
SHRIEK!

Enabling development teams to move fast

PostgreSQL at Zalando

About us

Valentine Gogichashvili

Database Engineer @Zalando

twitter: **@valgog**

google+: **+valgog**

email: valentine.gogichashvili@zalando.de

About us

Oleksii Kliukin
Database Engineer @Zalando
PostgreSQL Contributor
twitter: **@alexeyklyukin**
email: oleksii.kliukin@zalando.de


ZALANDO
SHOP
NEU

zalando

One of Europe's largest online fashion retailers

15 countries

3 fulfillment centers

15 million active customers

1.5 billion € revenue Q1-Q3 2014

640+ million visits in first half-year 2014

Visit us at tech.zalando.com

SALANDO
SHOP


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

Some more numbers

200+ deployment units (WARs)

1300+ production tomcat instances

80+ database master instances

90+ different database schemas

300+ developers and 200+ QA and PM

10 database engineers

Even more numbers

- > 4.0 TB of PostgreSQL data
- Biggest instances (not counted before)
  - `eventlogdb (3TB)`
    - 20 GB per week
  - `riskmgmtdb (5TB)`
    - 12 GB per day

Biggest challenges

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

Agenda

How we

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

Agenda

How we

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

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

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

Accessing data

ORM

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

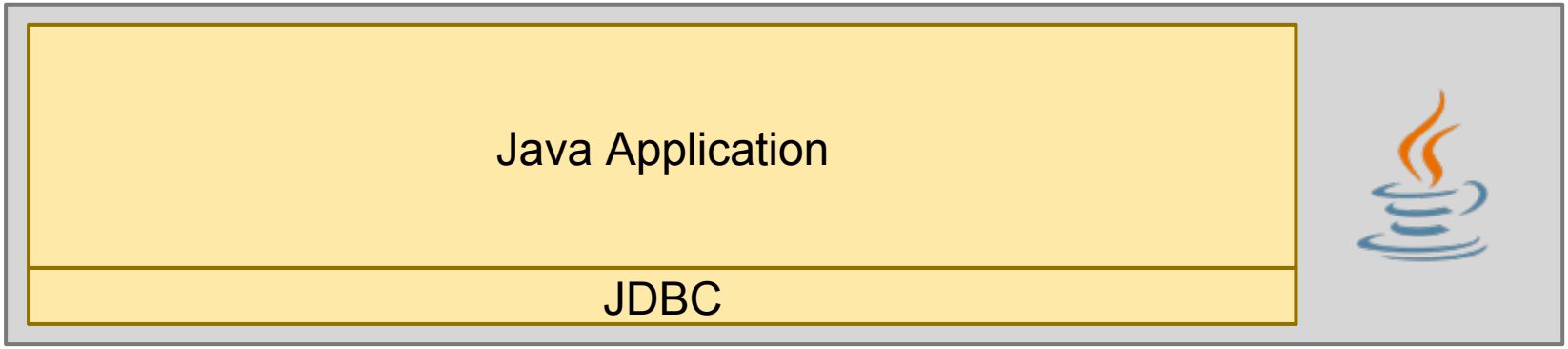
- error prone transaction management
- you have to reflect your tables in your code
- all business logic inside your application
- schema changes are not easy

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 Application
JDBC


Database Tables

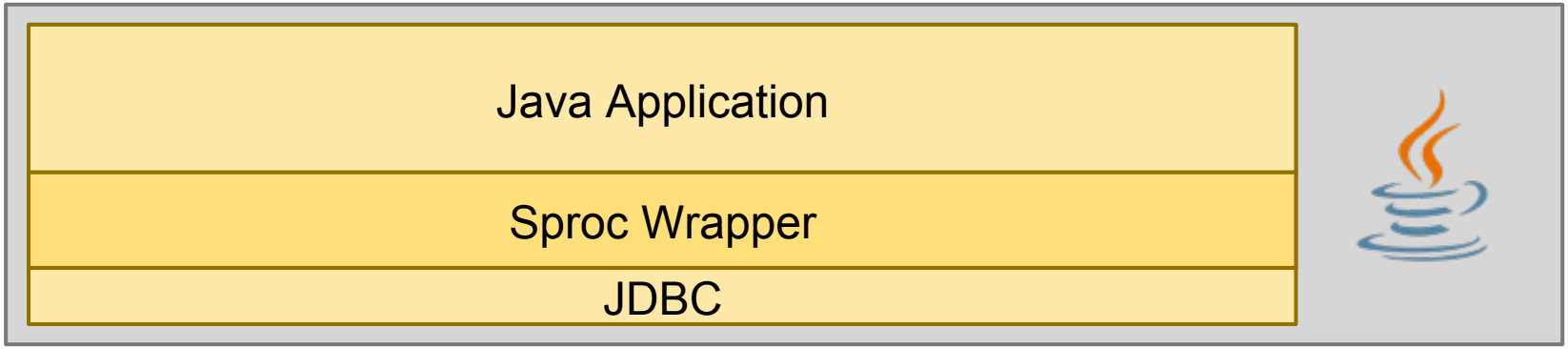
Java Application

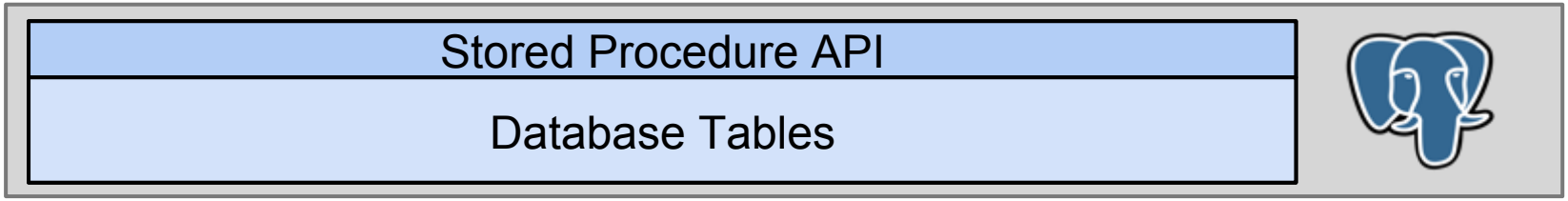
JDBC

Stored Procedure API

Database Tables

Java Sproc Wrapper





Java Sproc Wrapper

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;
```

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
@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

Stored Procedures
for developers

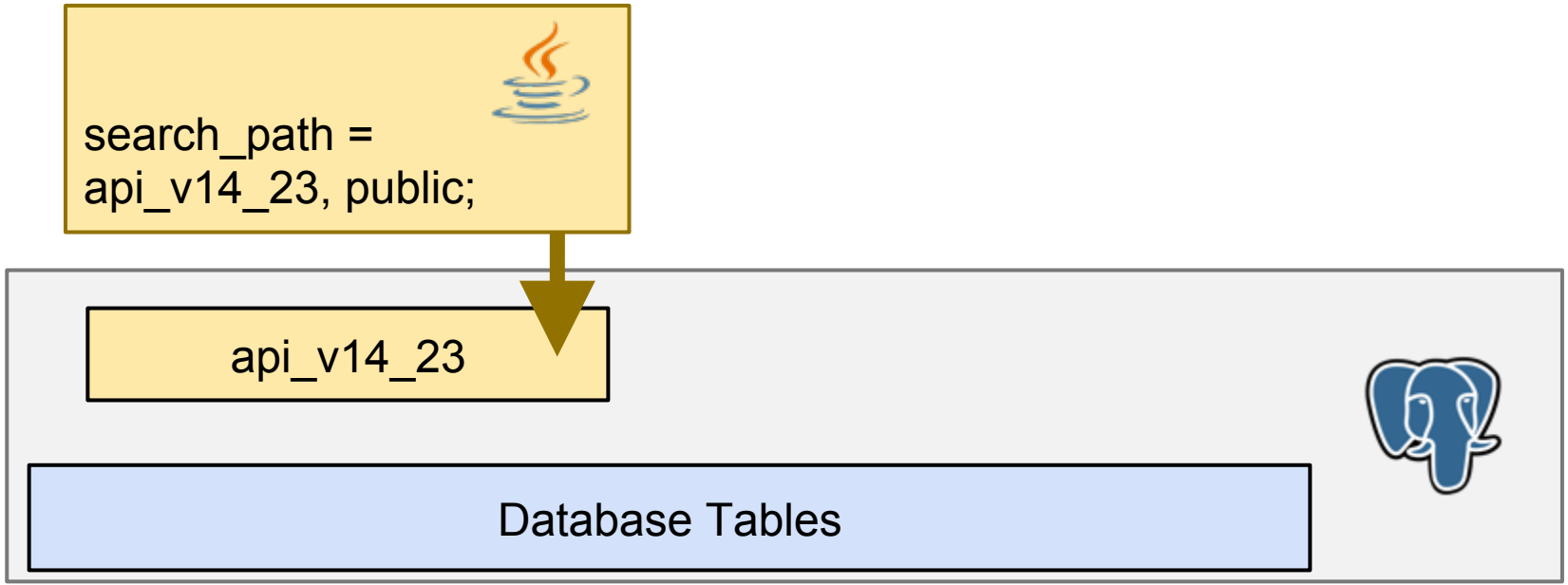
- CRUD operations need too much code
- Developers have to learn SQL
- Developers can write bad SQL
- Code reviews are needed

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

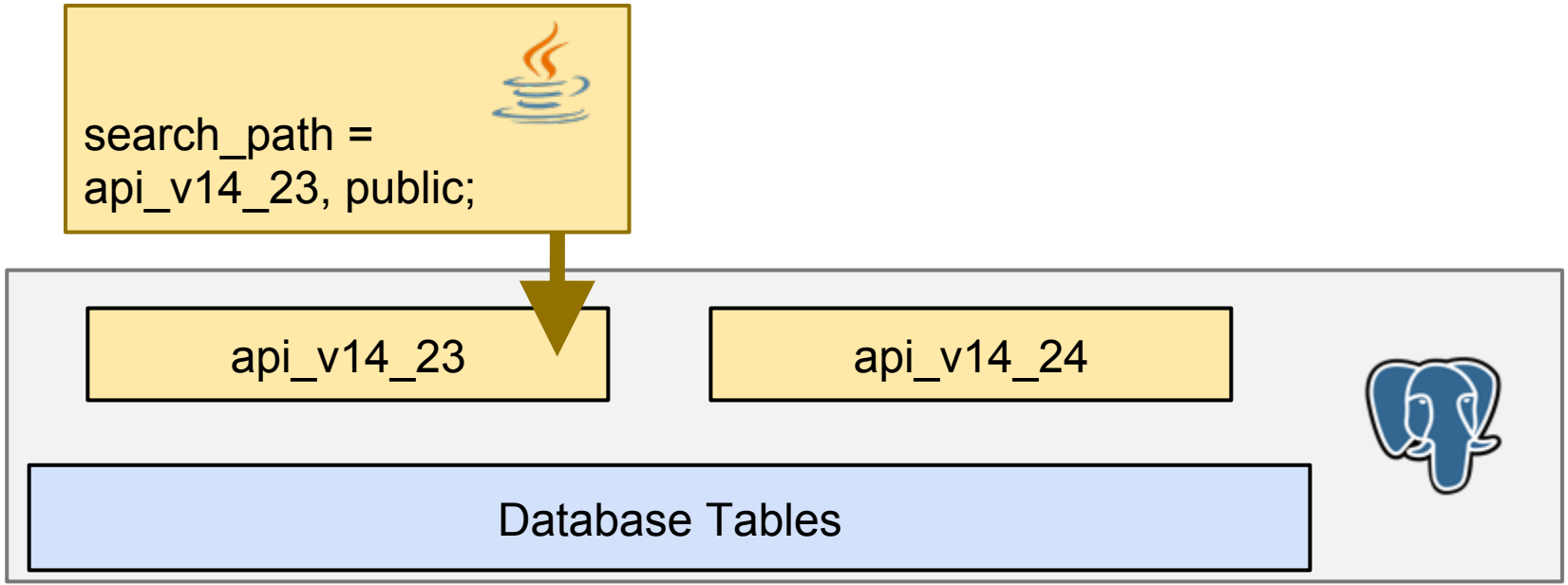
Horror story

- Never map your data manually
- Educate developers
- Educate yourself

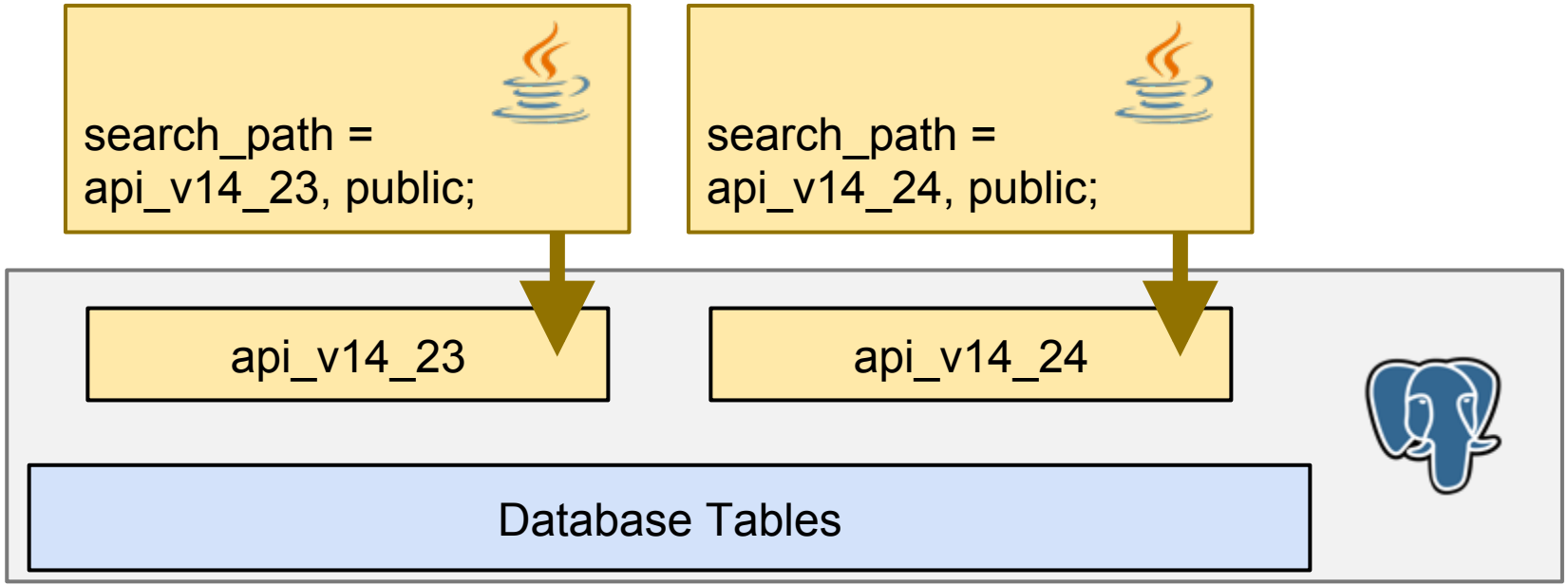
Stored Procedure
API versioning



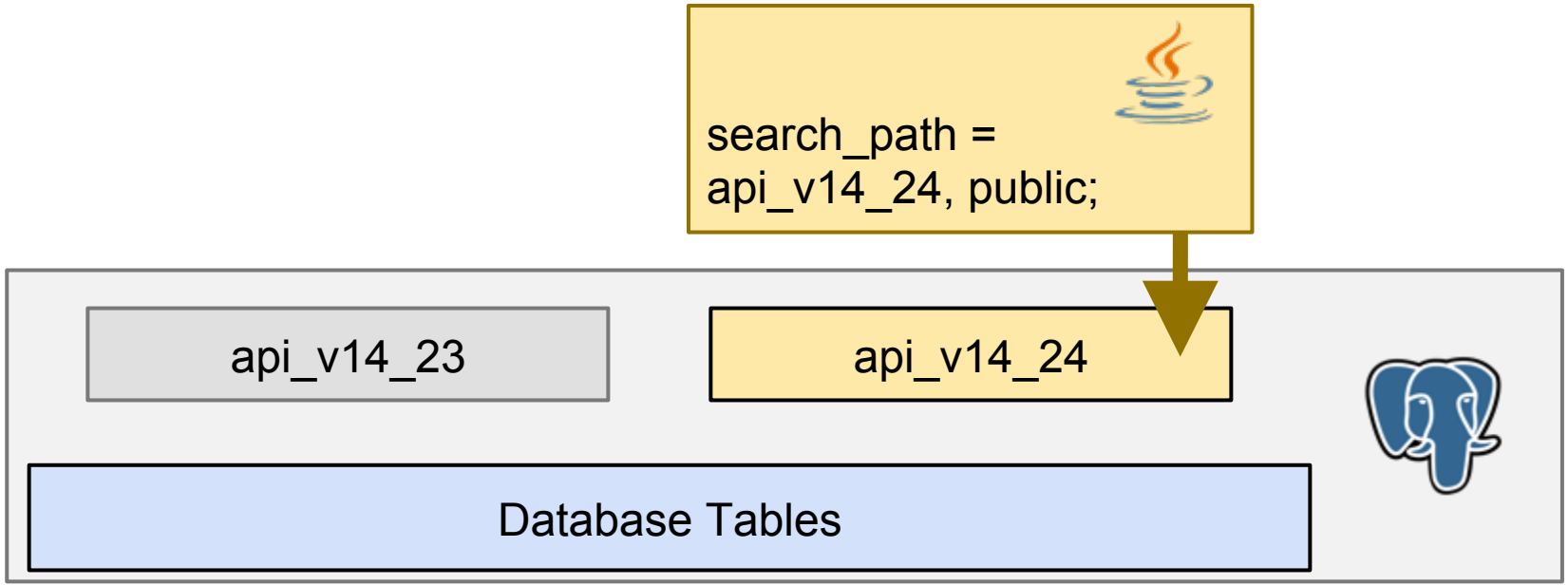
Stored Procedure
API versioning



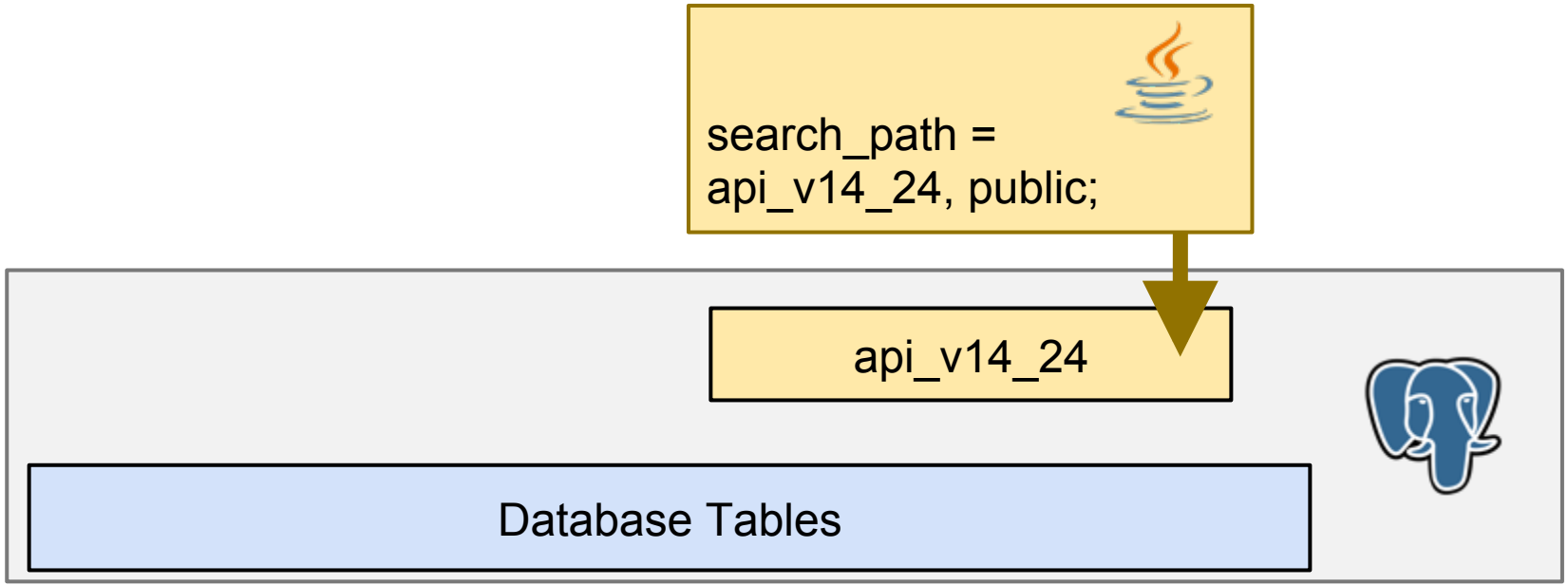
Stored Procedure
API versioning



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
- monitor

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`
  (will be much better in PostgreSQL 9.4)

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

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;
```

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

DBDIFF SQL

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

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

    SET statement_timeout TO ‘3s’;

    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

Easy schema changes

No downtime due to migrations or

deployment since we use PostgreSQL

Easy schema changes

One downtime due to migrations or

deployment since we use PostgreSQL

Horror story

- Invest in staging environments
- Do not become a bottleneck for developers
- Educate developers
- Educate yourself

Agenda

How we

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

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

Sharded database

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

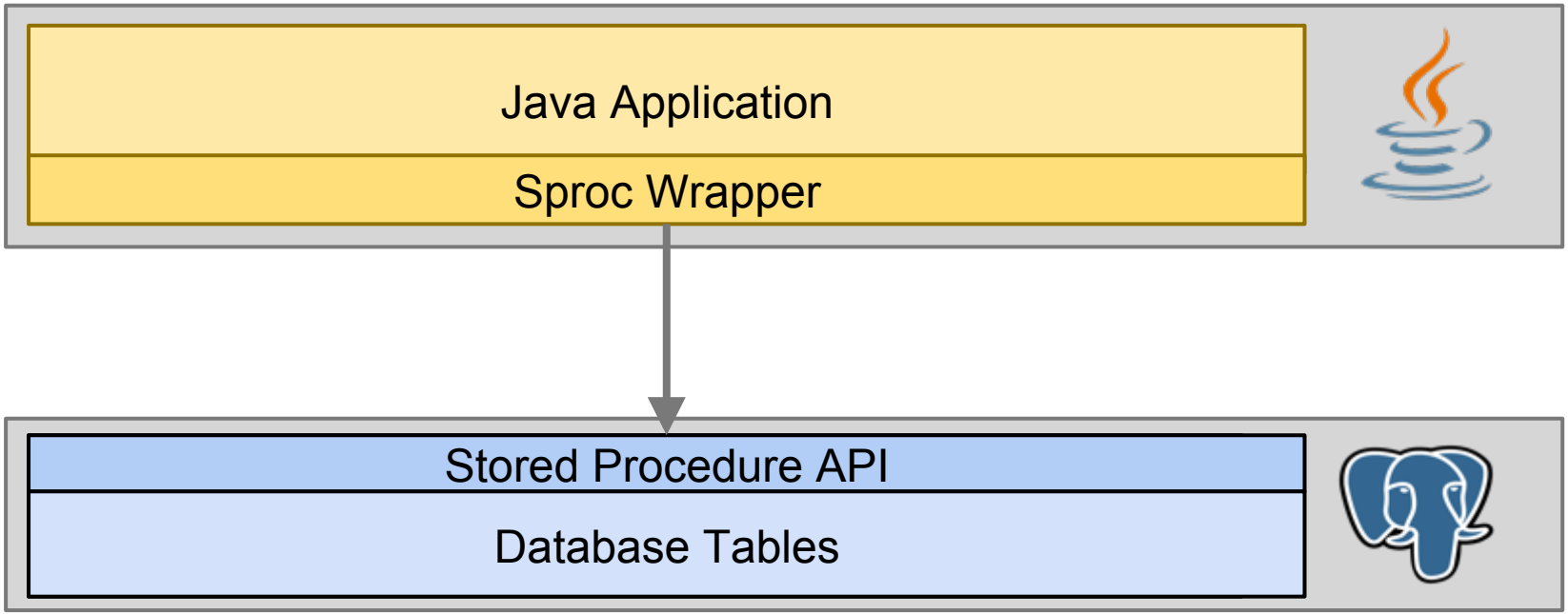
Sharded database

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

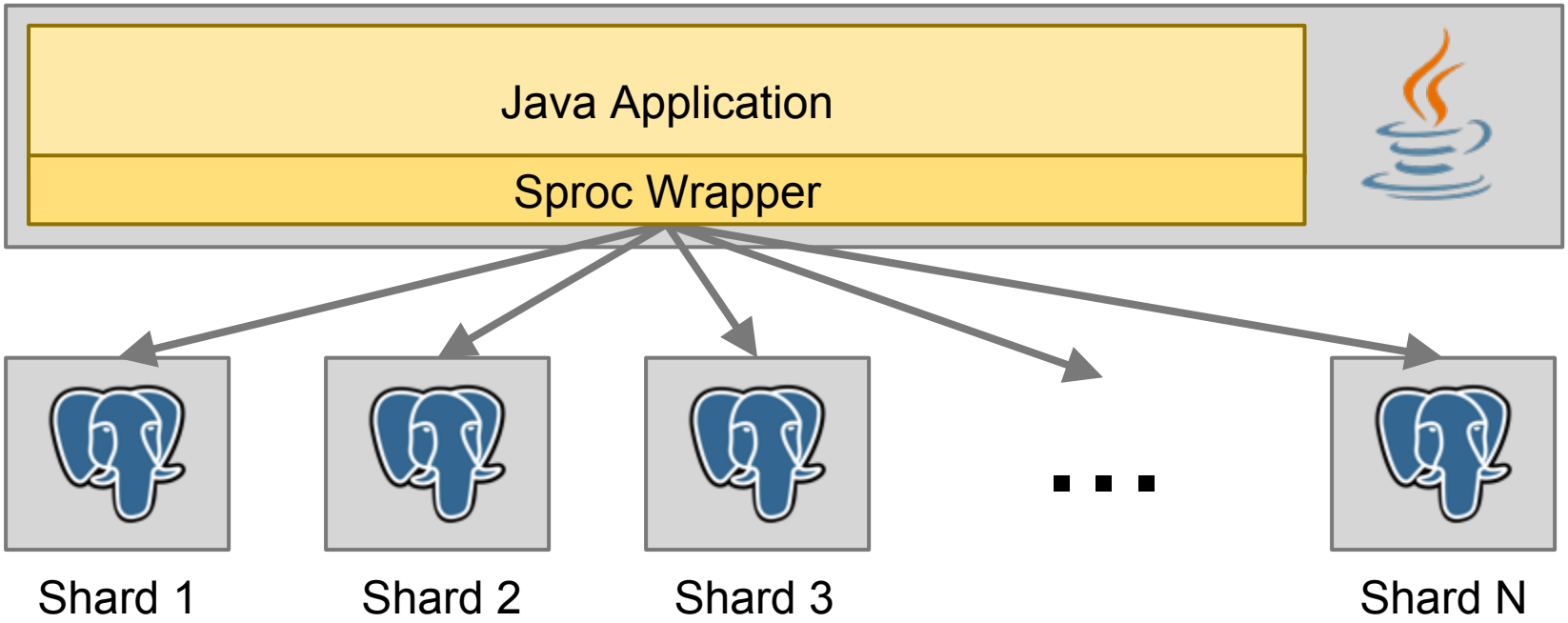
Sharded database

- 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);
```

Entity lookup strategies

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

Agenda

How we

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

Monitoring

pg_view

```
postgres@z-integrationdb: ~ 134x29
z-integrationdb up 117 days, 10:08:47 32 cores Linux 3.2.0-48-generic load average 0.73 0.63 0.51                    22:10:13
sys: utime 1.7   stime 0.3   idle 98.0  iowait 0.0   ctxt 2500 run 3   block 0
mem: total 251.9GB free 34.0GB buffers 1.6GB  cached 196.4GB dirty 10.0MB limit 127.8GB as 33.6GB left 94.2GB
integration93 9.3 database connections: 8 of 800 allocated, 2 active
type dev                fill   total   left   read  write    await path_size path
data mapper/vg01-data1   0.0   2.2TB  1.3TB                          37.1MB /data/postgres/pgsql_integration93/9.3/data
xlog sda9                0.0 119.0GB 88.5GB    0.0    0.3      0.0    64.0MB /data/postgres/pgsql_integration93/9.3/data/pg_xlog/
  pid type    s utime stime guest read write   age db            user          query
 4595 backend S   0.0   0.0   0.0  0.0   0.0 01:37 integr..ory_db vgogichashvili idle in transaction
 5019 backend S   0.0   0.0   0.0  0.0   0.0 01:02 integr..ory_db vgogichashvili select 'Cool tool' from pg_sleep(19000);

s: System processes   f: Freeze output   u: Measurement units   a: Autohide fields   t: No trim   r: Realtime   h: Help      v.1.1.0
```

Monitoring

- Nagios/Icinga was replaced by ZMON2
- Dedicated 24x7 monitoring team
- Custom monitoring infrastructure ZMON2

PGObserver

perftables/

Possible Table access/growth issues report

Hostname: bm.db.zalando

Timeframe: 2013-10-26 2013-10-29 show

Host	Schema	Table	Date	Scan change %	Scans1	Scans2	Size1	Size2
bm.db.zalando	zbm_data	sales_rule_set	2013-10-28	84.21 (50)	38	70	2022 MB	2022 MB

perfindexes/

Possible Index issues report

Hostname: catalog1.db.zalando

show

Invalid indexes (in total size of 0 bytes)

Hostname	Table name	Index name	Index size	% of table's indexes	Table size
catalog1.db.zalando	zcat_data.article_config	zcat_data.article_config_c1_c2_null_null_null_uidx	0 bytes	0.0% of 87 MB	42 MB

Duplicate indexes

Hostname	Table name	Table size	Index definition	Count
catalog1.db.zalando	zcat_data.article_simple	171 MB	CREATE INDEX X ON zcat_data.article_simple USING btree (as_simple_sku_id)	2
catalog1.db.zalando	zcat_data.article_config	130 MB	CREATE INDEX X ON zcat_data.article_config USING btree (ac_config_sku_id)	2
catalog1.db.zalando	zcat_data.article_model	47 MB	CREATE INDEX X ON zcat_data.article_model USING btree (am_model_sku_id)	2
catalog1.db.zalando	zcat_commons.size	12 MB	CREATE INDEX X ON zcat_commons.size USING btree (s_size_chart_code, s_code)	2
catalog1.db.zalando	zcat_commons.price_level	96 kB	CREATE INDEX X ON zcat_commons.price_level USING btree (pl_level)	2

Unused indexes

Hostname	Table name	Index name	Scans	Index size	% of table's indexes	Table size

PGObserver

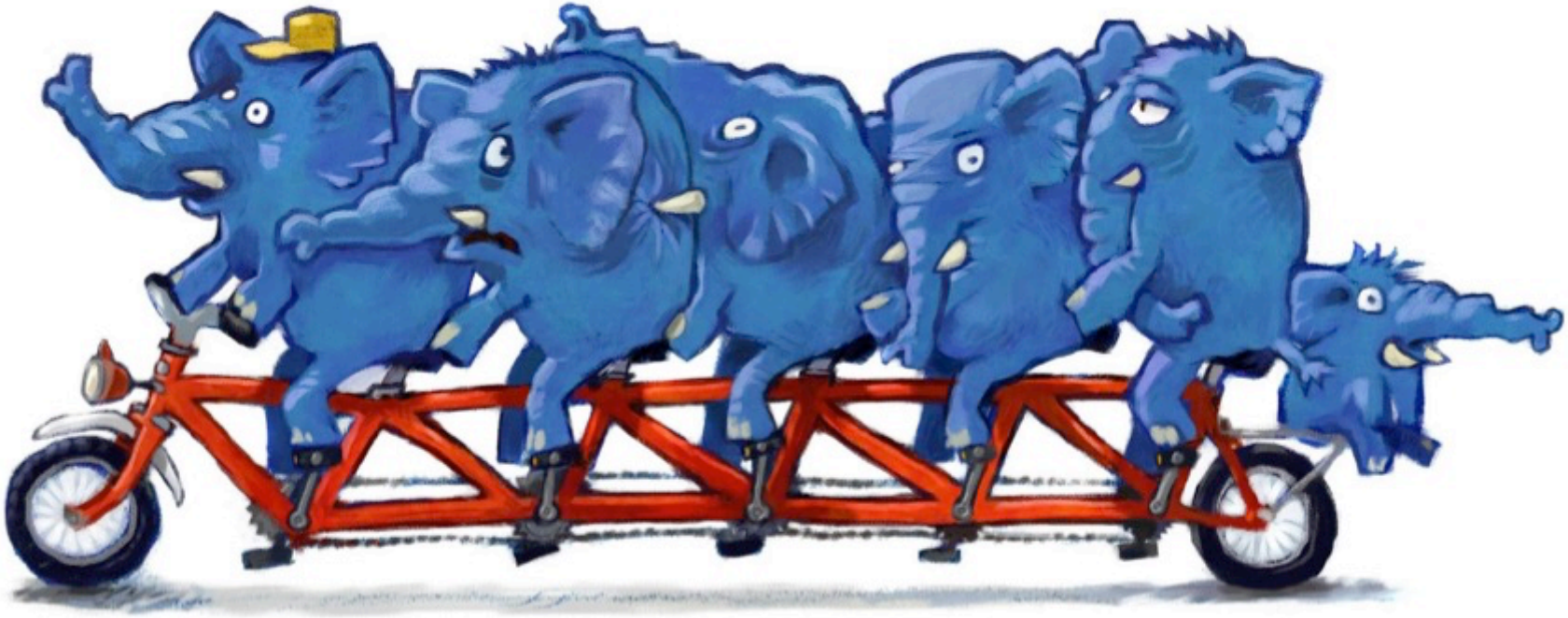
zalando

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



Thank you!