Trees and Hierarchies in SQL

by Eduard Hildebrandt

In most projects we have to deal with some kind of trees or hierarchies!

Think about…

- Categories
- Organisation
- Threads
- Folders
- Components
- …

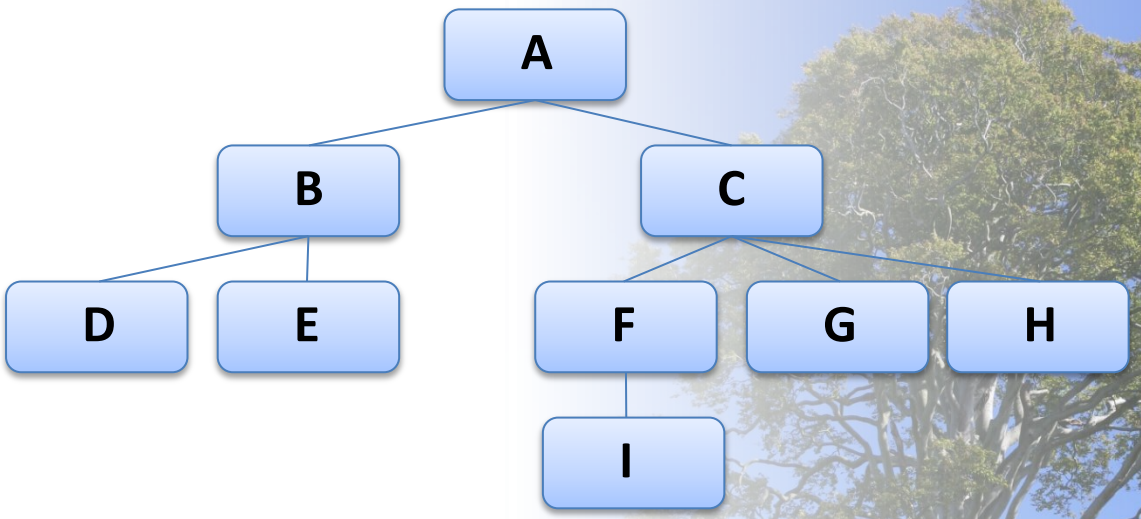
How can we store an retrieve trees and hierarchies effenciently from relational databases?

Agenda

- Adjacency List Model
- Closure Table Model
- Path Enumeration Model
- David Chandler Model
- Modified David Chandler Model
- Nested Set Model
- Hyprid Model
- Frequent Insertion in Nested Sets

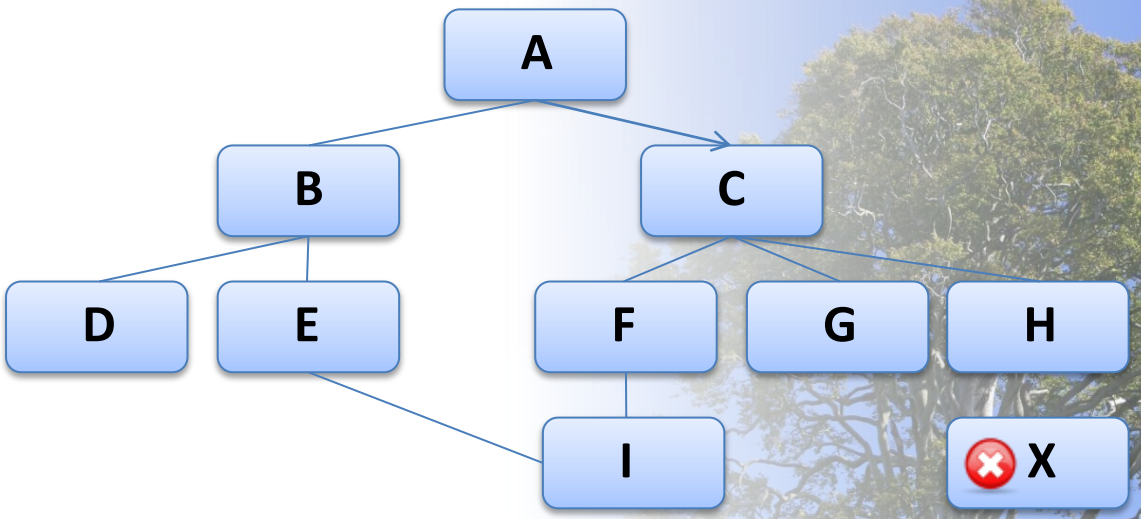
What is a „tree“?

A tree is a connected, undirected, acyclic grap.



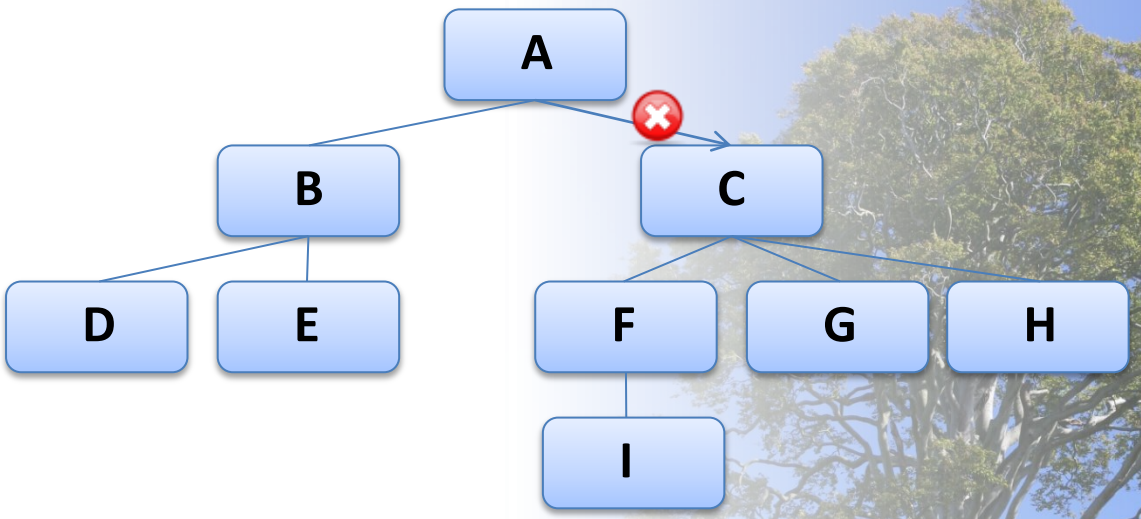
What is a „tree“?

A tree is a **connected**, undirected, acyclic grap.



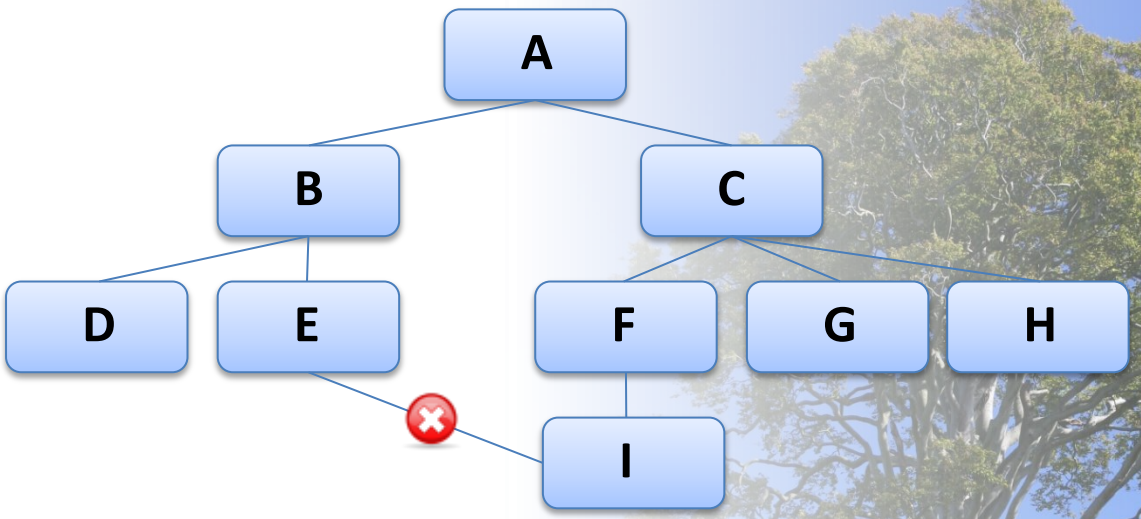
What is a „tree“?

A tree is a connected, **undirected**, acyclic grap.

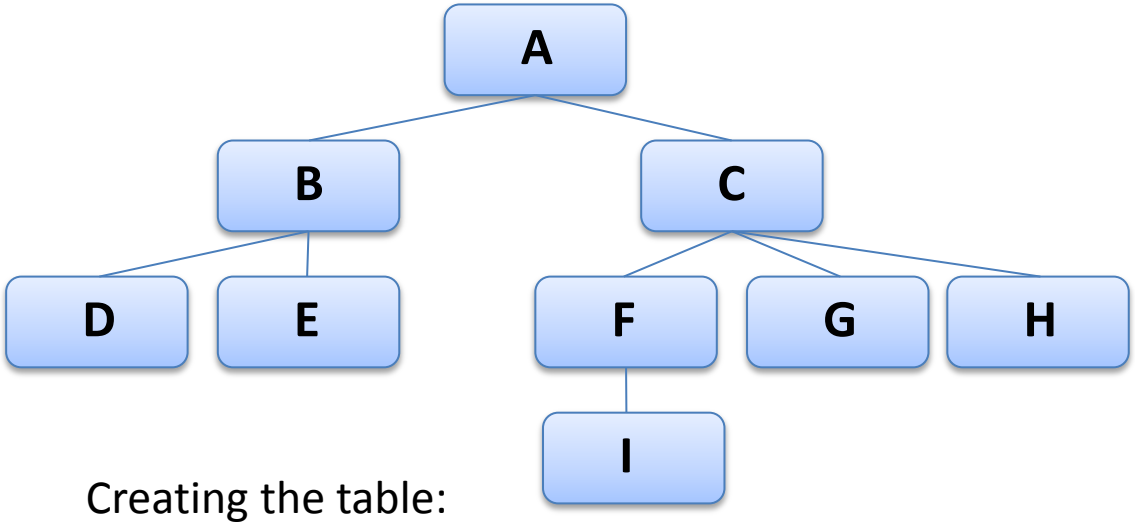


What is a „tree“?

A tree is a connected, undirected, **acyclic grap**.



Adjacency List Model



title	parent
A	null
B	A
C	A
D	B
E	B
F	C
G	C
H	C
I	F

Creating the table:

```
CREATE TABLE nodes
    (title VARCHAR(255) NOT NULL PRIMARY KEY,
    parent VARCHAR(255));
```

Finding the root node:

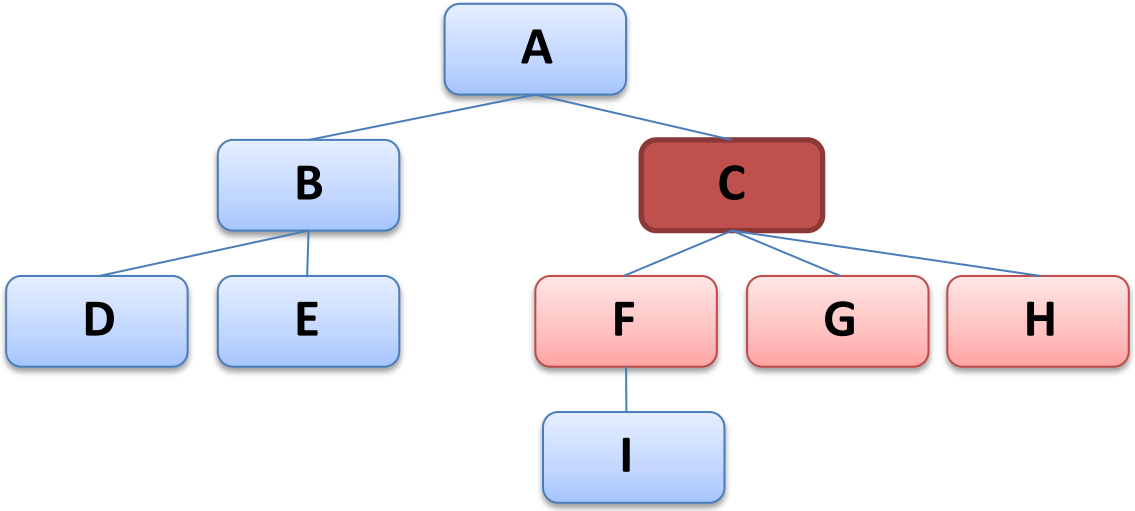
```
SELECT *
    FROM nodes
    WHERE parent IS NULL;
```

Finding the leaf nodes:

```
SELECT node1.*
    FROM nodes AS node1
    LEFT JOIN node AS node2 ON node1.title = node2.parent
    WHERE node2.title = 'A';
```

Adjacency List Model

UPDATE anomalies



title	parent
A	null
B	A
C	A
D	B
E	B
F	C
G	C
H	C
I	F

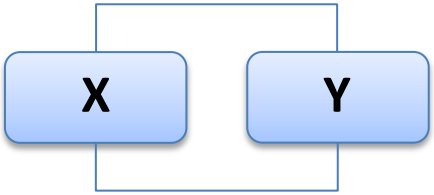
```
UPDATE nodes
   SET title = 'X'
   WHERE title = 'C';
```

```
START TRANSACTION;
UPDATE nodes
   SET title = 'X'
   WHERE title = 'C';
UPDATE nodes
   SET parent = 'X'
   WHERE parent = 'C';
COMMIT;
```

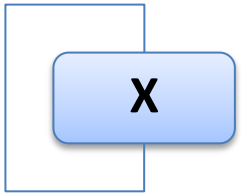
Adjacency List Model

INSERT anomalies

```
INSERT INTO nodes (title, parent) VALUES
 ('X', 'Y'),
 ('Y', 'X');
```




```
INSERT INTO nodes (title, parent) VALUES
 ('X', 'X');
```



Parent must exist:

```
ALTER TABLE nodes ADD CONSTRAINT parent_fk FOREIGN KEY (parent)
      REFERENCES nodes (id);
```

Node can not be it's own parent:

```
CHECK (title <> parent)
```

Prevent double connections:

```
UNIQUE (title, parent)
```

Connected graph (edges = nodes – 1):

```
CHECK ((SELECT COUNT(*) FROM nodes) - 1
        = SELECT COUNT(parent) FROM nodes;
```

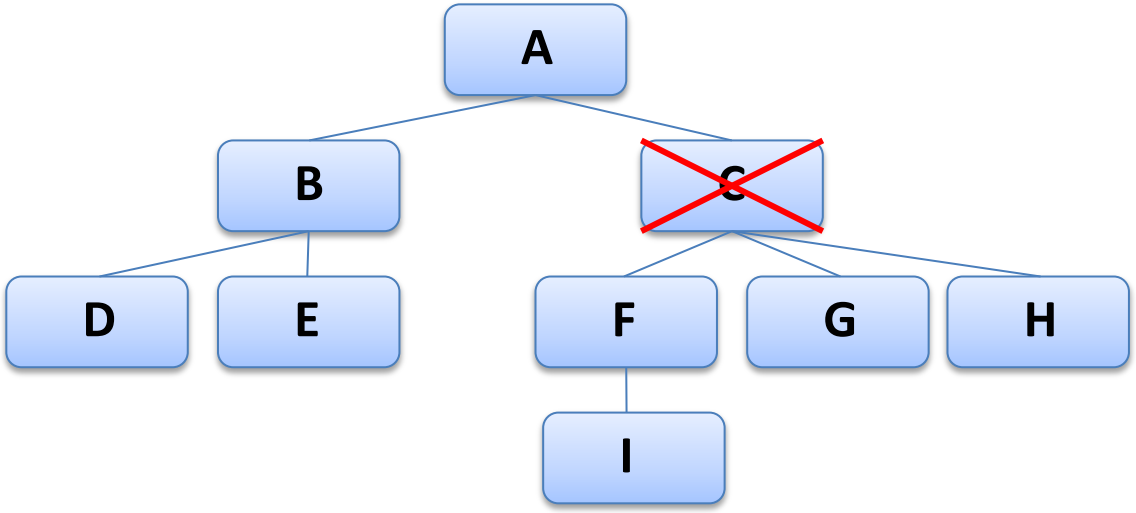
One root only:

```
CHECK ((SELECT COUNT(*) FROM nodes WHERE parent IS NULL) = 1);
```

Adjacency List Model

DELETE anomalies

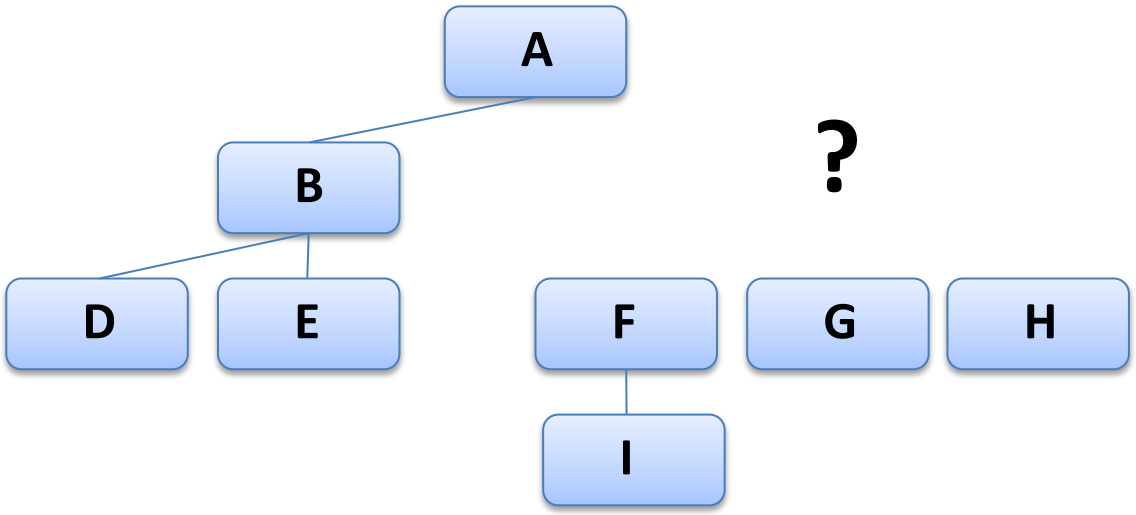
```
DELETE FROM nodes WHERE title = 'C';
```



Adjacency List Model

DELETE anomalies

What happens with nodes F, G, H and I?

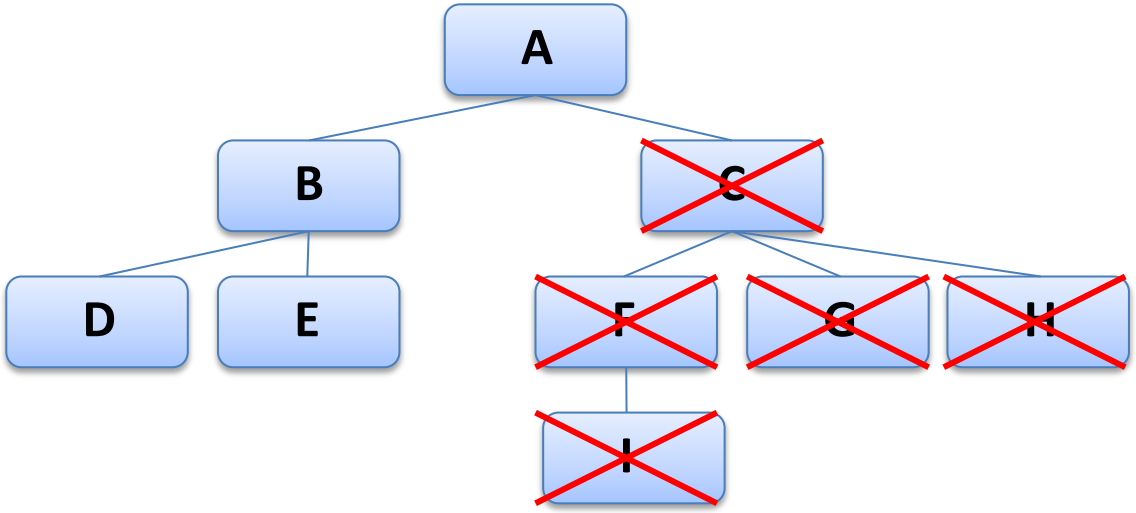



```
ALTER TABLE nodes ADD CONSTRAINT parent_fk FOREIGN KEY (parent)
      REFERENCES nodes (id)
      ON DELETE ...;
```

Adjacency List Model

DELETE anomalies

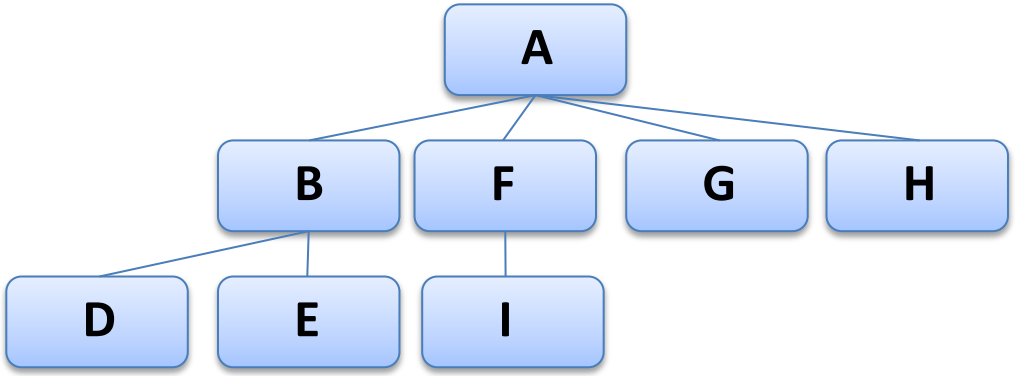
Solution 1: delete all children



Adjacency List Model

DELETE anomalies

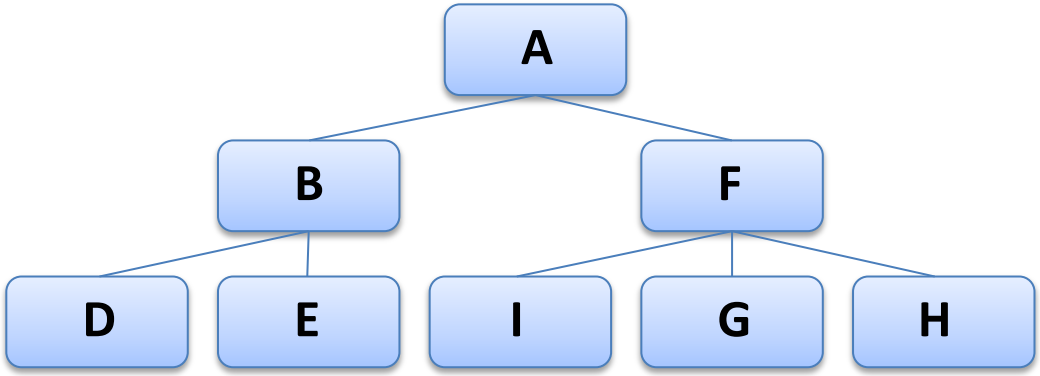
Solution 2: move children to the level of the parent



Adjacency List Model

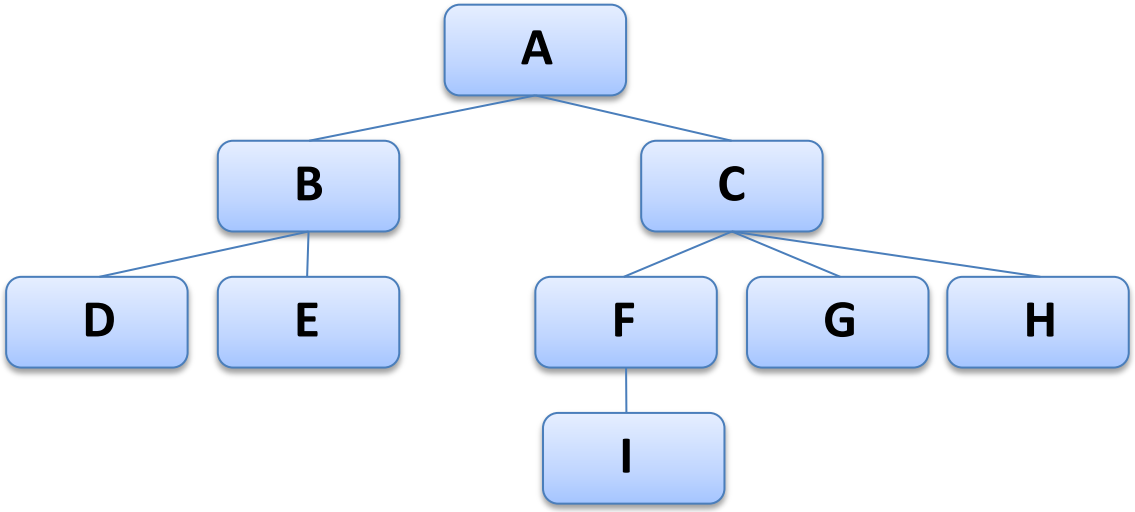
DELETE anomalies

Solution 3: replace parent with the first child



Adjacency List Model

with self -references



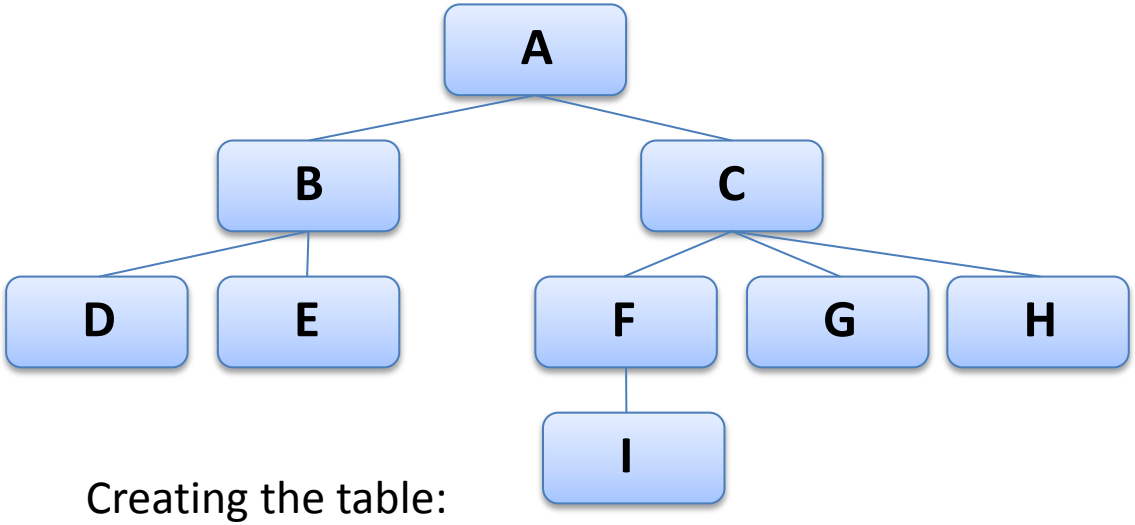
title	parent
A	A
B	A
C	A
D	B
E	B
F	C
G	C
H	C
I	F

Creating the table:

```
CREATE TABLE nodes
    (title VARCHAR(255) NOT NULL PRIMARY KEY,
    parent VARCHAR(255) NOT NULL);
```

Avoids NULL in the parent column!

Path Enumeration Model

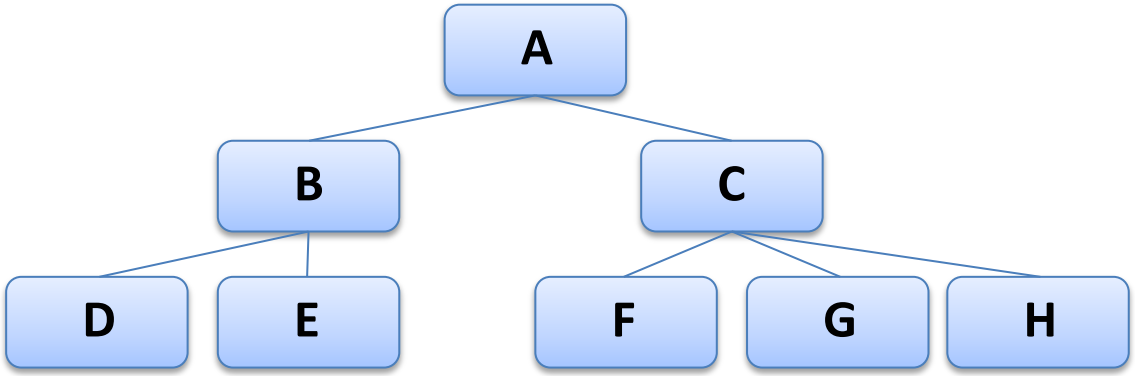


title	path
A	A
B	A/B
C	A/C
D	A/B/D
E	A/B/E
F	A/C/F
G	A/C/G
H	A/C/H
I	A/C/F/I

Creating the table:

```
CREATE TABLE nodes
    (title VARCHAR(255) NOT NULL PRIMARY KEY
     -- don't use a separater in primary key
     CHECK (REPLACE (title, '/', '') = title,
    path VARCHAR(1024) NOT NULL);
```

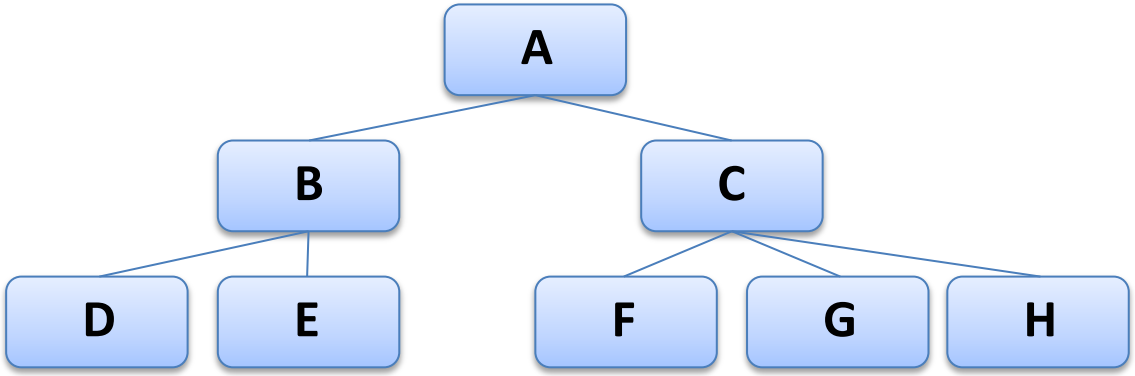
Closure Table Model



ID	parent
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H

Ancestor	Descendant
1	2
1	4
1	5
2	4
2	5
1	3
1	6
1	7
1	8
3	6

David Chandler Model



ID	T	L1	L2	L3
1	A	1	0	0
2	B	1	1	0
3	C	1	2	0
4	D	1	1	1
5	E	1	1	2
6	F	1	2	1
7	G	1	2	2
8	H	1	2	3

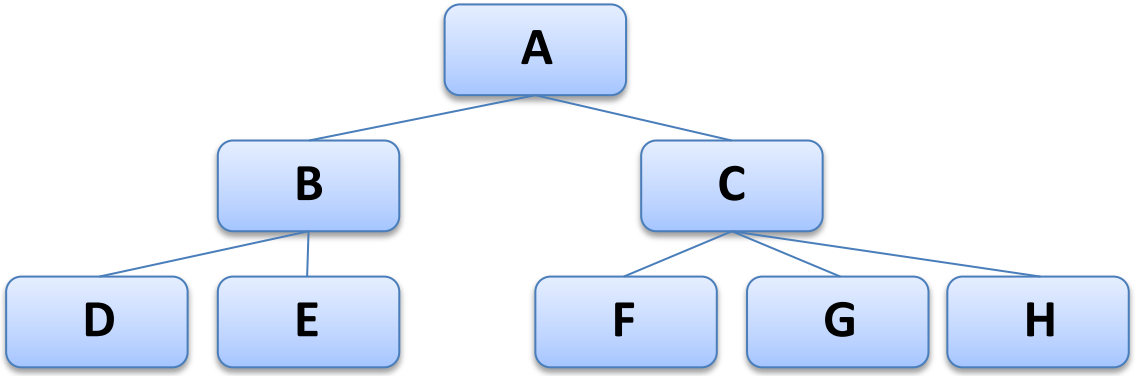
Creating the table:

```
CREATE TABLE nodes
    (id INTEGER NOT NULL PRIMARY KEY
     title VARCHAR(255) NOT NULL
     L1 INTEGER NOT NULL,
     L2 INTEGER NOT NULL,
     L3 INTEGER NOT NULL);
```

Algorithm is patened by David Chandler!

Hierarchy level is fixed!

Modified David Chandler Model



ID	T	L1	L2	L3
1	A	1	0	0
2	B	1	2	0
3	C	1	3	0
4	D	1	2	4
5	E	1	2	5
6	F	1	3	6
7	G	1	3	7
8	H	1	3	8

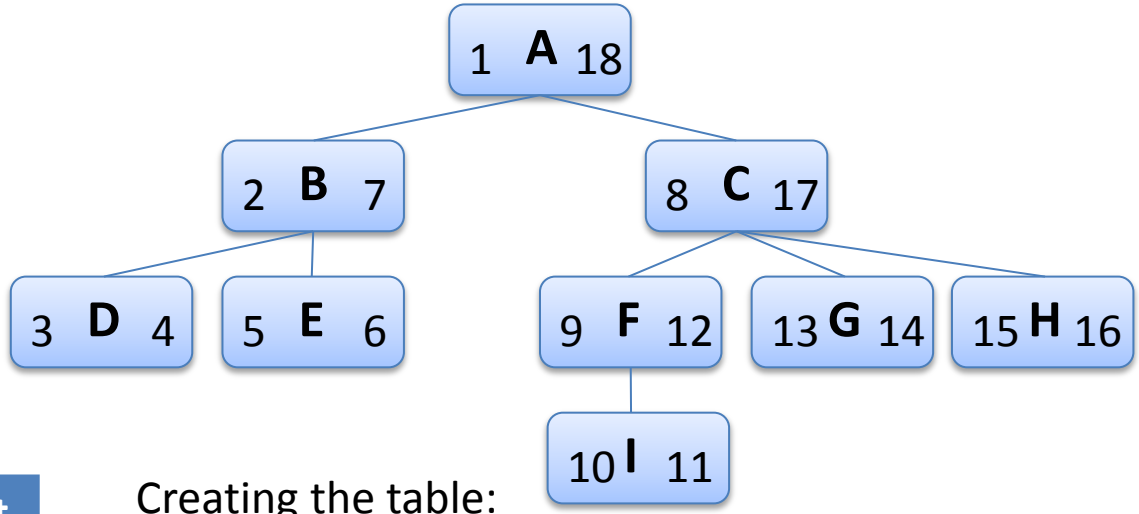
Creating the table:

```
CREATE TABLE nodes
    (id INTEGER NOT NULL PRIMARY KEY
     title VARCHAR(255) NOT NULL
     L1 INTEGER NOT NULL,
     L2 INTEGER NOT NULL,
     L3 INTEGER NOT NULL);
```

Not sure about implecations of David Chandlers patent…

Hierarchy level is fixed!

Nested Set Model



title	lft	rgt
A	1	18
B	2	7
C	8	17
D	3	4
E	5	6
F	9	12
G	13	14
H	15	16
I	10	11

Creating the table:

```
CREATE TABLE nodes
    (title VARCHAR(255) NOT NULL PRIMARY KEY,
    lft INTEGER NOT NULL,
    rgt INTEGER NOT NULL);
```

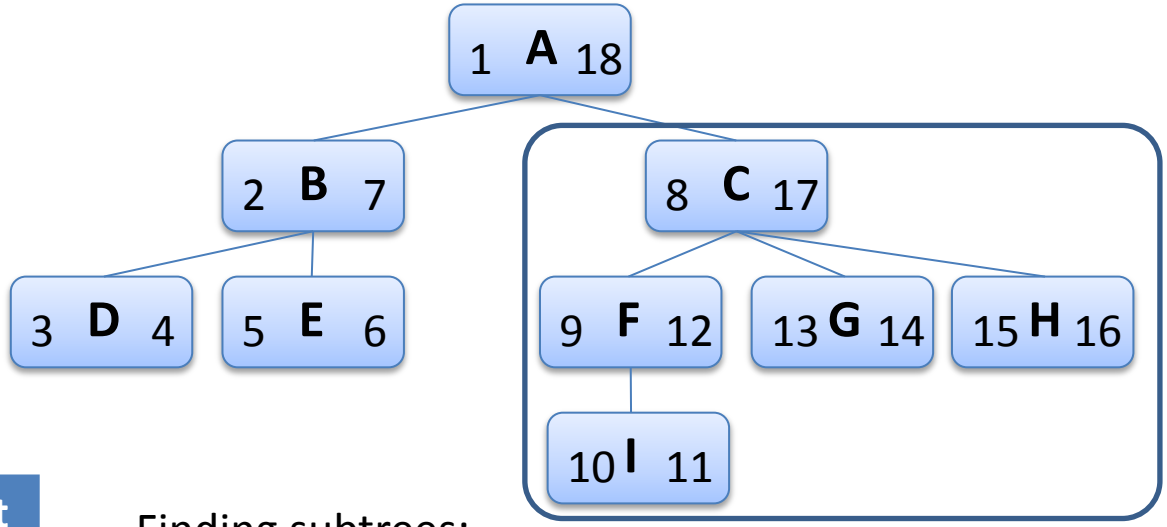
Finding the root node:

```
SELECT *
    FROM nodes
    WHERE lft = 1;
```

Finding the leaf nodes:

```
SELECT *
    FROM nodes
    WHERE lft = (MAX(rgt) - 1)
```

Nested Set Model



title	lft	rgt
A	1	18
B	2	7
C	8	17
D	3	4
E	5	6
F	9	12
G	13	14
H	15	16
I	10	11

Finding subtrees:

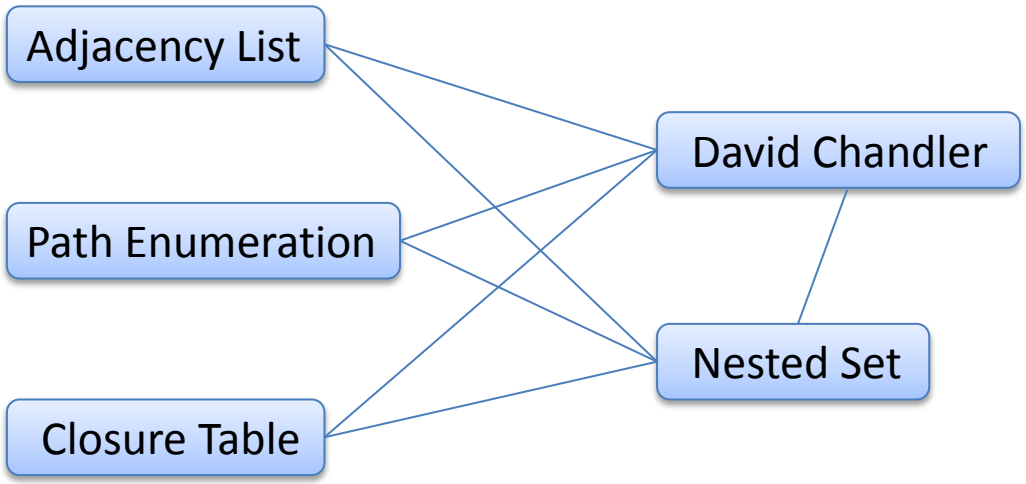
```
SELECT * FROM nodes
   WHERE lft BETWEEN lft AND rgt
   ORDER BY lft ASC;
```

Finding path to a node:

```
SELECT * FROM nodes
    WHERE lft < ? AND rgt > ?
    ORDER BY lft ASC;
```

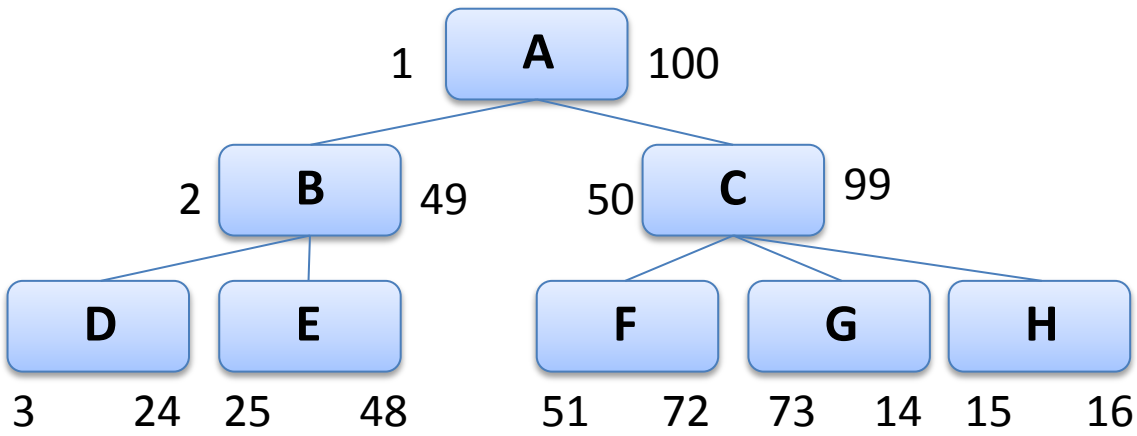
Hybrid Models

Combining models



Nested Set Model

Problem: Tree must be reorganized on every insertion (table lock).



Possible solution: Avoid overhead with larger spreads and bigger gaps

Nested Set Model

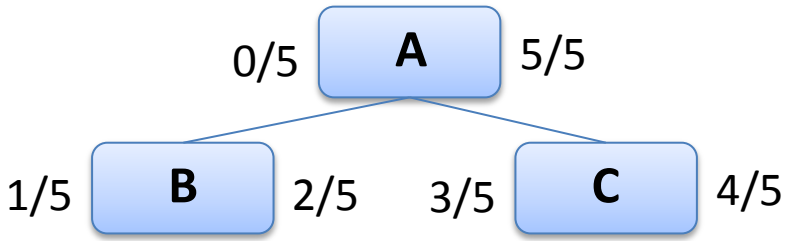
What datatype shout I use for left and right values?

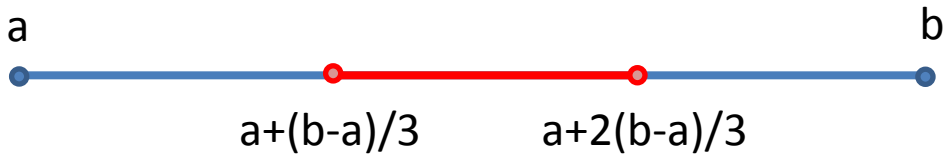
INTEGER

FLOAT / REAL / DOUBLE

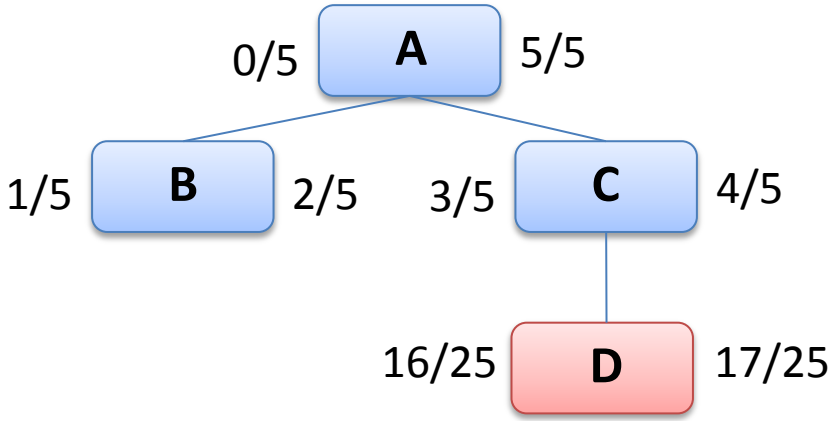
NUMERIC / DECIMAL

Nested Set Model
with rational numbers





Nested Set Model
with rational numbers



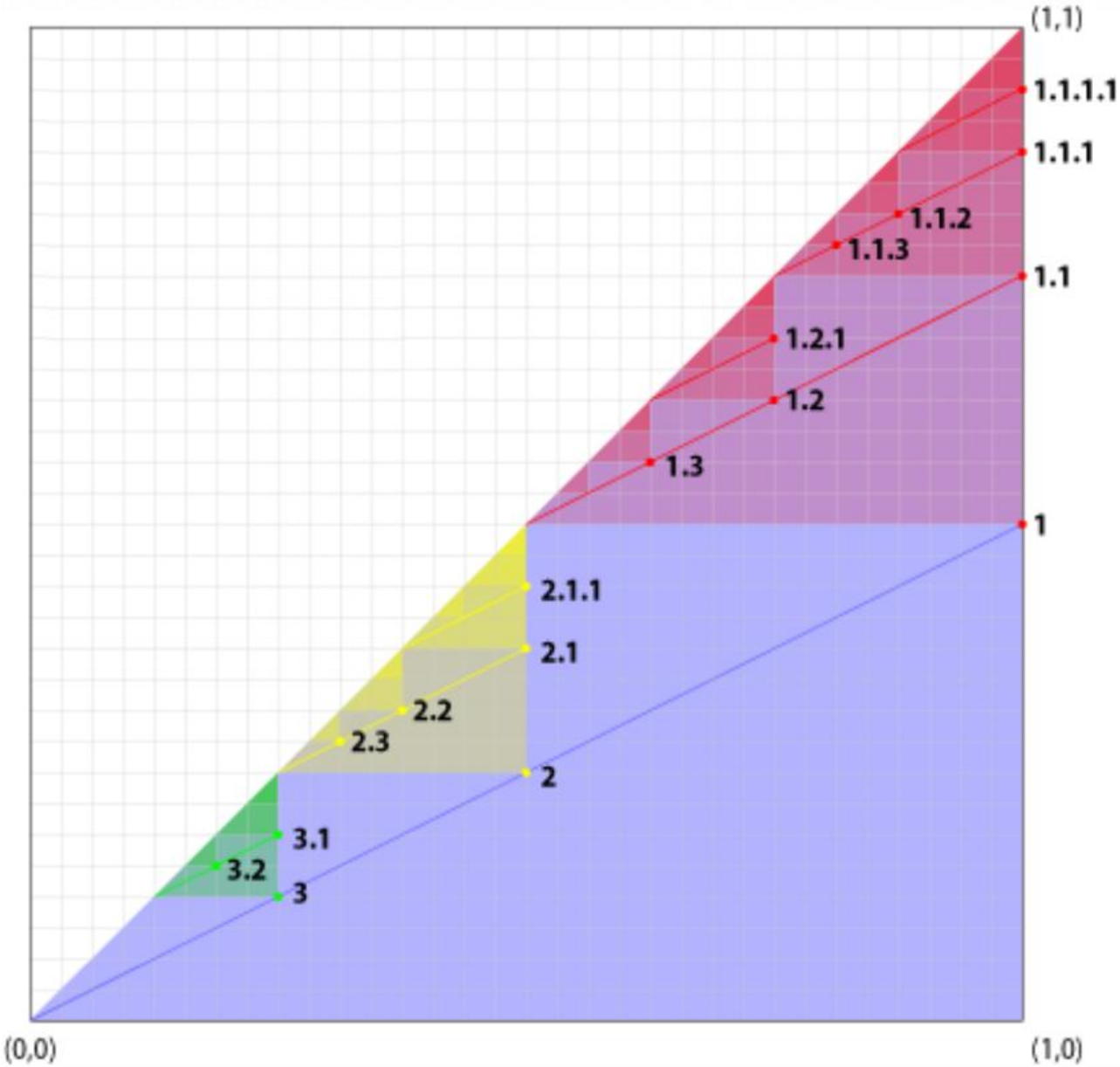
T	ln	ld	rn	rd
A	0	5	5	5
B	1	5	2	5
C	3	5	4	5
D	16	25	17	25

Adding an aditional child node is alway possible.
No need to reorganize the table!

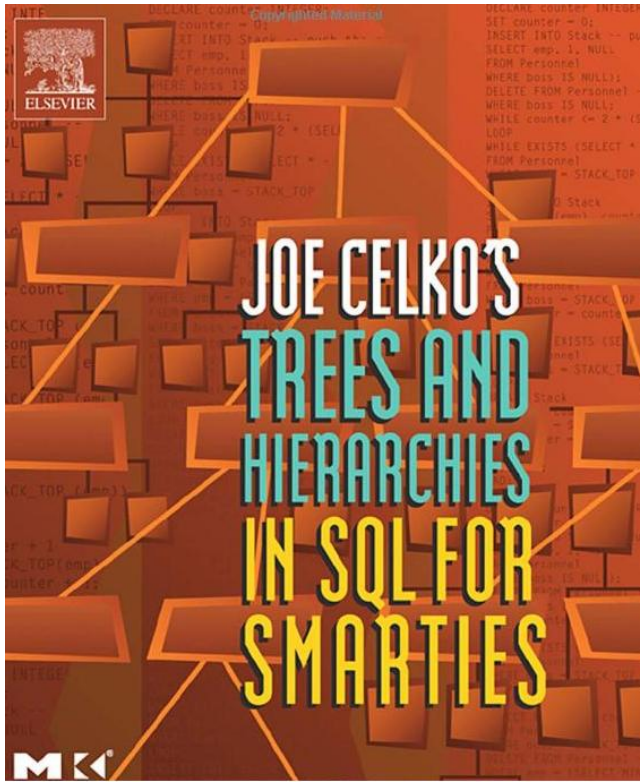
Nested Set Model

Binary Fraction

node	X	Y
1	1	1/2
1.1	1	3/4
1.1.1	1	7/8
1.1.1.1	1	15/16
1.1.2	7/8	13/16
1.1.3	13/16	25/32
1.2	3/4	5/8
1.2.1	3/4	11/16
1.3	5/8	9/16



Literature



Titel: Trees and Hierarchies in SQL
Autor: Joe Celko
Verlag: Morgan Kaufmann
ISBN: 1558609202

Q&A


THANK YOU!