
DATAGUARDS
Excellence in Data Technologies
PostgreSQL
für Oracle-Administratoren
ORACLE Gold Partner

Themen

- Geschichte und Einführung
- Gemeinsamkeiten und Unterschiede zu Oracle
- Die offene Architektur von PostgreSQL
- Einstieg für Oracle-DBAs
- Applikationen für PostgreSQL entwickeln
- Migration von Oracle und Mischbetrieb
- PostgreSQL in die IT-Landschaft einbinden

Geschichte und Einführung

- Ursprung: POSTGRES-Projekt UC Berkeley Version 1 1987
- Version 3 1993 Rule System
- Projekt mit der Version 4.2 1993 beendet (Supportanforderungen)
- 1995: Als Postgres95 ins Web gestellt
- Seit 1996: Ständige Weiterentwicklung als PostgreSQL Open Source
- Aktuell: Version 11.2

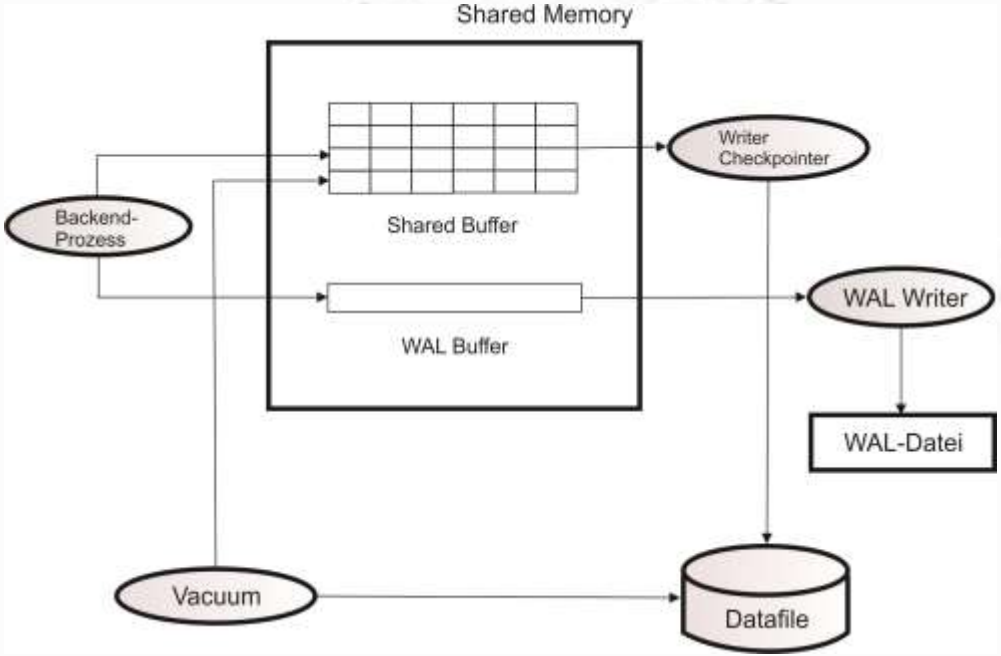
Gemeinsamkeiten mit Oracle

- Läuft auf allen populären Server-Plattformen
- Robuste Transaction Engine
- Designed als Client/Server-Architektur
- Unterstützt ANSI-SQL und Non-ANSI-SQL
- PL/pgSQL und PL/SQL
- Hoher Kompatibilitätsgrad mit Steigerungsmöglichkeiten
- Einsatzgebiete

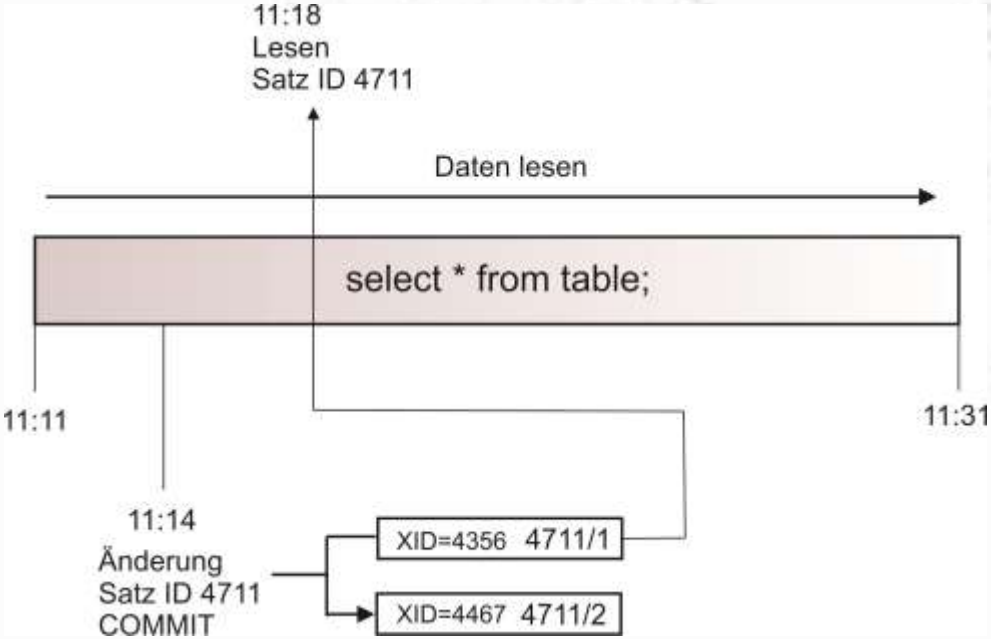
Einsatz von PostgreSQL

- Aus welchen Gründen würden Sie PostgreSQL einsetzen / nicht einsetzen?

Architektur-Übersicht



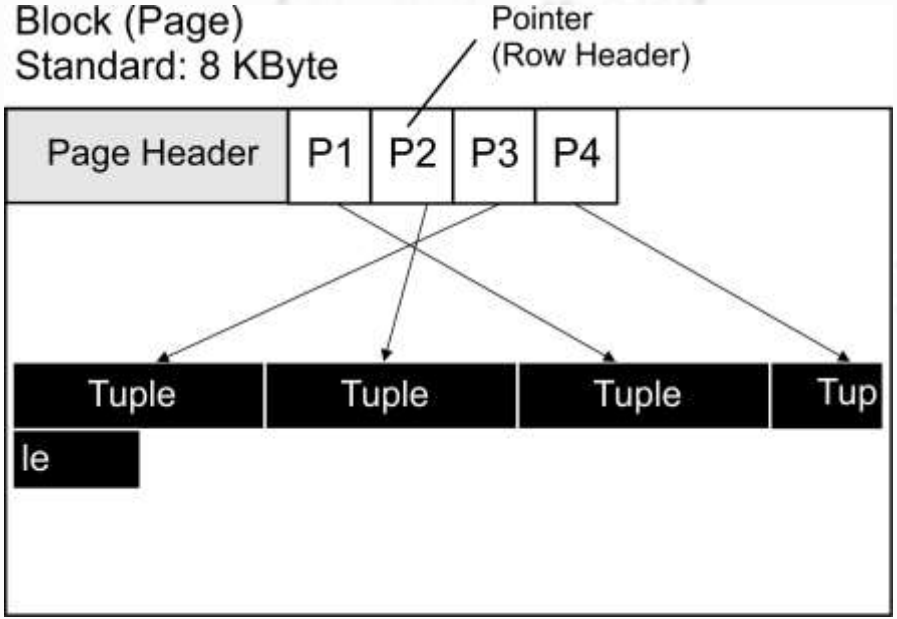
Multiversion Concurrency Control (MVCC)



Open Source - Quellcode

```
typedef struct BufferDesc
{
  BufferTag         tag;           /* ID of page contained in buffer */
  int               buf_id;        /* buffer's index number (from 0) */
  pg_atomic_uint32  state;         /* state of the tag,containing flags,refcount usagecount */
  int               wait_backend_pid; /* backend PID of pin-count waiter */
  int               freeNext;      /* link in freelist chain */
  LWLock            content_lock;  /* to lock access to buffer contents */
} BufferDesc;

* Buffer state is a single 32-bit variable where following data is combined.
 * - 18 bits refcount
 * - 4 bits usage count
 * - 10 bits of flags
 * Combining these values allows to perform some operations without locking
 * the buffer header, by modifying them together with a CAS loop.
```

Aufbau eines Datenblocks (Page)



Identifikation eines Datensatzes

Tuple Identifier (TID): Blocknummer, Offset des Pointers

```
(Lutz@localhost:5432)[hanser]> SELECT ctid,* FROM categories;
  ctid   | category | categoryname
---------+----------+--------------
 (0,1)   |        1 | Action
 (0,2)   |        2 | Animation
 (0,3)   |        3 | Children
 (0,4)   |        4 | Classics
```

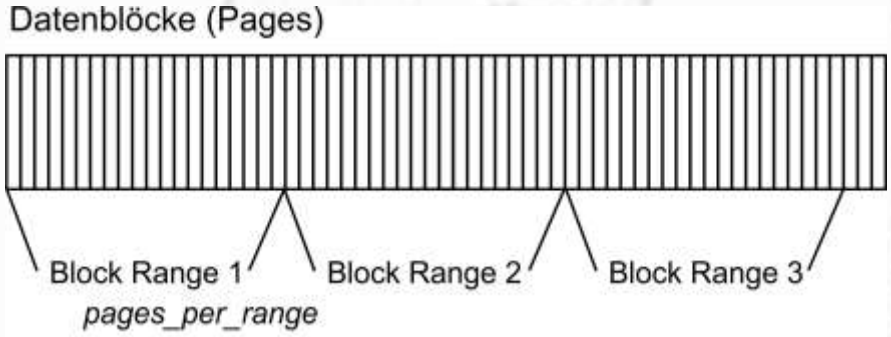
Rohdaten eines Datenblocks ausgeben

```
(Lutz@localhost:5432)[hanser]> SELECT t_ctid, lp_off, t_data
> FROM heap_page_items(get_raw_page('categories',0));

 t_ctid | lp_off |                t_data
--------+--------+-----------------------------------
 (0,1)  |   8152 | \x010000000f416374696f6e
 (0,2)  |   8112 | \x0200000015416e696d6174696f6e
 (0,3)  |   8072 | \x03000000134368696c6472656e
 (0,4)  |   8032 | \x0400000013436c617373696373
```

Features für DWH und große Datenbanken

- Native Table Partitioning (List, Range)
- Paralleles SQL
  - Parallel Bitmap Scans
  - Parallel Btree-Index Scans
  - Parallel Merge Joins
- BRIN Indexe
- Query Rewrite (Rule System)

BRIN Index



Block	Minimum	Maximum
1	2018-01-02	2018-01-06
2	2018-01-04	2018-01-12
3	2018-01-10	2018-01-22

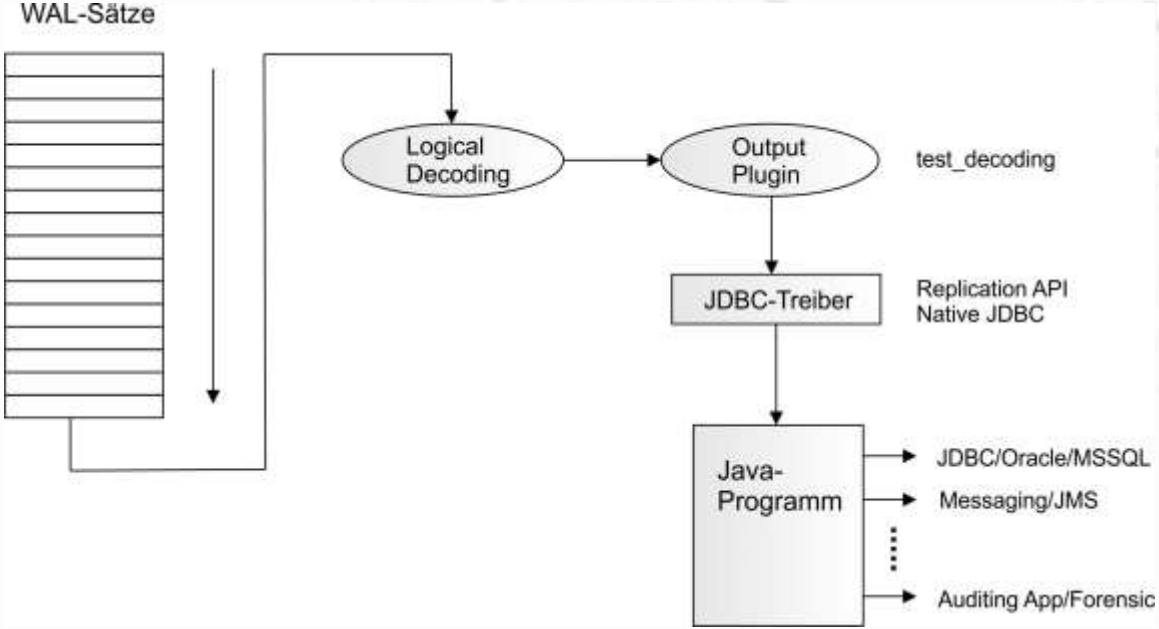
BRIN Index

```
(Lutz@localhost:5432)[hanser]> SELECT * FROM temperature LIMIT 10;
 location_id |       t_time        | t_celsius
-------------+---------------------+-----------
           1 | 2017-01-01 00:00:00 |        17
           1 | 2017-01-01 00:00:01 |        14
           1 | 2017-01-01 00:00:02 |        13
           1 | 2017-01-01 00:00:03 |        14

(Lutz@localhost:5432)[hanser]> CREATE INDEX i_temperature_brin
> ON temperature USING BRIN (t_time)
> WITH (pages_per_range = 128, autosummarize = true);

(Lutz@localhost:5432)[hanser]> \di+ i_temper*
                                 List of relations
 Schema |        Name          | Type  |    Table    |  Size
--------+----------------------+-------+-------------+--------
 public | i_temperature_brin   | index | temperature |  96 KB
 public | i_temperature_btree  | index | temperature | 674 MB
```

Replikation

- Physikalische Replikation (Streaming Replication)
  - Kaskadenförmige Replikation
  - Hot Standby (Active Data Guard)
  - Synchrone Replikation
  - Automatisches Failover mit Observer
- Logische Replikation (ab Version 10)
  - Publisher Subscriber-Prinzip
- Logical Decoding (ab Version 9.6)

Logical Decoding



Die offene Architektur

- Sprachen als Bestandteil des Datenbank-Servers
- SQL-Erweiterungen mit verschiedenen Sprachen
- Das Extension-Netzwerk

SQL erweitern

Folgende Erweiterung der SQL Engine sind möglich:

- Sprachen
- Datentypen
- Funktionen
- Operatoren
- Index-Zugriffsmethoden
- Datenaggregationen

Sprachen in der Datenbank

```
(Lutz@localhost:5432)[hanser]> SELECT lanname, lanispl, lanpltrusted
 > FROM pg_language;

 lanname  | lanispl | lanpltrusted
----------+---------+--------------
 internal | f       | f
 c        | f       | f
 sql      | f       | t
 plpgsql  | t       | t
 plperl   | t       | t
```

C-Programm als SQL-Erweiterung

```
#include "postgres.h"
#include "fmgr.h"
PG_MODULE_MAGIC;
PG_FUNCTION_INFO_V1(add_func);
Datum add_func(PG_FUNCTION_ARGS) {
  int32 arg = PG_GETARG_INT32(0);
  PG_RETURN_INT32(arg + 1);
}

(postgres@[local]:5432)[postgres]> SELECT add_func(10);
 add_func
----------
       11
```

Zugriffsmethoden in PostgreSQL

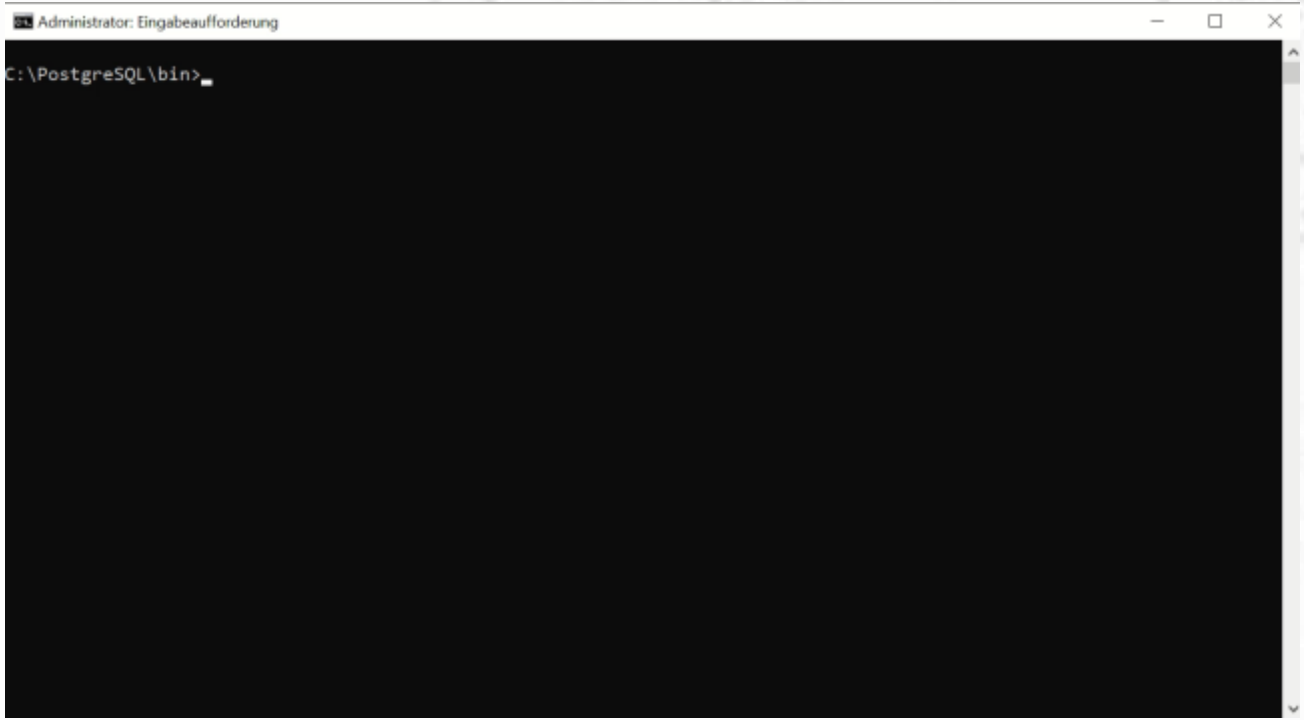
Vorhandene Zugriffsmethoden abfragen:

```
(Lutz@localhost:5432)[hanser]> SELECT * FROM pg_am;
 amname |  amhandler   | amtype
--------+--------------+--------
 btree  | bthandler    | i
 hash   | hashhandler  | i
 gist   | gisthandler  | i
 gin    | ginhandler   | i
 spgist | spghandler   | i
 brin   | brinhandler  | i
```

Erweiterung: Index auf Basis Bloom Filter

- Bloom Filter: Speicher-effiziente Struktur zum Testen ob ein Element Bestandteil eines Sets ist
- Bloom Filter können „False Positives“ liefern: Re-Verification Step
- Die Wahrscheinlichkeit von „False Positives“ kann ermittellt werden
- Bloom Filter Index ist am effektivsten für viele Attribute und viele willkürliche Kombinationen der Bedingungen getestet werden müssen
- Ein einzelner traditioneller BTree-Index kann schneller sein, allerdings sind in der Regel mehrere erforderlich um ein Äquivalent zu liefern
- Analog zu BRIN sind Bloom Filter sehr klein

Erweiterung: Index auf Basis Bloom Filter



Einstieg für Oracle DBAs

- Parameter: listen_addresses
- pg_hba.conf
- shutdown: smart, fast, immediate
- Tools: pgsql, pgAdmin 4, SQLDeveloper
- postgresql.conf

WAL-Archivierung einschalten

- Parameter anpassen:

```
archive_command = ‘copy “%p“ “C:\\PostgreSQL\\archive\\%f“‘
archive_mode = on
wal_level = replica | logical  (nicht minimal!)
```

- Neustart des Clusters

- Einen Log Switch durchführen

Online-Sicherung

```
 (Lutz@localhost:5432)[hanser]> SELECT pg_start_backup('Daily Backup',false,false);
 pg_start_backup
-----------------
 23/60000028

$tar -cvf backup.tar $PGDATA

(Lutz@localhost:5432)[hanser]> SELECT * FROM pg_stop_backup(false,false);
     lsn     |                              labelfile
-------------+-------------------------------------------------------------------
 23/60000130 | START WAL LOCATION: 23/60000028 (file 000000010000002300000060)
             | CHECKPOINT LOCATION: 23/60000060
             | BACKUP METHOD: streamed
             | BACKUP FROM: master
             | START TIME: 2019-04-08 15:14:38 CEST
             | LABEL: Daily Backup
             | START TIMELINE: 1
```

Online-Sicherung

.backup-Datei in $PGDATA/pg_wal:

```
START WAL LOCATION: 23/60000028 (file 000000010000002300000060)
STOP WAL LOCATION: 23/60000130 (file 000000010000002300000060)
CHECKPOINT LOCATION: 23/60000060
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2019-04-08 15:14:38 CEST
LABEL: Daily Backup
START TIMELINE: 1
STOP TIME: 2019-04-08 15:16:34 CEST
STOP TIMELINE: 1
```

PostGIS

- Open Source-Produkt der OSGeo Foundation
- PostgreSQL wurde gewählt wegen Offenheit (neue Datentypen, Indextypen, Operatoren und Funktionen)
- Speicherung geometrischer und geografischer Daten
- Wird als Extension in der PostgreSQL-Datenbank installiert

PostGIS

```
(postgres@localhost:5432)[hanser_postgis]> SELECT * FROM tankstellen;
 ts_id |   lat    |   lon
-------+----------+----------
     1 | 23.8092 | -79.2344
     2 | 23.8487 | -79.2094
     3 | 23.4849 | -79.4616

(postgres@localhost:5432)[hanser_postgis]> SELECT AddGeometryColumn(
> 'public', 'tankstellen', 'geom_col', 2163, 'POINT', 2);

(postgres@localhost:5432)[hanser_postgis]> CREATE INDEX i_geom_tankstellen
> ON tankstellen USING GIST(geom_col);
```

Applikationen für PostgreSQL

Am häufigsten verwendete Sprachen:

- Java / JDBC
- C-Library libpq
- PHP
- Perl-DBI
- PL/pgSQL

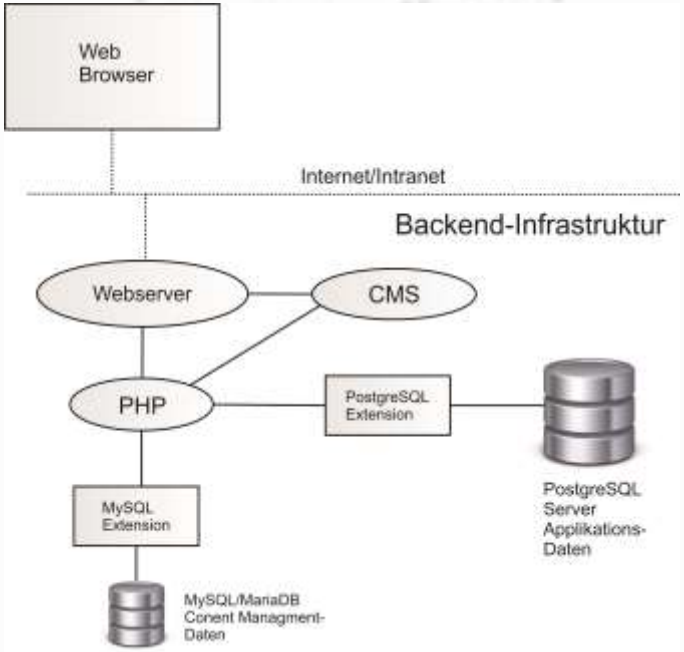
PL/pgSQL

```
CREATE FUNCTION get_actor_name (INTEGER)
RETURNS text AS $$
DECLARE
  v_prodid ALIAS FOR $1;
  v_title TEXT;
  v_actor TEXT;
BEGIN
  SELECT INTO v_title, v_actor
    title, actor
    FROM products WHERE prod_id = v_prodid;
    RETURN v_title || ' - ' || v_actor;
END;
$$ LANGUAGE 'plpgsql';
```

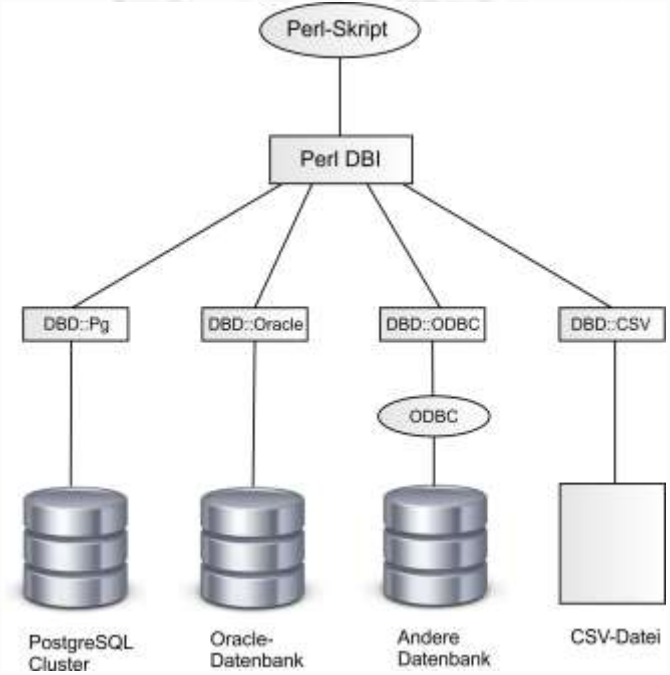
Library libpq

```
#include <stdlib.h>
#include <libpq-fe.h>
int main(int argc, char **argv) {
  const char *conninfo;
  PGconn     *conn;
  conninfo = "host=localhost port=5432 dbname=postgres
user=postgres password=postgres connect_timeout=5";
  conn = PQconnectdb(conninfo);
  if (PQstatus(conn) == CONNECTION_OK) {
    printf("Verbindung erfolgreich hergestellt.\n");
  } else {
    printf("Verbindung fehlgeschlagen.\n");
  }
  PQfinish(conn);
  return EXIT_SUCCESS;
}
```

PHP-Applikationen



Perl DBI


Perl-Skript
Perl DBI
DBD::Pg
DBD::Oracle
DBD::ODBC
DBD::CSV
ODBC
PostgreSQL Cluster
Oracle-Datenbank
Andere Datenbank
CSV-Datei

Perl DBI

```
#!/usr/bin/perl
use DBI;
use strict;
my $dsn = $ARGV[0];
my $user = $ARGV[1];
my $password = $ARGV[2];
my $conn = DBI->connect($dsn, $user, $password, { AutoCommit => 0} );
printf "Status: %s\n", $conn->state;
$conn->{ AutoCommit } = 1;
my $stmt = qq(SELECT version(););
my $sth = $conn->prepare($stmt);
my $rset = $sth->execute() or die $DBI::errstr;
while (my @row = $sth->fetchrow_array()) {
  print $row[0] . "\n";
}
$conn->disconnect;
```

Migration von Oracle

- Unterschiedliche Datentypen
- Behandlung von NULL, leeren Strings, Datum- und Zeittypen
- Sequenzen
- Steigerung der Kompatibilität
- Portierung von PL/SQL Code
- Datenübernahme

Datentypen (Auswahl)

Oracle	PostgreSQL	Bemerkung
NUMBER	NUMERIC	Konvertierung nach FLOAT8 kann erforderlich sein.
CHAR	CHAR	Oracle: Limit 2000 Bytes. PostgreSQL: Limit 10485760 Bytes.
CLOB, NCLOB	**TEXT**	**CLOB ist LOB von Zeichenketten. Maximale Größe 4 GByte. Zusätzliche Portierung erforderlich.**
BLOB	BYTEA oder Large Object.	BLOB kann bis zu 4 GByte groß werden. BYTEA max. 1 GByte. Large Object kann max. 4 TByte goß werden. Spezielle Funktion für Zugriff nötig.

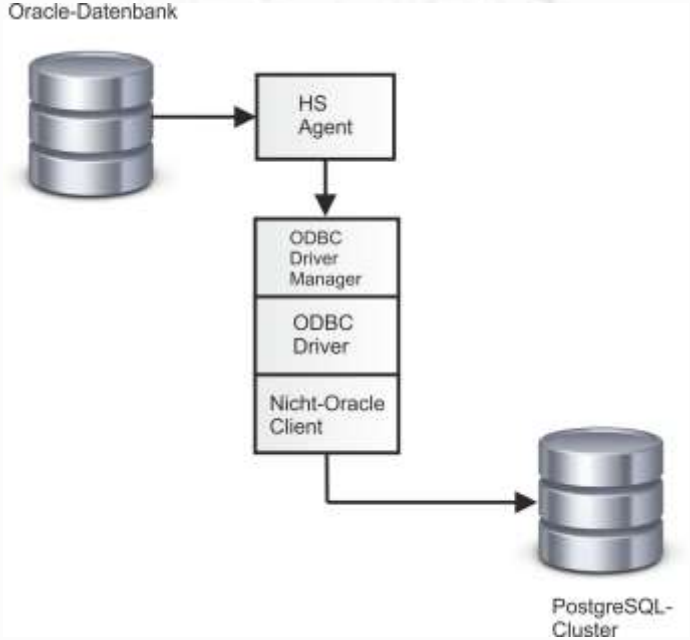
Ora2Pg

- Export eines kompletten Datenbank-Schemas (Tabellen, Views, Indexe, Sequenzen, Primär- und Fremdschlüssel).
- Entladen von Privilegien für Benutzer und Rollen.
- Export von Partitionen vom Typ Range- und List-Partitioning.
- Entladen von Oracle BLOB-Spalten nach PostgreSQL-Typ "BYTEA".
- Grundlegende Konvertierung von PL/SQL-Code nach PL/pgSQL.
- Detaillierter Bericht des Inhalts der Oracle-Datenbank.
- Führt ein Assessment über Aufwand und Schwierigkeitsgrad der Migration durch.

Ora2Pg Datenmigration

```
$ ora2pg -c ora2pg.conf
[==============>] 7/7 tables (100.0%) end of scanning.
[>             ] 0/7 tables (0.0%) end of scanning.
[==============>] 7/7 tables (100.0%) end of table export.
[==============>] 25/25 rows (100.0%) Table COUNTRIES (25 recs/sec)
[==>           ]  25/215 total rows (11.6%) - (0 sec., avg: 25 recs/sec).
[==============>] 27/27 rows (100.0%) Table DEPARTMENTS (27 recs/sec)
[=====>        ]  52/215 total rows (24.2%) - (1 sec., avg: 52 recs/sec).
[==============>] 107/107 rows (100.0%) Table EMPLOYEES (107 recs/sec)
[===========>   ] 159/215 total rows (74.0%) - (2 sec., avg: 79 recs/sec).
[==============>] 19/19 rows (100.0%) Table JOBS (19 recs/sec)
[===========>   ] 178/215 total rows (82.8%) - (2 sec., avg: 89 recs/sec).
[==============>] 10/10 rows (100.0%) Table JOB_HISTORY (10 recs/sec)
[============>  ] 188/215 total rows (87.4%) - (3 sec., avg: 62 recs/sec).
[==============>] 23/23 rows (100.0%) Table LOCATIONS (23 recs/sec)
[=============> ] 211/215 total rows (98.1%) - (3 sec., avg: 70 recs/sec).
[ [==============>] 2/2 procedures (100.0%) end of procedures export.
[==============>] 1/1 views (100.0%) end of output.
[==============>] 3/3 sequences (100.0%) end of output.
[==============>] 1/1 triggers (100.0%) end of output.
```

Mischbetrieb - Datenbanklink



Verwendung von Datenbanklinks

```
HR@ORA122> CREATE TABLE sales
  2  AS SELECT * FROM "sales"@postgres.world;

HR@ORA122> INSERT INTO "public"."sales2"@postgres.world VALUES (v1,v2,v3,v4,v5);

HR@ORA122> SELECT c."firstname",c."lastname",c."zip",o."orderid",o."totalamount"
  2  FROM customers c, "orders"@postgres.world o
  3  WHERE c."customerid" = o."customerid„ AND "country" = 'US‘;

HR@ORA122> DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@postgres.world('create table
regions(region_id integer,region_name VARCHAR(50))');
```

PostgreSQL in der IT-Landschaft

- Sicherung und Wiederherstellung
- Desaster Recovery
- Monitoring
- Verfügbarkeit
- Datensicherheit und Auditing
- Performance und Skalierbarkeit
- Schnittstellen und Kommunikation
- Support
- Cloud

F&A

PostgreSQL für Oracle DBAs

DATAGUARDS

Excellence in Data Technologies