An Adventure in Data Modeling

The Entity-Attribute-Value Data Model

Mark Wong
PostgreSQL Major Contributor
markwkm@postgresql.org
EDB Performance Engineer
mark.wong@enterprisedb.com

SCaLE 19x, Los Angeles, California

July 2022

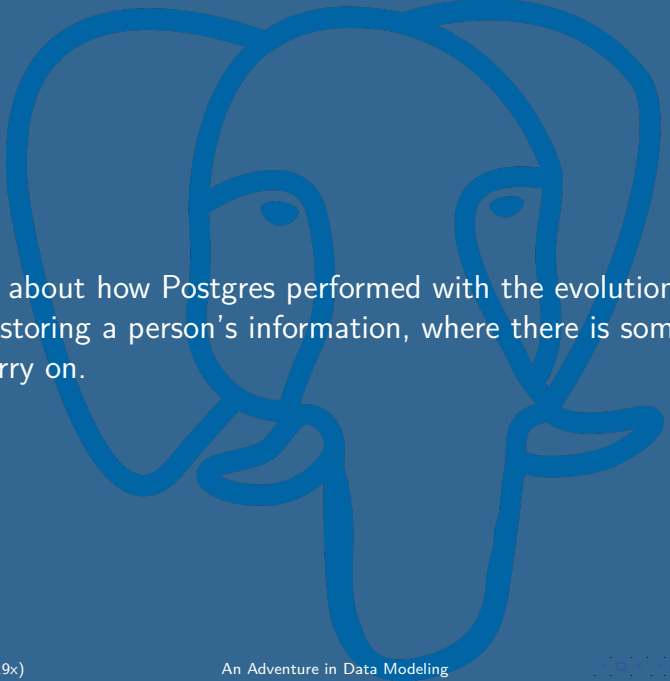
Agenda

- Introductions — Who am I?
- The Data Modeling Story
  - The original data model
  - What is EAV
  - How well it worked
  - Open questions

Introduction

- Employed by EDB `https://enterprisedb.com` 
- PostgreSQL Major Contributor
- Director at United States PostgreSQL Association `https://postgresql.us`
- Portland PostgreSQL Users Group `https://meetup.com/pdxpug/`
- PostgreSQL Exhibitions (North America)

Synopsis



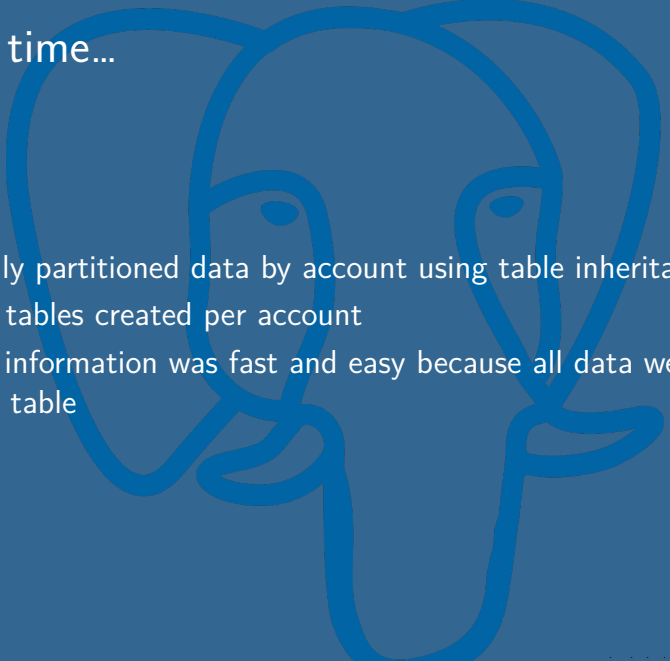
This is a story about how Postgres performed with the evolution of the data model around storing a person's information, where there is some stumbling, and how to carry on.

What kind of information?

A person's information includes an email address and any additional attributes that the customer wants to track, for example:

- first name
- last name
- favorite database

Once upon a time...



- Horizontally partitioned data by account using table inheritance
- 12+ child tables created per account
- Exporting information was fast and easy because all data were contained in a single table

Example of exporting information

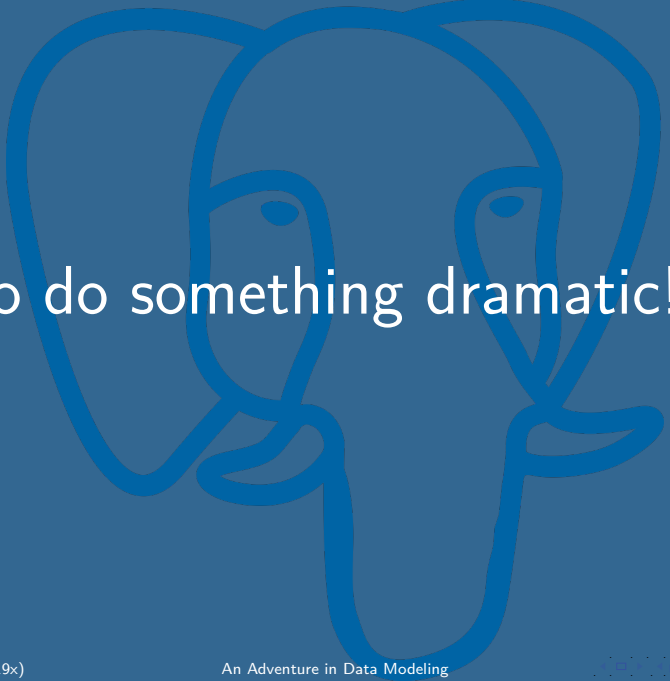
- Information on the PostgreSQL Contributors (not verified):

```
COPY info_1996
TO 'info-1996.csv' (FORMAT CSV);
```

email	first_name	last_name	favorite_dbms
laetitia.avrot at gmail.com	Lætitia	Avrot	PostgreSQL
lubennikovaav at gmail.com	Anastasia	Lubennikova	PostgreSQL
shayslerpgx at gmail.com	Stacey	Haysler	PostgreSQL

What was wrong?

- Over 40,000 accounts in the system
- Hard to mine data
  - At least 40,000 locks, one per child table
  - Compounded by joining with other tables
- Administration related issues
  - Over one million objects in the system (tables, indexes, sequences, etc.)
  - `ALTER TABLE` required to add or remove an attribute
  - Backups with **pg_dump** takes more than 24 hours



Time to do something dramatic!

Goals

Highlighting a few of the changes that occurred:

- Apply the Entity-Attribute-Value data model
- Horizontally partition with a hash
- Target 1 GB of data per partition

Entity-attribute-value model (EAV) is a data model to describe entities where the number of attributes (properties, parameters) that can be used to describe them is potentially vast, but the number that will actually apply to a given entity is relatively modest.

...

EAV is also known as object-attribute-value model, vertical database model and open schema.

`https://en.wikipedia.org/wiki/Entity-attribute-value_model`

What was known before applying EAV

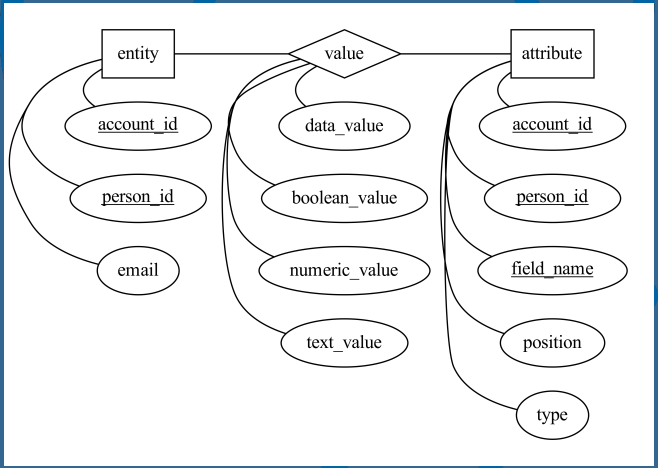
- Pros
  - Simple three table model
  - Use cheaper DML statements instead of expensive DDL statements when adding or removing member attributes
- Cons
  - Data will need to be queried differently
  - Data type checking either done using multiple tables or multiple columns (opted for latter)

EAV table descriptions

Three tables make up the model:

- Entity: an entity table containing attributes that all individuals must have, i.e. unique identifiers
- Attribute: an entity table containing the custom attributes that users defines, e.g. favorite database management system
- Value: a relationship table containing the values of defined attributes

EAV ER Diagram



Application Pivots Data

Before pivot:

email	field_name	value
rl at pcorp.us	first_name	Regina
rl at pcorp.us	last_name	Obe
rl at pcorp.us	favorite_dbms	PostgreSQL

After pivot:

email	first_name	last_name	favorite_dbms
ashutosh.bapat.oss at gmail.com	Ashutosh	Bapat	PostgreSQL

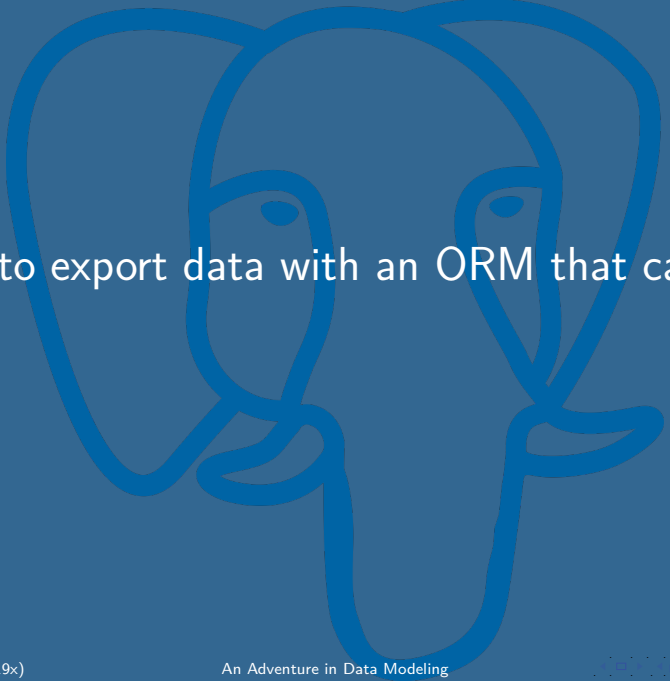
How much data are we dealing with?

Data set sizes

account	people	fields	values
1	1,000,000	120	120,000,000
2	2,000,000	50	100,000,000
3	700,000	100	70,000,000

How long to export data?

Digression for application developers.



How long to export data with an ORM that can pivot data?

Data pivots with an ORM

account	people	fields	values	runtime
1	1,000,000	120	120,000,000	DNF
2	2,000,000	50	100,000,000	DNF
3	700,000	100	70,000,000	4 Hours

How long to export data? (using `crosstab`)

`https://www.postgresql.org/docs/current/tablefunc.html`

Data pivots with `crosstab`

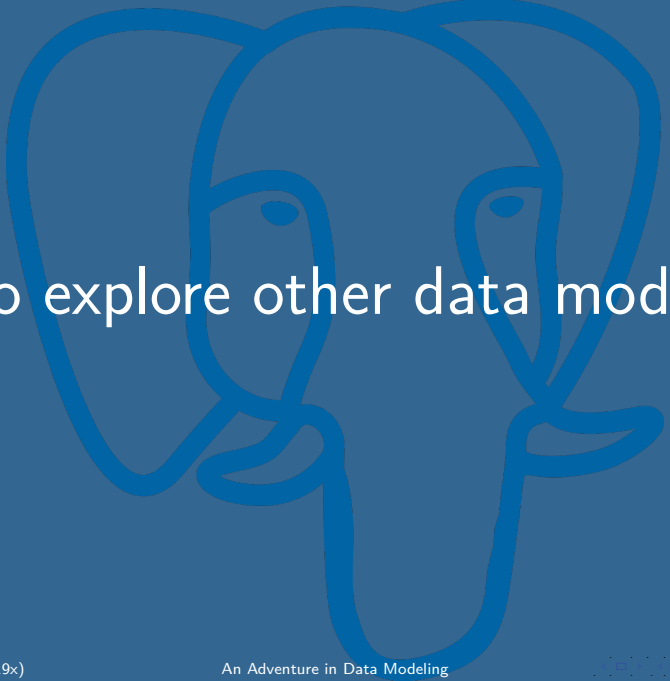
account	people	fields	values	orm	crosstab
1	1,000,000	120	120,000,000	DNF	22 min
2	2,000,000	50	100,000,000	DNF	17 min
3	700,000	100	70,000,000	4 Hours	10 min

Not all obstacles have been removed

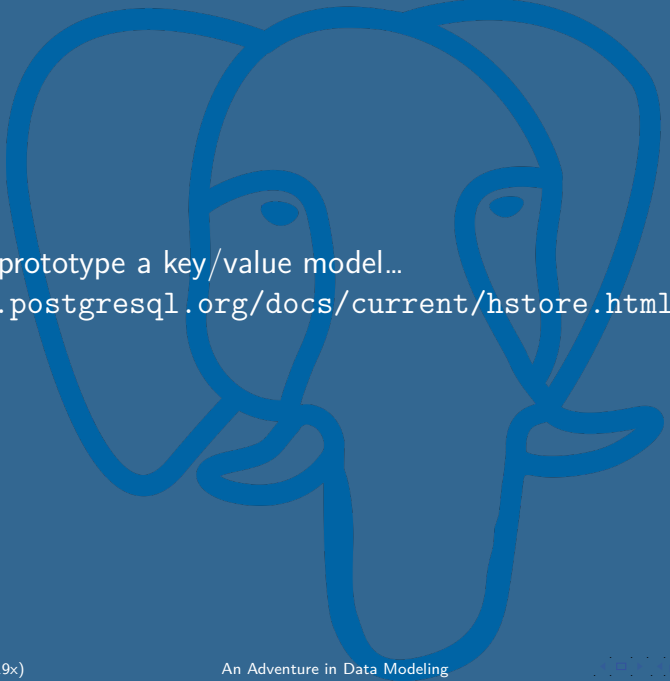
- Exports expected to take too long if accounts grow somewhere between 5 to 10 million members
- Importing member data faces similar challenges in order to perform well

What we knew after having EAV

- Retrieving data from EAV model is inefficient
- Performance issues begin when pivoting millions of rows+



Time to explore other data models



Use *hstore* to prototype a key/value model...
`https://www.postgresql.org/docs/current/hstore.html`

Cons to *hstore* data type

Things to note before going in:

- No strict types; everything is a string
- No referential integrity constraints; cannot create a foreign key between an *hstore* key and a table column
- Native support may vary in higher level database connectivity libraries

What does *hstore* look like

Put the **attributes** into the **entity** table as the *hstore* column field.

email	field
obartunov at gmail.com	"first_name"=>"Oleg", "last_name"=>"Bartunov", "favorite_dbms"=>"PostgreSQL"

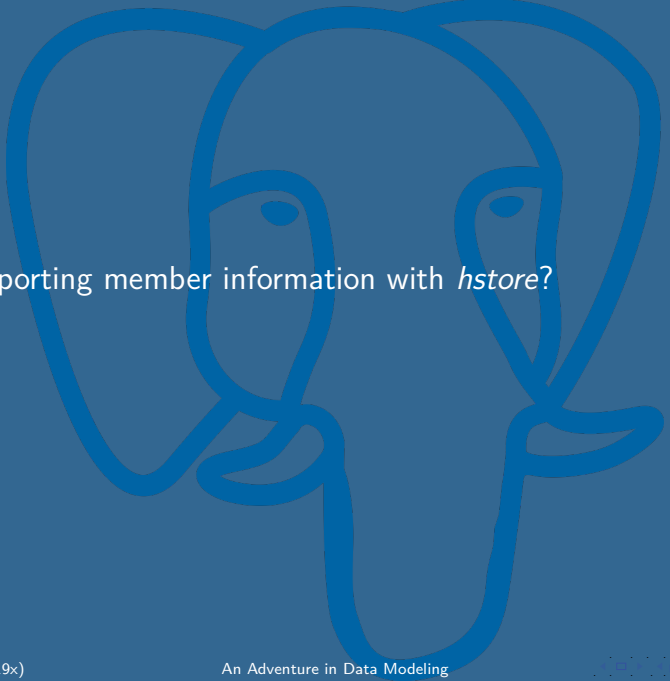
Converting to *hstore* seems fast

Approximately 2 minutes to transform a single partition:

```
WITH u AS (
    WITH t AS (
        SELECT member_id, shortcut_name,
               CASE WHEN f.field_type = 'text' THEN mf.text_value
                    WHEN f.field_type = 'numeric' THEN mf.numeric_value::TEXT
                    WHEN f.field_type = 'boolean' THEN mf.boolean_value::TEXT
                    WHEN f.field_type = 'date' THEN date_value::TEXT
                    ELSE NULL END AS value
        FROM field f, member_field mf
        WHERE f.field_id = mf.field_id
    )
    SELECT member_id,
           string_agg(hstore(shortcut_name, value)::TEXT, ',')::HSTORE AS hst
    FROM t GROUP BY member_id
)
UPDATE member
SET field = hst
FROM u
WHERE u.member_id = member.member_id;
```

Exporting member information with *hstore*

```
COPY (
    SELECT email,
           field -> 'name_first' AS first_name,
           field -> 'name_last' AS last_name,
           field -> 'favorite_dbms' AS favorite_dbms
    FROM member m
    WHERE m.account_id = 1986
)
TO 'info-1996.'csv (FORMAT CSV);
```



How fast is exporting member information with *hstore*?

Exporting member information is pretty fast

account	people	fields	values	orm	crosstab	hstore
1	1,000,000	120	120,000,000	DNF	22 min	???
2	2,000,000	50	100,000,000	DNF	17 min	???
3	700,000	100	70,000,000	4 Hours	10 min	15 sec

Changes in disk space utilization

For just one of our partitions:

- EAV:
  - **entity** table; 1,000,000 rows; 334 MB
  - **value** table; 6,000,000 rows; 843 MB
- Non EAV:
  - **entity** table size 505 MB
  - Net change: -672 MB, 43% reduction in disk space consumption (including indexes)

Final Thoughts

- No perfect solution
- EAV's performance significant issue
- Unstructured data can help
- Tranform data in the database
- Yet another data model?

Thank you!