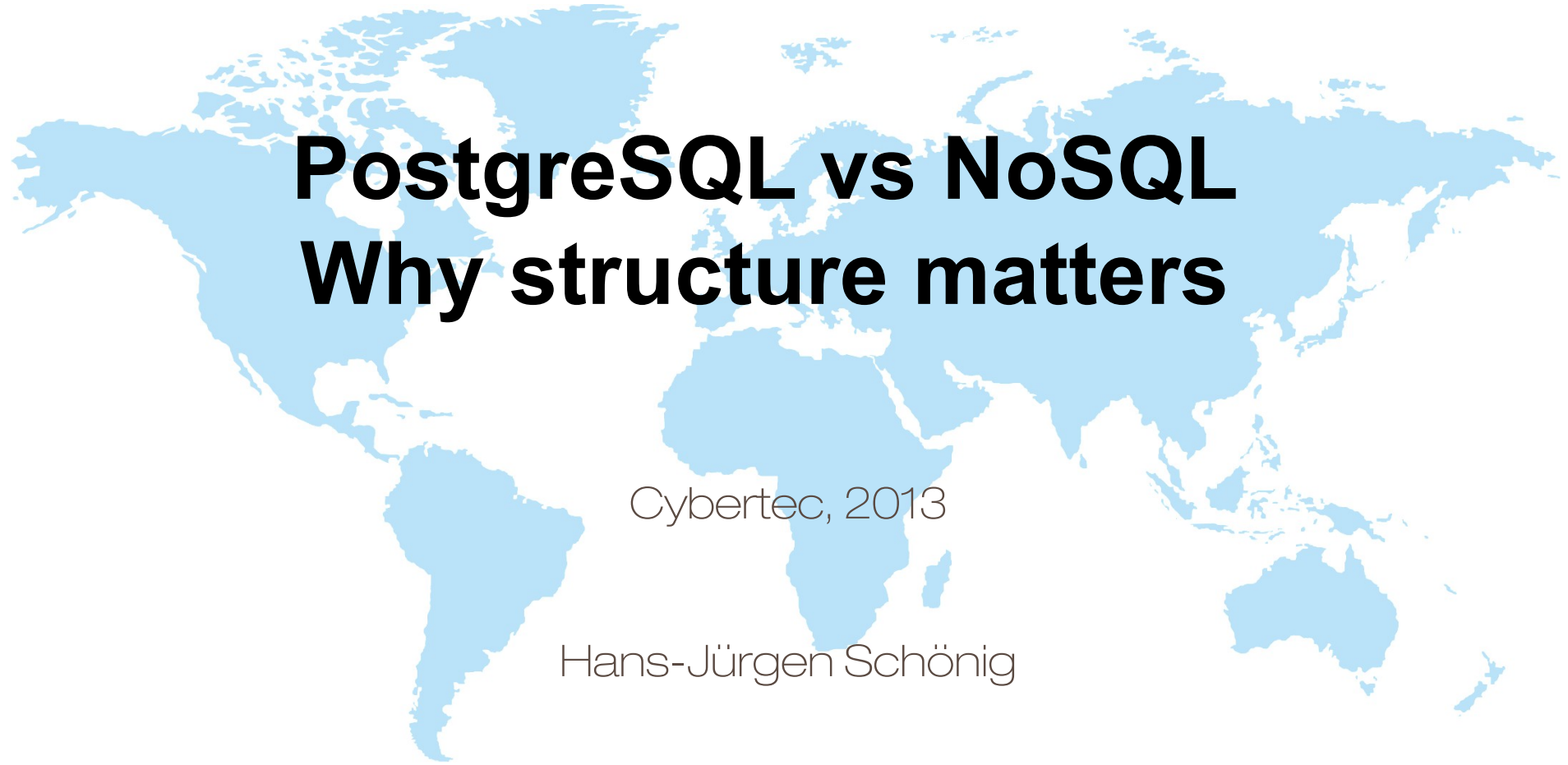


PostgreSQL vs NoSQL
Why structure matters

Cybertec, 2013

Hans-Jürgen Schönig

Recent developments:

- **“Document” stores have become more popular**
  - store “schemaless” data
  - key value stores

- **SQL is considered to be ...**
  - “old school”
  - non-scalable
  - less flexible

Clash of cultures:

CYBERTEC
The PostgreSQL Database Company

- **"Store first": The NoSQL way**

  - able to pump anything into the DB
  - NoSQL offers flexible schemas / no schema

- **"Structure first": The SQL way**

  - SQL is said to enforce a data structure
  - Data is expected to be correct
  - Storing "anything" is said to be messy

Can you really live without structure?

CYBERTEC
The PostgreSQL Database Company

- **There is no such thing as “no structure”**
  - Some structure logic must either be

    => in the database
    => on the application side

  - Write a script to count “anything” and group it

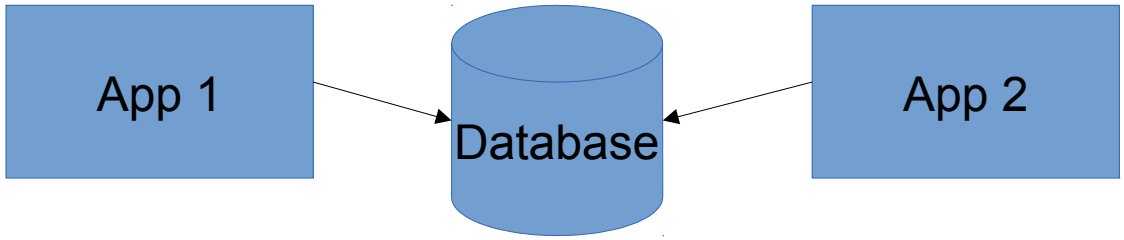
    => this is impossible
    => you have to know what to count

NoSQL: Secretly enforced schemas

- **Where is the schema then?**
  - Schema is secretly enforced by the application
  - Everybody has to play by the rules
  - “Wrong” data might make your code fail

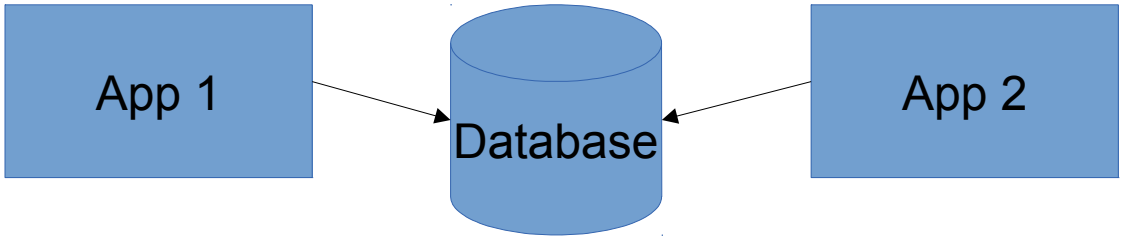
SQL enforces proper data ...

- Data structure is CENTRALLY enforced



- What if App2 uses slightly different keys?
- App1 and App2 have to agree on some common ground
- NoSQL does NOT enforce common ground
- Less room for errors in SQL

Potential NoSQL issues (1):



- App 1:

{"id":1,"long":35.6,"lat":27.4,"location":"some office"}

- App 2:

{"id":1,"lng":35.6,"lat":27.4,"location":"some office"}

=> if structure is not enforced, things go south
=> try to analyze longitude in this example
DISASTER !

Law and order in NoSQL

- **Developers must agree on field types**

  - agreeing on “field types” is not schemaless

  - enforced structure is less error prone than “agreements”

- **“well, developers have to stick to the rules”**

  => did you ever get caught speeding?
  => do you always separate your waste before you throw it away?

Don't expect ...

- developers to turn into angels because of NoSQL

- this will not happen for sure

- enforced structure is less error prone than "agreements"

- enforced rules are GOOD rules

=> SQL reduces the odds of failure and legacy
=> you can rely on sane data types

PostgreSQL: The NoSQL way (1)

CYBERTEC
The PostgreSQL Database Company

- **PostgreSQL can store “unstructured data”**

  - you don't have to normalize everything to death

    => normalization is not a must
    => it helps to organize data

    => be pragmatic !

  - you can use **json** and **hstore** to handle documents

PostgreSQL: The NoSQL way (2)

CYBERTEC
The PostgreSQL Database Company

- **Relational and document-oriented can coexist**

  - store documents the document way

  - store relational data the relational way

  => take the BEST of both worlds
  => PostgreSQL has them both

PostgreSQL: It is that simple ...

- Store the data the way YOU want

Relational:

```
test=# SELECT * FROM t_oil LIMIT 3;
 country | year | production
---------+------+------------
 USA     | 1965 |       9014
 USA     | 1966 |       9579
 USA     | 1967 |      10219
(3 rows)
```

As document:

```
test=# SELECT row_to_json(t_oil) FROM t_oil LIMIT 3;
                    row_to_json
-----------------------------------------------------
 {"country":"USA","year":1965,"production":9014}
 {"country":"USA","year":1966,"production":9579}
 {"country":"USA","year":1967,"production":10219}
(3 rows)
```

PostgreSQL: Strong support for JSON (1)

CYBERTEC
The PostgreSQL Database Company

- PostgreSQL support for JSON

- PostgreSQL has a native data type called JSON

- it is an ordinary field type

- you can index parts of the JSON document
  just like any other data

- JSON can be passed on to web developers
  directly.

PostgreSQL: Strong support for JSON (2)

- **As easy as it can be ...**

```
CREATE TABLE t_document (
    id          serial,          -- “relational”
    name        text,
    author      text,
    doc         json             -- “document oriented”
);
```

PostgreSQL: Flexibility ...

- **PostgreSQL is as flexible as NoSQL**
  - Add columns on the fly:

```
ALTER TABLE t_oil ADD COLUMN continent text;
```

  - Drop columns on the fly:

```
ALTER TABLE t_oil DROP COLUMN continent;
```

PostgreSQL: Scalability

- **PostgreSQL is fast and scalable**
  - Scalability can be achieved through:
    - => Master / slave replication
    - => Sharding (using PL/Proxy)
    - => Postgres-XC
      (a cluster solution for write as well as read scalability)
  - PostgreSQL is highly optimized

PostgreSQL: Development time

CYBERTEC
The PostgreSQL Database Company

- SQL code can be written easily and FAST

- Example: A moving average ...

```
SELECT *,
       avg(production)
            OVER ( PARTITION BY country
                        ORDER BY year
                        ROWS BETWEEN 2 PRECEDING
                              AND 2 FOLLOWING)
FROM t_oil;
```

- It takes some 30 seconds to write this

- PostgreSQL takes care of the best algorithm

PostgreSQL: Compatibility

- **SQL is an industry standard**
  - Many vendors support SQL
  - Examples are widely available
  - PostgreSQL takes care of the best algorithm
- **NoSQL market is fragmented**
  - No standards
  - Still changes in APIs and libraries

=> why take the risk?

Any questions?

Hans Jürgen Schönig
hs@cybertec.at, www.postgresql-support.de