Hierarchical organization of information, in relational databases

Demian Horia

University of Oradea, Faculty of Economics, No 1, University Street , 410087, Bihor, Tel. 0259 408 796

email: horia_demian@yahoo.com

Abstract: In this paper I will present different types of representation, of hierarchical information inside a relational database .I also will compare them to find the best organization for specific scenarios.

Keywords: hierarchy, relational databases

In many situations it is necessary to organize information like a tree. The classical tree looks like the one presented in the figure below

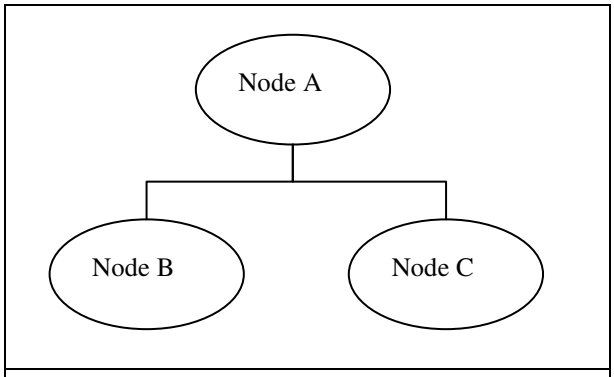


Fig. 1

To represent this information inside of relational databases we need, a table with the following structure:

Nod
idNod
idNodPrec
Nod

idNod	idNodPrec	Nod
1	1	Node A
2	1	Node B
3	1	Node C

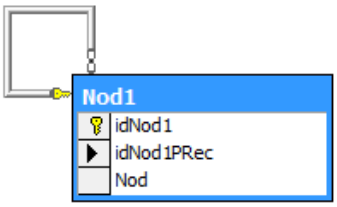
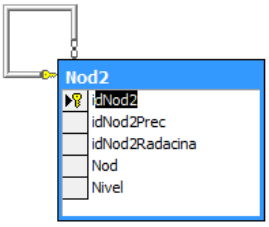
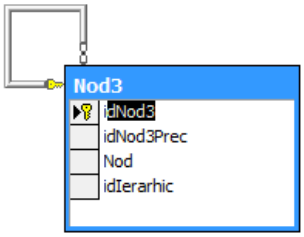
Fig. 2

In my studies I will try to find other methods for organization of hierarchical data, and I will compare them, to find the best solution. In practice there are a few very necessary operations. These operations were used in my tests.

For these tests I used three methods of representation:

1. Classical organization
2. Organization with reference to root node
3. Organization with hierarchical key

To better understand these types of organization I will present the table structure for each one

		
1. Classical organization	2. Organization with reference to root node	3. Organization with hierarchical key

Data looks like this:

1. Classical organization

idNod	IdNodPrec	Nod
1	1	Desktop
2	1	Horia
3	1	Public
4	1	Computer
5	4	Local Disk (C:)
...	...	...

Fig. 3 Organizarea tabelara a unui arbore

2. Organization with reference to root node

idNod	IdNodPrec	idNodRadacina	Nod	Level
1	1	1	Desktop	1
2	1	1	Horia	2
3	1	1	Public	2
4	1	1	Computer	2
5	4	1	Local Disk (C:)	3
....				

3. Organization with hierarchical key

idNod	**idNodPrec**	**Nod**	**idIerarhic**
1	1	Desktop	**1.**
2	1	Horia	**1.1.**
3	1	Public	**1.2.**
4	1	Computer	**1.3.**
5	4	Local Disk (C:)	**1.3.1.**

…	…	…	

One of the operations which I have tested was to obtain a sub tree beginning from a specific node.

1 In case of classical organization and organization with reference to root we need to create a function like this one

```
CREATE FUNCTION dbo.FN_NOD1_ARBORE ( @idNod1      INT)
        RETURNS @ARBOREDEST TABLE (
                                        IDNOD1          INT
                                        ,IDNOD1PREC     INT
                                )
AS
BEGIN
        DECLARE @ARBORENOD TABLE (
                                                IDNOD1          INT
                                                ,IDNOD1PREC     INT
                                        )

        DECLARE @AROREPREC TABLE (
                                                IDNOD1          INT
                                                ,IDNOD1PREC     INT
                                        )

        DECLARE @KOD                            SMALLINT
        SET @KOD        = 0
                --INSERAREA NODUrilor DE PLECARE(cele de ultim nivel pentru care exista produse publicate)
                INSERT INTO @ARBORENOD
                SELECT
                                N.IDNOD1
                                ,N.IDNOD1PREC
                        FROM
                                NOD1 N
                        WHERE N.IDNOD1 = @IDNOD1                        --nodul de la care pornim

                INSERT INTO @ARBOREDEST  --ACEASTA VA CONTINE DOAR parintii
                        SELECT
                                IDNOD1
                                ,IDNOD1PREC
                        FROM  @ARBORENOD
        WHILE (@KOD = 0 ) BEGIN
                --CAT TIMP MAI EXISTA NODURI CE FAC REFERINTA CATRE NODURI DIN PARINTE CONTINUAM PARCURGEREA ARBORELUI
                DELETE FROM @ARBOREPREC  --GOLIREA BUFFERULUI DE        noduri parinte
                INSERT INTO @ARBOREPREC  --ACEASTA VA CONTINE DOAR nodurile de nivel inferior ce nu au mai fost inserate
                        SELECT DISTINCT
                                A.IDNOD1
                                ,A.IDNOD1PREC
                        FROM NOD1       A
                                INNER JOIN @ARBORENOD AN
                                ON              A.IDNOD1PREC = AN.IDNOD1
                                        AND     A.IDNOD1PREC<>A.IDNOD1 --NU FACE REFERINTA SPRE EL
```

```
                IF @@ERROR<>0 OR @@ROWCOUNT=0 BEGIN
                        SET @KOD = 1  --NU MAI SUNT COPII
                END
                INSERT INTO @ARBOREDEST
                        SELECT
                                IDNOD1
                                ,IDNOD1PREC
                        FROM @ARBOREPREC
                --GOLIREA BUFFERULUI DE NODURI
                DELETE FROM @ARBORENOD
                --COPIEREA COPIILOR IN NOD
                INSERT INTO @ARBORENOD
                        SELECT
                                IDNOD1
                                ,IDNOD1PREC
                        FROM @ARBOREPREC
        END
        RETURN
END
```

3. In case of organization with hierarchical key we can use a simple SELECT command

```
Select  A.*
        From Nod3 A
        Where A.idIerarhic LIKE'1.3.%'
```

	1. Classical organization	2. Organization with reference to root node	3. Organization with hierarchical key
Space used for data	200 840 KB	243 272 KB	302 432 KB
Space used for indexes	59 288 KB	176 696 KB	208 064 KB
Number of records	5 380 845	5 380 845	5 380 845
Time in which we can obtain the records for the seven level	531 441 records in 41 seconds	531 441 records in 3 seconds	531 441 records in 11 seconds
Obtaining the child nodes for a specific node	9 records In 0 seconds	9 records In 0 seconds	9 records In 0 seconds
Obtaining a subtree order by level	66 430 records in 1 seconds	66 430 records in 4 seconds	66 430 records in 2 seconds
Obtaining the leaf nodes	4 782 972 records in 24 seconds	4 782 972 records in 29 seconds	4 782 972 records in 99 seconds
Obtaining the leaf nodes of a	59 049 records in 4 seconds	59 049 records in 2 seconds	59 049 records in 3 seconds

specific subtree			

For these tests I used SQL Server 2000 Desktop Engine, installed on a computer with Intel Core 2 Duo processor at 2GHZ, with 2GB of Ram.

After I obtain this results I decide to compare them, using global utility method.

	Criteriu	Criteriu	Criteriu	Criteriu	Criteriu	Criteriu	Criteriu	Criteriu	R
Coeficienti	1	1	1	1	1	1	1	1	
Minim/Maxim	Minim	Minim	Minim	Minim	Minim	Minim	Minim	Minim	
Variante\Criterii	spatiu ocupat	spatiu ocupat	obtinerea nod	obtinerea cop	obtinerea unu	Noduri frunza	Noduri frunza	overload de pı	
Arbore1	0	0	41	0	1	24	4	1	3.00
Arbore2	0	0	3	0	4	29	2	2	3.60
Arbore3	0	0	11	0	2	99	3	4	1.95

Situation 1.

	Criteriu	Criteriu	Criteriu	Criteriu	Criteriu	Criteriu	Criteriu	Criteriu	Re
Coeficienti	1	1	1	1	1	1	1	1	
Minim/Maxim	Minim	Minim	Minim	Minim	Minim	Minim	Minim	Minim	
Variante\Criterii	spatiu ocupat	spatiu ocupat	obtinerea nod	obtinerea cop	obtinerea unu	Noduri frunza	Noduri frunza	overload de pı	
Arbore1	200840	59288	41	0	1	24	4	1	5.000
Arbore2	243272	176696	3	0	4	29	2	2	4.393
Arbore3	302432	208064	11	0	2	99	3	4	1.956

Situation 2.

Observations:

1. When the space used by data is not important, the second method of organization hierarchical data was the best choice.
2. If space used for data need to be very small, the classical organization of hierarchical data was the best choice.
3. The third method gave us very good performance in situation in which we need to obtain a sub tree.
4. In each of this situation we obtain a very long time when we are trying to obtain the leaf nodes.

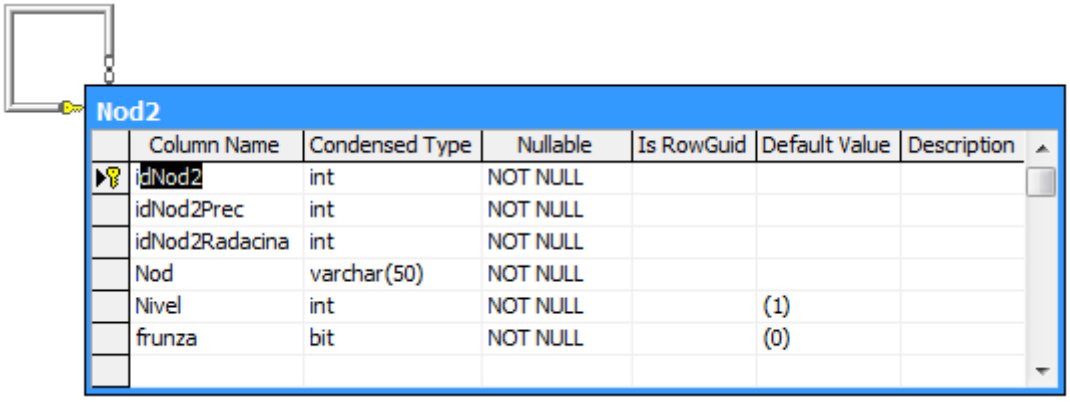
To improve performance I decided to try to reduce the time in which we obtain the leaf nodes. To do this we had to modify the structures of this table and to have additional information in these. I

	1. Classical organization	2. Organization with reference to root node	3. Organization with hierarchical key
Space used	217 944 KB	251 584 KB	307 152 KB
Space used by indexes	59 888 KB	176 712 KB	207 672 KB
Number of records	5 380 845	5 380 845	5 380 845
Level 7 nodes	531 441 records in 9 seconds	531 441 records in 2 seconds	531 441 records in 10 seconds
Command tested	select * from	select IDNOD2,IDNOD2PREC from nod2	SELECT IDNOD3,IDNOD3PREC

	dbo.fn_nod1_Arbore_nivel(10,7)	where idNOD2RADACINA = 10 AND NIVEL = 7	FROM nod3 WHERE idierarhic like '10.%.%.%.%.%.%.' and not idierarhic like'%.%.%.%.%.%.%.%.'
Obtaining child nodes	9 records in 0 seconds	9 records in 0 seconds	9 records in 0 seconds
Command tested	select * from nod1 where idnod1prec=66440	select * from nod2 where idnod2prec=66440	select * from nod1 where idnod1prec=66440
Obtaining a subtree order by level	66 430 records in 1 seconds	66 430 records in 1 seconds	66 430 records in 2 seconds
Command tested	select * from dbo.fn_Nod1_Arbore(21)	select * from dbo.fn_Nod2_Arbore(21)	select idnod3,idnod3prec from nod3 where idierarhic like '10.1.2.%'
Obtaining leaf nodes	4 782 972 records in 26 seconds	4 782 972 records in 28 seconds	4 782 972 records in 33 seconds
Command tested	select * from nod1 where frunza=1	select * from nod2 where frunza=1	select * from nod3 where frunza=1
Obtaining leaf nodes for a subtree	59 049 records in 2 seconds	59 049 records in 2 seconds	59 049 records in 2 seconds
Command tested	select * from dbo.fn_nod1_arbore(21) n inner join nod1 n1 on n.idnod1 = n1.idnod1 and frunza = 1	select * from dbo.fn_nod2_arbore(21) n inner join nod2 n1 on n.idnod2 = n1.idnod2 and frunza = 1	select * from nod3 n3 where n3.idierarhic like '10.1.2.%' and n3.frunza = 1

Conclusion:

The best result which I obtain was when the structure of table look like this one:



Bibliography:

1. Mihut, I. coord. –„Management”, „Babes-Bolyai” University, Cluj-Napoca, „1 Decembrie” – Alba Iulia University, 1998
2. Horia Demian – „Baze de date”, University of Oradea Publishing House, 2001.