier new type.

How It Works

After trying out the queries, there will be a further explanation, to help you relate what you have done to what you have just learned.

Chapter 1

Database Design

Database

A database is a place where data are stored in columns, and rows in a table, just like a spreadsheet, a database consist of one or several tables. A table consists of many columns, known as fields, and each field consist of many rows, called records. Data stored in a table, can be retrieved, updated or even deleted through executing a set of instruction to a database. This set of instruction is what we call SQL statement.

When the first database was created, its design was not in perfect form. The model of the design was to store data in a single stream of bytes. This is known as a flat-file database. A flat-file database is inefficient, given the lack of scalability and storage capacity.

Relational Database

A relational database model is designed to contain several tables that can be joined together via the use of common related fields. The link of two or more tables is achieved through the use of primary key and foreign keys, known as a relationship. The advantage of a relational database over a flat-file database is its ability to store data in different tables, with minimal duplication.

Primary Key (PK)

A primary key is an identifier that uniquely identifies a record stored in a table. By assigning a primary key to a particular field in a table, we can uniquely retrieve, update or delete certain records from a table. A primary key, can relate to other primary key created in another table. A primary field cannot be null, means it must be populated with value. A user cannot insert a value in a primary field twice, as a primary field is a unique field, and it cannot contain two rows of records with the same value.

Foreign Key (FK)

A primary key is known as a foreign key, if it links to a primary key of another table. A value entered in a foreign field, should be the same value entered in the primary field of another table. You could not enter a value as a foreign key that are not initially entered or exist in a primary field of another table.

Normalization Concept

Normalization is a process that shows the method or way of designing a well-structured database. Under normalization methodology, we can restructure database by simply following the below main three steps:

- 1) First Normal Form
- 2) Second Normal Form
- 3) Third Normal Form

1) First Normal Form

In the first normal form, a database designer is required to identify the type and group of data that each data item will fall in, and then decide which data should be used as the basis of creating individual table to contain them. Let's take an example of creating a phone book database. A phone book, generally consist of name, date of birth, address, phone number, place of work, and other details. We know that our main item data is Name and Location. So, we create a table called Name_T to store the name, date of birth, and phone number of each individual. We also create a table called Location_T to hold data on address and place of work.

Our next task, under the first normal form, is to eliminate repeating groups of data. We know that, under the Name_T table, it is very likely that two or more person, may share the same name, and as for the address, it is possible that more than one person could be staying in the same place, therefore we could end up typing the same name or address

twice in each of the tables. In order to ensure there is no duplication of data in each table, we need to identify a particular field to be a primary key.

A primary key is a unique identifier that identifies particular records in a table, and it ensures that a value entered in its field can never be re-entered twice. This enforces data integrity and consistency. In our Name_T table, we assign the field Name to be a primary field, and set it as primary key, and change the fieldname Name to Name_ID. We then, set the field address as a primary key, changing its fieldname address to Address_ID. We then create another table called, Customer Details_T table to store the Name_ID and Address_ID field. By assigning the Name and Address as a primary key, we can now have more than one record that shares the same name and location.

2) Second Normal Form

No other non-key field is independent of the primary key. We must ensure that all existing fields in a table must depend on the primary key. We know that the Name_T table, contains the date of birth field, and it is possible that more than one person has the same date of birth, thus, we need to create a separate table specifically to hold the date of birth data, and we rename the date of birth to DOB_ID, and set it as a primary key.

3) Third Normal Form

When we reached the second normal form, we almost complete normalizing our database structure. In the third normal form, it's basically ensuring that all non-key fields are now fully dependent on the primary key. We identify one more field that brings us to our third normal form. We could have more than one person working in the same place, thus, it is logical to create the field place of work as a separate table. We rename the existing fieldname place of work to POW_ID, giving reference to a primary key created in a new table called Place_of_Work_T table.

What is SQL?

SQL, an abbreviation for Structured Query Language, is a language used to execute a set of instruction directed to a database. When you go to an auto-teller machine, to withdraw money, you need to press certain button, to instruct the machine what to do, when you go to the Internet, you use your keyboard or mouse to navigate or search for your favorite website, you are telling your machine what to do. All this are possible with the help of SQL.

It is a universal language that receives instruction from a “front-end” object that will then compile and send the instruction back to a “back-end” object. The front-end object is an application tool, such as VB, C++, and the back-end object, is a database system, that helps to store data. The instruction received from a front-end application, generally perform the following task:

- 1) Select existing data
- 2) Insert new data
- 3) Update existing data
- 4) Delete existing data

SQL is a language governed by the American National Standards Institute (ANSI), a standard committee that consists of database experts from industry and software vendors. Thus, SQL is a universal and open language, meaning that, it is not owned by any industry.

Transact-SQL

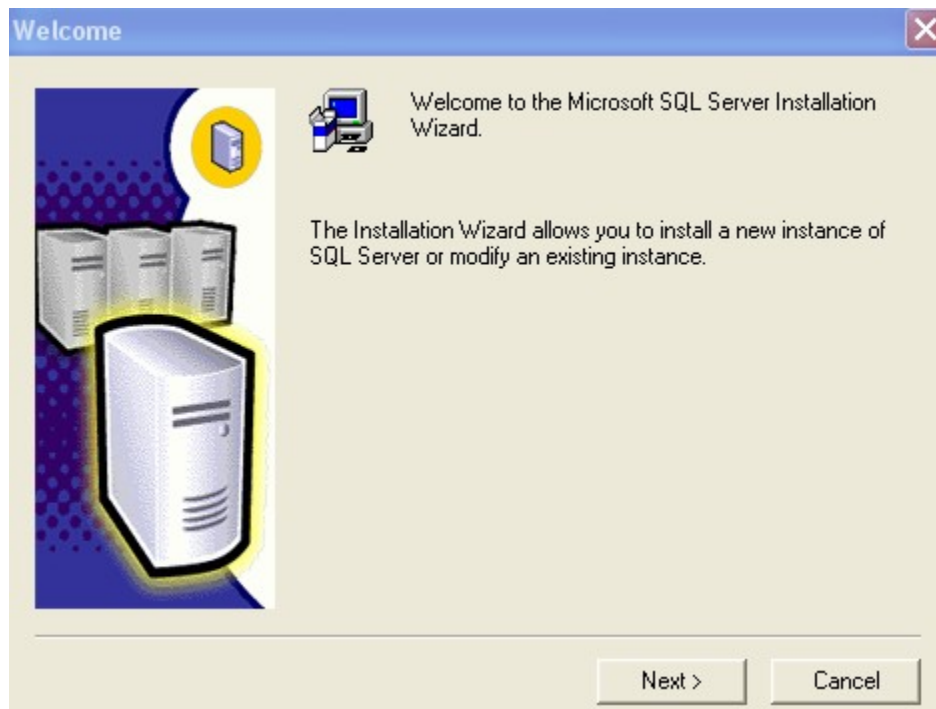
DBMS, or database management system, is a software product that holds and store data. A number of famous DBMS worth noting, are, IBM DB2, MySQL, Sybase Adaptive Server, Oracle, Microsoft Access, and Microsoft SQL Server. These various DBMS, would have their own type of SQL version, generally differ in terms of syntax and features, but, they all complied to the American National Standards Institute (ANSI) SQL Standard.

In our exercise contained in this book, we will be using Microsoft SQL Server as our DBMS in employing the use of Transact Structured Query Language (T-SQL), Microsoft's version of SQL.

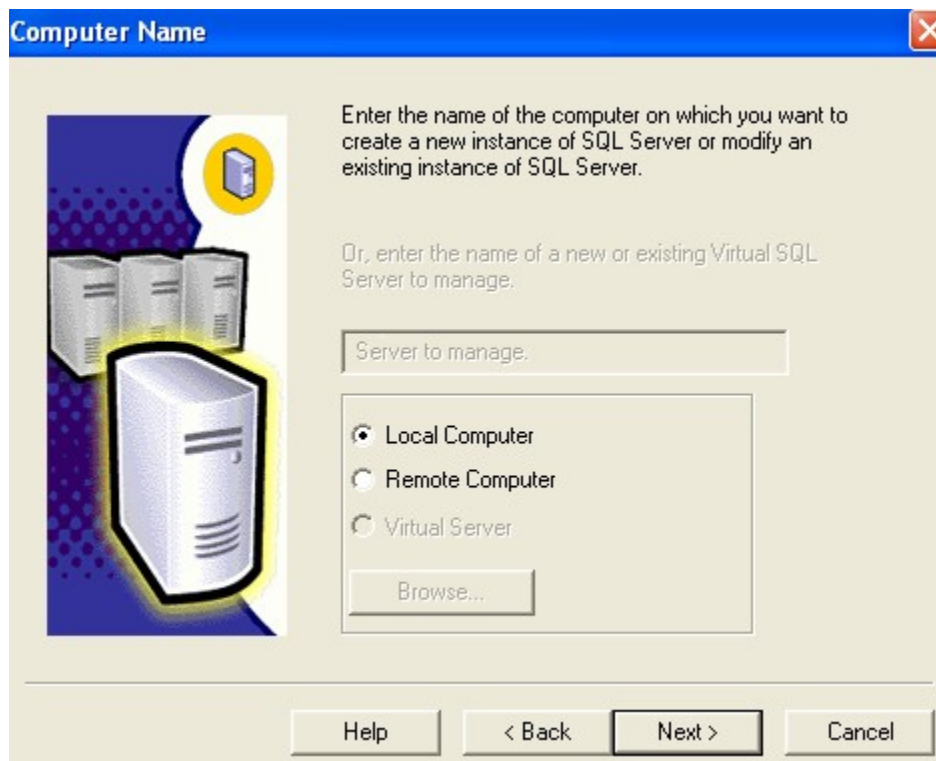
Installing Microsoft SQL Server 2000 (Personal Edition)

In order to try out some of the query that we are going to build in subsequent chapter, we need to install the Microsoft SQL Server, at least version 7.0 or higher. For the purpose of our case study, Microsoft SQL Server 2000 for Personal Edition would be used. You can choose to install other version in your workstation, but you need to check the minimum requirement before you begin installing other version of Microsoft SQL Server.

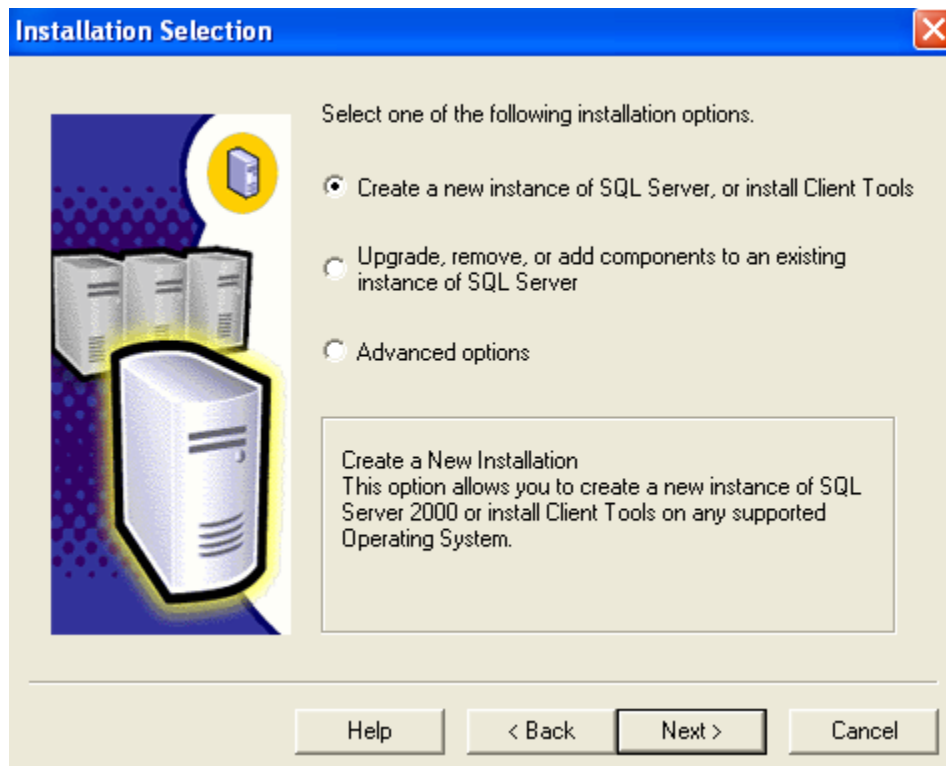
1. To install Microsoft SQL Server 2000 (Personal Edition), insert the SQL Server 2000 CD. If Auto run is not enabled, double-click the Autorun.exe program to begin the installation process.
2. Next, you will see a pop-up Welcome screen that will lead you to installing Microsoft SQL Server. Click on the Next button to proceed to the next step.



3. The next step allows us to select the name of the computer on which you want to install Microsoft SQL Server. By default, the installation Wizard will select the Local Computer option. As this is where we want to install our Microsoft SQL Server, we will accept the default on this screen.



4. Click on the Next button to proceed to the Installation Selection screen. Under this screen, we will accept the default option, which will create a new instance of SQL Server in your selected local computer. Click on the Next button to proceed to the next dialog box.



5. In the User Information screen, by default the name and company name will be automatically filled with the same information you have given when you first installed your operating system. If you prefer, you can change the name and company name before you click on the Next button. Click on the Next button once you have changed the name and the company name.

User Information

Enter your name below. It is not necessary to enter a company name.

Name:

Company:

< Back Next > Cancel

6. In this screen, you will be required to read the terms and condition of the License Agreement. Press the Page Down key to see the rest of the agreement. Once you have read and agreed to the License Agreement, proceed to the next step by clicking on the Yes button.

Software License Agreement

Please read the following License Agreement. Press the PAGE DOWN key to see the rest of the agreement.

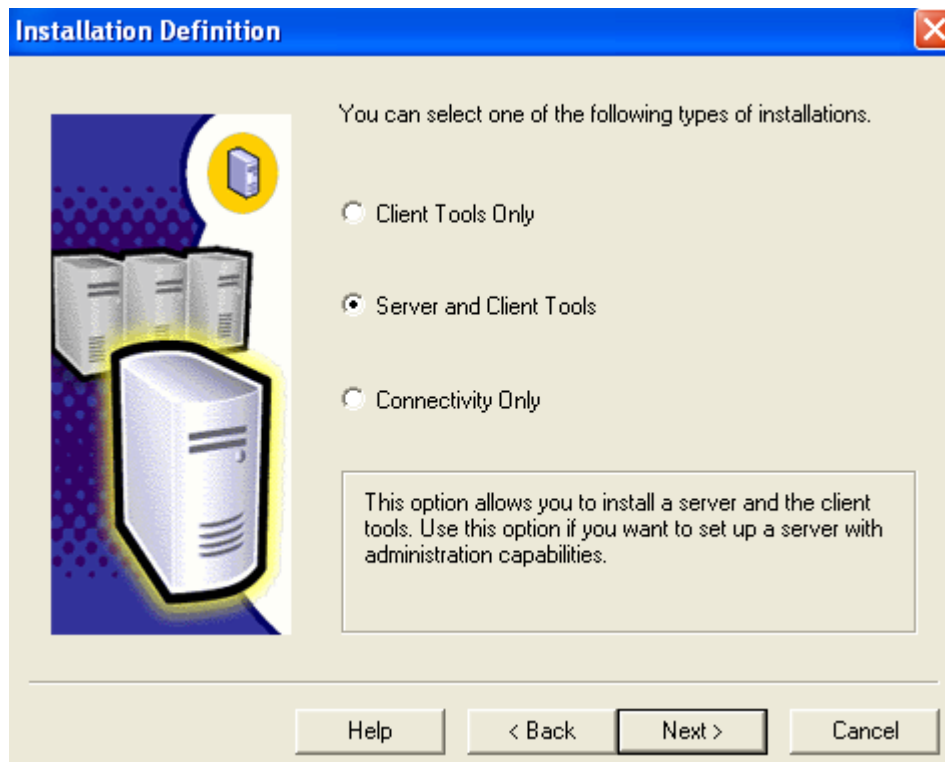
ADDENDUM TO THE MICROSOFT END USER LICENSE AGREEMENT FOR MICROSOFT SQL SERVER 2000

The software accompanying this Addendum, Microsoft SQL Server Personal Edition (the "Client Software") is provided to you for use under the terms and conditions of the end user license agreement you acquired with Microsoft SQL Server (Standard or Enterprise Edition) (the "EULA"). Please refer to the EULA for license rights and requirements associated with Client Software. The Client Software is deemed part of the Product (as defined in the EULA), and as such, if you do not have a validly licensed copy of the Product, you are not authorized to use the Client Software. Any capitalized terms used in this Addendum shall have the same meaning as set forth in the EULA, unless otherwise set forth in this Addendum. All terms and conditions of the EULA remain in full force and effect.

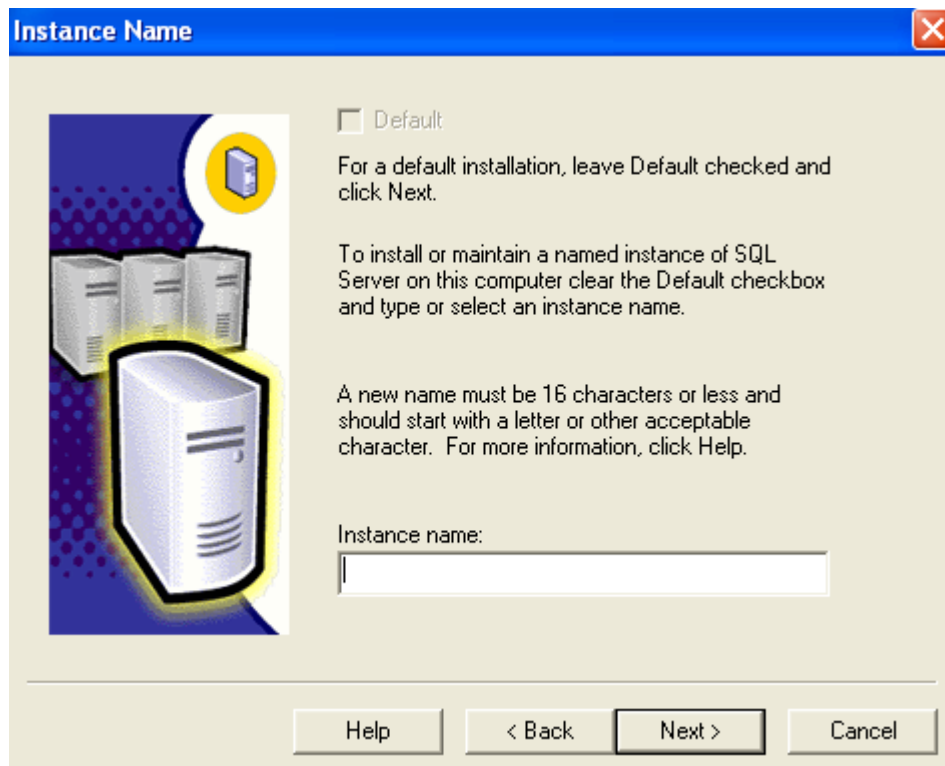
Do you accept all the terms of the preceding License Agreement? If you choose No, Setup will close. To install Microsoft SQL Server 2000, you must accept this agreement.

< Back Yes No

7. Under the Installation Definition screen, you need to select the type of installation that you want to install in your local computer. For our purpose, we will accept the default second option, as we want to make use of the Server and Client tools with administration capabilities. Click the Next button to move to the next screen.



8. The Installation Wizard will detect any installed version of SQL Server on your desktop. If this is your first installation, the default checkbox will be checked by default. If you have previously installed SQL Server, the default checkbox would be grayed out, and you would have to give an instance name for this current installation. The new instance name must be 16 characters or less and should begin with a letter or other acceptable character. Proceed to our next screen once you have given an instance name.



9. On this screen, you are required to select the type of setup. For our purpose, we would want to customize our installation, so let's choose the third option. By default, the program files and data files are directed to windows C drive. If you prefer to relocate the installation folder, you can do so by clicking on the Browse button. Click the Next button once you have completed this step.

Setup Type [X]

Click the type of Setup you prefer, then click Next.

☐ Typical Installed with the most common options. Recommended for most users.

☐ Minimum Installed with minimum required options.

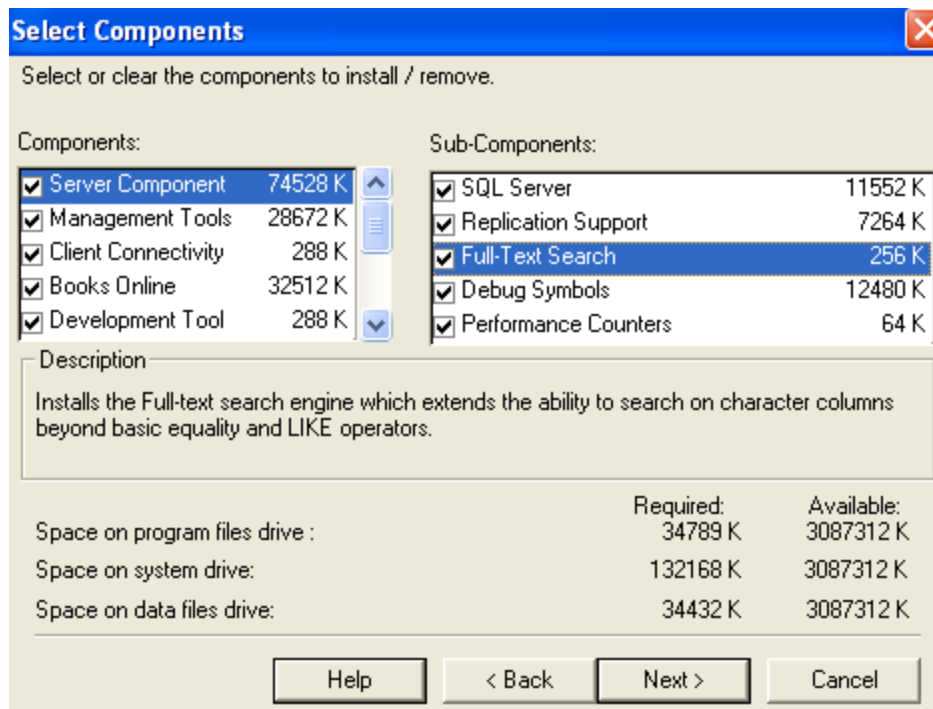
☒ Custom You may choose the options you want to install. Recommended for advanced users.

Destination Folder

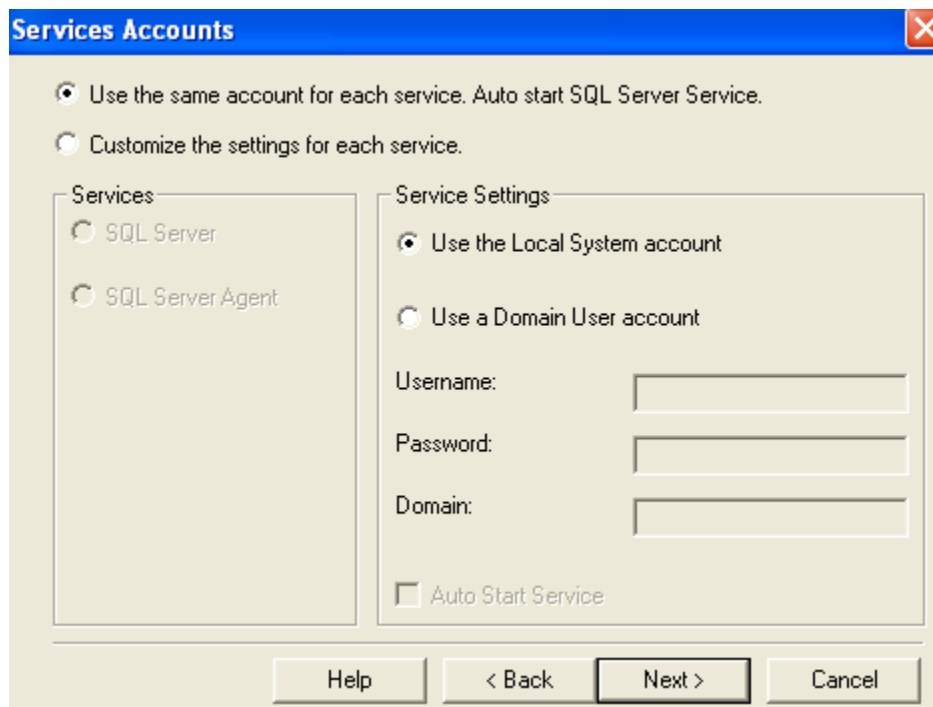
Program Files	Browse...	C:\Program Files\Microsoft SQL Server
Data Files	Browse...	C:\Program Files\Microsoft SQL Server

	Required:	Available:
Space on program files drive:	34657 K	3088336 K
Space on system drive:	107945 K	3088336 K
Space on data files drive:	34432 K	3088336 K

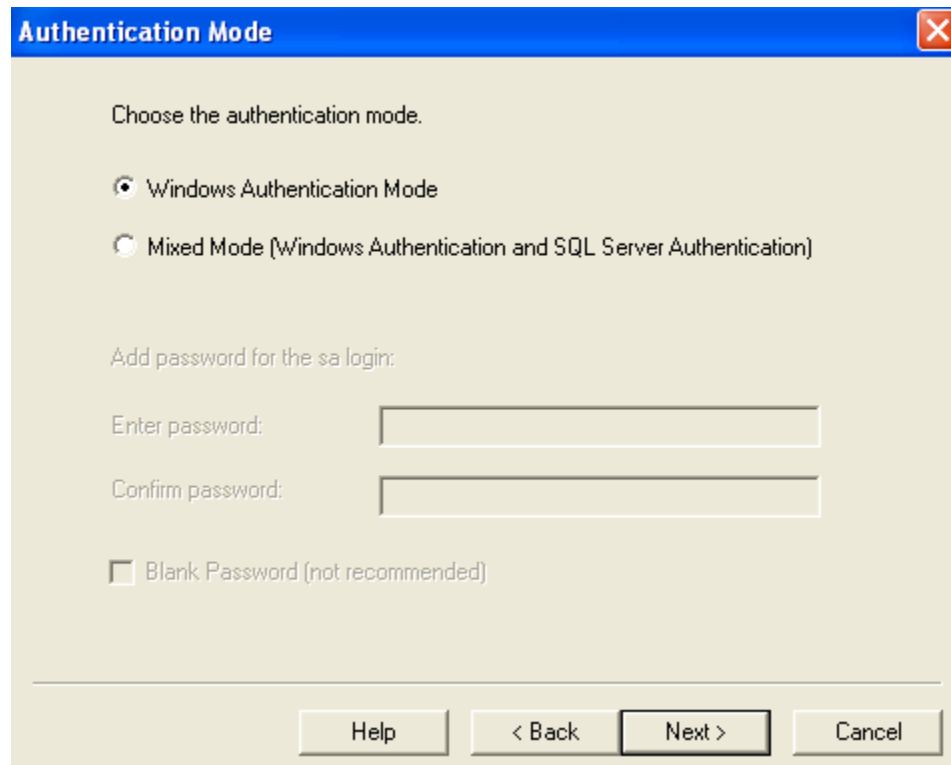
- Under the Select Component screen, you can view all the main and sub-component that allows you to select and install in your local computer. For our purpose, we will select all components; together will its individual sub-components. Under the Description label, you can view the function of each sub-component by checking on each of the sub-component checkbox. Click next, once you have selected all components.



11. Under the Services Accounts screen, you will have the option of selecting different account for each service or assigning the same account for each service. For our purpose, we choose the first option, as this would eliminate any unnecessary problem that a domain user account would normally encounter. Click on the Next button to proceed to our next screen.



12. This screen allows you to configure the type of authentication mode that you prefer in order to gain access to SQL Server. If you choose the first option, SQL Server will use windows domain user account to verify the authenticity of the user, before granting access to SQL Server. If you choose the second option, SQL Server will require an additional level of validation that would require a user login password. For our purpose, we will select the windows authentication mode option.



Authentication Mode

Choose the authentication mode.

☒ Windows Authentication Mode

☐ Mixed Mode (Windows Authentication and SQL Server Authentication)

Add password for the sa login:

Enter password:

Confirm password:

☐ Blank Password (not recommended)

Help < Back Next > Cancel

13. Under the Collation Settings, will allow you to specify a set of guidelines that determine how information is being compared and collated in SQL Server. For our purpose, we select the SQL Collations option.

Collation Settings

Windows Locale

Change the default settings only if you must match the collation of another instance of SQL Server or the Windows locale of another computer.

☐ Collation designator:

Sort order

☐ Binary

☐ Case sensitive

☐ Accent sensitive

☐ Kana sensitive

☐ Width sensitive

☒ SQL Collations (Used for compatibility with previous versions of SQL Server).

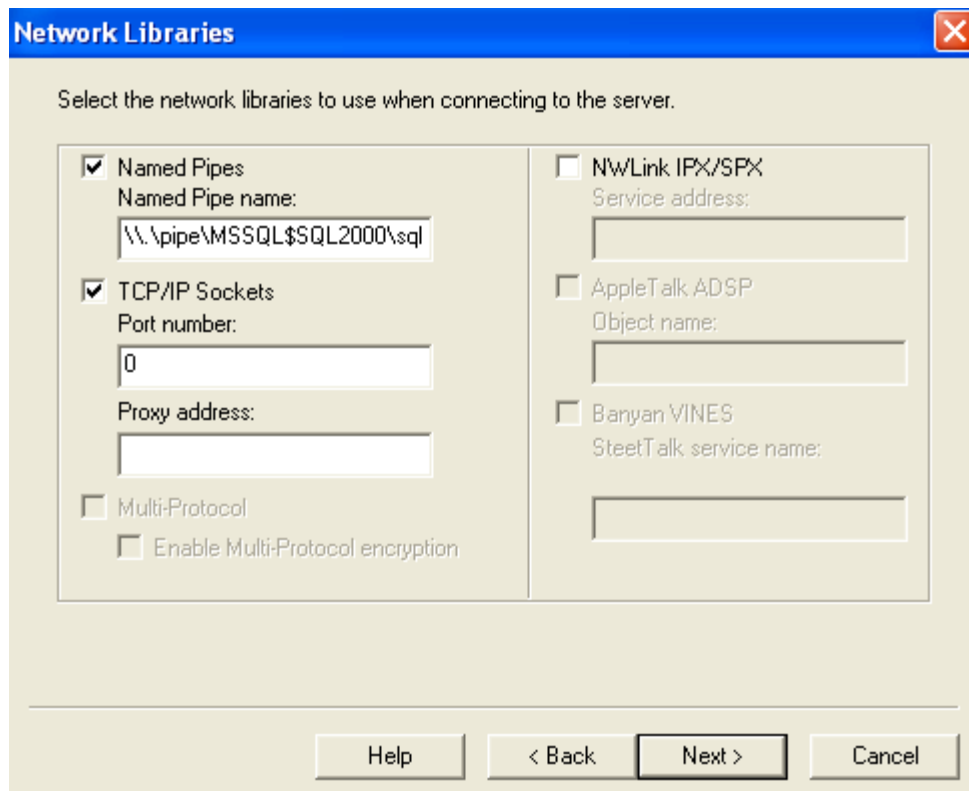
Strict compatibility with version 1.x case-insensitive databases, for use with the 850 (M

Dictionary order, case-sensitive, for use with 1252 Character Set.

Dictionary order, case-insensitive, for use with 1252 Character Set.

Help < Back Next > Cancel

14. SQL Server uses network libraries to pass network packets of information between SQL Server and its clients. By default, SQL Server is configured to listen to packets from clients via the Named Pipes shown on below textbox. If you are installing a named instance, the instance name would appear on the textbox, with a 0 port number specified. If you are installing SQL Server for the first time, the port number 1433 is specified, by default. Once you have specified the Named Pipe name and the port number click on the Next button to move to our next step.



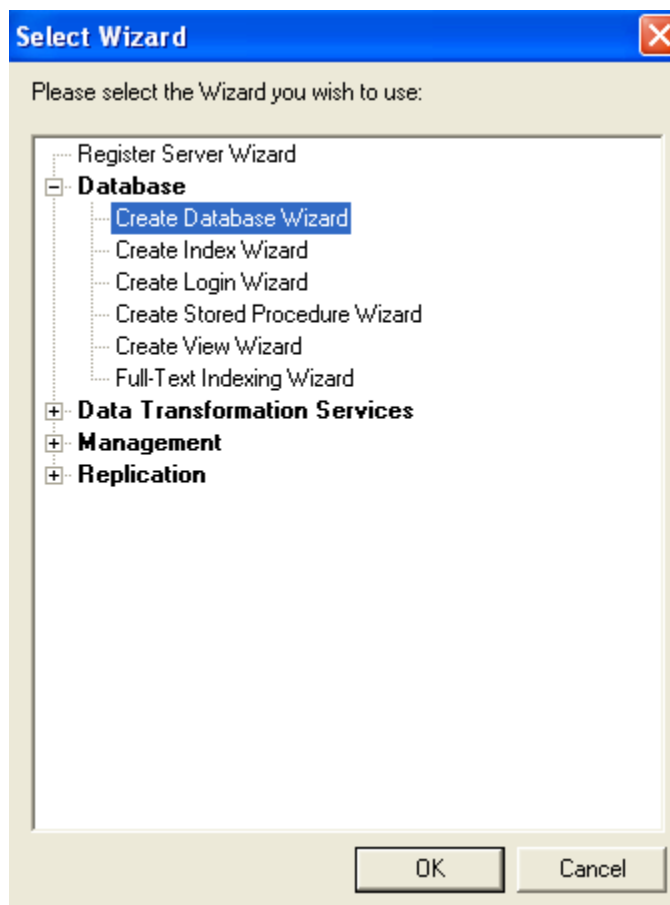
15. This is the last setup screen that will begin installing your SQL Server 2000 in your local computer. Click the next button to begin the installation, or click the Back button if you want to change some of the previous setting.



Creating a Database

Before we begin to discuss how to create tables, we need to create a database first.

1. First, open your Enterprise Manager
2. Next, expand the Microsoft SQL Servers root, SQL Server group and then the instance of SQL Server 2000 that you installed.
3. Click on the Database folder, and select Tools from the Menu bar, and then click on Wizard.
4. Expand the Database, and then select Create Database Wizard. You would see a screen as shown below. Next Click on the OK button to proceed to the next step.



5. The next step of the Wizard will show you a summary of the installation guide that we will go through in the following setup process. Click the Next button to proceed to the next screen.



6. In our next screen, we specify a name for our database. Give a name of AIS, which represent the abbreviation of Accounting Information System in the Database name textbox. We can change the default location directory for the database and transaction log file, if you want to, by clicking on the three dotted button beside the textbox. For our purpose, we shall accept the default location for both database and transaction log file. The transaction log file is used to record all transactions performed in your database and can be used for recovery purposes.

Create Database Wizard - (local)

Name the Database and Specify its Location

Specify a name for the database, following the rules for identifiers. Type or select a location for the database.

Database name:

Database file location:
 ...

Transaction log file location:
 ...

< Back Next > Cancel

- Next, we specify the name and the size of our database files. By default, 1 megabyte size is allocated, but we change it to 5 megabytes for our current database. Click on the Next button to proceed to the next screen.

Create Database Wizard - (local)

Name the Database Files

Specify the name of one or more files in which the database is contained. Specify the initial file size for each of the files.

Database files:

File Name	Initial Size (MB)
AIS_Data	5

< Back Next > Cancel

8. We want to allow our database to grow automatically and with unrestricted file size, thus we accept the default options as specified on the below screen. Proceed to the next step by clicking on the Next button.

The screenshot shows a Windows-style dialog box titled "Create Database Wizard - (local)". The main heading is "Define the Database File Growth", followed by the instruction: "Specify whether the database files should grow automatically, or grow only when you enlarge them." There is a yellow circular icon with a database cylinder and a sunburst. The dialog contains two main radio button options: "Do not automatically grow the database files" (unselected) and "Automatically grow the database files" (selected). The "Automatically grow" option is expanded to show two sub-options: "Grow the files in megabytes (MB):" with a spinner box set to "1", and "Grow the files by percent:" (selected) with a spinner box set to "10". Below these is a "Maximum file size" section with two radio button options: "Unrestricted file growth" (selected) and "Restrict file growth to MB:" with a spinner box set to "3011". At the bottom are three buttons: "< Back", "Next >" (highlighted), and "Cancel".

9. We are required to specify the size for our transaction log file. The default 1 megabyte is sufficient for us, thus proceed to the next step by clicking on the Next button.

Create Database Wizard - (local)

Name the Transaction Log Files

Specify the name of one or more files in which the transaction log is contained.
Specify the initial file size for each of the files.

Transaction log files:

File Name	Initial Size (MB)
AIS_Log	1

< Back Next > Cancel

10. Similarly, we also need to specify the size for our transaction log files. By default, the below setting is specified and it should be sufficient for our purpose. Click on the Next button to proceed to the next screen.

Create Database Wizard - (local)

Define the Transaction Log File Growth

Specify whether the transaction log files should grow automatically, or grow only when you explicitly enlarge them.

☐ Do not automatically grow the transaction log files
☒ Automatically grow the transaction log files

☐ Grow the files in megabytes (MB): 1
☒ Grow the files by percent: 10

Maximum file size

☒ Unrestricted file growth
☐ Restrict file growth to MB: 3011

< Back Next > Cancel

11. This is the last screen in the Database Wizard. You can now begin to create your first database by clicking on the Finish button, or if you decide to change your previous setting, you can do so by moving backward. Click the Finish button once you are ready to create your database.



Summary

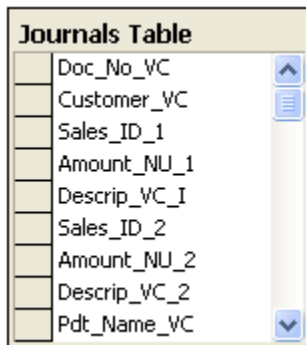
In this Chapter, we have learned what are a database, how to create relationship between tables, and the concept of normalization. We then learned how to install SQL Server and creating a database.

To summaries, in this chapter, we have discussed:

- The advantages of relational database versus a flat-file database
- Normalization Concept (First Normal Form to Third Normal Form)
- The use of Primary and foreign keys
- What is SQL, and its functionality
- How to install SQL Server 2000 (Personal Edition)
- How to create database with Database Wizard

Chapter 2

Developing the Journals Table



Doc_No_VC
Customer_VC
Sales_ID_1
Amount_NU_1
Descrip_VC_I
Sales_ID_2
Amount_NU_2
Descrip_VC_2
Pdt_Name_VC

Normalizing Journal Table

This chapter will illustrate how an accounting database is being designed. We will begin to adopt the normalization concept to break our table into several tables (First Normal Form to Third Normal Form). Above table contains a list of fields that holds information that consist duplication of data. This table is not normalized and we will begin to identify the key elements that are to become the primary key in a table and as a foreign key in a separate table.

The Journal table would contain the double-entry information of an entry performed by a user. MaxCorp is in the business of trading computer hardware and software. It needs to have a system to keep track on all its business transaction, and also a system that can produce relevant accounting reports on every close of each month. Dave, the Accountant, would normally raise an invoice to a customer on a Sales Form screen, and information such as date of invoice, product, customer name, pricing would be entered on the face of the Sales Form screen, and would be stored in the above fields created in the Journal Table.

First Normal Form

We understand that, MaxCorp's customer may receive several invoices in a week from its billing department, and some, merely once in a year, depending on the number of purchases made by its customer. As shown on the table above, currently the Journal Table can only fit in two transactions for each customer and Dave would have to insert a new line of records, having entering the Customer name again, if the same customer buy from MaxCorp for the third time. This is inefficient, as Dave, is repeating groups of data, by entering the customer name twice. To begin our first step of normalization, we will break the Journal Table into a Sales Table and a Journal Table. We will then, assign an Inv_ID_VC fieldname as the primary key in the Sale Table.

By creating a separate Sales Table, Dave would be able to raise as many invoice to a customer without having to repeat its customer name again. By creating a separate Sale Table, we are eliminating duplication of data and making use of the storage space of each field more efficiently, as each customer's name are only created once.

Second Normal Form

We aware that, a customer can buy more than one type of product from MaxCorp, thus, we need to further break up the Journals Table into a Product Table, a Sales Table and a Journals Table. We will assign the fieldname, Pdt_ID_VC as the primary key in the Product Table, and as a foreign key in the Sales Table referencing to a particular product residing in the Product Table. We are establishing a many-to-many relationship between the Sales and Product Table, by connecting these tables via a common field, where a customer may buy different types of products from MaxCorp, and a product could be purchased by different types of customers.

Third Normal Form

So far, we have created two tables out of Journals Table, the Sale Table and Product Table. At this point, our task is incomplete, Dave has voiced his concern on the issue of data integrity. He is worried, especially on numerical fields that hold important figures, vital to the preparation of logical and comprehensive financial reports to MaxCorp's management. He wants to have a database that runs on a real-time basis, where whenever an invoice is issued, an entry would be automatically posted in the Journals, when a collection is made from a customer or payment made to a creditor, an entry would also be posted in the Journals, without having the need for manual entry, simply said, a real-time processing system.

Having understood Dave requisition, we need to redefine the structure of the database design; we know that, he wants a real-time processing for all the posting of journals, the self-creating double-entry records for each level of order processing performed by Dave.

After a brainstorming session with Dave, we understood that, he wants a database that consist a group of tables, representative of each accounting module, to have a direct interface with the Journals Table. The Journal Table is the central repository that records all back-end double-entries performed by the client-application for all order processing transaction, and all front-end transaction performed by the user.

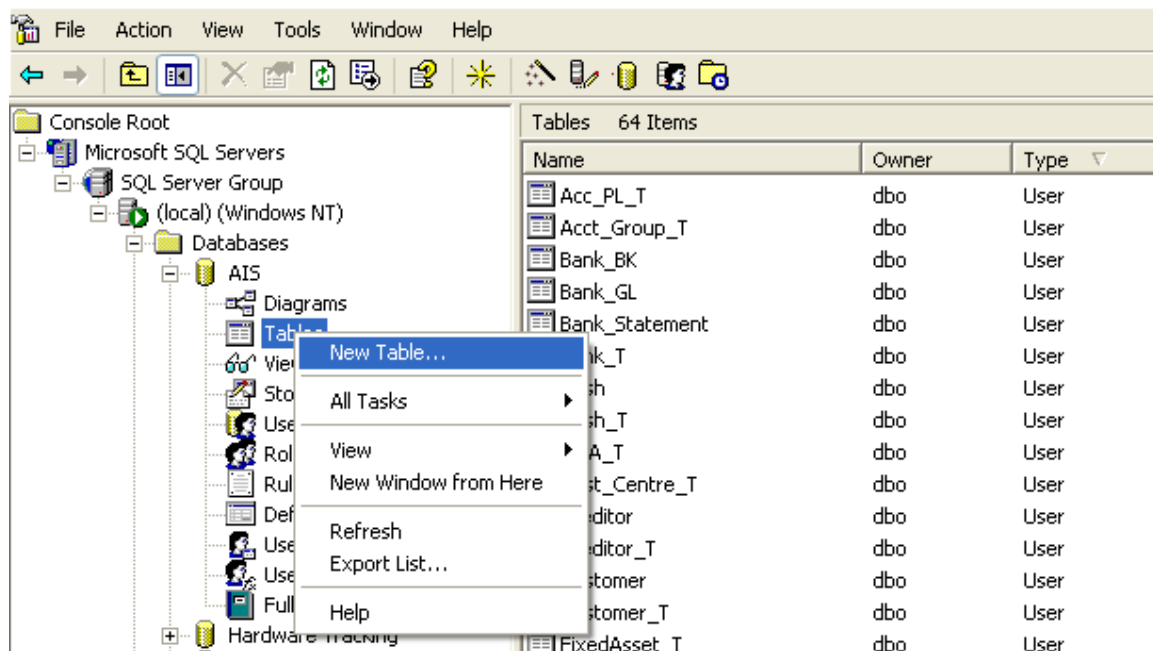
Dave explained, that, for each double-entry performed, a debit, and a credit entry would take place, example, if MaxCorp need to bill a customer, a debit and credit amount would be posted in the Journal Table, debiting an X amount in a Debtor account and crediting the corresponding amount in the Sales account. When he collects from its customer, and then making payment to its creditors on its purchase due, all these transaction would be recorded also as a double-entry in the Journals Table.

With the above knowledge shared by Dave, we need to break the Journals Table further, into another table, known as a Chart of Accounts Table (COA). This Table allows Dave to create account code (GL_ID) that uniquely identify each elements of accounts name, that eventually make up as one of the elements in the COA Table.

Before any transaction is being posted as a double-entry in the Journals Table, user would need to drill-down the dropdown list box, to select the appropriate account code (GL_ID), if it does not exist, user would be prompted to create the account code beforehand in the COA Table. Some of the posting can be pre-defined by the user during set-up stage, for example, double-entry for a customer billing can be pre-defined to debit and credit to a fix Sales and Debtors account, and some transaction would require user selection of account code during order processing process.

A list of important fields with their data type attribute, has been identified in the COA Table, which would have the GL_ID field assigned as primary key, referencing the Journal Table as a foreign key.

Now, Create the COA Table by first Opening the SQL Server 2000, then, select Database | AIS | Table, right click on your mouse, and select New Table as illustrated on below snapshot:



Key in the following fieldnames, data-type, length, and its Allow Null attributes in the COA_T Table as shown below:

COA_T				
	Column Name	Data Type	Length	Allow Nulls
🔑	GL_ID	int	4	
	GL_Name_VC	varchar	20	
	BS_Category_VC	varchar	20	
	Segment_VC	varchar	10	✓
	Status_BT	bit	1	

Designing Chart of Accounts Table

The GL_Name_VC field would hold the different type of account name, in which are uniquely identified by an account code located in the GL_ID field. The BS_Category_VC field would hold the types of accounts category, to which each individual accounts name will fit in. Segment_VC field stores the different type of business units to be created in MaxCorp. Lastly, user has the option to freeze the account code, by changing its status from active to a non-active account code, preventing the selection of the account code for posting purposes.

Key in the following fieldnames, data-type, length, and its Allow Null attributes in the Journal_T Table as shown below:

Journal_T				
	Column Name	Data Type	Length	Allow Nulls
	Doc_No_VC	varchar	10	
	GL_ID	int	4	
	Inv_ID_VC	varchar	10	
	Amount_NU	numeric	9	
	Descrip_VC	varchar	10	
	Date_DT	datetime	8	
	Period_TI	tinyint	1	
	Year_SI	smallint	2	

Designing Journal Table

The Journal table would have a field, called Doc_No_VC that will hold a set of double-entry records performed by a user or by an auto-generated entry performed by the system. Each journal entry would require the following information to be provided, in the remaining fieldnames, the GL_ID field, that will capture the account code of each journal entry, the invoice number (under Inv_ID_VC field) that is assigned as a foreign key, referencing to a particular set of records in the Sale_T table, amount of each transaction posting (under Amount_NU), date of the transaction posting (under Date_DT), Description, Period and Year.

Key in the following fieldnames, data-type, length, and its Allow Null attributes in the Sale_T Table as shown below:

Sale_T *				
	Column Name	Data Type	Length	Allow Nulls
	Doc_No_VC	varchar	10	
	Pdt_ID_VC	varchar	10	
	Cust_Name_VC	varchar	10	
	Inv_ID_VC	varchar	10	

Designing Sales Table

This table contains information on sales transacted with each customer, holding information on product, customer and invoice number. We assign the Inv_ID_VC field as a primary field, as invoice number would be the best candidate for being a unique identifier and as a foreign key in the Journal_T table that will uniquely identify a row of records related to a transaction's double-entry located separately in the Journal Table. The Sales Table merely hold a number of key identifier that are linked to several other tables, namely, the Pdt_ID_VC would link to a Product Table, giving further details, on name, description, category and supplier of each product items. We shall explore further on this table as we develop further in our database design under Chapter 5.

Product_T *				
	Column Name	Data Type	Length	Allow Nulls
🔑	Pdt_ID_VC	varchar	10	
	Pdt_Name_VC	varchar	10	

Designing Product Table

We will assign Pdt_ID_VC as the primary key field for Product Table, which holds key information of each product's name. We will get to know the usefulness of account set field, as we touches on Chapter 3 onwards.

Summary

In this Chapter, we have created the Journal Table, and then we learned how to normalize this table into several tables, going through the normalization process, from first normal form to third normal form. In Chapter 3, we will discuss how to normalize the Product Table further, giving us an insight on the types of fields needed in a Product table.

To summaries, in this chapter, we have discussed:

- The Normalization Concept (First Normal Form to Third Normal Form)
- The use of Primary and foreign keys
- The purpose of a Chart of Accounts and its relevant fields
- The purpose of a Journal Table and its relevant fields
- The purpose of a Sales Table and its relevant fields
- The purpose of a Product Table and its relevant fields

Chapter 3

Developing the Inventory Table

Product_T *				
	Column Name	Data Type	Length	Allow Nulls
🔑	Pdt_ID_VC	varchar	20	
	Pdt_Name_VC	varchar	30	
	Pdt_Descrip_VC	varchar	40	✓
	Pdt_Category_VC	varchar	20	
	Cred_ID_VC	varchar	20	
	Pdt_AcctSet_VC	varchar	20	

Normalizing Inventory Table

As discussed in Chapter 2, the Product Table was created to keep track on each product item's name, description, supplier code and its account set code. This table is still not in perfect form, as Dave, would also want to keep track on the movement of each of the product sold, the costing method applied on each product items and the pricing of each of its product items. We will again, follow the step-by-step normalization process to further identify key elements that should be broken down further away from the Product Table.

First Normal Form

We know that, not all products in MaxCorp are purchased from the same source of supplier, it could be a product sourced from different supplier or a supplier could also be supplying MaxCorp more than one product type. It is logical, at this point, to create a separate table for supplier, referencing the Product Table via a foreign key named Cred_ID_VC, being a primary key assigned in the Creditor Table.

Second Normal Form

At this stage, we have further identified another key element, the product category, as MaxCorp will also need to group its sale by category, in order to better analyze the marketability and profitability of each product type. We branch out the Pdt_Category_VC field, to create a separate table, named, Product Table_Category, assigning the field Pdt_Category_VC, as the primary key, having linked to the Product Table, as a foreign key.

Third Normal Form

We have not created any field to capture the quantity movement and balance of each product items. Create a table named, Stock_Movement Table to capture the physical movement of each product sold by MaxCorp, and a table named, Stock_Balance Table to monitor the current quantity balance of each product items.

Inventory Table

Stock_Movement_T				
	Column Name	Data Type	Length	Allow Nulls
	Doc_No_VC	varchar	20	
	Date_DT	datetime	8	
	Pdt_ID_VC	varchar	20	
	Qty_NU	numeric	5	
	Unit_Price_TI	tinyint	1	
	Descrip_VC	varchar	20	

Designing Stock Movement Table

This table would keep track of each product movement in MaxCorp warehouse. A row of records would be inserted into this table when MaxCorp delivers products to its customer, receiving incoming goods from its supplier, or even transferring goods between different warehouse locations. In our table, we only maintain one location for storing MaxCorp's goods, and should MaxCorp expands its business further in future, it may establish branches that would require more than one warehouse to store its product, then MaxCorp may need to create a separate table just to store warehousing information. The Descrip_VC field would accommodate the delivery note and goods receive note numbering for each product items moving in and out from MaxCorp's warehouse.

Enter the following fieldnames, data-type, length, and its Allow Null attributes as shown above:


Stock_Balance_T				
	Column Name	Data Type	Length	Allow Nulls
🔑	Doc_ID_VC	varchar	10	
	Pdt_ID_VC	varchar	20	
	Qty_NU	numeric	5	
	Unit_Price_TI	tinyint	1	
	Date_DT	datetime	8	

Designing Stock Balance Table

This table will hold the latest balance of each product items, after accumulating or deducting the previous balance of each product items, to illustrate this further, assuming that, MaxCorp purchase 2 units of item A at \$1.50 each. The system would record this information in this table after updating the Stock_Movement Table. When item A is sold to a customer, the system would then insert a new record in the Stock_Movement Table and will update the balance of item A in Stock_Balance Table simultaneously. For each update in the Stock_Balance Table, the system will first identify the matching product item in the Stock_Balance Table, if it exist, the system would update its quantity and unit price, by replacing its current quantity and unit price, and if, it is a new product item, a new record would then be inserted. This table plays a significant influence on product

pricing. We will have a better understanding, as we discuss further on the costing method field created in the Product AccountSet Table.

Enter the following fieldnames, data-type, length, and its Allow Null attributes as shown below:

Product_AccountSet_T				
	Column Name	Data Type	Length	Allow Nulls
	Pdt_AcctSet_VC	varchar	20	
	Cost_Method_VC	varchar	5	
	Stock_IN	int	4	
	Payable_IN	int	4	
	Shipment_IN	int	4	

Designing Product Account Set Table

The Cost_Method_VC field plays an important role in determining how a product item should be priced, when a product is sold to a customer. At present, we have two alternative in pricing MaxCorp product, the first-in-first-out (FIFO) method and the cost average method. Applying the first method, would price MaxCorp's goods on the oldest price first, as for the latter, at an average price. The quantity and pricing of each product item is referenced to the Stock_Balance Table.

Besides determining the pricing method of its out-going goods, Dave would also want the double-entries to flow into the Journal Table, to record the cost of the out-going goods and the cost of purchasing the goods from its suppliers. To achieve this, we need to assign a default inventory account code to record the cost of in-coming and out-going of each product item. In the Stock_IN field, a clearing account (Payable_IN) would need to be assigned to place its purchase cost temporarily, while awaiting billing from respective supplier and a clearing account (Shipment_IN) is assigned to capture its shipment cost, while pending subsequent billing to its customer.

Enter the following fieldnames, data-type, length, and its Allow Null attributes as shown below:

Product_Category_T				
	Column Name	Data Type	Length	Allow Nulls
	Pdt_Category_VC	varchar	20	
	Category_Name_VC	varchar	30	
	GL_ID	int	4	

Designing Product Category Table

This table contains three important fields, with the last field requiring Dave, to assign two default account code for effecting the double-entries into the Journal Table, each time a user raises an invoice to its customer. We will assign the field Pdt_Category_VC as the primary key, having reference to the Product Table. Firstly, under each product category, Dave would need to assign a default account code for the cost of goods sold amount, to record the pricing cost of its product, secondly, a default account code to record the sales amount of each invoices raised from the client application. We will illustrate this further, using a case study, as we turn to Chapter 11.

Summary

In this Chapter, we have created five Tables that made up the inventory group, the Product_Table, Product_AccountSet_Table, Product_Category_Table, Stock_Movement_T and the Stock_Balance_Table.

To summaries, in this chapter, we have discussed:

- How to normalize the inventory table by breaking up into several tables.
- The purpose of creating GL_ID field in Product Category Table
- The purpose of creating Stock_IN, Payable_IN and Shipment_IN fields in Product_AccountSet Table
- The purpose of creating the Stock_Balance Table
- The purpose of creating the Stock_Movement Table

Chapter 4

Developing the Purchase Table

Purchase Table	
	CreditorName_VC
	Product_1_ID
	Product_2_ID
	Unit_Price_IN
	Date_DT

Normalizing Purchase Table

The Purchase table would hold information on product code, supplier name, unit price and the date of purchase, transacted by MaxCorp. As you can see from the above table,

the field Product_1_ID and Product_2_ID allows Dave to assign two type of product supplied by each of its supplier, while this might comes useful, but, how about those suppliers that only provide one type of product, the second field, would remain idle, and unpopulated. What if a supplier supply more than 2 types of product? Again, this would result in the inefficient use of storage space. To restructure the design of the above table, we again, would run through the normalization procedure, breaking down the table one at a time.

First Normal Form

As discussed previously, we would need to replace the two product_ID fields with a foreign key, referencing a many-to-many relationship between the Product and the Purchase Table. We will name this field, Pdt_ID_VC, which is a primary key assigned in the Product Table. Now, this table would be able to hold as many types of product that are stored in the Product Table. We would also want to truncate the field Unit_Price off the Purchase Table, as this information has already been created under the Stock Balance Table.

Second Normal Form

We understand that, MaxCorp purchase many types of product from its supplier, assuming, if a supplier supply more than two types of products to MaxCorp, it would have to insert another new records in the Purchase Table, having repeating the supplier name twice. This would result in duplication of information, and to eliminate this, we would have to create a separate table specifically to store information on supplier. Thus, we have to replace the CreditorName_VC field with Cred_ID_VC, and assign this field as a foreign key referencing to the Creditor Table.

Third Normal Form

Notice that the above table, does not have a primary key field, and we know that a primary key should be one that uniquely identify a row of record residing in the Purchase Table. We will assign the purchase invoice number as the primary key, as it holds the key to accessing and identifying each individual records maintained in the Purchase Table. Do not forget, we will also need to create a field to hold the journal number, dictating the double-entries for each purchase transaction.

Enter the above Column names, data type, length and allow null attributes for the Purchase Table, shown below:

Purchase Table

Purchase_T				
	Column Name	Data Type	Length	Allow Nulls
	Cred_ID_VC	varchar	20	
	Pdt_ID_VC	varchar	20	✓
	Doc_No_VC	varchar	20	
	Date_DT	datetime	8	
	Status_BT	bit	1	
🔑	Inv_ID_VC	varchar	30	

We will set the fieldname `Inv_ID_VC` as a primary key, which would be a suitable unique identifier to each individual row of records maintained in the Purchase Table, in our case, we shall insert the purchase invoice number in this field. We will create the `Cred_ID_VC` field, and assign this field as a foreign key, referencing to the Creditor Table. We have now, established a one-to-many relationship between Purchase Table and Creditor Table, where one creditor may refer to more than one purchase records residing in the Purchase Table. Similarly, we would also want to assign the field `Pdt_ID_VC` as a foreign key, as one product can have more than one reference to the Purchase Table. We shall include a field to keep track on the payment status of each supplier, with an attribute of 1, denoting a paid status, and 0 for unpaid status.

Designing Creditor Table

Enter the below Column names, data type, length and allow null attributes for the Creditor Table, shown below:

Creditor_T				
	Column Name	Data Type	Length	Allow Nulls
🔑	Cred_ID_VC	varchar	20	
	Cred_Name_VC	varchar	20	
	Cred_Add_VC	varchar	50	✓
	Cred_Contact_VC	varchar	20	✓
	Credit_Term_TI	tinyint	1	
	Cred_Code_IN	int	4	

As mentioned previously, a creditor may have several purchase records residing in the Purchase Table; therefore, we will create a separate table for the supplier of MaxCorp. We set the `Cred_ID_VC` as the primary key for this table and as a foreign key in the Purchase Table. The fieldname, `Credit_Term_TI`, would record the credit term, in days given for every purchase made by MaxCorp. This field will hold the key, to calculating the ageing period of each purchase invoice. We shall discuss more on this area, when we move towards Chapter 10. The fieldname `Cred_Code_IN` allows Dave to assign the default account code for each supplier. This will be useful, when Dave begin to process its suppliers' invoices. As for now, we shall maintain one account code across all suppliers. The amount posted for each invoice will interface with the clearing account, as defined in `Product_AccountSet_Table` under the fieldname: `Payable_TI`.

Summary

In Chapter 4, we have learned how to break down the Purchase Table into three separate tables, the Purchase Table, the Product Table and the Creditor Table.

To summaries, in this chapter, we have discussed:

- How to normalize the Purchase Table by breaking up and eliminating some duplicated field.
- The process and motive of creating a separate table for Supplier
- The purpose of creating Credit_Term_TI field in Creditor Table
- The purpose of creating Cred_Code_IN field in Creditor Table

Chapter 5

Developing the Sales Table

Sale_T *	
	Cust_ID_VC
	Pdt_ID_VC
	Doc_No_VC
	Date_DT
	Status_BT
	Inv_ID_VC

Normalizing Sale Table

In Chapter 2, we have learned how to create Sale Table for MaxCorp, now; it is time to break down this table as we discover more missing elements that we will include in the Sale table, as we progress further from here.

First Normal Form

It is generally logical, to have repeated sale coming from any of MaxCorp's customer, and Dave would need to record the billing transaction details in the Sale Table. Notice, the above table has one field created for each customer, and Dave would have to repeat entering the same customer name again if there is recurring sale coming from the same customer. This would significantly consume a great amount of space, and would definitely slow down the performance of your database server. What we witness here, is duplication of unnecessary information. We can reduce the amount of unnecessary space consumption by restructuring the above Sale Table. We can start, by, replacing the

Cust_ID_VC field with an identity fieldname that identify a particular customer name residing in the Customer Table. Before we go into replacing the original field, we need to create a separate Customer Table to store all customer name and details that has trading activity with MaxCorp.

Second Normal Form

Again, the above table would have a field that would be assigned a primary key element, here, we adopt similar concept as what we have achieved in identifying the primary key for the Purchase Table. We will assign the sales invoice number as the default primary key element. Right click on the Inv_ID_VC field, and set it as primary key. We have to link the fieldname Pdt_ID_VC to its primary key, by highlighting the row on the Product Table, left-click on your mouse, while holding it, drag your mouse towards the Sale Table and release.

Third Normal Form

While MaxCorp is cautious in making timely payment to its supplier, it is equally important as well in ensuring that all debts due are collected in time. Currently, MaxCorp do not have field that store credit term allocated for each of its sales transaction, we will create this field to allow the user to keep track on the ageing of all debts due to MaxCorp. We will include this field in the Customer Table.

Enter the above fieldnames, data type, length and the allow nulls attribute in the Sale Table.

Sale Table

Sale_T *				
	Column Name	Data Type	Length	Allow Nulls
	Cust_ID_VC	varchar	20	
	Pdt_ID_VC	varchar	20	
	Doc_No_VC	varchar	20	
	Date_DT	datetime	8	
	Status_BT	bit	1	
	Inv_ID_VC	varchar	20	

As mentioned under the second normal form, we will assign the Inv_ID_VC as the primary key and unique identifier to individual sale record stored in Sale Table. We could keep track on the collection status of each customer's invoices, by setting the Status_BT field as a bit data type, specifying 1 as paid, and 0 as unpaid status.

Enter the below fieldnames, data type, length and the allow nulls attribute in the Customer Table.

Designing Customer Table

Customer_T				
	Column Name	Data Type	Length	Allow Nulls
🔑	Cust_ID_VC	varchar	20	
	Cust_Name_VC	varchar	30	
	Cust_Add_VC	varchar	50	✓
	Cust_Contact_VC	varchar	20	✓
	Credit_Term_TI	tinyint	1	
	Cust_Code_VC	varchar	10	

Notice, that, this table has similar attribute as in the Creditor Table. It contains customer details elements, name, address, contact and credit term. The Cust_Code_VC field will hold the user defined account code for capturing the debtor amount in the Journal Table. We could assign a different debtor account code for each customer, but, the code would need to be created in the COA Table beforehand.

Summary

In Chapter 5, we have discovered the key element that makes up the Sale Table, the importance of creating a customer table and the functions of each fields created in the Sale and Customer Table.

To summaries, in this chapter, we have discussed:

- How to normalize the Sale Table by breaking up and eliminating some duplicated field.
- The purpose of creating a separate table for customer
- The purpose of creating Credit_Term_TI field in Customer Table
- The purpose of creating Cust_Code_VC field in Customer Table

Chapter 6

Developing the Cash Table

Cash Table	
	Chq_No_VC
	Descrip_VC
	Amt_MO
	Date_DT

Normalizing Cash Table

The Cash Table would hold information on payment and collection. Dave has mentioned the key elements that he wants, as shown on the Cash Table drawn above. This table looks simple, but beyond its surface, it holds duplicate information, and some key elements are missing. In the following discussion, we will analyze further on how we can reshape the design of the Cash Table that would benefit MaxCorp more in terms of speed and space efficiency.

First Normal Form

We will attempt to filter out duplicated data contained in the Cash Table. The Amt_MO and Date_DT have been created in the Journal Table, thus we can remove these two fields. We can also do away with the Descrip_VC column, as we have included it in the Journal Table too. We will include the Doc_No_VC field in the Cash Table, as this field would allow us access to the three fields that we have removed from the Cash Table.

Second Normal Form

Dave requested, that, he would like to have some cash flow reports generated for MaxCorp. He would like to review reports that cover on MaxCorp's forecast and bank reconciliation. For this special purpose, we will include two additional key fields in the Cash Table, the Cash_Type_VC and the Cash_Category_VC fields. The Cash_Type_VC field will categories the type of expenditure or income arising from MaxCorp payment and collection. The Cash_Category_VC would be the sub category of the main category type created in the Cash_Type_VC. We will discover this in depth, when we go to Chapter 12.

Third Normal Form

We also want to identify the bankers that, are servicing MaxCorp, for this reason, we shall create an identity field for each of MaxCorp's banker. Create a fieldname: Bank_Code_VC, next create a new table called Bank_T, to store the name, address, contact of each of MaxCorp's bankers. We would also want to interface all payment and collection with the Creditor Table and Customer Table, create the two fields named, AR_Code_IN and AP_Code_IN in the Bank Table.

Enter the following column names, data type, length and allow null attributes in the Cash Table as follows:

Cash_T *				
	Column Name	Data Type	Length	Allow Nulls
	Doc_No_VC	varchar	20	
	Cash_Type_VC	varchar	20	
	Cash_Category_VC	varchar	20	
	Chq_No_VC	varchar	20	
	Bank_Code_VC	varchar	10	

Cash Table

We have redefined each of the fieldname created in the above Cash Table, with 4 newly created fieldnames. The first field constitutes a primary key field, as this field would hold a row or a set of records of payment or collection contained in the Journal Table, referencing it as a foreign key in the Journal Table. Each time, a payment or collection transaction processing is made, the user, would have to define the type of expenditure or income under the Cash_Type_VC field. We will go deeper on this topic when we begin to create reports for Dave, under Chapter 12.

Enter the following column names, data type, length and allow null attributes in the Bank Table as follows:

Bank_T *				
	Column Name	Data Type	Length	Allow Nulls
?	Bank_Code_VC	varchar	10	
	Bank_Add_VC	varchar	10	✓
	Bank_Contact_VC	varchar	10	✓
	AR_Code_IN	int	4	
	AP_Code_IN	int	4	

Designing Bank Table

Under this table, set the Bank_Code_VC as the primary key field. Next we would want to provide Dave an option, to select the mode of transaction under the cash form screen. Dave would want to have three choice of transaction, first, an option to select a debtor account code, that will interface with the Cust_Code_VC field, created in the Customer_Table for collection transaction, second, to select a creditor account code, to interface with the Cred_Code_IN field located under the Creditor Table for effecting the payment transaction, and, thirdly, an option to select account code from the COA Table for each cash transaction purposes.

Summary

In Chapter 6, we learned, that, brainstorming with the user, would open up some new ideas and requirement, as what Dave have personally shared his special needs of some cash flow reports from the Cash Table. Then, we discovered the need to include some mandatory fields, just to meet Dave's expectation.

To summarize, in this chapter, we have discussed:

- How to normalize the Cash Table by breaking up and eliminating some duplicated field.
- The purpose of creating Cash_Category_VC field in the Cash Table.

- The purpose of creating AR_Code_IN and AP_Code_IN field in Bank Table

Chapter 7

Developing the Asset Table

Asset Table	
	FA_Code_VC
	FA_Descrip_VC
	FA_Amt_MO
	Supplier_VC

Normalizing Asset Table

Dave, have requested that MaxCorp, would want to have a master list for all of its existing assets that would contain some relevant information as depicted on the above table. The asset code, to identify the asset physically, their description, amount paid for the asset, and also the name of its provider. It seems, that, we are close to achieving Dave's requirement, by looking at the above four fields. But, we are still far from reaching our goal at this stage of our design. We will detect missing fields and expanding the columns in the asset table, as we progress further in our below discussion.

First Normal Form

Notice the above table, contains a field that store value for each of MaxCorp's asset, in which we want to eliminate from this table. As mentioned, under Chapter 2, we will contain all debit and credit amount in one central table, the Journal Table, in which a set of double-entry are grouped under a journal number located in the Doc_No_VC fields. Thus, we will include the Doc_No_VC field in the asset table, in replace of the fieldname:FA_Amt_MO.

Second Normal Form

Moving forward, it is unlikely that, MaxCorp will purchase its asset from its same source of supplier. We may want to create a separate table to keep the name and addresses of each of its supplier. Wait, we have previously created a table for creditor, so, let's add the fieldname Cred_Code_IN, but, then again, we may even by pass this field, as we can reference to the Creditor Table via the Doc_No_VC field. We will go into this in more detail under Chapter 13, where we will start creating reports from the asset table.

Third Normal Form

Dave has requested that, he would like to see a summary of all MaxCorp's asset, by type, segregated by different category of treatment. In order to achieve this, we need to create the first field, to hold the asset type, and the second field to contain the treatment category. We shall understand this clearly, as we begin developing the relevant report under Chapter 13.

Enter the following column names, data type, length and allow null attributes in the Asset Table as follows:

FixedAsset_T *				
	Column Name	Data Type	Length	Allow Nulls
	FA_ID_IN	int	4	
	FA_Type_VC	varchar	20	
	FA_Category_VC	varchar	20	
	FA_Descrip_VC	varchar	50	
	Doc_No_VC	varchar	20	

Asset Table

As we can see, from the table above, we have expanded the fields in the asset table, to include one more important field, the Doc_No_VC that would allow user the access to each particular asset's value, date of purchase and its provider name. We can retrieve all this, by joining the following table in our query, Asset table | Journal Table | Purchase Table | Creditor Table. We will discuss this in more detail as we touches on creating reports for Dave in Chapter 13.

Summary

In Chapter 7, we learned that, MaxCorp wants to maintain a master list of all of its existing asset, then we start to create table to store all information related to its assets, and created some additional fields to meet some of Dave's required reports.

To summarize, in this chapter, we have discussed:

- How to normalize the Asset Table by creating and eliminating some duplicated field.
- The purpose of creating FA_Type_VC field in the Asset Table.
- The purpose of creating FA_Category_VC field in the Asset Table.

Introductory Chapters

If you find the contents of this book interesting, and would like to find out how to create SQL script to generate your own customized financial report, you may find the following Chapters interesting in the author's complete version on 'Accounting Database Design'.

The Complete version includes the following Chapters :

1. Chapter 8 : Creating Reports from Journals Table
2. Chapter 9 : Creating Reports from Inventory Table
3. Chapter 10 : Creating Reports from Purchase Table
4. Chapter 11 : Creating Reports from Sales Table
5. Chapter 12 : Creating Reports from Cash Table
6. Chapter 13 : Creating Reports from Asset Table

Download Link for Complete version : <http://www.accountingdes.com>

-End-