

One of the most important reasons that database systems were developed in the late 1960's was to provide a computerized system that would allow for implementing relationships between tables in the database. Prior to the creation of database systems, computer file systems could store data and retrieve the data from single files. To illustrate this concept, consider the Deuce Hardware Co. business that supplies hardware items (nails, screws, hammers and so on) to contractors. A contractor puts in an order for items to one of the Deuce stores, and an employee in the store is assigned to fill the order from the stores inventory. We'll keep it simple and assume that there is always a sufficient supply of all items in the store's inventory.

The old computer system for Deuce Hardware Co. used COBOL programs that accessed data in different files, for example, a file named STORES for data on each store, and a file named EMPLOYEES to store data on each employee.

The Deuce Hardware Co. now wants to develop an Access database to keep track of all stores' orders. They set up a table named STORES for all of their stores' data, and they create a table EMPLOYEES of employees in all stores. We start with just these two tables and will add other tables later.

The STORES table has these fields:

StoreNo	City	Phone
---------	------	-------

And the EMPLOYEES table has these fields:

EmpNo	EmpName	Payrate
-------	---------	---------

Deuce could retrieve data from either of these tables to answer questions, called queries, such as these:

Query 1: What city is store S3 in?

Query 2: What stores are in Charlotte?

Query 3: What is the payrate for employee named Ann Thomas?

Query 4: What employees have payrates of \$10.00 or more?

Each of the above queries is answered by retrieving data from only one of the two tables, just as the old COBOL programs retrieved data from the separate files. For example, Query 1 – What city is store S3 in? is performed by retrieving the row for store S3 from the STORES table and getting the city value in that row. Similarly, Query 2 retrieves rows in STORES that have the city value 'Charlotte', and Query 3 retrieves the row in EMPLOYEES for employee 'Ann Thomas', and Query 4 retrieves the rows in EMPLOYEES with payrates greater than or equal to \$10.00.

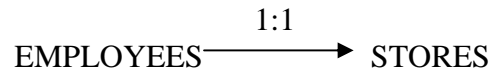
Now consider that each employee is employed in just one store, and each store employs some number of employees. There are two relationships illustrated here, the relationship between the employee and the store the employee is employed at, and the relationship of the store and the employees in that store. These relationships are conventionally diagrammed as

EMPLOYEES → STORES

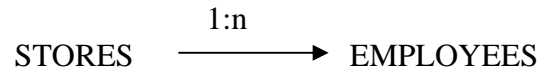
and

STORES → EMPLOYEES

In addition, the “EMPLOYEES → STORES” relationship has each employee related to just one store and so it is referred to as a one-to-one relationship, denoted 1:1, while the “STORES → EMPLOYEES” relationship allows each store to employ one or more employees and so this is called a one-to-many relationship, denoted as 1:n (read as ‘one to n’). These relationships now can now be diagrammed as



and



In a 1:n relationship such as STORES → EMPLOYEES, we will call the first file, STORES, the 1:-file (one-to-file) and the second file, EMPLOYEES, the :n-file (to-n-file).

While one-to-one relationships occur, it is the one-to-many relationships that are the important interest of database systems.

Here are some queries to show how a 1:n relationships is used to retrieve data from the database:

Query 5: Who are the employees in stores in Charlotte?

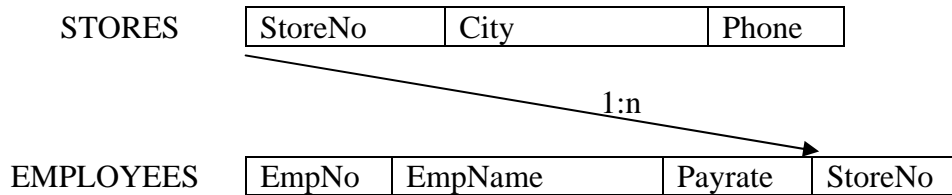
Query 6: What is the phone number of the store where employee Dawn Evans works?

In Query 5, the data about the store’s city is in the STORES table, while the data about the employees’ names is in the EMPLOYEES table. Consequently, to solve this query requires knowing which employees are employed in stores that have city value Charlotte. This is a two-file query. Likewise, for Query 6, the phone number of a store in the STORES file while the name of an employee is in the EMPLOYEES file, so this is also a two-file query.

The Access database management system (dbms) provides for relating two tables and using this relationship in queries. Access is referred to as a “relational” dbms, named for the mathematical theory of “relations” (and functions) that was the basis for this approach to dbms’s. The method used in a relational dbms to establish a relationship between the rows of 1: table (one-to table) such as STORES to the to-many or :n table (to-n table) such as EMPLOYEES is to include an additional field in the :n table that is same as the primary key field of the one-to table. This additional field in the :n table is called a **foreign key** field since it’s the primary key of another table. For the STORES → EMPLOYEES 1:n relationship, this means that the EMPLOYEES table must have a StoreNo field included in it, so now the EMPLOYEES table would have this structure (these fields)

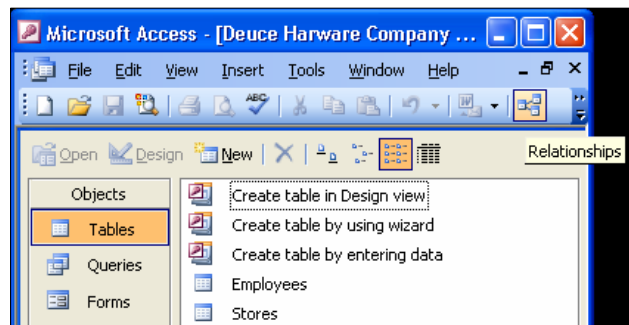
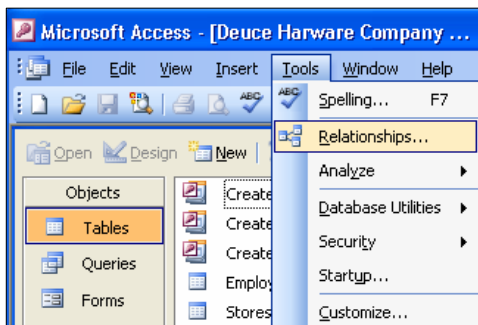
	Primary Key			Foreign Key
EMPLOYEES	EmpNo	EmpName	Payrate	StoreNo

and the 1:n relationship is diagrammed as

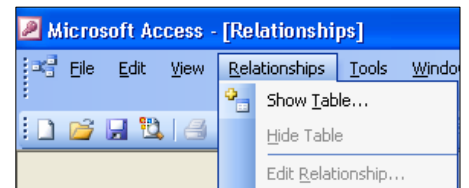


Project Step 1: Create a Deuce Hardware Co. database in Access, and create two tables in this database, a STORES table and an EMPLOYEES table, where each table has the fields in the diagram above. Make the StoreNo field and the EmployeeNo field the primary keys of the STORES and EMPLOYEES table.

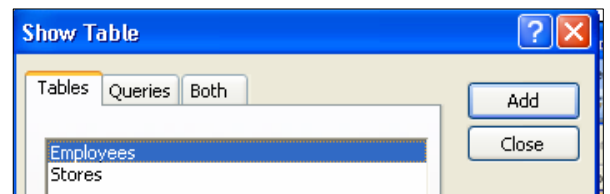
Now create the relationship between the STORES table to the EMPLOYEES table by using the Relationships window. You can get the Relationships window displayed by clicking Tools | Relationships, that is, click Tools on the menu bar, and then click Relationships on the Tools menu. Alternatively, click the Relationships button on the tools bar below the main menu bar. These two diagrams illustrate these ways of displaying the Relationships window.



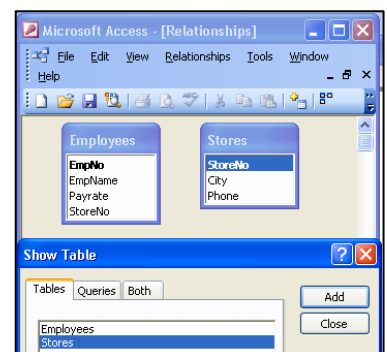
The Relationships window appears. To create the relationship between two tables in Access, you must add the tables to the Relationships window by clicking on the Relationships item in the main menu bar of the Relationships window, as illustrated here to the right.



Then click the Show Table item to get the Show Table window. Your Employees and Stores tables are listed in this Show Tables window since these are the only two tables in your database.



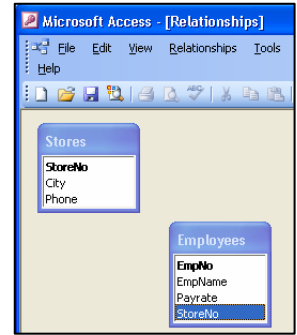
Now click on the Employees table in the Show table window, and click the Add button. The Employees table is then added to the Relationships window. Likewise, add the Stores table to the Relationships window, so your Relationships window looks like this:



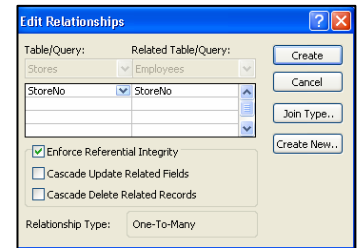
Click the close button of the Show Tables window.

The Relationships window has the Employees tables on the left and the Stores window on the right. Since a 1:n relationship between two tables is conventionally displayed with the 1:-table displayed to the left and above (when possible) and the :n-table is displayed below and to the right (when possible), then let's move these tables around in the Relationships window.

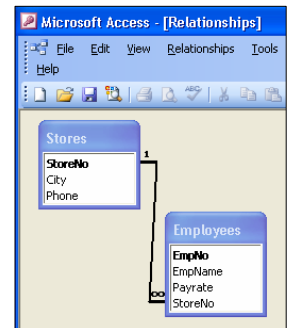
Click on the title bar of the Employees table and drag it to the right and lower and then click on the title bar of the Stores table and drag it to the left and higher. Your Relationships window might now look like this:



Now you create the actual 1:n relationship from Stores to Employees by clicking on the StoreNo field in the Stores table and dragging it to the StoreNo field in the Employees table. While you are dragging it, the cursor changes shape and finally when you get it over the StoreNo field in the Employees table, it looks like a little rectangle with some unreadable squiggles in it. When you release the mouse button after dragging, a new window, the Edit Relationships window appears. In the Edit Relationships window, the Stores table and the StoreNo field are listed under the Table/Query column and the Employees table is listed under the Related Table/Query column. Below these two columns, there is a check box for the Enforce Referential Integrity requirement. Check this box as illustrated here:



Finally, click the Create button in the Edit Relationships window. The result is that the Relationships window is redisplayed and now there is a line connecting the StoreNo field in the Stores table to the StoreNo field of the Employees table. Your Relationships window should look like this:



Notice the “1” on the line next to the StoreNo field of the Stores table, and the infinity sign next to the StoreNo field of the Employees box in the Edit Relationship window. They indicate that the database will enforce a 1:n relationship from the Stores table to the Employees table. This has two important consequences regarding the data you try to put into the database:

- If you try to enter an employee data record in the Employees table, but that record's StoreNo value is not the same as the StoreNo value of a record in the Stores table, then Access will not allow the data to be stored. Below, you will see is how this error looks when you try this illegal operation.
- If you try to delete a record from the Stores table, but there are Employees records (one or more) still related to this store record, then Access will not allow this store record to be deleted.

To see how these referential errors occur and how Access handles them, first create the following records in the stores table (go to the Access main screen, and open the Stores table in the Datasheet view and enter the data into the table):

	StoreNo	City	Phone
+	S1	Rock Hill	803-329-1234
+	S2	Charlotte	704-543-9876
+	S3	Charlotte	704-456-2222
+	S4	Fort Mill	803-396-4400

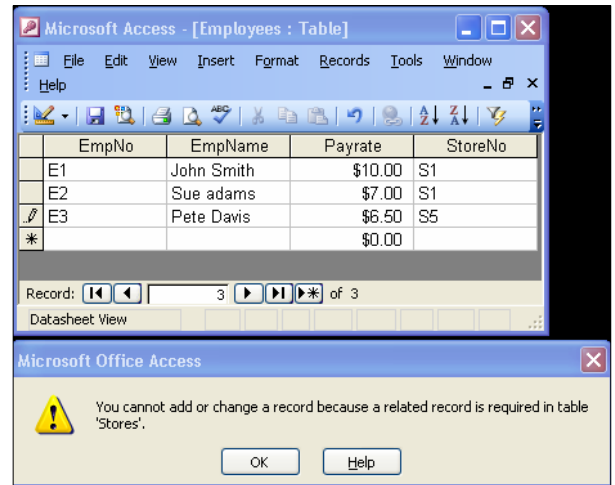
Next open the datasheet view of the Employees table, and enter these “valid” employee records:

EmpNo	EmpName	Payrate	StoreNo
E1	John Smith	\$10.00	S1
E2	Sue adams	\$7.00	S1

Now let’s enter an invalid employee record. Enter E3, Pete Davis, payrate of 6.50 and a StoreNo of S5. The result, after you press Enter to try to store this record is shown here:

You get the message that you “cannot add or change a record because a related record is required in the table ‘Stores’.” This is the first violation of referential integrity listed above.

Try creating another violation of referential integrity by trying to delete the S1 record in the Stores table, that is, open the Stores table and click on the S1 record and press the Del key, or click the Edit main menu item and click Delete. You will get an error message.



Fill in the data in your Deuce Hardware Co. database with these values.

STORES Table

Stores		
StoreNo	City	Phone
S1	Rock Hill	803-329-1234
S2	Charlotte	704-543-9876
S3	Charlotte	704-456-2222
S4	Fort Mill	803-396-4400

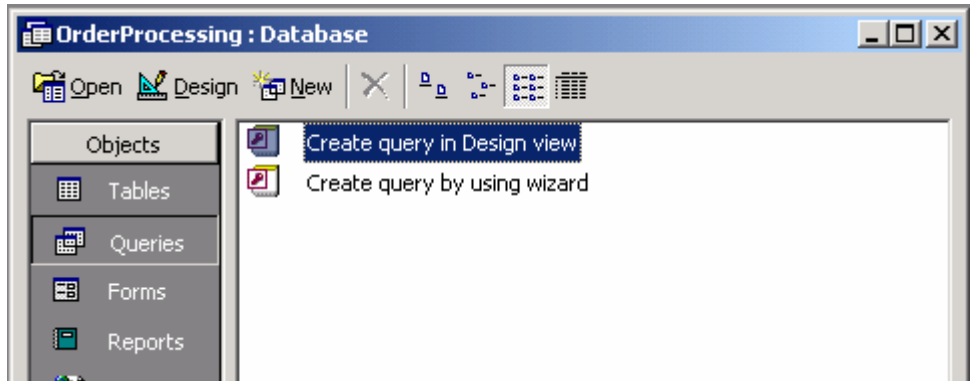
EMPLOYEES Table

Employees			
EmpNo	EmpName	Payrate	StoreNo
E1	John Smith	\$10.00	S1
E2	Sue Adams	\$7.00	S1
E3	Pete Davis	\$6.50	S2
E4	Joe Rich	\$11.00	S2
E5	Dawn Evans	\$9.50	S1
E6	Dale Baker	\$8.00	S3
E7	Kim Jones	\$12.00	S3

Queries in Access Using a 1:n Relationship

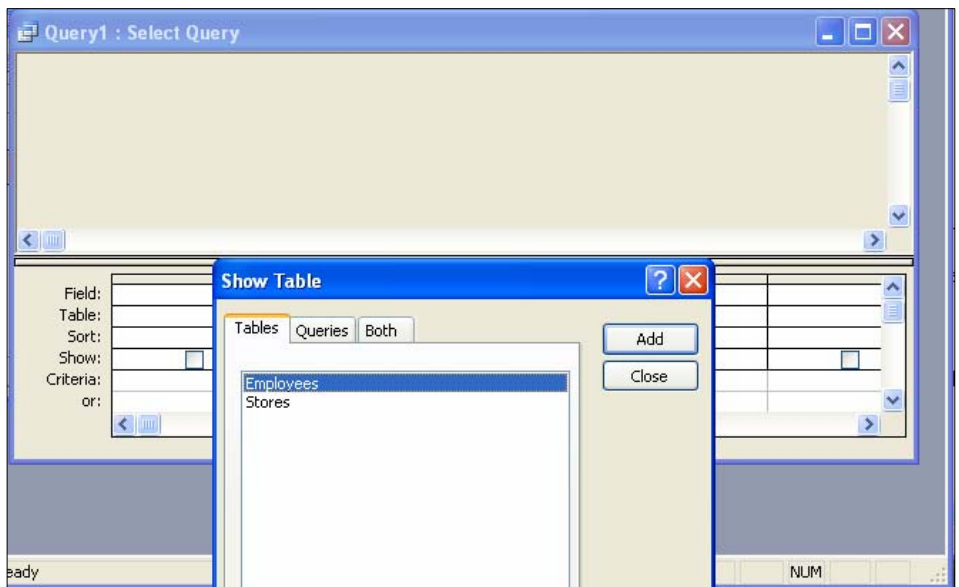
You have previously learned how to create a query on a single table, then execute the query and view or print the result of the query. Now we will cover how to create, execute and view a two-table query.

Click on the "Queries" button in the Objects pane at the left side of the database window.



Click on the "Create query in Design view" to get the Query design window.

The Show Table window is automatically displayed.



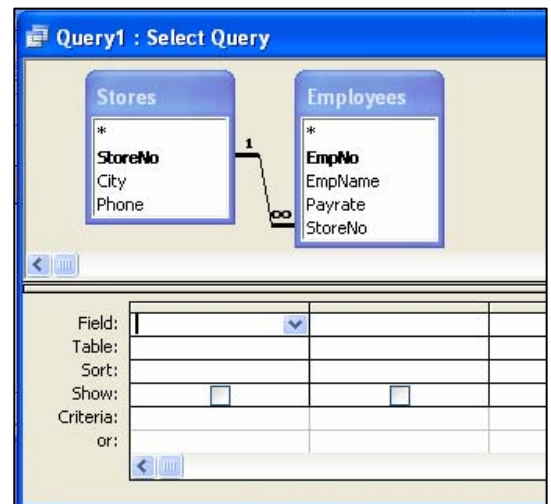
Click on the table name of a table that you want included in the query, and then click the Add button.

Do this for each table to be included in the query, then


First click on the Stores table and Add it, then click on the Employees table and add it. Close the Show Table window.

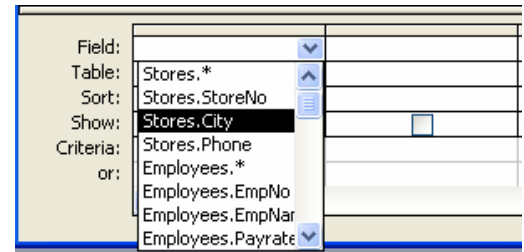
You can display the Show Table window at any time by right clicking on the title bar of the design window, or by clicking the View, Show Table on the main menu bar.

When two tables with a 1:n relationship between them are included in the query window, the relationship is automatically diagrammed.

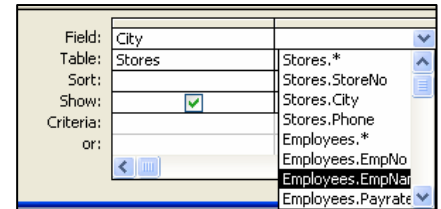


To create a Select query, you must select fields to add to the design grid in the query window. There are two ways to select fields, either by using the drop down menu in the "Field:" box of the design grid, or dragging them from a table that is displayed above the grid.

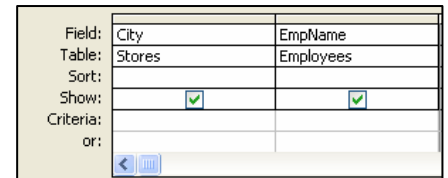
To select fields in the "Fields:" box, click on the arrow  in the box to the right of "Field:" The list of the fields in all of the tables that you added to the query design window is displayed in a drop down menu, as illustrated here to the right. Click on any field in this menu to select it. That field is now displayed in the box and the table name for the table containing that field is displayed in the "Table:" box, as illustrated below.




For the Query 5 example, “Who are the employees in stores in Charlotte?”, select the Stores.City item for the first field. Then move over to the second field, and select the Employees.EmpName item

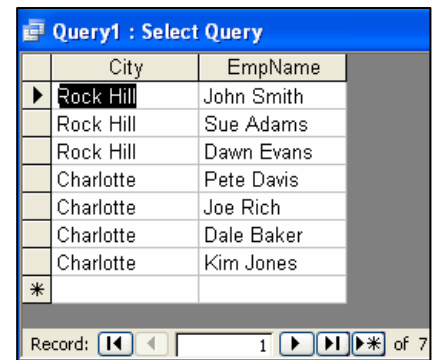


Now your query design should have the City (from the Stores table) and the EmpName (from the Employees table) in the first two fields of the query:



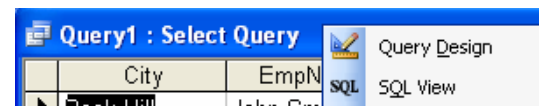
You now execute your query by clicking Query | Run from the menus or click the run button  on the tools bar.

The query window displays the result of the query:

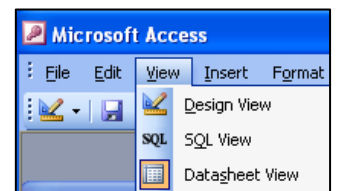


Note in this result, the employees for both Rock Hill and for Charlotte are displayed. To get only the employees for Charlotte, we must restrict the City value in the query to have a value “Charlotte”.

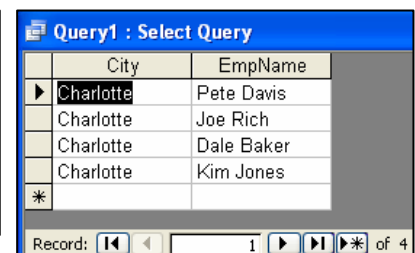
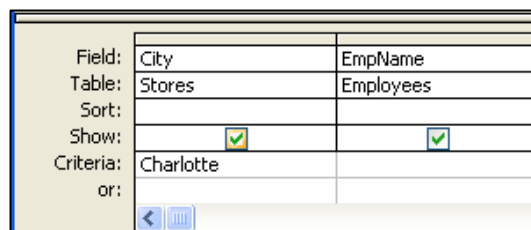
Switch to the Query Design view of the query from this Datasheet view by RIGHT clicking on the title bar of this query window and then click on the Query Design item of the drop down menu.



Alternatively, click View | Design View on the main menu bar:



Now enter the value “Charlotte” in the Criteria row under the City field and run the query again to display only employees in Charlotte stores.



You may print and/or save your query in this database.

Another method for choosing fields for the query is to select one a field from the field list in a table displayed in the query window, and then drag that field to a column in the design grid.

This table below from the Microsoft Access Help window below tells you the different ways of doing this.

To select	Do this
A field	Click the field name.
A block of fields	Click the first field in the block, hold down SHIFT, and click the last field.
Noncontiguous fields	Hold down CTRL as you click the fields.
All fields	Double-click the title bar of the field list or click the asterisk (*).

Notes

- Once you've selected fields in the field list, you must drag them to the design grid.
- When you drag more than one field at a time, Microsoft Access places each field in a separate column. If you drag the asterisk to the grid, Microsoft Access places the table or query name in one column and attaches a period and asterisk to the name (for example, Categories.*). After either operation, the datasheet looks the same.
- Instead of dragging, you can also add fields by double-clicking the name in the field list or selecting a field directly from the list box in the **Field** row on the grid.

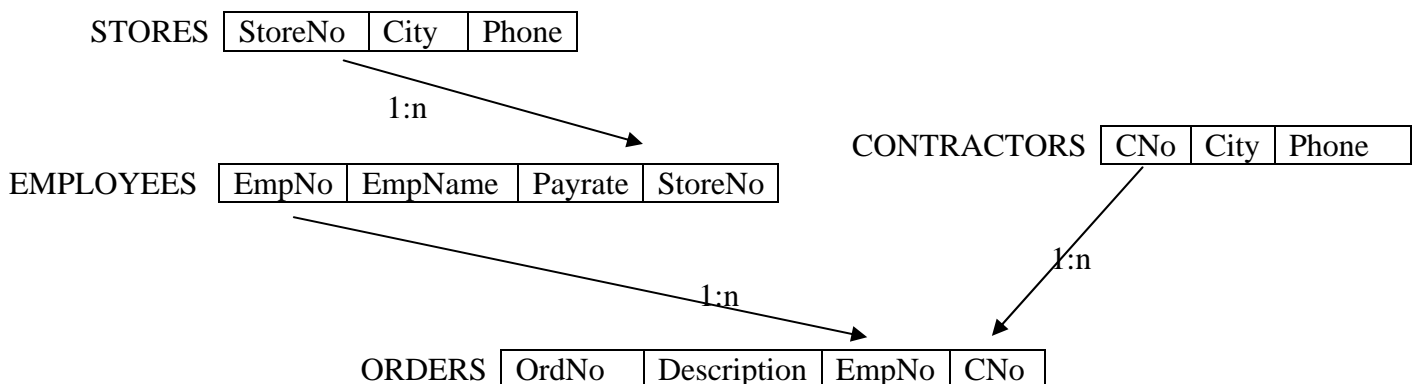
Exercise 1: Create and execute these queries:

Query 6: What is the phone number of the store where employee Ann Thomas works?

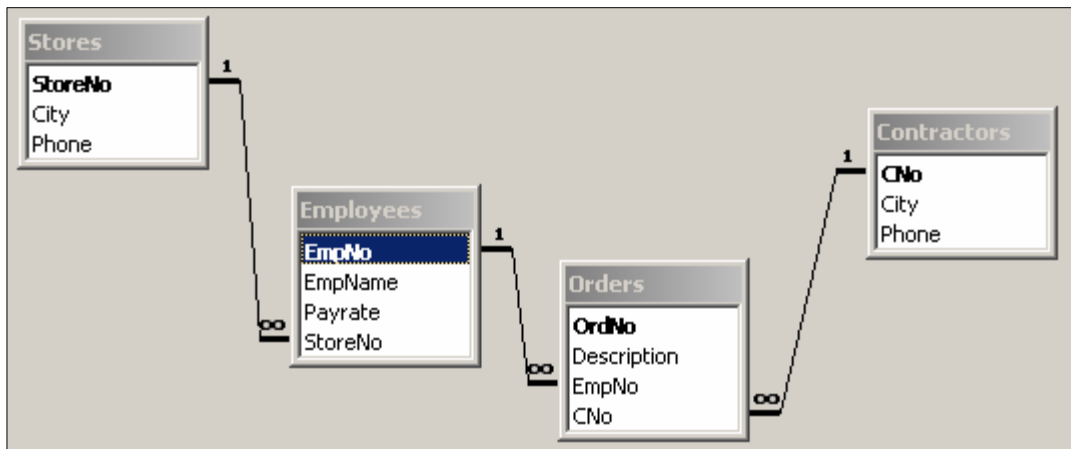
Query 7: List the employee names and their payrates for stores in Rock Hill.

Query 8: List the store number and city which have employees with payrates greater than \$9.99, and include the employee's names and payrate.

Exercise 2: Create two more tables in your Deuce Hardware database, one for CONTRACTORS and one for ORDERS, and create these two 1:n relationships: EMPLOYEES → ORDERS and CONTRACTORS → ORDERS, and enforcing referential integrity.



The Relationships Window would look something like this:



Now, enter these data into those tables.

Contractors

	CNo	City	Phone
	C1	Rock Hill	803-325-5555
	C2	Fort Mill	803-691-4599
	C3	Charlotte	704-225-1597
	C4	Charlotte	704-228-1234
	*		

Orders

	OrdNo	Description	EmpNo	CNo
	N1	Hammer 16 oz	E1	C2
	N2	Bucket 2 gal	E3	C1
	N3	Nails 8p	E2	C3
	N4	Screen wire	E5	C4
	N5	Power Nailer	E3	C3
	N6	Paint 1 gal	E4	C4
	*			

Finally, create and run these queries.

Query 9: List all of the orders for store number S1.

Query 10: List all of the orders for stores in Rock Hill.

Query 11: List all contractors with orders for store S1.

Query 12: List the descriptions of all orders for the employee named Pete Davis.