Postgres PROFESSIONAL

May the Force of hierarchical data be with you

Teodor Sigaev, Oleg Bartunov PGConf.EU, 2019

Our projects in Postgres

Hierarchical data

Example: Web-site about astronomy

```
                       TOP
                     /   |  \
             Science Hobbies Collections
                 /       |              \
        Astronomy   Amateurs_Astronomy Pictures
           /  \                            |
Astrophysics  Cosmology                Astronomy
                                        /  |    \
                                 Galaxies Stars Astronauts
```

- Typical queries:
  - Navigation by categories
  - All items about Astronomy
  - For given item find all related one

Hierarchical data

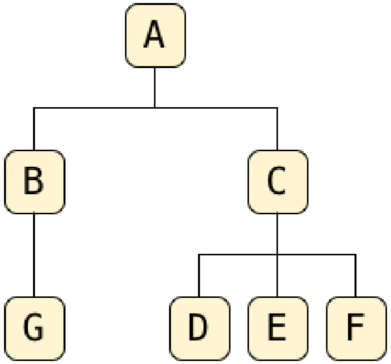
Example: Web-site about astronomy

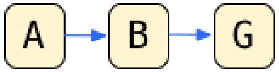
```
                       TOP
                     /   |  \
             Science Hobbies Collections
                 /       |              \
        Astronomy   Amateurs_Astronomy Pictures
           /  \                            |
Astrophysics  Cosmology                Astronomy
                                        /  |    \
                                 Galaxies Stars Astronauts
```

- Typical schema
  id, cat_id, item — very relational, need traverse the tree every time, can be slow
- Materialized the path — replace cat_id by path from the root

Ltree

- Ltree — an official extension (contrib/ltree) implementing support of materialized path in PG since 8.0 (inital release July 13, 2002, PG 7.2)
  - Provides data types, functions, operators and indexes

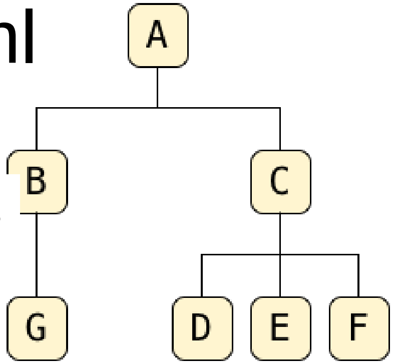




```
SELECT 'A.B.G'::ltree AS "path_to_G";
 path_to_G
-----------
 A.B.G
(1 row)
```

Ltree definitions

- Ltree — a data type representing materialized path
  https://www.postgresql.org/docs/current/ltree.html



- **A label** of a node is a sequence of alphanumeric characters and underscores. Labels must be less than 256 bytes long.

  (Extending set of allowed symbols https://commitfest.postgresql.org/25/1977/)

- **A label path** is a sequence of zero or more labels separated by **dots**, for example L1.L2.L3, representing a path from the root of a hierarchical tree to a particular node. The length of a label path must be less than 65kB, but keeping it under 2kB is preferable.

  Example: Top.Countries.Europe.Russia

Ltree data types

- *ltree* stores a label path.
- *Lquery* — a query for matching *ltree.*
  (A star symbol (*) matches zero or more labels)

`foo`	Match the exact label path foo
`*.foo.*`	Match any label path containing the label foo
`*.foo`	Match any label path whose last label is foo
`*{n}`	Match exactly n labels
`*{n,}`	Match at least n labels
`*{n,m}`	Match at least n but not more than m labels
`*{,m}`	Match at most m labels – same as `*{0,m}`
`@`	Case-insensitive match
`*`	Prefix match
`%`	Match words (separated by _)

- *ltxtquery* represents a full-text-search-like pattern for matching *ltree* values, ltxtquery matches words without regard to their position in the label path.

Lquery example

Lquery is flexible query language for *ltree.*

```
Top.*{0,2}.sport*@.!football|tennis.Russ*|Spain
a.  b.     c.      d.               e.
```

This query will match any label path that:

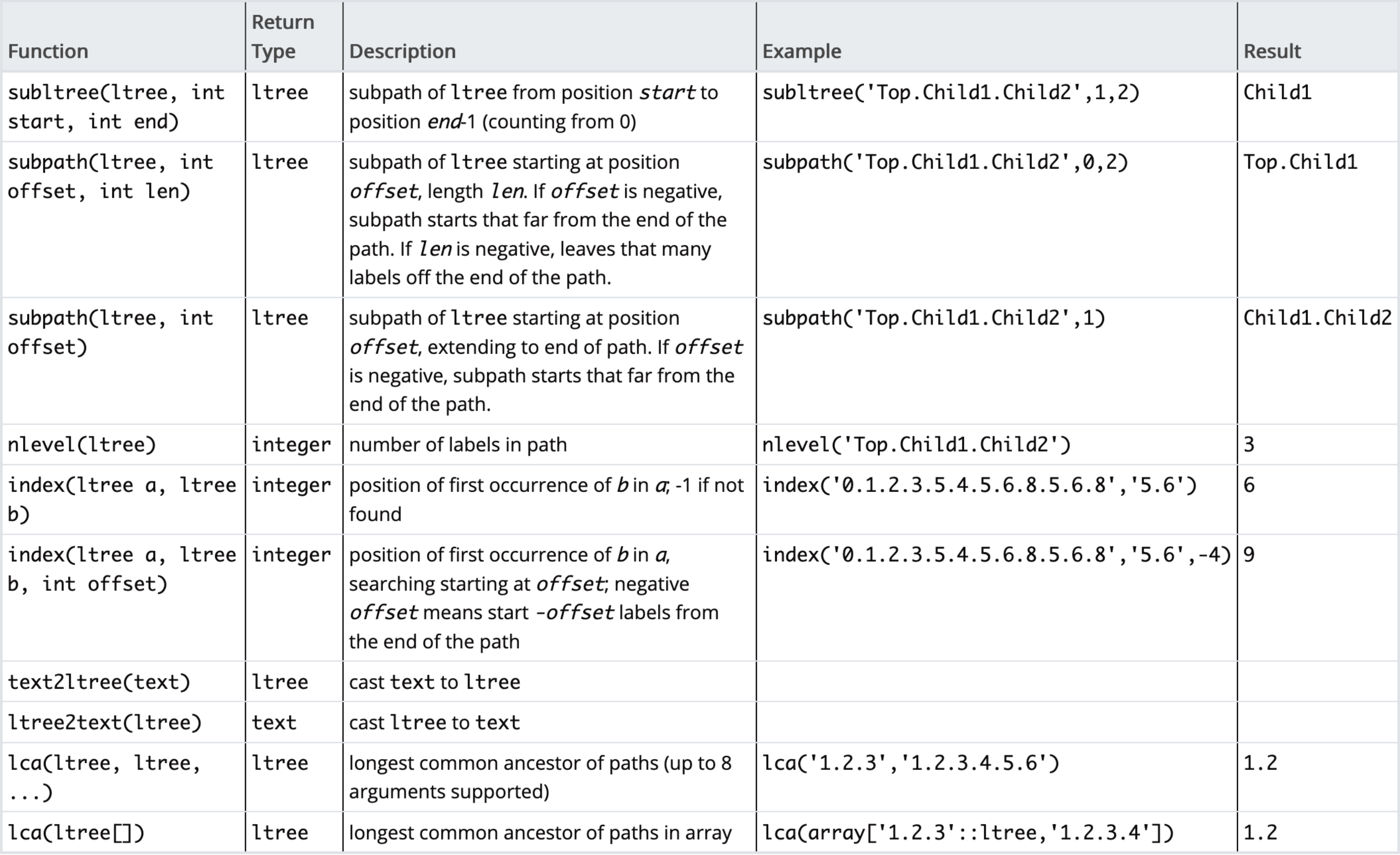
- a. - begins with the label Top
- b. - and next has zero to two labels before
- c. - a label beginning with the case-insensitive prefix sport
- d. - then a label not matching football nor tennis
- e. and then ends with a label beginning with 'Russ' or exactly matching 'Spain'.

Ltree operators

- Comparison operators =, <>, <, >, <=, >=

- ltree @> ltree - is left argument an ancestor of right (or equal)?
- ltree <@ ltree - is left argument a descendant of right (or equal)?
- ltree ~ lquery - does ltree match lquery?
- ltree ? lquery[] - does ltree match any lquery ?
- ltree @ ltxtquery - does ltree match ltxtquery?
- + many others, see https://www.postgresql.org/docs/current/ltree.html#id-1.11.7.30.9

Ltree functions



Function	Return Type	Description	Example	Result
`subltree(ltree, int start, int end)`	`ltree`	subpath of `ltree` from position *`start`* to position *`end`*-1 (counting from 0)	`subltree('Top.Child1.Child2',1,2)`	`Child1`
`subpath(ltree, int offset, int len)`	`ltree`	subpath of `ltree` starting at position *`offset`*, length *`len`*. If *`offset`* is negative, subpath starts that far from the end of the path. If *`len`* is negative, leaves that many labels off the end of the path.	`subpath('Top.Child1.Child2',0,2)`	`Top.Child1`
`subpath(ltree, int offset)`	`ltree`	subpath of `ltree` starting at position *`offset`*, extending to end of path. If *`offset`* is negative, subpath starts that far from the end of the path.	`subpath('Top.Child1.Child2',1)`	`Child1.Child2`
`nlevel(ltree)`	`integer`	number of labels in path	`nlevel('Top.Child1.Child2')`	`3`
`index(ltree a, ltree b)`	`integer`	position of first occurrence of *`b`* in *`a`*; -1 if not found	`index('0.1.2.3.5.4.5.6.8.5.6.8','5.6')`	`6`
`index(ltree a, ltree b, int offset)`	`integer`	position of first occurrence of *`b`* in *`a`*, searching starting at *`offset`*; negative *`offset`* means start -*`offset`* labels from the end of the path	`index('0.1.2.3.5.4.5.6.8.5.6.8','5.6',-4)`	`9`
`text2ltree(text)`	`ltree`	cast `text` to `ltree`		
`ltree2text(ltree)`	`text`	cast `ltree` to `text`		
`lca(ltree, ltree, ...)`	`ltree`	longest common ancestor of paths (up to 8 arguments supported)	`lca('1.2.3','1.2.3.4.5.6')`	`1.2`
`lca(ltree[])`	`ltree`	longest common ancestor of paths in array	`lca(array['1.2.3'::ltree,'1.2.3.4'])`	`1.2`

Ltree indexes

contrib/ltree provides indexing support for ltree

- B-tree index over ltree:
  - <, <=, =, >=, >
- GiST index over ltree:
  - <, <=, =, >=, >, @>, <@, @, ~, ?
- GiST index over ltree[]:
  - @>, <@, @, ~, ?

Ltree example

```
                        Top
                     /   |  \
             Science Hobbies Collections
                 /       |              \
        Astronomy   Amateurs_Astronomy Pictures
           /  \                            |
Astrophysics  Cosmology                Astronomy
                                        /  |    \
                                 Galaxies Stars Astronauts
```

```
SELECT path FROM test WHERE path <@ 'Top.Science';
                path
------------------------------------
 Top.Science
 Top.Science.Astronomy
 Top.Science.Astronomy.Astrophysics
 Top.Science.Astronomy.Cosmology
(4 rows)
```

Ltree example

```
                        Top
                     /   |  \
             Science Hobbies Collections
                 /       |              \
        Astronomy   Amateurs_Astronomy Pictures
           /  \                            |
Astrophysics  Cosmology                Astronomy
                                        /  |    \
                                 Galaxies Stars Astronauts
```

```
SELECT path FROM test WHERE path ~ '*.Astronomy.*';
                     path
-----------------------------------------------
 Top.Science.Astronomy
 Top.Science.Astronomy.Astrophysics
 Top.Science.Astronomy.Cosmology
 Top.Collections.Pictures.Astronomy
 Top.Collections.Pictures.Astronomy.Stars
 Top.Collections.Pictures.Astronomy.Galaxies
 Top.Collections.Pictures.Astronomy.Astronauts
(7 rows)
```

Ltree example

```
                        Top
                     /   |  \
             Science Hobbies Collections
                 /       |              \
        Astronomy   Amateurs_Astronomy Pictures
           /  \                            |
Astrophysics  Cosmology                Astronomy
                                        /  |    \
                                 Galaxies Stars Astronauts
```

```
SELECT path FROM test WHERE path ~ '*.!pictures@.*.Astronomy.*';
                path
------------------------------------
 Top.Science.Astronomy
 Top.Science.Astronomy.Astrophysics
 Top.Science.Astronomy.Cosmology
(3 rows)
```

Ltree example

```
                        Top
                     /   |  \
             Science Hobbies Collections
                 /       |              \
        Astronomy   Amateurs_Astronomy Pictures
           /  \                            |
Astrophysics  Cosmology                Astronomy
                                        /  |    \
                                 Galaxies Stars Astronauts
```

```
SELECT path FROM test WHERE path @ 'Astro* & !pictures@';
                path
------------------------------------
 Top.Science.Astronomy
 Top.Science.Astronomy.Astrophysics
 Top.Science.Astronomy.Cosmology
(3 rows)
```

Ltree example

```
                        Top
                     /   |  \
             Science Hobbies Collections
                 /       |              \
        Astronomy   Amateurs_Astronomy Pictures
           /  \                            |
Astrophysics  Cosmology                Astronomy
                                        /  |    \
                                 Galaxies Stars Astronauts
```

```
SELECT subpath(path,0,2)||'Space'||subpath(path,2) FROM test WHERE
path <@ 'Top.Science.Astronomy';
                 ?column?
------------------------------------------
 Top.Science.Space.Astronomy
 Top.Science.Space.Astronomy.Astrophysics
 Top.Science.Space.Astronomy.Cosmology
(3 rows)
```

GiST: RD-Tree (Signature tree)

- label signature — labels hashed to the specific position of '1'

w1 -> S1: 01000000   ltree: w1.w2.w3

w2 -> S2: 00010000

w3 -> S3: 10000000

- Query (ltree) signature — superposition (bit-wise OR) of signatures

S: 11010000

- Bloom filter

Q1: 00000001 – exact not

Q2: 01010000 - may be contained in the document, **false drop**

- Signature is a lossy representation of ltree
  - + fixed length, compact, + fast bit operations
  - - lossy (false drops)

GiST: RD-Tree (Signature tree)

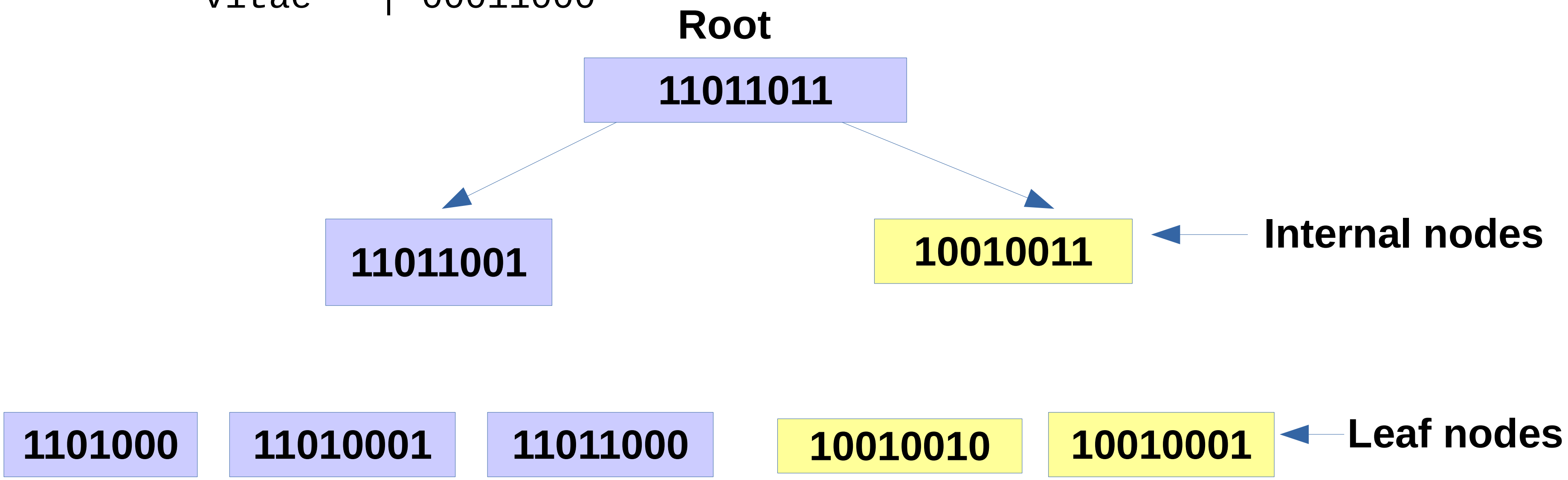
- Latin proverbs

```
id |        proverb
---+-----------------------
 1 | Ars.longa.vita.brevis
 2 | Ars.vitae
 3 | Jus.vitae.ac.necis
 4 | Jus.generis.humani
 5 | Vita.nostra.brevis
```

GiST: RD-Tree (Signature tree)

```
  labels    |  signature
  ---------+-----------
   ac        | 00000011
   ars       | 11000000
   brevis    | 00001010
   generis   | 01000100
   humani    | 00110000
   jus       | 00010001
   longa     | 00100100
   necis     | 01001000
   nostra    | 10000001
   vita      | 01000001
   vitae     | 00011000
```

QUERY



Root

11011011

11011001

10010011

Internal nodes

1101000

11010001

11011000

10010010

10010001

Leaf nodes

RD-Tree (GiST)

```
 id |        proverb          | signature
----+------------------------+-----------
  1 | Ars.longa.brevis       | 11101111
  2 | Ars.vitae              | 11011000
  3 | Jus.vitae.ac.necis     | 01011011
  4 | Jus.generis.humani     | 01110101
  5 | Vita.nostra.brevis     | 11001011
```

False drop

RD-Tree (GiST)

- Problems
  - Not very good scalabilty with increasing of cardinality of labels and records.
  - Index is lossy, need check for false drops (Recheck in EXPLAIN ANALYZE)

GIN over ltree

- Put ltree as is in entry tree of GIN (length limit)
- Parent — cut last label and do lookup
- Child — range scan starting with given ltree until keys has the same prefix

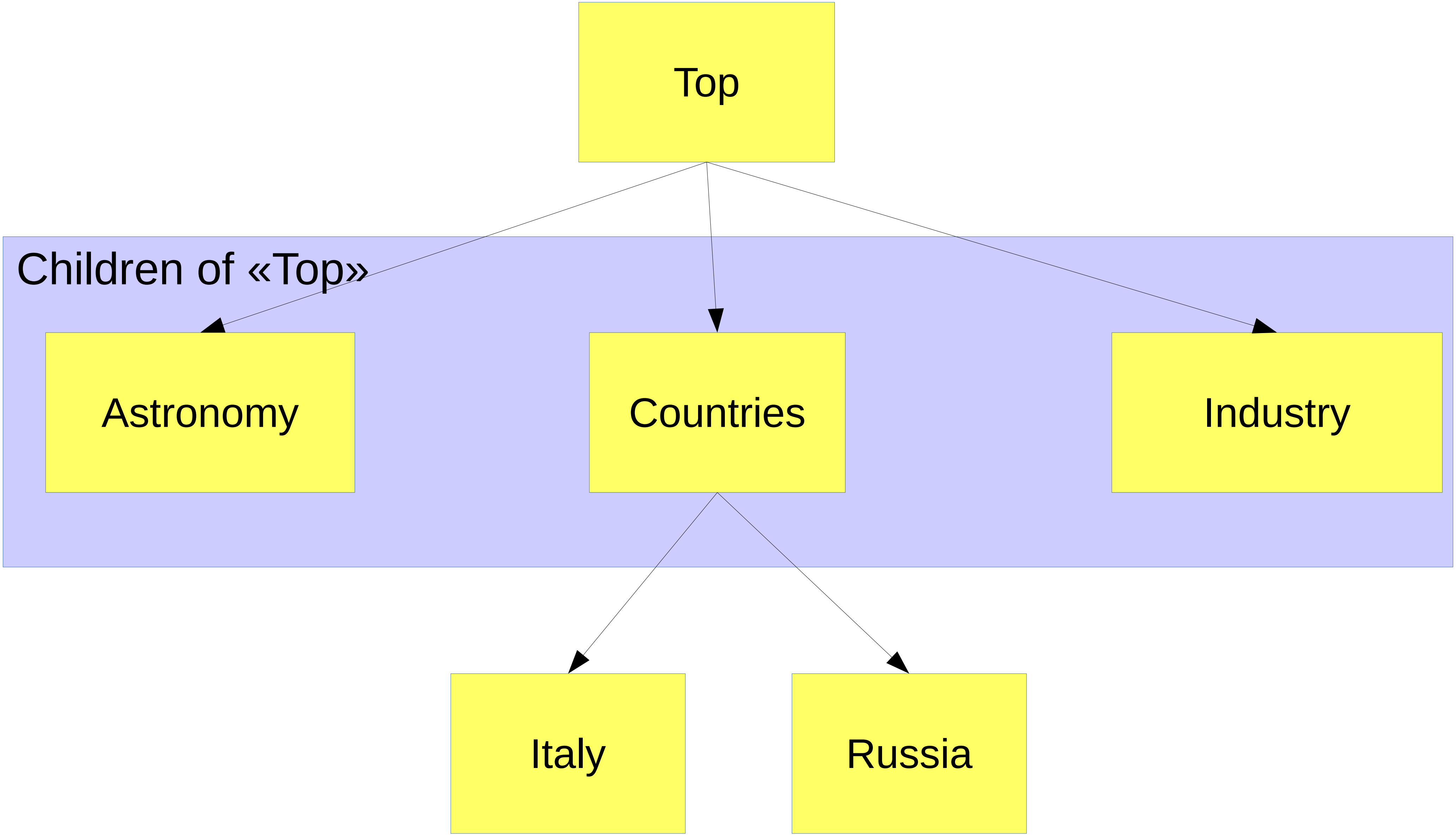
DMOZ catalog

- 332778 nodes
- 2335790 resources
- ~2.5 Gb with indexes

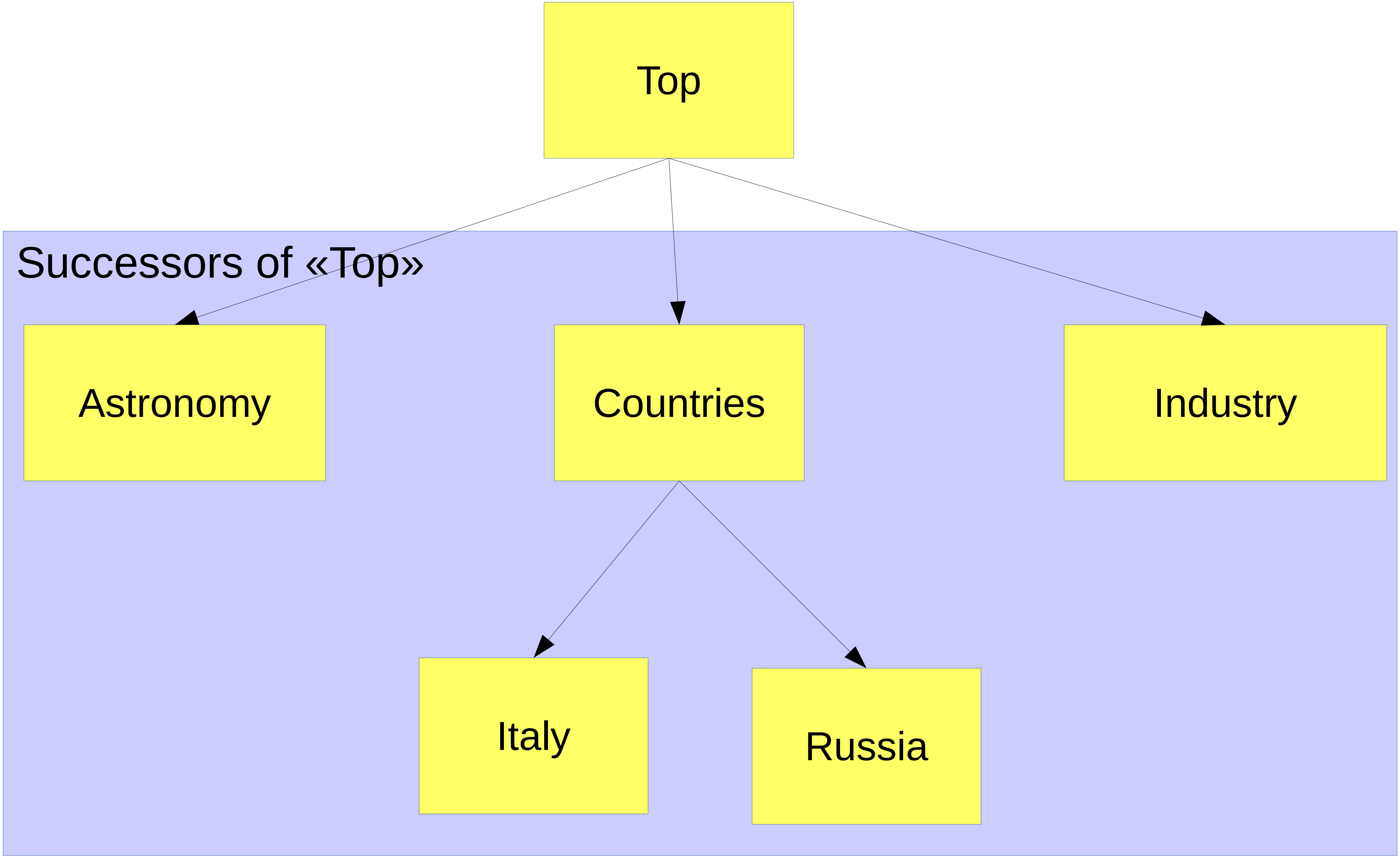
What to test

- Tree navigation
  - Get children
  - Get successors
  - Get predecessors (path to the root)
  - Get siblings
- Resource retrieval
  - Get resources linked to current node
  - Get resources linked to successors of current node

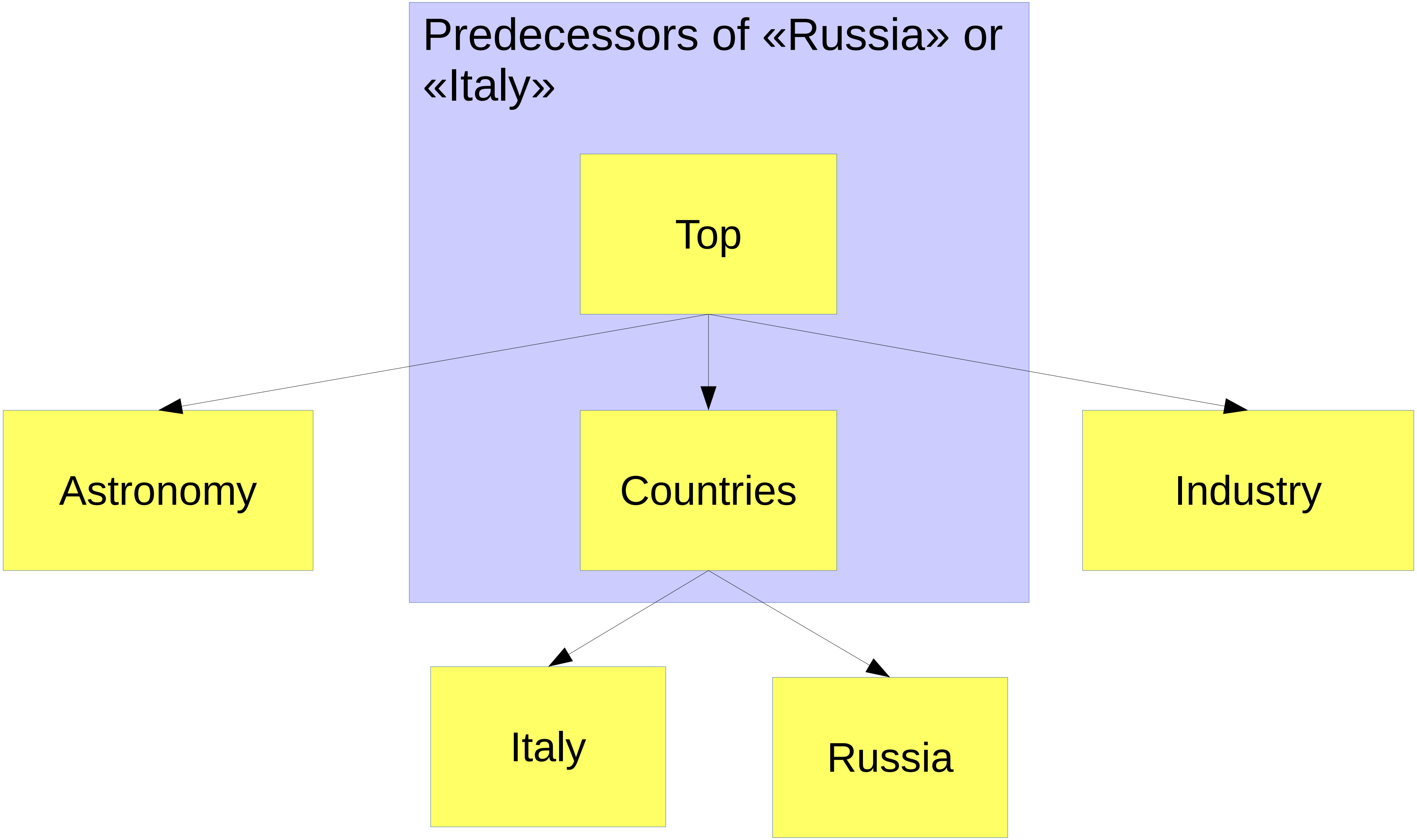
Tree naming


Top
Children of «Top»
Astronomy
Countries
Industry
Italy
Russia

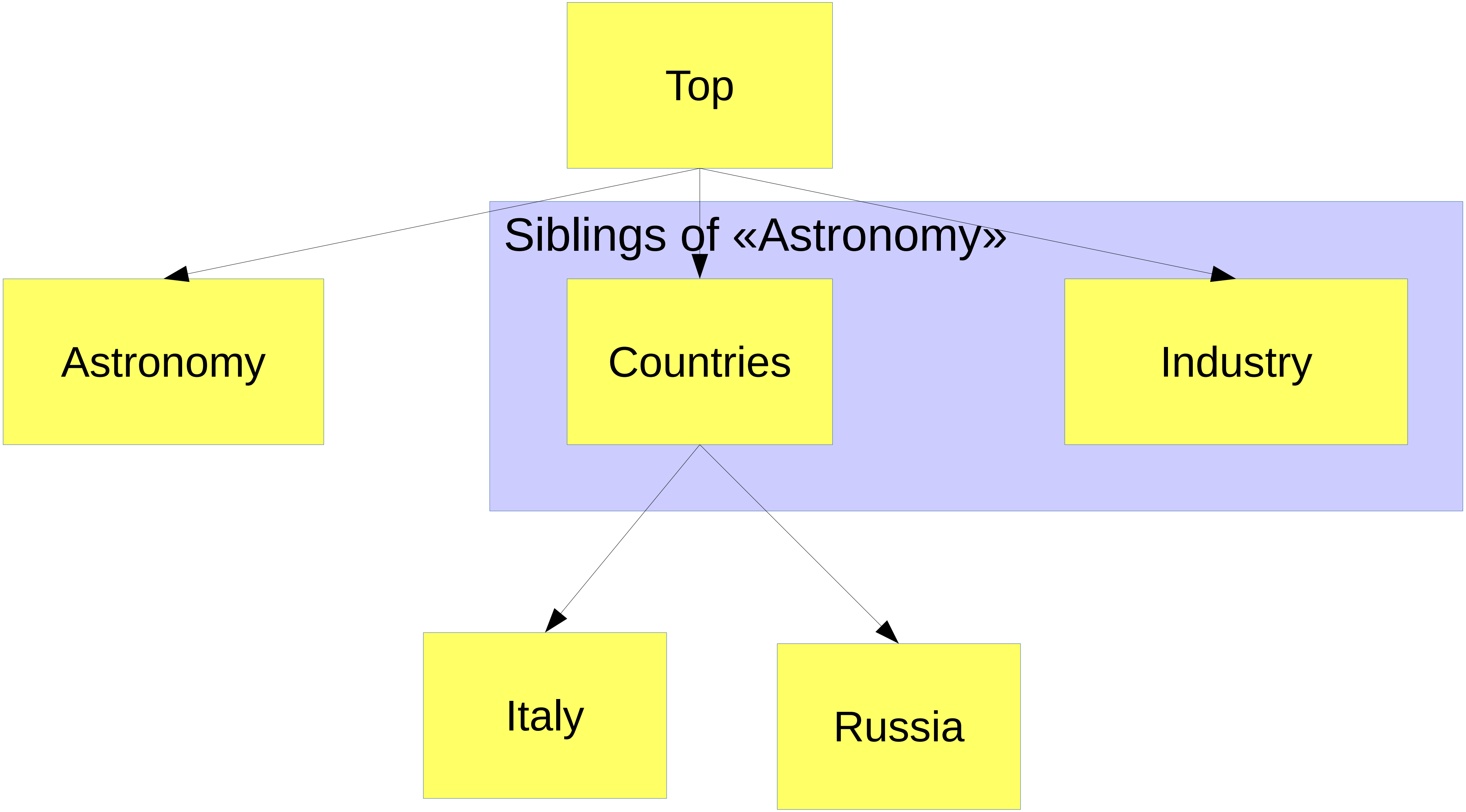
Tree naming


Top
Successors of «Top»
Astronomy
Countries
Industry
Italy
Russia

Tree naming


Predecessors of «Russia» or «Italy»

Tree naming


Top
Siblings of «Astronomy»
Astronomy
Countries
Industry
Italy
Russia

How to store

Store hierarcy

- Parent id
- Ranges
- Ltree

Store linked resources

- Many-to-many table (node_id, resource_id)
- List node's id
- List node's ltree

Nodes

```
 Table "public.dmozv"
   Column   |  Type   | Collation | Nullable | Default
------------+---------+-----------+----------+---------
 id         | integer |           |          |
 name       | text    |           |          |
 path       | ltree   |           |          |
 parentid   | integer |           |          |
 children   | integer |           |          |
 low        | integer |           |          |
 high       | integer |           |          |
 childorder | integer |           |          |
Indexes:
    "dmozv_id_idx" UNIQUE, btree (id)
    "dmozv_lh_idx" UNIQUE, btree (low, high)
    "dmozv_path_idx" gist (path)
    "dmozv_pc_idx" btree (parentid)
```

Parent id

```
Table "public.dmozv"
  Column    |  Type    | Collation | Nullable | Default
------------+----------+-----------+----------+---------
id          | integer  |           |          |
name        | text     |           |          |
path        | ltree    |           |          |
parentid    | integer  |           |          |
children    | integer  |           |          |

  id   |           name            | parentid
-------+---------------------------+----------
 19269 | Characters                |    15597
 19270 | Gamera                    |    19269
 19271 | Vampira                   |    19269
 19272 | The_Rocketeer             |    19269
 19273 | Snowmiser_and_Heatmiser   |    19269
 19274 | Hopalong_Cassidy          |    19269
```

Ranges

```
 Table "public.dmozv"
   Column   |  Type   | Collation | Nullable | Default
------------+---------+-----------+----------+---------
 id         | integer |           |          |
 name       | text    |           |          |
 low        | integer |           |          |
 high       | integer |           |          |

  id   |          name           |  low   |  high
-------+-------------------------+--------+--------
 19269 | Characters              | 100000 | 100055
 19270 | Gamera                  | 100010 | 100010
 19271 | Vampira                 | 100050 | 100050
 19272 | The_Rocketeer           | 100040 | 100040
 19273 | Snowmiser_and_Heatmiser | 100030 | 100030
 19274 | Hopalong_Cassidy        | 100020 | 100020
```

Path

```
Table "public.dmozv"
   Column   |  Type   | Collation | Nullable | Default
------------+---------+-----------+----------+----------
 id         | integer |           |          |
 name       | text    |           |          |
 path       | ltree   |           |          |

  id   |          name           |                        path
-------+-------------------------+----------------------------------------------------
 19269 | Characters              | Top.Arts.Movies.Characters
 19270 | Gamera                  | Top.Arts.Movies.Characters.Gamera
 19271 | Vampira                 | Top.Arts.Movies.Characters.Vampira
 19272 | The_Rocketeer           | Top.Arts.Movies.Characters.The_Rocketeer
 19273 | Snowmiser_and_Heatmiser | Top.Arts.Movies.Characters.Snowmiser_and_Heatmiser
 19274 | Hopalong_Cassidy        | Top.Arts.Movies.Characters.Hopalong_Cassidy
```

Resources

```
               Table "public.resource"
 Column |   Type    | Collation | Nullable | Default
--------+-----------+-----------+----------+---------
 id     | integer   |           |          |
 title  | text      |           |          |
 url    | text      |           |          |
 ids    | integer[] |           |          |
 path   | ltree[]   |           |          |
Indexes:
    "r_id_idx" UNIQUE, btree (id)
    "r_idpath_idx" gist (ids gist__intbig_ops)
    "r_path_idx" gist (path)
```

How to test

- PostgreSQL 12.0
- Intel(R) Core(TM) i7-3520M CPU @ 2.90GHz, 2/4 cores, 16Gb

-
  ```
  % cat node_select.sql
  \set nid random(1, 332778)
  select id, name from dmozv where id = :nid;
  ```
- pgbench
  ```
    -n -T 60 -c 4 -j 2 -f node_select.sql dmoz
        ~35000 tps (the same for resources)
    + -M prepared
        ~67200 tps (64000 tps for resources)
  ```

Tree navigation: children

- Path:

  ```
  select a.id, a.name
    from dmozv a, dmozv i  where
    i.id = :nid and
        a.path  ~ (i.path::text || '.*{1}')::lquery;
  ```

- Ranges:

  :(

- Parent id:

  ```
  select id, name from dmozv where parentid=:nid;
  ```

Tree navigation: successors

- Path:

```
select a.id, a.name from dmozv a, dmozv i
    where i.id = :nid and
       i.path @> a.path;
```

- Ranges:

```
select a.id, a.name from dmozv a, dmozv i  where i.id = :nid
and
   i.low <= a.low and a.high <= i.high;
```

- Parent id:

```
with recursive a as (
    select id, name, parentid from dmozv where id = :nid
    union all select d.id, d.name, d.parentid from  dmozv d, a
          where d.parentid = a.id)
select id, name from a;
```

Tree navigation: predecessors

- Path:

```
select a.id, a.name from dmozv a, dmozv i
      where i.id = :nid and
               i.path <@ a.path;  (was @>)
```

- Ranges:

```
select a.id, a.name from dmozv a, dmozv i
   where i.id = :nid and
        i.low >= a.low and a.high >= i.high; (was <=)
```

- Parent id:

```
with recursive a as (
   select id, name, parentid from dmozv where id = :nid
   union all select d.id, d.name, d.parentid  from dmozv d, a
        where a.parentid = d.id)
select id, name from a;  (was d.parentid = a.id)
```

Tree navigation: siblings

Parent id:

```
select
    b.id, b.name
from
    dmozv b, dmozv n
where
    n.id=:nid and n.parentid=b.parentid;
```

Tree navigation: siblings

Path:

```
select
    b.id, b.name
from
    dmozv b, dmozv n
where
    n.id=:nid and
    b.path ~
    (subpath(n.path, 0, -1)::text ||  .*{1}')::lquery;
```

Uuuuu.. magick

Tree navigation: siblings

Ranges: no way :(

Or I don“t know how

Result for tree navigation

Test	Not prepared	Prepared
Children	**GiST (GIN)**	**GiST (GIN)**
parentid	35200	68000
path	13100 (16000)	22000(32900)
Successors		
parentid	12800	36000
path	10400(18000)	15400(40600)
ranges	533	543
Predesessors		
parentid	12600	33600
path	5000 (14900)	5800(27700)
ranges	532	544
Siblings		
path	7800(4600)	11100 (5700)
parentid	12200	45000

Resources: only node

- List ids:

```
select r.id, r.title from resource r where
       r.ids && ARRAY[int4(:nid)];
```

- List paths:

```
select r.id, r.title from resource r, dmozv d where
    r.path && ARRAY[d.path] and d.id = :nid;
```

- Join:

```
 select r.id, r.title from resource r,
                      dmoz_resource dr where
     dr.nid = :nid and dr.rid = r.id;
```

Resources: node with successors

- List ids + range:

```
select r.id, r.title
       from resource r, dmozv a, dmozv i where
  r.ids && ARRAY[a.id] and i.id = :nid and
  i.low <= a.low and a.high <= i.high;
```

- List ids + parent id:

```
with recursive a as (
    select id, name, parentid from dmozv
          where id = :nid
    union all
    select d.id, d.name, d.parentid from
        dmozv d, a where
                  d.parentid = a.id)
select r.id, r.title from a, resource r where
    r.ids && ARRAY[a.id];
```

Resources: node with successors

- Join + range:

```
select r.id, r.title
 from resource r, dmozv a, dmozv i, dmoz_resource dr
     where
 dr.nid = a.id and dr.rid = r.id and i.id = :nid and
     i.low <= a.low and a.high <= i.high;
```

- Join + parent id:

```
with recursive a as (
    select id, name, parentid from dmozv where id = :nid
    union all select d.id, d.name, d.parentid
         from dmozv d, a where d.parentid = a.id)
select r.id, r.title
         from a, resource r, dmoz_resource dr
where
         dr.nid = a.id and dr.rid = r.id;
```

Resources: node with successors

- Path:

```
select r.id, r.title
   from resource r, dmozv d
   where
d.id=:nid and r.path <@ d.path;
```

Result for resources

Test	Not prepared	Prepared
Only node	**GiST (GIN)**	**GiST (GIN)**
ids	270	840
paths	82 (16400)	83 (30600)
join	11700	31500
With successors		
ids+range	56	56
ids+parent id	69	72
join+range	8	8
join+parent id	3300	6900
path	95(11200)	97(17100)

Non-test notes

- Hard update ranges
- Risk of infinite loop for parent id in WITH RECURSIVE (limit recursion?)

New hopes

- Any UTF8 (thanks to Dmitry Belyavsky)
  Top.“Книги“.“Научная фантастика“
- Statistic for ltree (nothing unusual, common problem for non-scalar data such as geo, FTS, json etc)
- GIN (use together with FTS — search documents linked to successors)
- SP-GiST — native hierarchical storage
- Better testing
  - uniform distribution is not a model of real life
  - zipfian distribution

Test set

http://sigaev.ru/misc/dmoz.tgz

PROFESSIONAL Postgres

Questions?!

obartunov@postgrespro.ru

teodor@postgrespro.ru