Was ist neu in PostgreSQL 8.4?

FrOSCon – 23. August 2009

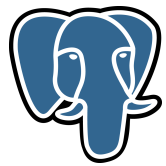
Andreas 'ads' Scherbaum

Web: http://andreas.scherbaum.la/
E-Mail: andreas[at]scherbaum.biz
PGP: 9F67 73D3 43AA B30E CA8F 56E5 3002 8D24 4813 B5FE

23. August 2009

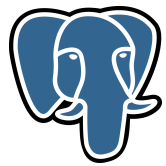
Was ist PostgreSQL?

- Relationale Datenbank
- BSD Lizenz, weltweit aktive Community
- Zahlreiche Features und Funktionen (Foreign Keys, Transaktionen, Trigger)
- Läuft auf zahlreichen Betriebssystemen und diverser Hardware
- Weitgehendes Einhalten der SQL-Standards – Dokumentation der Abweichungen
- Im Schnitt pro Jahr ein Minor-Release mit neuen Features
- Version 8.4 ist vor kurzem erschienen



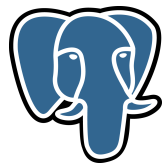
Was ist PostgreSQL?

- Relationale Datenbank
- BSD Lizenz, weltweit aktive Community
- Zahlreiche Features und Funktionen (Foreign Keys, Transaktionen, Trigger)
- Läuft auf zahlreichen Betriebssystemen und diverser Hardware
- Weitgehendes Einhalten der SQL-Standards – Dokumentation der Abweichungen
- Im Schnitt pro Jahr ein Minor-Release mit neuen Features
- Version 8.4 ist vor kurzem erschienen



Der Dozent

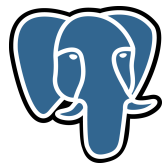
- Name: Andreas Scherbaum
- Selbstständig im Bereich Datenbanken, Linux auf Kleingeräten, Entwicklung von Webanwendungen
- Arbeit mit Datenbanken seit 1997, mit PostgreSQL seit 1999
- Gründungsmitglied der Deutschen und der Europäischen PostgreSQL User Group
- Board of Directors – European PostgreSQL User Group



Was bedeutet dieses komische .la?

http://andreas.scherbaum.la/

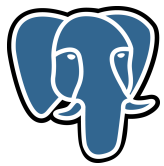
- .la ist die TLD von Laos
- .la wird von Los Angeles genutzt und verwaltet
- LA ist das KFZ-Kennzeichen von Landshut/Niederbayern
- Dort habe ich längere Zeit gewohnt und meine Frau fand die Domain schön



Was bedeutet dieses komische .la?

http://andreas.scherbaum.la/

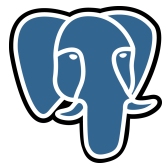
- .la ist die TLD von Laos
- .la wird von Los Angeles genutzt und verwaltet
- LA ist das KFZ-Kennzeichen von Landshut/Niederbayern
- Dort habe ich längere Zeit gewohnt und meine Frau fand die Domain schön



Was bedeutet dieses komische `.la`?

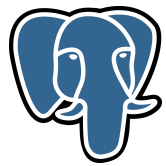
`http://andreas.scherbaum.la/`

- `.la` ist die TLD von Laos
- `.la` wird von Los Angeles genutzt und verwaltet
- `LA` ist das KFZ-Kennzeichen von Landshut/Niederbayern
- Dort habe ich längere Zeit gewohnt und meine Frau fand die Domain schön



1. Juli 2009 – PostgreSQL 8.4

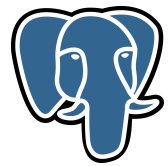
Die PostgreSQL Global Development Group hat **Version 8.4** veröffentlicht und setzt damit die **kontinuierliche Entwicklung** der fortschrittlichsten Open Source Datenbank der Welt fort. Dieses Release enthält eine **Fülle an Verbesserungen** um die Administration, Anfragen und das Programmieren von PostgreSQL-Datenbanken noch einfacher als zuvor zu gestalten. Mit **293 neuen oder verbesserten Features** in Version 8.4 gibt es noch mehr Gründe, PostgreSQL für Ihr nächstes Projekt zu wählen.



PostgreSQL Flyer

10 Dinge die PostgreSQL 8.4 nicht tun wird :-(

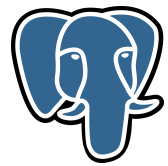
1. PostgreSQL 8.4 zerstört nicht den Regenwald.
2. Krebs wird nicht verursacht, es wird nur bei der Heilung geholfen.
3. Es schädigt nicht die Ozonschicht.
4. Es wird kein Beitrag zur Erderwärmung geleistet.
5. Es versursacht keine Hautverunreinigungen.
6. Es belastet Ihre Leber nicht.
7. Gefährdet weder Wale noch sonstige vom Aussterben bedrohte Arten.
8. Es verschärft nicht die globale Finanzkrise.
9. Es hält Sie nicht vom Singen unter der Dusche ab.
10. Es wird Ihren Zähnen nicht schaden.



PostgreSQL Flyer

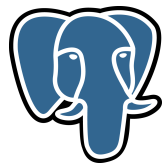
10 Dinge die PostgreSQL 8.4 sicher tun wird :-)

1. Rekursive Abfragen erlauben den Abruf eines ganzen Web-Forums oder Kommentar-Baumes mit einer einzigen Abfrage.
2. Paralleles Wiederherstellen ermöglicht es, ein Backup bis zu 8 Mal schneller zurückzuspielen.
3. Windowing Funktionen mit deren Hilfe komplexe Berichte vereinfacht werden.
4. Neue Werkzeuge zur Überwachung erlauben eine raschere und einfachere Analyse der Last auf dem Datenbank Server.
5. Die neue "Visibility-Map" und die Statistikverbesserungen vereinfachen den Betrieb von sehr großen Datenbanken.



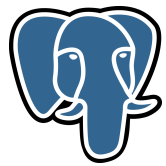
10 Dinge die PostgreSQL 8.4 sicher tun wird :-)

6. Erweiterte Syntax und Möglichkeiten für "Stored Procedures" sind einfacher und flexibler im Gebrauch.
7. Spaltenbezogene Berechtigungen, die es ermöglichen den Zugriff auf sensible Daten besser zu regulieren
8. `pg_migrator` macht das Upgrade von 8.3 nach 8.4 zum Kinderspiel für die meisten Benutzer.
9. Datenbankbezogene Kollationen, somit ist eine Verwendung in mehrsprachigen Umgebungen viel einfacher.
10. Viele komplexe Abfragen laufen deutlich schneller.



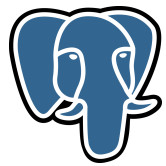
Performance

- `default_statistics_target` und `column statistics`
  - Defaultwerte von 10 auf 100 erhöht
  - Maximalwert von 10.000 jetzt möglich (vorher 1.000)
  - Ändern mit `ALTER TABLE ... ALTER COLUMN ... SET STATISTICS ...`



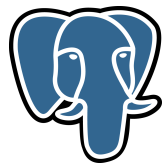
Performance

- `posix_fadvise()` Erweiterung und `effective_io_concurrency`
  - Konfigurationsparameter: `effective_io_concurrency(integer)`
  - Gibt die Anzahl der parallel möglichen I/O-Operationen per Disk an
  - Werte von 1 bis 1000, 0 schaltet asynchrone I/O-Requests aus
  - Guter Startwert ist die Anzahl der Festplatten (z. B. im RAID)
  - Asynchrone I/O-Requests benötigen die POSIX fadvise Funktion (`posix_fadvise()`)



Performance

- Hash-Methoden für `DISTINCT` / `UNION` / `INTERSECT` / `EXCEPTION`
  - Früher war eine Sortieroperation über die Ergebnismenge notwendig
  - Nun können Hash-basierte Methoden Verwendung finden
  - Das geschieht direkt im Planer, Änderungen sind nicht notwendig



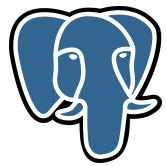
Kollation per Datenbank

- Angaben zur Kollation sind jetzt per Datenbank, nicht mehr nur per Cluster

Beispiel (Kollation per Datenbank)

```
CREATE DATABASE russisch WITH
       ENCODING = UTF8,
       LC_COLLATE = ru_RU.utf8,
       LC_CTYPE = ru_RU.utf8;
```

  - Datenbanken mit verschiedenen Sortierungen sind jetzt möglich
  - Leider immer noch keine Möglichkeit, verschiedene Tabellen (oder Datensätze) verschiedenen Sprachen (bzw. Sortierungen) zuzuordnen



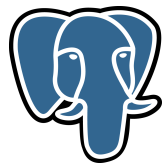
Kollation per Datenbank

- Angaben zur Kollation sind jetzt per Datenbank, nicht mehr nur per Cluster

Beispiel (Kollation per Datenbank)

```
CREATE DATABASE russisch WITH
       ENCODING = UTF8,
       LC_COLLATE = ru_RU.utf8,
       LC_CTYPE = ru_RU.utf8;
```

- Datenbanken mit verschiedenen Sortierungen sind jetzt möglich
- Leider immer noch keine Möglichkeit, verschiedene Tabellen (oder Datensätze) verschiedenen Sprachen (bzw. Sortierungen) zuzuordnen

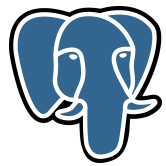


Free-Space-Map Konfiguration ist Geschichte

Beispiel (Wer kennt das: Free-Space-Map Konfiguration)

```
max_fsm_pages = 153600
max_fsm_relations = 1000
```

- `Free-Space-Map` wird jetzt automatisch konfiguriert
- Die neue `Visibility Map` verfolgt durch Transaktionen verursachte Änderungen
  - netter Nebeneffekt: `VACUUM` weiß vorher, welche Speicherseiten geändert wurden

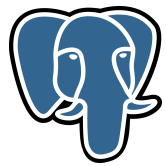


Free-Space-Map Konfiguration ist Geschichte

Beispiel (Wer kennt das: Free-Space-Map Konfiguration)

```
max_fsm_pages = 153600
max_fsm_relations = 1000
```

- `Free-Space-Map` wird jetzt automatisch konfiguriert
- Die neue `Visibility Map` verfolgt durch Transaktionen verursachte Änderungen
  - netter Nebeneffekt: `VACUUM` weiß vorher, welche Speicherseiten geändert wurden



pg_autovacuum Tabelle ist ebenfalls Geschichte

- Die `pg_autovacuum` Tabelle wurde entfernt
- Stattdessen wird die Konfiguration mittels `ALTER TABLE` geändert
  - Vorteil: diese Einstellungen werden endlich im Backup mitgesichert
  - Nachteil: Skripten ist nicht mehr ohne weiteres möglich

Beispiel (Autovacuum ausschalten)

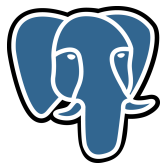
```
ALTER TABLE ... SET (autovacuum_enabled = off);
ALTER TABLE ... SET (toast.autovacuum_enabled = off);
```

paralleles Wiederherstellen

- `pg_restore` kann neuerdings mehrere Verbindungen beim Wiederherstellen nutzen
  - Anforderung: direkte Verbindung zur Datenbank
  - Anforderung: Backup im Custom Format notwendig (-F c)

Beispiel (Schneller Backups wiederherstellen)

```
pg_restore --dbname=test_neu --jobs=2 backup.dump
```

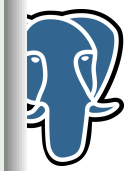


EXPLAIN VERBOSE gibt Spalten aus

- Die neue VERBOSE Option für EXPLAIN gibt die Spalten aus

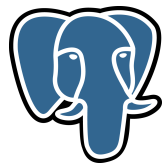
Beispiel (Spalten anzeigen)

```
test=# explain verbose select * from pg_views ;
                                        QUERY PLAN
-----------------------------------------------------------------------------
 Hash Left Join  (cost=1.14..11.79 rows=84 width=136)
   Output: n.nspname, c.relname,
       pg_get_userbyid(c.relowner), pg_get_viewdef(c.oid)
   Hash Cond: (c.relnamespace = n.oid)
   ->  Seq Scan on pg_class c  (cost=0.00..9.07 rows=84 width=76)
         Output: c.relname, c.relowner, c.oid, c.relnamespace
         Filter: (relkind = 'v'::"char")
   ->  Hash  (cost=1.06..1.06 rows=6 width=68)
         Output: n.nspname, n.oid
         ->  Seq Scan on pg_namespace n  (cost=0.00..1.06 ....
               Output: n.nspname, n.oid
(10 Zeilen)
```



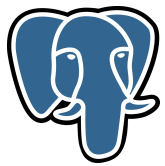
Update

- `pg_migrator` erlaubt das Updaten einer Datenbank von 8.3 auf 8.4 – ohne Dump & Reload



Kleinigkeiten

- Die `pg_settings` Tabelle zeigt jetzt die Defaultwerte
- Bei einem Deadlock werden alle beteiligten Anfragen ausgegeben
- GIN-Indexe über mehrere Spalten
- `auto_explain` Modul loggt automatisch den Anfrageplan (z. B. für langsame Anfragen oder innerhalb von Funktionen)
- `pg_stat_user_functions` protokolliert Anzahl Aufrufe und durchschnittliche Laufzeit von Funktionen
- SSL-Zertifikate jetzt auch clientseitig
- ...



spaltenbasierte Zugriffsrechte

- Zugriffsrechte lassen sich spaltenbasiert vergeben
- Diese Rechte greifen nur, wenn es keine tabellenbasierten Rechte gibt

Beispiel (Beispieltabelle)

```
CREATE TABLE spaltenbasiert (
  id        SERIAL        PRIMARY KEY,
  daten     TEXT
);

INSERT INTO spaltenbasiert (daten)
VALUES ('23'), ('42');
```

spaltenbasierte Zugriffsrechte

Beispiel (Zugriffsrechte vergeben)

```
REVOKE ALL ON spaltenbasiert FROM PUBLIC;
GRANT SELECT (daten) ON spaltenbasiert TO PUBLIC;

\dp spaltenbasiert
Zugriffsrechte
-[ RECORD 1 ]------------+----------------
Schema                   | public
Name                     | spaltenbasiert
Typ                      | table
Zugriffsrechte           | ads=arwdDxt/ads
Column access privileges | daten:
                         :   =r/ads
```

spaltenbasierte Zugriffsrechte

- als normaler Benutzer

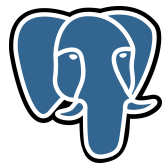
Beispiel (auf Tabelle zugreifen)

```
SELECT id, daten FROM spaltenbasiert;
ERROR:  permission denied for relation spaltenbasiert

SELECT daten FROM spaltenbasiert;
 daten
-------
 23
 42
(2 Zeilen)
```

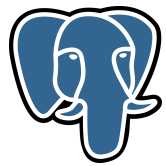
psql

- Autovervollständigung verbessert
- \timing akzeptiert jetzt "on" und "off"
- bessere Darstellung von ENUMs
- referenzierte Tabellen werden jetzt angezeigt
- \l+ zeigt jetzt die Größe der Datenbanken an
- \dt+ zeigt jetzt die Größe der Tabellen an
- \df zeigt keine Systemfunktionen mehr an
- \ef erlaubt das Editieren von Funktionen



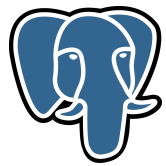
Kleinigkeiten

- `generate_subscripts()` erlaubt eine vereinfachte Handhabung von Arrays
- `generate_series()` versteht sich jetzt auch auf Datumsangaben
- `array_agg()` und `unnest()` aggregieren und zerlegen ein Array
- Funktionen können eine variable Anzahl an Parametern sowie Defaultparameter bekommen
- `pl/pgSQL` kennt jetzt `CASE`
- dynamische Anfragen in `pl/pgSQL` dürfen jetzt Platzhalter nutzen (`EXECUTE USING`)
- `RETURN QUERY` liefert das Ergebnis einer Anfrage zurück – keine lästige Schleife mehr notwendig
- `TRUNCATE` führt nun Statement-basierte Trigger aus



Views erweitern

- Views lassen sich (ohne Löschen und Neuerstellen des View) um neue Spalten am Ende erweitern
- Die bisherigen Spalten dürfen sich allerdings nicht ändern



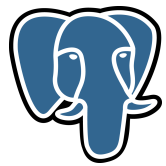
Sequenzen neustarten

- Beim Löschen von Tabellen mittels Truncate lassen sich die Sequenzen zurücksetzen

Beispiel (Schneller Backups wiederherstellen)

```
TRUNCATE TABLE ... RESTART IDENTITY;
```

- Der Neustart der Sequenz ist nicht transaktionssicher



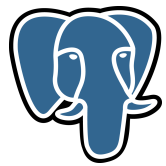
Sequenzen neustarten

- Beim Löschen von Tabellen mittels Truncate lassen sich die Sequenzen zurücksetzen

Beispiel (Schneller Backups wiederherstellen)

```
TRUNCATE TABLE ... RESTART IDENTITY;
```

- Der Neustart der Sequenz ist nicht transaktionssicher

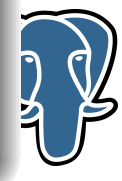


Kleinigkeiten

- Parameter für `LIMIT` kann nun ein Ausdruck oder ein Subselect sein
- Schlüsselwort `AS` für Spaltenaliase ist nun optional
- `TABLE` ist ein Alias für `SELECT * FROM tablename`
- `citext` Contrib-Modul erlaubt Suchen in Textfeldern ohne Beachtung von Groß- und Kleinschreibung
- ...

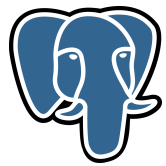
Beispiel (Schneller Backups wiederherstellen)

```
SELECT ...
  FROM ...
 LIMIT (SELECT anzahl
          FROM konfiguration);
```



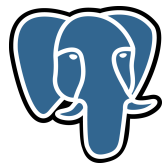
Kleinigkeiten

Oh, just one more thing!



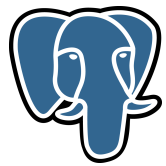
Windowing Functions, CTE und rekursive Anfragen

- Windowing Functions erlauben das Aggregieren von Daten über einen beliebigen unabhängigen Ausschnitt der Anfrage
  - Typischer Anwendungsfall: Data Warehouse
- Common Table Expressions (CTE) erlauben benannte Unteranfragen, die in anderen Teilen der aktuellen Anfrage genutzt werden können
  - Notwendigkeit für temporäre Tabellen entfällt
  - rekursive Anfragen sind möglich



Windowing Functions, CTE und rekursive Anfragen

- Windowing Functions erlauben das Aggregieren von Daten über einen beliebigen unabhängigen Ausschnitt der Anfrage
  - Typischer Anwendungsfall: Data Warehouse
- Common Table Expressions (CTE) erlauben benannte Unteranfragen, die in anderen Teilen der aktuellen Anfrage genutzt werden können
  - Notwendigkeit für temporäre Tabellen entfällt
  - rekursive Anfragen sind möglich



Sissa ibn Dahirs und die Reiskörner

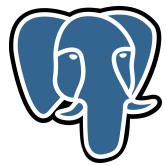
Beispiel (Anzahl Reiskörner auf dem Schachbrett)

```
SELECT feld, koerner_einzeln,
   SUM(koerner_einzeln) OVER (ORDER BY feld)
                                AS koerner_summe
  FROM (
  WITH  RECURSIVE reis AS
       (SELECT 1 AS feld, 1::DECIMAL AS koerner_einzeln
 UNION  ALL
SELECT  feld + 1, koerner_einzeln * 2
  FROM  reis
 WHERE  feld < 64)
SELECT  *
  FROM  reis) AS rekursion;
```

Sissa ibn Dahirs und die Reiskörner

Beispiel (Anzahl Reiskörner auf dem Schachbrett)

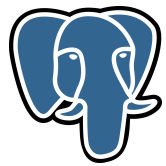
```
 feld |    koerner_einzeln    |     koerner_summe
------+------------------------+-----------------------
    1 |                      1 |                      1
    2 |                      2 |                      3
 ...
   63 | 4611686018427387904 |  9223372036854775807
   64 | 9223372036854775808 | 18446744073709551615
(64 Zeilen)
```



Baumstruktur

Beispiel (einen Baum erzeugen)

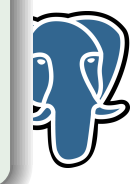
```
CREATE TABLE orte (
  id INT NOT NULL PRIMARY KEY,
  gehoert_zu INT,
  name TEXT NOT NULL UNIQUE
);
```



Baumstruktur

Beispiel (den Baum befüllen)

```
-- irgendwo muss ein Baum beginnen
INSERT INTO orte (id, gehoert_zu, name)
     VALUES (1, NULL, 'Deutschland');
INSERT INTO orte (id, gehoert_zu, name)
     VALUES (2, 1, 'Sachsen-Anhalt');
INSERT INTO orte (id, gehoert_zu, name)
     VALUES (3, 1, 'Niedersachsen');
INSERT INTO orte (id, gehoert_zu, name)
     VALUES (4, 2, 'Magdeburg');
INSERT INTO orte (id, gehoert_zu, name)
     VALUES (5, 4, 'Magdeburg/Hasselbachplatz');
INSERT INTO orte (id, gehoert_zu, name)
     VALUES (6, 3, 'Hannover');
```



Baumstruktur

Beispiel (rekursive Abfrage)

```
WITH RECURSIVE rekursion(eintrag, name, pfad) AS (
  -- das ist der Kopf des Baumes, ohne Zugehoerigkeit
  SELECT id, name, ARRAY[id] FROM orte
   WHERE gehoert_zu IS NULL
  UNION ALL
  SELECT o.id, o.name,
         rekursion.pfad || ARRAY[o.id]
    FROM orte o
    JOIN rekursion ON (o.gehoert_zu = rekursion.eintrag)
   WHERE id NOT IN (rekursion.eintrag)
)
  SELECT * FROM rekursion ORDER BY pfad;
```

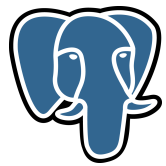
Baumstruktur

Beispiel (Ergebnis)

```
 eintrag |            name             |   pfad
---------+-----------------------------+-----------
       1 | Deutschland                 | {1}
       2 | Sachsen-Anhalt              | {1,2}
       4 | Magdeburg                   | {1,2,4}
       5 | Magdeburg/Hasselbachplatz | {1,2,4,5}
       3 | Niedersachsen               | {1,3}
       6 | Hannover                    | {1,3,6}
(6 Zeilen)
```

Ausblick auf PostgreSQL 8.5 (WIP)

- Query Optimizer wird überarbeitet
- `UNIQUE` wird `DEFERRABLE`
- `DROP COLUMN/CONSTRAINT IF EXISTS`
- neue `EXPLAIN` Ausgabe Formate: XML und JSON
- Handhabung partitionierter Tabellen wird vereinfacht
- Replikation!
- Alpha Releases
- Updateable Views?
- ...

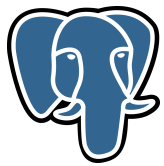


Wichtiger Termin

Wichtiger Termin

`Podiumsdiskussion: OpenSQL Camp`

- 16:30 Uhr
- Raum C 120 / OpenSQLCamp



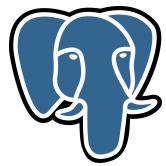
Wichtiger Termin

Wichtiger Termin

`PGDay.eu 2009`

- 6./7. November 2009
- in Paris
- Für Anwender und Entwickler
- Sponsoren sind gern gesehen

Webseite: `http://www.pgday.eu/`



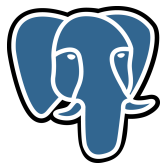
Wichtiger Termin

Wichtiger Termin

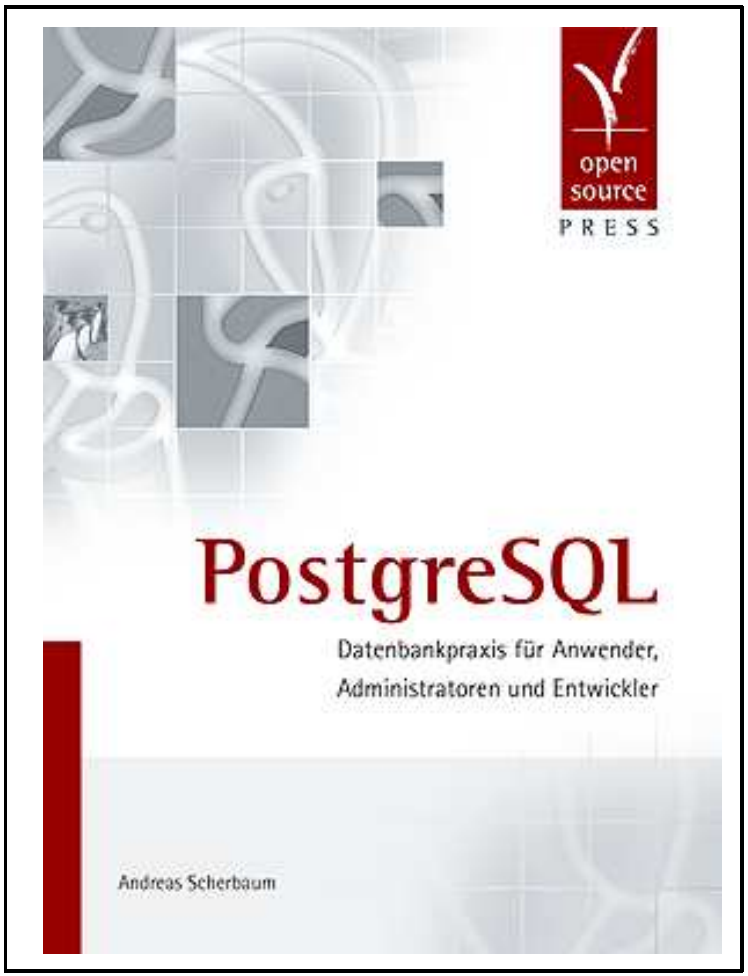
FOSDEM 2010

- 6./7. Februar 2010
- in Brüssel
- größte Open-Source Konferenz in Europa

Webseite: `http://www.fosdem.org/`

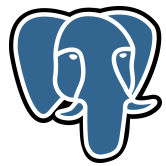


PostgreSQL Buch



PostgreSQL – Datenbankpraxis für Anwender, Administratoren und Entwickler

Draußen bei Open Source Press erhältlich
Umfang: ca. 550 Seiten



Ende

http://andreas.scherbaum.la/

Fragen?

Andreas 'ads' Scherbaum <andreas@scherbaum.biz>
PostgreSQL User Group Germany
European PostgreSQL User Group