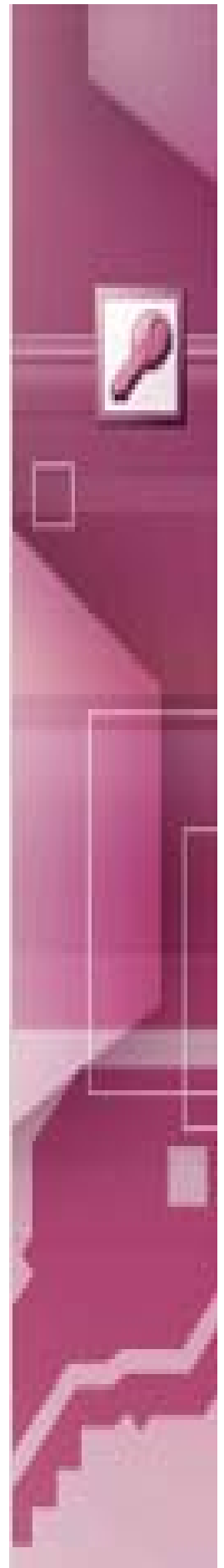


# ***INTRODUCTION TO MICROSOFT ACCESS 2010***

v. 2.0

October 2012

*nikos dimitrakas*



## Table of contents

<b>1 Introduction</b> .....	<b>4</b>
<b>1.1 Microsoft Access</b> .....	<b>4</b>
<b>1.2 Prerequisites</b> .....	<b>4</b>
1.2.1 Literature .....	4
<b>1.3 Structure</b> .....	<b>5</b>
<b>2 The Case</b> .....	<b>5</b>
<b>3 The Access Environment</b> .....	<b>8</b>
<b>3.1 Configuration</b> .....	<b>10</b>
<b>3.2 SQL</b> .....	<b>10</b>
<b>4 Creating A Database</b> .....	<b>11</b>
<b>4.1 Creating Tables</b> .....	<b>11</b>
4.1.1 Design.....	11
4.1.2 DDL.....	18
4.1.3 Defining Other Restrictions.....	20
<b>4.2 Working With Relationships</b> .....	<b>24</b>
4.2.1 Simple Foreign Keys .....	27
4.2.2 ISA Inheritance.....	30
4.2.3 Composite Foreign Keys .....	33
4.2.4 Multiple Relationships Between The Same Two Tables .....	35
4.2.5 Recursive Relationships .....	35
<b>5 Querying A Database - Working With Data</b> .....	<b>37</b>
<b>5.1 Preparing The Database With Data</b> .....	<b>37</b>
5.1.1 Using SQL.....	37
5.1.2 Using Datasheets .....	39
5.1.3 Using Forms .....	41
<b>5.2 Writing SQL</b> .....	<b>41</b>
<b>5.3 Reusing Queries</b> .....	<b>43</b>
<b>6 Forms</b> .....	<b>45</b>
<b>6.1 Simple Forms</b> .....	<b>45</b>
<b>6.2 Lookups</b> .....	<b>52</b>
<b>6.3 Master-Detail Constructs</b> .....	<b>65</b>
<b>6.4 Forms Based On Queries</b> .....	<b>77</b>
<b>6.5 Non-Data Forms</b> .....	<b>83</b>
<b>7 Reports</b> .....	<b>85</b>
<b>7.1 Simple Reports</b> .....	<b>85</b>
<b>7.2 Reports That Combine Many Tables</b> .....	<b>86</b>

7.3 Reports Based On Queries .....	90
7.4 Grouping And Sorting .....	91
7.5 Subreports .....	97
<b>8 Macros .....</b>	<b>103</b>
<b>9 Other Useful Tips .....</b>	<b>107</b>
9.1 Tip 1 - Lookups For Tables .....	107
9.2 Tip 2 - Viewing Subtables.....	108
9.3 Tip 3 - Sorting And Filtering .....	112
9.4 Tip 4 - SQL Parameters.....	112
9.5 Tip 5 - Nesting SELECT Statements – COUNT(DISTINCT).....	113
9.6 Tip 6 - Application Start-Up .....	114
9.7 Tip 7 - Concatenating Columns .....	115
9.8 Tip 8 - Using Forms To Find Records.....	116
9.9 Tip 9 - Keys And Indexes .....	116
9.10 Tip 10 - Multiple Subforms .....	117
9.11 Tip 11 - Division In Access .....	118
9.12 Tip 12 - Object Dependencies.....	119
9.13 Tip 13 - Copying Objects Between Databases .....	119
9.14 Tip 14 - Handling NULL .....	120
9.15 Tip 15 - Business Rules .....	120
9.16 Tip 16 - Set Operators .....	122
9.17 Tip 17 - Multimedia .....	122
9.17.1 Storage Outside The Database.....	128
9.18 Tip 18 - Compacting And Repairing A Database .....	129
9.19 Tip 19 - Linking External Data.....	130
9.19.1 Creating An ODBC Alias .....	131
9.19.2 Linking To The MySQL Tables From Access .....	133
9.19.3 Working With Linked Tables .....	135
9.20 Tip 20 - Working With Dates And Times .....	138
<b>10 Other Resources .....</b>	<b>140</b>
10.1 Web Sites.....	140
10.2 Books .....	140
<b>11 Epilogue.....</b>	<b>141</b>

# 1 Introduction

This compendium contains an introduction to the most commonly used functionalities found in Microsoft Access 2010, from how to create a database and define referential integrity to how to create forms and reports. As an added bonus, most of the information in this compendium also applies to earlier versions of Microsoft Access.

**It is strongly recommended that you read through (or at least look through) the entire compendium before you start working with it in front of a computer.** There are many references back and forth in this compendium, and therefore you should find it quite helpful to have acquired an idea beforehand of what is coming in later chapters.

Any comments or feedback that you may have about this compendium are greatly appreciated. Send any such comments or feedback to the author at [nikos@dsv.su.se](mailto:nikos@dsv.su.se).

The latest version of this compendium, all the files needed to complete the tutorial, relevant links and other information are available at <http://coursematerial.nikosdimitrakas.com/access/>.

## 1.1 Microsoft Access

Microsoft Access integrates a database management system and a rapid application development environment in the same package. It provides almost all basic relational database functionalities, and it extends this with facilities for rapid application development. Advanced development can also be done in Microsoft Access by using the also integrated Visual Basic environment.

Microsoft Access 2010 is included in the list of software offered by Microsoft within the MSDN Academic Alliance agreement. This means that any student at KTH/ICT or SU/DSV is entitled to one free licence for MS Access 2010. If you want to download Microsoft Access 2010 (or any other Microsoft software covered by Microsoft Dreamspark), go to [https://msdn60.e-academy.com/kgth\\_it/](https://msdn60.e-academy.com/kgth_it/) or [https://msdn60.e-academy.com/su\\_ids/](https://msdn60.e-academy.com/su_ids/).

From now on in this compendium we will refer to Microsoft Access 2010 as Access.

## 1.2 Prerequisites

This is a tutorial about Access, so the reader must already be familiar with conceptual modeling, relational database theory and some basic programming. Later in this compendium we will start working with a small case. We will skip to having a ready conceptual model, so we will assume that some conceptual modeling of our case was already done. The translation of the conceptual schema into a relational database schema will be shown, but not in any detail.

### 1.2.1 Literature

While working with this compendium it is recommended that you have some sort of reference literature on relational databases and SQL. Here are some recommended books:

- ❖ Connolly, Begg: *Database Systems A Practical Approach to Design, Implementation and Management*, Addison Wesley
- ❖ Elmasri, Navathe, *Fundamentals of Database Systems*, Addison-Wesley



There are many more books that will do just fine, but these two are mentioned here since they are the ones used for courses at SU/DSV and KTH/ICT.

## 1.3 Structure

This compendium has the following simple structure:

1. A short introduction (which you are reading right now)
2. A description of the case used throughout the compendium (chapter 2)
3. Creating a database for our case in Access (chapter 4)
  - a. Defining and creating the tables (section 4.1)
  - b. Defining relationships and referential integrity (section 4.2)
4. Querying the database (chapter 5)
  - a. Populating (putting some data in) the database so that we have something to query about (section 5.1)
  - b. Writing SQL statements to query the database (section 5.2)
  - c. Creating and using views (section 5.3)
5. Creating forms for input and for working with the data in the database (in a more user-friendly way) (chapter 6)
6. Creating reports for presenting data from the database (chapter 7)
7. Creating macros to do things that can't be done with just queries, forms and reports (chapter 8)
8. Finally there are some more tips and links to more information (chapters 9, 10 and 11)

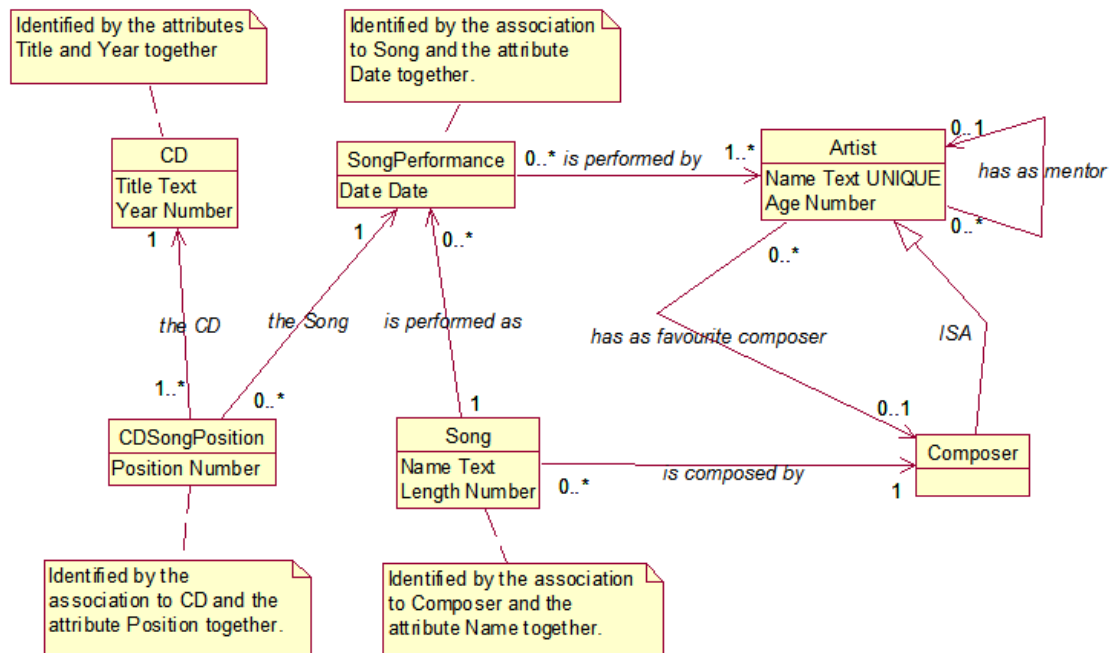
## 2 The Case

The case used in this compendium has been specifically designed in order to be both small and cover all the things to be discussed in the chapters to follow. The same case will be used for all the exercises in the rest of the compendium.

The system we are going to build will manage the following information:

- There are many artists, and for each of them we know their name and age. No two artists have the same name.
- Some of the artists are also composers.
- Composers compose songs. A composer never composes two songs with the same name.
- For each composed song we know its name and length (in seconds).
- Each song can be performed on particular dates. Each song performance can involve many different artists. The same song cannot be performed twice on the same date.
- A particular performance of a song can be included on a CD. A CD can contain many different songs performed by different artists.
- For each CD we know the order of the songs.
- Each CD has a title and a year (when it was released). No two CDs released the same year have the same title.
- A CD can only contain songs performed the same year or earlier (for obvious reasons).
- Many artists have a mentor, who is another artist. The mentor must be older than the artist.
- An artist can have a favorite composer.

The information above has been modeled into the following conceptual model:

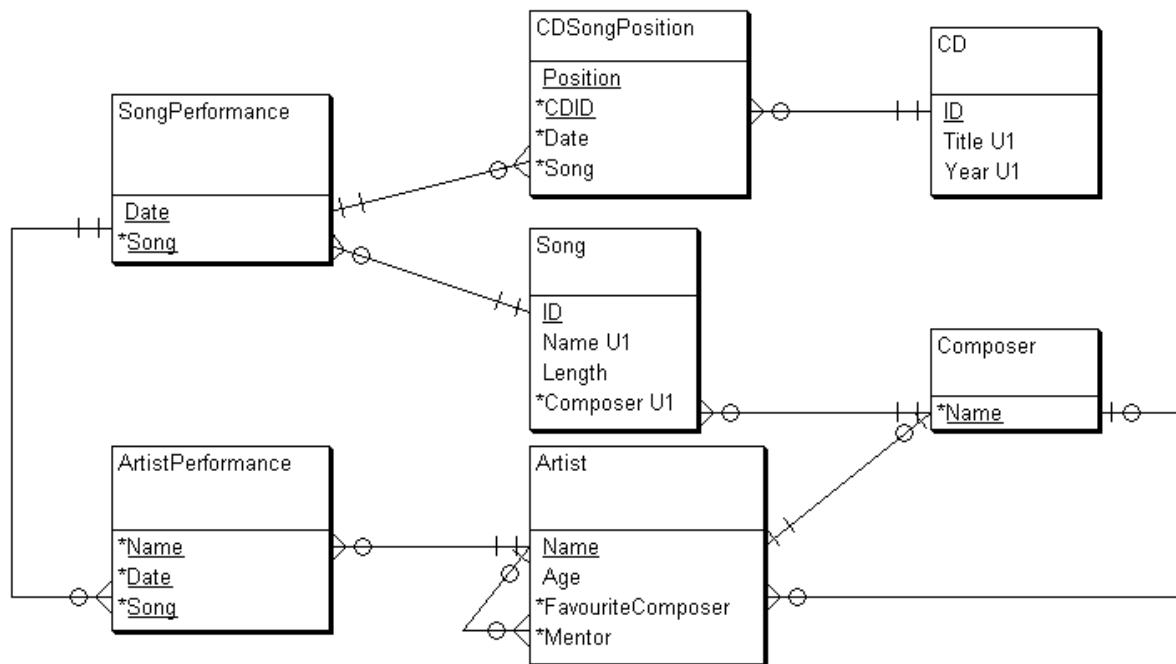


**Figure 1 Conceptual model of the case**

The arrows on the conceptual model are only there to help read the associations, for example “A song is composed by a composer” instead of “A composer is composed by a song”. The only two things not modeled are the facts that “A CD can only contain songs performed the same year or earlier” and that “The mentor must be older than the artist”. These will be handled as business rules, and we will see how we can add such restrictions in our database system later.

Before we can implement our database, the conceptual model has to be translated into a logical relational database schema. In this schema we will still not specify any Access specific information. We will specify primary keys, foreign keys, data types, and other restrictions.

The following figure shows the logical database schema created from the conceptual model. Primary keys are shown as underlined columns, while an asterisk (\*) indicates columns that constitute foreign keys. The columns CD.Title, Song.Name and Artist.Name are of data type STRING (or VARCHAR). The columns CD.ID, CD.Year, Artist.Age, Song.ID and Song.Length are of data type INTEGER. The column SongPerformance.Date is of data type DATE. All foreign key columns are automatically of the same data type as the referenced columns. Remember that keys can be composite, i.e. consist of more than one column, and that there can be several foreign keys in a table, possibly even sharing some column(s).



**Figure 2 Logical relational database schema of the case**

The schema of Figure 2 can also be shown in the following textual notation. The advantage of this textual notation is that the foreign keys are specified explicitly and there is no room for confusion.

**Tables (primary keys are underlined):**

- Artist (Name, Age, FavouriteComposer, Mentor)
- Composer (Name)
- Song (ID, Name, Length, Composer)
- SongPerformance (Date, Song)
- ArtistPerformance (Name, Date, Song)
- CD (ID, Title, Year)
- CDSongPosition (Position, CDID, Date, Song)

It is also possible to include the data types in this notation. The table Artist could be written instead as follows:

Artist (Name STRING, Age INTEGER, FavouriteComposer STRING, Mentor STRING)

**Foreign keys (foreign key on the left, referenced primary (or alternate) key on the right):**

- Artist.FavouriteComposer << Composer.Name
- Artist.Mentor << Artist.Name
- Composer.Name << Artist.Name
- Song.Composer << Composer.Name
- SongPerformance.Song << Song.ID
- ArtistPerformance.Name << Artist.Name
- ArtistPerformance.(Date, Song) << SongPerformance.(Date, Song)
- CDSongPosition.CDID << CD.ID
- CDSongPosition.(Date, Song) << SongPerformance.(Date, Song)

The first row says that the column FavouriteComposer of the table Artist is a foreign key to the column Name of the table Composer. For composite keys there is no difference: The last row, for example, says that the columns Date and Song of the table CDSongPosition constitute together a foreign key to the primary key of the table SongPerformance, namely the columns Date and Song.

Other constraints:

UNIQUE (CD.Title, CD.Year)

UNIQUE (Song.Name, Song.Composer)

In the next chapter we will see how we can create an Access database based on the relational database schema that we acquired earlier.

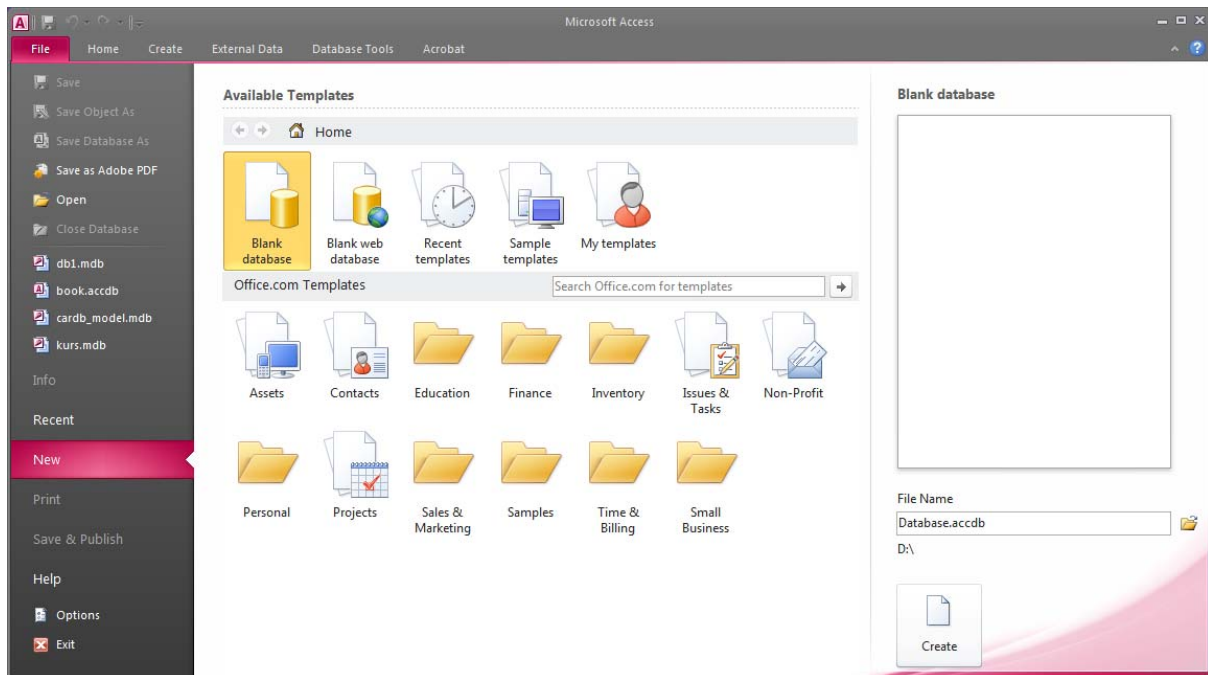
Our case also includes the following information needs and user interface:

1. Show all CD titles produced in 1999!
2. Show all songs in a particular CD!
3. Which CDs include songs written by Jerry Goldsmith?
4. Which song has been performed the most times?
5. How many distinct songs has each artist performed in?
6. Which artist has performed in at least one song of each CD?
7. Which artist has performed in at least one song of each composer?
8. Which songs has each composer composed?
9. A form for registering a new CD in the database.
10. A form for registering a new Artist in the database.
11. A form for registering song performances and artists performing them.
12. A report that shows the content of each CD (back cover style).
13. A report that shows information about each CD including which artists and composers that are related to the CD.
14. A report that shows for each composer the songs that they have composed and which performances of them exist and in which CDs these performances are included.

### **3 The Access Environment**

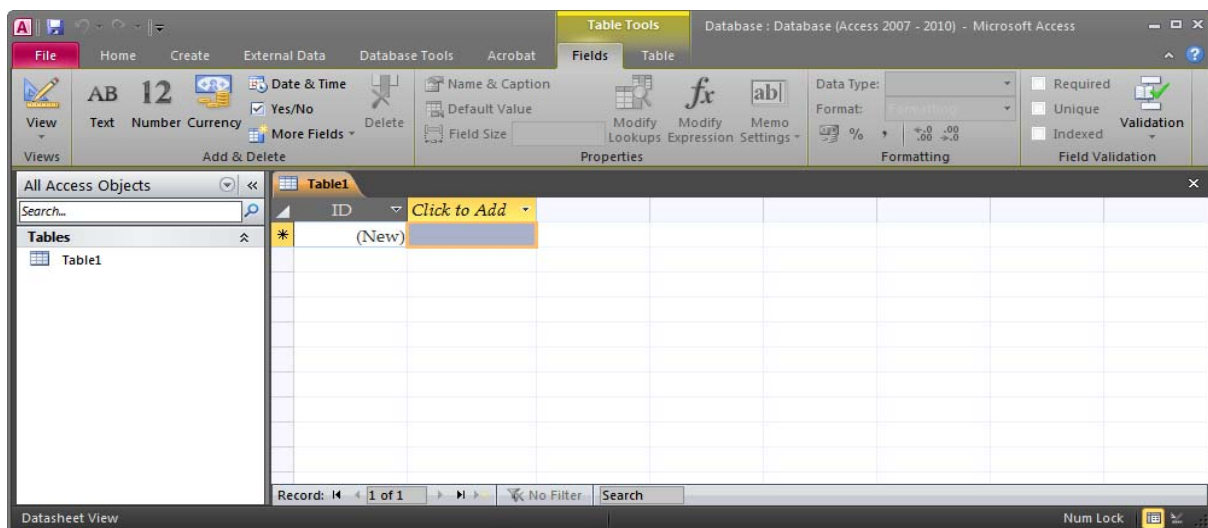
As we mentioned in section 1.1, Access is both a database management system and an application development environment. Access uses the basic philosophy of the other products in Microsoft Office, which means that a database (and accompanying application) is stored in a file (similar to Word, Excel and PowerPoint). Access files use the extension "accdb". Creating such a file is the equivalent to creating a database (done in other products with the SQL command CREATE DATABASE).

Starting Access without opening a particular file shows a welcome menu for creating a new file (a new database):



On the left side of the window there is the main menu and under "New" we have the Blank database template (and some other templates) which will create a new file. The file name can be specified on the right side before pressing "Create".

Once a new database has been created, Access will automatically suggest that we create a table:



On the top of the window we have the menu (File, Home, Create, etc.). Each option has its own toolbar and options will be active or inactive based on the current selection. The menu and corresponding toolbars is called "Ribbon". On the left side we have the object browser. Objects are not to be confused with objects in object-oriented programming. In Access we have six types of objects: Tables, Queries (views), Forms, Reports, Macros and Modules. All objects of these types will be shown in the object browser. The object browser is the control center of our database and is also known as the "Navigation Pane". From here we can open any table, query or other object in order to use it or modify it. The main area of the window (to the right of the object browser) is where we work with any objects we may open.

### 3.1 Configuration

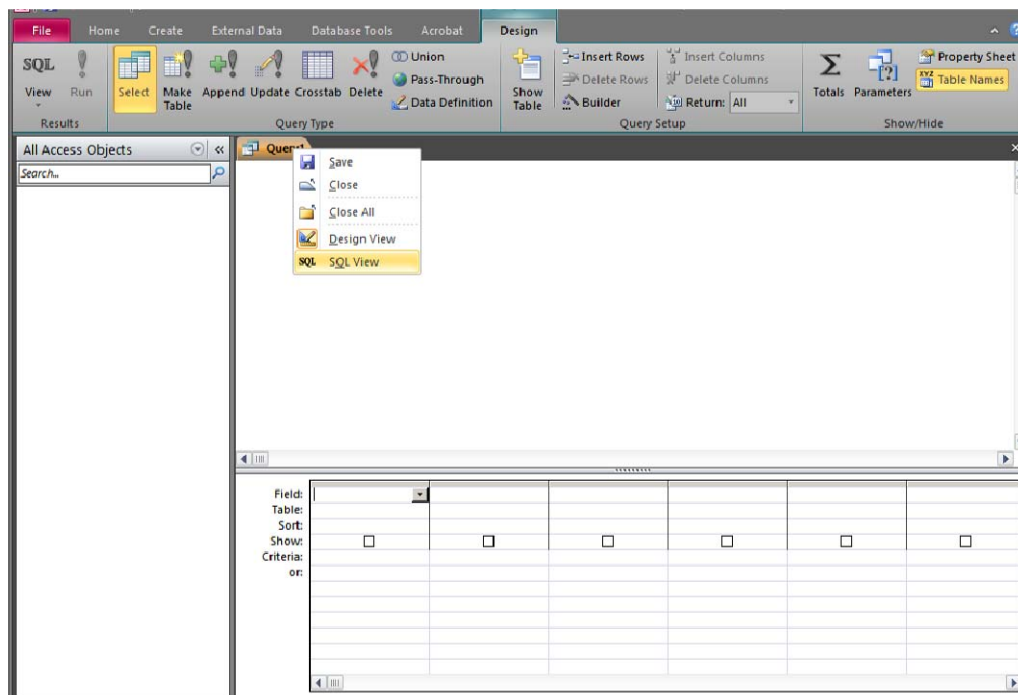
Access and the current database can be configured from File > Options. Some interesting settings are available under Current Database and under Object Designers.

### 3.2 SQL

Access is a relational database management system and thus supports SQL. But Access seems to encourage users to use wizards and graphical tools, so SQL is not really up, front and center. In order to write and execute SQL, we have to first create a query and then switch to the SQL view. A query can be created from Create > Query Design. Access will immediately suggest that we add tables to a graphical design of our query:



After ignoring the "Show Table" pop-up window, we will have the option of switching to the SQL view either by pressing "SQL" on the ribbon (under Design) or by right-clicking on the query's tab selector and selecting "SQL View":



Queries can be saved as database objects and each query can contain one SQL statement (SELECT, UPDATE, INSERT, DELETE, CREATE TABLE, etc). Saving SELECT statements as queries is equivalent to creating views using CREATE VIEW, which is not supported in Access.

## 4 Creating A Database

Given the relational database schema created in chapter 2, we will now create a database in Access. We will start by creating the tables, defining their columns, data types and primary keys, and alternate keys if any. When all the tables are in place we will move on to establishing the relationships between tables, i.e. the foreign key relationships.

Before we can start creating tables though, we have to create a database. A database in Access is a file with the extension "accdb". Note that, with the exception of Access 2007, the previous versions use a different format and the extension "mdb".

Start Access from the start menu and create a blank database as described in the previous chapter. Place the new file at a suitable location. You can always move the file later if you want.

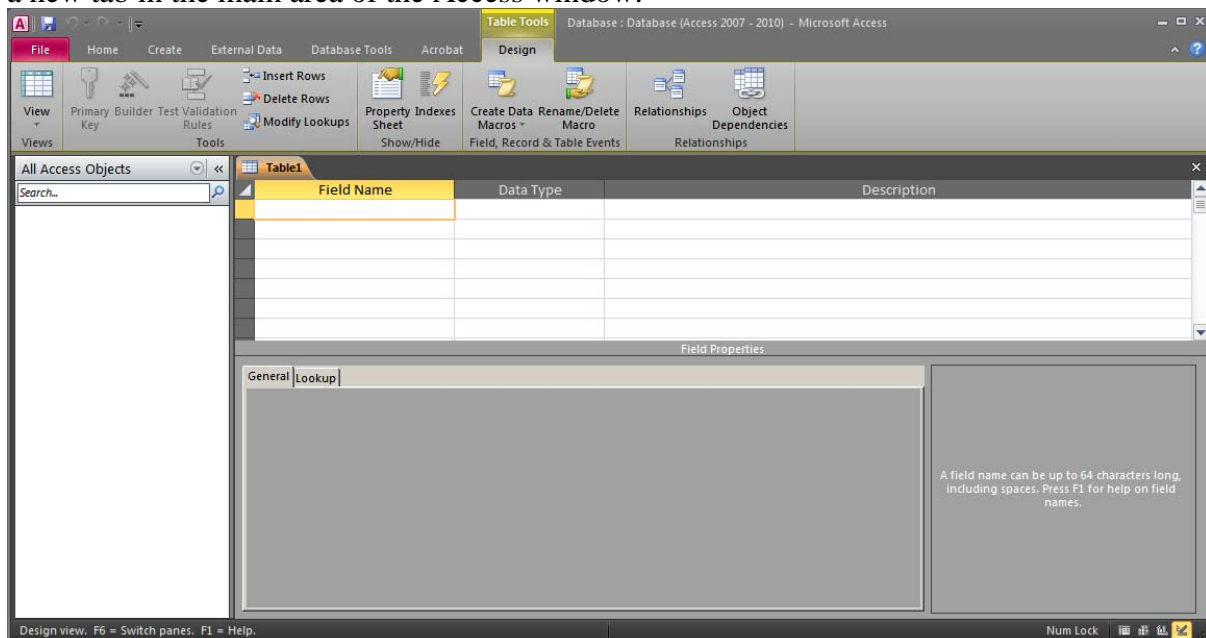
Access starts by suggesting that a new table be created. Close the new table without saving.

### 4.1 Creating Tables

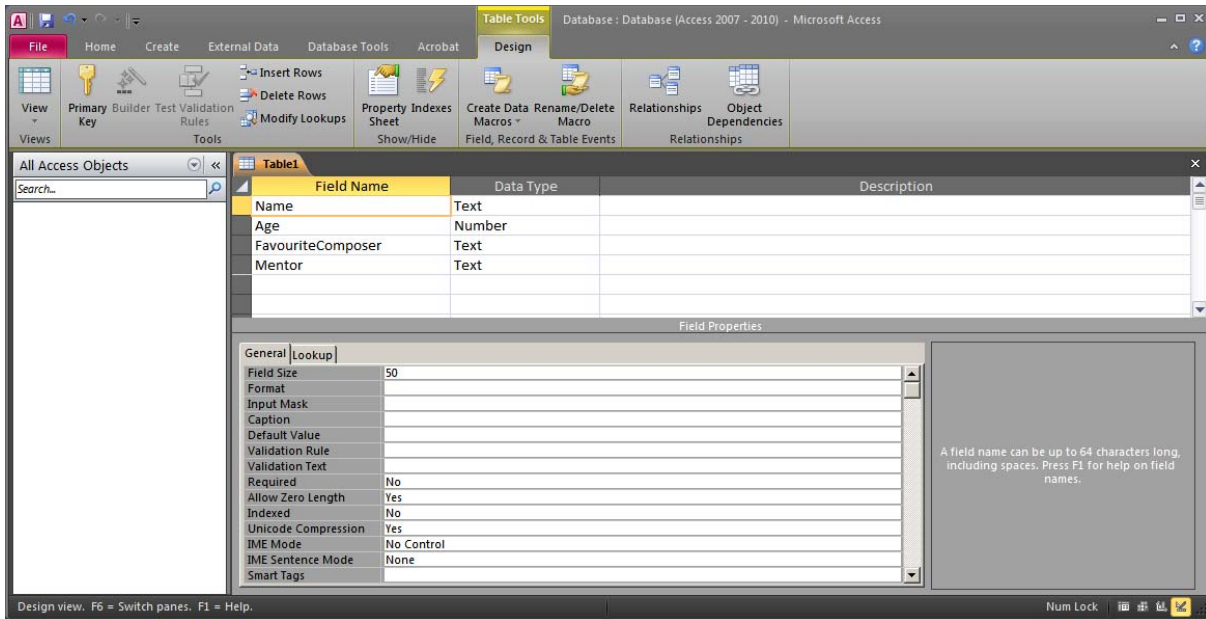
There are many ways to create tables in Access, but we will only look at two of them. The first way is using a special design view where we can specify the names of the columns, the data types, primary keys and other field restrictions (for example NOT NULL, alternate keys, unique fields, etc.). The second way that we will look at, is specifying a DDL statement (i.e. a CREATE TABLE statement) for each table and then simply running the DDL statements.

#### 4.1.1 Design

To create a table in design view, select Create > Table Design from the menu. This will open a new tab in the main area of the Access window:



In this window we can specify the structure of a new table. We start by specifying the columns of the table Artist. The columns that we have in this table are Name, Age, FavouriteComposer, and Mentor. They are all strings except from the column Age. So we can fill in the fields in the table design tab:



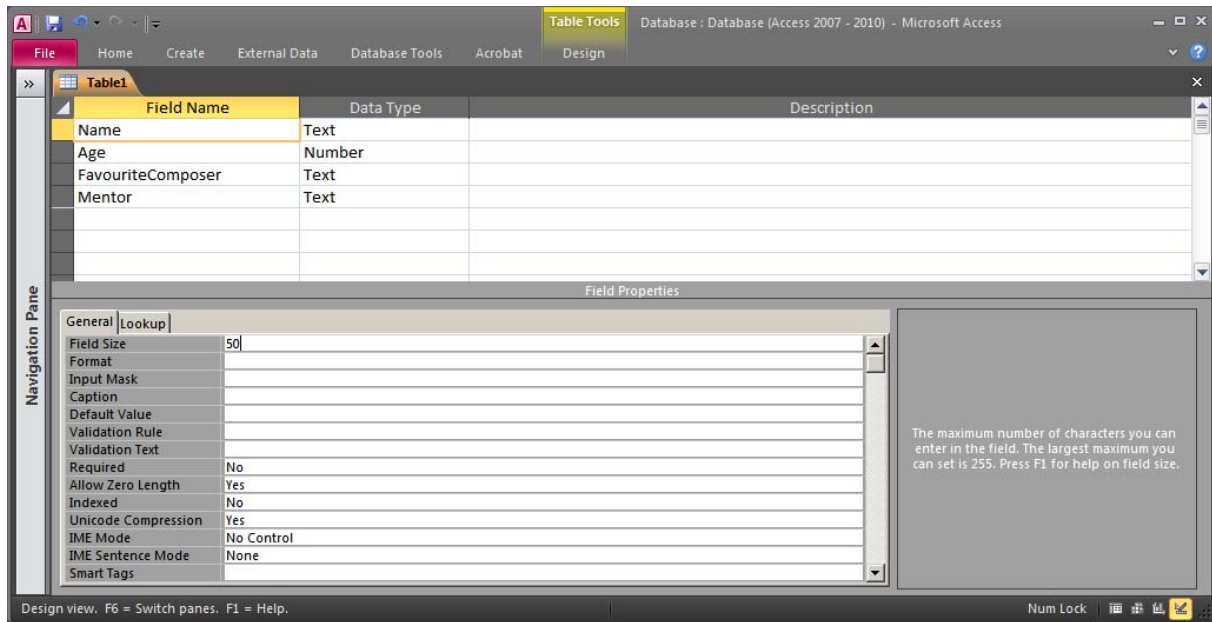
The word "Name" is a reserved word in Access, so a warning may appear when naming a column "Name". We can ignore the warning and continue. When using column names that are reserved words, we may have to enclose the column names within “[“ and “]” in SQL and in other contexts where Access may otherwise get confused.

Access has a data type called Text, which is equivalent to String, and a data type Number, which can be used as an integer.

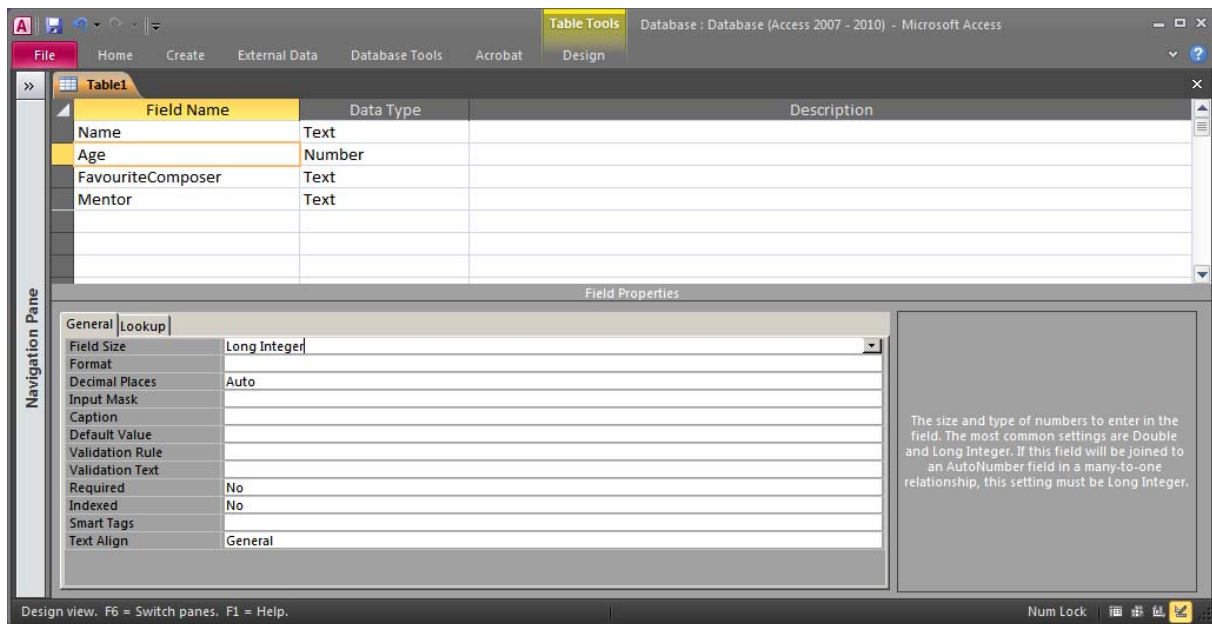
For each column (called “field” in Access) of the table we can specify more details. The Field Properties shown in the lower half of the tab belong to the field that is selected in the upper half. The active field is highlighted with a different color. So in the image above the field properties shown apply to the field Name.

The available field properties depend on the data type of the selected field. For example a Text field has a property Field Size. If you want to see help on a particular field property, simply place the cursor on that field property. The text to the right will give you a short explanation of that field property. In the image below, the cursor has been placed in the first field property (Field Size) (and the ribbon and the object browser have been minimized):

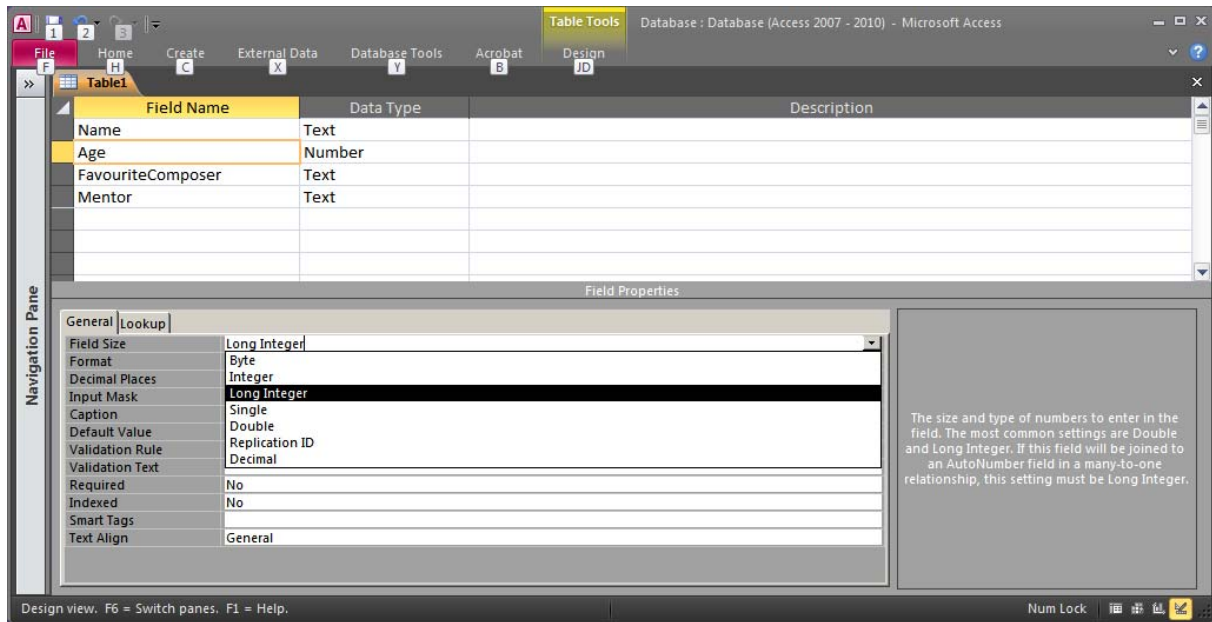




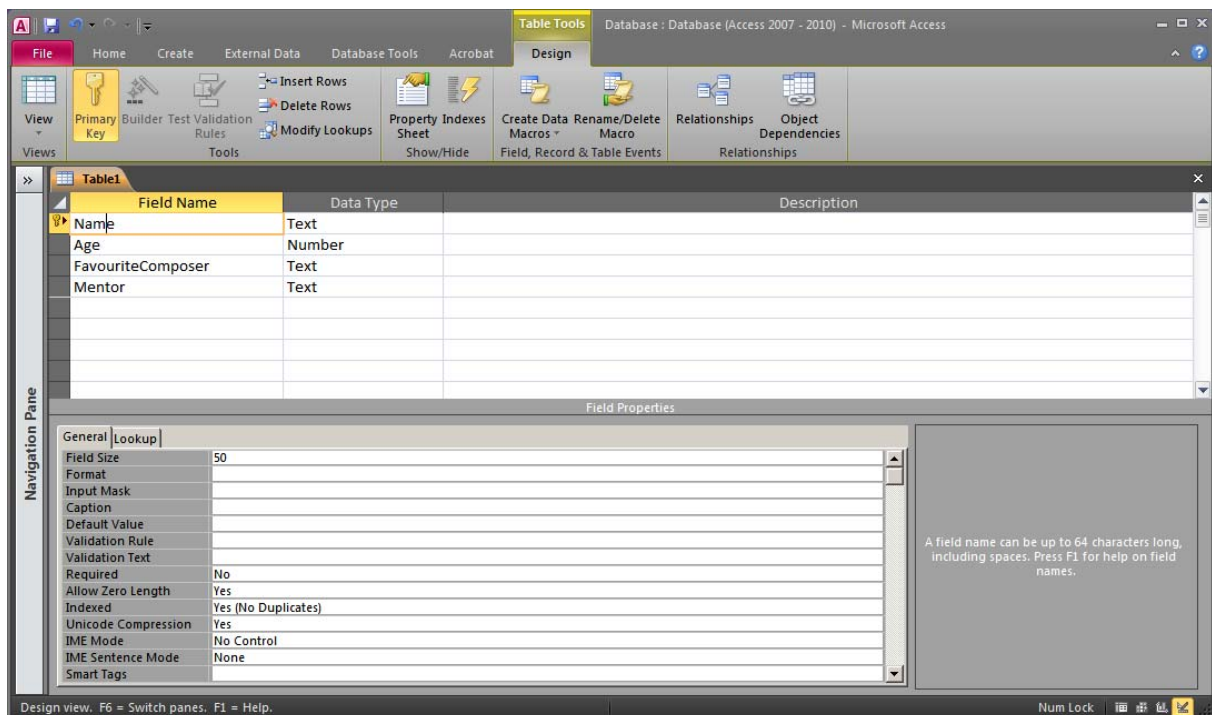
A field of data type Number has different properties:



A Number field has also a property Field Size, but here we can only select one of the available choices:

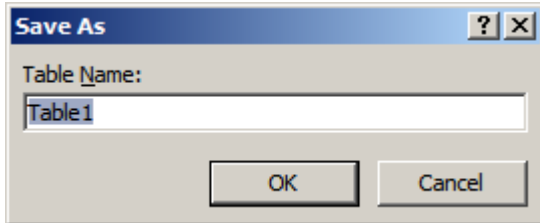


Now that we have specified all the columns of the table, we can also specify the primary key for the table. We can simply select the field that we want to use as primary key and press the Primary Key button on the ribbon (under Design). The selected field will then be marked with a key symbol (on the left of the field name) to indicate that it has been selected as primary key:



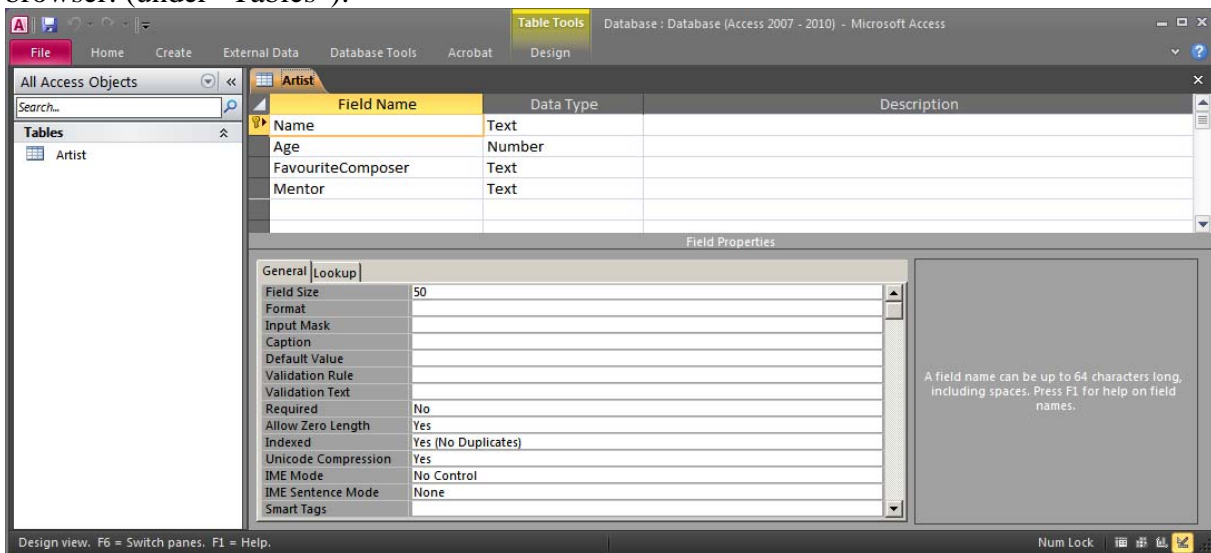
When specifying each field's data type, we can also specify whether this field should accept NULL. By setting the property Required to Yes, we specify the field as NOT NULL. The column Name is set as primary key, so it is implicitly NOT NULL. The column Age on the other hand must be explicitly set as Required. The remaining columns should allow NULL according to our model.

We can specify the table name now by either trying to save the table or trying to close the table design. To save the table press Ctrl-S or select File > Save from the menu, or right-click on the table tab selector and select Save. Access will ask you to specify the name of the new table:

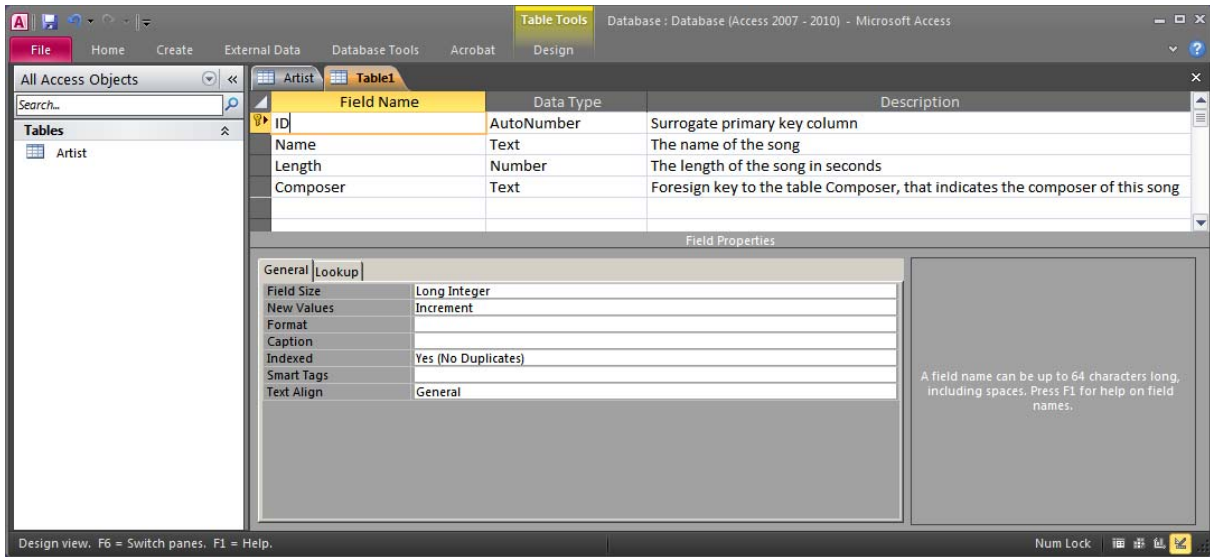


Specify the table name (Artist) and press OK.

The table name is now visible in the tab selector and the table has appeared in the object browser: (under “Tables”):

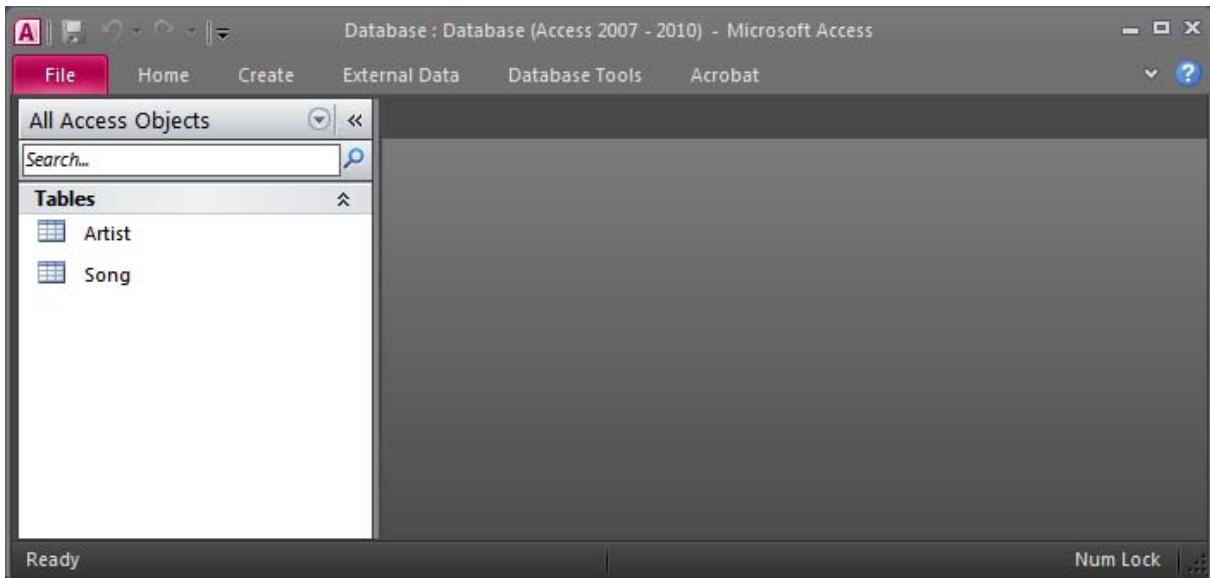


Now, let's create the table Song. This table has a surrogate primary key called ID. We decided previously to let this be an integer field. Access provides a data type AutoNumber, which can be quite useful in this case. An AutoNumber column is managed by Access. Whenever a new row is added, Access calculates a new unique value for this column. An AutoNumber is actually a Long Integer so this is important to remember when we later create foreign key columns that must refer to AutoNumbers. We can create the table Song according to the following screenshot:



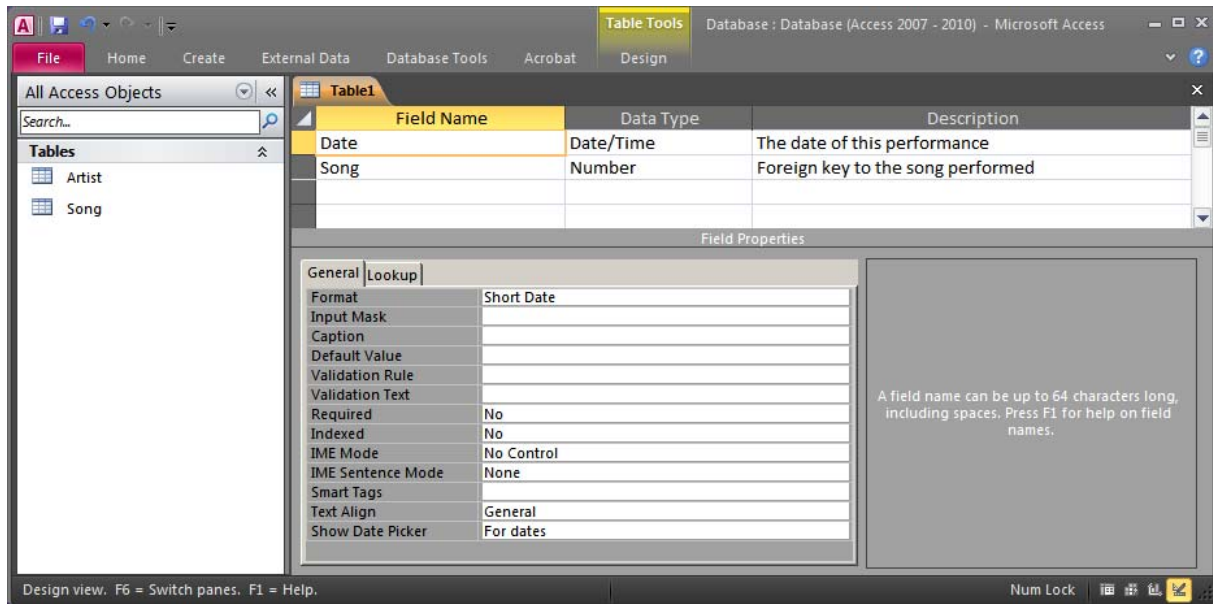
As you can see in the image above, it is also possible to write comments for each field. This can be useful, especially when the name of a field is not very intuitive, but generally, we should strive to have informative field names. When you have defined all four columns, their data types and the primary key, you can save the table and close the table design window.

We have now two tables in our database window:

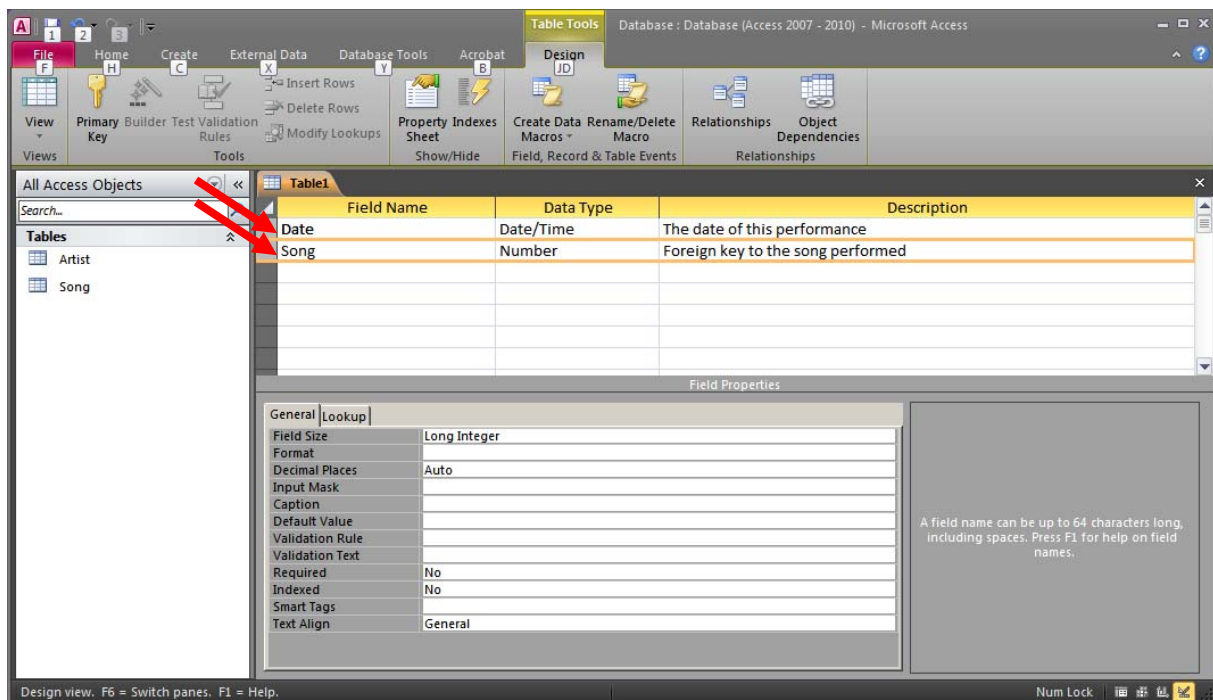


Now let's create a table with a composite primary key. SongPerformance is such a table.

Once again we start by selecting "Create" and "Table Design". We define the columns of our table and their data types as we did before. The column Date can be defined to be of data type Date/Time. We can then define in the field property "Format" that this field should be a "Short Date". We should now have the following:

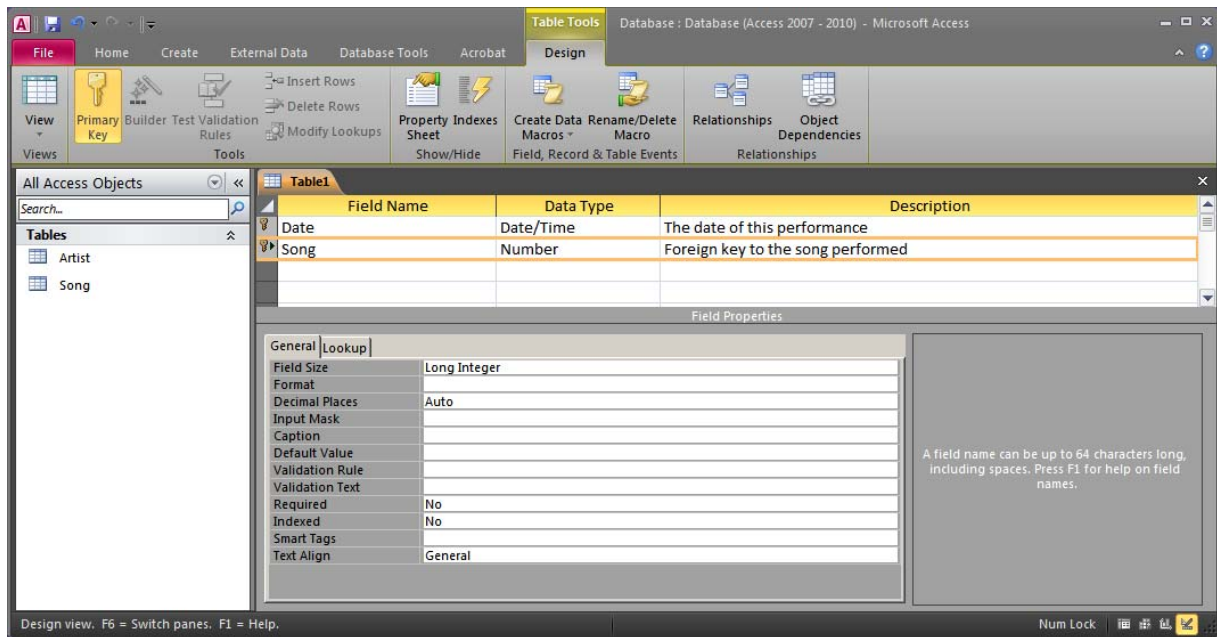


The only thing missing now is the primary key. In order to define a composite primary key, we must select all the primary key columns and then press the primary key button on the ribbon. We can do this by holding down the Ctrl-key and clicking on the square on the left of the relevant fields.



When all primary key columns have been highlighted, we can press the primary key button to indicate that all these columns together constitute the table's primary key. A primary key symbol will be shown next to each of the fields:





#### 4.1.2 DDL

Another way to create a table in Access is by writing a CREATE TABLE statement and then executing it.

We can for example create the table CD with the following statement:

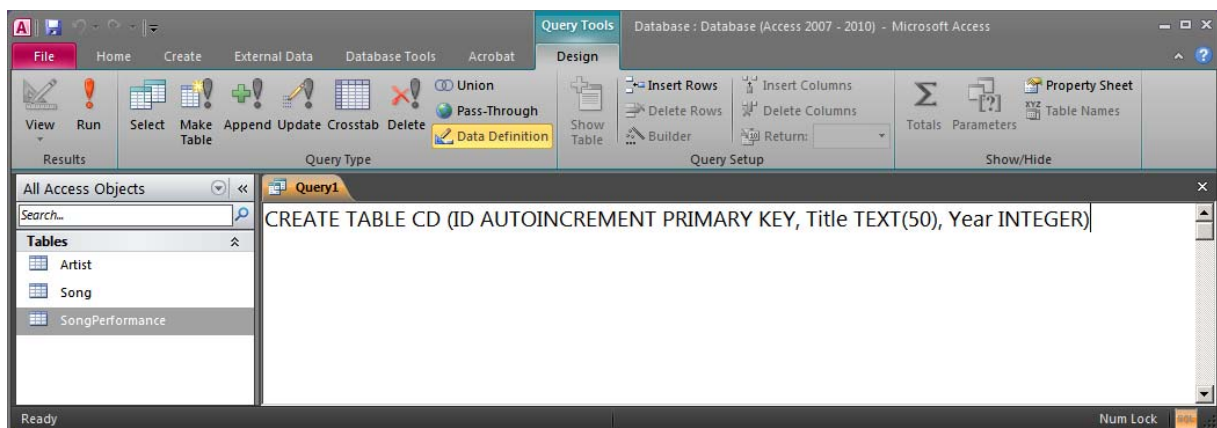
```
CREATE TABLE CD (ID AUTOINCREMENT PRIMARY KEY, Title TEXT(50), Year INTEGER)
```


AUTOINCREMENT is equivalent to the AutoNumber that we used in the design view in the previous section.

TEXT(50) is a text field that is up to 50 characters long.

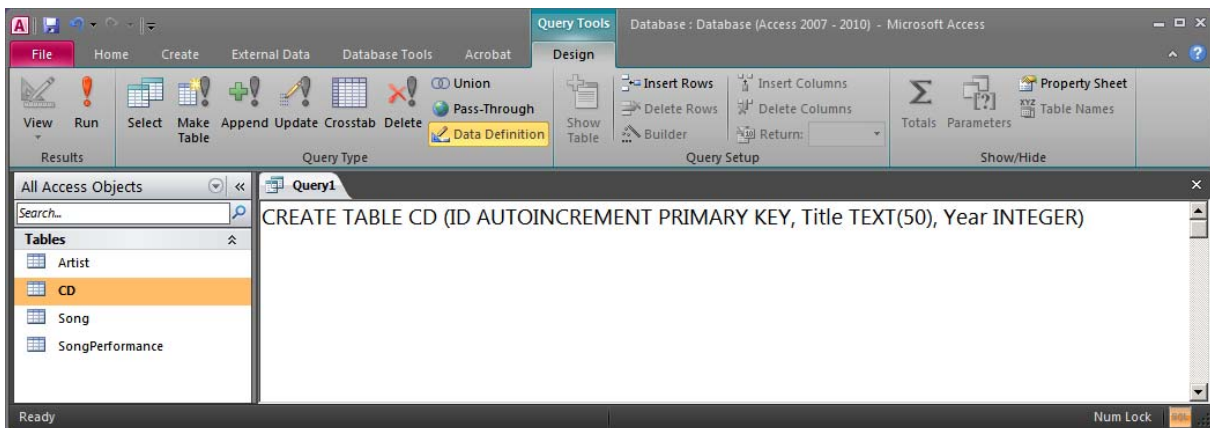
INTEGER is equivalent to a Number field of type Long Integer.

The only thing we need now is a Query object which we can use to execute SQL (as we saw in section 3.2). Once we have a Query object in SQL view we can write our CREATE TABLE statement:

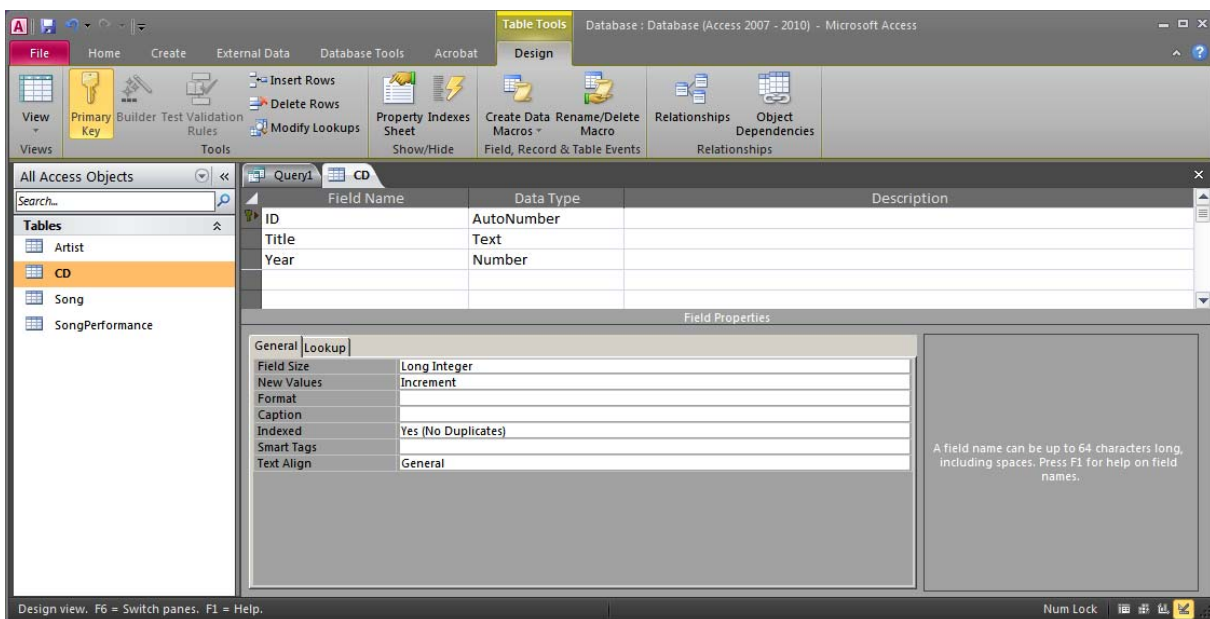


You can now either execute the statement directly by pressing the Run button (  ) on the toolbar, or save the statement as a query object in order to execute it later. Press the Run

button to execute the statement. A new table (CD) will immediately appear in the object browser:



By right-clicking on the new table and selecting "Design View", we can examine the table and possibly make changes:

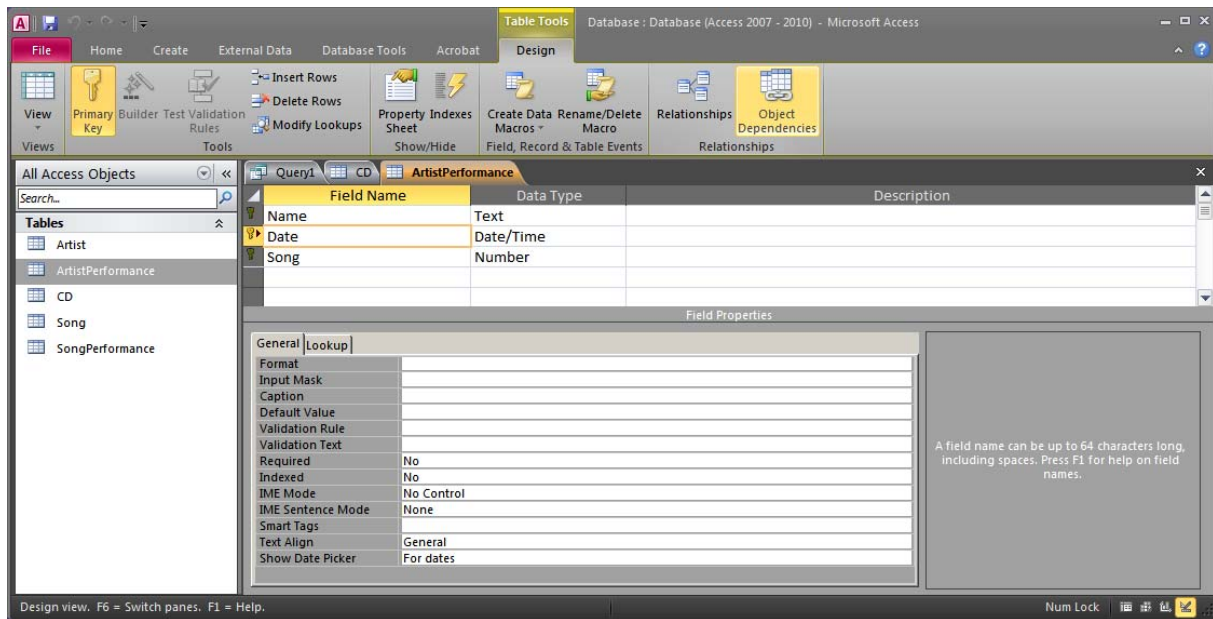


By repeating the same steps we can create the table ArtistPerformance with the following SQL statement:

```
CREATE TABLE ArtistPerformance (Name TEXT(50), [Date] DATE, Song INTEGER, PRIMARY KEY (Name, [Date], Song))
```

Observe that the word "Date" is a reserved word in Access. In order to indicate that we want to have a column with that name, we must enclose the column name within "[ " and " ]".

After running this CREATE TABLE statement, we can examine our new table in the design view:

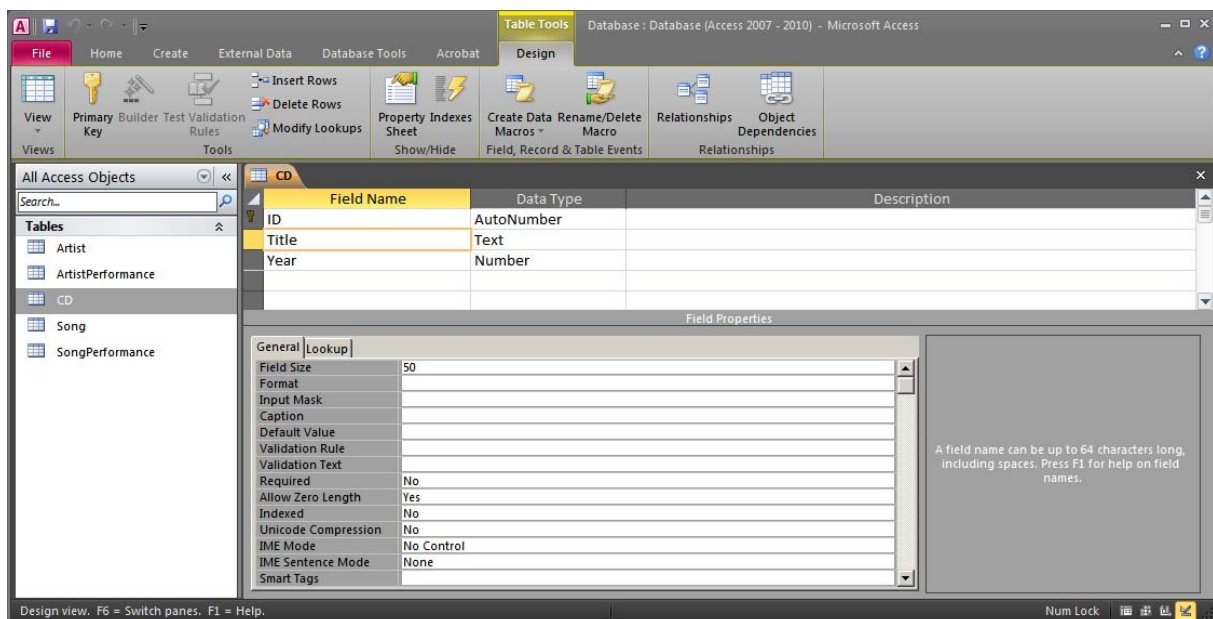


We may for example choose "Short Date" as the format of the column Date, since this was not specified in the CREATE TABLE statement.

### 4.1.3 Defining Other Restrictions

Just defining the columns of a table and its primary key is most of the times not enough. There are often other restrictions that have to be specified. For example we may want to define that the columns Title and Year in the table CD should not be left empty (cannot contain the value null). We may even want to restrict the value of the column Year to a specific interval, for example between 1980 and 2050. Simple rules like these can be defined in the table design view. Let's fix the table CD to include the restrictions mentioned above.

Open the table CD in the table design view. Now activate the field Title to show its field properties:

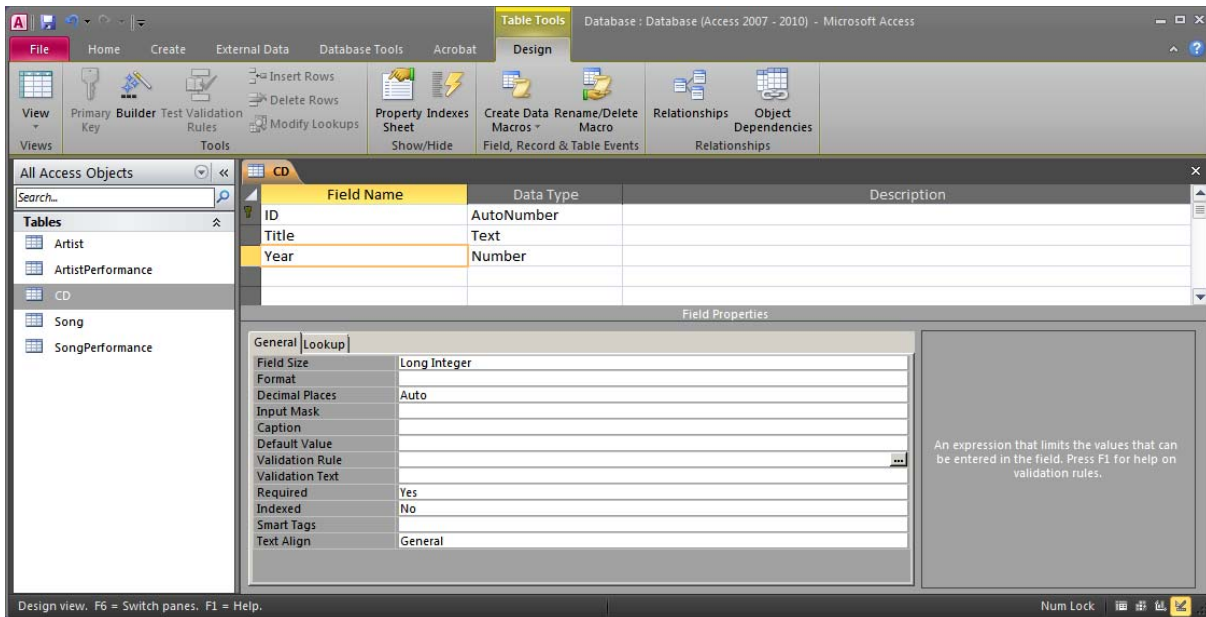


As we said earlier, we want to make sure that there is always a value in this column. The property Required can take care of that. Change the property value to Yes.

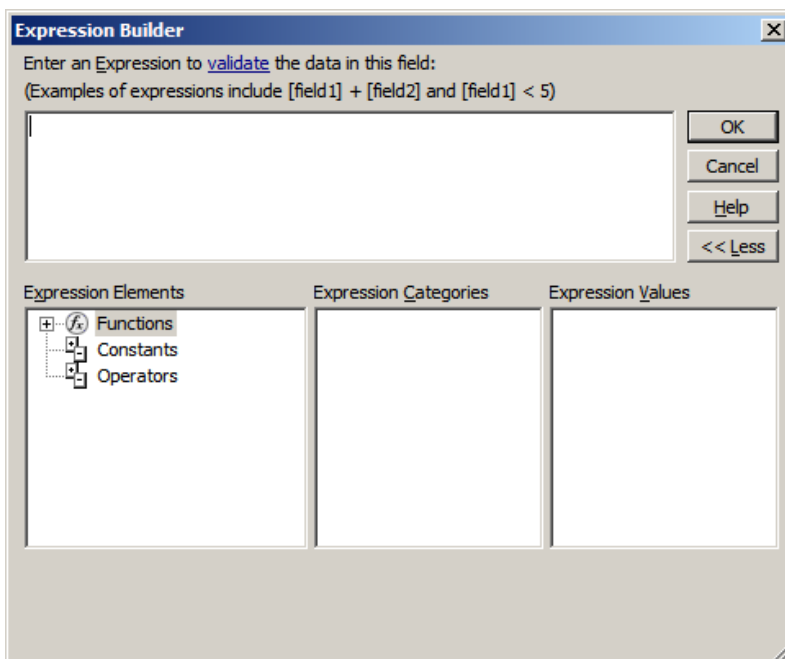


- ① You can shift property values quickly by simply double clicking on them. Double click on Yes to turn it into a No, and vice versa.

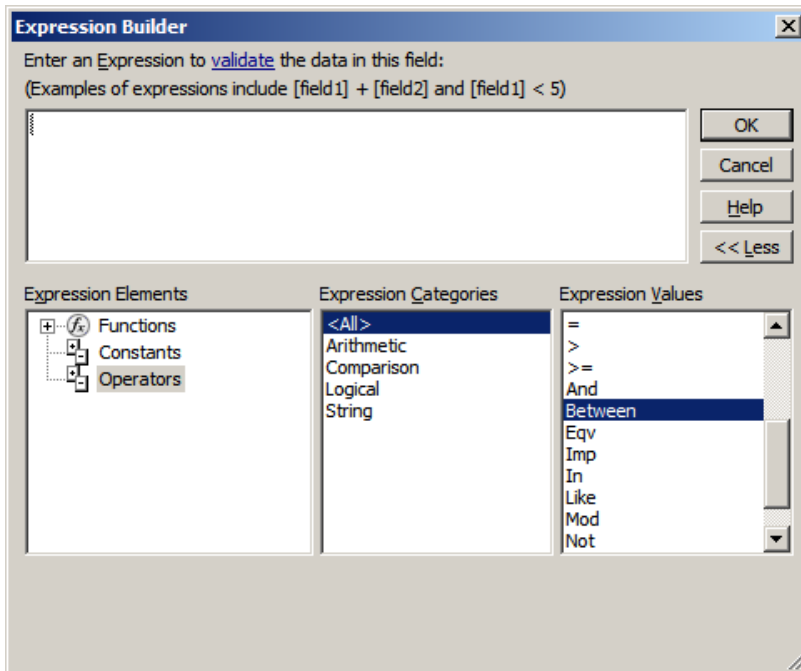
Now activate the field Year and do the same as for Title, i.e. set its Required property to Yes. For the column Year we also want to restrict the possible values. For this we can use the property Validation Rule. Activate the property (by placing the cursor there) and then you will see a little button on the right side of the property:



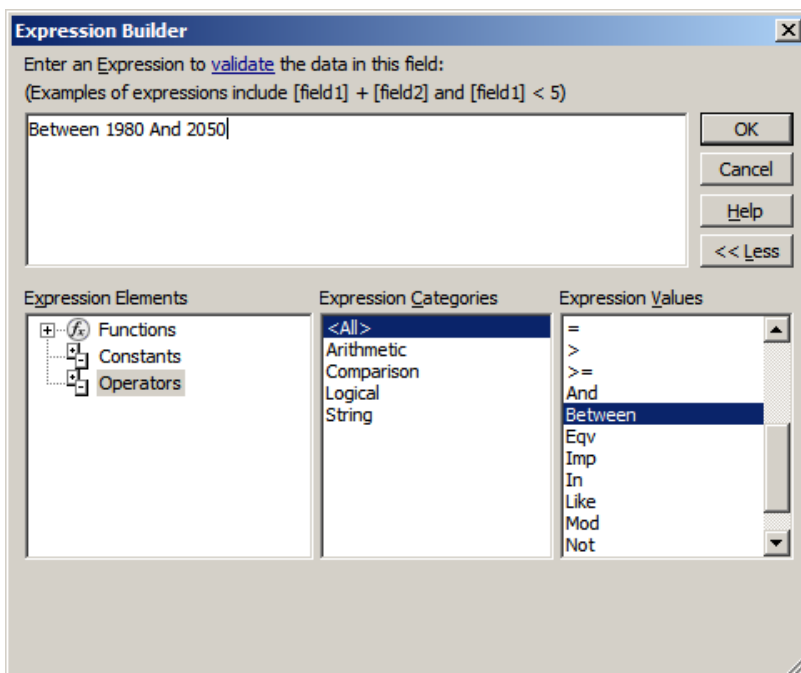
Press it and a new window will appear. This is the Expression Builder that allows us to create small logical/mathematical formulas:



Under Operators we can find what we need, namely Between:

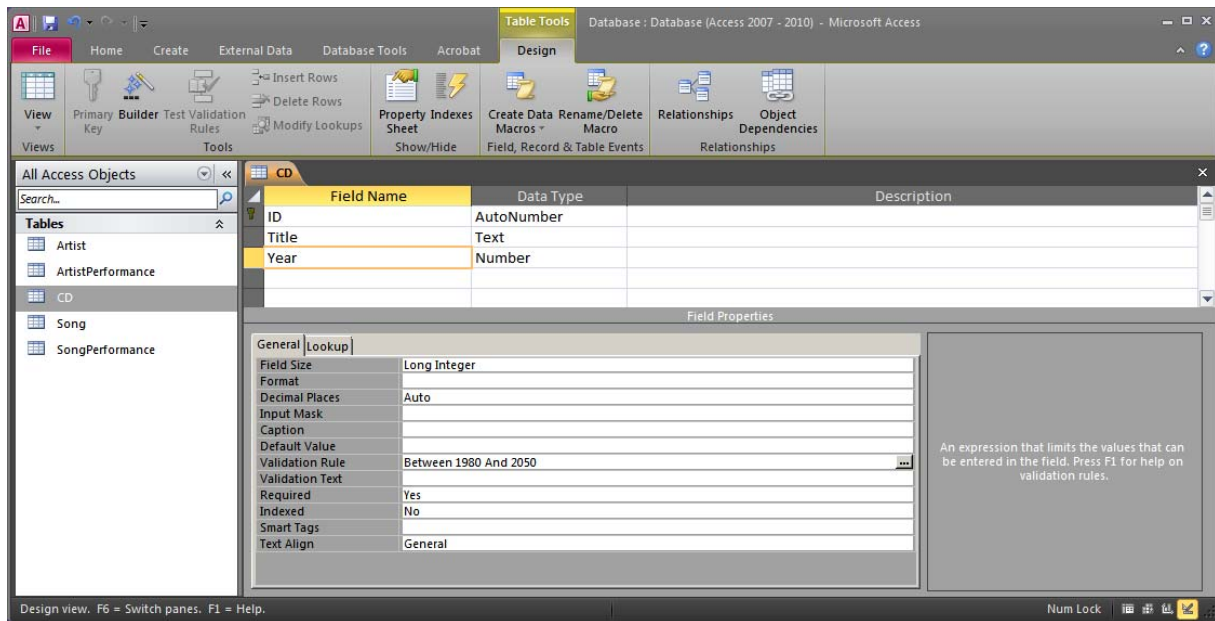


Double click on Between to add it into your formula and then substitute the two «Expr» with the appropriate values:



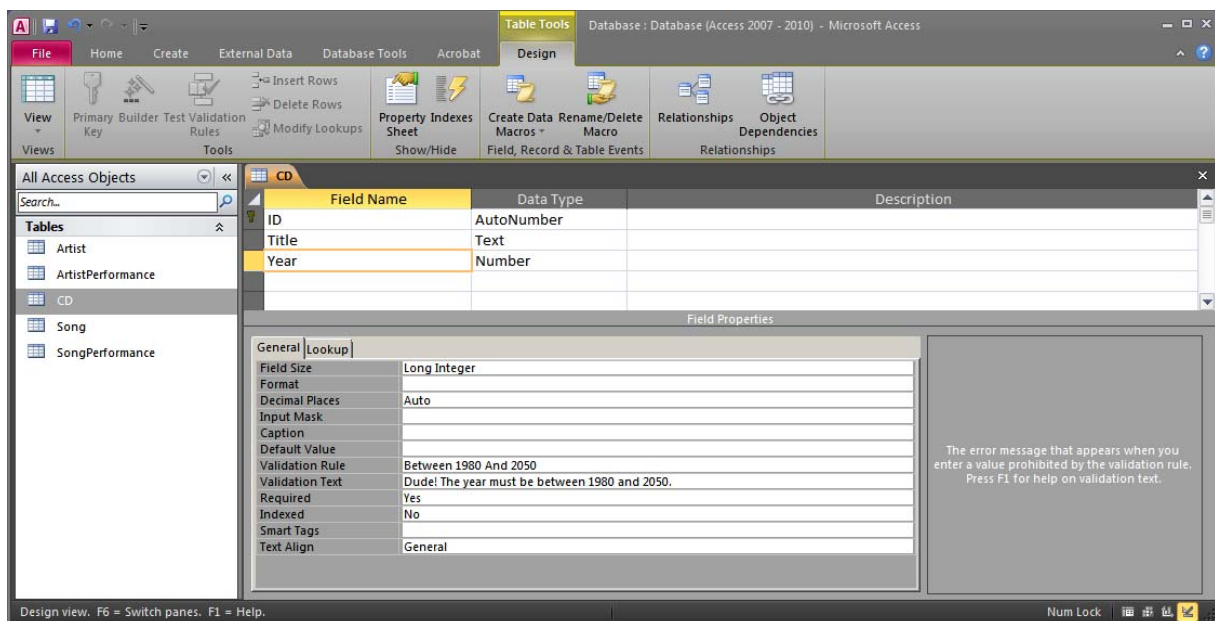
① It is of course possible to write your expression directly, without having to find the keywords in the menus.

Now press OK and the newly created expression will appear as the field property value:



- ① It is of course also possible to write the expression directly in the field property without opening the Expression Builder

When a user enters a value, Access will always check if it is correct in respect to the validation rule. If the value breaks the rule, then an error message will be signaled to the user. Access has a default message, but also allows us to specify what message should be displayed. We can specify a better message in the property Validation Text:



You can now create the rest of the tables with any of the two techniques described in this chapter. You can also add the appropriate restrictions on columns of all the tables (like specifying that Song.Length must be greater than zero and that Artist.Age must be between 1 and 200). In order to create composite alternate keys (two or more columns that together don't allow duplicates) you will need to create an index as described in section 9.9. The next section assumes that all the tables have been created. A ready database with all the tables created exists at <http://coursematerial.nikosdimitrakas.com/access/>.

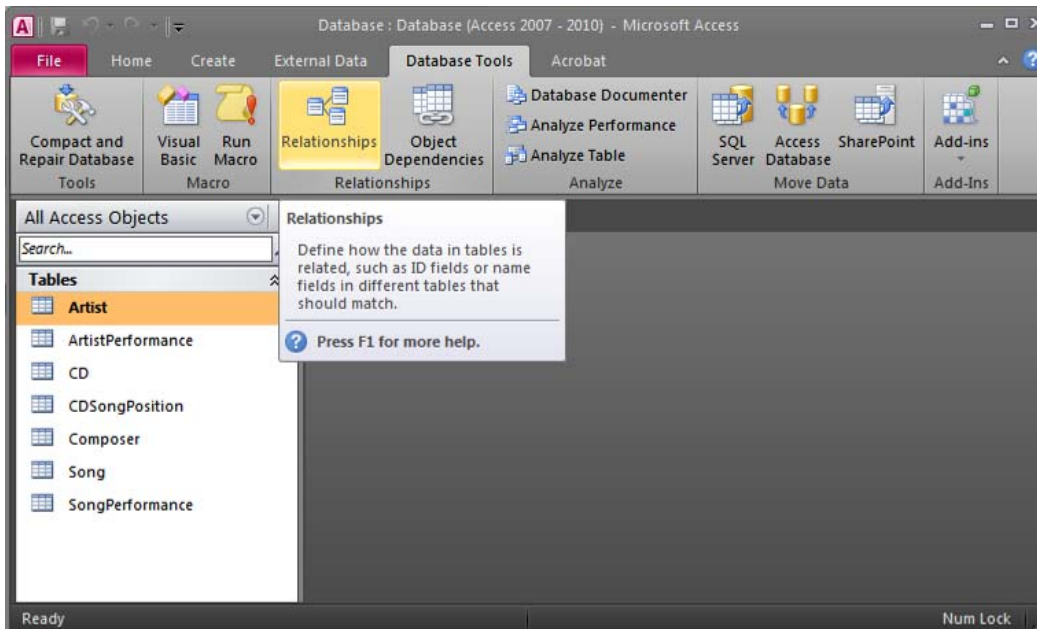
If you choose not to create the tables yourself, then download the database that contains them before continuing to the next section.

## 4.2 Working With Relationships

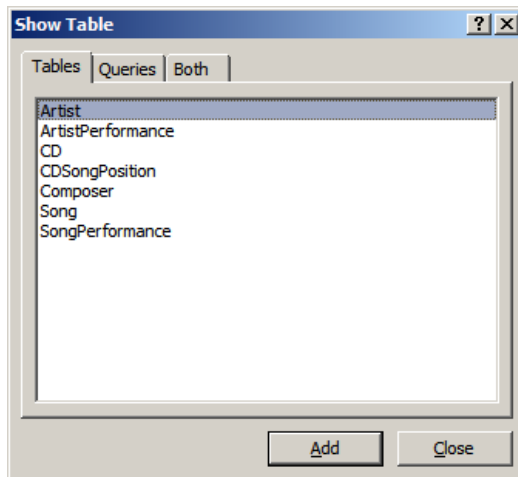
All the tables are now created, but they are not in any way related to each other. There is no referential integrity specified, and no foreign key rules either.

In Access, relationships between tables can be created graphically with a simple drag-and-drop principle. Relationships in Access are more than just foreign key relationships. They are also used by the system when working with wizards for creating forms and reports; more about this in later chapters. In this section we will focus on defining referential integrity and referential actions. The five subsections that follow describe the most common types of relationships that we can have in a relational database.

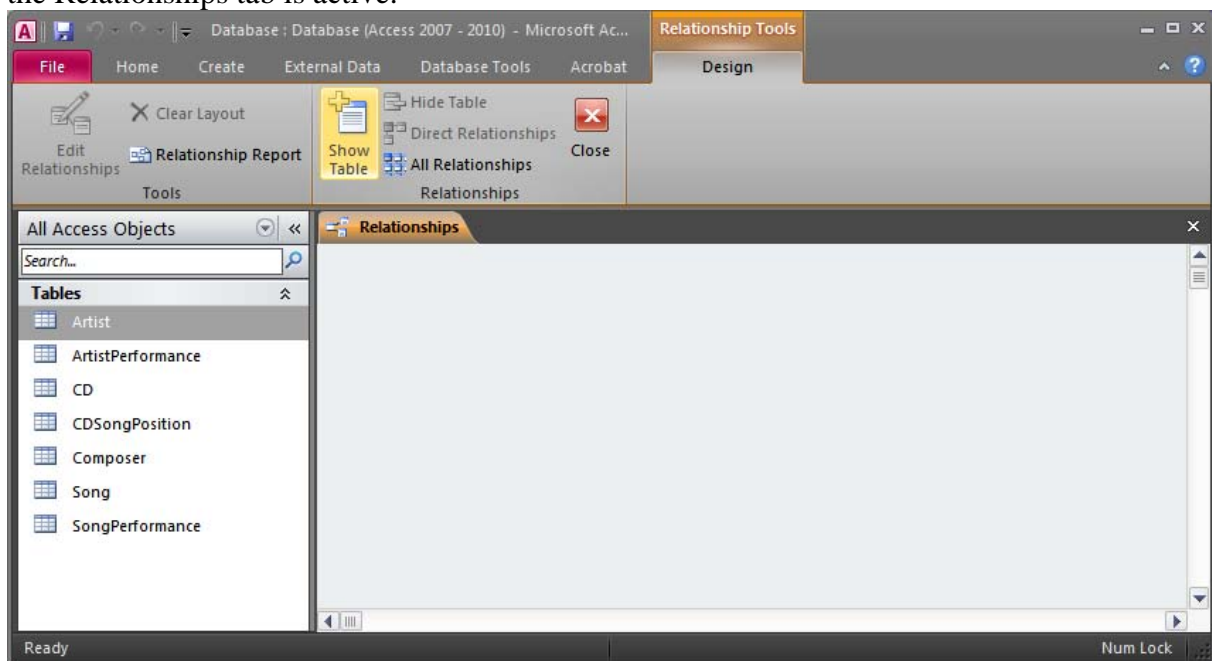
In order to create table relationships in Access we must use the Relationships window. This can be invoked from the ribbon under "Database Tools":



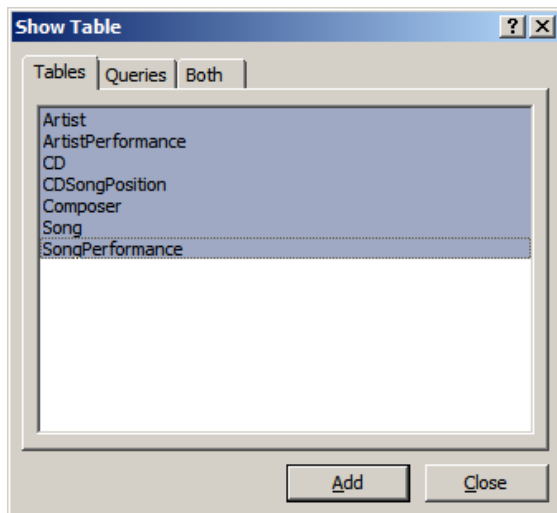
When you open the Relationships button for the first time, Access will automatically open the Relationships tab and show the Show Table dialog:



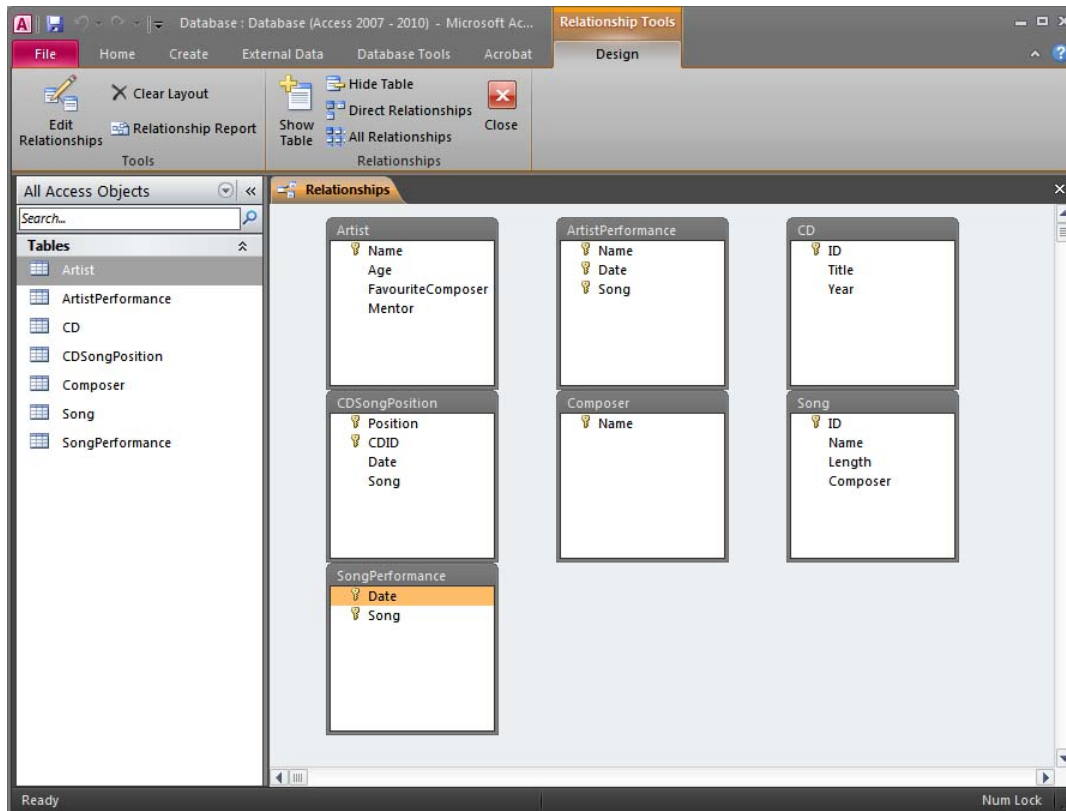
This dialog can also be invoked later by pressing the Show Table button on the ribbon while the Relationships tab is active:



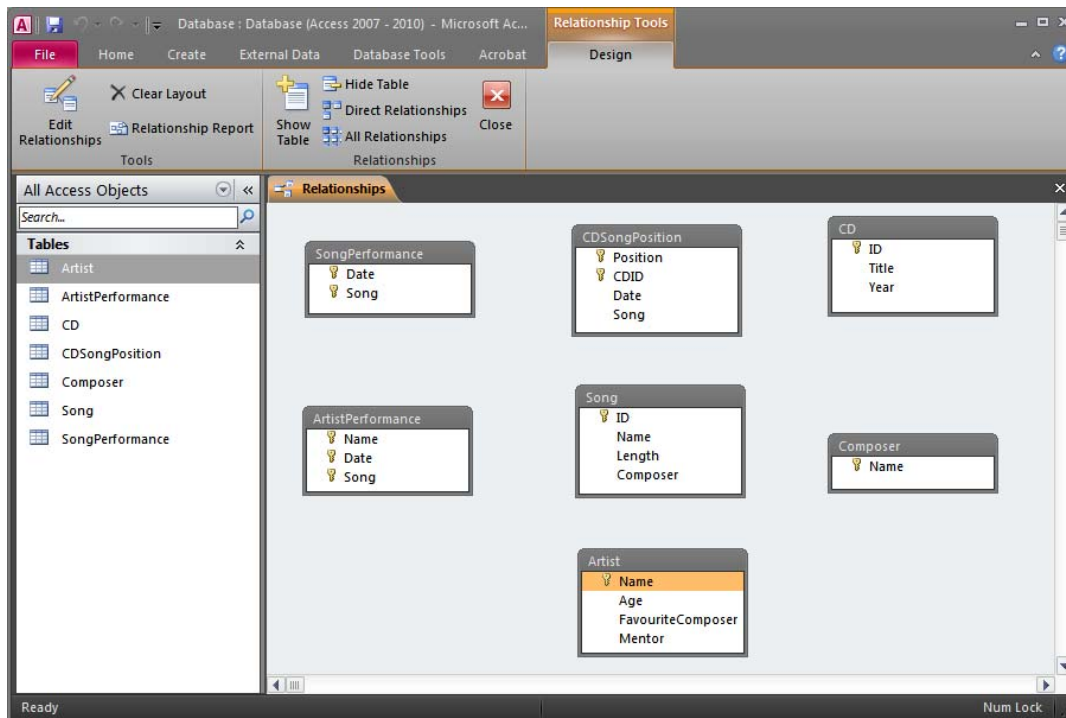
This dialog allows us to choose which tables to show on the Relationships tab. The relationships view in Access is like a diagram that shows graphically all the relationships between the different tables; not unlike the graphical relational schema shown in chapter 2. Highlight the first table in the list, hold down Shift and click on the last table name on the list to highlight them all:



Press Add to add them to the diagram and press Close to return to the relationships diagram:



We can now resize and move the tables as needed. We can for example put them in the same way as they were in the graphical database schema in chapter 2:



- ① Observe that the tables shown here are just graphical representations of the tables in our database. Deleting a table here does not delete the table from the database, just from the relationships diagram.

We are now ready to create relationships.

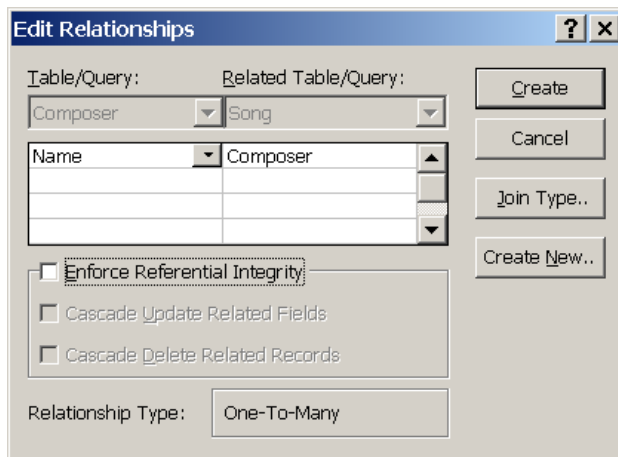
#### 4.2.1 Simple Foreign Keys

The first type of relationship, and probably the most common one, is when we have one column in one table that is a foreign key to another table (with a one-column primary key<sup>1</sup>). For example the column Composer in the table Song is a foreign key to the primary key of the table Composer. In order to define this relationship, we have to select the primary key column and drag and drop it on the foreign key column<sup>2</sup>. So, just drag the column Name of the table Composer to the column Composer of the table Song. The Edit Relationships dialog will then appear:

<sup>1</sup> Or alternate key

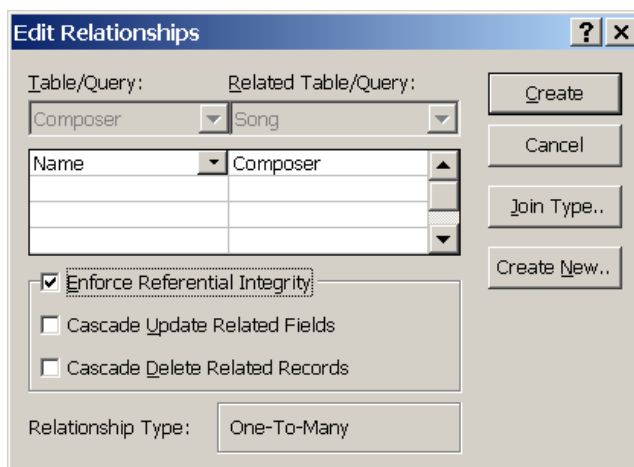
<sup>2</sup> Dragging and dropping the foreign key onto the primary key will also have the same effect if the multiplicity is one-to-many.





This dialog suggests a relationship based on the columns that we dragged and dropped. The foreign key is automatically placed on the right side and the referenced primary key on the left side. If we want the database to check that the values of the foreign key exist as values of the referenced primary key, then we should check the Enforce Referential Integrity check box. There are some rare cases that we would not want referential integrity, but in this compendium we will always have referential integrity for all relationships we create. Any relationship created without referential integrity would not formally be a foreign key relationship.

When we activate the referential integrity, a selection of referential actions becomes available:



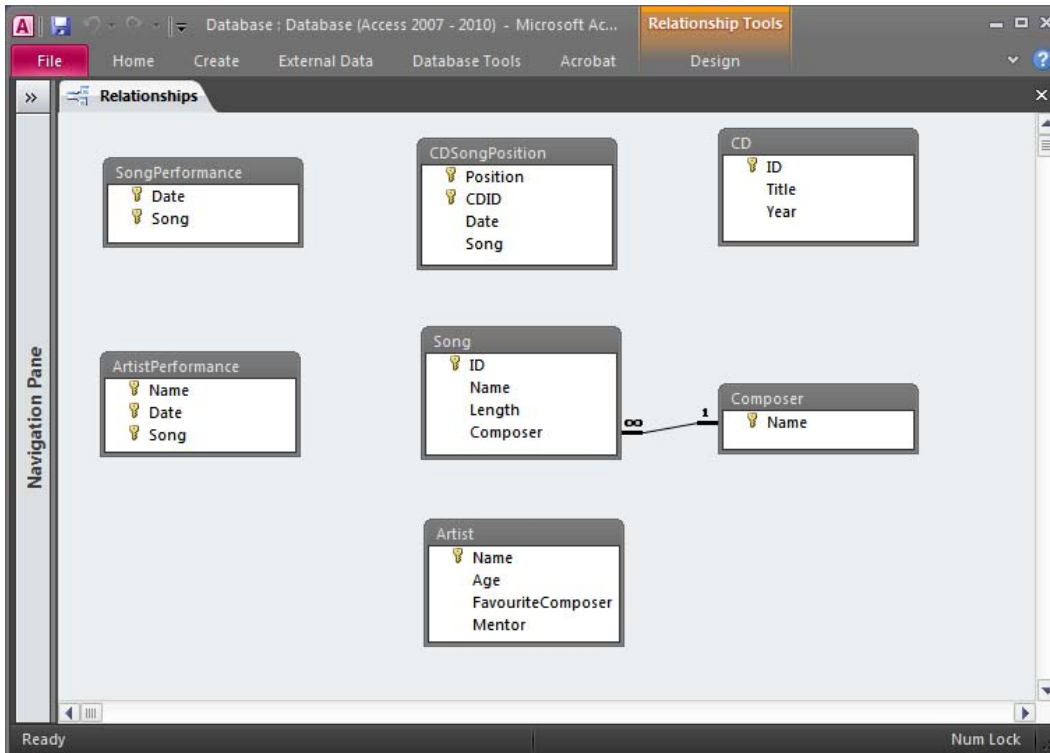
The two choices correspond to “ON UPDATE CASCADE” and “ON DELETE CASCADE” of standard SQL. Setting these options depends on the behavior we want our database to exhibit. In this case we could say that the first one is appropriate while the second one is not. This means that we want the value of the column Song.Composer to be changed whenever the value of the column Composer.Name is changed, but we do not want to delete all the songs of a composer every time a composer is deleted (or there is an attempt to delete a composer). Instead the system will restrict deletion of composers that have composed at least one song. Should we want to remove a composer, we would have to remove all of their songs first.

The Edit Relationships dialog shows us, as a bonus, what the relationship type is for the selected tables and columns. This is derived by the definition of the columns (whether they are candidate keys, unique, etc). If the Relationship type is not as you intended it to be then

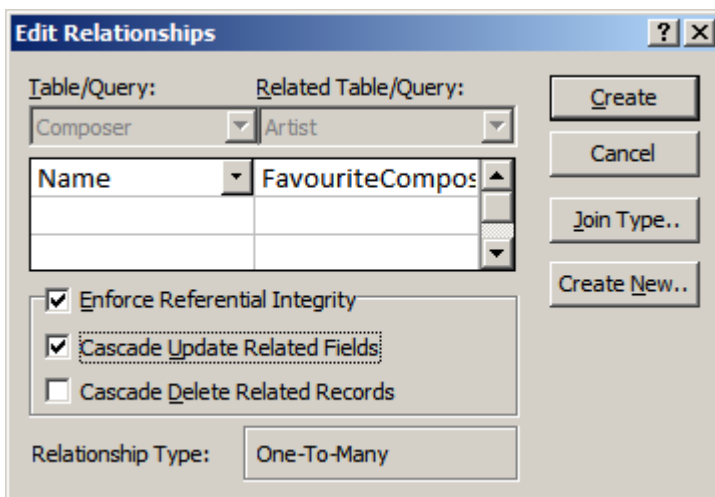


the tables have not been defined correctly. In this case One-To-Many is just fine. One composer can compose many songs and one song has to be composed by one composer.

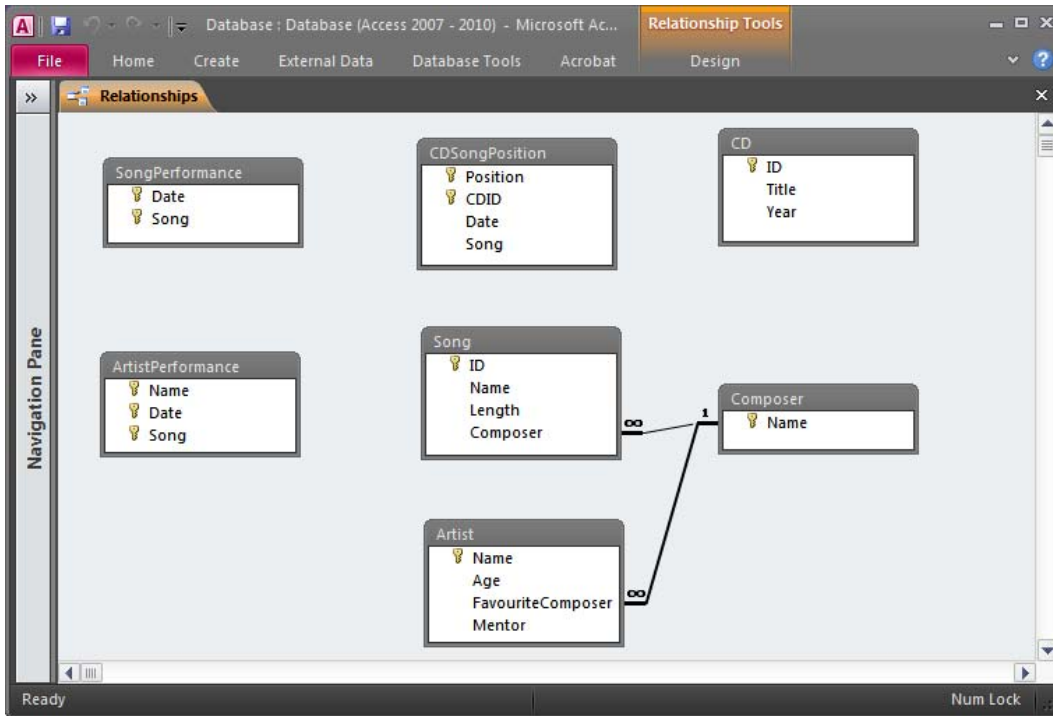
When we are done specifying the relationship, we press Create and then Access will show the new relationship graphically:



The same way, we can create the relationship between Artist and Composer (the artist's favorite composer):



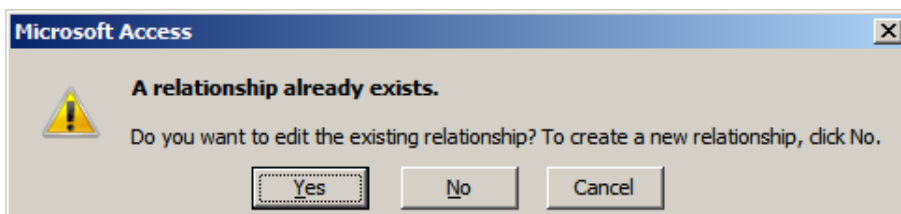
and then:



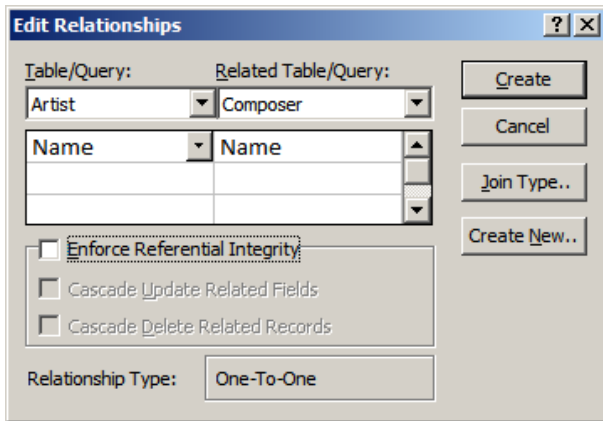
① Should we want to edit an already existing relationship, we can simply double-click on it and the Edit Relationships dialog we show up.

#### 4.2.2 ISA Inheritance

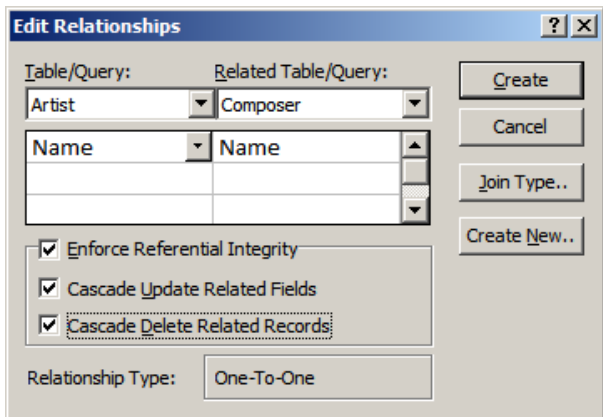
ISA relationships are also quite common. An ISA relationship is not different from any other relationship in relational databases. The implied inheritance is not managed automatically and the relationship is nothing more than a simple One-To-One relationship. When defining an ISA relationship in Access (and any other relational database), it is important to define the relationship in the right direction. In our case we have an ISA relationship that says that a composer is an artist. It would be wrong to create a relationship that says instead, that an artist is a composer, since this would require that all artists must be composers. It is therefore important to create the relationship from the general to the specific, i.e. from the artist to the composer. We can create this relationship by dragging the primary key of the table Artist and dropping it onto the primary key of the table Composer (that also serves as foreign key). Access will then detect that there is already a relationship between these tables and will ask us if we want to edit the old relationship or create a new one:



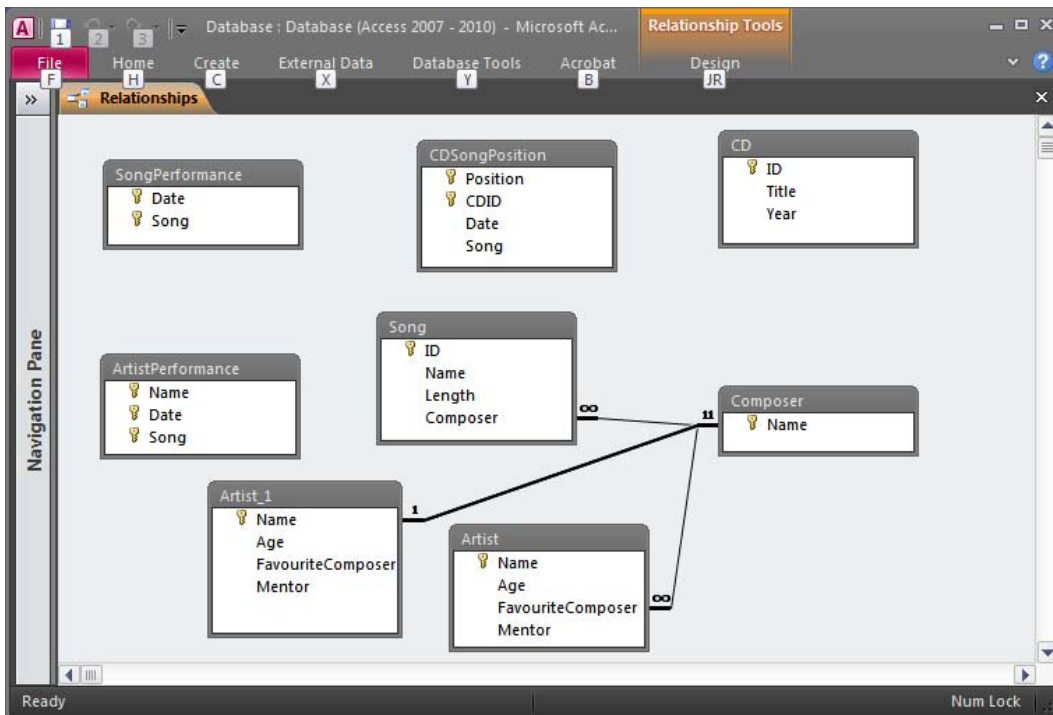
We, of course, want to create a new relationship and answer therefore No. The Edit Relationships dialog will then suggest the following relationship:



We can see that the table Artist is on the left side (at this stage, when creating relationships, always double check that the tables are placed correctly left-to-right), which indicates that it is the master table of this relationship, i.e. a composer cannot be created unless there is a corresponding Artist. The Relationship Type is detected to be One-To-One, which is exactly what we expected it to be. We must not forget to activate the referential integrity and also choose the appropriate referential actions. In this case both of them seem reasonable, so we check both boxes:

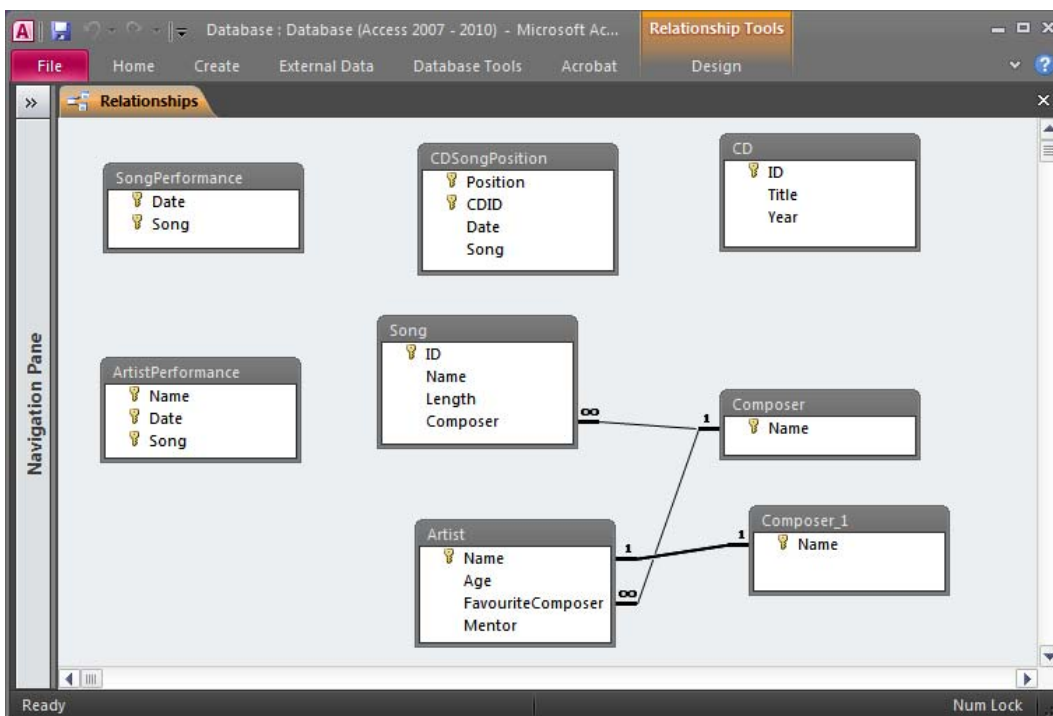


We create the relationship and return to the relationships diagram:

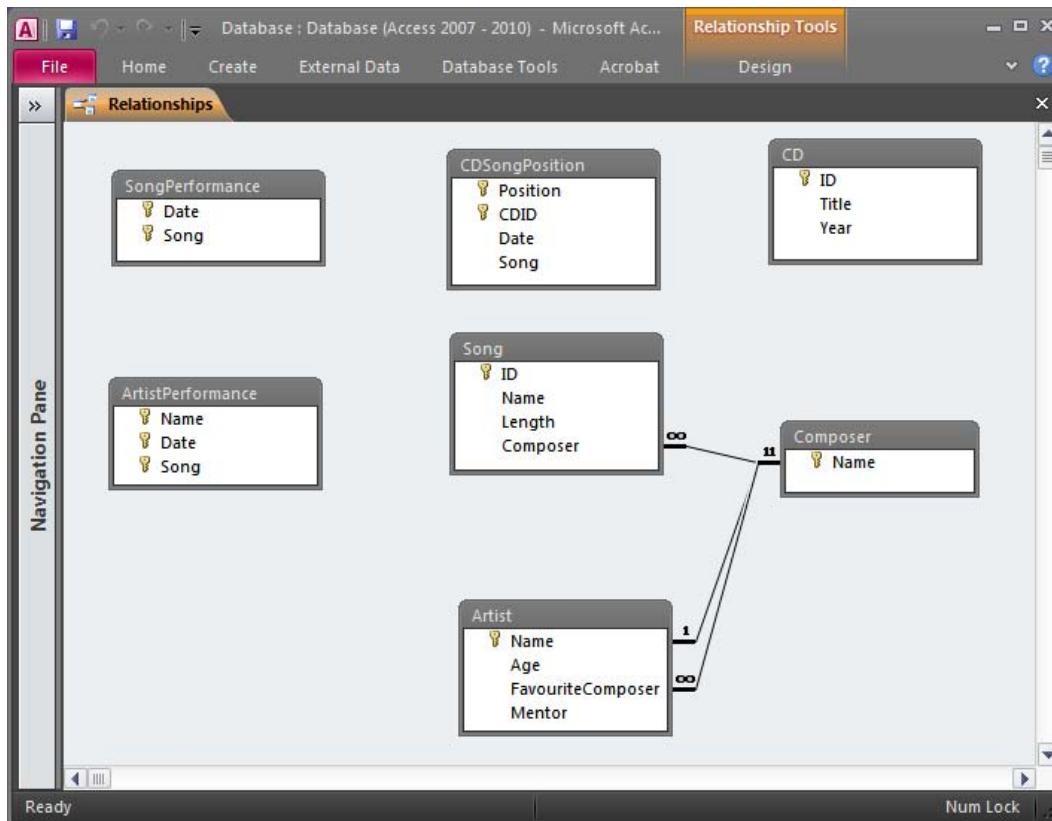


We can now see a strange new table Artist\_1 in our diagram. It is nothing to worry about. This is simply a second graphical representation of the table Artist in the relationships view and not an additional table in the database. Access created this automatically because the tables Artist and Composer already had a relationship. In this way we can distinguish between the two relationships.

It is also possible to create two graphical representations of the table Composer instead. Even though the diagram looks a little different, the relationship is identical to the one we created previously:



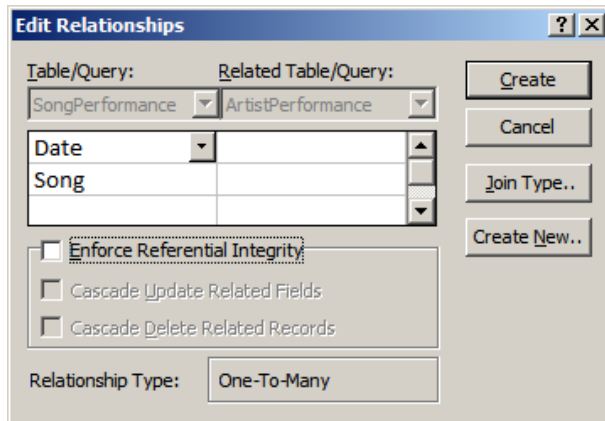
If we don't want to see the same table twice we can simply move the two graphical representations of the same table to the same position (so that the one hides the other):



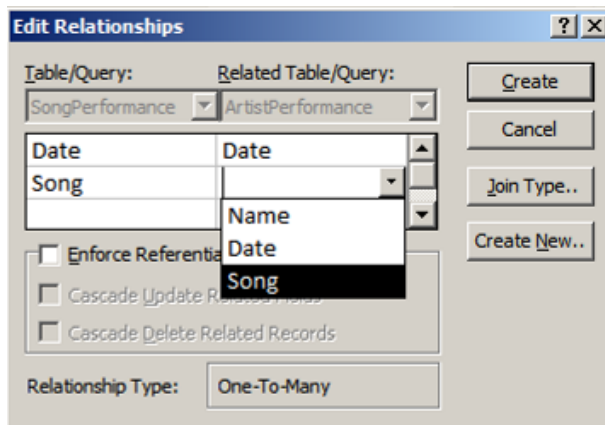
Artist\_1 is now behind Artist, or Composer\_1 is behind Composer.

#### 4.2.3 Composite Foreign Keys

Tables sometimes have composite primary keys (and perhaps also composite alternate keys). When such tables need to be referred to then the foreign key needs also be composite. In our case we have such an example with the table SongPerformance. SongPerformance has a composite primary key (columns Date and Song). The table ArtistPerformance has a foreign key to the table SongPerformance (columns Date and Song). The procedure of creating this relationship is not different than before. Highlight the referenced primary key (use the Control-key to select all the columns of the primary key) and drag and drop it onto the table that contains the foreign key. The Edit Relationship dialog will open, but this time the columns of the related table are not automatically filled in:

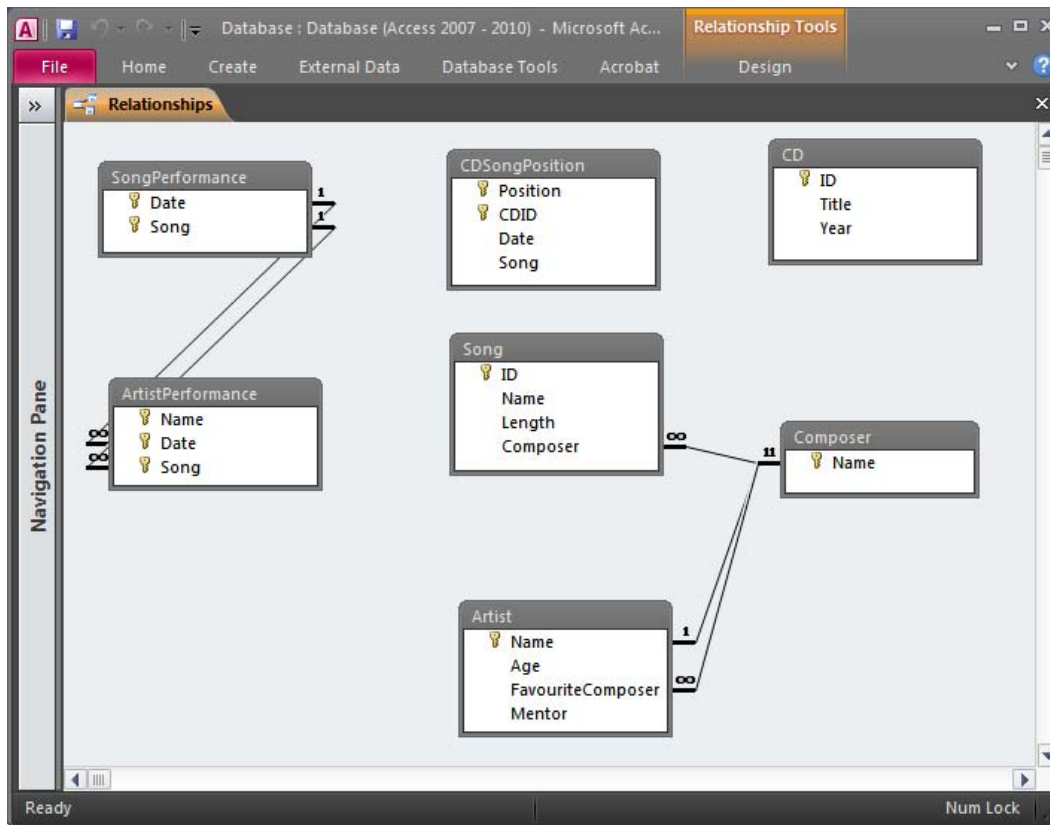


Select the correct columns by using the drop-down list or simply write the column names:



Then activate the referential integrity and create the new relationship.

The relationship is now visible in the relationships diagram:



Since there are two columns that are linked, Access draws two lines for the same relationship. This can be confusing, but both lines together indicate one and the same relationship. If they were two different relationships, then Access would have created a new graphical representation of one of the two tables as it did in the previous section.

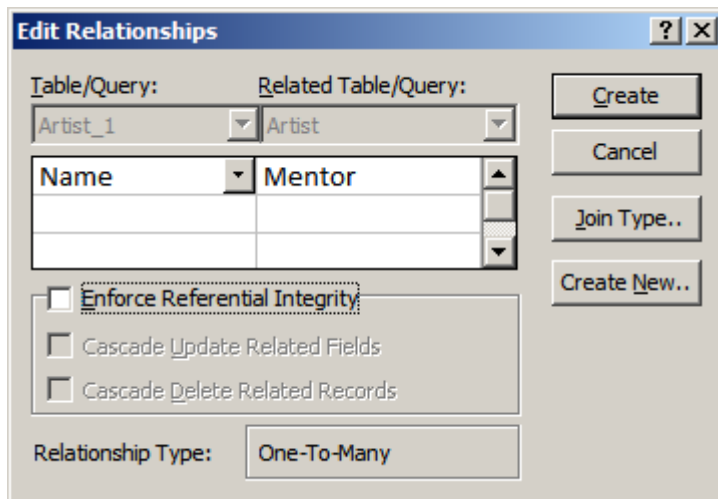
#### 4.2.4 Multiple Relationships Between The Same Two Tables

As we saw in section 4.2.2, when we create more than one relationship between the same two tables, Access will automatically add a new graphical representation of one of the two tables in the relationships diagram. This is to help us distinguish between the two relationships. This is especially useful when we have overlapping foreign keys and generally composite keys.

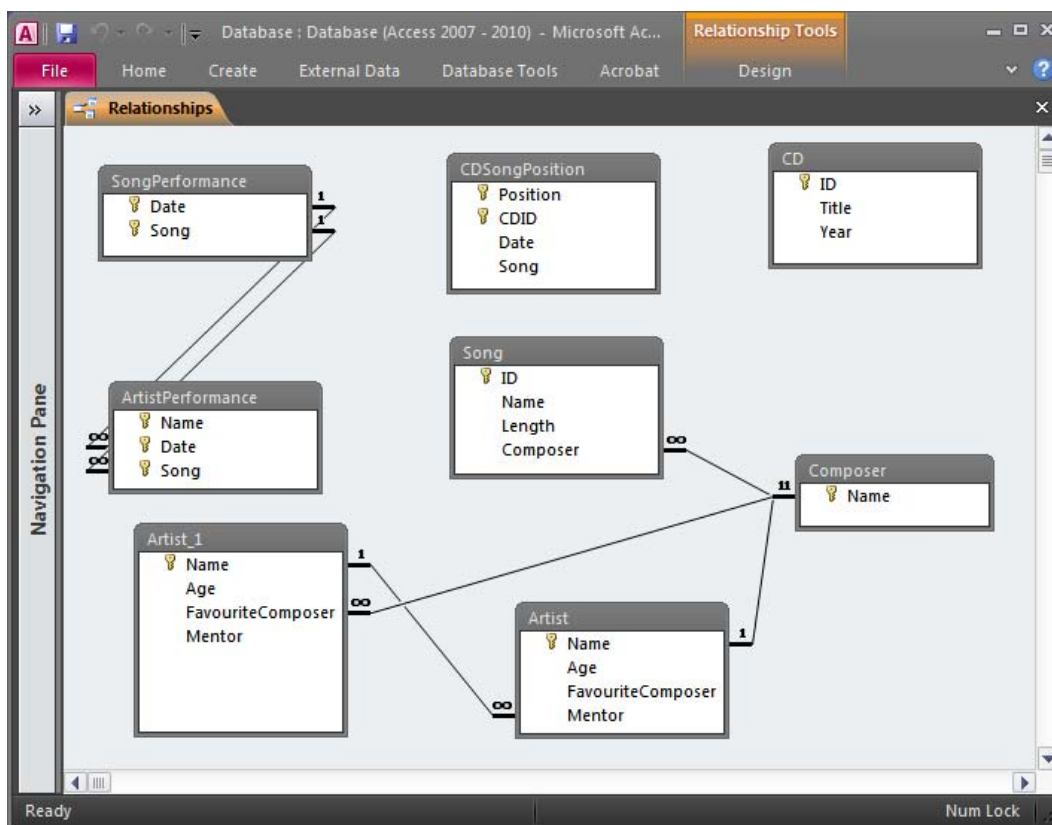
#### 4.2.5 Recursive Relationships

In some cases a table may need to refer to itself. In our case we have the table Artist that has a column Mentor. This column is intended to point out another Artist who is this artist's mentor. It is not a problem to define this relationship in Access. In order to do this we need to have two graphical representations of the table Artist in our diagram. In our case there is already a second graphical representation of the table Artist (that was created when we created the ISA relationship between Artist and Composer in section 4.2.2). If there isn't one, we can create one by opening the Show Table dialog (from the Design toolbar on the ribbon or by right-clicking on the diagram area).

Drag and drop the referenced primary key column (Name) from the one graphical representation of the table Artist to the foreign key column (Mentor) in the other graphical representation of the same table (probably labeled Artist\_1). The Edit Relationship dialog will then show up:



The suggested relationship is as we expected: One artist has one mentor and the same artist can be the mentor of many artists. We activate the referential integrity and create the relationship:



In this case it is probably preferable to let both Artist and Artist\_1 be visible. If we place Artist\_1 behind Artist then we will not be able to see the new relationship.

We have now looked at all the types of relationships that are common in relational databases and how to create them in Access. Complete the database with the rest of the relationships (they are all defined in chapter 2). A version of the database including all the relationships is available at <http://coursematerial.nikosdimitrakas.com/access/>.



## 5 Querying A Database - Working With Data

In this chapter we will look at how to work with queries in Access. We will look at different ways of querying the database, but before we do that we have to populate, i.e. prepare, the database with data. With no data there is no way to verify that our queries work. It is not only important to have data in the database. It is equally important to have both enough, and varying data. It is important that the data in the database can represent all possible classes of cases that can occur. For example it is important to have an artist with no performance, an artist with just one performance, an artist with many performances, an artist with many performances of the same song, an artist with many performances in different years and so on. In order to achieve this kind of variation we need to have at least 5-10 rows in the strong entity tables<sup>3</sup> (for example: Artist, Composer, CD), 10-20 rows in tables that reference those tables (for example: Song), and 20+ rows in the tables that reference more than one strong entity table directly or indirectly (for example: CDSongPosition, SongPerformance, ArtistPerformance). In general, the weaker the table, the more variation can occur in it.

In the next section we will look at some ways for adding data into the database.

### 5.1 Preparing The Database With Data

In this section we will look at three ways of adding data in an Access database. The data inserted into the tables in this section are just a fraction of what we should have in order to fulfill the requirements described above.

Since it would be a time consuming process to input all the data, a database with data has been prepared and is available at <http://coursematerial.nikosdimitrakas.com/access/>.

The following subsections describe three basic ways of adding data to an Access database.

#### 5.1.1 Using SQL

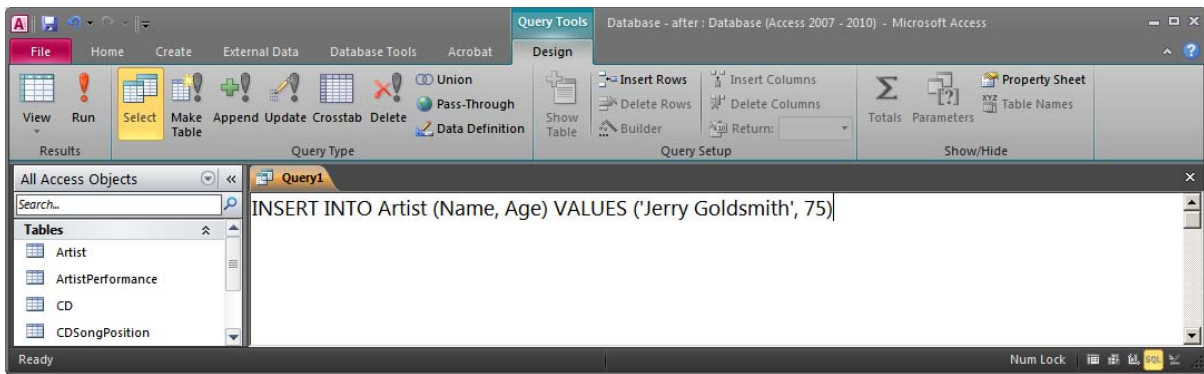
For those that dislike the graphical facilities provided in Access, there is the possibility of writing SQL statements to add data to the database tables. For example we could write the following INSERT statement in order to add a new Artist in our database:


```
INSERT INTO Artist (Name, Age) VALUES ('Jerry Goldsmith', 75)
```

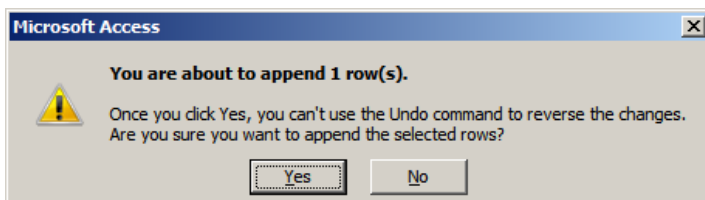
In order to run this in Access we have to create a new query and then go to the SQL mode (as we saw in section 3.2): Create a new query in design mode, close the Show Table dialog and press the SQL button the toolbar. We can now add our SQL statement in the text area:

---

<sup>3</sup> By strong entity tables, we mean here strong entities as described in, for instance, Database Systems. They are tables whose data is not dependent on the existence of data in other tables. Kind of like parent tables.

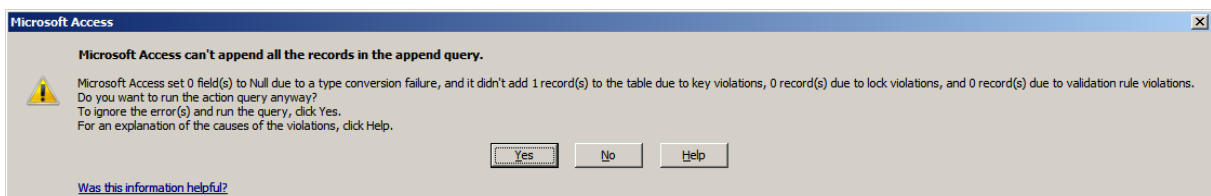


By simply clicking the Run button (  ) on the toolbar the INSERT statement will be executed and a new row will be inserted in the table Artist (as long as the data does not contradict any integrity rules). Depending on the Access settings, Access may ask you to confirm the insertion:



If it does, simply press Yes.

The new row is now in the table. If we try to execute the same INSERT statement again the database will detect the value 'Jerry Goldsmith' already exists and will inform us that the primary key rule of this table prohibits the insertion of the new row:

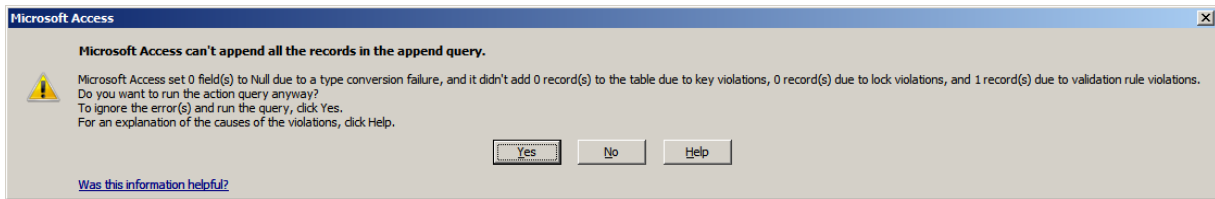


(Pressing Yes or No in this case will not make any difference, but the sensible answer is No)

We can also try to add another Artist with an illegal Age value (say 300). We write the following SQL statement:

```
INSERT INTO Artist (Name, Age) VALUES ('Vangelis', 300)
```

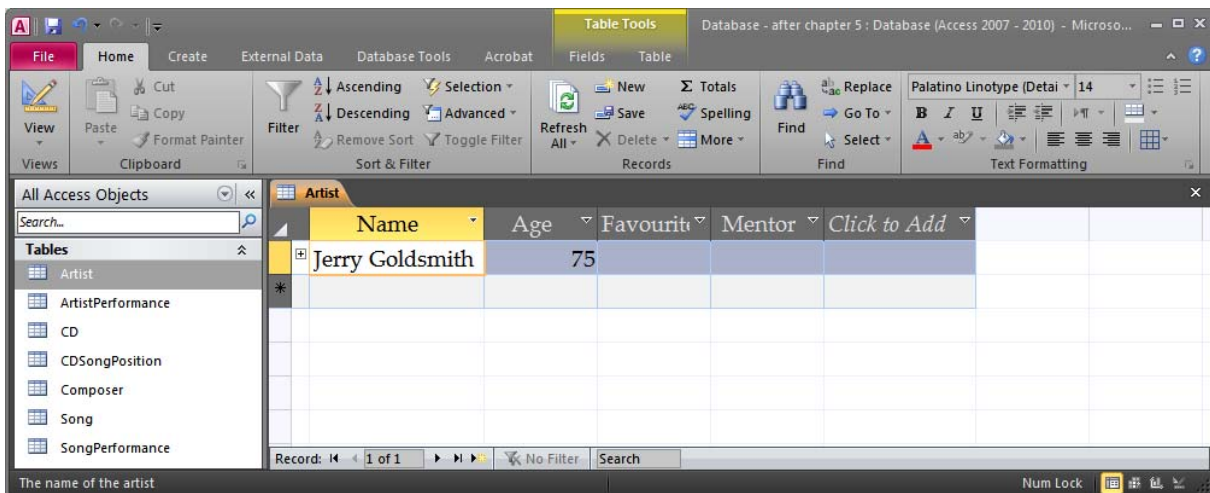
If we run this query, then Access will tell us that there was a validation rule violation. Sadly (and strangely) Access will not show us the specific text message in this mode. The error message looks like this:



The message that we defined: (“No way. The age must be between 1 and 200”), is not shown when we work in the SQL mode, but it will show up when working in the other modes described in the two following sections.

### 5.1.2 Using Datasheets

A more common way to work with table data in Access is working with datasheets. A datasheet looks just like an Excel sheet, but each column is a column of the table and each row is a row in the table. Let's continue adding artists in our table Artist. Open the table Artist by double clicking on it in the object browser (or right click and choose "Open"). The table Artist will open in the datasheet mode:

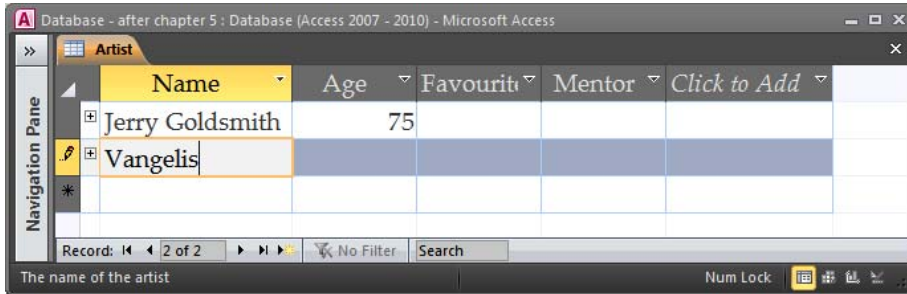


We can see now that a row is already in the table. It is the row we added earlier with the INSERT statement.

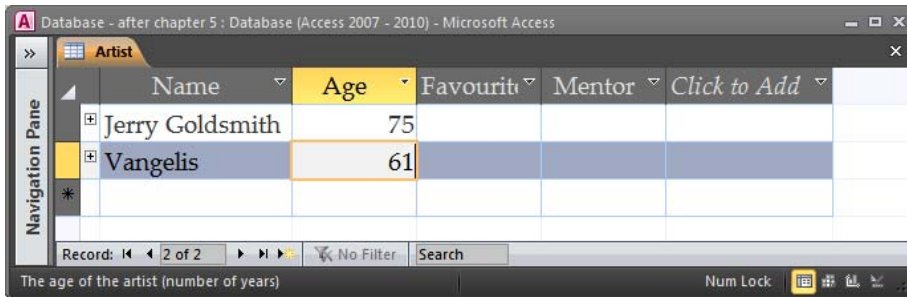
This view provides also other user-friendly features:

- On the top of the datasheet we can see the names of the columns.
- On the bottom we can see a navigator (Record: 1 of 1) which shows us the total number of records in the table, the number of the current record and also allows us to move to other records or create a new one (the button).
- On the status bar we can see the comment about the active field of the table. Place the cursor to the field Age and the status bar will change.
- In this mode Access will also recommend the default value for fields that have a default. Numerical fields have by default the default value 0. If this is not appropriate, it can be changed in the table design view (see section 4.1.1).
- In this mode we have the possibility to sort and filter the rows of the table. There are options for this under Home on the ribbon.
- It is also possible to add new columns to the table, but this should be avoided and done through the design view instead. To deactivate this option uncheck File > Options > Current Database > Enable design changes for tables in Datasheet view.

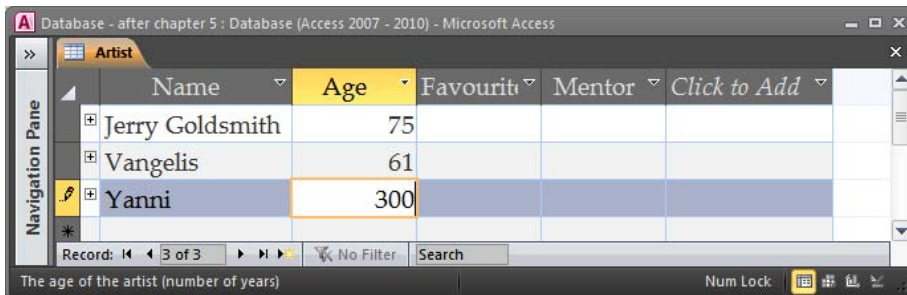
Let's now add a new row in the table. Simply place the cursor on the last row (the one with the star on the left) and type in the appropriate values at each cell:



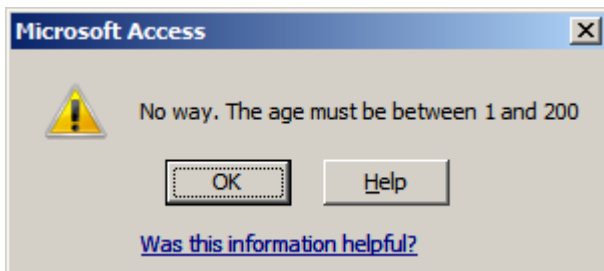
and then



We can now try to add a new artist with an invalid age (300 as before):



The moment we try to leave the cell, Access will try to validate the value and will show the appropriate error message:

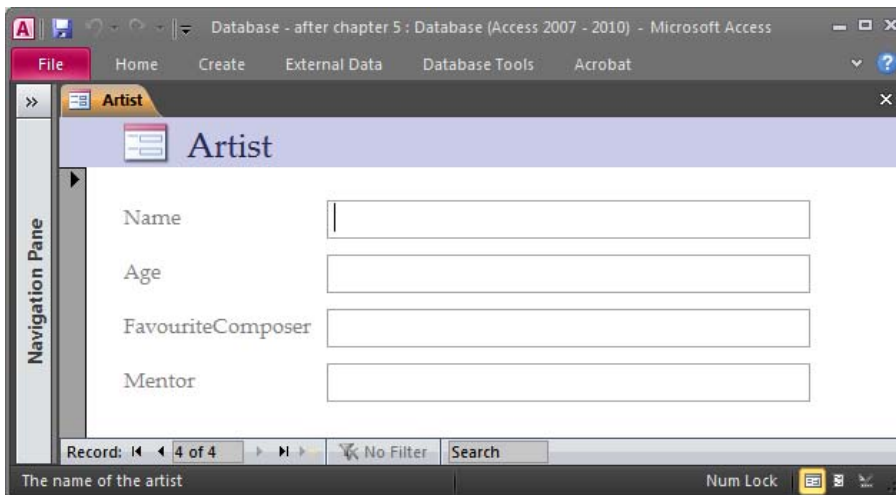


We can now change the value to the correct one (50).

- ① It is important to know that the new row is not stored in the table until we move out of it. We can move to the next row or to any old row in order to force Access to store the newly created row.

### 5.1.3 Using Forms

Another way to feed data into the database is by using forms. Creating forms can be from extremely simple to extremely advanced, depending on the functionality of the form. Chapter 6 discusses forms in detail so we are not going to create any form in this section. Below you can see an example of a form that can be used to input values in the table Artist:



We will see how to create this form in sections 6.1 and 6.2. The forms created in chapter 6 can be quite useful for inputting data in the database. Working with forms can actually save a lot of time, especially when the table we are filling in has many foreign keys.

In this section (section 5.1) we looked at different ways of putting data in the database. We barely created any rows in this section, and as we mentioned earlier it is important to have a database with enough data before we can start querying it. The rest of the sections of this chapter are about querying the database, so for that purpose there is a ready-made database with a bunch of data. It can be downloaded from <http://coursematerial.nikosdimitrakas.com/access/>. The data in this database has been created through both forms and datasheets, but all the forms have been removed so that the database is exactly as it was before, except from the data in the tables.

## 5.2 Writing SQL

With our database now populated with data, we can run some queries, and get some meaningful answers. In this section we will just create a couple of simple SQL SELECT statements and run them in Access. If we look back at chapter 2, we can see that one of the information needs defined is: “*Show all CD titles produced in 1999!*”

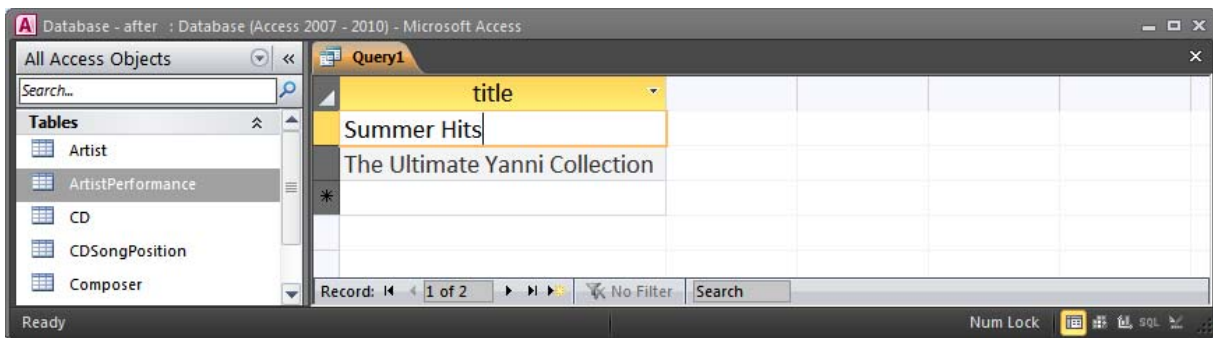
This can easily be solved with the following SELECT statement:

```
SELECT title
FROM CD
WHERE year=1999
```

We can now execute this SQL statement by creating a new query in design view and then switching to the SQL mode (as we did in section 4.1.2). We can insert our SQL statement in the text area and we are ready to receive the result:

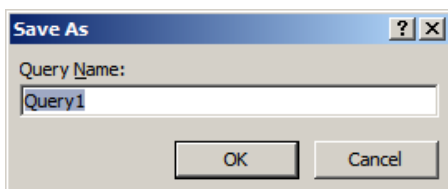


There are three ways to show the result. We can press the Run button (🚀), we can press the View button (📄) on the ribbon, or we can right click on the query name and choose "Datasheet View". They will all switch to a datasheet view that presents the result:



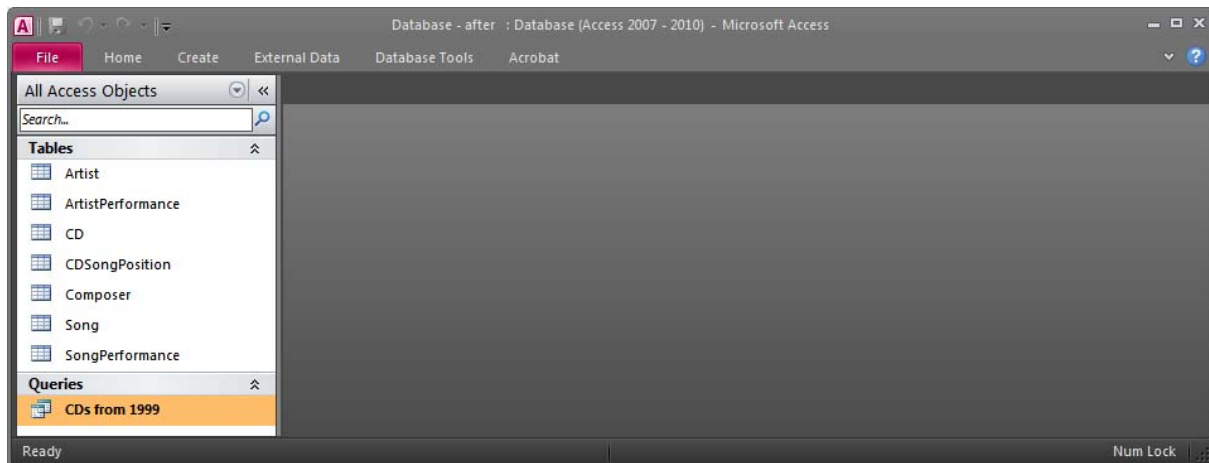
We can from this view return to the SQL edit mode by clicking the SQL View button on the ribbon or by right clicking on the query name and selecting "SQL View".

We can even save this query if we want. We can either press the Save button (💾) above the ribbon, we can right click on the query name and choose "Save" or we can just close the query tab and let Access ask us whether to save or not. In any case Access will ask for a name for the new query:



We can call it something intuitive; for example "CDs from 1999" (and press OK). The new query object is now available in object browser:





- ① If you want to edit the SQL statement of the query, right click on it in the object browser and press the "Design View" and Access will open the query automatically in the SQL editing mode.
- ① If you want to see the result of the query, double click on it in the object browser and Access will automatically run the query and show the result.
- ① Queries in Access are what other products call views.

### 5.3 Reusing Queries

In the case description in chapter 2, there was also the following information need: “Which CDs include songs written by Jerry Goldsmith?”

This can be done with the following SQL statement:

```
SELECT DISTINCT title
FROM CD, CDSongPosition cdsp, SongPerformance sp, Song s
WHERE CD.ID = cdsp.CDID
AND cdsp.Song = sp.Song
AND cdsp.Date = sp.Date
AND sp.Song = s.ID
AND s.Composer = 'Jerry Goldsmith'
```

But let's assume either that this is too big and confusing, or that a part of it is very commonly used. We can therefore create a view of one part of the query and then use this view in order to create the final result. Let's say that we often need to know the composer of the song performances. It can therefore be good to create a view that can give us this information without having to join Song and SongPerformance every time. We could easily create this view in standard SQL (which does not work in Access):

```
CREATE VIEW SongPerformanceComposer AS
SELECT sp.Song AS Song, sp.Date AS Date, s.Composer AS Composer
FROM SongPerformance sp, Song s
WHERE sp.Song = s.ID
```

Then we could write a simpler SELECT statement in order to produce the final result. We can in this statement's FROM clause include the view we created earlier:

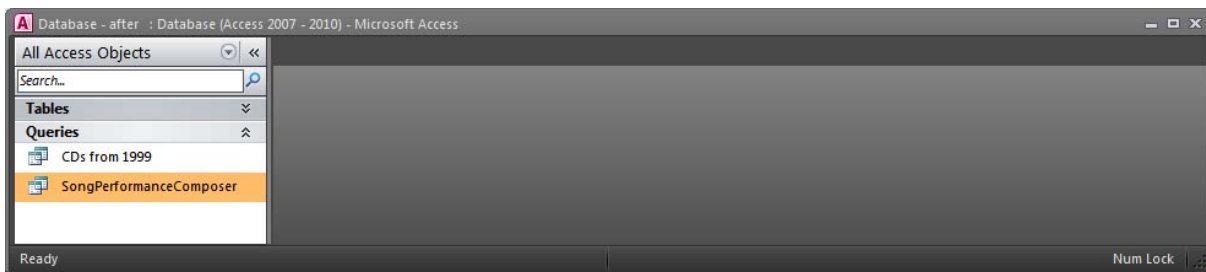


```
SELECT DISTINCT title
FROM CD, CDSongPosition cdsp, SongPerformanceComposer spc
WHERE CD.ID = cdsp.CDID
AND cdsp.Song = spc.Song
AND cdsp.Date = spc.Date
AND spc.Composer = 'Jerry Goldsmith'
```

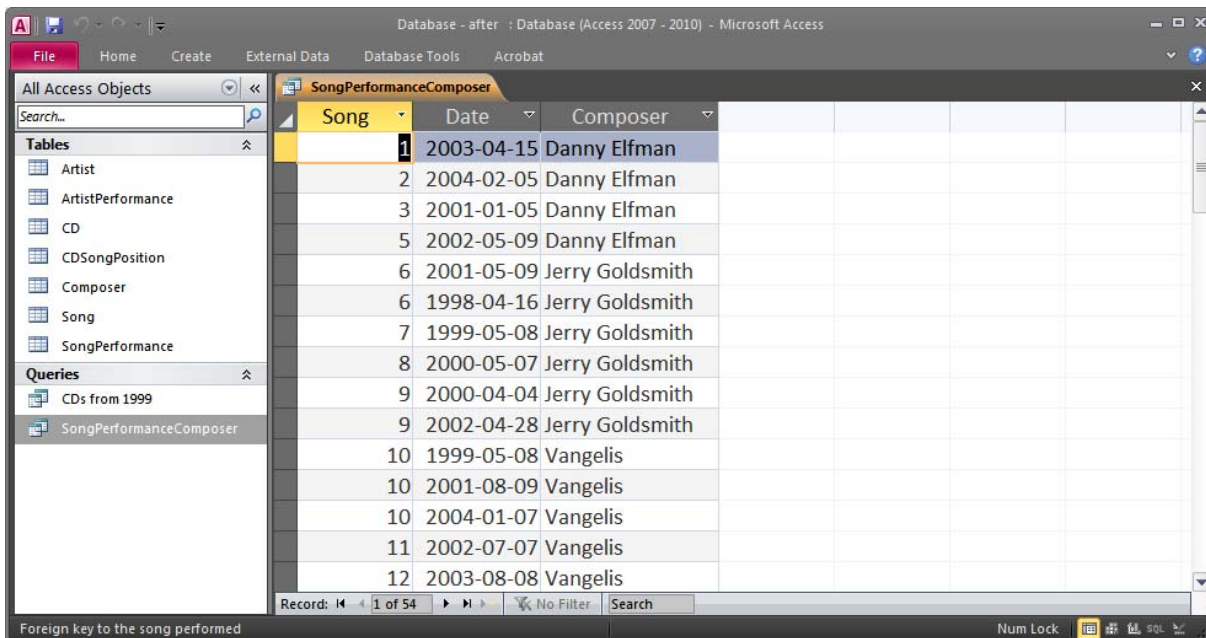
In Access, creating a view is the same as creating a query. A query is a view. So we can create a query for the following SQL statement and save it as "SongPerformanceComposer". This query would be the equivalent to the view that the CREATE VIEW statement earlier creates.

```
SELECT sp.Song AS Song, sp.Date AS [Date], s.Composer AS Composer
FROM SongPerformance sp, Song s
WHERE sp.Song = s.ID
```

The query is now visible in the object browser:



We can also open it and see the result:

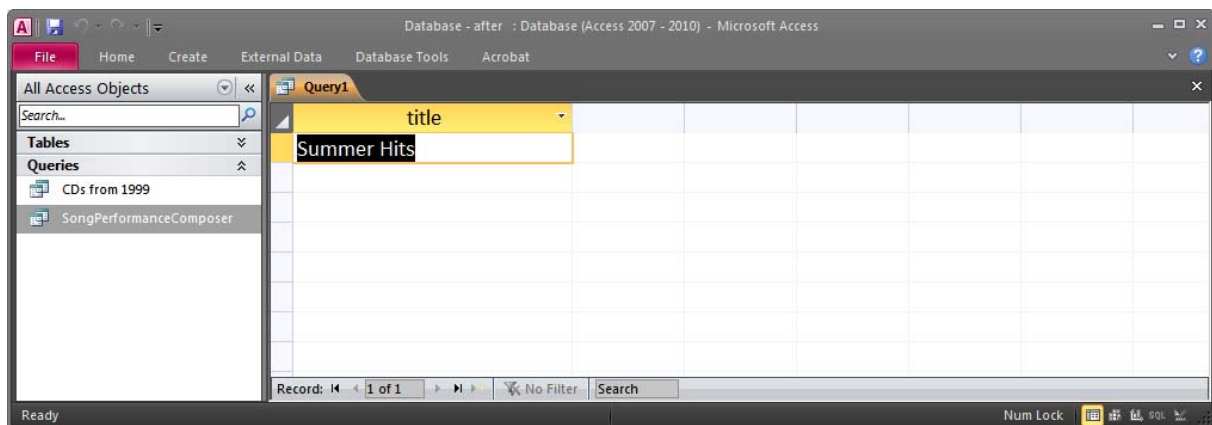


We can now create a new query (in the usual way) for the following SQL statement, which uses the other query in order to produce the final result:

```
SELECT DISTINCT Title
FROM CD, CDSongPosition cdsp, SongPerformanceComposer spc
WHERE CD.ID = cdsp.CDID
AND cdsp.Song = spc.Song
AND cdsp.Date = spc.Date
AND spc.Composer = 'Jerry Goldsmith'
```

In the FROM clause of the statement we have now an object that is not a table. Instead, it is a query.

We can now run our latest query and receive the result:



We can save this query as "CDs With Jerry Goldsmith".

A database including all the progress so far is available to download at <http://coursematerial.nikosdimitrakas.com/access/>.

## 6 Forms

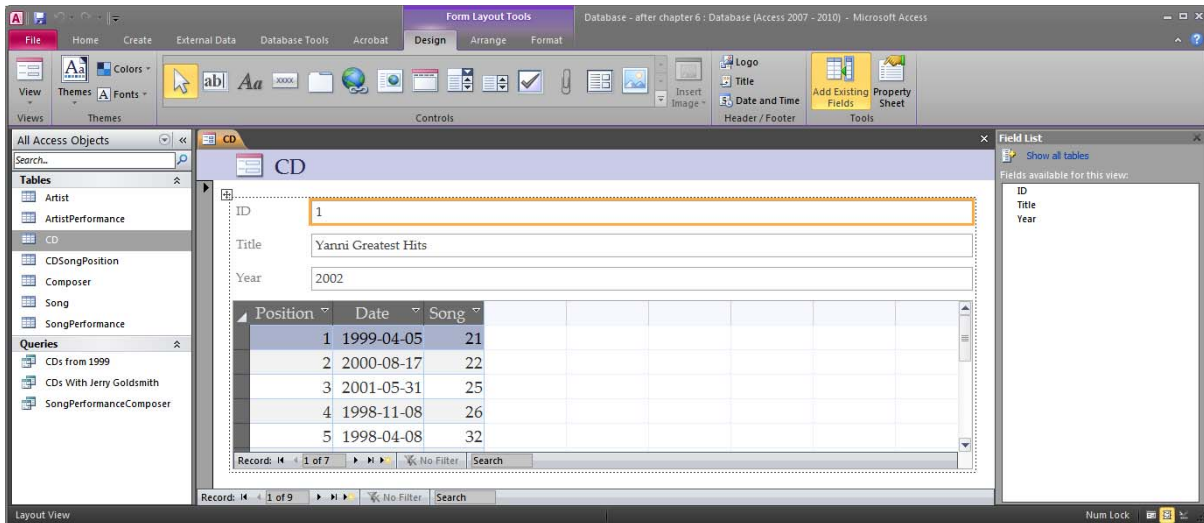
Forms can be used both for adding and editing data, as well as for browsing and presenting data. In section 5.1 and especially in subsection 5.1.3, we discussed entering data in the database and also doing it with forms. An example of a form was shown there, but we did not see how we could create it. In this chapter we will see how to create forms. We will look at some common types of forms, from the simplest possible to more complex master-detail forms.

### 6.1 Simple Forms

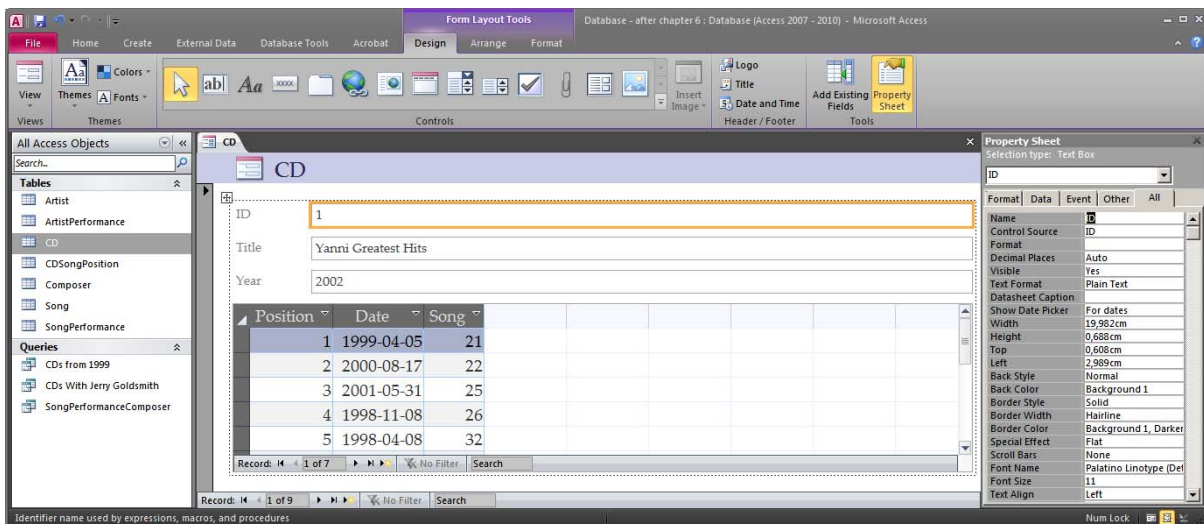
By simple form, we mean a form that only has simple input fields for adding data to a particular table. Such forms should only be used with tables that have no foreign keys, since then all the columns of the table can be filled in by the user and the values are independent of any values in other tables. In our case description in chapter 2, we had the following user interface need: "A form for registering a new CD in the database". The table CD is independent of all other tables. No column of the table CD depends on values of other tables. We can therefore create a simple form for this table.

Under the Create option in the ribbon, there is an option "Form". This will create a simple form for the object selected in the object browser. Select the table CD in the object browser

and press Create > Form. Access will create a new (unsaved) form with one field for each column in the table. Access has also created a subform based on the foreign key relationship that CD has to CDSongPosition:



On the right side we have the Field List and the ribbon has automatically switched to Design. The Field List shows the columns that are available in the table selected as the source of the form. In this case it is the table CD. On the ribbon we have the option to show the Property Sheet instead of the Field List. The Property Sheet shows all the properties of the selected form object:



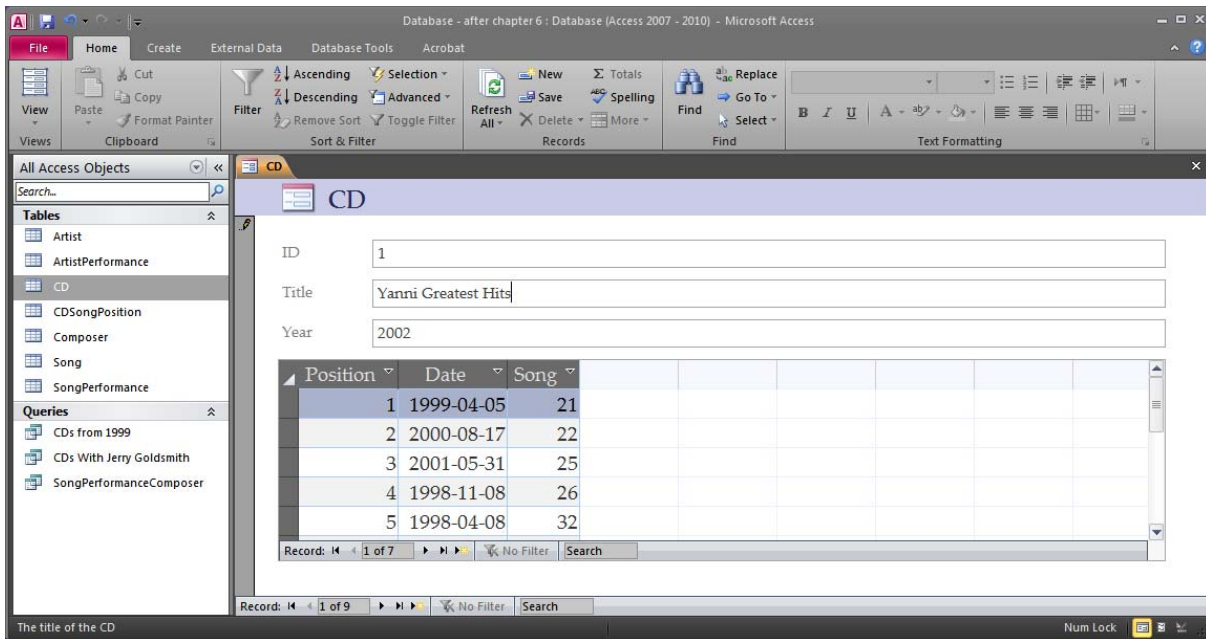
On the ribbon we have a set of controls. These are objects that can be added to a form. Every control has specific properties. Some controls are meant to be linked to database columns so that the data in the database can be visualized in a specific way in the form. For example, an Image control can be used to display an image stored in the database.

A form can be viewed in three different ways: Form View, Design View and Layout View. The Form View is for when the form is being used. The Design View and the Layout View are used to create the form, add controls, configure them, layout them, etc. The Design View and the Layout View are very similar, but in Layout View, live data from the tables is used to show what the form will look like. The ribbon has three tabs that together constitute the

"Form Design Tools". There is the Design tab with all the controls and some general options for configuring forms, there is the Arrange tab which is used to manage the layout of the controls on the form, and there is the Format tab that is used to select fonts and colors.

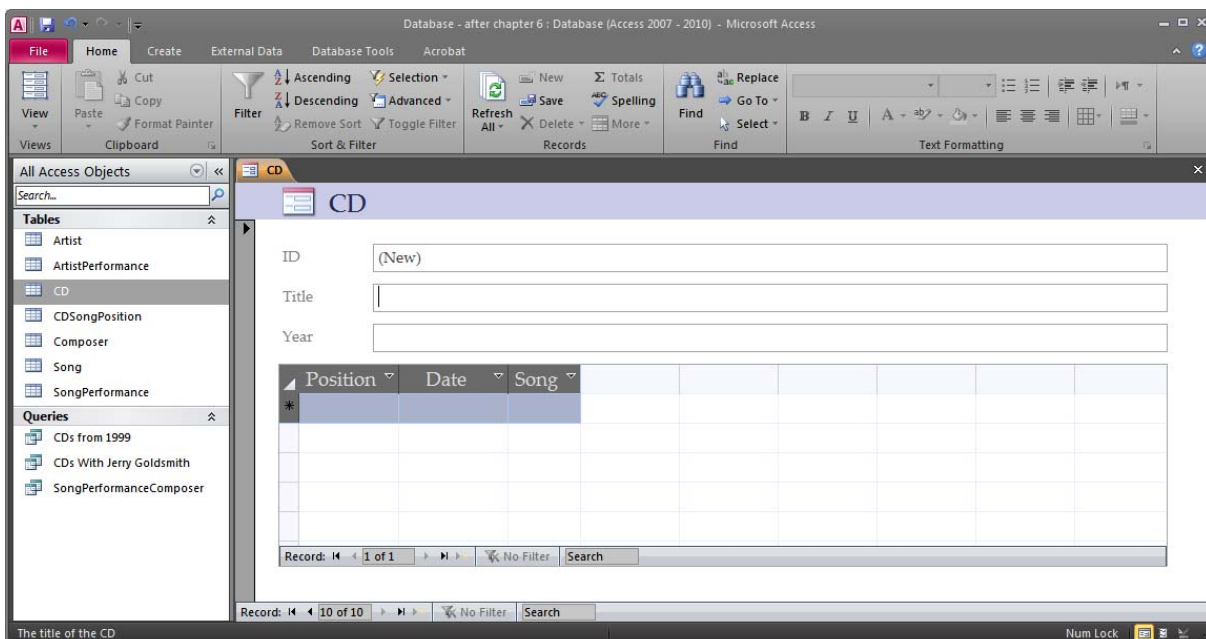
All automatically generated forms offer basic browsing functionality with a record navigator and a record selector, as well as the possibility to edit existing records or add new records.

The created form looks like this in Form View and we may edit any field:

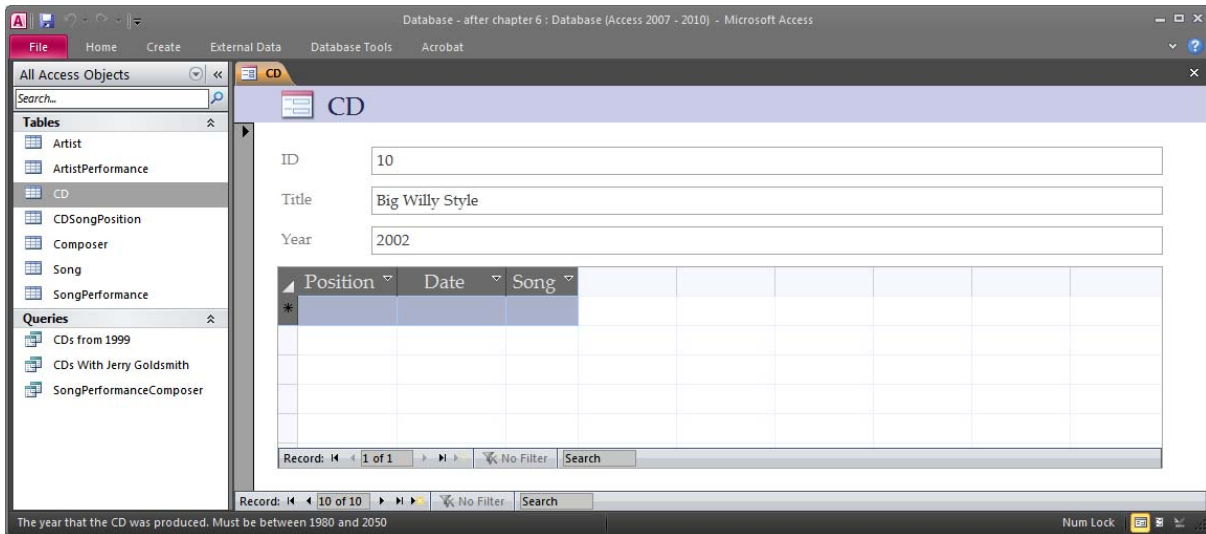


When a record is being edited, Access marks this with a little pencil symbol (🖋️) on the left.

We can also add a new CD by creating a new record (by pressing the ➕-button at the bottom):

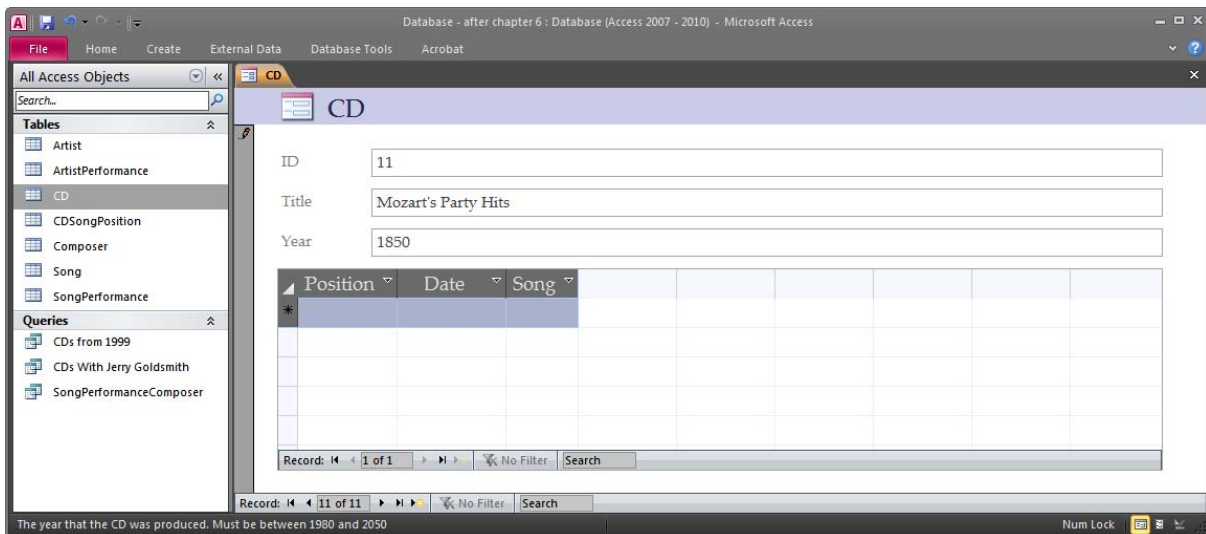


The ID field is automatically set by Access, so we only need to fill in the Title and the Year:

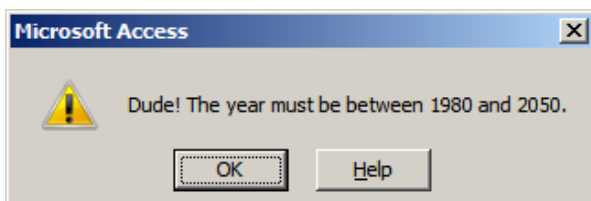


We can see that even in this view, Access shows the comment for the activated field in the status bar of the window.

We can also try to add another row but with an invalid Year value (say 1850):

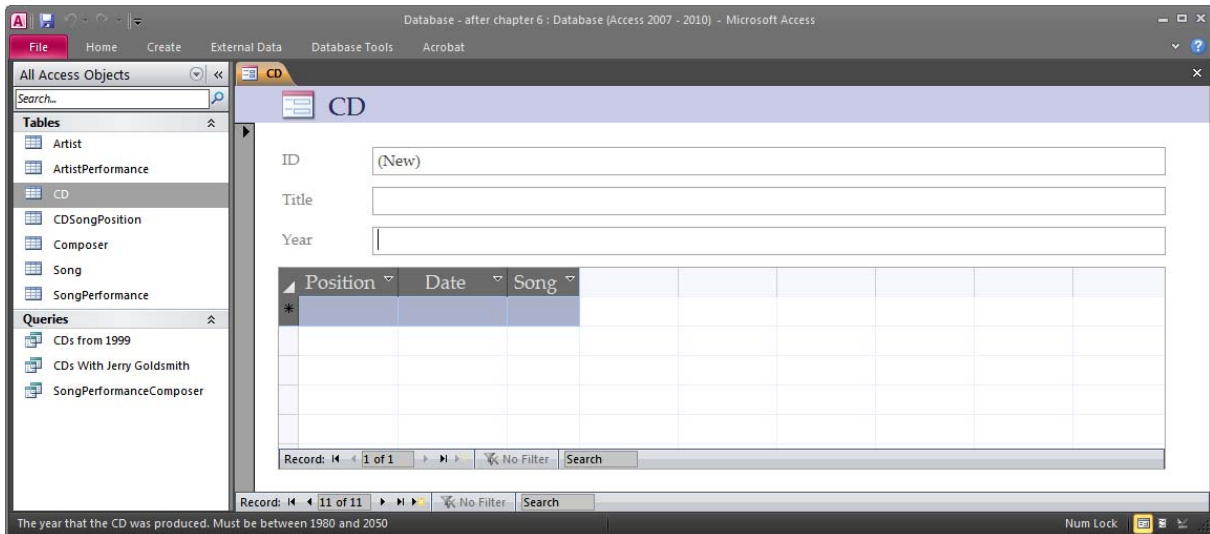


The moment we try to move the cursor out of the Year field, Access will detect the violation and show the specified Validation Text:

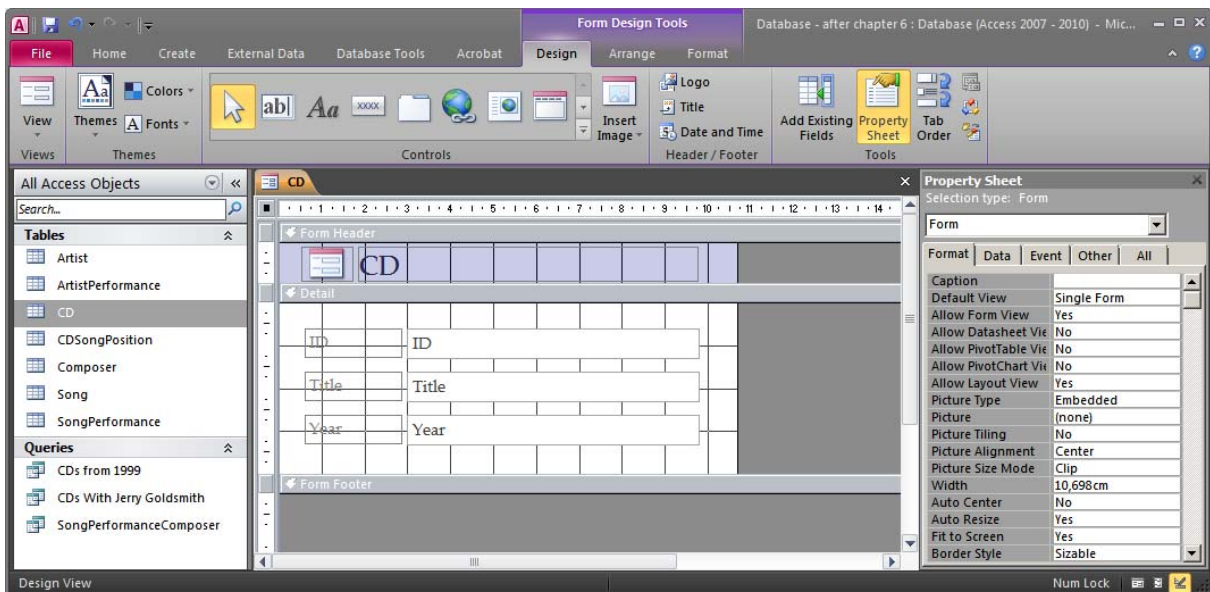


We can cancel the new record by simply pressing the Escape key (twice) after we press OK at the pop-up message dialog. The record will return to each original blank state:





If we don't like the layout of this automatically generated form, we have the possibility to switch to the Design View and change it. To switch to the Design View, press the Design View button on the right side of the status bar or select Design View from the ribbon on the Home tab. In the Design View (or in the Layout View) we move around the controls, add or remove controls, configure their format, etc. We could for example remove the grid showing all the CDSongPositions and change the width of the fields.

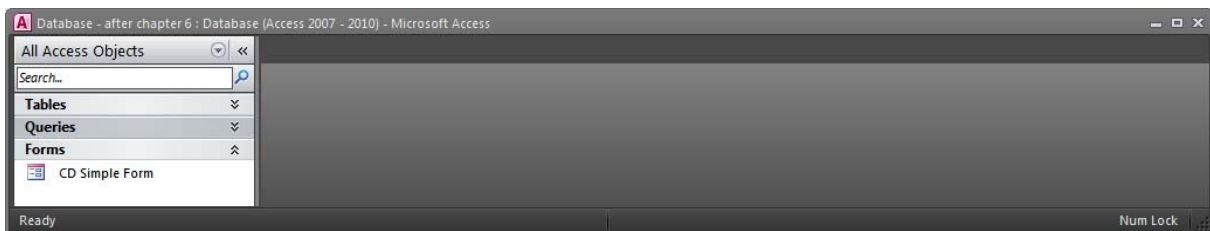


There are two basic ways of changing the design of the form and the form controls: Through the ribbon and through the Property Sheet. When a form object (a control or a section of the form) is selected, the Property Sheet will show all the relevant properties and their current values. Most values can be changed and the Property Sheet will offer all applicable options. The same applies to the options provided directly on the ribbon. For example the Font property shows the font in use and offers a list of possible fonts to choose from.

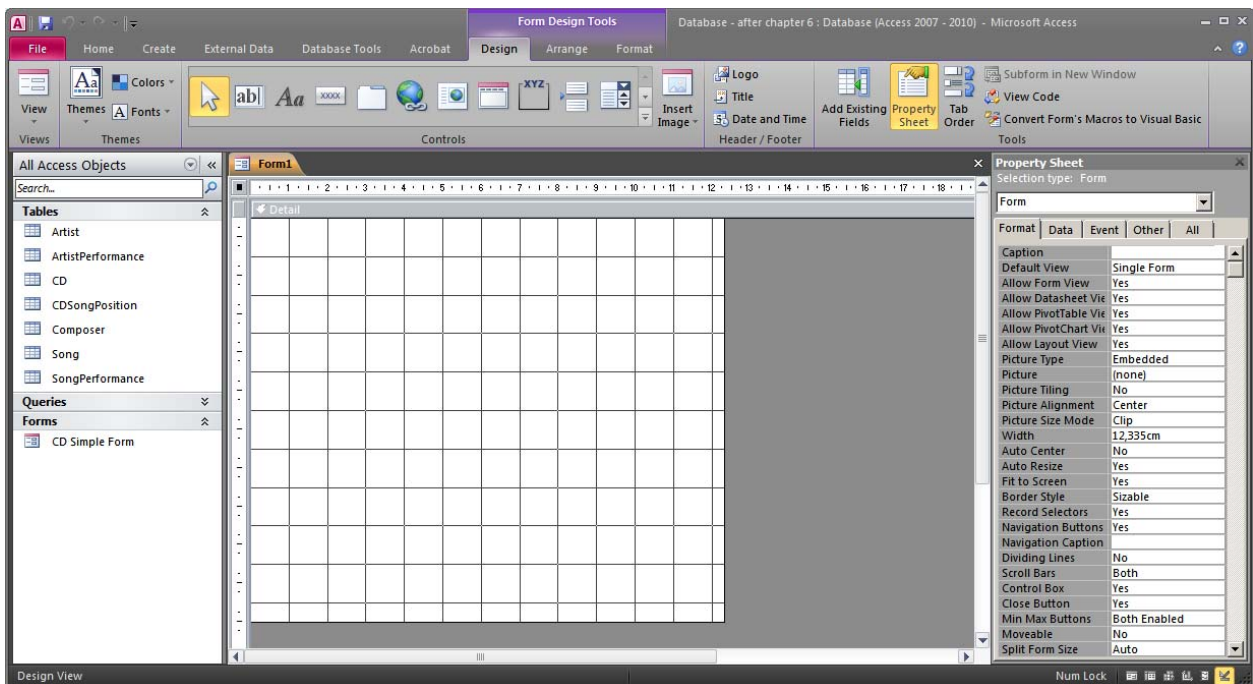
After playing around for a bit, the form may look like this:



A form can be saved at any time, but Access will ask if we try to close an unsaved form. We save our form as "CD Simple Form". The form will now be available in the object browser under Forms:



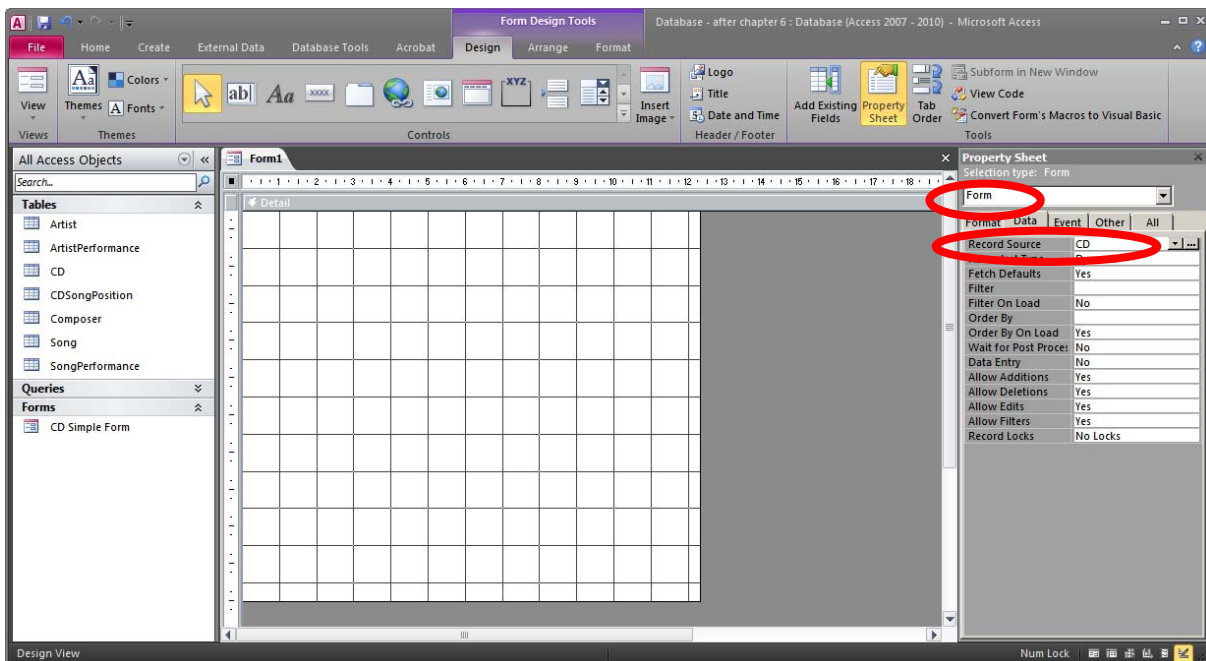
We can also create a form without using Access' AutoForms. We can create an empty form in Design View and then manually add all the components we want. Create a new empty form by selecting Create > Form Design from the ribbon. An empty form will appear and it is now up to us to design it:



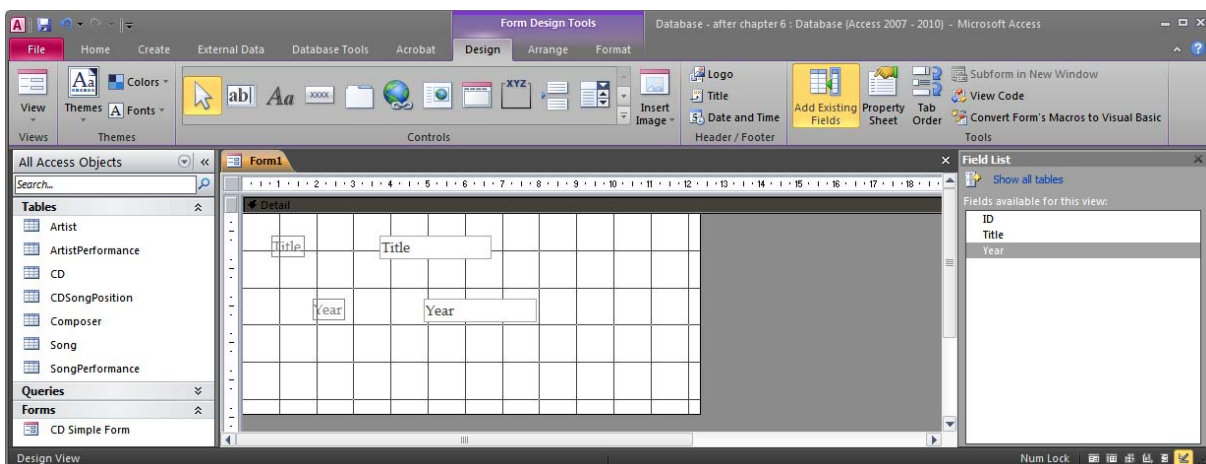
The first thing we have to do is to specify the table that is going to be the data source for this form (i.e. the table that contains the columns and data that will appear in the form). We can do this in the Property Sheet. Select the table CD as the Record Source for the form. The



property Record Source is available under Data (and under All) when the Property Sheet shows the properties of the form:

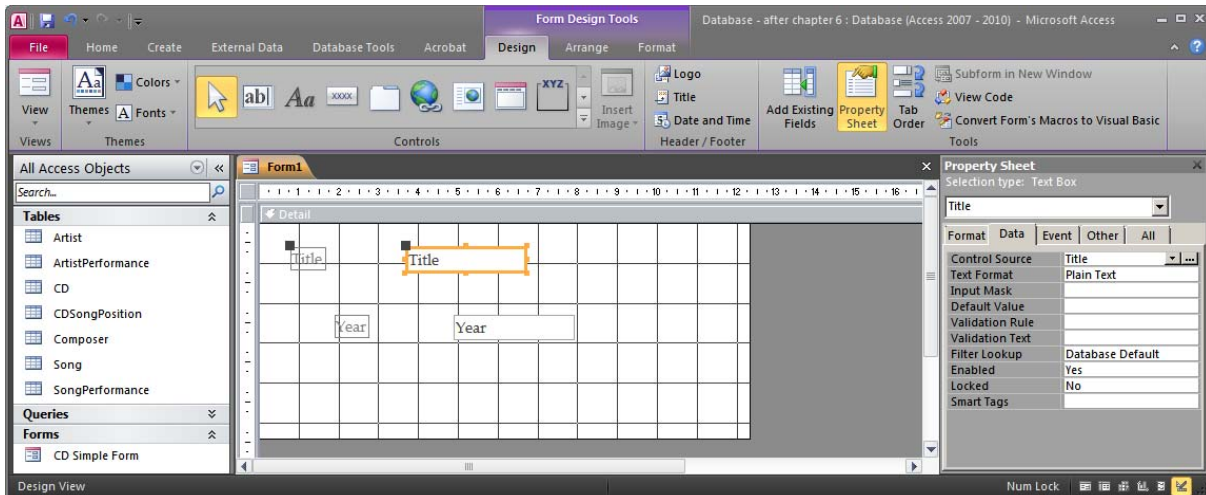


When a Record Source has been selected the Field List becomes available. The Field List shows the columns of the selected Record Source and we can drag and drop them into the form. This will create a label and a text field (Text Box) for the selected column of the table. Drag and drop the fields Title and Year onto the free area of the form so that it looks something like this:



The column ID is not visible in the form, but it exists of course in the table. Since the column ID is an Autonumber, Access will take care of the values for new rows automatically.

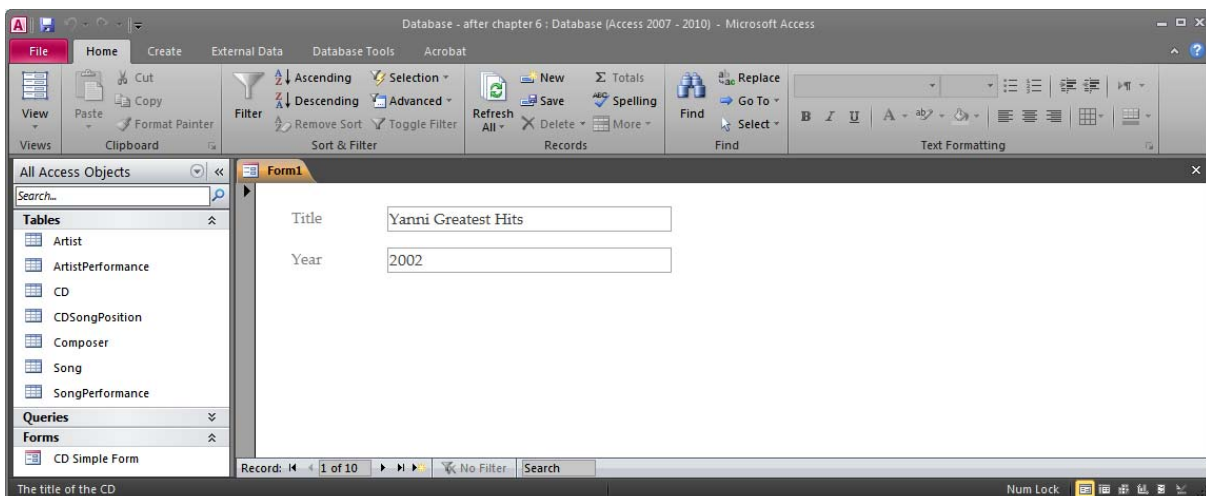
Now activate the Title Text Box and look at the Property Sheet under Data:



The value of the property Control Source is Title. This means that this Text Box is linked to the column Title of the form's Record Source (the table CD). This means that whatever we see or type in this field is stored in the Title column of the current row of the table CD.

We may also improve the layout of the form by choosing an appropriate layout from the Arrange tab of the ribbon.

Switch to the Form View to see and use the form:



We can now save this form as "CD Designed Simple Form".

## 6.2 Lookups

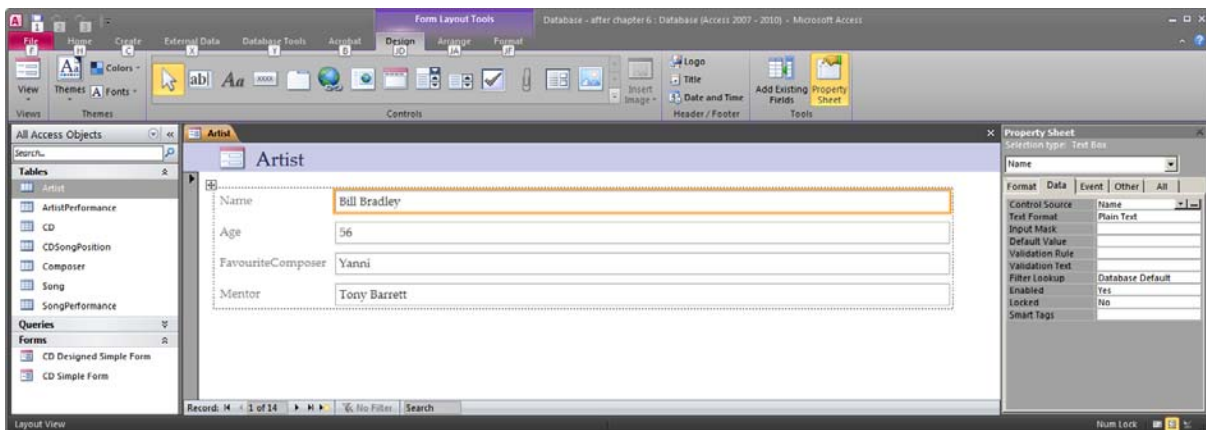
In this section we will see how we can create forms for tables that have one or more foreign keys. The main difference from the forms created in the previous chapter is that some of the columns of the table may not accept just any value. The value of a foreign key column is dependent on the values that exist in the referenced table.

We can start with a form that was required in the case in chapter 2: "A form for registering a new Artist in the database". The table Artist has two foreign keys. One of them (FavouriteComposer) references the primary key of the table Composer, while the other (Mentor) references the primary key of the table Artist. Our form should therefore provide a

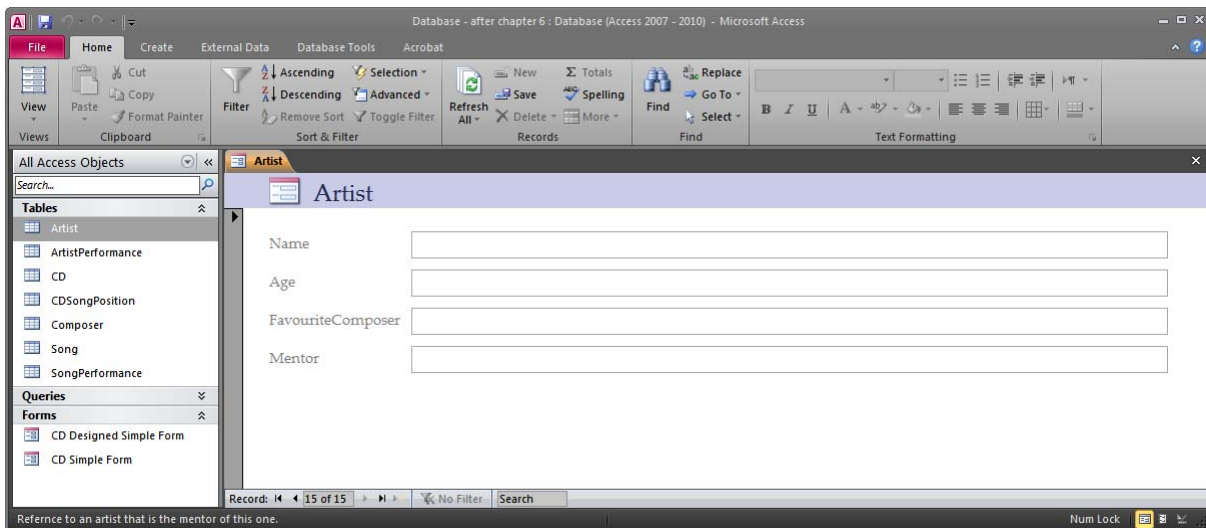
possibility of selecting one of the existing composers as the favourite composer and one of the existing artists as the mentor.

We can either create a form from scratch (as we did in the previous section), or we can let Access create a standard form for us and then modify it. In this example we will do the latter.

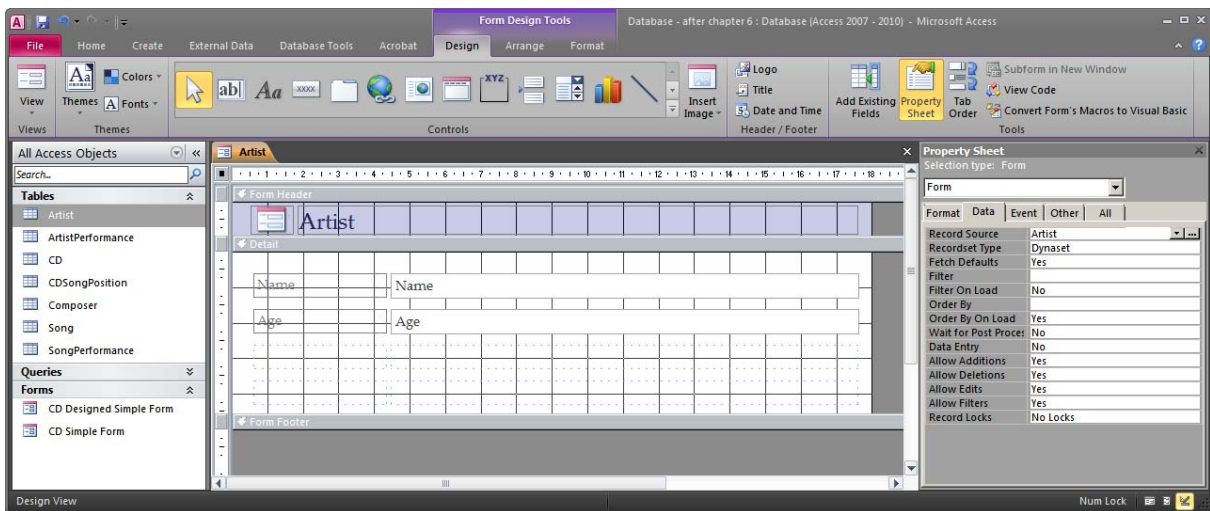
Select the table Artist in the object browser and select Create > Form from the ribbon:



Access has created one Text Box per column in the table. If we try to use this form, we must specify the values of every column manually. This is of course not practical because we need to remember exactly which values are available in the referenced primary keys to maintain referential integrity.

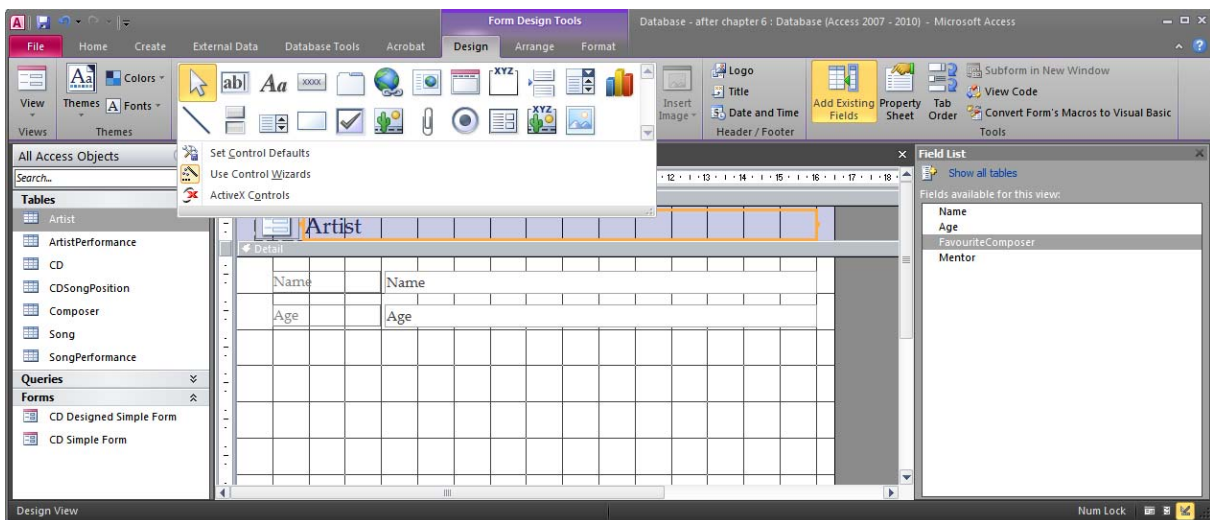


The first two Text Boxes are fine because it is up to the user to write the name and age of the artist. The third and fourth, though, are not so good. They require that the user knows exactly which composers and artists there are in the system and requires that the user doesn't make any spelling mistakes. We will therefore switch them to Combo Boxes. To do this we need to switch to the Design View and then remove the Text Boxes, and add Combo Boxes (from the ribbon). There is also an option to change a Text Box to a Combo Box (by right-clicking on the Text Box, but then we will need to configure the Combo Box manually. By adding a new Combo Box from the ribbon, a wizard will assist us in the configuration. First remove the two Text Boxes (and their associated Labels):

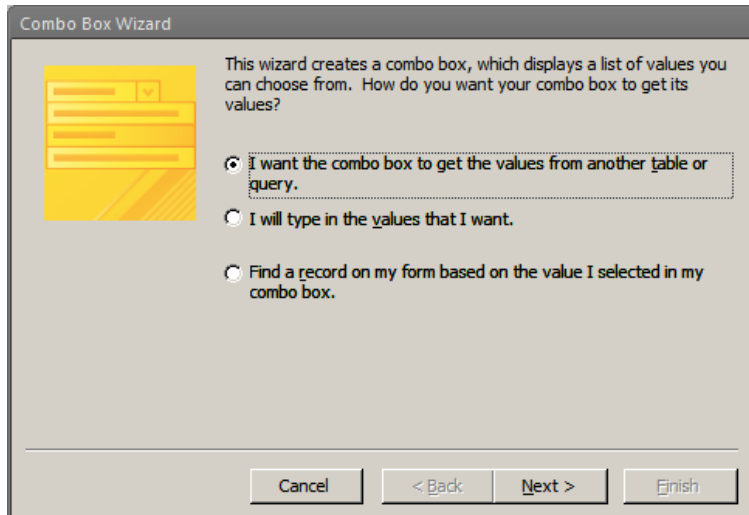


The dotted lines that remain are part of the selected layout. The default layout is a Stacked layout where all the Labels are on the left and all the Text Boxes are on the right and they all have the same width and spacing. When we add a new control, we may add it in the existing layout by dropping it at the right place.

We can now add two Combo Boxes: one for the column FavouriteComposer and one for the column Mentor. To do this we need to locate the Combo Box control under Design on the ribbon. When adding complicated controls, Access will provide a wizard for configuring the new control, but only if the option "Use Control Wizards" is selected:

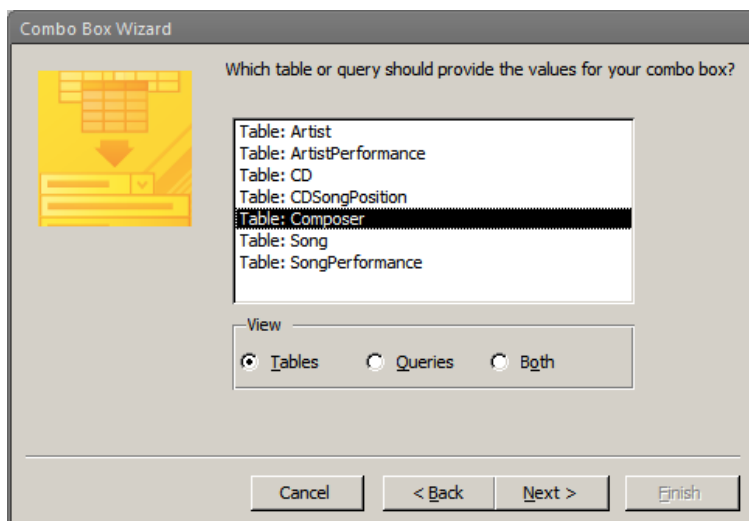


Now, we are ready to add our first Combo Box onto the form. Press the Combo Box button ( ). Move the mouse to the place on the form where you want to place the new Combo Box (the mouse cursor will change to indicate that you are about to add a new Combo Box). The position is probably not so important, since we can add it to the existing layout later. As soon as we have clicked to create the new Combo Box, the Combo Box Wizard will appear:



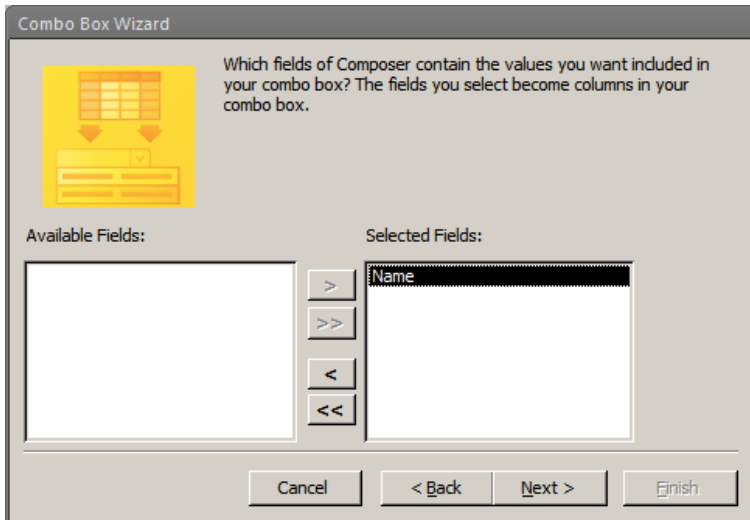
There are three alternatives, but the first one is what we want in this case. We want to fill the Combo Box with values of a table. Select the first alternative and press Next.

The wizard will now ask us to select which table or query we want to use. We want to use this Combo Box to let the user select an existing composer as the new artist's favourite composer. We choose therefore the table Composer:

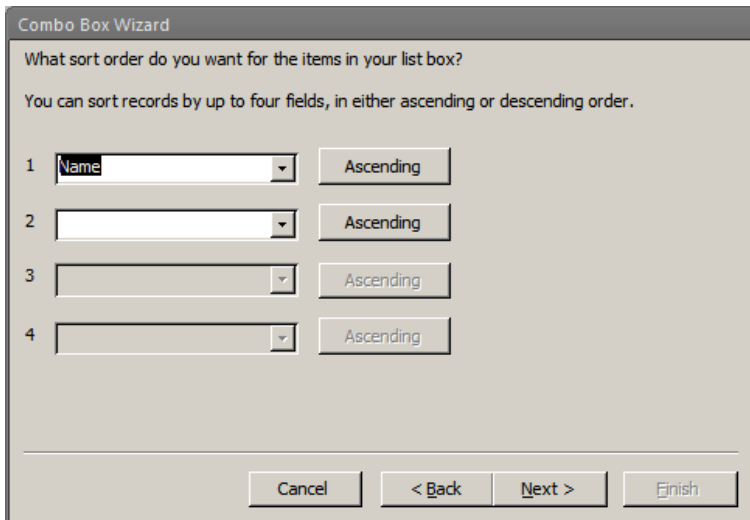


We can now press Next again and the wizard will ask us to select which columns of the table Composer that we want to include in the combo box. There is only one column, so there is not so much to think about. Just add the column Name to the selected fields:

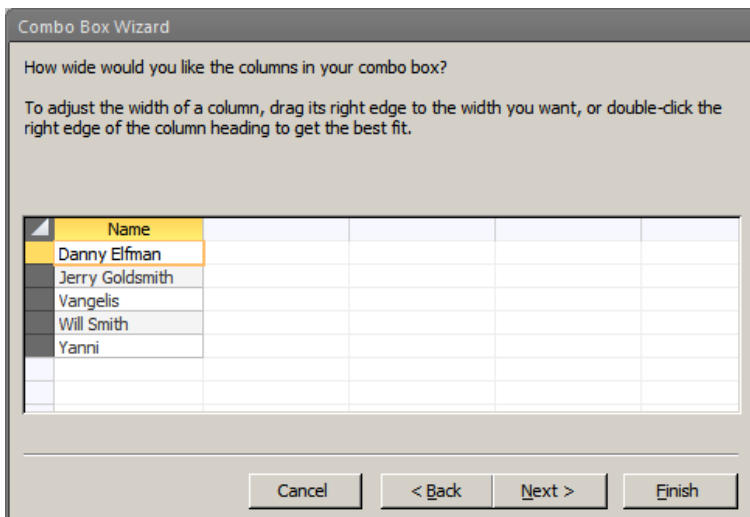




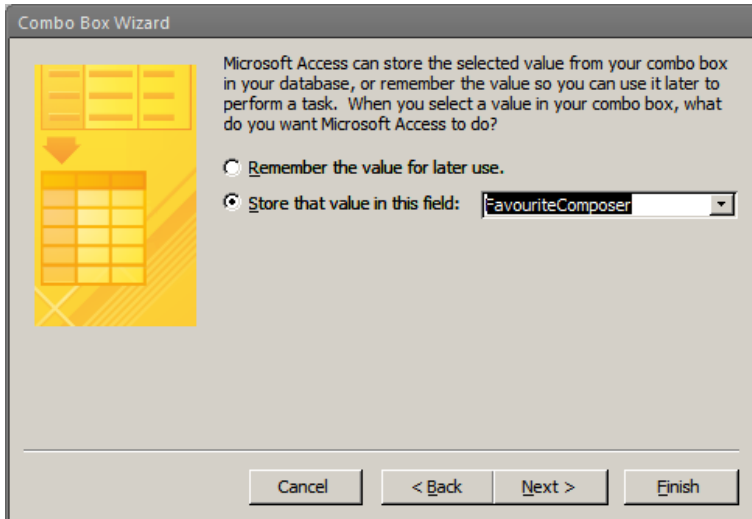
Press Next and the wizard will ask us to choose how the items of the combo box should be ordered. We can, for example, order the composers alphabetically:



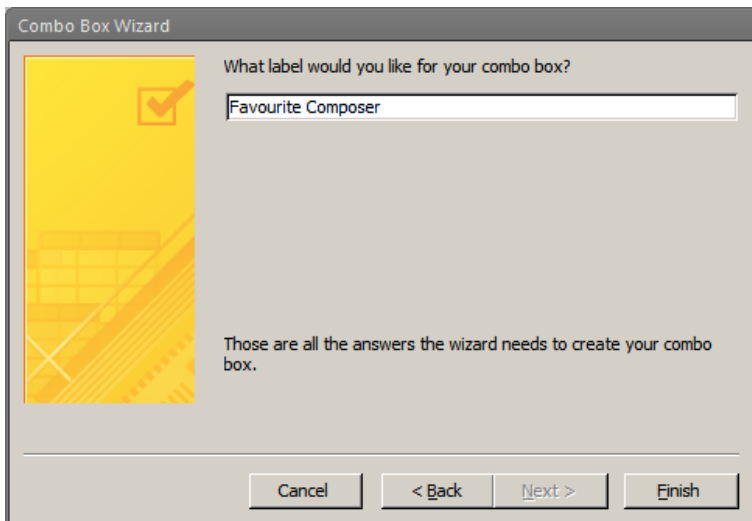
Press next and Access will display the available values in the defined order:



Here we can also modify the width of the column(s) of the Combo Box. This looks fine so we move on to the next step. This is where we finally connect our Combo Box to the table of the form. Here we can instruct Access to place the selected value of the Combo Box in a particular field of the table Artist. As we said earlier this Combo Box will help us choose the favourite composer, so the selected value should be stored in the column FavouriteComposer of the table Artist:

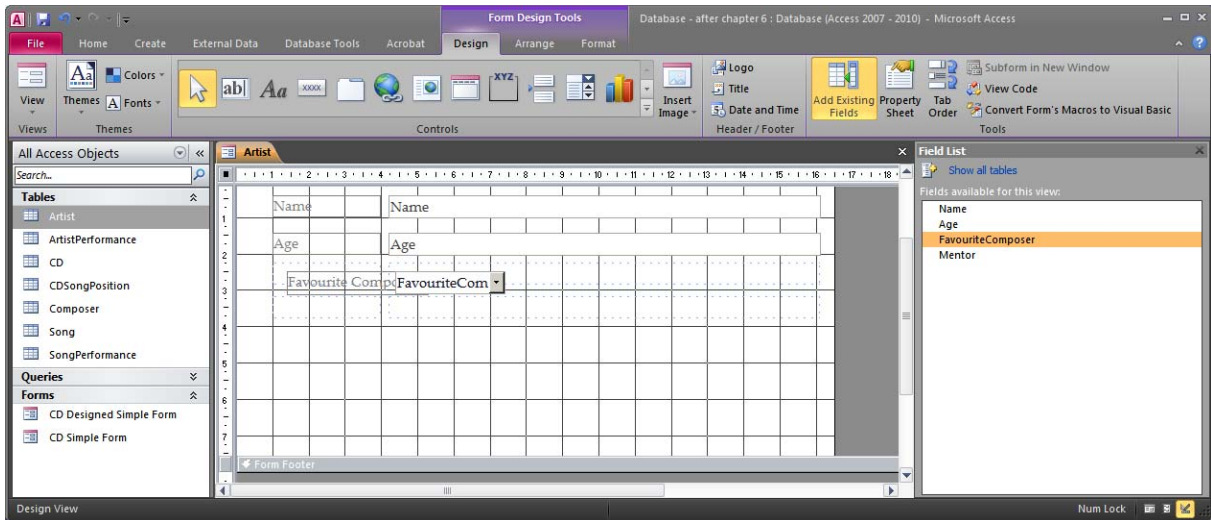


We can now move on to the final step of the wizard where we just need to define the text of the Label of the new Combo Box:

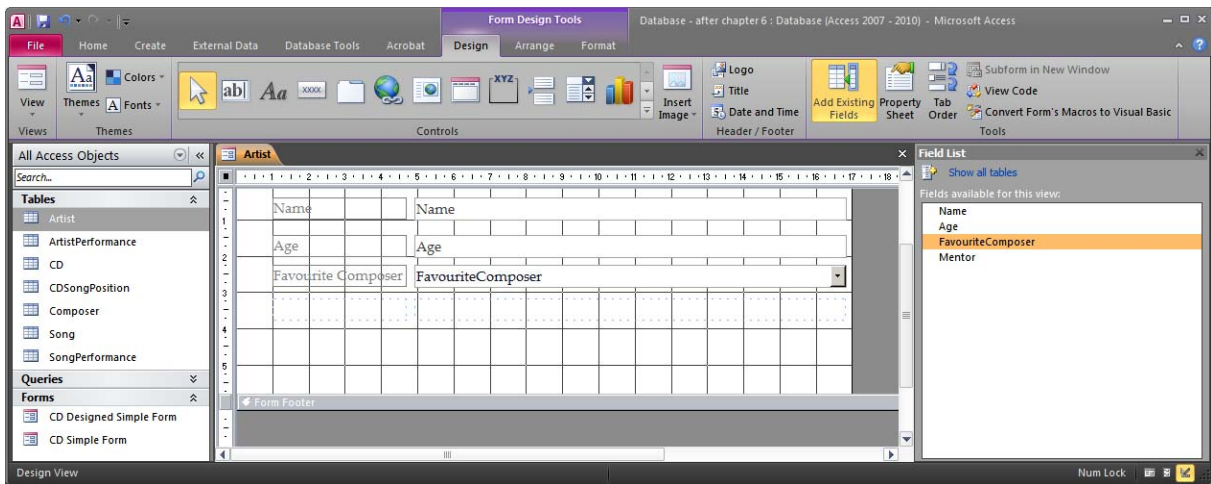


We can now press Finish and look at the form:

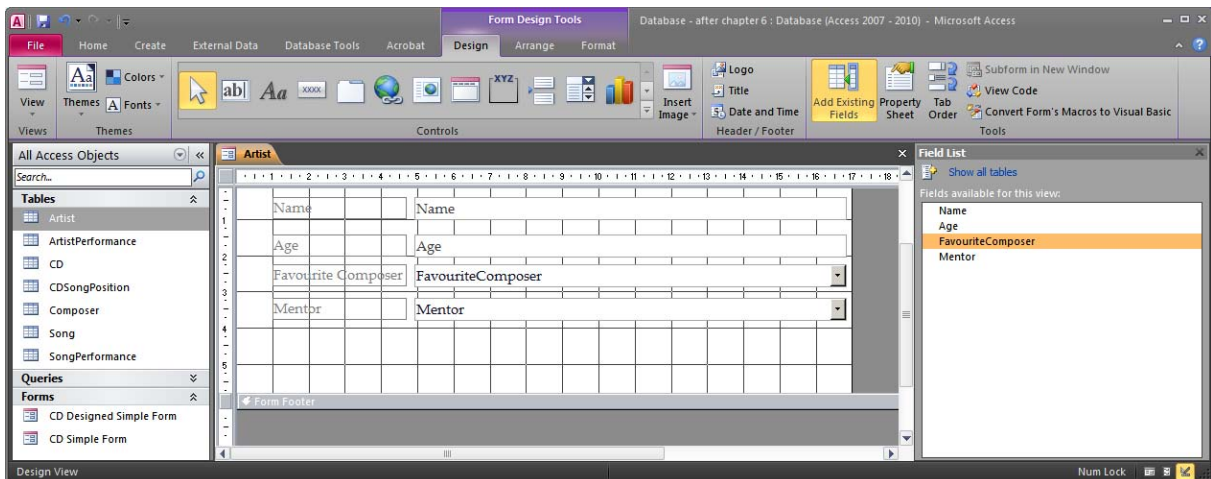




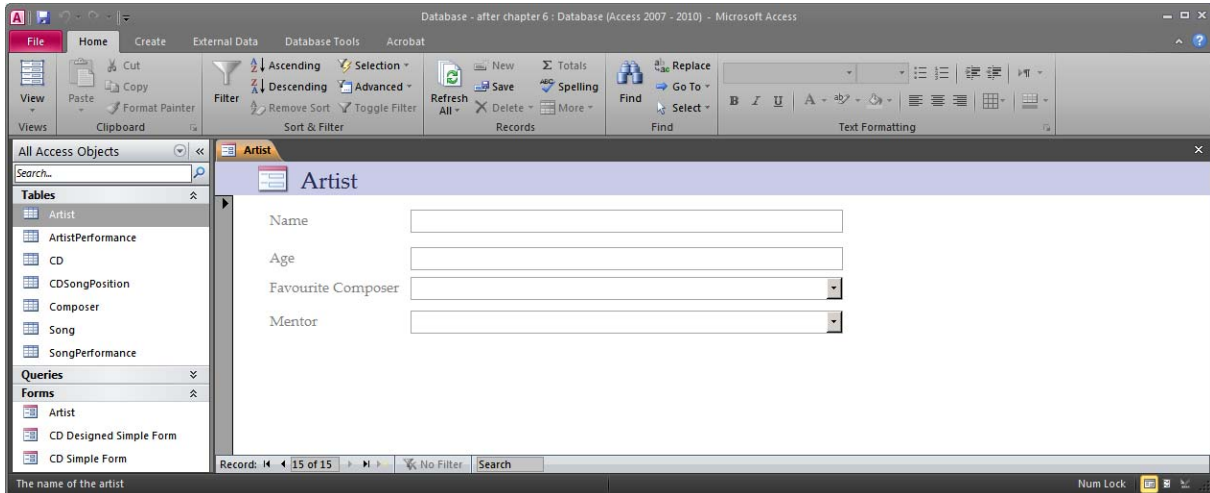
We can see the Label on the left and the Combo Box on the right. They were not placed exactly where we wanted, so we can drag and drop the Combo Box into the existing layout. The Label will follow automatically:



We can now add one more Combo Box for the column Mentor. The process is exactly the same, but this time we select the table Artist, we store the value in the column Mentor, and we can also set the Label to "Mentor". Here is the form with both Combo Boxes in place:



We can now switch to the Form View and see how our form works. Browse through the records. Notice that any change you make here is immediately stored in the table (as soon as you leave the current record). So if you want to add a new record, you must first move to a new record and then input the new values. Otherwise you will be changing an existing record in the table. We can at this point also save our new form as "Artist":



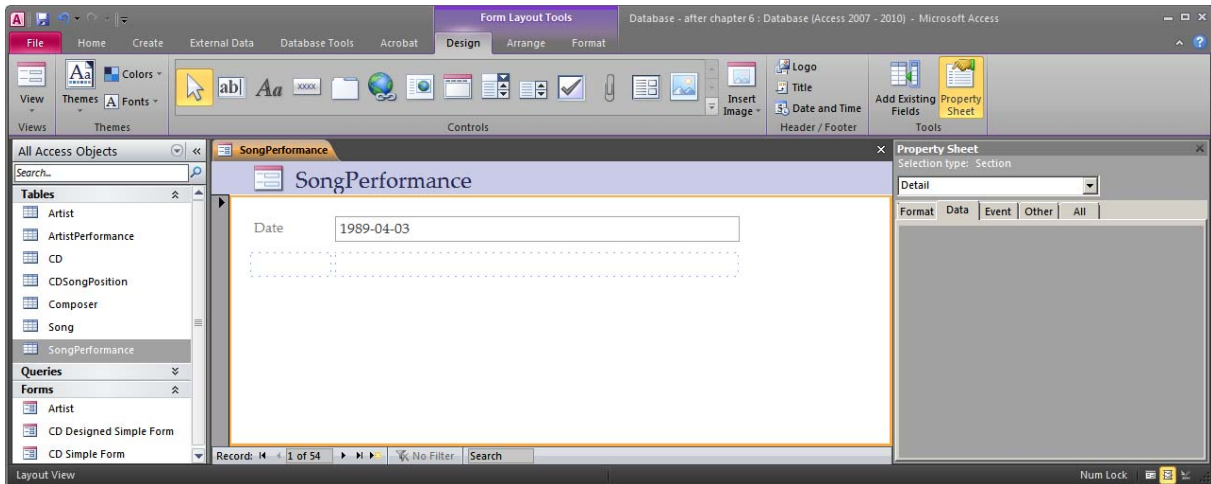
- ① Queries and tables may not have the same name, but a form can without problem have the same name as a table or query.

We will modify this form a little bit later to add the business rule restriction: "The mentor must be older than the artist". We will do this in section 9.15.

What we saw above is a very simple case where the primary key of the referenced table was what the user saw in each Combo Box. But there is also the possibility that a foreign key references a surrogate key that is not so useful for the user to see in a combo box. This can be illustrated with another form required in our case: *A form for registering song performances and artists performing them.*

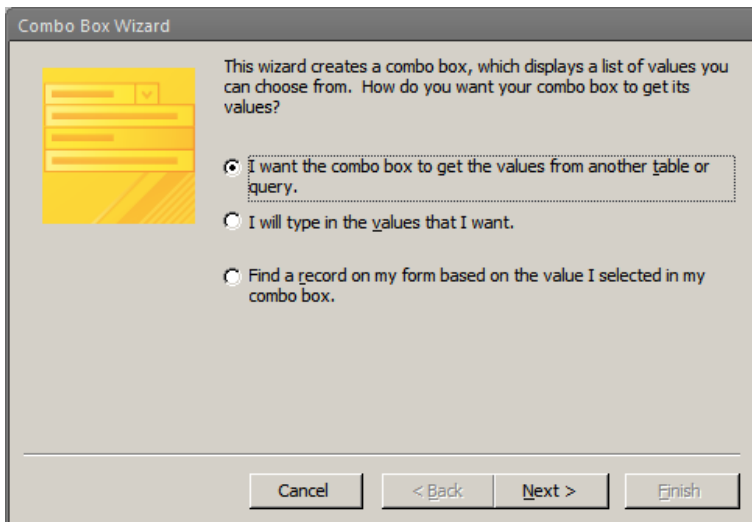
We can start by making a form for only the first half of the sentence above: *A form for registering song performances.* In this case the form will be able to register a new row in the table SongPerformance. This table has a foreign key (column Song) that references the primary key of the table Song (column ID). But it would be useless for a user to see a bunch of ID values in a combo box. We will therefore show in the combo box the name and the composer of the song, and let Access link the ID values.

We start by creating a new form for the table SongPerformance. We can use an AutoForm to get a form quickly. We can then remove the Song Text Box so that we can add a combo box instead. Our form should now look like this (before adding the Combo Box):

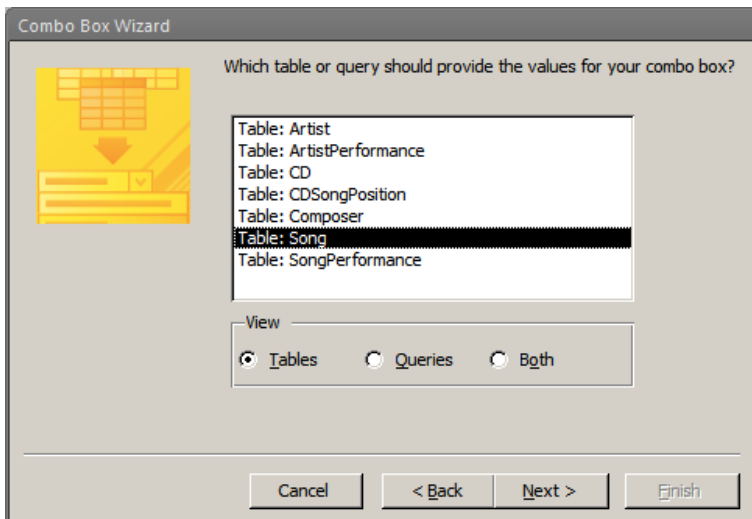


We can now add a Combo Box under the Date field and go through the wizard.

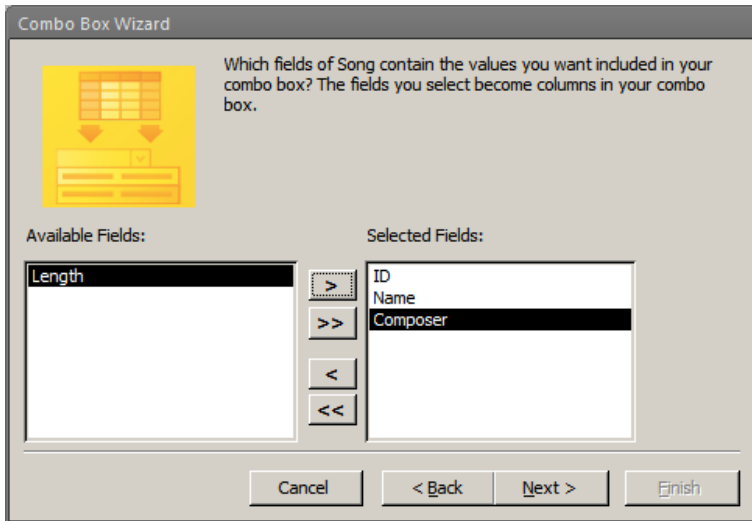
First we say that we want to look up the values in a table:



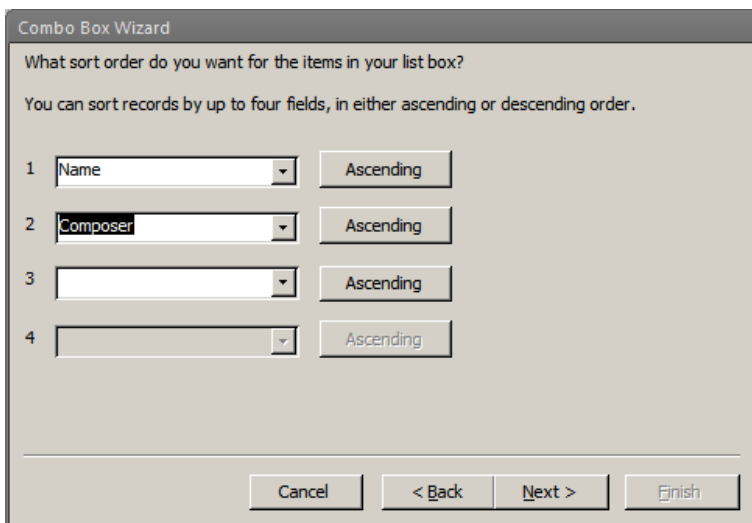
Then we specify the table Song as the value provider:



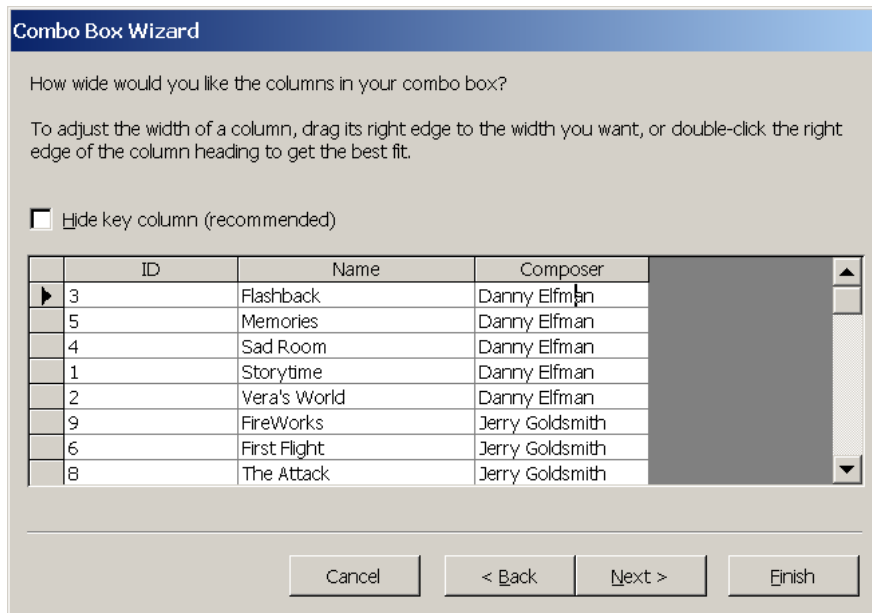
Next we have to say which fields we want to include in our combo box. It is important to take both the columns that we want to show to the user, and the columns that are referenced by the foreign key. This means that we have to take the columns Name, Composer and ID:



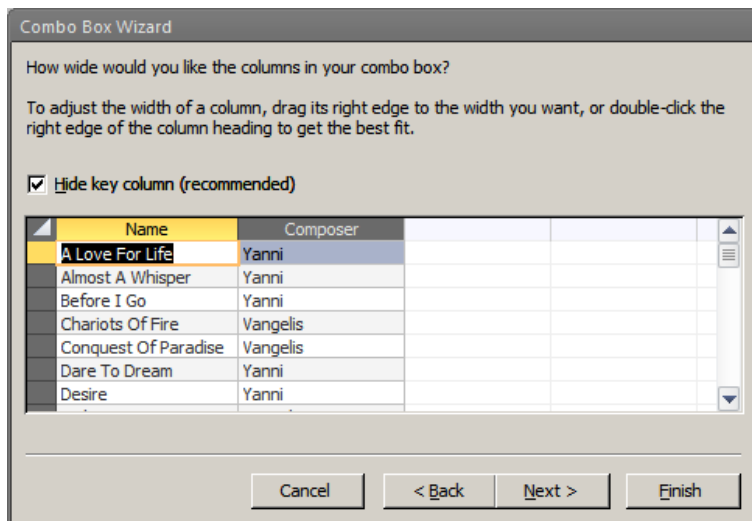
Next we can decide the order. For example by name and then by composer (when more songs have the same name):



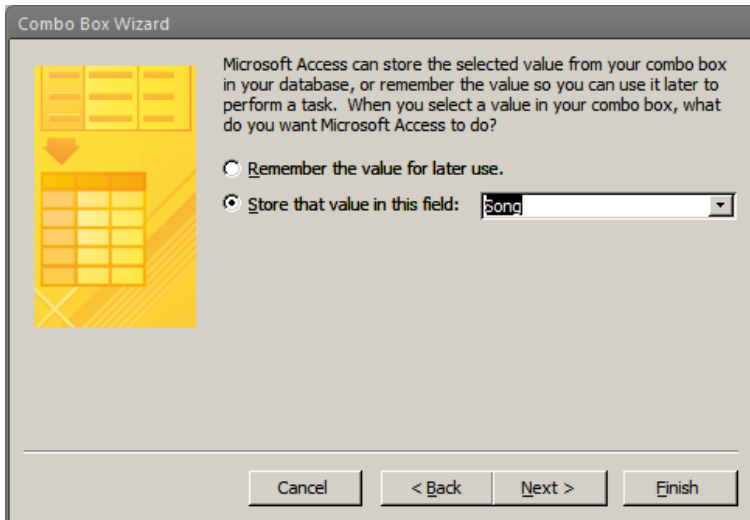
The next step now provides an extra possibility, namely to hide the primary key column. If we unhide the primary key then all three columns will be visible in the combo box.



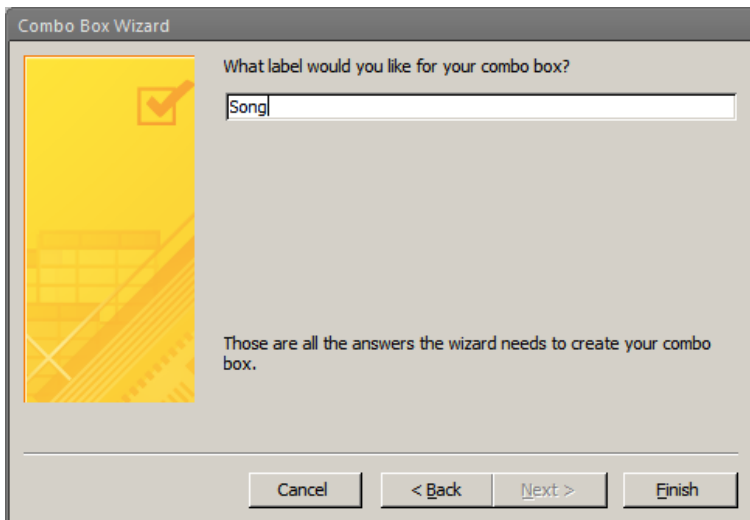
We can also rearrange the order and width of the columns if necessary. But we leave the ID column first and hidden, because the wizard may get confused otherwise:



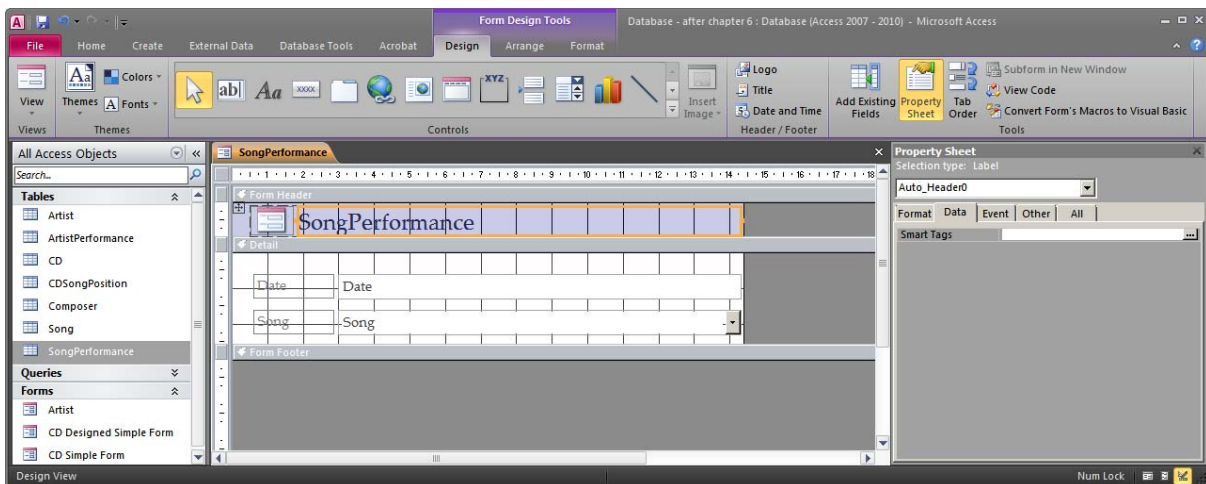
Since the ID column is a surrogate key for the table Song, it is best to hide it from the user. Even though the column ID is not visible to the user, Access can still place the ID value in the applicable column of the table SongPerformance. Choose therefore to store the value of the Combo Box in the column Song (which is the foreign key). Access will automatically place the ID value of the selected song in the column Song of the current row of the table SongPerformance.



Press next to specify the label for the new combo box:

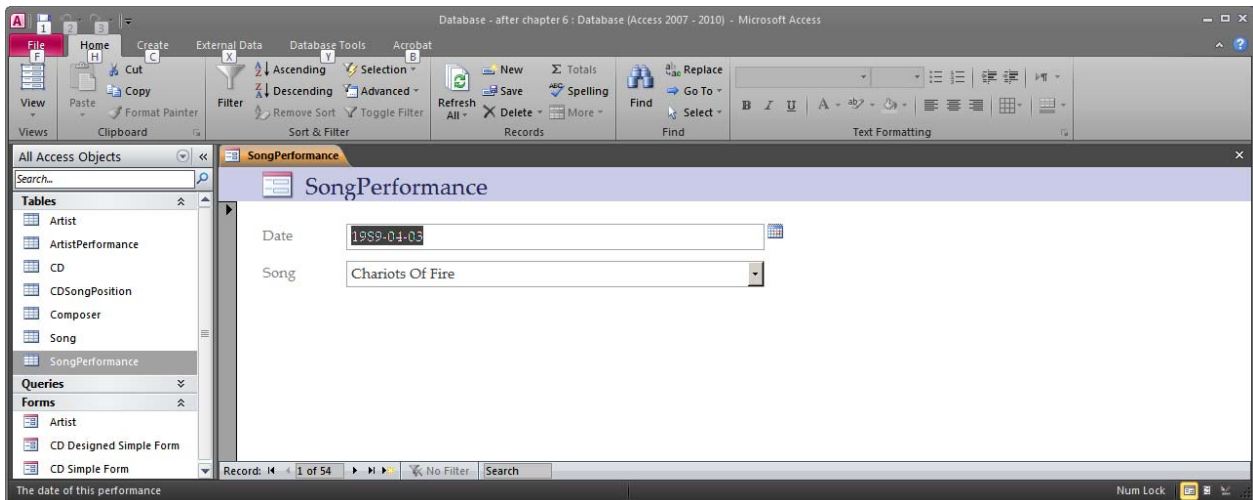


Press Finish to return to the form design view where we can modify the position and size of our new combo box and label:



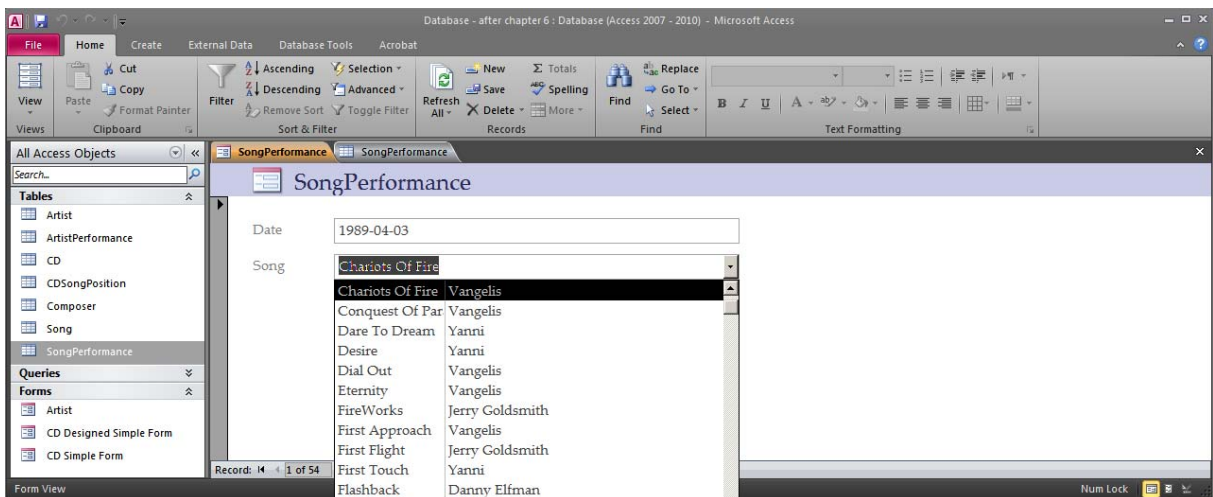


We can now switch to the Form View and browse through the records to see how the Combo Box works:



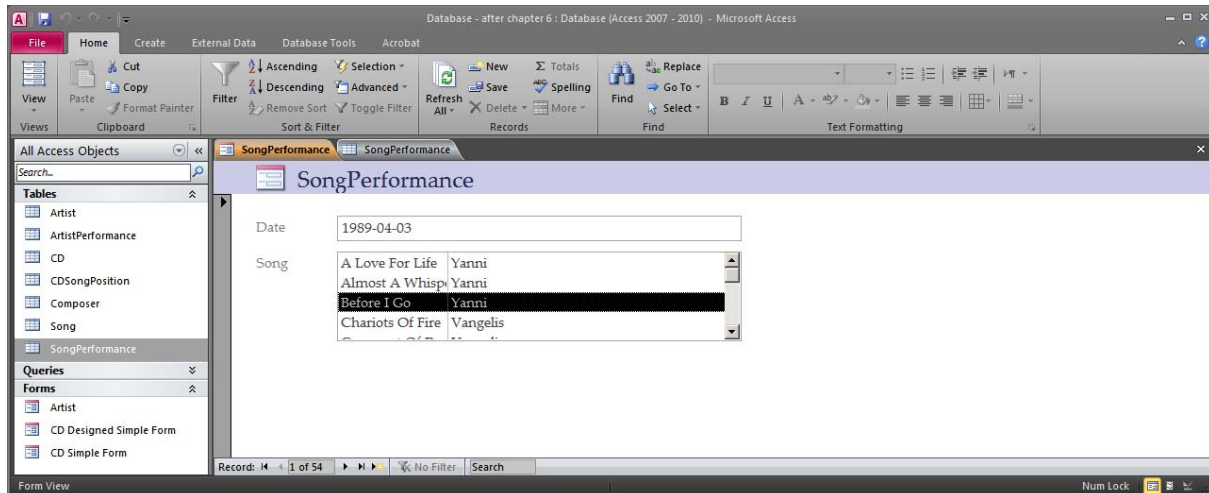
Access has also automatically added a little icon next to the Text Box for the Date column. This icon allows us to choose a date from a special pop-up calendar. Access added this because the column was defined to be a Short Date. This so called "Date Picker" can be configured through the property "Show Date Picker".

Sadly Access only shows the first visible column when the combo box is closed, while it shows all visible columns when the combo box is open:



There are some ways to fix this. One simple solution is to transform the Combo Box to a List Box. This can easily be achieved by right-clicking on the Combo Box control (while in Design View) and selecting Change To → List Box. Access will then transform the Combo Box into a List Box while preserving all other settings of the control. If you open the form now it should look much better:

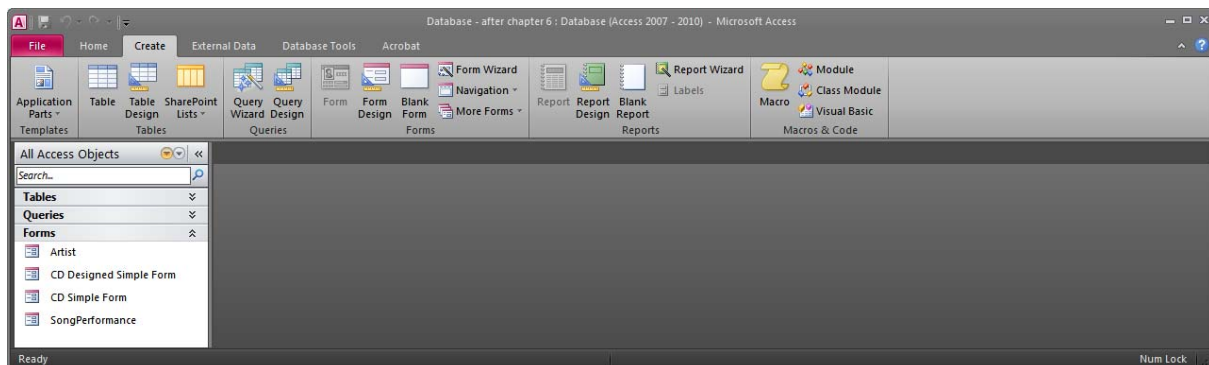




It is also possible to keep the combo box style and make sure that both the composer's name and song's name show. This can be done by specifying a query as the source of the Combo Box, which is described partly in section 6.4 and in section 9.7.

The width of the individual columns inside the List Box or Combo Box is configured in the property Column Widths. The properties Column Count, Column Widths, Row Source, Control Source and Bound Column must of course not contradict each other.

We can save our form as "SongPerformance". We have now 4 forms:



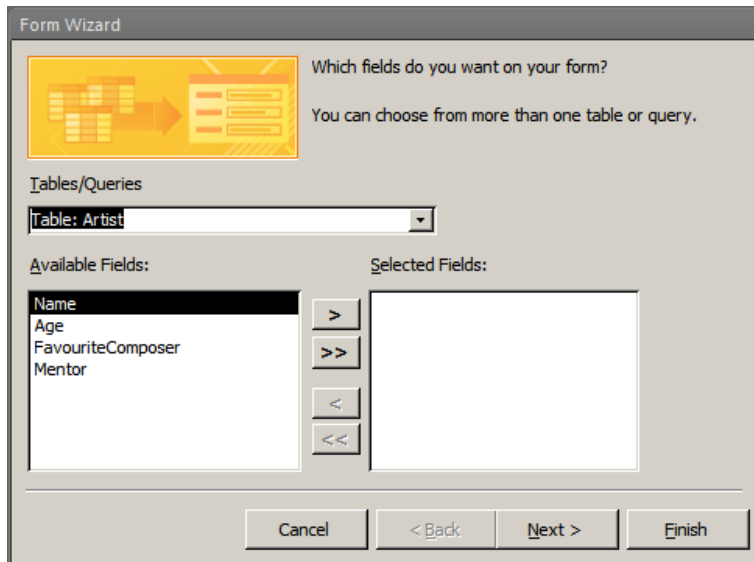
This technique of using Combo Boxes or List Boxes is quite useful, but it has some limitations. The main problem is that it cannot manage composite keys. For example, if we were to try to make an ArtistPerformance form with a Combo Box for selecting a SongPerformance then we would reach a dead-end because there is no way to tell Access to link both the column Date and the column Song. There are several ways to work around this problem. One way is by using a master-detail form (presented in the next section), another is to use a macro (presented in chapter 8).

### 6.3 Master-Detail Constructs

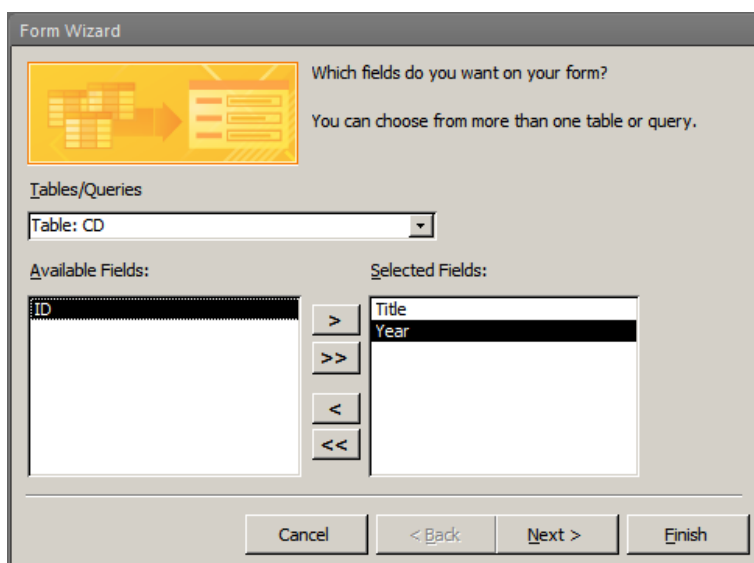
Sometimes it is useful to create forms that contain other forms. The content of the subform is then dependent on the current row of the main form. This kind of structure is very common and not at all difficult to create. The form Access created automatically in section 6.1 used such a construct.

In our case in chapter 2 we had the following information need: *Show all songs in a particular CD*. We could of course solve this as an SQL statement, but we could also create a form for this. Access has a form wizard that can produce a master-detail form structure for this kind of scenarios. We will look at the wizard way first, and then we will look at how we can make the same structure manually.

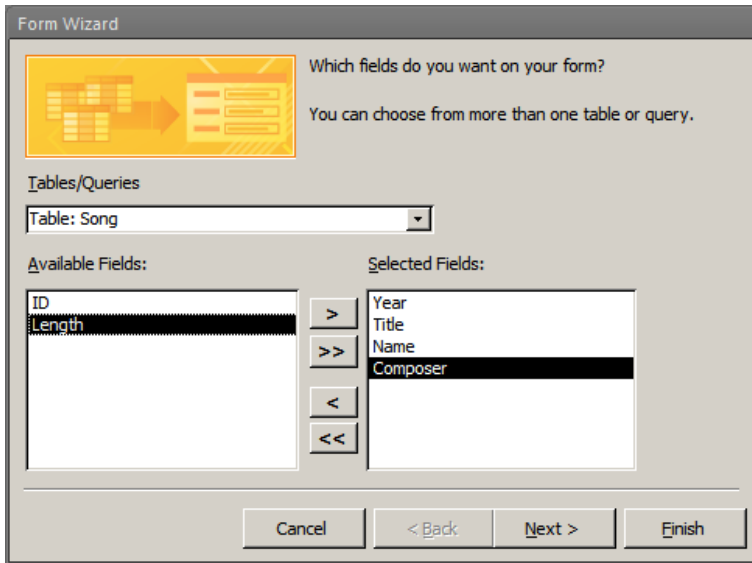
From the Create tab of the ribbon, choose "Form Wizard". The Form Wizard will then appear to guide us through the process of defining our form:



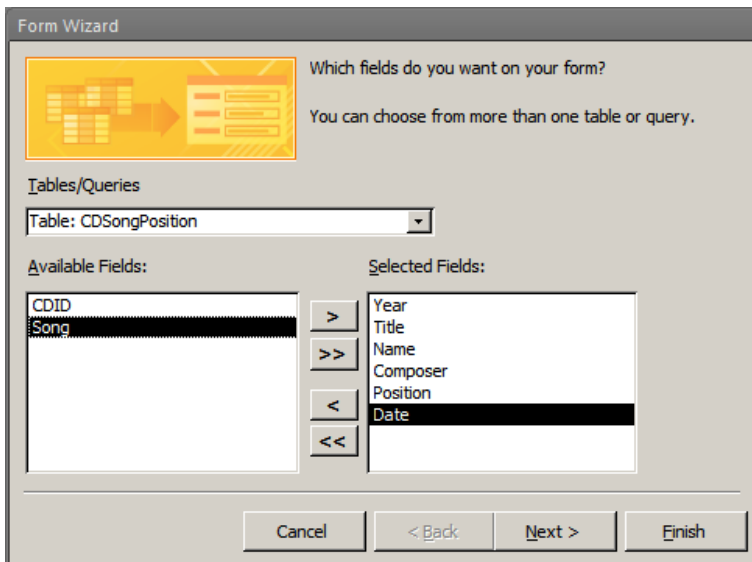
In this first step we must specify which columns that we are interested in having in the form. In our case we want to see the CD's title and year and the song names with their composer's name and the performance date. We shall show the songs in the correct order (as they appear on the CD). We must therefore specify in the wizard that we want to include these columns. We can start by selecting the table CD and then adding the columns Title and Year:



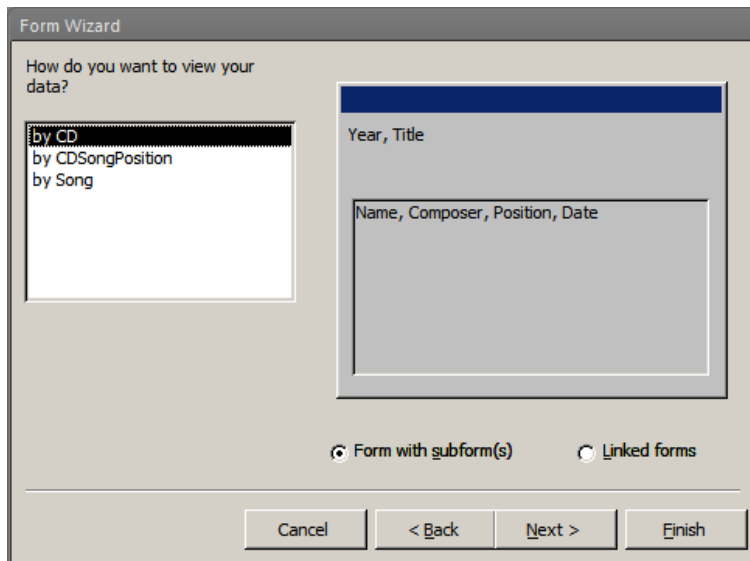
Then we can select the table Song and add the columns Name and Composer:



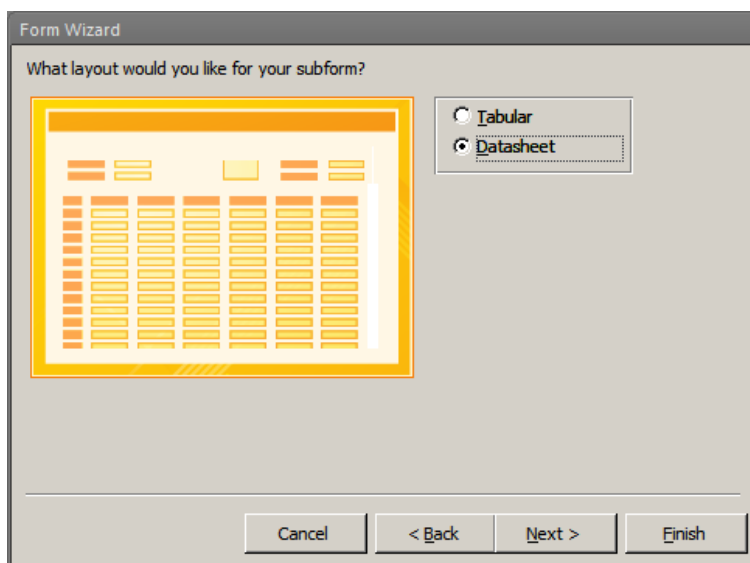
And we can also add the columns position and date from the table CDSongPosition:



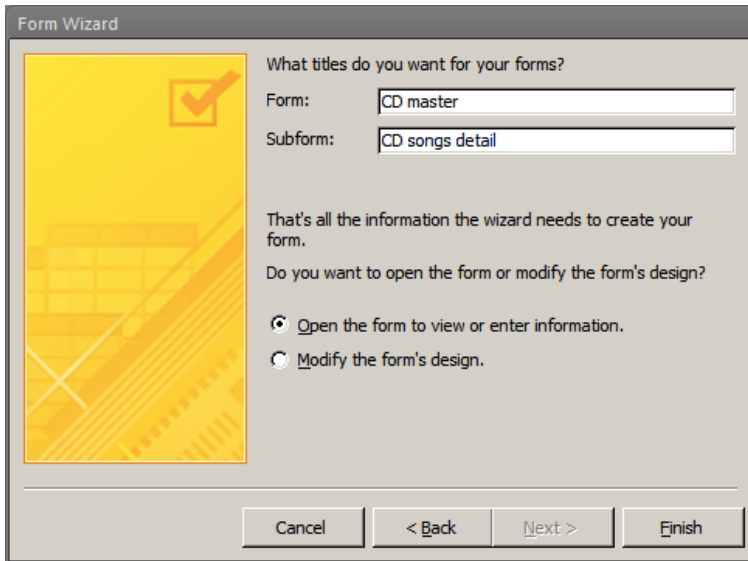
We can press Next, and Access will give us some suggestions of how to organize the selected columns. This is done based on the relationships that we have defined in our database (in section 4.2). The alternative that fits our needs is the first one ("by CD") and we want to realize this as "Form with subforms":



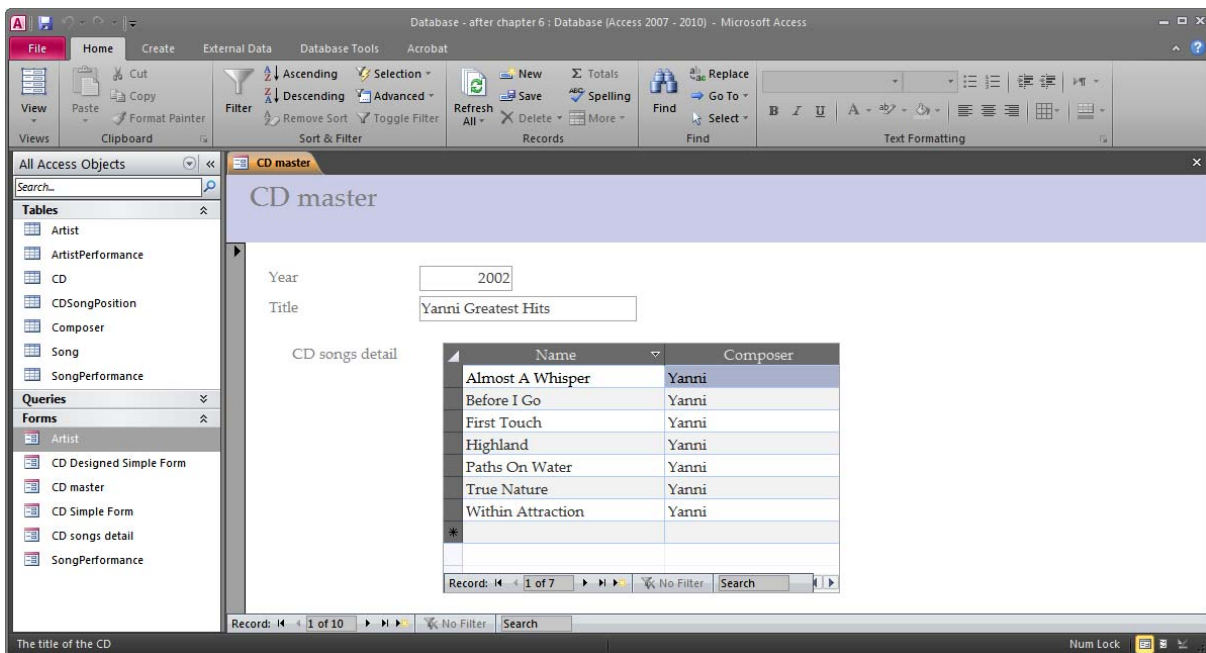
If Access does not suggest the structure we want, we can still achieve it, but it will have to be done manually as we will see later in this section. In the next step we can define the layout of the subform. Any of the two choices will do fine, but let's take Datasheet:



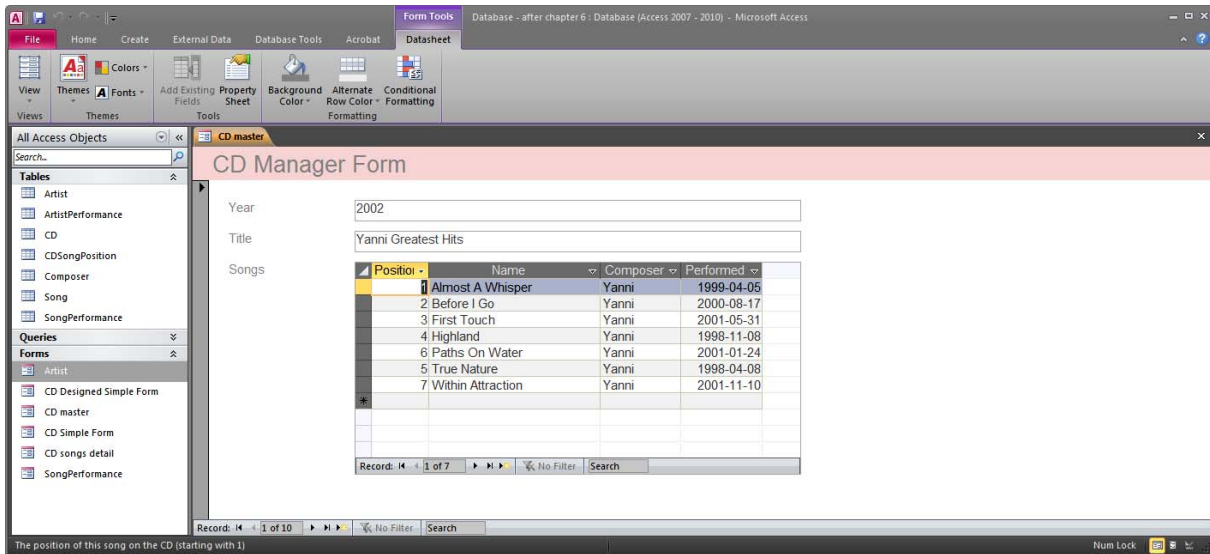
Finally we can define the names of the forms. We can call them "CD master" and "CD songs detail". We can also let Access open the form directly after we press Finish:



We can now improve the layout if necessary. The form created by the wizard may look like this:

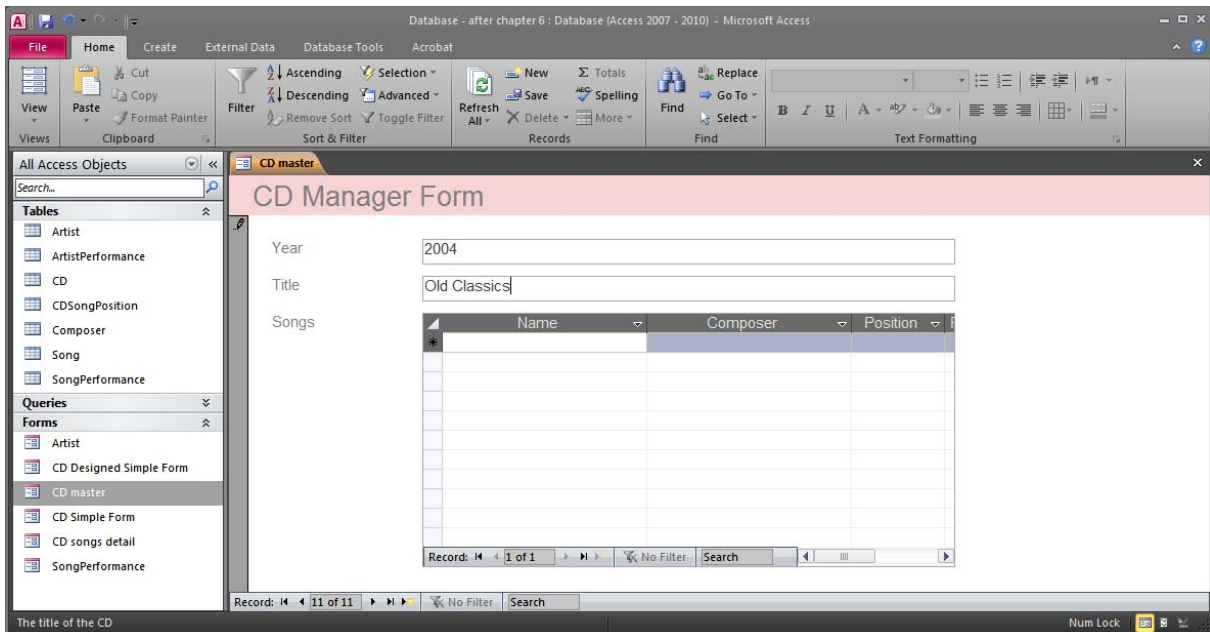


We can quickly go to the Design View to resize the subform, and change and move the Label, etc. The subform is actually just another form control of the main form. Access also allows us to edit the subform directly here, but we can also open the subform separately if we want. The subform is now bigger and with better column widths so the layout is improved:



We can see here that there are two sets of record navigation buttons. The lower one is for browsing CDs, while the other one is for moving among songs.

This kind of form has some limitations as well. For example we cannot add or change the songs of a CD. This is because not all the required fields are available on the form. But we can use the form for adding new CDs in the table CD:



Adding songs in this CD requires that we add rows in the table CDSongPosition, and to do that we have to identify a particular SongPerformance, and this form has not been designed to do this.

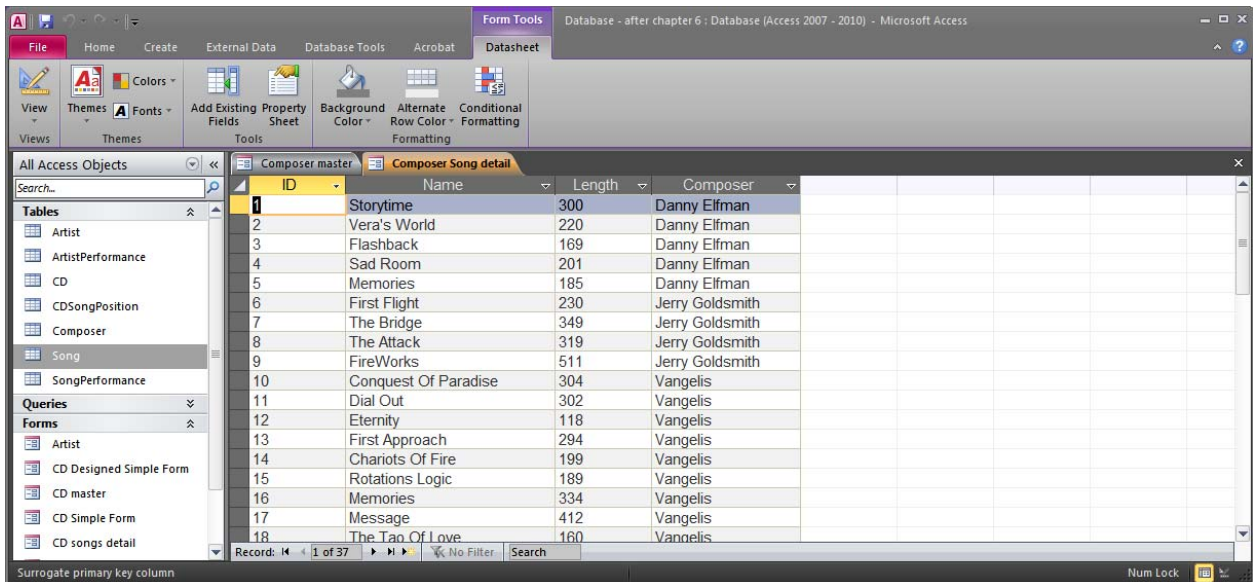
Creating the same form manually (no wizard) can be a little more time consuming, but can help you in understanding how the master form and the detail form are connected. Since the subform in this case is a form based on a query we will look at this example in the next section. In this section we will look at a simpler case of a master-detail form structure.



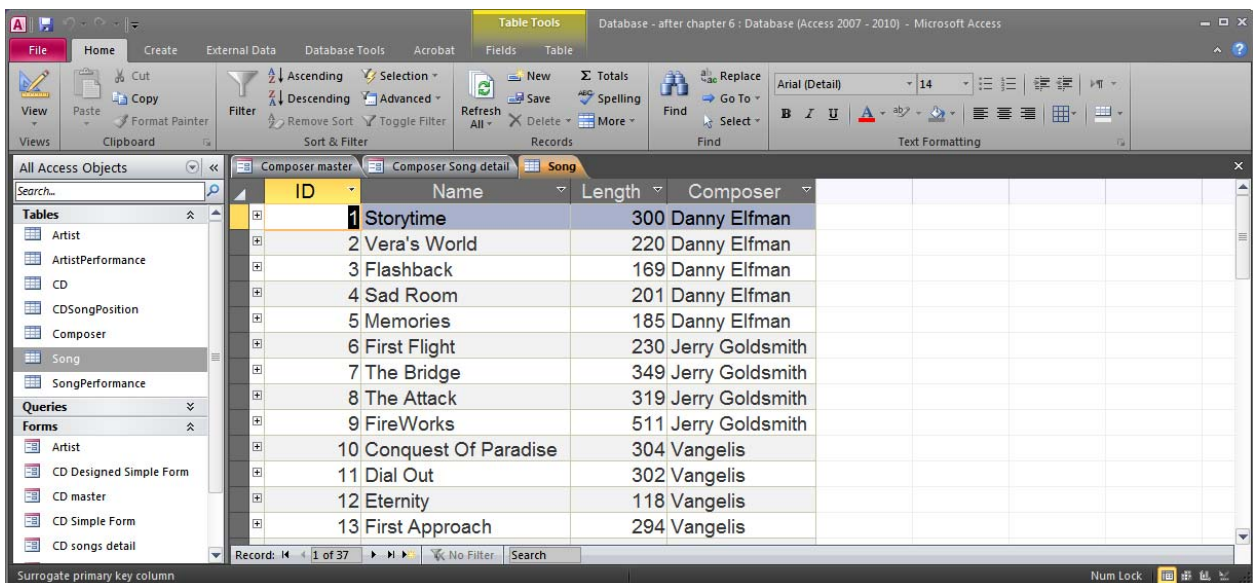
In our case in chapter 2 we asked the following: *Which songs has each composer composed?* This can of course be solved as a query, but we will try to make a form with a subform. The main form will allow us to browse through the composers in our database, while the subform will show the songs composed by the selected composer. We can start by making two forms (independent of each other) and then connecting them.

The first form is a very simple form for the table Composer. You can create this by selecting the table Composer in the object browser and press Create > Form on the ribbon. Save it as "Composer master".

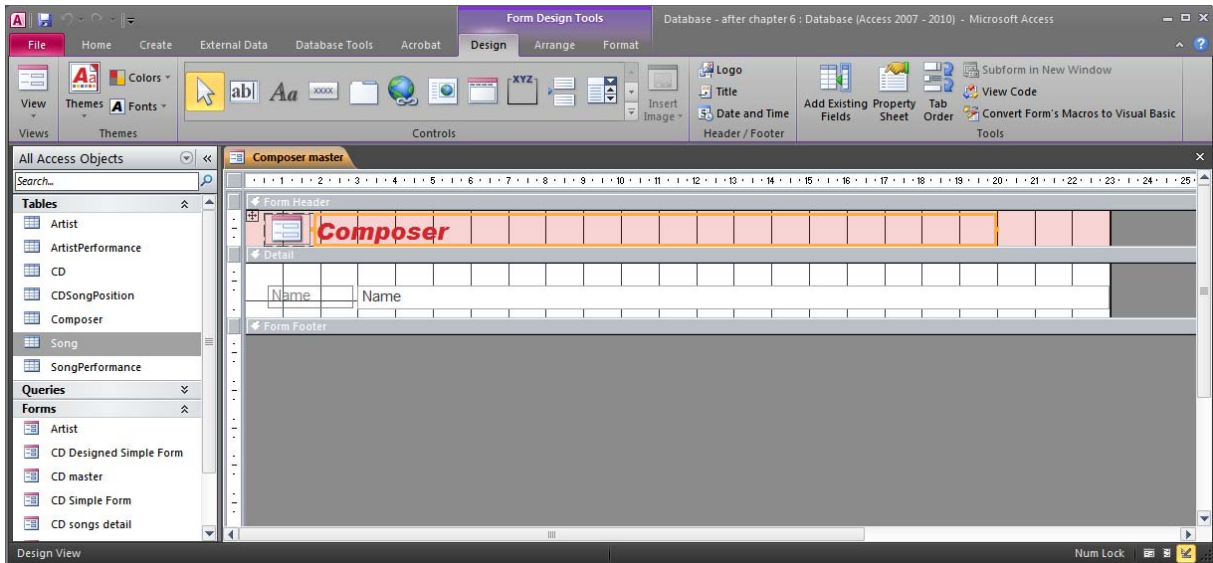
The second form is also a simple form for the table Song. You can create this as a Datasheet form by selecting the table in the object browser and then pressing Create > More Forms > DataSheet on the ribbon. Save this form as "Composer Song detail". This form looks just like when opening the table:



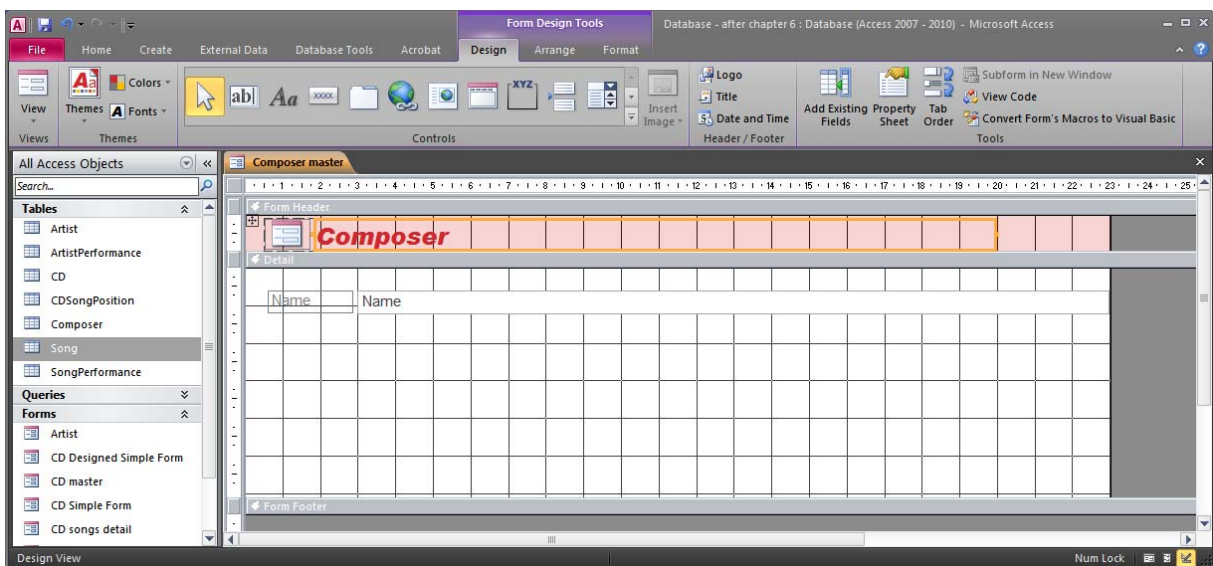
Compare it to the table:



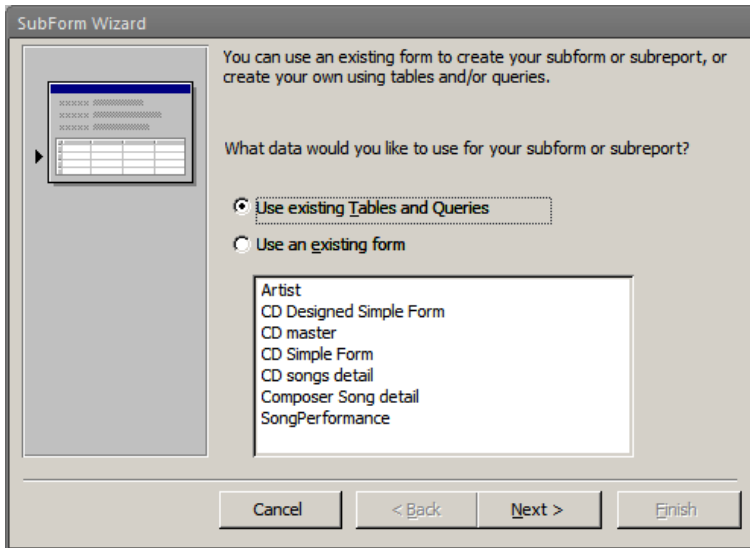
Now that we have both forms ready and saved, we can see how we can combine them. To do this we have to open the master form in design mode. It is the master form that will contain the detail form. Select the master form and open it in design mode:



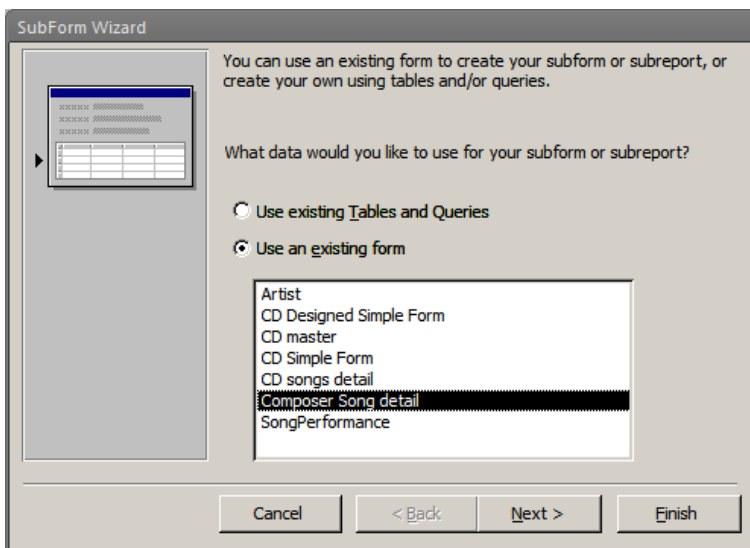
We can see that there is not so much space, so we have to make more space. Use the mouse to expand the "Detail" part of this form:



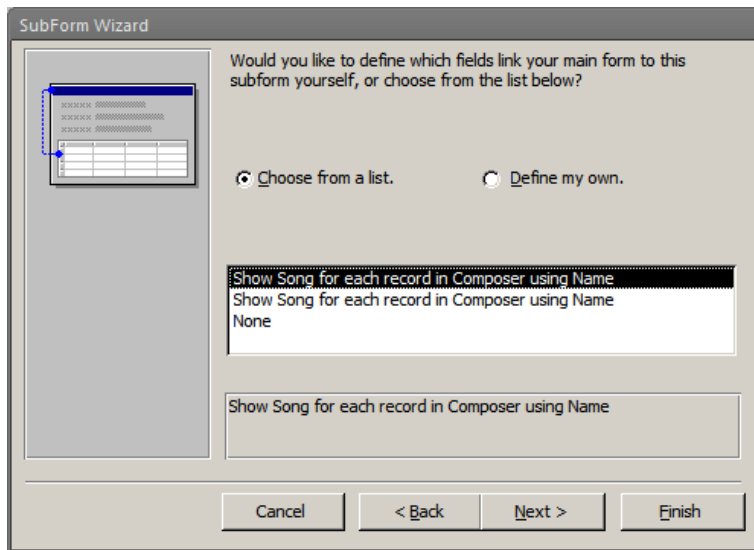
We have now enough space to add our subform. On the ribbon there is a special form control named Subform/Subreport. Make sure that the "Use Control Wizards" option is pressed and click to activate the Subform/Subreport button ( ). Now click on the form at the place you want to have the subform. The Subform Wizard will then appear:



In the first step we are asked to select either a table or query, or an existing form. If we select a table or query then the wizard will create a form for that table or query. Since we have already created a form we can select that instead:

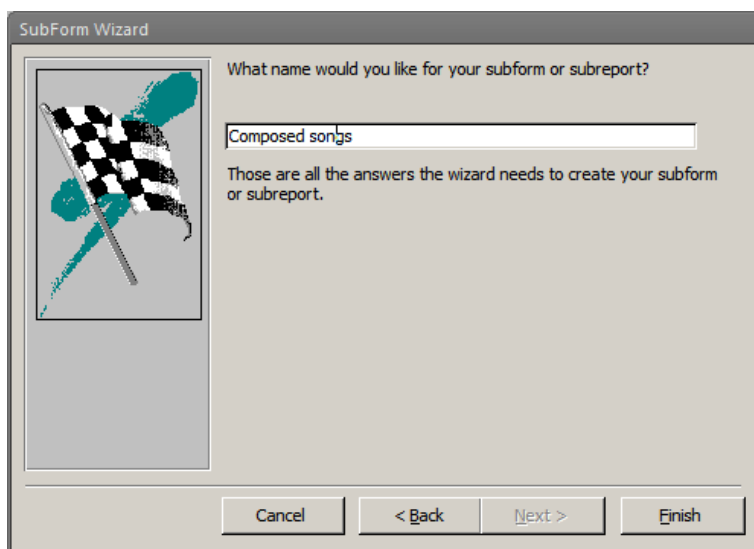


In the next step we can select the option "Show Song for each record in Composer using Name" (which for some strange reason appears twice):

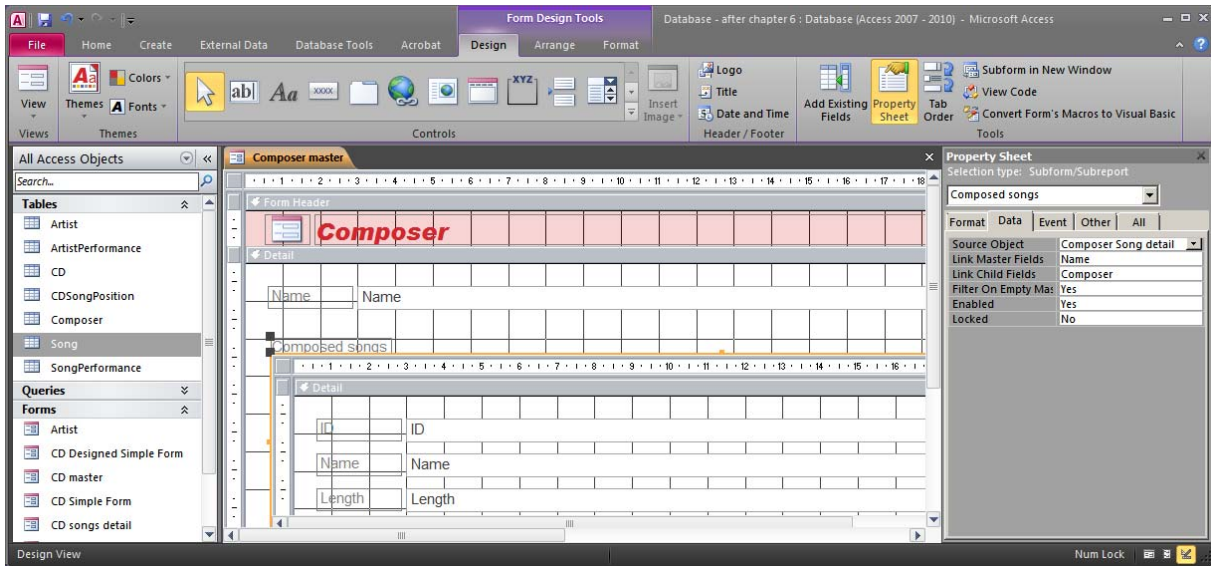


- ① The options available here are based on the relationships between tables that we defined in section 4.2. If the option you were expecting is not available here it is most probably because that relationship has not been defined.

The final step of the wizard asks us to define the name of the subform. This name is also the text of the label created next to the subform. We can therefore give it the name "Composed songs":



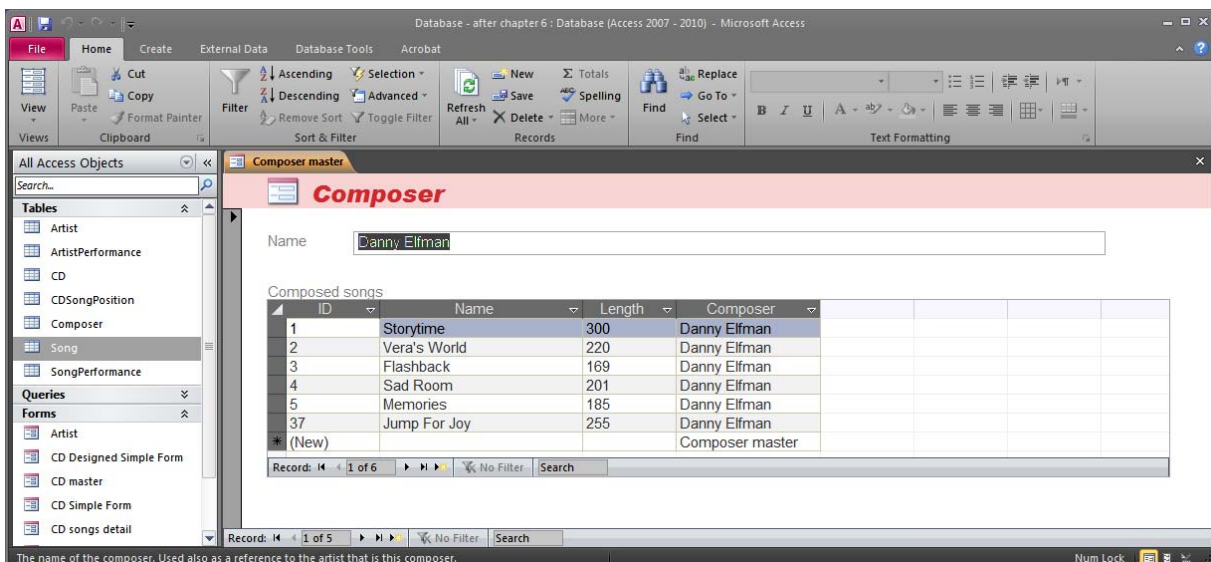
We press Finish and our subform is now linked. We can also see how the linking is done, by examining the properties of the subform in the property sheet. Select the subform component and look under "Data":



The Link Child Fields property specifies the fields of the subform to be linked (in this case the field Composer). The Link Master Fields specifies the fields of the main form to be linked (in this case the field Name). This means that every time the value of the field Name of the main form is changed, the subform will be refreshed to only show records that have the same value in the field Composer.

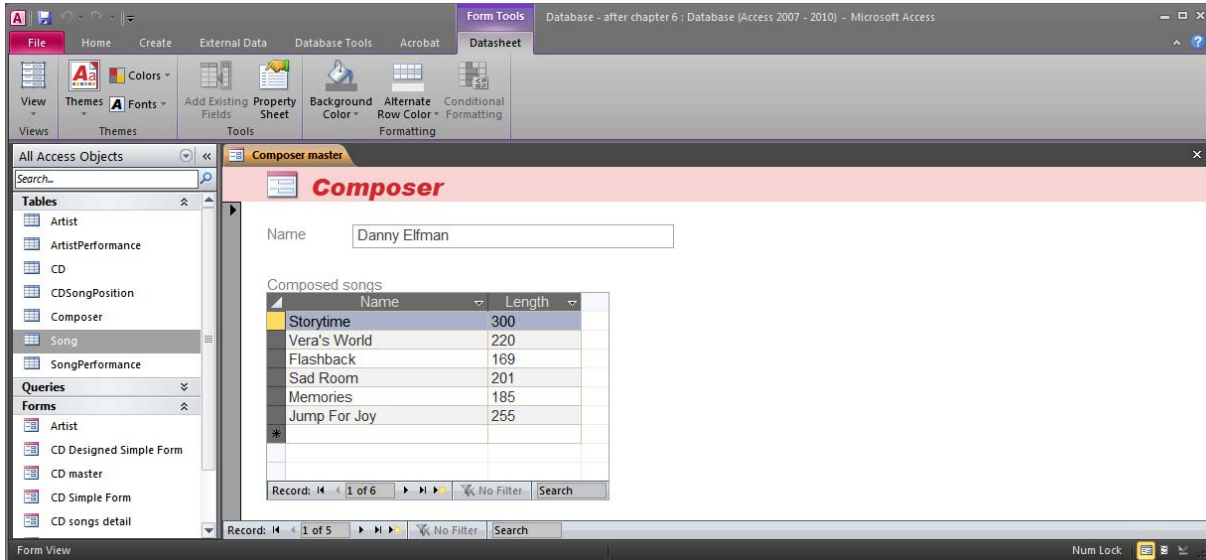
- ① If the two forms are linked with two or more columns, then they must all be specified in the Link Child Fields and Link Master Fields properties in the same order and separated by semicolon (;).

We can now view our form:

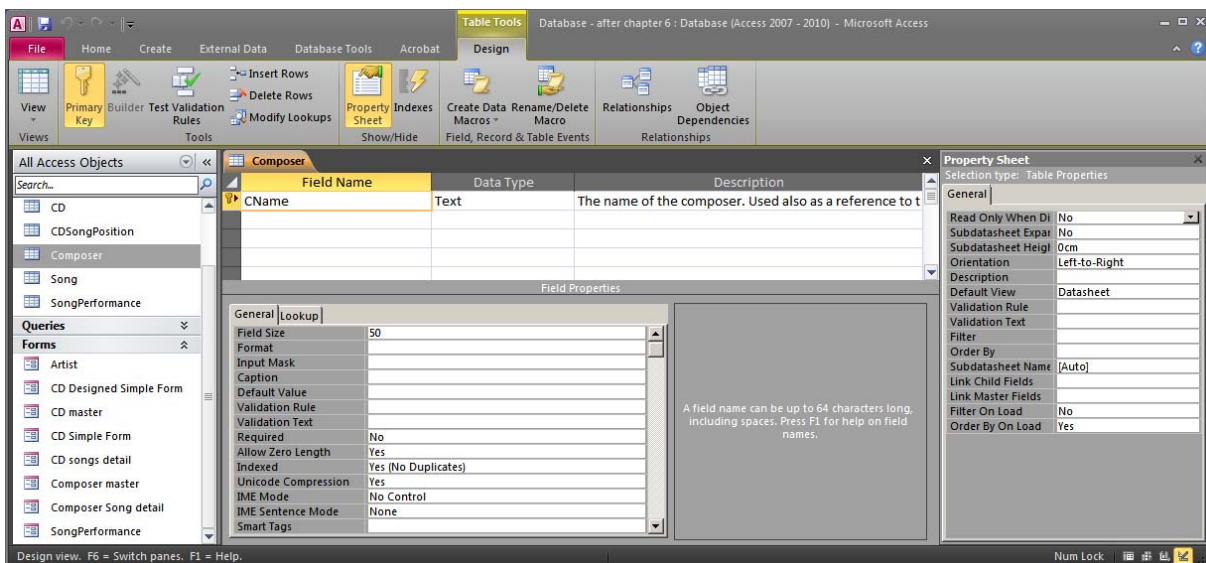


One thing that we can fix here is removing the ID column and the Composer column from the subform. The ID column is not interesting for the user, and the Composer column is always the same as the name of the current composer (in the master form). Here is the final version:



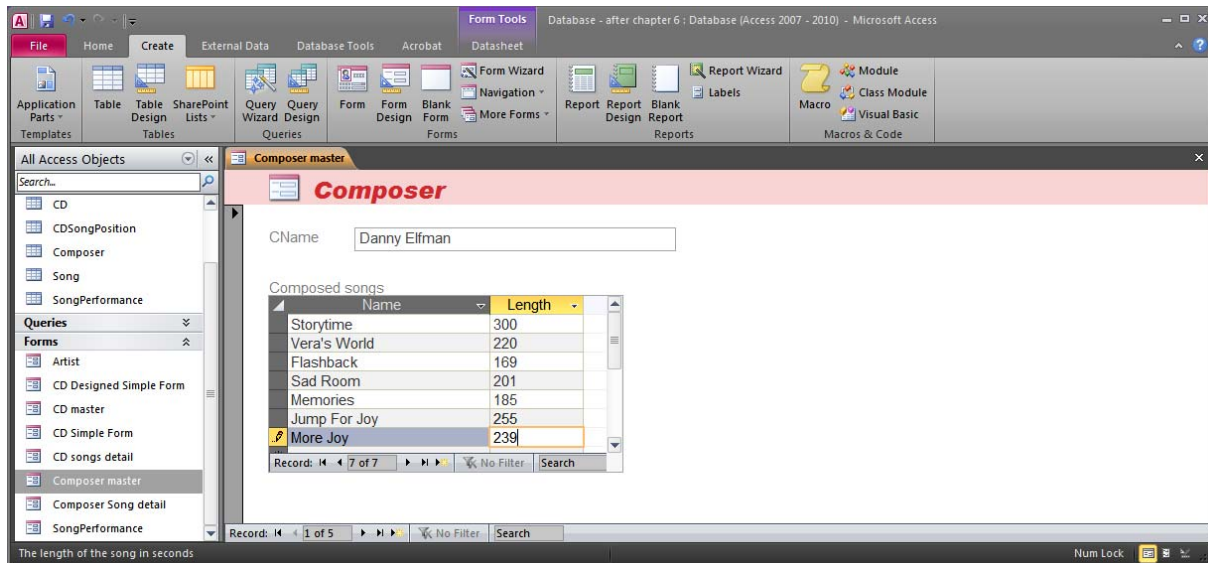


This form can actually be used for adding both composers and songs (but a composer has to already be an existing artist). There is one problem though. There is a bug in Access and the form gets confused just because the column Name in composer is called “Name”. The word “Name” is a special keyword, so when the subform refers to a field called Name Access gets confused. The only way to fix this is by changing the name of the column to something else (for example CName). Close the forms (save if necessary), open the table Composer in design view and change the column Name to CName:



Access will update all references to the column, so the form should now work fine. Try adding a new song (“More Joy” – 239 seconds) for Danny Elfman:





## 6.4 Forms Based On Queries

So far we have only created forms that are based on tables (except from the one that we created with the Form Wizard). But it can also be necessary to create forms that are based on queries. We can either let the Form Wizard create the form query or we can create the query ourselves in advance. This is what we are going to do here. We will create and save a query, and then we will create a form that is based on that query. As we mentioned previously, we will create the same structure we created in the beginning of section 6.3, but this time we will do it manually.

What we want to have is a form that allows us to browse through the rows in the table CD, and then in a subform we want to see the songs included in the selected CD. The main form will therefore be based on the table CD (nothing strange about that). The subform has to have a source that contains all the information we want to show and all the columns needed to make the connection to the current row the main form. So we need the columns Name and Composer from the table Song and then we also need the column CDID from the table CDSong position. We can do this with one of the following two SQL statements:

The first version is what you may be more used to:

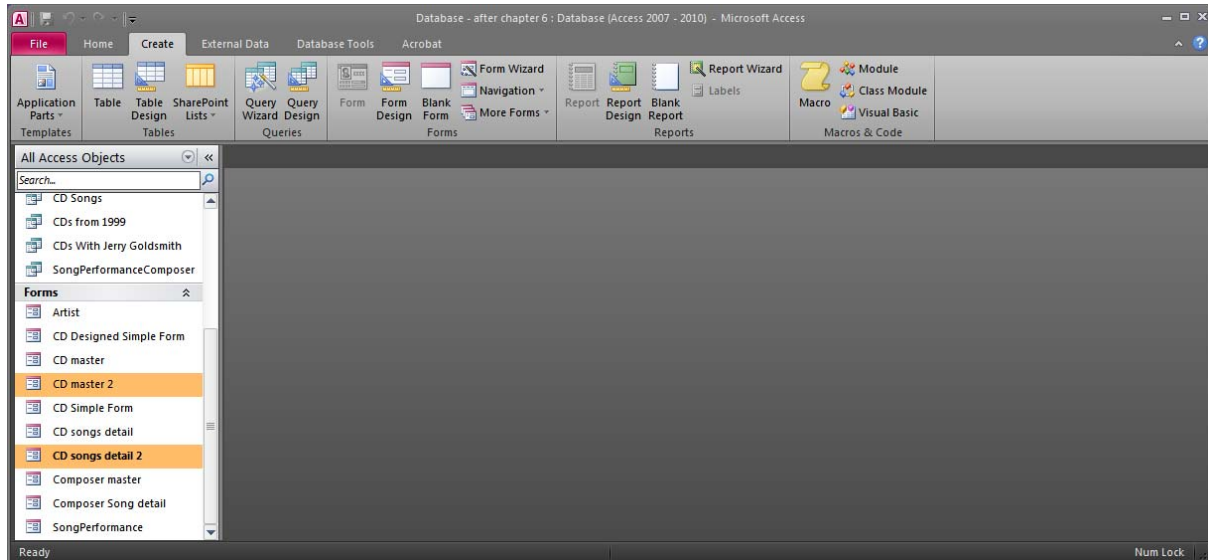
```
SELECT CDSongPosition.CDID, Song.Name, Song.Composer
FROM Song, SongPerformance, CDSongPosition
WHERE SongPerformance.Song = CDSongPosition.Song
AND SongPerformance.Date = CDSongPosition.Date
AND Song.ID = SongPerformance.Song
```

The second version is what Access generates when this is done through the Form Wizard:

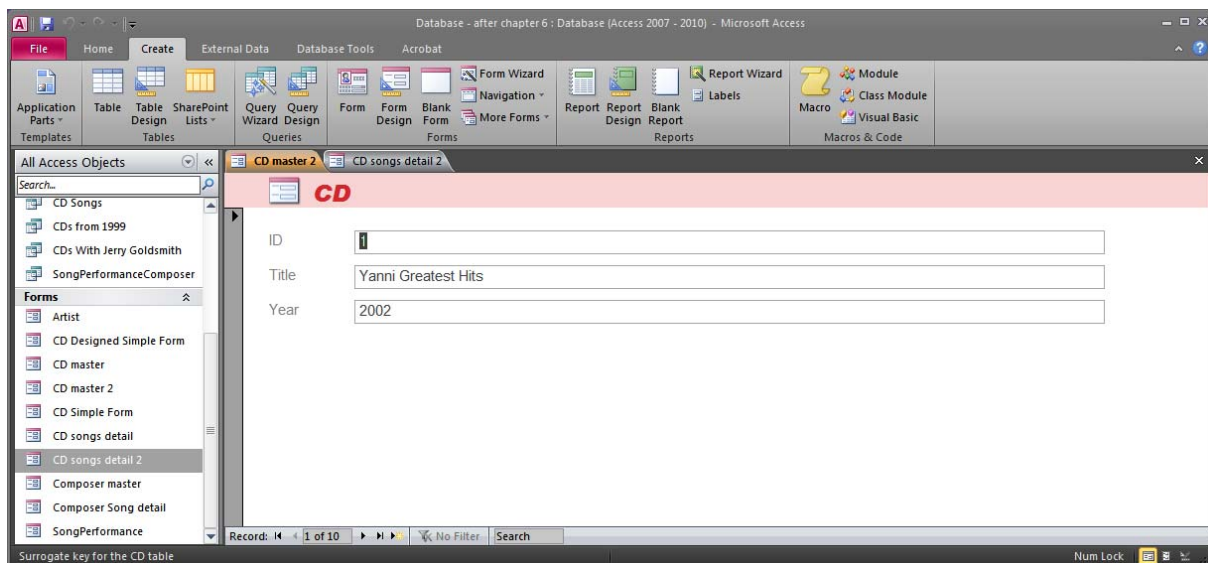
```
SELECT CDSongPosition.CDID, Song.Name, Song.Composer
FROM Song INNER JOIN (SongPerformance INNER JOIN CDSongPosition ON
(SongPerformance.Song = CDSongPosition.Song) AND (SongPerformance.Date =
CDSongPosition.Date)) ON Song.ID = SongPerformance.Song
```

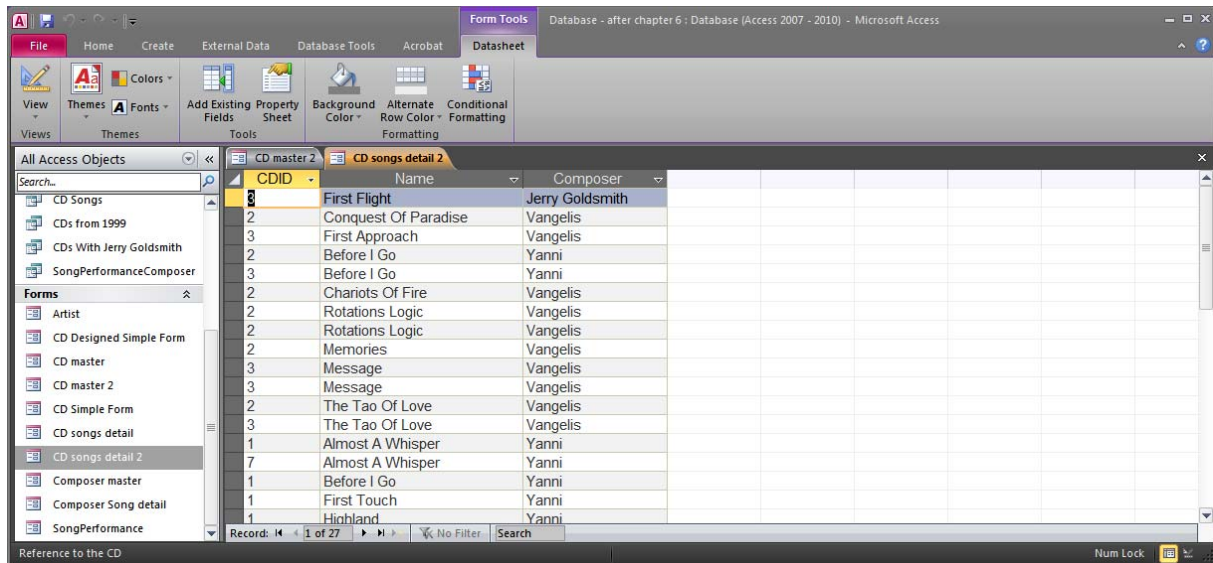
You can use whichever you like best. Create a new query, switch to the SQL mode, write the SQL statement and save the query as “CD songs”.

We can now use this query to create a form. Create a Form for the table CD and a Datasheet Form for the query CD songs. Save the forms as “CD master 2” and “CD songs detail 2” respectively. The two forms are now available in the object browser:



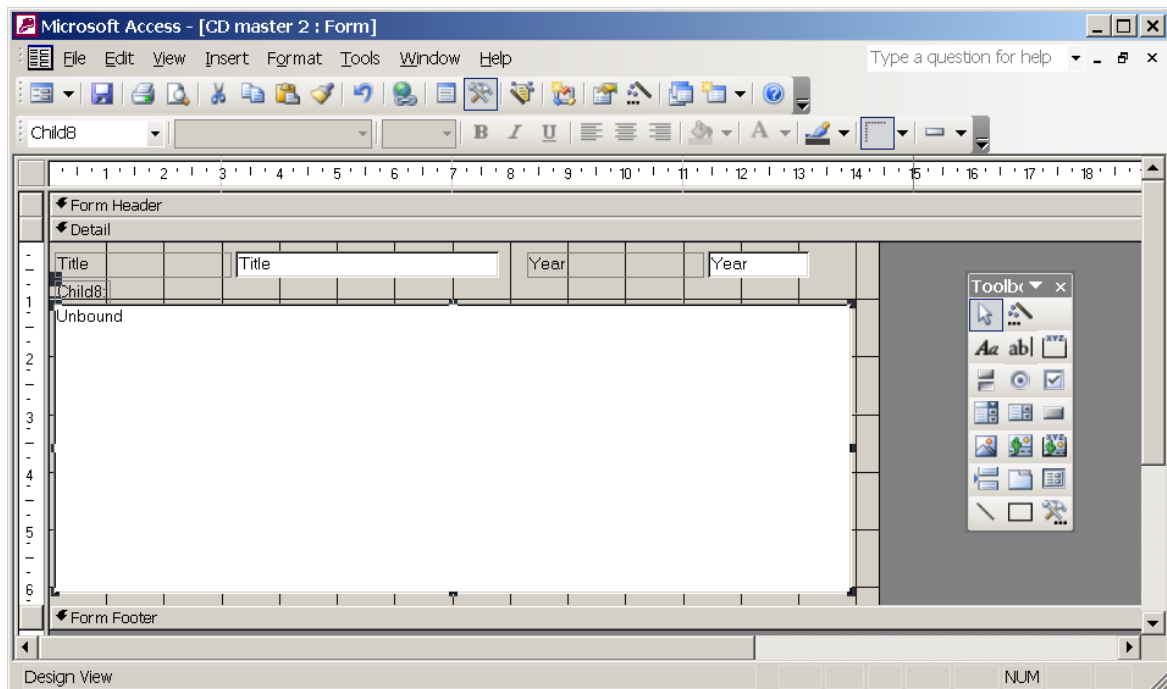
We can also open them individually and see that they work and adjust their layout if necessary:



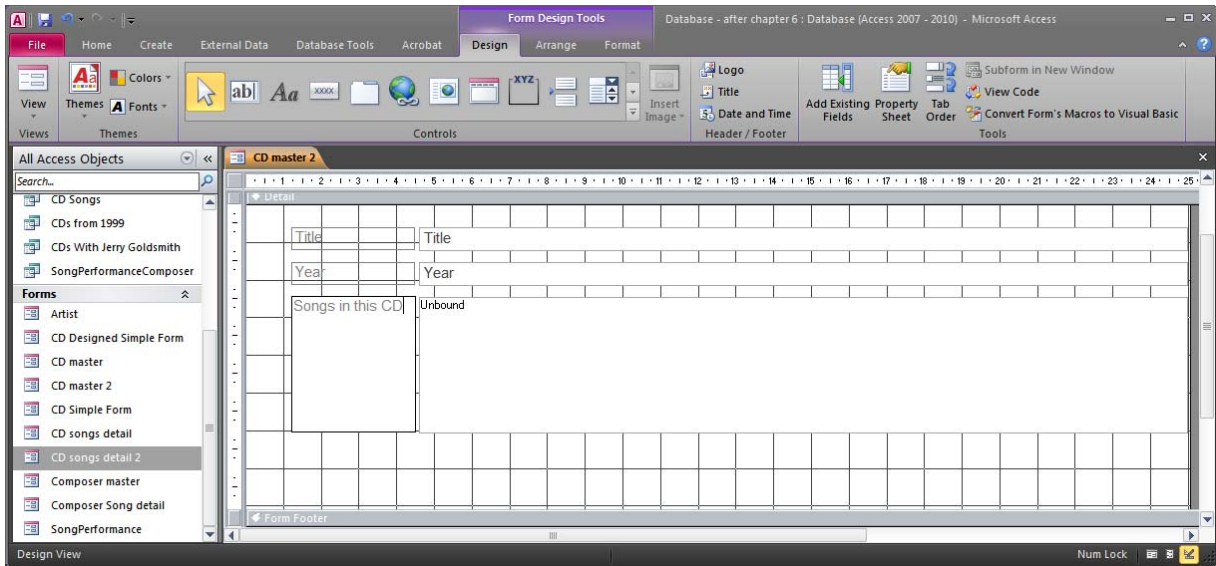


We can also see that they have some unnecessary fields. The ID and CDID fields are not relevant for showing. We need them for linking the two forms, but not for showing to the user. Edit the forms to remove them.

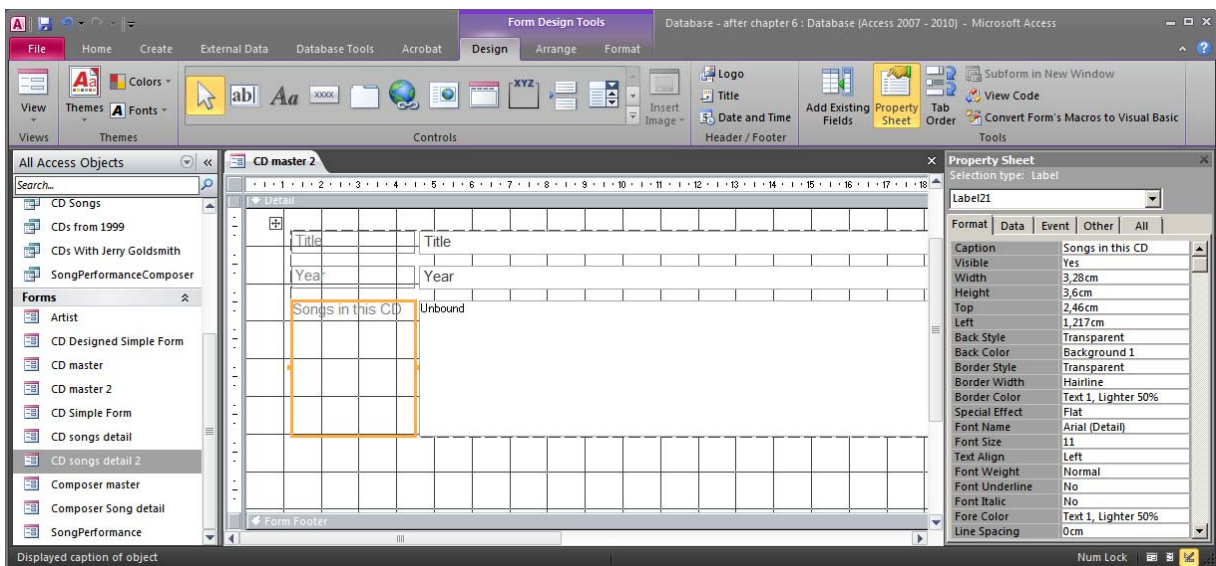
We can now do the linking. This time we will try to do it without the SubForm Wizard. We can start by opening the form “CD master 2” in the design mode. Make some space for the subform, but don’t add a subform yet. Before adding a subform, make sure to turn off the Control Wizards. We do that by pressing the Use Control Wizards button (on the ribbon) so that it is not active. We can now add a subform to our form. The wizard will not appear and we will just have an unbound subform component:



We can start by fixing the subform’s label to “Songs in this CD” and fixing the layout. We can do this directly on the form:

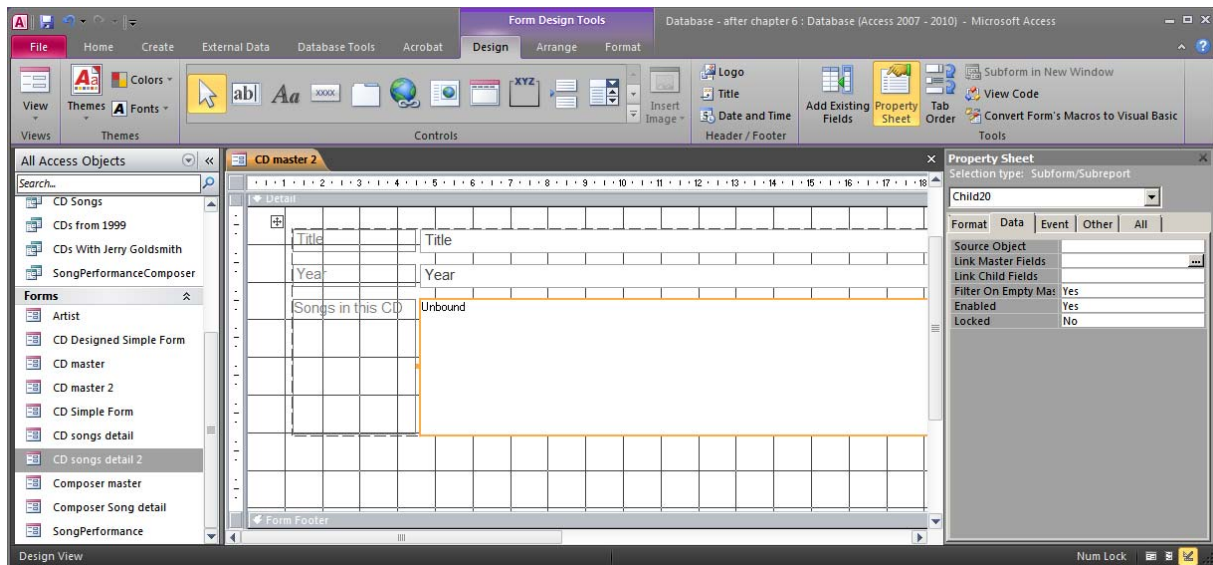


Or of course we can use the property sheet and edit the property Caption:

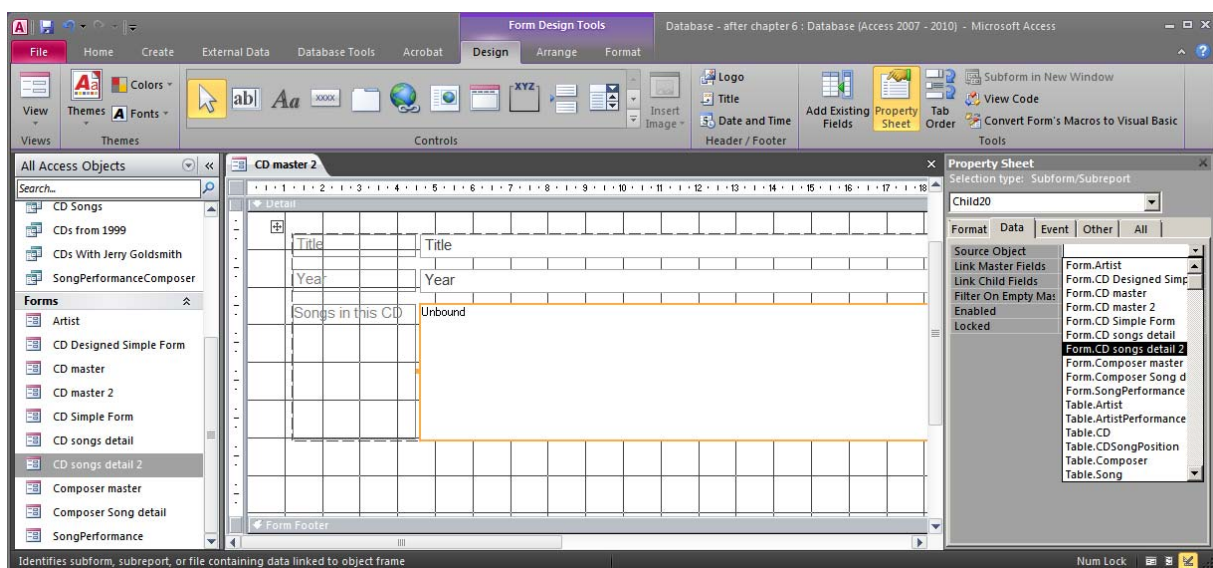


Now, select the unbound subform component and look at its properties (under Data):

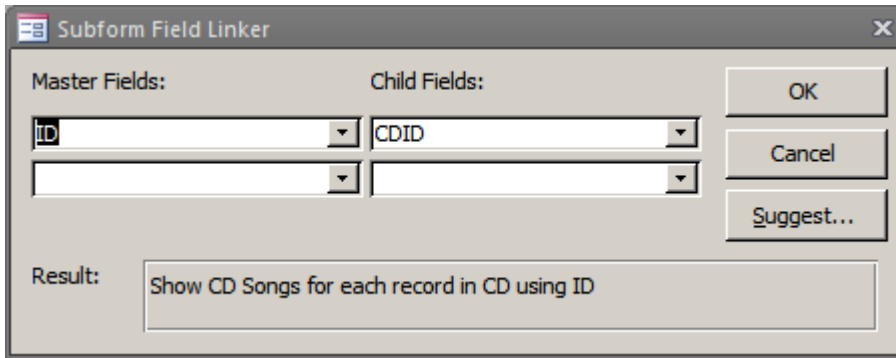




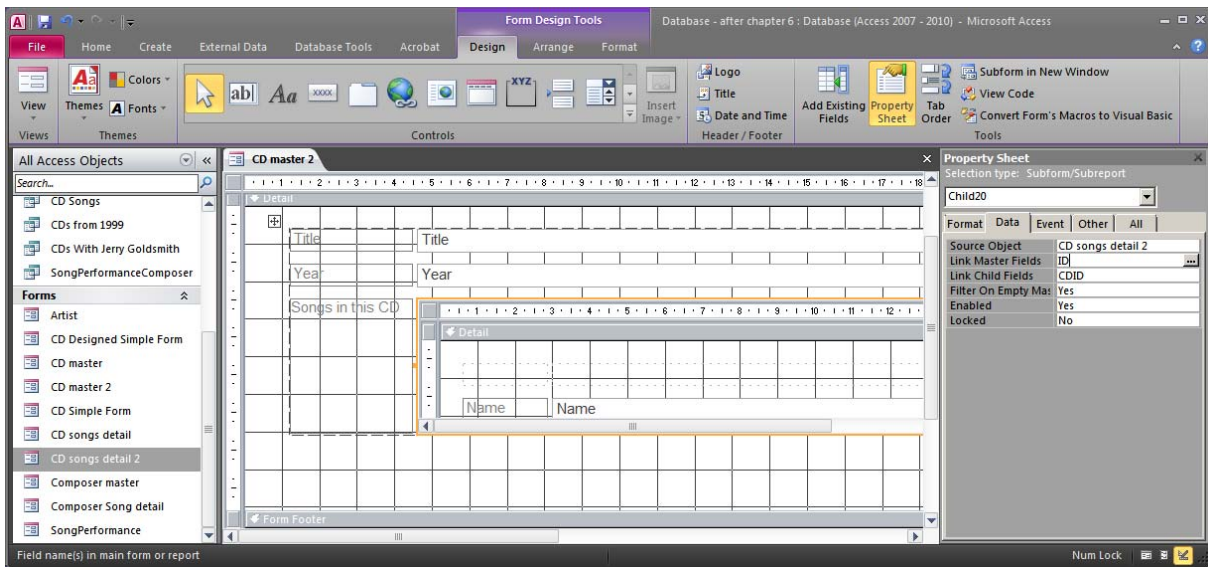
We have three properties that need to be specified before the form “CD songs detail 2” has been linked as the subform. The first one is the property Source Object. In this property we can specify which form should be used as the subform. We can simply select the “CD songs detail 2” from the drop down list:



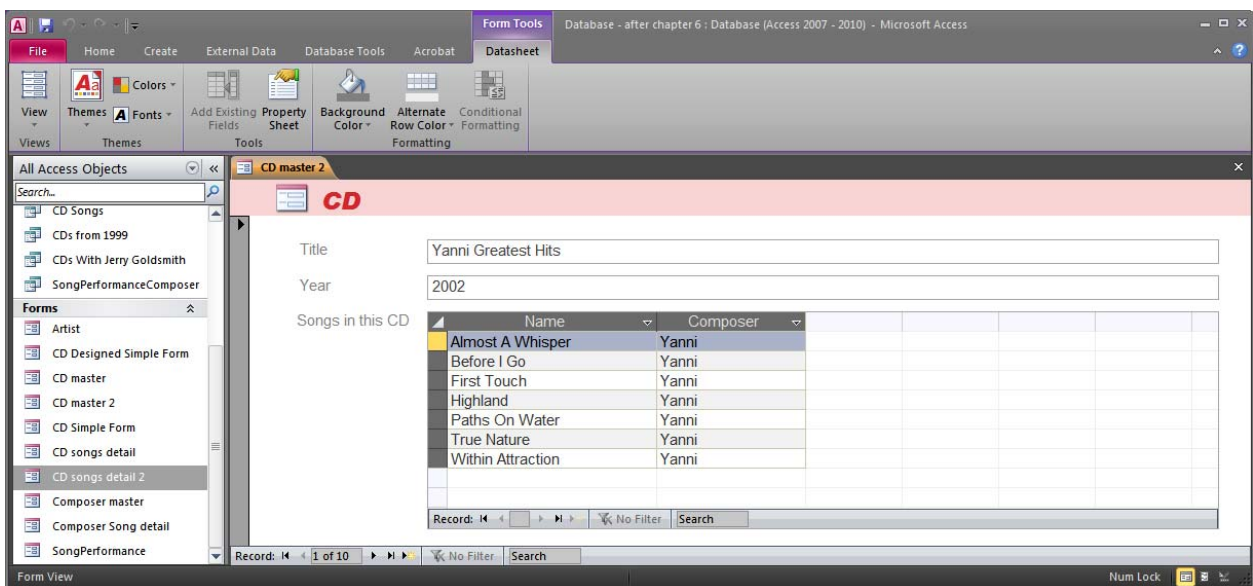
Next we must define the fields that should be used to link the main form to the subform. The Link Child Fields property must be set to the name of the column in the subform that should be linked, i.e. CDID. The Link Master Fields property must similarly be set to ID. We can do this manually or we could press the little ellipsis (...) to open the Subform Field Linker:



Either way the properties should look like this:



We can now try our form:

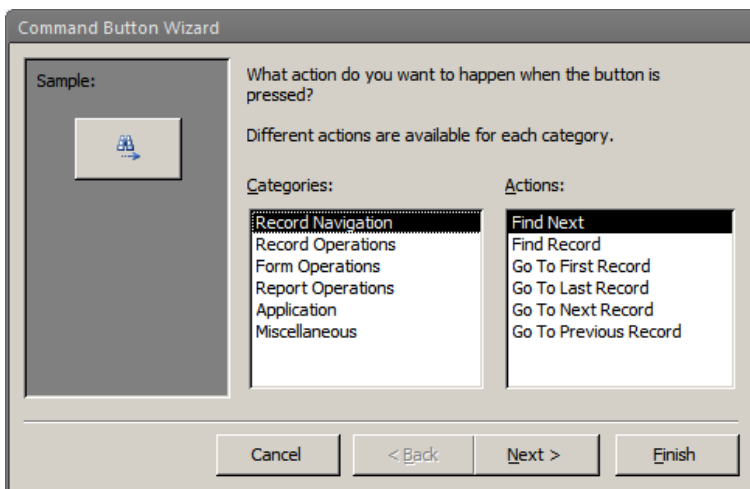


It works just like the one we created with the wizard in the previous section.

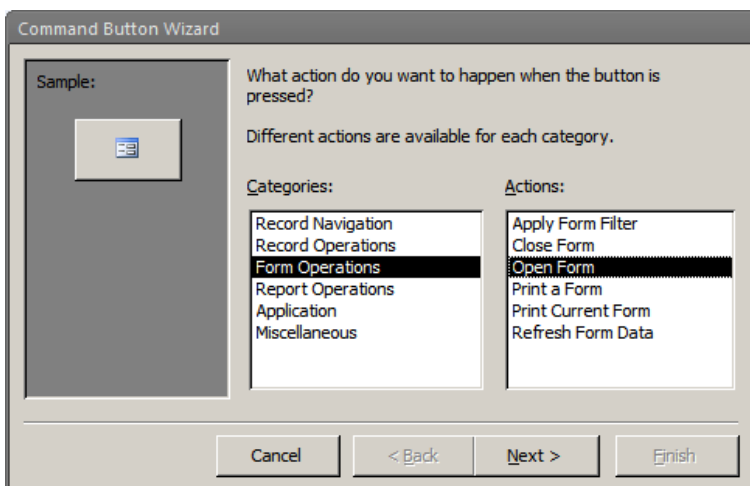


## 6.5 Non-Data Forms

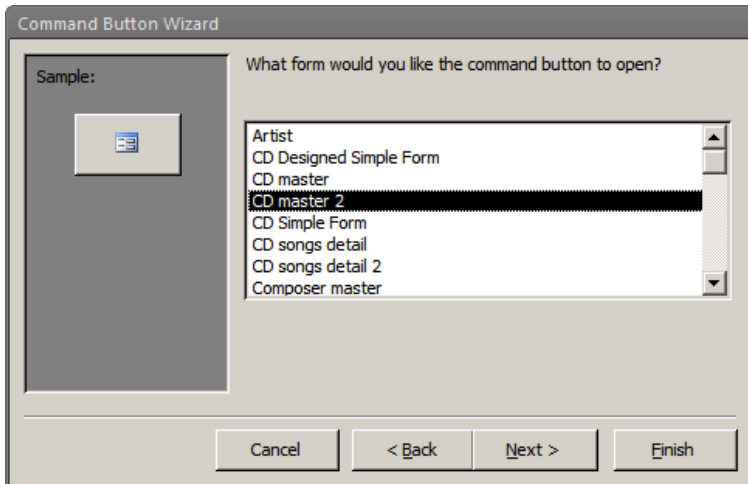
Forms can also be used for menus and navigation in a bigger application. We can for example make a form that has two buttons: One for opening the form created in section 6.4 and one for opening the form created in section 6.2. This form is different from the ones we created in previous sections in that it does not have a record source. All the other forms had a record source, i.e. a table or a query from where the form retrieved data (and also saved data to). The form we will create now will not have any link to data. We can start by creating a blank form in design mode (by selecting Create > Blank Form on the ribbon). We can then add a Button control to this form (available on the ribbon under Design). The Command Button Wizard will pop up as soon as we add a button on our form (provided that Control Wizards are activated):



In this wizard we can select an action and a layout (from a predefined list of functions and layouts) for our new button. We want a “Form Operation” to open another form:



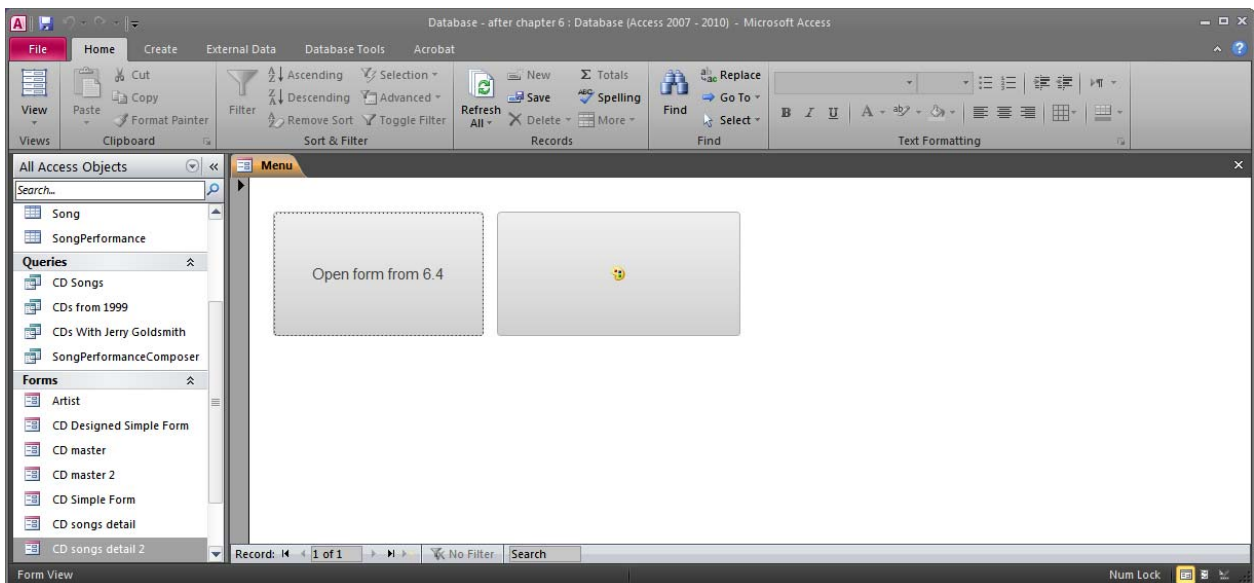
The wizard will then ask us to specify the form to be opened:



The rest of the steps are mostly about layout and giving your button a name and can also be skipped by selecting Finish.

We can now add a second button for the second form.

The default layout of the buttons is just a little icon, but this can be changed either in the wizard or later in the property sheet. The properties Picture, Caption, and ControlTip Text are relevant to this. See if you can make the form look like this:



A database with all the forms that we created in this chapter is available at <http://coursematerial.nikosdimitrakas.com/access/>.

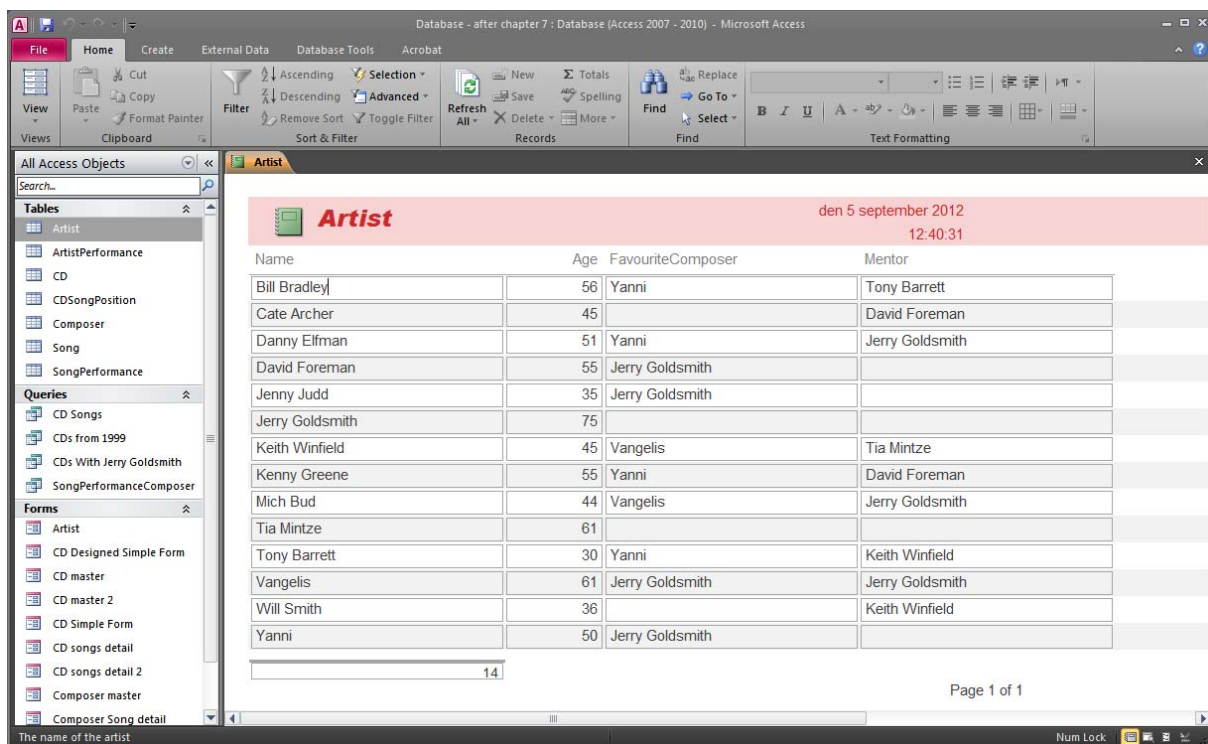
## 7 Reports

Reports are very similar to forms, but they are more static. You can think of a report as something that would be a preview of a printout. Creating reports is similar to creating forms. A report can be based on a table or query and we can have subreports, just like with forms. In the sections that follow, we will look at some examples of reports. Since we already worked a lot with forms, we will not go into the same level of detail in this chapter.

### 7.1 Simple Reports

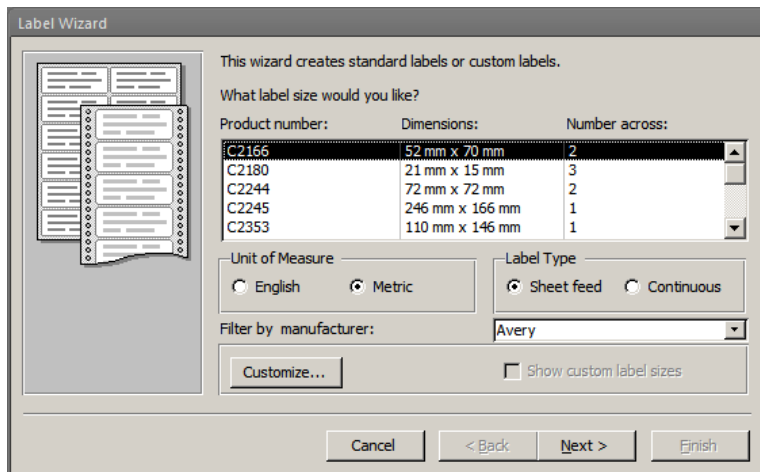
A report can be created by selecting Create > Report on the ribbon. The table or query that is selected in the object browser when we press Create > Report will become the record source of the report. As with forms, reports can be created in other ways (as blank reports or with the help of a wizard). In this section we will just look at the simplest type which is a report based on one table.

We can select the table Artist in the object browser and press Create > Report on the ribbon. The new report will look like this:

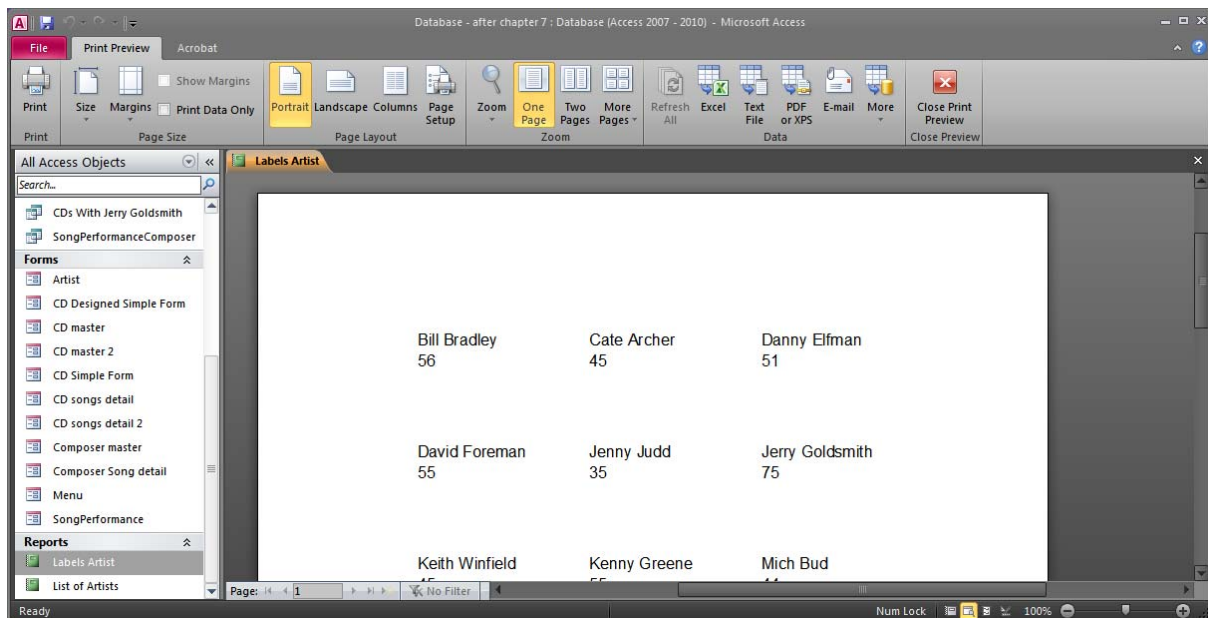


We can save this as "List of Artists". The report can also be viewed in Design View and in Layout View. These views can be used to modify the report and all the report controls on it.

Access can make many different types of simple reports. It can for example make mailing labels. Just select the Create > Labels (with the table Artist selected in the object browser):



Answer the questions of the wizard and Access will create the report. It can look something like this:

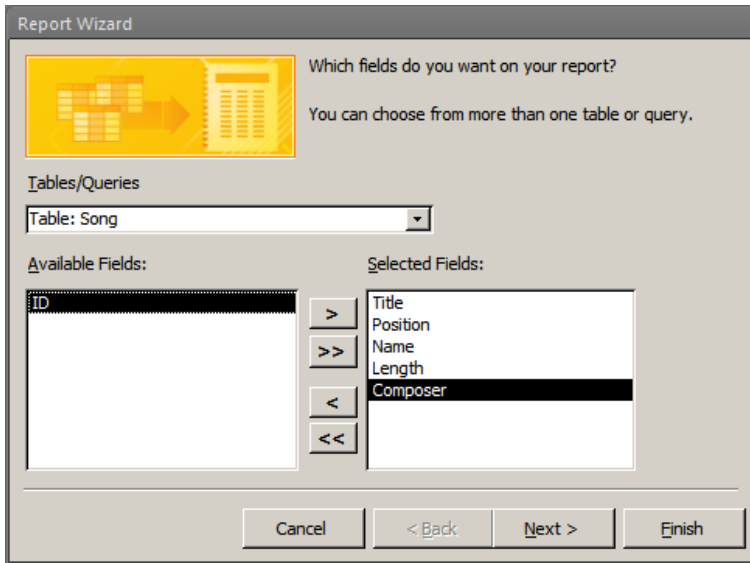


## 7.2 Reports That Combine Many Tables

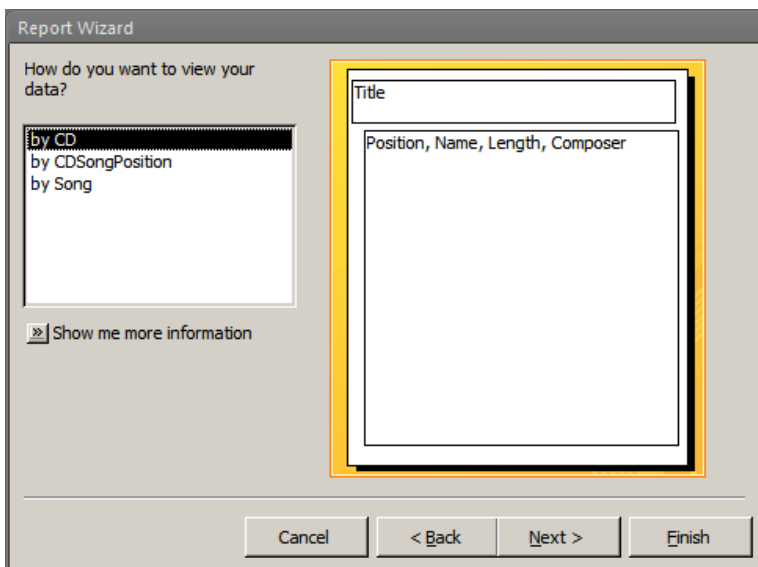
We can also make reports that combine many tables. This is exactly the same as it was for forms. In our case in chapter 2 we had the following need: *A report that shows the content of each CD (back cover style).*

To do this we need the tables CD, CDSongPosition and Song. The table SongPerformance is also needed for the connection between Song and CDSongPosition, but we have nothing to show from it.

We can start by selecting to make a new report with a wizard. The wizard will ask us to add all the relevant fields. We can add CD.Title, CDSongPosition.Position, Song.Name, Song.Length and Song.Composer:



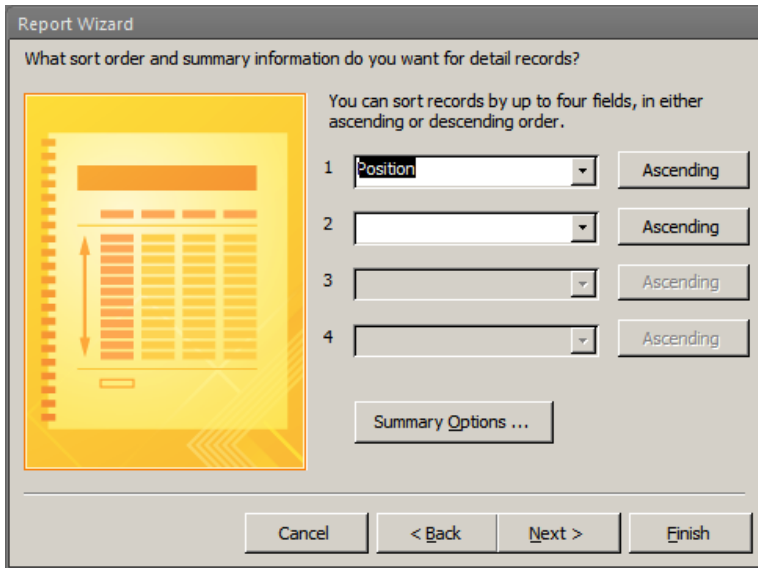
Next, we have to choose the structure of our report. This is similar to what we did in the Form Wizard in section 6.3. We can select the option "by CD"<sup>4</sup>:



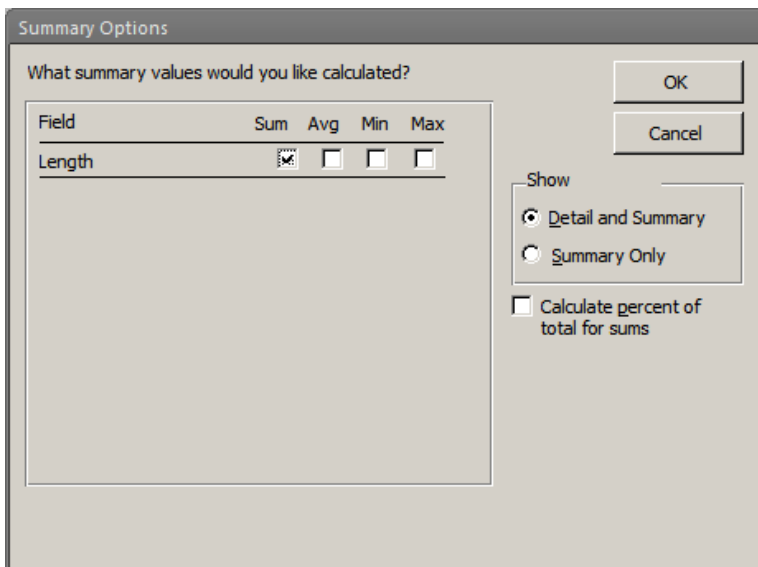
We don't need any more grouping level so we move on to sorting. The important thing is that the songs appear in the correct order, so we can sort them by Position:

---

<sup>4</sup> The wizard actually creates grouping levels and sorts the data accordingly. In section 7.4 we will see how we can do this manually.

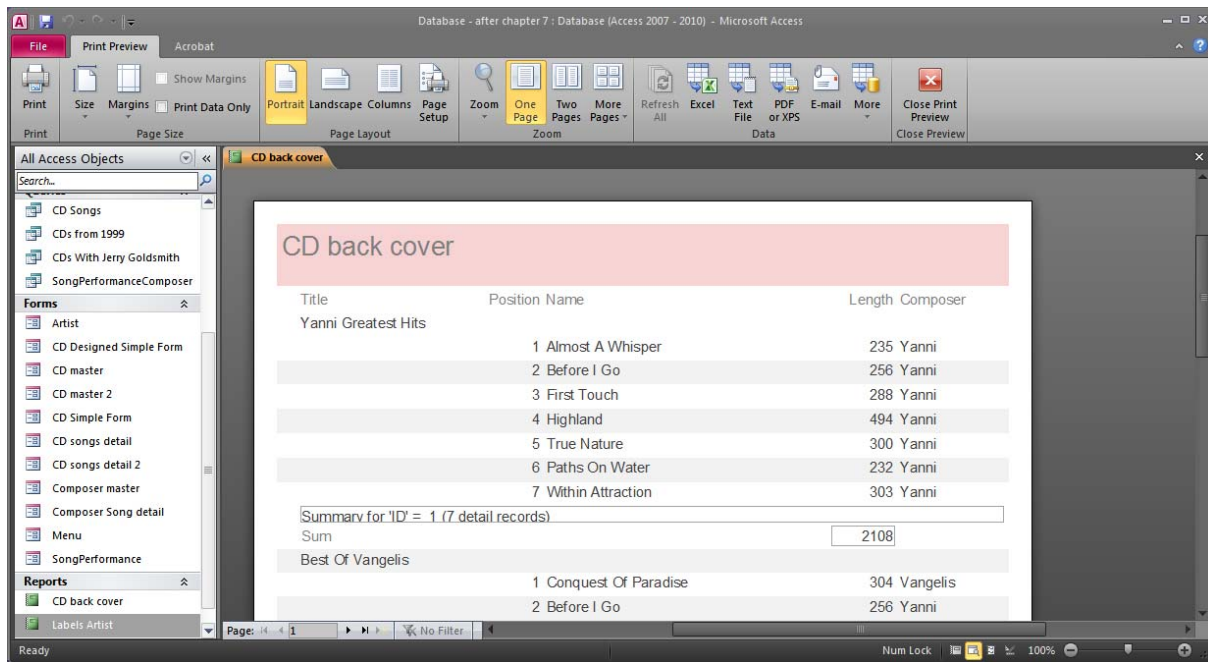


On this window we also have the option to create summary information. We can use this to add a total of seconds per CD in our form. Press the Summary Options... button to see the available options. Choose the Sum of Length and the option Detail and Summary:

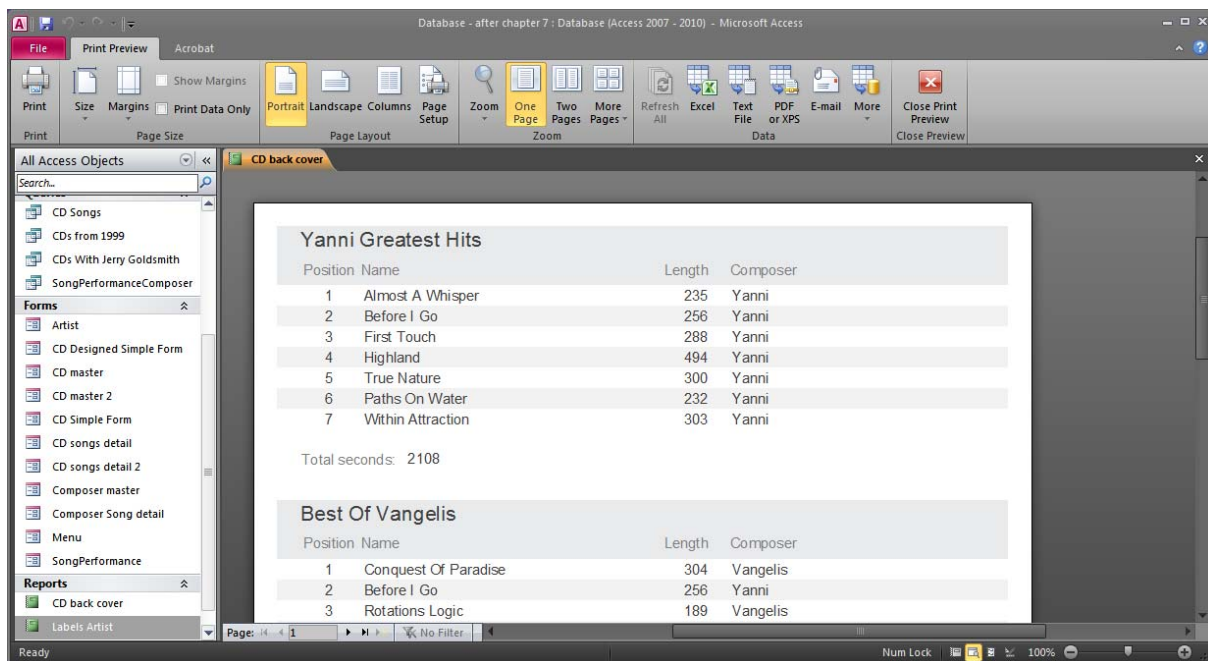


The next step is about layout, so choose something that you like. Finally give a name to the report to be generated; for example "CD back cover". Press Finish and Access will generate the report:

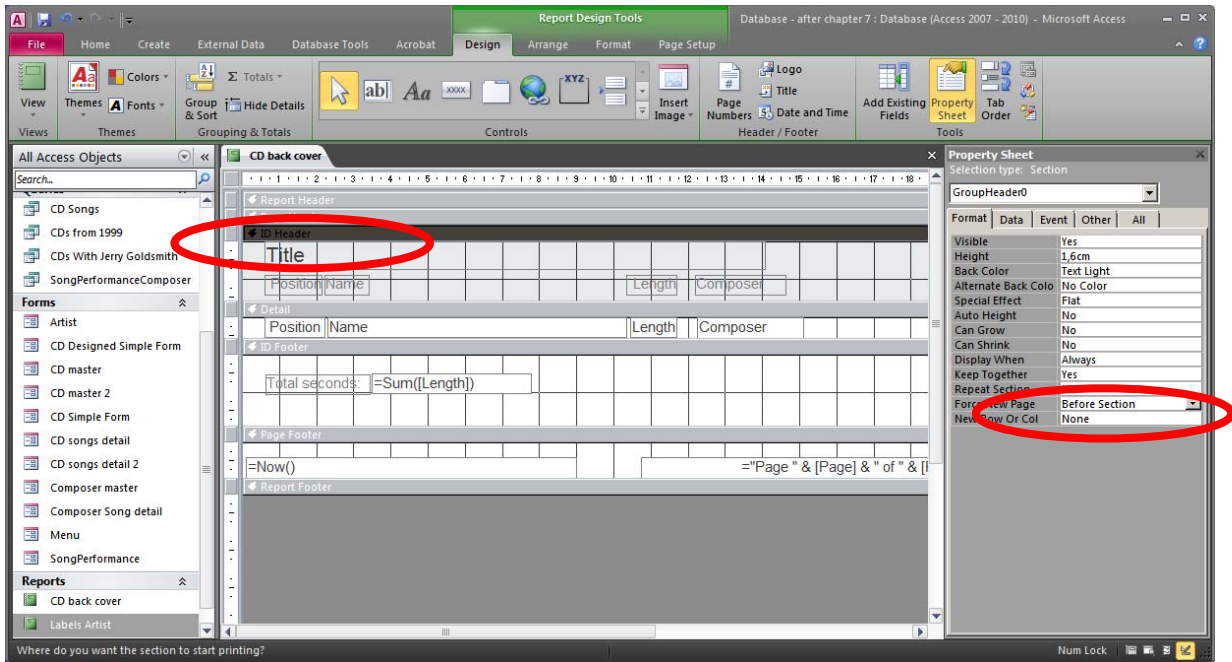




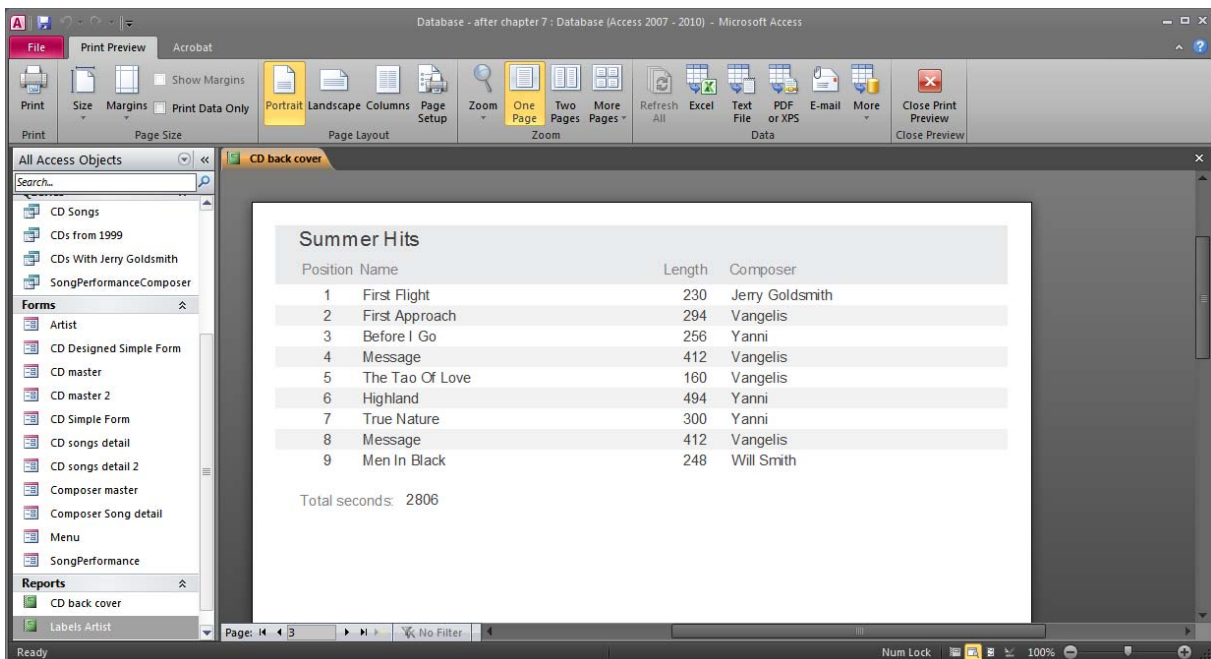
We can see that the wizard has generated all the things we wanted, but there are some things we may want to change or remove. We can do that in design view. We can for example remove the "Summary for 'ID' = ...". We may also want to remove some of the field labels. We can start by doing that. Our report can now look like this:



There is one more thing that we might want to change. We might want every CD to appear on a new page. We switch back to the design view and select the ID Header. In the property sheet we can now set the property "Force New Page" to "Before Section":



We can now look at our report and browse between the pages:



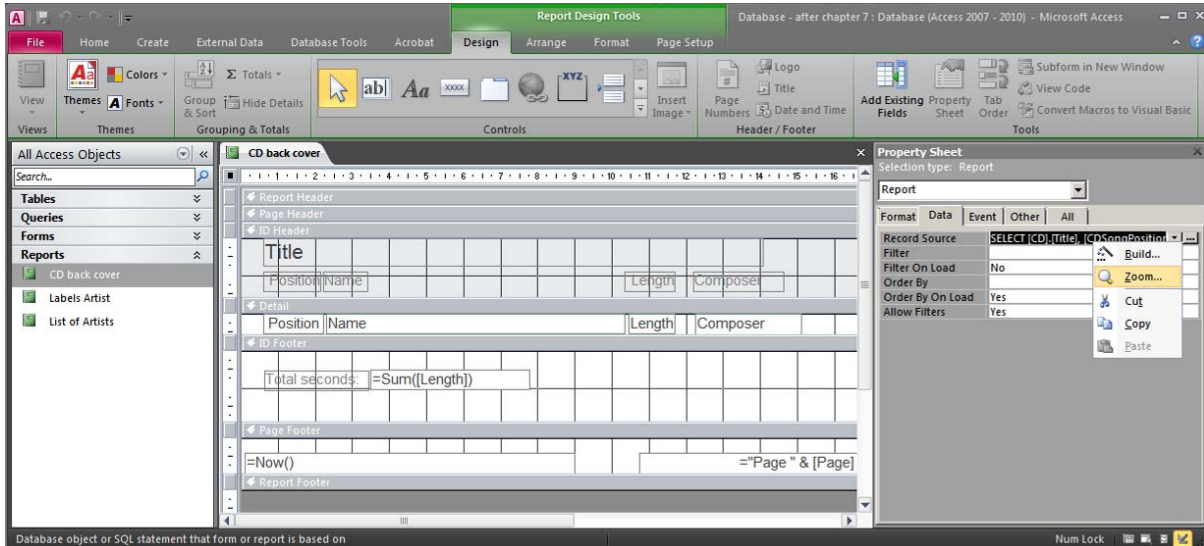
### 7.3 Reports Based On Queries

Making a report that is based on a query instead of a table is no different than making a report with a table. The wizard in the previous section actually generated a query for the report. The query was:

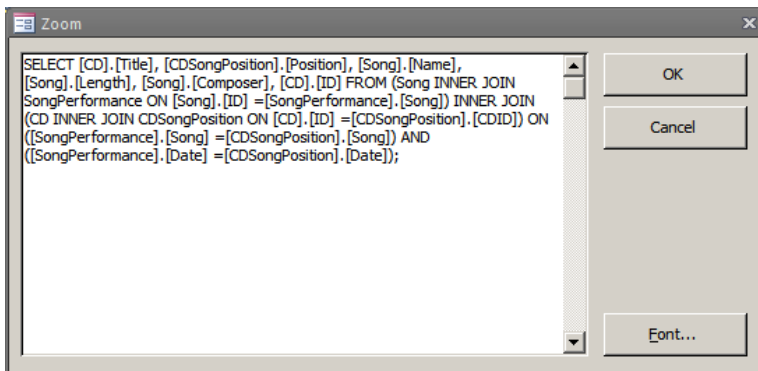
```
SELECT CD.Title, CDSongPosition.Position, Song.Name, Song.Length,
Song.Composer, CD.ID FROM (Song INNER JOIN SongPerformance ON
Song.ID=SongPerformance.Song) INNER JOIN (CD INNER JOIN CDSongPosition
ON CD.ID=CDSongPosition.CDID) ON
```

(SongPerformance.Date=CDSongPosition.Date) AND  
(SongPerformance.Song=CDSongPosition.Song);

The query can be accessed in the report properties under Data. To see the entire SQL statement, right click on it and select Zoom:



And a new window will appear with the query. Table names and column name are inside [] which ensures the query works even if table names or column names use special characters or reserved words:



If we had this query as a stored query (a view) in our database, we could then just write the name of the query object in the Record Source property.

## 7.4 Grouping And Sorting

In section 7.2, we let the wizard take care of the grouping and sorting in our report. But it is also possible to specify the grouping and sorting manually. To illustrate this we can create a query and then make a report in design view (no wizards) based on that query. We can try to satisfy the following need that was in our case: “A report that shows for each composer the songs that they have composed and which performances of them exist and in which CDs these performances are included”.

We can first create a query that gathers all the necessary data:

```
SELECT Composer.CName, Song.Name, SongPerformance.Date, CD.Title, CD.Year
FROM CD, Composer, Song, SongPerformance, CDSongPosition
WHERE Composer.CName = Song.Composer
AND Song.ID = SongPerformance.Song
AND SongPerformance.Song = CDSongPosition.Song
AND SongPerformance.Date = CDSongPosition.Date
AND CD.ID = CDSongPosition.CDID
```

Save the query as “Composed Songs Info”.

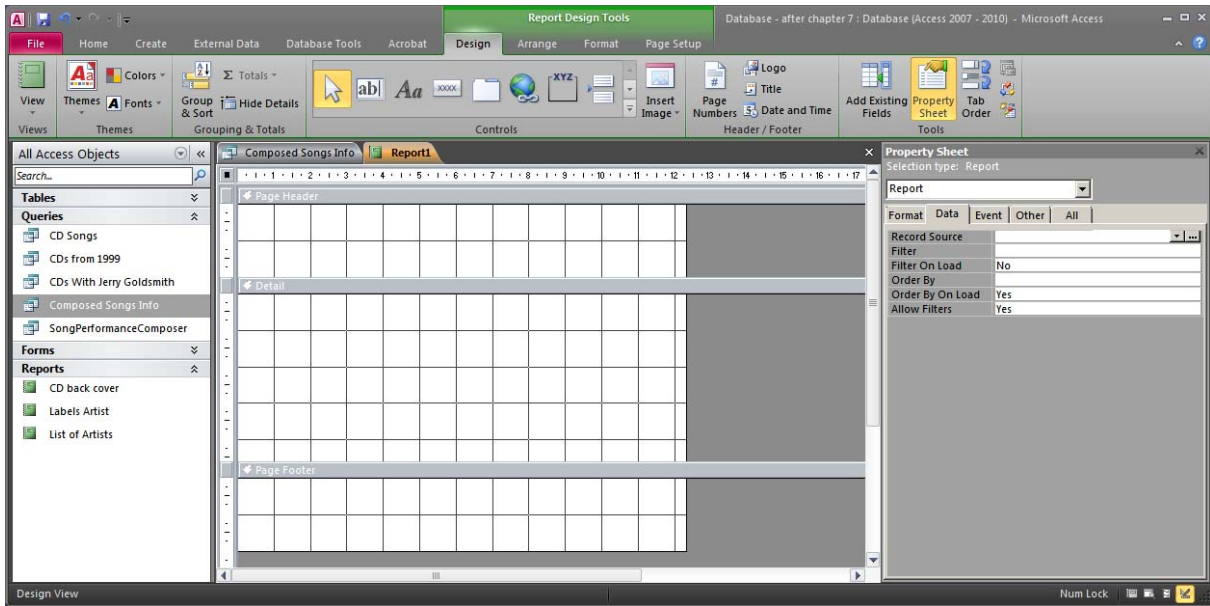
The result of this query looks like this:

CName	Name	Date	Title	Year
Yanni	Almost A Whisper	1999-04-05	Yanni Greatest Hits	2002
Vangelis	Conquest Of Paradise	2001-08-09	Best Of Vangelis	2001
Jerry Goldsmith	First Flight	1998-04-16	Summer Hits	1998
Yanni	Until The Last Moment	1999-05-15	The Ultimate Yanni Collection	1999
Yanni	Before I Go	2000-08-17	Yanni Greatest Hits	2002
Yanni	Before I Go	1989-04-03	Best Of Vangelis	2001
Vangelis	First Approach	1998-04-02	Summer Hits	1998
Yanni	Almost A Whisper	1999-04-05	The Ultimate Yanni Collection	1999
Yanni	First Touch	2001-05-31	Yanni Greatest Hits	2002
Vangelis	Rotations Logic	1999-04-08	Best Of Vangelis	2001
Yanni	Before I Go	1989-04-03	Summer Hits	1998
Yanni	True Nature	1998-04-08	The Ultimate Yanni Collection	1999
Yanni	Highland	1998-11-08	Yanni Greatest Hits	2002
Vangelis	Rotations Logic	2001-05-17	Best Of Vangelis	2001
Vangelis	Message	1995-08-14	Summer Hits	1998
Yanni	Until The Last Moment	1999-05-15	The Ultimate Yanni Collection	1999
Yanni	True Nature	1998-04-08	Yanni Greatest Hits	2002
Vangelis	The Top Of Love	1991-05-11	Best Of Vangelis	2001

As we can see the same composer appears many times, and the same song may have been performed many times and each performance may be included in many different CDs. It can therefore be good to add some grouping levels in our report so that the same information doesn't appear over and over again.

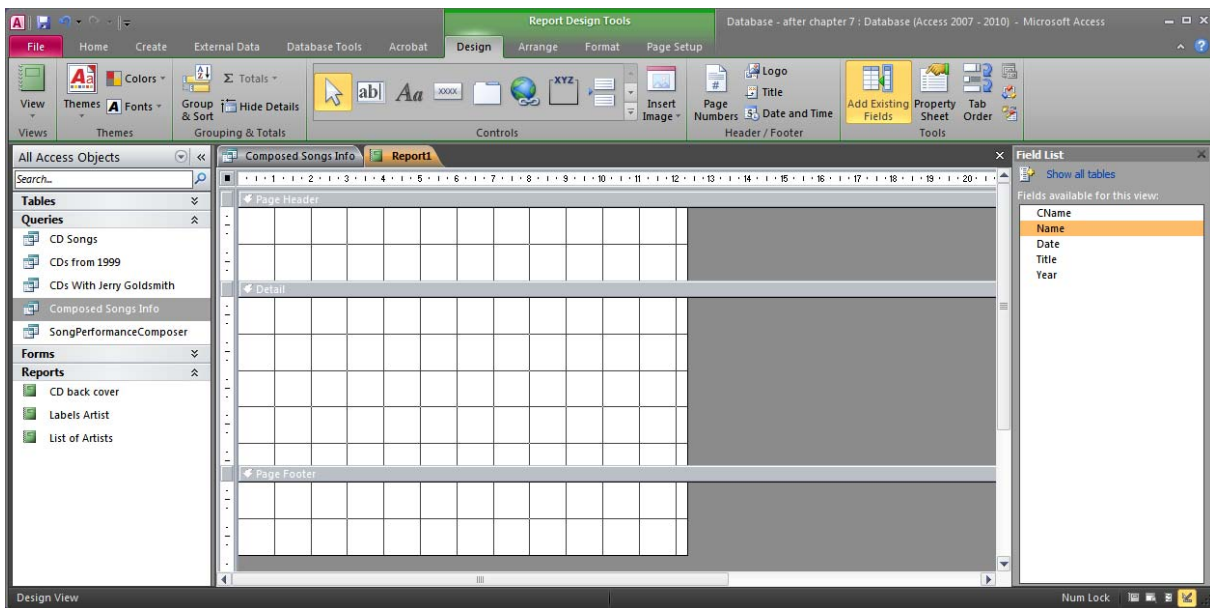
We can start by creating a report in design view based on the new query. Press Create > Report Design on the ribbon and a new blank report will appear:



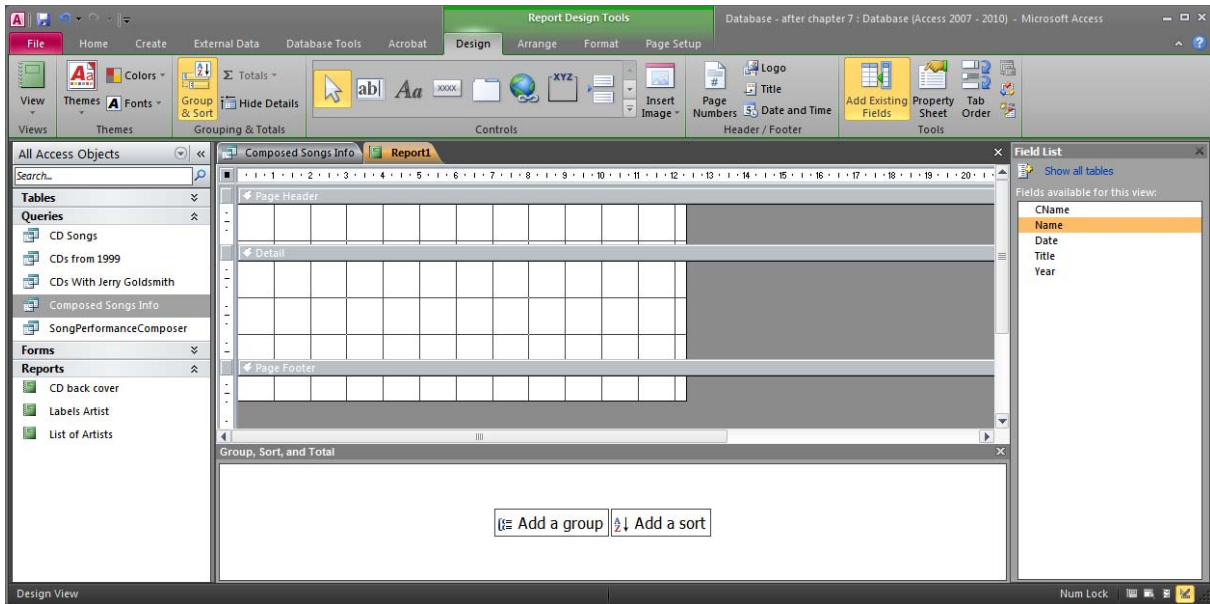


The new report has no Record Source specified and has by default a Page Header, a Detail, and a Page footer.

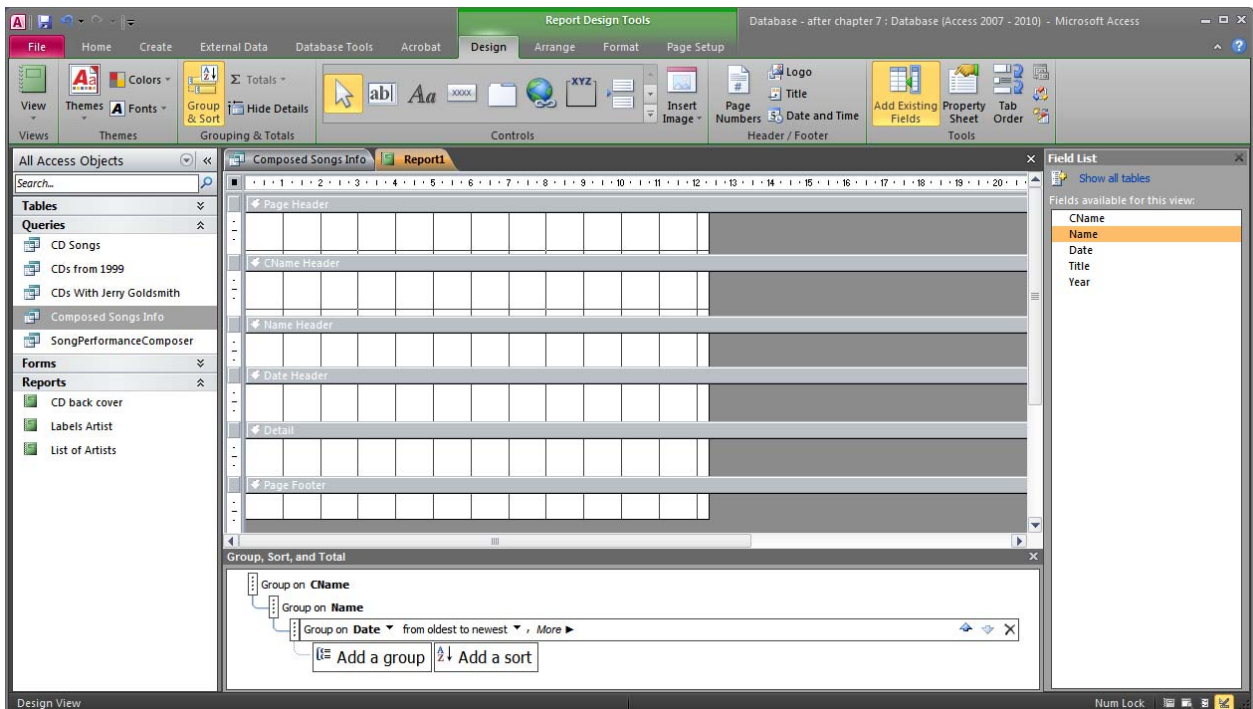
We can start by selecting our stored query as the Record Source. The columns of the query are now available in the Field List:



No fields have been placed anywhere on the report yet. Before we add any fields, we can open the Sorting and Grouping dialog by either pressing the Sorting and Grouping button on the ribbon (under Design) or by right-clicking anywhere on the report and selecting "Sorting and Grouping". Either way the "Group, Sort, and Total" pane will come up:

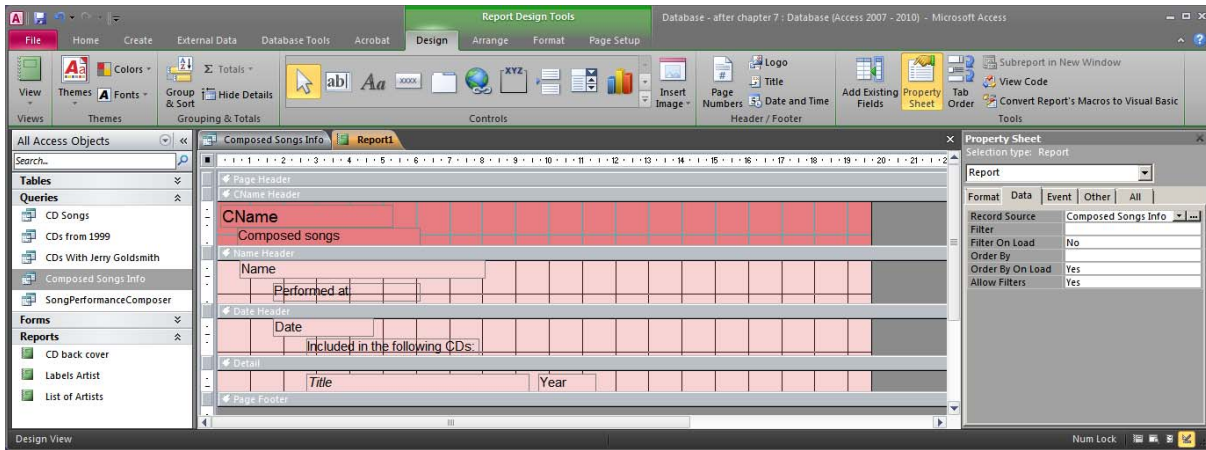


Here we can define the three grouping levels that are relevant in our case. First we group by composer, then we group by song, and then we group by song performance. For a composer we have the column CName, for the song we have the column Name, and for the Performance we have the column Date. We can also specify whether we want a group header or a group footer, whether to sort in ascending or descending order and how to handle page breaks. If all group levels have a group header and no group footer (which is the default) the report should look like this:

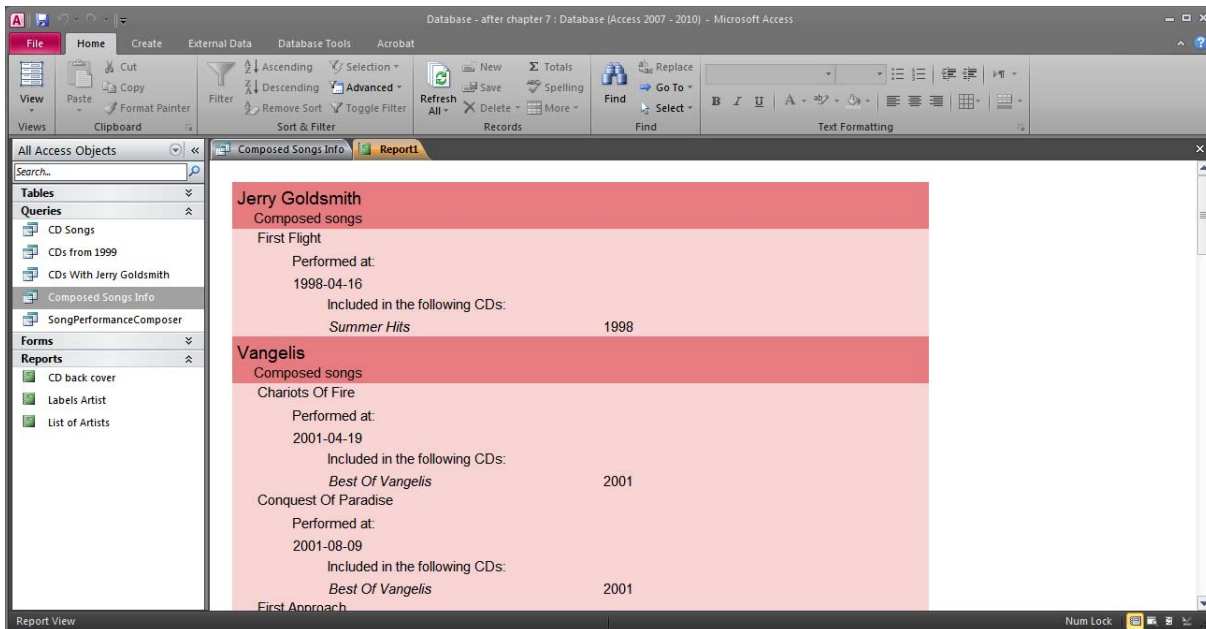


Our report now has many levels. We can now place the appropriate fields on each level (drag and drop from the Field List). We can also adjust the fields' size, colour, etc. We can also add necessary labels. The report could now look like this:

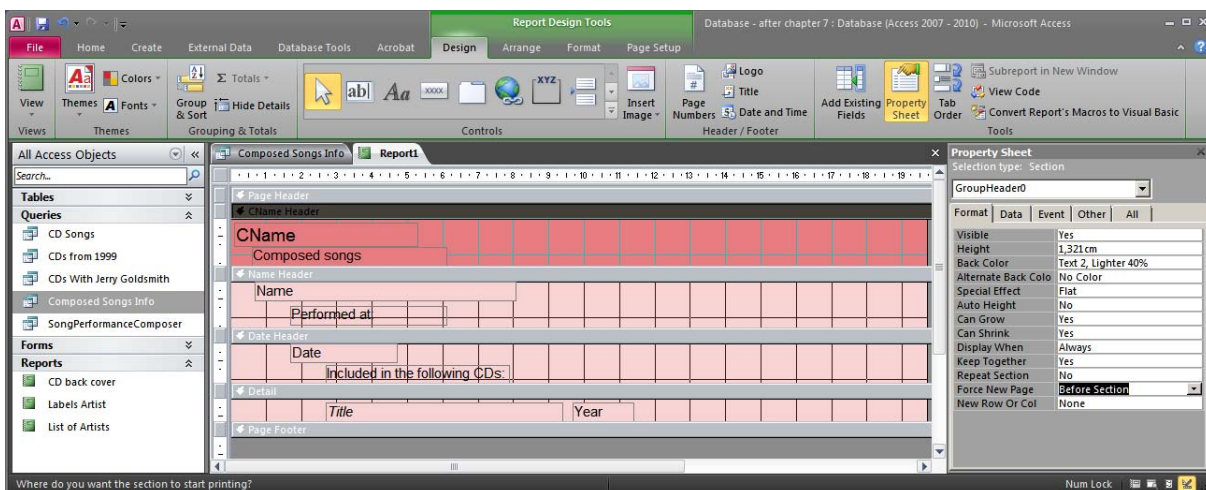




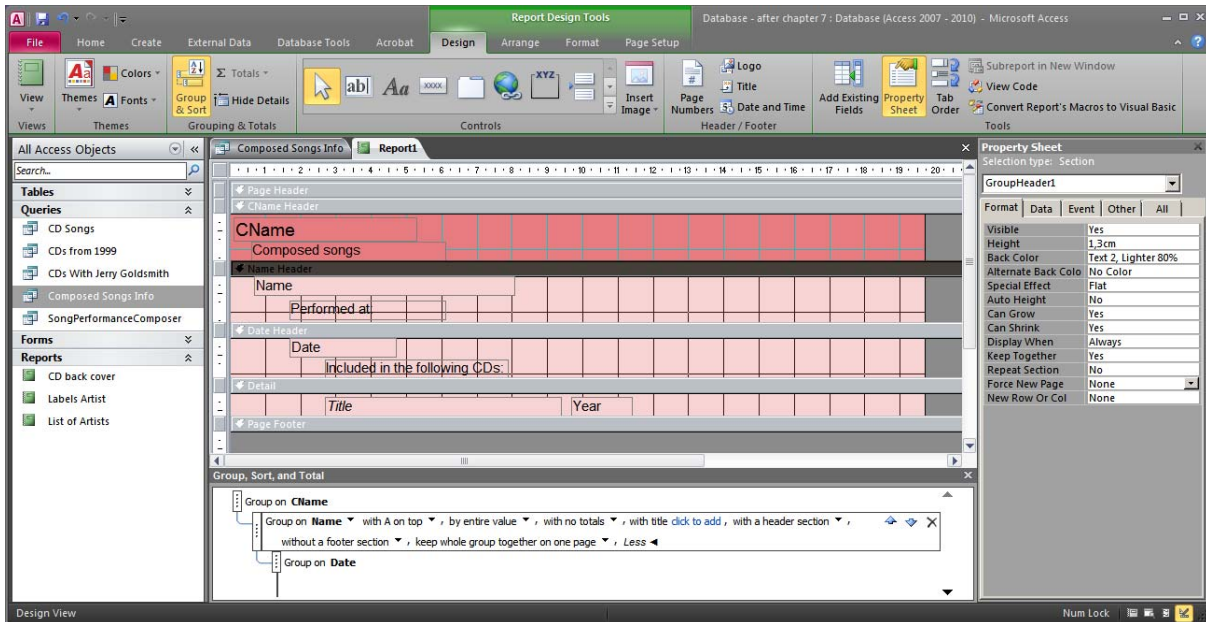
We can now switch to the Report View and see if we are satisfied:



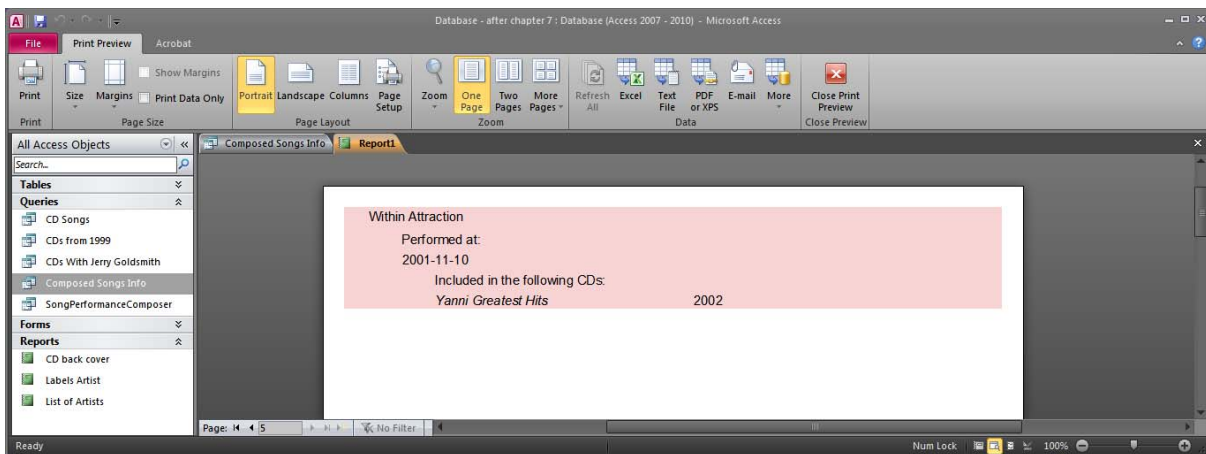
It looks fine, but maybe it would be better to change page for each new composer. We can switch back to the design view, select the CName Header and change its property Force New Page to Before Section:



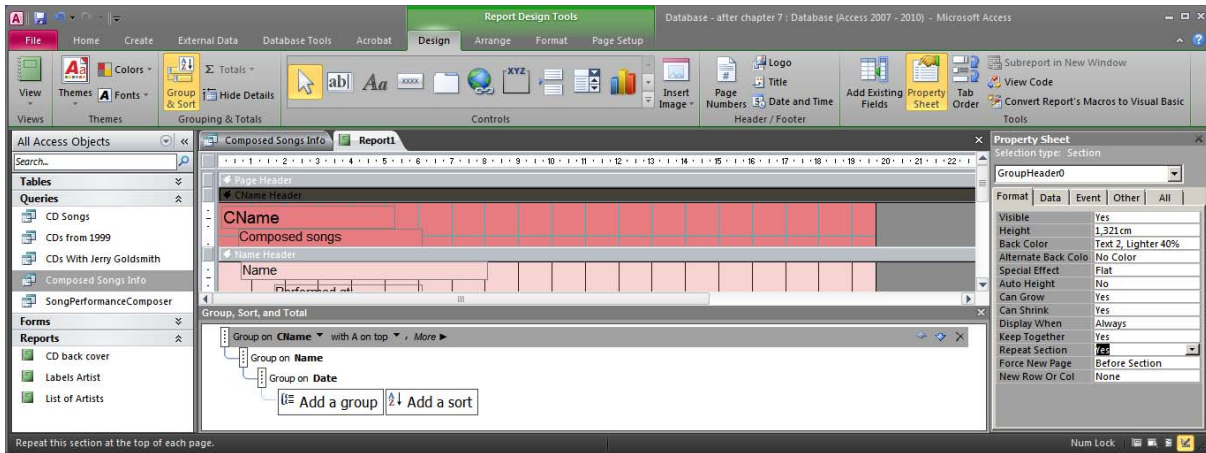
Our report is now better, but we can notice that if one composer doesn't fit on one page, the page break can be anywhere (for example after the label "Performed at:"). We can instruct Access to make better choices on page breaks by going back to the Sorting and Grouping and selecting the option "keep whole group together on one page" (for example at the song level):



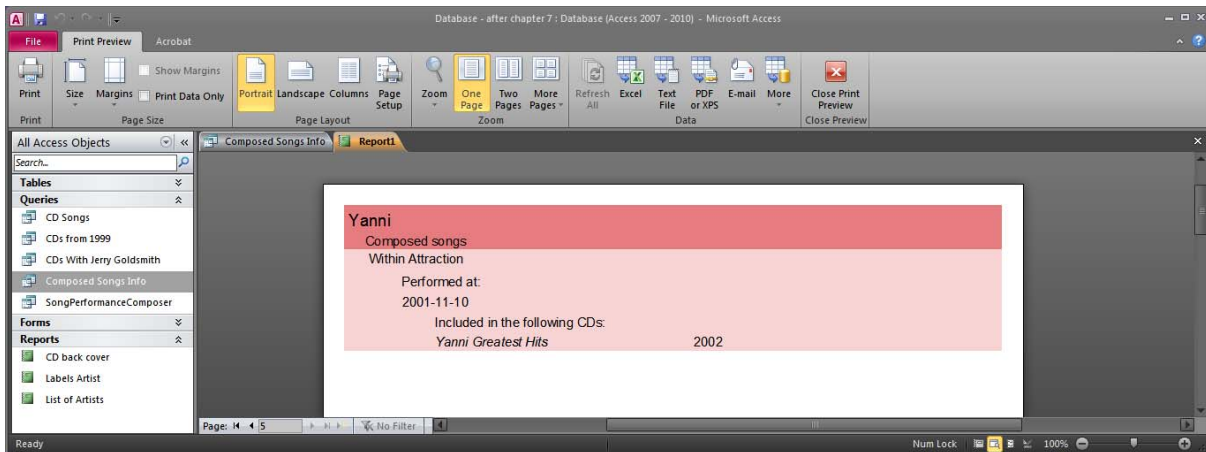
This will make sure that if a song doesn't fit on a page it will not start on that page. So now we have the following effect:



When a composer's songs are spread over more than one page, we can only see the composer name on the first page. To fix this we can select the CName Header and change its property Repeat Section to Yes.



This will make sure that the composer's name appears on every page about this composer:



Save the report as “Song Performances on CDs by Composer”.

## 7.5 Subreports

Similarly to the way we created forms with subforms in section 6.3, we can create reports with subreports. This possibility can be especially useful when our report contains many aspects of one concept. As an example we will create a report that shows the different artists and the different composers for every CD. We can also define the report to include the number of songs and total length of each CD. Each CD will be shown on a new page. Here is how a page of the report should look like:

Title	Summer Hits
Year	1998
Number of songs	9
Length in seconds	2806
Composers	Artists
Jerry Goldsmith	Bill Bradley
Vangelis	Cate Archer
Will Smith	Danny Elfman
Yanni	David Foreman
	Jenny Judd
	Jery Goldsmith
	Keith Winfield
	Kenny Greene
	Mich Bud
	Tia Mintze
	Tony Barrett
	Will Smith

In order to achieve this we must create one main report and then place in it two subreports; one for the composers and one for the artists.

Each of the reports must have a source (a table or a query). In this case we need to have three queries since the data is spread over several tables. The queries must contain one or more columns that can be used to connect the corresponding reports. This linking column will in this case be the id of the CD. We can create the following three queries in advance so that we can use them later when creating the reports:

Query for main report (CD info for Report):

```
SELECT CD.ID, CD.Title, CD.Year, COUNT(*) AS songcount, SUM(Length) AS
cdlength
FROM CD, CDSongPosition, Song
WHERE CD.ID=CDSongPosition.CDID
AND SONG.ID=CDSongPosition.Song
GROUP BY CD.ID, CD.Title, CD.Year
```

Query for composer subreport (CD composers for Report):

```
SELECT DISTINCT CDSongPosition.CDID, Song.Composer
FROM Song, SongPerformance, CDSongPosition
WHERE SongPerformance.Song=CDSongPosition.Song
AND SongPerformance.Date=CDSongPosition.Date
AND Song.ID=SongPerformance.Song
```

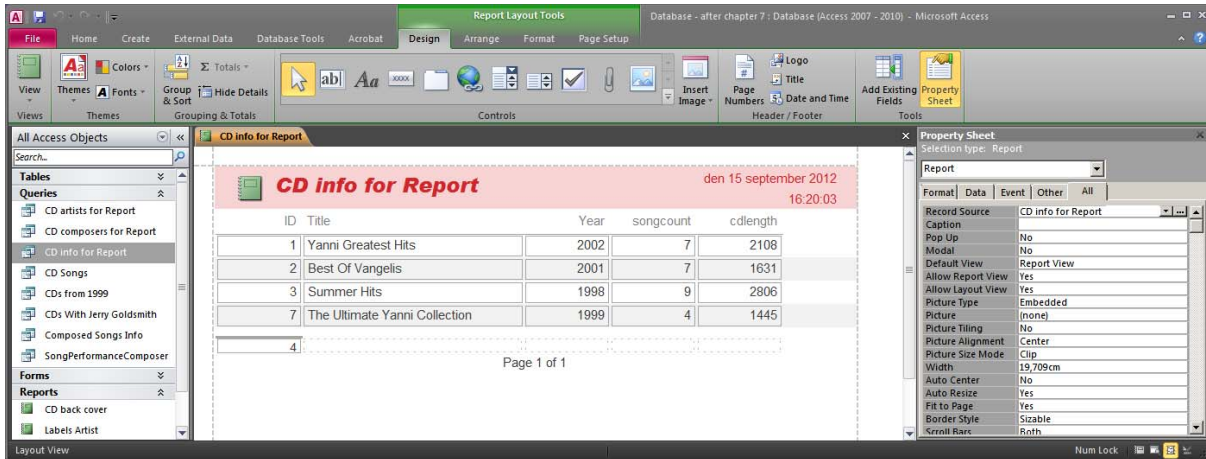
Query for artist subreport (CD artists for Report):

```
SELECT DISTINCT CDSongPosition.CDID, ArtistPerformance.Name
FROM SongPerformance, ArtistPerformance, CDSongPosition
WHERE SongPerformance.Song=ArtistPerformance.Song
AND SongPerformance.Date=ArtistPerformance.Date
AND SongPerformance.Song=CDSongPosition.Song
AND SongPerformance.Date=CDSongPosition.Date
```

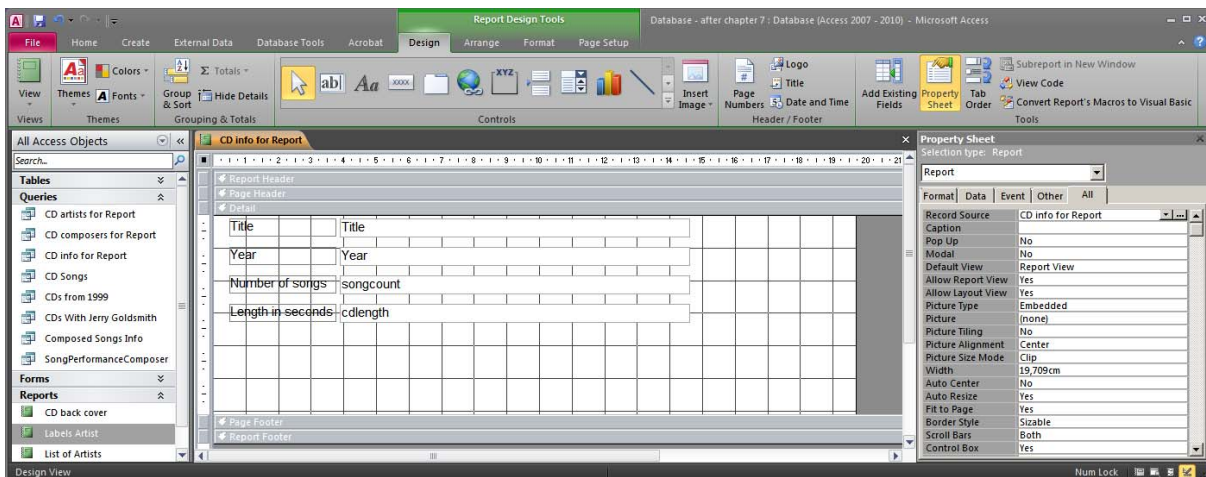
With these queries created, we can start creating the reports. The three queries have been given the names within the parentheses.



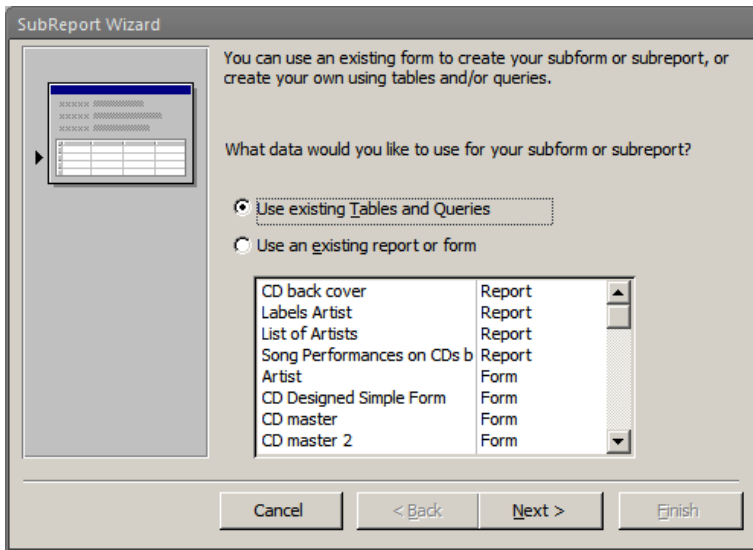
We can, of course, create the three reports independently and link them later, but here we will create the subreports from within the main report. This means that we have to start with the main report. Select the query for the main report in the object browser and press Create > Form on the ribbon. A standard report will be created:



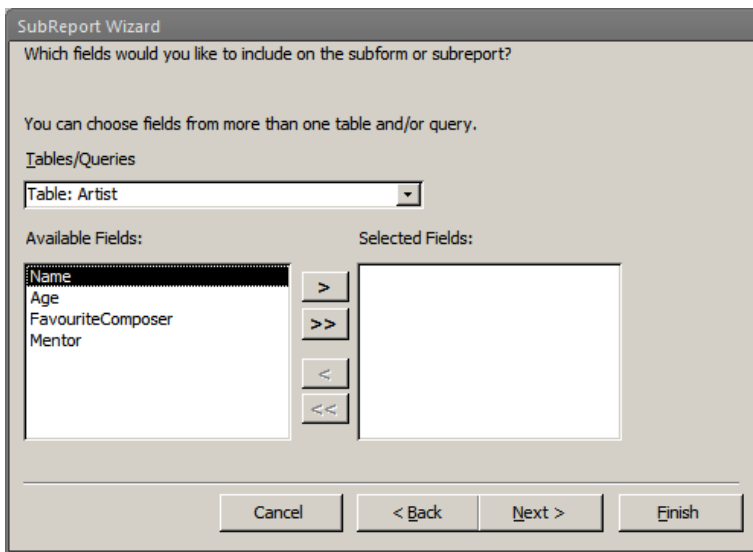
Make the necessary modifications to the layout and it could look like this:



Now make some space to put the subreports (also in the detail part of the report). Click on the subform/subreport button on the ribbon (under Design) to select it. Then click on the report to create a subreport. Alternatively, drag and drop the relevant query from the object browser onto the report (in Design View). The SubReport Wizard should appear. If you dragged and dropped the query onto the report, the first to steps will be skipped.

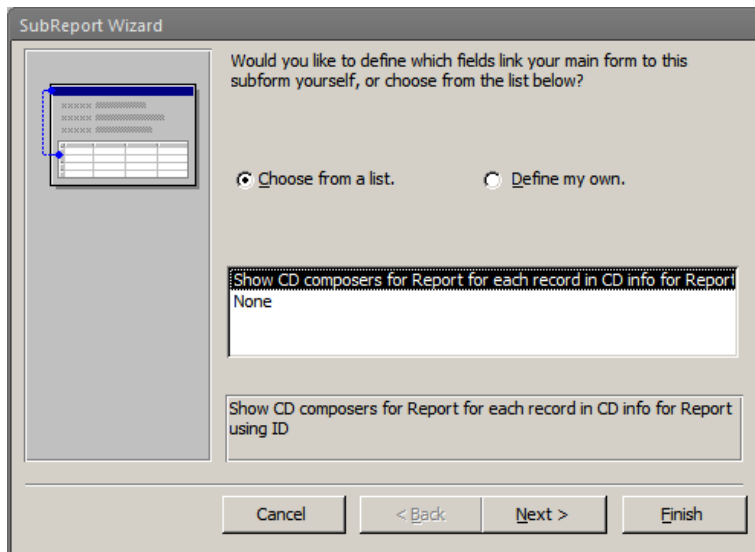


As we decided earlier we will create the subreports based on the queries that we created in advance. Select "Use existing Tables and Queries" and press Next. Now select the relevant query and the relevant columns:

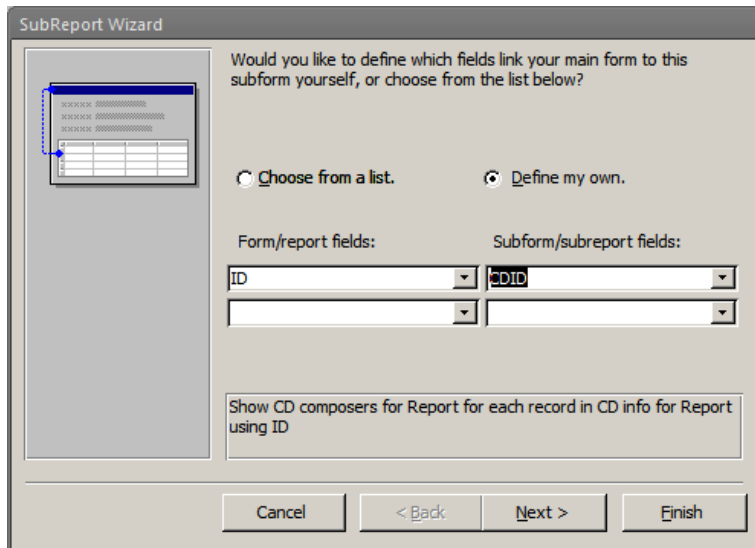


At the next step the wizard suggests possible ways to link the subreport to the main report. The only suggestion is the one we intended to use:



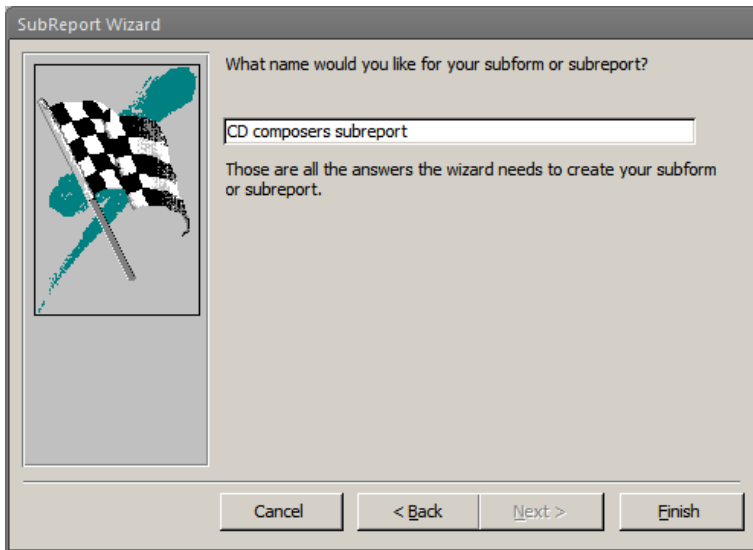


Alternatively, we can define it ourselves:

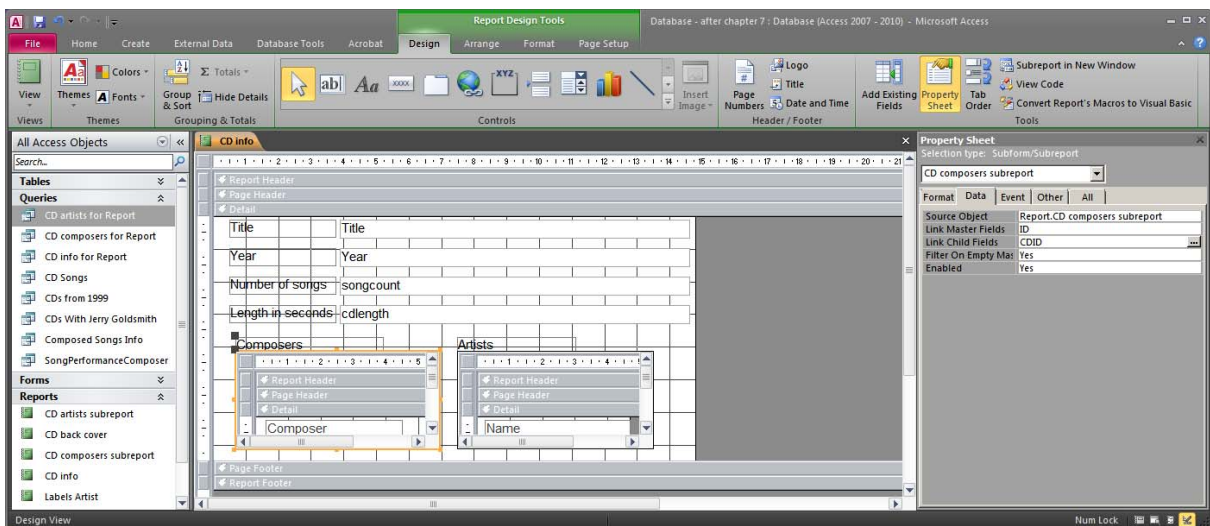


Either way the result will be the same.

At the last step of the wizard we can define a name for the subreport (which will show up as a normal report in the object browser).

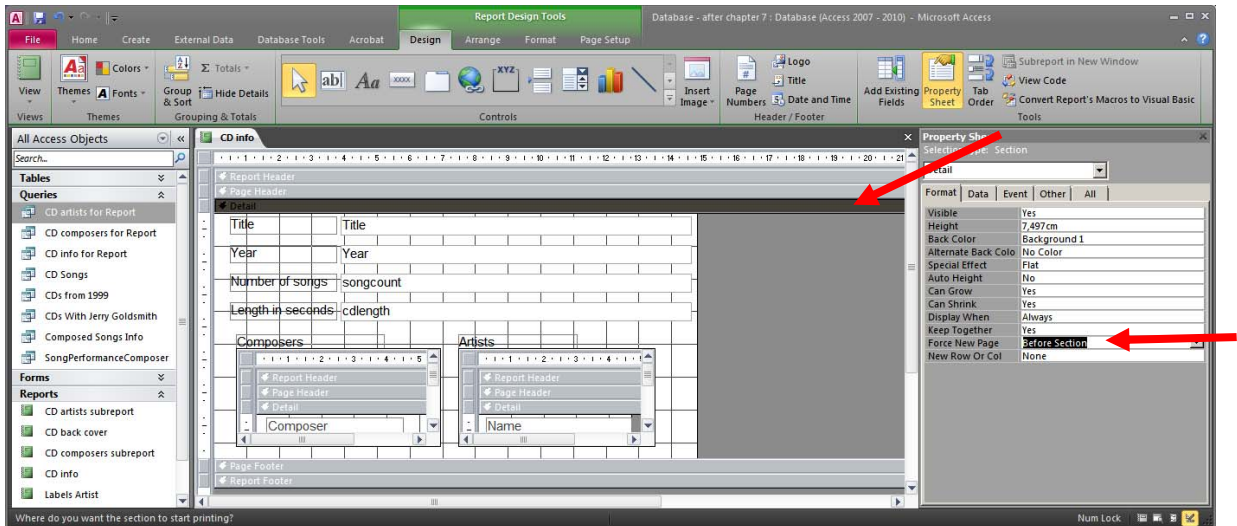


We can now repeat the process for the second subreport. We can of course also edit the layout of the subreports. Our report should now contain two subreports that are both linked to the main report using the id of the CD. This is also visible in the Property Sheet:



So if you want to link two reports without using the wizard, you can open the Property Sheet and set the appropriate properties.

Finally, we can define that we only want one CD per page. To do this we specify that there should be a page break before each CD (i.e. each Detail block):

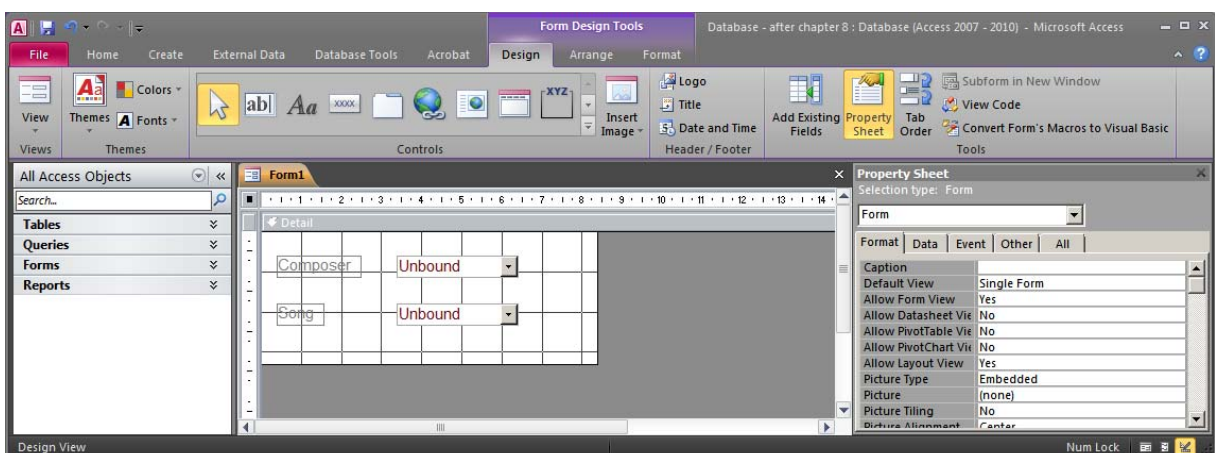


## 8 Macros

Macros can be used in order to do something a little more complicated, but not that complicated that it would require writing code. A macro can for example execute a query, activate or refresh a form or form component, etc.

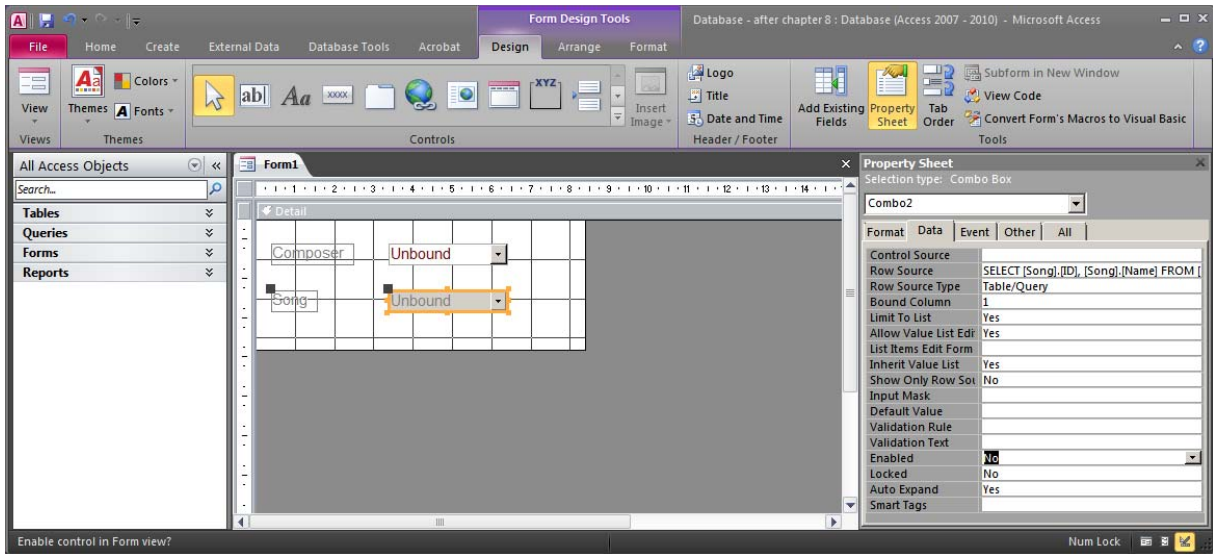
In this chapter we will look at a very simple macro that refreshes a Combo Box when another Combo Box has changed value. What we want to do is to have two Combo Boxes, one for composers and one for songs. We want the song Combo Box to be inactive until we have selected a composer, and then activate it and let it contain only the songs composed by the selected composer. This can be combined with other stuff to make a form for, for example, registering new song performances. We will only look at the macro related parts though.

Create a blank form in Design View. Place a Combo Box based on the table Composer on the form. Now place a second Combo Box on the form, this one getting its values from the table Song. It is enough to include the column Name from the table Song. We have now a form that looks like this:

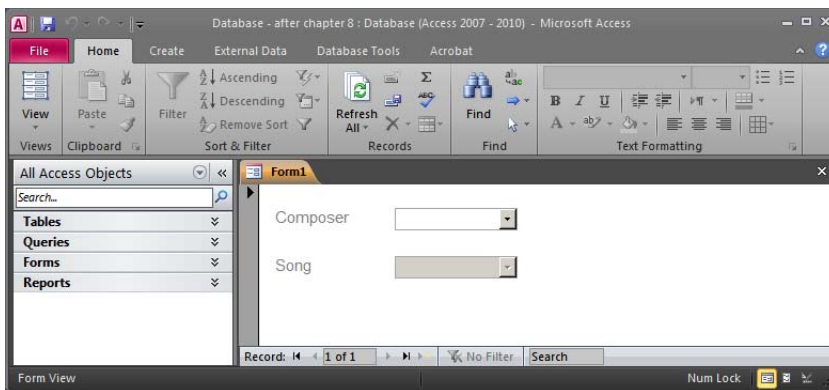


We can switch to the Form View and see that all the composers and all the song are visible.

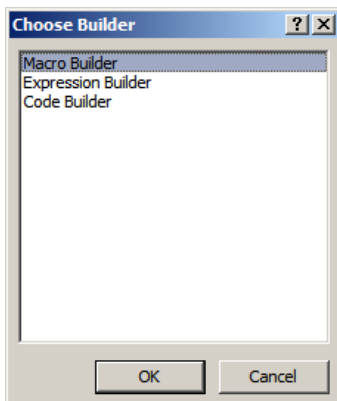
We can switch back to Design View. We can start by deactivating the second Combo Box. We can deactivate it by setting its property Enabled to No:



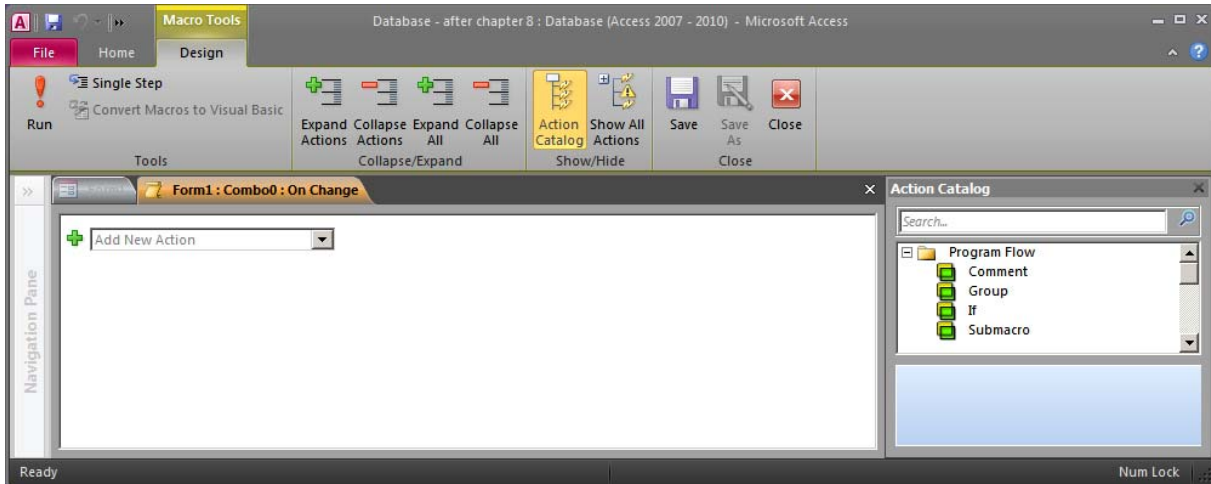
The Combo Box will still be visible, but the user won't be able to use it until we activate it:



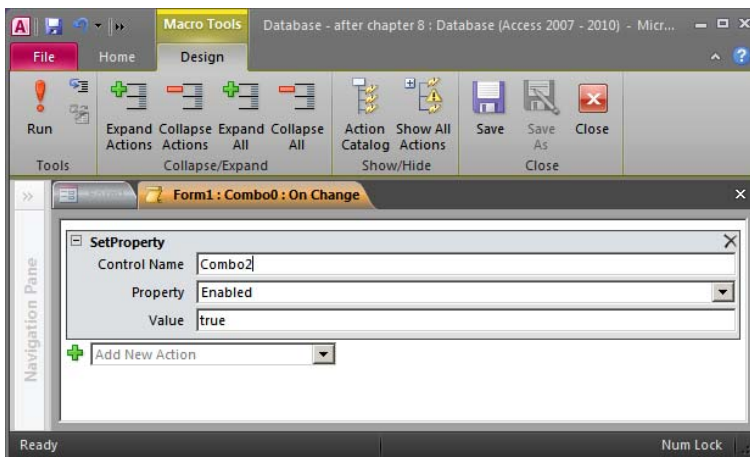
We can now see how we can activate it when the first Combo Box has been changed. We look at the properties of the first Combo Box (the one with the composers). Under Event we can find the property (event) On Change. Activate the property and click on the ellipsis to show the Choose Builder dialog:



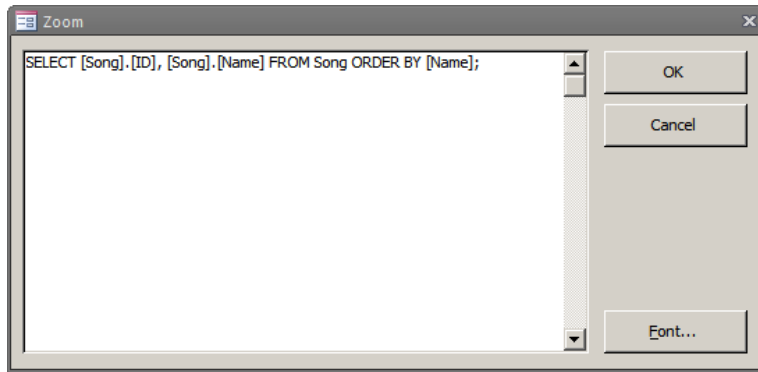
We can select between the Expression Builder (which we can also use to create validation rules as we saw in chapter 4), the Macro Builder, and the Code Builder (where we can write Visual Basic code). Highlight the Macro Builder and press OK. Access will create a new macro and show it in Design View:



Here we can define a bunch of actions that should occur when this macro is run. We can select any of the many predefined actions, and then, dependent on the selected action, specify the applicable properties. In order to activate our Combo Box we need to set its Enabled property to Yes (or true). So the action that can help us do that, is the action SetProperty. Select this action and its properties will become available:



The properties we need to specify for this action are: The name of the control whose property we want to change (in this case Combo2), the name of the property (in this case Enabled), and the new value for the property (in this case true). Access will provide suggestions when possible. We can save and close the macro and try the form. As soon as we select a value in the first ComboBox, the second one becomes active. However, the songs in the second combo box are still not filtered based on the selected composer. To do that, we need to alter some properties of the second Combo Box. Under Data there is a property Row Source. We can zoom into it and edit the automatically generated SQL statement (right click and select Zoom...). We can now see the Zoom window where we can edit our SQL statement:



The SQL statement as it is now has no connection to the current value of the first Combo Box. We can add this as a condition in the WHERE clause. The full SQL statement should be changed to the following:

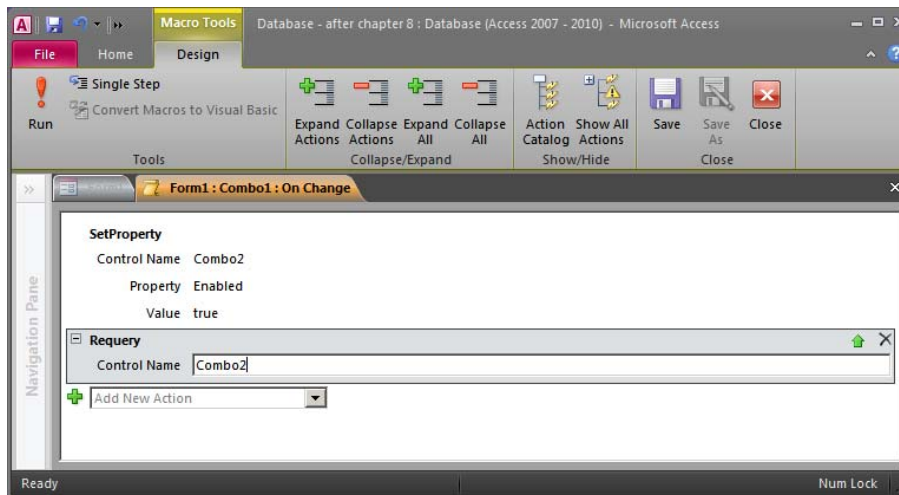
```
SELECT Song.ID, Song.Name  
FROM Song  
WHERE Song.Composer = Combo1  
ORDER BY Song.Name
```

Combo1 is the name of the first Combo Box. Both these names can be changed to other more intuitive names in the Property Sheet. Instead of Combo1, we could write Forms!Form1!Combo1 which would be the qualified name of the Combo Box control. But since this query will only be used in the context of this form, Combo1 is enough. If we later change the name of the control or the form, we will have to change the query manually, as Access will not automatically update it. It is therefore good practice to specify the names of components and objects before starting referring to them.

We have now instructed the second Combo Box to select value from the table Song where the composer is the composer currently selected in the first Combo Box. Let's try out our form!

Well, the first time we selected a composer the list of songs got updated, but when we choose another composer then the list of songs remained unchanged. This is where our macro comes in handy. We can go back to the property On Change of the first Combo Box and then go back to the macro editor by clicking on ellipsis next to the property. We can now add a second action to our macro. This time we want to refresh a component, so we can use the action Requery for this. The action Requery has only one property. This property must be set to the name of the form control whose Row Source is to be refreshed (in our case Combo2):





We can now save the macro and return to our form. We can now try our form again and we will conclude that it works as we want it to.

We can now save our form, perhaps giving it a better name like “Songs based on composer”.

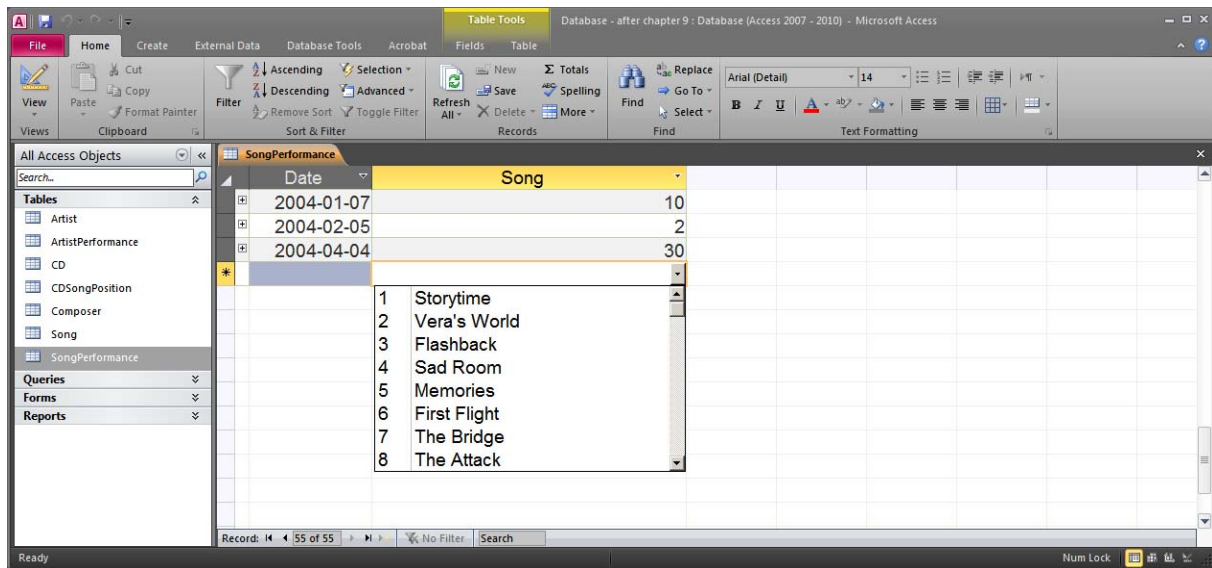
A version with the database with everything created so far is available, as usual, at <http://coursematerial.nikosdimitrakas.com/access/>.

## 9 Other Useful Tips

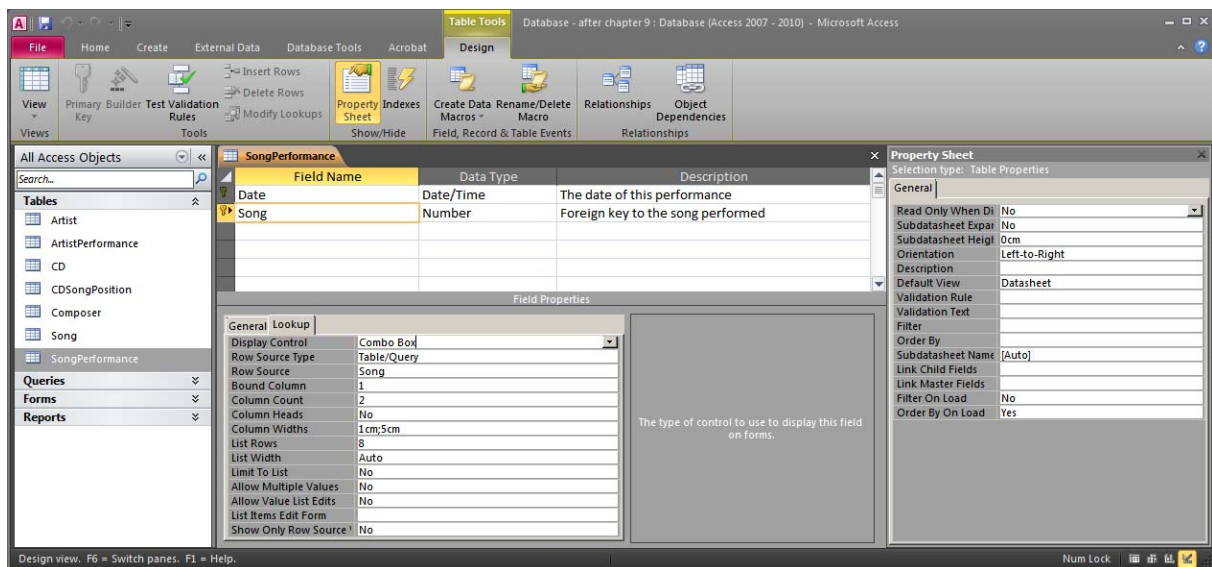
The tips presented in this chapter can be useful in many situations. There are many things in Access that can be useful, but not that easy to know how to do them. The following tips cover things that many find useful when working with Access.

### 9.1 Tip 1 - Lookups For Tables

When inserting values to a table using the datasheet mode, it can be useful to have a Combo Box that helps us select a value for foreign keys. This is similar to what we did when we created forms, but it can also be done without having a form. What we have to do is to define in the Table Design View where Access should look up values for a particular column. We could for example do this for the table SongPerformance, so that we can select the song like this:



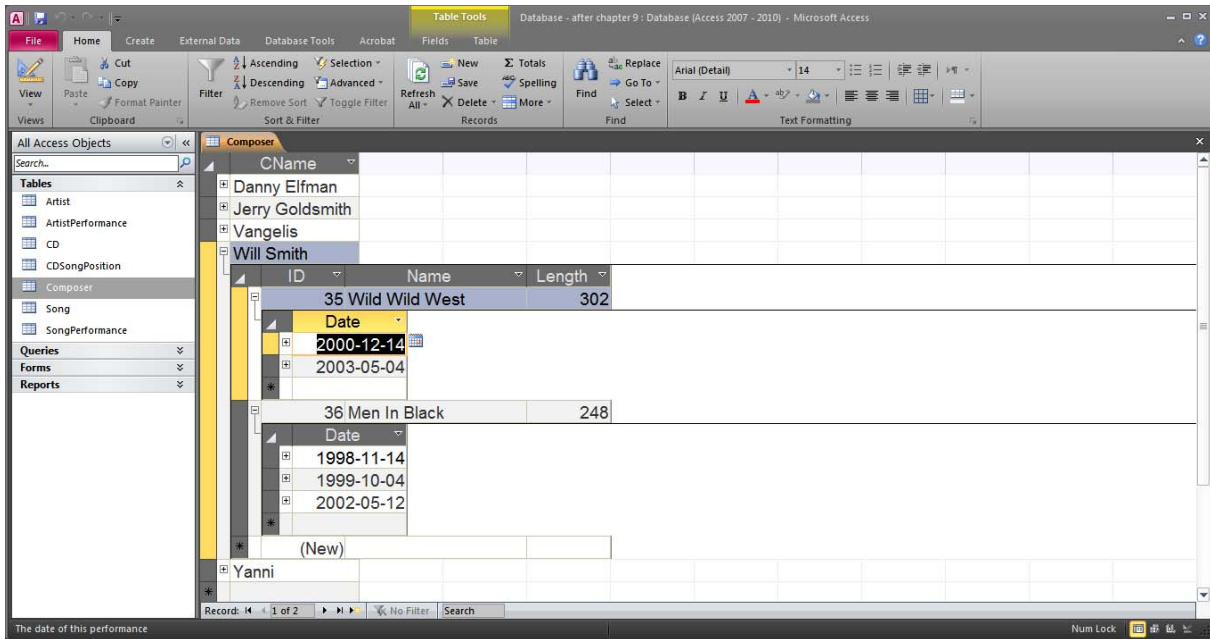
In order to do this we must look at the lookup properties of the column Song in the Table SongPerformance. We have to set the Display Control property and then the Source Row, Bound Column and Column Count properties (perhaps even Column Widths and List Rows):



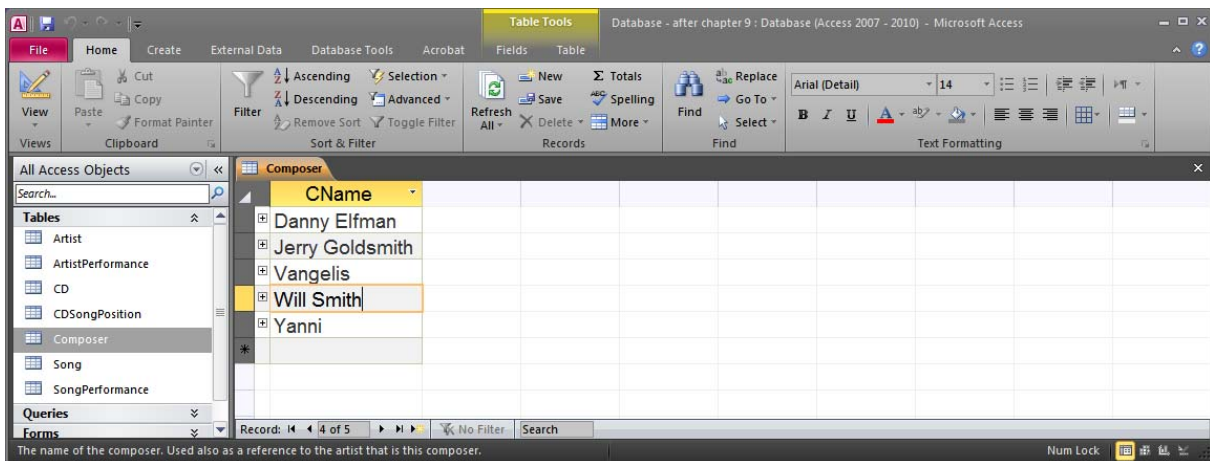
The Bound Column value tells Access which column from the table Song that should be linked to the column Song. The Column Count tells Access how many columns from the table Song that should be displayed in the combo box.

## 9.2 Tip 2 - Viewing Subtables

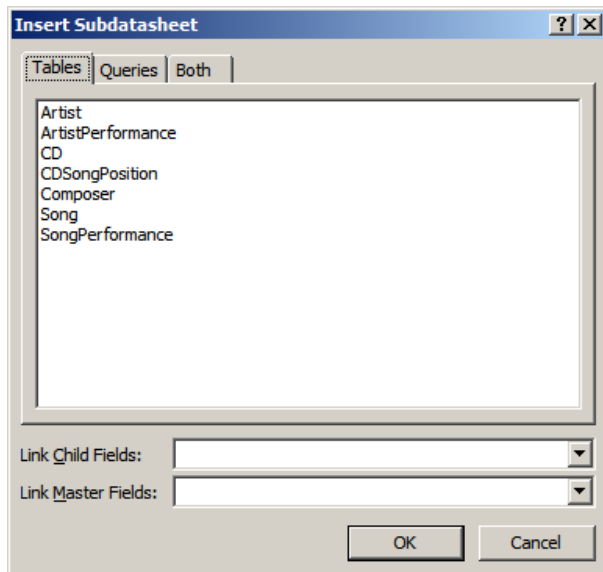
Another way to speed up data input and viewing without making forms, reports or queries, is by browsing the data in tree structures. We can for example open the table Composer and then see the songs composed by each composer and the performances of each song. This would look like this:



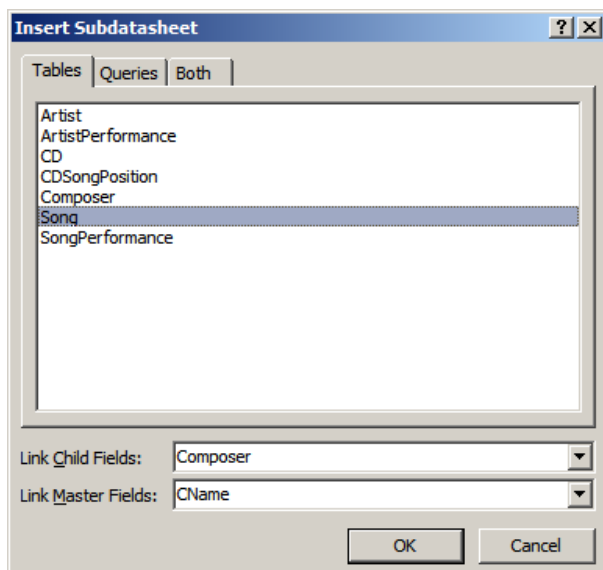
To do this we can click on the little plus-sign (+) next to the name of a composer. When we open the table it looks just like this:



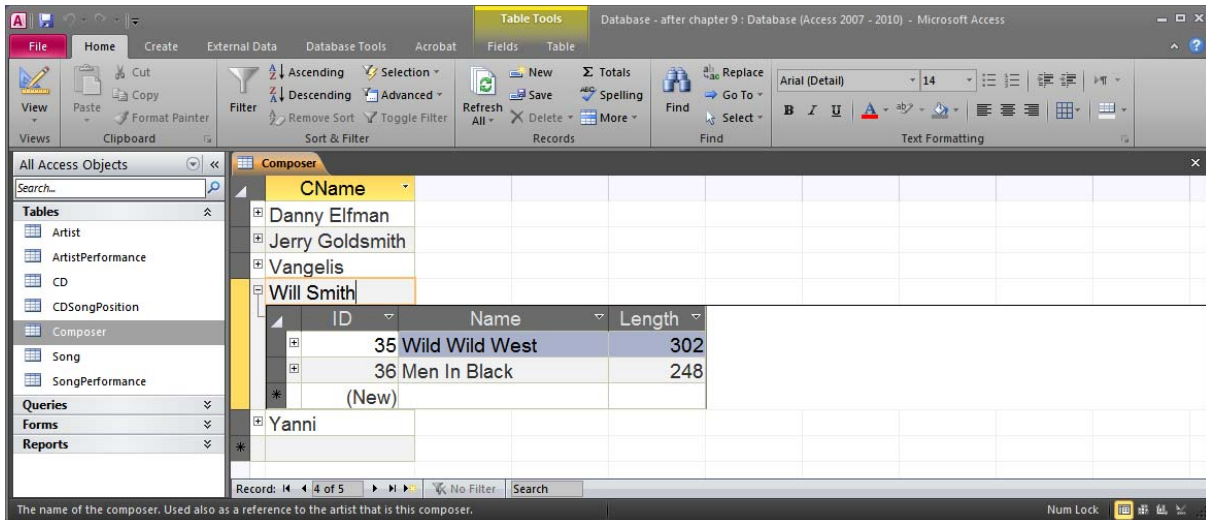
When we press on any of the plus-signs next to the composer names, Access will automatically open a tree based on the relationships that this table has. If there are many relationships, then Access will ask us to choose which one we want to use:



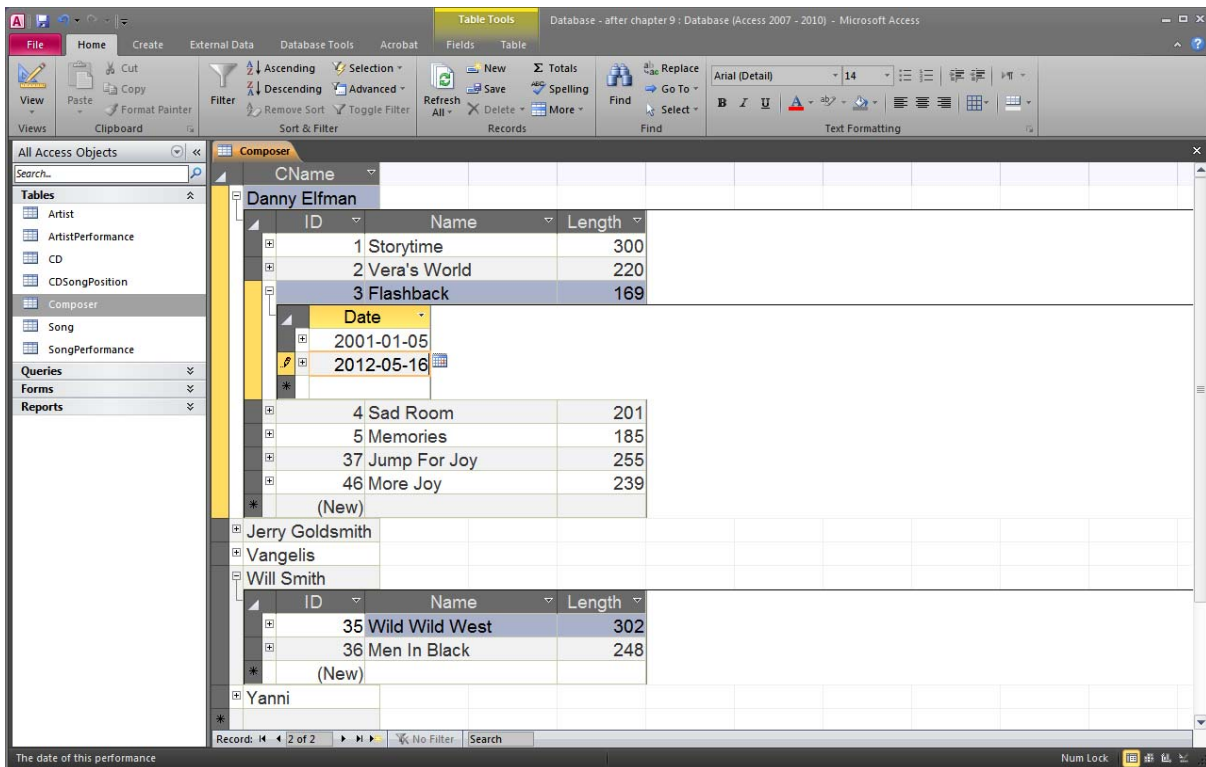
By just clicking on the table Song, Access will suggest the columns used for this connection:



The songs of the selected composer will now be visible:



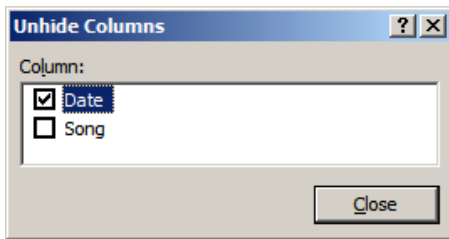
We can do the same thing to go deeper into the structure, and see for example song performances. In this structure we can also insert new rows (on any level/table). We can for example add a new song performance of the song Flashback:



If we want to close a tree structure and then select a different subtable we must place the cursor on the relevant level and then select, on the ribbon, Home > Records > More > Subdatasheet > Remove. If this option is unavailable, the subdatasheet can be configured in the Property Sheet while the table is in Design View.

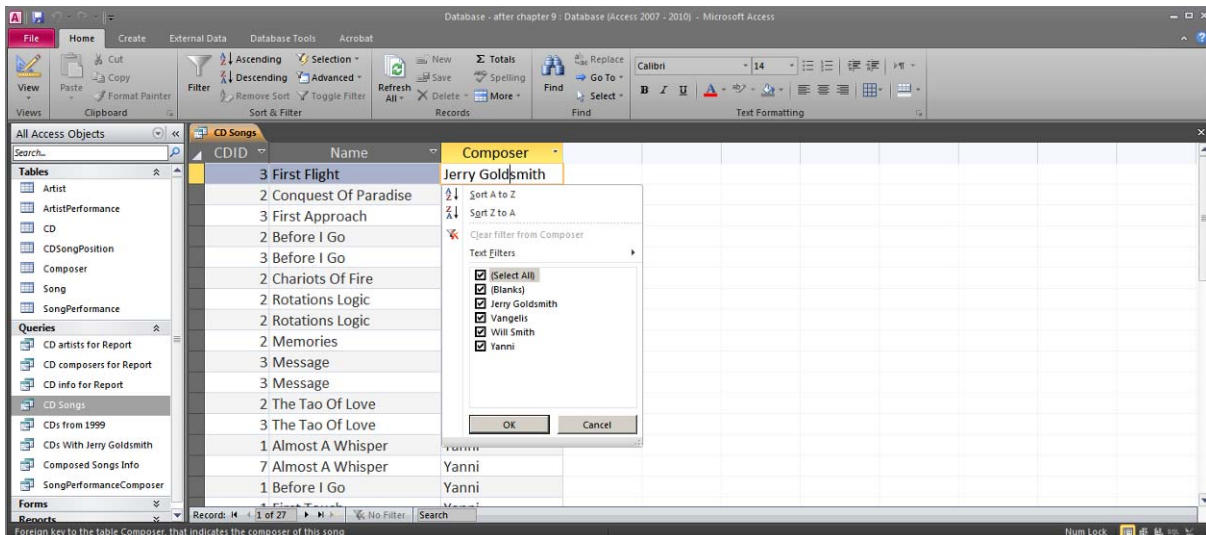
When working with subdatasheets, Access will automatically hide the columns that are visible in the upper level (in the example above, the foreign key column Song in the table SongPerformance is hidden). If a column is hidden but we still want to see it, then we can

unhide it by opening the Unhide Columns dialog from the Home > Records > More menu on the ribbon. The Unhide Columns dialog should appear and we can select what we want to see:



### 9.3 Tip 3 - Sorting And Filtering

When we are looking at the contents of a table, or the result of a query, it can be interesting to do some quick sorting or filtering (which of course can also be done in the query). There are many options for filtering and sorting in Access on the ribbon under Home in the Sort & Filter group. By simply pressing the Filter button while a particular column is selected, a filter pop-up menu will appear:



There are also buttons for sorting that will sort the table (or query result) on the active column. Note, that sorting settings may become permanent if we save the changes to the design of the query or table. The sorting can then be removed by selecting Home > Remove Sort.

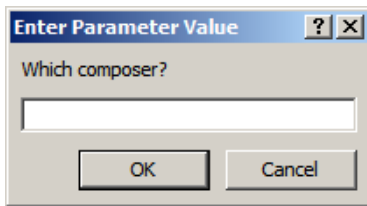
### 9.4 Tip 4 - SQL Parameters

A quite cool feature in Access is the possibility to specify parameters in queries. Access will then ask the user to provide the values of the parameters when the query is executed. For example we may want to see which songs are composed by a specific composer, but we don't want to hardcode a particular composer's name in our query. We can write a SELECT statement like this:

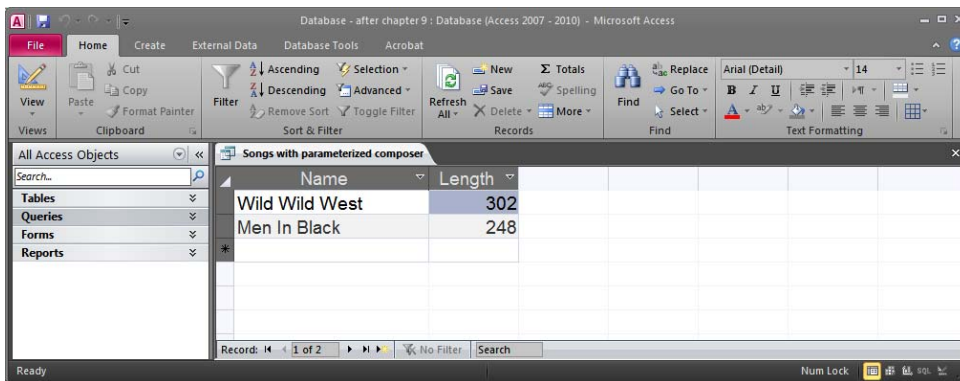
```
SELECT Name, Length
FROM Song
WHERE Composer = [Which composer?]
```



We can save this query as “Songs with parameterized composer” and then try to execute it. Access will immediately detect that there is a parameter and ask the user for a value:



We can type “Will Smith” and see the result:



Anything that is not a column, table, query or other database object is considered a parameter. Access will try to match any name to column, table, etc in the current scope, then it will move to the next scope and the user will be the final scope.

- ① The same principle can be used to connect the value of a form component to a query (as we did in chapter 8).

### 9.5 Tip 5 - Nesting SELECT Statements – COUNT(DISTINCT)

In Access it is possible to have nested SELECT statements in the FROM clause. Generally, we would then like to write the code straightforward like this:

```
SELECT Name, Age
FROM (SELECT * FROM Artist WHERE Age>40) AS newtable
WHERE Age<60
```

However, in some cases this notation, i.e. using normal parentheses, may not work (depending on the number of nested levels or the operations performed on each level). In order to get around this problem we may have to use the following notation:

```
SELECT Name, Age
FROM [SELECT * FROM Artist WHERE Age>40]. AS newtable
WHERE Age<60
```

The nested SELECT statement is now placed within “[“ and “]” followed by a full stop (.).

For the above SQL statement it is not necessary to have a nested SELECT. It is only used as an example.

One particular situation when we may need to use a nested SELECT statement is when we want to do a COUNT(DISTINCT). As this is not supported in the current version of Access, we must first have a nested SELECT DISTINCT in order to later (in the outer SELECT) do a normal COUNT(). Here is an example:

We would like to know how many artists that are featured in each CD. We would normally do this using a COUNT(DISTINCT), in accordance with standard SQL, like this:

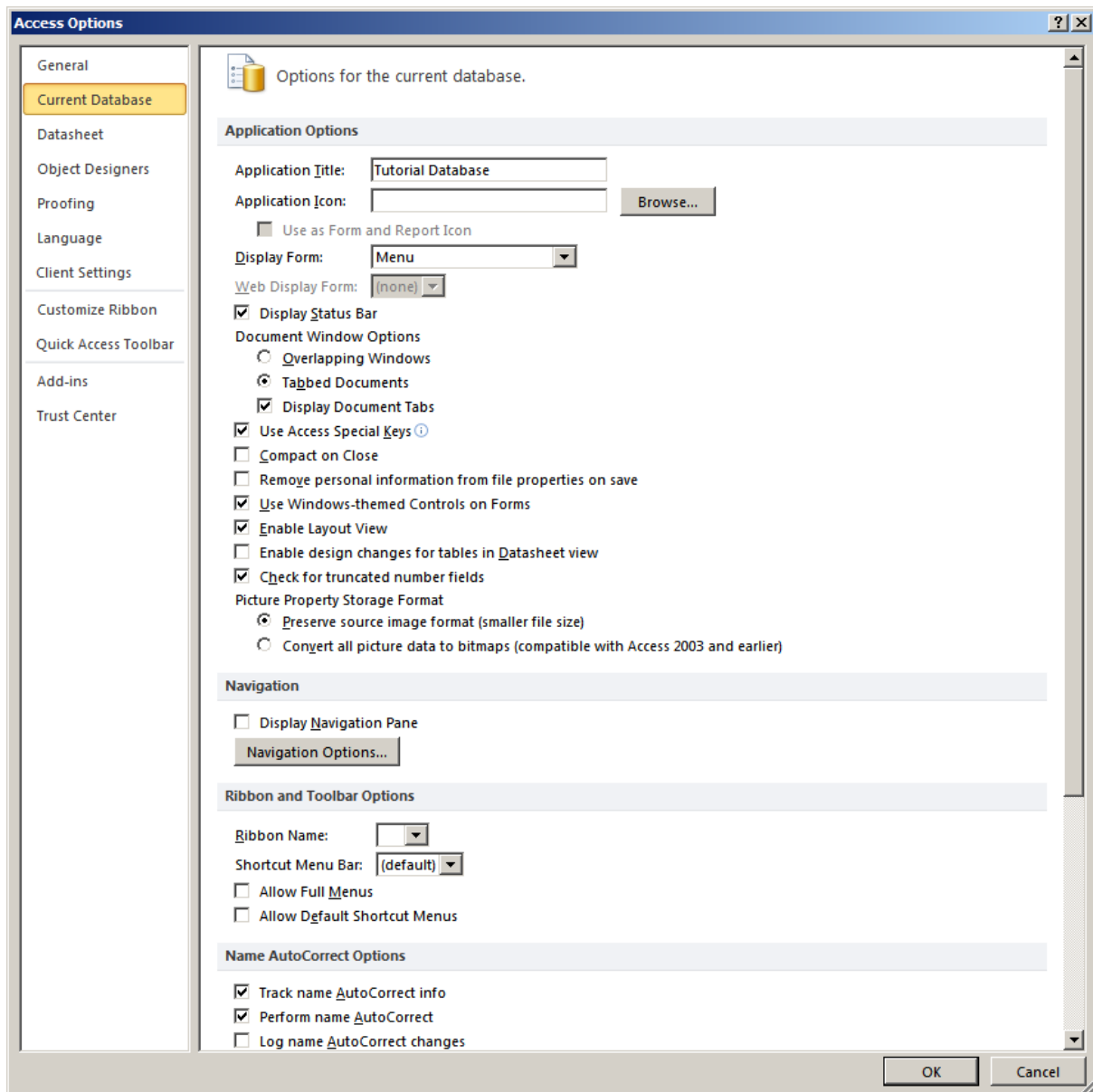
```
SELECT CD.ID, CD.Title, COUNT(DISTINCT ap.Name) AS ArtistAmount
FROM CD, CDSongPosition sp, ArtistPerformance ap
WHERE CD.ID = sp.CDID
AND sp.Date = ap.Date
AND sp.Song = ap.Song
GROUP BY CD.ID, CD.Title
```

This statement is correct, but not supported by Access. In order to achieve the same thing in Access, we would have to split the statement into two, according to this:

```
SELECT ID, Title, COUNT(Name) AS ArtistAmount
FROM (SELECT DISTINCT CD.ID, CD.Title, ap.Name
      FROM CD, CDSongPosition sp, ArtistPerformance ap
      WHERE CD.ID = sp.CDID
      AND sp.Date = ap.Date
      AND sp.Song = ap.Song) AS innertable
GROUP BY ID, Title
```

## **9.6 Tip 6 - Application Start-Up**

When we create an Access application with many forms it is often so that we want a particular form to automatically open at start up. We may also want to control which menus that should be available in the Access window (so that the users don't get access to the database other than through forms). All of this can be arranged in the Access Options under Current Database:



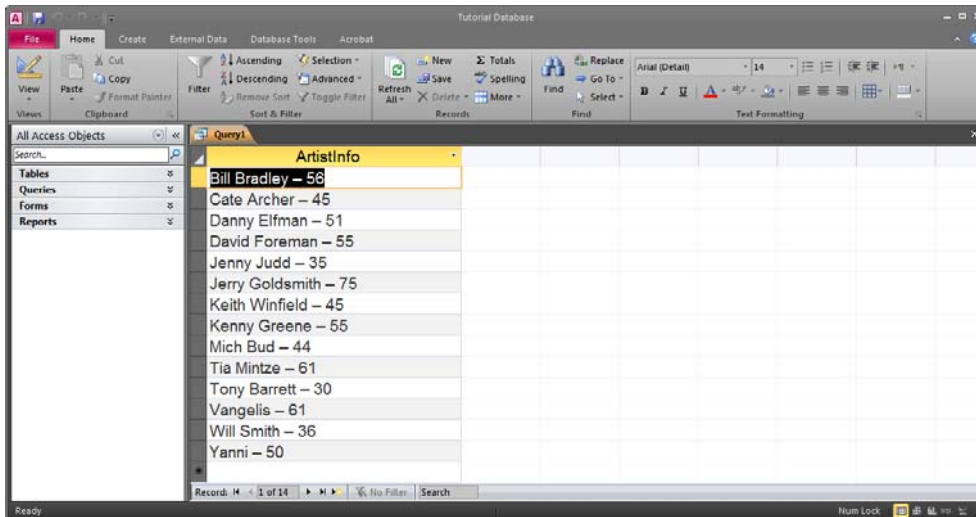
If we choose to hide or deactivate certain options and menus and then open the database file, we will no longer have access to the options. To bypass these startup settings, we must press and hold the shift button while opening the database file. Then we can open the options and reconfigure the settings.

### 9.7 Tip 7 - Concatenating Columns

When we work with forms, it can sometimes be useful to have one column instead of many. We can in such cases concatenate columns in the SELECT clause of our SQL statement. We can for example write the following SELECT statement:

```
SELECT Name & ' - ' & Age AS ArtistInfo
FROM Artist
```

The result would be just one column:



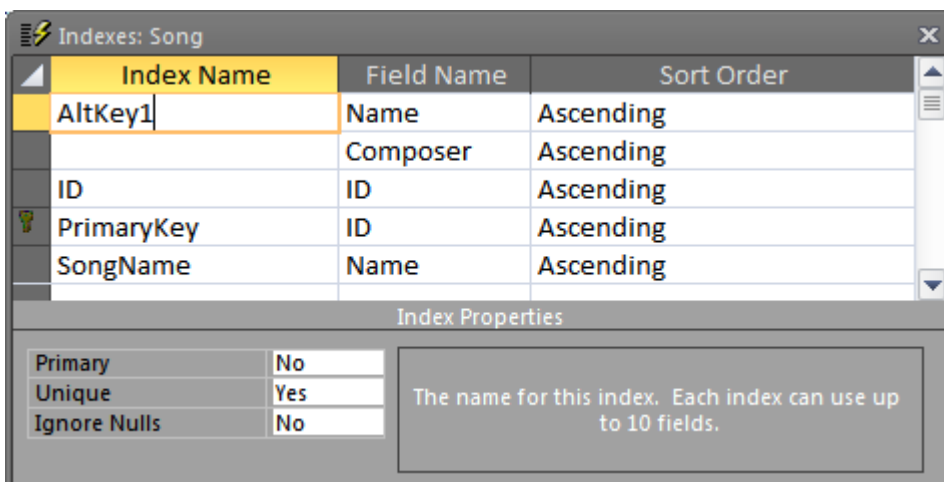
① This is actually nothing specific for Access. This is included in standard SQL. The actual keyword or symbol used for concatenation may vary from product to product.

### 9.8 Tip 8 - Using Forms To Find Records

A combo box can be used in a form as a search field for finding a record. The form will automatically move to the first record that matches the value of the combo box. To add such functionality, add a combo box to your form and select the appropriate choice in the wizard.

### 9.9 Tip 9 - Keys And Indexes

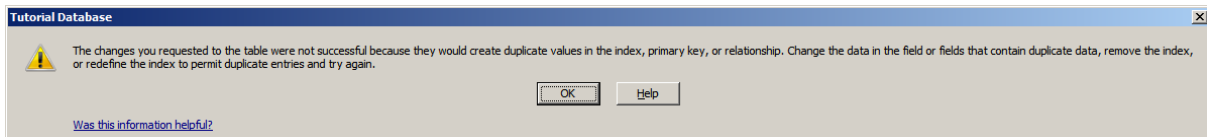
When designing a table it is possible to define primary keys and other rules. But it is also possible to do this in the Indexes window. In this window we can also see other keys and indexes that Access has created. These may be created due to table relationships or orderings that we have applied to the table. For example the indexes of the table Song may look like this (open the Indexes window by pressing the Indexes button on the ribbon under Design while in Table Design View):



Here we can add our own indexes and keys. For example we can add the alternate key (shown above as AltKey1) for the table Song. This is a composite key that combines the column Composer and the column Name, that is to say, a composer may not compose two songs with the same name. To add a new index, we must specify an Index Name and then the fields

(columns) that are included in this index. The first row specifies the index name and the first field in the index, as well as the sort order. Any field in a row without an Index Name immediately under that, will also belong to that index definition. We must also define the appropriate property values for our new index (in this case Unique Yes):

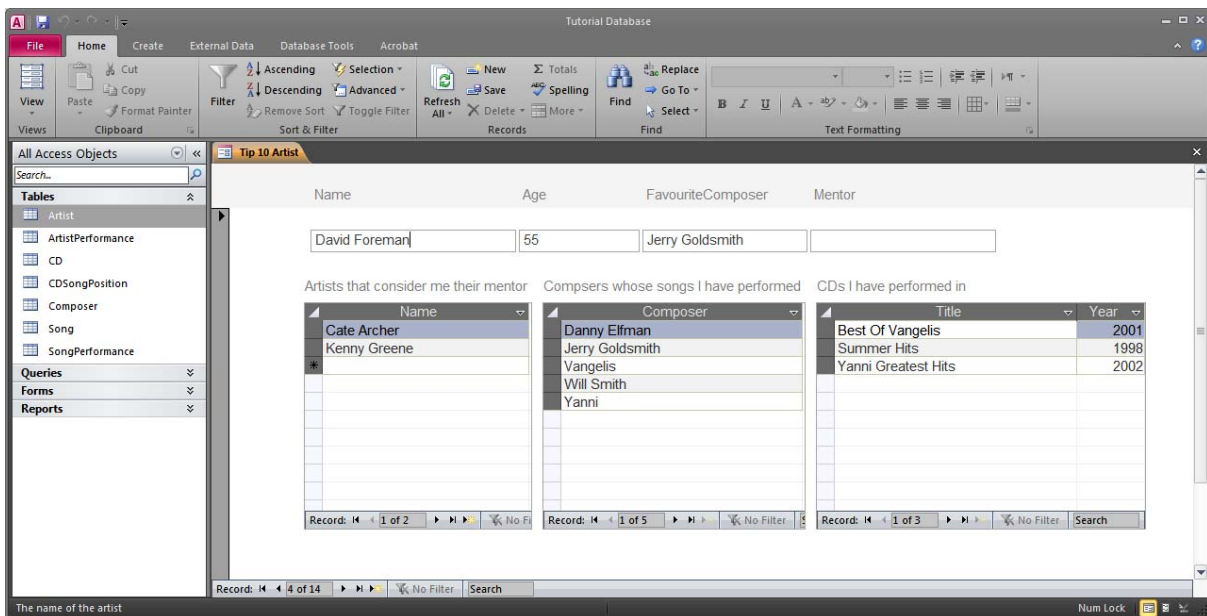
If we try to add a new song with a combination of Composer and Name that already exists, the database will stop us:



- ① If you experience that a table behaves strangely, then check that there are no unwanted indexes. Note that Access creates non-unique indexes automatically for fields with certain names, such as id or name.

### 9.10 Tip 10 - Multiple Subforms

We saw how we can create forms with subforms in sections 6.3 and 6.4. But sometimes we may want to have a multiple subform structure. This kind of structure is not different to make than when we have a single subform. That is, there is no difference when making them manually, but the wizard cannot manage this kind of forms. The following form is a single form based on the table Artist with three subforms. One that shows the artists that have the current artist as mentor, one that shows the composers that this artist has performed songs of, and one that shows the CDs that this artist has performed in:



We may want to have more than two levels of forms, i.e. subforms in subforms. This is possible to achieve and really no different from just working with two levels.

### 9.11 Tip 11 - Division In Access

In Access division works with both NOT EXISTS and NOT IN. In our case we had two examples of information needs that require a division. Here are three possible solutions for the need "Which artist has performed in at least one song of each composer?":

The double NOT EXISTS variant:

```
SELECT Artist.Name
FROM Artist
WHERE NOT EXISTS (SELECT *
    FROM Composer
    WHERE NOT EXISTS (SELECT *
        FROM ArtistPerformance ap, SongPerformance sp, Song s
        WHERE s.ID=sp.Song
        AND sp.Date = ap.Date
        AND sp.Song = ap.Song
        AND ap.Name = Artist.Name
        AND s.Composer = Composer.CName));
```

The NOT EXISTS – NOT IN variant:

```
SELECT Artist.Name
FROM Artist
WHERE NOT EXISTS (SELECT *
    FROM Composer
    WHERE CName NOT IN (SELECT Composer
        FROM ArtistPerformance ap, SongPerformance sp, Song s
        WHERE s.ID=sp.Song
        AND sp.Date = ap.Date
        AND sp.Song = ap.Song
        AND ap.Name = Artist.Name));
```

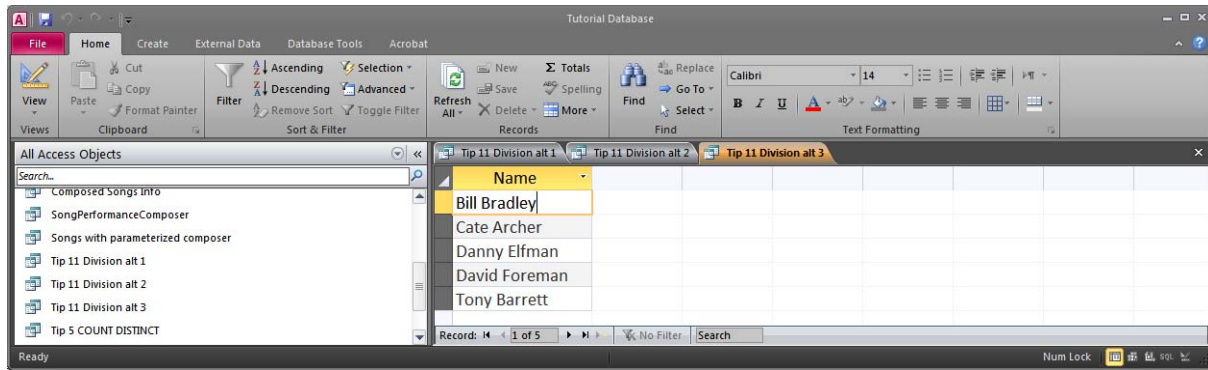
And the hard way:

```
SELECT innertable.Name
FROM [SELECT DISTINCT a.Name, Composer
    FROM Artist a, Song s, SongPerformance sp, ArtistPerformance ap
    WHERE s.ID=sp.Song
    AND sp.Date = ap.Date
    AND sp.Song = ap.Song
    AND ap.Name = a.Name]. AS innertable
GROUP BY innertable.Name
HAVING COUNT(innertable.Composer) = (SELECT Count(*) FROM Composer);
```

The nested table in the FROM clause is required because Access does not support COUNT(DISTINCT *column*) that is specified in standard SQL.

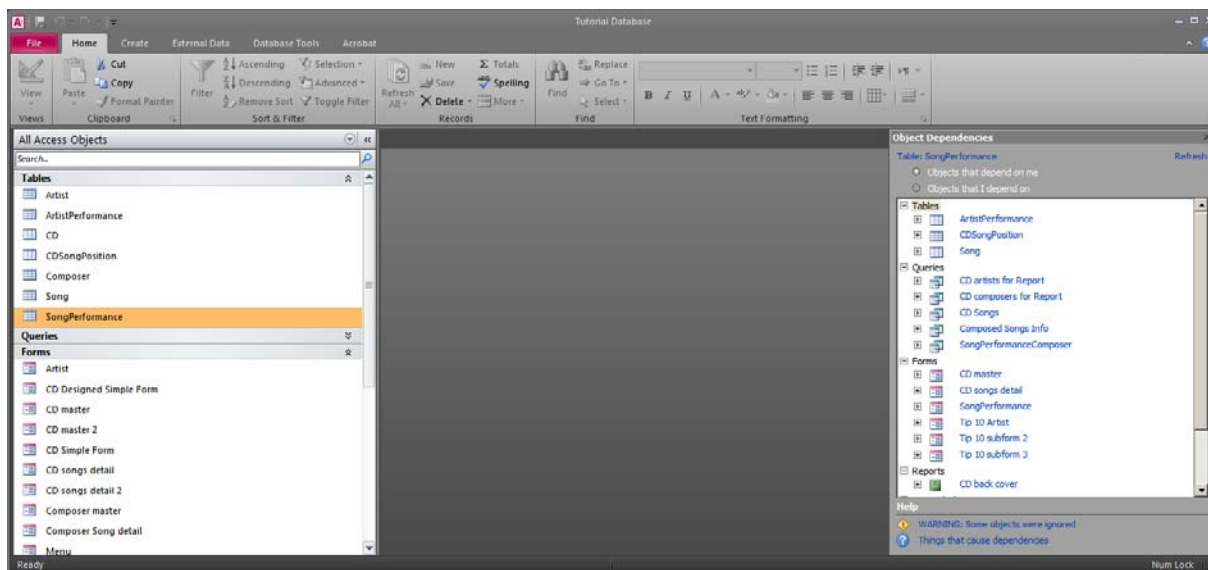
All three variants are equivalent and their result is a list of the five artists that have performed songs of all composers:





## 9.12 Tip 12 - Object Dependencies

Another interesting feature of Access is the possibility to see all dependencies between objects. This can be useful when we want to delete an object. We can look at its dependences and then decide if it is okay to remove it or not. Let's say for example that we would want to see if the table `SongPerformance` has any dependencies. We can highlight the table in the Object browser and select Object Dependencies from the ribbon under Database Tools. We can now see all the dependencies of the table:



We can see that there are other tables, queries, forms and reports that depend on this table. We can also switch to see what this table depends on by selecting "Objects that I depend on" instead of "Objects that depend on me".

## 9.13 Tip 13 - Copying Objects Between Databases

When working in groups, it is often so that some queries, forms, and reports are developed by one person in one database (.accdb or .mdb file), while others are in a separate database. At the end the goal is to have all the database objects in one file. This is not a problem when working in Access. We can copy and paste objects between databases. We must first open the database that contains the object to be copied, copy it, open (in the same Access Window) the other database and paste. It is also possible to just do a drag and drop (or copy-paste) between two databases open in different Access windows. When copying forms and reports, make sure that all the queries, macros, etc. are also available in the target database; otherwise the form or report will not function properly.

### 9.14 Tip 14 - Handling NULL

Sometimes it is necessary to return something where there is NULL in the database. For example we may not want to just have an empty field in a form or report when an artist doesn't have a mentor. The function NZ (in other products called COALESCE) can help us with that. In the following SQL statement we tell Access to return a specific value whenever the value of the column Mentor is null:

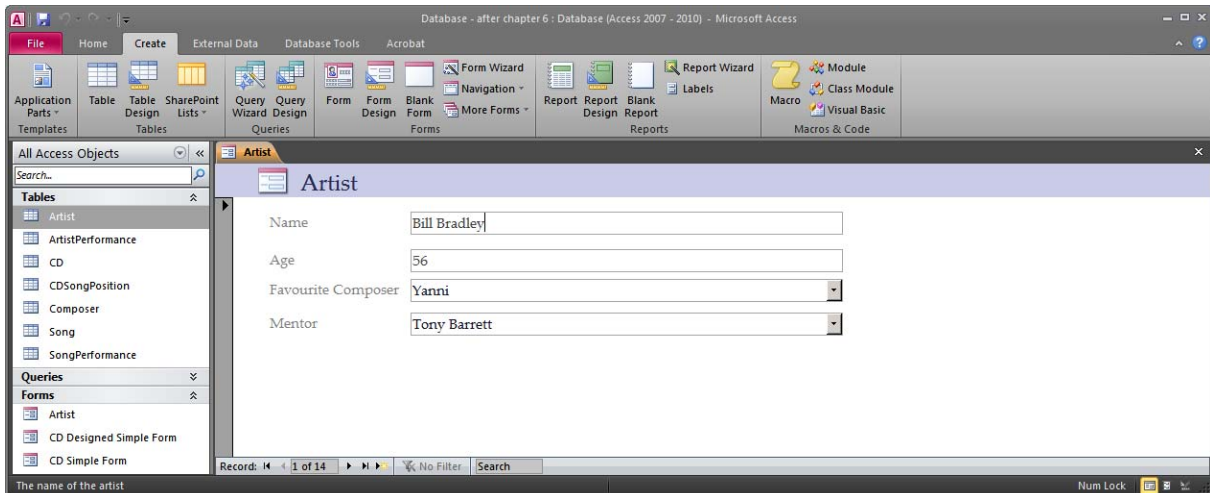
```
SELECT Artist.Name, NZ(Artist.Mentor, "Self-inspired") as Mentor
FROM Artist;
```

The result would look like this:

Name	Mentor
Bill Bradley	Tony Barrett
Cate Archer	David Foreman
Danny Elfman	Jerry Goldsmith
David Foreman	Self-inspired
Jenny Judd	Self-inspired
Jerry Goldsmith	Self-inspired
Keith Winfield	Tia Mintze
Kenny Greene	David Foreman
Mich Bud	Jerry Goldsmith
Tia Mintze	Self-inspired
Tony Barrett	Keith Winfield
Vangelis	Jerry Goldsmith
Will Smith	Keith Winfield
Yanni	Self-inspired

### 9.15 Tip 15 - Business Rules

It is considered a good idea by many, perhaps most, database theorists and practitioners to always model business rules as close to the database as possible. Some simple rules were already included in our database in section 4.1.3. Using the Validation Rule property for columns is not always possible though. Some rules are too complex to be expressed there. The next level is adding the business rules as validation rules in forms. This way, the user interface will restrict the user from making an invalid choice. We already saw a few ways of controlling the available data in forms in chapter 6 (for example by look-up Combo Boxes and query based forms) and in chapter 8 (by macros). In this section we will look at a very useful function, namely DLookup. This function can be used to look up a value in another table or query. We can now modify the form we created in section 6.2 so that we can restrict the user from choosing a mentor that is not older than the artist. Remember the form?



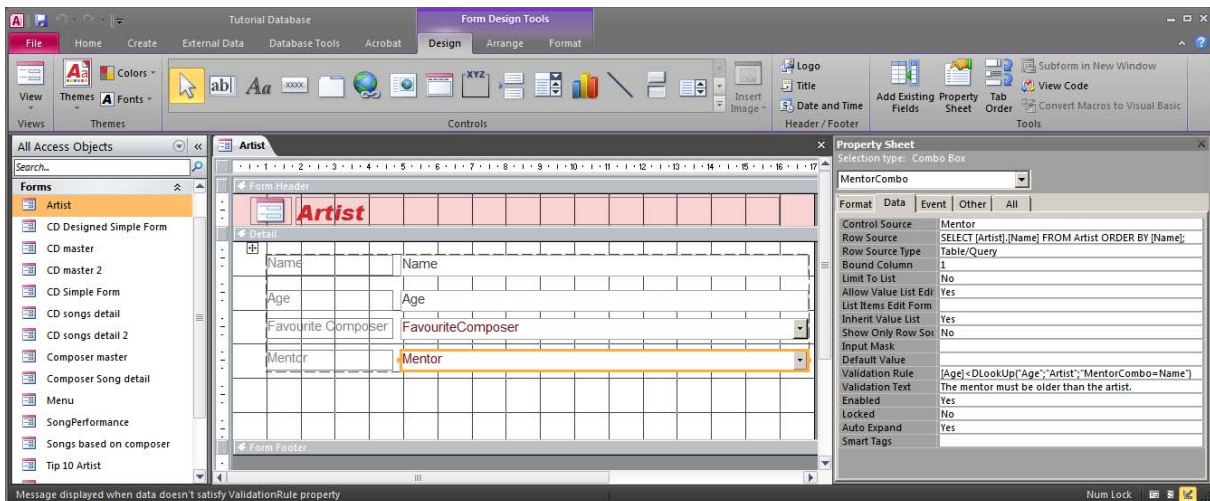
We can now add a validation rule for the Mentor combo box. We switch to the Design View and look at the properties of the Combo Box. The property we want to change is the Validation Rule (and maybe also the Validation Text so that we can give a custom error message to the user).

The property Validation Rule can have the following value:

`[Age]<DLookUp("Age";"Artist";"MentorCombo=Name")`

This checks that the Age of the current artist is smaller than the Age of the Artist whose Name is equal to the current artist's Mentor. MentorCombo is the name of the Combo Box control.

We can also add an error message so the properties will look like this:



Try the form now! Will Smith, for example, who is young, will not be allowed to be the Mentor of Jerry Goldsmith, who is older:



Another way to achieve this restriction would be to make the Row Source of the Combo Box dependent on the value of the column Age. Perhaps deactivate the mentor Combo Box until a value has been entered in the age Text Box. This could be similar to what we did in chapter 8.

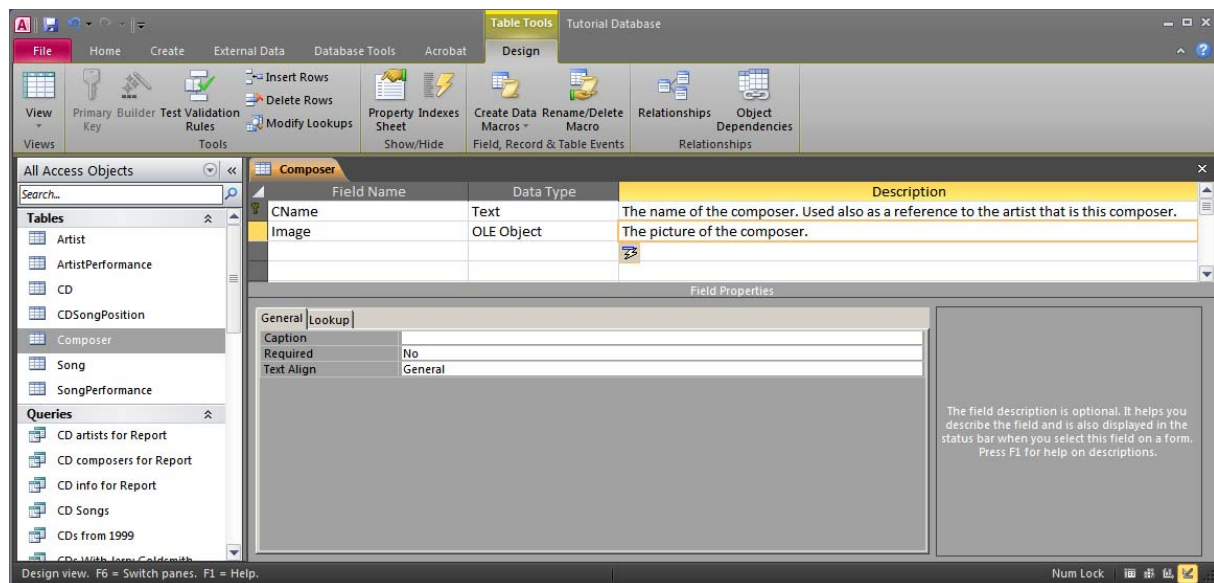
### 9.16 Tip 16 - Set Operators

Access does not support all set operators. In fact, the only one supported is the UNION operator. Intersection and difference must therefore be implemented by combining other operators like OR, AND, EXISTS, IN, NOT, etc.

### 9.17 Tip 17 - Multimedia

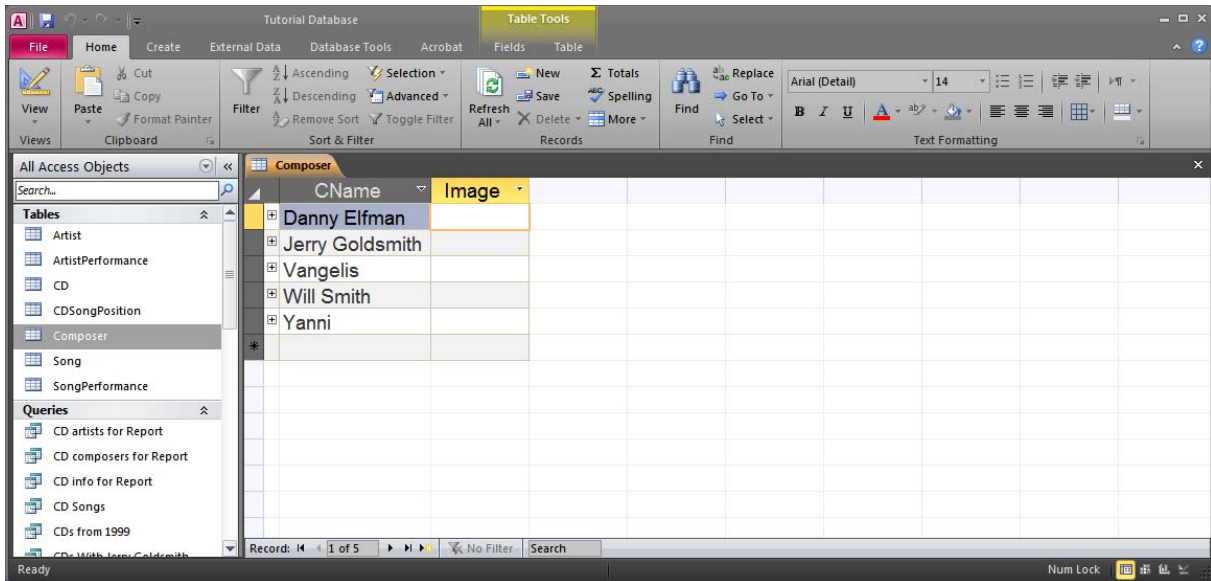
The database that we have seen so far uses only simple data types, i.e. text, numbers and dates. But Access supports other, more complex, types as well. For example we can store images and sounds in an Access database. Apart from storing multimedia in the database, Access also offers us the possibility to show and use the multimedia data in forms and reports.

In this section we will have a look at how to both store an image in the database, and show it in a form. To do this we will modify the table composer and add a field for the composer's image. To do this we simply open the table in design mode and create a new field of type OLE Object:

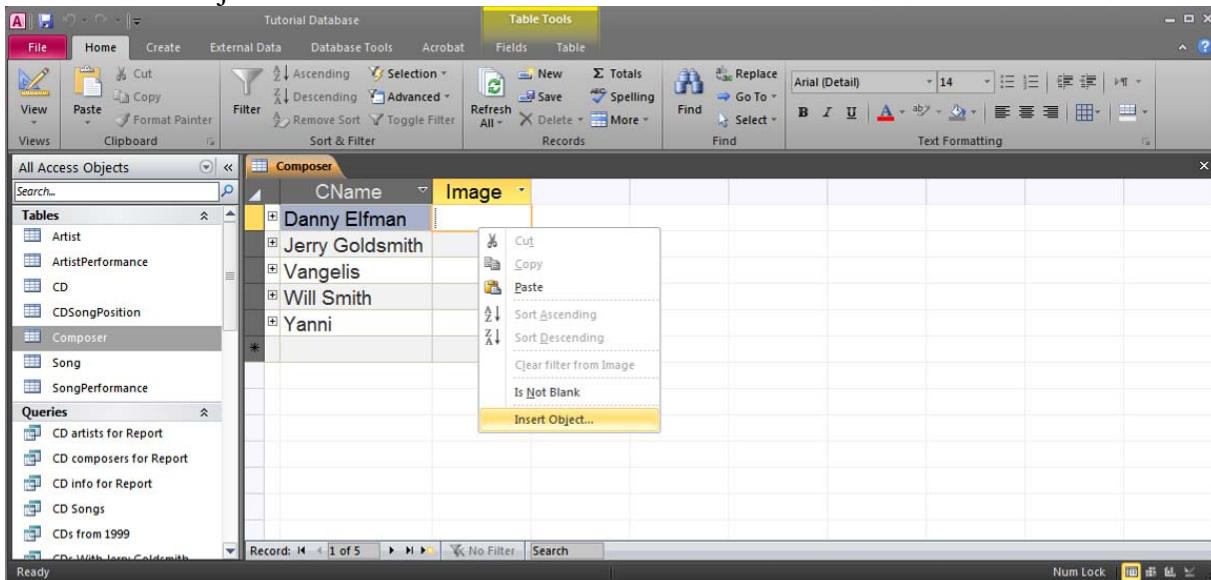


If we now open the table (after saving it), we will see that there is a field called Image, but we cannot edit it directly:

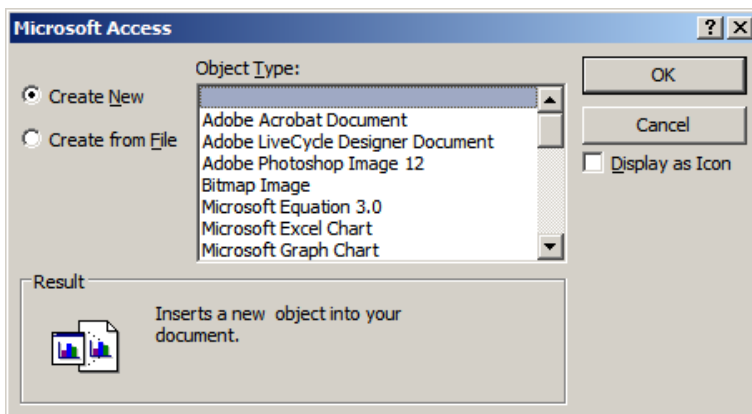


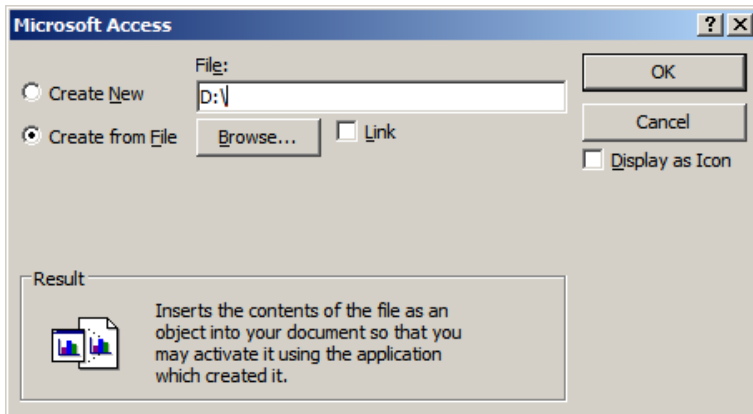


What we can do is right-click on a cell where we want to place an picture (a picture file) and select "Insert Object...":

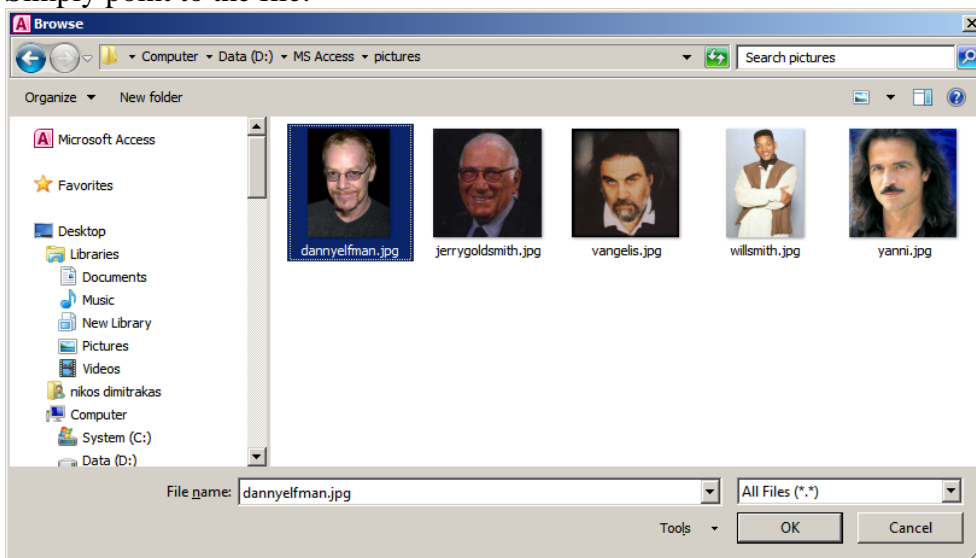


We will then see this dialog where we will choose "Create from File"

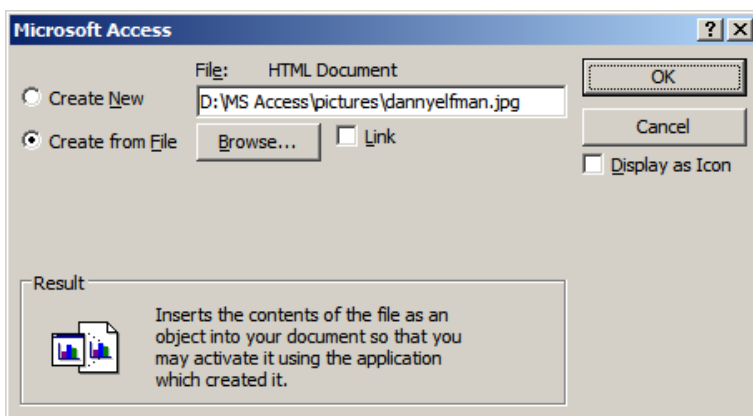




Simply point to the file:



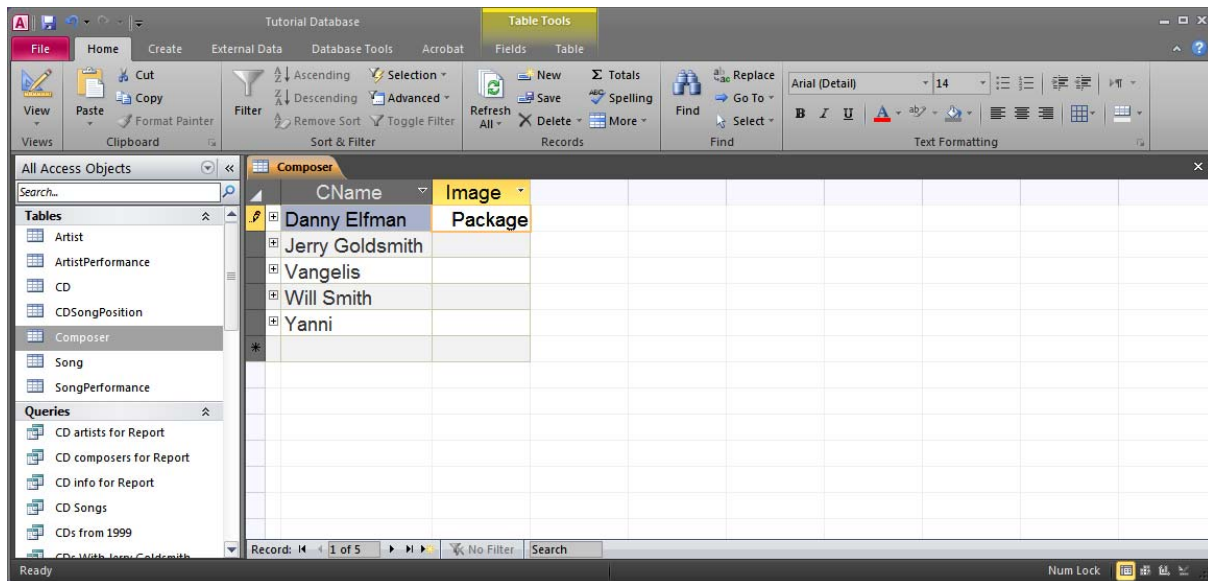
Press OK:



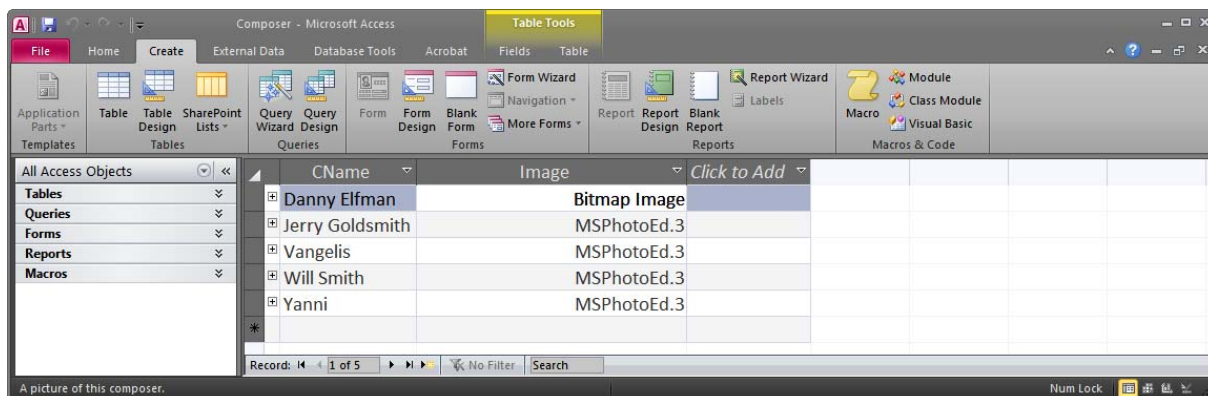
And then press OK again!

In the table view we can now see that there is a value in that cell:



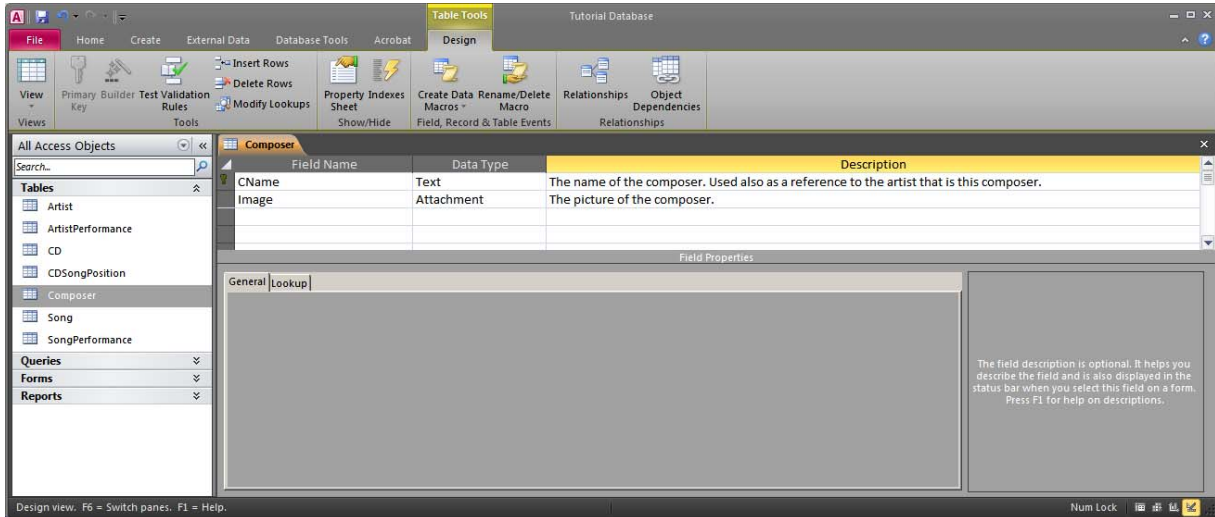


Note that if the value is "Package" then the picture may not be displayed correctly in forms and reports. The value should be instead, "Bitmap Image" for a bmp file or for example "Microsoft Photo Editor 3.0 Photo" for jpg or gif files. If we insert an image file and the value is just "Package", this indicates that there is no support in Access for that file type. If we get the value "Package" when inserting a jpg or gif file, then we probably need to install Microsoft Photo Editor, which is included in earlier versions of Microsoft Office (for example Office XP). When the pictures have been identified correctly by Access, the table should look like this:

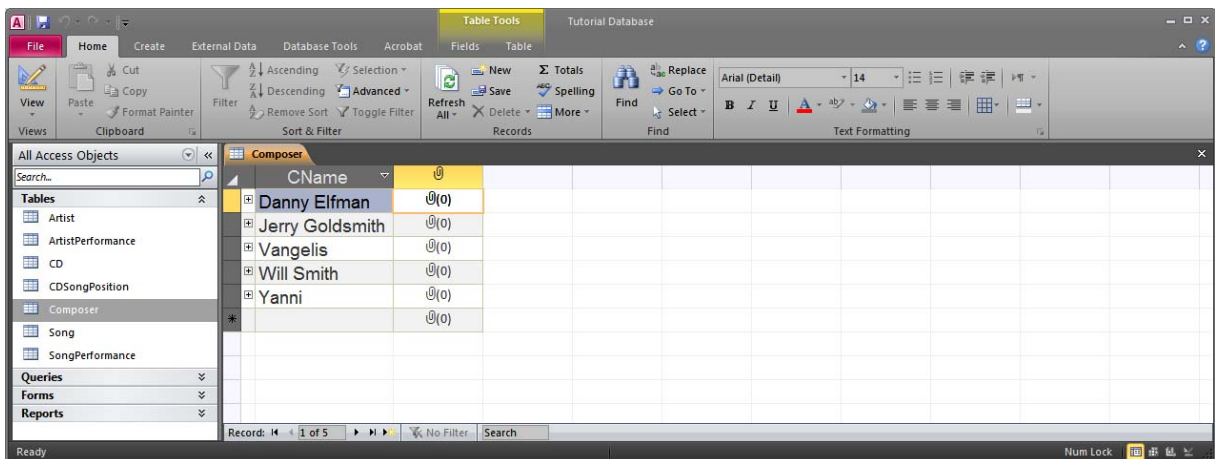


We can repeat the process for all composers that we want to store their picture in our database.

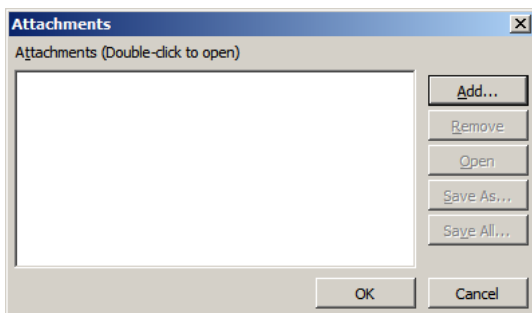
Another possibility is to use the data type Attachment. In that case the column Image will be able to contain the image files as attachments. This data type is supposed to be more flexible than OLE Object and uses storage more efficiently. However, it is not available in older versions of Access. We can remove the Image column and create it again with the data type Attachment:



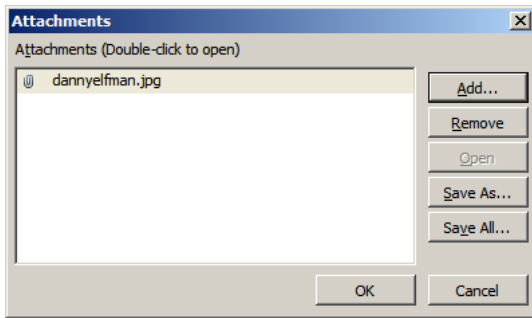
Adding data in the column Image is similar to when we had OLE Object as data type. We start by opening the table and the new column will show how many attachments each row has:



We can now right-click on a particular cell and choose Manage Attachments. The Attachments dialog will appear where we can add and remove attachments:



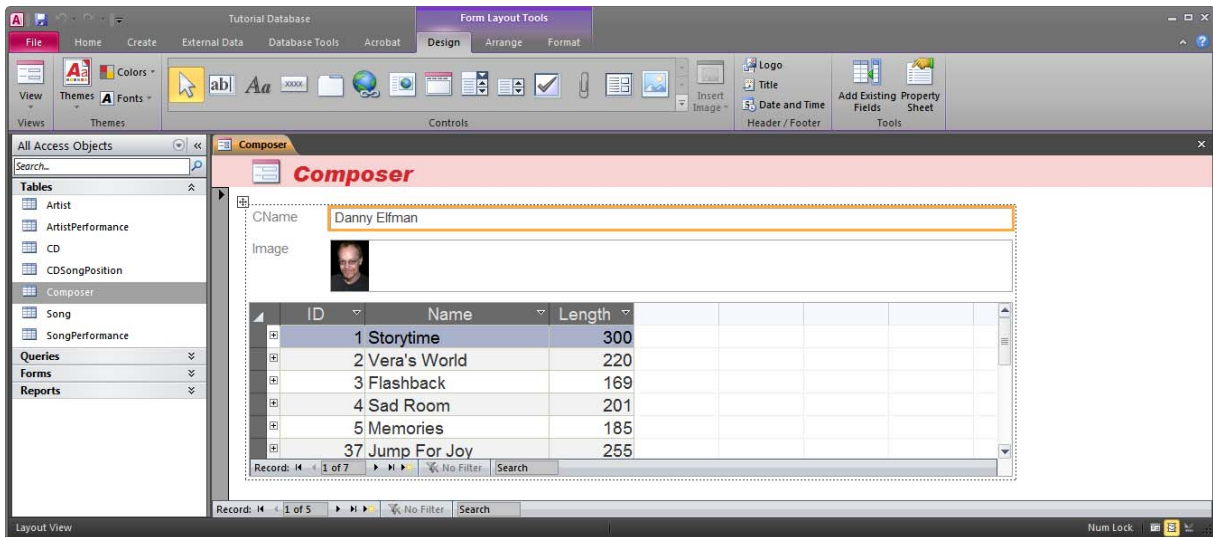
We can add the appropriate file as an attachment:



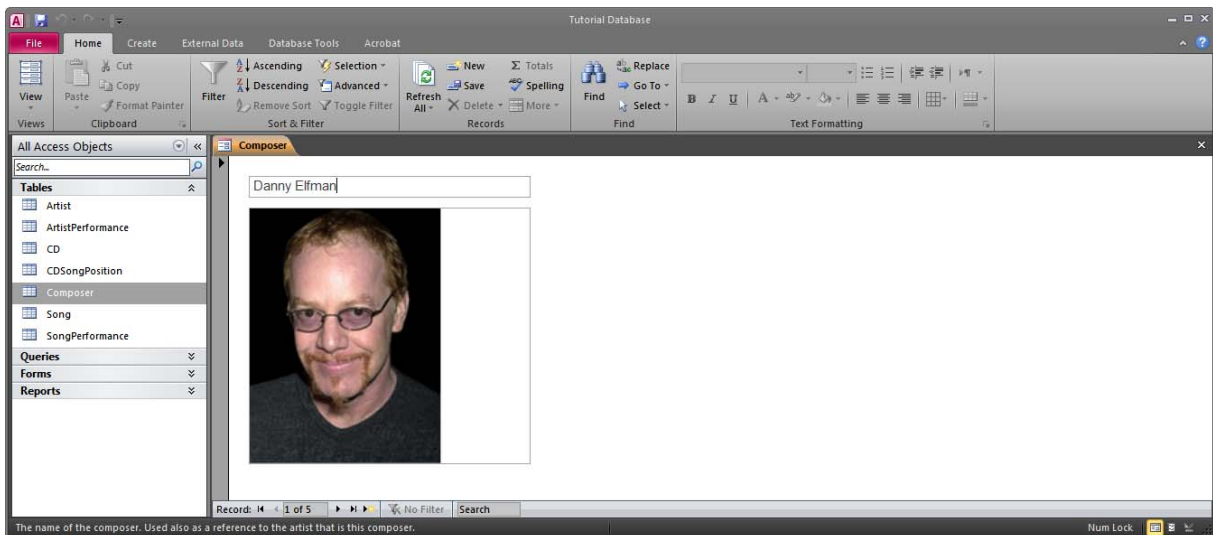
We can repeat this until all the rows have one attachment.

We can now make a quick form that will allow us to browse the composers with their pictures. We can create a new form in Design View based on the table Composer:

This will create a blank form, which will be connected to the table Composer. By default the created form will have a subform with the composed songs and the image will be too small:



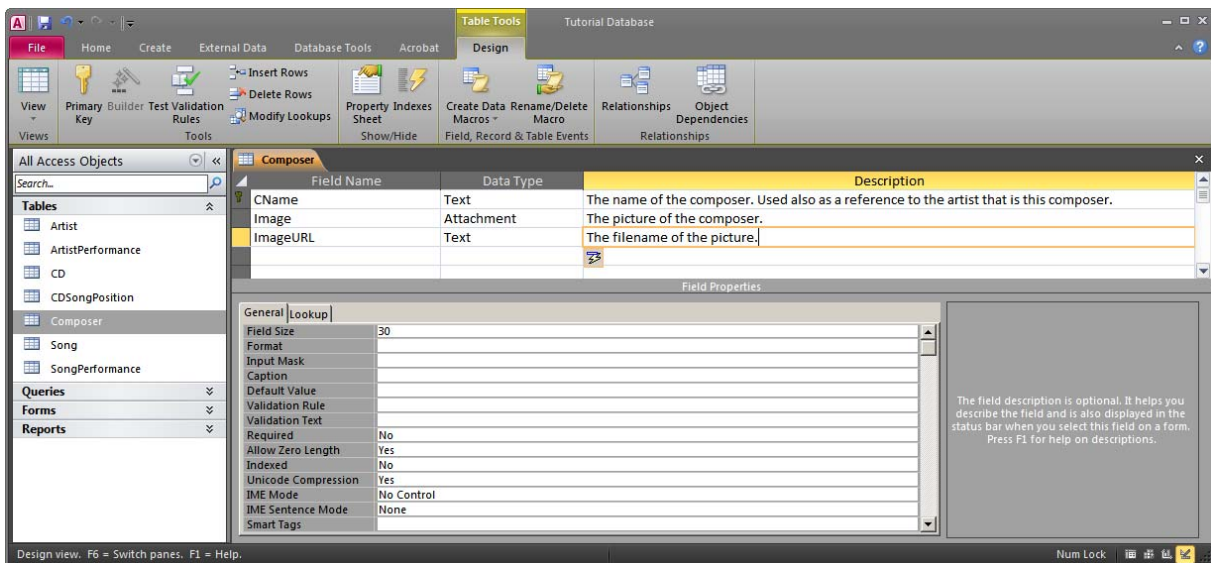
We can modify the layout:



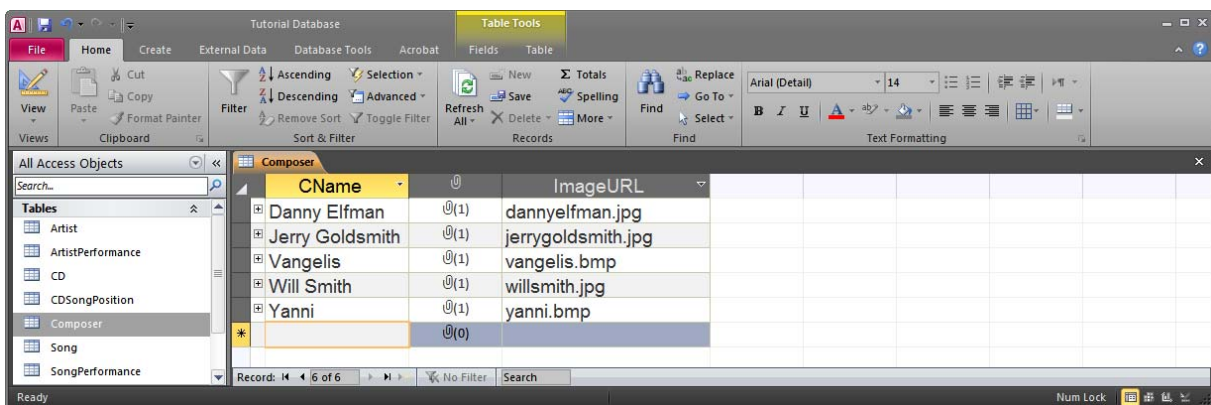
The component that shows the image is an Image control that is associated to the column Image (through its property Control Source). It can also be configured to zoom or stretch the picture (property "Picture Size Mode").

### 9.17.1 Storage Outside The Database

If the ways of embedding images into the database described earlier are not suitable to our needs, storing the images in the file system and storing the filenames in the database could be an option. In such case, the database will not have control of the images, so if an image (or a folder) is moved or renamed, the database will not find the image (or images). We could try this solution by adding a column to the table Composer:

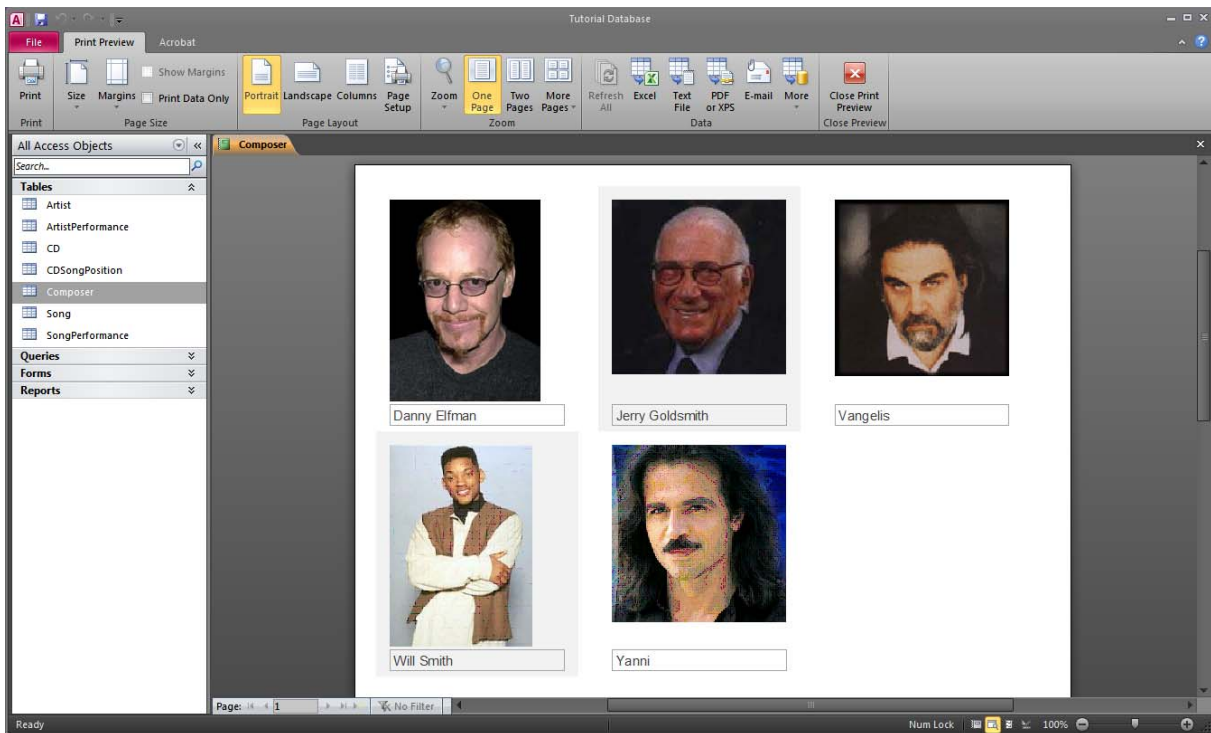


We can now update the content of the table:

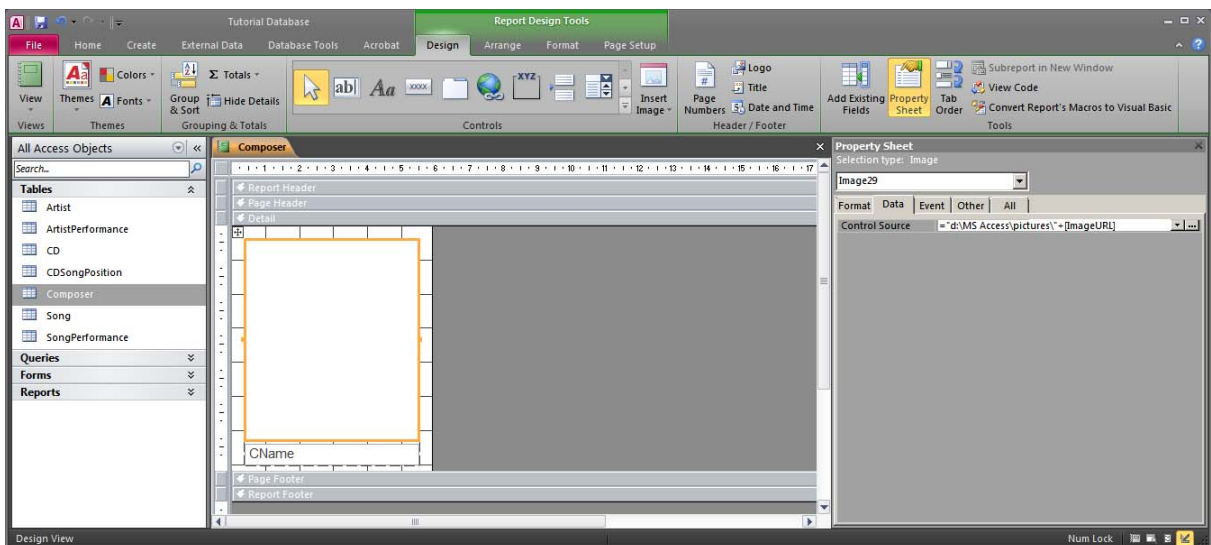


If we now create a form or report and want to show the images, we will need to have a component that can load the image from the file system, given a filename. The Image control can do exactly that. We can create a form that looks like this (in Print Preview):





In Design View we have a report that has the table Composer as Record Source and an Image control that has a Control Source that is a filename. We take the value of the column ImageURL and append it to the full path to the directory:



The Control source is the concatenation of "d:\MS Access\pictures" and the value of the column ImageURL. If the directory or file does not exist, Access will simply display nothing.

### 9.18 Tip 18 - Compacting And Repairing A Database

Databases built in Access are stored in one single file. This file can for different reasons become unnecessarily big or sometimes inconsistent. One reason why a file can grow in size is the use of multimedia. All the multimedia content that we add to the database will be stored inside the database file. Removing values and multimedia objects from the database may not automatically mean that the file becomes smaller. Working with different versions of Access

can also be a cause of inconsistencies in the database. If you have reasons to believe that there is something strange with your database, you can always use the Compact and Repair Database tool in Access (found under Database Tools on the ribbon).

### 9.19 Tip 19 - Linking External Data

In certain cases, it may be useful to use Access together with some other database manager. This could be the case when the entire database is implemented in another database manager and you want to use Access to build a user interface. You may also want to combine data from several databases build in different systems. This can easily be achieved with Access and ODBC (Open DataBase Connectivity).

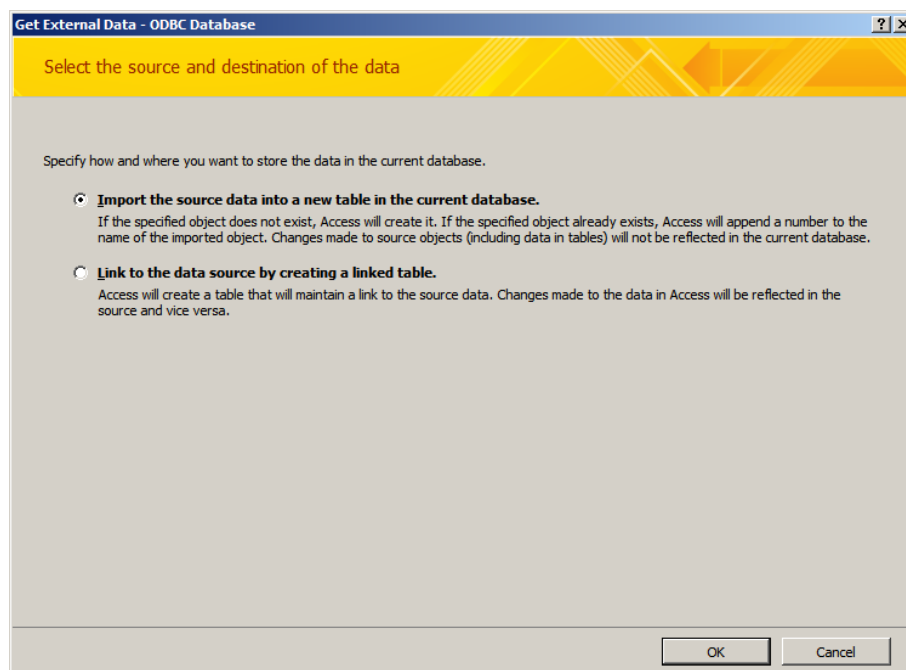
An ODBC database can be any database for which there is an ODBC driver. There are ODBC drivers for most database management systems. In this section we will work with a test database in MySQL. The process is the same for any database manager though. We will work with a database called testdb. This database has two tables: person and car, where a car is owned by a person. The database tables are created according to the following SQL statements:

```
CREATE TABLE Person (name VARCHAR(20) NOT NULL PRIMARY KEY, birthdate DATE, salary REAL)
```

```
CREATE TABLE Car (carID VARCHAR(10) NOT NULL PRIMARY KEY, color VARCHAR(12) NOT NULL, owner VARCHAR(20) NOT NULL, FOREIGN KEY (owner) REFERENCES Person(name))
```

We will assume that the tables have been created and populated.

In the "External Data" tab on the ribbon, press "ODBC Database". The "Get External Data" dialog will appear:

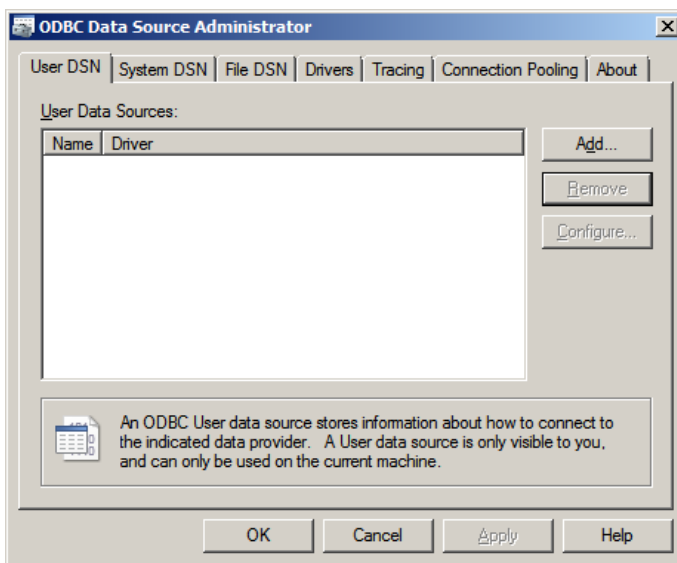


We can now choose whether to import data or link to some other database. We would like to link to the database described earlier through ODBC. This requires that the database has an ODBC alias. The next section describes how to create an ODBC alias.



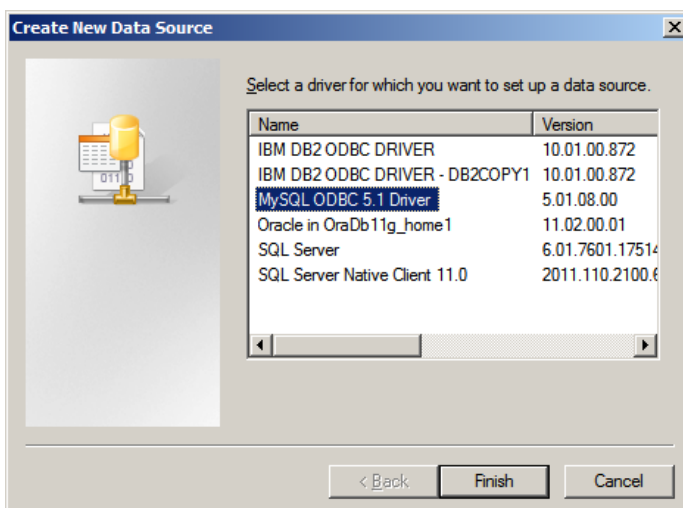
### 9.19.1 Creating An ODBC Alias

In order to link a database we must have an ODBC alias. This is basically a name we can use to refer to the original database without knowing what that database is called or what database manager it is implemented in. We can create an ODBC alias (also known as DSN – Data Source Name) either in advance, or while trying to link or import data in Access. To do this in advance, we must open the ODBC manager (ODBC Data Source Administrator). We can open the ODBC manager directly by executing the command "odbcad32.exe" or by locating the corresponding icon in the Windows Control Panel. When we open the ODBC manager it looks something like this:

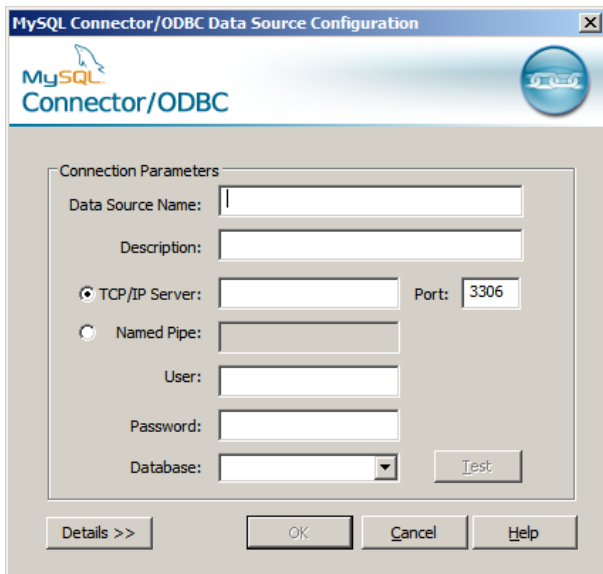


There are several tabs here, but the interesting ones are the first two. The System DSN tab contains any DSNs created to be available to all the users of the computer, while the User DSN contains DSNs that are available only to the current user. As long as you use the same Windows account there won't be any difference.

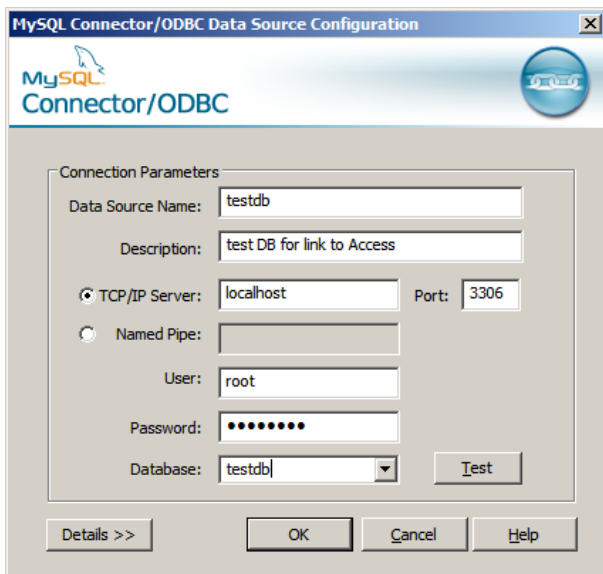
We can press Add... And a Wizard will appear to help us create a new DSN. We can start by selecting the appropriate driver:



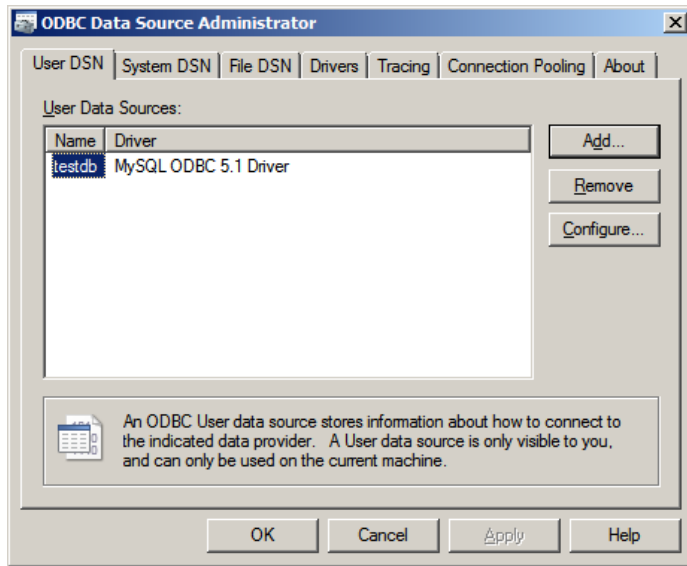
Press Finish, and the Wizard will initiate another wizard specific to the selected driver:



We select the correct database and give it a name, and also specify the server details:



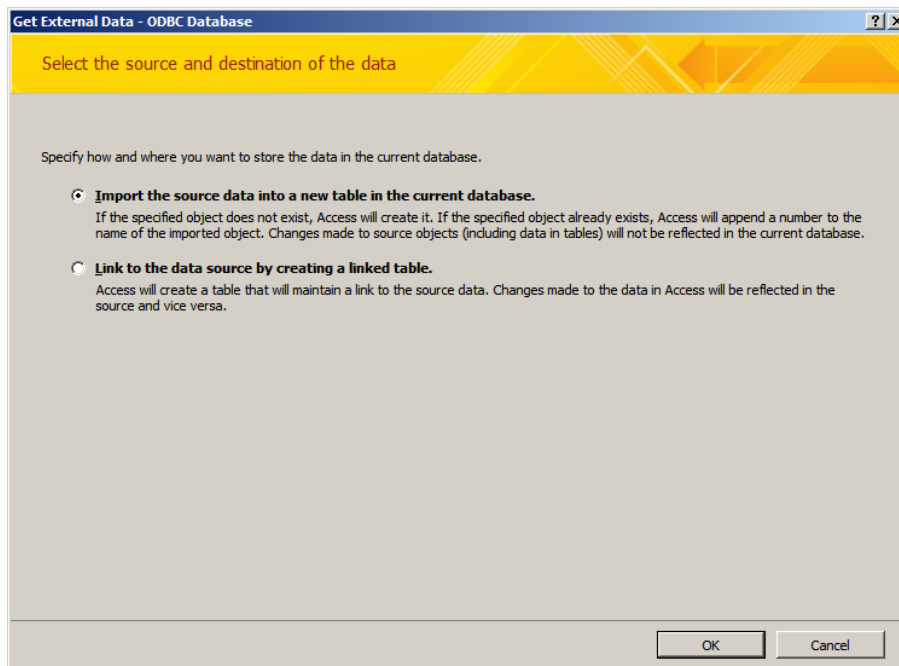
Press OK, and the new DSN is ready:



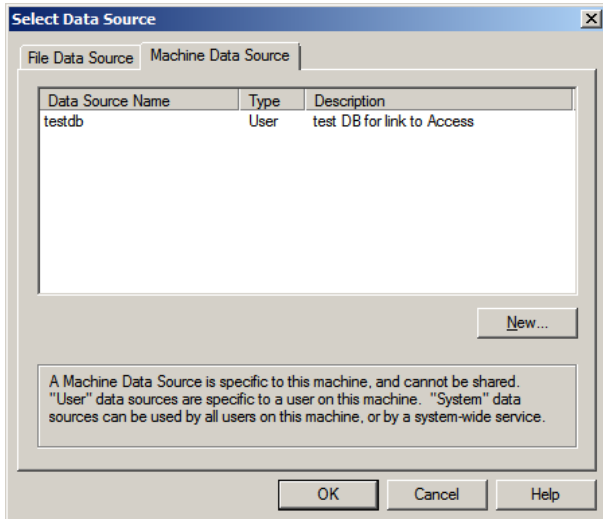
We can press the Configure... button to see the configuration and change it if necessary.

### 9.19.2 Linking To The MySQL Tables From Access

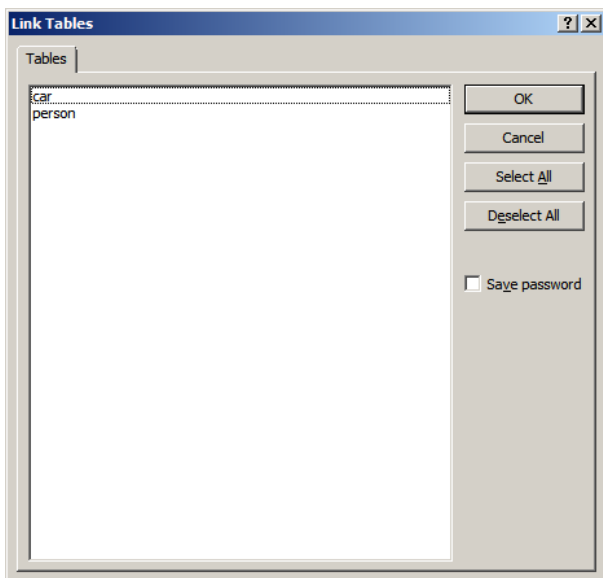
Now that the ODBC DSN is ready, we can link our two tables (from MySQL to Access). Back in Access, we have the "Get External Data" dialog:



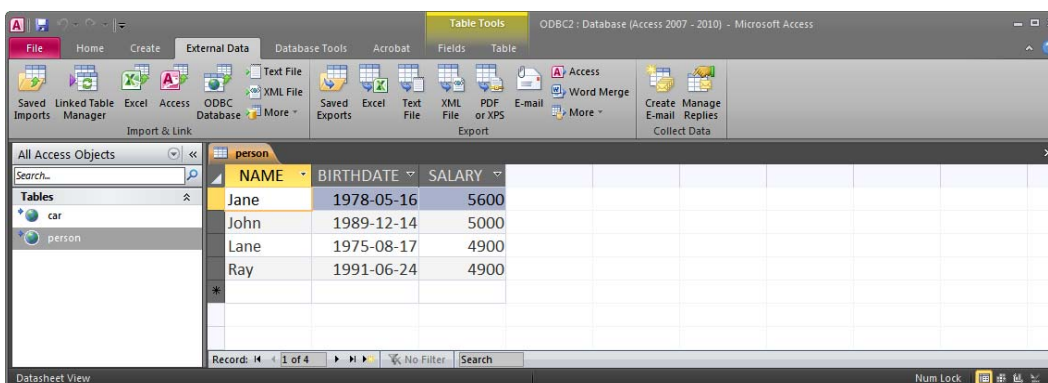
We select "Link to the data by creating a linked table", and press OK. Access will show a new dialog allowing us to select an existing ODBC DSN (or create a new one):



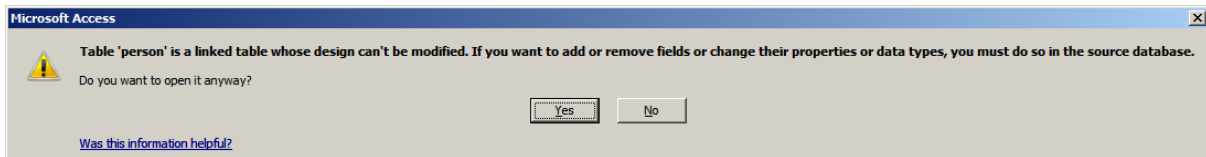
We can select the correct DSN and press OK. A new dialog appears showing all the available tables (and views):



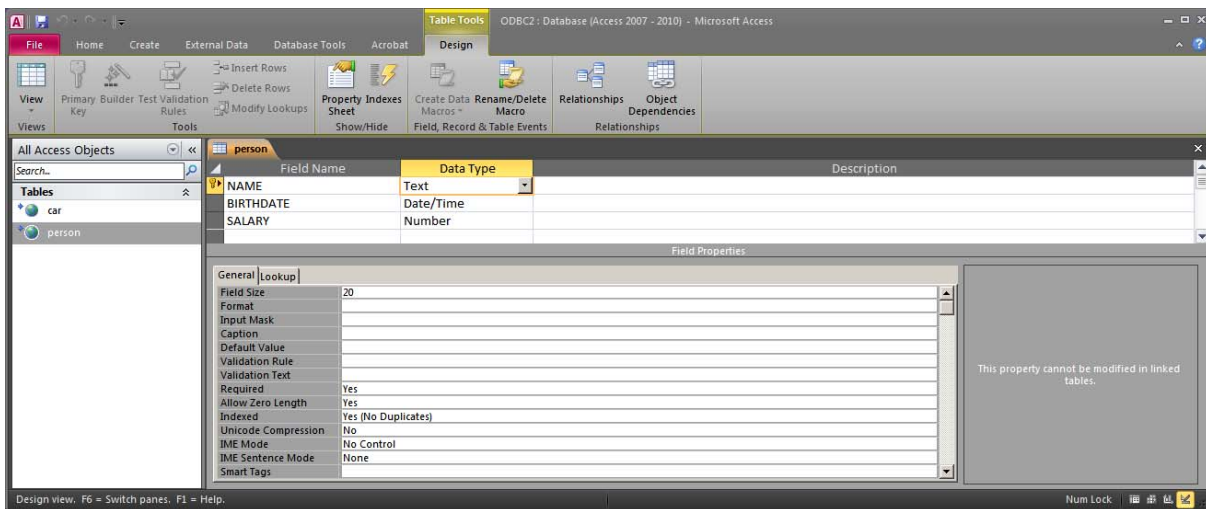
We can select the ones we want to link (both of them) and press OK. The linked tables are now visible in the Object browser (with a special icon indicating that they are linked tables). We can also open them and see their data:



If we try to design a linked table, we will not be allowed to change anything:



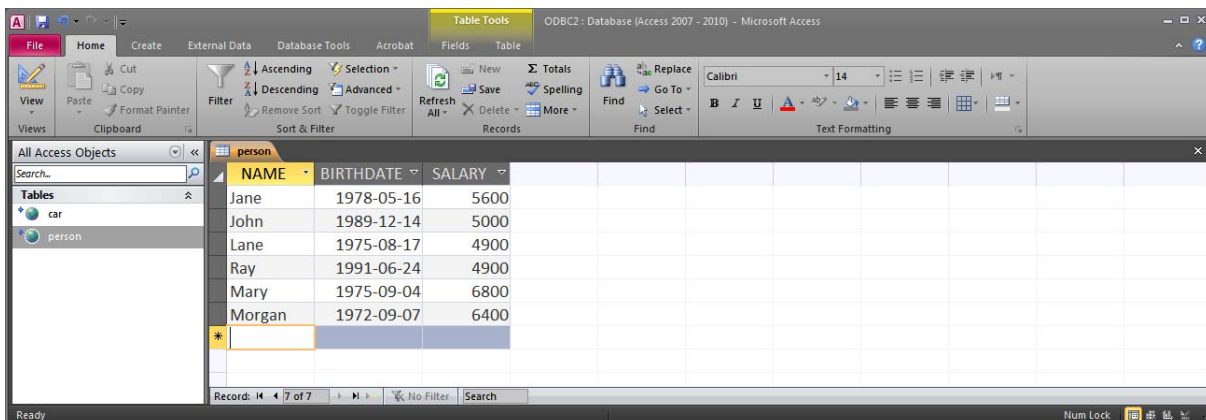
Press yes to see the table definition:



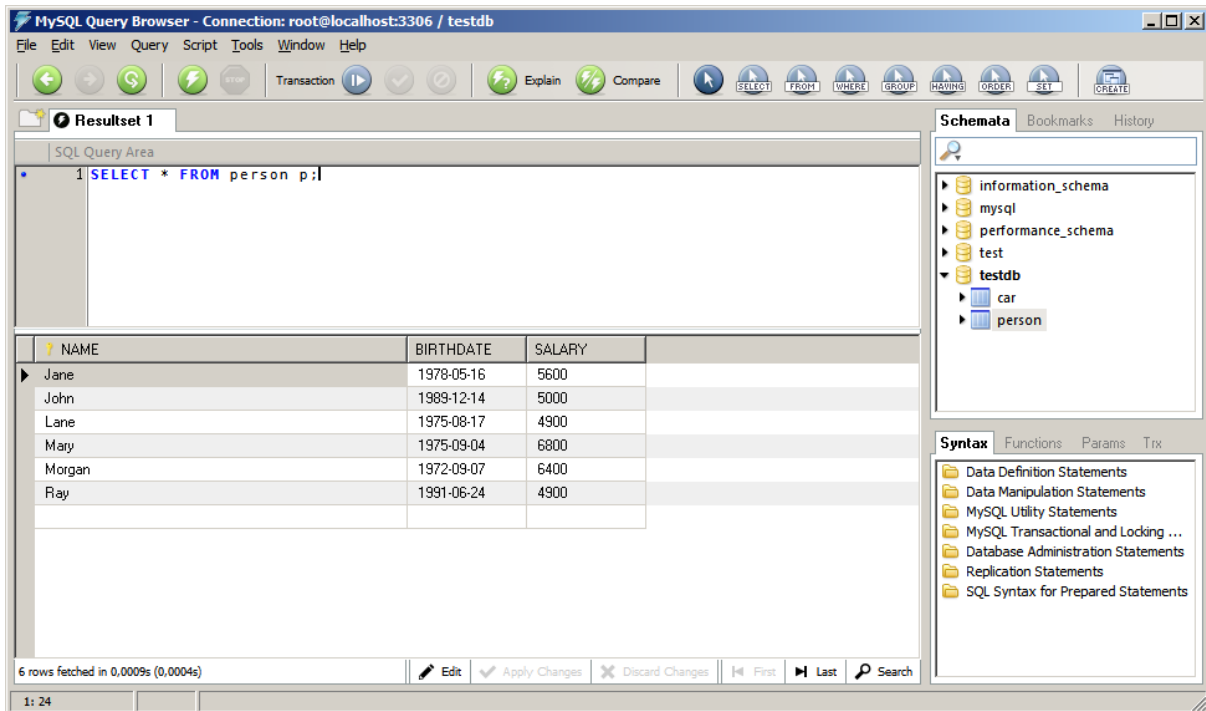
We can see that Access has retrieved all the details about the columns including data types, primary keys, NOT NULL restrictions, etc.

### 9.19.3 Working With Linked Tables

We can now open the table person, and modify some data (add two rows):



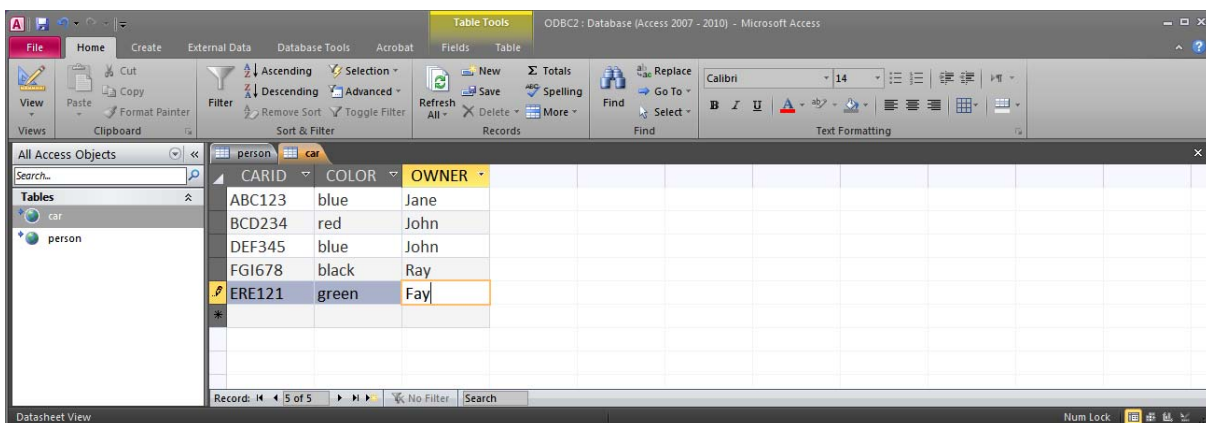
If we now ask MySQL to show the content of the table person we will see that the two new rows are there:



All the data is stored in the MySQL database. Access uses and modifies the data, but nothing is stored in Access.

When linking tables, foreign key rules are not imported to Access. But they are still maintained in the linked database. This means that if we try to add a car with an owner that does not exist, it will be MySQL that complains about it, and not Access.

We can try to insert the following car:



The moment we try to insert this new row, MySQL will send an error that Access kindly shows:



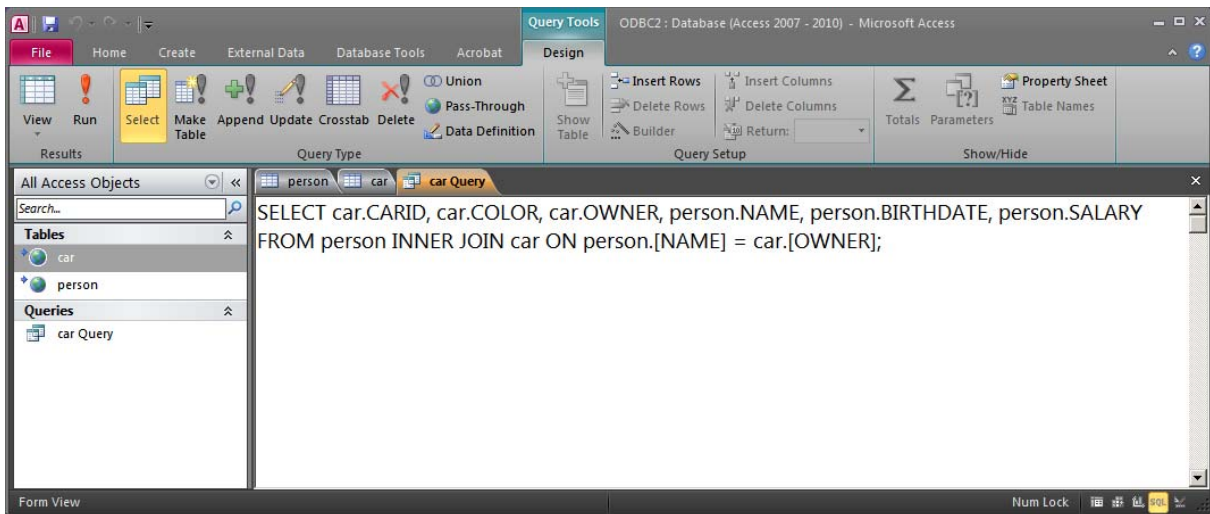


As you can see from the message, Access has no idea what the problem is. It just knows that there was an error.

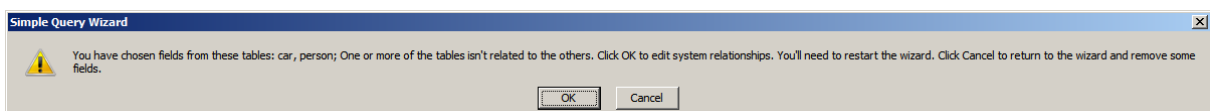
When working with linked tables, it can, however, be useful to also link the tables in Access. This will enable the wizards to identify relationships to be used when automatically building queries, forms and reports. This kind of linking will not have any effect on referential integrity. It is only useful for the Access wizards. We can add the linked tables in the relationships window and link them:



Try to create a query using the Simple Query Wizard with both tables. Access will automatically use the relationship in order to make a JOIN condition:



Try the same without the relationship and you will receive the following message:



So if we want to use the Access wizards efficiently, we will have to define all the relationships between the linked tables.

## 9.20 Tip 20 - Working With Dates And Times

Access, as well as all other database management systems, offers special data types for working with dates and times. In Access there is a data type called Date/Time, which can be configured for different formats of just the date, just the time or both date and time. In the database created in chapter 4, we used this data type to represent the date of a specific song performance.

Dates and times can be used for many operations. We can calculate the difference between dates or times in years, days, hours, etc. In order to do things like that, we must be able to convert the date or time value to the appropriate value. Access offers several functions that can be used for such purposes. Date and time representations are dependent on the regional settings of your Windows, and possibly the installation language of Access. You may therefore experience that your system does not behave exactly as described in this section. The configuration used while composing this section is based on an English version of Windows and Access with regional settings set to Swedish.

The function **DatePart** is a very useful function that makes it possible to retrieve only one part of a date or time:

```
DatePart("yyyy", "2003-2-15") returns the year: 2003
DatePart("m", "2003-2-15") returns the month: 2
DatePart("d", "2003-2-15") returns the day of the month: 15
DatePart("y", "2003-2-15") returns the day of the year: 46
DatePart("w", "2003-2-15") returns the day of the week: 7
DatePart("ww", "2003-2-15") returns the week of the year: 7
DatePart("h", "14:23:47") returns the hour: 14
DatePart("n", "14:23:47") returns the minute: 23
DatePart("s", "14:23:47") returns the second: 47
```

The function **DateDiff** can be used to retrieve the difference between two date/time values. You can choose to retrieve the difference in years, months, weeks, days, hours, etc.:

```
DateDiff("h", "14:23:47", "19:21:33") returns the difference in hours (ignoring the minutes): 5
DateDiff("n", "14:23:47", "19:21:33") returns the difference in minutes: 298
DateDiff("s", "14:23:47", "19:21:33") returns the difference in seconds: 17866
DateDiff("yyyy", "1999-2-13", "2003-10-4") returns the difference in years: 4
DateDiff("m", "1999-2-13", "2003-10-4") returns the difference in months: 56
DateDiff("d", "1999-2-13", "2003-10-4") returns the difference in days: 1694
DateDiff("w", "1999-2-13", "2003-10-4") returns the difference in weeks: 242
```

If the first date/time value is greater than the second, then the result will be negative:

```
DateDiff("n", "19:21:33", "14:23:47") : -298
```

The function **DateAdd** can be used to manipulate a date/time value. The function can be used to add (or subtract) years, months, hours, etc.:

```
DateAdd("yyyy", 5, "2003-10-4") adds 5 years to the specified date. Returns 2008-10-04
DateAdd("m", 5, "2003-10-4") adds 5 months to the specified date. Returns 2004-03-04
DateAdd("d", 5, "2003-10-4") adds 5 days to the specified date. Returns 2003-10-09
DateAdd("h", 5, "11:00") adds 5 hours to the specified time. Returns 16:00:00
DateAdd("h", -5, "11:00") adds -5 hours to the specified time. Returns 06:00:00
```

It is of course possible to combine several functions:

DateAdd("h", 5, DateAdd("n", 20, DateAdd("d", 5, "2003-10-4")))

Adds 5 hours, 20 minutes and 5 days to the specified date. Returns 2003-10-09 05:20:00

The functions **Year(value)**, **Month(value)**, **Day(value)**, **Hour(value)**, **Minute(value)**, **Second(value)** and **Weekday(value)**, return the year, month, day, hour, minute, second and weekday of the specified parameter. The parameter must of course be a valid date/time value.

The function **Now()** returns the current timestamp (time and date), the function **Date()** returns the current date and the function **Time()** returns the current time. Note: use Date() or Time(), if you only need the date or time respectively, but not both. Using Now() might create unexpected results in some cases when doing comparisons.

The function **DateSerial** can be used to create a new date. This function requires that you specify the year, month and day:

DateSerial(2004, 10, 12) creates and returns the following date: 2004-10-12

There is an equivalent function **TimeSerial** for time values.

There is also a possibility to create date/time values from string representations of dates and times. This is in most cases not necessary though, since Access accepts the string representation directly. However, it can still be useful to mention this possibility. For this, there are two functions **DateValue** and **TimeValue**:

TimeValue("4:35:17 PM") returns a new time value: 16:35:17

TimeValue("14:12") returns a new time value: 14:12:00

DateValue("2004-12-13") returns a new date value: 2004-12-13

DateValue("12/13/2004") returns a new date value: 2004-12-13

The two can be combined:

DateValue("12/13/2004") + TimeValue("4:35:17 PM") returns 2004-12-13 16:35:17

The expression above would be equal to this one:

TimeValue("12/13/2004 4:35:17 PM") + DateValue("12/13/2004 4:35:17 PM")

If you just try to use DateValue("12/13/2004 4:35:17 PM"), then the time part will be ignored. The result would only contain the date part: 2004-12-13

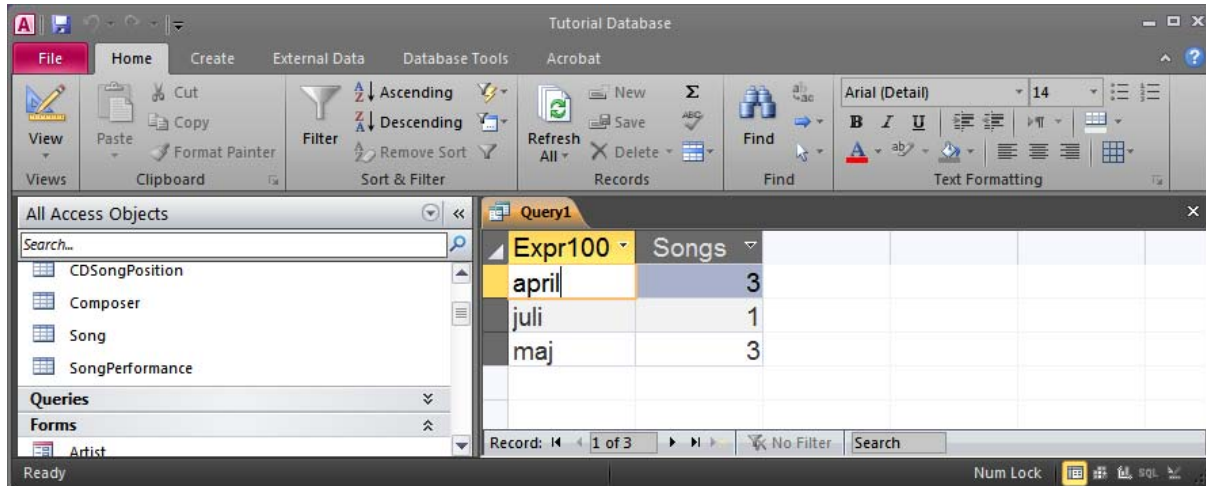
The functions **MonthName** and **WeekdayName** can be used to retrieve the name of the specified month and weekday as a string:

MonthName(Month("2003-12-14")) returns: December, december, diciembre etc. based on your regional settings.

WeekdayName(Weekday("2003-12-14")) returns: Monday, måndag, lunes etc. based on your regional settings.

All of the functions described in this section can of course be combined with each other and with other functions and SQL operators. You can for example write a query to retrieve the number of songs performed per month during 2002:

```
SELECT MonthName(Month(date)), COUNT(*) AS Songs
FROM songperformance
WHERE Year(Date) = 2002
GROUP BY MonthName(Month(date))
```



Another more direct way to use dates in SQL in Access is to use square brackets around the string representation of the date as follows. Here is the same query as above, but with the WHERE condition defined in an alternative way:

```
SELECT MonthName(Month(date)), COUNT(*) AS Songs
FROM songperformance
WHERE Date BETWEEN #2002-01-01# AND #2002-12-31#
GROUP BY MonthName(Month(date));
```

## 10 Other Resources

In the sections that follow, there are some references to interesting web sites and books about Access. Not all of them are for Access 2010, but most of the information applies to most versions of Access.

### 10.1 Web Sites

There are many good websites with information about Access. Just use a search engine to find a page relative to a specific Access issue.

The MSDN site is one of the best resources on the Internet for developers using any Microsoft product or technology. The main site can be reached at <http://msdn.microsoft.com/> from where one can search for Access. This is the official site for Access.

### 10.2 Books

There are hundreds of books about the different Microsoft Access versions out there and you will probably do fine with most of them. The different versions don't have so many differences (not in the basic functionality anyway), so a book about Access 2007 will do fine when working with any version from Access 2007 to Access 2010. Access 97 - Access 2003 have a different user interface, but most functionality remains the same.

It is not at all necessary to have a book. The resources in the Access help and on the Internet should be more than enough.

## 11 Epilogue

This covers the most commonly used functionality of Access. I would like to encourage you all to play around with Access. This is the best way to learn all the tricks. It is also important that you are not afraid to try to combine techniques covered in different chapters. You can for example combine something that we discussed in the chapter about forms with something that was first introduced when we were working with reports and also combine that with a macro.

I hope you have enjoyed this tutorial. Please give me feedback!

The Author

*nikos dimitrakas*