

Tech debt

58 Issues - 11 Points

Live #1899

Upgrade 2 RDS 9.6 instances before Jan 18, 2022

tech debt

Live #2019

Investigate using changesets for publishing shared packages

tech debt

Live #2022

Reuse new modal component in Live CMS

Live CMS changes

tech debt

Live #2013

LiveblogContainer: Move out processing logic

tech debt

Live #2015

Use translations in the new settings form

Rewrite edit settings pane to R...

tech debt

Live #2007

Critical security patches for RDS-es

tech debt

Comments #1383

Resolve fasten-static dependency in comments-front

tech debt

Comments #1176

Opinion service terraform

Product metrics

0 Issues - 0 Points

Bugs

59 Issues - 6 Points

Live #2021

Tags do not work for logged in users

bug

Live #2024

live-cms: broken user messages stories

bug

tech debt

Live #2023

Deleting 'Intro Title' or 'Simple Intro text' does not remove it from the front end

bug

Live #2016

Blank circle next to journalist name (pinned entry)

bug

Live #1977

Comments breaking on Linkify malformed URI

bug

Comments #1374

Remove stars :focus outline on click

bug

Live #1960

Users are able to post messages when liveblog is closed

bug

Live #1850

Scrolling and content-loading problems on sports liveblogs

Backlog

190 Issues - 41 Points

Live #2020

Edit entry should edit feeds too

Live CMS changes

Live #1964

Split settings into liveblog and feed

Sprint #21

Rewrite edit settings pane to R...

Live #1955

Test feeds on production before old liveblogs migration

Sprint #21

Backend changes

Live #2018

Rename fallback rack

Live migration to Convex

Live #2017

Check if data is present on datadog dashboard

Prepare deployment step for a...

Live #2011

Setup alerts

Live migration to Convex

Live #2010

Cleanup of Heroku and terraform

Live migration to Convex

Live #2009

Setup autoscaling for services

Live migration to Convex

Comments #1369

Check if we have budget alerts on AWS for comments

Sprint #18

Waiting

1 Issue - 0 Points

Live #1969

Add log in requirement to 'see more' button

Sprint #21

Todo

2 Issues - 0 Points

High priority

Comments #1382

Sprint #21 theme

Sprint #21

Live #1978

Analyze Supernytt data

Sprint #21

Migration of data

In Progress

3 Issues - 0 Points

Live #1963

Use content editor for liveblog summary

Sprint #21

Rewrite edit settings pane to R...

Live #1986

Prepare Convex deployment for live-message-consumer

Sprint #21

Prepare deployment step for a...

tech debt

Live #1944

(Step 4) Adjust live-components to use feed settings along with liveblog settings for backward compatibility

Sprint #21

Live component changes

In Review

5 Issues - 0 Points

Live #1963

Make with f

Sprint

Live C

Live #1963

Article not w

Sprint

Live #1963

(Step a migr entry-

Sprint

Backe

Live #1963

(Step delete

Sprint

Live co

Live #1963

(Step user n

Sprint

Live C

What and who?

What and who?

- Modeling problems in a relational database

What and who?

- Modeling problems in a relational database
- SQL, constraints, transactions...

What and who?

- Modeling problems in a relational database
- SQL, constraints, transactions...
- examples from **Postgres**


What and who?

- Modeling problems in a relational database
- SQL, constraints, transactions...
- examples from Postgres
- beginners and advanced

Settings

Livefeed

Livefeed is open










Marcin Kasprowicz

Add title

0/100

Write text here

NEWS VALUE

NEWS LIFETIME

SHORT


MID

LONG


MEMES :)
 IN POLISH
 IN ENGLISH

Memes :)

Send meme




i gâr 18:49

DEL
 


NV: 0.5
 VISIBLE ONLY FOR EDITORS

Pin
 Edit
 Delete




Taki Michau
 Złoty jest tak słaby, że 10 zł ledwo dycha

10



i gâr 18:45


DEL
 

NV: 0.5
 VISIBLE ONLY FOR EDITORS

Pin
 Edit
 Delete

W całej Polsce:
- Gdzie patrzy?

Na Śląsku:



Twitter

Messages

0

Autoload new messages

New (0)
 Published (4)

No user messages to display...

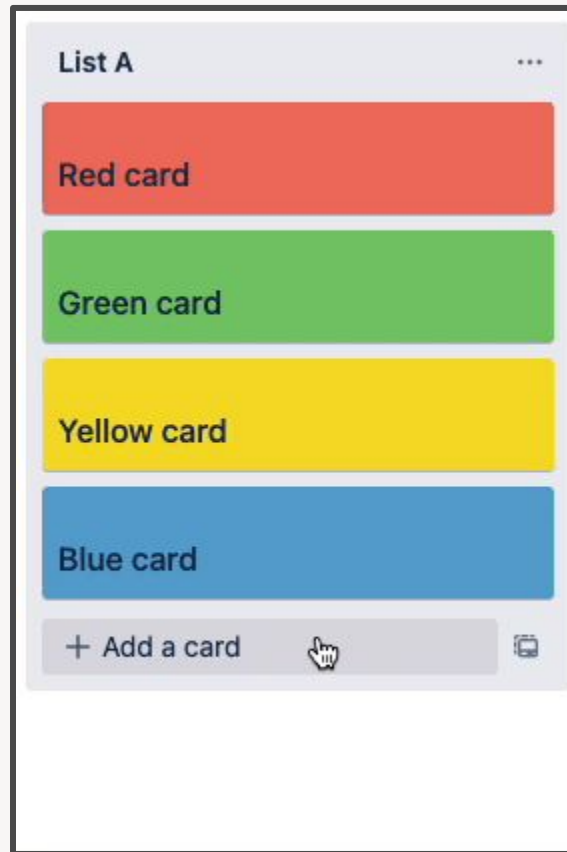
Add guest editor

External feeds

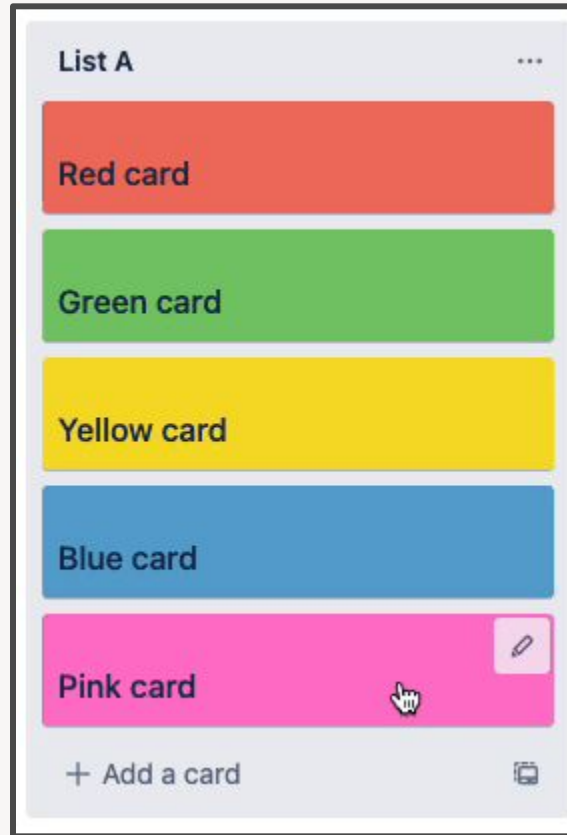


As a user.....

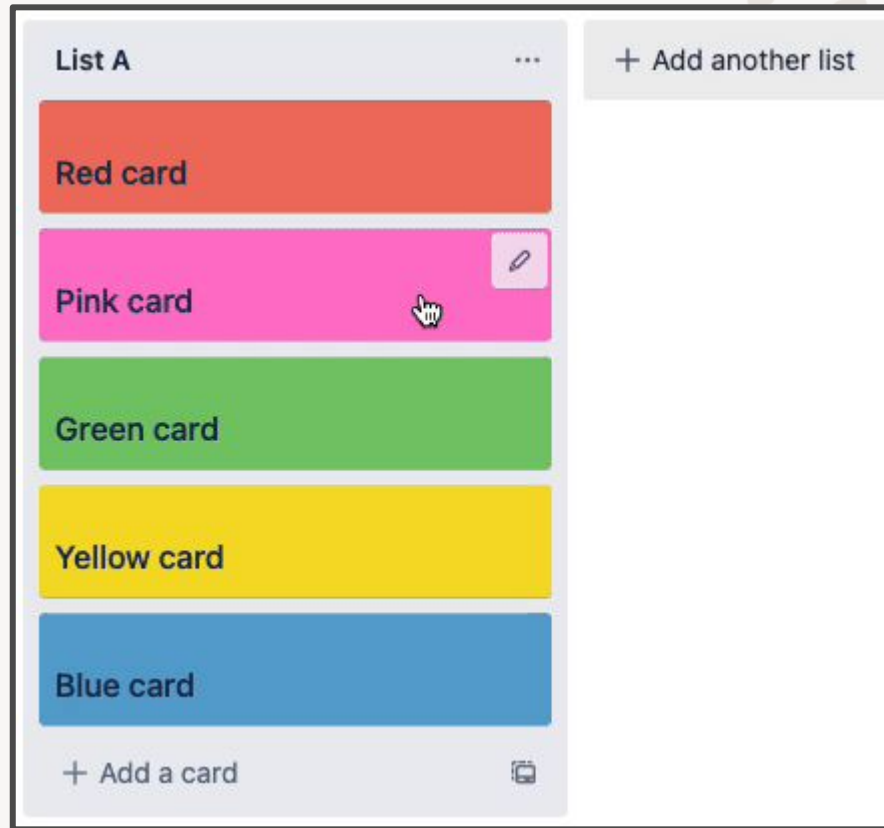
As a user, I can add
a new card



As a user, I can
rearrange cards



As a user, I can
remove a card





How to model it?

Identifiers in a parent - array

```
# lists table
```

id	cards
A	[red, blue, green]

```
# cards table
```

id	description
red	blah
blue	As I user...
green	foo bar

Identifiers in a parent - array

```
# lists table
```

id	cards
A	[red, blue, green]

```
# cards table
```

id	description
red	blah
blue	As I user...
green	foo bar

What is wrong with that?

Auxiliary attribute - placement

```
# lists table
```

id

A

```
# cards table
```

id	description	list_id	placement
red	blah	A	1
blue	As I user...	A	2
green	foo bar	A	3

Auxiliary attribute - placement

```
# lists table
```

id
A

```
# cards table
```

id	description	list_id	placement
red	blah	A	1
blue	As I user...	A	2
green	foo bar	A	3

What is wrong with that?

Trying to be smart - rank

```
# lists table
```

id

A

```
# cards table
```

id	description	list_id	rank
red	blah	A	100
blue	As I user...	A	200
green	foo bar	A	300

Trying to be smart - rank

lists table

id
A

cards table

id	description	list_id	rank
red	blah	A	100
blue	As I user...	A	200
green	foo bar	A	300

id	description	column_id	rank
red	blah	A	100
green	foo bar	A	150
blue	As I user...	A	200



Trying to be smart - rank


```
# lists table
```

id
A

```
# cards table
```

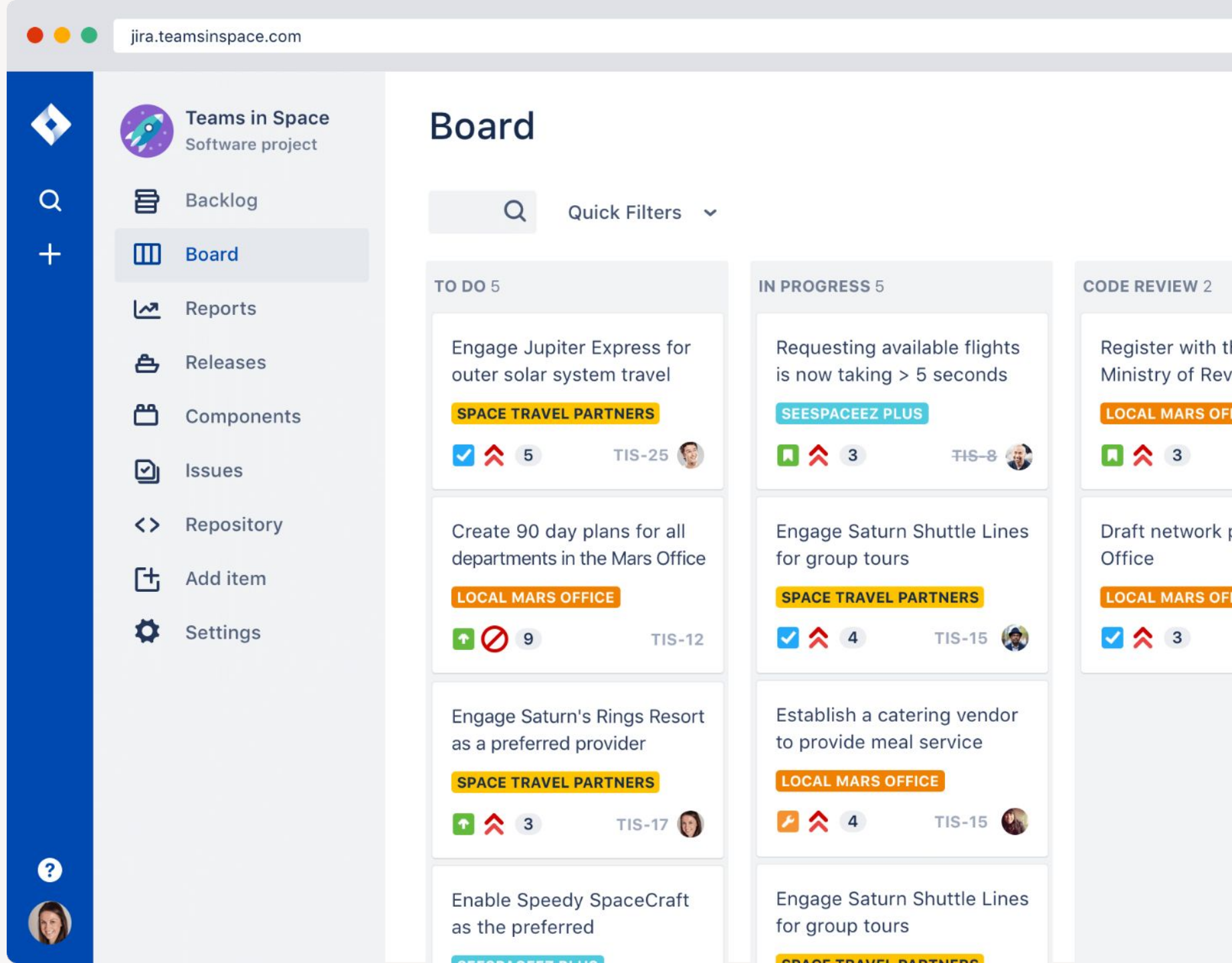
id	description	list_id	rank
red	blah	A	100
blue	As I user...	A	200
green	foo bar	A	300

id	description	column_id	rank
red	blah	A	100
green	foo bar	A	150
blue	As I user...	A	200



What is wrong with that?

lexorank



Winner?

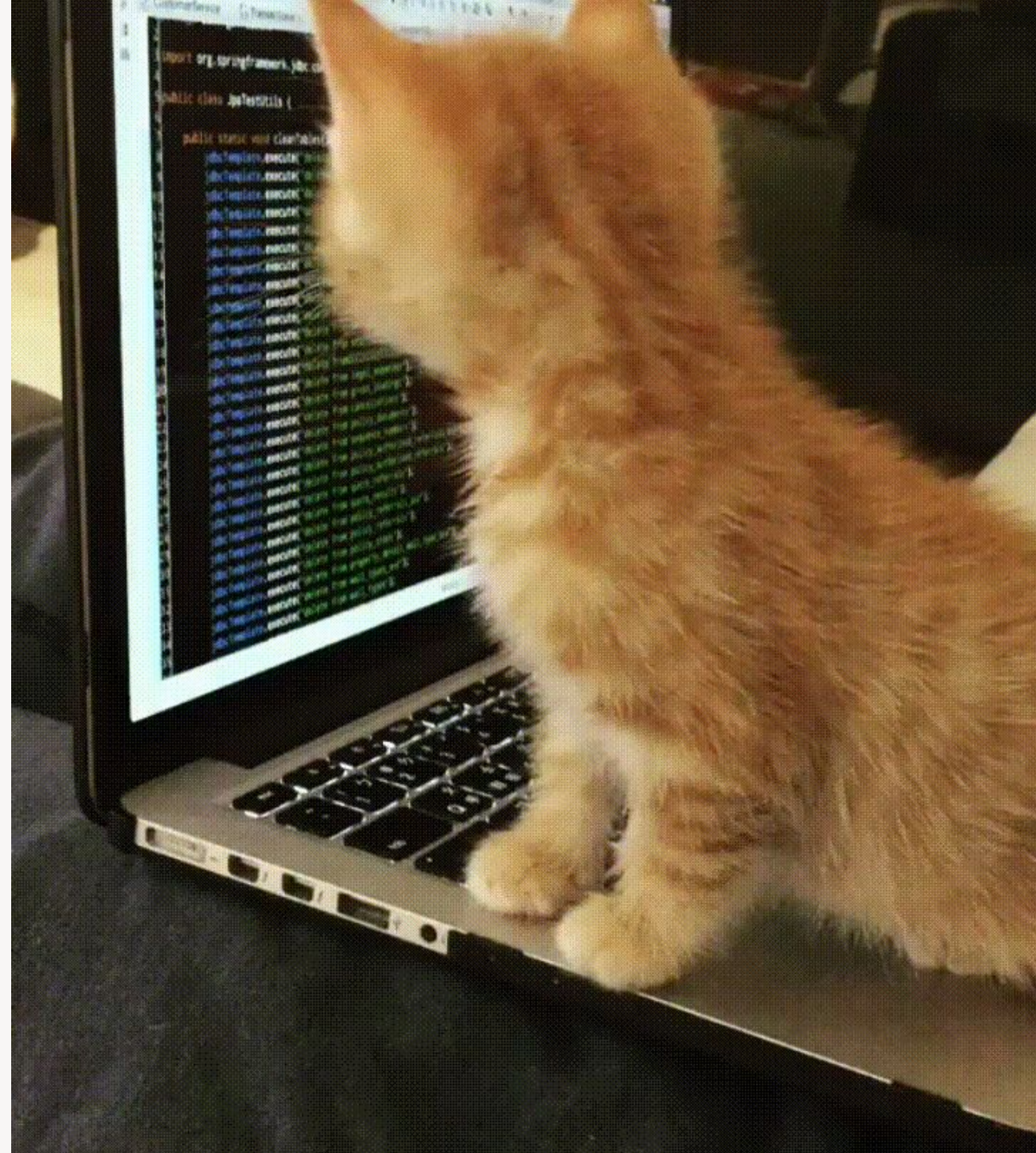
- 5 cards are absolute maximum,
2-3 commonly
- rare rearrangements

Winner?

- 5 cards are absolute maximum,
2-3 commonly
- rare rearrangements

placement

Implementation



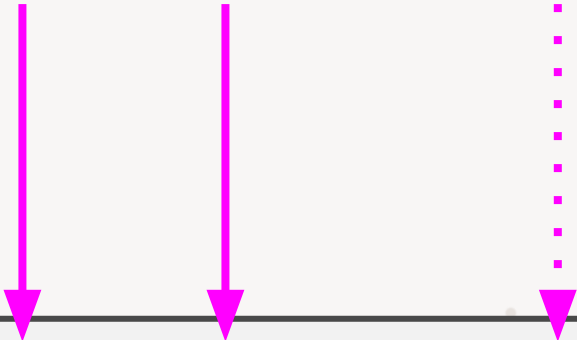
Schema

id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me	A	4	null

Deleting...



Schema



id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me	A	4	null

```
CREATE UNIQUE INDEX cards_listid_placement_deletedat_unique  
ON cards (list_id, placement)  
WHERE deleted_at IS NULL
```

Insert

id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me	A	4	null

```
INSERT INTO cards (id, list_id, placement)
SELECT
  :id,
  :list_id,
  (SELECT MAX(placement) + 1 FROM cards WHERE list_id = :list_id AND deleted_at IS NULL)
```

Delete

id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me	A	4	null

```
WITH placement_of_removed AS (  
  SELECT placement  
  FROM cards  
  WHERE id = :id AND deleted_at IS NULL  
)  
UPDATE cards  
SET  
  placement = (  
    CASE WHEN placement > (SELECT * FROM placement_of_removed)  
    THEN placement - 1 🙌  
    ELSE placement 🙌  
  )  
  ,  
  deleted_at = (  
    CASE WHEN id = :id  
    THEN NOW() 🗑️  
    ELSE NULL  
  )  
)  
WHERE list_id = :list_id AND deleted_at IS NULL
```

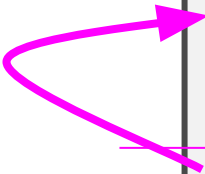
Rearrange

```
WITH helper AS (  
  SELECT  
    placement AS old_placement,  
    CASE  
      WHEN placement = :new_placement THEN 0  
      WHEN placement > :new_placement THEN 1  
      ELSE -1  
    END AS direction  
  FROM cards  
  WHERE id = :id  
)  
  
UPDATE cards  
  
SET  
  placement = (  
    CASE  
      WHEN id = :id  
        THEN :new_placement  
      WHEN placement  
        BETWEEN  
          LEAST(old_placement, :new_placement)  
          AND  
          GREATEST(old_placement, :new_placement)  
        THEN placement + direction  
      ELSE  
        placement  
    END  
  )  
  
FROM cards c JOIN helper h ON TRUE  
WHERE list_id = :list_id AND deleted_at IS NULL
```

id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me	A	4	null

Rearrange


```
WITH helper AS (  
  SELECT  
    placement AS old_placement,  
    CASE  
      WHEN placement = :new_placement THEN 0 🖐️  
      WHEN placement > :new_placement THEN 1 🖐️  
      ELSE -1 🖐️  
    END AS direction  
  FROM cards  
  WHERE id = :id  
)  
  
UPDATE cards  
  
SET  
  placement = (  
    CASE  
      WHEN id = :id  
        THEN :new_placement  
      WHEN placement  
        BETWEEN  
          LEAST(old_placement, :new_placement)  
          AND  
          GREATEST(old_placement, :new_placement)  
        THEN placement + direction  
      ELSE  
        placement  
    END  
  )  
  
FROM cards c JOIN helper h ON TRUE  
WHERE list_id = :list_id AND deleted_at IS NULL
```




id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me	A	4	null

Rearrange

```
WITH helper AS (  
  SELECT  
    placement AS old_placement,  
    CASE  
      WHEN placement = :new_placement THEN 0 🖐️  
      WHEN placement > :new_placement THEN 1 🖐️  
      ELSE -1 🖐️  
    END AS direction  
  FROM cards  
  WHERE id = :id  
)  
  
UPDATE cards  
  
SET  
  placement = (  
    CASE  
      WHEN id = :id  
        THEN :new_placement  
      WHEN placement  
        BETWEEN  
          LEAST(old_placement, :new_placement)  
          AND  
          GREATEST(old_placement, :new_placement)  
        THEN placement + direction  
      ELSE  
        placement  
    END  
  )  
  
FROM cards c JOIN helper h ON TRUE  
WHERE list_id = :list_id AND deleted_at IS NULL
```



old_placement	direction
4	1



id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me	A	4	null

Rearrange

```
WITH helper AS (  
  SELECT  
    placement AS old_placement,  
    CASE  
      WHEN placement = :new_placement THEN 0 🖐️  
      WHEN placement > :new_placement THEN 1 🖐️  
      ELSE -1 🖐️  
    END AS direction  
  FROM cards  
  WHERE id = :id  
)  
  
UPDATE cards  
  
SET  
  placement = (  
    CASE  
      WHEN id = :id  
        THEN :new_placement  
      WHEN placement  
        BETWEEN  
          LEAST(old_placement, :new_placement)  
          AND  
          GREATEST(old_placement, :new_placement)  
        THEN placement + direction  
      ELSE  
        placement  
    END  
  )  
  
FROM cards c JOIN helper h ON TRUE  
WHERE list_id = :list_id AND deleted_at IS NULL
```

old_placement	direction
4	1

id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me	A	4	null

id	list_id	placement	old_placement	direction
red	A	1	4	1
blue	A	2	4	1
green	A	3	4	1
black	A	4	4	1

Rearrange

```
WITH helper AS (  
  SELECT  
    placement AS old_placement,  
    CASE  
      WHEN placement = :new_placement THEN 0 🖐️  
      WHEN placement > :new_placement THEN 1 🖐️  
      ELSE -1 🖐️  
    END AS direction  
  FROM cards  
  WHERE id = :id  
)  
  
UPDATE cards  
  
SET  
  placement = (  
    CASE  
      WHEN id = :id  
        THEN :new_placement  
      WHEN placement  
        BETWEEN  
          LEAST(old_placement, :new_placement)  
          AND  
          GREATEST(old_placement, :new_placement)  
        THEN placement + direction  
      ELSE  
        placement  
    END  
  )  
  
FROM cards c JOIN helper h ON TRUE  
WHERE list_id = :list_id AND deleted_at IS NULL
```

old_placement	direction
4	1

id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me	A	4	null

id	list_id	placement	old_placement	direction
red	A	1	4	1
blue	A	2	4	1
green	A	3	4	1
black	A	4	4	1

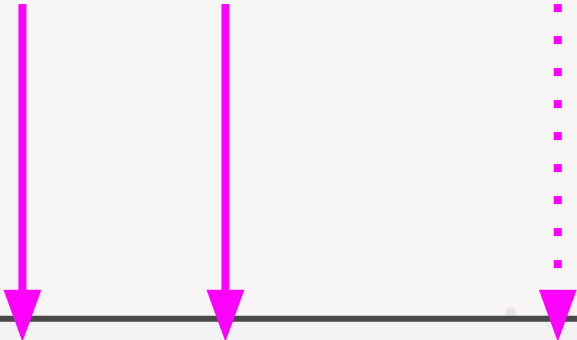
id	list_id	placement
red	A	(+1) 2
blue	A	(+1) 3
green	A	(+1) 4
black	A	(:new_placement) 1

duplicate key value violates
unique constraint

"cards_listid_placement_de
letedat_unique"



Drop index



id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me	A	4	null

```
DROP INDEX cards_listid_placement_deletedat_unique
```

Drop index

id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	
pink	not relevant	A	3	2021-11-
black	move me	A	4	

DROP INDEX card deletedat_unique

id	list_id	placement
red	A	(+1) 2
blue	A	(+1) 3
green	A	(+1) 4
black	A	(:new_placement) 1



Order of execution

id	list_id	placement	old_placement	direction
red	A	1	4	1
blue	A	2	4	1
green	A	3	4	1
black	A	4	4	1

Order of execution



id	list_id	placement	old_placement	direction
red	A	1	4	1
blue	A	2	4	1
green	A	3	4	1
black	A	4	4	1

Don't assume anything

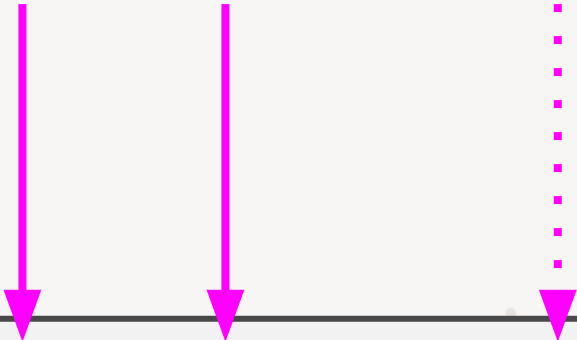
Order of execution



id	list_id	placement	old_placement	direction
red	A	1	4	1
blue	A	2	4	1
green	A	3	4	1
black	A	4	4	1

By default, a constraint check is applied on each update in the table, not at the end of the whole transaction.

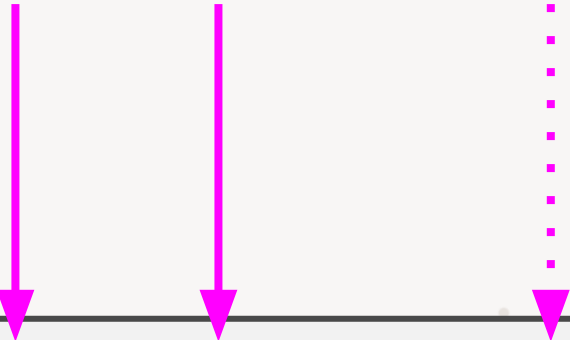
Defer



id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me			

```
ALTER TABLE cards
ADD CONSTRAINT cards_listid_placement_deletedat_unique
UNIQUE (list_id, placement) WHERE (deleted_at IS NULL)
DEFERRABLE INITIALLY DEFERRED
```


Defer



id	description	list_id	placement	deleted_at
red	blah	A	1	null
blue	As I user...	A	2	null
green	foo bar	A	3	null
pink	not relevant	A	3	2021-11-20T15:54:30
black	move me			

```
ALTER TABLE cards
ADD CONSTRAINT cards_listid_placement_deletedat_unique
UNIQUE (list_id, placement) WHERE (deleted_at IS NULL)
DEFERRABLE INITIALLY DEFERRED
```

Defer

id	description	list_id	placement	deleted_at
red				
blue	As I use			
green	foo			
pink	not rele			
black	mov			

ALTER TABLE cards
ADD CONSTRAINT cards_listid_placement_deletedat_unique_exclude
EXCLUDE (list_id WITH =, placement WITH =) **WHERE** (deleted_at **IS NULL**)
DEFERRABLE INITIALLY DEFERRED

ALTER TABLE cards
ADD CONSTRAINT cards_listid_placement_deletedat_unique
UNIQUE (list_id, placement) **WHERE** (deleted_at **IS NULL**)
DEFERRABLE INITIALLY DEFERRED

Defer

id	description	list_id	placement	deleted_at
red				
blue	As I use			
green	foo			
pink	not rele			
black	mov			

ALTER TABLE cards

ADD CONSTRAINT cards_listid_placement_deletedat_unique_exclude
EXCLUDE (list_id WITH =, placement WITH =) **WHERE** (deleted_at **IS NULL**)
DEFERRABLE INITIALLY DEFERRED

ALTER TABLE cards

ADD CONSTRAINT cards_listid_placement_deletedat_unique
UNIQUE (list_id, placement) **WHERE** (deleted_at **IS NULL**)
DEFERRABLE INITIALLY DEFERRED

- `red.deleted_at == null && black.deleted_at == null`
- `red.list_id == black.list_id`
- `red.placement == black.placement`

Defer

id	description	list_id	placement	deleted_at
red				
blue	As I use			
green	foo			
pink	not rele			
black	mov			

ALTER TABLE cards

ADD CONSTRAINT cards_listid_placement_deletedat_unique_exclude
EXCLUDE (list_id WITH =, placement WITH =) **WHERE** (deleted_at **IS NULL**)
DEFERRABLE INITIALLY DEFERRED

ALTER TABLE cards

ADD CONSTRAINT cards_listid_placement_deletedat_unique
UNIQUE (list_id, placement) **WHERE** (deleted_at **IS NULL**)
DEFERRABLE INITIALLY DEFERRED

- `red.deleted_at == null && black.deleted_at == null`
- `red.list_id == black.list_id`
- `red.placement == black.placement`

Summary





Schibsted