Good bye Sequence, hello Snowflake!

Jan Wieck
Principal Software Engineer

Basics of Sequences in PostgreSQL

- Sequences are used to generate unique identifiers.
- Sequences in PostgreSQL do not roll back.
- You cannot use a Sequence in PostgreSQL to create gap free identifiers.
- Sequences are also not allocated in chronological order according to any timestamp taken in the transaction.
- Never accept a design that requires the order to be strict.
- Use `bigserial` instead of `serial`

Sequences and Replication

- Do sequences need to be replicated?
  - Not really,
  - but what about failover?
  - We had a solution for this in previous systems.
- Are there other ways to avoid sequence replication?

Add problems of multi-master

- Network latency probably will prevent a central sequence generator.
- Every node needs its own number range.
- Allocating blocks of sequences is possible but requires bookkeeping.

```
CREATE SEQUENCE … START 1000 MAXVALUE 1999
```

- Need to monitor this.

Alternative number range

- Sequence with different `START` and `INCREMENT>1`
  Node1: `CREATE … START 1 INCREMENT 100;`
  Node2: `CREATE … START 2 INCREMENT 100;`

- Node ID visible in the last two digits.
- Maintenance free.
- Still can use `CACHE`.
- Has no relation to any order of allocation.

Snowflake Extension

https://github.com/pgEdge/snowflake

- Based on `int8` and a `sequence`.
- Designed as a drop-in replacement (WIP).
- Internally a 41 bit timestamp with ms resolution, a 10 bit node-ID and a 12 bit counter.

Snowflake Extension

- `snowflake.nextval(seqid regclass)`
- `snowflake.currval(seqid regclass)`
- `seqid` is optional (using a default sequence).
- `snowflake.format(int8)`

  `{"id": 1, "ts": "2024-04-05 19:46:51.022+00", "count": 0}`
- `snowflake.convert_sequence_to_snowflake(seqid)`
- `Does this create a somewhat usable order?`

Snowflake Example

```
db1=# CREATE TABLE t1 (
db1(#     id bigserial PRIMARY KEY,
db1(#     data integer
db1(# );
CREATE TABLE
db1=# SELECT
snowflake.convert_sequence_to_snowflake('t1_id_seq');
NOTICE:  EXECUTE ALTER TABLE public.t1 ALTER COLUMN id SET
DEFAULT snowflake.nextval('public.t1_id_seq'::regclass)
NOTICE:  ALTER SEQUENCE public.t1_id_seq NO CYCLE MAXVALUE 2
 convert_sequence_to_snowflake
-------------------------------
                              2
(1 row)
```

Snowflake Example

```
db1=# INSERT INTO t1 (data) SELECT generate_series(1, 10);
INSERT 0 10
db1=# select snowflake.format(id), data from t1 limit 4;
                        format                         | data
-------------------------------------------------------+------
 {"id": 1, "ts": "2023-01-01 00:00:00+00", "count": 1} |    1
 {"id": 1, "ts": "2023-01-01 00:00:00+00", "count": 2} |    2
 {"id": 1, "ts": "2023-01-01 00:00:00+00", "count": 3} |    3
 {"id": 1, "ts": "2023-01-01 00:00:00+00", "count": 4} |    4
(4 rows)
```

What’s next?

- Automatically change a `serial` or `bigserial` column into a `snowflake` on `CREATE TABLE` or `ALTER ADD COLUMN`.
- Any suggestions?
- Any questions?

pgEdge