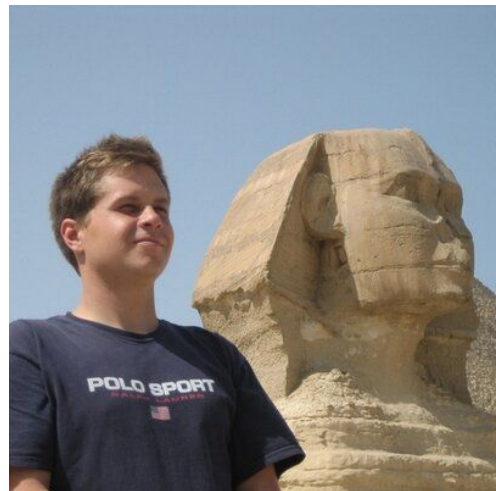


Rails (Anti-)Patterns

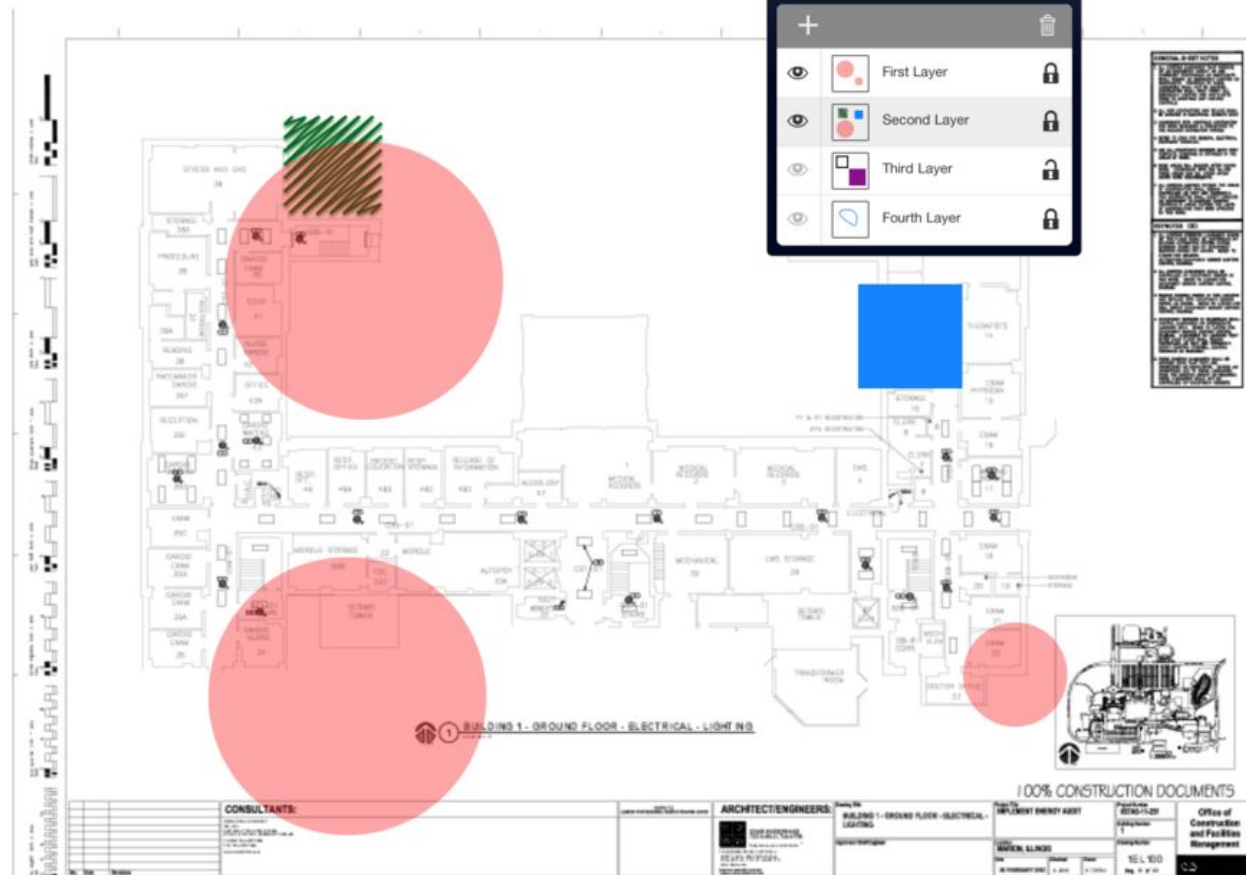
How *not* to design your database

```
{  
  "Brad Urani" : "Staff Engineer",  
  "Company" : " PROCORE ",  
}
```













What is Scale?

20,000,000 rows scanned / sec

30,000 transactions / sec

6 TB



1.5 TB RAM
64 Cores
8 TB HD

BUDGET

Export

Budget

Configurable Views

Forecasting

Legacy Budget Snapshots (0)

Change History (414)

CREATE NEW BUDGET LINE ITEM

Sub Job	Cost Code	Category	Original Budget	
N/A	Select a Cost Code		\$ 0.00	+ Add

Last synced with Sage 300 8 days 23 hours 13 minutes and 26 seconds ago.
Please note: While all JTD/Direct costs have been updated, there may be items in your ERP system that have not been added to the Budget in Procore.

COST CODE DETAIL

Cost Code	Category	Original Budget	Revised Budget	Committed Costs	Direct Costs	Pending Cost Changes	Projected Costs	Forecast To Complete	Estimated Cost At Completion	Projected Over/Under	Job To Date Costs
Division 01 - General Conditions											
01-010004 - A/E Fees	O	\$ 1,490,000.00	\$1,490,000.00	\$0.00	\$0.00	\$0.00	\$0.00	\$1,490,000.00	\$1,490,000.00	\$0.00	\$1,571,975.41
01-010004 - A/E Fees	S	\$ 2,100.00	\$2,100.00	\$0.00	\$0.00	\$0.00	\$0.00	\$2,100.00	\$2,100.00	\$0.00	\$0.00
01-010005 - A/E Reimbursables	O	\$ 300.00	\$300.00	\$0.00	\$297.50	\$0.00	\$297.50	\$2.50	\$300.00	\$0.00	\$297.50
01-010005 - A/E Reimbursables	S	\$ 6,000.00	\$6,000.00	\$0.00	\$0.00	\$0.00	\$0.00	\$6,000.00	\$6,000.00	\$0.00	\$0.00
01-010006 - Administrative Personnel	B	\$ 1,000.00	\$1,000.00	\$0.00	\$104,150.99	\$0.00	\$104,150.99	\$0.00	\$104,150.99	(\$103,150.99)	\$104,150.99
01-010006 - Administrative Personnel	L	\$ 3,689,355.00	\$3,689,355.00	\$0.00	\$643,129.53	\$0.00	\$643,129.53	\$3,046,225.47	\$3,689,355.00	\$0.00	\$643,129.53
01-010006 - Administrative Personnel	O	\$ 489,355.00	\$489,355.00	\$0.00	\$396.30	\$0.00	\$396.30	\$488,958.70	\$489,355.00	\$0.00	\$396.30
01-010006 - Administrative Personnel	PD	\$ 0.00	\$0.00	\$0.00	\$60,120.00	\$0.00	\$60,120.00	\$0.00	\$60,120.00	(\$60,120.00)	\$60,120.00
01-010007 - Admin. Personnel-Fleet Card	O	\$ 36,600.00	\$36,600.00	\$0.00	\$45,395.99	\$0.00	\$45,395.99	\$0.00	\$45,395.99	(\$8,795.99)	\$45,395.99
01-010007 - Admin. Personnel-Fleet Card	Z	\$ 0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00	\$0.00
01-010008 - Admin. Personnel-Car Rentals	O	\$ 30,000.00	\$30,000.00	\$0.00	\$5,448.29	\$0.00	\$5,448.29	\$24,551.71	\$30,000.00	\$0.00	\$5,448.29

Create Snapshot

Re-send to ERP

Lock Budget

IMPORT BUDGET CSV

Download Template

Choose File

No file chosen

Need Help?

Import

VIEWS

Cost Code Detail

Cost Code Summary (Read Only)

FILTERS

Sub Job

None

Division

All

Category

All

Clear All

Apply

CUSTOM REPORTS

+ New

Minimize Sidebar

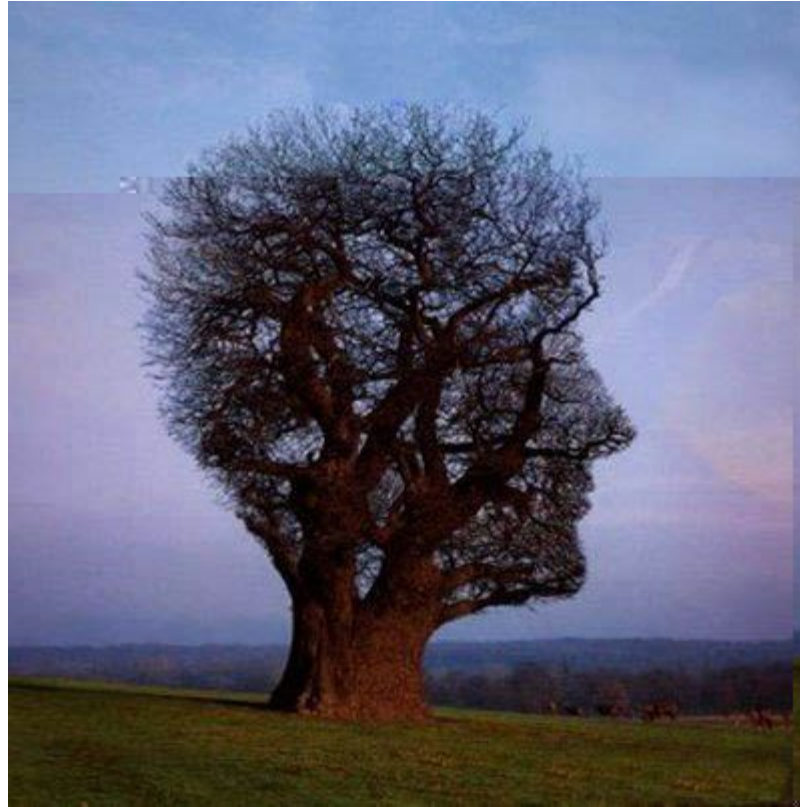
A photograph of a seal lying on its back on a sandy beach. The seal's mouth is wide open, showing its teeth and tongue, giving it a happy or content expression. Its flippers are visible, and the background shows the ocean with gentle waves. A blue speech bubble is drawn around the seal's head and the text.

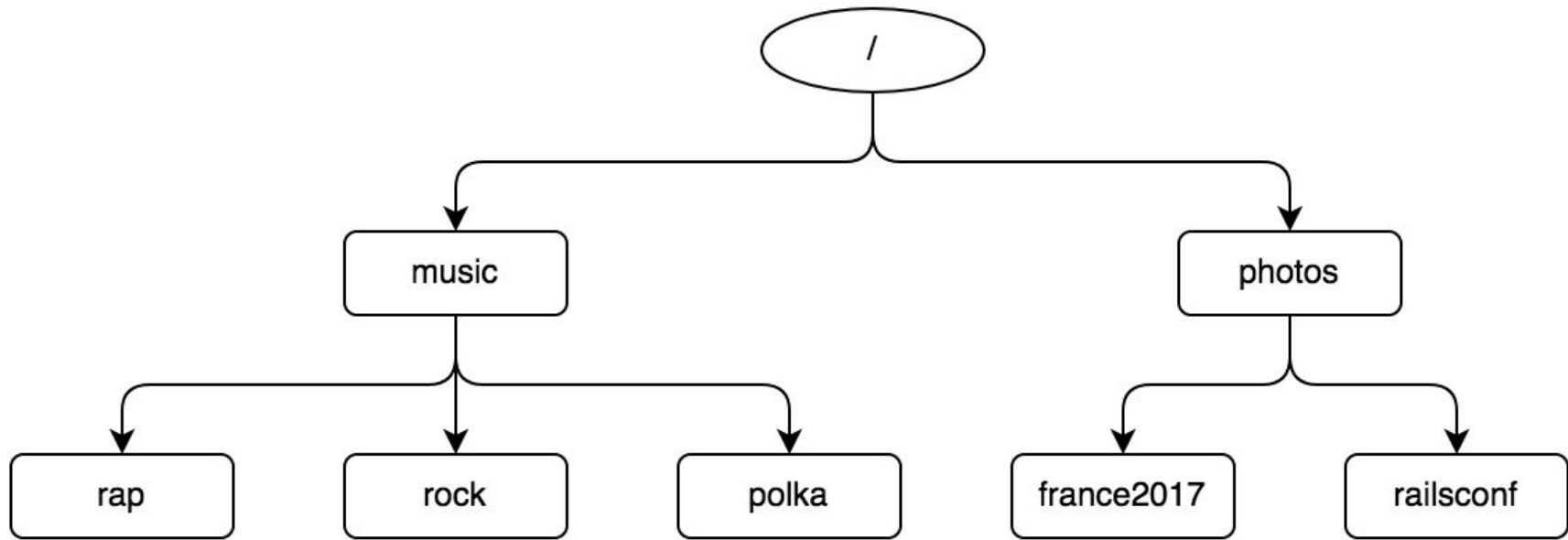
Don't forget
developer
happiness

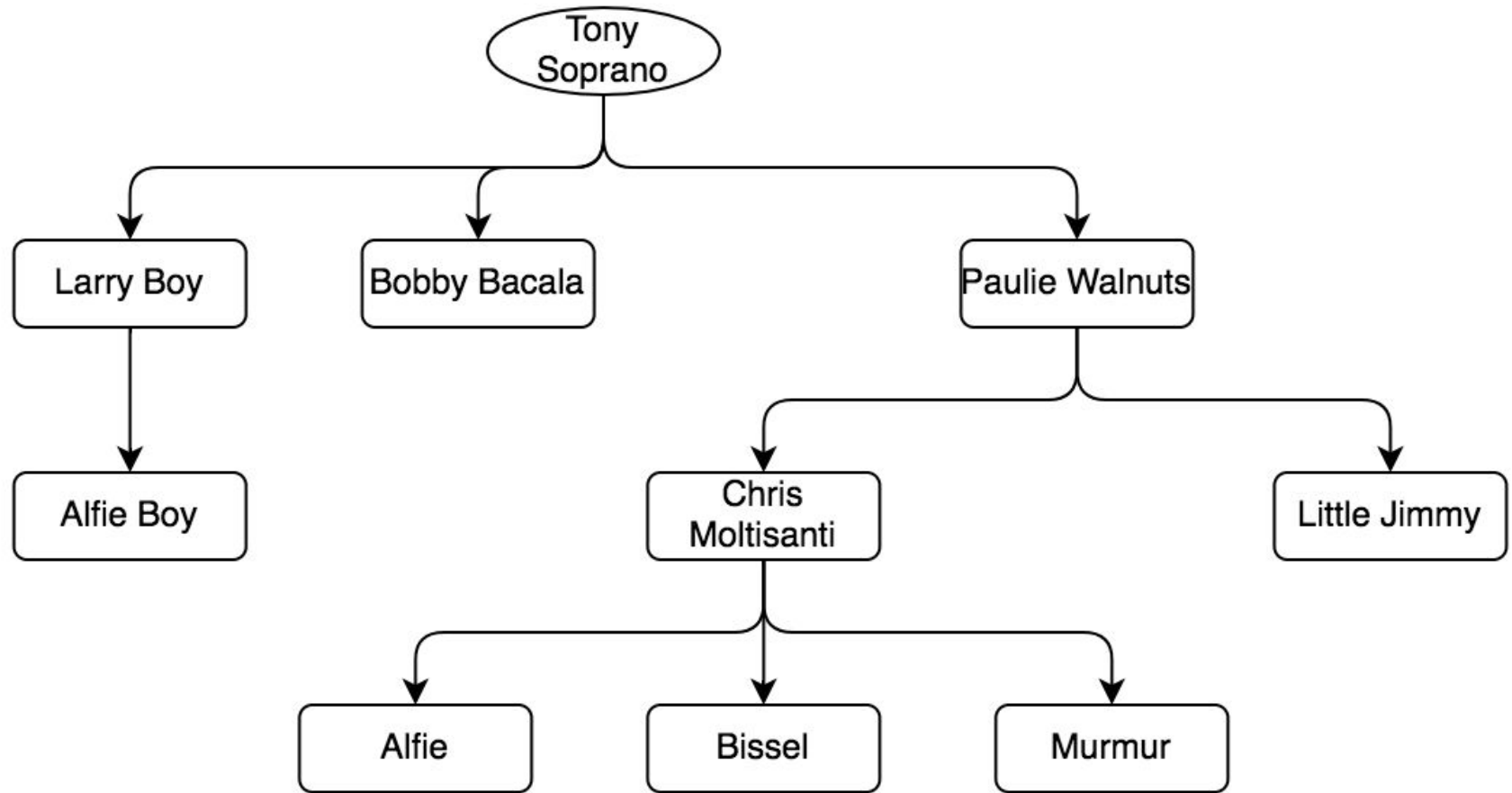
A close-up photograph of a seal's head, tilted back with its mouth wide open in a yawn or call. The seal's fur is dark and textured. A blue hand-drawn speech bubble originates from the seal's mouth and extends towards the left side of the frame. Inside the bubble, the text "A shark will probably eat me" is written in a blue, casual font.

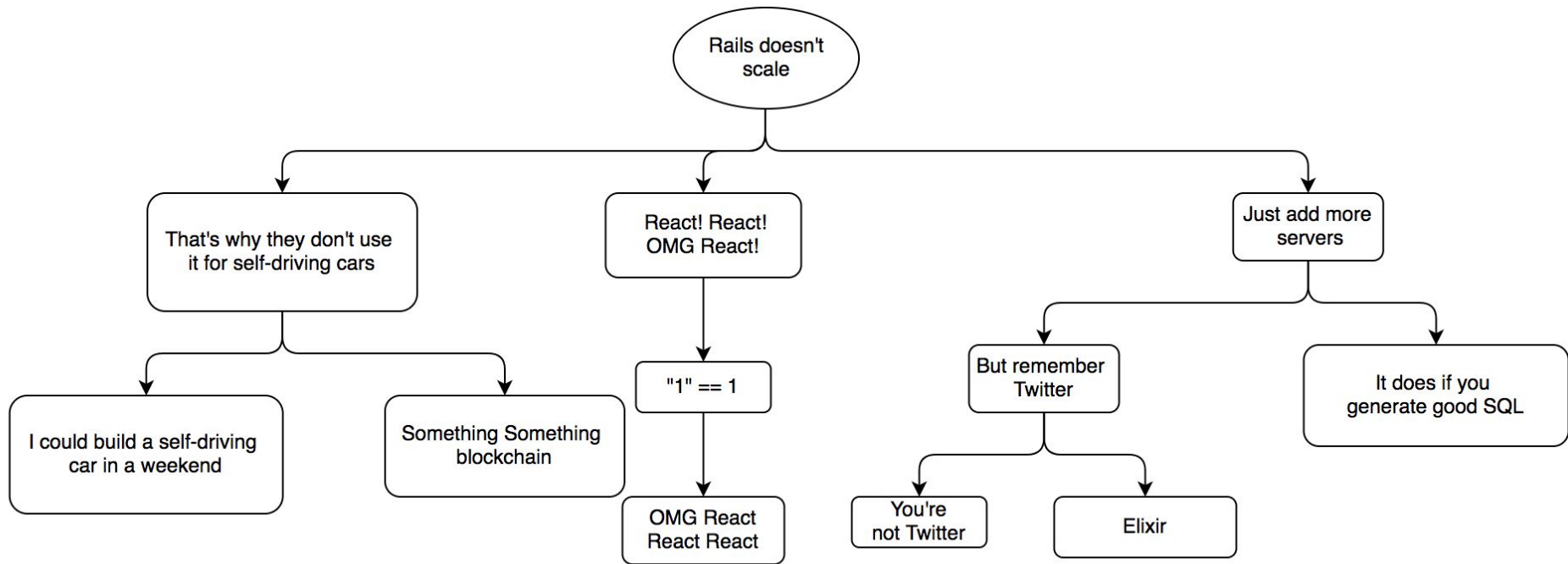
A shark
will
probably
eat me

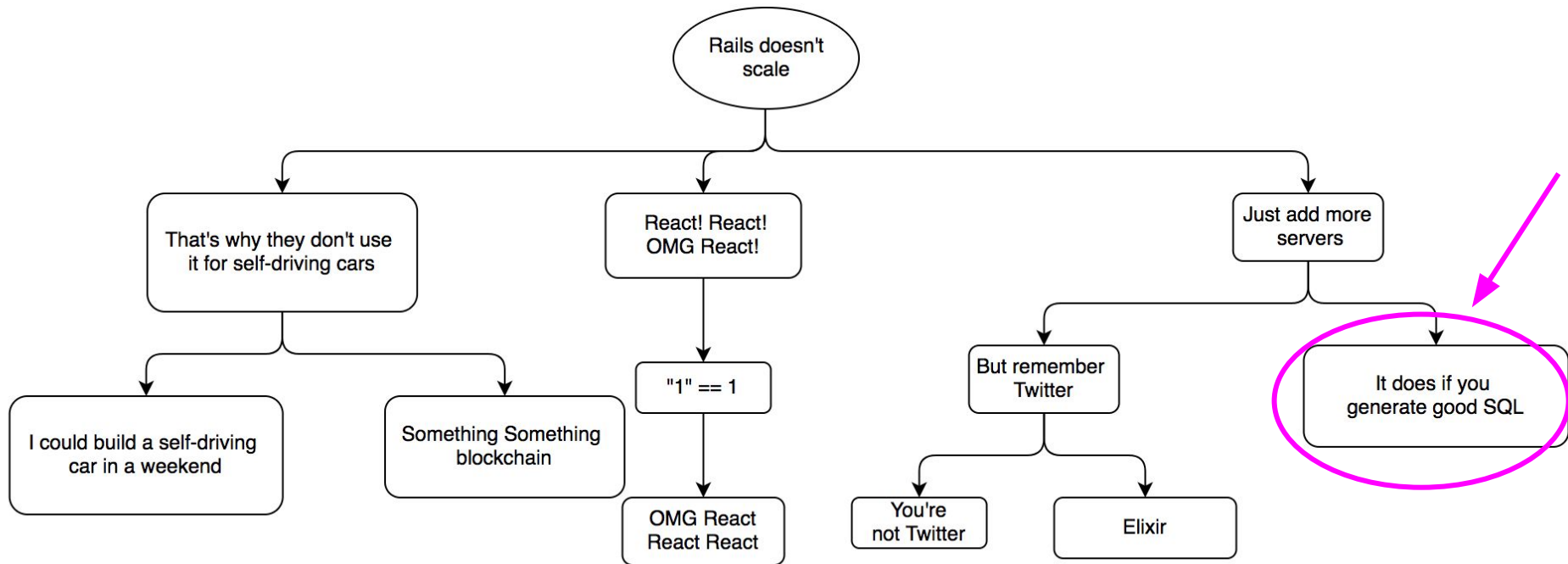
Trees











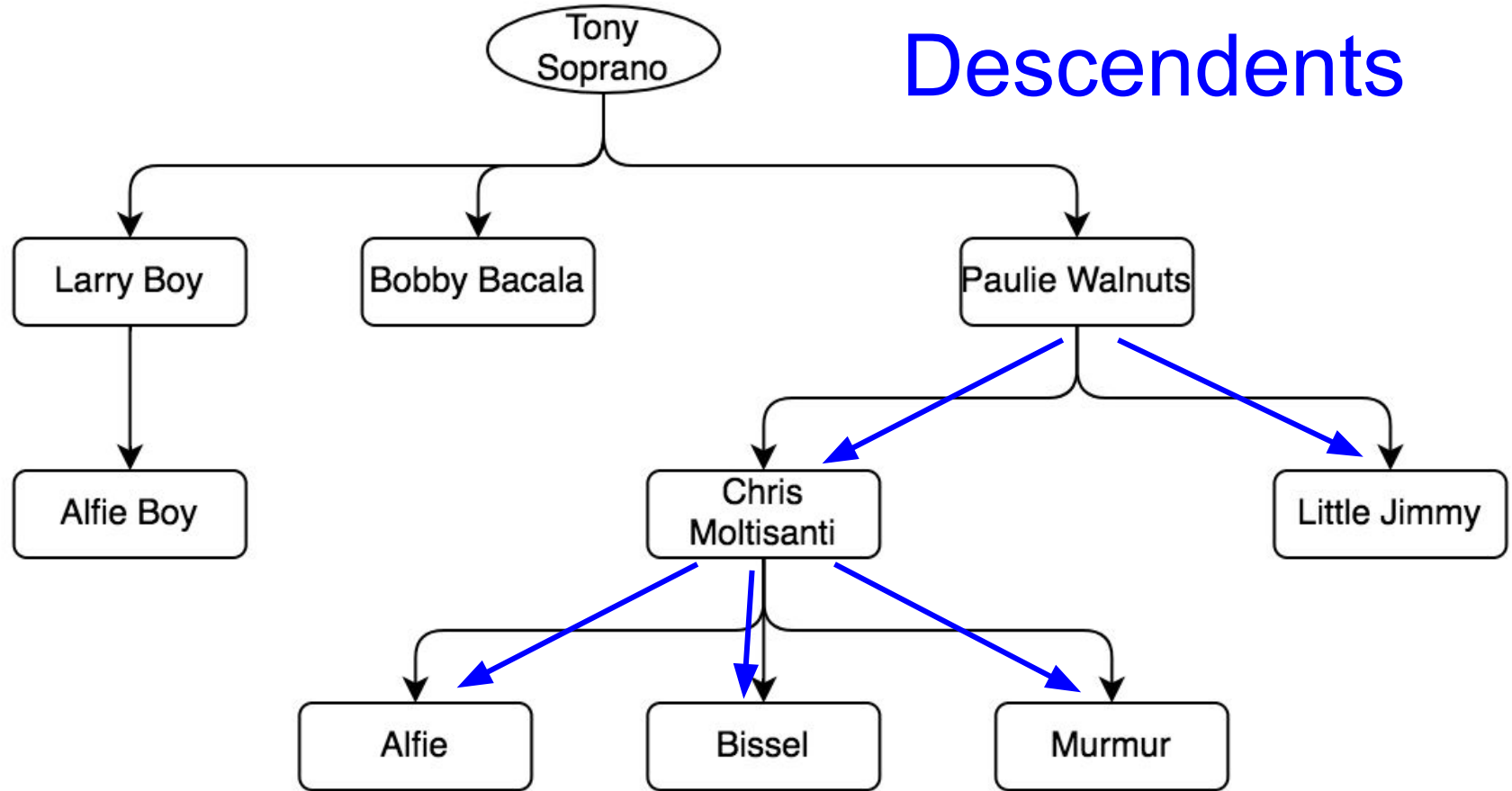
(Anti)-Pattern

Naive Tree



employees	
id	<u>manager_id</u>
1	NULL
2	1
3	1
4	2

Descendents




```
SELECT *  
FROM employees e1  
LEFT JOIN employees e2 ON e1.id = e2.manager_id  
LEFT JOIN employees e3 ON e2.id = e3.manager_id  
LEFT JOIN employees e4 ON e3.id = e4.manager_id  
LEFT JOIN employees e5 ON e4.id = e5.manager_id  
WHERE e1.manager_id IS NULL;
```

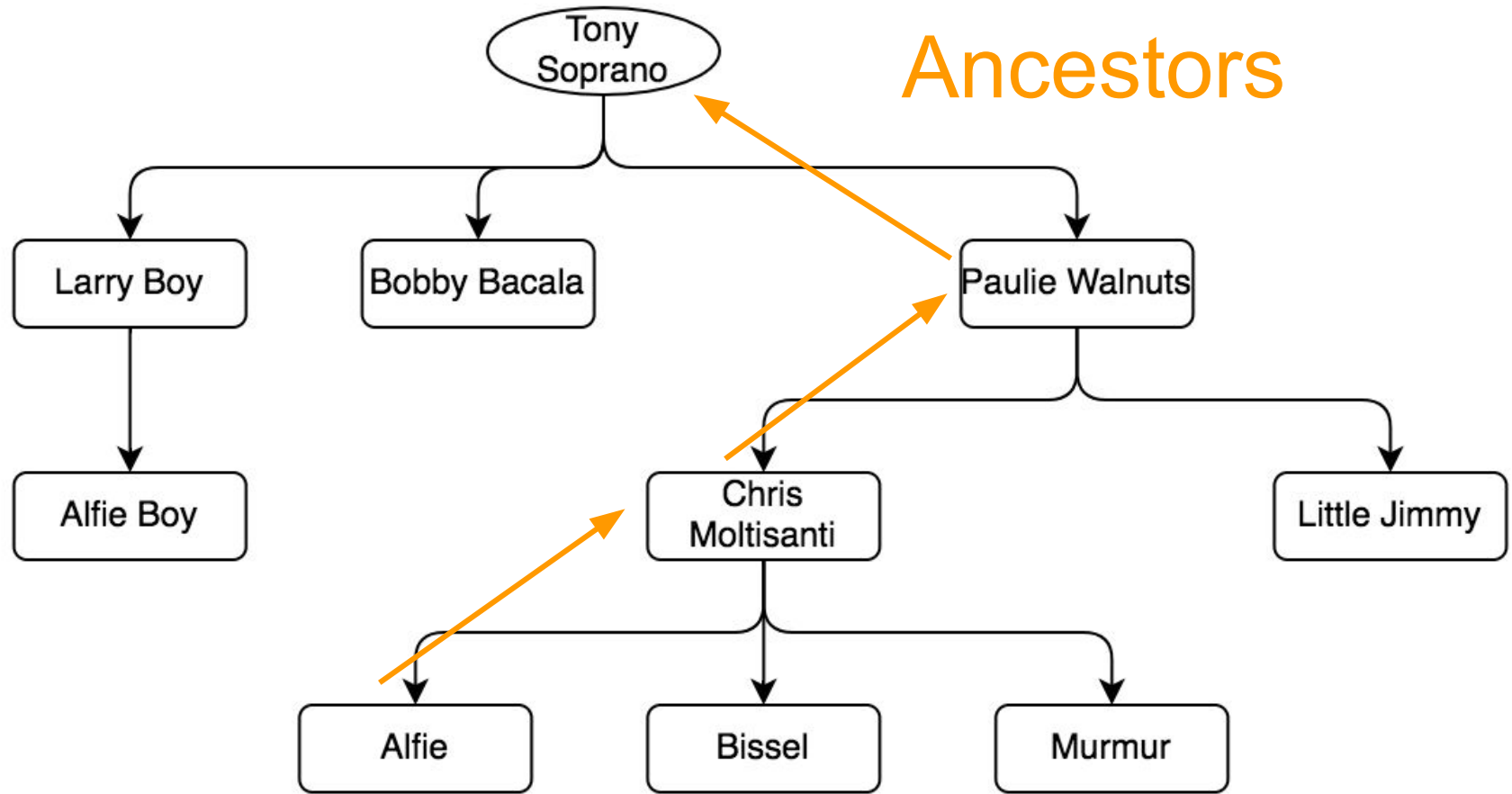
```
SELECT * FROM employees e WHERE e.manager_id IS NULL;  
  
SELECT * FROM employees e WHERE e.manager_id IN (1);  
  
SELECT * FROM employees e WHERE e.manager_id IN (2,3,4);  
  
SELECT * FROM employees e WHERE e.manager_id IN (5,6,7,8,9);  
  
SELECT * FROM employees e WHERE e.manager_id IN (10,11,12,13,14)
```



Recurse or
suffer!!!

```
(  
    cte_query_defini  
Anchor member  
UNION ALL  
    cte_query_defini  
Recursive member  
references cte_nam  
)  
-- Statement using
```

```
WITH RECURSIVE descendants AS
( SELECT parent, child AS descendant, 1 AS level
  FROM employees
  UNION ALL
    SELECT d.parent, s.child, d.level + 1
    FROM descendants AS d JOIN source s ON d.descendant = s.parent
)
SELECT *
FROM descendants
ORDER BY parent, level, descendant;
```



The Mob

- Tony Soprano
 - Larry Boy
 - Alfie Boy
 - Bobby Bacala
 - Paulie Walnuts
 - Chris Moltisanti
 - Alfie
 - Bissel
 - Murmur
 - Little Jimmie

Paginating

Counting

DELETING?

Reparenting

Updating?

Inserting


```

WITH RECURSIVE CTE AS (
    SELECT cc.*
           , nextval('erp_cost_codes_id_seq') AS new_id
           , ARRAY[cc.id] as path
           , false as cycle
    FROM cost_codes cc
    WHERE cc.biller_id = #{biller_id}
    AND cc.biller_type = '#{biller_type}'
    AND cc.parent_id IS NULL
    AND cc.deleted_at IS NULL
    UNION ALL
    SELECT cc.*
           , nextval('erp_cost_codes_id_seq') AS new_id
           , path || cc.id
           , cc.id = ANY(path)
    FROM CTE
    INNER JOIN cost_codes cc ON CTE.id = cc.parent_id
    WHERE cc.biller_id = #{biller_id}
    AND cc.biller_type = '#{biller_type}'
    AND cc.deleted_at IS NULL
    AND NOT cycle
)

INSERT INTO erp_cost_codes (id, parent_id, biller_id, biller_type, procore_cost_code_id, code, name, sortable_code, parent_sortable_code, standard_cost_code_id, created_at, updated_at)
SELECT C1.new_id as id
     , C2.new_id as parent_id
     , #{erp_biller_id || 'NULL'} AS biller_id
     , #{erp_biller_type ? "" + erp_biller_type + "" : 'NULL'} AS biller_type
     , C1.id as procore_cost_code_id
     , C1.code
     , C1.name
     , C1.sortable_code
     , C2.sortable_code as parent_sortable_code
     , C1.standard_cost_code_id
     , clock_timestamp() as created_at
     , clock_timestamp() as updated_at
    FROM CTE C1
LEFT JOIN CTE C2 on
    C1.parent_id = C2.id
ON CONFLICT DO NOTHING;

```

```

WITH RECURSIVE CTE AS (
    SELECT cc.*
        , nextval('erp_cost_codes_id_seq') AS new_id
        , ARRAY[cc.id] as path
        , false as cycle
    FROM cost_codes cc
    WHERE cc.biller_id = #{biller_id}
    AND cc.biller_type = '#{biller_type}'
    AND cc.parent_id IS NULL
    AND cc.deleted_at IS NULL
    UNION ALL
    SELECT cc.*
        , nextval('erp_cost_codes_id_seq') AS new_id
        , path || cc.id
        , cc.id = ANY(path)
    FROM CTE
    INNER JOIN cost_codes cc ON CTE.id = cc.parent_id
    WHERE cc.biller_id = #{biller_id}
    AND cc.biller_type = '#{biller_type}'
    AND cc.deleted_at IS NULL
    AND NOT cycle
)

INSERT INTO erp_cost_codes (id, parent_id, biller_id, biller_type, procore_cost_code_id, code, name, sortable_code, parent_sortable_code, standard_cost_code_id, created_at, updated_at)
SELECT C1.new_id as id
    , C2.new_id as parent_id
    , #{erp_biller_id || 'NULL'} AS biller_id
    , #{erp_biller_type ? "" + erp_biller_type + "" : 'NULL'} AS biller_type
    , C1.id as procore_cost_code_id
    , C1.code
    , C1.name
    , C1.sortable_code
    , C2.sortable_code as parent_sortable_code
    , C1.standard_cost_code_id
    , clock_timestamp() as created_at
    , clock_timestamp() as updated_at
FROM CTE C1
LEFT JOIN CTE C2 on
    C1.parent_id = C2.id
ON CONFLICT DO NOTHING;

```

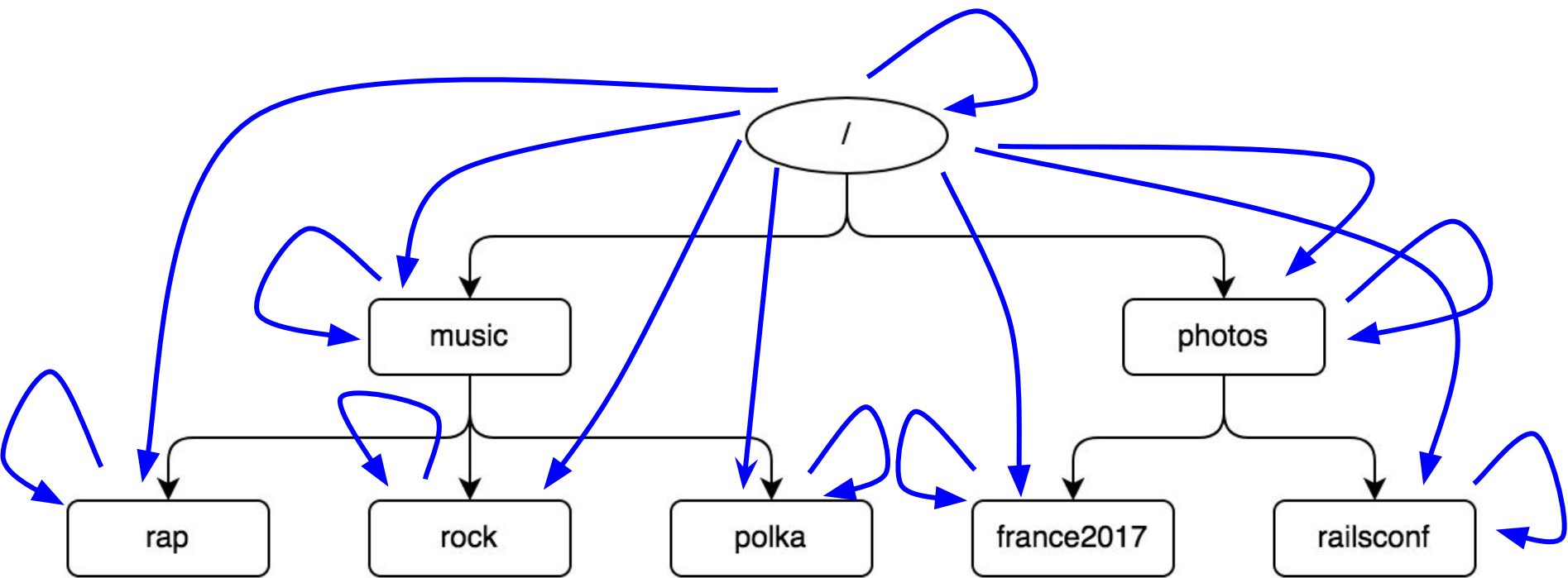
Naive Trees

- Simple
- Referential Integrity
- Disk Space

Naive Trees

- You don't need descendants
- You can cache (changes don't happen often)

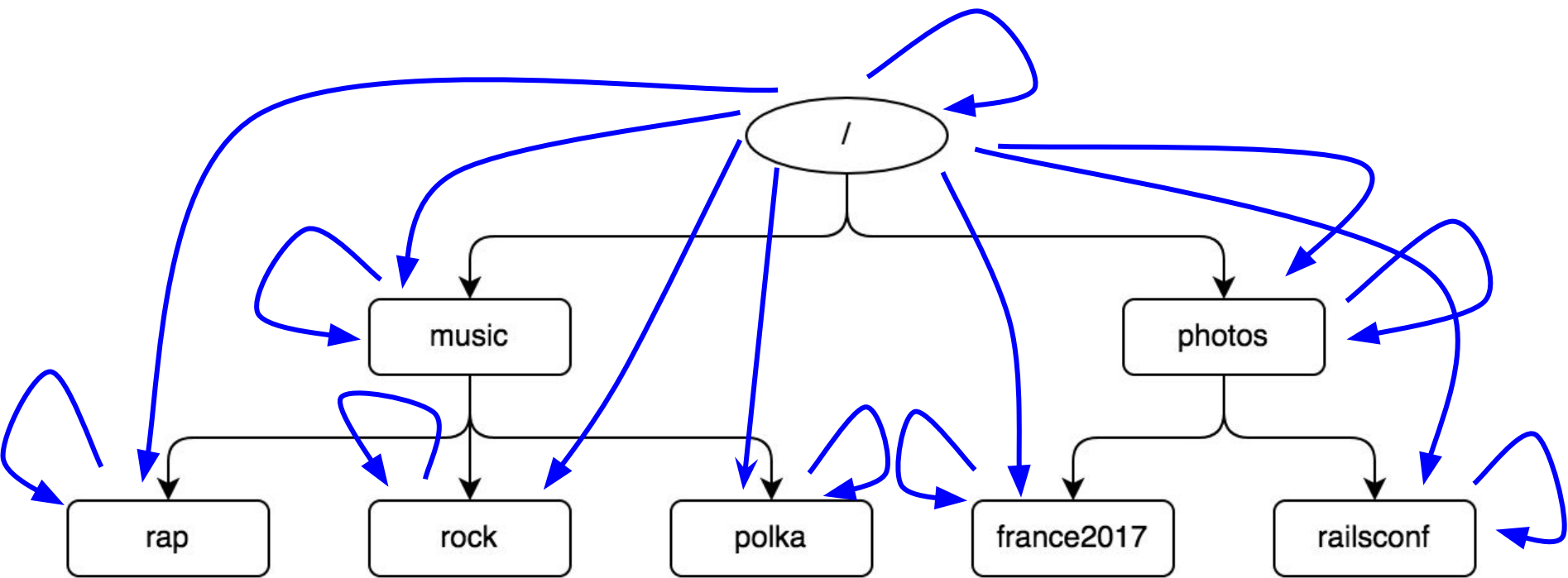
Closure Tree



id	parent_id	name
1	NULL	/
2	1	music
3	1	photos
4	2	rap
5	2	rock
6	2	polka

id	parent_id	name
1	NULL	/
2	1	music
3	1	photos
4	2	rap
5	2	rock
6	2	polka

ancestor_id	descendent_id
1	1
1	2
1	3
1	4
1	5
2	2
2	4
2	5
2	6
3	3
4	4
5	5
6	6



Descendents

```
SELECT f.*  
FROM folders f  
JOIN folder_trees ft ON f.id = ft.descendent_id  
WHERE t.ancestor_id = 2;
```

Performance:

- ancestors: 1 SELECT.
- descendants: 1 SELECT.
- siblings: 1 SELECT.

<> Code

Issues 44

Pull requests 3

Projects 0

Wiki

Insights

Easily and efficiently make your ActiveRecord models support hierarchies <https://closuretree.github.io/closure...>

closure-tree

ruby

nested-hashes

tree-structure

rails

activerecord

ancestry

nested-set

hierarchy

hierarchies

descendants

parent

child

870 commits

12 branches

80 releases

51 contributors

MIT

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download



mceachen prep v6.6.0

Latest commit 9baa885 on Jul 11

gemfiles	prep 6.5.0, add ar 5.1 appraisal	4 months ago
img	added docs for digraph and updated to 4.1.0.rc1	4 years ago
lib	prep v6.6.0	2 months ago
spec	Additional specificity for Rails 5.1	2 months ago
.gitignore	Codacy	11 months ago
.rspec	don't rebuild in the middle of prepending siblings.	3 years ago
.travis.yml	prep 6.5.0, add ar 5.1 appraisal	4 months ago
.yardopts	switch from rdoc to yard	6 years ago
Appraisals	prep 6.5.0, add ar 5.1 appraisal	4 months ago
CHANGELOG.md	prep v6.6.0	2 months ago
Gemfile	Bye bye rails 4.0	2 years ago
MIT-LICENSE	* version bump	a year ago
README.md	s/mceachen/ClosureTree/	2 months ago

```
class Employee < ActiveRecord::Base  
  has_closure_tree  
end
```

ancestors
descendants
parent
children
root
leaves
depth

```
Person.find_or_create_by_path([  
    {name: 'Grandparent'},  
    {name: 'Parent'},  
    {name: 'Child'}  
])
```

Mutation:

- create: 2 INSERTs
- reparent: 3 INSERT/UPDATES

closure_tree (the gem)

- Speed
- No recursion
- Referential integrity
- Single table inheritance
- Deterministic ordering
- MySQL, PostgreSQL, and SQLite
- Easy migration

closure_tree

- No pgbouncer (concurrency is hard)
- Possibly not as fast as...

Materialized Path

id	path	name
1	1/	/
2	1/2/	music
3	1/3/	photos
4	1/2/4/	rap
5	1/2/5/	rock
6	1/2/6/	polka

Descendents

```
SELECT *  
FROM folders f  
WHERE f.path LIKE '1/4/%';
```

<> Code

! Issues 86

🔗 Pull requests 17

📁 Projects 0

📖 Wiki

Insights ▾

Organise ActiveRecord model into a tree structure

🕒 321 commits

🔗 5 branches

📦 11 releases

👤 49 contributors

📄 MIT

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾

 **kbrock** committed on **GitHub** Merge pull request [#357](#) from tilsammans/readme-generate-index-migration ... Latest commit 3b042ad 19 days ago

📁 gemfiles	Dropping official support for rails 4.0 and 4.1	4 months ago
📁 lib	Fix order_by_ancestry_and	a month ago
📁 test	mac specific tests	a month ago
📄 .coveralls.yml	Fixing coveralls	4 years ago
📄 .gitignore	Ignore locked gemfiles	2 years ago
📄 .travis.yml	Add ruby 2.4 support in travis.	4 months ago
📄 Appraisals	Dropping official support for rails 4.0 and 4.1	4 months ago
📄 CHANGELOG.md	bump 3.0.1	2 months ago
📄 Gemfile	ruby 1.9.3 support	11 months ago
📄 MIT LICENSE	Update copyright date	a year ago

ancestry

- Limits ~ 23 level deep
- No referential integrity

Ancestors x 10 runs

ancestry	0.01
----------	------

closure_tree	0.03
--------------	------

Descendents x 10 runs

ancestry	0.014
----------	-------

closure_tree	0.02
--------------	------

But...

`descendants.count`

Closure tree is ~175X faster!

Paginating

Counting

DELETING?

Reparenting

Updating?

Inserting

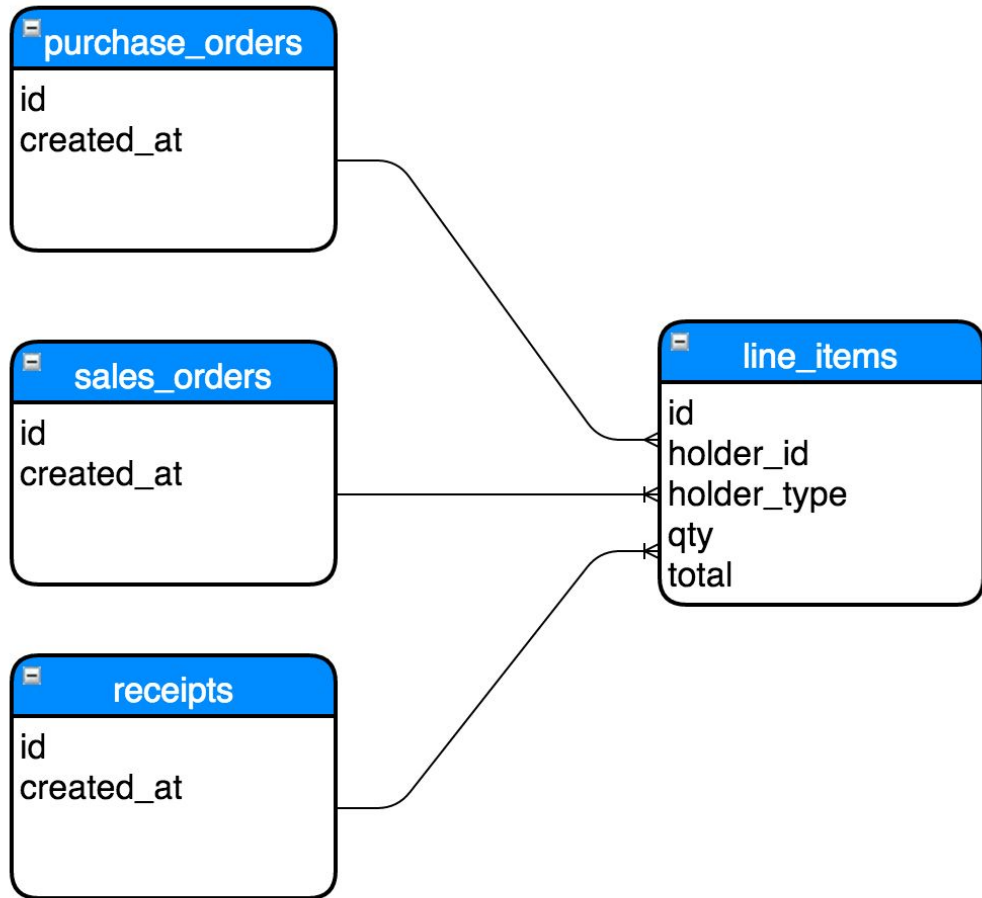
A man in a brown hoodie is shown in profile, focused on his work. He is sitting at a desk, typing on a laptop. The background is dark and filled with floating binary code (0s and 1s) and glowing rectangular shapes, suggesting a digital or data environment. A bright light emanates from the laptop screen.

**Focus on
the reads!**

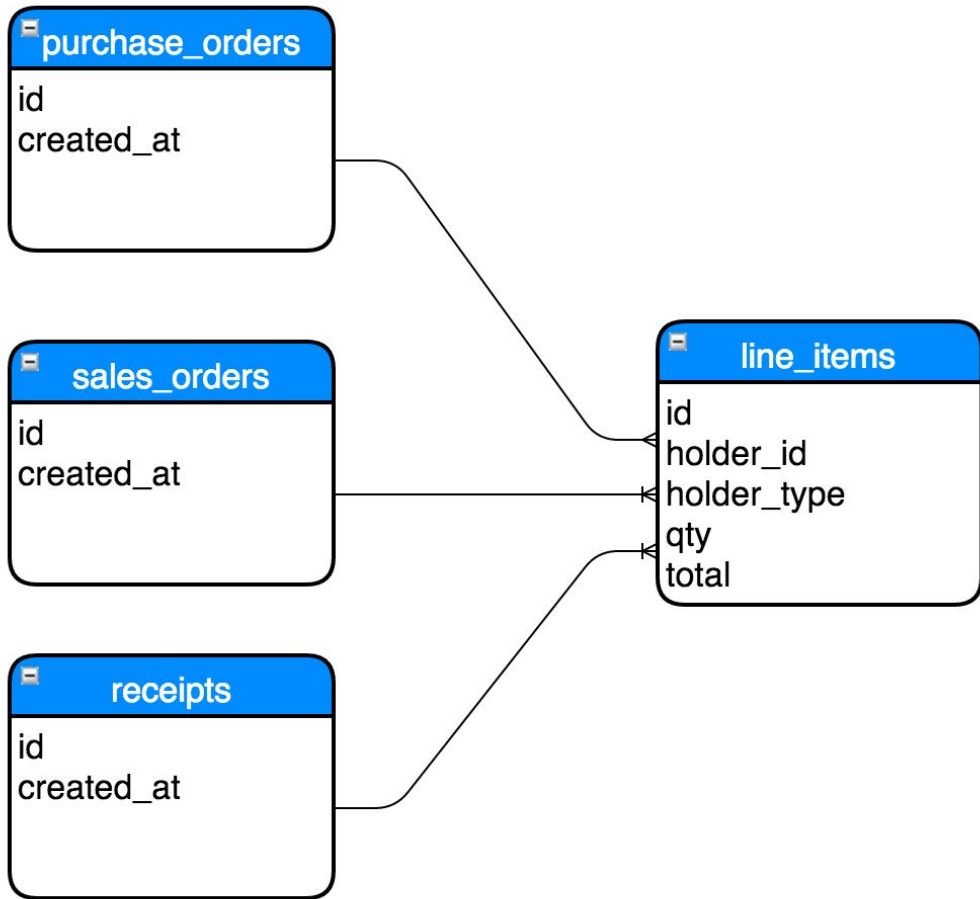


(Anti)-Pattern

Polymorphic Relationship

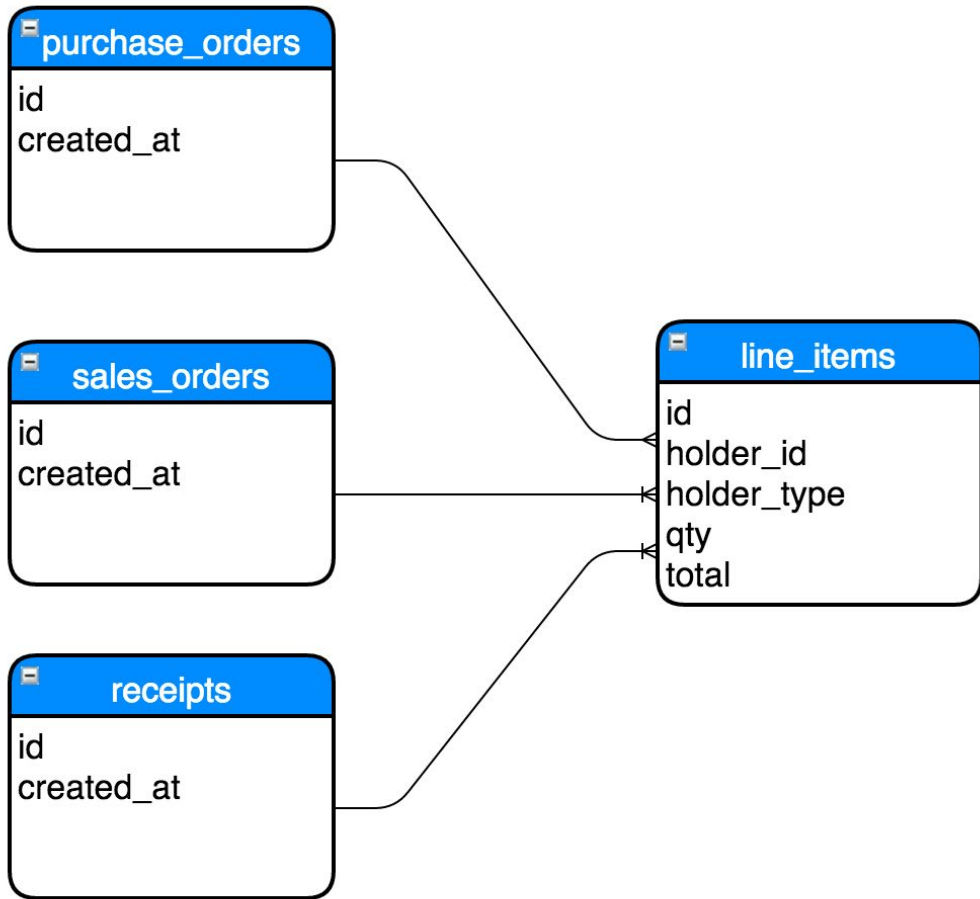


id	holder_id	holder_type
1	1	SalesOrder
2	1	SalesOrder
3	1	SalesOrder
4	2	PurchaseOrder
5	2	PurchaseOrder
6	2	PurchaseOrder



Advantages

- Built into Rails
- Easy to add more

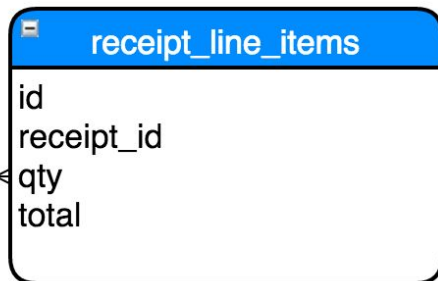
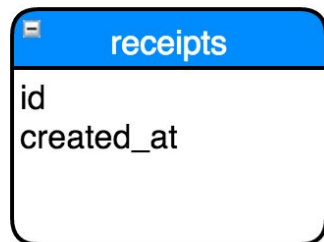
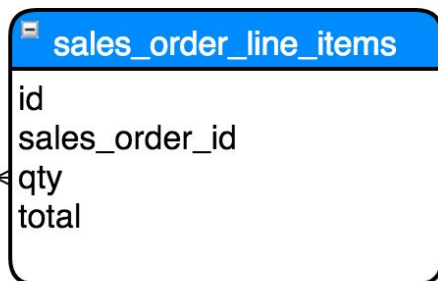
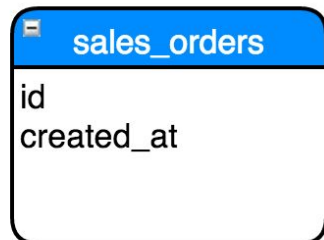
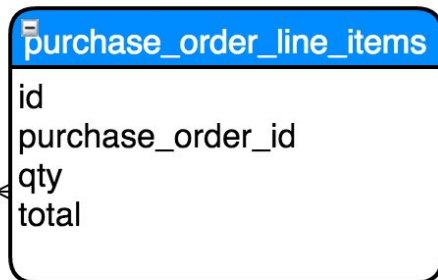
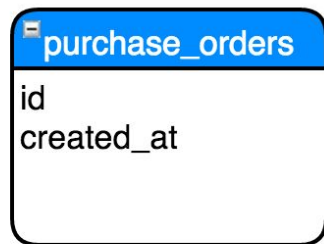


Disadvantages

- **No foreign key**
- No cascading delete
- Confusing column names
- 2 join conditions
- `holder_type` is extra data

Separate Tables

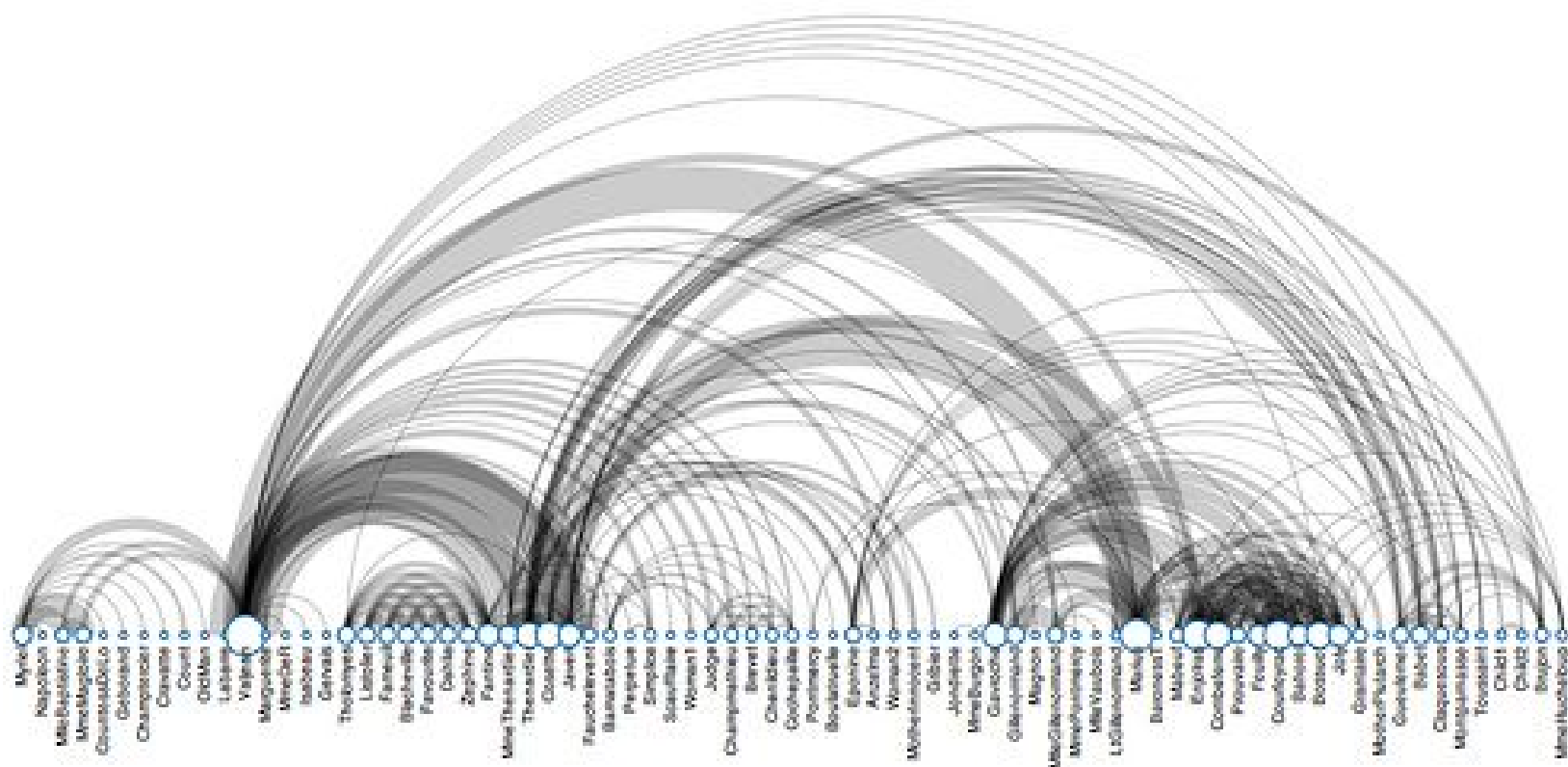


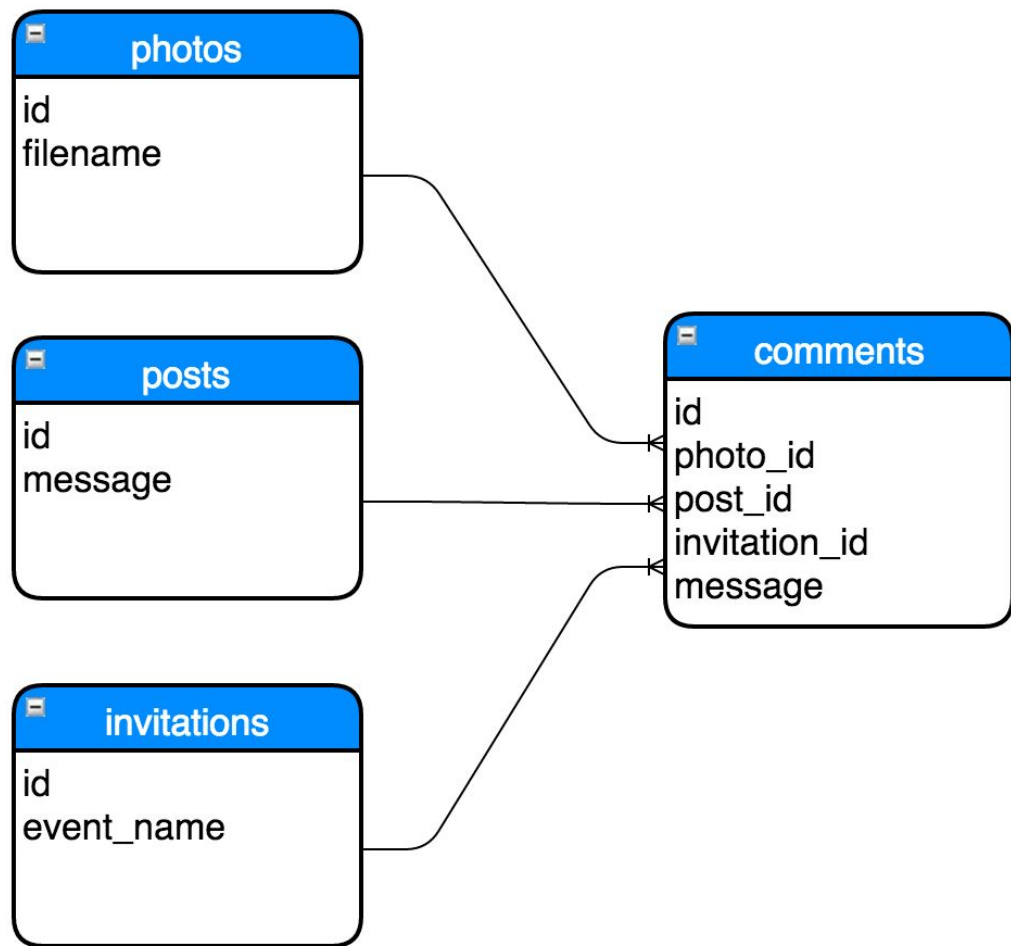




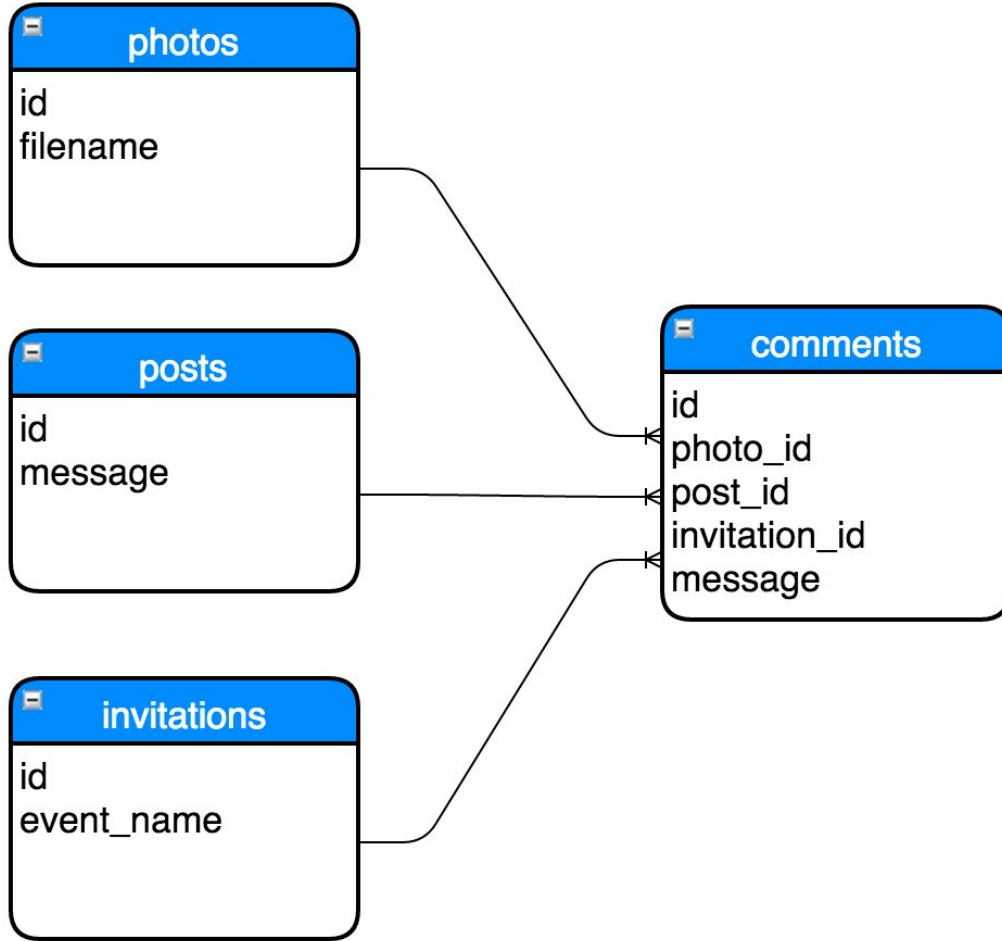
It's OK
to not
be DRY

Exclusive Arcs



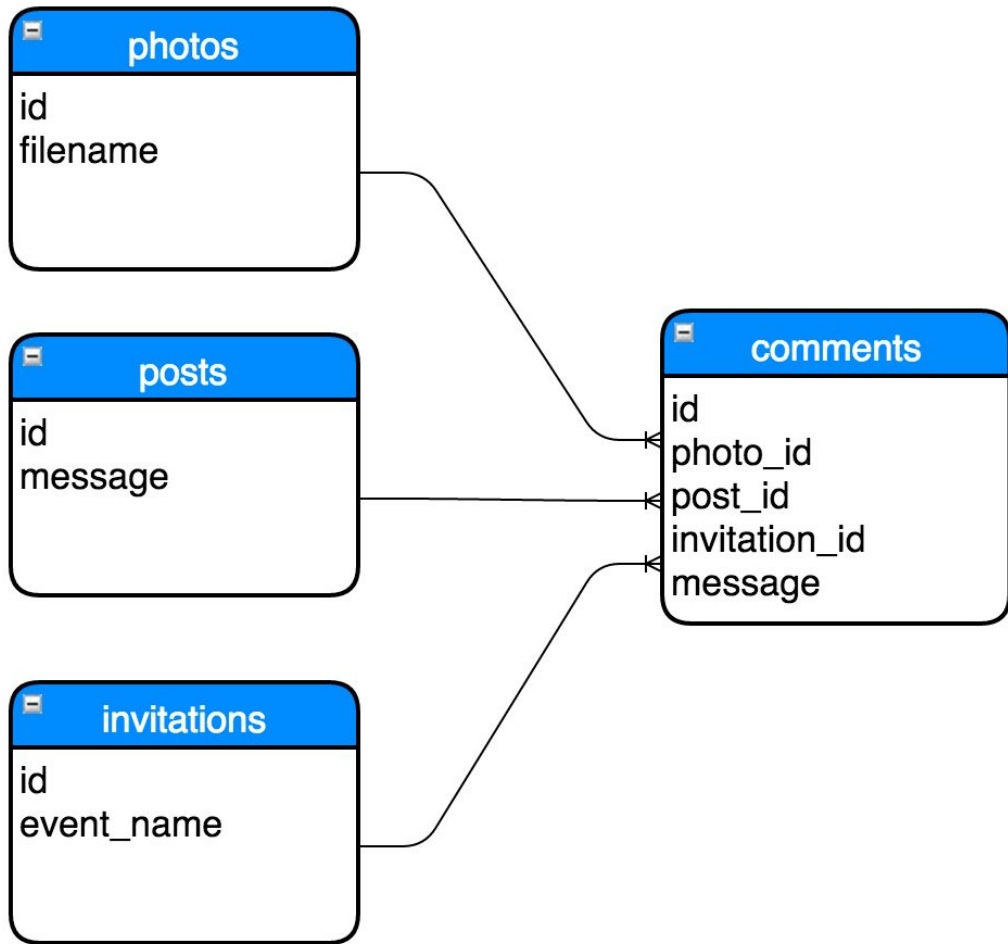


id	comment	post_id	photo_id	invitation_id
1 ...		1	NULL	NULL
2 ...		2	NULL	NULL
3 ...		NULL	NULL	18
4 ..		NULL	NULL	18
5 ...		NULL	NULL	21
6 ...		NULL	4	NULL



Advantages

- **Foreign key constraints**
- Cascading delete

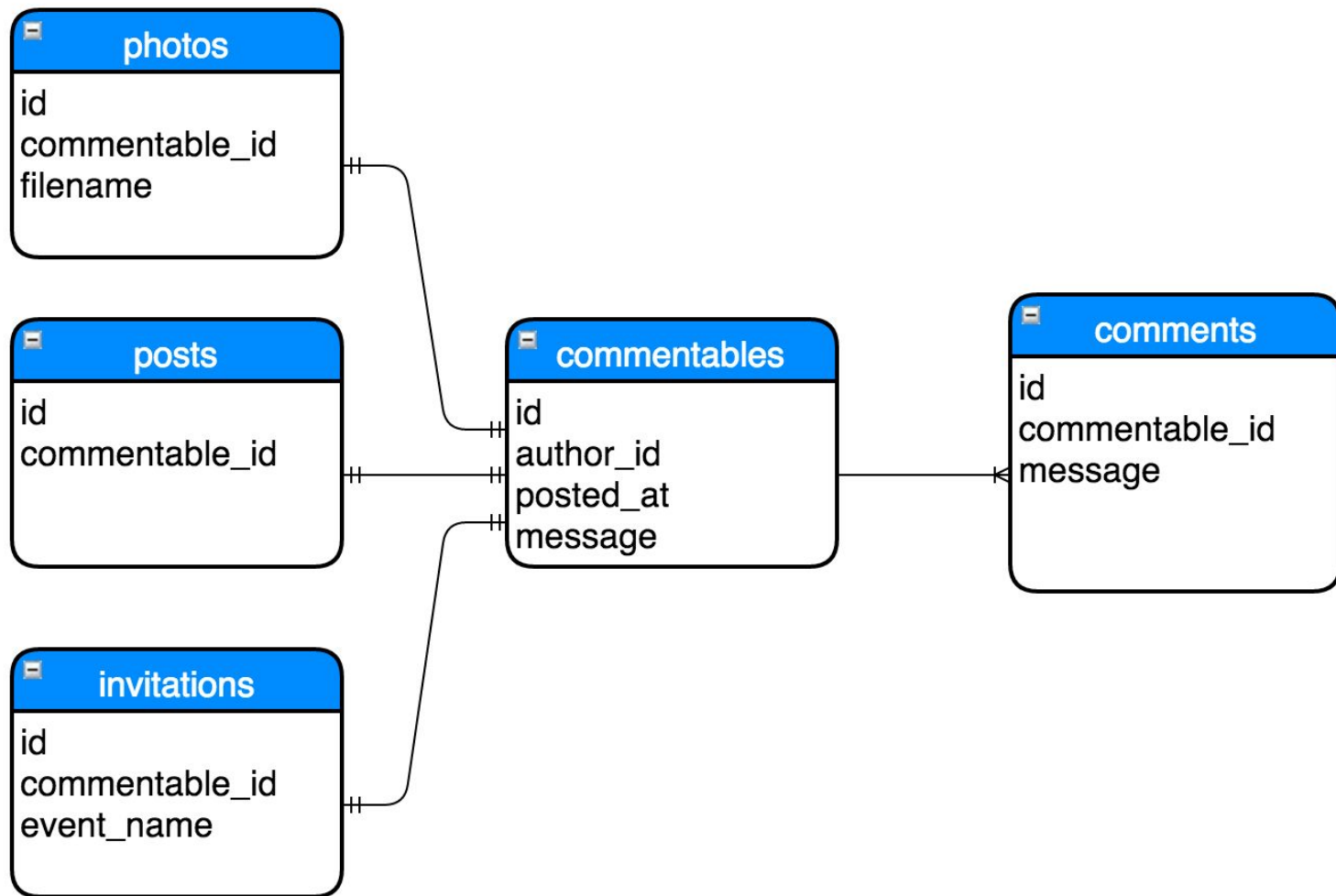


Disadvantages

- Possible mistakes
- More migrations

Base Parent Table



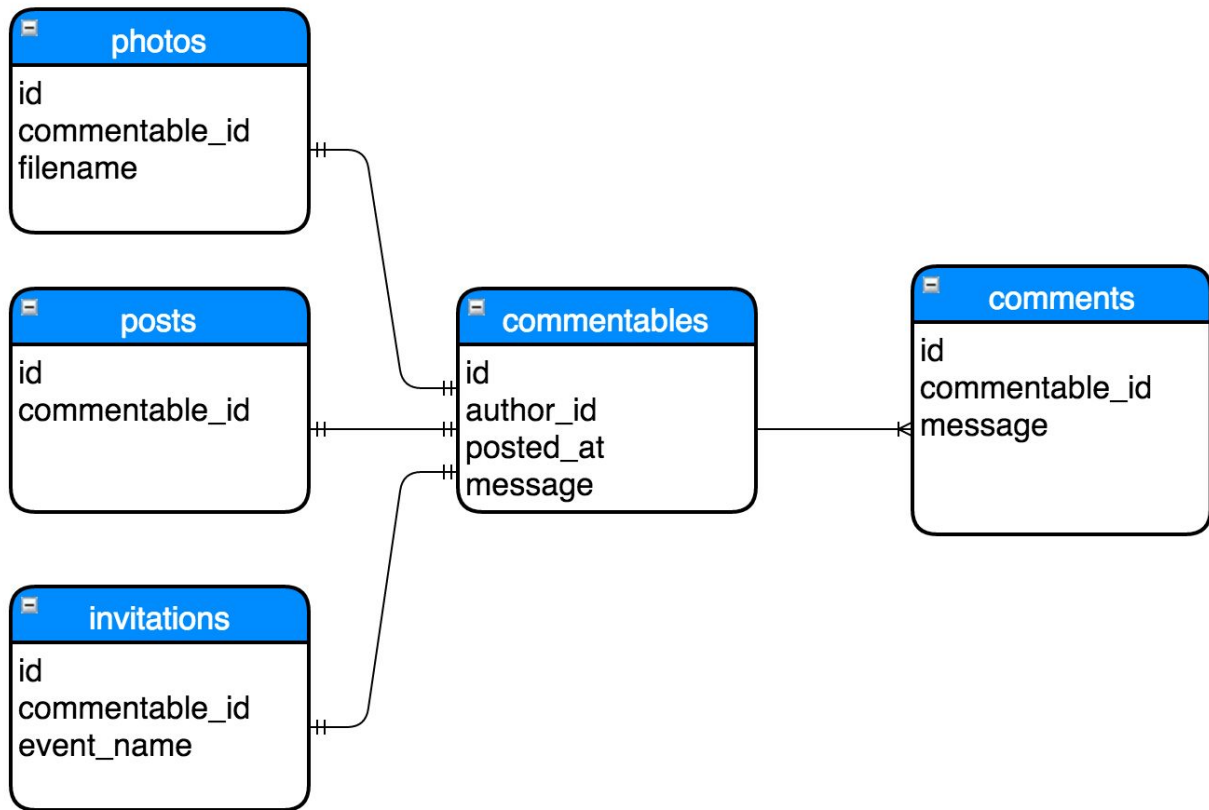




1 to 1

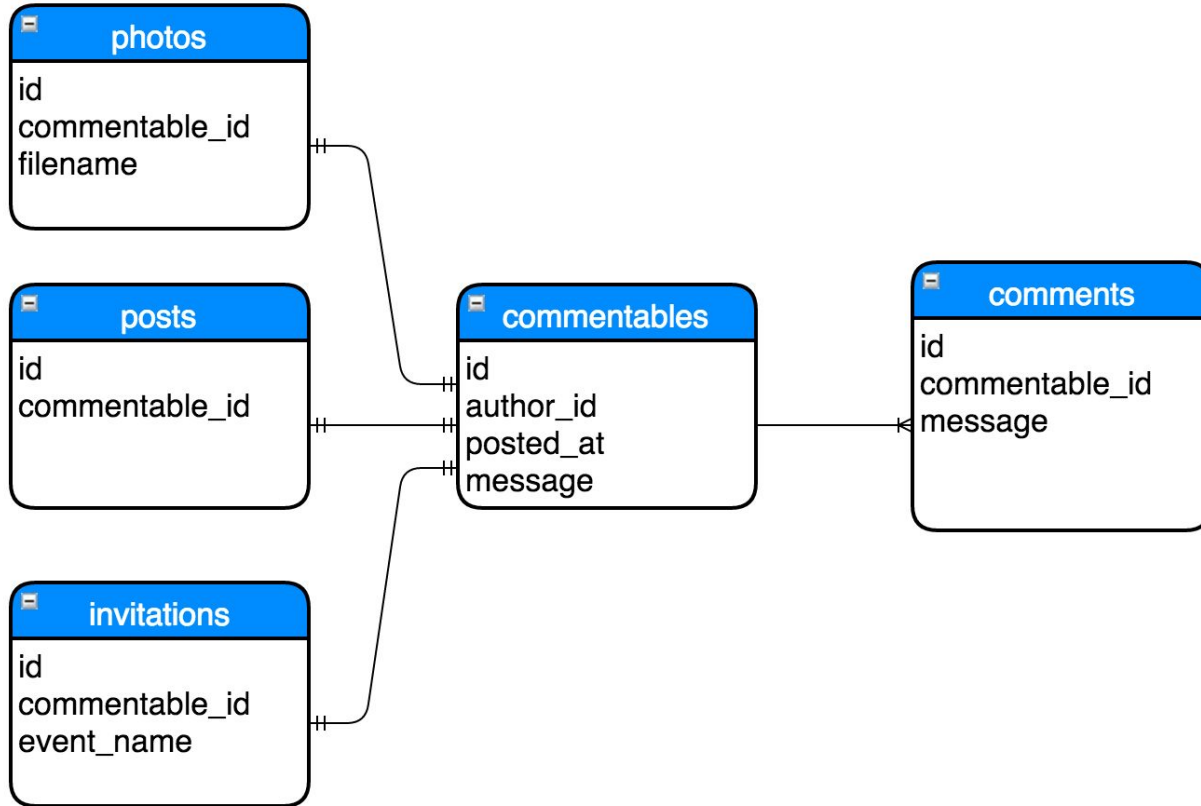
1 to Many





Advantages

- No mistakes
- DRY



Disadvantages

- 2 Joins
- 1-way street
- Errors still possible

Dozens of possible parent types....

Dozens of possible parent types....

POLYMORPHIC RELATIONSHIP

Simple and *mostly* safe...

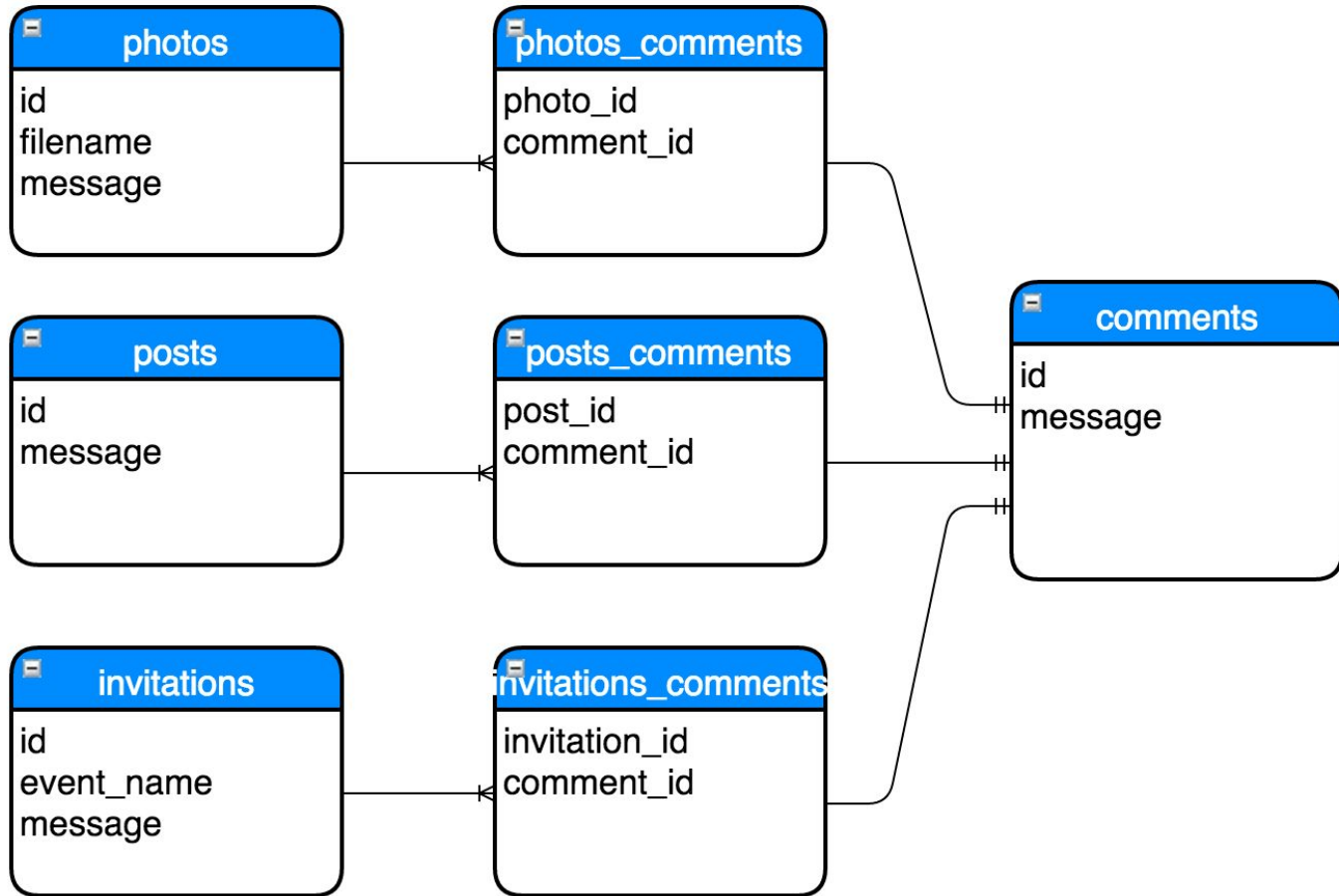
Simple and *mostly* safe...

EXCLUSIVE ARCS

Completely safe...

Completely safe...

BASE PARENT TABLE





Performance vs. Simplicity
Correctness vs. Convenience

MAKE
GOOD
CHOICES

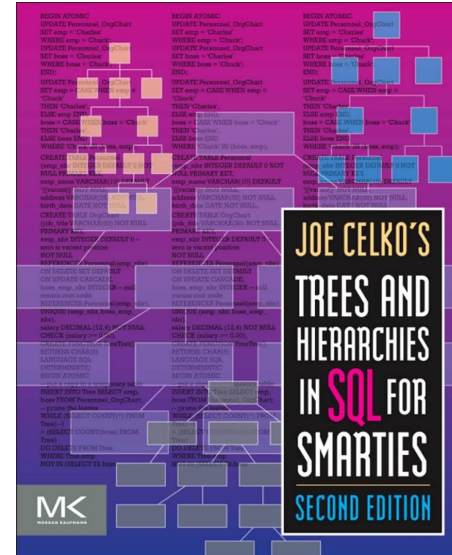
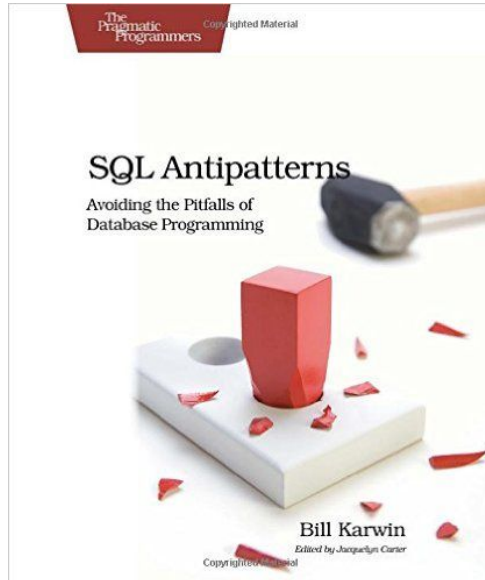


A photograph of a seal lying on its back on a sandy beach. The seal's mouth is wide open, showing its teeth and tongue, as if it is yawning or laughing. A blue speech bubble is superimposed over the seal's head and upper body, containing the text "I avoided recursive SQL Hell!". The background shows the ocean with gentle waves and a cloudy sky.

I avoided
recursive
SQL Hell!

Resources

- http://www.hilman.io/blog/2015/09/comparing-ancestry-and-closure_tree/



Who am I?

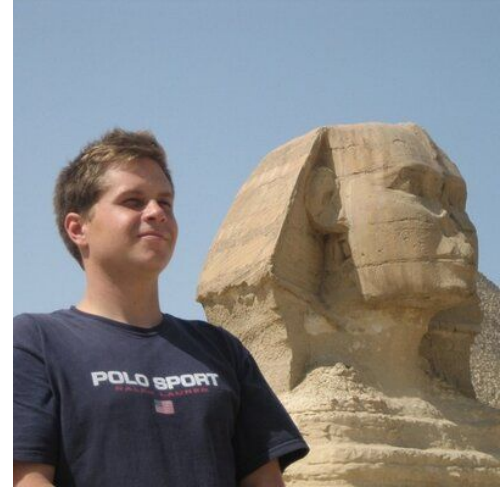
I tweet at



@BradUrani



linkedin.com/in/bradurani



I work in Santa Barbara at







