

# 7

## Joins

Up to this point we have used SQL SELECT statements to retrieve information on single tables. However, as we learned in earlier chapters, relational databases contain information among many related tables. This chapter will introduce you to JOIN operations that allow queries to retrieve rows from multiple tables.

### CHAPTER OBJECTIVES

The objectives for this chapter are as follow:

- Understand how to use INNER JOINS.
- Understand how to use OUTER JOINS.
- Understand how to use SELF JOINS.
- Understand how to perform multiple table joins.
- Understand how to avoid Cartesian products.

## Referential Integrity

Figure 1 shows a relationship between the Employees table and EmployeesDependents table. The relationship is a one-to-many relationship where each employee can have many dependents, but each dependent can belong to only one employee. The relationship is linked by connecting the primary key from the Employees table (EmployeeID) to the foreign key of the EmployeesDependents table, also called EmployeeID. These are referred to as common columns and they are linked to form referential integrity constraints.

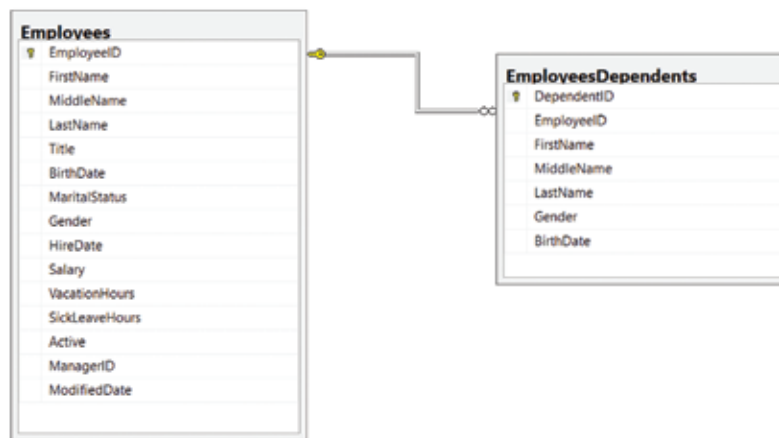


Figure 1

Referential integrity is necessary to ensure the integrity of the database. For example, any **EmployeeID** that exists as a foreign key in the **EmployeesDependents** table **MUST** exist as a primary key in the **Employees** table. Otherwise that would produce what would be considered “orphan” rows. Orphan rows are rows containing foreign keys whose primary key has been deleted.

## INNER JOIN

The INNER JOIN is used to retrieve all data from tables that have matching keys. In our example of Employees and their dependents, an INNER JOIN will only retrieve rows from the Employees table if it finds an EmployeeID match in the EmployeesDependents table. To understand this let's first look at all the rows in the EmployeesDependents table in Figure 2.

	DependentID	EmployeeID	FirstName	MiddleName	LastName	Gender	BirthDate
1	1	9	Martha	S	Eaton	F	2010-06-07
2	2	9	William	M	Eaton	M	2008-11-02
3	3	34	Michael	T	Erickson	M	2005-07-01
4	4	72	Robert	NULL	Ralls	M	1999-04-25
5	5	72	Marcia	A	Ralls	F	1998-09-17
6	6	72	Barbara	C	Ralls	F	2001-11-12
7	7	72	Theodore	NULL	Ralls	M	2005-08-01
8	8	72	Patricia	M	Ralls	F	2003-12-22
9	9	96	Samuel	NULL	Ajenstat	M	1995-04-01
10	10	96	Heidi	NULL	Ajenstat	F	1997-10-07
11	11	96	Wendy	NULL	Ajenstat	F	1999-08-27
12	12	102	Joseph	L	Bischoff	M	1993-08-07
13	13	94	Christopher	A	Hillmann	M	1994-02-17
14	14	94	Jason	P	Hillmann	M	1995-07-21
15	15	71	Sheila	M	Valdez	F	1997-08-22
16	16	71	Carla	C	Valdez	F	1998-12-02
17	17	71	Maria	N	Valdez	F	2000-08-01

Figure 2

Figure 2 shows all the dependents in the EmployeesDependents table for a total of 17 rows. Notice that the EmployeesDependents table has its own primary key with column name DependentID. The foreign key is highlighted in blue and is connected to the primary key in the Employees table.

Let's now look at rows from the Employees table. However, since there is a total of 102 employees I'll just show a subset in Figure 3. I've highlighted the rows in the subset of employees that have dependents in the EmployeesDependents table. You can see that there are many more employees who don't have any dependents. When we perform an INNER JOIN these employees will NOT show up in the result set because they have no related data in the EmployeesDependents table.

EmployeeID	FirstName	MiddleName	LastName	Title	BirthDate	MaritalStatus	Gender	HireDate
2	Kevin	F	Brown	Marketing Assistant	1977-06-03	S	M	1997-02-26
3	Roberto	NULL	Tamburello	Engineering Manager	1964-12-13	M	M	1997-12-12
5	Thierry	B	D'Hofo	Tool Designer	1949-08-29	M	M	1998-01-11
6	David	M	Bradley	Marketing Manager	1965-04-19	S	M	1998-01-20
9	Gail	A	Erickson	Design Engineer	1942-10-29	M	F	1998-02-06
11	Josief	H	Goldberg	Design Engineer	1949-04-11	M	M	1998-02-24
12	Teri	Lee	Duffy	Vice President of Engineering	1961-09-01	S	F	1998-03-03
28	Ashvini	R	Sharma	Network Administrator	1967-04-28	S	M	1999-01-05
30	Paula	M	Baneiro de Mattos	Human Resources Manager	1966-03-14	M	F	1999-01-07
34	Susan	W	Eaton	Stocker	1968-03-20	S	F	1999-01-08
35	Vamsi	N	Kuppe	Shipping and Receiving Clerk	1967-04-19	M	M	1999-01-08
41	Peng	J	Wu	Quality Assurance Supervisor	1966-04-19	M	M	1999-01-10
42	Jean	E	O'Brien	Information Services Manager	1966-01-13	S	F	1999-01-12
44	A. Scott	NULL	Wright	Master Scheduler	1959-10-19	S	M	1999-01-13
46	Sanya	E	Hampadungasataya	Marketing Specialist	1977-06-21	S	M	1999-01-13
47	Wille	T	Johnson	Recruiter	1968-08-18	S	M	1999-01-14
49	Christian	E	Kleinerman	Maintenance Supervisor	1966-02-18	M	M	1999-01-15
54	Tengiz	N	Kharatishvili	Control Specialist	1980-05-29	S	M	1999-01-17
59	Deborah	E	Smyth	Accounts Receivable Speci...	1966-04-07	M	F	1999-01-19
66	Jenaina	Baneiro G...	Bueno	Application Specialist	1975-03-03	M	F	1999-01-24
70	Mindy	C	Martin	Benefits Specialist	1974-12-22	M	F	1999-01-26
71	Wendy	Beth	Kahn	Finance Manager	1974-11-12	S	F	1999-01-26
72	Kim	T	Ralls	Stocker	1974-06-01	S	F	1999-01-27
77	Sean	P	Alexander	Quality Assurance Technician	1966-04-07	S	M	1999-01-29
78	Phyllis	I	Murphy	Business and Development	1970-07-06	S	F	1999-01-29

Figure 3

The INNER JOIN statement to retrieve employees with dependents is as follows.

```
SELECT *
FROM Employees
INNER JOIN EmployeesDependents ON Employees.EmployeeID =
EmployeesDependents.EmployeeID
```

Notice that I had to identify the table name for each EmployeeID so that SQL Server understands which table I'm referring to. This is called "qualifying." If I didn't qualify the tables SQL Server would find the EmployeeID ambiguous, as it doesn't know which table I am referring to in the SELECT statement and the error shown in Figure 4 would be the result.

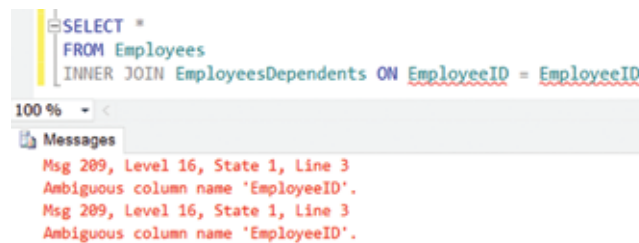


Figure 4

It is common procedure to use alias names in place of table names to minimize the amount of typing required as well as the length of the SELECT statement. In the example below I've used the letter "E" as an alias for the Employees table and the letter "D" as an alias for the EmployeesDependents table. You are free to use any alias and length you'd like but usually single letters are used. I selected "E" to signify employees and "D" to signify dependents.

```
SELECT E.EmployeeID,
       E.FirstName EmployeeFirstName,
       E.LastName EmployeeLastName,
       Title,
       DependentID,
       D.FirstName DependentFirstName,
       D.LastName DependentLastName,
       D.Birthdate
FROM Employees E
INNER JOIN EmployeesDependents D ON E.EmployeeID = D.EmployeeID
ORDER BY E.LastName
```

The results in Figure 5 show rows of all employees who have children. You will notice the result set shows some rows with duplicate employee names but that is just because those employees have more than one dependent. If you look to the right in Figure 5 you can see different dependent names. Employees that only have one dependent will show only once appropriately. And for those employees who don't have dependents, well, they don't show in the result set at all because they don't have related information in the EmployeesDependents table.

SQLQuery1.sql - (Laner-PC\Owner (S1))

```

SELECT E.EmployeeID,
       E.FirstName EmployeeFirstName,
       E.LastName EmployeeLastName,
       Title,
       DependentID,
       D.FirstName DependentFirstName,
       D.LastName DependentLastName,
       D.BirthDate
FROM Employees E
INNER JOIN EmployeesDependents D ON E.EmployeeID = D.EmployeeID
ORDER BY E.LastName

```

100 %

Results Messages

	EmployeeID	EmployeeFirstName	EmployeeLastName	Title	DependentID	DependentFirstName	DependentLastName	BirthDate
1	102	Dan	Bacon	Application Specialist	12	Joseph	Bechhoff	1993-08-07
2	34	Susan	Eaton	Stockler	3	Michael	Erickson	2005-07-01
3	9	Gail	Erickson	Design Engineer	1	Martha	Eaton	2010-06-07
4	9	Gail	Erickson	Design Engineer	2	William	Eaton	2008-11-02
5	71	Wendy	Kahn	Finance Manager	15	Sheila	Valdez	1997-08-22
6	71	Wendy	Kahn	Finance Manager	16	Carla	Valdez	1998-12-02
7	71	Wendy	Kahn	Finance Manager	17	Maria	Valdez	2000-08-01
8	72	Kim	Ralls	Stockler	4	Robert	Ralls	1999-04-25
9	72	Kim	Ralls	Stockler	5	Marcia	Ralls	1998-09-17
10	72	Kim	Ralls	Stockler	6	Barbara	Ralls	2001-11-12
11	72	Kim	Ralls	Stockler	7	Theodore	Ralls	2005-08-01
12	72	Kim	Ralls	Stockler	8	Patricia	Ralls	2003-12-22
13	94	Candy	Spoon	Accounts Receivable Specialist	13	Christopher	Hilman	1994-02-17
14	94	Candy	Spoon	Accounts Receivable Specialist	14	Jason	Hilman	1995-07-21
15	96	William	Vinn	Schucklin Assistant	9	Samuel	Auerstet	1995-04-01

Query executed successfully. (local) (10.0 RTM) Owner-PC\Owner (S1) HumanResou

Figure 5

In the SELECT statement you will notice that I used the alias names to qualify the table for all the columns except Title and DependentID. This is because Title occurs only in the Employees table and DependentID occurs only in the EmployeesDependents table so there is no ambiguity there. However, for the sake of consistency, you should qualify all your column names so that anybody who looks at your code can identify the table the data comes from. See the same example below with all columns qualified in Figure 6.

SQLQuery1.sql - (Laner-PC\Owner (S1))

```

SELECT E.EmployeeID,
       E.FirstName EmployeeFirstName,
       E.LastName EmployeeLastName,
       E.Title,
       D.DependentID,
       D.FirstName DependentFirstName,
       D.LastName DependentLastName,
       D.BirthDate
FROM Employees E
INNER JOIN EmployeesDependents D ON E.EmployeeID = D.EmployeeID
ORDER BY E.LastName

```

100 %

Results Messages

	EmployeeID	EmployeeFirstName	EmployeeLastName	Title	DependentID	DependentFirstName	DependentLastName	BirthDate
1	102	Dan	Bacon	Application Specialist	12	Joseph	Bechhoff	1993-08-07
2	34	Susan	Eaton	Stockler	3	Michael	Erickson	2005-07-01
3	9	Gail	Erickson	Design Engineer	1	Martha	Eaton	2010-06-07
4	9	Gail	Erickson	Design Engineer	2	William	Eaton	2008-11-02
5	71	Wendy	Kahn	Finance Manager	15	Sheila	Valdez	1997-08-22
6	71	Wendy	Kahn	Finance Manager	16	Carla	Valdez	1998-12-02
7	71	Wendy	Kahn	Finance Manager	17	Maria	Valdez	2000-08-01
8	72	Kim	Ralls	Stockler	4	Robert	Ralls	1999-04-25
9	72	Kim	Ralls	Stockler	5	Marcia	Ralls	1998-09-17
10	72	Kim	Ralls	Stockler	6	Barbara	Ralls	2001-11-12
11	72	Kim	Ralls	Stockler	7	Theodore	Ralls	2005-08-01
12	72	Kim	Ralls	Stockler	8	Patricia	Ralls	2003-12-22
13	94	Candy	Spoon	Accounts Receivable Specialist	13	Christopher	Hilman	1994-02-17
14	94	Candy	Spoon	Accounts Receivable Specialist	14	Jason	Hilman	1995-07-21

Figure 6

## Outdated Join Syntax

Join conditions can be specified in either the FROM clause or WHERE clause.

Thus far we have covered the newer form of join conditions according to the American National Standards Institute where the join condition is formed in the FROM clause. This is the SQL-92 standard. However, the older standard forms the join condition in the WHERE clause and is still considered valid today. You may see it used depending on where you're employed.

As an example of the older version, I've written the SELECT statement below, based on the query we've already worked on involving Employees and EmployeesDependents. The results are identical to our work earlier where the query uses the join in the FROM clause.

```
SELECT E.EmployeeID,
       E.FirstName EmployeeFirstName,
       E.LastName EmployeeLastName,
       E.Title,
       D.DependentID,
       D.FirstName DependentFirstName,
       D.LastName DependentLastName,
       D.Birthdate
FROM Employees E, EmployeesDependents D
WHERE E.EmployeeID = D.EmployeeID
```

## Cartesian Product

When SQL combines tables in a join it first considers all possible combinations of the tables. This is called a Cartesian product. For example, if we use the two tables we've been working on, Employees and EmployeesDependents we see there are 102 rows in the Employees table and 17 rows in the EmployeesDependents table. That means there are 1,734 combinations ( $102 \times 17$ ).

Let's try this out. If we issue the following, SELECT statement without a join and order by EmployeeID we see the results of a Cartesian product in Figure 7.

```
SELECT E.EmployeeID,
       E.FirstName EmployeeFirstName,
       E.LastName EmployeeLastName,
       E.Title,
       D.DependentID,
       D.FirstName DependentFirstName,
       D.LastName DependentLastName,
       D.Birthdate
FROM Employees E, EmployeesDependents D
ORDER BY E.EmployeeID
```

	EmployeeID	EmployeeFirstName	EmployeeLastName	Title	DependentID	DependentFirstName	DependentLastName	Birthdate
1	2	Kevin	Brown	Marketing Assistant	1	Martha	Eaton	2010-06-07
2	2	Kevin	Brown	Marketing Assistant	2	William	Eaton	2008-11-02
3	2	Kevin	Brown	Marketing Assistant	3	Michael	Erickson	2005-07-01
4	2	Kevin	Brown	Marketing Assistant	4	Robert	Ralls	1999-04-25
5	2	Kevin	Brown	Marketing Assistant	5	Marcia	Ralls	1998-09-17
6	2	Kevin	Brown	Marketing Assistant	6	Barbara	Ralls	2001-11-12
7	2	Kevin	Brown	Marketing Assistant	7	Theodore	Ralls	2005-08-01
8	2	Kevin	Brown	Marketing Assistant	8	Patricia	Ralls	2003-12-22
9	2	Kevin	Brown	Marketing Assistant	9	Samuel	Ajenstat	1995-04-01
10	2	Kevin	Brown	Marketing Assistant	10	Heidi	Ajenstat	1997-10-07
11	2	Kevin	Brown	Marketing Assistant	11	Wendy	Ajenstat	1999-08-27
12	2	Kevin	Brown	Marketing Assistant	12	Joseph	Bischoff	1993-08-07
13	2	Kevin	Brown	Marketing Assistant	13	Christopher	Hilman	1994-02-17
14	2	Kevin	Brown	Marketing Assistant	14	Jason	Hilman	1995-07-21
15	2	Kevin	Brown	Marketing Assistant	15	Shella	Valdez	1997-08-22
16	2	Kevin	Brown	Marketing Assistant	16	Carla	Valdez	1998-12-02
17	2	Kevin	Brown	Marketing Assistant	17	Maria	Valdez	2000-08-01
18	3	Roberto	Tamburello	Engineering Man...	1	Martha	Eaton	2010-06-07
19	3	Roberto	Tamburello	Engineering Man...	2	William	Eaton	2008-11-02
20	3	Roberto	Tamburello	Engineering Man...	3	Michael	Erickson	2005-07-01
21	3	Roberto	Tamburello	Engineering Man...	4	Robert	Ralls	1999-04-25
22	3	Roberto	Tamburello	Engineering Man...	5	Marcia	Ralls	1998-09-17
23	3	Roberto	Tamburello	Engineering Man...	6	Barbara	Ralls	2001-11-12
24	3	Roberto	Tamburello	Engineering Man...	7	Theodore	Ralls	2005-08-01
25	3	Roberto	Tamburello	Engineering Man...	8	Patricia	Ralls	2003-12-22

Figure 7

As can be seen from the partial result set in Figure 7, EmployeeID number 2 is being matched with every DependentID. Then the next EmployeeID number 3 is going through the same match and it will continue that way through all the employees. An appropriately written JOIN clause will prevent a Cartesian product. In this case joining the Employees table EmployeeID joined to the EmployeesDependents table EmployeeID will result in the proper number of rows returned.

## Know Your Data

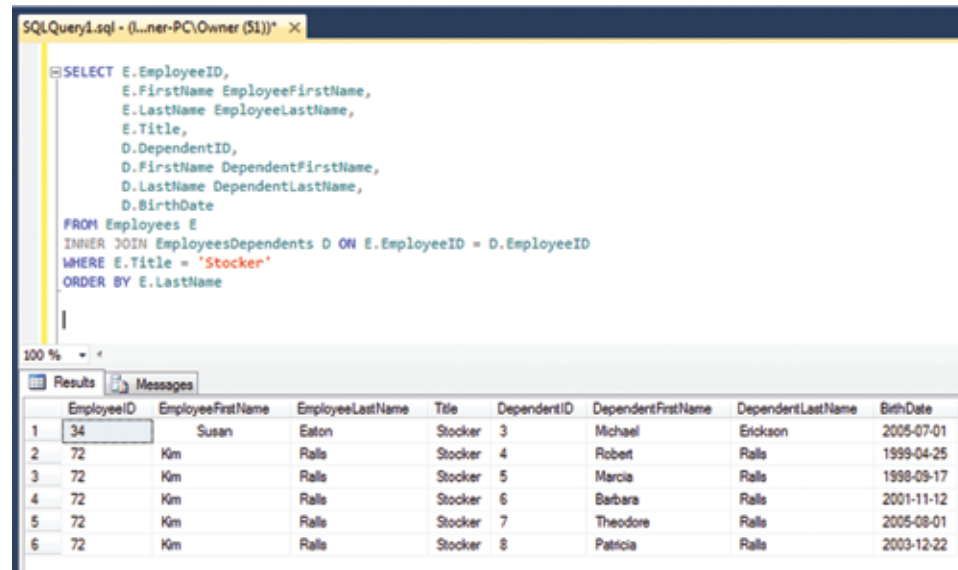
It's important that you understand the detail of your data to make sure PROPER results are returned. As you can see, we may be able to retrieve result sets as we did with the Cartesian product, but that doesn't guarantee our SQL statement is correct.

Before executing your SQL statements, you should understand what the result sets will contain. To do this, you should study the data in your tables and perform multiple tests before providing the ending results to your customer.



## Using the WHERE Clause with Joins

You are free to use the WHERE clause with joins to filter your result sets. In addition, the WHERE clause can apply to any of the tables used in the join. Figure 8 shows an example of joining the Employees and EmployeesDependents table and using the WHERE clause on the Employees table to limit the result set to employees with the title of "Stocker." Notice I've included the dependents' birth dates in this query.



```

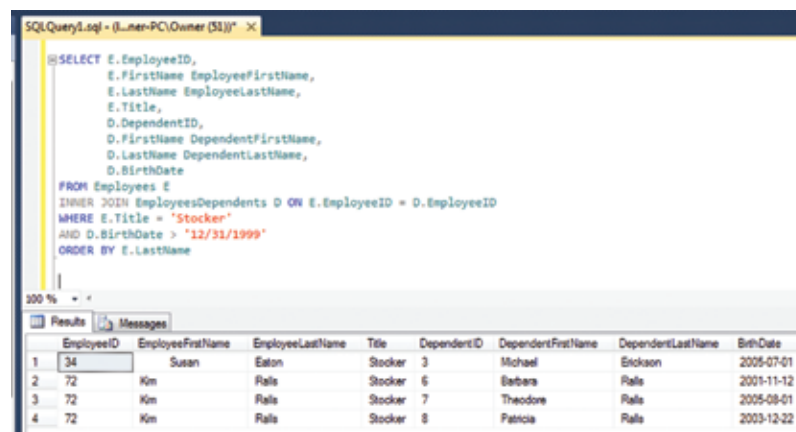
SELECT E.EmployeeID,
       E.FirstName EmployeeFirstName,
       E.LastName EmployeeLastName,
       E.Title,
       D.DependentID,
       D.FirstName DependentFirstName,
       D.LastName DependentLastName,
       D.BirthDate
FROM Employees E
INNER JOIN EmployeesDependents D ON E.EmployeeID = D.EmployeeID
WHERE E.Title = 'Stocker'
ORDER BY E.LastName

```

	EmployeeID	EmployeeFirstName	EmployeeLastName	Title	DependentID	DependentFirstName	DependentLastName	BirthDate
1	34	Susan	Eaton	Stocker	3	Michael	Erickson	2005-07-01
2	72	Kim	Rails	Stocker	4	Robert	Rails	1999-04-25
3	72	Kim	Rails	Stocker	5	Marcia	Rails	1998-09-17
4	72	Kim	Rails	Stocker	6	Barbara	Rails	2001-11-12
5	72	Kim	Rails	Stocker	7	Theodore	Rails	2005-08-01
6	72	Kim	Rails	Stocker	8	Patricia	Rails	2003-12-22

Figure 8

Figure 9 shows an example of filtering the result set even further by finding dependents who have birth dates greater than 12/31/1999. As a result, dependents Robert and Marcia are not included in the result set.



```

SELECT E.EmployeeID,
       E.FirstName EmployeeFirstName,
       E.LastName EmployeeLastName,
       E.Title,
       D.DependentID,
       D.FirstName DependentFirstName,
       D.LastName DependentLastName,
       D.BirthDate
FROM Employees E
INNER JOIN EmployeesDependents D ON E.EmployeeID = D.EmployeeID
WHERE E.Title = 'Stocker'
AND D.BirthDate > '12/31/1999'
ORDER BY E.LastName

```

	EmployeeID	EmployeeFirstName	EmployeeLastName	Title	DependentID	DependentFirstName	DependentLastName	BirthDate
1	34	Susan	Eaton	Stocker	3	Michael	Erickson	2005-07-01
2	72	Kim	Rails	Stocker	6	Barbara	Rails	2001-11-12
3	72	Kim	Rails	Stocker	7	Theodore	Rails	2005-08-01
4	72	Kim	Rails	Stocker	8	Patricia	Rails	2003-12-22

Figure 9

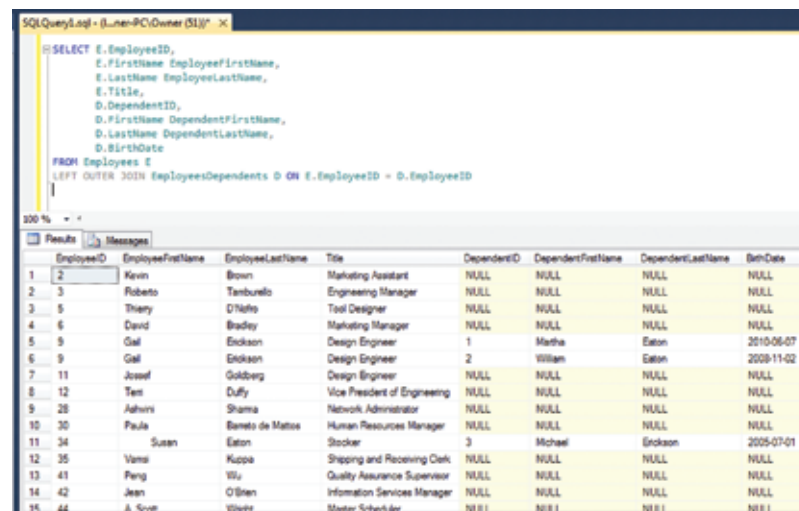


## OUTER JOIN

While INNER JOINS only return rows where tables have a match. OUTER JOINS are used when the result set needs to contain rows from one table regardless of whether there is a match in a second table. For example, not all employees have dependents. But, what if your manager wanted to see a list of all employees and their dependents, if they exist. The syntax would be as follows:

```
SELECT E.EmployeeID,
       E.FirstName EmployeeFirstName,
       E.LastName EmployeeLastName,
       E.Title,
       D.DependentID,
       D.FirstName DependentFirstName,
       D.LastName DependentLastName,
       D.BirthDate
FROM Employees E
LEFT OUTER JOIN EmployeesDependents D ON E.EmployeeID = D.EmployeeID
```

LEFT OUTER JOIN tells SQL Server that the leftmost tables in our join should include all rows regardless of whether there is a match or not. The leftmost table in our example is Employees and as you can see from the subset in Figure 10, all employees are displayed. If they don't have dependents NULL values are placed in the columns that belong to EmployeesDependents. For those employees that have dependents, matches were found, therefore EmployeesDependents data is retrieved.



EmployeeID	EmployeeFirstName	EmployeeLastName	Title	DependentID	DependentFirstName	DependentLastName	BirthDate
1	Kevin	Brown	Marketing Assistant	NULL	NULL	NULL	NULL
2	Roberto	Tamburello	Engineering Manager	NULL	NULL	NULL	NULL
3	Thierry	D'Neef	Tool Designer	NULL	NULL	NULL	NULL
4	David	Bradley	Marketing Manager	NULL	NULL	NULL	NULL
5	Gail	Ekkoen	Design Engineer	1	Martha	Eaton	2010-06-07
6	Gail	Ekkoen	Design Engineer	2	William	Eaton	2009-11-02
7	Josef	Goldberg	Design Engineer	NULL	NULL	NULL	NULL
8	Toni	Duffy	Vice President of Engineering	NULL	NULL	NULL	NULL
9	Adrian	Shams	Network Administrator	NULL	NULL	NULL	NULL
10	Paula	Boneto de Mattos	Human Resources Manager	NULL	NULL	NULL	NULL
11	Susan	Eaton	Stocker	3	Michael	Ekkoen	2005-07-01
12	Vamsi	Koppa	Shipping and Receiving Clerk	NULL	NULL	NULL	NULL
13	Peng	Wu	Quality Assurance Supervisor	NULL	NULL	NULL	NULL
14	Jean	O'Brien	Information Services Manager	NULL	NULL	NULL	NULL
15	A. Scott	Wheeler	Master Scheduler	NULL	NULL	NULL	NULL

Figure 10

Let's look at another example to further explain the OUTER JOIN. Let's say a golf course has members and they assign members to golf carts. A member can be assigned to only one golf cart at a time and a golf cart can only have one member assigned to it. This is a one-to-one relationship. The tables are displayed below in Figure 11.

MEMBERS							GOLF CARTS					
	MemberID	First	Last	Mi	Sex	DOB	CartID	Member...	Make	Color	Year	
1	1	David	Bradley	M	M	1965-04-19	1	2	Club Car	Blue	2005	
2	2	Gail	Erickson	A	F	1942-10-29	2	NULL	Club Car	Red	2007	
3	3	Macheel	Smith	A	M	1949-04-11	3	NULL	Honda	White	2003	
4	4	Angela	Dimauro	M	F	1961-09-01	4	3	Toyota	Black	2009	
5	5	Ashvini	Sharma	R	M	1967-04-28	5	4	Honda	Silver	2008	
6	6	Paula	Barreto de Mattos	M	F	1966-03-14	6	NULL	Club Car	Orange	2001	
7	7	Susan	Eaton	W	F	1968-03-20	7	NULL	Honda	Silver	2002	
8	8	Vamsi	Kuppe	N	M	1967-04-19	8	6	Club Car	Green	2010	
9	9	Rebecca	Mercurio		F	1965-04-19						
10	10	Mike	Berel	A	M	1964-09-22						
11	11	Jodie	Foster	F	F	1971-09-14						
12	12	Louise	Scott	C	F	1975-11-01						
13	13	Scott	Ridgway		M	1987-02-11						
14	14	Sean	Combs	T	M	1990-05-23						
15	15	Rose	Burgeo		F	1985-12-24						

Figure 11

The MemberID is the primary key for the MEMBERS table and MemberID is the foreign key for the GOLF CARTS table. The GOLF CARTS table also has a primary key entitled CartID.

If we performed an INNER JOIN then only members with golf carts assigned to them would be retrieved along with the golf carts they've been assigned. This is because only matches show up. See the INNER JOIN results in Figure 12.

```

SELECT M.MemberID,
       M.First,
       M.Last,
       M.DOB,
       G.CartID,
       G.Make,
       G.Color,
       G.Year
FROM Members M
INNER JOIN GolfCarts G ON M.MemberID = G.MemberID

```

	MemberID	First	Last	DOB	CartID	Make	Color	Year
1	2	Gail	Erickson	1942-10-29	1	Club Car	Blue	2005
2	3	Macheel	Smith	1949-04-11	4	Toyota	Black	2009
3	4	Angela	Dimauro	1961-09-01	5	Honda	Silver	2008
4	6	Paula	Barreto de Mattos	1966-03-14	8	Club Car	Green	2010

Figure 12

Let's say the manager of the golf course wants to see a list of all members regardless if they have golf carts assigned to them, he also wants to see the golf cart information when applicable. The SQL Syntax is shown below and the result set follows in Figure 13.

```
SELECT M.MemberID,
       M.First,
       M.Last,
       M.DOB,
       G.CartID,
       G.Make,
       G.Color,
       G.Year
FROM Members M
LEFT OUTER JOIN GolfCarts G ON M.MemberID = G.MemberID
```

```
SELECT M.MemberID,
       M.First,
       M.Last,
       M.DOB,
       G.CartID,
       G.Make,
       G.Color,
       G.Year
FROM Members M
LEFT OUTER JOIN GolfCarts G ON M.MemberID = G.MemberID
```

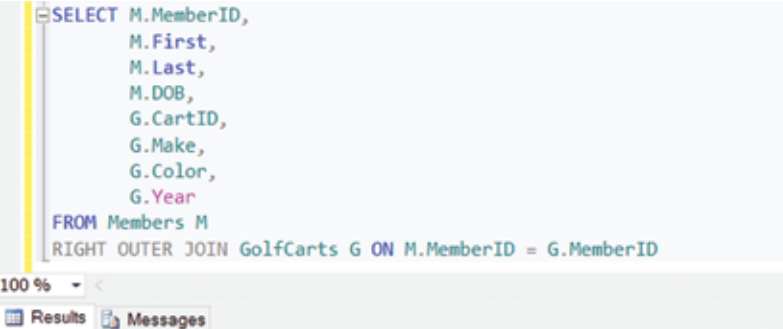
	MemberID	First	Last	DOB	CartID	Make	Color	Year
1	1	David	Bradley	1965-04-19	NULL	NULL	NULL	NULL
2	2	Gail	Erickson	1942-10-29	1	Club Car	Blue	2005
3	3	Michael	Smith	1949-04-11	4	Toyota	Black	2009
4	4	Angela	Dimauro	1961-09-01	5	Honda	Silver	2008
5	5	Ashvini	Sharma	1967-04-28	NULL	NULL	NULL	NULL
6	6	Paula	Barreto de Mattos	1966-03-14	8	Club Car	Green	2010
7	7	Susan	Eaton	1968-03-20	NULL	NULL	NULL	NULL
8	8	Vamsi	Kuppa	1967-04-19	NULL	NULL	NULL	NULL
9	9	Rebecca	Mercurio	1965-04-19	NULL	NULL	NULL	NULL
10	10	Mike	Berel	1964-09-22	NULL	NULL	NULL	NULL
11	11	Jodie	Foster	1971-09-14	NULL	NULL	NULL	NULL
12	12	Louise	Scott	1975-11-01	NULL	NULL	NULL	NULL
13	13	Scott	Ridgway	1967-02-11	NULL	NULL	NULL	NULL
14	14	Sean	Combs	1990-05-23	NULL	NULL	NULL	NULL
15	15	Rose	Bungeo	1985-12-24	NULL	NULL	NULL	NULL
16	16	Willie	Peppin	1971-09-12	NULL	NULL	NULL	NULL
17	17	Michael	Smith	1989-02-04	NULL	NULL	NULL	NULL
18	18	Mark	Thompson	1988-02-01	NULL	NULL	NULL	NULL

Figure 13

Notice the NULL values in the GOLF CART table columns. This is because member with NULL values don't have golf carts assigned to them. But the manager wanted to see them in the list anyway because he wants to see all members whether they have golf carts. Because we performed a LEFT OUTER JOIN where the MEMBERS table is on the left side of the JOIN operation, all members are shown. But not all golf carts are shown.

What would happen if we performed a RIGHT OUTER JOIN? In other words, let's show all golf carts regardless if it has a match in the MEMBERS table. Since the GOLF CART table is on the right side of the JOIN operation we will issue a RIGHT OUTER JOIN statement as shown below. The result set follows in Figure 14.

```
SELECT M.MemberID,
       M.FirstName,
       M.LastName,
       M.DOB,
       G.CartID,
       G.Make,
       G.Color,
       G.Year
FROM Members M
RIGHT OUTER JOIN GolfCarts G ON M.MemberID = G.MemberID
```



	MemberID	First	Last	DOB	CartID	Make	Color	Year
1	2	Gail	Erickson	1942-10-29	1	Club Car	Blue	2005
2	NULL	NULL	NULL	NULL	2	Club Car	Red	2007
3	NULL	NULL	NULL	NULL	3	Honda	White	2003
4	3	Michael	Smith	1949-04-11	4	Toyota	Black	2009
5	4	Angela	Dimauro	1961-09-01	5	Honda	Silver	2008
6	NULL	NULL	NULL	NULL	6	Club Car	Orange	2001
7	NULL	NULL	NULL	NULL	7	Honda	Silver	2002
8	6	Paula	Barreto de Mattos	1966-03-14	8	Club Car	Green	2010

Figure 14

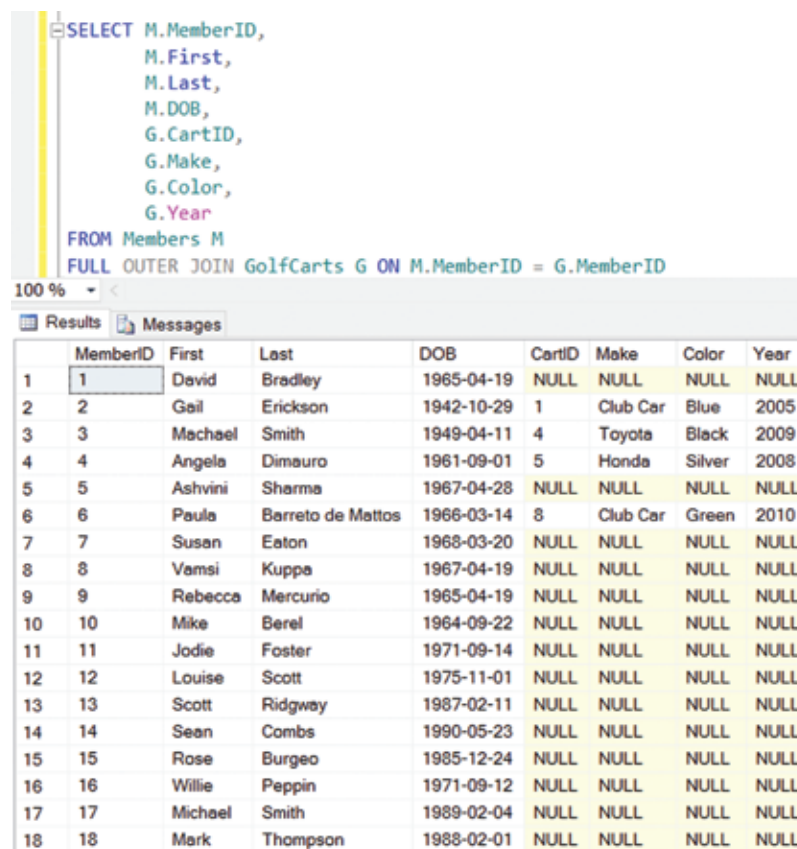
Notice that all the golf cart information is visible while only member information for those who have golf carts assigned are shown. NULL values exist in the member information columns where golf carts don't have members assigned to them.

When performing OUTER JOINS remember that the position of the table you want all information to show from dictates whether it is a LEFT or RIGHT OUTER JOIN. If the table on the left of the JOIN is the table, you want to see all information on then it is a LEFT OUTER JOIN. If it is on the right of the JOIN, then it is a RIGHT OUTER JOIN.

## FULL OUTER JOIN

The FULL OUTER JOIN returns all data from both tables no matter if there is a match or not. In other words, it performs both a LEFT and RIGHT OUTER JOINS. The syntax is shown below and the result set follows in Figure 15.

```
SELECT M.MemberID,
       M.First,
       M.Last,
       M.DOB,
       G.CartID,
       G.Make,
       G.Color,
       G.Year
FROM Members M
FULL OUTER JOIN GolfCarts G ON M.MemberID = G.MemberID
```



	MemberID	First	Last	DOB	CartID	Make	Color	Year
1	1	David	Bradley	1965-04-19	NULL	NULL	NULL	NULL
2	2	Gail	Erickson	1942-10-29	1	Club Car	Blue	2005
3	3	Macheel	Smith	1949-04-11	4	Toyota	Black	2009
4	4	Angela	Dimauro	1961-09-01	5	Honda	Silver	2008
5	5	Ashvini	Sharma	1967-04-28	NULL	NULL	NULL	NULL
6	6	Paula	Barreto de Mattos	1966-03-14	8	Club Car	Green	2010
7	7	Susan	Eaton	1968-03-20	NULL	NULL	NULL	NULL
8	8	Vamsi	Kuppa	1967-04-19	NULL	NULL	NULL	NULL
9	9	Rebecca	Mercurio	1965-04-19	NULL	NULL	NULL	NULL
10	10	Mike	Berel	1964-09-22	NULL	NULL	NULL	NULL
11	11	Jodie	Foster	1971-09-14	NULL	NULL	NULL	NULL
12	12	Louise	Scott	1975-11-01	NULL	NULL	NULL	NULL
13	13	Scott	Ridgway	1987-02-11	NULL	NULL	NULL	NULL
14	14	Sean	Combs	1990-05-23	NULL	NULL	NULL	NULL
15	15	Rose	Burgeio	1985-12-24	NULL	NULL	NULL	NULL
16	16	Willie	Peppin	1971-09-12	NULL	NULL	NULL	NULL
17	17	Michael	Smith	1989-02-04	NULL	NULL	NULL	NULL
18	18	Mark	Thompson	1988-02-01	NULL	NULL	NULL	NULL

Figure 15

Notice the result set in Figure 15 contains NULL values on both tables but all the information is shown for both tables as well.

## SELF JOINS

A SELF JOIN is required when the relationship exists in a single table. Our Employees table is a table that contains a relationship to itself. The table contains all employees and related information including the manager they report to. Since managers are also employees they exist in the same Employees table. Let's look at this. The SELECT statement below lists employees' names and their manager ID. The results are shown in Figure 16.

```
SELECT EmployeeID,  
       FirstName,  
       LastName,  
       ManagerID  
FROM EMPLOYEES
```

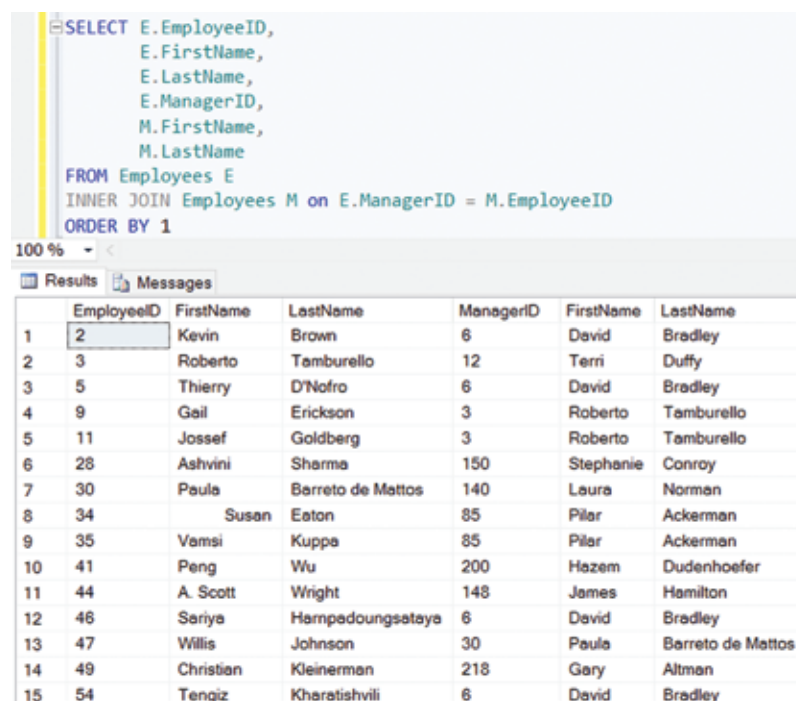
	EmployeeID	FirstName	LastName	ManagerID
1	2	Kevin	Brown	6
2	3	Roberto	Tamburello	12
3	5	Thierry	D'Nofro	6
4	6	David	Bradley	109
5	9	Gail	Erickson	3
6	11	Jossef	Goldberg	3
7	12	Terri	Duffy	109
8	28	Ashvini	Sharma	150
9	30	Paula	Barreto de Mattos	140
10	34	Susan	Eaton	85
11	35	Vamsi	Kuppa	85
12	41	Peng	Wu	200
13	42	Jean	O'Brien	109
14	44	A. Scott	Wright	148
15	46	Sariya	Harnpadoungsataya	6

Figure 16

The colored arrows in Figure 16 point to the names of a few of the managers of employees. From this small subset of data, we can see that Kevin Brown's manager's name is David Bradley, Gail Erickson's manager is Roberto Tamburello, and Roberto Tamburello's manager is Terri Duffy.

What if we want to see the actual names of the managers and not just the ManagerID within each row. The following SELF JOIN is necessary to accomplish this. Notice the highlight in the query above where I used the alias "M" for the manager's first and last name. The results are shown in Figure 17.

```
SELECT E.EmployeeID,
       E.FirstName,
       E.LastName,
       E.ManagerID,
       M.FirstName,
       M.LastName
FROM Employees E
INNER JOIN Employees M on E.ManagerID = M.EmployeeID
ORDER BY E.EmployeeID
```



```
SELECT E.EmployeeID,
       E.FirstName,
       E.LastName,
       E.ManagerID,
       M.FirstName,
       M.LastName
FROM Employees E
INNER JOIN Employees M on E.ManagerID = M.EmployeeID
ORDER BY 1
```

	EmployeeID	FirstName	LastName	ManagerID	FirstName	LastName
1	2	Kevin	Brown	6	David	Bradley
2	3	Roberto	Tamburello	12	Terri	Duffy
3	5	Thierry	D'Nofro	6	David	Bradley
4	9	Gail	Erickson	3	Roberto	Tamburello
5	11	Jossef	Goldberg	3	Roberto	Tamburello
6	28	Ashvini	Sharma	150	Stephanie	Conroy
7	30	Paula	Barreto de Mattos	140	Laura	Norman
8	34	Susan	Eaton	85	Pilar	Ackerman
9	35	Vamsi	Kuppa	85	Pilar	Ackerman
10	41	Peng	Wu	200	Hazem	Dudenhoefer
11	44	A. Scott	Wright	148	James	Hamilton
12	46	Seriya	Hampedoungsataya	6	David	Bradley
13	47	Willis	Johnson	30	Paula	Barreto de Mattos
14	49	Christian	Kleinerman	218	Gary	Altman
15	54	Tengiz	Kharatishvili	6	David	Bradley

Figure 17



SELF JOINS can be confusing because they are joining to the same table. But because it requires two alias names to refer to the same table you can visually see a partition in table. In our example, I used the letter “E” to represent employees and the letter “M” to represent managers. I’m able to retrieve the managers’ names because I’m connecting the Employees table (E) ManagerID to the Manager table (M – really the Employees table) EmployeeID to retrieve the name of the Manager.

## Joining on Multiple Tables

Often, joins will require using more than two tables. The diagram below in Figure 18 shows three related tables. We know that many employees switch departments as they grow in their careers. Therefore, many employees can belong to many departments at different times in their employment with the same company. We track this by start and end dates. Likewise, departments can have many employees assigned to them. Therefore, this scenario is a many-to-many relationship.

When we have many-to-many relationships, as discussed in earlier chapters, we need to create a so-called bridge table to make the relationship one-to-many. That bridge table will contain a composite key consisting of the EmployeeID and DepartmentID. Additional information will be the start and end date the employee was assigned to the department and the date the table was modified to reflect the change in departments.

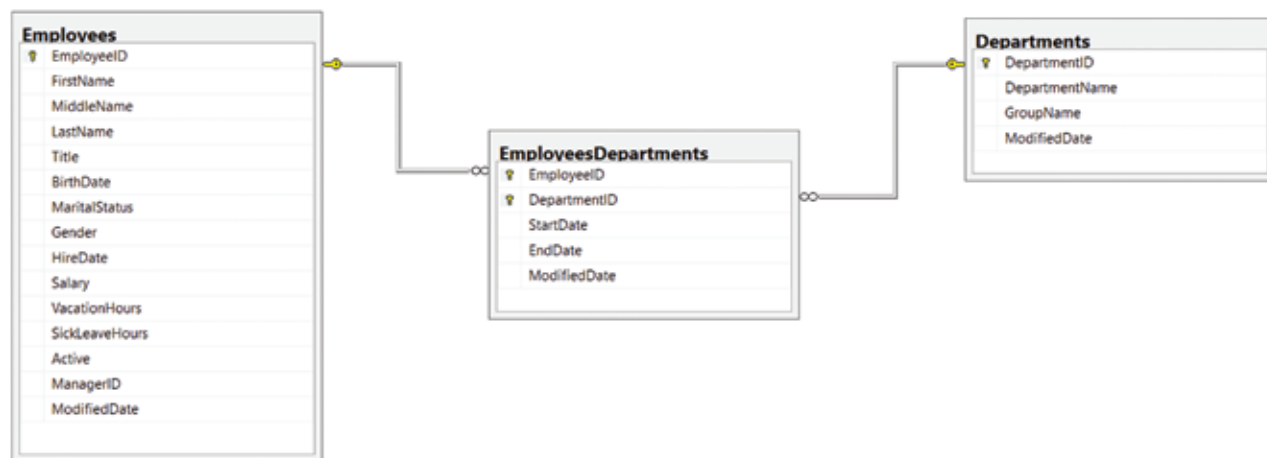
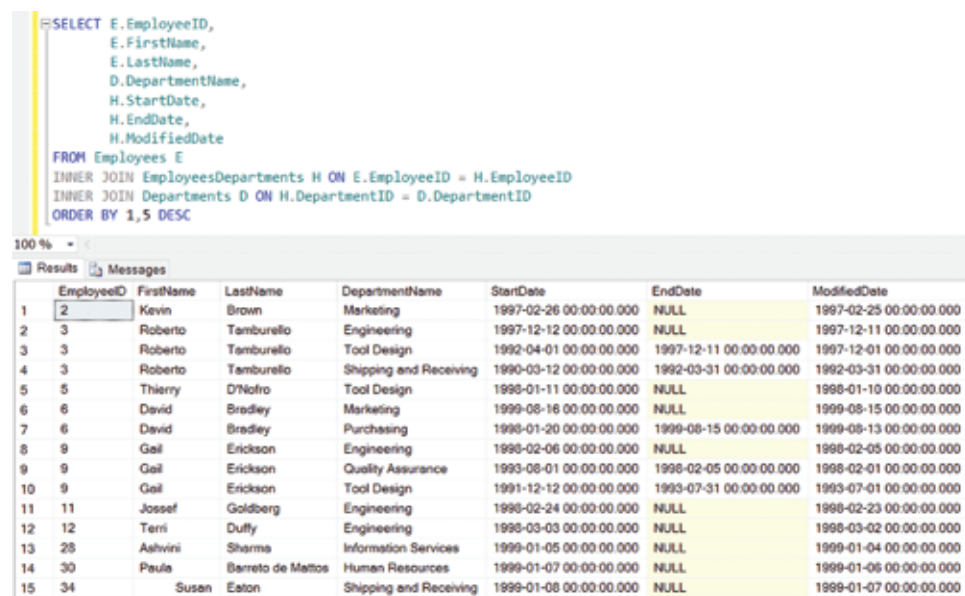


Figure 18

Let's say the manager wants to see a list of employee names and the various departments they were assigned to during their careers, along with the dates in which they were assigned to them. The query below joins three tables to achieve this request. The result set is shown in Figure 19.

```
SELECT E.EmployeeID,
       E.FirstName,
       E.LastName,
       D.DepartmentName,
       H.StartDate,
       H.EndDate,
       H.ModifiedDate
FROM Employees E
INNER JOIN EmployeesDepartments H ON E.EmployeeID = H.EmployeeID
INNER JOIN Departments D ON H.DepartmentID = D.DepartmentID
ORDER BY E.EmployeeID, H.StartDate DESC
```



```
SELECT E.EmployeeID,
       E.FirstName,
       E.LastName,
       D.DepartmentName,
       H.StartDate,
       H.EndDate,
       H.ModifiedDate
FROM Employees E
INNER JOIN EmployeesDepartments H ON E.EmployeeID = H.EmployeeID
INNER JOIN Departments D ON H.DepartmentID = D.DepartmentID
ORDER BY 1,5 DESC
```

	EmployeeID	FirstName	LastName	DepartmentName	StartDate	EndDate	ModifiedDate
1	2	Kevin	Brown	Marketing	1997-02-26 00:00:00.000	NULL	1997-02-25 00:00:00.000
2	3	Roberto	Tamburello	Engineering	1997-12-12 00:00:00.000	NULL	1997-12-11 00:00:00.000
3	3	Roberto	Tamburello	Tool Design	1992-04-01 00:00:00.000	1997-12-11 00:00:00.000	1997-12-01 00:00:00.000
4	3	Roberto	Tamburello	Shipping and Receiving	1990-03-12 00:00:00.000	1992-03-31 00:00:00.000	1992-03-31 00:00:00.000
5	5	Thierry	D'Nofro	Tool Design	1998-01-11 00:00:00.000	NULL	1998-01-10 00:00:00.000
6	6	David	Bradley	Marketing	1999-08-16 00:00:00.000	NULL	1999-08-15 00:00:00.000
7	6	David	Bradley	Purchasing	1998-01-20 00:00:00.000	1999-08-15 00:00:00.000	1999-08-13 00:00:00.000
8	9	Gail	Erickson	Engineering	1998-02-06 00:00:00.000	NULL	1998-02-05 00:00:00.000
9	9	Gail	Erickson	Quality Assurance	1993-08-01 00:00:00.000	1998-02-05 00:00:00.000	1998-02-01 00:00:00.000
10	9	Gail	Erickson	Tool Design	1991-12-12 00:00:00.000	1993-07-31 00:00:00.000	1993-07-01 00:00:00.000
11	11	Jossef	Goldberg	Engineering	1998-02-24 00:00:00.000	NULL	1998-02-23 00:00:00.000
12	12	Terri	Duffy	Engineering	1998-03-03 00:00:00.000	NULL	1998-03-02 00:00:00.000
13	28	Ashvini	Sharma	Information Services	1999-01-05 00:00:00.000	NULL	1999-01-04 00:00:00.000
14	30	Paule	Barreto de Mattos	Human Resources	1999-01-07 00:00:00.000	NULL	1999-01-06 00:00:00.000
15	34	Susan	Eaton	Shipping and Receiving	1999-01-08 00:00:00.000	NULL	1999-01-07 00:00:00.000

Figure 19

We can retrieve EmployeeID, FirstName, and LastName from the Employees table. We join to the EmployeesDepartments table using EmployeeID to retrieve the StartDate, EndDate, and ModifiedDate. However, to retrieve the actual DepartmentName we need to join to the Departments table and we do this using the DepartmentID from the EmployeesDepartments table to join to the DepartmentID in the Departments table. The query is then ordered by EmployeeID ascending and StartDate descending to display the latest department first.

Notice Roberto Tamburello in the result set in Figure 19. We can view the dates he was assigned to the various departments.

What if the manager wanted to add the state where the employee lives to the result set above. We would then need to join to a fourth table called Addresses as shown below. Notice the yellow highlighted line showing the join to the Addresses table. The result set follows in Figure 20.

```
SELECT E.EmployeeID,
       E.FirstName,
       E.LastName,
       A.State,
       D.DepartmentName,
       H.StartDate,
       H.EndDate,
       H.ModifiedDate
FROM Employees E
INNER JOIN EmployeesDepartments H ON E.EmployeeID = H.EmployeeID
INNER JOIN Departments D ON H.DepartmentID = D.DepartmentID
INNER JOIN ADDRESSES A ON E.EmployeeID = A.EmployeeID
ORDER BY E.EmployeeID, H.StartDate DESC
```

	EmployeeID	FirstName	LastName	St...	DepartmentName	StartDate	EndDate	ModifiedDate
1	2	Kevin	Brown	WA	Marketing	1997-02-25 00:00:00.000	NULL	1997-02-25 00:00:00.000
2	3	Roberto	Tamburello	WA	Engineering	1997-12-12 00:00:00.000	NULL	1997-12-11 00:00:00.000
3	3	Roberto	Tamburello	WA	Tool Design	1992-04-01 00:00:00.000	1997-12-11 00:00:00.000	1997-12-01 00:00:00.000
4	3	Roberto	Tamburello	WA	Shipping and Receiving	1990-03-12 00:00:00.000	1992-03-31 00:00:00.000	1992-03-31 00:00:00.000
5	5	Thierry	D'Alecy	WA	Tool Design	1998-01-11 00:00:00.000	NULL	1998-01-10 00:00:00.000
6	6	David	Bradley	WA	Marketing	1999-05-15 00:00:00.000	NULL	1999-05-15 00:00:00.000
7	6	David	Bradley	WA	Purchasing	1998-01-20 00:00:00.000	1999-05-15 00:00:00.000	1999-05-13 00:00:00.000
8	9	Gail	Erickson	WA	Engineering	1998-02-06 00:00:00.000	NULL	1998-02-05 00:00:00.000
9	9	Gail	Erickson	WA	Quality Assurance	1998-05-01 00:00:00.000	1998-02-05 00:00:00.000	1998-02-01 00:00:00.000
10	9	Gail	Erickson	WA	Tool Design	1991-12-12 00:00:00.000	1998-07-31 00:00:00.000	1998-07-01 00:00:00.000
11	11	Josef	Goldberg	WA	Engineering	1998-02-24 00:00:00.000	NULL	1998-02-23 00:00:00.000
12	12	Terri	Duffy	WA	Engineering	1998-03-03 00:00:00.000	NULL	1998-03-02 00:00:00.000
13	28	Adhvi	Sharma	WA	Information Services	1999-01-05 00:00:00.000	NULL	1999-01-04 00:00:00.000
14	30	Paula	Barneto de Matos	WA	Human Resources	1999-01-07 00:00:00.000	NULL	1999-01-06 00:00:00.000
15	34	Susan	Eaton	WA	Shipping and Receiving	1999-01-08 00:00:00.000	NULL	1999-01-07 00:00:00.000

Figure 20

## Summary

This chapter exposed you to JOIN operations that retrieve information from related tables. You now understand the purpose of INNER JOINS in retrieving information on matching columns as well as the OUTER JOINS which may not have matching columns to retrieve all information from a table regardless of matches. SELF JOINS were covered to show that a relationship can exist among one table as in employees and managers. In addition, the dangers of producing Cartesian products with improper joins were discussed. You also understand that multiple joins can be used to retrieve information dispersed among multiple tables.

## Exercises

1. Create a list of employee names and their full addresses using the INNER JOIN.
2. Create a list of employee names and their full addresses using the INNER JOIN for employees who live either in Massachusetts or California.
3. Create an example of a Cartesian product using the Employees and Addresses table.
4. Use a LEFT OUTER JOIN to list all employee first and last names as well as the gender and birth-dates of their dependents if they have any.
5. Use a RIGHT OUTER JOIN to list all employee first and last names as well as the gender and birthdates of their dependents if they have any (Same as above but change location of tables).
6. Use a SELF JOIN to select the employee first name, last name, title as well as the manager's first name, last name, and title.
7. Retrieve the employee first name, last name, address, city, state, zip, department name, and group name for the employee named David Bradley.
8. Retrieve the employee first name, last name, dependent first name, last name, and gender for female dependents.
9. Retrieve the employee first name, last name, dependent first name, last name, birth date for dependents who have birth dates less than or equal to 11/02/2008.
10. Use LEFT OUTER JOINS to list all Employees, their Department Names, and their dependent names, if any.
11. Use RIGHT OUTER JOINS to list all Employees, their Department Names, and their dependent names, like you did in the question above.
12. Retrieve the employee first name, last name, dependent first name, last name, and gender for dependents who were born in or before 1998.
13. Retrieve all employees belonging to the department "Purchasing" using INNER JOINS.
14. Retrieve the first name, last name, and title of the manager responsible for the Employee with the first name "Ben" and last name "Miller."
15. Retrieve the employee first name, last name, department name, and group name for the employee named Gail Erickson.