Journal of Information Systems Education, Vol. 18(1)

Teaching Tip

Recursive Joins to Query Data Hierarchies in Microsoft Access

Mohammad Dadashzadeh
Decision and Information Sciences Department
Oakland University
Rochester, Michigan 48309, USA

ABSTRACT

Organizational charts (departments, sub-departments, sub-sub-departments, and so on), project work breakdown structures (tasks, subtasks, work packages, etc.), discussion forums (posting, response, response to response, etc.), family trees (parent, child, grandchild, etc.), manufacturing bill-of-material, product classifications, and document folder hierarchies are all examples of hierarchical data. Although relational databases can represent such hierarchical data with ease, relational query languages such as Structured Query Language (SQL) and Query-By-Example (QBE) fail to support users in formulating natural queries involving transitive closure of such hierarchical data (e.g., listing all descendants of an individual in a family tree scenario). This paper presents a simple approach for teaching users how to overcome this shortcoming and formulate the required recursive joins in order to query such data hierarchies in Microsoft Access.

Keywords: Database Management Systems, Hierarchical Data, SQL, Recursive Joins, Transitive Closure, Microsoft Access

1. INTRODUCTION

Hierarchical data occurs frequently in database development. Consider the following table representing departments, their employee count, and the department to which they report to (Steinbach, 2003):
DEPT(DNO, Name, EmployeeCount, ReportsToDNO)

A sample instance of the table DEPT is shown below:

DNO	Name	EmployeeCount	ReportsToDNO
9	Final Building	15	12
10	Pre Building	18	12
11	Q&A	4	19
12	Manufacturing	10	19
13	West	10	18
14	South	7	18
15	East	9	18
16	North	7	18
17	IT	3	20
18	Sales	4	20
19	Production	3	20
20	Samples & Co.	3	

The natural hierarchical structure defined by *ReportsTo* relationship is, of course, more noticeable in the following figures representing, respectively, the corresponding entity-relationship diagram and the associated organizational chart for the shown instance:

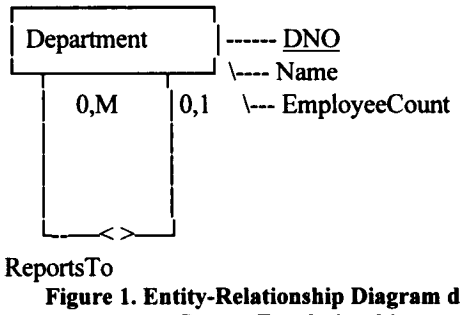


Figure 1. Entity-Relationship Diagram depicting *ReportsTo* relationship

Although relational databases can represent such hierarchical data with ease, relational query languages such as SQL and QBE do not support the most natural queries against data hierarchies. To fix ideas, consider the request to produce the output shown below (Exhibit 1) representing all departments reporting at one level or another to the "Production" department.

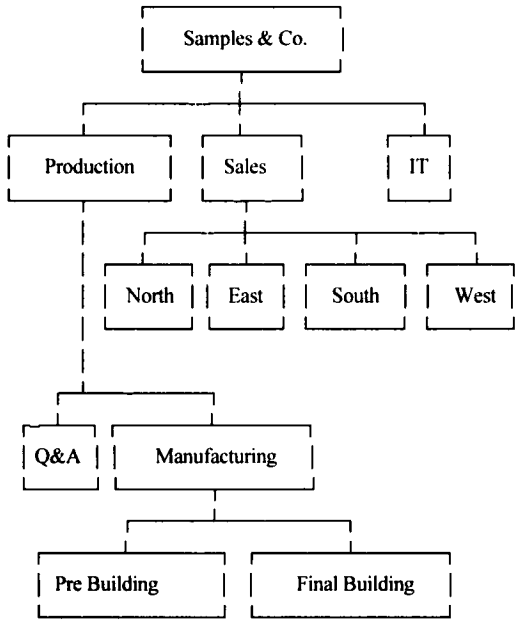


Exhibit 1. Organizational Chart

DNO	Name	Level
11	Q&A	1
12	Manufacturing	1
9	Final Building	2
10	Pre Building	2

Figure 2. *ReportsTo* data in the sample instance of DEPT table in organizational chart format

Basically, the output requires successive joins of the DEPT table with itself until no new rows can be added to the result, reflecting the completed traversal of the tree starting at the "Production" node. Standard SQL implementations such as in Microsoft Access do not support the required recursive joins. Oracle and DB2 have added proprietary extensions to their implementations of SQL to support such recursion. Specifically, the desired output is produced in Oracle (Loney and Koch, 2002) using the following query:

```
SELECT          DNO, level
FROM            DEPT
CONNECT BY PRIOR ReportsToDNO = DNO
START WITH      Name = "Production"
```

Oracle provides the built-in pseudo column *level* when using CONNECT BY PRIOR. However, the starting node is considered to be at level one and its immediate descendants are considered to be at level two.

IBM's DB2 (Melnyk and Zikopoulos, 2001) supports recursive queries in an implementation similar, but not identical, to that prescribed by the ANSI/ISO SQL-99 standard (Gulutzan and Pelzer, 1999). Specifically, the desired output is produced in DB2 using the following recursive query:

```
WITH Temp( DNO, ReportsToDNO, Level ) AS
((SELECT        root.DNO, root.ReportsToDNO, 1
FROM            DEPT root
WHERE           Name = "Production")

UNION ALL

(SELECT         child.DNO, child.ReportsToDNO,
                parent.level + 1
FROM            DEPT child, Temp parent
WHERE           child.ReportsToDNO = parent.DNO))

SELECT          *
FROM            Temp;
```

In this Teaching Tip, we present a solution for providing similar functionality in Microsoft Access. The solution is based on building the transitive closure of the relationship (see Figure 3 below) using embedded SQL programming.

ReportsTo

DNO	ReportsToDNO	Level
17	20	1
18	20	1
19	20	1
16	18	1
15	18	1
14	18	1
13	18	1
11	19	1
12	19	1
10	12	1
9	12	1
13	20	2
14	20	2
15	20	2
16	20	2
12	20	2
11	20	2
9	19	2
10	19	2
9	20	3
10	20	3

Figure 3. Transitive closure of *ReportsTo* relationship in the sample instance of DEPT table

2. COMPUTING TRANSITIVE CLOSURE IN MICROSOFT ACCESS

In mathematics, the transitive closure of a binary relation ***R*** on a set ***X*** is the smallest transitive relation on ***X*** that contains ***R*** (Wikipedia 2006). If ***X*** is the set of humans (alive or dead) and ***R*** is the relation "parent-of" so that ***xRy*** means that *x* is a parent of *y*, then the transitive closure of ***R*** is defined recursively as

the relation, ***S***, "ancestor-of," where *x**S**z* means that there exists a *y* such that *x**R**y* and *y**S**z*.

Using our example, the DEPT table defines the following relation:

*DNO **ImmediatelyReportsTo** ReportsToDNO*

In turn, the transitive closure of the relation can be defined (and computed) recursively as *A* ***ReportsTo*** *Z*, if there exists a row X in DEPT (such that X.DNO = *A* and X.ReportsToDNO = *B*) and a row Y in *ReportsTo* table (such that Y.DNO = *B* and Y.ReportsToDNO = *Z*).

The basic algorithm for computing a transitive closure (such as *ReportsTo*) using iteration (instead of recursion) can be sketched as follows:

```
/* Initialize the transitive closure table ReportsTo to contain all
reporting relationships at level 1 */

Let ReportsTo:=
        SELECT  DNO, ReportsToDNO
        FROM    DEPT
        WHERE   ReportsToDNO IS NOT NULL;

/* Initialize auxiliary table Temp to be the same as ReportsTo */

Let Temp:= ReportsTo;

Repeat

/* Compute all reporting relationships at the next level */
 Let Temp: =
        SELECT  DEPT.DNO, Temp,ResportsToDNO
        FROM    DEPT, Temp
        WHERE   DEPT.ReportsToDNO =
                Temp.DNO;

/* If Temp is not empty, then append it to ReportsTo */
 If |Temp| <> 0 Then
        Let ReportsTo:= ReportsTo Union Temp;

Until |Temp| = 0;
```

Stepping through the algorithm as implemented in Microsoft Access (see Appendix A), we observe the partial results shown in Figures 4-6 on the way to the final *ReportsTo* table given earlier in Figure 3.

With the transitive closure table (*ReportsTo*) produced, it is now possible to formulate queries against the data hierarchy with ease in both SQL and QBE. For example, the following Access SQL query will list all departments reporting at one level or another to the "Production" department:

```
SELECT ReportsTo.DNO, DEPT.Name,
        ReportsTo.Level
FROM    (ReportsTo INNER JOIN DEPT
        ON ReportsTo.DNO = DEPT.DNO)
        INNER JOIN DEPT AS DEPT_1
        ON ReportsTo.ReportsToDNO =
        DEPT_1.DNO
WHERE DEPT_1.Name = "Production"
```

Departments Reporting to Production

DNO	Name	Level
11	Q&A	1
12	Manufacturing	1
9	Final Building	2
10	Pre Building	2

Similarly, to determine the total number of employees for the "Production" department including all its sub-departments the following Access SQL query can be employed to produce the desired result:

Total Employee Count for Production Department

TotalEmployeeCount
50

```
SELECT Sum([Dept].[EmployeeCount]) +
        [Dept_1].[EmployeeCount]
        AS TotalEmployeeCount
FROM    DEPT, ReportsTo, [Dept_1]
WHERE DEPT.DNO = ReportsTo.DNO
        AND
        ReportsTo.ReportsToDNO = DEPT_1.DNO
        AND
        DEPT_1.Name = "Production"
GROUP BY        DEPT_1.DNO,
                DEPT_1.EmployeeCount
```

The solution presented above is based on computing the transitive closure and storing it in a separate table. Therefore, as the underlying data is updated, there is a need to re-compute the transitive closure. To avoid expensive (time consuming) re-computation, a more sophisticated incremental evaluation system can be designed (Pang et al., 2005).

3. CONCLUSIONS

Embedded SQL programming is a standard topic of coverage in the traditional course on database management systems in the IS curricula. With the expressive power of SQL and QBE, developing a motivating example of when embedded SQL programming becomes necessary is difficult. This is especially true in the context of retrieving data (via SELECT) as opposed to update/data entry processing. In this paper, we have presented an ideal opportunity to teach students the need for embedded SQL programming using the Microsoft Access VBA (Visual Basic for Applications) programming environment.

Computing the transitive closure of a relation is a necessary operation for formulating queries against data hierarchies that are often present in database design situations. Whether we are capturing course pre-requisite relationships or product/ subassembly bill-of-material relationships, the need for computing the transitive closure of such inherently recursive relationship types becomes quickly evident. The algorithm presented in this paper and implemented as re-usable VBA code in Appendix A provides a general solution to computing transitive closures, thereby making queries against data hierarchies a reasonably simple user undertaking in Microsoft Access.

DEPT

DNO	Name	EmployeeCount	ReportsToDNO
9	Final Building	15	12
10	Pre Building	18	12
11	Q&A	4	19
12	Manufacturing	10	19
13	West	10	18
14	South	7	18
15	East	9	18
16	North	7	18
17	IT	3	20
18	Sales	4	20
19	Production	3	20
20	Samples & Co.	3	

TEMP

DNO	ReportsToDNO	Level
17	20	1
18	20	1
19	20	1
16	18	1
15	18	1
14	18	1
13	18	1
11	19	1
12	19	1
10	12	1
9	12	1

ReportsTo

DNO	ReportsToDNO	Level
17	20	1
18	20	1
19	20	1
16	18	1
15	18	1
14	18	1
13	18	1
11	19	1
12	19	1
10	12	1
9	12	1
13	20	2
14	20	2
15	20	2
16	20	2
12	20	2
11	20	2
9	19	2
10	19	2

RESULT

DNO	ReportsToDNO	Level
13	20	2
14	20	2
15	20	2
16	20	2
12	20	2
11	20	2
9	19	2
10	19	2

Figure 4. Results from the Initial Iteration

DEPT

DNO	Name	EmployeeCount	ReportsToDNO
9	Final Building	15	12
10	Pre Building	18	12
11	Q&A	4	19
12	Manufacturing	10	19
13	West	10	18
14	South	7	18
15	East	9	18
16	North	7	18
17	IT	3	20
18	Sales	4	20
19	Production	3	20
20	Samples & Co.	3	

RESULT

DNO	ReportsToDNO	Level
9	20	3
10	20	3

TEMP

DNO	ReportsToDNO	Level
13	20	2
14	20	2
15	20	2
16	20	2
12	20	2
11	20	2
9	19	2
10	19	2

ReportsTo

DNO	ReportsToDNO	Level
17	20	1
18	20	1
19	20	1
16	18	1
15	18	1
14	18	1
13	18	1
11	19	1
12	19	1
10	12	1
9	12	1
13	20	2
14	20	2
15	20	2
16	20	2
12	20	2
11	20	2
9	19	2
10	19	2
9	20	3
10	20	3

Figure 5. Results from the Second Iteration

DEPT

DNO	Name	EmployeeCount	ReportsToDNO
9	Final Building	15	12
10	Pre Building	18	12
11	Q&A	4	19
12	Manufacturing	10	19
13	West	10	18
14	South	7	18
15	East	9	18
16	North	7	18
17	IT	3	20
18	Sales	4	20
19	Production	3	20
20	Samples & Co.	3	

RESULT

DNO	ReportsToDNO	Level

TEMP

DNO	ReportsToDNO	Level
9	20	3
10	20	3

ReportsTo

DNO	ReportsToDNO	Level
17	20	1
18	20	1
19	20	1
16	18	1
15	18	1
14	18	1
13	18	1
11	19	1
12	19	1
10	12	1
9	12	1
13	20	2
14	20	2
15	20	2
16	20	2
12	20	2
11	20	2
9	19	2
10	19	2
9	20	3
10	20	3

Figure 6. Results from the Third (final) Iteration

4. REFERENCES

Gulutzan, P. and Trudy, P. (1999) SQL-99 Complete, Really. CMP Books, Gilroy, CA.

Loney, K. and Koch, G. (2002) Oracle9i: The Complete Reference. McGraw-Hill, New York, NY.

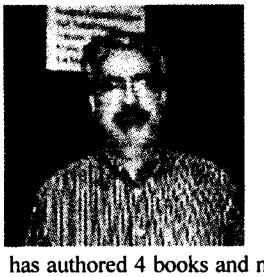
Melnyk, R. B. and Zikopoulos, P. C. (2001) DB2: The Complete Reference. McGraw-Hill, New York, NY.

Pang, C., Dong, G., and Ramamohanarao, K. (2005) "Incremental Maintenance of Shortest Distance and Transitive Closure in First-Order Logic and SQL," *ACM Transactions on Database Systems,* 30(3), pp. 698-721.

Steinbach, T. (2003) "Migrating Recursive SQL from Oracle to DB2 UDB," http://www-128.ibm.com/developerworks/db2/library/techarticle/0307steinbach/0307steinbach.html (7/4/2006).

Wikipedia (2006) "Transitive Closure," http://en.wikipedia.org/wiki/Transitive_closure (7/4/2006).

AUTHOR BIOGRAPHY



Mohammad Dadashzadeh has been affiliated with University of Detroit (1984-1989) and Wichita State University (1989-2003) where he served as the W. Frank Barton Endowed Chair in MIS. Since 2003, he has been serving as Professor of MIS and Director of the Applied Technology in Business (ATiB) Program at Oakland University. Dadashzadeh has authored 4 books and more than 40 articles on information systems and has served as the editor-in-chief of *Journal of Database Management.*

APPENDIX 1
Microsoft Access VBA Subroutine to Build *ReportsTo* Transitive Closure Table

The VBA code given in this appendix re-builds the *ReportsTo* transitive closure table from the DEPT(DNO, Name, EmployeeCount, ReportsToDNO) table. A generalized version with user-specifiable parameters identifying the input and output tables and the parent and child columns is available from the author.

```
Sub BuildReportsTo()
'This VBA subroutine re-builds the ReportsTo table.
DoCmd.SetWarnings False

'Empty ReportsTo table if it exists ...
If TableExists("ReportsTo") Then
 DoCmd.RunSQL ("DELETE FROM ReportsTo")
End If

'Initialize ReportsTo table with Level One ReportsTo data ...
If TableExists("ReportsTo") Then

 Let SQLcmd =
 "INSERT INTO " & _
 "ReportsTo(DNO, ReportsToDNO, [Level]) " & _
          "SELECT DNO, ReportsToDNO, 1 " & _
           "FROM DEPT " & _
          "WHERE ReportsToDNO IS NOT NULL"
Else
 Let SQLcmd =
          "SELECT DNO, " & _
          "ReportsToDNO, 1 As [Level] " & _
 "INTO ReportsTo " & _
 "FROM DEPT " & _
 "WHERE ReportsToDNO IS NOT NULL"
End If
DoCmd.RunSQL (SQLcmd)

'Initialize auxiliary table Temp to be the same as ReportsTo
...
DoCmd.CopyObject , "TEMP", acTable, "ReportsTo"

'Compute all reporting relationships at the next level ...
Let CurrentLevel = 2
Let Done = False

Do While Not Done
 Let SQLcmd =
          "SELECT DEPT.DNO, " & _
          "TEMP.ReportsToDNO, " & _
 Str(CurrentLevel) & " AS [Level] " & _
 "INTO RESULT " & _
          "FROM TEMP, DEPT " & _
          "WHERE DEPT.ReportsToDNO = TEMP.DNO"

 DoCmd.RunSQL (SQLcmd)
'If the Result of joining is not empty, then append all the
'reporting relationships found to ReportsTo and continue
'with Temp as the results found ...

 Set rst = CurrentDb.OpenRecordset("RESULT")

 If rst.RecordCount = 0 Then
 Let Done = True
 Else
 Let SQLcmd =
 "INSERT INTO " & _
 "ReportsTo(DNO, ReportsToDNO, [Level]) " & _
 "SELECT DNO, ReportsToDNO, [Level] " & _
 "FROM RESULT"

 DoCmd.RunSQL (SQLcmd)
 DoCmd.RunSQL ("DELETE FROM TEMP")
 Let SQLcmd =
          "INSERT INTO " & _
          "TEMP(DNO, ReportsToDNO, [Level]) " & _
          "SELECT DNO, ReportsToDNO, [Level] " & _
          "FROM RESULT"
 DoCmd.RunSQL (SQLcmd)
 End If

 rst.Close

 DoCmd.RunSQL ("DELETE FROM RESULT")
 Let CurrentLevel = CurrentLevel + 1

Loop

'Delete auxiliary tables produced ...
DoCmd.RunSQL ("DROP TABLE TEMP")
DoCmd.RunSQL ("DROP TABLE RESULT")

DoCmd.SetWarnings True
End Sub

Function TableExists(TableName As String) As Boolean
'Returns True if a table named TableName exists ...

Let ExistsFlag = False
On Error GoTo NotFound
   DoCmd.OpenTable TableName, acViewNormal,
   acReadOnly
Let ExistsFlag = True
DoCmd.Close

NotFound:
On Error GoTo 0 'Reset the error handler.

TableExists = ExistsFlag

End Function
```

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096