SphereEx

PostgreSQL Distributed & Secure Database Ecosystem Building

Trista Pan, panjuan@apache.org

About Me

Trista Pan

SphereEx Co-Founder & CTO
Apache Member
AWS Data Hero
Tencent Cloud TVP
Apache ShardingSphere PMC
Apache brpc (Incubating) & Apache AGE (Incubating) mentor
China Mulan Community Mentor

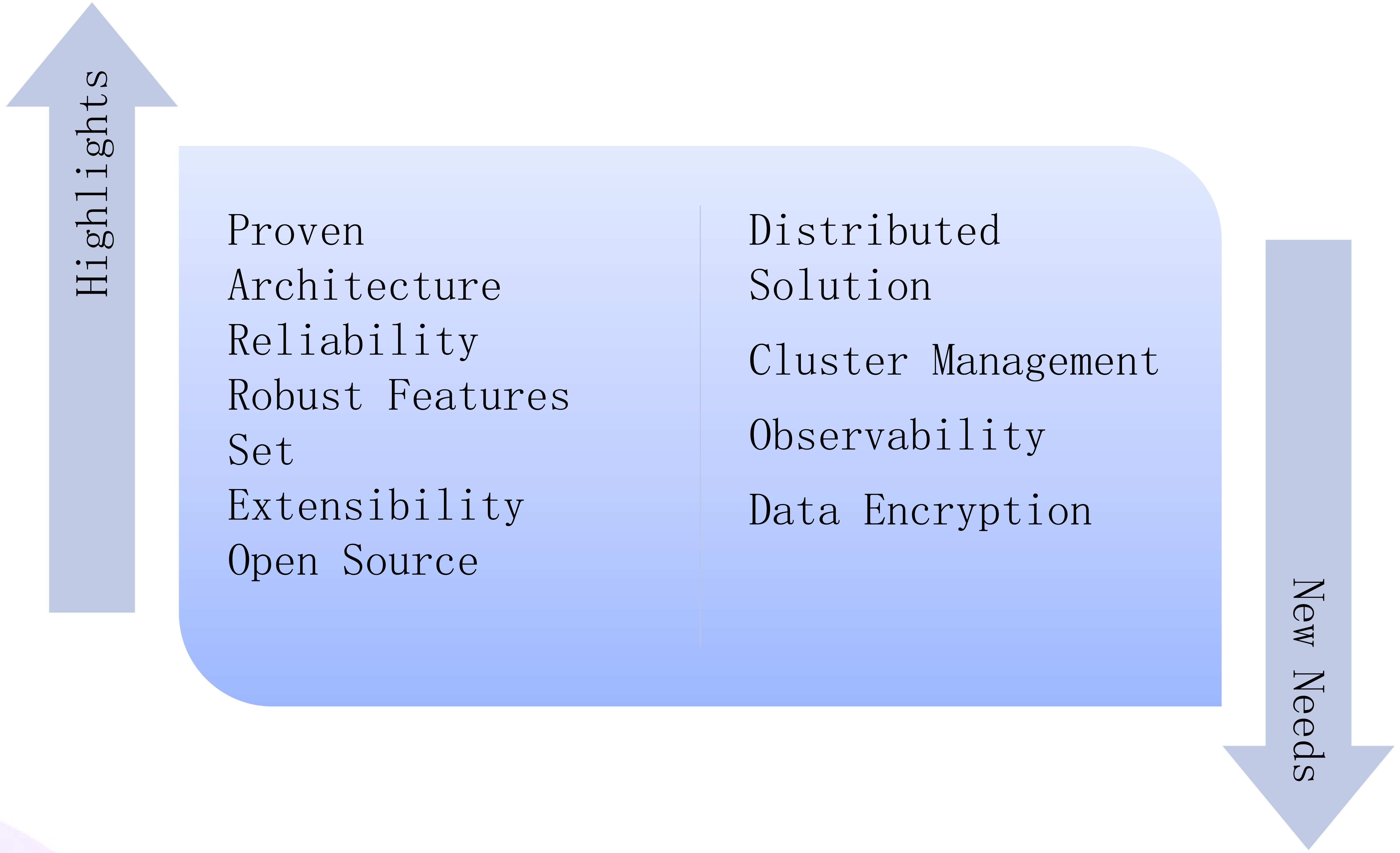
Bio: https://tristazero.github.io
LinkedIn: https://www.linkedin.com/in/panjuan
GitHub: https://github.com/tristaZero
Twitter: @tristaZero
Project Twitter: @ShardingSphere

Contents

- Why You’re Expecting More From PostgreSQL
- What is Apache ShardingSphere?
- What is the Value of this Project?
- Distributed SQL’s Kernel Processing
- Distributed PostgreSQL Database Solution
- Data Encryption with PostgreSQL

New Needs

PostgreSQL: The World's Most Advanced Open Source Relational Database



- Proven Architecture
- Reliability
- Robust Features Set
- Extensibility
- Open Source

- Distributed Solution
- Cluster Management
- Observability
- Data Encryption

Apache ShardingSphere Community Overview

Why Open Source Software

- Build new software easily and fast.
- Prevent vendor lock effectively.
- Proved to be secure and reliable.
- Fast Upgrade & Low Switching Cost
- Sound community ecosystem & quality software.

+

THE APACHE SOFTWARE FOUNDATION

- The World's Largest Open Source Foundation
- 227M+ Lines of Code in Stewardship
- 300+ Top-Level Projects

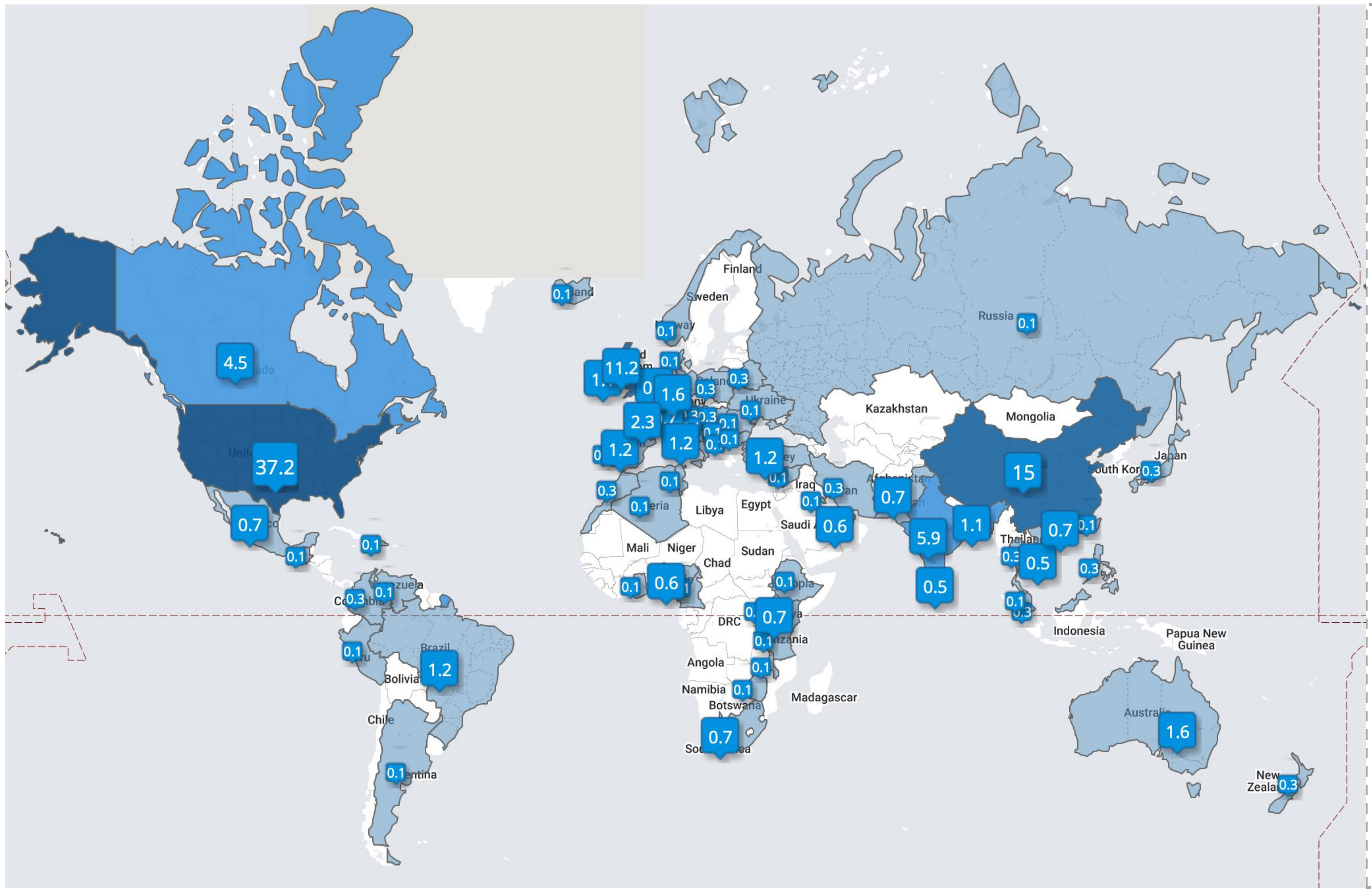
POWERED BY ShardingSphere APACHE

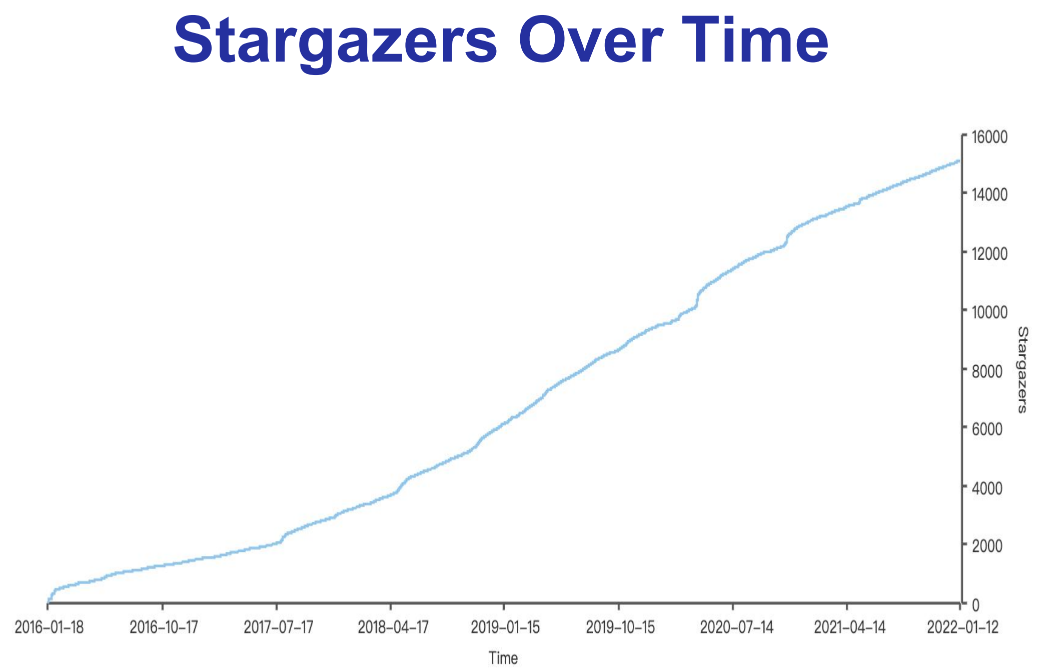
- **15000+** Stars
- **300+** Contributors
- **5000+** Forks
- **8000+** Pull Requests

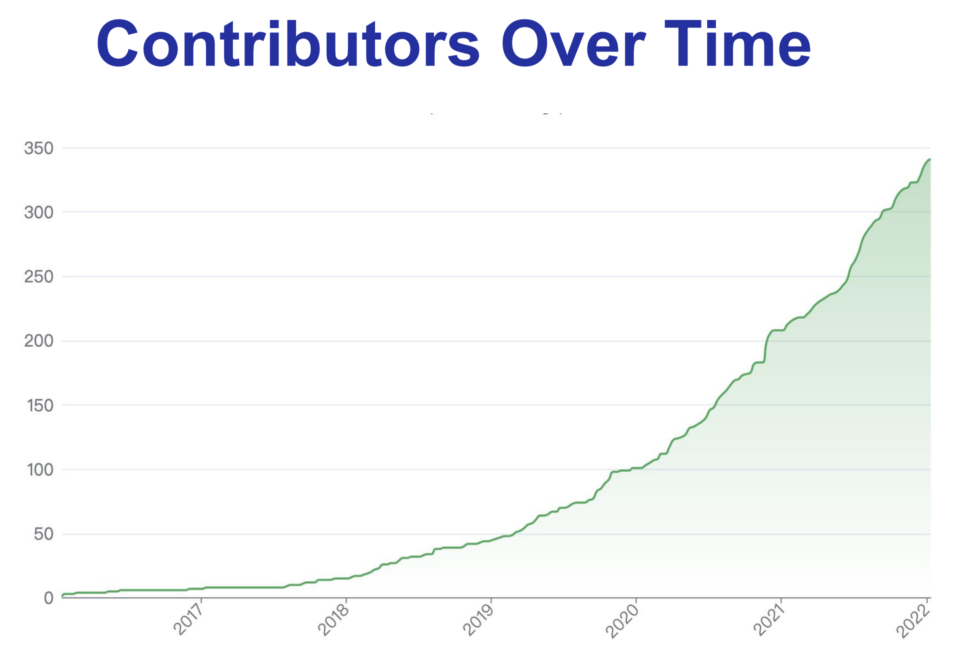
- ShardingSphere is one of the Top-Level Apache Projects.
- Top Chinese Led Apache Project, 2020
- Top 10 Apache Project by Number of Commits, 2021

Apache ShardingSphere's Global Development

Twitter Social Followers





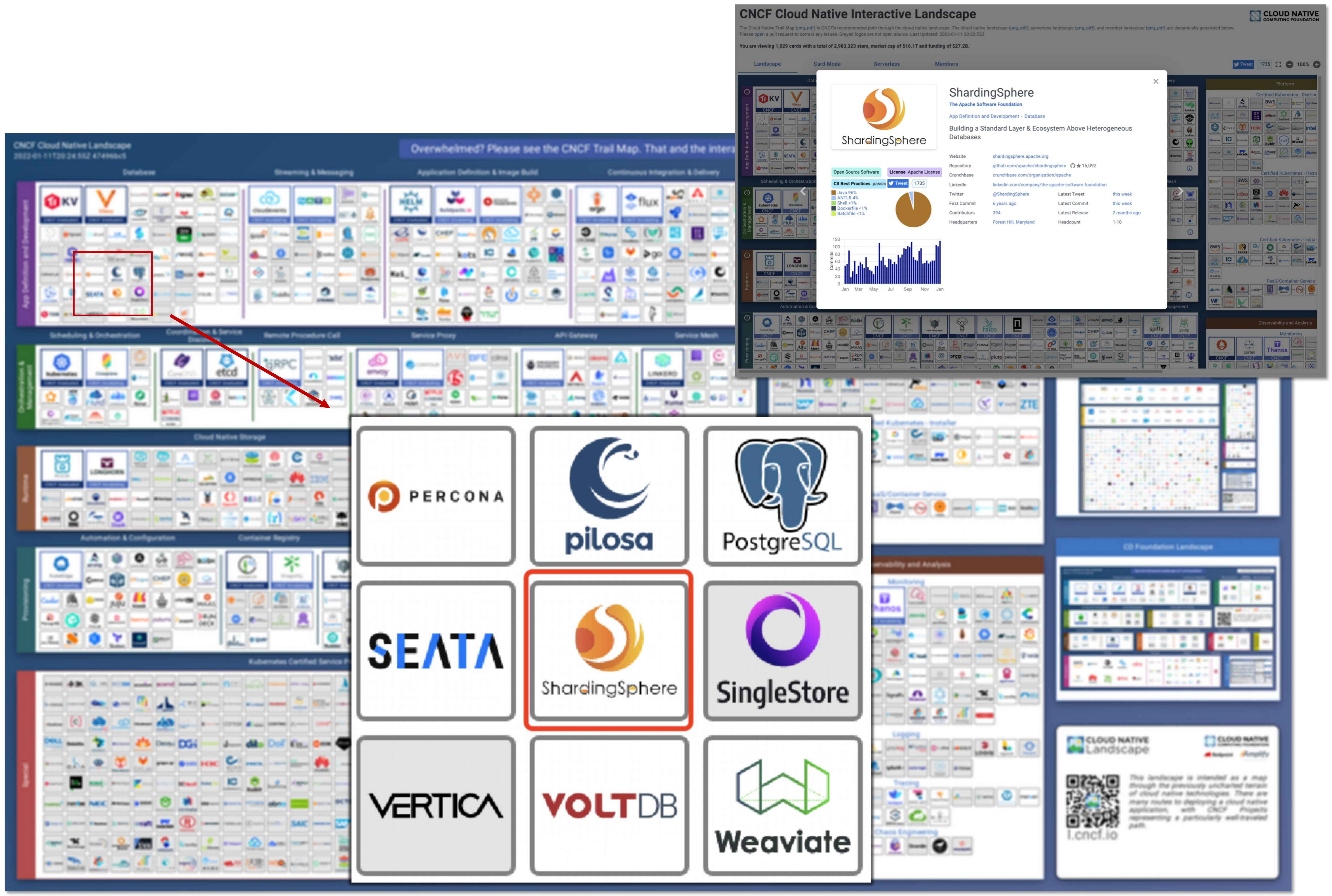


Apache ShardingSphere GitHub Stargazers & Contributors Over Time

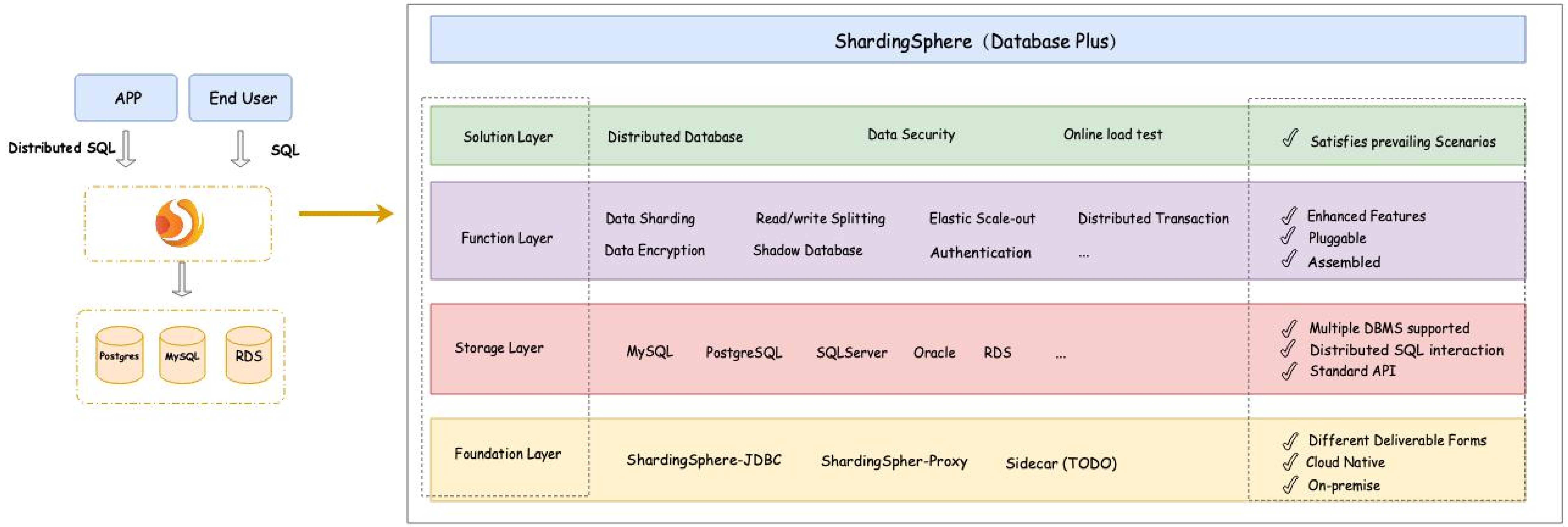
CNCF：Cloud Native Computing Foundation

- As a part of the Linux Foundation, CNCF's mission is to make cloud native computing ubiquitous.

- Currently listed are 1,029 cards with a total of 2,984,816 Stars, a market cap of $16.3T and funding of $27.2B.


CNCF Cloud Native Interactive Landscape
CNCF Cloud Native Landscape
Overwhelmed? Please see the CNCF Trail Map. That and the intera
ShardingSphere
The Apache Software Foundation
App Definition and Development · Database
Building a Standard Layer & Ecosystem Above Heterogeneous Databases
PERCONA
pilosa
PostgreSQL
SEATA
ShardingSphere
SingleStore
VERTICA
VOLTDB
Weaviate

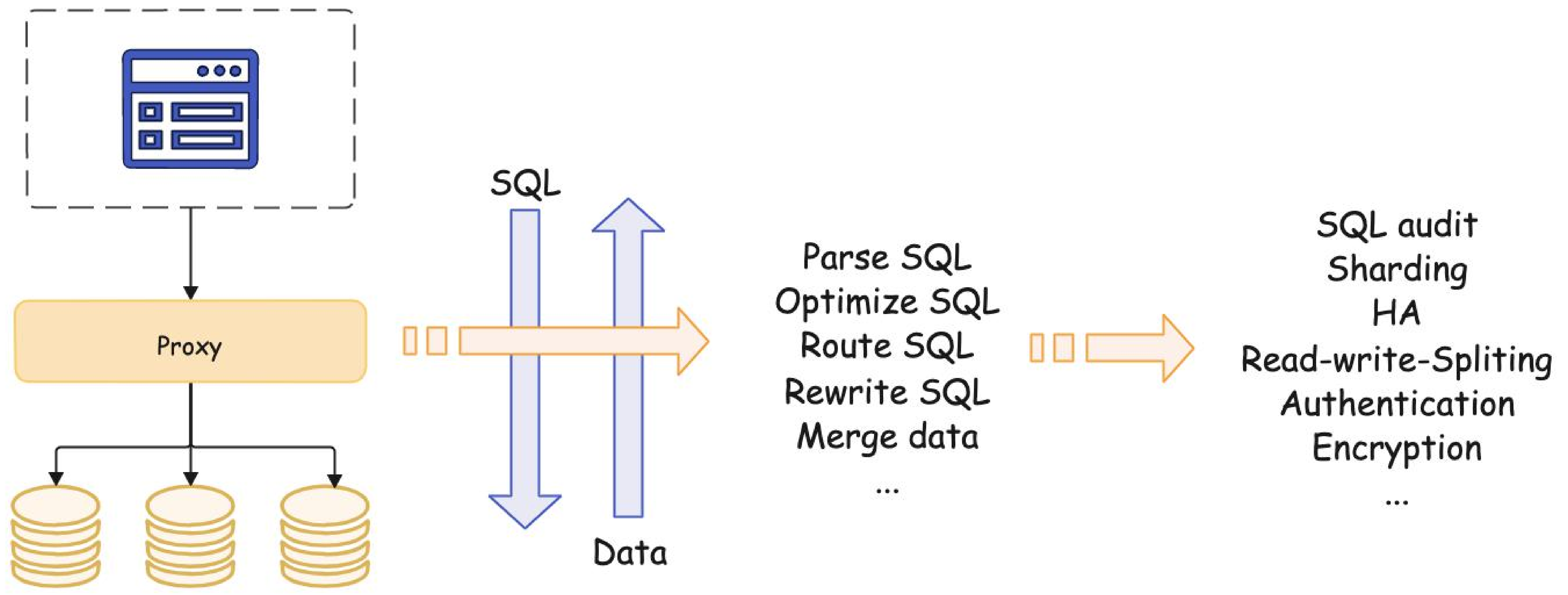
Apache ShardingSphere



ShardingSphere (Database Plus)

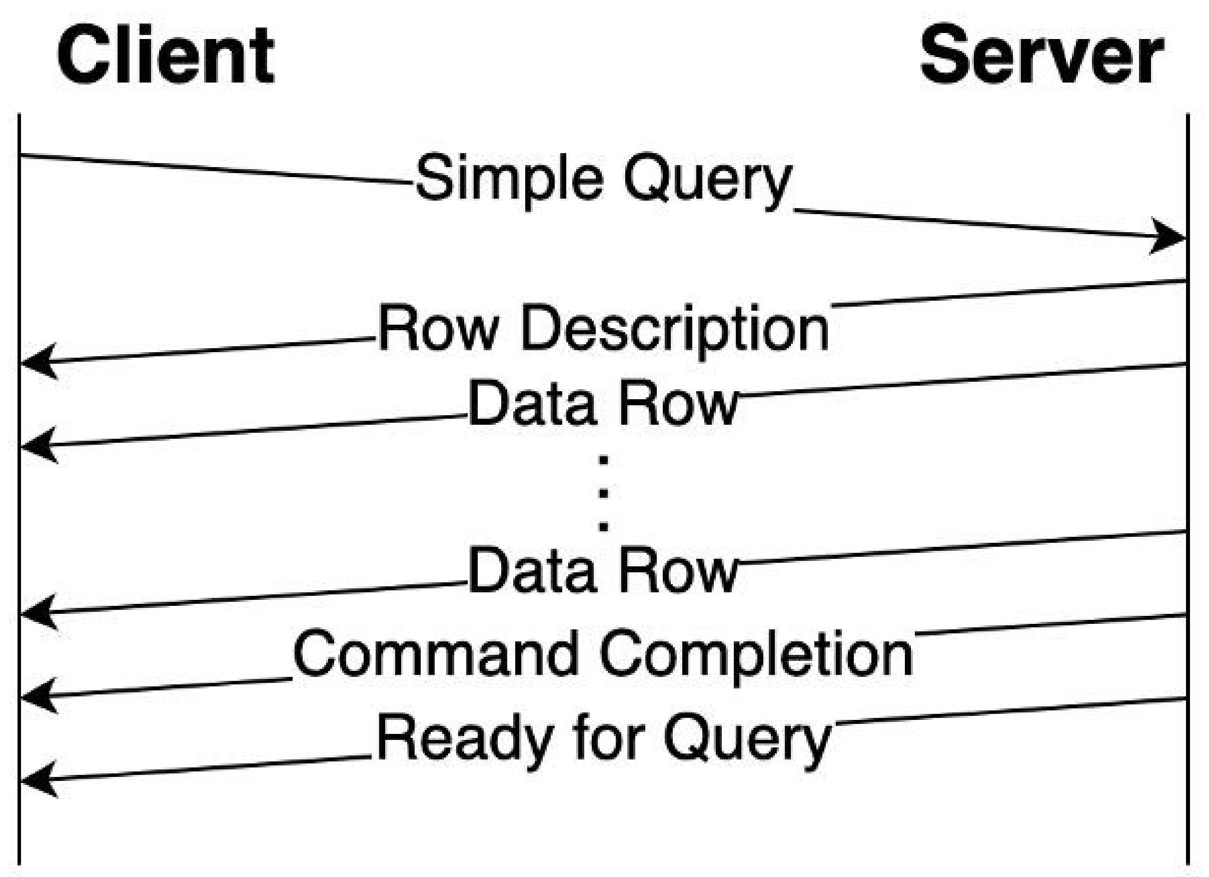
Solution Layer	Distributed Database	Data Security	Online load test		✓ Satisfies prevailing Scenarios
Function Layer	Data Sharding Data Encryption	Read/write Splitting Shadow Database	Elastic Scale-out Authentication	Distributed Transaction ...	✓ Enhanced Features ✓ Pluggable ✓ Assembled
Storage Layer	MySQL PostgreSQL	SQLServer Oracle	RDS ...		✓ Multiple DBMS supported ✓ Distributed SQL interaction ✓ Standard API
Foundation Layer	ShardingSphere-JDBC	ShardingSpher-Proxy	Sidecar (TODO)		✓ Different Deliverable Forms ✓ Cloud Native ✓ On-premise

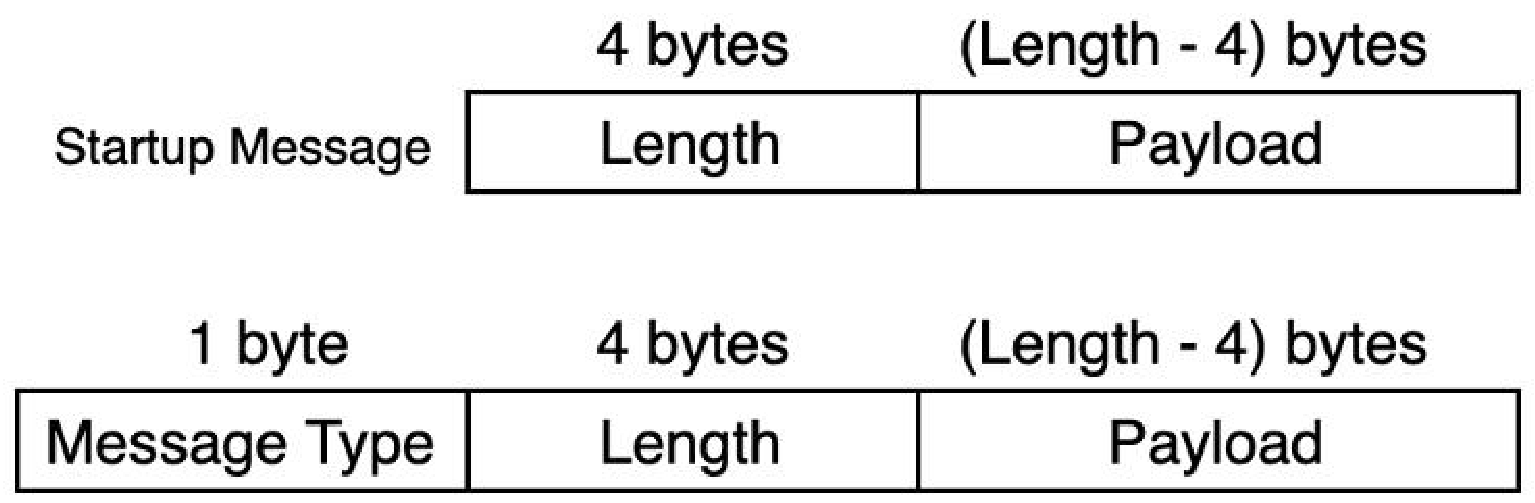
Kernel Process

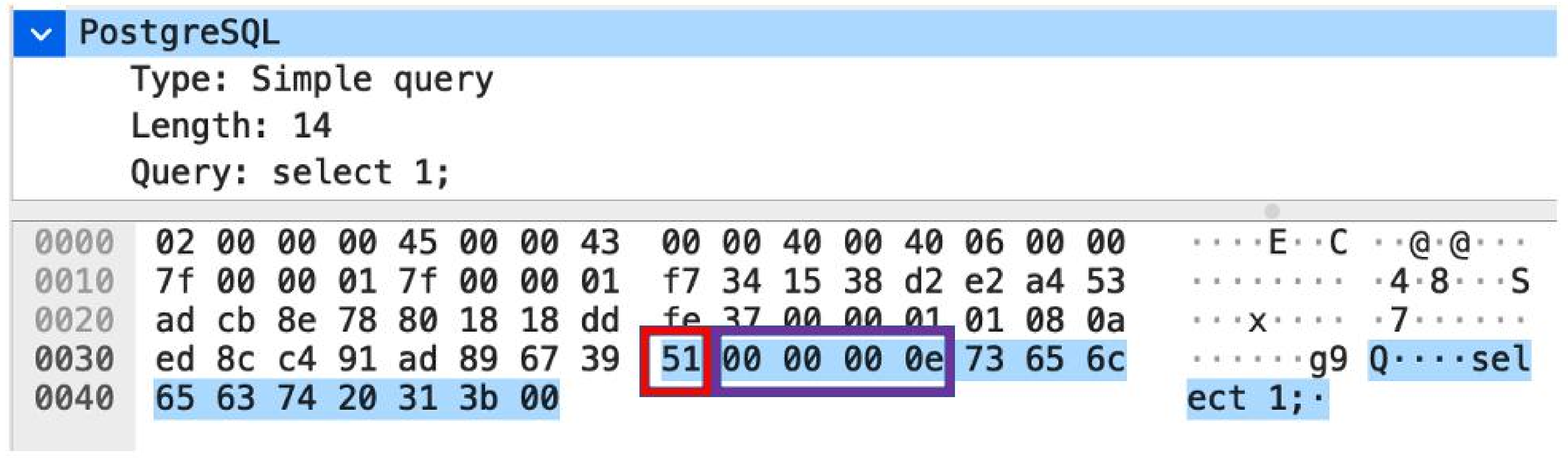


Kernel Process

```
biz_order=# select order_id, create_time from t_order limit 3;
      order_id       |        create_time
---------------------+-------------------------
 614158771245527040 | 2021-09-01 10:32:44.893
 614127204259311616 | 2021-09-01 07:55:18.471
 608625285797490688 | 2021-08-31 18:17:42.173
(3 rows)
```

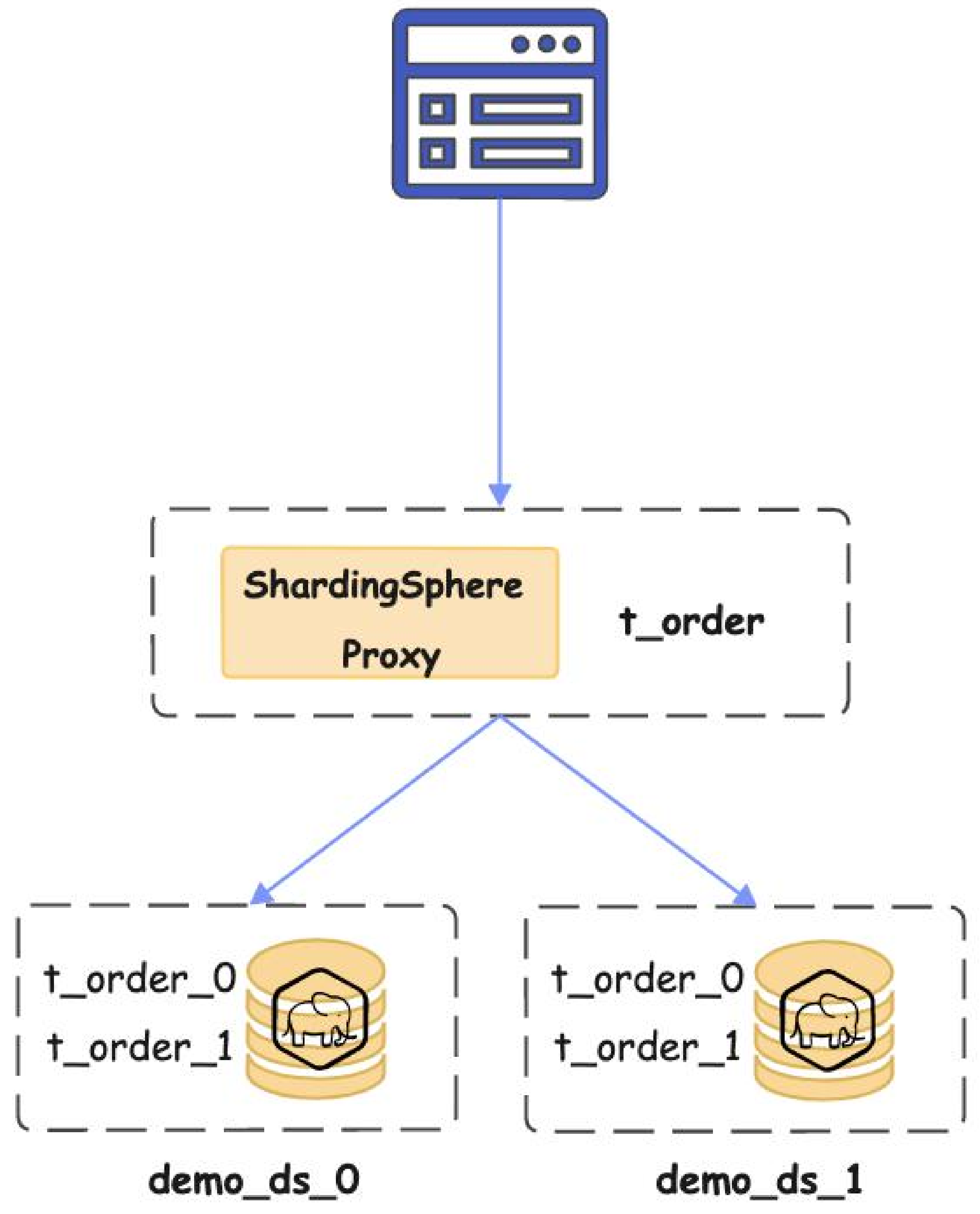


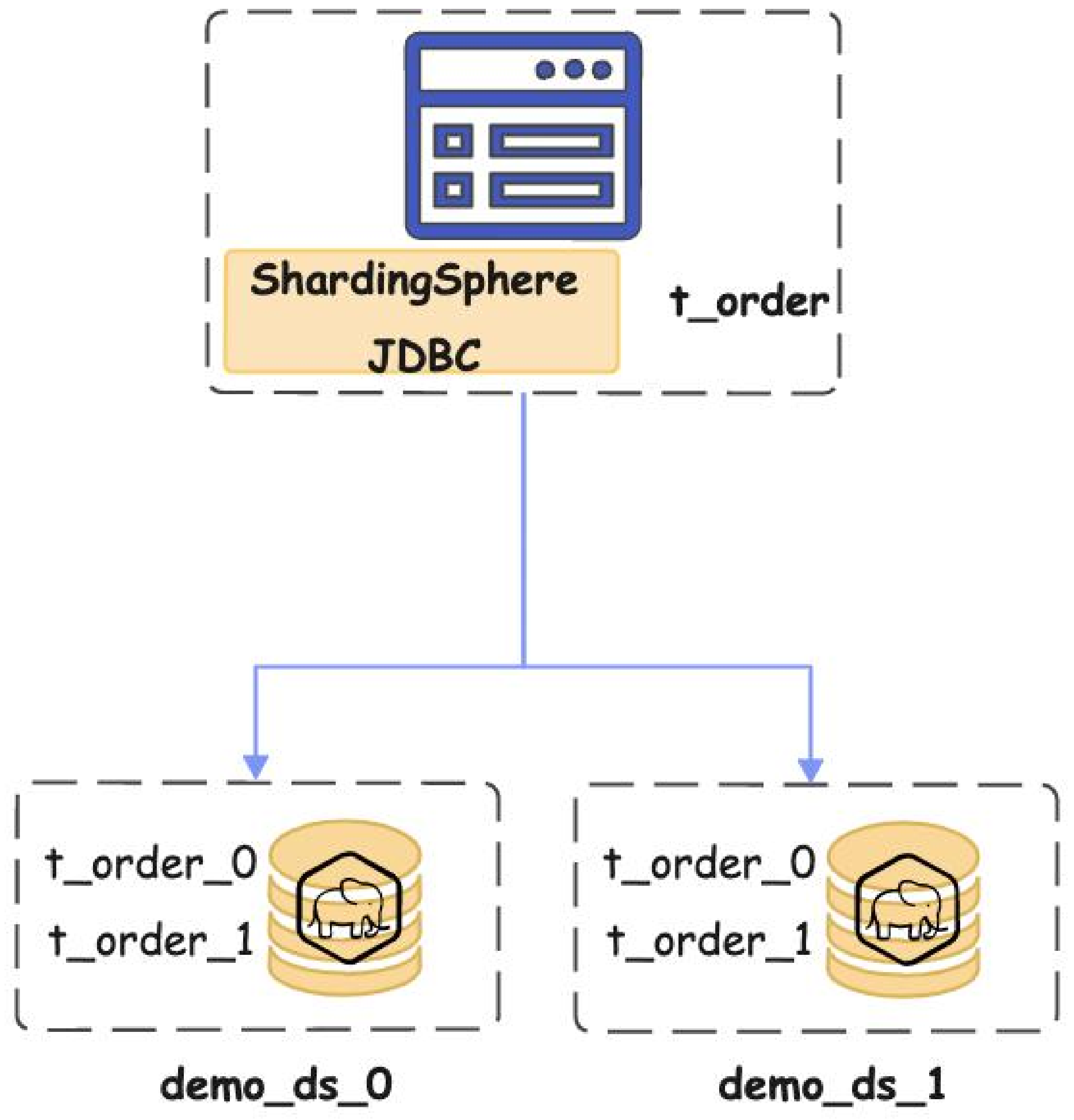




```
PostgreSQL
    Type: Simple query
    Length: 14
    Query: select 1;

0000  02 00 00 00 45 00 00 43  00 00 40 00 40 06 00 00   ····E··C ··@·@···
0010  7f 00 00 01 7f 00 00 01  f7 34 15 38 d2 e2 a4 53   ········ ·4·8···S
0020  ad cb 8e 78 80 18 18 dd  fe 37 00 00 01 01 08 0a   ···x···· ·7······
0030  ed 8c c4 91 ad 89 67 39  51 00 00 00 0e 73 65 6c   ······g9 Q····sel
0040  65 63 74 20 31 3b 00                               ect 1;·
```

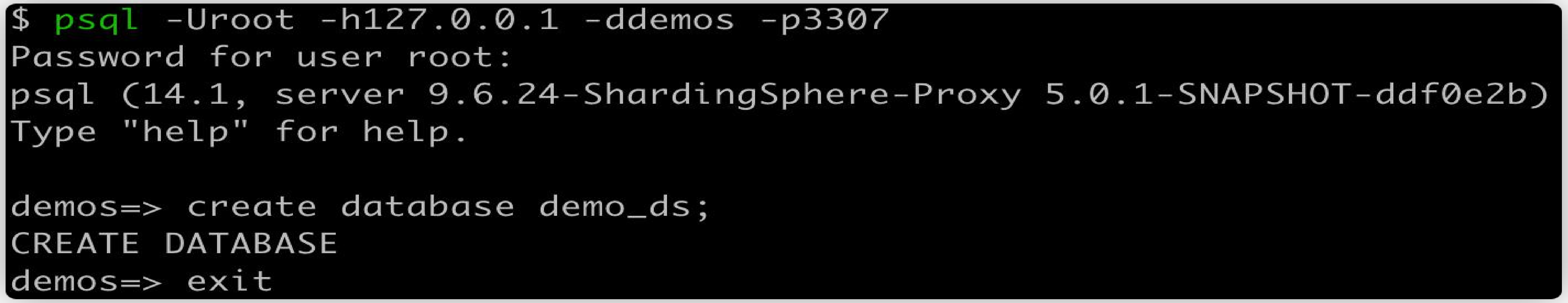
Distributed Postgres Database Solution





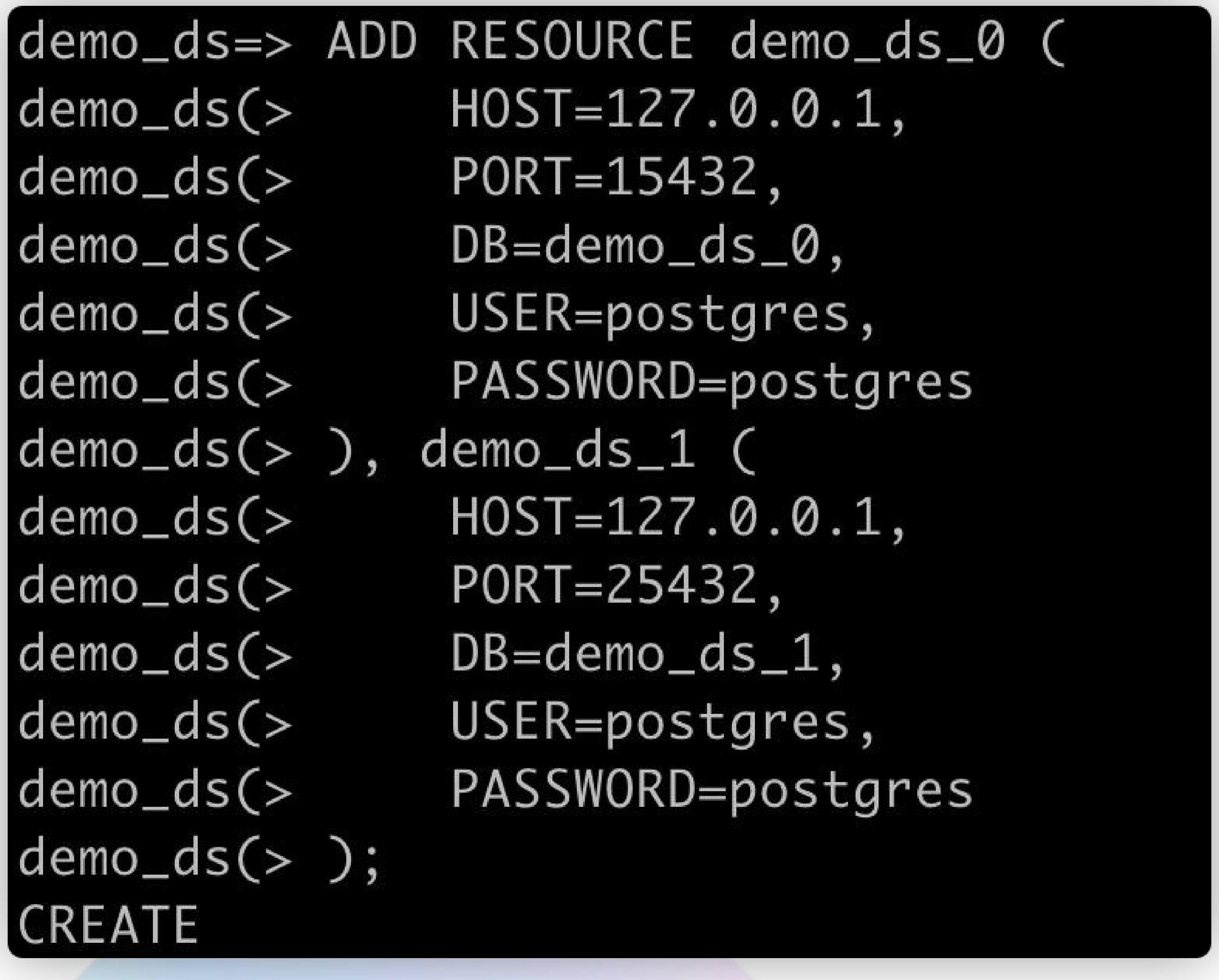
Distributed Postgres Database Solution

1. Create Database



```
$ psql -Uroot -h127.0.0.1 -ddemos -p3307
Password for user root:
psql (14.1, server 9.6.24-ShardingSphere-Proxy 5.0.1-SNAPSHOT-ddf0e2b)
Type "help" for help.

demos=> create database demo_ds;
CREATE DATABASE
demos=> exit
```

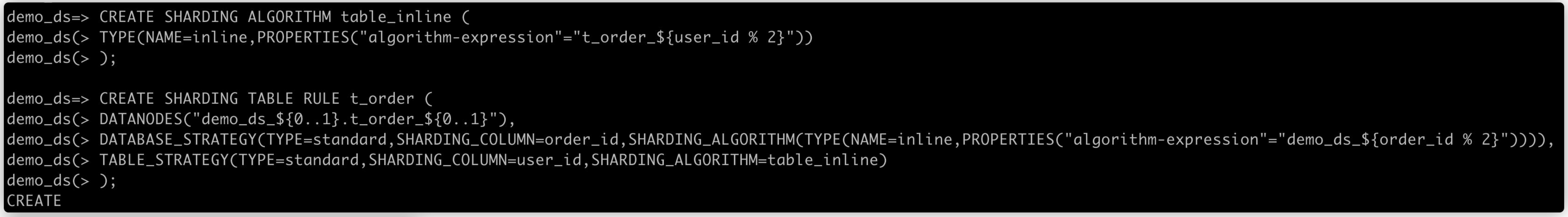
2. Add Data Source



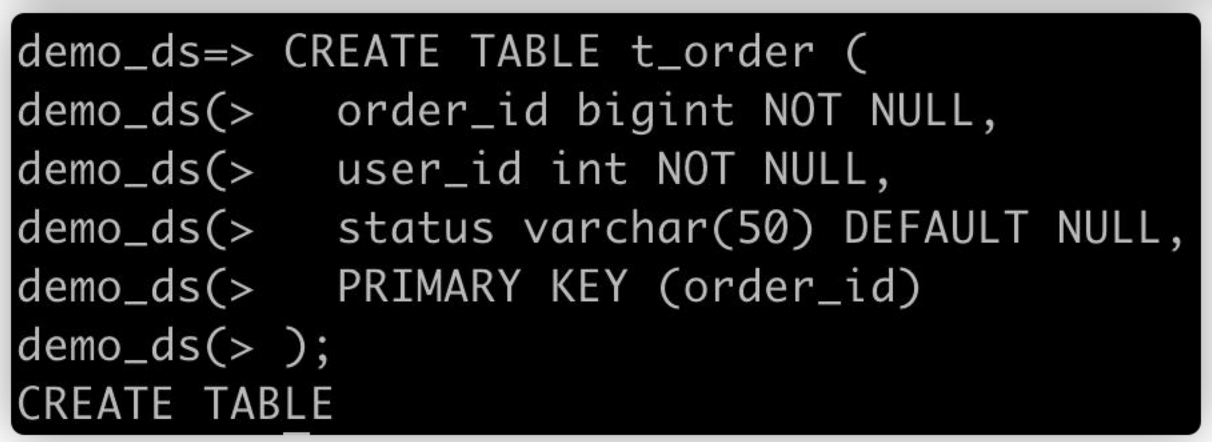
```
demo_ds=> ADD RESOURCE demo_ds_0 (
demo_ds(>      HOST=127.0.0.1,
demo_ds(>      PORT=15432,
demo_ds(>      DB=demo_ds_0,
demo_ds(>      USER=postgres,
demo_ds(>      PASSWORD=postgres
demo_ds(> ), demo_ds_1 (
demo_ds(>      HOST=127.0.0.1,
demo_ds(>      PORT=25432,
demo_ds(>      DB=demo_ds_1,
demo_ds(>      USER=postgres,
demo_ds(>      PASSWORD=postgres
demo_ds(> );
CREATE
```

Distributed Postgres Database Solution

3. Create Rules & Tables

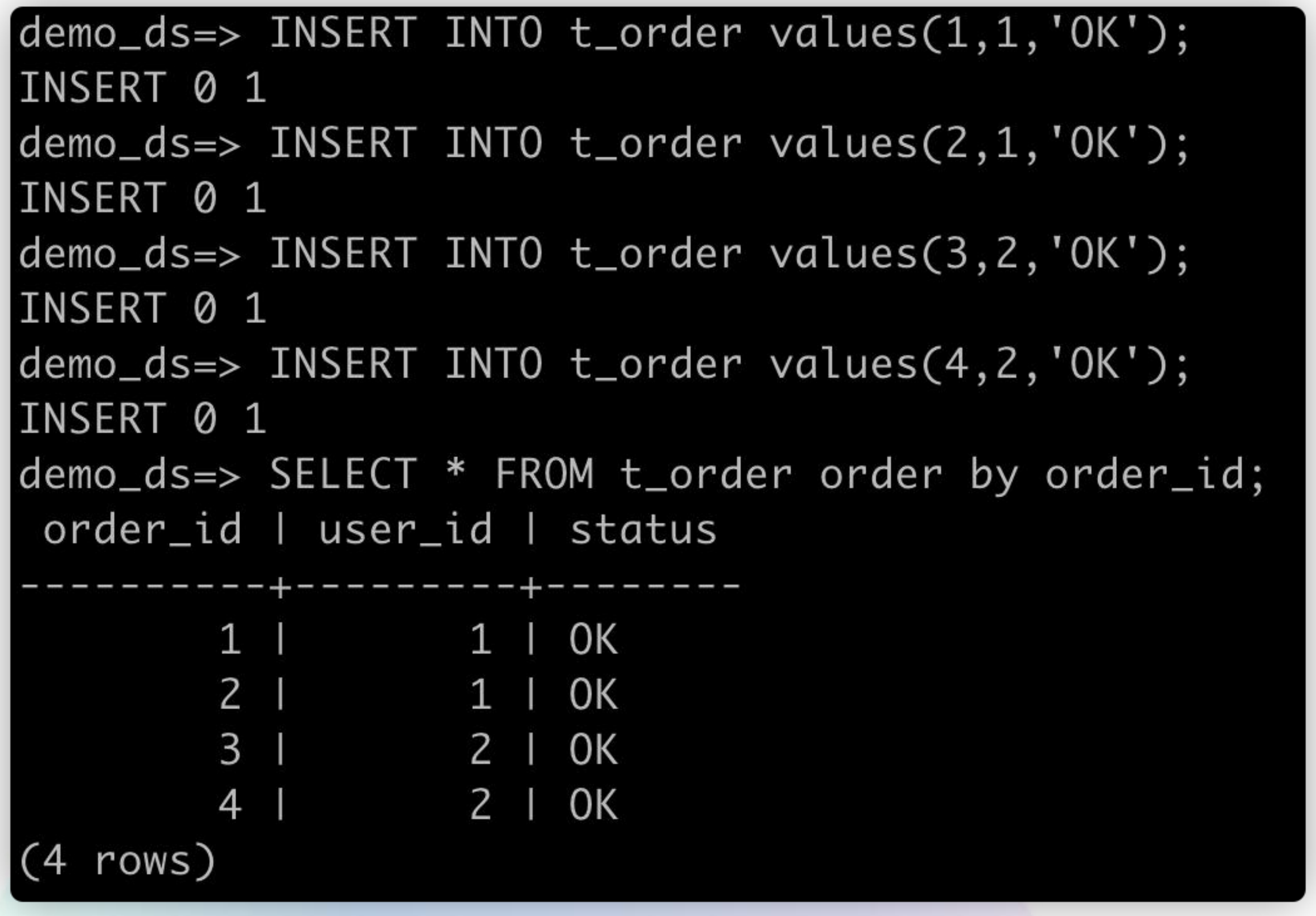


```
demo_ds=> CREATE SHARDING ALGORITHM table_inline (
demo_ds(> TYPE(NAME=inline,PROPERTIES("algorithm-expression"="t_order_${user_id % 2}"))
demo_ds(> );

demo_ds=> CREATE SHARDING TABLE RULE t_order (
demo_ds(> DATANODES("demo_ds_${0..1}.t_order_${0..1}"),
demo_ds(> DATABASE_STRATEGY(TYPE=standard,SHARDING_COLUMN=order_id,SHARDING_ALGORITHM(TYPE(NAME=inline,PROPERTIES("algorithm-expression"="demo_ds_${order_id % 2}")))),
demo_ds(> TABLE_STRATEGY(TYPE=standard,SHARDING_COLUMN=user_id,SHARDING_ALGORITHM=table_inline)
demo_ds(> );
CREATE
```



```
demo_ds=> CREATE TABLE t_order (
demo_ds(>   order_id bigint NOT NULL,
demo_ds(>   user_id int NOT NULL,
demo_ds(>   status varchar(50) DEFAULT NULL,
demo_ds(>   PRIMARY KEY (order_id)
demo_ds(> );
CREATE TABLE
```

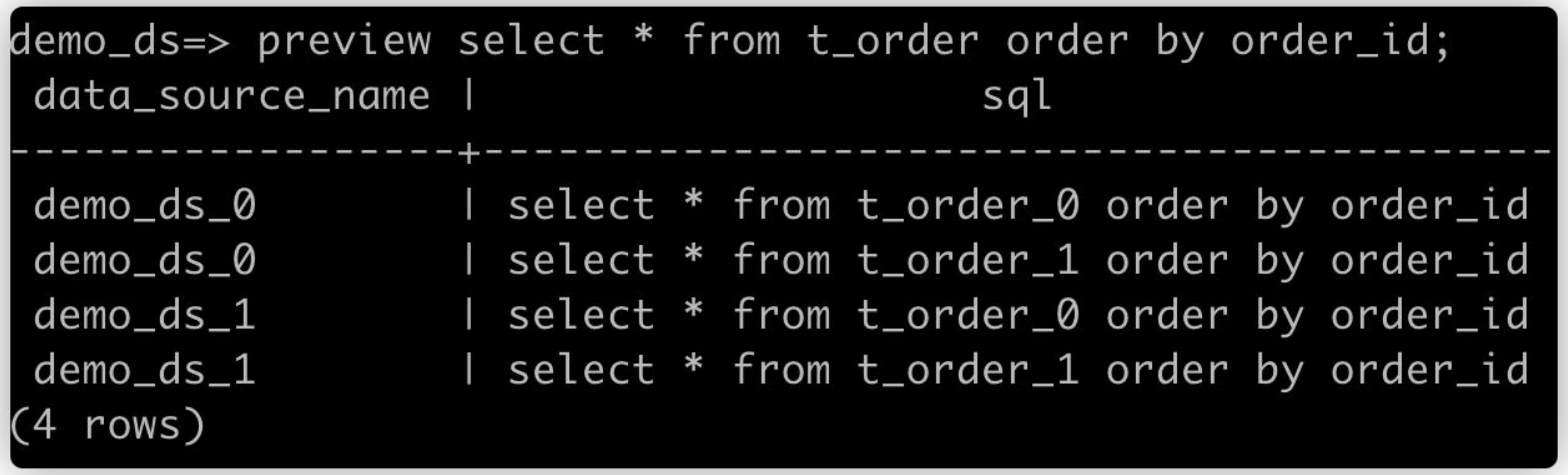
4. Insert Data



```
demo_ds=> INSERT INTO t_order values(1,1,'OK');
INSERT 0 1
demo_ds=> INSERT INTO t_order values(2,1,'OK');
INSERT 0 1
demo_ds=> INSERT INTO t_order values(3,2,'OK');
INSERT 0 1
demo_ds=> INSERT INTO t_order values(4,2,'OK');
INSERT 0 1
demo_ds=> SELECT * FROM t_order order by order_id;
 order_id | user_id | status
----------+---------+--------
        1 |       1 | OK
        2 |       1 | OK
        3 |       2 | OK
        4 |       2 | OK
(4 rows)
```

Distributed Postgres Database Solution

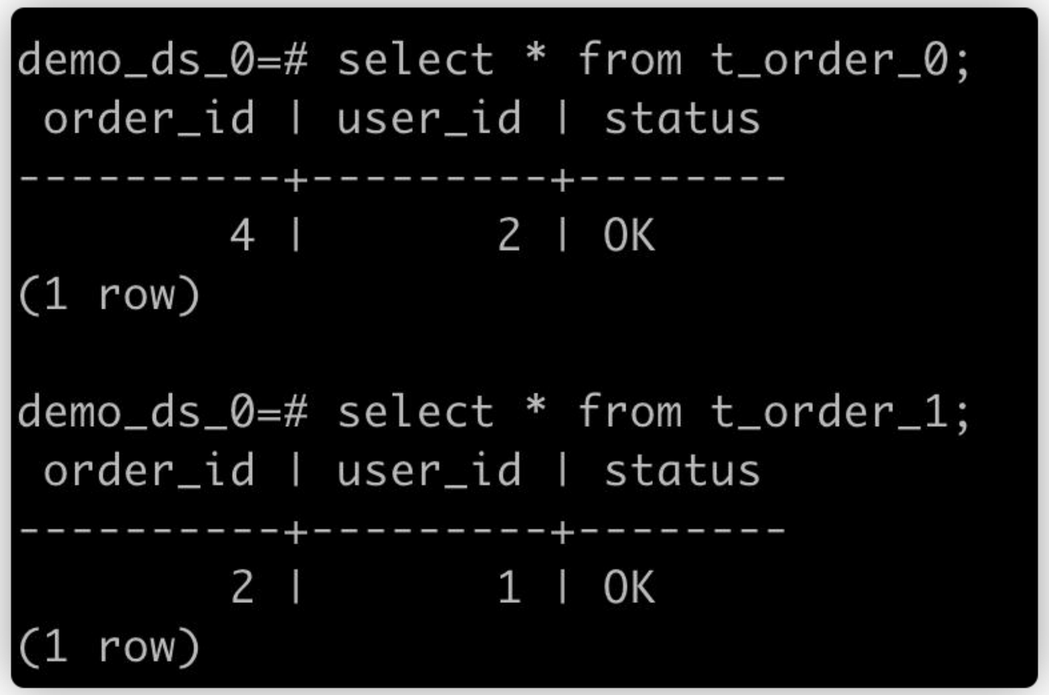
5. Preview SQL



```
demo_ds=> preview select * from t_order order by order_id;
 data_source_name |                    sql
------------------+----------------------------------------
 demo_ds_0        | select * from t_order_0 order by order_id
 demo_ds_0        | select * from t_order_1 order by order_id
 demo_ds_1        | select * from t_order_0 order by order_id
 demo_ds_1        | select * from t_order_1 order by order_id
(4 rows)
```

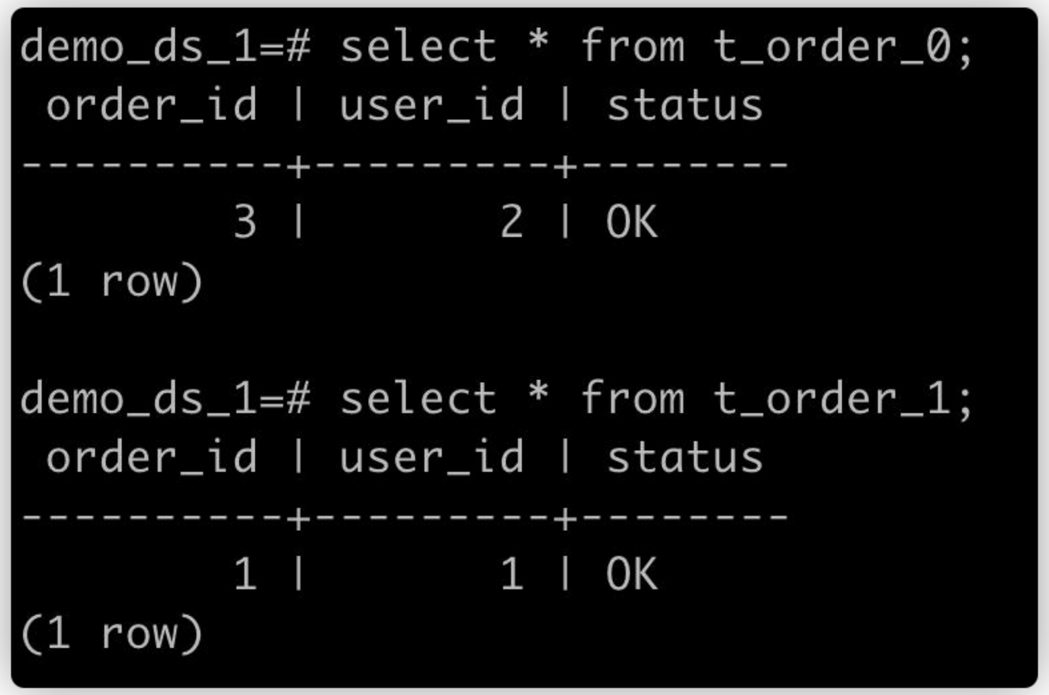
6. The Data in Actual PostgreSQL

demo_ds_0



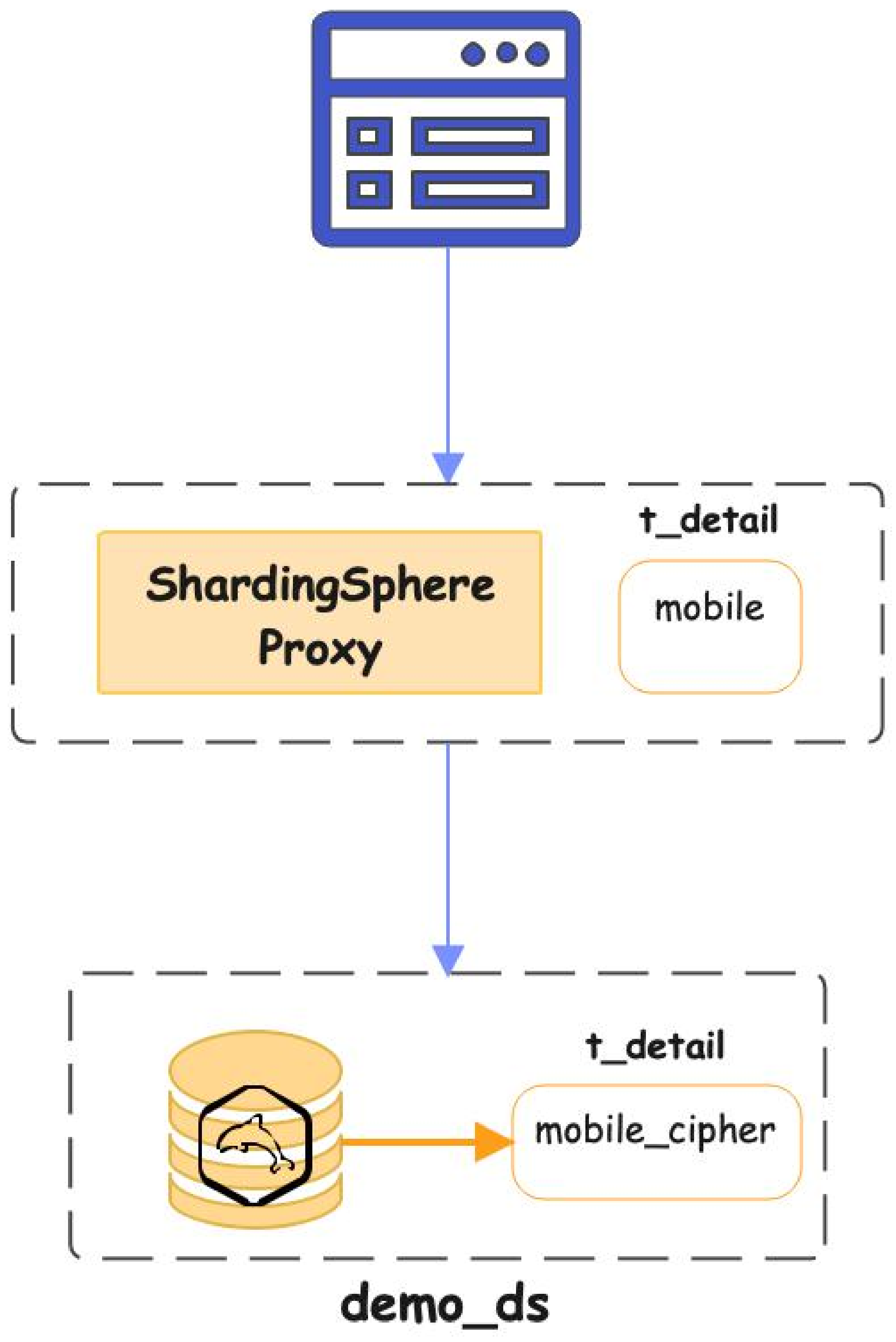
```
demo_ds_0=# select * from t_order_0;
 order_id | user_id | status
----------+---------+--------
        4 |       2 | OK
(1 row)

demo_ds_0=# select * from t_order_1;
 order_id | user_id | status
----------+---------+--------
        2 |       1 | OK
(1 row)
```

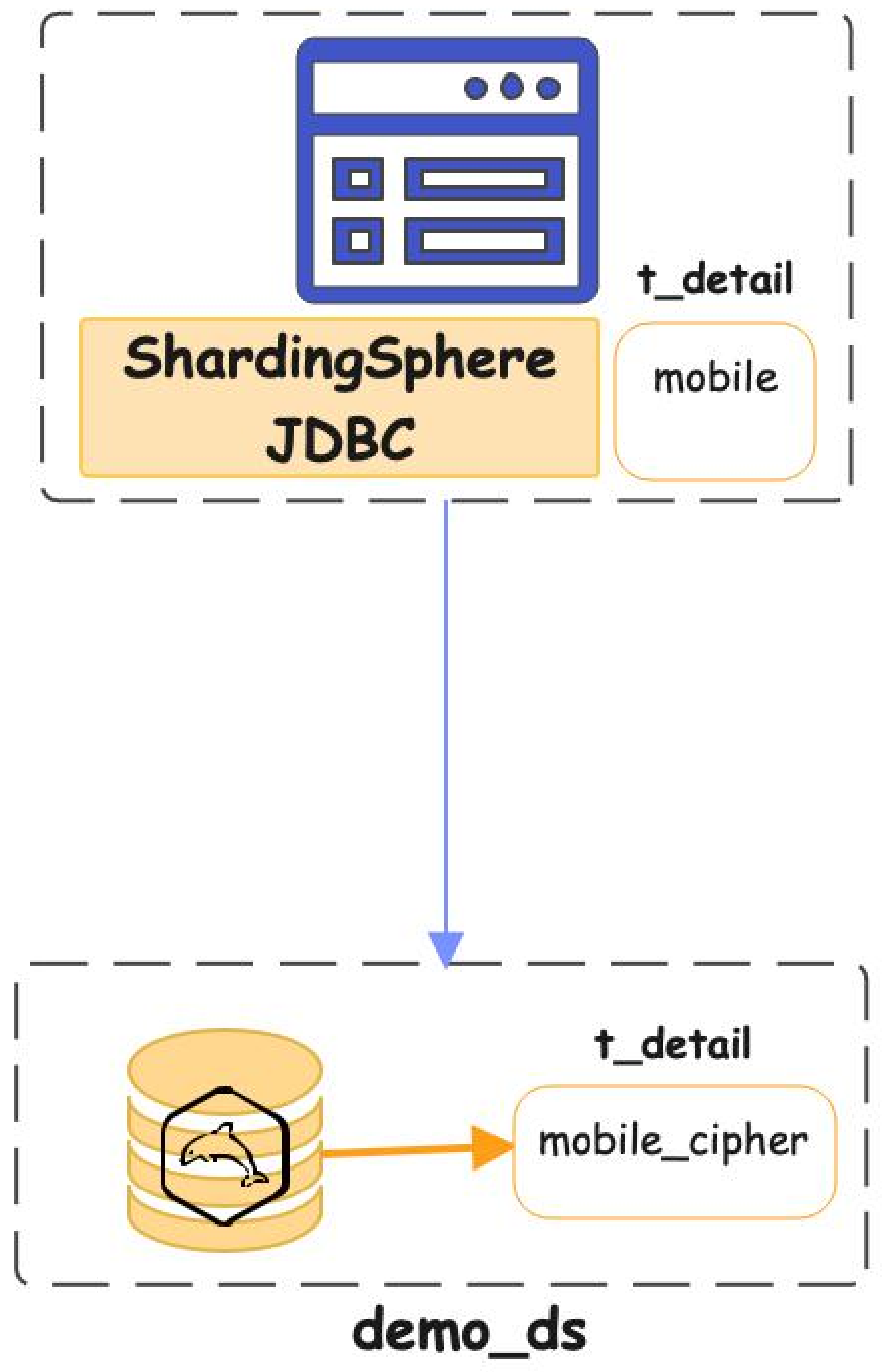
demo_ds_1



```
demo_ds_1=# select * from t_order_0;
 order_id | user_id | status
----------+---------+--------
        3 |       2 | OK
(1 row)

demo_ds_1=# select * from t_order_1;
 order_id | user_id | status
----------+---------+--------
        1 |       1 | OK
(1 row)
```

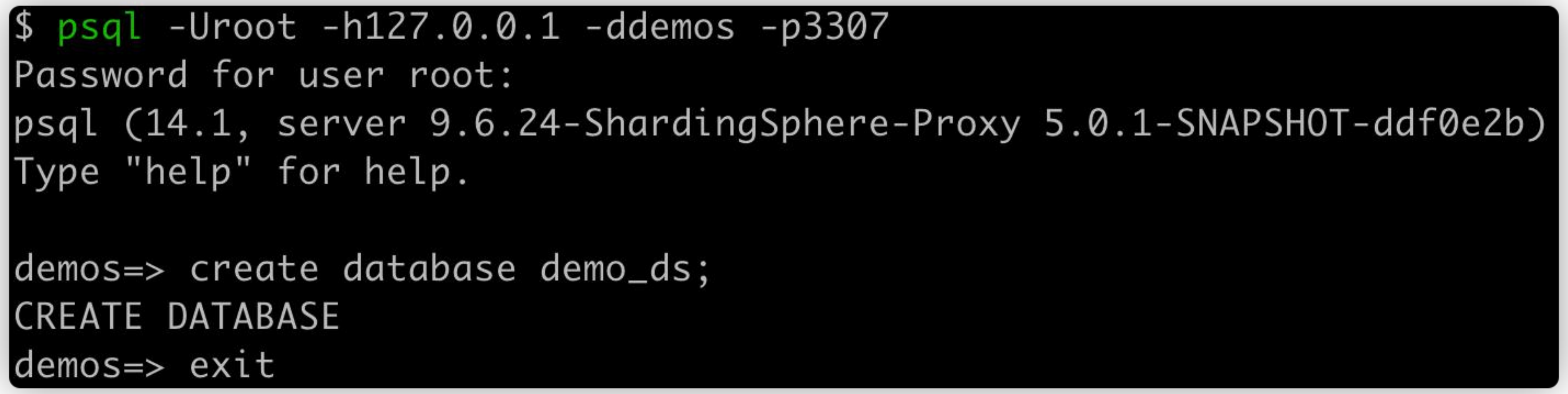
Secure Postgres Database Solution





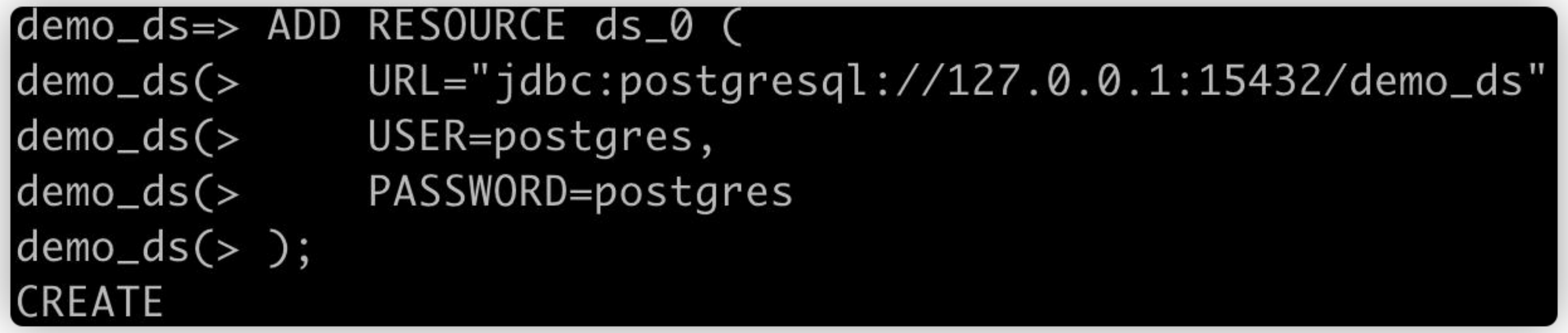
Secure Postgres Database Solution

1. Create Database



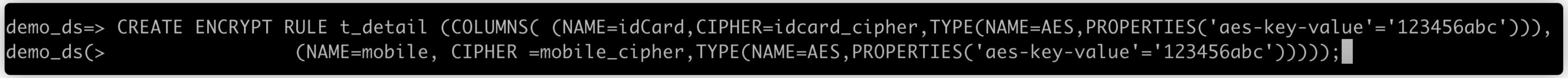
```
$ psql -Uroot -h127.0.0.1 -ddemos -p3307
Password for user root:
psql (14.1, server 9.6.24-ShardingSphere-Proxy 5.0.1-SNAPSHOT-ddf0e2b)
Type "help" for help.

demos=> create database demo_ds;
CREATE DATABASE
demos=> exit
```

2. Add Data Source



```
demo_ds=> ADD RESOURCE ds_0 (
demo_ds(>     URL="jdbc:postgresql://127.0.0.1:15432/demo_ds"
demo_ds(>     USER=postgres,
demo_ds(>     PASSWORD=postgres
demo_ds(> );
CREATE
```

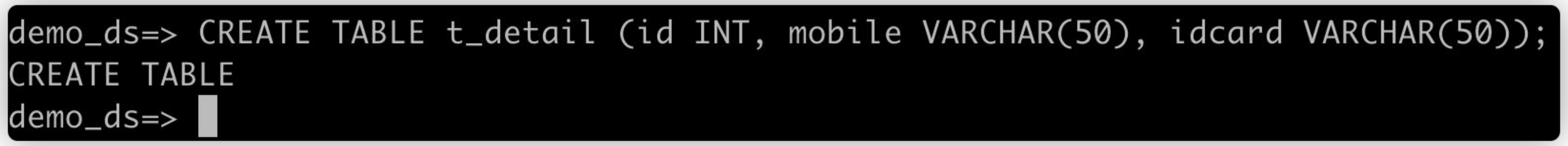
3. Create Rule



```
demo_ds=> CREATE ENCRYPT RULE t_detail (COLUMNS( (NAME=idCard,CIPHER=idcard_cipher,TYPE(NAME=AES,PROPERTIES('aes-key-value'='123456abc'))),
demo_ds(>                  (NAME=mobile, CIPHER =mobile_cipher,TYPE(NAME=AES,PROPERTIES('aes-key-value'='123456abc')))));
```

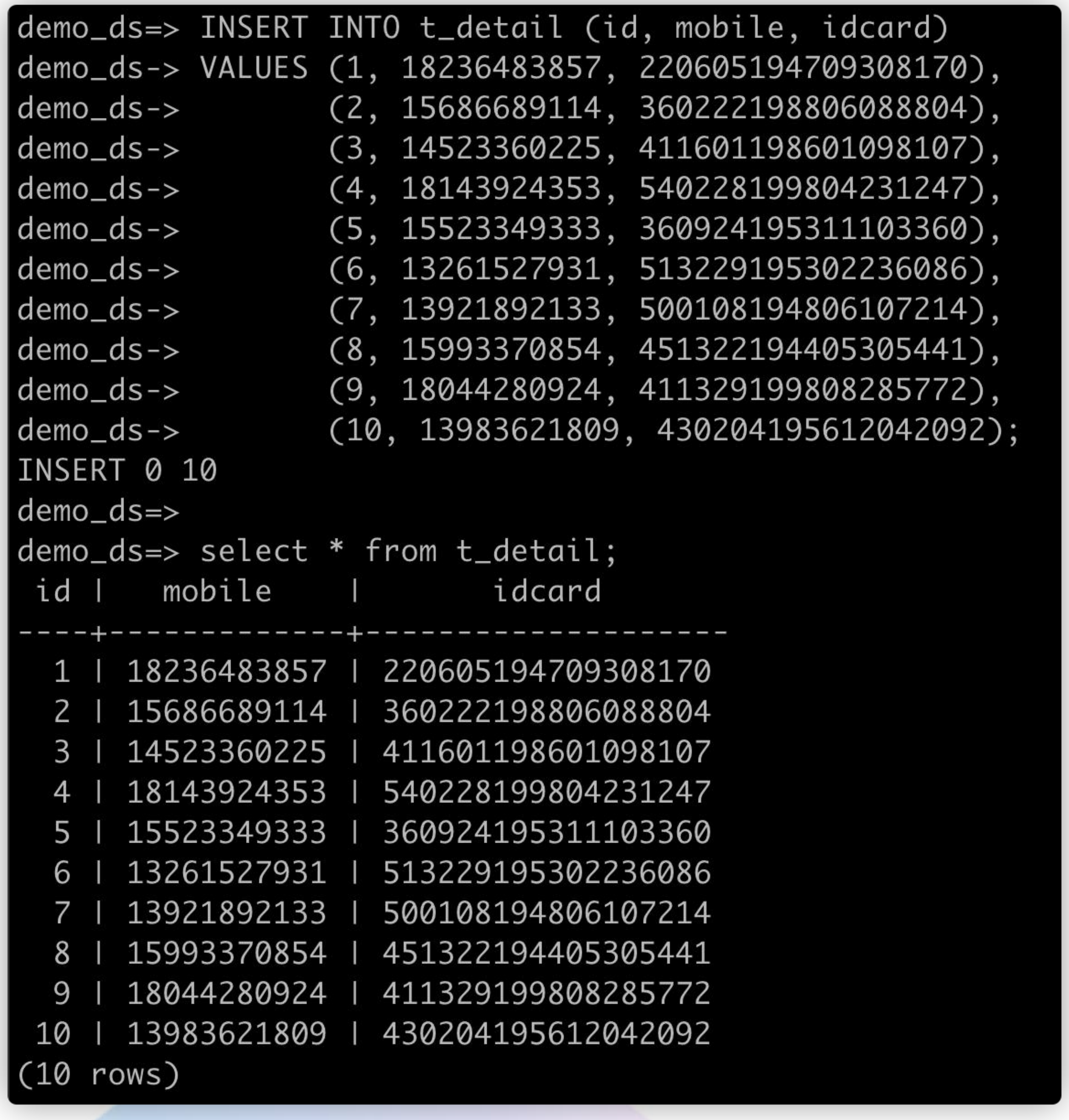
Secure Postgres Database Solution

4. Create Table



```
demo_ds=> CREATE TABLE t_detail (id INT, mobile VARCHAR(50), idcard VARCHAR(50));
CREATE TABLE
demo_ds=>
```

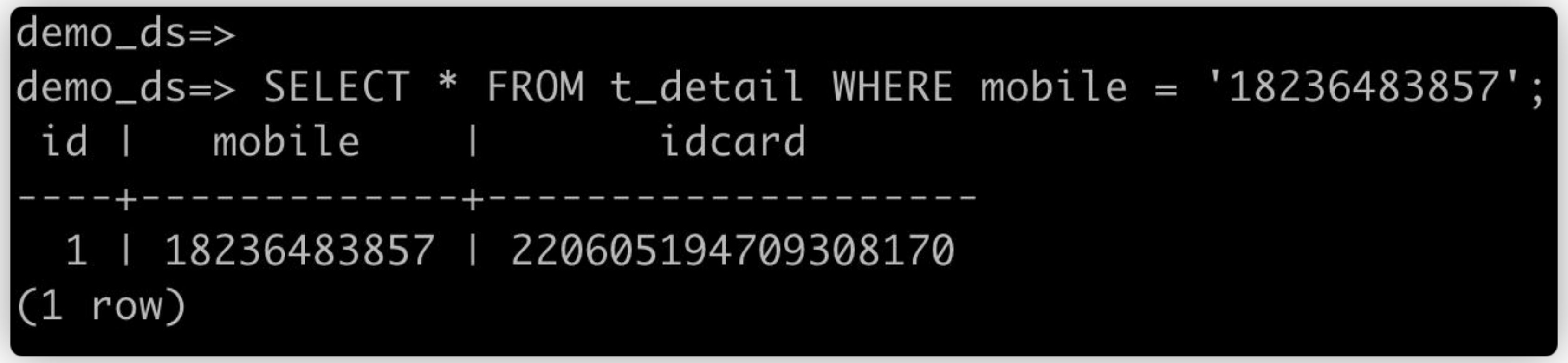
5. Insert Data



```
demo_ds=> INSERT INTO t_detail (id, mobile, idcard)
demo_ds-> VALUES (1, 18236483857, 220605194709308170),
demo_ds->        (2, 15686689114, 360222198806088804),
demo_ds->        (3, 14523360225, 411601198601098107),
demo_ds->        (4, 18143924353, 540228199804231247),
demo_ds->        (5, 15523349333, 360924195311103360),
demo_ds->        (6, 13261527931, 513229195302236086),
demo_ds->        (7, 13921892133, 500108194806107214),
demo_ds->        (8, 15993370854, 451322194405305441),
demo_ds->        (9, 18044280924, 411329199808285772),
demo_ds->        (10, 13983621809, 430204195612042092);
INSERT 0 10
demo_ds=>
demo_ds=> select * from t_detail;
 id |   mobile    |       idcard
----+-------------+--------------------
  1 | 18236483857 | 220605194709308170
  2 | 15686689114 | 360222198806088804
  3 | 14523360225 | 411601198601098107
  4 | 18143924353 | 540228199804231247
  5 | 15523349333 | 360924195311103360
  6 | 13261527931 | 513229195302236086
  7 | 13921892133 | 500108194806107214
  8 | 15993370854 | 451322194405305441
  9 | 18044280924 | 411329199808285772
 10 | 13983621809 | 430204195612042092
(10 rows)
```

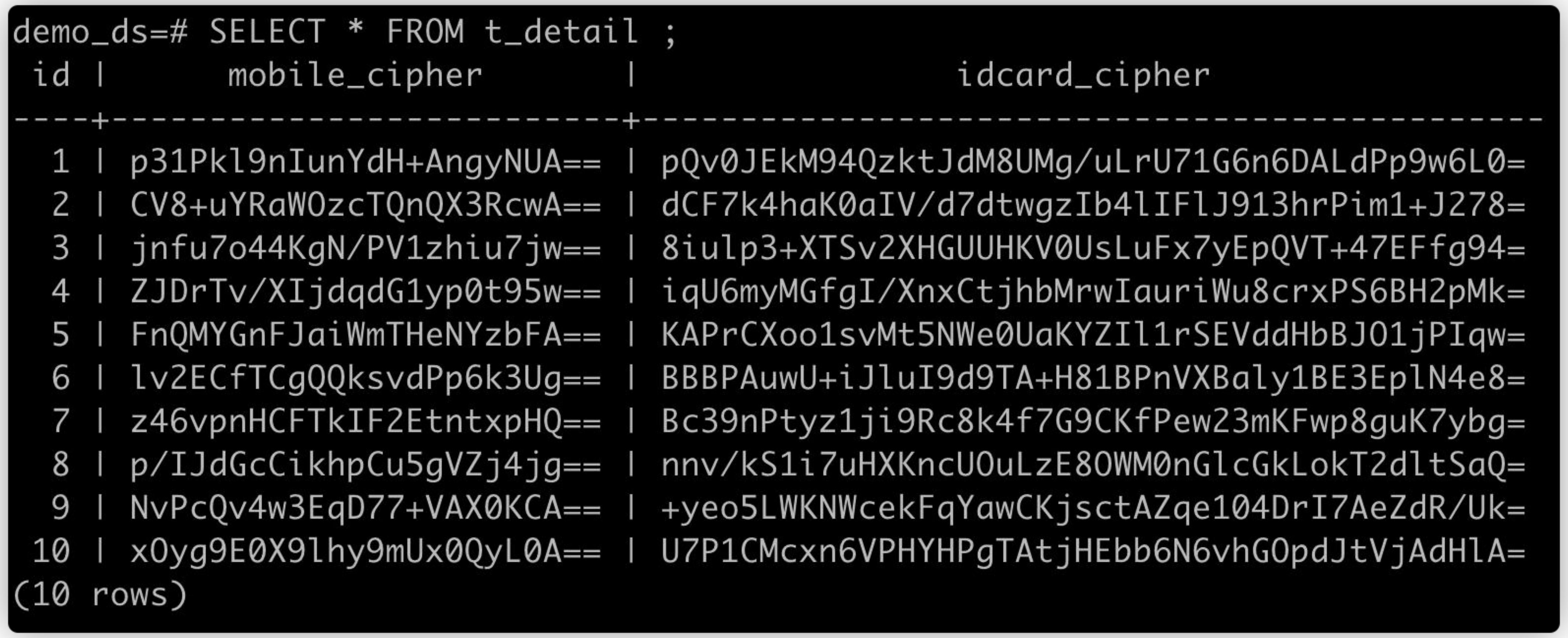
Secure Postgres Database Solution

6. Check Data from Proxy



```
demo_ds=>
demo_ds=> SELECT * FROM t_detail WHERE mobile = '18236483857';
 id |    mobile     |       idcard
----+-------------+-------------------
  1 | 18236483857 | 220605194709308170
(1 row)
```

7. The Data in Actual PostgreSQL



```
demo_ds=# SELECT * FROM t_detail ;
 id |      mobile_cipher        |                idcard_cipher
----+-------------------------+----------------------------------------------
  1 | p31Pkl9nIunYdH+AngyNUA== | pQv0JEkM94QzktJdM8UMg/uLrU71G6n6DALdPp9w6L0=
  2 | CV8+uYRaWOzcTQnQX3RcwA== | dCF7k4haK0aIV/d7dtwgzIb4lIFlJ913hrPim1+J278=
  3 | jnfu7o44KgN/PV1zhiu7jw== | 8iulp3+XTSv2XHGUUHKV0UsLuFx7yEpQVT+47EFfg94=
  4 | ZJDrTv/XIjdqdG1yp0t95w== | iqU6myMGfgI/XnxCtjhbMrwIauriWu8crxPS6BH2pMk=
  5 | FnQMYGnFJaiWmTHeNYzbFA== | KAPrCXoo1svMt5NWe0UaKYZIl1rSEVddHbBJO1jPIqw=
  6 | lv2ECfTCgQQksvdPp6k3Ug== | BBBPAuwU+iJluI9d9TA+H81BPnVXBaly1BE3EplN4e8=
  7 | z46vpnHCFTkIF2EtntxpHQ== | Bc39nPtyz1ji9Rc8k4f7G9CKfPew23mKFwp8guK7ybg=
  8 | p/IJdGcCikhpCu5gVZj4jg== | nnv/kS1i7uHXKncUOuLzE8OWM0nGlcGkLokT2dltSaQ=
  9 | NvPcQv4w3EqD77+VAX0KCA== | +yeo5LWKNWcekFqYawCKjsctAZqe104DrI7AeZdR/Uk=
 10 | xOyg9E0X9lhy9mUx0QyL0A== | U7P1CMcxn6VPHYHPgTAtjHEbb6N6vhGOpdJtVjAdHlA=
(10 rows)
```

SphereEx

Thank You!

Any questions?

Bio: https://tristazero.github.io
LinkedIn: https://www.linkedin.com/in/panjuan
GitHub: https://github.com/tristaZero
Twitter: @tristaZero
Project Twitter: @ShardingSphere