Changing your huge table's data types in production

Jimmy Angelakos  
Senior PostgreSQL Architect  
EDB  

FOSDEM  
07/02/2021
Motivation

- Era of “Big Data”
- PostgreSQL seeing heavier usage
- PostgreSQL performance getting better
- Find your DB facing rapid growth
- Too rapid growth?
Why change types?

• Incorrect data type
  – VARCHAR(42) not enough → TEXT

• Non-optimal data type
  – TEXT id ‘12816750’ (9 bytes) vs INTEGER (4 bytes)

• Running out of IDs
  – Max INTEGER is +2,147,483,647
  – Whoops!
But I can change types!

• Yes, if they are compatible
• “Binary coercible”
  – No conversion function invocation
e.g. XML → TEXT (but not TEXT → XML)
• “Binary compatible”
  – Same internal representation
e.g. TEXT ↔ VARCHAR
ALTER TABLE ALTER TYPE

• ALTER TABLE ALTER `column_name` TYPE `data_type`

• USING `expression` needed if there is no implicit cast
  – May need to DROP DEFAULT & add a new one after

• Needs index rebuild
So what’s the problem?

- Requires ACCESS EXCLUSIVE lock
  - No reads, writes allowed to other transactions
- Effectively prevents usage of the table in production
- Causes table rewrite, if not binary coercible
  - S L O W
  - Requires double the disk space
Scenario

One possible scenario

- Huge table in production (1.7B rows)
- PK column is INT, rapidly approaching 2.1B rows
- BIGINT is seen as the solution
- Not binary compatible (8 bytes vs 4 bytes)
- Cannot be taken offline
What now?

One possible concurrent solution

- Add new BIGINT column
- Write procedure to copy values to new column in batches
- Write trigger to replicate changes from old column
- Drop old column, rename new column
- Make new column PK
Small details

- Need to create sequence for new PK
- Need to create index for new PK
- After conversion, perform all DDL in one transaction
- Minimum possible locking/blocking

- Test system: Intel i7-9750H, 64GB RAM, NVMe SSD
- Table with 1.7*10^9 rows of 170 bytes each
CREATE TABLE largetable (id INT NOT NULL, content TEXT);

INSERT INTO largetable
    SELECT i, 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur sodales arcu non pulvinar venenatis. Morbi ut enim efficitur.'
    FROM generate_series(1,1700000000) AS i;

INSERT 0 1700000000
Time: 1945398.859 ms (32:25.399)
CREATE SEQUENCE largetable_id_id_seq START 17000000001;
CREATE SEQUENCE
ALTER TABLE largetable
ALTER id SET DEFAULT nextval('largetable_id_id_seq');
ALTER TABLE
CREATE UNIQUE INDEX ON largetable(id);
CREATE INDEX
Time: 1585770.840 ms (26:25.771)
ALTER TABLE largetable
ADD PRIMARY KEY USING INDEX largetable_id_id_idx;
ALTER TABLE
Time: 8.534 ms
Data “in production” (i)

test=> \d largetable;
  Table "public.largetable"
  Column | Type | Nullable |                Default
  ------+------+----------+----------------------------------------
    id  | integer | not null | nextval('largetable_id_seq'::regclass)
  content | text |          |
Indexes:
  "largetable_id_idx" PRIMARY KEY, btree (id)

test=> \dt+ largetable;
  List of relations
  Schema | Name       | Type | Owner | Size  | Description
  ------+------------+------+-------+-------+-------------
    public | largetable | table | test  | 265 GB | (1 row)
### Data “in production” (ii)

```sql
test=> \textbf{TABLE} largetable \textbf{LIMIT} 5;

<table>
<thead>
<tr>
<th>id</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur s</td>
</tr>
<tr>
<td>2</td>
<td>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur s</td>
</tr>
<tr>
<td>3</td>
<td>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur s</td>
</tr>
<tr>
<td>4</td>
<td>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur s</td>
</tr>
<tr>
<td>5</td>
<td>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur s</td>
</tr>
</tbody>
</table>

(5 rows)

```sql
test=> \textbf{SELECT} \textbf{n\_live\_tup}
test-> \textbf{FROM} pg\_stat\_user\_tables \textbf{WHERE} relname='largetable';

<table>
<thead>
<tr>
<th>n_live_tup</th>
</tr>
</thead>
<tbody>
<tr>
<td>17000000000</td>
</tr>
</tbody>
</table>

(1 row)
```
Add the new column

• With zeros (instantaneous):

```
ALTER TABLE largetable
ADD COLUMN id_new BIGINT
NOT NULL
DEFAULT 0;
```

Time: 13.249 ms
Build the trigger function

- Replicates incoming changes while conversion is running

```sql
CREATE FUNCTION largetable_trig_func()
RETURNS TRIGGER AS $$
BEGIN
    NEW.id_new := NEW.id;
    RETURN NEW;
END $$ LANGUAGE plpgsql;
```

CREATE FUNCTION
Time: 21.512 ms
Add the trigger

- Replicates incoming changes while conversion is running

```sql
CREATE TRIGGER largetable_trig
BEFORE INSERT OR UPDATE ON largetable
FOR EACH ROW
EXECUTE FUNCTION largetable_trig_func();
```

Time: 12.576 ms
CREATE PROCEDURE largetable_sync_proc() AS $$
DECLARE r RECORD;
DECLARE count BIGINT := 0;
DECLARE batchsize BIGINT := 100000;
DECLARE cur CURSOR FOR SELECT id FROM largetable;
BEGIN
    FOR r IN cur LOOP
        UPDATE largetable
        SET id_new = id
        WHERE id = r.id;
        count := count + 1;
        IF (count % batchsize = 0) THEN
            COMMIT;
        END IF;
    END LOOP;
    COMMIT;
RETURN$$ LANGUAGE plpgsql;
... with progress notices

... BEGIN
  FOR r IN cur LOOP
    UPDATE largetable
    SET id_new = id
    WHERE id = r.id;
    count := count + 1;
    IF (count % batchsize = 0) THEN
      IF (count % (batchsize * 10) = 0) THEN
        RAISE NOTICE '% rows done', count;
      END IF;
      COMMIT;
    END IF;
  END LOOP;
  COMMIT;
END LOOP;
COMMIT;
RETURN;
END ...
test=> **CALL** largetable_sync_proc();
NOTICE: 10000000 rows done
NOTICE: 20000000 rows done
NOTICE: 30000000 rows done
...
...
...
<table>
<thead>
<tr>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>56000006</td>
</tr>
</tbody>
</table>

(1 row)

Thu 14 Jan 2021 09:09:43 GMT (every 1s)

<table>
<thead>
<tr>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>6900014</td>
</tr>
</tbody>
</table>

(1 row)
7 hours later...
test=> CALL largetable_sync_proc();
NOTICE: 10000000 rows done
NOTICE: 20000000 rows done
NOTICE: 30000000 rows done
...
...
...
NOTICE: 16980000000 rows done
NOTICE: 16990000000 rows done
NOTICE: 17000000000 rows done

CALL
Time: 25583914.664 ms (07:06:23.915)
Our table now looks like:

```sql
test=> TABLE largetable LIMIT 5;

<table>
<thead>
<tr>
<th>id</th>
<th>content</th>
<th>id_new</th>
</tr>
</thead>
<tbody>
<tr>
<td>100001</td>
<td>Lorem ipsum dolor sit amet, con...</td>
<td>100001</td>
</tr>
<tr>
<td>100002</td>
<td>Lorem ipsum dolor sit amet, con...</td>
<td>100002</td>
</tr>
<tr>
<td>100003</td>
<td>Lorem ipsum dolor sit amet, con...</td>
<td>100003</td>
</tr>
<tr>
<td>100004</td>
<td>Lorem ipsum dolor sit amet, con...</td>
<td>100004</td>
</tr>
<tr>
<td>100005</td>
<td>Lorem ipsum dolor sit amet, con...</td>
<td>100005</td>
</tr>
</tbody>
</table>
```

(5 rows)
CREATE UNIQUE INDEX CONCURRENTLY largetable_id_new_idx ON largetable(id_new);

CREATE INDEX
Time: 4662236.271 ms (01:17:42.236)
DO $$
DECLARE new_start BIGINT;
BEGIN
SELECT max(id) + 1 FROM largetable INTO new_start;
EXECUTE 'CREATE SEQUENCE largetable_id_bigint_seq ' 'START ' || new_start;
ALTER TABLE largetable ALTER id_new
SET DEFAULT nextval('largetable_id_bigint_seq');
ALTER TABLE largetable DROP id;
ALTER TABLE largetable RENAME id_new TO id;
ALTER TABLE largetable ADD CONSTRAINT largetable_id_pkey
PRIMARY KEY USING INDEX largetable_id_new_idx;
DROP TRIGGER largetable_trig ON largetable;
COMMIT;
END $$ LANGUAGE plpgsql;
NOTICE: ALTER TABLE / ADD CONSTRAINT USING INDEX will rename index "largetable_id_new_idx" to "largetable_id_pkey"

Time: 451.049 ms
Thank you =)
Twitter: @vyruss

Photo: River Broom Valley, Northwest Highlands, Scotland