
SHRIEK!

Elephants in fashion

How Zalando uses PostgreSQL to drive the biggest online fashion store in Europe

About me

Valentine Gogichashvili

Database Engineer @Zalando

twitter: @valgog

google+: +valgog

email: valentine.gogichashvili@zalando.de

zalando


ZALANDO
SHOP
NEU
%

zalando

Europe's largest online fashion retailer

14 countries

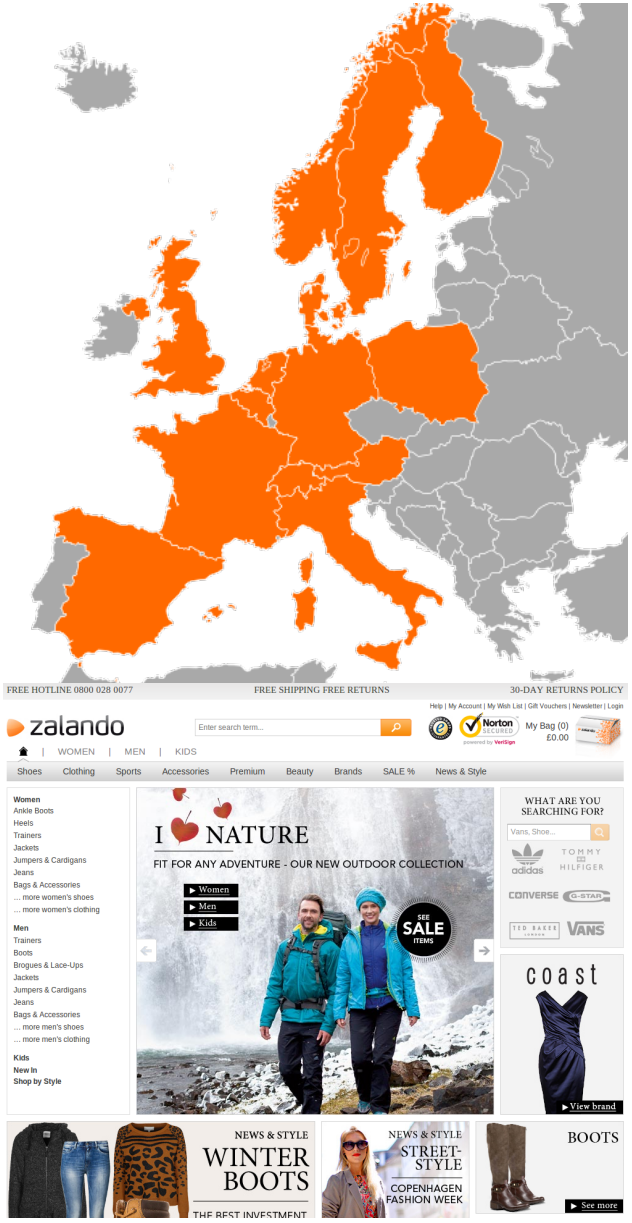
3 warehouses

15 million customers

1+ billion € revenue 2012

150,000+ products

100+ million visits per month



ZALANDO
SHOP


SOLR
Zalando
SHOP
MOBILE
LOUNGE
3rd PARTY
TOOLING
SERVICE 1
SERVICE 2
SERVICE 3
SERVICE 5
SERVICE 6
POSTGRE SQL
BI
LOGISTICS

Some more numbers

90+ deployment units (WARs)

800+ production Tomcat instances

90+ database master instances

50+ different databases

200+ developers

8 database engineers

zalando

Even more numbers

- > 3.0 TB of PostgreSQL data
- Biggest instances (not counted before)
  - `eventlogdb (3TB)`
    - 20 GB per week

Biggest challenges

- Constantly growing
- Fast development cycles
- No downtimes are tolerated

Agenda

How we

- access data
- change data models without downtimes
- shard without limits

- install
- configure
- monitor

Agenda

How we

- **access data**
- change data models without downtimes
- shard without limits

Accessing data

- customer
  - bank account
  - order -> bank account
    - order position
  - return order -> order
    - return position -> order position
  - financial document
    - financial transaction -> order

Accessing data

NoSQL

- map your object hierarchy to a document
- (de-)serialization is easy
- transactions are not needed

- No SQL
- implicit schemas are tricky

Accessing data

ORM

- is well known to developers
- CRUD operations are easy
- all business logic inside your application
- developers are in their comfort zone

- error prone transaction management
- you have to reflect your tables in your code
- all business logic inside your application
- schema changes are not easy

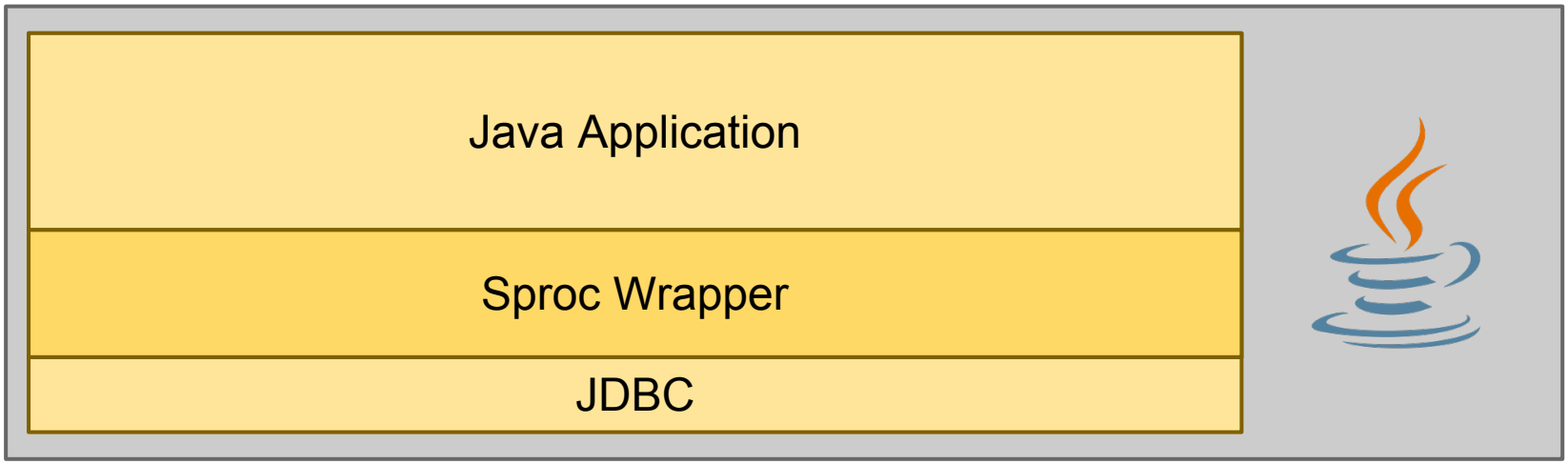
Accessing data

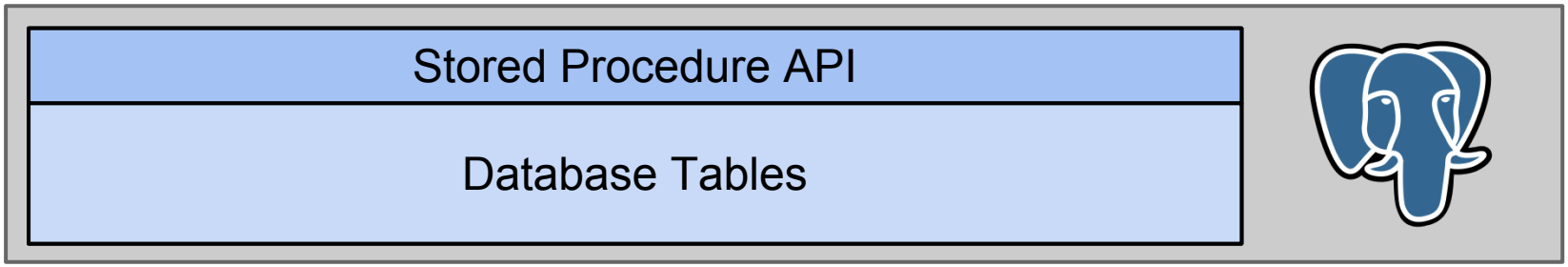
Are there alternatives to ORM?

Stored Procedures

- return/receive entity aggregates
- clear transaction scope
- more data consistency checks
- independent from underlying data schema

Java Sproc Wrapper





Java Sproc Wrapper

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;
```

Java Sproc Wrapper

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;
```

zalando

Java Sproc Wrapper

JAVA

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

SQL

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

Java Sproc Wrapper

JAVA

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

SQL

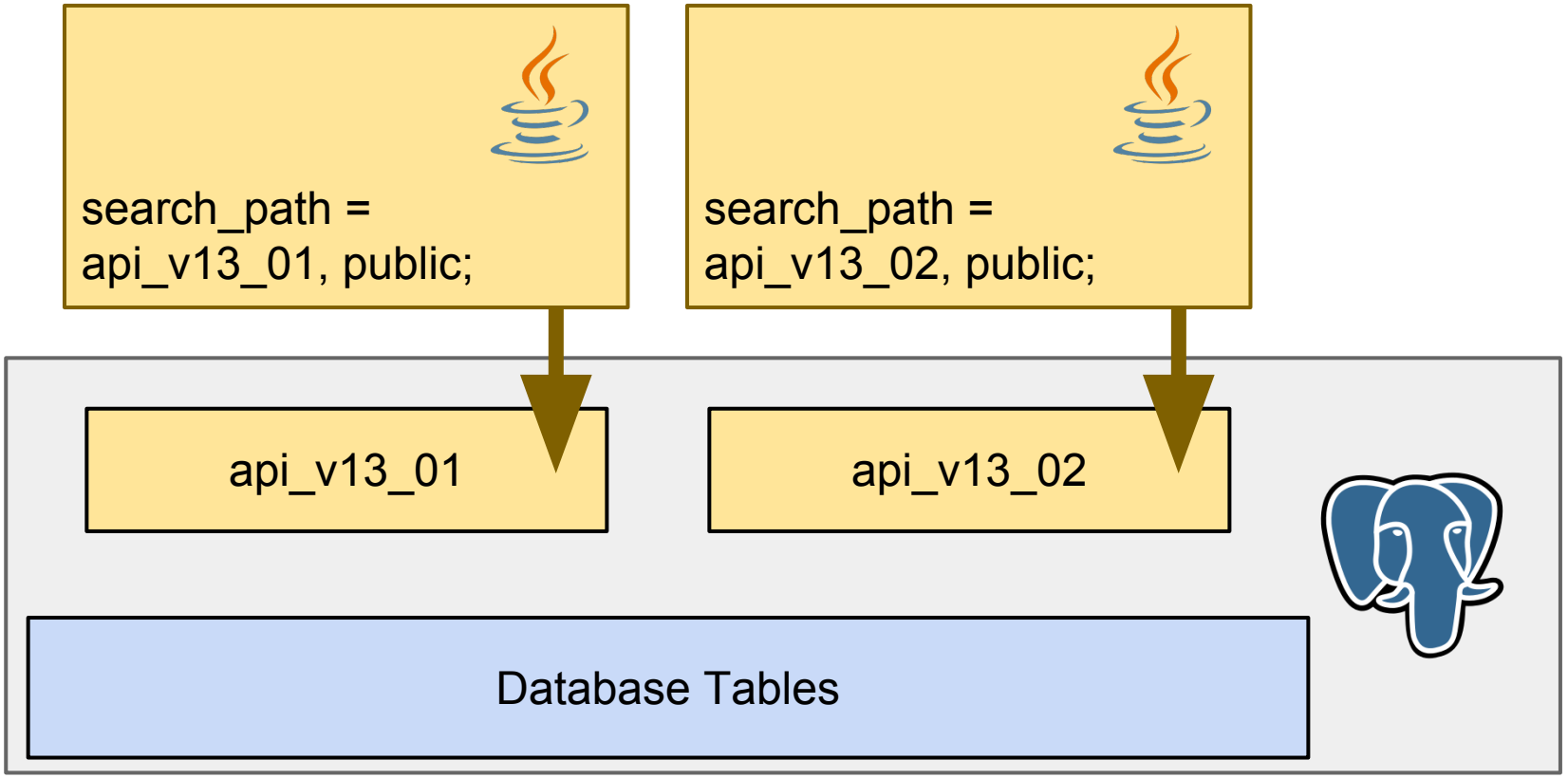
```sql
CREATE FUNCTION find_orders(p_email text,
                            OUT order_id int,
                            OUT order_created timestamptz,
                            OUT shipping_address order_address)
                RETURNS SETOF record
AS $$
      SELECT o_id, o_created,
         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

Stored Procedures
for developers

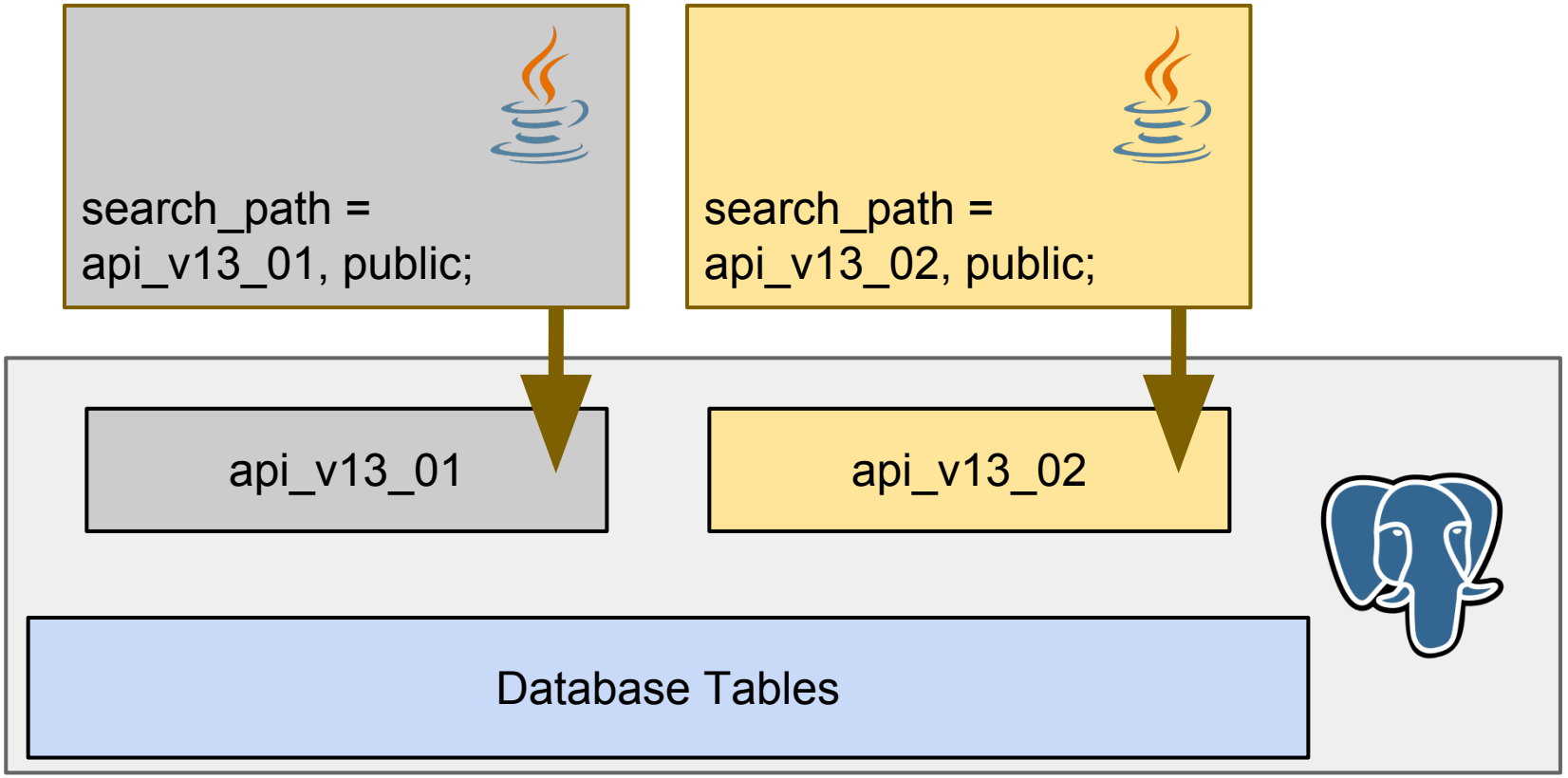
- CRUD operations need too much code
- Developers have to learn SQL
- Developers can write bad SQL
- Code reviews are needed

- Use-case driven
- Developers have to learn SQL
- Developers learn how to write good SQL

Stored Procedure
API versioning



Stored Procedure
API versioning



Stored Procedure
API versioning

- Tests are done to the whole API version
- No API migrations needed
- Deployments are fully automated

Agenda

How we

- access data
- **change data models without downtimes**
- shard without limits

Easy schema changes

- PostgreSQL
  - Schema changes with minimal locks with:
    `ADD/RENAME/DROP COLUMN`
    `ADD/DROP DEFAULT VALUE`
  - `CREATE/DROP INDEX CONCURRENTLY`
  - Constraints are still difficult to `ALTER`

Easy schema changes

- Stored Procedure API layer
  - Can fill missing data on the fly
  - Helps to change data structure without application noticing it

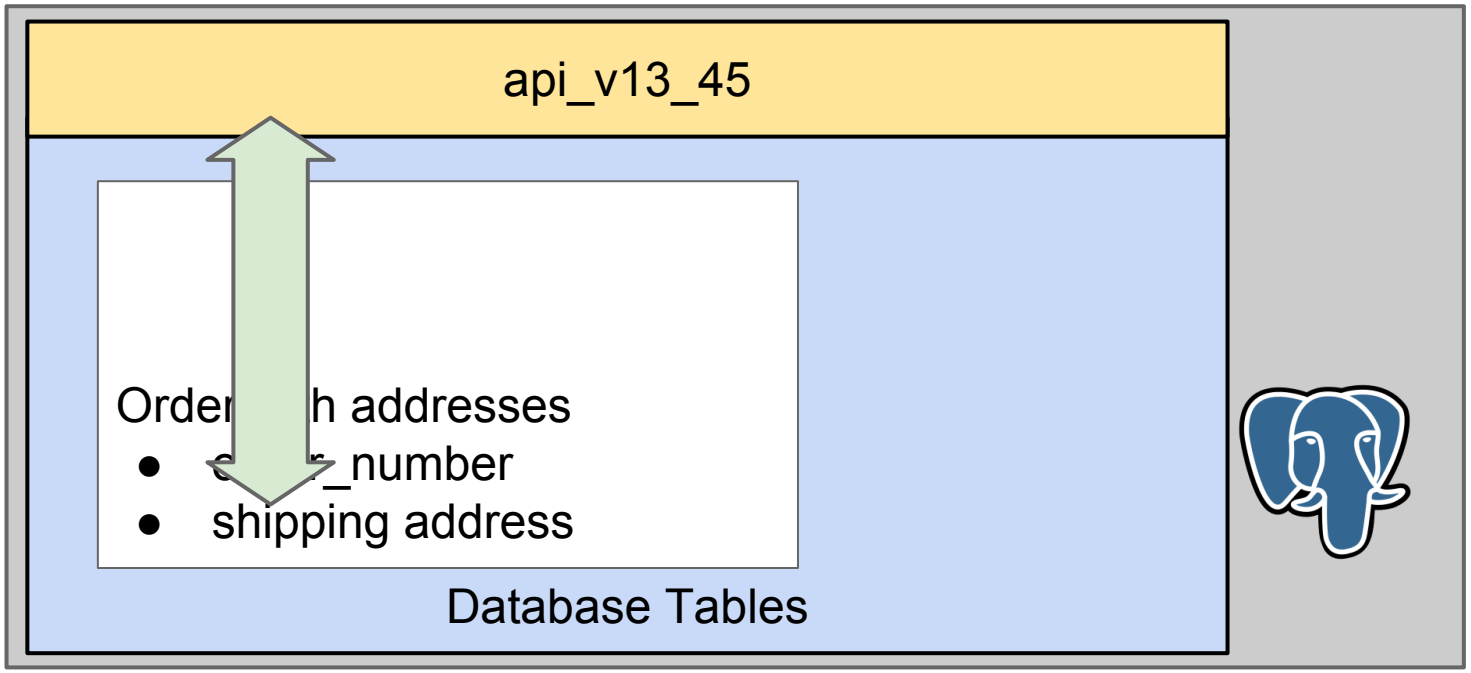
Easy schema changes

- Read and write to *old* structure
- Write to both structures, *old* and *new.*
  Try to read from *new*, fallback to *old*
- Migrate data
- Read from new, write to *old* and *new*

Easy schema changes

search_path =
api_v13_45, public;

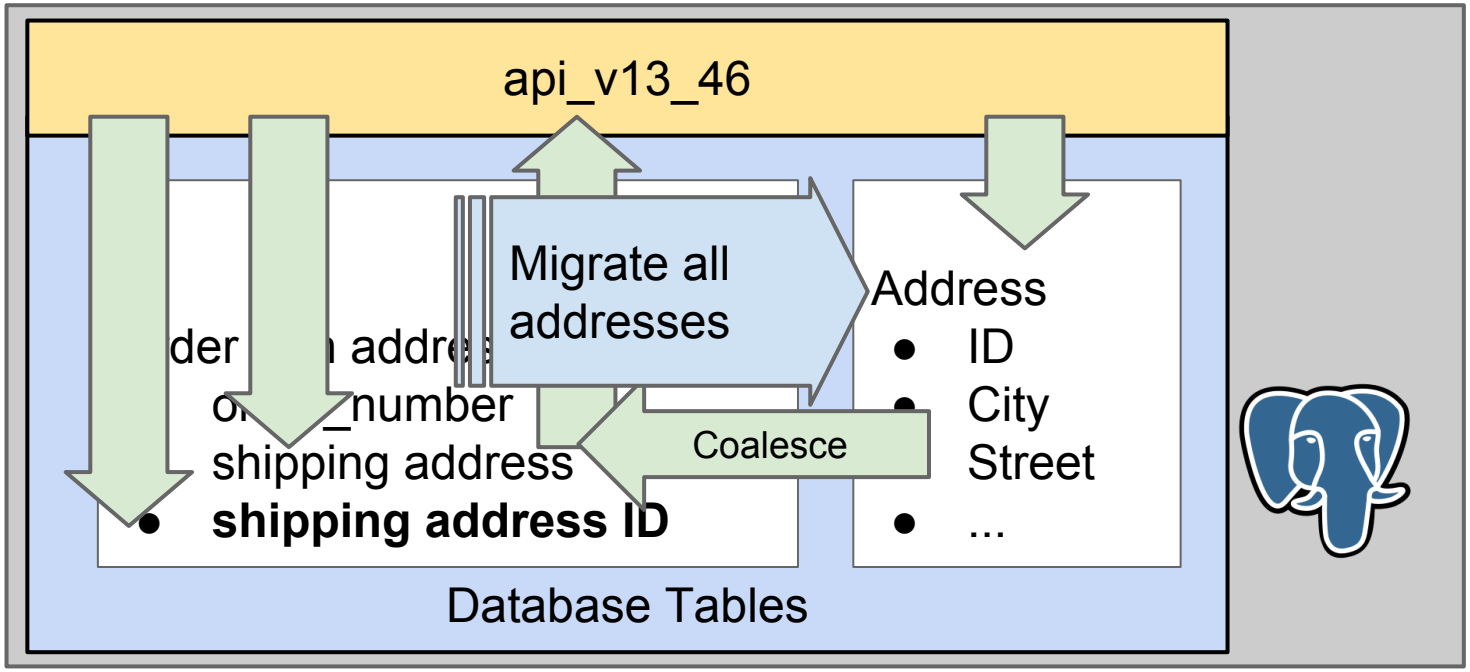
Read and write to old structure



Easy schema changes

search_path =
api_v13_46, public;

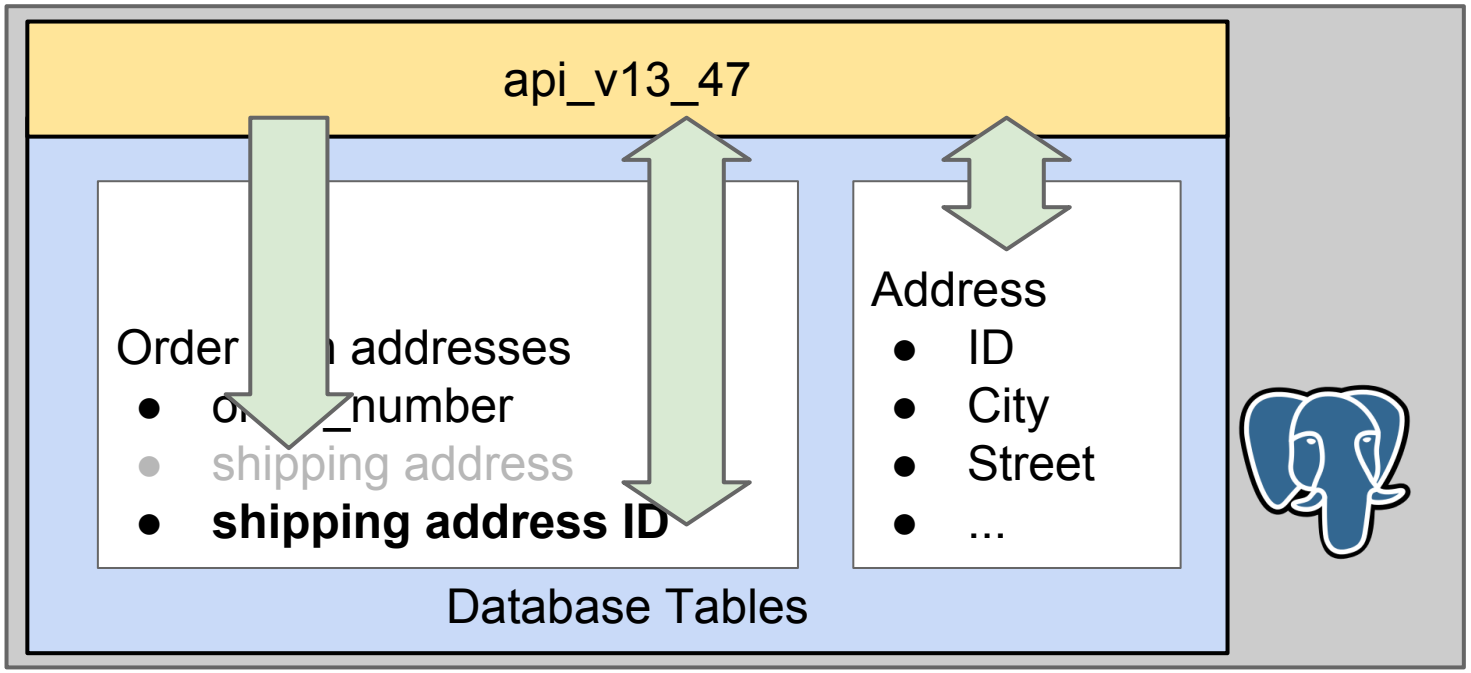
Write to both structures, old and new
Try to read from new, fallback to old



Easy schema changes

search_path =
api_v13_47, public;

Read from new
Write to both structures, old and new

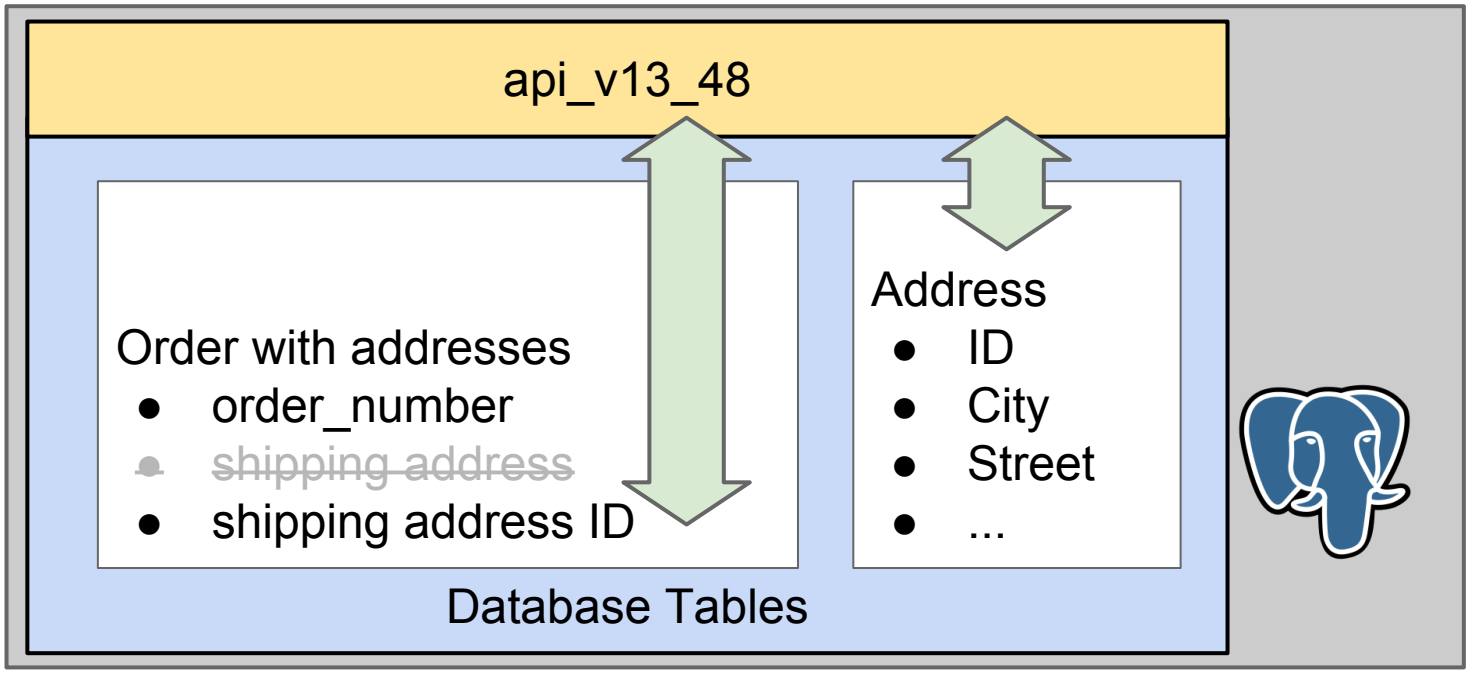


zalando

Easy schema changes

search_path =
api_v13_48, public;

Read and write to new
Drop old structures



zalando

Easy schema changes

- Schema changes using SQL script files
  - SQL scripts written by developers (DBDIFFs)
  - registering DBDIFFs with `Versioning`
  - should be reviewed by DB guys
  - DB guys are rolling DB changes on the live system

Easy schema changes

DBDIFF SQL

```
BEGIN;
    SELECT _v.register_patch('ZEOS-5430.order');

    CREATE TABLE z_data.order_address (
      oa_id int SERIAL,
      oa_country z_data.country,
      oa_city varchar(64),
      oa_street varchar(128), ...
    );

    ALTER TABLE z_data."order" ADD o_shipping_address_id int
      REFERENCES z_data.order_address (oa_id);
COMMIT;
```

zalando

Easy schema changes

DBDIFF SQL

```
BEGIN;
    SELECT _v.register_patch('ZEOS-5430.order');

    \i order/database/order/10_tables/10_order_address.sql

    ALTER TABLE z_data."order" ADD o_shipping_address_id int
      REFERENCES z_data.order_address (oa_id);
COMMIT;
```

Easy schema changes

Overview of R13_00_44

Warning! 11 patch names exists in multiple files!

Project	Database	Diff	Reviewed	Integration	Release	Patch	LIVE
		backend - 18/19					
de.zalando.admin/admin-backend							
	admin						
		ZEOS-24617.admin	A S	1 / 1	1 / 1	1 / 1	1 / 1
de.zalando/bm							
	bm						
		ORDER-453.bm	S A	0 / 1	1 / 1	1 / 1	1 / 1
de.zalando/config-service							
	config						
		ZEOS-21566.data	A A S	1 / 1	1 / 1	1 / 1	1 / 1
		ZEOS-24840.data	S A	1 / 1	1 / 1	1 / 1	1 / 1
		ZEOS-25486.data	A	0 / 1	1 / 1	0 / 1	1 / 1

zalando

Easy schema changes

purchasing - 6/10						
de.zalando/purchasing-backend						
	purchase					
		ZEOS-19134.1.purchase	A	0 / 1	0 / 1	0 / 1	0 / 1
		ZEOS-23911.purchase	A S	0 / 1	1 / 1	1 / 1	1 / 1
		ZEOS-24134.purchase	S A	1 / 1	1 / 1	1 / 1	1 / 1
		ZEOS-24484.purchase	A S	0 / 1	1 / 1	1 / 1	1 / 1
		ZEOS-24597.purchase	A S	1 / 1	1 / 1	1 / 1	1 / 1
		ZEOS-25078.purchase	S A	0 / 1	1 / 1	1 / 1	1 / 1
		ZEOS-25272.purchase		1 / 1	0 / 1	0 / 1	0 / 1
		ZEOS-25425.purchase		0 / 1	1 / 1	1 / 1	1 / 1
		ZEOS-25428.purchase.data		1 / 1	1 / 1	0 / 1	0 / 1
		ZEOS-25521.purchase.data		0 / 1	0 / 1	0 / 1	0 / 1
shared - 1/1						
de.zalando/zalando-db-commons						
	commons					
		ORDER-405.db-commons	S A	16 / 58	20 / 57	19 / 57	17 / 59

zalando

Easy schema changes

No downtime due to migrations or

deployment since we use PostgreSQL

Agenda

How we

- access data
- change data models without downtimes
- **shard without limits**

One big database

- Joins between any entities
- Perfect for BI
- Simple access strategy
- Less machines to manage

One big database

- Data does not fit into memory
- OLTP becomes slower
- Longer data migration times
- Database maintenance tasks take longer

zalando

zalando

Sharded database

- Data fits into memory
- IO bottleneck wider
- OLTP is fast again
- Data migrations are faster
- Database maintenance tasks are faster

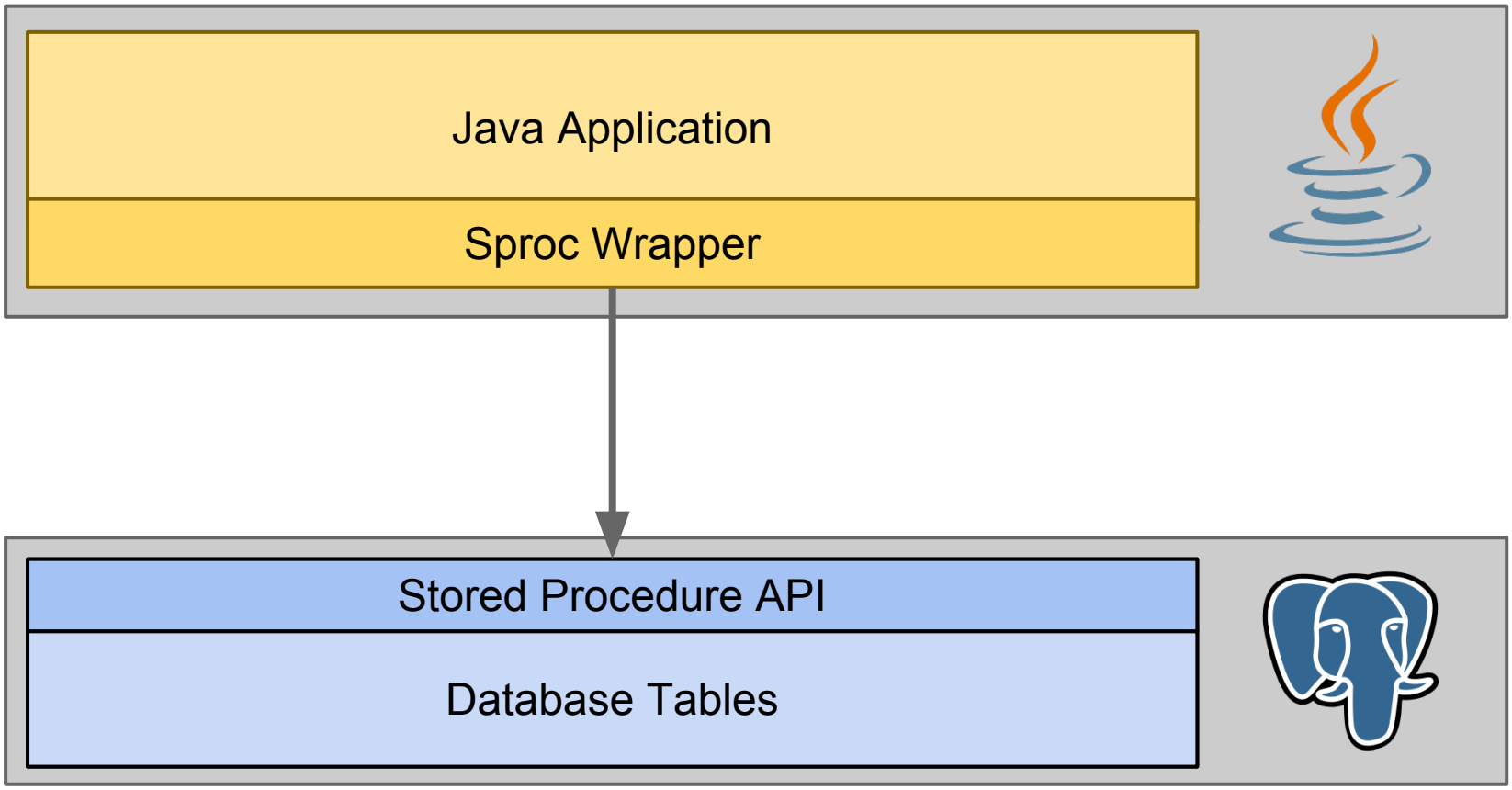
Sharded database

- Joins only between entities aggregates
- BI need more tooling
- Accessing data needs more tooling
- Managing more servers needs more tooling

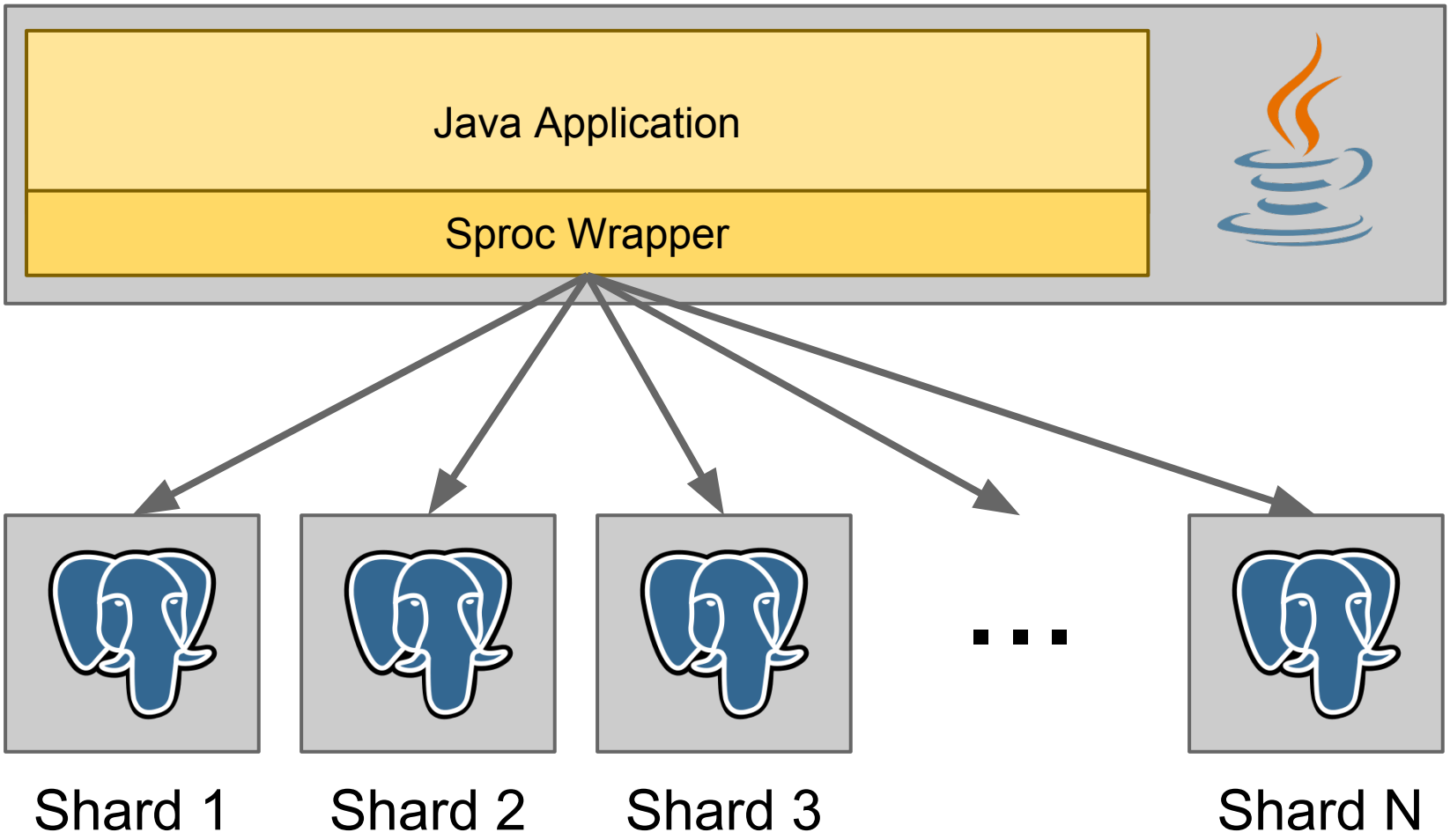
Sharded database

- Need more tooling

Sharding without limits



Sharding without limits



Sharding with Java Sproc Wrapper

JAVA

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

JAVA

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

JAVA

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

Sharding with Java Sproc Wrapper

Entity lookup strategies

- search on all shards (in parallel)
- hash lookups
- unique *shard aware* ID
  - Virtual Shard IDs (pre-sharding)

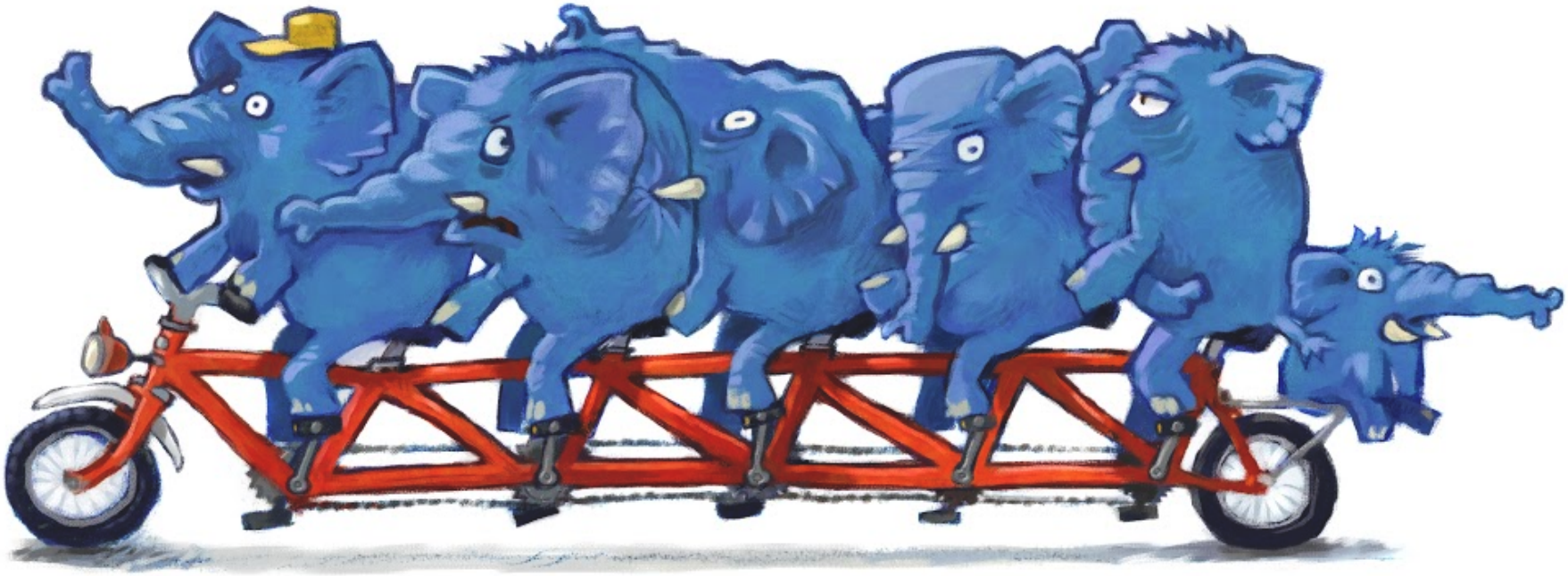
Links

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

zalando



zalando