pg_recall

a time machine for your PostgreSQL data

https://github.com/mreithub/pg_recall/
(https://github.com/mreithub/pg_recall/)

In [2]:
```
\connect host=localhost user=manuel database=postgres nopassword
```

```
ok
```

what it’s for

keeps track of data changes to your tables

- look at past states of your tables
- list changes for individual keys
- allows you to implement tools to automate that
  - undo/redo buttons
  - Revision history page
  - a backend for customer service where they can see what the user did
  - look for data/users that behave oddly

for...

- user data
- logging metrics

... data that doesn't change *too often*

design goals

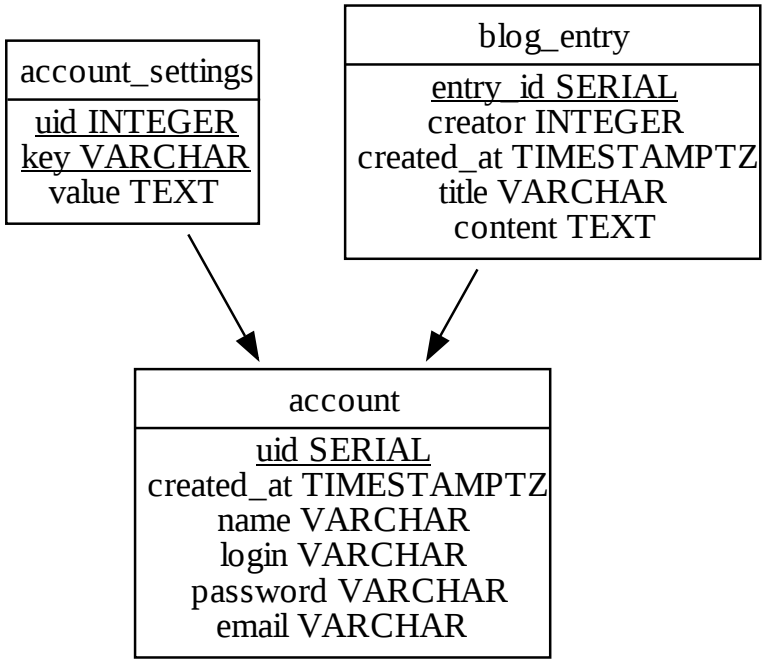
CRUD transparency

simplicity

flexibility

little overhead

Demo



In [35]:
```
BEGIN; -- run all this in a transaction (for predictable timing)
```

```
ok (took 0.00ms)
```

In [36]:

```
CREATE TABLE IF NOT EXISTS account (
  uid SERIAL PRIMARY KEY,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),

  name VARCHAR(200) NOT NULL,
  login VARCHAR(100) NOT NULL,
  password VARCHAR(200) NOT NULL,
  email VARCHAR(200) NOT NULL
);
CREATE UNIQUE INDEX IF NOT EXISTS idx_account_login ON account(lowe
r(login));

CREATE TABLE IF NOT EXISTS account_settings (
  uid INTEGER NOT NULL,
  key VARCHAR(100) NOT NULL,
  value TEXT NOT NULL,

  PRIMARY KEY (uid, key),
  FOREIGN KEY (uid) REFERENCES account(uid)
);
 §
CREATE TABLE IF NOT EXISTS blog_entry (
  entry_id SERIAL PRIMARY KEY,
  creator INTEGER NOT NULL,
  created_at TIMESTAMPTZ NOT NULL DEFAULT now(),

  title VARCHAR(200) NOT NULL,
  content TEXT NOT NULL,

  FOREIGN KEY (creator) REFERENCES account(uid)
);
```

ok (took 0.00ms)

Installation

run `make install` in the source directory

In [37]:
```
CREATE EXTENSION IF NOT EXISTS btree_gist;
CREATE EXTENSION IF NOT EXISTS recall WITH VERSION '0.9.5';

SELECT recall.enable('account_settings', '3 months');
SELECT recall.enable('blog_entry', '6 months');
```

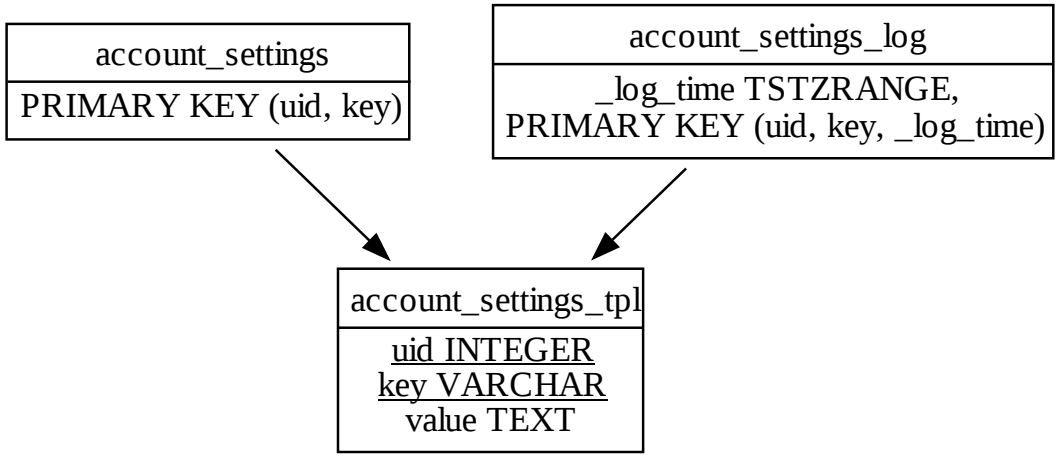
enable

1 rows (took 0.00ms)

In [38]:
```
-- cheating a little
CREATE OR REPLACE FUNCTION pretendToWait(t INTERVAL) RETURNS void A
S $$
  UPDATE recall.account_settings_log SET _log_time = tstzrange(LOWE
R(_log_time)-t, UPPER(_log_time)-t);
  UPDATE recall.blog_entry_log SET _log_time = tstzrange(LOWER(_lo
g_time)-t, UPPER(_log_time)-t);
$$ LANGUAGE sql;
```

ok (took 0.00ms)

Table inheritance



In [7]:
```
\dt
```

Schema	Name	Type	Owner
public	account	table	manuel
public	account_settings	table	manuel
public	blog_entry	table	manuel

3 rows (took 0.00ms)

In [8]:
```
\dt recall
```

Schema	Name	Type	Owner
recall	_config	table	manuel
recall	account_settings_log	table	manuel
recall	account_settings_tpl	table	manuel
recall	blog_entry_log	table	manuel
recall	blog_entry_tpl	table	manuel

5 rows (took 0.00ms)

Some data...

In [39]:
```
INSERT INTO account (uid, name, login, password, email)
VALUES (12, 'John Doe', 'jdoe', 'very secure password', 'jdoe@exampl
e.com')
RETURNING uid;
```

uid
12

1 rows (took 0.00ms)

In [40]:
```
INSERT INTO account_settings (uid, key, value) VALUES
(12, 'get_newsletter', true),
(12, 'enable_spellcheck', false);
```

2 rows (took 0.00ms)

In [41]:
```
INSERT INTO blog_entry (entry_id, creator, title, content) VALUES
(123   , 12, 'Welcome to my new bog', 'This is sooooo super excitin
g!'),
(124, 12, 'House warming party', 'I want to invite you all to my hou
se warming party next tuesday at 123 Some Place')
RETURNING entry_id;
```

entry_id
123
124

2 rows (took 0.00ms)

Changes

In [42]:
```
-- fix a typo
SELECT pretendToWait('5 minutes');
UPDATE blog_entry SET title = 'Welcome to my new blog'
  WHERE entry_id = 123;
```

```
1 rows (took 0.00ms)
```

In [43]:
```
-- enable spell check to prevent typos in the future
SELECT pretendToWait('5 minutes');
UPDATE account_settings SET value = true
  WHERE uid = 12 AND key = 'enable_spellcheck';
```

```
1 rows (took 0.00ms)
```

In [44]:
```
-- remove the second blog entry
SELECT pretendToWait('5 minutes');
DELETE FROM blog_entry WHERE entry_id = 124;
```

```
1 rows (took 0.00ms)
```

Let's have a look

In [15]:
```
SELECT * FROM account;
```

uid	created_at	name	login	password	email
12	2016-04-04 20:52:30.577405+02:00	John Doe	jdoe	very secure password	jdoe@example.com

1 rows (took 0.00ms)

In [16]:
```
SELECT * FROM account_settings;
```

uid	key	value
12	get_newsletter	true
12	enable_spellcheck	true

2 rows (took 0.00ms)

In [17]:
```
SELECT * FROM blog_entry;
```

entry_id	creator	created_at	title	content
123	12	2016-04-04 20:52:30.577405+02:00	Welcome to my new blog	This is sooooo super exciting!

1 rows (took 0.00ms)

Going back in time

In [46]:
```
SELECT recall.at('blog_entry', now() - interval '10 minutes');
SELECT * FROM blog_entry_past;
```

entry_id	creator	created_at	title	content
123	12	2016-04-04 21:16:47.405329+02:00	Welcome to my new blog	This is sooooo super exciting!
124	12	2016-04-04 21:16:47.405329+02:00	House warming party	I want to invite you all to my house warming party next tuesday at 123 Some Place

2 rows (took 0.00ms)

Behind the scenes

In [19]:
```
SELECT uid, key, value, _log_time::text FROM recall.account_settings_log;
```

uid	key	value	_log_time
12	get_newsletter	true	["2016-04-04 20:37:30.577405+02",)
12	enable_spellcheck	false	["2016-04-04 20:37:30.577405+02","2016-04-04 20:47:30.577405+02")
12	enable_spellcheck	true	["2016-04-04 20:47:30.577405+02",)

3 rows (took 0.00ms)

In [20]:

```
SELECT entry_id, creator, created_at, title, content, _log_time::tex
t
FROM recall.blog_entry_log;
```

entry_id	creator	created_at	title	content	_log_tim
123	12	2016-04-04 20:52:30.577405+02:00	Welcome to my new bog	This is sooooo super exciting!	["2016-0 20:37:30 04-04 20
123	12	2016-04-04 20:52:30.577405+02:00	Welcome to my new blog	This is sooooo super exciting!	["2016-0 20:42:30
124	12	2016-04-04 20:52:30.577405+02:00	House warming party	I want to invite you all to my house warming party next tuesday at 123 Some Place	["2016-0 20:37:30 04-04 20

3 rows (took 0.00ms)

In [21]:
```
SELECT * FROM recall._config;
```

tblid	ts	log_interval	last_cleanup	pk
account_settings	2016-04-04 20:52:30.577405+02:00	90 days, 0:00:00	None	['u 'ke
blog_entry	2016-04-04 20:52:30.577405+02:00	180 days, 0:00:00	None	['e

2 rows (took 0.00ms)

In [22]:
```
ROLLBACK;
```

ok (took 0.00ms)

design choices

storage is (relatively) cheap

- logs rows, not individual fields
- detects unchanged rows
- each table has a retention interval

separate log tables

timestamps identify revisions

In [23]:
```
BEGIN;
SELECT now();
```

now
2016-04-04 21:05:00.405573+02:00

1 rows (took 0.00ms)

In [31]:
```
SELECT now();
```

now
2016-04-04 21:05:00.405573+02:00

1 rows (took 0.00ms)

In [32]:
```
ROLLBACK;
```

ok (took 0.00ms)

`tstzrange` as revision identifier

In [34]:
```
SELECT '[2011-01-01,2011-03-01)'::tsrange @> '2011-01-10'::timestamp
```

?column?
True

1 rows (took 0.00ms)

no constraints in the log table

Restrictions

it protects user data, not schema changes

manual cleanup

storage overhead

depends on `btree_gist`

PostgreSQL features

extension support

range types

- no-overlap
- non-empty

GiST[1] indexes

[1] Generalized Search Tree

the `btree_gist` extension contains GiST index implementations for built in types

table inheritance

pl/pgsql

Future

- automatic partitioning
- website
- ports?
- ...

Similar projects

PostgreSQL

- TimeTravel for PostgreSQL (http://www.databtech.com/eng/index_timetravel.htm) (GNU GPLv3)
- A PL/pgSQL Trigger Procedure For Auditing (http://www.postgresql.org/docs/current/static/plpgsql-trigger.html#PLPGSQL-TRIGGER-AUDIT-EXAMPLE) in the PostgreSQL docs

Others

- Temporal queries in SQL:2011
- Oracle FlashBack (https://docs.oracle.com/cd/B28359_01/appdev.111/b28424/adfns_flas
- CouchDB's Document Revisions (http://docs.couchdb.org/en/1.6.1/intro/api.html#revisions)
- EclipseLink JPA History (https://wiki.eclipse.org/EclipseLink/Examples/JPA/History)

Questions?

feel free to talk to me afterwards, tweet me at @mreithub (https://twitter.com/mreithub) or send an email to manuel@reithuber.net (mailto:manuel@reithuber.net?subject=pg_recall) .

The project's open source and can be found at: https://github.com/mreithub/pg_recall/ (https://github.com/mreithub/pg_recall/)

These slides were written using jupyter.org (https://jupyter.org) (extended by RISE (https://github.com/damianavila/RISE/) as well as my own postgres_kernel (https://github.com/mreithub/postgres_kernel))

Cleanup

In [ ]:
```
-- cleanup
SELECT recall.disable('account_settings');
SELECT recall.disable('blog_entry');
```

In [ ]:
```
DROP TABLE blog_entry, account_settings, account;
```