
SHRIEK!

Why Zalando trusts in PostgreSQL

A developer's view on using the most advanced open-source database

Henning Jacobs - Technical Lead Platform/Software
Zalando GmbH
Valentine Gogichashvili - Technical Lead Platform/Database
Zalando GmbH

GOTO Berlin 2013, 2013-10-18

Who we are

Henning Jacobs <henning.jacobs@zalando.de>

with Zalando since 2010

NO DBA, NO PostgreSQL Expert

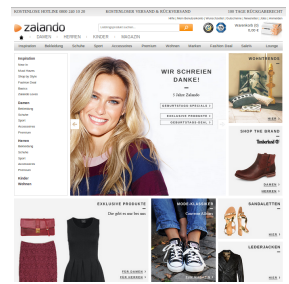
Valentine Gogichashvili <valentine.gogichashvili@zalando.de>

with Zalando since 2010

DBA, PostgreSQL Expert

About Zalando

- 14 countries
- > 1 billion € revenue 2012
- > 150,000 products
- 3 warehouses
- Europe's largest online fashion retailer

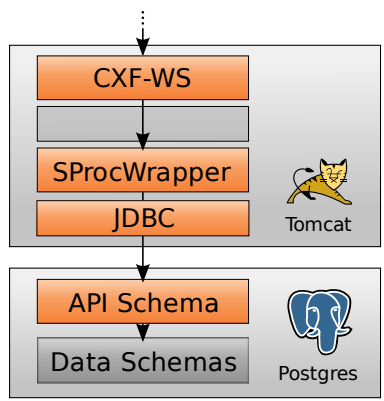


Our ZEOS Platform

Our Main Stack

Java → PostgreSQL

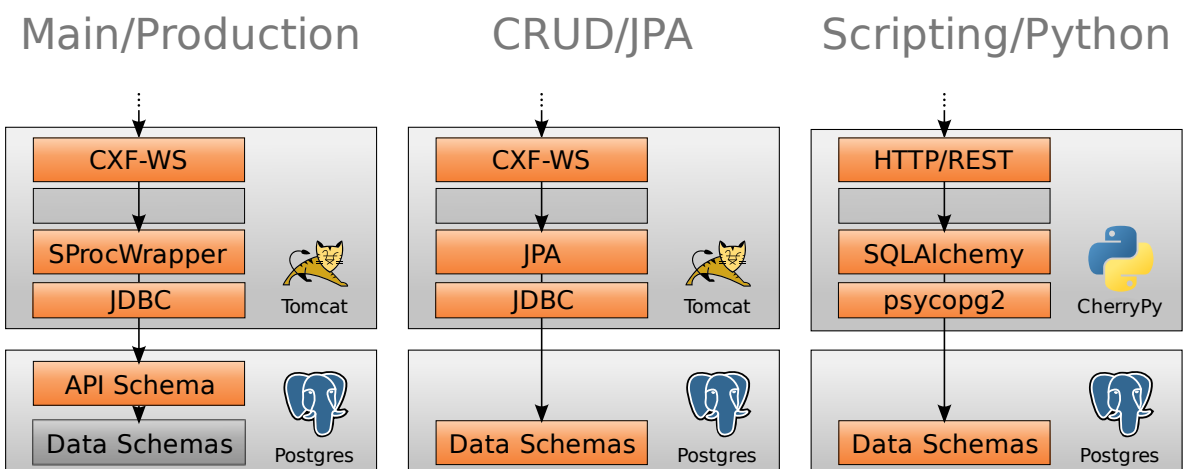
Main/Production



Java 7, Tomcat 7, PostgreSQL 9.0–9.3

Our Main Stacks

Different Use Cases — same Database

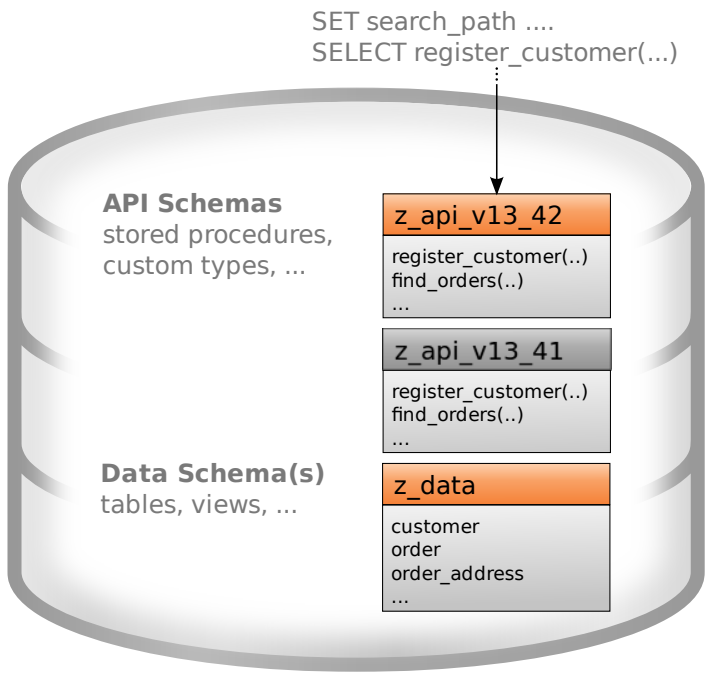


Java 7, Tomcat 7, Python 2.7, PostgreSQL 9.0–9.3

Some Numbers

- > 90 deployment units (WARs)
- > 800 production Tomcat instances
- > 50 different databases
- > 90 database master instances
- > 5 TB of PostgreSQL data
- > 200 developers, 8 DBAs

Stored Procedures



Stored Procedures

1:1 Mapping using SProcWrapper

JAVA

```java
@SProcService
public interface CustomerSProcService {
    @SProcCall
    int registerCustomer(@SProcParam String email, @SProcParam Gender gender);
}
```

SQL

```sql
CREATE FUNCTION register_customer(p_email text, p_gender z_data.gender)
RETURNS int AS
$$
    INSERT INTO z_data.customer (c_email, c_gender)
         VALUES (p_email, p_gender)
      RETURNING c_id
$$
LANGUAGE 'sql' SECURITY DEFINER;
```

Stored Procedures

Complex Types

JAVA

```java
@SProcCall
List<Order> findOrders(@SProcParam String email);
```

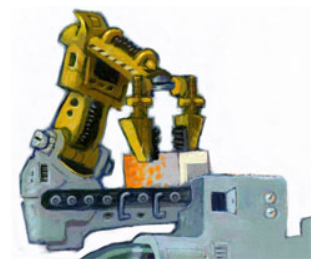
SQL

```sql
CREATE FUNCTION find_orders(p_email text,
    OUT order_id int,
    OUT order_date timestamp,
    OUT shipping_address order_address) RETURNS SETOF RECORD AS
$$
      SELECT o_id, o_date, ROW(oa_street, oa_city, oa_country)::order_address
        FROM z_data.order
        JOIN z_data.order_address ON oa_order_id = o_id
        JOIN z_data.customer ON c_id = o_customer_id
       WHERE c_email = p_email
$$
LANGUAGE 'sql' SECURITY DEFINER;
```

Stored Procedures

Experience

- Performance benefits
- Easy to change live behavior
- Validation close to data
- Simple transaction scope
- Makes moving to new software version easy
- Cross language API layer
- CRUD ⇒ JPA



Stored Procedures

Rolling out database changes

- API versioning
  - Automatic roll-out during deployment
  - Java backend selects "right" API version
- Modularization
  - shared SQL gets own Maven artifacts
  - feature/library bundles of Java+SQL
- DB-Diffs
  - SQL scripts for database changes
  - Review process
  - Tooling support

Stored Procedures & Constraints

Protect Your Most Valuable Asset: Your Data

Constraints

Ensuring Data Quality

Simple SQL expressions:

```SQL
CREATE TABLE z_data.host (
    h_hostname     varchar(63) NOT NULL UNIQUE CHECK (h_hostname ~ '^[a-z][a-z0-9-]*$'),
    h_ip           inet        UNIQUE CHECK (masklen(h_ip) = 32),
    h_memory_bytes bigint      CHECK (h_memory_bytes > 8*1024*1024)
);
COMMENT ON COLUMN z_data.host.h_memory_bytes IS 'Main memory (RAM) of host in Bytes';
```

Combining check constraints and stored procedures:

```SQL
CREATE TABLE z_data.customer (
    c_email        text        NOT NULL UNIQUE CHECK (is_valid_email(c_email)),
    ..
);
```

Constraints

What about MySQL?

The MySQL manual says:

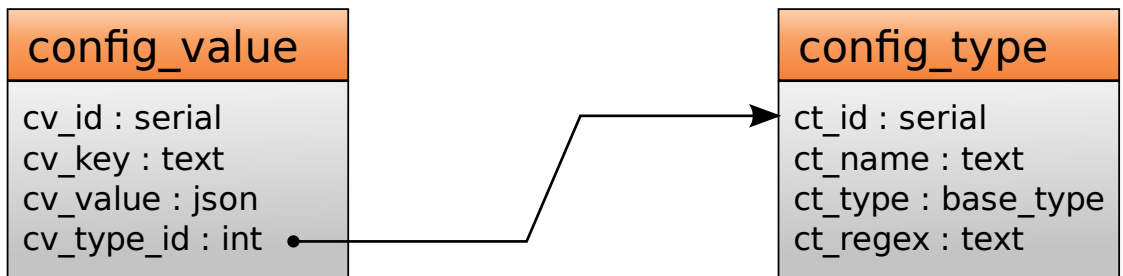
> *The CHECK clause is parsed but ignored by all storage engines.*

Open Bug ticket since 2004: http://bugs.mysql.com/bug.php?id=3464

http://dev.mysql.com/doc/refman/5.7/en/create-table.html

Constraints

Custom Type Validation using Triggers



key:
recipientEmail

value:
"john.doe@example.org"

type:
EMAIL_ADDRESS

Custom Type Validation using Triggers

SQL

```
CREATE FUNCTION validate_value_trigger() RETURNS trigger AS
$$
    BEGIN
        PERFORM validate_value(NEW.cv_value, NEW.cv_type_id);
    END;
$$
LANGUAGE 'plpgsql';
```

SQL

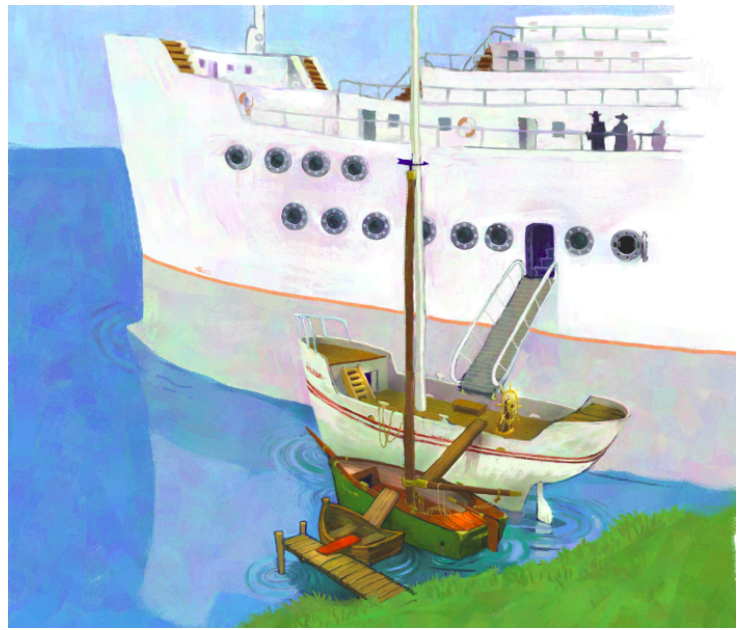
```
CREATE FUNCTION validate_value(p_value json, p_type_id int) RETURNS void AS
$$
    import json
    import re
    # ... Python code, see next slide
$$
LANGUAGE 'plpythonu';
```

Custom Type Validation with PL/Python

PYTHON

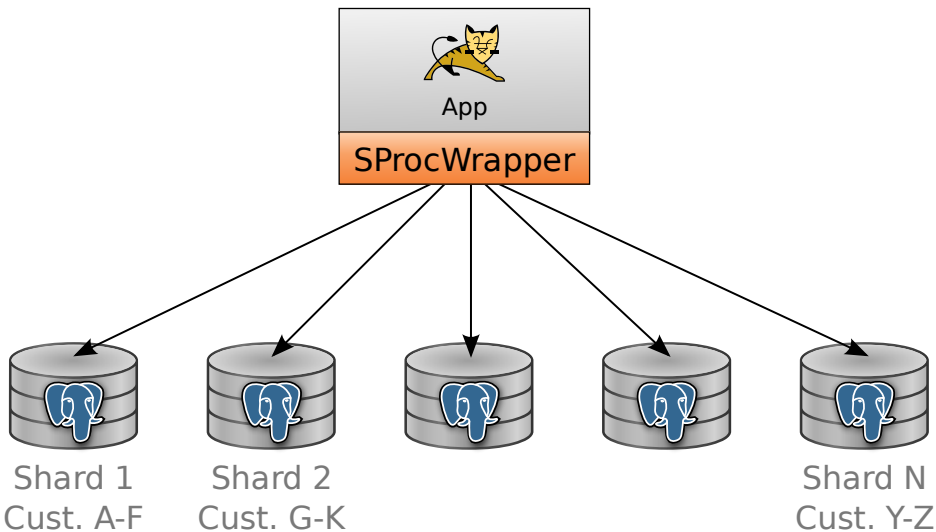
```
class TypeValidator(object):
    @staticmethod
    def load_by_id(id):
        tdef = plpy.execute('SELECT * FROM config_type WHERE ct_id = %d' % int(id))[0]
        return _get_validator(tdef)

    def check(self, condition, msg):
        if not condition:
            raise ValueError('Value does not match the type "%s". Details: %s' %
                             (self.type_name, msg))

class BooleanValidator(TypeValidator):
    def validate(self, value):
        self.check(type(value) is bool, 'Boolean expected')

validator = TypeValidator.load_by_id(p_type_id)
validator.validate(json.loads(p_value))
```

Scaling?



Sharding

Scaling Horizontally



Sharding

SProcWrapper Support

Sharding customers by ID:

```java
@SProcCall
int registerCustomer(@SProcParam @ShardKey int customerId,
                     @SProcParam String email, @SProcParam Gender gender);
```

Sharding articles by SKU (uses MD5 hash):

```java
@SProcCall
Article getArticle(@SProcParam @ShardKey Sku sku);
```

Collecting information from all shards concurrently:

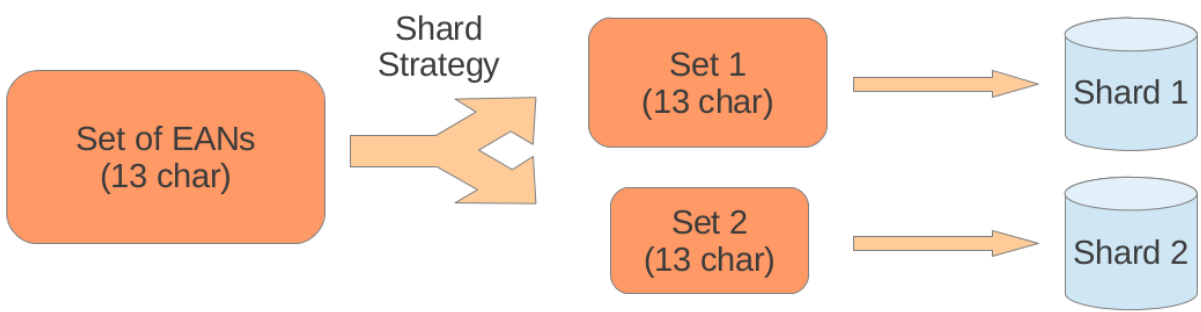
```java
@SProcCall(runOnAllShards = true, parallel = true)
List<Order> findOrders(@SProcParam String email);
```

Sharding

Auto Partitioning Collections

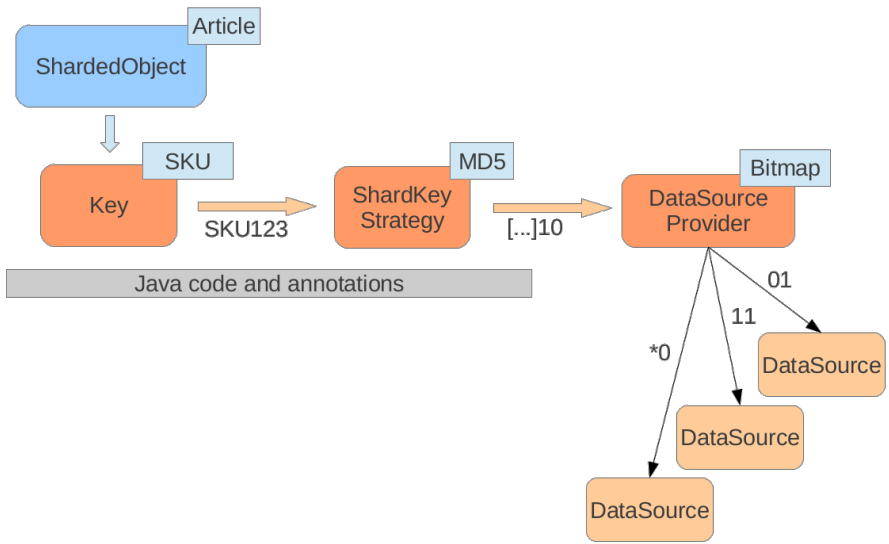
JAVA

```
@SProcCall(parallel = true)
void updateStockItems(@SProcParam @ShardKey List<StockItem> items);
```



Sharding

Bitmap Data Source Providers



Sharding

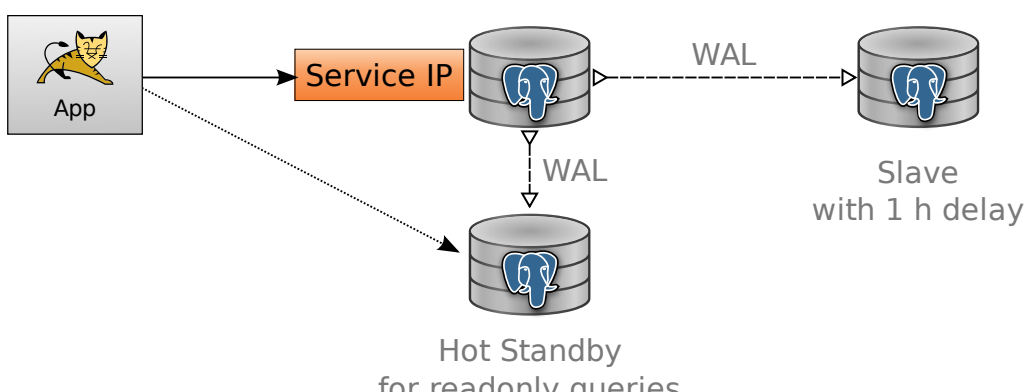
Experience

- **Scalability** without sacrificing any SQL features
- Start with a **reasonable number of shards** (8)
- Some **tooling required**
- **Avoid caching if you can** and scale horizontally

Fail Safety

Replication

- All databases use **streaming replication**
- Every database has a (hot) standby and a delayed slave



Fail Safety

Failover

- **Service IP** for switching/failover
- **Monitoring** with Java and custom web frontend
- Failover is manual task

Fail Safety

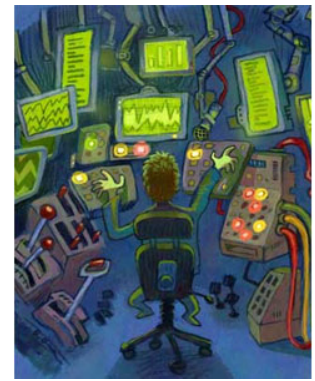
General Remarks

- **Good hardware**
  - G8 servers from HP
  - ECC RAM, RAID
- **No downtimes allowed**
  - Major PostgreSQL version upgrades?
- Two data centers
- Dedicated 24x7 team
- Maintenance
  - Concurrent index rebuild, table compactor

Monitoring

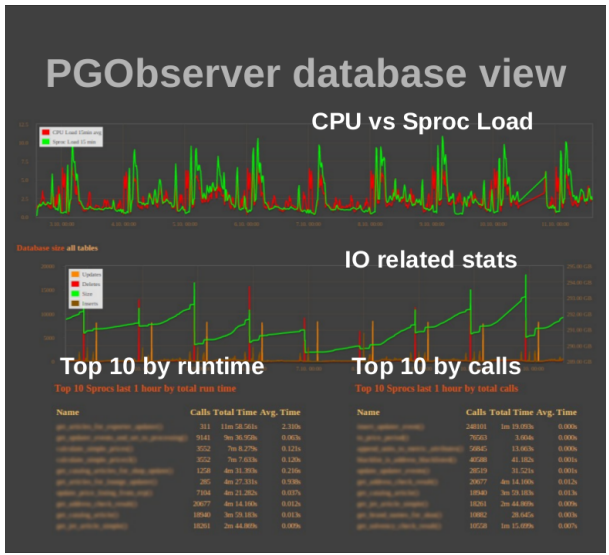
You need it...

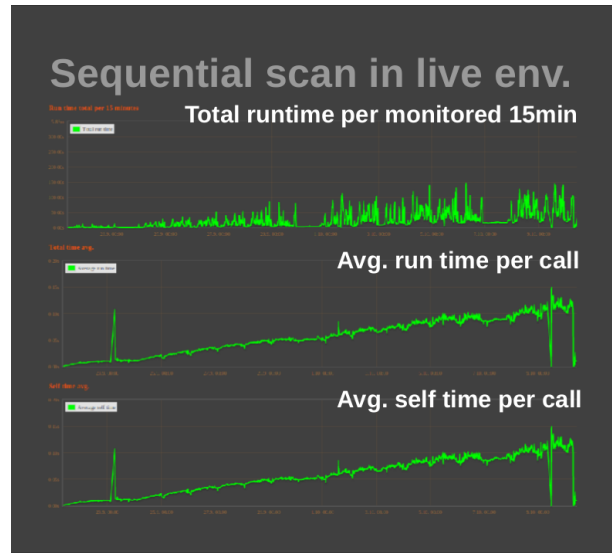
- Nagios/Icinga
- PGObserver
- pg_view



Monitoring

PGObserver

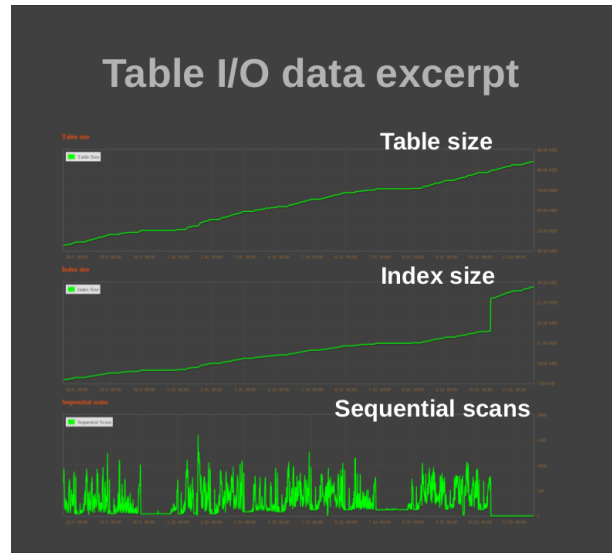




Monitoring

PGObserver

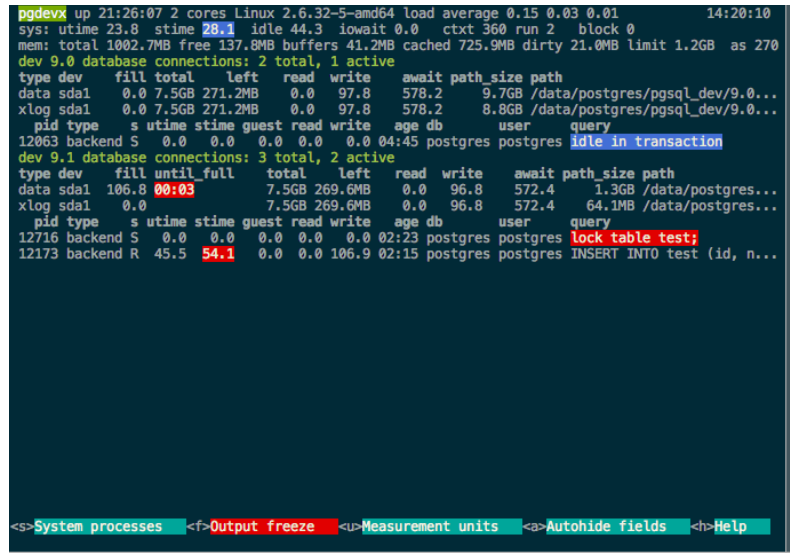
- Locate hot spots
  - Frequent stored procedure calls
  - Long running stored procedures
  - I/O patterns
- Helps tuning DB performance
- Creating the right indices often a silver bullet



Monitoring

pg_view

- Top-like command line utility
- DBA's daily tool
- Analyze live problems
- Monitor data migrations



NoSQL

Relational is dead?

> *If your project is not particularly vital and/or your team is not particularly good, use relational databases!*

Martin Fowler, GOTO Aarhus 2012

NoSQL

Relational is dead?

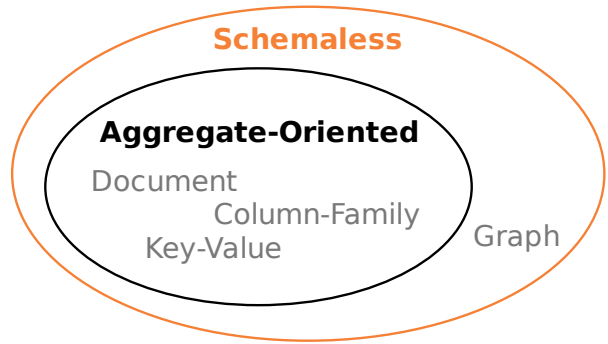
The key goals of F1's design are:

> *3. Consistency: The system must provide ACID transactions, [..]*
> *4. Usability: The system must provide full SQL query support [..]*
> *Features like indexes and ad hoc query are not just nice to have, but absolute requirements for our business.*

F1: A Distributed SQL Database That Scales

NoSQL – Comparison

- Aggregate oriented?
  - SProcWrapper
  - Changes?
- Schemaless?
  - ⇒ Implicit Schema
  - HStore, JSON
- Scaling?
- Auth*?
- Use cases? ⇒ "Polyglot Persistence"



YeSQL

PostgreSQL at Zalando

- Fast, stable and well documented code base
- Performs as well as (and outperforms some) NoSQL databases while retaining deeper flexibility
- Scaling horizontally can be easy
- Know your tools — whatever they are!

Thank You!

Please Visit also

- **SProcWrapper** – Java library for stored procedure access
  github.com/zalando/java-sproc-wrapper
- **PGObserver** – monitoring web tool for PostgreSQL
  github.com/zalando/PGObserver
- **pg_view** – top-like command line activity monitor
  github.com/zalando/pg_view

tech.zalando.com

Please rate this session!