If you can't Beat 'em, Join 'em!

Integrating NoSQL data elements into a relational model

This presentation and SQL file is on SlideShare and PGCon2015

http://www.slideshare.net/jamesphanson/pg-no-sqlbeatemjoinemv10sql

Jamey Hanson
jhanson@freedomconsultinggroup.com
jamesphanson@yahoo.com
@jamey_hanson

Freedom Consulting Group
http://www.freedomconsultinggroup.com

PGCon 2015,
Ottawa, ON CA
19-Jun-2015

NoSQL hype check

NoSQL is ~~magic~~ ~~a panacea~~ 42 ~~a hot mess~~
really useful in some situations, not applicable in other situations - and here to stay.

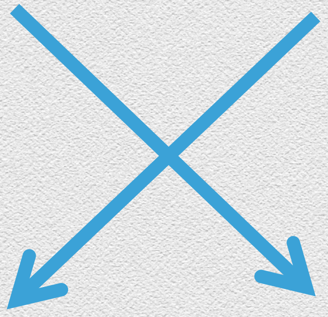
Q: How can PostgreSQL *thrive* in a mixed NoSQL environment?

A: By integrating NoSQL data types and features plus understanding where PostgreSQL is - and is not - a good fit.

Stages of NoSQL acceptance …

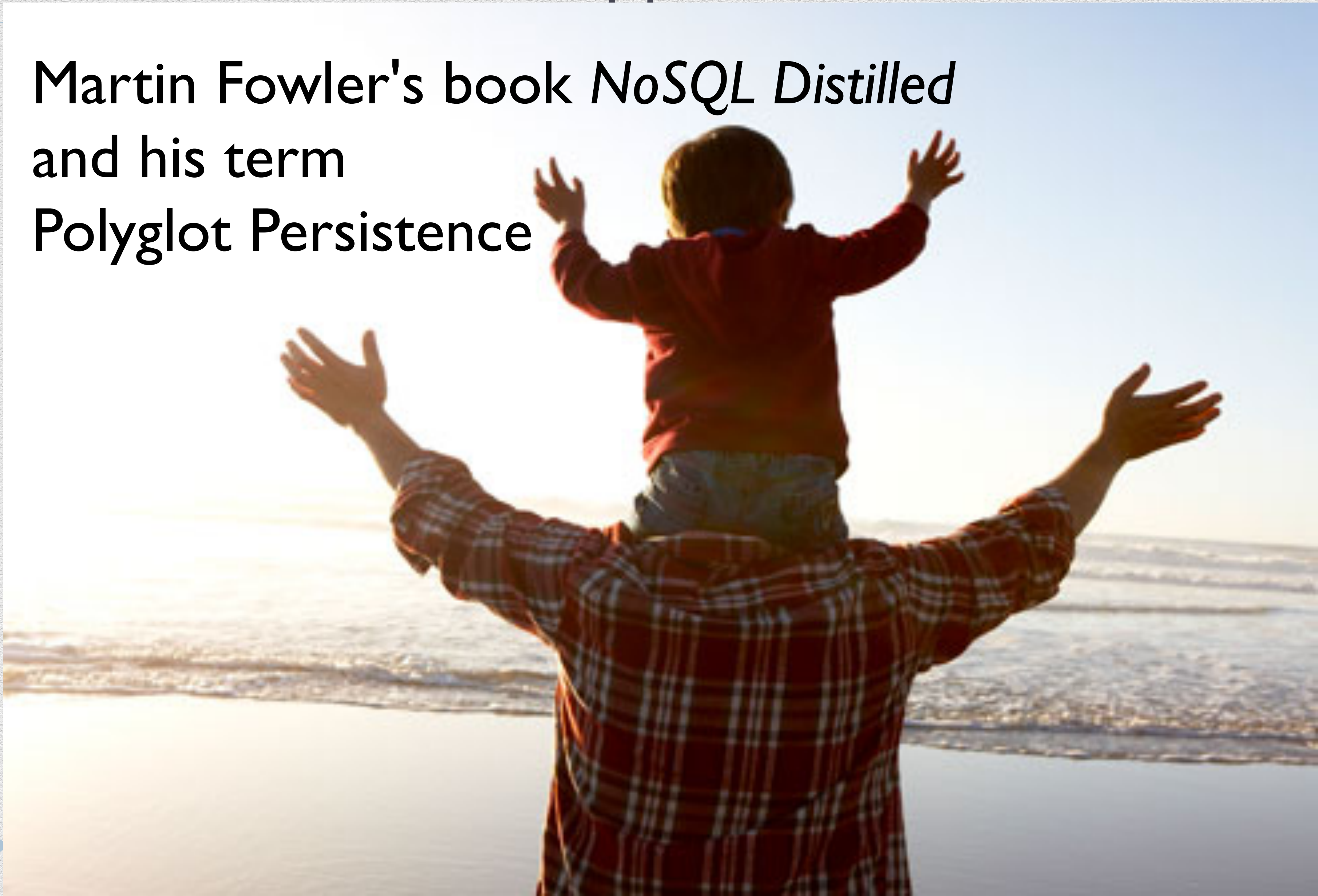
Reverse the traditional approach

Given that I have PostgreSQL,
how can I leverage NoSQL data?



Given that I have NoSQL data,
how can I leverage PostgreSQL?

The framework and approach come from



Martin Fowler's book *NoSQL Distilled*
and his term
Polyglot Persistence

About the author

Jamey Hanson
jhanson@freedomconsultinggroup.com
jamesphanson@yahoo.com

Manage a team for Freedom Consulting Group migrating applications from Oracle to Postgres Plus Advanced Server and PostgreSQL in the government space. We are subcontracting to EnterpriseDB

~~Overly~~ certified: PMP, CISSP, CSEP, OCP in 5 versions of Oracle, Cloudera developer & admin. Used to be NetApp admin and MCSE. I teach PMP and CISSP at the Univ. MD training center

Alumnus of multiple schools and was C-130 aircrew

A. JAMES CLARK SCHOOL OF ENGINEERING

PITT | SWANSON ENGINEERING

Carnegie Mellon Tepper SCHOOL OF BUSINESS

What is NoSQL … for the next 45 min?

- Document store (MongoDB)
- Wide column store (Cassandra)*
  a.k.a. Column family database
- Key-value store (Redis)
- Graph DBMS (Neo4j)
- Search engine (Solr)**

Categories adapted from db-engines.com and Martin Fowler, martinfowler.com/nosql.html

* Not covered in this presentation.

** See PGConf 2015 NYC "Full Text Search with Ranked Results"

Why NoSQL (vs. RDBMS/PostgreSQL)? 1

- Too much data for a single machine to process. RDBMS is a "single-or-few" machine architecture.*

  NoSQL has expectations of sharding across large cluster while RDBMS does not.*

  - Shard – divide database into aggregates of related business data and spread the entire database across a cluster.
    Sharding incorporates (changes to) application design.

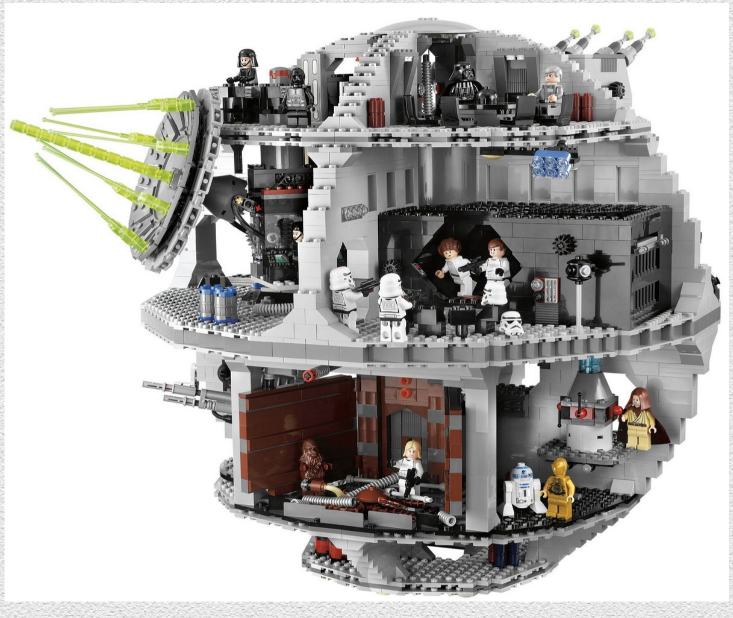
* Sharded RDBMSs have been developed but they are more difficult than NoSQL sharding and have not been as successful.

Why NoSQL (vs. RDBMS/PostgreSQL)? 2

- Because RDBMSs store data in very small pieces in lots of different places, which does not match object-oriented methodology and is inconvenient for developers.
  - A.k.a. Object-relational Impedance Mismatch, which is handled by Object Relational Mapping (ORM) tools such as Hibernate.

RDBMS

NoSQL



Why NoSQL (vs. RDBMS/PostgreSQL)? 3

- Because RDBMS data models are difficult to modify as data structures and business needs change.
  - RDBMS models must be consistent for all the data in a table. It is not possible to have legacy data use on structure, new data use a different structure and keep them all in the same table(s).

Why RDBMS/PostgreSQL (vs. NoSQL)?

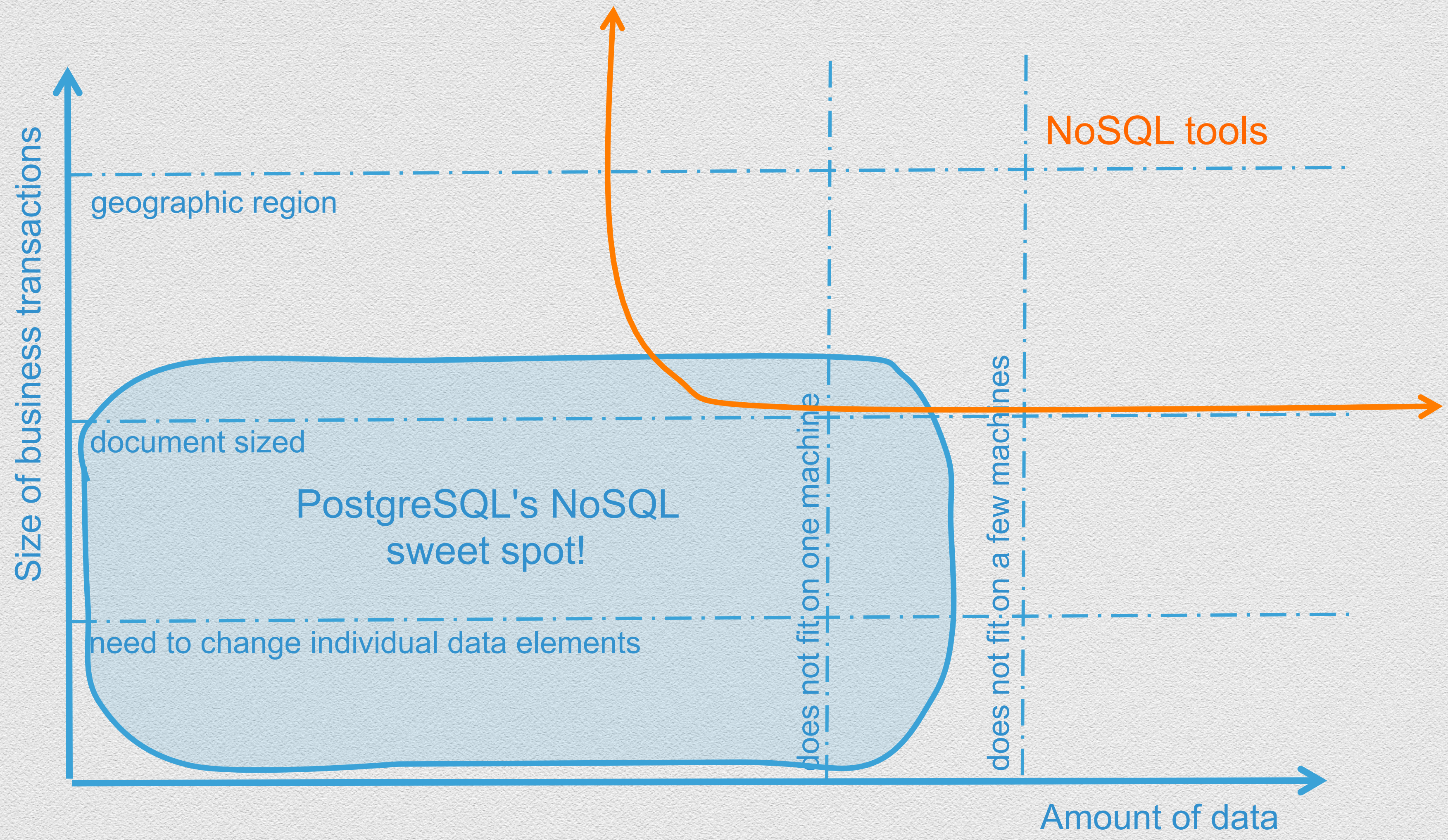
- Because RDBMS is the incumbent.
  - Installed everywhere, widely understood, mature technology that still has active development – such as this conference.
- Because RDBMS have transactions and a consistent view of the data.
  - Sometimes you need to change a small piece of data and you need every connection to see that change instantly.
- Because most data sets are *not* Google-sized.
  - A single machine easily can process terabytes of data.
- Because RDBMS are better at finding relationships* and enforcing data integrity. (*Graph databases are an exception.)
  - Sometimes you *want* to stop bad-data from loading.

Q: Where does this leave us?

- A: With *Polyglot Persistence**.
  Organizations will have relational *and* NoSQL databases … our job is to match the business needs + data to the technology.
  * *Polyglot Persistence* was coined by Martin Fowler.
  It refers to using multiple database tools and architectures.
- This presentation is about identifying where PostgreSQL is a great fit and demonstrating how to integrate NoSQL data into PostgreSQL's relational model.

PostgreSQL can *thrive* – not just survive – in a world that includes NoSQL.

PostgreSQL NoSQL data sweet spot


NoSQL tools
geographic region
document sized
PostgreSQL's NoSQL sweet spot!
need to change individual data elements
does not fit on one machine
does not fit on a few machines
Size of business transactions
Amount of data

On to the technical parts …

- Scenario: You are given ~ 1 million JSON files and need to decide how to handle them. (i.e. Do I need MongoDB?)
  - Q: Can you process this data on a single/few servers?
    A: Easily!
  - Q: How large are the business transactions?
    (element-level, document-level or other?)
    A: I have no idea.*

- Perfect for PostgreSQL integrating NoSQL data.

* PostgreSQL can handle most answers.
NoSQL can (generally) only handle document-level or larger transactions.

What is JSON?

- JavaScript Object Notation:
  A widely accepted, human readable open standard for transmitting optionally-nested key-value pairs.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY"
    "postalCode": "10021-3100"
  }...
```

Also applies to XML … but it's not as cool

What's in our JSON files?

- 10,000 files from the Million Song Database (MSD)
  - http://labrosa.ee.columbia.edu/millionsong/lastfm
  - http://labrosa.ee.columbia.edu/millionsong/sites/default/files/lastfm/lastfm_subset.zip
- Each file includes the song's:
  - track_id
  - artist
  - title
  - 0-*N* key-value pairs of similar track_id's and weights.
  - 0-*N* key-value pairs of song tags and weights.

How do I load JSON files into PostgreSQL?

- Create a table with JSONB* data type.

```
CREATE TABLE j_songs (
 id SERIAL PRIMARY KEY,
 song JSONB
);
```

- Use COPY command to load each file.

```
COPY j_songs (song) FROM '/NoSQL/TRAAFD.json'
CSV QUOTE e'\x01' DELIMITER e'\x01';
```

NOTE: The `e'\x01'` parameter handles embedded quotes.

*There are very few reasons to use JSON over JSONB

How do I load JSON files into PostgreSQL?

- Requires some Linux work … but not too bad.

- Extract JSON files into OS postgres's `~/NoSQL`

```
$ unzip ~/NoSQL/lastfm_subset.zip
```

- Create a symbolic link for each JSON file from `~/NoSQL` to `$PGDATA/ExtFiles`

```
$ find ~/NoSQL/lastfm_subset -name *.json|
xargs -i ln -s {} $PGDATA/ExtFiles/
```

How do I load JSON files into PostgreSQL?

- Use `pg_ls_dir`* to generate the file-loading SQL

```
SELECT 'COPY nosql.j_songs(song) FROM
  ''ExtFiles/' || pg_ls_dir('ExtFiles') ||
  ''' CSV QUOTE e''\x01''
  DELIMITER e''\x02'';';
```

* `pg_ls_dir` can list directory contents under `$PGDATA`.
This is why we created symbolic links in under `$PGDATA`
We could also have used `COPY` command in the files' original location.

Switch to SQL interactive
Prepare and loading JSON files.

NOTE: The SQL statements are in the file PG_NOSQL_BeatEmJoin_vXX.sql, which is loaded in SlideShare and the PGCon web page.

Exploring JSON data

- Return tags

```
SELECT DISTINCT jsonb_object_keys(song)...
```

- Return values

```
SELECT
   song ->> 'title' AS title, -- return TEXT
   song -> 'artist' AS artist, -- return JSON
FROM j_songs ...
```

- Match tags

```
WHERE song @> '{"artist":"Arctic Monkeys"}'::JSONB
```

Switch to SQL interactive
Exploring JSON data and indexing

Present JSON as an RDBMS relation

- Some interfaces – and a lot of existing code – require an RDBMS structure.
  - JPA (a.k.a. Hibernate) SQL cannot interact with non-RDBMS structures.
- Present JSON data as a view or materialized view.

```
CREATE OR REPLACE VIEW v_songs AS
SELECT
  song ->> 'track_id' AS track_id,
  song ->> 'artist' AS artist,
  song ->> 'title'
```

Switch to SQL interactive
Presenting JSON as a view and/or materialized view

Transform `tags` and `similars` to HSTORE

- The `tags` and `similars` JSON elements contain arrays of key-value-pairs.
  - Convert them to HSTORE
  - `track_id`, `artist` and `title` are present in every JSON file – so we can turn them into columns.

```
CREATE TABLE h_songs (
  track_id TEXT PRIMARY KEY,
  artist TEXT,
  title TEXT,
  tags HSTORE,
  similars HSTORE);
```

FREEDOM consulting group

Use JSON and HSTORE operators to convert

- Return elements in the arrays with **jsonb_array_elements**

```
jsonb_array_elements (song -> 'tags') ->> 0
   AS tag_key,
```

- Build the HSTORE column with **HSTORE** and **array_agg** operators.

```
HSTORE (array_agg(tag_key), array_agg(tag_value))
```

Switch to SQL interactive
Convert from JSON to HSTORE

HSTORE also has operators and indexes

- Select records with a specific tag and value.

```
SELECT
   artist,
   title,
   tags -> 'latin'
FROM h_songs
WHERE
   tags ? 'latin'
   and (tags -> 'latin')::INTEGER > 67;
```

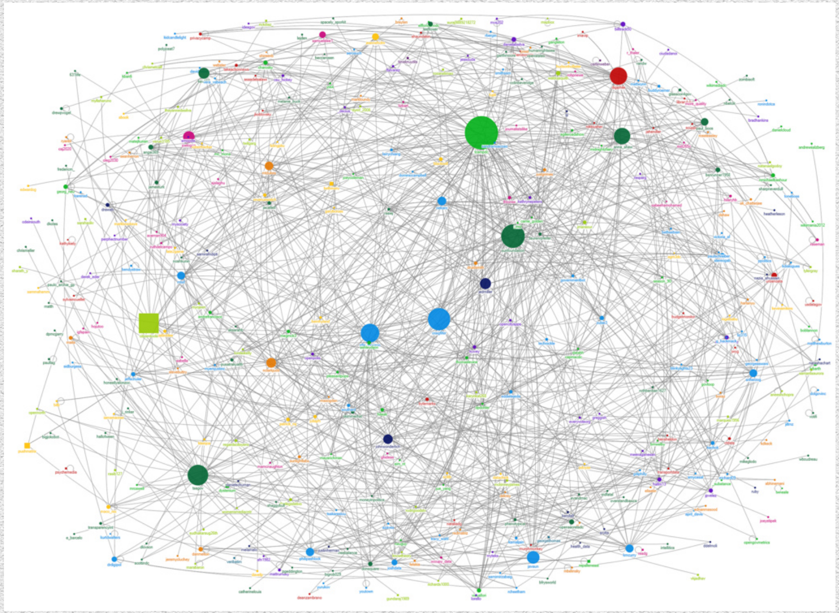
Switch to SQL interactive

Explore and index HSTORE key-value pairs

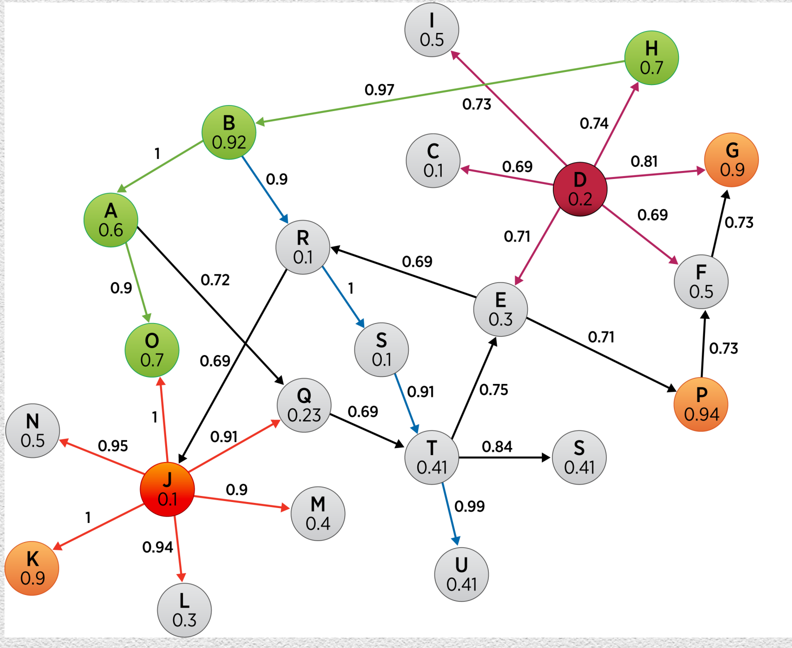
Can also explore songs data as a graph

- The `similars` element (key-value pairs of similar_song => weight) ~ edges on a graph.
  - Neo4j is the leading NoSQL graph database.

Neo4j-sized graph



PostgreSQL-sized graph



FREEDOM consulting group

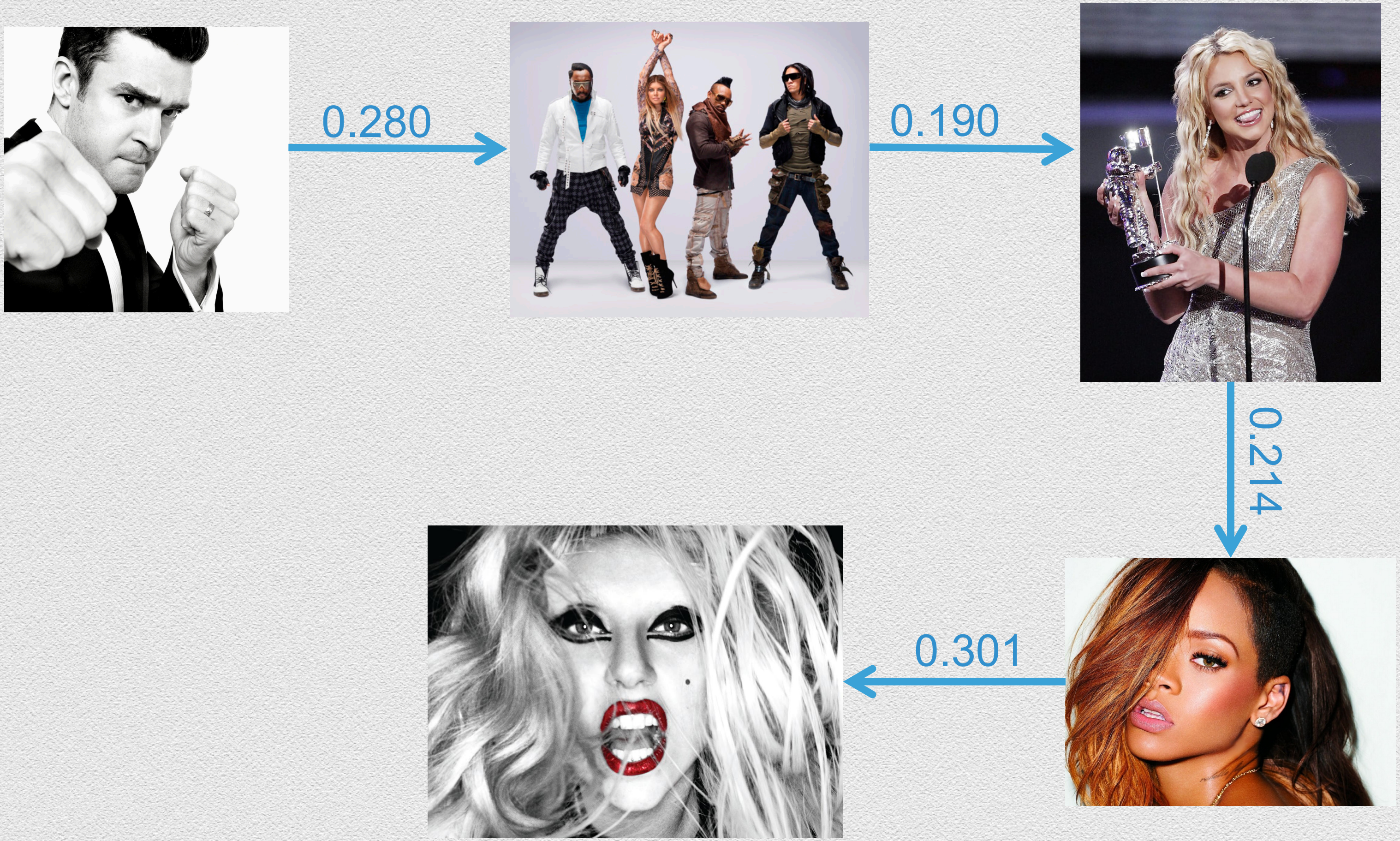
Use recursive query to find path

- Example adapted from PostgreSQL documentation.
- Transform songs data format into:
  - track_id
  - similar_track_id (a.k.a. link)
  - weight
- Filter our data set to 'rock' songs with relatively strong links … because recursive queries are expensive.
- Answer the burning question:
  Is there a path of related songs from Lady Gaga *Poker Face* to Justin Timberlake *What Goes Around … Comes Around*?

Switch to SQL interactive

Explore recursive query and graphing

Graph song results



Summary (1)

- Assertion: NoSQL is here to stay.
  our task is to *thrive* within Polyglot Persistence world.
- 5-ish types of NoSQL databases.
  PostgreSQL plays nice with 4 of them:
  - Document store (MongoDB)
  - ~~Wide column store (Cassandra)~~
  - Key-value store (Redis)
  - Graph DBMS (Neo4j)
  - Search engine (Solr)*

*See "Full Test Search with Ranked Results" from PGConf 2015.

Summary (2)

- PostgreSQL can load, interact with and present NoSQL data in a relational structure.
- PostgreSQL's sweet spot:
  - Data volume that fits well on one or a few servers.
  - Transaction boundaries from element to document level.
  - Want to enforce (some) referential integrity.
  - Want to find relations within data.

NOTE: This is what most organizations call "my real data".

- Leave edge cases to NoSQL tools.

Summary (3)

- Database architecture rules are more subtle and complex now.
- It is too simplistic to think *If it's not in at least 3NF – it's wrong.*
  - Know your business needs.
  - Know your data.
  - Know the strengths and limitations of the relational model plus NoSQL.

Seek to thrive in a world of Polyglot Persistence.

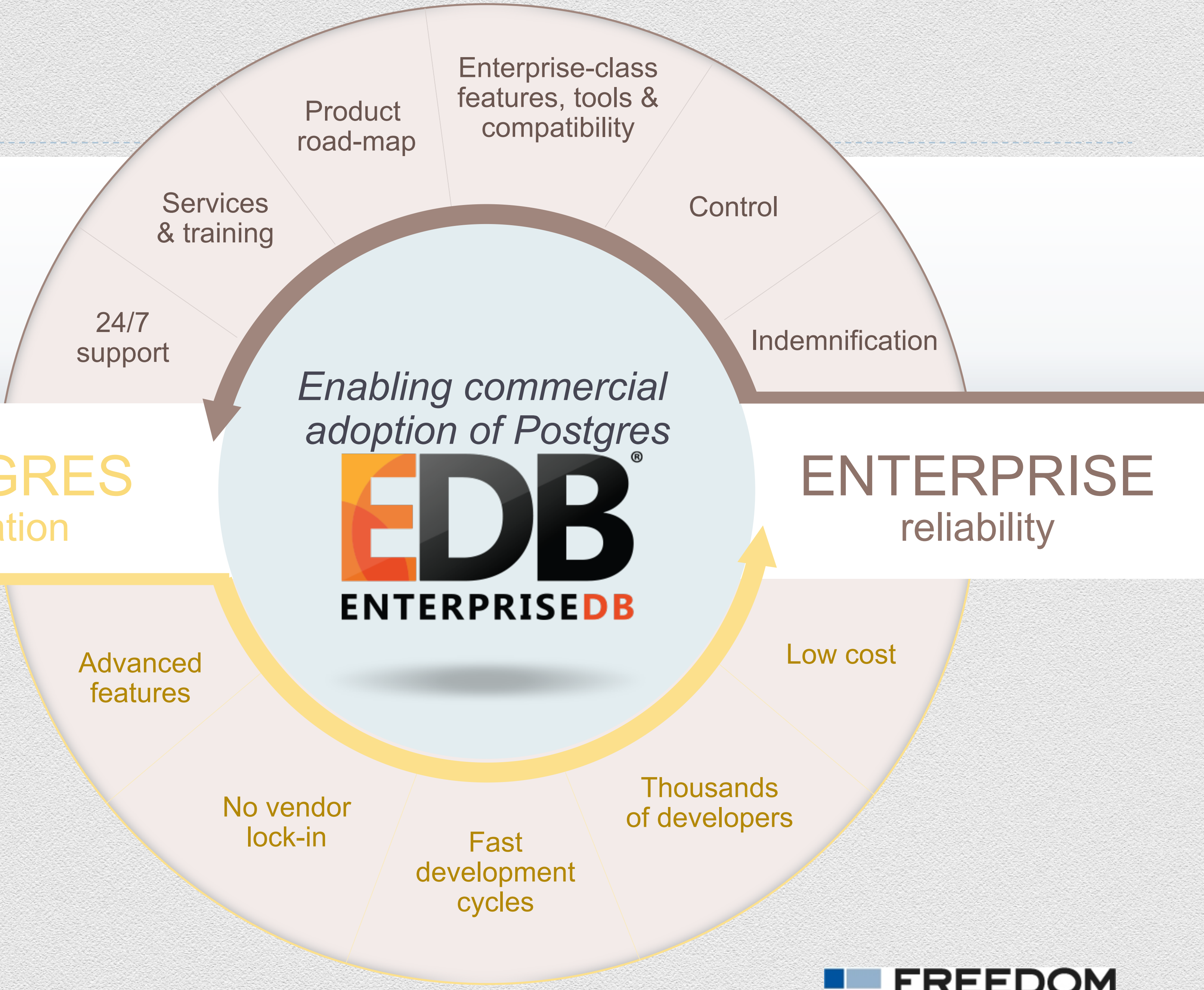
Are there any Questions or follow up?

LIFE. LIBERTY. TECHNOLOGY.

“

Freedom Consulting Group is a talented, hard-working, and committed partner, providing hardware, software and database development and integration services to a diverse set of clients.

”



FREEDOM consulting group