CHAPTER 1

RELATIONAL DATABASE DESIGN

INFOCUS

The key to a good database lies in effective and efficient design.

But just how do you go about converting your task or job into a database application? How do you know if it is created efficiently and that the application has been designed to be able to grow with your needs in the future?

The answer to these questions lies in database design. In this chapter we will explore some of the aspects and techniques associated with the design of good databases.

In this session you will:

- ✓ gain an understanding of how to design a relational database
- ✓ gain an understanding of how to scope the database system
- ✓ gain an understanding of how to determine inputs for a database
- ✓ gain an understanding of database normalisation
- ✓ gain an understanding of how to test for ***First Normal Form***
- ✓ gain an understanding of ***Second Normal Form*** (2NF)
- ✓ gain an understanding of how our case study looks after ***Second Normal Form***
- ✓ gain an understanding of how ***Third Normal Form*** can be applied to a database
- ✓ gain an understanding of indexing in databases.

DESIGNING A RELATIONAL DATABASE

As you will soon see, creating a database file with table structures in Microsoft Access is relatively easy to do. As a consequence there is an inclination amongst less experienced users to create a database file without giving much thought to the overall objective and ***design*** of the application. This is not the right way to approach database development.

Planning

Experienced database developers know that the more time and effort that they put into the overall planning of the database, the easier it will be to use and operate.

There are many ways of approaching the development of a database system and these ways vary depending upon the complexity of the system. Most developers use a ***top down approach*** – starting with the overall concepts, and then identifying the components required.

The Three Steps of Database Design

There are three main steps to designing an effective relational database:

1 Scope the System

Scoping is all about determining what the system should be able to do. You begin by establishing the overall aim of the system. Ask yourself one fundamental question: *"What is the database that I will construct supposed to do?"*

2 Determine the Inputs and Build A Prototype Table Structure

Once you have an idea of what information is required from the system you can determine what data needs to be entered into it. Here you determine the data that will be required to be entered into the system to achieve what the system is supposed to do. Once the inputs have been determined you can construct a test table or tables that will be used to store the data. This prototype table is often drawn on paper.

3 Normalise the Data

Arguably the most academic of the steps, normalising involves applying a set of database design rules to the prototype table. As you apply the rules, you adjust the table or tables until you have arrived at a network of tables that ensure efficient data storage and eliminate all possible examples of redundancy (repetition).

SCOPING THE SYSTEM

Scoping a database system requires you to determine what the system should ultimately be able to do. Scoping statements for systems are often written as a general aim. The aim is then expanded with a series of objectives which outline how the aim will be achieved through the system. It is important to think through as many scenarios and possible uses as possible.

How to Scope a System

To ***scope*** a new database system you begin by establishing the overall aim of the system. A useful approach is to ask the question: "What is the database that I will construct supposed to do?" Some sample aims might be:

My system is supposed to track employee expenses.

My system is supposed to track my fleet of hire cars.

My system is supposed to track customer purchases.

Once you have done this you can create the specific objectives of the system that will help it achieve its aim. For example, using the first aim above, you might have the following objectives:

To track employee expenses my system should:

- *show me the daily expense details for each employee*
- *produce expense reports*
- *produce a list of current employees.*

There would most likely be many more objectives – the objectives above are designed to give you an idea of what is required.

Scoping – A Case Study

In this courseware we will construct a relational database for the fictional company ***Alpheius Global Enterprises (AGE)*** that helps track expense claim forms.

At ***AGE***, goods, services and products can only be purchased through a rigorous purchase orders system. However, sometimes employees need to make quick purchases. At other times employees may be interstate or overseas, or entertaining guests and visitors, where it is not feasible or practical to raise an official purchase order. In these circumstances the employees can make purchases using their own money or credit cards and then claim the expense back on AGE. When this occurs employees are required to complete an ***Expense Form*** with the Admin & Accounts department. The money can be paid either by cheque or paid into the employee's salary.

Up until now the Admin & Accounts department have been recording these expense claim details in a spreadsheet. After only three months they already have over 700 forms entered into the spreadsheet. The big problem is that the spreadsheet is becoming unwieldy and doesn't allow them to produce worthwhile information without a lot of rearranging of the layout and the data.

So it has been decided to build an ***Expense Claim*** system in Microsoft Access. The scope of our project is to build an efficient Expense Claim system that allows AGE to track expense claims from employees.

The objectives of the system are to:

- produce reports showing all of the claims by type (e.g. meals, accommodation, etc), and by person
- produce a monthly expenditure report for each department
- analyse expenses
- enable easy entry of data by people not overly-familiar with computers or Access.

DETERMINING THE INPUTS

Once you have an idea of what information is required from the system, you can determine what data needs to be entered into it. This is really a brainstorming session where you try and anticipate all of the fields (columns) that you will need to meet the overall system objectives. When you have ***determined these inputs*** you can use them to create a prototype table.

How to Determine the Inputs

Start by having a look at the system aim and objectives. Imagine that the data to meet these objectives will be placed in one large table. Then make a list of all of the data columns that you would require in that table.

Inputs - The Alpheius Global Enterprises Case Study

To meet the objectives of our system we will need to record:

- the claim details (reference number, amount, type of expense)
- who submitted the claim
- the department of the person submitting the claim
- the various types of claims
- and perhaps some personal details about the employees for statistical analysis later

The information above will next need to be placed into a test table – not necessarily in Access, but maybe in a prototype spreadsheet or even on paper. We have created a mock structure and taken the first few records from a spreadsheet system to create the table shown below.

	A	B	C	D	E	F	G	H	I	J	K	L
1	No	Employee	Date	Accom A	Accom B	Postage	Gifts	Coffee	Other	Department	Date of Birth	Salary
2	1	Peter Dawson	02-Jan-08	$132.00	$0.00	$0.00	$0.00	$0.00	$0.00	Executive	12-Jul-54	$140,000.00
3	2	Julianne Kerr	02-Jan-08	$145.00	$0.00	$0.00	$0.00	$0.00	$0.00	Executive	05-Feb-60	$250,000.00
4	3	Syed Ali	02-Jan-08	$0.00	$0.00	$0.00	$27.06	$0.00	$0.00	Sales & Marketing	28-Nov-74	$54,000.00
5	4	Belinda Moore	02-Jan-08	$0.00	$0.00	$3.59	$0.00	$0.00	$0.00	Sales & Marketing	04-Dec-82	$51,000.00
6	5	Charles Morris	02-Jan-08	$0.00	$0.00	$16.99	$0.00	$0.00	$0.00	Administration	20-Dec-77	$84,000.00
7	6	Vivienne Clark	02-Jan-08	$154.50	$0.00	$0.00	$0.00	$0.00	$0.00	Administration	22-Nov-61	$80,000.00
8	7	Augustine Millson	02-Jan-08	$125.50	$0.00	$0.00	$0.00	$0.00	$0.00	Administration	07-Dec-78	$85,000.00
9	8	Elizabeth Dangaard	02-Jan-08	$0.00	$0.00	$0.00	$0.00	$0.00	$48.39	Research & Development	25-Aug-58	$25,000.00
10	9	Victor Brown	02-Jan-08	$0.00	$0.00	$0.00	$0.00	$18.26	$0.00	Administration	02-Apr-73	$81,000.00
11	10	George Samuelson	02-Jan-08	$0.00	$0.00	$0.00	$0.00	$7.72	$0.00	Administration	01-Dec-87	$98,000.00
12	11	Victoria McDonald	02-Jan-08	$123.44	$0.00	$0.00	$0.00	$0.00	$0.00	Research & Development	19-Apr-47	$65,000.00
13	12	Lance Williams	02-Jan-08	$0.00	$237.66	$0.00	$0.00	$0.00	$0.00	Administration	03-May-75	$83,000.00
14	13	Augustine Millson	02-Jan-08	$0.00	$0.00	$0.00	$0.00	$0.00	$0.00	Administration	07-Dec-78	$85,000.00
15	14	Augustine Millson	16-Jan-08	$155.60	$0.00	$0.00	$0.00	$0.00	$0.00	Administration	07-Dec-78	$85,000.00
16	15	Peter Dawson	02-Feb-08	$246.53	$0.00	$0.00	$0.00	$0.00	$0.00	Executive	12-Jul-54	$140,000.00
17	16	Julianne Kerr	02-Feb-08	$244.12	$0.00	$0.00	$0.00	$0.00	$0.00	Executive	05-Feb-60	$250,000.00
18	17	Syed Ali	02-Feb-08	$0.00	$0.00	$0.00	$43.23	$0.00	$0.00	Sales & Marketing	28-Nov-74	$54,000.00
19	18	Belinda Moore	02-Feb-08	$0.00	$0.00	$6.24	$0.00	$0.00	$0.00	Sales & Marketing	04-Dec-82	$51,000.00
20	19	Charles Morris	04-Feb-08	$0.00	$0.00	$18.32	$0.00	$0.00	$0.00	Administration	20-Dec-77	$84,000.00
21	20	Vivienne Clark	04-Feb-08	$294.54	$0.00	$0.00	$0.00	$0.00	$0.00	Administration	22-Nov-61	$80,000.00
22	21	Augustine Millson	04-Feb-08	$128.44	$0.00	$0.00	$0.00	$0.00	$0.00	Administration	07-Dec-78	$85,000.00
23	22	Elizabeth Dangaard	04-Feb-08	$0.00	$0.00	$0.00	$0.00	$0.00	$70.42	Research & Development	25-Aug-58	$25,000.00
24	23	Victor Brown	06-Feb-08	$0.00	$0.00	$0.00	$0.00	$22.46	$0.00	Administration	02-Apr-73	$81,000.00
25	24	George Samuelson	06-Feb-08	$0.00	$0.00	$0.00	$0.00	$13.82	$0.00	Administration	01-Dec-87	$98,000.00
26	25	Victoria McDonald	06-Feb-08	$123.66	$0.00	$0.00	$0.00	$0.00	$0.00	Research & Development	19-Apr-47	$65,000.00
27	26	Lance Williams	06-Feb-08	$0.00	$255.44	$0.00	$0.00	$0.00	$0.00	Administration	03-May-75	$83,000.00

Expenses

NORMALISING A DATABASE

Just because you have placed your data in a table structure doesn't necessarily mean that you have an efficient system. Indeed, the complete opposite may well be the case. There are actually five tests that can be applied to the tables of your system that will ensure that the system is functional and well designed. The application of these tests is known as ***normalising***.

What Is Normalising?

Normalising is actually a sequential set of tests, each to be performed only when the former has been satisfied. While the explanation of these tests may sound complicated, they are quite logical and based on common sense.

The point of the tests is to eradicate duplication of data and redundancy of data. While the table on the previous page may seem well-structured and organised, it is extremely inefficient and would be troublesome to implement in a relational database such as Microsoft Access.

In reality, only the first three forms are sufficient to ensure that you have a sound system (the other two tests were developed for special circumstances).

The First Three Tests of Normalisation

1 First Normal Form (1NF)

A table has passed the first test (called ***first normal form – 1NF***) when all of the fields cannot be divided any further and contain only a single value. In technical terms all fields in a table or tables should be ***atomic*** – which means that data cannot be divided any further.

2 Second Normal Form (2NF)

All fields in a table need to refer to a ***key*** field – this key field is often a ***primary key*** field which identifies each field as ***unique***. To do this, additional tables are often required to be developed and the duplicates from the main table are extracted and placed into these secondary, or lookup tables. In second normal form each table in your system must be given a ***primary key*** – a unique entry for each record. Each field in the table must refer to that, and only that, primary key. The purpose of this is to make it impossible to have duplicate records in a table.

3 Third Normal Form (3NF)

In the third normal form, all fields in a table must be ***mutually independent*** – in other words one field cannot rely on another. The best example of this would be in an invoicing application. It would be tempting to create at least three fields: ***Quantity***, ***Price***, and ***Total***. The *Total* field is really the price multiplied by the quantity.

FIRST NORMAL FORM (1NF)

A table has passed the first normalisation test (called ***first normal form – 1NF***) when all of the fields cannot be divided any further and contain only a single value. This sometimes takes on two variations. The first is where columns such as people's name or address can be further broken down. The second is where repeating columns for the same type of data appear.

Alpheius Global Enterprises Case Study

The first reason our table (shown earlier in the chapter) is not in first normal form is because the ***Employee*** field can be further broken down. We need to break our name fields into *First Name* and *Last Name*. This will make it much easier to sort the data by ***Last Name*** and to locate records for specific employees based on their ***Last Name***.

The second reason that our table is not in first normal form, is because there are ***repeating groups*** of fields in the table – this is a bit harder to explain. Currently there are seven types of expenses catered for in the table: *Accommodation A*, *Accommodation B*, *Postage*, *Meals*, *Tea and Coffee*, *Gifts*, and *Other*. What we are doing in each of these fields is storing the appropriate amount of expenditure. This type of layout is suited to a spreadsheet but is a disaster in a database and causes problems because it is more difficult in a database to sum across several columns and secondly, you'll have a mountain of work to do when you want to add an eighth category. All of the fields in the table are really just ***expense*** types. A much better way to represent these fields is in an ***Expense*** field and an ***Amount*** field.

After first normal form your table will appear as shown below.

No	Date	First Name	Last Name	Expense	Amount	Department	Date of Birth	Salary
1	02-Jan-08	Peter	Dawson	Accommodation A	$132.00	Executive	12-Jul-54	$140,000.00
2	02-Jan-08	Julianne	Kerr	Accommodation A	$145.00	Executive	05-Feb-60	$250,000.00
3	02-Jan-08	Syed	Ali	Gifts	$27.06	Sales & Marketing	28-Nov-74	$54,000.00
4	02-Jan-08	Belinda	Moore	Postage	$3.59	Sales & Marketing	04-Dec-82	$51,000.00
5	02-Jan-08	Charles	Morris	Postage	$16.99	Administration	20-Dec-77	$84,000.00
6	02-Jan-08	Vivienne	Clark	Accommodation A	$154.50	Administration	22-Nov-61	$80,000.00
7	02-Jan-08	Augustine	Millson	Accommodation A	$125.50	Administration	07-Dec-78	$85,000.00
8	02-Jan-08	Elizabeth	Dangaard	Other Expenses	$48.39	Research & Development	25-Dec-58	$25,000.00
9	02-Jan-08	Victor	Brown	Coffee and Tea Expenses	$18.26	Administration	02-Apr-73	$81,000.00
10	02-Jan-08	George	Samuelson	Coffee and Tea Expenses	$7.72	Administration	01-Dec-87	$98,000.00
11	02-Jan-08	Victoria	McDonald	Accommodation A	$123.44	Research & Development	19-Apr-47	$65,000.00
12	02-Jan-08	Lance	Williams	Accommodation B	$237.66	Administration	03-May-75	$83,000.00
13	02-Jan-08	Augustine	Millson	Meals	$52.86	Administration	07-Dec-78	$85,000.00
14	16-Jan-08	Augustine	Millson	Accommodation A	$155.60	Administration	07-Dec-78	$85,000.00
15	02-Feb-08	Peter	Dawson	Accommodation A	$246.53	Executive	12-Jul-54	$140,000.00
16	02-Feb-08	Julianne	Kerr	Accommodation A	$244.12	Executive	05-Feb-60	$250,000.00
17	02-Feb-08	Syed	Ali	Gifts	$43.23	Sales & Marketing	28-Nov-74	$54,000.00
18	02-Feb-08	Belinda	Moore	Postage	$6.24	Sales & Marketing	04-Dec-82	$51,000.00
19	04-Feb-08	Charles	Morris	Postage	$18.32	Administration	20-Dec-77	$84,000.00
20	04-Feb-08	Vivienne	Clark	Accommodation A	$294.54	Administration	22-Nov-61	$80,000.00
21	04-Feb-08	Augustine	Millson	Accommodation A	$128.44	Administration	07-Dec-78	$85,000.00
22	04-Feb-08	Elizabeth	Dangaard	Other Expenses	$70.42	Research & Development	25-Dec-58	$25,000.00
23	06-Feb-08	Victor	Brown	Coffee and Tea Expenses	$22.46	Administration	02-Apr-73	$81,000.00
24	06-Feb-08	George	Samuelson	Coffee and Tea Expenses	$13.82	Administration	01-Dec-87	$98,000.00
25	06-Feb-08	Victoria	McDonald	Accommodation A	$123.66	Research & Development	19-Apr-47	$65,000.00
26	06-Feb-08	Lance	Williams	Accommodation B	$255.44	Administration	03-May-75	$83,000.00

First Normal Form

SECOND NORMAL FORM (2NF)

First normal form eliminates duplication *horizontally* across the field columns. ***Second normal form (2NF)*** is designed to eliminate duplication in the records *vertically* down the table. This is usually done by identifying common ***entities*** and breaking a large table up into smaller ***entity*** tables which take on the role of *lookup* tables.

Alpheius Global Enterprises Case Study

After ***first normal form***, it is clear that there is data duplication in our table. For example, the names and other details of the employees are repeated each time the same employee incurs an expense transaction. It is clear from this, that *employees* form one entity while their *expense transactions* form another.

Second normal form requires that separate tables be created for each of these ***entities***, with the two tables linked using a common field as follows:

Original Structure

Expenses
No (pkey)
Date
First Name
Last Name
Expense
Amount
Department
Date of Birth
Salary

Revised Structure

Expenses
No (pkey)
Date
Expense
Amount
EmpNo → ***Employees***
EmpNo (pkey)
First Name
Last Name
Department
Date of Birth
Salary

We now have a two-table structure where one records the transactions (***Expenses***) and the other is used as a lookup table with information relating to employees (***Employees***). The two are linked using the primary key of the lookup table (***EmpNo***).

A closer study shows that we still have duplication in the ***Expenses*** table. Each time we add a new transaction we have to record the description of that expense: *Accommodation A*, *Accommodation B*, *Postage*, etc. In reality expenses themselves are a separate entity to expense transactions, so we can therefore further split our existing ***Expenses*** table so that our second revision looks as follows:

Revision 2

Expense Transactions
No (pkey)
Date
ExpTypeNo → ***Expenses***
ExpTypeNo (pkey)
Description
MaximumValueAllowed
Amount
EmpNo → ***Employees***
EmpNo (pkey)
First Name
Last Name
Department
Date of Birth
Salary

The advantage of putting entities into their own tables is that we can add more information about the entity that may be useful for reporting later – here we've added the ***MaximumValueAllowed*** field which will help us report if employees are overspending.

Another useful feature of splitting entities into tables is that we can create additional support entities. For example, in our case study it has been decided that we could add some more personal details about employees, such as home phone number, next of kin, and the like. Because this information should really be kept more confidential and discreet we can place it into a separate table linked back to the ***Employees*** table as shown below:

Revision 3

Expense Transactions
No (pkey)
Date
ExpTypeNo → ***Expenses***
ExpTypeNo (pkey)
Description
MaximumValueAllowed
Amount
EmpNo → ***Employees***
EmpNo (pkey) → ***Personal Details***
EmpNo (pkey)
First Name
Last Name
Department
Date Started
Date of Birth
FullTime
WeeklyHours
Home Phone
Next of Kin
Relationship
Salary

SECOND NORMAL FORM – CASE STUDY

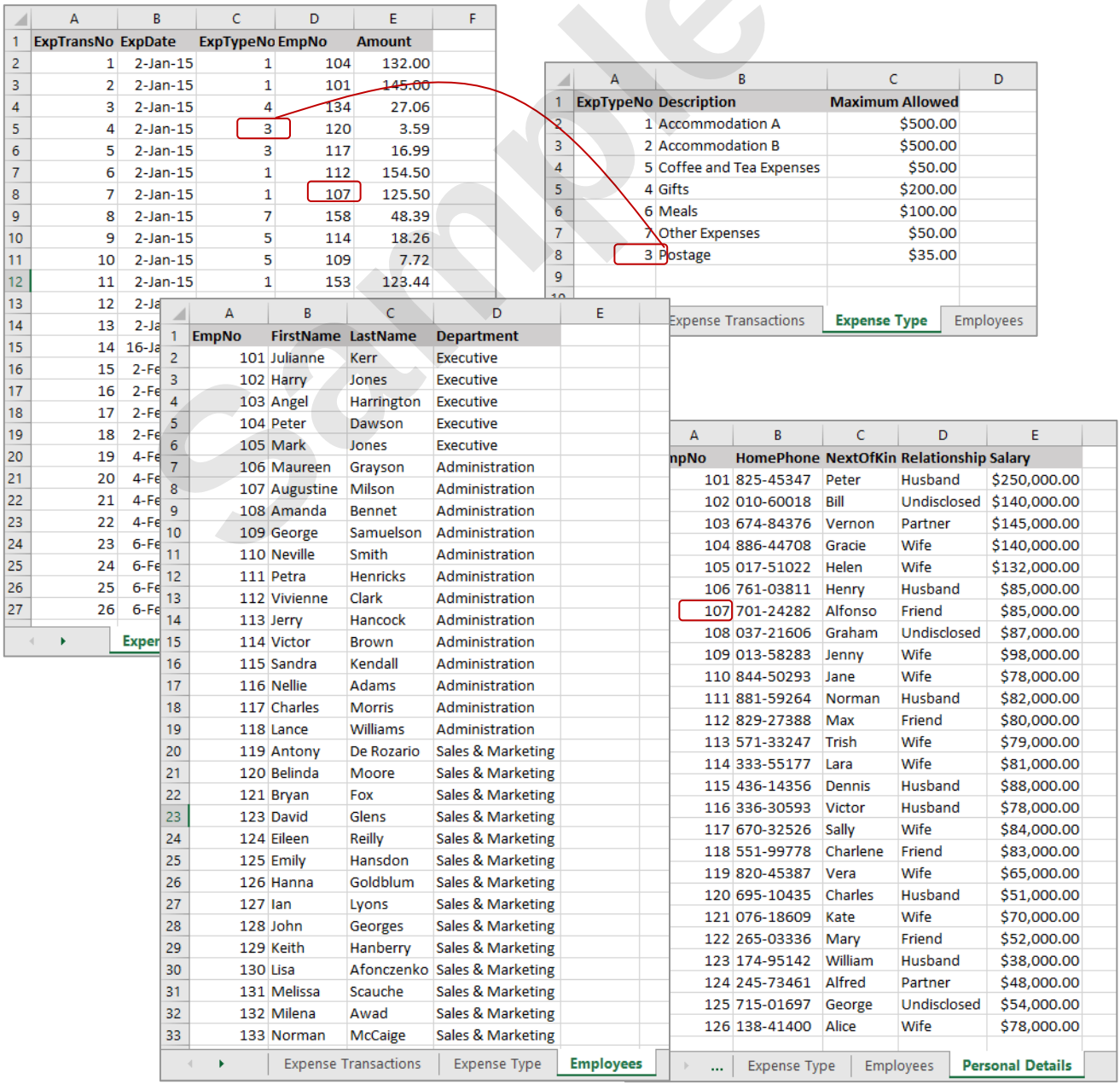
The tables for our case study, complete with the sample data, will work as shown below. Arguably, ***Second Normal Form*** takes the greatest amount of thinking and time to achieve and is often the result of working through tables and the case study several times. However, it is well worth the time and effort and will result in a far more efficient database.

1 Lookup Tables

The idea in our case study after *Second Normal Form* is that data for employees, their personal details, and expense types should only be entered once – into special tables known as ***lookup tables***. These ***lookup tables*** have at least one field that identifies each record as unique – the ***primary key***.

2 Transaction Tables

All other tables that require this information can reference the data using the same value as the primary key fields. The ***Expenses*** table is a ***transaction table*** – it has records that look up values in the other tables using codes that match the primary key values in the lookup tables. While each transaction is unique, there may be many transactions that are for the same employee or expense type.



ExpTransNo	ExpDate	ExpTypeNo	EmpNo	Amount
1	2-Jan-15	1	104	132.00
2	2-Jan-15	1	101	145.00
3	2-Jan-15	4	134	27.06
4	2-Jan-15	3	120	3.59
5	2-Jan-15	3	117	16.99
6	2-Jan-15	1	112	154.50
7	2-Jan-15	1	107	125.50
8	2-Jan-15	7	158	48.39
9	2-Jan-15	5	114	18.26
10	2-Jan-15	5	109	7.72
11	2-Jan-15	1	153	123.44

ExpTypeNo	Description	Maximum Allowed
1	Accommodation A	$500.00
2	Accommodation B	$500.00
5	Coffee and Tea Expenses	$50.00
4	Gifts	$200.00
6	Meals	$100.00
7	Other Expenses	$50.00
3	Postage	$35.00

EmpNo	FirstName	LastName	Department
101	Julianne	Kerr	Executive
102	Harry	Jones	Executive
103	Angel	Harrington	Executive
104	Peter	Dawson	Executive
105	Mark	Jones	Executive
106	Maureen	Grayson	Administration
107	Augustine	Milson	Administration
108	Amanda	Bennet	Administration
109	George	Samuelson	Administration
110	Neville	Smith	Administration
111	Petra	Henricks	Administration
112	Vivienne	Clark	Administration
113	Jerry	Hancock	Administration
114	Victor	Brown	Administration
115	Sandra	Kendall	Administration
116	Nellie	Adams	Administration
117	Charles	Morris	Administration
118	Lance	Williams	Administration
119	Antony	De Rozario	Sales & Marketing
120	Belinda	Moore	Sales & Marketing
121	Bryan	Fox	Sales & Marketing
123	David	Glens	Sales & Marketing
124	Eileen	Reilly	Sales & Marketing
125	Emily	Hansdon	Sales & Marketing
126	Hanna	Goldblum	Sales & Marketing
127	Ian	Lyons	Sales & Marketing
128	John	Georges	Sales & Marketing
129	Keith	Hanberry	Sales & Marketing
130	Lisa	Afonczenko	Sales & Marketing
131	Melissa	Scauche	Sales & Marketing
132	Milena	Awad	Sales & Marketing
133	Norman	McCaige	Sales & Marketing

EmpNo	HomePhone	NextOfKin	Relationship	Salary
101	825-45347	Peter	Husband	$250,000.00
102	010-60018	Bill	Undisclosed	$140,000.00
103	674-84376	Vernon	Partner	$145,000.00
104	886-44708	Gracie	Wife	$140,000.00
105	017-51022	Helen	Wife	$132,000.00
106	761-03811	Henry	Husband	$85,000.00
107	701-24282	Alfonso	Friend	$85,000.00
108	037-21606	Graham	Undisclosed	$87,000.00
109	013-58283	Jenny	Wife	$98,000.00
110	844-50293	Jane	Wife	$78,000.00
111	881-59264	Norman	Husband	$82,000.00
112	829-27388	Max	Friend	$80,000.00
113	571-33247	Trish	Wife	$79,000.00
114	333-55177	Lara	Wife	$81,000.00
115	436-14356	Dennis	Husband	$88,000.00
116	336-30593	Victor	Husband	$78,000.00
117	670-32526	Sally	Wife	$84,000.00
118	551-99778	Charlene	Friend	$83,000.00
119	820-45387	Vera	Wife	$65,000.00
120	695-10435	Charles	Husband	$51,000.00
121	076-18609	Kate	Wife	$70,000.00
122	265-03336	Mary	Friend	$52,000.00
123	174-95142	William	Husband	$38,000.00
124	245-73461	Alfred	Partner	$48,000.00
125	715-01697	George	Undisclosed	$54,000.00
126	138-41400	Alice	Wife	$78,000.00

THIRD NORMAL FORM (3NF)

With ***third normal form (3NF)*** all fields in a table must be mutually exclusive – in other words, one field cannot rely on another. This concept of reliance is yet another example of redundancy and often occurs when a field that could be calculated from existing fields is included in the table. For example, a ***Tax Rate*** can be stored, but the actual ***Tax*** amount can be calculated.

Third Normal Form

The best example of a database requiring the ***Third Normal Form*** (3NF) test would be in an invoicing application. It would be tempting to create at least three fields: ***Quantity***, ***Price***, and ***Total***. The ***Total*** field is really the price multiplied by the quantity.

This table would not be in third normal form because the ***Total*** field relies on both the ***Quantity*** field and the ***Price*** field. In databases the ***Total*** can easily be calculated for you. Therefore any field that can be calculated from existing fields in the data should be removed.

Another example might be an ***Age*** field. For example, let's say in our case study we recorded the age of the employees instead of their date of birth. The data would quickly become inaccurate. So while we might enter Michelle's age as 41, within a year it will be wrong. Computers can calculate age from a fixed date such as date of birth. So in ***Third Normal Form*** instead of having an *Age* field you would have a *Date Of Birth* field.

Alpheius Global Enterprises Case Study

Fortunately, in our case study, we do not have this issue with any of our tables.

DATABASE INDEXING

Indexes in books are used to help you find topics more quickly. Indexes in databases are used to help Access find records more quickly. In databases with large volumes of data, or multi-user environments, indexes can significantly improve the performance of your database and therefore the satisfaction of your users. This page examines index concepts in more detail.

What is an Index?

An index is a list of pointers to the location of data. Access uses the index to find data in the same way that you would use an index in a book – it looks up the location of the data in the index.

What are Indexes Used For?

Access uses indexes to ***sort*** or ***search*** for data. Given that indexes hold pointers rather than the data itself, they can be re-sorted and searched much more quickly than the database. The sort order is determined by the data type of the field that the index is created on.

For example, an index on a text field can be used to sort the records in ascending or descending alphabetical order. An index on a numeric field can be used to sort the records in ascending or descending numerical order. An index on a date field can be used to sort the records in chronological order – most recent to oldest, or vice versa.

The Primary Key

In order for a relational database to locate information in separate tables, each record must be unique in some way. The field that contains the unique value is the ***primary key***. The primary key is the main index for a table and is indexed automatically. Each table must have a primary key.

Examples of Fields Used for the Primary Key

Some fields are more suitable than others for use as a primary key. For example, *LastName* would not be useful because it is likely that there will be duplicates. *EmployeeNo*, however, will be unique for each employee and is therefore a far better candidate. Any ID number that is unique to a record is perfect. If your data does not include ID numbers, you can use the ***AutoNumber*** data type to assign a unique number to each record.

Using Additional Indexes

Indexes are used to locate data, but they can also slow the operation of the database down. You should only index a field (or fields) if all of the following criteria are satisfied:

- The data type of the field is ***Text***, ***Number***, ***Currency*** or ***Date/Time***
- The field is one you expect to search on frequently e.g. *LastName*
- The field is one you expect to sort on frequently e.g. alphabetic order
- You expect that most of the values stored in the field will be different. An index will not speed up queries if many of the values in the field are the same.

Single-Field Indexes

Single field indexes are those which refer to one field only. For example, you may decide to create an index on the *DeptNo* field in the *Employee* table because it is a field that you frequently use to access information from the *Department* table via a join in a query.

Multi-Field Indexes

Multi-field indexes are used when you often search or sort by two or more fields at a time. For example, if you often search for a combination of the *LastName* and *FirstName*, it makes sense to create a multiple-field index on both fields.

CHAPTER 2

INFOCUS

CREATING A RELATIONAL DATABASE

A database application requires the creation of a database file and appropriate table structures. In Microsoft Access your complete relational database application is stored in one database file.

After you have planned your system, the first task is to create a new database file which becomes the repository for all of the tables, reports, forms, and other objects of your system.

When the database has been created, you can populate it with the necessary tables and data.

In this session you will:

- ✓ learn how to create a new database file
- ✓ learn how to create the lookup tables
- ✓ learn how to define a primary key for a table
- ✓ learn how to save and close a new table design
- ✓ learn how to create a second lookup table in a database
- ✓ learn how to create the transactions table
- ✓ learn how to create the details table.

CREATING A NEW DATABASE FILE

In Microsoft Access 2016 all elements of your database; tables, reports, forms, and the like, are stored in one file with the file extension ***accdb***. This is what is commonly referred to as the database file – not to be confused with the tables where your data is stored. Before you can create tables, or reports, or forms, or any other object, you need to create a new database file.

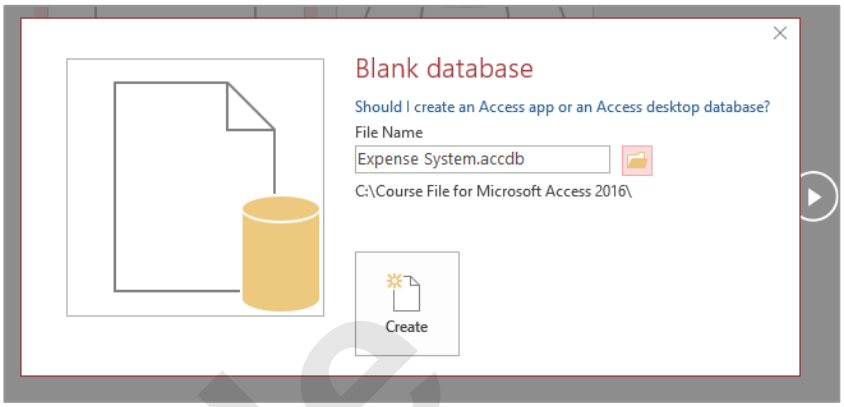
Try This Yourself:

Before you begin, ensure that Access has started...

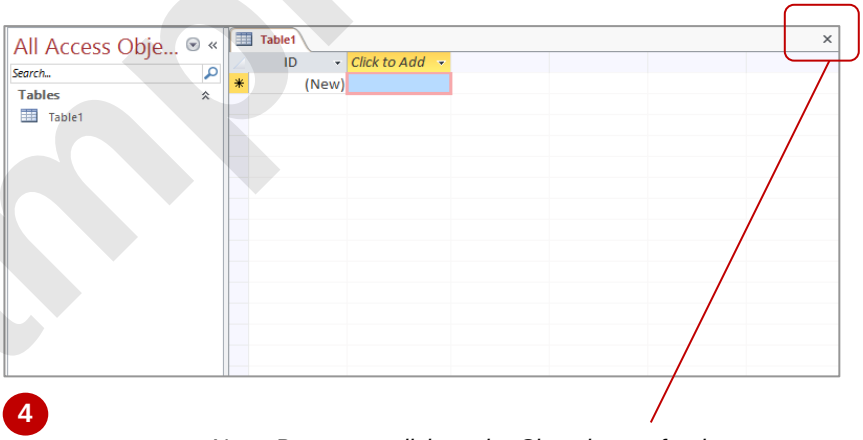
1. Click on the ***File*** tab to display the ***New*** area in the ***Backstage***
2. Click on the ***Blank desktop database*** template and type **Expense System** in ***File Name***

   We'll save it where the other course files are located…
3. Click on **[Browse]** to display the ***New Database*** dialog box, then locate and click on the ***Course Files for Microsoft Access 2016*** folder and click on **[OK]**

   The course files folder is now where the database will be saved…
4. Click on **[Create]** to create the new database
5. Click on ***Close*** to close the automatic table (***Table1***) that has appeared



3



4

Note: Be sure to click on the Close button for the database object when closing tables, queries, forms etc – if you click on the Close button in the very top right corner, you will close Access

For Your Reference…

To ***create*** a ***new database file***:

1. Click on the ***File*** tab and click on **New**
2. Click on ***Blank database***, type the ***File Name***, click on **[Browse]** and choose a save location
3. Click on **[OK]** then click on **[Create]**

Handy to Know…

- All new Access 2016 database files will be saved in the same format as Access 2007 and 2010 files (***.accdb***). If you need to provide the file to other users who may be using earlier versions, you can save it as an Access 2003 or earlier file (***.mdb***), but the file may lose some functionality.