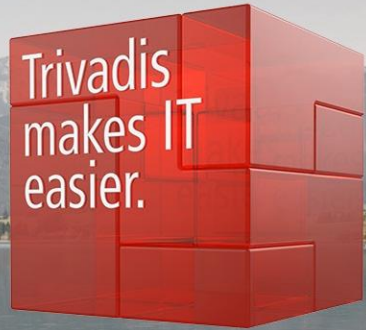


Oracle und Postgres

Zusammenspiel und Backup/Recovery

AOUG Breakfast - 22.02.2019

Roland Stirnimann



@rstirnimann_ch

BASEL ■ BERN ■ BRUGG ■ BUCHAREST ■ DÜSSELDORF ■ FRANKFURT A. M. ■ FREIBURG I. BR. ■ GENEVA
HAMBURG ■ COPENHAGEN ■ LAUSANNE ■ MANNHEIM ■ MUNICH ■ STUTTGART ■ VIENNA ■ ZURICH

trivadis
makes IT easier. ■ ■ ■

■ Agenda

1. Postgres mit Oracle verbinden

- Anwendungsbeispiele
- Tools
- Oracle Foreign Data Wrapper

2. Backup und Restore

- Backup Funktionalität in Postgres
- Vergleich zu Oracle
- 3rd Party Backup-Tools

Postgres mit Oracle verbinden

■ Anwendungsbeispiele

RDBMS Wechsel, Migration


- Migration von Oracle nach Postgres
 - Struktur
 - Daten
- Einmaliger Vorgang (Projekt)
- Applikation nicht vergessen!
 - Konnektivität (Treiber)
 - Persistenz Layer

Permanenter Datenaustausch



- Postgres liest oder schreibt Daten von/nach Oracle
- Permanent bestehende Verbindung
- Postgres als Daten-Proxy, z.B. zum Visualisieren von Daten mit Grafana (Daten-Pull)
- Dedizierte DML Operationen in zentrale Oracle Datenbank

■ Tools

RDBMS Wechsel, Migration

- EDB Migration Toolkit 
- orafce - Oracle Funktionen in Postgres
- ora2pg - Perl Modul für Export von Daten und Struktur
- ora_migrator - verwendet Oracle FDW

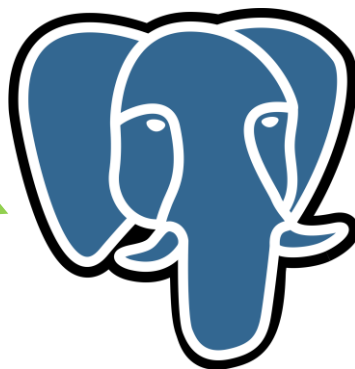
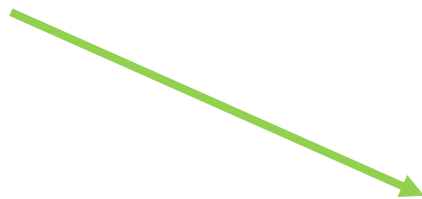
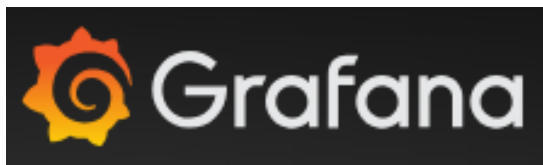
Permanenter Datenaustausch

- Datenbank Link in EDB Postgres Advanced Server mit Oracle Kompatibilitätsmodus 
- Oracle Foreign Data Wrapper (FDW) für Postgres (OCI)
- Golden Gate (ODBC) – nur Delivery 

■ Oracle Foreign Data Wrapper

- Postgres Server Extension
- SQL/MED basierender Foreign Data Wrapper für Oracle
- Transparenter Zugriff auf Oracle Tabellen/Views als wären sie lokal
- Entwickler Laurenz Albe - https://github.com/laurenz/oracle_fdw
- Open Source
- Linux and Windows Supported

PoC Setup mit Grafana



■ Oracle Client

- Oracle Instant Client mit SDK Package als OS User postgres

```
cd /opt/oracle/product
unzip instantclient-basic-linux.x64-18.3.0.0.0dbru.zip
unzip instantclient-sdk-linux.x64-18.3.0.0.0dbru.zip
unzip instantclient-sqlplus-linux.x64-18.3.0.0.0dbru.zip
```

- Erforderliche Variablen setzen (.bash_profile) und Verbindungstest

```
export ORACLE_HOME=/opt/oracle/product/instantclient_18_3
export LD_LIBRARY_PATH=${ORACLE_HOME}:${LD_LIBRARY_PATH}
export PATH=${PATH}:${ORACLE_HOME}

sqlplus orauser/pw@//ldb04:1521/REPO1.world
```


■ Installation oracle_fdw

- Postgres Development Package ist erforderlich für Header Dateien

```
yum install postgresql10-devel
```

- oracle_fdw Extension installieren und Postgres neu starten (Library laden!)

```
cd /tmp
tar -xvzf oracle_fdw-ORACLE_FDW_2_1_0.tar.gz
cd oracle_fdw-ORACLE_FDW_2_1_0

make
make install

pg_ctl restart
```

■ Postgres Server Extension erstellen

- Foreign Data Wrapper Extension bezieht sich auf ein Oracle Home

```
psql
```

```
create extension oracle_fdw;
```

```
select oracle_diag();
```

```
oracle_fdw 2.1.0, PostgreSQL 10.5, Oracle client 18.3.0.0.0,  
ORACLE_HOME=/opt/oracle/product/instantclient_18_3
```

```
\dx
```

List of installed extensions

Name	Version	Schema	Description
oracle_fdw	1.1	public	foreign data wrapper for Oracle access

■ Foreign Server und User Mapping

- Ein Foreign Server bezieht sich auf eine Oracle Instanz

```
create server oradb_repo1 foreign data wrapper oracle_fdw  
options (dbserver '//1db04:1521/REPO1.world' );
```

- Ein User Mapping bezieht sich auf einen Oracle Benutzer

```
create user mapping for postgres server oradb_repo1 options  
(user 'TVDCAPMAN', password 'abc');
```

TIPP: Gross-/Kleinschreibung beachten für Oracle Namen

Foreign Table

- Foreign Table entspricht einer Oracle Tabelle oder View
- Weitere Optionen (`options`) verfügbar, z.B. `readonly`

```
create foreign table DATABASE_STAT (  
STAT_ID      numeric OPTIONS (key 'true') not null,  
SNAP_ID      numeric OPTIONS (key 'true') not null,  
VALUE        numeric not null,  
RAC_FACTOR   numeric  
)  
server oradb_repo1 options (schema 'TVDCAPMAN', table  
'DATABASE_STAT');  
  
select count(*) from database_stat;
```

■ Postgres Dictionary Views und Berechtigungen

■ Dictionary Views in Postgres

- pg_foreign_data_wrapper - \dx und \dew
- pg_foreign_server - \des
- pg_foreign_table - \dE und \det
- pg_user_mapping und pg_user_mappings (View) - \deu

■ Erforderliche Privilegien in Oracle Schema

- CREATE SESSION
- SELECT Privileg auf v\$sql und v\$sql_plan für EXPLAIN VERBOSE
- Berechtigung auf gewünschte Applikationstabellen

■ Gut zu wissen...

- PL/SQL Tabellen Funktionen sind nicht unterstützt für Abfragen
- Postgres Tabelle kann mehr oder weniger Spalten besitzen
- Für UPDATE und DELETE muss die KEY Klausel definiert sein
- Datentypen müssen korrekt übersetzt werden, z.B.
 - CLOB zu TEXT
 - NUMBER zu NUMERIC
- Push Down von WHERE, ORDER BY und JOIN Operationen
- Statistiken auf Foreign Tables können erstellt werden mit ANALYZE
- IMPORT FOREIGN SCHEMA Funktion für Strukturübernahme

Backup und Restore

■ Backup und Restore in Postgres Übersicht

- Logische Sicherung (SQL Dump) – Data Pump in Oracle
 - Restore einzelner Objekte aber ohne Recovery
 - Vier Formate: plain (default), custom, directory und tar (`--format=p|c|d|t`)
- Physikalische Sicherung – RMAN in Oracle
 - Base Backup inkl. WAL Logs
 - Zwei Formate: plain (default) und tar (`--format p|t`)
 - Filesystem Backup mit low-level API
- Point-in-Time-Recovery
- Timelines – analog Inkarnationen in Oracle

■ Basis-Funktionalität in Postgres

■ Logische Sicherung

- pg_dump
- pg_dumpall (ganzer Cluster)

■ Logische Wiederherstellung

- psql (plain Format)
- pg_restore

■ Physikalische Sicherung

- pg_basebackup
- Low-level API (start/stop Backup)
- Betriebssystem-Befehle

■ Physikalische Wiederherstellung

- Betriebssystem-Befehle

■ Write-Ahead Logging - WAL

- Standardmässig unter \$PGDATA in `pg_wal` oder `pg_xlog` (Version < 10)
- Parameter `wal_level`, `archive_mode` und `archive_command` setzen (`postgresql.conf`)
- Standardgrösse 16MB pro WAL Log (änderbar via `initdb`)
- WAL Sequence Number analog Log Sequence in Oracle
- Jeder WAL Eintrag kriegt eine Log Sequence Number (LSN)
 - Byte Offset im WAL Log für den Eintrag
 - Erhöht sich mit jedem Eintrag
- Checkpoints schreiben dirty Blocks vom Buffer in die Datendateien
 - Log wird für Instanz-Recovery nicht mehr gebraucht (recycle oder löschen)
 - Gleiches Prinzip wie in Oracle

■ WAL Archivierung - Konfiguration

■ Log Switch erzwingen und WAL Logs mit Sequence Number

```
select pg_switch_wal();      -- < v10 pg_switch_xlog()
```

```
ls -lr /u01/pgdata2/pg_wal/  
0000000100000000100000002C
```

■ WAL Parameter setzen und Cluster neu starten

```
vi $PGDATA/postgresql.conf  
wal_level=replica  
archive_mode=on  
archive_command=<OS Command>  
  
pg_ctl restart -D $PGDATA
```

■ WAL Archivierung im Detail

- archive_command darf ein existierendes Log **nicht überschreiben (test)**
- Return Code muss 0 sein wenn copy erfolgreich war

```
archive_command = 'test ! -f /archbck/%f && cp %p /archbck/%f'
```

```
# test ! -f /archbck/0000001000000A9000000065 && cp  
pg_wal/0000001000000A9000000065 /archbck/0000001000000A9000000065
```

- WAL Dateien bleiben stehen, wenn archive_command fehlschlägt
- Kann zu Stillstand führen → Panic Shutdown aber kein Datenverlust

```
scp: /pgbackup/pgdata1/archived_wals/0000010000000000000002: No such file or directory  
2019-02-18 21:01:10 CET LOG: archive command failed with exit code 1
```

■ Tablespaces in Postgres

- Postgres hat zwei Tablespaces, die beim initialisieren des Clusters angelegt werden

```
postgres=# \db+
```

List of tablespaces

Name	Owner	Location	Access privileges	Options	Size	Description
pg_default	enterprisedb				4298 MB	
pg_global	enterprisedb				717 kB	

- Tablespaces können irgendwo liegen und brauchen somit ein Verzeichnis

```
mkdir /u01/tablespaces
```

```
chown enterprisedb:enterprisedb /u01/tablespaces
```

■ Tablespaces anlegen

■ Einen neuen Tablespace anlegen in Postgres

```
create tablespace data_app1 location '/u01/tablespaces' ;
```

```
ls -Gg $PGDATA/pg_tblspc
```

```
lrwxrwxrwx 1 16 Feb 19 22:38 24647 -> /u01/tablespaces
```

```
ls -Gg /u01/tablespaces
```

```
drwx----- 2 4096 Feb 19 22:38 PG_10_201707211
```

■ Eine Tabelle erstellen im neuen Tablespace

```
create table mytbl (a int) tablespace data_app1;
```

■ Tablespaces Backup - Plain Format

- Ein plain Backup muss `--tablespace-mapping` Option verwenden

```
pg_basebackup -D backup --format p
```

```
pg_basebackup: directory "/u01/tablespaces" exists but is not empty  
pg_basebackup: removing contents of data directory "backup/"
```

```
pg_basebackup -D backup --format p \  
  --tablespace-mapping /u01/tablespaces=$HOME/backup/tablespaces
```

- Der Link wird beim Backup auf die neue Lokation umgesetzt

```
ls -l backup/pg_tblspc backup/tablespaces
```

```
lrwxrwxrwx 1 37 Feb 21 11:05 24647 -> /home/enterprisedb/backup/tablespaces  
drwx----- 3 4096 Feb 21 11:05 PG_10_201707211
```

■ Tablespaces Backup - Tar Format

- Ein Tar Backup erstellt pro benutzerdefinierter Tablespace eine Tar-Datei

```
pg_basebackup -D backup --format t
```

- Diese Dateien liegen im Backup Verzeichnis

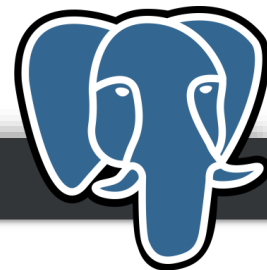
```
ls -Gg backup
```

```
-rw-rw-r-- 1          2560 Feb 21 12:47 24647.tar  
-rw-rw-r-- 1 4510689280 Feb 21 12:47 base.tar  
-rw----- 1    16779264 Feb 21 12:47 pg_wal.tar
```

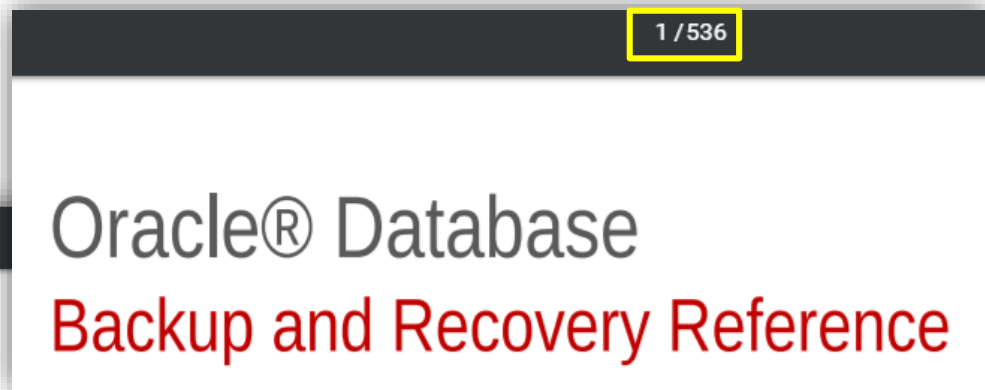
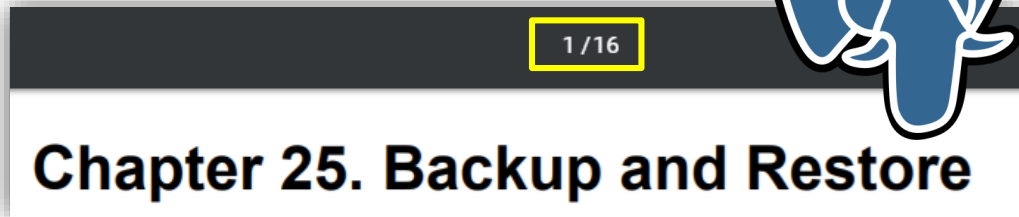

■ Tablespaces Restore

- Restore im **Plain** Format kann 1:1 verwendet werden
 - Zurückholen vom Backup Storage/Tape
 - Tablespace-Pfad ist bereits korrigiert während Backup Vorgang
- Restore im **Tar** Format muss an den richtigen Stellen entpackt werden
 - Entpacken nach \$PGDATA
 - Entpacken der Tablespaces Tar Dateien an entsprechende Lokation
 - Remapping während Restore ist möglich (siehe tablespace_map Datei in base.tar)

■ Vergleich Postgres und Oracle



16
versus
1471



■ Was hat Community Postgres nicht?

- RMAN ähnliches Tool
- Repository mit Metadaten (Catalog) und Backup Management
- Parallelisierung
- Flashback
- Dedizierte Restore und Recovery Kommandos
- Recovery einzelner Teile wie Datendatei/Tablespace oder Blöcke
- Datenbank-Duplizierung

ABER, brauchen wir diese erweiterten Funktionen wirklich immer!

Backup/Restore Basis-Funktionalität klappt in Postgres einwandfrei.

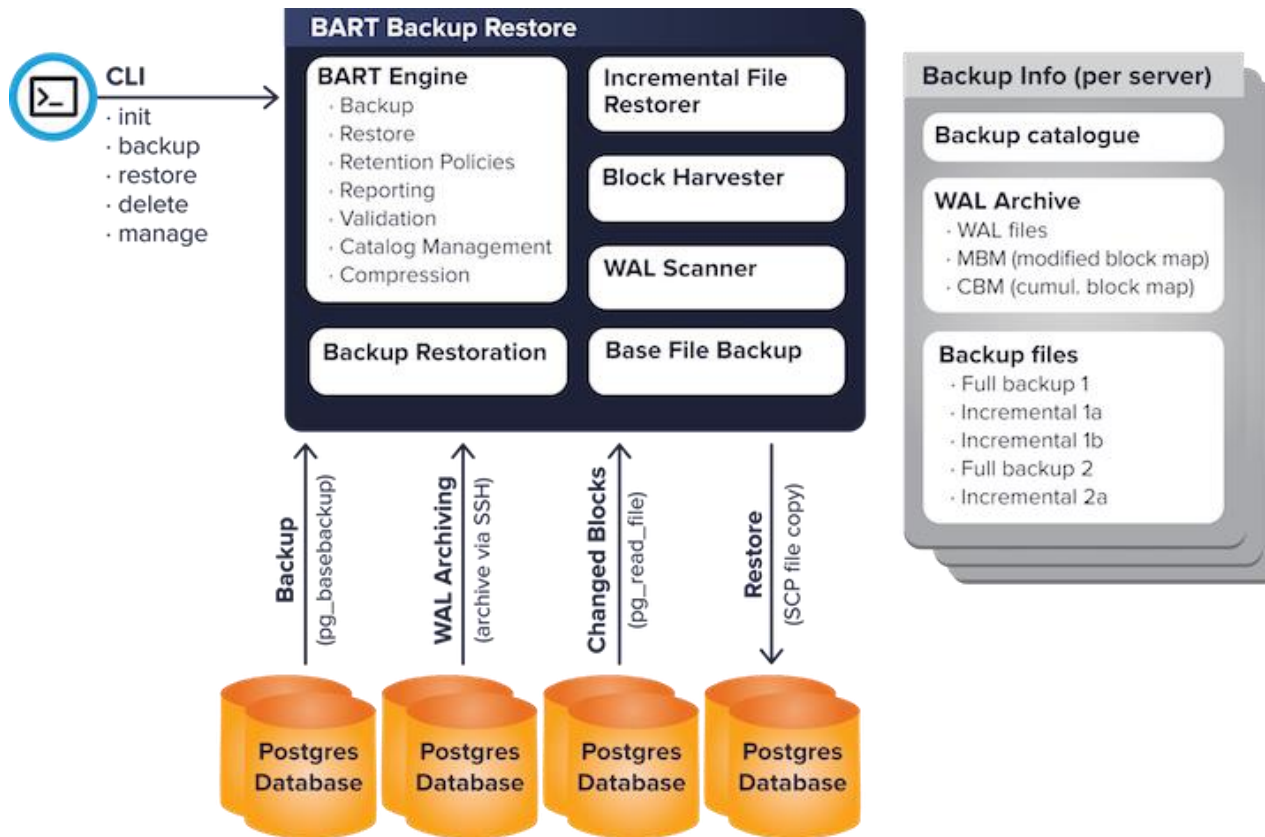
■ 3rd Party Backup-Tools

- **BART** von Enterprise DB (EDB Postgres Advance Server) - Kostenpflichtig
 - Einziges Tool mit inkrementellem Backup auf Block Level
 - Long-Term Backups mit Keep Option und Catalog
- **pgBackRest** von Crunchy Data – MIT License
 - Inkrementelles/differentielles Backup auf Basis von geänderten Dateien
 - Restore einzelner Datenbanken
- **Barman** von 2ndQuadrant - GNU General Public License 3
 - Inkrementelles Backup auf Basis von geänderten Dateien mit Hard Links
 - Backup Management und automatisches Löschen

■ EDB Backup und Recovery Tool - BART

- Vereinfachte Backup und Recovery Administration für EDB Postgres Advanced Server und Postgres (Community Version)
- Online full base backups and block-level incremental backups
- Backup and recovery management of the database servers
- Centralized catalog for backup metadata
- Retention policy support for defining and managing backups
- Verifies backup data with checksums
- Simplifies the point-in-time recovery process





■ BART Installation

- Download und Konfiguration vom EDB Repository (<http://yum.enterprisedb.com/>)

```
rpm -ivh /tmp/edb-repo-latest.noarch.rpm
# EDB Repository Login einfügen
vi /etc/yum.repos.d/edb.repo
yum install edb-bart
```

- BART Benutzer erstellen

```
mkdir /u02/pgbackup
useradd bartuser
passwd bartuser
chown bartuser:bartuser /u02/pgbackup/
```

■ BART Host-Konfiguration

- /usr/edb/bart/bin zu \$PATH hinzufügen
- \$LD_LIBRARY_PATH ergänzen mit einem \$PGHOME/lib Verzeichnis
- SSH Keys austauschen zwischen bartuser und postgres Benutzer
- **BART Host-Konfiguration** erstellen unter /usr/edb/bart/etc/bart.cfg

[BART]

```
bart_host= bartuser@192.168.2.100
backup_path = /u02/pgbackup
pg_basebackup_path = /u00/app/enterprisedb/product/10.5/bin/pg_basebackup
logfile = /home/bartuser/bart.log
scanner_logfile = /home/bartuser/bart_scanner.log
thread_count = 8
```


■ Instanz-Zugriff konfigurieren für BART

- .pgpass Datei erstellen auf **BART Host** für remote Instanz-Zugriff ohne Passwort

```
vi ~/.pgpass
192.168.2.100:5444:*:repuser:edb
192.168.2.100:5445:*:repuser:edb
chmod 600 ~/.pgpass
```

- Postgres Benutzer erstellen und pg_hba.conf anpassen auf Postgres Server

```
create user repuser superuser password 'edb';

vi /u01/pgdata1/pg_hba.conf
host      templat1      repuser      192.168.2.0/24      md5
host      replication   repuser      192.168.2.0/24      md5
```

■ BART Instanz-Konfiguration

- **Pro Instanz eine Konfiguration** erstellen auf BART Host in bart.cfg Datei

[LDB04_PGDATA2]

```
host = 192.168.2.100
port = 5445
user = repuser
backup_name = bck_pgdata2_%year-%month-%day_%hour-%minute-%second
retention_policy = 2 BACKUPS
xlog_method = fetch
copy_wals_during_restore = enabled
cluster_owner = enterprisedb
description = "Production Server - pgdata1 10.5"
allow_incremental_backups = enabled
remote_host = enterprisedb@192.168.2.100
tablespace_path=24647=/u01/tablespaces
```

■ BART initialisieren und validieren

- Bart Catalog initialisieren
- Host- und Instanz-Konfiguration validieren

```
bart init
bart CHECK-CONFIG
bart CHECK-CONFIG -s ldb04_pgdata2
bart show-servers
```

- Bart Scanner starten damit inkrementelle Backups erstellt werden können
 - Scant WAL Logs nach Änderungen und merkt sich die Blöcke
 - Analog Block Change Tracking in Oracle (modified block map Datei)

```
bart-scanner --daemon
```

■ BART Backup durchführen und anzeigen

■ Full und inkrementeller Backup erstellen

```
bart backup -s ldb04_pgdata2 -z --backup-name "initial_full"  
bart backup -s ldb04_pgdata2 -Fp --backup-name "inc_01" \  
  --parent "initial_full"
```

■ Backups anzeigen

```
bart show-backups  
bart show-backups -s ldb04_pgdata2 -i "inc_01" -t
```

■ Instanz-Restore mit BART

- Es wurde komplett alles gelöscht in \$PGDATA
- Optional kann mit `-g` Wiederherstellungszeitpunkt definiert werden

```
bart RESTORE -s ldb04_pgdata2 -i "initial_full" \  
-p /u01/pgdata2 -t 1
```

```
INFO: restoring backup 'initial_full' of server 'ldb04_pgdata2'  
INFO: base backup restored  
INFO: copying WAL file(s) to enterprisedb@192.168.2.100:/u01/pgdata2/archived_wals  
INFO: creating recovery.conf file  
INFO: archiving is disabled  
INFO: permissions set on $PGDATA  
INFO: restore completed successfully
```

```
pg_ctl start -D $PGDATA
```

■ Fazit

- Oracle bleibt in Sachen Features unerreicht
- BART bietet jedoch einiges an Komfort gegenüber pg_basebackup
 - Details zu Backups können angezeigt werden
 - Restore Befehl kopiert erforderliche Dateien zurück
 - tablespace_map und recovery.conf Dateien werden generiert
- Ähnliche Unterstützung bieten auch die beiden anderen 3rd Party Tools



PostgreSQL Foreign Data Wrapper for Oracle: https://github.com/laurenz/oracle_fdw

BART: <https://www.enterprisedb.com/products/edb-postgres-platform/edb-backup-and-recovery-tool>

Barman: <https://www.pgbarman.org/>

pgBackRest <https://pgbackrest.org/>

Q & A

Roland Stirnimann
Business Development Manager

Mathias Zarick
Principal Consultant

