Hierarchical Queries

The following is chapter 5 from my Common Table Expressions book in its entirety.

READER NOTE: *Please run the CTEBookSetup.sql script in order to follow along with the examples in this chapter. The script mentioned in this chapter is available at http://SteveStedman.com.*

Hierarchical CTEs

When looking at the table of contents for a book it is generally represented as an outline of some kind utilizing indentation, and a type of parent/child relationship between the chapters, and the sections in those chapters. Some items show up with no indentation indicating that they are the top level, followed by some indentation showing the next level, and finally more indentation to show the lower levels. It is a very simple concept for the person reading the book to understand, and it has been used for a very long time in the publishing world.

In SQL Server the use of tables does not follow this hierarchical structure, instead the table is made up of rows and columns. With a column to identify the parent relationship and with a well written CTE, you can walk through the hierarchical structure and display those relationships.

Once we have mastered the recursive query we can then fix it up a bit by adding formatting to make it look more like a hierarchical tree.

READER NOTE: *Please run the CTEBookSetup.sql script in order to follow along with the examples in this chapter. The script mentioned in this chapter is available at http://SteveStedman.com.*

READER NOTE: *Please run the FillMoreDepartments stored procedure in the cte_demo database as shown here. This will add additional department hierarchy levels into the departments table.*

```
USE [cte_demo];
execute FillMoreDepartments;
```

Hierarchical Tree Path

A tree path is used to describe the entire path through a tree. Some examples of tree paths are the Windows file system or the category levels that might be shown in an online store.

A great example of a tree path is a file system. A full path to a directory would be considered a tree path as shown in Figure 5.1.

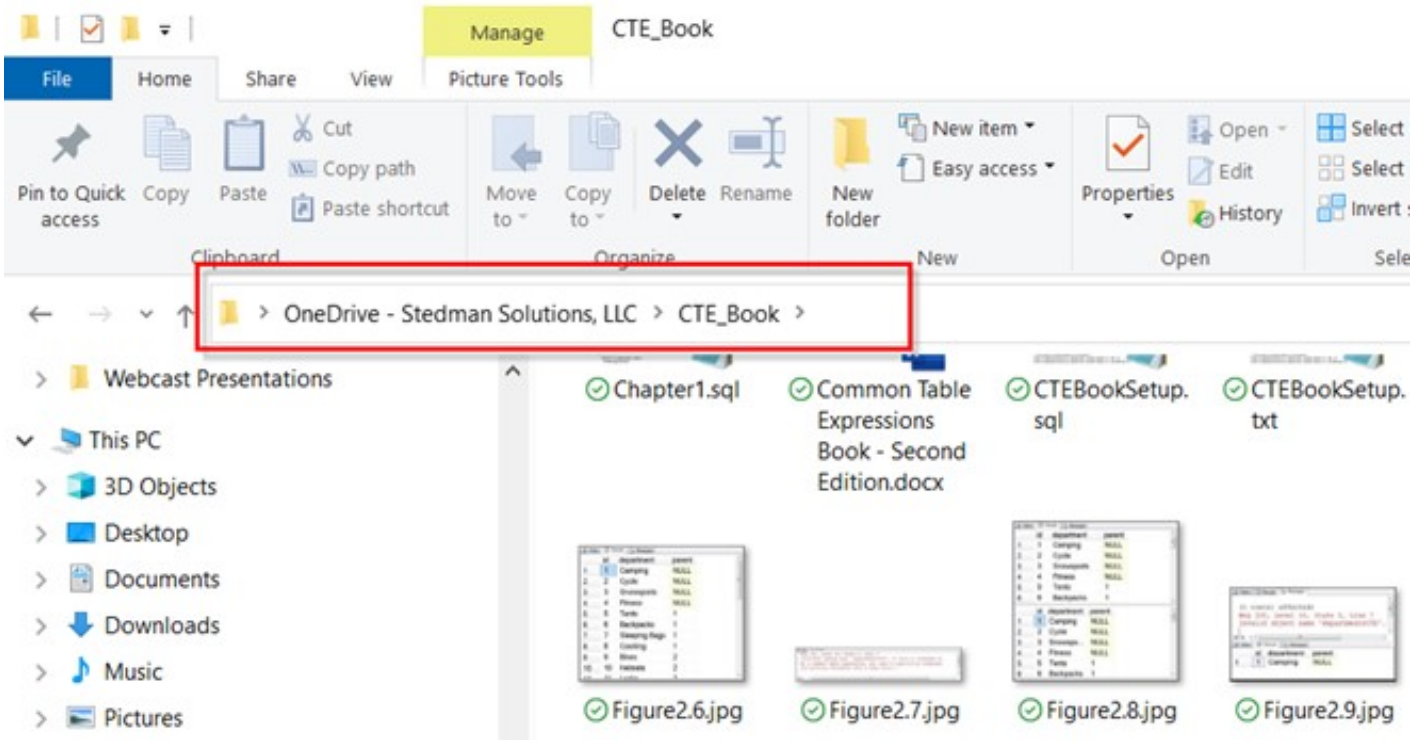


Figure 5.1 File path shown in Windows Explorer.

Another example is the path shown in the Windows Command Prompt where the current directory is shown with all Bloody well right its parents all the way to the root of the drive.

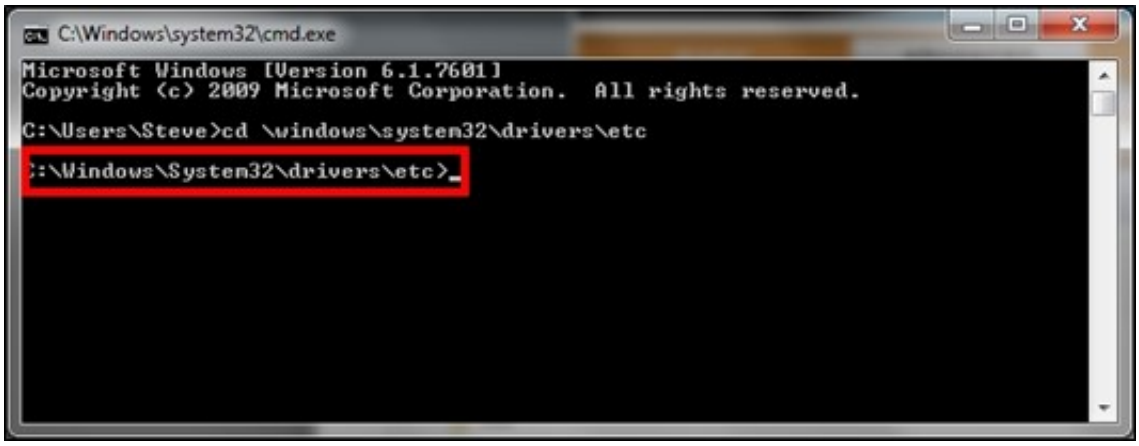


Figure 5.2 File path shown in the Windows command prompt.

Another name for tree path that is commonly used in web development is the term “bread crumbs”. The current level and all parent levels are shown as links back to any point in the hierarchy of web pages.

The tree path is very easy to build in a CTE by concatenating or adding on each of the tree levels as the query recursively links the data together. To build the tree path, the following algorithm could be applied:

1. At the top level, use the current department as the full tree path.
2. For each level of recursion add on the current department with a separator to the current path.

To implement this let’s start with the original departmentsCTE query from the last chapter as seen in the following code:

```
;WITH departmentsCTE AS
(
  SELECT id, Department, parent, 0 as Level
    FROM Departments
   WHERE parent is NULL
   UNION ALL
  SELECT d.id, d.Department, d.Parent,
         DepartmentsCTE.Level + 1 as Level
    FROM Departments d
   INNER JOIN departmentsCTE ON DepartmentCTE.id = d.Parent
)
SELECT *
  FROM departmentsCTE
 ORDER BY parent;
```

To this query we will add in a new result column called TreePath. The new column needs to be added to both the anchor query and the recursive query.

For the anchor query the full path (TreePath) is just the current department so we add a new column into the anchor query:

```
cast(Department as varchar(8000)) as TreePath
```

For the recursive query we add a reference to the existing TreePath, followed by a separator and then the current department.

```
cast(DepartmentCTE.TreePath
  + ' -> '
  + d.Department as varchar(8000)) as TreePath
```

Since we set the varchar size to 8000 in both queries we will not get the error seen in Figure 5.3. We will get an error if we do not match the data types like the following code :

```
cast(Department as varchar(7000)) as TreePath
```

```
cast(DepartmentCTE.TreePath
  + ' -> '
  + d.Department as varchar(8000)) as TreePath
```

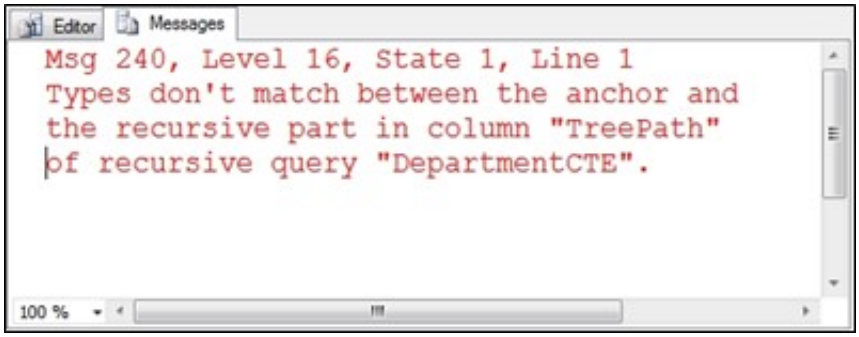


Figure 5.3 Error message if the recursive and anchor query columns are not the same type.

Given that we just have to modify two lines of the query in order to add the TreePath and since we know how to build the recursive query from the previous chapter, it shouldn't take much time to get a tree path query in place.

After adding TreePath into the anchor query and the recursive query, we end up with the following query:

```
;WITH departmentsCTE AS
(
  SELECT id, Department, parent, 0 as Level,
        cast(Department as varchar(8000)) as TreePath
    FROM Departments
   WHERE parent is NULL
   UNION ALL
  SELECT d.id, d.Department, d.Parent,
        DepartmentsCTE.Level + 1 as Level,
        cast(DepartmentCTE.TreePath
          + ' -> '
          + d.Department as varchar(8000)) as TreePath
    FROM Departments d
   INNER JOIN departmentsCTE ON DepartmentCTE.id = d.Parent
)
SELECT *
```

```
 FROM departmentsCTE
ORDER BY parent;
```

Editor | Results | Messages

	id	Department	parent	Level	TreePath
2	2	Cycle	NULL	0	Cycle
3	3	Sn	NULL	0	Snowsports
4	4	Fit			Fitness
5	28	Gifts	NULL	0	Gifts
6	29	Clearance	NULL	0	Clearance
7	5	Tents	1	1	Camping -> Tents
8	6	Backpacks	1	1	Camping -> Backpacks
9	7	Sleeping Bags	1	1	Camping -> Sleeping Bags
10	8	Cooking	1	1	Camping -> Cooking
11	9				Cycle -> Bikes
12	10		2	1	Cycle -> Helmets
13	11	Locks	2	1	Cycle -> Locks
14	12	Ski	3	1	Snowsports -> Ski
15	13	Snowboard	3	1	Snowsports -> Snowboard
16	14	Snowshoe	3	1	Snowsports -> Snowshoe
17	15	Running	4	1	Fitness -> Running
18	16	Swimming	4	1	Fitness -> Swimming
19	17	Yoga	4	1	Fitness -> Yoga
20	18	1 Person	5	2	Camping -> Tents -> 1 Person
21	19	2 Person	5	2	Camping -> Tents -> 2 Person
22	20	3 Person	5	2	Camping -> Tents -> 3 Person

Anchor query or top level departments.

Recursive query or sub-departments.

Figure 5.4 Tree path shown with anchor query results, and recursive results split out.

To improve even further on the query we can filter out just the TreePath column and order it by TreePath for a better visual representation.

```
;WITH departmentsCTE AS
(
  SELECT id, Department, parent, 0 as Level,
         cast(Department as varchar(8000)) as TreePath
    FROM Departments
   WHERE parent is NULL
   UNION ALL
  SELECT d.id, d.Department, d.Parent,
         DepartmentsCTE.Level + 1 as Level,
         cast(DepartmentCTE.TreePath
           + ' -> '
           + d.Department as varchar(8000)) as TreePath
    FROM Departments d
   INNER JOIN departmentsCTE ON DepartmentCTE.id = d.Parent
)
SELECT TreePath
  FROM departmentsCTE
 ORDER BY TreePath;
```

	TreePath
1	Camping
2	Camping -> Backpacks
3	Camping -> Cooking
4	Camping -> Sleeping Bags
5	Camping -> Tents
6	Camping -> Tents -> 1 Person
7	Camping -> Tents -> 2 Person
8	Camping -> Tents -> 2 Person -> Backpacking
9	Camping -> Tents -> 2 Person -> Family Camping
10	Camping -> Tents -> 2 Person -> Mountaineering
11	Camping -> Tents -> 2 Person -> Mountaineering -> Lightweight
12	Camping -> Tents -> 2 Person -> Mountaineering -> Standard
13	Camping -> Tents -> 2 Person -> Mountaineering -> Ultra-lightweight
14	Camping -> Tents -> 3 Person
15	Camping -> Tents -> 4 Person
16	Clearance
17	Cycle
18	Cycle -> Bikes
19	Cycle -> Helmets
20	Cycle -> Locks
21	Fitness
22	Fitness -> Running

Figure 5.5 Query results with just the tree path column, sorted by the tree path column.

The ordering of the TreePath column with the ORDER BY TreePath clause helps with the formatting by grouping departments by their parent department.

If we wanted the Department tree path to be represented more like the file system path shown earlier, the path separator could be changed.

Replace this: + ' -> '

With this: + '\'

for the results shown in Figure 5.6:

Editor | Results | Messages

	TreePath
1	Camping
2	Camping\Backpacks
3	Camping\Cooking
4	Camping\Sleeping Bags
5	Camping\Tents
6	Camping\Tents\1 Person
7	Camping\Tents\2 Person
8	Camping\Tents\2 Person\Backpacking
9	Camping\Tents\2 Person\Family Camping
10	Camping\Tents\2 Person\Mountaineering
11	Camping\Tents\2 Person\Mountaineering\Lightweight
12	Camping\Tents\2 Person\Mountaineering\Standard
13	Camping\Tents\2 Person\Mountaineering\Ultra-lightweight
14	Camping\Tents\3 Person
15	Camping\Tents\4 Person

Figure 5.6 Tree path with a backslash separator.

With these tree path techniques we can quickly visualize hierarchical data without too much work. Generating a tree path this way is much easier than other methods of doing it outside of the database.

Hierarchical Formatting

I think back to middle school when I was learning how to write a paper of some kind. We were told to start with an outline, fill out the top level topics first then fill out each of the top levels with the details, indenting each level of the outline. Theoretically I could have any number of levels of indentation, but I was generally limited by the size of the paper that I was working with.

There is a similar concept to the paper size with hierarchical data in recursive CTEs where we run

out of room to display data on the screen or in the area we are working with. The tree path solution to show hierarchical data is very useful in some areas, but if space is limited, the indentation or outline style of showing hierarchical data might be a better option.

Let’s take a look at the following outline format. To take the tree path query and modify it to show the data in this format is quite easy.

```
Camping
   Backpacks
   Cooking
   Tents
      1 Person
      2 Person
         Backpacking
         Mountaineering
         Family Camping
      3 Person
      4 Person
Cycle
Fitness
```

To start with, the REPLICATE function will be used to duplicate the indentation for our levels. The REPLICATE function takes 2 parameters; the first parameter is a varchar which is the string or item to be replicated. The second parameter is the number of times to replicate the first parameter. The following example replicates “abc.” three times.

```
SELECT REPLICATE('abc.', 3);
```

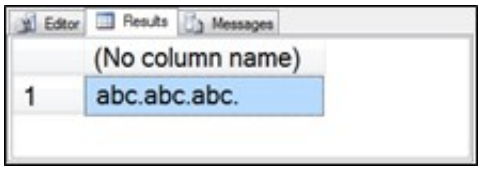


Figure 5.7 Output from the REPLICATE function.

Where the REPLICATE function becomes useful is when the depth of the hierarchical query is known. This depth is represented in the Level column of the recursive CTE query.

READER NOTE*: The next step will not run by itself and is there to show what this query looks like when it is partially done.*

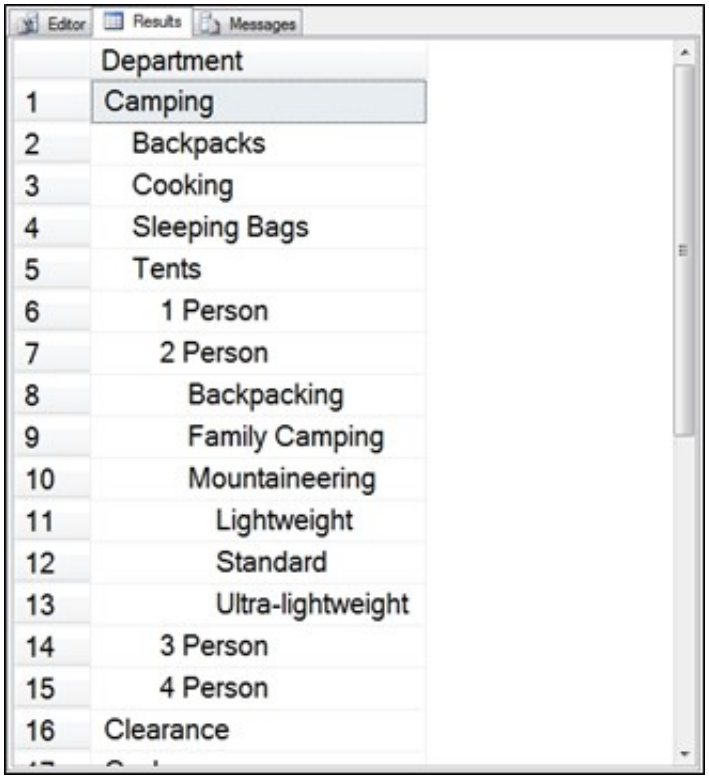
The way that the indentation will be built is to take the exact same tree path CTE query from the previous section, keeping the TreePath column formatted and sorted exactly as before. Note that there are no changes to the anchor or recursive part of the CTE. A new column will be added to the result set (called Department) which will replicate the indentation that is desired. The following example replicates four spaces for each level of the hierarchy.

```
SELECT REPLICATE('    ', Level)
    + Department as Department
```

Now drop that into the query that is calling the CTE.

```
;WITH departmentsCTE AS
(
  SELECT id, Department, parent, 0 as Level,
        cast(Department as varchar(8000)) as TreePath
    FROM Departments
   WHERE parent is NULL
   UNION ALL
  SELECT d.id, d.Department, d.Parent,
        DepartmentsCTE.Level + 1 as Level,
```

```
        cast(DepartmentCTE.TreePath
          + ' -> '
          + d.Department as varchar(8000)) as TreePath
    FROM Departments d
   INNER JOIN departmentsCTE ON DepartmentCTE.id = d.Parent
)
SELECT REPLICATE('    ', Level)
    + Department as Department
  FROM departmentsCTE
 ORDER BY TreePath;
```



	Department
1	Camping
2	Backpacks
3	Cooking
4	Sleeping Bags
5	Tents
6	1 Person
7	2 Person
8	Backpacking
9	Family Camping
10	Mountaineering
11	Lightweight
12	Standard
13	Ultra-lightweight
14	3 Person
15	4 Person
16	Clearance

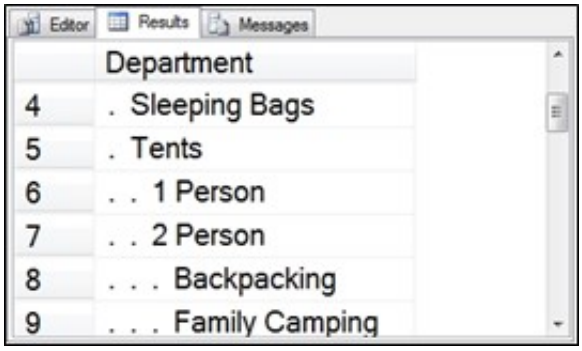
Figure 5.8 Tree path hierarchy shown with indentation.

In the previous example it is necessary to include the ORDER BY TreePath to get all the departments to show up in the right order with sub-departments being shown under their parents. Without this sorting the results would get the departments out of order, with no representation of

their place in the hierarchy.

The indentation can be altered for different effects with the following and many more options:

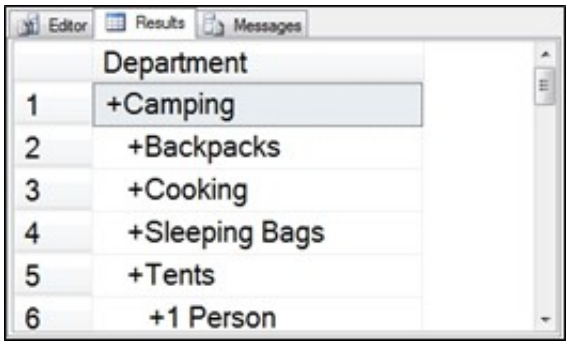
```
SELECT REPLICATE('.  ', Level)
    + Department as Department
```



	Department
4	. Sleeping Bags
5	. Tents
6	. . 1 Person
7	. . 2 Person
8	. . . Backpacking
9	. . . Family Camping

Figure 5.9 Each level is indented with a dot and 2 spaces.

```
SELECT REPLICATE('   ', Level) + '+'
    + Department as Department
```



	Department
1	+Camping
2	+Backpacks
3	+Cooking
4	+Sleeping Bags
5	+Tents
6	+1 Person

Figure 5.10 Each level is indented with 3 spaces and a plus.

With the REPLICATE function it is easy to customize the output from hierarchical queries to better represent the hierarchical data.

Hierarchy in Multiple Recursive Queries

The RoyalTreeCTE from the previous chapter was an interesting example of multiple recursive queries in the CTE, but it didn’t have a great visual representation of the data.

First we could apply the tree path method of formatting with the following query, which limits the results to only 3 levels of the hierarchy:

```
;WITH RoyalTreeCTE AS
(
  SELECT *, 0 as level,
        cast(name as varchar(4096)) as TreePath
    FROM Royalty WHERE id = 17
   UNION ALL
  SELECT r.*, level + 1 as level,
        cast(rt.TreePath + '-->father:' +
            r.name as varchar(4096)) as TreePath
    FROM Royalty r
   INNER JOIN RoyalTreeCTE rt on rt.father = r.id
   UNION ALL
  SELECT r.*, level + 1 as level,
        cast(rt.TreePath + '-->mother:' +
            r.name as varchar(4096)) as TreePath
    FROM Royalty r
   INNER JOIN RoyalTreeCTE rt on rt.mother = r.id
)
```

```
SELECT TreePath
  FROM RoyalTreeCTE
 WHERE level  ORDER BY TreePath;
```

Editor | Results | Messages

	TreePath
1	William Arthur Phillip Windsor
2	William Arthur Phillip Windsor-->father:Charles Philip Arthur Windsor
3	William Arthur Phillip Windsor-->father:Charles Philip Arthur Windsor-->father:Phillip Mountbatte
4	William Arthur Phillip Windsor-->father:Charles Philip Arthur Windsor-->mother:Elizabeth II Wind
5	William Arthur Phillip Windsor-->mother:Diana Frances (Lady) Spencer
6	William Arthur Phillip Windsor-->mother:Diana Frances (Lady) Spencer-->father:Edward John S
7	William Arthur Phillip Windsor-->mother:Diana Frances (Lady) Spencer-->mother:Frances Ruth

Figure 5.11 Tree path for the Royalty tree.

As shown in Figure 5.11 Williams parents are Charles and Diana. The parents of Charles are Phillip and Elizabeth II, and the parents of Diana are Edward and Frances. This is challenging to follow when there are additional levels to the tree.

The next option is to switch to indentation instead of tree path for this example. This can be done by changing the SELECT statement outside of the CTE as follows:

```
SELECT REPLICATE('.      ', Level) +
      Relationship + ': ' + name
```

In this example we still keep the ORDER BY TreePath to preserve the same order as before. Once we drop that into our query it will look like the following:

```
;WITH RoyalTreeCTE AS
(
   SELECT *, 0 as level,
         cast(name as varchar(4096)) as TreePath,
         cast('Root' as varchar(10)) as Relationship
     FROM Royalty WHERE id = 17
    UNION ALL
   SELECT r.*, level + 1 as level,
         cast(rt.TreePath + '-->father:' +
              r.name as varchar(4096)) as TreePath,
         cast('Father' as varchar(10)) as Relationship
     FROM Royalty r
    INNER JOIN RoyalTreeCTE rt on rt.father = r.id
    UNION ALL
   SELECT r.*, level + 1 as level,
         cast(rt.TreePath + '-->mother:' +
              r.name as varchar(4096)) as TreePath,
         cast('Mother' as varchar(10)) as Relationship
     FROM Royalty r
    INNER JOIN RoyalTreeCTE rt on rt.mother = r.id
)
SELECT REPLICATE('.      ', Level) +
      Relationship + ': ' + name
  FROM RoyalTreeCTE
 ORDER BY TreePath;
```

Editor | Results | Messages

	(No column name)
1	Root: William Arthur Phillip Windsor
2	. Father: Charles Philip Arthur Windsor
3	. . Father: Phillip Mountbatten, Duke of Edinburgh
4	. . . Father: Andreas, Prince of Greece
5	. . . . Father: William George I of the Hellenes
6	. . . . Mother: Olga Konstantinovna Romanova
7	. . . Mother: Alice, Princess of Battenbugr
8	. . . . Father: Louis Alexander von Battenburg
9	. . . . Mother: Victoria von Hessen und bei Rhein
10	. . Mother: Elizabeth II Windsor, Queene of England
11	. . . Father: George VI Windsor, King of England
12	. . . . Father: George V, King of England
13	. . . . Mother: Mary, Princess of Teck
14	. . . Mother: Elizabeth Angela Marguerite Bowes-Lyon
15	. . . . Father: Claude George Bowes-Lyon
16	. . . . Mother: Nina Cecilia Cavendish-Bentinck
17	. Mother: Diana Frances (Lady) Spencer
18	. . Father: Edward John Spencer
19	. . Mother: Frances Ruth Burke Roche

Figure 5.12 Improved formatting with indentation.

Looking at the overall hierarchy, it doesn’t look like the common family tree that has the individual in the middle, and the paternal and maternal on one side or the other, or the above and below split.

In order to improve the format, we simply need to improve the formatting. We have all the right people, just not in the right order.

To achieve the desired sorting, we change our format by adding another column to the anchor and recursive queries called Sorter, and we drop the tree path since we won’t be using that for sorting any more.

The sort algorithm is the following:

1. Assign a ‘b’ to the person at the root. Prince William in this case.
2. For any individuals’ father, add an ‘a’ to the sorter.
3. For any individuals’ mother, add a ‘c’ to the sorter.
4. When all sorting is complete, append a series of ‘b’s to the sorter so that shorter sort strings end up in the middle.

With this logic added to the code there are 4 lines changed from the previous CTE examples. The Sorter column was added to the anchor query setting the Sorter value to ‘b’, this covers item #1 in the sort algorithm. Next, a Sorter column is added to the recursive query for the father’s side, and it adds the value of ‘a’ to the Sorter column, meeting item #2 in the algorithm. Then the Sorter column is added to the mother’s recursive query with the value of ‘c’ added to the Sorter column, this meets item #3 in the algorithm. The final change is to add a series of ‘b’s to the Sorter column in the ORDER BY clause, which meets #4 in the algorithm. If the maternal side was needed to appear above instead of the paternal side, simply swap the ‘a’ and ‘c’ values for the Sorter in the mother and father recursive queries.

Sorting related changes is shown in bold in the following query:

```
;WITH RoyalTreeCTE AS
(
  SELECT *, 0 as level,
        cast('b' as varchar(4096)) as Sorter,
        cast('Top' as varchar(10)) as Relationship
    FROM Royalty WHERE id = 17
   UNION ALL
  SELECT r.*, level + 1 as level,
        cast(Sorter + 'a' as varchar(4096)) as Sorter,
        cast('Father' as varchar(10)) as Relationship
    FROM Royalty r
   INNER JOIN RoyalTreeCTE rt on rt.father = r.id
   UNION ALL
  SELECT r.*, level + 1 as level,
        cast(Sorter + 'c' as varchar(4096)) as Sorter,
        cast('Mother' as varchar(10)) as Relationship
    FROM Royalty r
```

```
    INNER JOIN RoyalTreeCTE rt on rt.mother = r.id
)
SELECT sorter, name
  FROM RoyalTreeCTE
 ORDER BY Sorter + 'bbbbbbbbbbbbb';
```

Editor | Results | Messages

	sorter	name
1	baaaa	William George I of the Hellenes
2	baaa	Andreas, Prince of Greece
3	baaac	Olga Konstantinovna Romanova
4	baa	Phillip Mountbatten, Duke of Edinburgh
5	baaca	Louis Alexander von Battenburg
6	baac	Alice, Princess of Battenbugr
7	baacc	Victoria von Hessen und bei Rhein
8	ba	Charles Philip Arthur Windsor
9	bacaa	George V, King of England
10	baca	George VI Windsor, King of England
11	bacac	Mary, Princess of Teck
12	bac	Elizabeth II Windsor, Queene of England
13	bacca	Claude George Bowes-Lyon
14	bacc	Elizabeth Angela Marguerite Bowes-Lyon
15	baccc	Nina Cecilia Cavendish-Bentinck
16	b	William Arthur Phillip Windsor
17	bca	Edward John Spencer
18	bc	Diana Frances (Lady) Spencer
19	bcc	Frances Ruth Burke Roche

Figure 5.13 The Royal family tree with William’s paternal ancestors in the list before him and his maternal ancestors after him.

At this point we have the right sort order, but the output still doesn’t look like any recognizable family tree layout. The next step is to use the REPLICATE function to do the indentation that we are looking for and to choose the right data.

```
SELECT REPLICATE('.           ', Level) +
      Relationship + ': ' + name
```

This gives a dot and eleven spaces of indentation for each level through the family tree. It includes the Relationship column of Top (for Prince William), Father or Mother for the rest, and then the person’s name.

```
;WITH RoyalTreeCTE AS
(
   SELECT *, 0 as level,
         cast('b' as varchar(4096)) as Sorter,
         cast('Top' as varchar(10)) as Relationship
     FROM Royalty WHERE id = 17
    UNION ALL
   SELECT r.*, level + 1 as level,
         cast(Sorter + 'a' as varchar(4096)) as Sorter,
         cast('Father' as varchar(10)) as Relationship
     FROM Royalty r
    INNER JOIN RoyalTreeCTE rt on rt.father = r.id
    UNION ALL
   SELECT r.*, level + 1 as level,
         cast(Sorter + 'c' as varchar(4096)) as Sorter,
         cast('Mother' as varchar(10)) as Relationship
     FROM Royalty r
    INNER JOIN RoyalTreeCTE rt on rt.mother = r.id
)
SELECT REPLICATE('.           ', Level) +
      Relationship + ': ' + name
  FROM RoyalTreeCTE
 ORDER BY Sorter + 'bbbbbbbbbbbbb';
```

	(No column name)
1	. . . . Father: William George I of the Hellenes
2	. . . Father: Andreas, Prince of Greece
3	. . . . Mother: Olga Konstantinovna Romanova
4	. . Father: Phillip Mountbatten, Duke of Edinburgh
5	. . . . Father: Louis Alexander von Battenburg
6	. . . Mother: Alice, Princess of Battenbugr
7	. . . . Mother: Victoria von Hessen und bei Rhein
8	. Father: Charles Philip Arthur Windsor
9	. . . . Father: George V, King of England
10	. . . Father: George VI Windsor, King of England
11	. . . . Mother: Mary, Princess of Teck
12	. . Mother: Elizabeth II Windsor, Queene of England
13	. . . . Father: Claude George Bowes-Lyon
14	. . . Mother: Elizabeth Angela Marguerite Bowes-Lyon
15	. . . . Mother: Nina Cecilia Cavendish-Bentinck
16	Top: William Arthur Phillip Windsor
17	. . Father: Edward John Spencer
18	. Mother: Diana Frances (Lady) Spencer
19	. . Mother: Frances Ruth Burke Roche

Figure 5.14 The Royal family tree for Prince William.

At this point the steps that we have taken to build the family tree are:

1. Build the hierarchy with a recursive CTE with multiple recursive members.
2. Use indentation to display the depth of recursion.
3. Use sorting to get the order that is desired.

The same logic could be applied to any set of hierarchical data to get similar results.

Summary

Hierarchical data can be built with any hierarchical representation as was shown in the previous chapter. Making the hierarchical data look good can sometimes be as important as finding it.

There are many choices on how to display hierarchical data. One is with a tree path where the current item is shown with a full path of its parents until the root of the tree is reached.

Editor | Results | Messages

	TreePath
9	Camping -> Tents -> 2 Person -> Family Camping
10	Camping -> Tents -> 2 Person -> Mountaineering
11	Camping -> Tents -> 2 Person -> Mountaineering -> Lightweight
12	Camping -> Tents -> 2 Person -> Mountaineering -> Standard
13	Camping -> Tents -> 2 Person -> Mountaineering -> Ultra-lightweight
14	Camping -> Tents -> 3 Person

Figure 5.15 Query results with just the tree path column, sorted by the tree path column.

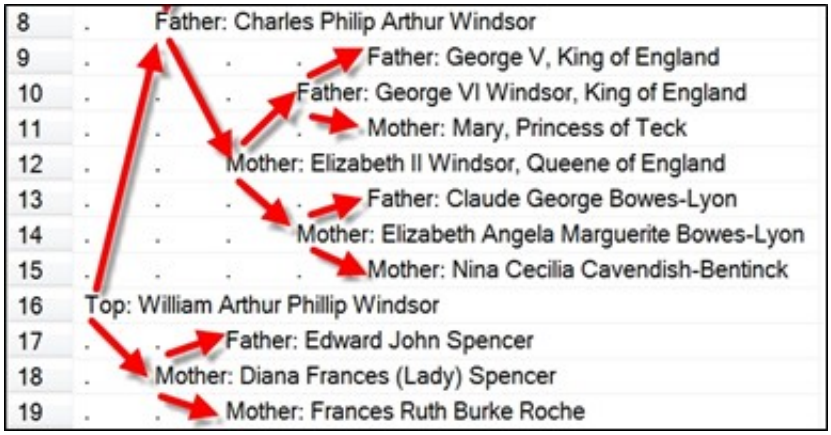
Another option besides tree path is the hierarchical display with indentation which can be achieved using the REPLICATE function with various indentation options.

Editor | Results | Messages

	Department
4	. Sleeping Bags
5	. Tents
6	. . 1 Person
7	. . 2 Person
8	. . . Backpacking
9	. . . Family Camping

Figure 5.16 Each level is indented with a dot and 2 spaces.

With some interesting sorting and indentation the family tree data can be shown in a format similar to a normal family tree layout. This was shown in the example of the Royal family tree example showing the ancestors of Prince William.



8	. Father: Charles Philip Arthur Windsor
9	. . . Father: George V, King of England
10	. . . Father: George VI Windsor, King of England
11	. . . Mother: Mary, Princess of Teck
12	. . Mother: Elizabeth II Windsor, Queene of England
13	. . . . Father: Claude George Bowes-Lyon
14	. . . Mother: Elizabeth Angela Marguerite Bowes-Lyon
15	. . . . Mother: Nina Cecilia Cavendish-Bentinck
16	Top: William Arthur Phillip Windsor
17	. . Father: Edward John Spencer
18	. Mother: Diana Frances (Lady) Spencer
19	. . Mother: Frances Ruth Burke Roche

Figure 5.17 The Royal family tree for Prince William.

The formatting of the CTE hierarchical output changes the results from just simple data to the visualization of data.

Points to Ponder - Hierarchical Queries

1. Recursive CTEs give the ability to access data, but hierarchical queries with the right formatting give the ability to visually understand the data.
2. Hierarchical queries can be easily built with a recursive CTE.

3. Tree path, also known as bread crumbs, can be easily created with a recursive CTE.
4. The REPLICATE function can be used to help indent hierarchical data based on the tree depth.
5. Hierarchical recursive CTEs perform better than a UNION ALL type non-CTE query that accomplishes similar results.

Review Quiz - Chapter Five

1. The ORDER BY clause must be used on hierarchical indentation CTEs in order for the formatting to make sense.
    - TRUE
    - FALSE
2. The REPLICATE function must be used with hierarchical CTEs in order to achieve indentation.
    - TRUE
    - FALSE
3. The following are common hierarchical formatting techniques, choose the two correct answers?
    - Tree branch
    - Tree path
    - Hierarchical indentation
    - Hierarchical consolidation

Answer Key

1. The ORDER BY clause must be used in a hierarchical CTE in order for the results to make sense. Without the sorting, the outline or indentation view won't place sub items in the right order. However, if using tree path instead of hierarchical indentation, the order isn't as important because the entire path is contained in each result. The correct answer is (a), TRUE.
2. The REPLICATE function is an excellent way to achieve hierarchical indentation but it is not

the only way. An alternative would be to build an indentation varchar column in the recursive member. The correct answer is (b), FALSE.

3. Tree path and hierarchical indentation are not the only options for hierarchical formatting, but they are the two most common. Both (b) and (c) are the correct answers.