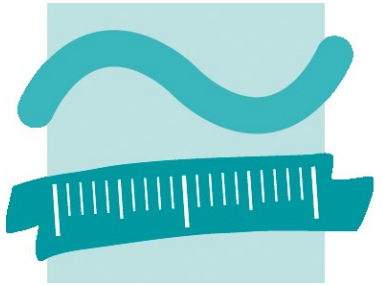
BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

JSONB AUDITS

PRO & CONTRA

Felix Kunde and Petra Sauer



BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN

University of Applied Sciences

German-speaking PostgreSQL Conference 2018 Berlin


BEUTH
University of Applied

Wirtschaftsatlas Berlin

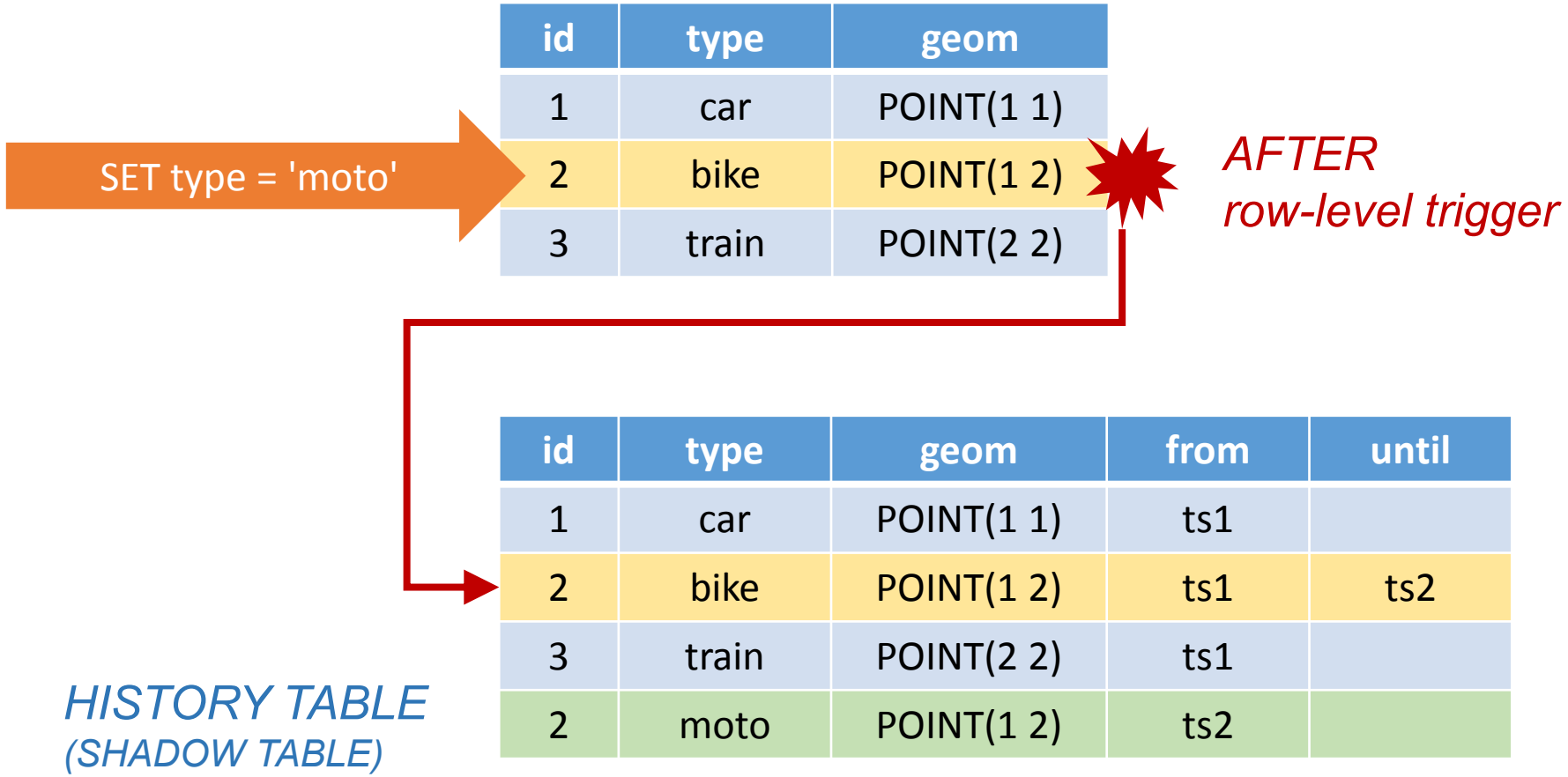
In the GIS scene more and more users are interested in data curation. Data creation has been too cost-intensive to just throw it away. Plus, a view into the past is desirable to measure effects of location-based decision making

MOTIVATION

- Track all data changes in the database
- Revisit previous data versions
- Undo changes of certain write operations
- Work against multiple branches of a database

HOW TO AUDIT?

IN A RELATIONAL WAY



id	type	geom
1	car	POINT(1 1)
2	bike	POINT(1 2)
3	train	POINT(2 2)

id	type	geom	from	until
1	car	POINT(1 1)	ts1	
2	bike	POINT(1 2)	ts1	ts2
3	train	POINT(2 2)	ts1	
2	moto	POINT(1 2)	ts2	

GOOD …

- Easy to setup
- Easy to query the past
- Does not break apps

, BUT …

- What to do with DDL changes?
- Rely on timestamp fields?
- Store complete tuples?

TOOLS

Tool	Method	Log type	Revision
timetravel	Audit Trail	Extra Columns	Timestamps
temporal_tables	Audit Trail	Shadow Tables	Timestamps
table_version	Audit Trail	Shadow Tables	UD revision
table_log	Audit Trail	Shadow Tables	Trigger seq
audit_trigger	Audit Trail	Generic (hstore)	Transactions
pgMemento	Audit Trail	Generic (jsonb)	Transactions
CyanAudit	Audit Trail	Generic (pivot)	Transactions
pgVersion	Version Control	Extra Columns	UD revision
QGIS Versioning	Version Control	Extra Columns	UD revision
GeoGig	Version Control	External (binary)	UD revision
Flyway	Migration	External (SQL)	UD revision
Liquibase	Migration	External (XML)	UD revision

WHY AUDIT IN JSONB?

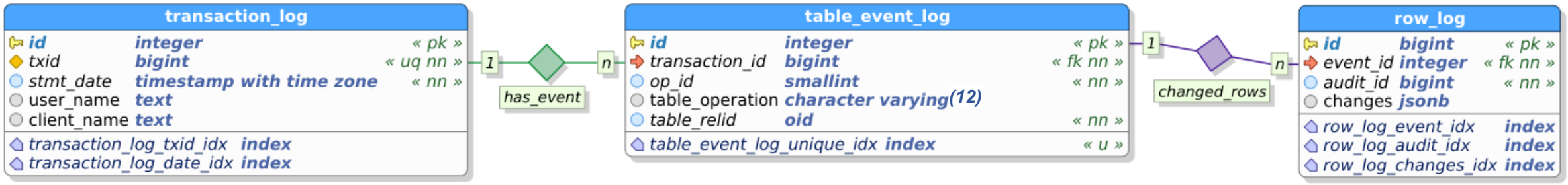
- Stop care about DDL changes
- Can be indexed, so queries are fast
- Store your JSON somewhere else
- *Everybody loves JSONB!!*

WHAT IS DIFFERENT?

- Relies on transaction IDs, not timestamps
- Stores only deltas in the logs
- Has a powerful undo feature

LOG TABLES

Transaction Metadata | Event Metadata | Data log



- When using JSONB everything could be stored in one log table
- pgMemento stores transaction and table event metadata in separate tables to facilitate the lookup for historic actions
- Less redundancy vs. higher logging overhead

DML-AUDITING

Surrogate Key (as PK can cover more columns)

id	type	geom	audit_id
1	car	POINT(1 1)	2
2	bike	POINT(1 2)	23
3	train	POINT(2 2)	42

txid = 2800000

SET type = 'moto'

insert

BEFORE statement-level trigger

id	txid	…
…	…	…
10	2800000	…

TRANSACTION_LOG

insert

id	transaction_id	op_id	Table_operation	table_relid
…	…	…	…	…
50	2800000	4	UPDATE	2005030

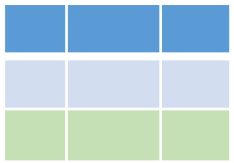
TABLE_EVENT_LOG

DML-AUDITING

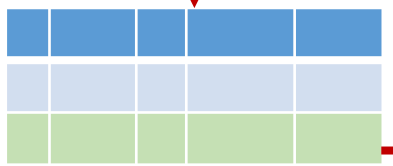
```
SELECT
  event_id
FROM
  table_event_log
WHERE
  transaction_id = txid_current()
  AND table_relid = 2005030
  AND op_id = 4;
```

id	type	geom	audit_id
1	car	POINT(1 1)	2
2	moto	POINT(1 2)	23
3	train	POINT(2 2)	42

AFTER row-level trigger



TRANSACTION_LOG



TABLE_EVENT_LOG

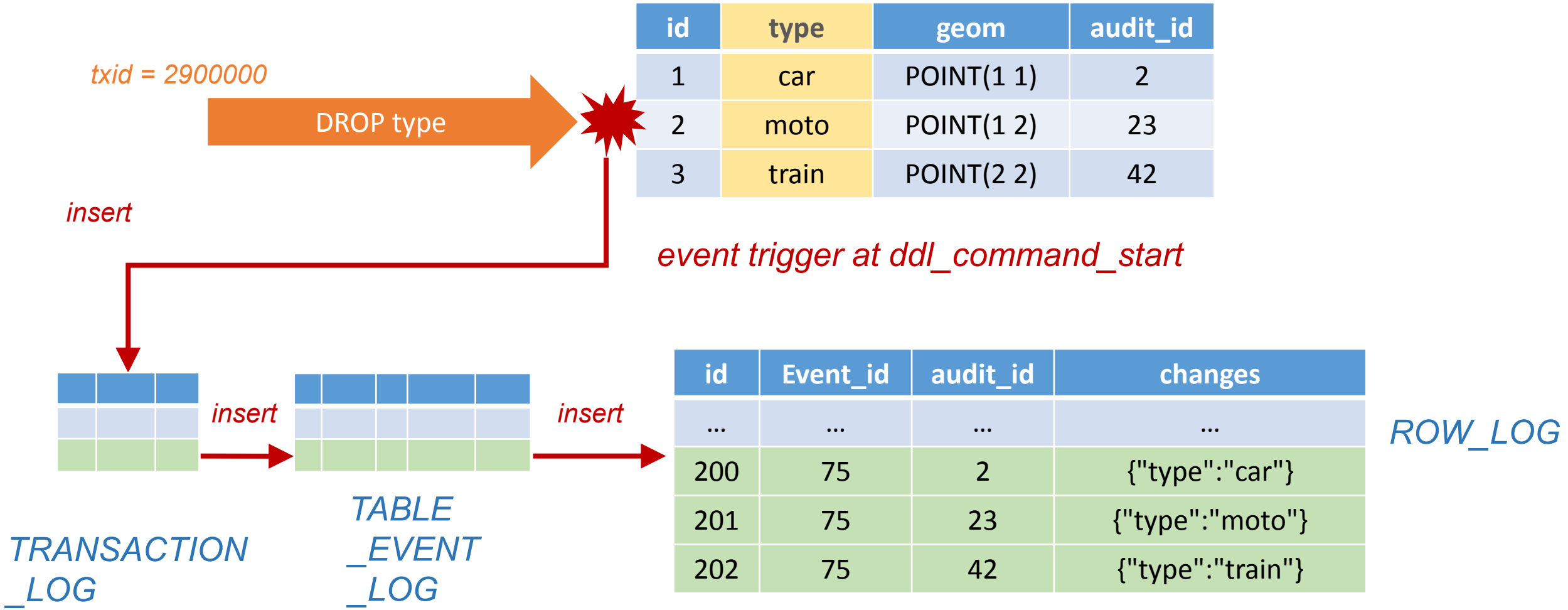
insert

id	Event_id	audit_id	changes
…	…	…	…
100	50	23	{"type":"bike"}

Surrogate Key as tracer

ROW_LOG

DDL-AUDITING



id	type	geom	audit_id
1	car	POINT(1 1)	2
2	moto	POINT(1 2)	23
3	train	POINT(2 2)	42

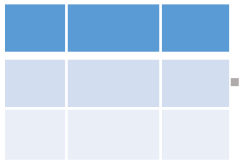
id	Event_id	audit_id	changes
…	…	…	…
200	75	2	{"type":"car"}
201	75	23	{"type":"moto"}
202	75	42	{"type":"train"}

DDL-AUDITING

id	geom	audit_id
1	POINT(1 1)	2
2	POINT(1 2)	23
3	POINT(2 2)	42

event trigger at ddl_command_end

```
SELECT * FROM pg_event_trigger_ddl_commands()
```



id	audit_table_id	column_name	data_type	txid_range
…	…	…	…	
100	1	type	text	[2700000,2900000)

AUDIT_TABLE_LOG

AUDIT_COLUMN_LOG

FAST QUERIES POWERED BY GIN INDEX

For which transactions column 'type' exists in the logs?

```
SELECT DISTINCT
  e.transaction_id
FROM
  pgmemento.table_event_log e
JOIN
  pgmemento.row_log r
  ON r.event_id = e.id
WHERE
  r.audit_id = 23
  AND (r.changes ? 'type');
```

Which tuples once contained certain combinations of key(s) and value(s)?

```
SELECT DISTINCT
  audit_id
FROM
  pgmemento.row_log
WHERE
  changes @> '{"type": "bike"}'::jsonb;
```

Imagine the whole tuple is stored every time.
More overhead on data processing.

TIME FOR PAIN

RESTORE PREVIOUS VERSIONS OF TUPLES

THE LIFE OF A TUPLE

	id	name	type	Status	Audit_id
BIRTH	NULL				10
GROWING UP				{"status":„hello world"}	10
MARRIAGE		{"name":"foo",		"status":"alone"}	10
WISDOM			{"type":"phd"}		10
DEATH	{Id:1,	"name":"bar",	"type":"prof",	"status":"happy"}	10

Time / transactions

HOW WAS LIFE BEFORE MARRIAGE?

	id	name	type	Status	Audit_id
BIRTH			NULL		10
GROWING UP				{"status":„hello world"}	10
Uh oh, where is the log?!					
MARRIAGE		{"name":"foo",		"status":"alone"}	10
WISDOM			{"type":"phd"}		10
DEATH	{Id:1,	"name":"bar",	"type":"prof",	"status":"happy"}	10

Time / transactions

STRATEGY 1:
ROLLING BACK

- Concat all JSONB logs in reverse order
- Starting from recent state (or delete event) until requested point in time
- JSONB trick: duplicate keys get overwritten
- Too much overhead, if history is long

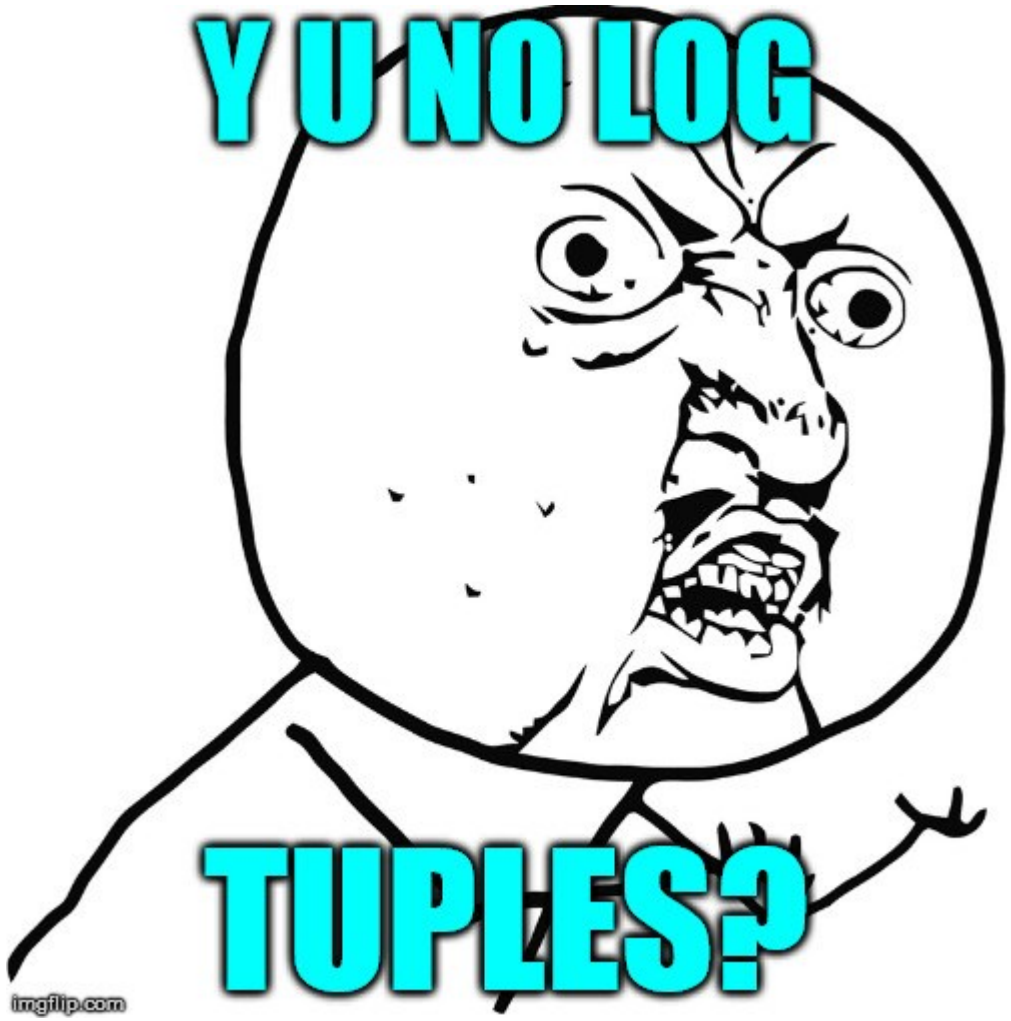
STRATEGY 2:
JSONB QUERIES

- Check audit_column_log
- For each column find the first entry in logs after requested point in time
- Feed result to jsonb_build_object
- Produces giant queries, but still quite fast

RESTORE – PART 3

```
SELECT
  p.*
FROM
  generate_log_entries(1,2800000,'my_table') entries
LATERAL (
  SELECT
    *
  FROM
    jsonb_populate_record(
       null::my_table,
       entries
    )
) p;
```

Works only with a template. Could be the actual table, but to be correct in case of any DDL changes, a temporary template can be created on the fly with information from audit_column_log.


Y U NO LOG
TUPLES?
imgflip.com

WELL…

- … why would you use JSONB anyway then?
- Relational is faster and easier
- Takes up more space on your disk
- Your app might not like it

REVERT

- Query all changes and referenced events for a given txid (or range of txids) in reverse order
- Loop over result set and perform the opposite event
- Consider dependencies between tables in order to avoid foreign key violations

REVERT

op_id	Event	Reverse Event	Log Content
1	CREATE TABLE	DROP TABLE	-
2	ALTER TABLE ADD COLUMN	ALTER TABLE DROP COLUMN	-
3	INSERT	DELETE	NULL
4	UPDATE	UPDATE	Changed fields of changed rows
5	ALTER TABLE ALTER COLUMN	ALTER TABLE ALTER COLUMN	All rows of altered columns
6	ALTER TABLE DROP COLUMN	ALTER TABLE ADD COLUMN	All rows of deleted columns
7	DELETE	INSERT	All fields of deleted rows
8	TRUNCATE	INSERT	All fields of table
9	DROP TABLE	CREATE TABLE	All fields of table (logged as truncate)

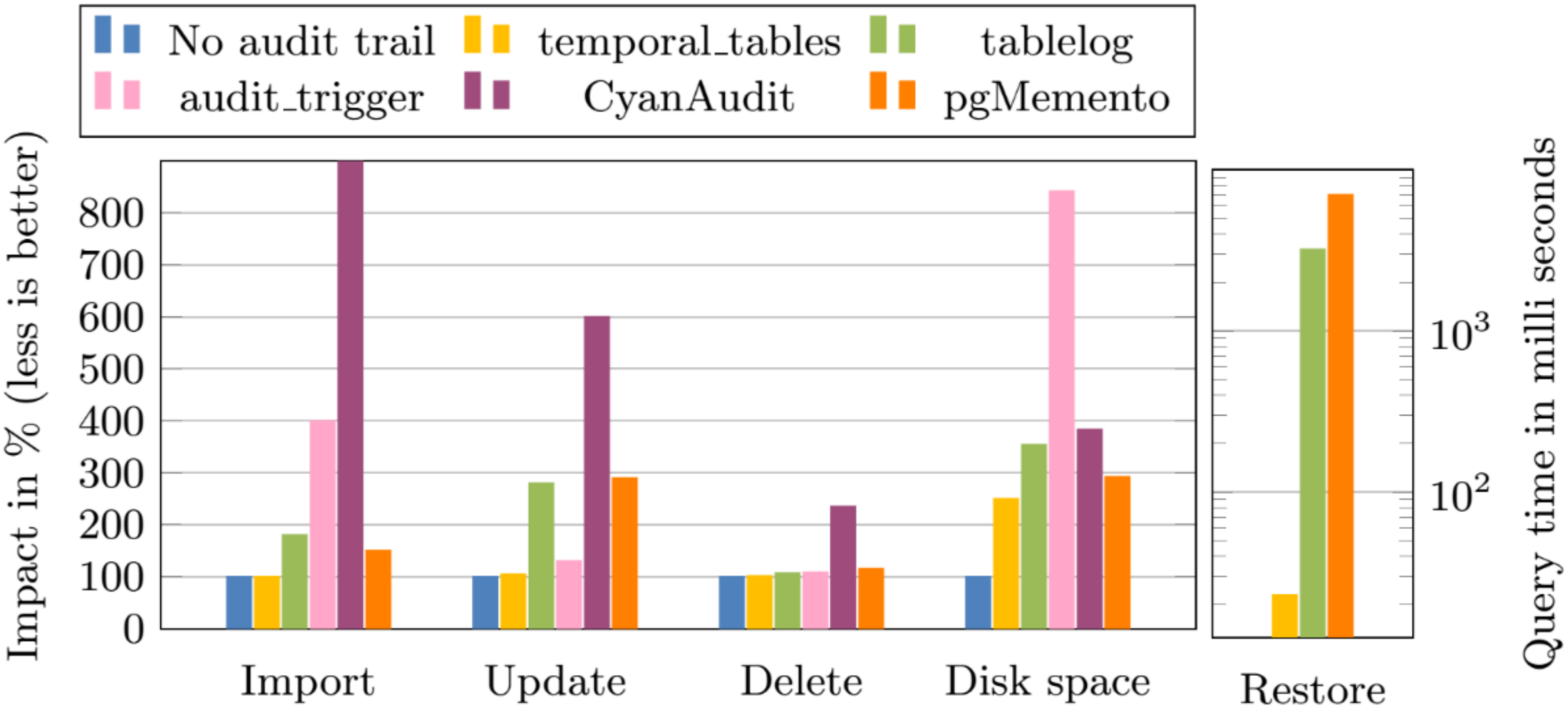
JSONB > QUERIES

- Updates are pretty easy to setup *(ok we could produce deltas also here and not during trigger phase)*
- Could lead to a branching concept …
- What about long running transactions? (typical GIS workflows – many edits, commit once)

TO DOs

- Branching concept
- Log tables for more DB objects
- Extending the test suite
- Maybe: Logical decoding instead of triggers

PERFORMANCE

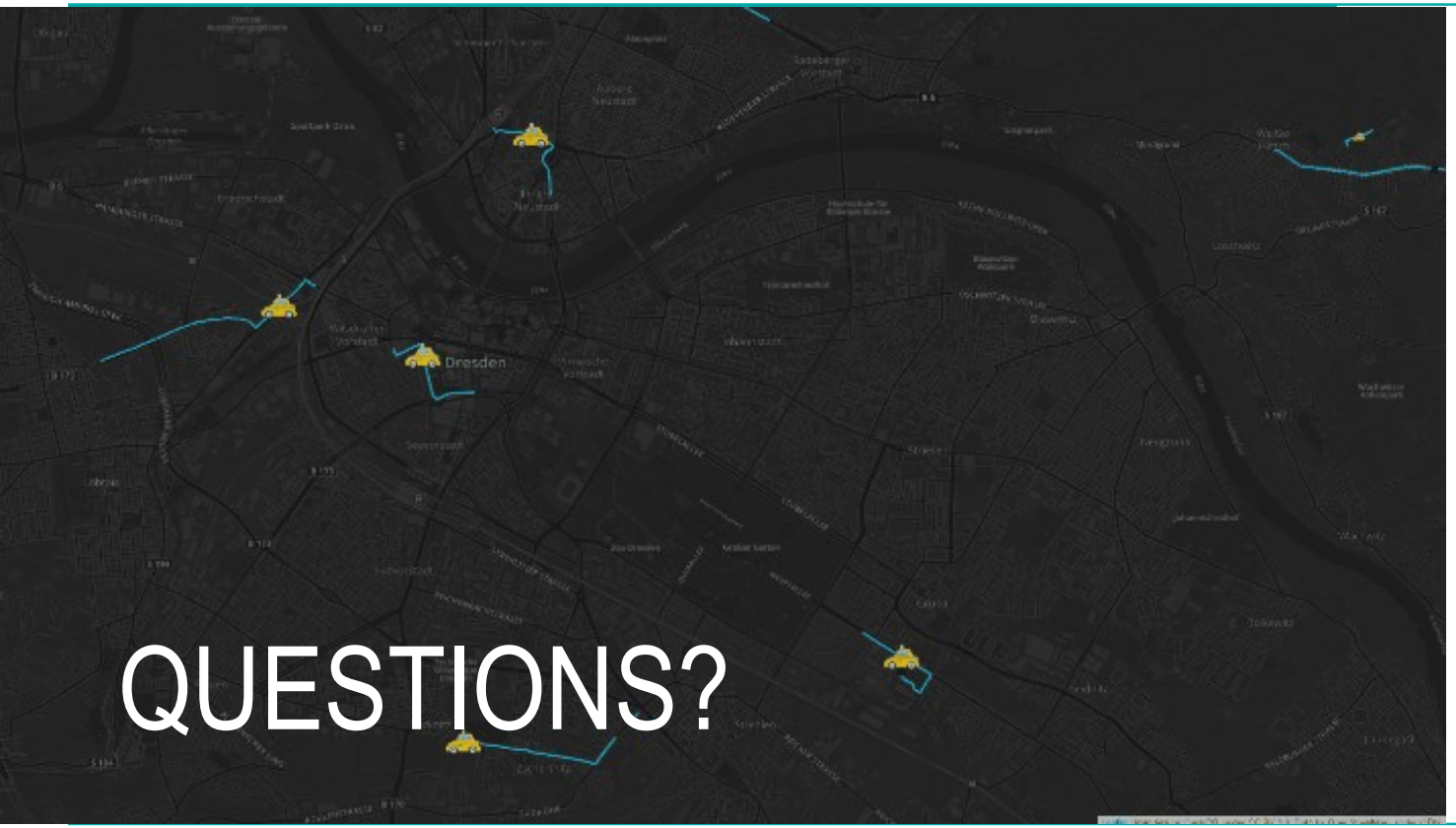


Kunde F., Sauer P. (2017) pgMemento – A Generic Transaction-Based Audit Trail for Spatial Databases. In: Gertz M. et al. (eds) Advances in Spatial and Temporal Databases. SSTD 2017. Lecture Notes in Computer Science, vol 10411. Springer, Cham

TECHNICAL DETAILS

- Written entirely in PL/pgSQL
- Requires at least PostgreSQL 9.5
- Repo: github.com/pgmemento
- LGPL v3 Licence

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences



Felix Kunde

fkunde[at]beuth-hochschule.de

Petra Sauer

Sauer[at]beuth-hochschule.de

Funded by:

DLR

Bundesministerium für Wirtschaft und Energie

Smart Data