2ndQuadrant®
PostgreSQL

MVCC and Vacuum explained

Boriss Mejías
Consultant - 2ndQuadrant
Air Guitar Player

Prelude

ACID

ACID

Atomicity, Consistency, Isolation, Durability

BASE

Basically Available, Soft state,
Eventually consistent

Wikipedia: Milky Way

Wikipedia: ALMA

MVCC

- Multi-Version Concurrency Control
- Multiple versions of each row
- Only one version is visible to each observer
- Time-consistent view of the whole database is always available for each session: Snapshot

Everything is a transaction

```
INSERT INTO key_value (key, value)
     VALUES (42, 'towel');
```

Everything is a transaction

```
BEGIN;
INSERT INTO key_value (key, value)
     VALUES (42, 'towel');
COMMIT;
```

Concurrency

```
BEGIN;

SELECT value
  FROM key_value
 WHERE key = 42;

COMMIT;
```

```
BEGIN;
UPDATE key_value
   SET value = 'foo'
 WHERE key = 42;

COMMIT;
```

Time

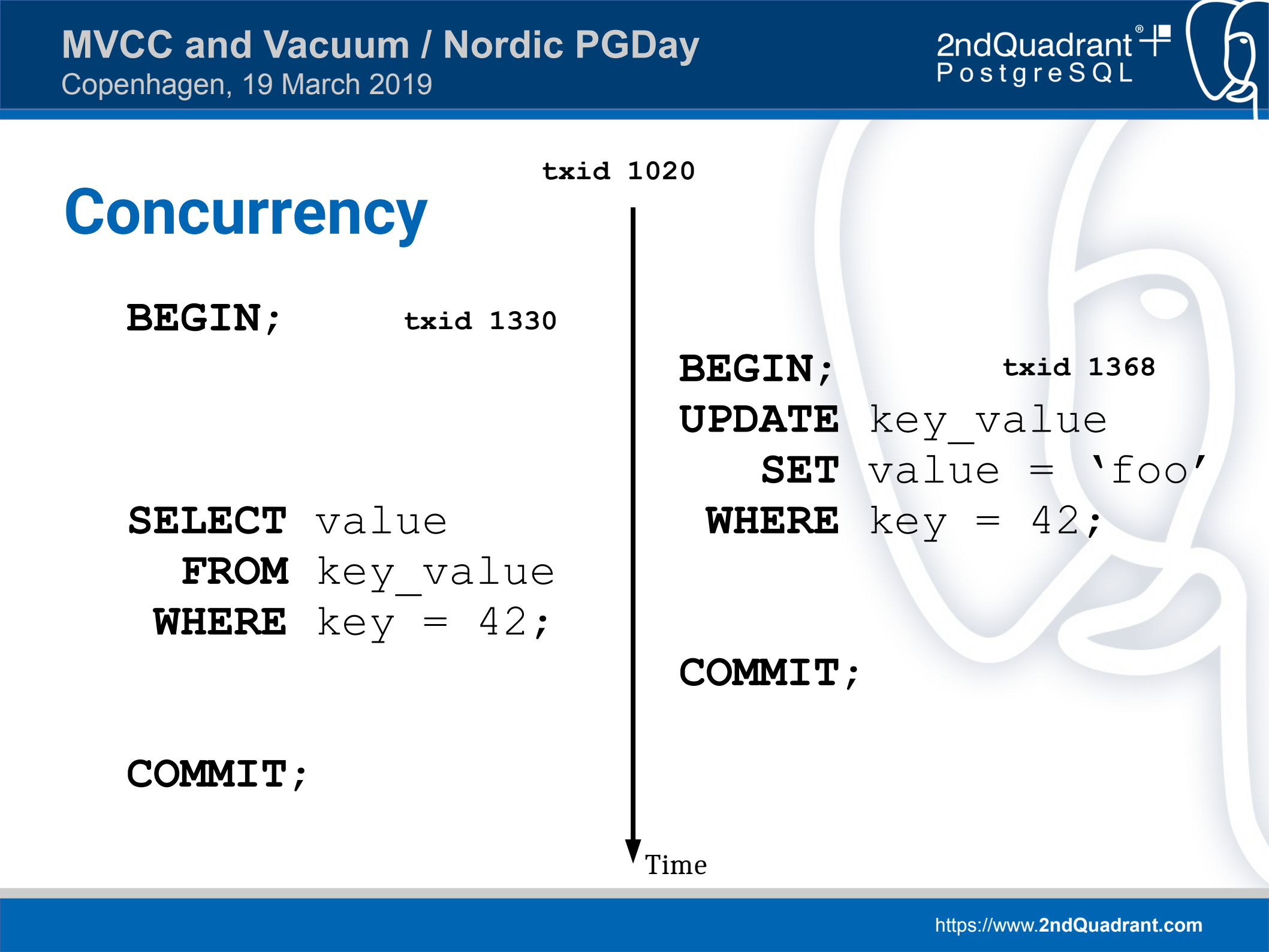
MVCC Basic Components

- Each transaction has a full transaction id
  - `SELECT txid_current();`
- Each row has visibility information
  - xmin: creation txid
  - xmax: when row was deleted/updated
- Each transaction has a status
  - Commit log / pg_xact

SQL Write Queries

- **`INSERT`**: sets xmin
- **`DELETE`**: sets xmax
- **`UPDATE`**: delete and insert → sets xmax and creates new version with xmin

Concurrency



Multi-Versions of the row

```
xmin xmax    key    value
1020 0       42     'towel'

UPDATE →
xmin xmax    key    value
1020 1368    42     'towel'
1368 0       42     'foo'
```

Commit state of txid 1368 will be “commit” or “rollback”

Dead Rows

- **`DELETE`**: creates a dead row
- **`UPDATE`**: *delete* and insert → dead row
- **`ROLLBACK`** of writes: creates dead rows

- Too many dead rows → “bloated table”

Interlude

Mind the Transaction Isolation Level

- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE
- ~~READ UNCOMMITTED~~

Concurrency

```
BEGIN;
SELECT value
  FROM key_value
 WHERE key = 42;


SELECT value
  FROM key_value
 WHERE key = 42;
COMMIT;
```

```
BEGIN;
UPDATE key_value
   SET value = 'foo'
 WHERE key = 42;
COMMIT;
```

Time

Vacuum

Vacuum – Dead Rows Maintenance

- **`VACUUM`** `table;`
- Remove dead rows
- Unless the dead row is potentially visible for at least one session

Concurrency

```
BEGIN;
SELECT value
  FROM key_value
 WHERE key = 42;


SELECT value
  FROM key_value
 WHERE key = 42;
COMMIT;
```

```
BEGIN;
UPDATE key_value
   SET value = 'foo'
 WHERE key = 42;
COMMIT;

VACUUM;
```

Time

Table pg_stat_user_tables

```
-[ RECORD 1 ]-------+-----------
relid               | 24652
schemaname          | public
relname             | key_value
n_tup_ins           | 2789000
n_tup_upd           | 6712
n_tup_del           | 1789000
n_tup_hot_upd       | 0
n_live_tup          | 1000000
n_dead_tup          | 5666
```

VACUUM key_value;

```
-[ RECORD 1 ]-------+------------
relid               | 24652
schemaname          | public
relname             | key_value
n_tup_ins           | 2789000
n_tup_upd           | 6712
n_tup_del           | 1789000
n_tup_hot_upd       | 0
n_live_tup          | 1000000
n_dead_tup          | 0
```

Vacuum – Dead Rows Maintenance

- **`VACUUM`** `table;`
- Remove dead rows
- Unless the dead row is potentially visible for at least one session
- Long running transactions can be an issue
  - Check ‘idle in transaction’ in pg_stat_activity

Vacuum Command

- **`VACUUM`**;
  - Vacuums the entire database
- **`VACUUM`** `tablename;`
  - Vacuums table tablename
- **`VACUUM ANALYZE`** `tablename;`
  - Vacuums and analyzes the table
- `$ vacuumdb --jobs=N`
  - Multiple concurrent vacuums

Vacuum effects

- **`VACUUM`** locks against DDL
- Only one vacuum per table
- `INSERT`, `DELETE`, `UPDATE` can still run

Vacuum effects

- **`VACUUM`** locks against DDL
- Only one vacuum per table
- `INSERT`, `DELETE`, `UPDATE` can still run

divertimento or delude

- **`VACUUM FULL`** works differently
  - It locks everything
  - It creates a new table
  - Think of a butterfly

Autovacuum

Autovacuum

- Runs **`VACUUM`** and **`ANALYZE`**
- Runs all the time
- No scheduling, just nap times
- It cancels itself to avoid blocking user's actions

Autovacuum basic parameters

- `autovacuum = on`
- `autovacuum_naptime = 1min`
- `autovacuum_max_workers = 3`
- `log_autovacuum_min_duration = -1`

Autovacuum triggered

```
threshold + (rows * scale_factor)
```

- autovacuum_vacuum_threshold = 50
- autovacuum_vacuum_scale_factor = 0.2

- autovacuum_analyze_threshold = 50
- autovacuum_analyze_scale_factor = 0.1

Table pg_stat_user_tables

```
-[ RECORD 1 ]-------+-----------
relid               | 24652
schemaname          | public
relname             | key_value
n_tup_ins           | 2789000
n_tup_upd           | 6712
n_tup_del           | 1789000
n_tup_hot_upd       | 0
n_live_tup          | 1000000
n_dead_tup          | 5666
```

Table pg_stat_user_tables

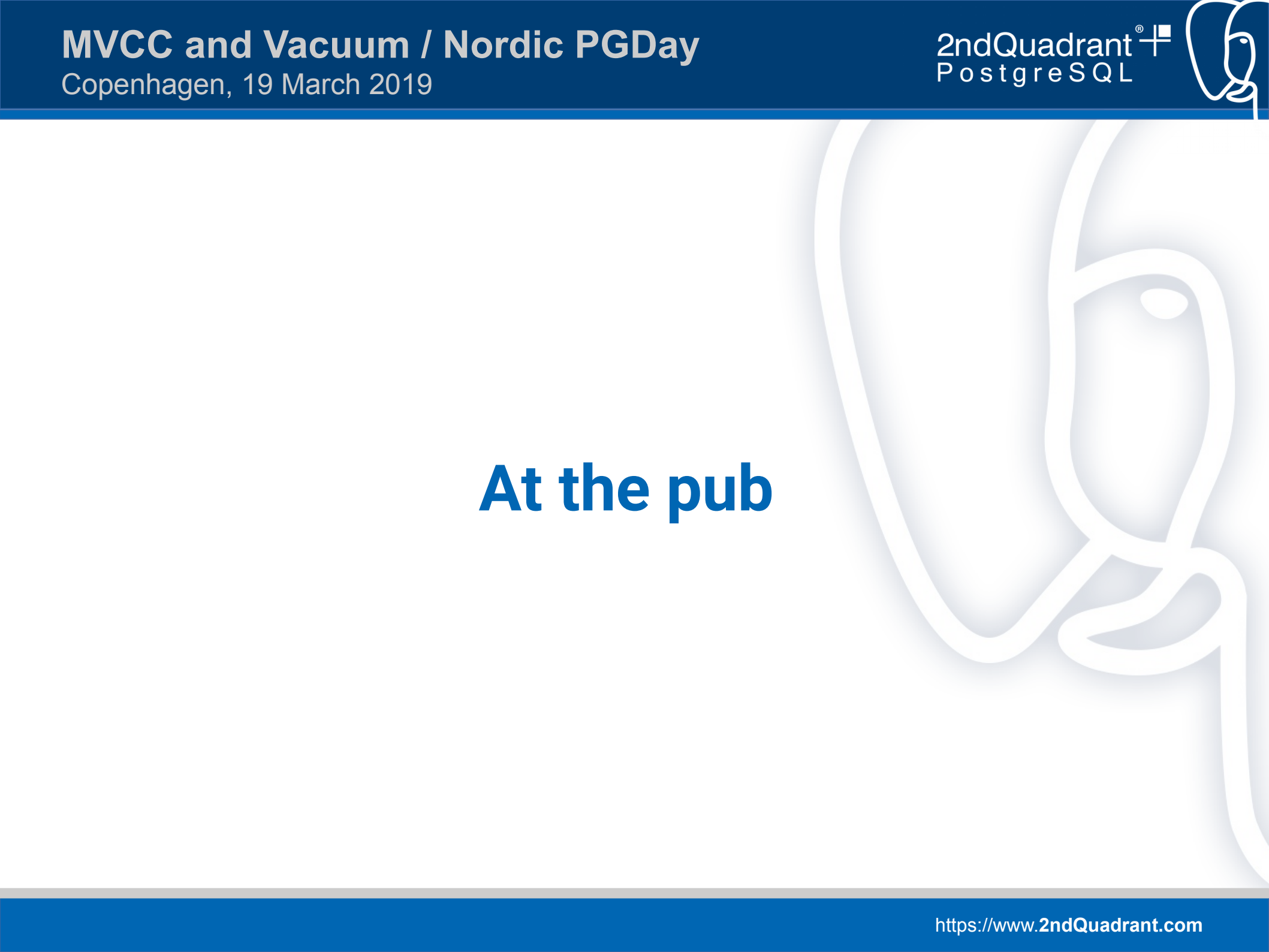
```
-[ RECORD 1 ]-------+------------
relname             | key_value
last_vacuum         | 2019-02-19 12:58:42
last_autovacuum     | 2019-03-16 07:06:06
last_analyze        | 2019-02-19 12:58:42
last_autoanalyze    | 2019-03-16 07:06:06
vacuum_count        | 7
autovacuum_count    | 6128
analyze_count       | 10676
autoanalyze_count   | 7
```

Autovacuum – How much work?

- `autovacuum_vacuum_cost_limit = 200`
- `vacuum_cost_page_hit = 1`
- `vacuum_cost_page_miss = 10`
- `vacuum_cost_page_dirty = 20`

Table pg_stat_progress_vacuum

```
-[ RECORD 1 ]------+------------
pid                | 27666
datid              | 18613
datname            | lamuella
relid              | 24652
phase              | performing final cleanup
heap_blks_total    | 1
heap_blks_scanned  | 1
heap_blks_vacuumed | 1
index_vacuum_count | 0
max_dead_tuples    | 291
num_dead_tuples    | 0
```

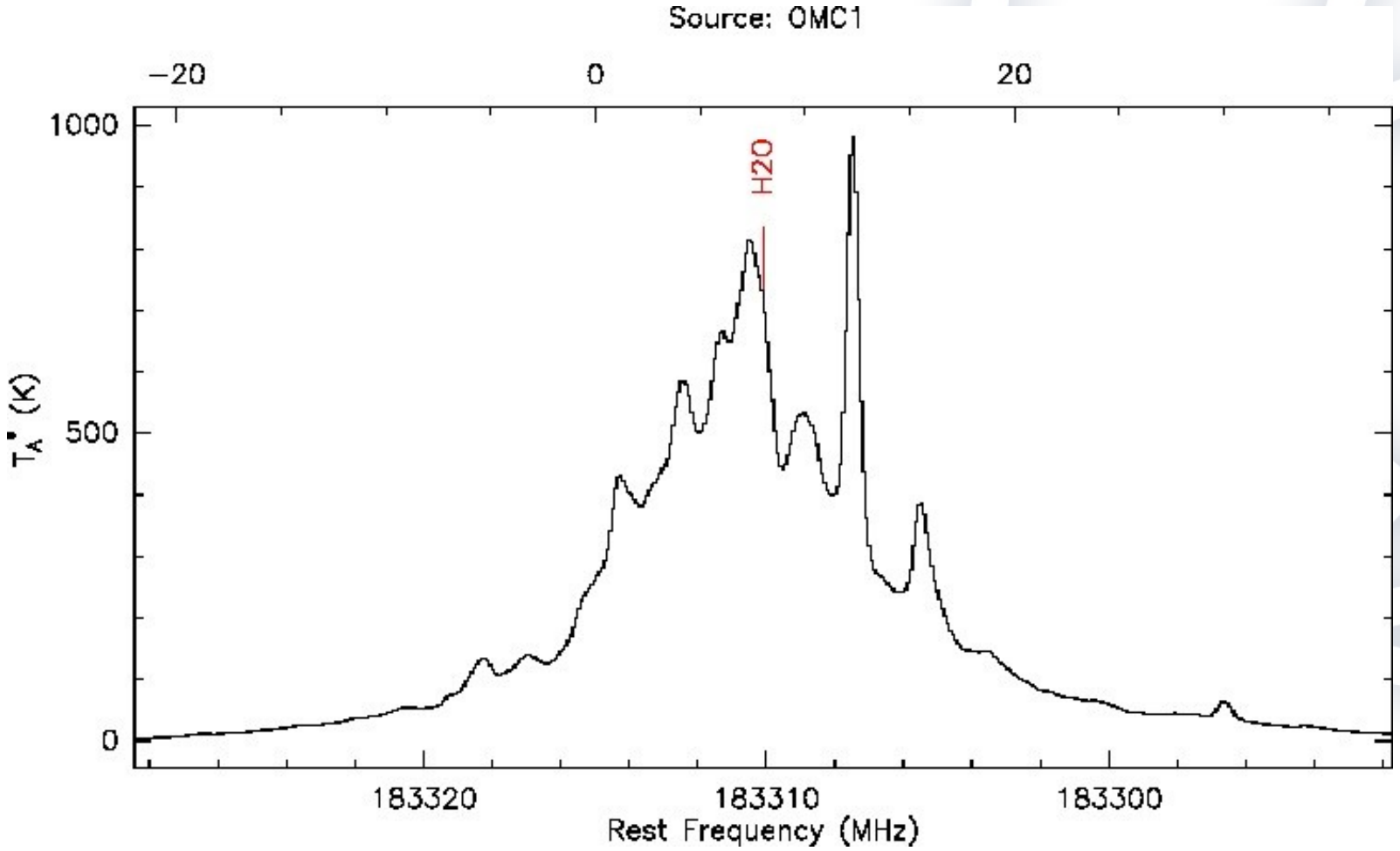


At the pub

Freezing

- Transaction ids are 4 byte ints
- Counter wraps every 4 billion xids
- Old rows get “frozen”
  - Replace xid with the FrozenTransactionId

Finale


Source: OMC1
−20
0
20
1000
500
0
H2O
T_A^* (K)
183320
183310
183300
Rest Frequency (MHz)

MVCC and Vacuum

- MVCC crucial to provide ACID properties
- It creates dead rows → needs maintenance
- VACUUM remove dead rows
- Autovacuum does it for you

Thanks and Remember Rule #6

Boriss Mejias
boriss.mejias@2ndquadrant.com
@tchorix