Dear Access Lover,

About a year ago, I decided it was time to write about basics for Access, so I did and it was 30 pages. Several folks told me screen shots would be nice – so here you are! The file size got a lot bigger and it has more than doubled in size, but I hope it is still short enough that you will read it all <smile>. This document covers essentials in Access and prepares you for programming with VBA (Visual Basic for Applications).

I wish to thank Allen Browne for including my Access Basics document on his fantastic website, and the following folks (all esteemed Microsoft MVPs -- Most Valuable Professionals) for much appreciated edits and comments: Allen Browne, Brent Spaulding, John Mishefske, John Viescas, Truitt Bradley, Marshall Barton (edited my first version too) and an extra-special thanks to Tom Wickerath, who was so incredibly thorough and went through this document several times.

This is never done… but I figured I will give you what I have so far <smile>

Over the past year, hundreds of you have written with wonderful comments that brightened up my day. Thank you so much! I hope you like the changes!

Print this document, get a highlighter, make a nice pot of tea, get comfy in your favorite chair, relax ... and ... enjoy!

Use the Table of Contents (at the end) to find specific pages … or just start reading ☺

If you want to send a message, I love hearing what you have to say.

Warm Regards,

Crystal
Microsoft MVP Access 2008
strive4peace2008@yahoo.com
*

(: have an awesome day :)
*

I do remote programming and training with great success. If you want information for yourself or to pass on, for now or later, email me and request it.

© Copyright 2008 Crystal Long
All rights reserved. Feel free to give (not sell) this document in its entirety to anyone you wish, but no part of it is allowed to be included in another publication or writing or video or audio or any product that is sold without first obtaining written permission from me. Thanks!
1. Types of Applications

**Perspective**

Before we get into Access, let's put it into perspective. There are six main categories of applications. If you learn a package in each of these categories, you will have a well-rounded foundation on which to build.

**Word processing**

A word processor gives you blank electronic paper to write, edit, and produce text. Although many packages give you capabilities for incorporating graphics, the main function is to work with text. Word processing software gives you the ability to format text such as **bold**, *italics*, and *underline* … and much more! Examples of when you would use a word processor would be to write letters, papers, articles, and books. The use of word processors is widespread. In the Microsoft Office suite, the word processing application is Word. Other word processing packages include WordPerfect and Open Office Writer. WordPad is a simple word processor that comes with Windows and can be found in the Accessories section of Programs.

If you are using Microsoft Word, you do have some basic database capability – Word has a "merge" feature that allows you to keep track of, for instance, names and addresses in one document and, using another document, write a letter with placeholders for merging.

When you set up information using Word tables, you also have basic sorting and formula capability … but if you find that you have mostly a list of information – and especially if you want to relate one list of information to another or filter it by certain criteria, you should be looking at Access.

**Spread sheets**

A spreadsheet gives you a work sheet with rows and columns. The intersection of each row and column is called a *cell*. You can put text, numbers, or a formula (such as to add a column of numbers) into each cell. A file, called a workbook, can contain many work sheets. Spread sheets are generally used for financial and other applications where calculations, graphs, and "what-if" analysis will be used. Spreadsheets are among the most popular uses for computers. VisiCalc, a spreadsheet that was originally developed to run on an Apple computer, was responsible for the huge initial sales of personal computers in the late 70s and early 80s. In the Microsoft Office suite, the spread sheet application is Excel. Lotus 1-2-3 and Quattro were two popular spreadsheet applications that are no longer made. Open Office's Calc is another alternative to Excel.

A lot of confusion originates with when to use a spreadsheet and when to use a database. Many databases do start out in Excel – and then they grow to the point where it is better to convert data to Access. Excel has sorting and filtering capabilities … but they are not the main focus of the package. Due to the nature of a spreadsheet, sometimes data conversion to Access can be a big task! Spreadsheets allow you to put any kind of information anywhere you like … this is a no-no with databases. If your column has a defined data type of *number*, you cannot, for instance, write, "ask Clint for more details" for the value, it must be a number.

If you find yourself needing a lot of comboboxes that lookup values from another source and relational capability, or complex filtering and sorting, it is time to convert what you have in Excel to Access. Due to the lack of structure requirements of a spreadsheet, data set up in spreadsheets is often redundant (repeating information from one row to another) and not relational. When data is converted into a database, this information should be normalized.
Databases

A database gives you a way to structure information into rows (records) and columns (fields). Each of these collections is a relation, also called a table. Databases give you capabilities to sort and filter information. Where databases really shine is in being able to relate information in one table to another. A phone book is a simple example of a database, as is a hand-written check register, your mother’s recipe collection, and even an Excel spreadsheet. Though a phonebook is an example of a simple database and could be tracked with Excel, it has too many records (rows) to contain all the listings for the country. While Excel continues to expand the limit on the number of rows, it does have trouble with filtering (using the Auto-Filter option available) when there are lots of entries. A more complex example requiring a relational database management system would be tracking all the expenses, sales, and accounting allocations for a company.

In the Microsoft Office suite, the database application is Microsoft Office Access, hereinafter simply referred to as Access. Other desktop (PC) database applications include dBase (no longer made), FileMaker, and Open Office Base. High-end databases include MySQL, Microsoft SQL Server, Oracle, and Sybase.

Unlike other applications, you can't just start typing. A database needs to be planned out and a structure set up before you add data. If you do not plan and just create columns/fields as you need them, you will miss many advantages of a structured, relational database; and you will likely produce an application that will be difficult to query and create reports, and is very error prone.

Access is the hardest application in the Office suite to learn for good reason; it is powerful and you need to first design tables, fields, and relationships before you create your data. Access gives you the ability to create forms for easy entry, and reports for formatted output … but forms and reports should not be designed until the structure of the information is solid. A good database design can take time, and the simplest solution is not often easy to see.

Graphics

With graphics software, you can create drawings, images, animations, presentations, multimedia, and much more. No one package usually does all these things well. In the Microsoft Office suite, the graphics application is PowerPoint, which is geared toward creating presentations and also provides basic tools for creating drawings and editing images. Other graphics packages include CorelDraw, Corel Photo-Paint, Adobe Photoshop, Adobe Illustrator, Adobe Flash, IrfanView, Open Office Impress, Microsoft Movie Maker, Sony Vegas, and ULead. Paint is a simple program you can use to create fun graphics that comes with Windows and can be found in the Accessories section of Programs.

Communication

Communications software allows you to exchange text, images, and files with others. Types of communication include email, VOIP (voice-over internet protocol), instant messaging, and newsgroup participation. Outlook Express often comes with Windows and acts as an email and newsgroup client. In Windows Vista, Outlook Express has been replaced by Windows Mail. In the Microsoft Office suite, Outlook (not Express) provides email and personal contact management capability. Other email packages include Mozilla Thunderbird and Netscape Communicator. VOIP software (softphone) allows you to make telephone calls using the internet.
Web Browsers

A web browser enables you to render text, and images on web pages. Microsoft Internet Explorer comes with Windows; other web browsers include Mozilla Firefox, Apple Safari, Opera (Opera), and Netscape Navigator. The basic language that web pages render is HTML (Hyper Text Markup Language). Many web pages incorporate non-standard features such as video. Plug-Ins allow web browsers to render a specific file format whose capability is not inherently built-in, or gives other additional capability. Plugs-Ins include Acrobat Reader, Adobe Flash Player, Java, QuickTime, Real Player, Shockwave, Microsoft Silverlight, and Windows Media Player. Some plug-ins, such as Windows Media Player and Acrobat Reader are also stand-alone applications.
2. Database Objects

Intro

Starting with mathematical number crunching in an engineering setting, most of the data I worked with was not stored; it was initially input and calculated using equations. The results could range from a few lines to several thousand lines. Getting input from the user during execution was near to impossible.

After college, I began to see the world of databases. The logic used was still sequential, but, unlike the programs I had written in the past, now storing information, as opposed to calculating it, was the primary purpose.

With the advent of Windows and the overwhelming markets for a graphical user interface (GUI), I moved to Access when it came out in the early 90s. It took time to translate my sequential way of thinking to that of event-driven system acting on objects.

Nearly every day since I started working with Access, I learn more about it. It is an incredible application. This document is just the tip of the iceberg… but it's a good start.

"The more I learn, the more I realize I don’t know."

— Albert Einstein

An Access Application is a Container

Folks often use the term "Application" and "Database" interchangeable when referring to Access. Technically, "application" would be the correct term but you will usually see "database" being used in this document.

Analogous to a bucket of seashells, an Access application is a container for many different types of objects. The main objects can be divided into 6 categories: Tables, Queries, Forms, Reports, Macros, and Modules.

Like a toaster or a washing machine that is composed of other distinct parts, each main object type in Access contains other object types.

Tables

The most important objects are tables, which is where data is stored and defined. Tables are composed of rows (records) and columns (fields). Within each table, indexes (lists that Access makes much like indexes in the back of a book to make finding things easier) can be defined.

As a general rule, Tables are what all other objects are ultimately based on and act upon. Tables can be resident or linked to other sources. When you open a table, you see a datasheet view of the information.

Tables are connected to each other by defining Relationships. While you do not have to define a relationship to link tables, it is best to do so.
Datasheet View of a Table

Each row of data is a record. Each column is a field. Within a record, the field describes an attribute of the record. The field names, or captions, are displayed across the top as column headings. In my opinion, you should not use captions in table definitions because they mask the real names. In other words, it is best to see the real field names when you look at the datasheet view of a table.

Figure 2-1 Datasheet View of a Table

<table>
<thead>
<tr>
<th>PID</th>
<th>Gender</th>
<th>TypeID</th>
<th>Human</th>
<th>MainName</th>
<th>FirstName</th>
<th>MiddleName</th>
<th>DOB</th>
<th>IsActive</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>F</td>
<td>48</td>
<td>Ackley</td>
<td>Cynthia</td>
<td>T</td>
<td></td>
<td>7/20/1984</td>
<td></td>
</tr>
<tr>
<td>937</td>
<td>M</td>
<td>48</td>
<td>Actipes</td>
<td>George</td>
<td>S</td>
<td></td>
<td>5/17/1981</td>
<td></td>
</tr>
<tr>
<td>936</td>
<td>M</td>
<td>48</td>
<td>Actipes</td>
<td>Moses</td>
<td></td>
<td></td>
<td>9/2/1978</td>
<td></td>
</tr>
<tr>
<td>762</td>
<td>M</td>
<td>48</td>
<td>Albaugh</td>
<td>Seth</td>
<td></td>
<td></td>
<td>12/17/1990</td>
<td></td>
</tr>
<tr>
<td>733</td>
<td>M</td>
<td>48</td>
<td>Albaugh</td>
<td>Marc</td>
<td>T</td>
<td></td>
<td>7/3/1990</td>
<td></td>
</tr>
<tr>
<td>1239</td>
<td></td>
<td></td>
<td>All Natural</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>877</td>
<td>M</td>
<td>48</td>
<td>Allen</td>
<td>Roger</td>
<td>P</td>
<td></td>
<td>1/7/1888</td>
<td></td>
</tr>
<tr>
<td>1363</td>
<td>O</td>
<td>48</td>
<td>American School of Beauty</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>F</td>
<td>48</td>
<td>Anderson</td>
<td>Heather</td>
<td>E</td>
<td></td>
<td>5/14/1977</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>M</td>
<td>48</td>
<td>Anderson</td>
<td>Walt</td>
<td></td>
<td></td>
<td>4/22/1980</td>
<td></td>
</tr>
<tr>
<td>1315</td>
<td></td>
<td></td>
<td>Andy's Alam systems</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1202</td>
<td></td>
<td></td>
<td>Angelo's Floor Covering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1313</td>
<td>O</td>
<td>48</td>
<td>Applegate Ace Hardware</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>810</td>
<td>M</td>
<td>48</td>
<td>Arbor</td>
<td>Jordan</td>
<td></td>
<td></td>
<td>9/21/1977</td>
<td></td>
</tr>
<tr>
<td>811</td>
<td>M</td>
<td>48</td>
<td>Arbor</td>
<td>Raul</td>
<td>P</td>
<td></td>
<td>5/20/1979</td>
<td></td>
</tr>
<tr>
<td>1054</td>
<td>F</td>
<td>50</td>
<td>Ashley</td>
<td>Gretchen</td>
<td></td>
<td></td>
<td>9/13/1981</td>
<td></td>
</tr>
</tbody>
</table>

Design View of a Table

When you design a table, you create the field names, specify what type of data the field will hold (like text, number, or date), create a description that explains more about the field, and specify other properties such as size, format, default value, and whether or not the field will be indexed.

Each table should have one field (or a combination of fields) that will be unique to each record. This field or combination can be designated as a Primary Key. It is common to use an AutoNumber for the primary key. An AutoNumber is a special form of the Long Integer data type whereby Access automatically assigns the values. Although, to define a field to be a Long Integer, you use Number as the Data Type and Long Integer as the Field Size, Access does actually treat a Long Integer as a specific Data Type.
Without tables, a database normally has little value as this is the where data is typically stored (exceptions are database properties, INI and other external files and the registry). Designing tables is discussed in greater detail in the Normalization section of this document.

**Queries**

The most common use of a Query is to **SELECT** information, specify criteria (filter), and sort. Queries show information from one or more tables as records (rows) and fields (columns). Records can be sorted; they can also be filtered so that only certain records show. For example, let us assume you are printing a Denver phonebook from a database that contains information for the whole country; you can specify criteria to limit records to those whose city is 'Denver'. Then you can sort the list alphabetically by last name then first name.

If you frequently find yourself looking at the same information (perhaps for different criteria such as a date range or a region), it would be good to define a Report instead of using a Query to show the data. When a Report is based on a Query, that query is the **RecordSource**.

The best object to use for changing data is a Form. A Query can be used for a form **RecordSource**. Word of caution: even though a query can pull from many tables, good practice dictates that you should only change data in one table on each form. If you wish to add or update data in a related table at the same time, it is best to use a form/subform(/subsubform) scenario.

Another thing a query could be used for is to specify the **RowSource** of a combo or listbox. As for Properties... in most instances where you can use a table, you can also use a query – and vice versa. This is why tablenames and querynames cannot be the same.

Queries give you a way to quickly answer ad-hoc questions: list all sales for a particular quarter; identify duplicate information so you can fix it; find unmatched records so you can fill in missing data; switch (translate) rows into columns using a Crosstab (Pivot in Excel) so you can see, for instance, sales in columns under the months -- derived from daily amounts that are being stored.

In addition to **selecting** data, queries can perform **actions** to manipulate data such as **Delete** (get rid of Records), **Update** (modify records), **Append** (add records), and **Make Tables**.
The statement defining a Query is stored using SQL (Structured Query Language). Many SQL statements can be displayed graphically on the QBE (Query-By-Example) grid in Design View. As with tables, opening a query that selects information uses the datasheet view to display the information.

In this example, three tables have been used to show information:

1. **t_PEOPLE** has fields describing a person such as first name, last name, gender; or main name for company. Each record is uniquely identified using a field called PID (People ID).

2. **t_Types** is used to categorize records such as: Friend, Family, Professional, Auto, Hospital, and Art Supplies. Each record in this table is uniquely identified using a field calledTypeID. By storing a correspondingTypeID in the People table, a short number can be stored in the People table and the longer text can be displayed at any time from the Types table. The Types table is commonly referred to as a lookup table.

3. **t_eAddresses** stores email address. It is linked to the People table using a field called PID, which is short for PeopleID. Since PID is such a common field, it is abbreviated.

### Datasheet View of a Select Query

The datasheet view of a query looks like the datasheet view of a table.

**Figure 2-3 Datasheet View of a Select Query**

In Figure 2-3, the Fullname column shows a combination of:

- Main name
  - In the case of a human, this is the last name. For a company, it is the company name.
• First name and Middle name or initial
  Some human records have middle name or initial specified and some don't. Companies don't have first names or middle names, so that information is ignored – as well as the commas and spaces that act as separators.

• Suffix
  Some have suffix (Jr., III) and some don't.

In the second column, type of contact, the word 'Typ' is used since 'Type' has a special meaning to Access (reserved words will be discussed later)

The third column shows the email address.

**Design View of a Select Query**

We are starting you out with a slightly complicated example; the simple one comes next <smile>. Figure 2-4 shows the QBE (Query-By-Example) grid and fieldlists for the data displayed in the select query of Figure 2-3.

**Figure 2-4 Design View of a Query**

The arrow pointing to the Types table on the Join Line signifies that this relationship is an outer join instead of an equi-join... this means that People records will show even if a related record is not present in the t_Types table.

<table>
<thead>
<tr>
<th>Field:</th>
<th>Table:</th>
<th>Sort:</th>
<th>Show:</th>
<th>Criteria:</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_People: [MainName] &amp; (&quot;+suffix&quot;) &amp; (&quot;+FirstName&quot;) &amp; (&quot;+MiddleName&quot;)</td>
<td>t_Types</td>
<td></td>
<td>Ascending</td>
<td></td>
</tr>
<tr>
<td></td>
<td>t_eAddresses</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**SQL View of a Select Query**

The SQL statement is what Access stores internally to render a query.

**Figure 2-5 SQL View of a Select Query**

When Ampersand & is used to combine values, it doesn't matter if one of the terms is not filled out. Whatever is specified will display.

```sql
SELECT [MainName] & (' '+[suffix]) & (' '+[FirstName]) & (' '+[MiddleName]) AS Fullname
    . t_Types.Typ
    . t_eAddresses.eAddress
FROM t_Types
    RIGHT JOIN t_people
    ON t_Types.TypeID = t_people.TypeID
    INNER JOIN t_eAddresses
    ON t_people.PID = t_eAddresses.PID
ORDER BY [MainName] & (', '+[suffix]) & (', '+[FirstName]) & (', '+[MiddleName]);
```

If the FirstName is Null, the literal comma won't show either. + is used so that if anything inside the parentheses doesn't have a value, the whole term within parentheses will be unknown (null).

When you specify the calculated field to be sorted, this is the SQL that Access will construct. It is more efficient, however, to use this as your **Order By** clause:

**ORDER BY** [MainName], [FirstName], [MiddleName];

The **semi-colon** at the end of the SQL statement indicates the end of the statement.

**Calculated Field**

This is a calculated field and will be called 'Fullname'

```sql
[MainName] & (', '+[suffix]) & (', '+[FirstName]) & (', '+[MiddleName]) AS Fullname
```

This equation concatenates (combines) information from main name, first name, middle name, and suffix into one place. Literal values, such as spaces and commas are concatenated where needed. Note that literal values can be enclosed in single or double quotes … the quote marks just have to be balanced.

**Definition:** **Concatenate** is used in database terminology to mean 'to combine', or connect, or link together. In databases, one reason it is important to separate data into the smallest possible unit is because it is **much** easier to concatenate than it is to parse.

**Definition:** **Parse** is a term meaning 'to separate'. If you store someone's full name in one field and decide you want just the first name, you would need to parse that out from the whole field.
Difference between + and &

& and + are both Operators

The standard Concatenation Operator is ampersand (&). If a term that is concatenated is Null (has no data; unknown), all terms will display if you use ampersand.

The Addition Operator is the plus sign (+) ... but, even if one of the terms has a value, the result will be Null if any term is Null (kind of like multiplying by 0). As in math, what is enclosed in parentheses will be evaluated first.

Null + "anything" = Null
Null & "anything" = "anything"

"something " + "anything" = "something anything"
"something " & "anything" = "something anything"

no difference because both of the terms have a value

Null + "" = Null
Null & "" = ""

(Null + " ") & "Lastname" = "Lastname"
(Null & " ") & "Lastname" = " Lastname"

in the second case, the parentheses do not make a difference, each term is concatenated -- and note the space in the result before Lastname

Do you see the difference between using + and using & ? For instance, if you want to add a space between first and last name but you are not sure that first name will be filled out, you can do this:

(Firstname + " ") & Lastname

What is in the parentheses is evaluated first -- then it is concatenated to what comes next

You might also want to do this:

(Firstname + " ") & (Middlename + " ") & Lastname

Combining + and & in an expression gives you a way to make the result look right without having to test if something is not filled out.

What if firstname is filled but nothing else? There will be a space at the end. Usually, this is not a problem but if you want to chop it off, you can wrap the whole expression in the Trim function, which truncates leading and trailing spaces.

Trim((Firstname + " ") & (Middlename + " ") & Lastname)
**Fieldlist**

*Figure 2-6 Fieldlist*  

A *fieldlist* is a list showing fields that are in a table or query. The titlebar shows the name of the table or query and its fields are listed below.

To add a fieldlist to a query:

1. **Click** the Show Table icon

2. OR
3. from the menu, choose →
4. **Query, Show Table …**

**Joins**

A *join* is a line between two fieldlists showing how they relate to each other.

In the query design example, the line between the table with People information and the table with eMail addresses is an *equi-join* (aka *INNER JOIN*). This means that only people records that also have an email address will be included.

When you **Right-Click** on a join line, you can choose to edit the Join Properties. You can also **Double-Click** a join line to bring up the properties window directly.

*Figure 2-7 Join Properties – Equi-Join*
An equi-join is not always desired. Another fieldlist in the example shows the type person (or company). What if the type is not filled out (TypeID field in the people table) but we have an email address? In that case, we still want the person and the email address to show. How do we do that? We can use an outer join, which is also called a Left join or a Right join depending on how the SQL statement is written.

**Figure 2-8 Join Properties - Right Join**

<table>
<thead>
<tr>
<th>Join Properties</th>
<th>Right: Table Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Left Table Name</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>t_Types</td>
<td></td>
</tr>
<tr>
<td><strong>Left Column Name</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TypeID</td>
<td></td>
</tr>
<tr>
<td><strong>Right Column Name</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TypeID</td>
<td></td>
</tr>
</tbody>
</table>

- **1:** Only include rows where the joined fields from both tables are equal.
- **2:** Include ALL records from 't_Types' and only those records from 't_PEOPLE' where the joined fields are equal.
- **3:** Include ALL records from 't_PEOPLE' and only those records from 't_Types' where the joined fields are equal.

**Data Example**

To better understand what is happening with the Join lines, assume that we have the following data in our tables:

<table>
<thead>
<tr>
<th>t_People</th>
<th>t_Types</th>
<th>t_eAddresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>PID</td>
<td>Fullname</td>
<td>TypeID</td>
</tr>
<tr>
<td>768</td>
<td>Camfield, Greyson D.</td>
<td>48</td>
</tr>
<tr>
<td>769</td>
<td>Camfield, Haden E.</td>
<td>50</td>
</tr>
<tr>
<td>35</td>
<td>Camou, Bruce Matthew</td>
<td>54</td>
</tr>
<tr>
<td>36</td>
<td>Camou, Lucille</td>
<td>55</td>
</tr>
<tr>
<td>1214</td>
<td>Campbell Car Wash</td>
<td>56</td>
</tr>
<tr>
<td>970</td>
<td>Campbell, Joslyn B.</td>
<td>57</td>
</tr>
<tr>
<td>971</td>
<td>Campbell, Laura</td>
<td>58</td>
</tr>
<tr>
<td>709</td>
<td>Cannon, Dalton M.</td>
<td>69</td>
</tr>
<tr>
<td>710</td>
<td>Cannon, Quinn G.</td>
<td>70</td>
</tr>
<tr>
<td>898</td>
<td>Carlson, Dylan H.</td>
<td>71</td>
</tr>
<tr>
<td>899</td>
<td>Carlson, Evan G.</td>
<td>76</td>
</tr>
<tr>
<td>1253</td>
<td>Carlton Camp Lodge</td>
<td>77</td>
</tr>
<tr>
<td>1340</td>
<td>Carpets To Go</td>
<td>78</td>
</tr>
<tr>
<td>1007</td>
<td>Carroll, Kaitlyn</td>
<td>79</td>
</tr>
<tr>
<td>1006</td>
<td>Carroll, Meredith V.</td>
<td>80</td>
</tr>
<tr>
<td>59</td>
<td>Carson Jr., Joel</td>
<td>81</td>
</tr>
<tr>
<td>60</td>
<td>Carson, Vanna S.</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Red indicates there is no corresponding

Bold means that this TypeID is used in the list of people

**Not all people shown have email addresses**

In this example, 'Campbell, Laura'
**PID in the email addresses table – so this name will not show in the query.**

The results of our earlier query example run on this data would net the following results:

<table>
<thead>
<tr>
<th>Fullname</th>
<th>Typ</th>
<th>eAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camfield, Haden E.</td>
<td>Friend</td>
<td><a href="mailto:Haden.Camfield@hotmail.com">Haden.Camfield@hotmail.com</a></td>
</tr>
<tr>
<td>Camou, Bruce Matthew</td>
<td>Friend</td>
<td><a href="mailto:Bruce.Camou@yahoo.com">Bruce.Camou@yahoo.com</a></td>
</tr>
<tr>
<td>Camou, Lucille</td>
<td>Friend</td>
<td><a href="mailto:Lucille.Camou@bellsouth.net">Lucille.Camou@bellsouth.net</a></td>
</tr>
<tr>
<td>Campbell, Joslyn B.</td>
<td>Friend</td>
<td><a href="mailto:Joslyn.Campbell@someschool.org">Joslyn.Campbell@someschool.org</a></td>
</tr>
<tr>
<td>Campbell, Laura</td>
<td></td>
<td><a href="mailto:Laura.Campbell@Soupy.net">Laura.Campbell@Soupy.net</a></td>
</tr>
<tr>
<td>Carlson, Evan G.</td>
<td>Friend</td>
<td><a href="mailto:Evan.Carlson@earthlink.net">Evan.Carlson@earthlink.net</a></td>
</tr>
<tr>
<td>Carlton Camp Lodge</td>
<td>Hotel/Motel</td>
<td><a href="mailto:Vacations@CarltonCampLodge.com">Vacations@CarltonCampLodge.com</a></td>
</tr>
<tr>
<td>Carroll, Kaitlyn</td>
<td>Professional</td>
<td><a href="mailto:Kaitlyn.Carroll@Company.com">Kaitlyn.Carroll@Company.com</a></td>
</tr>
<tr>
<td>Carroll, Meredith V.</td>
<td>Professional</td>
<td><a href="mailto:Meredith.Carroll@yahoo.com">Meredith.Carroll@yahoo.com</a></td>
</tr>
<tr>
<td>Carson Jr., Joel</td>
<td>Friend</td>
<td><a href="mailto:Joel.Carson@sealane.com">Joel.Carson@sealane.com</a></td>
</tr>
<tr>
<td>Carson, Vanna S.</td>
<td>Friend</td>
<td><a href="mailto:Vanna.Carson@earthlink.net">Vanna.Carson@earthlink.net</a></td>
</tr>
<tr>
<td>Carter Home Appliance</td>
<td>Home Suppl/Serv</td>
<td><a href="mailto:CustomerService@CarterHomeAppliance.com">CustomerService@CarterHomeAppliance.com</a></td>
</tr>
</tbody>
</table>

Only people that have an email address are displayed, regardless of whether or not their type has been specified.

**Simple Query Example**

Now that we started you out with a complicated query that pulled data from three tables and used different types of joins … how about an easy example?

Figure 2-9 shows a simple list of people sorted by the main name (last name for humans or company name for companies) and then the first name.

**Figure 2-9 Simple Query Example**
Date Functions in Queries

In our table of people, one of the fields is DOB (Date of Birth). Suppose we want to list birthdays sorted by month and then day? This example shows some of the cool things you can do with dates! By the way, you can usually use the same functions in Excel that you see for Access!

Here are some of the results:

<table>
<thead>
<tr>
<th>BDay</th>
<th>Fullname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 25</td>
<td>Tyson Snider</td>
</tr>
<tr>
<td>Jan 26</td>
<td>Alfonso A. Tenney</td>
</tr>
<tr>
<td>Jan 27</td>
<td>Gary E. Summey</td>
</tr>
<tr>
<td>Jan 27</td>
<td>Rudy Bradley</td>
</tr>
<tr>
<td>Jan 28</td>
<td>Pedro R. McIntyre</td>
</tr>
<tr>
<td>Jan 30</td>
<td>Anna E. Deming</td>
</tr>
<tr>
<td>Jan 30</td>
<td>Roberto K. Gaylord</td>
</tr>
<tr>
<td>Feb 2</td>
<td>Cade R. Miller</td>
</tr>
<tr>
<td>Feb 2</td>
<td>Fiona E. Renner</td>
</tr>
<tr>
<td>Feb 4</td>
<td>Joshua B. Robinson</td>
</tr>
<tr>
<td>Feb 5</td>
<td>Braxton L. Hansford</td>
</tr>
</tbody>
</table>

Notice that January shows before February … even though, alphabetically, February comes first! How is that done?

How do we show 'Jan 25' from a date that looks like this: 1/25/47?

Here is the design view of the birthday query (DOB is DateOfBirth):

Figure 2-10 Design View of Query Showing Birthdays

There are only 2 columns that show (note the checkbox) – **Bday**, which displays as a month and day (mmm d) and **Fullname**. The other columns on the grid are for sorting or filtering.

The **Month** function is used to return the month number (1-12) from **DOB**. It is on the grid but the 'Show' box is not checked – notice the 'Ascending' is specified in the Sort cell.
The **Day** function is used to return the day number from **DOB** for sorting purposes. It also does not show.

If more than one person has a birthday on the same day, the list will then be sorted by **FirstName**. Choosing just first name for the sort is more efficient than specifying the calculated column.

Sorting is from left to right in the QBE grid for the three ascending sorts specified.

The **Bday** column uses the **Format** function to return the 3 character month abbreviation (mmm) and then the day of the month (d).

Records are only shown if **DOB** has a value – **Is Not Null** means the field is filled out <smile>

**Query Criteria**

*To be written.*

**Forms**

Forms provide a way to present data and/or menus and control the user's interaction. Forms can be **bound or unbound**. A **Bound** form is connected to a table or query by its **RecordSource** property (like a table or query – and remember that queries get their data from tables) and can be used to find, display, edit, and add data. **Unbound** forms, which have no **RecordSource**, can be used as menus or to hold subforms.

Forms are divided into different sections. The form header and footer sections display at the top and bottom. There are also optional Page Header and Page Footer sections that show at the top and bottom of each page when a form is printed.

The **Detail** section shows for each record.

In each section are controls. **Textbox**, **combobox**, **listbox**, **checkbox**, **option group**, **option button**, and **image** controls can be used to display and collect information. These types of controls can be bound to a field, or unbound, similar to binding a form to a **RecordSet**. **Command button** controls are often used for menus. **Tab controls** are used to organize information. Other types of controls include **label**, **line**, **rectangle**, as well as controls with complex information such as **subform/subreport**, calendar, video, and sound.

When a form is bound, controls that hold data can also be bound; in other words, this means that the **ControlSource** is a field in the table or query. Binding a form and then binding (connecting) controls on the form to fields in the **RecordSource** allows you to take advantage of many of the built-in features of Access like record navigation and data manipulation.

**Different Ways to View Data in a Form**

Forms can be set up to show one record at a time or multiple records at a time. They can also pivot information in a table layout or show graphs. The default in Access, is to allow all views for a form. However, if you are creating an application for other users, it is generally best to restrict the allowed view. We will focus on the 3 main types of forms:

- **Single Form View**
- **Continuous Form View**
- **Datasheet View**
**Field vs Control**

It is important to understand the difference between a field and a control. A field is a column in a table. A calculated field is an expression. A control can be a container for a field – but not all controls can be bound to fields – examples of controls that cannot be bound are label, line and rectangle controls.

The only way for a field to be placed on a form (or anything else for that matter) is in a control.

**Form View**

This is an example of a main form with subforms (creating a main form and subforms is covered in more detail later in this document). The main form is based on the People table. Subforms are used to enter information into related tables such as Addresses, Phones, and Email. A tab control is used to organize the subforms on the main form. Records displayed in subforms are linked to records from the parent table displayed on the main using the unique identifier for the parent table which, in this case, is **PID**.

**Figure 2-11 Form View - Main Form and Subforms**

**Design View of a Form**

This form uses 3 sections, Form Header, Detail, and Form Footer. Hidden controls (those with the Visible property set to No) use a white foreground color on a black background – that way they show up well in design view.

Figure 2-12 shows:

- Titlebar across the top indicating you are looking at an Access form named f_PEOPLE in a project named Contacts.
• Menu bar (File, Edit, View, …)
• The Form Design and Formatting toolbars are displayed on two rows.
• The Toolbox is docked on the left.
• The horizontal ruler across the top and vertical ruler on the left (from the menu: View, Rulers)
• Grid, which shows up as dots to help line up controls (from the menu: View, Grid)
• Three sections: Form Header, Detail, and Form Footer
• A Tab Control for switching between different types of information such as Addresses, Phone, and Email
• Two subform controls on the Address tab: one for main address information and the other for additional address lines.

Figure 2-12 Design View of a Form

Subform controls are used to contain forms for Addresses and extra lines for addresses. Being an Aussie, Microsoft Access MVP Allen Browne asked how I would handle foreign addresses and the
example he gave me had about 6 lines! As an American with somewhat standard addressing, I wasn't familiar with some of the outlandish addresses that some of you have! Well, one of the wonderful things about Access is that a table to hold what I call "standard" addresses could be set up – and then another related table allowing as many more lines as desired!

There are other subforms on other tabs – Phone, Email, Web Pages, Lists, and Relationships. On the "Change IDs" tab is a combobox to pick an ID to change all related references to so that a duplicate person can be deleted without losing any of the additional information that may have been entered.

While I realize all you have is this document and not the Contact Management application, I am leaving it up to your imagination to see some of the many ways a form can be used. If you do wish to see the Contact Management system this document references, it is available for download from:

**Contacts -- Names, Addresses, Phones, eMail, Websites, Notes**  

To download files from UtterAccess, you need to be a member – but it is free to join and is a great resource!

### Reports

Reports are used to format, calculate, and display data. The power of reports is in being able to sort and group up to 10 levels, make everything look nice with extensive formatting capabilities, and execute code in several places. Like forms, reports contain various types of controls.

**Print Preview of a Report**

Figure 2-13 shows a report with 2 columns. Its `RecordSource` is a query called q_Birthdays. In the lower-left corner is a navigation control to go to different pages of the report.

The number of birthdays listed is reported at the end of each month. Information is grouped by month and then sorted.
Figure 2-13 Print Preview of a Report - Two Columns

### Birthdays

<table>
<thead>
<tr>
<th>Month</th>
<th>Name</th>
<th>Birth</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>January</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Axel A. Creager</td>
<td>Sat 1977</td>
<td>30  M</td>
</tr>
<tr>
<td>2</td>
<td>Larry K. Conner</td>
<td>Thu 1976</td>
<td>31  M</td>
</tr>
<tr>
<td>3</td>
<td>Mary Sue Test</td>
<td>Tue 1980</td>
<td>27  F</td>
</tr>
<tr>
<td>4</td>
<td>Vance Fitzpatrick</td>
<td>Sat 1993</td>
<td>24  M</td>
</tr>
<tr>
<td>5</td>
<td>Alfred Gooden</td>
<td>Fri 1976</td>
<td>31  M</td>
</tr>
<tr>
<td>6</td>
<td>Martha Burk</td>
<td>Sun 1985</td>
<td>22  F</td>
</tr>
<tr>
<td>7</td>
<td>Solomon I. Dickson</td>
<td>Sun 1979</td>
<td>29  M</td>
</tr>
<tr>
<td>8</td>
<td>Marilyn Roden</td>
<td>Sat 1994</td>
<td>23  M</td>
</tr>
<tr>
<td>9</td>
<td>Roger P. Allen</td>
<td>Thu 1988</td>
<td>19  M</td>
</tr>
<tr>
<td>10</td>
<td>Roselyn A. Barrier</td>
<td>Wed 1981</td>
<td>25  F</td>
</tr>
<tr>
<td>11</td>
<td>Alyssa C. Roden</td>
<td>Sat 1976</td>
<td>31  F</td>
</tr>
<tr>
<td>12</td>
<td>Clark W. Starks</td>
<td>Wed 1978</td>
<td>32  M</td>
</tr>
<tr>
<td>13</td>
<td>Tamara Hartsaw</td>
<td>Fri 1979</td>
<td>25  F</td>
</tr>
<tr>
<td>14</td>
<td>Jared Townsend</td>
<td>Sun 1977</td>
<td>24  M</td>
</tr>
<tr>
<td>15</td>
<td>Noah Baldwin</td>
<td>Thu 1982</td>
<td>25  M</td>
</tr>
<tr>
<td>16</td>
<td>Ralph J. Layton</td>
<td>Fri 1977</td>
<td>30  M</td>
</tr>
<tr>
<td>17</td>
<td>Dean Porter</td>
<td>Thu 1981</td>
<td>26  M</td>
</tr>
<tr>
<td>18</td>
<td>Phoenix L. Farhay</td>
<td>Tue 1980</td>
<td>27  F</td>
</tr>
<tr>
<td>19</td>
<td>Vernon L. Barker</td>
<td>Thu 1981</td>
<td>25  M</td>
</tr>
<tr>
<td>20</td>
<td>Orval S. Souza</td>
<td>Mon 1984</td>
<td>23  M</td>
</tr>
<tr>
<td>21</td>
<td>Heath S. Branson</td>
<td>Wed 1979</td>
<td>28  M</td>
</tr>
<tr>
<td>22</td>
<td>Kenneth Sullivan</td>
<td>Wed 1978</td>
<td>29  M</td>
</tr>
<tr>
<td>23</td>
<td>Nicole A. Cole</td>
<td>Fri 1978</td>
<td>29  F</td>
</tr>
<tr>
<td>24</td>
<td>Terence Teli</td>
<td>Fri 1994</td>
<td>23  M</td>
</tr>
<tr>
<td>25</td>
<td>Alan Vanschoick</td>
<td>Mon 1986</td>
<td>27  M</td>
</tr>
<tr>
<td>26</td>
<td>Scarlet Wheeler</td>
<td>Sat 1978</td>
<td>29  F</td>
</tr>
<tr>
<td>27</td>
<td>Gregory G. Sweeney II</td>
<td>Sun 1977</td>
<td>30  M</td>
</tr>
<tr>
<td>28</td>
<td>Lillie Teter</td>
<td>Sat 1992</td>
<td>25  F</td>
</tr>
<tr>
<td>29</td>
<td>Neil Malcolm</td>
<td>Wed 1979</td>
<td>28  M</td>
</tr>
<tr>
<td>30</td>
<td>Jameson R. McKain</td>
<td>Mon 1987</td>
<td>25  M</td>
</tr>
<tr>
<td>31</td>
<td>Priscilla J. Jenkins</td>
<td>Fri 1980</td>
<td>20  F</td>
</tr>
<tr>
<td>32</td>
<td>Tommie Parks</td>
<td>Wed 1984</td>
<td>23  F</td>
</tr>
<tr>
<td>33</td>
<td>Tatiana D. Neese</td>
<td>Sun 1987</td>
<td>20  F</td>
</tr>
<tr>
<td>34</td>
<td>Tyson Snider</td>
<td>Thu 1977</td>
<td>30  M</td>
</tr>
<tr>
<td>35</td>
<td>Afonso A. Tenney</td>
<td>Fri 1979</td>
<td>29  M</td>
</tr>
<tr>
<td>36</td>
<td>Gary E. Sumney</td>
<td>Wed 1981</td>
<td>25  M</td>
</tr>
<tr>
<td>37</td>
<td>Rudy Bradley</td>
<td>Tue 1987</td>
<td>20  F</td>
</tr>
<tr>
<td>38</td>
<td>Pedro R. McIntyre</td>
<td>Wed 1981</td>
<td>26  M</td>
</tr>
<tr>
<td>39</td>
<td>Anna E. Deming</td>
<td>Fri 1976</td>
<td>31  F</td>
</tr>
<tr>
<td>40</td>
<td>Roberto K. Gaylord</td>
<td>Tue 1986</td>
<td>17  M</td>
</tr>
<tr>
<td></td>
<td><strong>February</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Cade R. Miller</td>
<td>Mon 1976</td>
<td>31  M</td>
</tr>
<tr>
<td>3</td>
<td>Fiona E. Renner</td>
<td>Tue 1987</td>
<td>25  F</td>
</tr>
<tr>
<td>4</td>
<td>Joshua B. Robinson</td>
<td>Mon 1985</td>
<td>22  M</td>
</tr>
<tr>
<td>5</td>
<td>Bronson L. Hansford</td>
<td>Mon 1990</td>
<td>17  M</td>
</tr>
<tr>
<td>6</td>
<td>Marlene J. Grimes</td>
<td>Mon 1979</td>
<td>28  F</td>
</tr>
<tr>
<td>7</td>
<td>Renee R. Austin</td>
<td>Wed 1975</td>
<td>32  F</td>
</tr>
<tr>
<td>8</td>
<td>Hope Hammond</td>
<td>Sun 1977</td>
<td>30  F</td>
</tr>
<tr>
<td>9</td>
<td>Maude A. Fiddler Jr.</td>
<td>Fri 1981</td>
<td>26  F</td>
</tr>
<tr>
<td>10</td>
<td>Steven G. Peterson</td>
<td>Sun 1983</td>
<td>24  M</td>
</tr>
<tr>
<td>11</td>
<td>Ronald S. Bundy</td>
<td>Tue 1989</td>
<td>18  M</td>
</tr>
<tr>
<td>12</td>
<td>Kevin J. Bowe</td>
<td>Thu 1982</td>
<td>25  M</td>
</tr>
<tr>
<td>13</td>
<td>Xander G. Graham</td>
<td>Fri 1977</td>
<td>30  M</td>
</tr>
<tr>
<td>14</td>
<td>Juliet K. Smith</td>
<td>Sun 1989</td>
<td>18  F</td>
</tr>
<tr>
<td>15</td>
<td>Miss Betty J. Zeigler</td>
<td></td>
<td>27  F</td>
</tr>
<tr>
<td>16</td>
<td>Ms. Dore Young</td>
<td>Tue 1989</td>
<td>17  M</td>
</tr>
<tr>
<td>17</td>
<td>Karen Davies</td>
<td>Mon 1982</td>
<td>25  F</td>
</tr>
<tr>
<td>18</td>
<td>Layle McCormick</td>
<td>Tue 1992</td>
<td>24  F</td>
</tr>
<tr>
<td>19</td>
<td>Scott Rapp</td>
<td>Wed 1976</td>
<td>29  M</td>
</tr>
<tr>
<td>20</td>
<td>Vincent Boll</td>
<td>Sat 1976</td>
<td>21  M</td>
</tr>
<tr>
<td>21</td>
<td>Clyde C. Benedum</td>
<td>Mon 1983</td>
<td>26  M</td>
</tr>
<tr>
<td>22</td>
<td>Jon R. Hart</td>
<td>Sun 1974</td>
<td>33  M</td>
</tr>
<tr>
<td>23</td>
<td>Pablo Watson</td>
<td>Sun 1970</td>
<td>27  M</td>
</tr>
<tr>
<td>24</td>
<td>Peter Franklin</td>
<td>Sat 1978</td>
<td>28  M</td>
</tr>
<tr>
<td>25</td>
<td>Joslyn Campbell</td>
<td>Fri 1982</td>
<td>25  F</td>
</tr>
<tr>
<td>26</td>
<td>Morris N. Baldwin</td>
<td>Fri 1982</td>
<td>25  M</td>
</tr>
<tr>
<td>27</td>
<td>Trinity S. McCormick</td>
<td>Fri 1982</td>
<td>25  F</td>
</tr>
<tr>
<td>28</td>
<td>Darien C. Bumett</td>
<td>Sat 1976</td>
<td>31  M</td>
</tr>
<tr>
<td>29</td>
<td>Hailey Harper</td>
<td>Thu 1989</td>
<td>27  F</td>
</tr>
<tr>
<td>31</td>
<td>Toby Knauff</td>
<td>Sat 1940</td>
<td>37  M</td>
</tr>
<tr>
<td>32</td>
<td>Jose Van</td>
<td>Wed 1982</td>
<td>25  F</td>
</tr>
<tr>
<td>33</td>
<td>Merilee R. Miles</td>
<td>Wed 1982</td>
<td>25  F</td>
</tr>
<tr>
<td>34</td>
<td>Yvonne C. Morrison</td>
<td>Sat 1942</td>
<td>25  M</td>
</tr>
<tr>
<td>35</td>
<td>Justine Ryan</td>
<td>Sat 1981</td>
<td>26  M</td>
</tr>
<tr>
<td>36</td>
<td>Marco J. Cheveaux</td>
<td>Sat 1982</td>
<td>26  M</td>
</tr>
<tr>
<td>37</td>
<td>Miles G. Knauff</td>
<td>Wed 1990</td>
<td>17  M</td>
</tr>
<tr>
<td>38</td>
<td>Owen Lahma</td>
<td>Tue 1989</td>
<td>18  M</td>
</tr>
</tbody>
</table>

**36 Birthdays for February**

### March

<table>
<thead>
<tr>
<th>Month</th>
<th>Name</th>
<th>Birth</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joyce T. Deaver</td>
<td>Sat 1980</td>
<td>27  F</td>
</tr>
<tr>
<td>2</td>
<td>Keaton Tenney</td>
<td>Sun 1991</td>
<td>26  M</td>
</tr>
<tr>
<td>3</td>
<td>Helena Everhart</td>
<td>Thu 1973</td>
<td>35  F</td>
</tr>
<tr>
<td>4</td>
<td>Keira Cunningham</td>
<td>Sat 1980</td>
<td>22  F</td>
</tr>
<tr>
<td>5</td>
<td>Amanda K. Paulus</td>
<td>Sat 1979</td>
<td>20  F</td>
</tr>
<tr>
<td>6</td>
<td>Vivian A. Siders</td>
<td>Thu 1988</td>
<td>19  F</td>
</tr>
</tbody>
</table>

34 Birthdays for January
**Design View of a Report**

**Report Sections**

Figure 2-14 shows the design view of a report with seven sections. The **Report Header** section is set to zero height, but it needs to show so that the **Report Footer** section, which counts all the birthdays listed on the report, can be used.

At the top of each page will be the information in the **Page Header** section. It appears that the **Page Footer** section has nothing … but if this report is limited, for instance to just people that are friends, the criteria is programmed to show in this area.

The **Mnth Header** section is used to break the report into months like January, and February.

The **Mnth Footer** section contains a calculated control to count the number of birthdays listed.

Finally, the **Detail** section is where the data for each record goes.

*Why use Mnth instead of Month for the calculated fieldname and, therefore, the section names?*

Notice in this report, we have group sections called 'Mnth Header' and 'Mnth Footer'. Why are they not called 'Month Header' and 'Month Footer'? It is important to avoid using reserved words (a link to Microsoft Access MVP Allen Browne's list to check for 'Problem names and reserved words' is listed in the Normalization section of this document).

The word "Month" has a special meaning to Access – **Month(date)** is a function that returns a month number, 1-12, from a date. It is a good idea to choose names that will not be confused with other things – and that is why Mnth has been spelled without the 'o' <smile>.

**Sorting and Grouping**

This report has 3 levels of sorting and grouping as seen in the **Sorting and Grouping** window. It is grouped by the month number (even though the month name is what shows up on the report). Then, it is sorted by the day number and then fullname. A common misconception is that sort orders applied in a query, used as a **RecordSource** for a report, should apply. In fact, you must use the **Sorting and Grouping** feature to specify the desired sorting.

**Fieldlist**

You also see the **Fieldlist** for the **RecordSource**, q_Birthdays.

**Properties**

Also showing is the **Properties** window.
Macros

Macros provide an easy way to increase the functionality of your application without having to write code. Macros are quick and easy to create but are limited in what tasks they can perform, and errors that might occur are difficult to handle. Macros can be convenient but they do have tradeoffs.
A macro stores a list of actions to be performed. There are about 50 actions to choose from, many of which have parameters that can be specified. Traditionally, macros could not use variables or programming logic nor have error handlers. Starting with Access 2007, macros can use temporary variables (tempvars) and include error-handling. However, in all prior versions, there was no provision for tempvars or error handling. Not being able to use variables or trap for errors and present a graceful error message are a couple reasons that macros gained a bad reputation with many developers.

**Design View of a Macro**

A typical macro action would be to open a particular form (OpenForm) when a command button is clicked. Notice that once OpenForm is specified as the Action, you are prompted for additional information in the lower pane such as the name of the form. Help is displayed on the right, depending on where your cursor is.

**Figure 2-15 Design View of a Macro**

Figure 2-16 shows a very simple macro that maximizes the windows within the Access application. Because it is named "autoexec", it will be automatically executed when the database is opened. Notice that the Action column gives you a list of choices to pick from. In the lower pane on the right is a description of the chosen action.

**Figure 2-16 AutoExec Macro**
**Macros for a Form**

In our database, we have a form to add an address as shown in Figure 2-17.

**Figure 2-17 Form to Add Address for Macro Example**

The user will type data into the form. Before the data is added or changed in the table that the form is based on, we will ask the user if they want to save changes. We will also want to let the user close the form.

To accomplish these two tasks, we will make a new macro group and call it `AddAddress` since it will be used for our AddAddress form. This is the name that will display in the Database Window.

**Figure 2-18 Macro Group Name shows in Database Window**

Our macro group will contain more than one macro so we will display the Macro Name column.

We also need to display the Condition column since we will ask the user a question and, based on their answer, we will choose to do something.

Here are the two macros we will create:

**SaveChanges**

The User will be given a message box with Yes and No buttons on the form BeforeUpdate event. If they choose No, two things will happen

1. *their changes will be undone*

   Action = RunCommand
   Command = Undo
2. the form BeforeUpdate event will be cancelled and their changes discarded
   Action = CancelEvent

CloseForm

We have a Close command button on the form and when the user clicks on it, the form will close.

Action = Close

Figure 2-19 Design View of Macro showing Macro Name and Condition Columns

<table>
<thead>
<tr>
<th>Macro Name</th>
<th>Condition</th>
<th>Action</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SaveChanges</td>
<td>MsgBox(&quot;Save Changes?&quot;, 4, &quot;Address&quot;) = 7</td>
<td>RunCommand</td>
<td>4 means Yes and No buttons will appear, 7 means No was chosen</td>
</tr>
<tr>
<td>CloseForm</td>
<td></td>
<td>Close</td>
<td>Cancel saving the record</td>
</tr>
</tbody>
</table>

Assigning macros to events

When you Click in an event property and the Click on the drop-down arrow, you will see a list of macros in your database. If you wish to write code, you can choose [Event Procedure], which is covered in the Database Objects section of this document.

Click in the upper left where the rulers intersect to select the form.

Figure 2-20 shows a red circle around the black square in upper left indicating that the form is selected as opposed to an object on the form.

On the Property Sheet, choose the Event tab, as shown by the blue circle.

Click in the BeforeUpdate Event and choose the SaveChanges macro in the AddAddress macro group.

Figure 2-20 Assign a Macro to the Form BeforeUpdate Event
The **SaveChanges** macro from the **AddAddress** macro group is now assigned to the **BeforeUpdate** event of the form as shown in Figure 2-21.

**Figure 2-21 Property Sheet for Form showing Macro Assignment**

![Property Sheet for Form showing Macro Assignment](image)

Next, we need to assign a macro to the Click event of our Close Command Button.

**Click** the command button to select it and then assign the **CloseForm** macro from the **AddAddress** macro group

The **CloseForm** macro from the **AddAddress** macro group will be assigned to the On Click event of our Close command button as shown in Figure 2-22.

**Figure 2-22 Macro Assigned to Click Event of Command Button**

![Macro Assigned to Click Event of Command Button](image)

In case you are wondering how we got a command button colored blue ... we didn't. A label control was used to make the blue 'Close' rectangle. Then a command button was laid on the top and its Transparent property was set to Yes. To demonstrate this better, the transparent command button has been moved slightly to the left as shown in **Error! Reference source not found.**
AutoKeys Macro

An AutoKeys macro gives you a way to set up global shortcut keys for your application. In an AutoKeys macro, the Macro Name parameter is used to define the keystroke.

Key names are surrounded with curly braces.

{ESC}
{F12}
{ENTER}

The carrot symbol ^ is used to mean Ctrl

The Plus symbol + is used to mean Shift

You can look at the Help for SendKeys to find the other codes that can be used for AutoKeys -- except for the Alt key, which is not defined for use in an AutoKeys macro.
AutoKeys Example

Figure 2-24 shows that we have defined $\text{Ctrl}+F$ to open the form to Find People.

Figure 2-24 AutoKeys Macro showing how to set a Global Shortcut key

Modules

Modules are used for storing Visual Basic for Applications (VBA) code (programs). Module code can do anything a macro can do (with the exception of AutoKeys) … and multitudes more! The ability to control Access using logic and loops is where the awesome power truly gets unleashed.

Unlike sequential programming, where a task is coded from input to report and cannot collect user input during this time, Access is event-driven … because the user clicked a button, or changed a record, or opened a form.

Code procedures are often short and respond to something that the user has requested or done such as clicked on a button or modified data in a control. Code can also be set on a timer or launched by other applications.

Code that is stored with a form or report is called a Class Module. Code that is stored in an independent module sheet is a Standard (or General) Module.

What is Me?

When you are in the code behind a form or report, Me refers to the form or report itself. Me is not necessary unless there is a conflict with names – using it makes things more clear. If you have Me.Total in code, you know it is referring to a control or a field named Total. If you just use Total, it could also mean a variable.
Another advantage of using Me is this: when you type Me. in code, Access IntelliSense will prompt you with a list of choices so it makes coding easier and faster. Let us assume we have a command button to close the form when the user clicks on it. The name of the command button is 'cmdClose' and this would be the Click event.

```vbnet
Private Sub cmdClose_Click()
    '--------------- ignore Errors
    On Error Resume Next
    '--------------- save record before closing form
    ' Add Note: Do this test for bound forms (forms with a record source) only.
    'test to see if record needs to be saved
    If Me.Dirty = True Then
        Me.Dirty = False
    End If
    '--------------- if record is still 'dirty' then it was not saved
    ' so don't close
    If Me.Dirty = True Then
        Exit Sub
    End If
    '--------------- close the form and do not save design changes
    DoCmd.Close acForm, Me.name, acSaveNo
End Sub
```

After you choose Close from the list of actions, you are prompted to specify other information, called arguments. Arguments are separated by commas, and can be positional or named arguments. Positional arguments must be specified in the correct order; named arguments can be given in any order, and without having to include more than one comma in a row. This example used positional arguments.

The first argument is the type of object. acForm is a built-in constant that evaluates to 2, which means the object is a form.

Next, Access wants to know the name of the object. Me.name is used to indicate the name of the form that the code is attached to.

Note: Instead of using Me, you could specify the name of the form as a string, however, doing so means that your code is less portable. If your form is a bound form (i.e. includes a RecordSource), you should save the record before closing the form. If the 'Dirty' property of the form is true, that means the current record has changes that are not saved. The reason that 'Dirty' is tested twice is because, after the first test, if the record still has unsaved changes then it did not pass validation and the user still needs to take action so the closing of the form should not happen. This will help prevent "losing" data, as discussed in this excellent tip by Access MVP Allen Browne:

**Losing data when you close a form**
http://allenbrowne.com/bug-01.html

Whenever you close a form that has been opened for the user to edit data, it is best to instruct Access not to save the form (you do not want to replace the version of the form that you, as the developer,
created and saved), which is why the constant, `acSaveNo`, is used and specifies that the form’s DESIGN should not be saved. This (`acSaveNo`) should always be specified to avoid problems; otherwise Access automatically saves property settings such as the Filter property.

**Testing for Conditions in Code**

Take a breath … we are going to look at more code ….

Many of you have probably heard the term "If-Statement". Life is full of choices. If one thing happens, it will cause another to happen. Simply put: If this, then that.

In terms of VBA, an IF statement tests a condition and, depending on the outcome, does one thing or another – or nothing. A condition is a numeric or string expression that evaluates to TRUE or FALSE.

The order that words have to be written is called syntax. Optional parts are shown in brackets.

```
If condition Then
    [statements]
[ElseIf condition-n Then
    [elseif_statements]]
[Else
    [else_statements]]
End If
```

*Example:*

```
If Me.ST = 'TX' Then
    MsgBox "Texas is a great state ... but it sure gets hot!"
Else
    MsgBox "If you move to Texas, make sure you have a swimming pool!"
End If
```

*WHERE:*

ST is the Name of a control containing the State Abbreviation on the form this code is behind.
Msgbox is a function built-in to Access to display a message.
**Design View of a Module**

Figure 2-25 shows the Visual Basic Editor (VBE) and is where you create and edit module code. The name of the module we are looking at, bas_crystal_code_general_07102, is highlighted in gray in the Project Explorer Window shown in the upper left. When you Double-Click on a module in the Project Explorer Window, its code will be shown on the right.

In the lower left, the Properties window is displayed and shows the Name property of the module. On the bottom right is the Immediate Window, formerly known as, and often still called, the Debug window. You can type a question mark followed by a function or sub and Access will execute it.

On the right is the code window. Remember the macro we just showed you to Maximize Access? The top routine does the same thing using VBA code – so don't be intimidated by code! It's fun! (geek bell is ringing loudly).

In code, comments are shown in green. These lines are added to describe what you are doing and are not part of the executable program code. Some of the keywords are shown in navy, and the rest of the program code is displayed in black.

The Object and Procedure comboboxes at the top give you a quick way to jump to specific places in code without scrolling through the module sheet.
You can use the Page Up and Page Down buttons to move up or down one screen at a time.

You can press Ctrl Page Up or Ctrl Page Down to move up one procedure at a time.

You can press Ctrl Page Down or Ctrl Page Up to move down one procedure at a time.

Ctrl Home moves the cursor to the top of the module.

Ctrl End moves the cursor to the end of the module.

Home moves the cursor to the beginning of a line.

End moves the cursor to the end of a line.

Indenting code is always a good idea. Press Tab to increase the indent or Shift Tab to decrease the indent. If you select one or more lines and press Tab or Shift Tab, you will increase or decrease the indent for all the lines selected.
IntelliSense

When you are typing code, starting with version 97, Access has really nice feature called IntelliSense that helps you write code by prompting you with choices. In this example, once DoCmd. is typed, Access shows you a list of macro actions (choices for DoCmd). As you type letters, the list index moves. So it IS easy to write code 😊

Objects, Containers, Collections

Everything tangible in Access is an object whether it is a table, query, form, field, or control. Objects are stored in containers and many objects of the same type are called a collection.

Properties and Methods

- Properties are like adjectives that describe an object.
- Methods are like verbs and define actions an object can do

For humans, properties include height, weight, eye color, hair color, and gender.

Human methods would be actions such as run, jump, eat, and sleep.

Table properties include name, date created, and a collection of fields. Field properties include the name, data type, size, and description.

Table actions include: add, delete, rename, and open.

Events

Events occur when the user does something such as Click on a form control, or change data in a form. When events happen, code can be launched.

Events that can be trapped at the form level include Open, Load (populate with data), Current (change which record is displayed), BeforeInsert (just before a record is added), AfterInsert (after a record is added), BeforeUpdate (before changes to a record are written), and AfterUpdate (after changes are written). There is a Timer event and you can specify the Time Interval to launch code as long as a form is open – be careful about using a timer, though, it can lead to strange problems.

Control events include Click, DoubleClick, BeforeUpdate, AfterUpdate, GotFocus, LostFocus, and MouseUp.

VBA

VBA (Visual Basic for Applications) is based on a simple set of logic control. Essential statements include testing conditions to determine which code to execute (IF, THEN, ELSE) and using loops to repeat tasks (DO WHILE, LOOP). Think in terms of discrete tasks that can be launched at the click of a button.

Code acts on objects and is triggered by events. Understanding the types of objects that are available and knowing the properties (describe an object, like adjectives describe nouns) and methods (actions that object can do, like verbs are for nouns) is paramount to writing VBA code.
Most of what you will learn as you dive into the exciting world of programming is that VBA is logical – it makes sense!... just as you would drive a car but not a piece of fruit; and eat fruit but not a car.

**Reference Libraries**

Much as a set of encyclopedias is a resource to look up information, reference libraries provide program code with additional information.

**VBA**

The VBA reference library includes generic functions (not specific to an application) such as:

* **math**

  - **Round** numbers. `Round(23.473,1) → 23.5`
  - **Abs** – get absolute value of a number. `Abs(-5) → 5`
  - **Sgn** – get sign of a number. `Sgn(-5) → -1`
  - **Rnd** – Random Number. `CLng(Rnd(Now())*100) → {Some random integer between 0 to 100}`

* **manipulate strings**

  - **Left** # letters. `Left("Access Database", 6) → "Access"`
  - **Instr** – determine position of a character or phrase. `Instr("Access Database", " ") → 7`
  - **Mid** – extract a sub string. `Mid("Smith, Mary",8, 4) → "Mary"
  - **Len** – get length of a string. `Len("Smith, Mary") → 11`

* **access the file system**

  - **Dir** – retrieve first filename from a particular location. `FileNam = DIR("c:\data\*.")`
  - **Name** – change filename and/or path. `Name "c:\data\MyFile.doc" AS "c:\data\Myfiles\NewName.doc"`
  - **FileCopy** – make a copy of a file. `FileCopy "c:\data\MyFile.xls", "c:\data\CopyOfMyFile.xls"`
  - **GetAttributes** of a spec such as Normal, Read-Only, Hidden, Directory. `GetAttr("c:\data") → 16`
  - 16 is the constant indicating a directory
  - **FileDialogTime** – date file last changed. `FileDialogTime("c:\data\AccessBasics.doc") → 8/26/2007`

* **convert (between data types – these are very important) --commonly known as "Type Conversion"**

  - **functions**
    - `CSng`, `CDbl`, `CCur`, `CInt`, `CLng`, `CDate`, `CStr`, etc.
      - to long integer: `CLng(1005.672) → 1006`
      - to date: `CDate(#8/26/07#) + 10 → 9/5/2007`
    - **Format** to change numbers* into strings. `Format(Date(), "dd-mmm-yy") → 26-Aug-07`
    - to change strings. `Format(" I want to be capital", ">") → I WANT TO BE CAPITAL`

*Dates are stores internally as numbers

* **Get and Set Dates and Times**

  - extract the **Day**, **Month**, or **Year** from a date. `Year( Date() ) → 2007`
  - **DateSerial** to create a date from year, month, day. `DateSerial(2007, 8+1, 26) → 9/26/2007`
  - extract the **Hour**, **Minute**, or **Second** from a date/time. `Minute(#8:30#) → 30`
  - **TimeSerial** to create a time from hour, minute, second. `TimeSerial(16, 30, 0) → 4:30:00 PM`
  - **DatePart** to extract intervals such as quarter, weekday, day of month. `DatePart("q",#1/1/2010#) → 1`
  - **DateAdd** to add/subtract specified intervals to/from date. `DateAdd("m", 3, #8/26/07#) → 11/26/2007`
**DateDiff** for difference between dates. \( \text{DateDiff("q", #1/1/07#, #12/5/07#)} \rightarrow 3 \)

**DateValue** to extract a date. \( \text{DateValue("Apr 3, 2007 8:30") \rightarrow 4/3/2007} \)

**TimeValue** to extract a time. \( \text{TimeValue("Apr 3, 2007 8:30") \rightarrow 8:30:00 AM} \)

There are functions for Financial calculations such as amortizing payments, Constants, Informational functions to check data type or errors, and much more. The VBA library is listed at the top of the order and cannot be moved because it is the first external library accessed to interpret code.

### Access

The objects in Access (tables, queries, forms, reports, macros, modules, etc) are defined in the Microsoft Access #.0 Object Library, where #.0 indicates the version number. The Access library is listed second and, like the VBA library, cannot be removed.

### ActiveX Data Objects

Provides the programmatic interface (i.e.: the VBA object types, methods and functions) which allow you to manipulate data using the ADO protocol. ADO database objects appear in the object browser as the **ADODB** library and is defined in the Microsoft ActiveX Data Objects #.# Library. ADO communicates with a variety of data sources through "**OLE DB Providers**". OLE DB Providers goes beyond the capabilities of ODBC (Open DataBase Connectivity). In simpler terms, ADO makes data stored in other formats outside of Access accessible, similar to DAO, but a broader range of sources can be communicated with.

### OLE

The ability to do things such as enable conversations with other applications is defined in the OLE (Object Linking and Embedding) Automation reference library, which uses COM (Component Object Model) technology.

### DAO

DAO stands for Data Access Objects. Using a Microsoft DAO library allows you to do things such as find, view and manipulate data and definitions in Databases, Tables (TableDefs), Queries (QueryDefs); to loop through objects a container holds (such as records of a table); and retrieve/put data from/into external files such as other databases. You can also do many of the same operations using ADO. There is some overlap between DAO and ADO but the methods they use are different.

The DAO Library was built-in to Access 97. Then, in versions 2000 and 2002, it had to be added by the user. Microsoft realized it was best to include it automatically and in version 2003 and above, DAO is again a default library.

### Excel

To get data from/ put data into Excel, and control the Excel objects, you can specify the Microsoft Excel #.# Object Library, which is not included by default. This is an example of using Early Binding. However, it is best to convert early bound automation code to use Late Binding (no checked library reference required), after you have finished writing and debugging your code.

For more information on binding:

**Late Binding in Microsoft Access, by Tony Toews**

[http://www.granite.ab.ca/access/latebinding.htm](http://www.granite.ab.ca/access/latebinding.htm)
Other libraries

There is a multitude of other libraries available in case you want to write code to include use of them such as MS Graph, HTML, Outlook, PowerPoint, Word, XML, and Scripting (the file system). If you are using the Object Browser to explore capabilities, you might temporarily reference another library to learn more about it. Be careful about keeping references to libraries that are not necessary. It is okay to reference one so that you can use the Object Browser to see what is in it, but then remove it if your code does not need it. The Object Browser is discussed in the Properties and Methods section of this document.

A broken reference will cause strange errors when you run an application. To "fix" this, you can often remove one of the libraries, then put it back. If any references are flagged as MISSING, uncheck it, click on the OK button to dismiss the References dialog.

Compile your code. If it compiles okay without the reference, then it is not needed. If you do need the reference, re-open this dialog and add the removed reference back in. If it does not appear in the list, use the Browse button to locate the file.

Note: You should not need to exit the database to fix a broken reference – but sometimes that is what it takes.

To see library references, from a module window, use this menu option Tools, References.Here are some articles on library references:

Solving Problems with Library References, by Allen Browne
http://allenbrowne.com/ser-38.html

Access Reference Problems, by Doug Steele
http://www.accessmvp.com/djsteele/AccessReferenceErrors.html

Another important consideration when referencing libraries is their order, or priority. Here is an article on Reference Priority.

ADO and DAO Library References in Access Databases, by Tom Wickerath
3. Normalizing Data
* storing data efficiently

|---   ...When you are planning your database...   ---|

Instead of thinking about what you want out ...

...think about what you have to put in and structure it well.

Okay, maybe I said that wrong, you do need to keep in mind what you will want to report, but that should not drive how you structure your tables; what you want out must be put in. Your table structure is driven by how things actually relate in the real world – looking at what you have from another perspective to see that can often be difficult.

Drawing an analogy to a building:

- Data structure is the foundation.
- Forms are walls.
- Reports are windows and skylights, since they allow you to view your data.

Organize your data so the structures are strong. You want what you build on that foundation to be stable and flexible.

Structuring data is an iterative process. Set up tables, create relations between tables and print relationship diagram, change tables, print relationship diagram, put data in, change tables, print relationship diagram, …

The best solution is simple... but it is the hardest to find.

It takes thought and foresight to design your structures well. And the simplest solution is rarely easy to see. Get the structure right before building anything else. If you have already built forms, queries, or other objects, fix the structure before building any more!

Just as you wouldn't build a house on sand without a strong foundation, your data structure is the foundation you will build on for Access. A good design is everything when it comes to Access. Structuring data is the most important thing to do well -- as you learn and your knowledge grows, it is important to translate that to data structure changes -- just as you would repair a crack in the foundation of a building the moment it is discovered.

It takes time to plan your database. Here are some ideas for you:

Think of the different "nouns" that you will track:

- Customers
- Addresses
- Cities
- Phones
- Notes
- Products
- Purchases
- Payments
- Campaigns

For each "noun", you describe it with "adjectives", or fields. Each table should have a primary key. It is common to use the AutoNumber field and "ID", such as CustomerID, ProductID, etc.
By convention, when "ID" is used in a fieldname, it indicates that field was originally created by an AutoNumber and it is a Long Integer data type.

Do not use "ID" as a fieldname all by itself as it is ambiguous and not descriptive.

**Don't repeat data structures**

If you see, for instance, that you have Department information in more than one table, create a Departments table with an AutoNumber DeptID and make a Long Integer DeptID in other tables to relate the data.

**Key Fields**

Key fields are used to link related tables together. The field names in the different tables should be the same (IMHO) when they represent the same piece of information. Long Integer key fields typically have "ID" in their name, such as ProductID, ProdID, TransactionID, TranID, and CustID.

**Data Type for Key Fields**

Don't use anything but Number (field size = Long Integer) or Text for key fields – and Long Integers are most efficient. Double-precision and single-precision numbers are not accurate for keys. If you do use a text field for a key, keep in mind that it needs to be short to be efficient. It takes about 1 byte to store each character whereas long integers only require 4 bytes to store.

Starting with JET 4 (Access 2000), each text character takes 2 bytes to store in order to support Unicode. This means that a 50-character text field takes 100 bytes to store – 25 times the space needed for a Long Integer.

**DefaultValue**

Change default value of all numeric foreign key fields to Null -- the default for all numbers is 0 (versions below Access 2007), which will never match with a sequential AutoNumber field -- not changing it will prevent you from being able to enforce referential integrity if it is not specified. If referential integrity is enforced, as it should be in most relationships (unless you have a good reason not to), a default value of 0 in a numeric foreign key field can cause other problems, such as append queries that fail.

To hammer that in...Access sets the DefaultValue of numeric fields to 0 -- this is not good for a foreign key because there won't be a key field to match it to: *It is okay if it is not filled out, but it is NOT okay if it is filled out with a value that doesn't match.*

For more discussion on setting the Default Value, see this thread: [http://www.utteraccess.com/forums/showflat.php?Cat=&Board=83&Number=1411316](http://www.utteraccess.com/forums/showflat.php?Cat=&Board=83&Number=1411316)

**Indexes**

You can also create indexes on fields in tables. For instance, you may have a combination of fields that should be unique, such as TestID and QuestionID, where you would not want the same question to appear on a test more than once. You can create a unique index on the combination.

Think of which fields you will be doing lookups on and build indexes, but do this judiciously as indexes are updated when records are changed, so they take extra time to maintain, and there is a limit to how many you can create. If you use dLookup, it will work faster on an indexed field.
This topic is so important and is covered a bit more in the **Relationships** section of this document. Here is a link you may wish to read as well:

**Use Microsoft Jet's ShowPlan to write more efficient queries, by Susan Sales Harkins and Mike Gunderloy**


**Lookup Fields in the table design**

Another thing Microsoft allows you do is define comboboxes as part of the table design. Don't do this. Data should be entered via forms, which is where you would set up comboboxes.

**The Evils of Lookup Fields in Tables**

http://www.mvps.org/access/lookupfields.htm

*Explore this website, mvps.org – it is an outstanding source of information.*

**Names**

DON'T use anything but letters, numbers, and underscores in fieldnames and tablenames.

Don't use special characters in names (%, &, /, etc). Personally, I don't even use spaces. Start all names with a letter not a number. Using numbers usually indicates that the data is not normalized anyway.

If you start a fieldname with a number, you WILL have problems, so don't ever do it. Access allows you to do many things that kick you in the butt later.

Think about how long text fields will be and set the Field size to something other than the default of 50 characters. For instance, for Americans, 30 is usually long enough for cities, 10 long enough for zips, 14 for phone numbers, 20 or 30 for last or first name. These are just guidelines, your data may need more space.

Keep names concise yet descriptive.

Name your fields and tables well. When you do create forms and reports, name the controls to something logical before building any code. Names like Text34 and Combo68 make it frustrating to code and even more so if you need help and others have to decipher what those names really mean. Personally, I like to make the **Name** property of the control the same as the **ControlSource** property whenever possible.

When you have command buttons, name them cmdClose, cmdOpenReportMenu, etc.

"Date" is a bad name for a field since that is a reserved word. Qualify fieldnames, like CourseDate and SaleDate. One reason to keep fieldnames short is consideration of the length of the field that the column heading will be over since column headings can't wrap when you open a table. Another is less to type when you are coding and … last but certainly not least – you can fit more on a relationship diagram (covered in the **Relationships** section of this document).

***

more considerations:

Although convention tells us to prefix tables with "tbl", I don't like to do that…takes a split second longer to scan for values. If you do want to group your tables, use something short, like t_

Which is easier to read?
tblCustomers
 t_Customers
 Customers

If you are going to use a long table prefix such as "tblABC" ...make it quicker to read the important part -- the actual table name.

Which can you read faster ...

tblABCLocation
 or
 tblABC_Location
 or
 tabc_Location

always keep Names concise yet descriptive

Table and query names need to be different. To make sure they are, I do recommend using 'q' at the beginning of a query – this also groups them together when presented with a list of tables and queries as well as makes it easy to see if the source of an object is a table or a query.

Reserved Words

In addition to naming your fields (tables, queries, etc) well, you also need to avoid using special words that Access might interpret to mean something different. A list of reserved words, and a free utility to scan your tables, can be found on Allen Browne's website:

Problem names and reserved words in Access, by Allen Browne
http://www.allenbrowne.com/AppIssueBadWord.html
Allen has a lot of great information on his site, I encourage you to explore it.

Numbers that aren’t numbers

If a "number" you plan to store can not be used in a mathematical expression, you may choose to store it as text, especially if it has symbols. For example, you would not do math with a telephone number, so you may wish to store it as text so that you can store the symbols and read the value easier (123) 555-1212, especially when you export your data. It is more efficient, however, to store numbers as numbers than text because the number of bytes of storage is less. Personally, I give that up for keeping the symbols.

In the case of a phone number, you could not even store it as a long integer because it has too many digits. You, therefore, would need to store it as a double-precision number and, because double-precision numbers are stored in floating point format, they are not accurate for exact comparisons.

On the other hand, a ranking, mass, or level, etc. can be used in math equations (even if you don't plan to use it that way!), so store that kind of data with a numeric data type.

InputMask

When you use the InputMask property on a text field, choose to store symbols in the field so when you do an export, they will be there. For instance, without storing symbols, a phone number would be 1234567890 when exported as opposed to 123-456-7890 or (123) 456-7890.

There are those that will disagree with me on this because it DOES make searching for phone numbers more difficult. There is a trade-off here … in my databases, I now use a CountryCode
(from the standard list) that links to a Countries table which also stores Phone InputMask; that information is used when phone numbers are created or edited. To search on phone number, I use a function to translate the phone number to just digits without mask symbols.

**Allow Zero Length**

Make sure Allow Zero Length is set to Yes for text fields if you are planning on importing or using APPEND queries to add data and fields may be empty strings. You need to be careful here as a ZLS (zero length string), or even a field with a space or two, looks like a Null value even though it is not.

There are (many!) others who do not agree with this and consider allowing zero length to be a problem. You need to decide for yourself what you will choose to do. Here is a link which cautions you NOT to AllowZeroLength:

**Problem Properties, by Allen Browne**


**Captions**

*Don't use captions for your fieldnames in your table designs!* This is a bad idea. Users should never enter information directly into a table. Therefore, the only people that should be opening the table directly are administrators and using captions hides the real field name. If you want to call "ItemID" something else like TaskID, then NAME it that – and use the Description property.

There are those who do like using captions as this will also be the Caption property for a label on a form that prompts for the field information.

**Descriptions**

*Fill out your field Descriptions!* This is what the status bar text will be set to when you slide that field onto a form. The StatusBarText shows up in the lower left corner of the screen on the Status Bar when you are in that field.

**Order of Fields**

The Order you list fields in the table design does not matter. Usually, all key fields are at the top of the structure and tracking fields, like DateAdded and DateEdited below are at the bottom.

**Tracking date record was created or modified**

It is a good idea to add these 2 fields to all your tables (except tables that are used to provide values for lists and won't be altered, such as a table of states in the United States). Let them be the last 2 fields.

```
DateAdded, date, DefaultValue = Now()
DateEdited, date -> set on the form BeforeUpdate event
```

the best way to use the DateAdded field is to set a DefaultValue of

```
=Now ()
```

in the table design.

For DateEdited, make sure it is your form RecordSet. Then, use the Form BeforeUpdate event to set the value

```
Me.DateEdited = Now()
```
**Combobox Example**

Do not store names in related tables, store the IDs. For instance, instead of storing "Sue Smith", store the PID (PersonID) that identifies her. This can be totally transparent to the user.

In most circumstances you should not store names in more than one place. For instance, if you have a People table, define a PID (or PeopleID) AutoNumber field. Then, in other tables, when you want to identify a person, you can use the key field. One way to do this...

Create an AutoNumber field in the People table -->

    PID, AutoNumber

then, in the other tables...

    PID, Long Integer, DefaultValue: leave blank

Then, when you want to put data in (which should be done from a form), you can set it up to pick names from a list but store the PID. Create a combobox control:

    Name --> PID
    ControlSource --> PID
    RowSource -->
        SELECT
            PID,
            LastName & ", " & Firstname AS Fullname,
            BirthDate
        FROM People
        ORDER BY LastName, Firstname
    BoundColumn --> 1
    ColumnCount --> 3
    ColumnWidths --> 0;2;1
        (etc for however many columns you have -- the ID column will be hidden)
    ListWidth --> 3.2
        (for a combobox, this is the sum of the column widths plus 0.2 for a scrollbar)

If you have a listbox, sometimes you need to make the WIDTH .01 more than the sum of the columns to prevent the horizontal scrollbar from appearing.

PID will be stored in the form RecordSource while showing you names from another table... a much better and more reliable method.

If you want to show other information from your combobox in other controls, you can use calculated fields. For instance

    textbox:
    Name --> BirthDate
    ControlSource --> = PID.column(2)
The reason that column 2 is referenced instead of column 3 is that column indexes start with 0, not 1, in Access.

**Combobox Properties to Set**

Rather than using a wizard to make your comboboxes, it is MUCH better to set the properties yourself, there aren't many of them, and then you know what was done 😊

**DATA TAB**

**ControlSource**

- Unbound if the control will be used to Find or Filter (etc) and its value will not be stored
- The name of a Field in the RecordSource of your form/report to display and edit data in your tables
- A calculation; for instance \( =\) [Quantity] \* [UnitCost]

Calculated controls must start with an equal sign, =

What appears IN the control when you are looking at the design view of a form or report is the ControlSource

Once you have chosen the ControlSource, copy the field name to the clipboard (use F2 to toggle between insertion point and selecting the whole value) and press Ctrl-C.

**OTHER TAB**

**Name**

This is how you will refer to your control in code, on queries, other forms, reports, etc -- so pick something logical! I like to make the Name the same as the ControlSource for bound controls.

Name cannot be the same as a fieldname in the RecordSource UNLESS it is bound to that field (in which case I recommend it)

click in the Name property and press Ctrl-V to paste the ControlSource value you copied

(back to) **DATA TAB**

**RowSourceType**

- 'Table/Query' if you are specifying a tablename (bad idea as you may change the order of your fields or modify your table structure), query name, or SQL statement as the RowSource
- 'ValueList' if you are specifying values as the RowSource
- 'FieldList' if your RowSource is a table/query/SQL and you want to list the field names
RowSource

Where Access will get the values to display for your list

I usually pick Table/Query as the RowSourceType and use an SQL statement for RowSource, something like this:

```
SELECT
    PID,
    [NameMain] & (', '+[NameFirst]) & (' ' & [NameMid]) AS Name,
    Code
FROM
    t_PEOPLE
ORDER BY
    [NameMain],[NameFirst],[NameMid];
```

If you are supplying a value list, such as choices for Gender --> M; Male; F; Female (assuming you will use 2 columns, and just store 1 letter)

**FORMAT TAB**

ColumnCount

The number of columns you wish Access to keep track of from the RowSource

*The above example has 3 columns*

*Column indexing starts at 0, so if I want to make a textbox control to show the Code associated with a name, assuming that PID is the name of my combobox, here is the equation for ControlSource --> =PID.column(2)*

ColumnHeads

True or False, depending on whether or not you want the column headings to be displayed as the first row when you drop the combo

ColumnWidths

A list, delimited by semi-colon, of the widths associated with each column

For our example --> 0";2.5";1"

*The first column will be hidden*

*The second column uses 2.5" to display the Name*

*The third column will be 1" to display the code*

ListRows

The number of rows displayed when you drop the combobox

By default, this value is 8

*I usually change it to 24*

ListWidth

How wide you want the combobox list to be

Add up the ColumnWidths and add 0.2" to allow for a scrollbar

\[ 0 + 2.5 + 1 + 0.2 \rightarrow 3.7" \]
Width

The width of the control

*In our example, the Name will show and we gave it a column width of 2.5 -- so, we should set of width of 2.5 as well (add .15" if you want to accommodate for the drop-down arrow)*

**EVENT TAB**

**OnMouseUp**

While this is not a critical property to set, it is very nice to make the list of a combo automatically drop when you click anywhere in the control (as opposed to just the arrow). Here would be the code in the [Event Procedure] -->

```
me.ActiveControl.Dropdown
```

**OTHER TAB**

**StatusBarText**

What will show up in the StatusBar (lower left corner of the screen) when you are clicked in the control

The display of the Status Bar can be toggled using --> Tools, Options, View tab --&gt; 'Status Bar' checkbox

*for example --> Name of Person or Company*

~~~

The list above is most important for a combo but here are other properties that you may decide to set, for instance:

**Format Tab** -- Format, Top, Left, BackStyle, BackColor, BorderStyle, BorderColor, ForeColor, FontName, etc

**Data Tab** -- InputMask, LimitToList, DefaultValue, Enabled, Locked

**Event Tab** -- launch code or a macro if you want something to happen – AfterUpdate of a specific control or the form, Click, BeforeUpdate to validate a specific control or another record before it is saved, etc.

**Other Tab** -- TabStop, ControlTipText, Tag

**Getting Help on Combobox Properties**

If you do not know what some of the properties are, click in the property on the property sheet and press the F1 key for Help -- Properties are very important as they describe an object (such as height, weight, eye color, etc describe a human)
More information on Normalization

While you are designing your database, the more you read on this subject, the better ideas you will get for your own needs. As with many of the ideas here, I say one thing and others say something different so you will just need to decide for yourself what is best. Here are links you may wish to study:

Access Junkie's (Jeff Conrad) Resource List, Database Design 101
http://www.accessmvp.com/JConrad/accessjunkie/resources.html#DatabaseDesign101

A Little Treatise on Normalization, by Jerry Dennison

Data Modeling, by George Hepworth (GroverParkGeorge)

Glossary of Terms and Definitions, by Jerry Dennison

General Primary Keys - Natural vs Surrogate, by Glenn Lloyd (argeedblu)

Microsoft: Object Hungarian Notation Naming Conventions for VB
http://support.microsoft.com/kb/q173738

Rules of Data Normalization
http://www.datamodel.org/NormalizationRules.html

Training Day 1 (TD1): Introduction to Relational Database Design, by The Gunny
http://www.access.qbuilt.com/html/rdb_design_boot_camp.html

Training Day 2 (TD2): First Normal Form, by The Gunny
http://www.access.qbuilt.com/html/rdb_design_boot_camp1.html

There are a lot of things to read! And it may be hard to justify the time on design when your boss wants a database he can use … but as the familiar saying goes: "Measure twice cut once". With Access, it is more like "Measure 50 times, cut once". The time you spend up-front creating a solid structure will save you endless headaches down the road.
4. Relationships, Referential Integrity, Indexes

Steps to Document Table Structure using Relationship Diagram

To make it easier to see your tables and relationships

1. Enforce referential integrity on all relationships (unless you have a good reason not to)

2. Put the table for the "1" side of the relationship on the left and the "many" side on the right. Because we read from left-to-right, it is more appropriate to visualize creation of data that way.

... data diagrams should FLOW, just as data does -- in the same direction so it is easier to visualize (very important to get the best ideas)

A trick if you have a field list longer than your screen (to get all fields to show) is to put it on the relationship diagram twice. Place the copy next to the first one (I usually place it to the right and line it up with the bottom). Adjust the height of the copy and the scrollbar position to show the fields at the end -- then take a screen shot.

To print it out, this is a good method:

3. Stretch all your field lists to show all fields and put all tables on even if they don’t have relationships to anything

4. Press Print Screen to copy screen to Windows clipboard

5. Open Paint (Start, Accessories, Paint)

6. Paste Ctrl V (at this point, I like to select part of the image, copy the selection, and make a new Paint document)

Crop the diagram so there is nothing extra in the picture

Change the background to white, it is easier to write on your diagram

Notice how tables are arranged so that the "1" side of the relationship is on the left and the "many" side is on the right.
7. Save As...
→ file type = JPG or PNG or GIF
→ filename = "Rel_dname.jpg" -- just keep overwriting
→ location: put it into the directory with your database

A JPG file is about 1/10th the file size with no visible loss of resolution (as long as you don't make it bigger) compared to the default format of bitmap (BMP).

8. Then, you can open Word or PowerPoint and Insert, Picture ➤ From File…

You can also skip steps 4-8 and print your relationship diagram from the relationship window using File, Print Relationships…

If you have a program like SnagIt, by TechSmith, you can skip a lot of these steps. SnagIt was used to do all the screen captures for this publication.

The reason I like to save the relationship diagram as an image(JPG or PNG or GIF) file is because I prefer to use PowerPoint so that I can add comments to the diagram -- also, most of the databases I work on have too many tables for one screen. Allen Browne also has a utility which is quite useful if you do not have a zillion tables in your database <smile>.

**Relationship Report With Extended Field Information By Allen Browne**
http://allenbrowne.com/AppRelReport.html

Keep a relationship diagram in front of you at all times whenever you work on a database, it is an invaluable reference!

Once your relationships are printed, put the printouts where you can spend time studying them … by your bed, in the bathroom, in your briefcase, etc.

Keep your relationship diagram in front of you whenever you are designing or developing the database. I like to put them in plastic sheet covers with cardboard behind them so they stand up better. Before any forms or reports are built, which is like building a first and second story of a house, the foundation has to set. The foundation of every database is the table structure and relationships.

**Steps to take if you can't Enforce Referential Integrity**

1. Change the DefaultValue for numeric foreign key fields to Null instead of 0 (or just delete it)

Access sets the DefaultValue of numeric fields to 0 -- this is not good for a foreign key because there won't be a sequential auto number field to match it to -- and if there is, it probably won't be right. If you want to ensure that the user fills out the data, use the form BeforeUpdate event (used for data validation) so that you can issue an understandable message.

2. Delete the 0's out of the foreign key data in your tables

This should take care of the problem

Now that you have the relationships more readable ... in addition to making the diagram easier to read, the purpose for enforcing referential integrity is so that a main record that has children does not get deleted leaving an :( orphan ...

**Indexes**

Another consideration is how many Indexes exist – and what Access creates automatically if you don't stop it. Each time you name a field with ID or Code, Num, or Key in it, Access creates an
Index. This is like the index in the back of a book -- makes things easier and faster to find. But each time a record is changed, all the indexes have to be updated. Whenever you are designing your tables, keep the indexes window open and delete extra indexes that you don't have a specific purpose for. Click the lightening bolt icon or from the menu → View, Indexes

Figure 4-1 Index Window showing Primary Key

Change the default so that Access does not create so many indexes while you are creating your tables (there IS a limit to how many a database can have).

From the menu in the database window
---> Tools, Options, Tables/Queries --> AutoIndex on Import/Create

Figure 4-2 Tools, Options - Delete AutoIndex Entries

... Personally, I delete everything there, so Access doesn't create any indexes for me. If I want them, I will make them myself.

If you are using Access 2007, or want more information on this, See this gem tip:
Prevent Automatic Indexes -- By Tom Wickerath
http://www.access.qbuilt.com/html/gem_tips.html#AutoIndex

I generally delete any index (or just make sure it is never created!) that is not needed for the Primary Key or a Unique Index. A table can have 32 indexes. It best NOT to create indexes on foreign keys that will be used in a relationship that enforces referential integrity (RI) because they will be redundant. Each time you create a RI relationship, Access creates a hidden index – and these hidden indexes count toward your limit!

Unique Indexes

Sometimes, you want to make sure certain fields are not repeated, such as the user value for CustomerCode or a ChartNumber for a patient. In these cases, make a UNIQUE index on that field by setting the Index property.

Figure 4-3 shows a unique index set on StateName in the States table.

Indexes --> Yes (No duplicates)

Primary Key indexes are always unique.

Figure 4-3 Unique Index on StateName

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST</td>
<td>Text</td>
<td>StateAbbreviation or Country</td>
</tr>
<tr>
<td>StateName</td>
<td>Text</td>
<td>StateName</td>
</tr>
<tr>
<td>IsActive</td>
<td>Yes/No</td>
<td>IsActive</td>
</tr>
</tbody>
</table>

Steps to make a Multi-Field, Unique Index

For resolution of the problem allowing a duplicate combination into the table that must be unique:, make a unique index in the table on a multi-field combination

1. From the table design, turn on the Indexes window (from the menu: View, Indexes)

2. Click on the first field in the table design and, in the lower pane, set the Index property to Yes (No Duplicates)

that will add a line to the Indexes window and the Unique property will be Yes.

3. In the row just below the index you just made, in the 2nd column of the indexes window, Click in the fieldname column, drop the list, and choose the 2nd fieldname
4. If you have another field to add, **Click** on the fieldname column in the next row down and choose the 3rd fieldname.

A Unique Index will disallow duplicate combinations into the table.

Using a Unique index is very helpful when you use Append queries to create data such as:

1. creating test question records for each student so you can quickly fill in the data
2. adding oil & gas production for a well, product, and a day
3. creating survey detail records for each person

This ensures that, even if the query to add records runs twice, duplicate records will not go in.

Figure 4-4 shows a multi-field unique index on the combination of three fields: PID, TypeID, and Phone.

**Figure 4-4 Multi-Field Unique Index**

- **Index Name**: PIDTypePhone
- **Field Name**: PID, TypeID, Phone
- **Sort Order**: Ascending
5. Form and Report Design

**Toolbox**

In the design view, it is a good idea to show the Toolbox. The toolbox enables you to easily add several different types of controls to your form or report.

Click the icon that looks like a Hammer and Wrench to toggle the Toolbox on and off. It is common to "dock" the toolbox on the left side of the screen.

![Toolbox](image)

**Figure 5-1 Toolbox**

- Select Objects
- Control Wizards On|Off
- Label Tool
-Textbox Tool
- Option Group Tool
- Toggle Button Tool
- Option Button Tool
- Checkbox Tool
- Combobox Tool
- Listbox Tool
- Command Button Tool
- Image Tool
- Unbound Object Frame Tool
- Bound Object Frame Tool
- Page Break Tool
- Tab Control Tool
- Subform/Subreport Tool
- Line Tool
- Rectangle Tool
- More Controls

**Customize Form/Report Design toolbar**

**Add Alignment and Order icons**

When you are in the design view of a form or report, Right-Click in a gray area to the right of your menu bar or an icon bar.

1. choose "Customize" from the shortcut menu
2. in the dialog box:
choose the Command tab

**Category → Form/Report Design**

3. Click in the gray area between the scroll box and the down arrow in the shaded gray area of the vertical scroll bar in the Commands pane 3 times, then Click the down arrow at the bottom of the scroll bar until Align Left is the top-most choice

**Figure 5-2 Customize Form/Report Design Toolbar**

4. Drag these 6 icons to an existing toolbar:
   - **Align Right** -- must have >1 control selected to be enabled
   - **Align Left** -- must have >1 control selected to be enabled
   - **Align Top** -- must have >1 control selected to be enabled
   - **Align Bottom** -- must have >1 control selected to be enabled
   - **Bring To Front**
   - **Send To Back**

   If you don't know what these icons do, while the customize window is open, Click on the **Description** command button for each one to see what is does.

   To be efficient, the Align button are a *must* for form and report design. The **Bring To Front** and **Send To Back** are nice to have too, since everything is "layered" and this gives you control on order.

**Best-Fit Control Height and Width**

To "Best-Fit", **Double-Click** any handle on a control except the big one in the upper left. This resizes:
• Width and Height depending on font used for labels and command buttons

• Height for data controls (since Access doesn't know how much data will be in there, it leaves the Width alone)

If a control is closer to the height of 2 or more lines, the height will be made to an even increment of the number of lines

**Properties and Methods**

In design view, it is good to show the property sheet, which displays information for selected item(s).

**Turn on Properties window**

When you are in the design view, to turn on/off the Properties window -->

1. from menu: **View, Properties**  
   OR  
2. **Right-Click** and choose **Properties** from the shortcut menu  
   OR  
3. **Double-Click** on the solid black square in the upper left where the rulers intersect  

and then **Click** on various objects. The properties window changes as you change what is selected. If you have multiple objects selected, the values for the properties they have in common will be displayed. Try it!

**Setting Properties vs. Resizing with Handles**

I like to make the width of controls exact -- like 0.25, 0.3, 0.4, 0.5, 0.6, 0.75, 1, 1.25, etc

This is especially handy for lining up labels to controls under them

**Selecting Objects**

You can select multiple controls

1. **Click** and **Drag** the mouse and everything your imaginary rectangle touches before you let go will be selected  
   OR  
2. **Click** the first object then **Shift-Click** additional objects to toggle the selection status without affecting what is already selected  
   OR  
3. **Click** (and, optionally **Drag**) in a ruler
Figure 5-3 Drag in ruler to Select Multiple Objects

While the mouse is down, you will see a line extend across (vertical ruler) or down (horizontal ruler). When you Click and Drag, the ruler will turn dark indicating where you started and stopped -- everything the line/rectangle touches will be selected

OR

4. Drop down the Object combo (left-most control on design toolbar) and select something by its control name

Figure 5-5 Select Object Combobox on Toolbar

© Copyright 2008 Crystal. All rights Reserved.
Select Form or Report

To select the form (or report), you may:

1. Click in the upper left corner where the rulers intersect

2. Click completely outside the design area in the dark gray space

3. Press \texttt{Ctrl} + \texttt{R}

4. from menu: \texttt{Edit}, \texttt{Select Form/Report}

Building Event Code

After selecting appropriate object, you can Click on the desired event property and then Click the builder button \(\ldots\) in the right margin. By default, you will see a dialog box with three choices. You'll only go directly to the VBE if you have previously selected the option to "Always Use Event Procedures" (Tools > Options | Form/Report tab). Another way to go directly to the VBE window is to first type \([\ldots]\) in the property (Access will fill in \([\text{Event Procedure}]\)) and then click the builder button or press \texttt{Ctrl} + \texttt{F2}

Access will provide the procedure declaration and the procedure end -- you put the statements in between.
Name property

Procedures are NAMED according to the object name (except the form object), so always change the Name property of any object to something logical before creating any event procedures. If you change the name after creating an event procedure, then you will have to change the name of the procedure too. For example, if you have a command button named cmdClose, its (simplified) event procedure may be something like this:

```vba
Private Sub cmdClose_Click()
    DoCmd.Close
End Sub
```

When the properties window is displayed and only one thing is selected, the Name appears in the title bar of the Properties window.

**Figure 5-8 Name of Selected Object Appears in titlebar of Properties Window**
If multiple items are selected, you will see "Multiple Selection" on the title bar of the Properties window. The Properties that are the same for all the selected objects will be listed.

**Figure 5-9 Multiple Items Selected: Properties in common are Listed**

![Image of multiple items selected with properties listed](image)

**ControlSource, SourceObject**

It is important to realize that the **Name** is NOT what is displayed *in* a control. If the control is (for instance a textbox or combobox), you will see the **ControlSource** displayed.

**Figure 5-10 Control with an Ambiguous Name**

![Image of control with ambiguous name](image)
Notice how 'Fullname' shows up in the control … but its NAME (what will be used to refer to it in code and expressions) is Text67, which is ambiguous as it does not describe the contents of the control. This is how Access will assign names when you create controls using the toolbox. It is important to change the Name property to something that makes sense.

For bound objects, I like to make the Name property match the source property (this does not, by the way, follow naming conventions, but for me, it eases confusion).

Figure 5-11 Change Name to match ControlSource for Bound Textbox Control

As always, avoid using spaces and special characters when naming objects. Use mixed case for readability and, if you want separation, use the underscore _ character.

If the object is a subform or subreport, what you see displayed is the name of the SourceObject.

Figure 5-12 Name of a Subform is not necessarily the same as the SourceObject
**RecordSource Property**

From the design view of a report or form:

1. Turn on **Properties** from menu: View, Properties  
   OR  
   Right-Click and choose Properties from the shortcut menu

   ![Properties Menu](image)

2. Select the report or form  
   **Click** in the upper left where the rulers intersect

   ![Rulers](image)

3. **Click** on the Data tab in Properties window

4. **Click** in the RecordSource property
Click on the drop-down arrow to see a list of Tables and Queries to choose from.

Notice the builder button to the right.

If your RecordSource is a query, clicking the builder button takes you to the query design screen. If the RecordSource is a table, you will be asked if you want to make a query.

Figure 5-15 shows a RecordSource based on a SQL Statement.
Figure 5-15 RecordSource can be SQL Statement

If you use a SQL statement as the RecordSource for a form, the combo box wizard will not work. Clicking the Builder Button takes you to the same kind of screen you would see if the RecordSource was a query. Figure 5-16 shows that Aliases have been used for the People (p) and the Types (T) tables. The asterisk means all fields will be used from the p table because this form is primarily for editing information in the People table. In this case, the only reason that Typ is pulled in as well is for sorting. It is not, however, visible on the form except in the TypeID combobox which shows Typ and collected TypeID (refer to Combobox Example).
If your report is based on a SQL statement, you can also press \( \text{Shift} + F2 \) to use the Zoom Box to change the SQL directly without the QBE grid.

**Builder Button**

*Using the Builder for RecordSource or RowSource*

The `RowSource` property for a combo or listbox is like the `RecordSource` property for a form or report -- you are choosing where the data for the list of choices will come from. The Builder Button ... will show when you **Click** in the `RowSource` property. Clicking on the builder button for a query takes you to the design of the query. If you **Click** the builder button and you have not filled anything out, you can use the QBE grid to design an SQL statement that Access will save in that property.

*Using the Builder to choose other Colors*

For choosing colors that are not on the color palette (like `ForeColor`, `BackColor`, `BorderColor`), **Click** the button in the property.

Once in the palette dialog box, expand the dialog box by clicking...
You can set the amounts for Red/Green/Blue or adjust Hue/Saturation/Luminosity. There is also a slider control with a triangle you can Drag up or down to change the Luminosity (brightness). I like to Drag it up and fade out colors, especially for BackColor.

Figure 5-17 Custom Color Palette - drag slider to lighten/darken color

While we are on the subject of colors … another cool trick is setting the BackColor of a control to something like pale yellow – then set the BackStyle to Transparent so the background color of the form shows through (I like to use white or a faded color). When you are IN the control, it will be the BackColor – another way you can easily indicate to the user where they are <smile>.

Using the Builder to go to Code

If an event has [Event Procedure] assigned, you can Click on the Builder button to go to the code.

On Click . . . . . . . . . . [Event Procedure] ....

If an event is assigned to a function in a standard module, you can Click on the Builder button to go to the code.

On Click . . . . . . . . . . =DeepAnalysis() ....

If the assigned function is a function behind the form and not in a standard module, the Builder Button .... will not take you there … you will need to go to the module sheet specifically (Click the 'Code' button on the toolbar or, from the menu, choose View, code…)

Then, either scroll <smile> or use the Procedure/Member combobox in the upper right-corner of the module window –where you currently see (Declarations).
If you are looking for an [Event Procedure], I find the quickest way there is the Builder Button from the object itself... but you can also choose the object name – where (General) is displayed in the above figure.

... then the event (member). Object events that have a procedure already defined are bolded.

Events

"Properties" listed on the Events tab specify how the event should be handled, such as OnCurrent for form, AfterUpdate for Control, etc.

Deleting a Control

Before you delete a control that has [Event Procedure] code, first select the control and then the Events tab on the Properties dialog. Follow all [Event Procedures] and delete the code before
deleting the control; otherwise, your code will become filled with "spiderwebs" (a "Tom-term") – all kinds of procedures for non-existent controls.

**Learning Properties and Methods**

Explore the property sheet. Get familiar with how properties are grouped on the tabs and the different properties for different objects.

**Help on Properties and Methods**

You can get help about any property by pressing $\texttt{F1}$ while in a property on the property sheet where you want more information.

**Object Browser**

... Find out more about references... or just get Help!

In a code window to View the Object Browser:

1. from the menu --> View, Object Browser

OR

2. Press $\texttt{F2}$

On the left, you will probably see a Project window and a Properties window below it

On the right, you will see the main Object Browser window

As you select a Class on the left (DateTime selected), its members appear in the pane on the right

When you are getting started, change the library to "VBA" (for instance, instead of <all libraries>) and look at the classes (categories) of functions -- Click on a class and then Click on a function in the right pane. When you see something you want help on, press the $\texttt{F1}$ key and switch to the Help window

The VBA library is the most basic library and a great place to start exploring.
To look up properties and methods for different objects like Forms, TableDefs, etc, change the library to Access.

To look up Ranges and Sheets, change the library to Excel.

Explore the different libraries you have to pick from and see what is available in each (these are added or removed using Tools, References...)

When you are in the Object Browser window, the library that each function/class is a member of is shown in the lower left corner of the window.
When you have an object selected, press **F1** to get help.

**General Help**

For general help, I find it interesting and informative to read the Help starting from the beginning of the Contents. In fact, if you have the desire to print a ream of paper, it would be good to print it like a book and read it so you can also take notes -- and you can read it away from the computer -- a good time to put new information into your head is just before you sleep ... let your subconscious figure it out!

**Code Behind Form (CBF) … or Report**

To see the code behind a form (anything said about code behind a form also applies to code behind a report): from the menu in the design view of a form, choose --> **View, Code**

This will take you to the module sheet that is stored 'behind' and with the form. When the form is copied, the module sheet tags along without you having to do anything special. This kind of code is a **Class Module** as opposed to a **Standard (General) Module**. When you **Click** on the Modules tab in the Database Window, you see a list of Standard Modules -- they are able to be used by any form, any report, any query ... anything in the database. Therefore, their scope is global.

**A Procedure** is a general term encompassing Subs and Functions. If you look at a **procedure declaration** you might see something like this:

---

Private Sub eAddress_DblClick(Cancel As Integer)

**Private** means that only this form (and module stored behind the form) can see and use this code -- its **scope** is limited to the form.

**Sub** means it is a Subroutine that does not return a value as opposed to a **Function** which does ... or maybe I should say 'can' -- return a value. A Function can do everything a Sub can -- a weird thing -- you cannot assign a Sub directly to a property event, it **must** be a Function by definition -- and there is no way to **use** a return value in these cases. Why Microsoft chose to use Functions instead of Subs in Access is a mystery. With this in mind, it seems odd that the **[Event Procedure]** code is a Sub -- not consistent! ... anyway, I digress...

**eAddress_DblClick** means that there is a control with the **Name** 'eAddress' and this is the **DoubleClick** event procedure -- so it will be launched (triggered) when the user double-clicks on the control.

**Cancel As Integer** -- **Cancel** is a parameter to the sub -- although most people don't think of "Cancelling" a **DoubleClick** event because you can simply exit it! **Cancel** is better understood on an event such as **BeforeUpdate** where you **Cancel** the **Update** to a control or prevent Access from saving a record if certain conditions are not met. **As Integer** means that **Cancel** is a whole number < 32K. Some event procedures have **Cancel** as a parameter but most do not. For instance, out of the 50 events that can be defined at the form level, 13 of them have **Cancel** as a parameter.
An easier way to create code for the **Double-Click** event of a control is this:

Turn on the **Property** sheet in the design view of a form  
(Right-Click anywhere and choose **Properties** from the shortcut menu)

**Click** on the control you want to see or set properties for -- you will see the control **NAME** (another property) in the title bar of the **Properties** window and all kinds of information that describe that control.

**Figure 5-22 Properties for a Control**

![Properties for a Control](image)

**Click** on the **Event** tab of the Properties window

**Click** in the **DoubleClick** event

Off to the right, you will see a **Builder** button -- **Click** on this and, if you have not yet created code, 'Code Builder' will be one of the choices. Access will create the declaration statement, then a blank line where you can start typing code, then an 'End sub' statement
When you create code like this, you will see [Event Procedure] in the property sheet for that event

Alternately, you could create code as a Private (or not) Function ... something like this:

```vba
Private Function SendEmailFromCrystal()  
    If IsNull(Me.eAddress) Then  
        MsgBox "eMail Address is not filled out" _  
        , , "Cannot send eMail"  
        Exit Sub  
    End If  
    Application.FollowHyperlink _  
        "mailto:" & Me.eAddress & "?subject=From%20Crystal"  
End Sub
```

then, directly on the property sheet, you could write this in the property, instead of the usual [Event Procedure] -->

= SendEmailFromCrystal ()
Rules For Code

Procedures can be in any order -- it is more efficient to put the procedures that are going to be used more often near the top of the code just so Access uses less time to find them when necessary. The main rules are this:

1. **Option** statements (called *Compiler Directives*) must be at the top

```
'capital letters will be equal to lowercase letters for comparison  ABC = abc
Option Compare Database
'venables must be defined (i.e.: Dim) before you can use them
Option Explicit
```

Using **Option Explicit** is very important. For more information, visit this link:

**Always Use Option Explicit, by Tom Wickerath**

2. variables that are global to the module (all procedures in the module can use it) are declared before any procedure code. Otherwise, if a variable will just be used by a particular procedure, it is declared in the procedure itself.

3. code has to be between a procedure declaration statement and the End Sub/Function statement -- you can't have loose code hanging around that is not inside a procedure unless it is a comment.

4. comments can be anywhere.

**Organizing Code and Other Information**

Here is a tip to organize code or SQL or instructions or links (etc) that you write or get from other sources:

Each time you write something that you think would be useful at a later date for yourself or to help someone, use NotePad to save it as a text file.

To put NotePad on your Start button:

1. click Start
2. choose --> All Programs, Accessories
3. Highlight NotePad
4. hold down the CTRL key while you drag it to your Start button and place it in the list (make sure you are still holding CTRL when you let go of the mouse to make a copy)

For example, here might be a few of your filenames:

- code_NotInList.txt
- code_FindRecord.txt
- tip_DebugCompile.txt
- tip_WindowsDefaults.txt
- sql_Syntax.txt
- help_FormatCodes_DateTime.txt
- step_DocumentingADatabase.txt
- UA_CodeTags.txt

Both FindRecord and NotInList need to be changed for the circumstances they are being used, but at least you have "shell" code to build around. In my FindRecord file, I have code that goes behind the form at the top since that is easiest to understand and what I most often give to other people. I also have a generic function that takes parameters and goes into a standard module that is below the first procedure. I also put the instructions to give the user in the text file.
Here are some prefix examples:

- code -- contains code
- blog -- link to a blog
- db -- has a database attachment you can download
- dfn -- definition
- eqn - equation
- help -- from the Application help
- idea -- something you want to write more about later
- link -- link
- sql -- SQL syntax
- step -- numbered steps to do something
- tip -- has instructions
- UA -- information specific to UtterAccess (my favorite on-line forum)

If you are not sure how you might search, use the filename to specify multiple search terms.

- code_LastDateOfMonth_DaysInMonth_EOM.txt
- code_CreateDates_FillComboBoxWithDayOfWeek_DOW.txt
  *(This might be instructions on creating a Dates table and code to create date records for 20 or so years, and then instructions to fill a combo box with all dates for a particular day of the week, like all the Tuesdays)*

If you open one of the text files to give the information to someone else, after you submit the post and it renders on the screen, copy the post link, subject, and date to the top of your text file. If the post is on UtterAccess, copy the URL (Uniform Resource Locator) from the AddressBar of your browser window and truncate all the information after the number. For instance:

--------------------------
filename --> link_MainformSubform.txt
--------------------------

12-6-07

Some rather basic use of relationships -- Mainform/Subform


--------------------------
When someone posts code or SQL or a link that you know will be a good reference, make a note of
(1) the link (2) a title (3) who wrote it and, once again, save text files. For example:

code_CloseOpenForms_ArvinMeyer.txt

code_PrinterEscapeSequence_datAdrenaline.txt

code_hideNavigationPane_2007_JohnViescas.txt

link_Access2007_JeffConrad.txt

link_Performance_mishej.txt

When you want to find "your" help, use the Windows search facility. For instance, you might be
looking for something on how to send printer escape sequences and you remember that
datAdrenaline (Brent Spaulding, Access MVP) wrote it.

Right-click on the top level Application directory for your code in Windows Explorer (or
My Computer) and choose --> Search

All or part of the filename --> *print*_dat*
to find --> code_PrinterEscapeSequence_datAdrenaline.txt

With the title and a link at the top of the document, you can just copy and paste for others. For
example:

---------------

sending hex code to printer ... windows default printer, by datAdrenaline (Brent)

---------------

If you know you have a file that covers the information you are looking for but you cannot
remember your search terms, search 'A word or phrase in the file' -- and then when you find it,
rename the file to include what you originally used to try to find it by searching on the name.

If a site has some good code that you know you will reference (two examples are www.mvps.org
and www.AllenBrowne.com ), I save things like this:

code_API_OpenFile_CommonDialog_KenGetz_mvps.txt

The link is there too, but if you also copy the code for yourself, it is convenient to name it starting
with 'code_'

~~~~~~~~~~~~~~

I also have a document with basic links that I give out often (like the Access Basics document posted
in the UtterAccess Tutorials forum and Allen Browne's reserved word list) -- but I keep this short or
it would be faster to use the Windows search <smile>

~~~~~~~~~~~~~~

You can also keep all your Access code in a database and reference it in all your applications ... but I
like using TXT files because not everything is code.
You might have code/info for other applications as well, like Excel, Word, and PowerPoint. Create folders for each application and put the text files under the application folder. Within an application folder, you may create several more folders under them (such as code written for specific versions, or stuff written by other MVPs) -- and when you search, start at the top level directory for that application.

The TXT file thing works real well for me (and I update them) so, like an engineer, I think, "why fix it if it isn't broken?"

Hope this information helps you get organized <smile>

**Creating a Main Form and Subform**

In my opinion, each form/subform should be based on just one table*. If you wish to manipulate information in more than one table, it is best to use a main form to for the "parent" table and subforms to display and edit information in each of the related tables.

*Note: This is a general guideline that helps when you are beginning to use Access. As you get better and realize that certain records must be created before related records, you can bend this rule.

Create the main form and the form(s) that will be used as subform(s) -- make sure to put the linking key fields on them (usually ID fields).

**to put a subform on a main form:**

Create a subform control on your main form using the subform/subreport tool in the toolbox (Cancel the wizard if it pops up and fill properties manually)

Then, from the design view of the main form:

1. **turn on the properties window** – Right-Click anywhere and choose Properties from the shortcut menu
2. **Click** ONE time on the subform control if it is not already selected
3. **Click** on the DATA tab of the Properties window

**SourceObject** --> drop list and choose the name of the form you will use as a subform (You can also **Drag** a form object from the database window and drop it on the main form. This automatically sets the **SourceObject** property.)

**LinkMasterFields** --> **MainID**

**LinkChildFields** --> **MainID**

If you have multiple fields, delimit the list with semi-colon

**LinkMasterFields** --> **MainID;Fieldname_main**

**LinkChildFields** --> **MainID;Fieldname_child**

**WHERE:**

-- **MainID** is replaced with *your* field name holding an AutoNumber field (usually) in the parent table and a Long Integer field in the child table. **Fieldname_main** is the fieldname in the main **RecordSet** – and it is best to actually put the field on the main form. **Fieldname_child** is the name of a field in the child **RecordSet** – and, once again, it is best that this field actually be ON the related subform.

Even though the Help for Access says that the linking fields do not have to be ON the forms, I find this not to be the case. It is best that you reference fields that are ON each of the respective
forms. If a control is bound, I usually make the Name of the control the same as the ControlSource (what is in it). This does not follow standards but I find that it eases confusion. There are those who disagree and insist that controls should be named according to convention such as:

Naming Conventions for Microsoft Access - Leszynski/Reddick Guidelines for Microsoft Access

Reddick VBA (RVBA) Naming Conventions
http://www.xoc.net/standards/default.asp

It is common to set the Visible property to No for the control(s) containing the field(s) used in LinkChildFields

4. while still on the subform control, Click the ALL tab in the Properties window -- change the Name property to match the SourceObject property (minus Form. in the beginning if Access puts it there).

Difference between Subform Control and Subform

The first Click on a subform control puts handles* around the subform control.

*black squares in the corners and the middle of each side -- resizing handles

Figure 5-25 Subform Control Selected
The subform control has properties such as

Name  
SourceObject  
LinkMasterFields  
LinkChildFields  
Visible  
Locked  
Left  
Top  
Width  
Height

The subform control is the container for the subform or subreport used as the SourceObject.

Figure 5-26 shows that the second Click gets you INTO the subform that is contained by the subform control …then when you first Click on the contained form, you will see a black square where the rulers intersect in the upper left of the "form" you are "in"

Figure 5-26 Form Contained by Subform Control is Selected

me.subform_controlname --> the subform control  
me.subform_controlname.form --> the form inside the subform control  
me.subform_controlname.form.controlname --> a control on the form contained by the subform control

The form that is contained by the subform control is an independent form -- you can open it directly from the database window, just as you can with any other form. It is often referred to as a "subform" because of the way it is being used.

It is advisable to edit forms used as subforms directly, instead of within the main form. Since I often do not disable Name AutoCorrect, oh-uh, I mean Name AutoCorrect, I have had trouble with Access putting property changes in the wrong place for RowSource and RecordSource. Since it occasionally happens there, for major changes, I go to the design view of the "sub" form directly from the database window when the main form is closed.

Failures caused by Name AutoCorrect  
http://allenbrowne.com/bug-03.html

© Copyright 2008 Crystal. All rights Reserved.
6. SQL Syntax

Don't let the SQL acronym intimidate you; it is not tough to learn the basics.

**SQL background**

SQL is Structured Query Language

It is a statement to get data out of one or more tables/queries. This is what Access stores for:

1. Queries (just shows the QBE grid for convenience -- choose View, SQL)
2. RowSource for Comboboxes and Listboxes (if list does not come directly from a table since a query is actually an SQL statement)
3. RecordSource for Reports and Forms (if list does not come directly from a table)

Queries can be different types, for instance:

1. Select
2. Crosstab
3. Make-Table
4. Update
5. Append
6. Delete

Select and Crosstab queries show data while the others are "action" queries" (Data Manipulation Queries) and do not display data -- so you cannot use them for record sources or for row sources.

To expand and reiterate:

Queries are a very powerful tool in Access -- you can do far more than simply use them to select data

... Queries can be ACTION Queries...*they DO things*, not show you things

They can add data to a table --> APPEND Query
They can make changes to a table --> UPDATE Query
They can delete records from a table --> Delete Query
They can make tables --> Make-Table Query

They can transpose data --> Crosstab Query (which is also, actually, SELECTing data from a table since there is no action)

Internally, Access stores an SQL (Structured Query Language) statement for queries*, not the visual representation you see when you design queries -- the QBE (Query-By-Example) grid is for your benefit, so you can use drag-n-drop and pick methods and visually look at things better.

*this is why it is important to get comfortable with looking at SQL statements

A great way to become more familiar with SQL is to LOOK at the SQL every time you make a query. Until you actually start to look, you never realize how easy and sensible it actually is... and it really helps to use Aliases (short abbreviations) for tablenames as it makes the SQL statement shorter.
**Types of Queries**

~~~ SELECT ~~~

*Select data from one or more tables and display the results in rows and columns.*

**BASIC SQL SYNTAX**

```
SELECT fieldlist
FROM tablename
IN anotherdatabase.mdb
WHERE conditions
GROUP BY fieldlist
HAVING conditions for fields that are grouped
ORDER BY fieldlist;
```

~~~ APPEND ~~~

*Add records to a table.*

An **Append Query** is a **Select** query preceded by

```
INSERT INTO Tablename (field1, field2, etc )
'~ for instance (and this does not use a table as it supplies actual values)
```

```
Dim strSQL as string
strSQL = "INSERT INTO [Tablename] " _
   & " ([TextField], [NumField], [DateField] ) " _
   & " SELECT ", & strValue & ", " _
   & numValue & ", " _
   & "," & datValue & "," _
   & ";"
CurrentDb.Execute strSQL, dbFailOnError
CurrentDb.TableDefs.Refresh
```

~~~ UPDATE ~~~

*Update records in a table.*

An Update Query first identifies the tables that are used

```
UPDATE table1
INNER JOIN table2 ON table1.keyfield = table2.keyfield
```

Then identifies what to change

```
SET table1.fieldtochange = expression
```

Then, if you have criteria...

```
WHERE tablename.strField = 'somevalue'
AND tablename.numField = 99
AND tablename.dateField = #1/1/06#
```
~~~ MAKE TABLE ~~~

Make a table from the resulting values.

```sql
SELECT fieldlist
INTO tablename
IN c:\path\anotherdatabase.mdb
FROM tablename
WHERE conditions to process before recordset is made
GROUP BY fieldlist
HAVING conditions for fields that are grouped
ORDER BY fieldlist;
```

~~~ DELETE ~~~

Delete records in a table.

```sql
DELETE A.*
FROM tblArticles AS A
WHERE (A.somefield=2);
```

~~~ CROSSTAB ~~~

Instead of displaying values for a field in different rows, display them across the top – like display months of a year from a date field.

```sql
TRANSFORM Count(B.Fieldname1) AS FieldAlias
SELECT
    A.Fieldname2,
    A.Fieldname3
FROM Table2 AS B
    INNER JOIN Table1 AS A
    ON B.someID = A.someID
GROUP BY
    A.Fieldname2,
    A.Fieldname3
PIVOT B.Fieldname1;
```

you can use equations to pivot (this will be column headings). For instance, if you want the column headings to be year and month, you can do this:

```sql
PIVOT Format([DateField].’yy-mm’);
```

if you want month names in chronological, instead of alphabetical, order, you can do this:

```sql
PIVOT Format([DateField].”mmm”) In
(”Jan”, ”Feb”, ”Mar”, ”Apr”, ”May”, ”Jun”, ”Jul”, ”Aug”, ”Sep”, ”Oct”, ”Nov”, ”Dec”)
```

The PIVOT clause can also be stated as PIVOT BY
If you also want a column that totals whatever you have in all the \textit{VALUE} columns (the expression after \texttt{TRANSFORM}), repeat your transform expression after the \texttt{SELECT} keyword and the \texttt{GROUP BY} keywords. For instance:

\begin{verbatim}
TRANSFORM Count(B.Fieldname1) AS FieldAlias
SELECT
  A.Fieldname2,
  A.Fieldname,
  Count(B.Fieldname1) AS FieldAlias2
FROM Table2 AS B
  INNER JOIN Table1 AS A
ON B.someID = A.someID
GROUP BY
  A.Fieldname2,
  A.Fieldname3,
  Count(B.Fieldname1)
PIVOT B.Fieldname1;
\end{verbatim}

here is an example:

\begin{verbatim}
TRANSFORM Sum(Qty) AS SumQty
SELECT
  Item,
  Sum(Qty) AS TotalQty
FROM Sales
  INNER JOIN Items
ON Sales.ItemID = Items.ItemID
GROUP BY
  Item,
  Sum(Qty)
PIVOT By Format(DateSale, "yy-mm_mmm");
\end{verbatim}

\begin{verbatim}
RESULTS:
\end{verbatim}

\begin{verbatim}
<table>
<thead>
<tr>
<th>Item</th>
<th>TotalQty</th>
<th>06-01_Jan</th>
<th>06-02_Feb</th>
<th>06-03_Mar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bells</td>
<td>15</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Whistles</td>
<td>9</td>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
</tbody>
</table>
\end{verbatim}
Data Definition Queries

There is another class of SQL statement, called Data Definition Queries, that can be used to create, modify or delete tables, fields, indexes and relationships. We will not go into examples here.

Aliases

Using Aliases for table names (query names) makes the SQL easier to read.

For calculated fields, it is best to assign your own field alias instead of letting Access use "expr1", "expr2", etc. Calculated fields MUST have aliases.

An Alias follows the keyword AS

Table Alias:

FROM t_People As P INNER JOIN t_Phones As Pho ON P.PID = Pho.PID

Field Alias:

SELECT Sale-Cost As Profit, etc…

Joins

When you are getting information from more than one table, the default join type is INNER JOIN. This means that only records in common will be displayed. For instance, if you have a table of Companies and a table of Jobs and not every Company has done a job, but you want to see ALL the companies anyway, you would use a Left Join or a Right Join for the type of relationship between tables.

From Companies AS C

LEFT JOIN Jobs AS J ON C.CompID = J.CompID

The join is specified as a LEFT JOIN because you want all records from Companies and any matching records from Jobs. [The On clause can have the tables in either order.] In this case, since C (Companies) is on the left side of the equal sign, the Join is a Left Join.

ON C.CompID = J.CompID --> C is on the LEFT side of the equal sign.

Parameters

If you specify data type in Query, Parameters, the SQL statement is preceded by (for example):

PARAMETERS [enter category] Text ( 50 );

while the criteria may be:

WHERE (B.Category=[enter category])

More information on SQL

For more information on SQL, one of the masters is John Viescas and he has written numerous books, all of which are fantastic. One of his books is titled, "SQL Queries for Mere Mortals"

Look At SQL!

Whenever you create a query, LOOK at the SQL statement and study it for a couple minutes -- it makes sense! Within a short period, the confusion will be gone...
from the menu of a query, choose:

**View, SQL**

The SQL statement can be selected, copied, then pasted into Word for formatting and printing (makes great wallpaper for your wall, not your computer ;) as you are learning) or into Notepad to have as a popup reference while you are working into design view, etc.

First, get comfortable with **SELECT** statements. Once you have them mastered, learn other forms.

Here is an example of the Select statements. Once you have them mastered, learn other forms.

**Figure 6-1 Datasheet View of a Phone List Query**

<table>
<thead>
<tr>
<th>Fullname</th>
<th>Type</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &amp; M Surplus</td>
<td>Store</td>
<td>(123) 456-7890</td>
</tr>
<tr>
<td>Ackerman, Ian T.</td>
<td>Friend</td>
<td>(123) 456-7890</td>
</tr>
<tr>
<td>Ackley, Cynthia T.</td>
<td>Friend</td>
<td>(678) 969-0645</td>
</tr>
<tr>
<td>Ackley, Cynthia T.</td>
<td>Friend</td>
<td>(678) 732-6511</td>
</tr>
<tr>
<td>Ackley, Steven</td>
<td>Friend</td>
<td>(42) 514-9794</td>
</tr>
<tr>
<td>Ali Natural</td>
<td>Health Store</td>
<td>(555) 545-4545</td>
</tr>
<tr>
<td>American School of Beauty</td>
<td>Beauty</td>
<td>(333) 489-7845</td>
</tr>
<tr>
<td>Anderson, Heather E.</td>
<td>Friend</td>
<td>(770) 637-4055</td>
</tr>
<tr>
<td>Anderson, Heather E.</td>
<td>Friend</td>
<td>(678) 937-3675</td>
</tr>
<tr>
<td>Anderson, Walt</td>
<td>Friend</td>
<td>(770) 128-7468</td>
</tr>
<tr>
<td>Anderson, Walt</td>
<td>Friend</td>
<td>(678) 994-3611</td>
</tr>
<tr>
<td>Andy's Alarm systems</td>
<td>Home Suppl/Serv</td>
<td>(456) 871-6512</td>
</tr>
<tr>
<td>Angelo's Floor Covering</td>
<td>Home Suppl/Serv</td>
<td>(456) 074-2131</td>
</tr>
<tr>
<td>Applegate Ace Hardware</td>
<td>Bldg Supplies</td>
<td>(123) 456-7892</td>
</tr>
<tr>
<td>Astrid's Upholstery</td>
<td>Services</td>
<td>(325) 635-4677</td>
</tr>
<tr>
<td>Backseat Painting</td>
<td>Services</td>
<td>(456) 056-7232</td>
</tr>
<tr>
<td>Baker, Debra T.</td>
<td>Friend</td>
<td>(678) 103-0103</td>
</tr>
<tr>
<td>Baker, Debra T.</td>
<td>Friend</td>
<td>(770) 851-9069</td>
</tr>
<tr>
<td>Baker, Richard</td>
<td>Friend</td>
<td>(404) 710-4565</td>
</tr>
<tr>
<td>Baker, Richard</td>
<td>Friend</td>
<td>(404) 594-3515</td>
</tr>
<tr>
<td>Barker Tires</td>
<td>Auto</td>
<td>(130) 564-7974</td>
</tr>
<tr>
<td>Barnett Auto Body</td>
<td>Auto</td>
<td>(415) 610-6543</td>
</tr>
<tr>
<td>Barricks, Cecilla F.</td>
<td>Friend</td>
<td>(90) 667-7332</td>
</tr>
<tr>
<td>Barricks, Cecilla F.</td>
<td>Friend</td>
<td>(803) 106-8148</td>
</tr>
<tr>
<td>Barricks, Jonathan</td>
<td>Friend</td>
<td>(912) 646-3705</td>
</tr>
</tbody>
</table>

… and the Design View of the same query:
Figure 6-2 Design View of a Phone List Query

... and the SQL View:

Figure 6-3 SQL View of a Phone List Query

```
SELECT p.[MainName] & (' '+p.[suffix]) & (', '+p.[FirstName]) & (' '+p.[MiddleName]) AS Fullname, Types.Typ, Phones.Phone
FROM (t_PEOPLE AS p
       LEFT JOIN t_Types AS Types ON p.TypeID = Types.TypeID)
       INNER JOIN t_Phones AS Phones ON p.PID = Phones.PID
ORDER BY p.MainName, p.FirstName;
```

Boldfacing the keywords and adding a few more line breaks gives this:

```
SELECT p.[MainName] & (' '+p.[suffix]) & (', '+p.[FirstName]) & (' '+p.[MiddleName]) AS Fullname, Types.Typ, Phones.Phone
FROM (t_PEOPLE AS p
       LEFT JOIN t_Types AS Types ON p.TypeID = Types.TypeID)
       INNER JOIN t_Phones AS Phones ON p.PID = Phones.PID
ORDER BY p.MainName, p.FirstName;
```
An Alias of 'p' has been used for the People table, which makes the SQL easier to read. The Alias is in this phrase:

```
t_people AS p
```

Can you spot another Alias?

There are actually 4 Aliases altogether, 3 for tables, and one for a calculated field.

**Using VBA to execute SQL statements**

You can execute action queries using VBA

```
dim strSQL as string
strSQL = "UPDATE tablename " 
& "SET fieldname = value " 
& " WHERE conditions;"
Debug.Print strSQL
CurrentDb.Execute strSQL
```

**Debug.Print**

Debug.Print strSQL

--> this prints a copy of the SQL statement to the Debug (Immediate) window

After you execute your code, open the Debug window

Press **Ctrl** to Goto the debug window -- look at the SQL statement

If the SQL statement has an error

1. Make a new query (design view)
2. choose View, SQL from the menu
(or SQL from the toolbar, first icon)
3. cut the SQL statement from the debug window -- select the statement then **Ctrl**
4. paste into the SQL window of the Query **Ctrl**
5. run ! from the SQL window

-- Access will tell you where the problem is in the SQL

**Subqueries**

```
SELECT field1,..., (subquery) AS alias FROM datasource
SELECT fieldlist FROM datasource WHERE field comparison operator (subquery)
SELECT fieldlist FROM datasource WHERE field ANY|SOME|ALL (subquery)
WHERE expression [NOT] EXISTS (subquery)
```
7. Delimiters

A delimiter is a character used to mark the boundaries of a particular value or to separate different parts of data. For instance, in this example:

8:30 am

A colon (:) is used to separate hours from minutes.

When we reference a file on a drive, we use the backslash \ to separate directory names from each other and from the file name:

C:\Data\Documents\MyFile.Doc

Access has several different delimiters and each means something different. Some delimiters, such as quotes and hash marks are balanced like parentheses around a value and tell Access what data type to assign to the value. Other delimiters, like bang and dot, tell Access that what comes next is a member or a property.

"String in Double Quotes"

Double quotes are used to delimit strings

"this is a string of text"

"Mary's lamb has a hyphen which is okay because double-quotes are used to delimit this string"

If you are delimiting a string with double quotes and want to use a double quote IN the string, use 2 of them.

"The width is 4"""

Will be displayed as ➔ The width is 4"

When you are concatenating field and literal values, use quotes around the literal values. for instance, there needs to be a comma and a space between City and State

=[City] & "", " & [State]

the brackets are not necessarily necessary -- they are often used when typing what to do so that you know it is a fieldname that is being referenced.

'String in Single Quotes'

Single quotes are also used to delimit strings in queries and control source expressions. If a value in the string contains a double-quote, like writing four inches as 4", you can delimit the string with single quotes.

'The width is 4''

There are many cases when single quotes don't work on the outside, like in an argument for dLookup (unless you are constructing it in code, then it does). In VBA code, you must use double-quotes as the outer delimiter when referencing strings.
#Date#

Hash marks are used to indicate that the value is a date/time data type. This is perhaps the data type that causes the most trouble. Internally, it is stored as a number yet it is shown to you in various formats. It also stores multiple pieces of information: year, month, day, hour, minute, and second.

Access stores date/times in a numeric format where the integer portion of the number represents the date and the decimal portion of the number represents time:

\[
\begin{align*}
1/1/100 &= -657,434 \\
1/2/100 &= -657,433 \\
12/30/1899 &= 0 \\
1/1/1950 &= 18,264 \\
1/1/2005 &= 38,353 \\
\text{12/31/2007} &= 39,447 \\
1/1/9999 &= 2,958,101 \\
12/31/9999 &= 2,958,465
\end{align*}
\]

Time is a fraction of the day

\[
\begin{align*}
12 \text{ noon} &= 0.5 \\
6\text{pm} &= 0.75 \\
\text{12/31/07, 12 noon} &= 39,447.5
\end{align*}
\]

If you have a control with just a date and you want to make sure it converts to a whole number, use:

**DateValue([control_or_fieldname])**

or

**cLng([control_or_fieldname])**

Likewise, if you only want the time component, you can use:

**TimeValue([control_or_fieldname])**

Since dates are whole numbers and times are the fractions, you can also do arithmetic operations on them. That is why you can subtract one date from another and get the number of days between the two.

If you have a date stored in text format, use cDate (convert to Date) around it:

**cDate([textDate_controlname_or_fieldname])**

… or concatenate the date with delimiters:

"#" & [textDate_controlname_or_fieldname] & "#"

Because dates can also have a time component, it is handy to use DateAdd, which let you specify the time interval (year, month, day, hour, etc) to add or subtract to calculate a new date or use DateDiff to get a numeric difference between dates for a specific interval.

The DateDiff function can be used to specify what time increment you want returned when you subtract dates. Likewise, there is a DateAdd function to add specific time increments to a date.

Dates are stored as floating point numbers. This makes them inaccurate for equality comparisons -- the best way to ensure you have only the Whole part of the number (the date), is to use the Integer portion of the number (the date) only -- this, in essence, is what DateValue does. In addition to showing the result in a date format, it strips off the decimals.
For more information on dates:

**International Dates in Access - Allen Browne**
http://allenbrowne.com/ser-36.html

**Brackets []**

Brackets are used around names that you make up, like fieldnames, form names, report names, etc. If you have been good and didn't use spaces or special characters, you usually don't have to type them. In code, you can skip them unless they are necessary. In a query, Access will adjust what you type – but pay attention! Sometimes Access erroneously thinks what you typed needs quotes not brackets (for instance, in an UpdateTo cell).

[MyFieldname]

**Take control references and variables out of a string**

When you reference a control or variable, it comes OUT of the string

```vba
dLookup("Fieldname","Tablename" _
,"someID=" & Me.ID_controlname)
```

If the control you are referencing is a string value, use quotes to delimit the value

```vba
dLookup("Fieldname","Tablename" _
,"SomeName='" & Me.Text_controlname & '""
```

if the control is a date value, use # to delimit the value

```vba
dLookup("Fieldname","Tablename" _
,"SomeDate=#" & Me.Date_controlname & '#")
```

**Line Continuation _**

While a Line Continuation character is not really a delimiter, it does have a special meaning when you are writing code. When you type a space and an underscore character at the end of a line, you are telling Access that the statement is continued on the next line.

**Bang !**

Use an exclamation point !, also called Bang (much easier to say <smile) when referring to a member of a collection

- `Forms!Formname` a particular form in the Forms collection
- `Reports!Reportname` a particular report in the Reports collection
- `Me!fieldname` a field in a form or report `RecordSource`
- `rs!fieldname` a field in a `RecordSet`

you can also identify members this way:

```vba
Forms("Formname")
Reports("Reportname")
```

this method offers greater flexibility because you can also do this:

```vba
Forms(stringVariableName)
```

And, in code …
'declare (dimension) variables you will use
dim rs as dao.RecordSet, mStr as string

'open a recordset
Set rs = CurrentDb.OpenRecordset("MyTable")

'if not at the end, store a value in a variable
If Not rs.Eof then
    mStr = rs!SomeStringField  'you HAVE to use Bang to refer to field in Recordset
End If

'close recordset and release object variable
rs.Close
Set rs = Nothing

The Bang allows you to "Late Bind", so if referring to a Field from a RecordSource, you do not have to have the RecordSource set in order to compile, because VBA will not verify that the object/field you specify after the bang is present upon Compile, VBA just assumes its there. When you are executing your code, VBA will try to resolve the reference and, if VBA can not resolve it, and error is raised. This concept of "late binding" is very handy if, for instance, you use VBA code to manipulate or set the RecordSource of your form upon form load.

Dot .

Use a Dot (period) when referring to a Property or Method that belongs to an object.

Me.NewRecord                NewRecord property (true or false) of a form
rs.AddNew                    AddNew record method of a Recordset
rs.Update                    Update (save) method of a Recordset
Me.controlname              Name of control in controls collection of form or report

You can use Me.fieldname, Me!fieldname, or Me.controlname when you are behind a form or report. I've heard that Bang has a slight speed advantage but I usually use Dot for the IntelliSense advantage while coding.
8. Where to Get More Help

There are several fantastic books on Access – visit your local bookstore or go to an on-line store like Amazon. Here are a few of my favorites:

**Access 2007 VBA Programmer's Reference** (Programmer to Programmer)
*by Teresa Hennig, Rob Cooper, Geoffrey L. Griffith, Armen Stein*

**Microsoft Office Access 2007 Inside Out**
*by John Viescas and Jeff Conrad*

**Access Developer's Handbook**
*by Paul Litwin, Ken Getz, and Mike Gilbert*
(Mike Gunderloy was co-author for version 2002)

**Expert Access 2007 Programming**
*by Rob Cooper and Michael Tucker*

**RibbonX : Customizing the Office 2007 Ribbon**
*Robert Martin, Ken Puls, Teresa Hennig, Oliver Stohr*

**Grover Park George On Access: Unleash the Power of Access**
*by George Hepworth*

**Access 2007 Programming**
*by Rob Cooper and Michael Tucker*

**Microsoft Office Access 2007 VBA (Business Solutions)**
*by Scott B. Diamond and Brent Spaulding*

*And so many more! Too numerous to mention!*

**Forums and Newsgroups**

Reading is a great way to learn but there is no substitute for interaction with others while you are figuring things out. There are a number of forums and newsgroups you can join for free and experienced folks graciously volunteer their time to answer your questions.

You are not alone! Here are 3 links to get you started:

**Utter Access**
http://www.utteraccess.com/

**Yahoo! Tech Groups -- MS Access Professionals**
http://tech.groups.yahoo.com/group/MS_Access_Professionals/

**Microsoft Communities**
http://www.microsoft.com/communities/default.mspx
Links
Aside from interactive communities, there are multitudes of websites to provide you with lots of great information, examples, and downloads. Here are some favorites:

The Access Web
http://www.mvps.org/access/

Allen Browne's tips for Microsoft Access
http://www.allenbrowne.com/tips.html

Stephen Lebans Website
http://lebans.com

Access Junkie (Jeff Conrad)
http://www.accessmvp.com/JConrad/accessjunkie.html

Tony's Main Microsoft Access Page
http://www.granite.ab.ca/accsmstr.htm

Roger's Access Library
http://www.rogersaccesslibrary.com/

Steve Schapel's Assorted Access Tips
http://accessstips.datamanagementsolutions.biz/

Viescas Consulting Links & Downloads
http://www.viescas.com/Info/links.htm

Lynn Trapp's LT Computer Designs
http://www.ltcomputerdesigns.com/

Arvin Meyer
http://www.datastrat.com/

Utter Angel's MS Access Downloads (Candace Tripp)

Q-Built Solutions Web site for Microsoft Access
http://www.access.qbuilt.com/

Tina's Tips to Get You Going
http://home.att.net/~california.db/tips.html

Access RunCommand Constants
http://www.tkwickenden.clara.net/

Database Solutions for Microsoft Access
http://www.databasedev.co.uk/
Access MVP  
http://www.accessmvp.com/

Access Links, by Tom Wickerath  
http://home.comcast.net/~tutorme2/samples/accesslinks.zip

Tutorials and Resources

Jeff Conrad's resources page  
http://www.accessmvp.com/JConrad/accessjunkie/resources.html

The Access Web resources page  
http://www.mvps.org/access/resources/index.html

MVP Allen Browne's tutorials  
http://allenbrowne.com/links.html#Tutorials

Luke Chung and FMS Free Resources  
http://www.fmsinc.com/free/index.html

Relational Database Design Articles, Tutorials and Tips  
http://www.databasedev.co.uk/general.html

Microsoft KnowledgeBase Search Page  
http://search.support.microsoft.com/search/?adv=1

Blogs

Clint Covington: Software design, Microsoft Office Access  
http://blogs.msdn.com/clintcovington

Access Team Blog  
http://blogs.msdn.com/access/

Alex Dybenko  
http://alexdyb.blogspot.com/

Tony's Microsoft Access Blog  
http://msmvps.com/blogs/access/

Access 2007 sites

Here are a couple sites devoted to Access 2007:

Access Junkie (Jeff Conrad, Microsoft)  
http://accessjunkie.com/default.aspx

Access-Freak (Oliver Stohr, freakazeud)  
http://www.access-freak.com/
# TABLE OF CONTENTS

1. **Types of Applications** ........................................................................................................................................ 2
   - Perspective ......................................................................................................................................................... 2
   - Word processing ............................................................................................................................................... 2
   - Spread sheets .................................................................................................................................................. 2
   - Databases ....................................................................................................................................................... 3
   - Graphics ......................................................................................................................................................... 3
   - Communication ............................................................................................................................................. 3
   - Web Browsers ............................................................................................................................................... 4

2. **Database Objects** ........................................................................................................................................ 5
   - Intro ............................................................................................................................................................... 5
   - An Access Application is a Container ......................................................................................................... 5
     - Tables ......................................................................................................................................................... 5
     - Queries ...................................................................................................................................................... 7
     - Query Criteria .......................................................................................................................................... 16
     - Forms ....................................................................................................................................................... 16
     - Reports ..................................................................................................................................................... 19
     - Macros ...................................................................................................................................................... 22
     - Modules ................................................................................................................................................... 28
     - IntelliSense .............................................................................................................................................. 33
   - Objects, Containers, Collections .................................................................................................................. 33
     - Properties and Methods .......................................................................................................................... 33
     - Events ....................................................................................................................................................... 33
   - VBA ............................................................................................................................................................. 33
     - Reference Libraries ............................................................................................................................... 34
     - VBA .......................................................................................................................................................... 34
     - Access ....................................................................................................................................................... 35
     - ActiveX Data Objects ............................................................................................................................. 35
     - OLE ........................................................................................................................................................... 35
     - DAO ......................................................................................................................................................... 35
     - Excel ......................................................................................................................................................... 35
     - Other libraries .......................................................................................................................................... 36

3. **Normalizing Data** ........................................................................................................................................ 37
   - Think of the different "nouns" that you will track: ....................................................................................... 37
   - Don't repeat data structures ....................................................................................................................... 38
   - Key Fields .................................................................................................................................................. 38
   - Data Type for Key Fields .......................................................................................................................... 38
   - DefaultValue .............................................................................................................................................. 38
   - Indexes ....................................................................................................................................................... 38
   - Lookup Fields in the table design .............................................................................................................. 39
   - Names ......................................................................................................................................................... 39
   - Reserved Words ....................................................................................................................................... 40
   - Numbers that aren't numbers ................................................................................................................... 40
     - InputMask ................................................................................................................................................. 40
     - Allow Zero Length .................................................................................................................................. 41
     - Captions .................................................................................................................................................... 41
     - Descriptions ............................................................................................................................................ 41
     - Order of Fields ....................................................................................................................................... 41
   - Tracking date record was created or modified .......................................................................................... 41
   - Combobox Example .................................................................................................................................. 42
   - Combobox Properties to Set ..................................................................................................................... 43
     - ControlSource ........................................................................................................................................ 43
6. SQL Syntax .............................................................................................................. 78
   SQL background ........................................................................................................ 78
   Types of Queries ........................................................................................................ 79
      ~~~ SELECT ~~~ ........................................................................................................ 79
      ~~~ APPEND ~~~ ...................................................................................................... 79
      ~~~ UPDATE ~~~ ....................................................................................................... 79
      ~~~ MAKE TABLE ~~~ ............................................................................................ 80
      ~~~ DELETE ~~~ ....................................................................................................... 80

5. Form and Report Design .......................................................................................... 52
   Toolbox ...................................................................................................................... 52
   Customize Form/Report Design toolbar ................................................................... 52
      Add Alignment and Order icons .............................................................................. 52
      Best-Fit Control Height and Width .......................................................................... 53
   Properties and Methods ............................................................................................ 54
      Turn on Properties window ...................................................................................... 54
      Setting Properties vs. Resizing with Handles ......................................................... 54
      Selecting Objects .................................................................................................... 54
      Select Form or Report ............................................................................................. 56
      Building Event Code ............................................................................................... 56
      Name property ........................................................................................................ 57
      ControlSource, SourceObject ................................................................................ 58
      RecordSource Property .......................................................................................... 60
      Builder Button ........................................................................................................ 63
      Events ...................................................................................................................... 65
      Deleting a Control .................................................................................................. 65
   Learning Properties and Methods .............................................................................. 66
      Help on Properties and Methods ............................................................................. 66
      Object Browser ....................................................................................................... 66
      General Help .......................................................................................................... 68
      Code Behind Form (CBF) ... or Report ................................................................... 68
      Rules For Code ........................................................................................................ 71
      Organizing Code and Other Information ............................................................... 72
      Creating a Main Form and Subform ....................................................................... 75

4. Relationships, Referential Integrity, Indexes .......................................................... 47
   Steps to Document Table Structure using Relationship Diagram .............................. 47
   Steps to take if you can't Enforce Referential Integrity ............................................. 48
   Indexes ...................................................................................................................... 48
   Unique Indexes .......................................................................................................... 50
   Steps to make a Multi-Field, Unique Index ............................................................... 50

More information on Normalization ........................................................................... 46

3. Creating a Main Form and Subform ........................................................................ 75
   Code Behind Form (CBF) ... or Report ................................................................... 68
   Rules For Code .......................................................................................................... 71
   Organizing Code and Other Information ................................................................... 72
   Creating a Main Form and Subform ........................................................................ 75
Access Basics for Programming: Table of Contents, List of Figures

~~~ CROSSTAB ~~ ................................................................. 80
Data Definition Queries ............................................................. 82
Aliases .................................................................................. 82
Joins ..................................................................................... 82
Parameters ........................................................................ 82
More information on SQL ...................................................... 82
Look At SQL! ........................................................................ 82
Using VBA to execute SQL statements ................................... 85
Subqueries ........................................................................... 85

7. Delimiters .............................................................................. 86
   "String in Double Quotes" .................................................. 86
   'String in Single Quotes' ..................................................... 86
   #Date# ............................................................................... 87
   Brackets [ ] .......................................................................... 88
   Take control references and variables out of a string ....... 88
   Line Continuation _ .......................................................... 88
   Bang ! .................................................................................. 88
   Dot . .................................................................................... 89

8. Where to Get More Help .......................................................... 90
   Forums and Newsgroups ...................................................... 90
   Links .................................................................................. 91
   Tutorials and Resources .................................................... 92
   Blogs .................................................................................. 92
   Access 2007 sites ................................................................ 92
LIST of FIGURES

Figure 2-1 Datasheet View of a Table
Figure 2-2 Design View of a Table
Figure 2-3 Datasheet View of a Select Query
Figure 2-4 Design View of a Query
Figure 2-5 SQL View of a Select Query
Figure 2-6 Fieldlist
Figure 2-7 Join Properties – Equi-Join
Figure 2-8 Join Properties - Right Join
Figure 2-9 Simple Query Example
Figure 2-10 Design View of Query Showing Birthdays
Figure 2-11 Form View - Main Form and Subforms
Figure 2-12 Design View of a Form
Figure 2-13 Print Preview of a Report - Two Columns
Figure 2-14 Design View of a Report
Figure 2-15 Design View of a Macro
Figure 2-16 AutoExec Macro
Figure 2-17 Form to Add Address for Macro Example
Figure 2-18 Macro Group Name shows in Database Window
Figure 2-19 Design View of Macro showing Macro Name and Condition Columns
Figure 2-20 Assign a Macro to the Form BeforeUpdate Event
Figure 2-21 Property Sheet for Form showing Macro Assignment
Figure 2-22 Macro Assigned to Click Event of Command Button
Figure 2-23 Transparent Command Button is drawn on top of a Label Control
Figure 2-24 AutoKeys Macro showing how to set a Global Shortcut key
Figure 2-25 VBE (Visual Basic Editor) – Design View of a Module
Figure 2-26 IntelliSense helps Write Code
Figure 4-1 Index Window showing Primary Key
Figure 4-2 Tools, Options - Delete AutoIndex Entries
Figure 4-3 Unique Index on StateName
Figure 4-4 Multi-Field Unique Index
Figure 5-1 Toolbox
Figure 5-2 Customize Form/Report Design Toolbar
Figure 5-3 Drag in ruler to Select Multiple Objects
Figure 5-4 Let go of the Mouse -- Anything the Line Touched is Selected
Figure 5-5 Select Object Combobox on Toolbar
Figure 5-6 Select Object using Combobox in Properties Window
Figure 5-7 Select Form (Or Report) by Clicking where Rulers Intersect
Figure 5-8 Name of Selected Object Appears in titlebar of Properties Window
Figure 5-9 Multiple Items Selected: Properties in common are Listed
Figure 5-10 Control with an Ambiguous Name
Figure 5-11 Change Name to match ControlSource for Bound Textbox Control
Figure 5-12 Name of a Subform is not necessarily the same as the SourceObject
Figure 5-13 Form RecordSource
Figure 5-14 RecordSource gives list of Tables and Queries
Figure 5-15 RecordSource can be SQL Statement
Figure 5-16 SQL Builder
Figure 5-17 Custom Color Palette - drag slider to lighten/darken color
Figure 5-18 Choose General Procedure from Combobox
Figure 5-19 Choose Code Object from Combobox
Figure 5-20 Object Events are Bold if the are Defined
Figure 5-21 Object Browser
Figure 5-22 Properties for a Control
Figure 5-23 Define an [Event Procedure]
Figure 5-24 Assign Function to an Event Property
Figure 5-25 Subform Control Selected
Figure 5-26 Form Contained by Subform Control is Selected
Figure 6-1 Datasheet View of a Phone List Query
Figure 6-2 Design View of a Phone List Query
Figure 6-3 SQL View of a Phone List Query