PGCONF.EU 2018, LISBON | OCTOBER 24, 2018

Data Modeling, Normalization and Denormalisation

Dimitri Fontaine Citus Data POSTGRESQL MAJOR CONTRIBUTOR

PostgreSQL



CURRENTLY WORKING AT

Citus Data

https://masteringpostgresql.com

Mastering PostgreSQL In Application Development



In Application Development



Dimitri Fontaine, PostgreSQL Major Contributor Ist Edition https://masteringpostgresql.com

Mastering PostgreSQL In Application Development -15% "pgconfeu2018"

Mastering PostgreSQL

In Application Development



Dimitri Fontaine, PostgreSQL Major Contributor Ist Edition





ROB PIKE, NOTES ON PROGRAMMING IN C

Rule 5. Data dominates.

"If you've chosen the right data structures and organized things well, the algorithms will almost always be self-evident. Data structures, not algorithms, are central to programming."

(Brooks p. 102)

Avoiding Database Anomalies

Update Anomaly

Employees' Skills

Employee ID	Employee Address	Skill
426	87 Sycamore Grove	Typing
426	87 Sycamore Grove	Shorthand
519 <	94 Chestnut Street	Public Speaking
519 <	96 Walnut Avenue	Carpentry

Insertion Anomaly

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201

•				
I.	424	Dr. Newsome	29-Mar-2007	?
Ι.				

Deletion anomaly

Faculty and Their Courses

Faculty ID	Faculty Name	Faculty Hire Date	Course Code
389	Dr. Giddens	10-Feb-1985	ENG-206
407	Dr. Saperstein	19-Apr-1999	CMP-101
407	Dr. Saperstein	19-Apr-1999	CMP-201



ANOTHER QUOTE FROM FRED BROOKS

Database Design and User Workflow

"Show me your flowcharts and conceal your tables, and I shall continue to be mystified. Show me your tables, and I won't usually need your flowcharts; they'll be obvious."

Tooling for Database Modeling

BEGIN;

```
create schema if not exists sandbox;
```

```
create table sandbox.category
 (
    id serial primary key,
    name text not null
 );
```

```
insert into sandbox.category(name)
    values ('sport'),('news'),('box office'),('music');
```

ROLLBACK;

Object Relational Mapping

- The R in ORM stands for relation
- Every SQL query result set is a relation



Object Relational Mapping

When mapping base tables, you end up trying to solve different complex issues at the same time

- User Workflow
- Consistent view of the whole world at all time

Normalization

Basics of the Unix Philosophy: principles

Clarity

• Clarity is better than cleverness

Simplicity

 Design for simplicity; add complexity only where you must.

Transparency

 Design for visibility to make inspection and debugging easier.

Robustness

• Robustness is the child of transparency and simplicity.

1st Normal Form, Codd, 1970

- There are no duplicated rows in the table.
- Each cell is single-valued (no repeating groups or arrays).
- Entries in a column (field) are of the same kind.

2nd Normal Form, Codd, 1971

"A table is in 2NF if it is in 1NF and if it has no partial dependencies."

"A table is in 2NF if it is in 1NF and if all nonkey attributes are dependent on all of the key. A partial dependency occurs when a non-key attribute is dependent on only a part of the composite key." Third Normal Form, Codd, 1971 BCNF, Boyce-Codd, 1974

- A table is in 3NF if it is in 2NF and if it has no transitive dependencies.
- A table is in BCNF if it is in 3NF and if every determinant is a candidate key.

More Normal Forms

- Each level builds on the previous one.
- A table is in **4NF** if it is in BCNF and if it has no multivalued dependencies.
- A table is in **5NF**, also called "Projection-join Normal Form" (**PJNF**), if it is in 4NF and if every join dependency in the table is a consequence of the candidate keys of the table.
- A table is in **DKNF** if every constraint on the table is a logical consequence of the definition of keys and domains.

Database Constraints

Primary Keys

create table sandbox.article

id bigserial primary key, category integer references sandbox.category(id), pubdate timestamptz, title text not null, content text

Surrogate Keys

Artificially generated key is named a surrogate key because it is a
substitute for natural key.
A natural key would allow preventing duplicate entries in our data set.

Surrogate Keys

insert into sandbox.article
 (category, pubdate, title)
 values (2, now(), 'Hot from the Press'),
 (2, now(), 'Hot from the Press')
 returning *;

Oops. Not a Primary Key.

```
-[ RECORD 1 ]-
id
          3
category 2
pubdate | 2018-03-12 15:15:02.384105+01
title
         | Hot from the Press
content
-[ RECORD 2 ]
id
           4
category | 2
pubdate | 2018-03-12 15:15:02.384105+01
title
         Hot from the Press
content
```

INSERT 0 2

Natural Primary Key

```
create table sandboxpk.article
```

```
(
```

- category integer references sandbox.category(id),
 pubdate
- pubdate timestamptz,
- title text not null,
- content text,

```
primary key(category, pubdate, title)
);
```

Update Foreign Keys

create table sandboxpk.comment

a_category integer not null, a_pubdate timestamptz not null, a_title text not null, pubdate timestamptz, content text,

primary key(a_category, a_pubdate, a_title, pubdate, content),

foreign key(a_category, a_pubdate, a_title)
 references sandboxpk.article(category, pubdate, title)
);

Natural and Surrogate Keys

create table sandbox.article

id	integer	generated always as identity,
category	integer	<pre>not null references sandbox.category(id),</pre>
pubdate	timestamptz	not null,
title	text	not null,
content	text,	

```
primary key(category, pubdate, title),
unique(id)
```

);

Other Constraints

Normalisation Helpers

);

- Primary Keys
- Foreign Keys
- Not Null
- Check Constraints
- Domains
- Exclusion
 Constraints

create table rates
 (
 currency text,
 validity daterange,
 rate numeric,

exclude using gist
(
 currency with =,
 validity with &&
)

Denormalization

Rules of Optimisation

@pleb

Rules of Optimization: Rule 1: Don't do it. Rule 2: Don't do it yet(experts only)







DONALD KNUTH

Premature Optimization...

"Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We should forget about small efficiencies, say about 97% of the time: **premature optimization is the root of all evil**. Yet we should not pass up our opportunities in that critical 3%."

"Structured Programming with Goto Statements" Computing Surveys 6:4 (December 1974), pp. 261–301, §1.

Denormalization: cache

- Duplicate data for faster access
- Implement cache invalidation

Denormalization example

\set season 2017

select drivers.surname as driver, constructors.name as constructor, sum(points) as points

from results

join races using(raceid)

join drivers using(driverid)

join constructors using(constructorid)

where races.year = :season

group by grouping sets(drivers.surname, constructors.name)
 having sum(points) > 150
order by drivers.surname is not null, points desc;

Denormalization example

```
create view v.season_points as
  select year as season, driver, constructor, points
    from seasons left join lateral
            select drivers.surname as driver,
                   constructors.name as constructor,
                   sum(points) as points
              from results
                   join races using(raceid)
                   join drivers using(driverid)
                   join constructors using(constructorid)
             where races.year = seasons.year
          group by grouping sets(drivers.surname, constructors.name)
          order by drivers.surname is not null, points desc
          as points on true
order by year, driver is null, points desc;
```

Materialized View

create materialized view cache.season_points as
 select * from v.season_points;

create index on cache.season_points(season);

Materialized View

refresh materialized view cache.season_points;

Application Integration

select driver, constructor, points from cache.season_points where season = 2017

and points > 150;

Denormalization: audit trails

- Foreign key references to other tables won't be possible when those reference changes and you want to keep a history that, by definition, doesn't change.
- The schema of your main table evolves and the history table shouldn't rewrite the history for rows already written.

History tables with JSONB

create schema if not exists archive;

create type archive.action_t
 as enum('insert', 'update', 'delete');

```
create table archive.older_versions
(
   table_name text,
   date   timestamptz default now(),
   action   archive.action_t,
   data   jsonb
);
```

Validity Periods

```
create table rates
 (
  currency text,
  validity daterange,
  rate numeric,
```

Validity Periods

select currency, validity, rate
 from rates
 where currency = 'Euro'
 and validity @> date '2017-05-18';

-[RECORD 1]----currency | Euro
validity | [2017-05-18,2017-05-19)
rate | 1.240740

Denormalization Helpers: Data Types

Composite Data Types

- Composite Type
- Arrays
- JSONB
- Enum

- hstore
- ltree
- intarray
- pg_trgm

Partitioning

Partitioning Improvements

PostgreSQL 10

- Indexing
- Primary Keys
- On conflict
- Update Keys

PostgreSQL 11

- Indexing, Primary Keys, Foreign Keys
- Hash partitioning
- Default partition
- On conflict support
- Update Keys



Schemaless with JSONB

Durability Trade-Offs

create role dbowner with login; create role app with login;

create role critical with login in role app inherit; create role notsomuch with login in role app inherit; create role dontcare with login in role app inherit;

alter user critical set synchronous_commit to remote_apply; alter user notsomuch set synchronous_commit to local; alter user dontcare set synchronous_commit to off;

Per Transaction Durability

SET demo.threshold TO 1000;

```
CREATE OR REPLACE FUNCTION public.syncrep_important_delta()
RETURNS TRIGGER
LANGUAGE PLpgSQL
```

```
AS
```

\$\$ DECLARE

```
threshold integer := current_setting('demo.threshold')::int;
delta integer := NEW.abalance - OLD.abalance;
BEGIN
```

```
IF delta > threshold
```

```
THEN
```

```
SET LOCAL synchronous_commit TO on;
END IF;
RETURN NEW;
END;
$$;
```



Horizontal Scaling

Sharding with Citus

Five Sharding Data Models and which is right?



- Sharding by Geography
- Sharding by EntityId
- Sharding a graph
- Time Partitioning

PGCONFEU 2018, LISBON | OCTOBER 24, 2018

Ask Me Two Questions!

Dimitri Fontaine Citus Data

The Art of PostgreSQL

Turn Thousands of Lines of Code into Simple Queries