Foreign Data Wrapper

leichter Zugriff
auf externe Datenquellen

Thomas Koch

Data Architect

Deutsche Bahn Connect GmbH

Deutsche Bahn Connect GmbH

Organisation

Die Produkte von DB Connect sind in vier Business Lines organisiert

Fleet Mobility

Flottenmanagement
DB Firmenrad

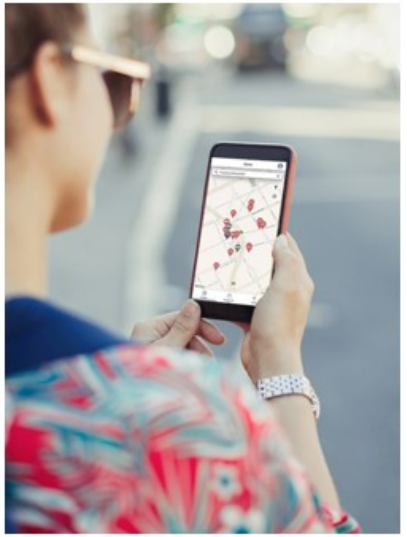
Category Management

Procurement
Remarketing

Shared Mobility

Call a Bike
Flinkster

Connected Mobility



Bonvoyo Mobilitätsbudget
Curbside Management

- ca. 24.000 Fahrzeuge
- ca. 59.000 bestellte Firmenräder

- Über 4500 Fahrzeuge im Flinkster Netzwerk in über 400 Städten
- Über 13000 Räder in 80 Städte & Kommunen

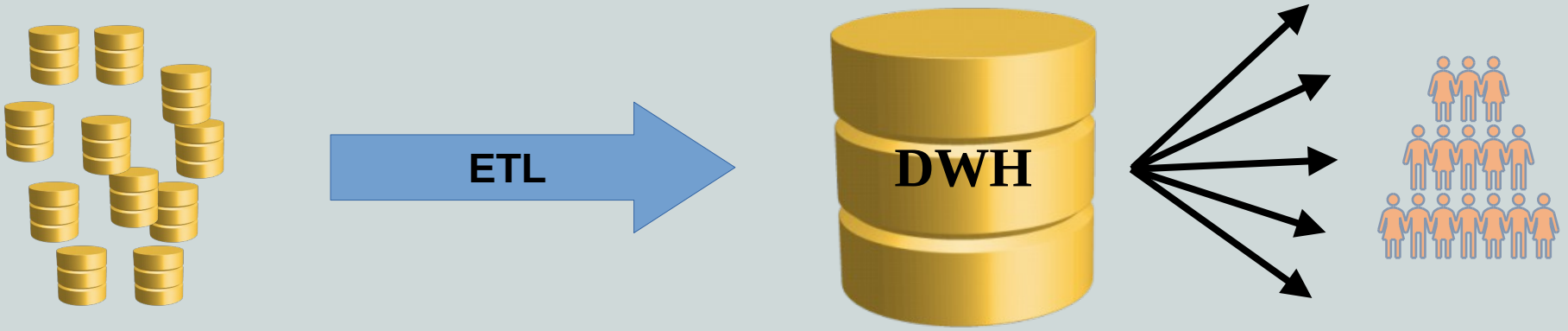
Architektonische Herausforderung

Mehrere Produkte

→ viele Software-Komponenten
→ viele Microservices
→ viele Datenbanken
→ viele Technologien

Viele Stakeholder

- Berichte für Kunden
- Berichte ans Management
- Daten ans Controlling
- ...



Probleme

- ETL sehr aufwändig
- Wenig automatisiert
- Monitoring kaum umgesetzt
- Tests nicht vorhanden

Probleme

- DWH benötigt viel Speicherplatz
- Nur das enthalten was angefordert
- ...

Data Lake als Alternative?

Mehrere Produkte

→ viele Software-Komponenten
→ viele Microservices
→ viele Datenbanken
→ viele Technologien

Viele Stakeholder

- Berichte für Kunden
- Berichte ans Management
- Daten ans Controlling
- ...

Probleme

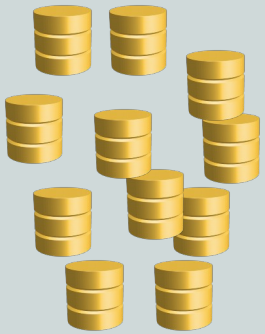
- Verwaltbar?
- Data Catalog?
- Weg zum Datensumpf verhindern?
- Benötigt auch viel Speicherplatz

90% der Data Lakes scheitern

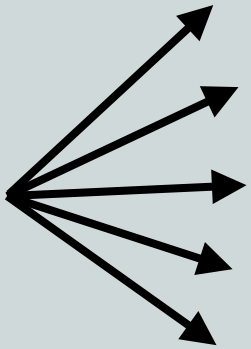
(Quelle: Gartner)

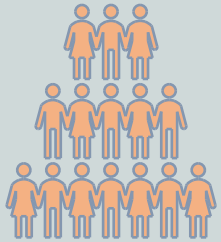
Suche nach anderer Alternative

Viele Datenbanken



Viele Stakeholder





Anforderungen

- Aktueller und rollenspezifischer Datenzugriff
- Unternehmensdatenmodell
- Mit Speicherplatz sparsam sein
- Soviel wie möglich automatisieren
- Testbar und monitoring-fähig
- Betreuung durch wenige Personen
- Dokumentiert

Zugriff auf andere Datenquellen?

Dblink Extension

- Verbindung zu einer anderen **PostgreSQL Datenbank**
- Nur innerhalb einer Session gültig
- Verwendung von derzeit 19 bereitgestellten Funktionen
- Erstellen einen Cursor und laden die externen Daten in die lokale Datenbank (fetch)

Foreign Data Wrapper

- implementiert SQL/MED (kurz für "Management of External Data")
  → Seit 2003 in ISO 9075-9 enthalten
- Für jede Datenquelle eine separate Extension
  - in contrib enthalten oder als externe Quelle
  - Datenquelle kann jede Datenpersistenz sein
- Speichert die Verbindungsparameter

Auswahl an Foreign Data Wrappers

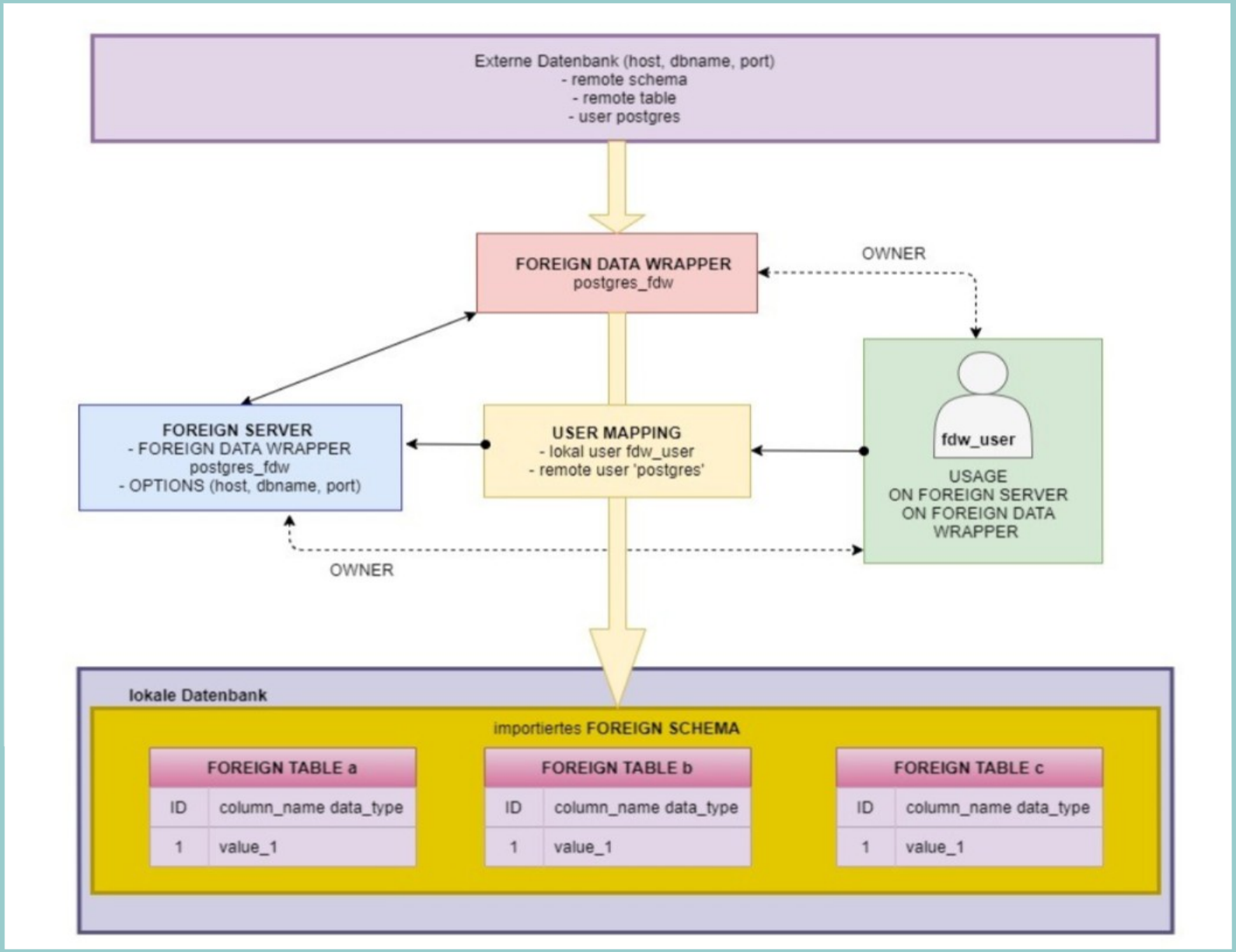
Data Source	Type	License	Code	Install
PostgreSQL	Native	PostgreSQL	git.postgresql.org	
Oracle	Native	PostgreSQL	github	
MySQL	Native		github	
Informix	Native	PostgreSQL	github	
DB2	Native		github	
Firebird	Native	PostgreSQL	github	
SQLite				
Sybase /				
MonetDB				

Data Source	Type	License	Code
CSV	Native	PostgreSQL	git.postgres
CSV	Multicorn	PostgreSQL	GitHub
CSV / Text Array	Native		GitHub
CSV / Fixed-length	Native		Git ub
CSV / gzipped	Multico n		Git ub
Compressed File	Native		GitHub
Document collecti	Native	PostgreSQL	GitHub
SON	Native	GPL3	G Hu
Multi-File	Multicorn	PostgreSQL	GitHub
Multi CDR	Native	PostgreSQL	GitHub
Parquet	Native	PostgreSQL	GitHub
pg_dump	Native	New BSD	GitHub
TAR Files	Native		GitHub
XML	Multicorn	PostgreSQL	GitHub
ZIP Files	Native		GitHub

Data Source	Type	License	Code	Install
BigTable or HBase	Native Rust Binding (RPGFFI)	MIT	Github	
Cassandra	Multicorn	MIT	Github	Rankactive
Cassandra2	Native	MIT	Github	
Cassandra	Multicorn	PostgreSQL	Github	
ClickHouse	Multicorn	BSD	Github	
ClickHouse	Native	Apache	Github	
ClickHouse	Native		Github	
CouchDB	Native	PostgreSQL	Github	PGXN
CouchDB	Native	PostgreSQL	Github	
GridDB	Native	PostgreSQL	Github	
nfluxDB	Native	PostgreSQL	Github	
Kafka	Native	PostgreSQL	GitHub	
Kyoto Tycoon	Native	MIT	Github	
Mong DB	Native	GPL3+	Github	PGXN
MongoDB	Multicorn	MIT	Github	
MongoDB	Multicorn		Github	
Neo4j	Multicorn	GPLv3	Github	
Neo4j	Native	?	Github	
Quasar	Native	Apache	Github	
Redis	Native	PostgreSQL	Github	
Redis	Native	BSD	Github	
RethinkDB	Multicorn	MIT	Github	
Riak	Multicorn	PostgreSQL	Github	

Und viele mehr
(ldap, ical, S3, Twitter...)

Foreign Data Wrapper – im Detail



Zeit für SQL

- Extension laden

```
CREATE EXTENSION postgres_fdw;
```

- Foreign Server erstellen
- *Options* abhängig von FDW

```
CREATE SERVER film_server
  FOREIGN DATA WRAPPER postgres_fdw
  OPTIONS (host 'foo', dbname 'foodb', port '5432');
```

Zeit für SQL

- USER Mapping anlegen (inkl. Passwort im Klartext)

```
CREATE USER MAPPING FOR bob SERVER film_server OPTIONS (user 'bob', password 'secret');
```

- Foreign-Tabellen anlegen mit exakter Spalten-Definition
- Foreign-Tabellen aus bestimmten Schema importieren

```
CREATE FOREIGN TABLE fdw.films (
    code        char(5) NOT NULL,
    title       varchar(40) NOT NULL,
    date_prod   date,
    kind        varchar(10)
)
SERVER film_server;

IMPORT FOREIGN SCHEMA foreign_films LIMIT TO (actors, directors)
    FROM SERVER film_server INTO fdw;
```

Generierung der FDW-Objekte

```
./generate_foreign_tables.sh -h <host> -p <port> -u <user> -d <database> -s <schema>
```

- Anfrage der Systemtabellen auf Fremd-Datenquelle
- Generierung der SQL-Anweisungen für Foreign-Table-Erstellung
- Ein Ergebnis aus der Bachelor Arbeit *„Konsolidierung einer heterogenen Datenbanklandschaft mit räumlichen Daten*" (2021, Herr Rastenes)
- Gilt vorerst nur für PostgreSQL-Quelldatenbanken

Testen mit pgTAP

- PgTAP ist ein unitTest-Framework
- Test von der FDW-Struktur
- Test der Lauffähigkeit
- Einbindung in Pipelines und Monitoringsystem
- Generierung der Tests (aus den Systemtabellen) möglich

```
SELECT has_extension('postgres_fdw');


SELECT fdw_privs_are(
  'postgres_fdw', 'bob', ARRAY['USAGE'],
  'Bob granted USAGE on fdw "postgres_fdw"'
);


SELECT server_privs_are(
  'film_server', 'bob', ARRAY['USAGE'],
  'Bob granted USAGE on "film_server"'
);
```

```
SELECT foreign_tables_are(
  'fdw',
  ARRAY['films', 'actors', 'directors']
);

SELECT foreign_table_owner_is(
    'fdw', 'films', 'bob',
    'films should be owned by bob'
);

SELECT performs_ok(
  $$
    SELECT code, title, date_prod, kind
     FROM fdw.films LIMIT 1
  $$,
  1000
);
```

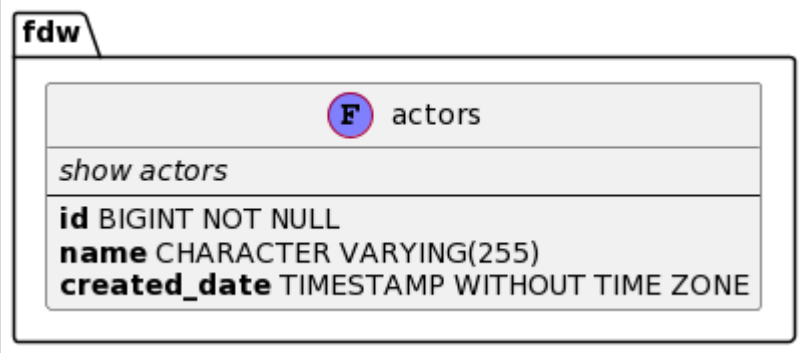
Quelle: https://pgtap.org/documentation.html

Generierung mit PlantUML

- Diagramme werden durch textuelle Notation beschrieben und als PNG- oder SVG-Grafik exportiert
- Generierung des Datenmodells (aus Systemtabellen möglich) für Dokumentation

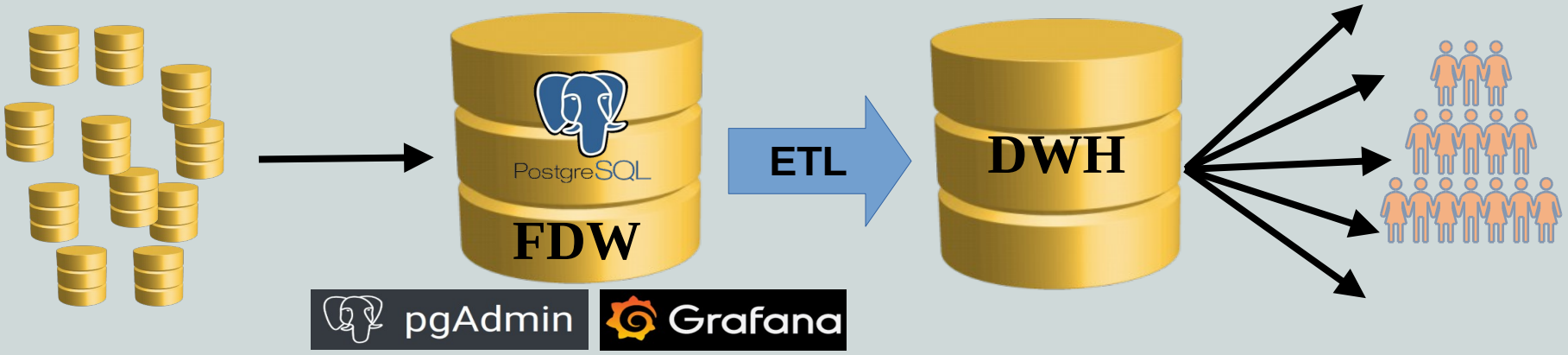
```
@startuml
!define FOREIGN(x) class x << (F,#8080FF) >>

FOREIGN(fdw.actors) {
<i>show actors</i>
--
<b>id</b> BIGINT NOT NULL
<b>name</b> CHARACTER VARYING(255)
<b>created_date</b> TIMESTAMP WITHOUT TIME ZONE
}
@enduml
```



Rückblick – wurde Aufgabe erfüllt?

Viele Datenbanken **Viele Stakeholder**



Anforderungen

- Aktueller und rollenspezifischer Datenzugriff ✓
- Unternehmensdatenmodell ✓
- Mit Speicherplatz sparsam sein ✓
- Soviel wie möglich automatisieren ✓
- Testbar und monitoring-fähig ✓
- Betreuung durch wenige Personen ✓
- Dokumentiert ✓

Konsequenzen

- Langsamere Abfragen
  - durch Netzwerk
  - durch Datenquellen-übergreifende Joins
- Jeder neuer FDW eine Herausforderung (Installation, Funktionsweise etc.)
- DWH trotzdem seine Daseinsberechtigung
- Stakeholder wechseln von DWH hin zur FDW-Datenquelle (als UI Grafana, Dashboards von allen gebaut)

Foreign Data Wrapper