Relational Database Concepts

Table of Contents

Introduction

Overview

Pre-Assessments

Each major section of this course begins with a pre-assessment. Its purpose is to help you to evaluate your knowledge of the topic for that section. Many of you will have gathered information about databases before undertaking this course, so you may use the pre-assessment to gauge how well you understand the concepts at hand.

The pre-assessment is not used by anyone else to measure you. Simply consider it one of many milestones in guiding you through the learning process.

Discussion Text

The text introduces concepts, practical examples and defines terminology. Feel free to annotate the discussion text as you read.

Apply Your Knowledge

Use this portion of the course to evaluate your mastery of the discussion text that precedes this section. Included are multiple choice questions, open ended questions and fill-in forms to help while working through the material.

The Strategy

In this self-paced course, you will be tracking information about software customers, the contacts you have with them and the software they buy from Microsoft. As you gain a firmer ground in relational database concepts, you will refine the way you organize the data and amass a vocabulary of database terminology. At every milestone, you'll test yourself by answering questions and filling in forms.

What Will You Need?

Because the goal of this course is to bolster your understanding of concepts (rather than teaching you product-specific skills), you will not need a computer at all. Everything you need is contained within the covers of the book. You are welcome to jot down notes and to highlight key passages.

It is a good idea to work with a pencil when filling in responses to *Pre-assessment* and *Apply Your Knowledge* materials.

Flat File Databases and Basic Terminology

Introduction

This section introduces a business scenario familiar to people who sell software. You will step through the basics of the design process, familiarizing yourself with the data. Along the way, you will learn fundamental database terminology.

Objective

After completing this section, you'll be able to:

- Understand the structure of a flat file database
- Reorganize data into multiple tables

Pre-Assessment

1. What are data?

2. What is a DBMS?

3. What is a table?

Perspective: Creating A Flat File Database

The best way to familiarize yourself with how a database works is to think about the kind of information you have to routinely record and track in the course of business.

The information you commonly work with in the course of business is composed of *data*. Data are the most common elements. For example, you keep a record of the addresses of customers. Company name and phone number are two data items.

A *database management system* (DBMS) is an application that aids in establishing, organizing, maintaining and processing data.

The data that database management systems work with are stored in *tables*. These are called tables because users can conceptualize the stored data as a two-dimensional object consisting of rows and columns.

Sketch it out on paper first

Before you create a database on the computer, it's wise to jot down which data items you will want to structure into tables.

For our first attempt, it helps to define a list of data items that a Microsoft salesperson would want to keep track of. The listed information helps to track customers, contacts, invoices and items ordered.

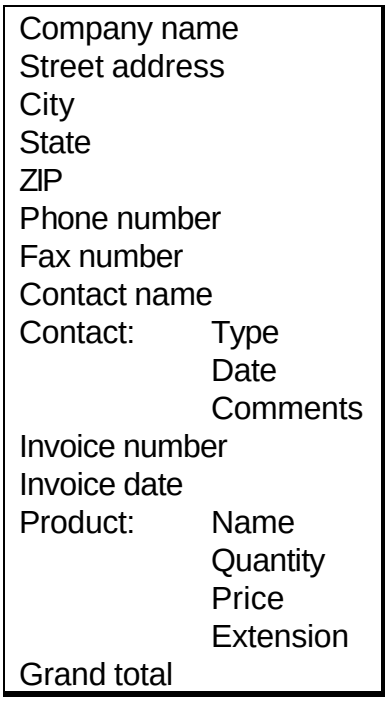


Figure 1. List of data items

Picture the data items written on a sheet of paper arranged with columns and rows. Each row in the table stores a set of related information about a customer:

Contact Name	Street Address	City	State	ZIP
Charles Henry	444 Wilson Terr.	Cucamonga	CA	99006
Charmaine McDoe	2209 Destiny St.	Kent	WA	87654

Figure 2. A small table

In the database world, rows in a table are commonly called *records*. The columns are referred to as *fields*.

If you wanted to know the state where customer Charmaine McDoe is located, you could find that information by visiting the record for Ms. McDoe and consulting the State field.

Analyze the paper design

Study the list of items specified in the List of data items on page 9. You probably noticed some severe limitations:

- Customers typically order more than one product on an invoice.
- It's useful to record more than one recent contact with a customer.

So you decide to modify the table. This new table adds several fields so you can record *two* recent contacts and *three* products per invoice:

Company name	
Street address	
City	
State	
ZIP	
Phone number	
Fax number	
Contact name	
Contact 1:	Type
	Date
	Comments
Contact 2:	Type
	Date
	Comments
Invoice number	
Invoice date	
Product 1	Name
	Quantity
	Price
	Extension
Product 2	Name
	Quantity
	Price
	Extension
Product 3	Name
	Quantity
	Price
	Extension
Grand total	

Figure 3. Revised list of salesperson's data items

The solution is not very practical:

- Two contacts does not make a very useful history of client activity.
- Customers often order many different products on an invoice.

- It's not a great idea to put *repeating groups* of data (like contacts and products) in a single table of data.

  Repeating groups bring with them:

  - **Artificial limits**- When you place a maximum on the number of items you can track, you are often setting an artificial limit on the data you are able to track.

    Example: No business wants to set a maximum on how many different products a customer can order at a time. Examine the Figure: Revised list of salesperson's data items, on page 11. The salesperson's table in has a limit of three products per invoice. What happens when a customer asks you for a fourth product?

  - **Inefficient storage**- Whenever you add another record to the table, you automatically require additional storage space for contacts and products you may not use.

    Example: To avoid setting a low limit on the number of products a customer can order, you set the repeating group up for 12 products. Every time you take an order, the DBMS must store blank space for 12 products— even when the customer orders one or two.

  - **Needlessly complex searches**- The most important objective of placing information in a database is to query it for subsets as you will see later. By placing data in repeating groups you would have to specify every repeating field.

    Example: You want to summarize how many copies of Excel were sold last month. If you have a repeating group for products, you will have to look in every field that holds the product. This would result in asking something like: "Show me all orders where Product 1 is Excel or Product 2 is Excel or Product 3 is Excel or ..."

  Until this point, there was no way to handle *multiple instances of information related to a single entity*. If a customer is the *entity*, we need a way to track multiple invoices, multiple contact events and multiple products.

  Single tables impose limits. As you have seen, a flat file system tends to encourage users to create repeating groups.

  Relational database management systems, however, manage multiple tables and link them as necessary. With a relational database system, you don't have to anticipate in advance how many items you will need to store; you simply add another record to the appropriate table. You add as many records as necessary to properly represent the transaction. If a customer orders ten different products, you can add ten invoice items to the proper table.

  In the next unit, you'll learn how to partition the data items into more than one table for a valid relational data design.

Apply Your Knowledge: Flat File Databases and Basic Terminology

1. List the fields contained in A small table (on page 10).

_____

_____

_____

_____

_____

2. How many records are contained in that table (A small table on page 10)?

_____

3. Give at least two reasons why repeating groups are undesirable:

a) _____

b) _____

c) _____

Normalizing Data

Goals

As you learned in the previous section, flat files are not an optimal way to store data. In this section you'll learn how to take the same data set and partition it into multiple tables that efficiently store data. You'll learn how the tables are linked to provide overviews of the information held in the DBMS.

Pre-Assessment

1. Define normalization.

2. What is the goal of first normal form?

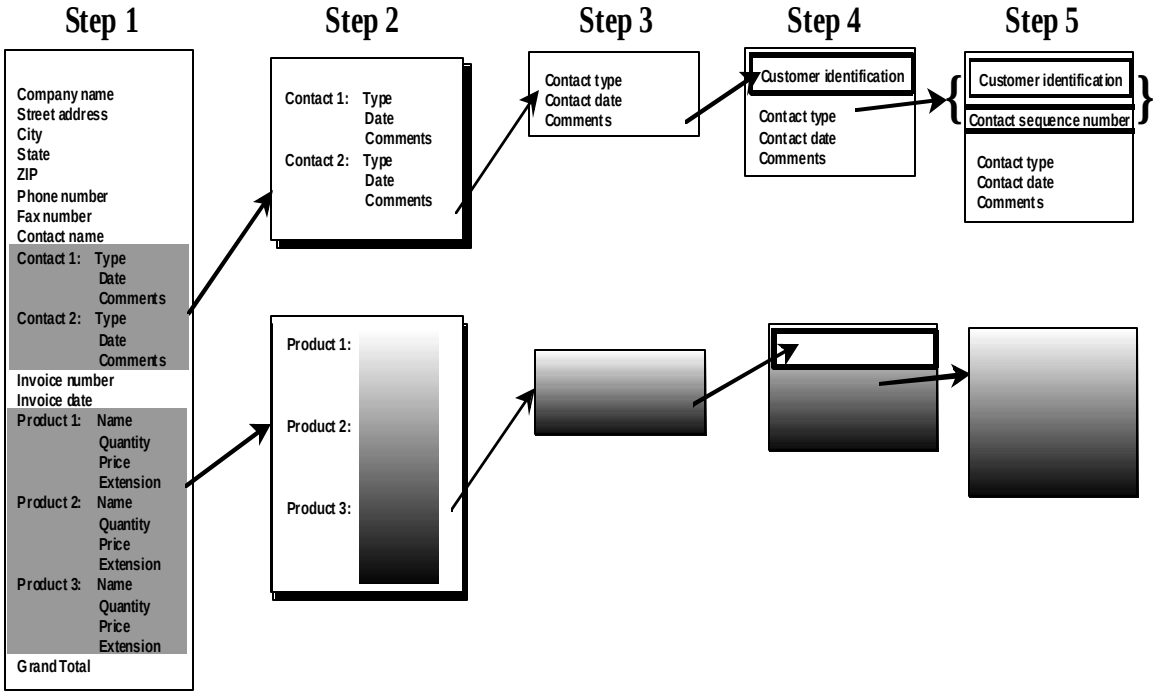
First Normal Form: Eliminating Repeating Groups

You are already aware that repeating groups are a poor data design approach. And you know that relational databases permit you to link multiple tables.

How do you take a flat file and turn that into multiple, linkable tables?

You do this in a series of steps. The process of partitioning data items by grouping them into a set of tables is called *normalization*. The steps are:

1. Identify the repeating groups.
2. Isolate the data items from repeating groups.
3. Design a new nonrepetitive table from the repeating group items.
4. Introduce a linking column.
5. Ensure new table has a unique key.



Here, in greater detail, are the steps:

1. Identify the repeating groups

The flat file includes two repeating groups of items. These repeating groups center on:

- Contact events (when you made a contact, what kind of contact was it?)
- Products ordered (name, price, quantity)

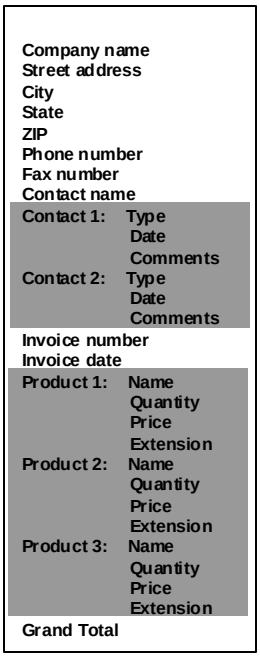


Figure 4. Repeating groups in the flat file (cross hatched sections)

2. Isolating the repeating groups

The first repeating group concerns contact information for two contact events. Remove these fields into a separate list, and delete them from the original flat file.

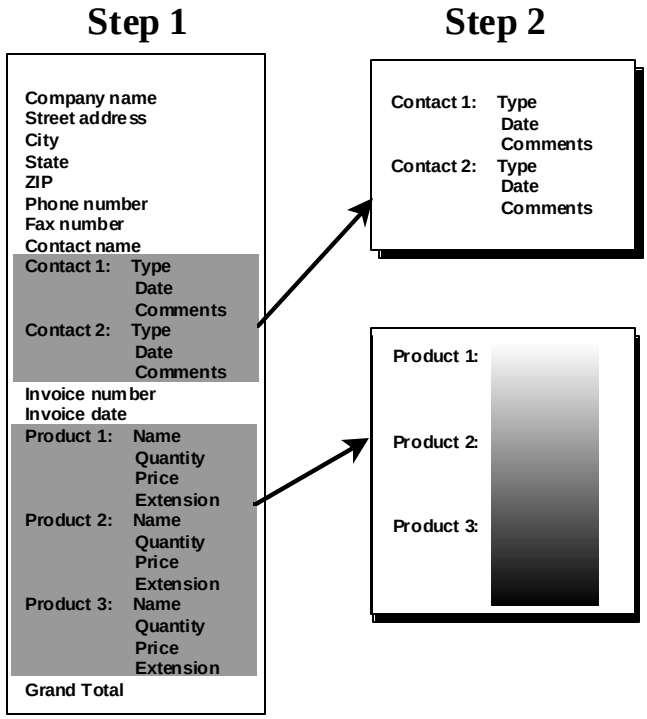


Figure 5. Isolating repeating groups

3. Design a new nonrepetitive table

You are now ready to create a **Contact Events** table. Each row will contain information about just one contact event.

When you remove the repetitive second event (table on the left), your new table will look like the table on the right:

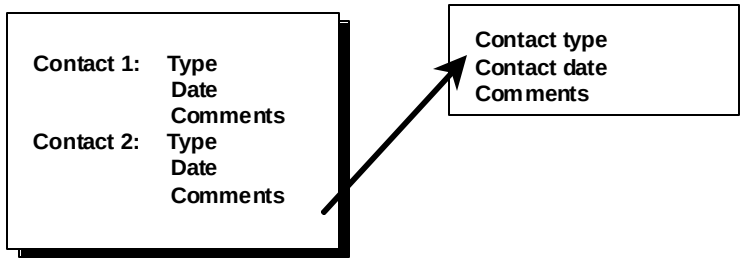


Figure 6. Creating the Contact Events table

4. Creating a Linking Column (Foreign Key)

But you are not done with **Contact Events** yet. When you look at an event in this new table, how will you know which customer this contact concerns?

You solve this problem by identifying the customer in each contact event row. If you add the Company name to the **Contact Events** table you will always know which customer the event pertains to.

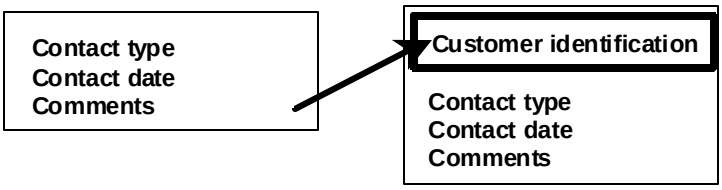


Figure 7. Contact Events table with linking column

5. Ensure the New table has a unique key

You now know which customer generated a contact event. But is the Company Name enough to help us "zero in" on a specific contact? Not unless each customer has only one contact in the entire lifetime of your business relationship with them.

What about using the contact date? Ask yourself a question: "Can I be certain I will only have one contact a day with any given customer? The answer is almost certainly no. You might discuss a contract with a client on the phone in the morning (contact number one) and then meet with them in the afternoon (contact number two) to finalize negotiations.

The solution to the problem is extremely simple. Every contact occurs in sequence. So create a field called Contact Number. This number will be incremented for each contact, making the first contact ever number 01, the next one 02, and so on.

DBMS's provide facilities to increment numbers for you. So every time someone enters a new contact event row into the table, the DBMS will automatically assign the next highest number.

The **Contact Events** table will now look like this:

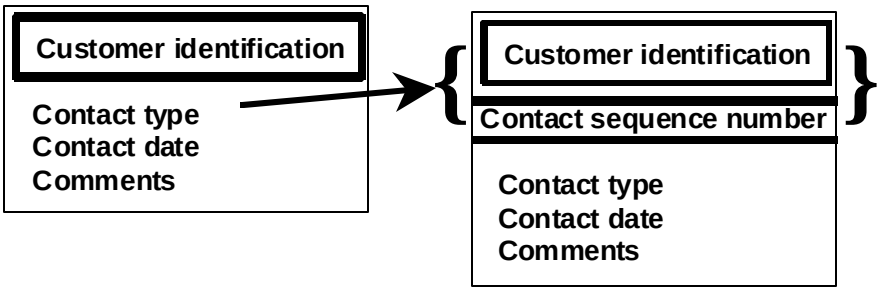
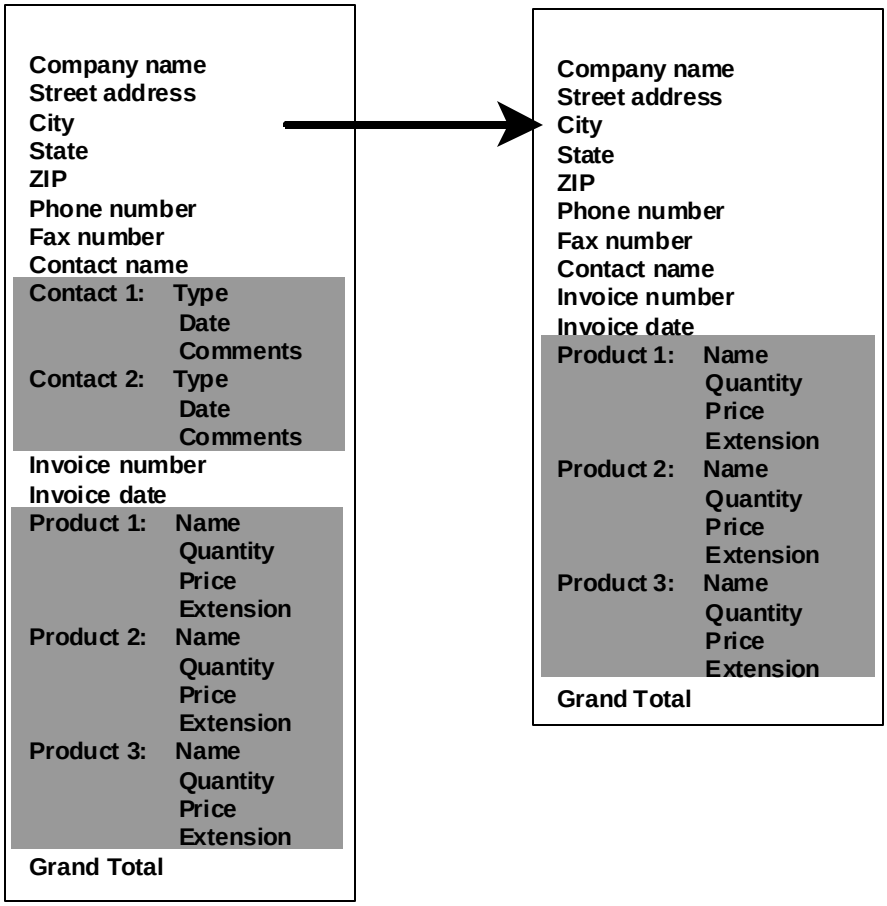


Figure 8. Contact Events table with unique key (at right)

Having isolated and removed the contract information from this flat file, it has essentially become a **Customer/Invoice** table.

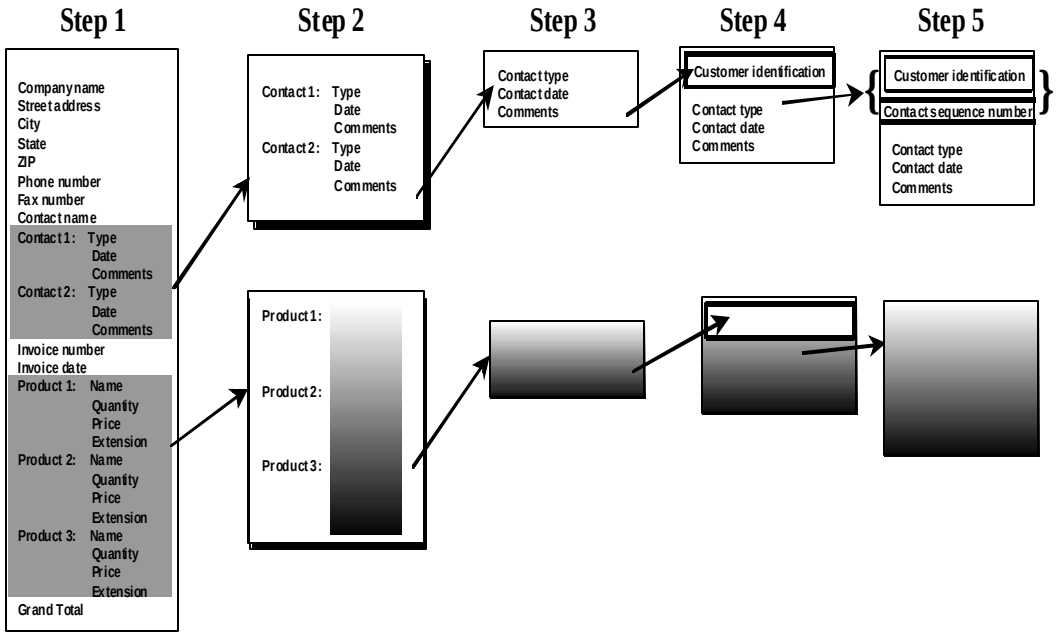


Apply Your Knowledge

Isolating Repeating Groups

Now that the contact events have been removed from the **Customer/Invoice** file, there is one remaining repeating group.

The purpose of this exercise it to isolate the repeating fields using the same approach as with contact events.

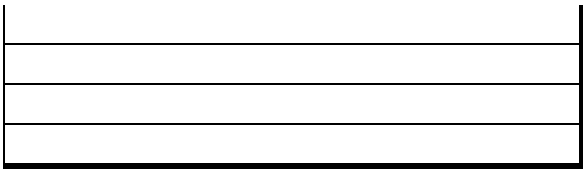


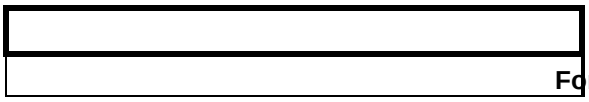
1. Fill in Form A with the names of the repeating group fields.

Product 1
Product 2
Product 3

Form A

2. Fill in Form B to create a new table without the repeating groups. (It is not required to fill in the unique key.)





Form B

Eliminating Redundancies

Goals

When you have completed this section you will understand:

- Eliminate redundant data from the Customer/Invoice table. Create unique keys.

Pre-Assessment

1. List reasons to avoid redundant data.

2. What is the a unique key? Why do you need unique keys?

Eliminating Redundant Information

First Normal Form

Now you have three tables, the **Customer/Invoice** table, **Contact Events** table, and **Invoice Line Items** table. By eliminating all the repeating groups you have put these tables into *first normal form.*

Now that you have eliminated the repeating groups, the **Customer/Invoice** table looks like this.

Company name
Street address
City
State
ZIP
Phone number
Fax number
Contact name
Invoice Number
Invoice Date
Grand total

Customer/Invoice Table

This table contains information about a Customer (company name, address, phone numbers and contact name) and information about an invoice (invoice number, date and grand total).

Can you see problems with this? It appears that you either expect a customer to have only one invoice, or you must repeatedly type all the customer information into each invoice.

Clearly you hope for many more than one invoice per customer, so it appears that you must re-type all customer information on each invoice.

There are several problems with this solution.

- It is tiresome to have to re-enter the same information repeatedly, and you are likely to get sloppy on each subsequent re-typing, leading to a high liklihood of error.
- If a customer cancels their first invoice, and you delete it from your table you will have lost their name and address along with the invoice information.
- If a customer moves to a new location you may be forced to change their address on several **Customer/Invoice** table entries.
- You can only record customer information in your table when you create an invoice.

These limitations indicate that you need to further refine the table design.

Will it improve the situation if you separate the Customer information from the Invoice information, creating two tables? If you put the customer specific information into a new **Customer** table, and the invoice specific information into an **Invoice** table they will look like this:

Company name
Street address
City
State
ZIP
Phone number
Fax number
Contact name

Customer Table

Invoice Number
Company name
Invoice Date
Grand total

Invoice Table

The tables are now defined so that each customer can have many invoices. You have included the Company Name in the invoice table so you can link an Invoice to a Customer, just as you did with **Contact Events** and **Invoice Line Items**. If the Company moves you only have to change the address once, in the **Customer** table. If you need to delete one of the invoices you can safely do so without risking the loss of customer information.

Furthermore, this table design supports your way of conducting business in other ways. For example, it is common to establish and develop a business relationship with a new customer well in advance of invoicing them for an order. Once the invoice information is isolated into its own table you can create an entry in the **Customer** table for prospects, and start tracking contact events, independently of recording an invoice.

A Primary Key Identifies Each Row Uniquely

Are there any remaining problems with the **Customer** and **Invoice** table designs?

Does the company name uniquely identify each customer? Probably not.

- You might have several customers with the same company name.
- You might not always enter the name exactly the same way.
- Customers might change the Company name.

While you might know that GM, Gen'l Motors, General Motors and General Motors, Inc. all refer to the same company, you cannot expect a DBMS to know that. DBMS's are as literal in searching and matching information as a word processor.

Is the Company Name a wise choice of column to link these two tables? No again. If you have two customers called Puget Sound Printers you will not be able to know which invoice belongs to which printer.

The best way to ensure that you can unambiguously identify a particular customer is with a unique code that will never change.

Example: On the Microsoft Retail Products sheet, every retail product has its own MS Part Number. This ensures you that you are ordering exactly the right product when you use that MS Part Number. A full version of Word for Windows has one MS Part Number while a 3.5" diskette upgrade version has another.

Imagine what would happen if the government kept track of things using names alone. How many people named John Smith would appear in the taxpayer database? How many towns in the United States are called Albany? Postal areas have ZIP codes and people have social security numbers.

Every customer doing business with you probably already has an account number. The account number is unique to each customer, which means that no two customers will ever have the same number.

So if you include the Customer Number in both tables both you and the DBMS can be sure that you can unambiguously identify a company, and can associate each invoices with the correct customer.

When you add the Customer Number to the tables they now look like this:

Customer Number
Company name
Street address
City
State
ZIP
Phone number
Fax number
Contact name

Customer Table

Customer Number
Invoice Number
Invoice Date
Grand total

Invoice Table

In relational data base terminology, the Customer Number is the *primary key* for the **Customer** table. Every column in the **Customer** table describes only the customer identified by that Customer Number.

You can now confidently say that:

- all repeating groups have been eliminated *(first normal form)*
- a primary key has been identified,
- each row contains only information about one customer

A database design that meets these three criteria is in *second normal form.* The process of normalization includes additional normal forms. However, it is not necessary to understand them to grasp database fundamentals.

Now that you have a key field, Customer Number, you will need to re-examine the **Contact Event** and **Invoice Line Items** tables. You will need to replace Customer Name with Customer Number as the linking field.

Once you do this, these tables will look like this:

Customer Number
Contact Number
Contact Type Contact Date Comments

Contact Events

Invoice Number
Line Item Number
Product name Quantity Price
Extension

Invoice Line Items

Apply Your Knowledge: Normalizing Data

1. A primary key is (select the correct choice):

   a. An optional feature in relational databases

   b. A column that must be in all tables to uniquely identify the contents of each row

   c. The main data item to look at in a database

   d. The method of normalizing data

Exercise: Implementing Second Normal Form

The purpose of this exercise it to isolate and remove the redundant fields using the same approach as with the Customer/Invoice table, and two identify primary keys.

Two things must be resolved to put the **Invoice Line Items** table into second normal form.

- There is one redundant field, and
- There is an inappropriate foreign key. (Consider the problem you had using Company Name as a key.)

Invoice Items

Products

Other Database Models

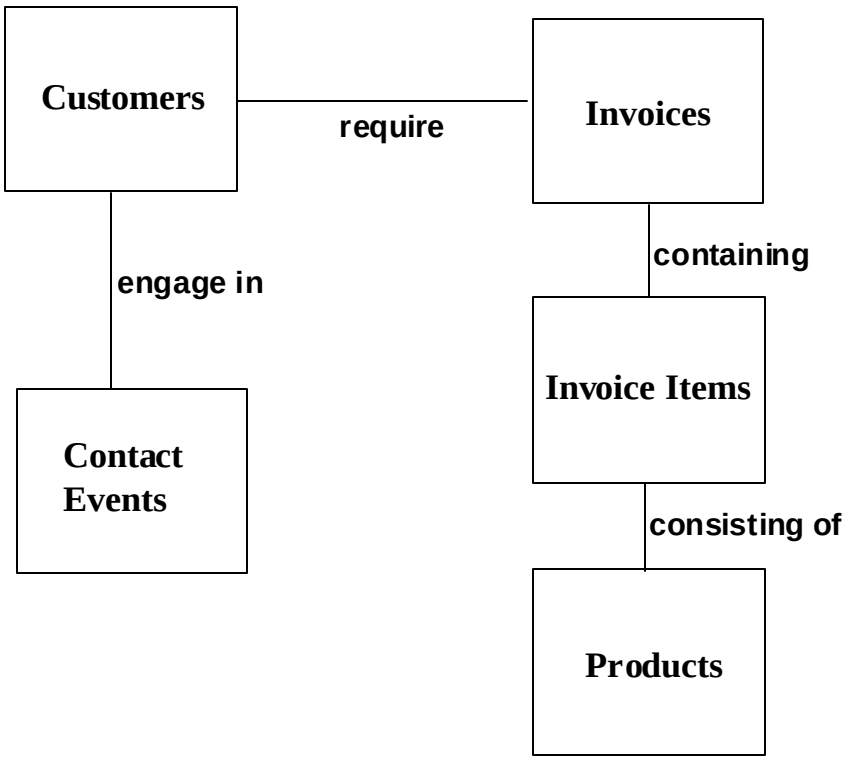
Normalization

To normalize a data set, you first determine what the *entities* are in the system. Entities tend to be people, events and things. In the case of the salesperson's database, the entities are:

- Customers
- Products

To help you in normalizing the data, it's helpful to sketch out a model depicting all the entities in the data set. The diagram also indicates how the entities are related to each other:

Figure 5. Data Relationship Diagram



This diagram places the entities in boxes with connecting lines. Along each connecting line is a very brief phrase describing the relationship between the two connected entities.

Once you have diagrammed the data relationships, it become much easier to normalize the data set.

You would use the MS Part number as the column that joins **Products** and **Invoice Items**. Every table in a relational database must have a *primary key* column that can give you direct access to only one row in the table. You can be certain that when you ask the DBMS for the row regarding MS Part Number 105-050v200 in the **Products** table that the DBMS will correctly show you information about Microsoft Money for Windows.

Since the DBMS must be depended upon to return correct results, the primary key must be *unique*. In the **Products** table, the MS Part Number is unique and this serves as a guaranty that the DBMS can find what you are looking for.

Once you can be sure that *every* table in a relational database has a unique primary key, you can be sure that the DBMS will properly join tables without losing any of the rows or incorrectly linking the row from one table to the related row in the other.

Apply Your Knowledge: Normalizing Data

2. Database experts use the terms column and row or field and record. Match up the equivalent terms:

I. Field	**a.** Column
II. Record	**b.** Row

I. _____
II. _____

3. What is a primary key and what is its main characteristic?

Why use a database instead of a spreadsheet?

Goals

Surveys show that the most popular way for PC users to record structured information is to use a spreadsheet. This section helps you to understand the particular strengths of a DBMS and to identify practical reasons why a database system is a better customer choice. This section introduces additional database terminology.

Pre-Assessment

List six reasons why you would use a database instead of a spreadsheet.

a) _____

b) _____

c) _____

d) _____

e) _____

f) _____

Perspective

Spreadsheets are ideal tools for analyzing and manipulating data. This is particularly valuable when you want to test multiple "what if" scenarios."

But spreadsheets are not well suited for storing and retrieving complex and interrelated information.

Example: A brand new product announcement appears on your electronic mail first thing in the morning and you take an order for it within minutes of the public announcement. By the end of the morning, you have entered orders from 25 corporate sites. Then an electronic mail message alerts you that the previously announced MS part number was incorrect. Worse yet, the incorrect part number is identical to an existing Microsoft product (Microsoft Word for Windows on 5.25" diskettes). The alert includes the corrected MS part number. You now have to find a way to correct the part numbers as listed in every order for the new product— without changing legitimate orders for Word for Windows on 5.25" diskettes.

Eliminating Data Redundancy

Since spreadsheets lack the means of joining data tables, users are more likely to create tables with repetitive entries as in our flat file example. Having to type data multiple times is an inefficient use of time and computer resources. It's also a likely source of data inconsistency— during the data entry process people are likely to use varying abbreviations and spellings:

21	B	C	D
22	Microsoft	One Microsoft Way	Word for Windows
23	Microsoft Corp.	1 Microsoft Way	Project
24	Microsoft Corp	One Microsoft Wy	Mail
25	Microsoft Corporation	One Microsoft	Excel

Figure 9. Redundant entries in a spreadsheet

Inconsistencies mean that searches for data subsets may not return all the data. What happens when there's a single change of address? If repeating groups exist in a spreadsheet table, users have to make multiple changes to the data.

Ensuring Data Integrity And Validation

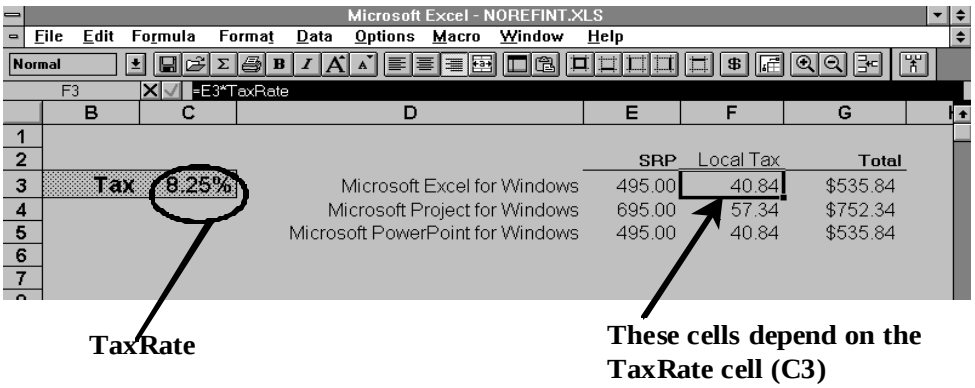
Spreadsheets lack tools for controlling what goes into the spreadsheet. The key to a good database is reliable information. A business's order entry system typically restricts data entry clerks to a specific list of products for sale. Database management systems test the correctness of data as they are entered so that incorrect product codes are rejected by the database and valid codes are displayed.

Enforcing Referential Integrity

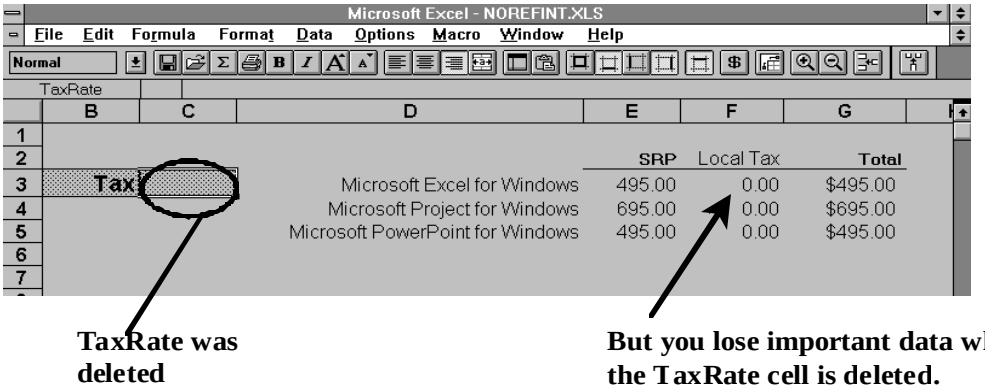
Because spreadsheet-based data is essentially stored in a flat file, there is no way to protect against the deletion of non-normalized data. Imagine you keep a spreadsheet listing orders.

There are no automatic protections that ensure that master cells in a spreadsheet will be retained. Unless you manually lock a cell upon which others are dependent, you could easily delete the contents of any cell; if other cells hold formulas depending on that master cell, these dependent cells are now in error.

Before



After



Example: A brand new product announcement appears on your electronic mail first thing in the morning. You take an order for it within minutes of the public announcement. You type into the spreadsheet the customer name and number, the product part number and quantity. Ten minutes later the customer calls to postpone the order. So you delete the order from your spreadsheet. Along with your deletion of the order goes the part number for the new item. What happens when another customer calls you an hour later to place a large order for the hot new product? You have lost the MS part number.

In a database, you would have established a product table. Upon the announcement of a new product, you would enter the product name and number into that table. As orders come it, the orders table would *refer* to the product table for valid product numbers. What happens if a product is superseded? Let's say that the development team makes a dramatic performance improvement in the product one month after the product starts shipping and a new product number has been assigned. All back orders for the item are automatically fulfilled with the new product number.

You must now decide how to enter this new information into your system. You might want to delete the first product number from your product table, or to change its product number to the newer one. However, you have already shipped the product to some customers and you may have back orders for it. If you delete the product from your product table, or change its number, those orders will be "orphaned", they will reference a product number no longer available.

A database system would not permit you to delete the old MS part number as long as there are tables depending on that part number. This protects referential integrity. You can be certain that any record containing the old MS part number will not be "orphaned" in relation to the product table.

On the other hand, a spreadsheet has no single place where you can change the part number so that the correct change ripples through the rest of the data. And if you change data, there is no mechanism to protect against orphaned transactions.

Limiting Data Sets

Before

Wilson	**NY**
Taylor	**CA**
Patrick	**WY**
Nostrand	**WA**
Dennison	**NJ**
Bennett	**WA**
Marks	**TN**
Taylor	**NY**
Spencer	**NV**
Powell	**AK**
Vinson	**NY**
Vee	**MD**
Alston	**MO**

After

Wilson	**NY**
Taylor	**NY**
Vinson	**NY**

Spreadsheets offer limited options when users want to evaluate subsets of data, but databases are designed specifically for that purpose.

Spreadsheets can usually be set up to permit one or two specific queries, but if a user decides to change the kinds of queries they do, they might have to dramatically change the spreadsheet design and even enter new data. Properly normalized databases can be queried in ways that the user never anticipated.

Providing Performance And Capacity

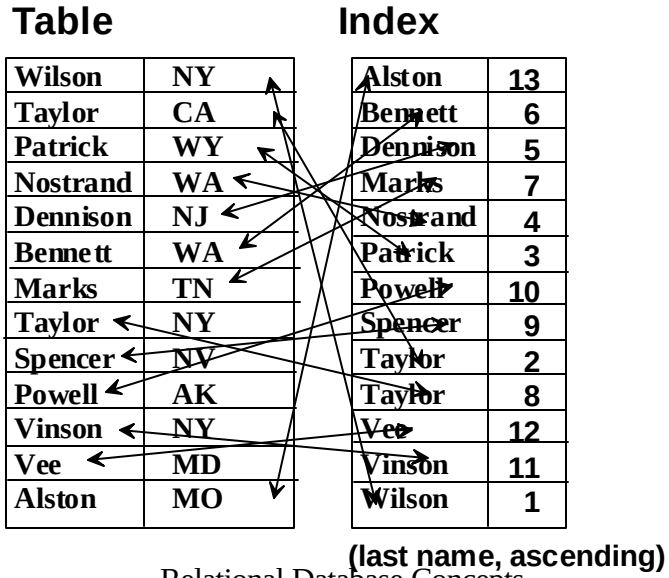
A spreadsheet slows down in proportion to the amount of data it holds. Databases are optimized to handle large amounts of data. They also provide indexes to speed access to data. An index file works similarly to an index in a book. It is organized in sorted order. Each entry in the index includes the full contents of the indexed item. Each entry also holds a notation (or *pointer*) to the location of that record in the table. When a user indexes on the last name field, he or she can rapidly access the record for any given last name in a table.

You can easily understand how an index works if you consider the strategy for researching with books. If you're looking for information about Alexander the Great in a world history textbook, you have two possible search strategies:

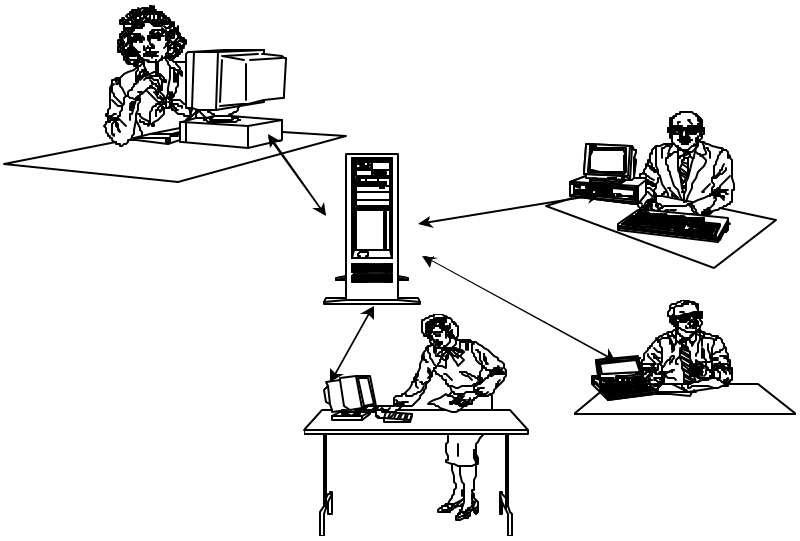
- You could start at the beginning of the book and scan backward until you find a reference to Alexander the Great.

- You could consult the index and then proceed to the page identified in the index.

The first method is sequential. It works well only when the data you are searching for happens to be at the beginning. You can rarely depend on this factor. The second approach is speedier.



Supporting Multiple User Concurrency

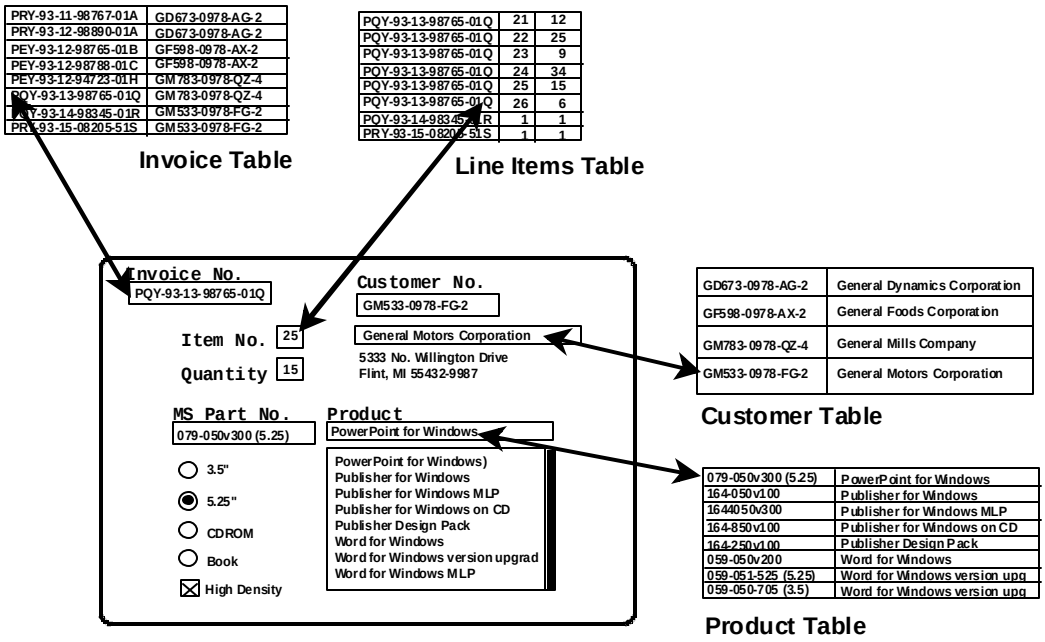


On a network a spreadsheet is available to one user at a time. Once the first user loads the spreadsheet into the computer's memory, other people attempting to work with that same spreadsheet receive a message that this particular spreadsheet file is in use by someone else. Only when the first user closes the spreadsheet can someone else gain access to it.

Most database management systems are conceived from the start as multiuser applications. Data can simultaneously be added, deleted and updated by multiple people and departments. Some users may be generating queries while others are performing data maintenance. This is especially useful when the database is in constant use.

Providing Flexible Data Entry

Spreadsheets can provide simple forms for data entry, but they lack support for multiple tables. Database systems give users the ability to customize data entry screens for ease of use, accuracy and productivity. Users can organize and group data entry fields on the screen as convenient. These can match familiar paper forms. Screen forms can automatically fill in certain fields based on multiple factors (Example: user types the first few characters of the product name and the data entry screen displays the full name of the product, its price, media configuration, operating system and the MS part number.)



Delivering Programmability

Spreadsheet macros were first conceived to replay keystrokes. Although branching logic has been added, spreadsheets lack the features of full programming languages. Database systems typically come with programming languages including debugging tools and error handling capabilities.

Supplying Flexible Reporting

Spreadsheet reports typically display columns and rows of numbers and text labels as they were entered by the user. Database management systems include reporting tools that allow users to summarize data group items and arrange the contents in a free form manner. Databases give users the ability to format their reports as they'd like to see them, but spreadsheet reports appear the same way the data is stored.

Apply Your Knowledge: Why use a database instead of a spreadsheet?

1. List at least two reasons why databases offer data entry features superior to spreadsheets.

   a.

   b.

   c.

2. What is a database technique that offers faster data access?

3. Define *referential integrity.*

Appendix: Suggested Responses

Flat File Databases and Basic Terminology

Pre-Assessment (Page 8)

1. What are data?

   Data are the individual components of the information people work with.

2. What is a DBMS?

   A DBMS is a database management system. DBMS's are applications that aid in establishing, organizing, maintaining and processing data.

3. What is a table?

   Database management systems store information in tables. A table consists of a two dimensional structure of columns and rows.

4. What are repeating groups?

   Repeating groups are clusters of fields that permit a single record to hold more than one instance of a data item.

Apply Your Knowledge: Flat File Databases and Basic Terminology (Page 13)

1. List the fields contained in A small table (on page 10).

   Customer Name

   Street Address

   City

   State

   ZIP

2. How many records are contained in that table (A small table on page 10)?

   2 records (for customers Henry and McDoe)

3. Give at least two reasons why repeating groups are undesirable: (material appears on page 12.)

   a) Artificial limits

   b) Inefficient storage

   c) Needlessly complex searches

Normalizing Data

Pre-Assessment (Page 23)

1. Define normalization.

Normalization is the process by which a database designer groups fields into multiple tables. The process entails meeting the criteria of a series of normal forms.

2. What is the goal of first normal form?

The goal of 1NF is to eliminate all repeating groups.

Excercise: Isolating Repeating Groups

1.

Product 1	Name
	Quantity
	Price
	Extension
Product 2	Name
	Quantity
	Price
	Extension
Product 3	Name
	Quantity
	Price
	Extension

Form A

2.

Invoice Number
Product Name
Quantity
Price
Extension

Form B

Apply Your Knowledge: Normalizing Data (Page 30)

1.

a. A column that must be in all tables to uniquely identify the contents of each row

2.

Invoices

Invoice number
Customer number
Invoice Date
Grand total*

* Note that the Grand total field is not absolutely necessary, since the DBMS can link this table to **Invoice Items** and calculate the grand total.

Invoice Items

Invoice number
Item Number
Product number
Quantity
Extension*

* Note that the Extension field is not absolutely necessary, since the DBMS can calculate the extension by linking this table to **Products** and multiplying price by quantity .

Products

Product Number
Product Name
Price

In real life this table might include additional fields for media format, operating system and other information specific to the product.

2.

I. b
II. a

3. A primary key is the column in a table that holds unique values. Primary keys provide a means for accessing any row in the table.

Why use a database instead of a spreadsheet?

Pre-Assessment (Page 33)

1. Eliminating data redundancy

2. Ensuring data integrity and validation

3. Enforcing referential integrity

4. Limiting data sets

5. Providing performance and capacity

6. Supporting multiple user concurrency

7. Providing flexible data entry

8. Delivering programmability

9. Supplying flexible reporting

Apply Your Knowledge: Why use a database instead of a spreadsheet? (Page 40)

1. List at least two reasons why databases offer data entry features superior to spreadsheets.

a. Support for multiple tables

b. Automatic display of information

c. User can organize and group fields at their convenience.

2. What is a technique that databases offer for faster data access?

Indexes store a sorted list of a field's (or a group of fields) contents. Each index entry holds the full text of the field and a pointer to the location in the table.

3. Define *referential integrity.*

Referential integrity protects against the deletion of a master data item upon which other tables are dependent. One could not delete product A from the Products table when there are outstanding orders for Product A.

Request for Feedback

Feedback Form