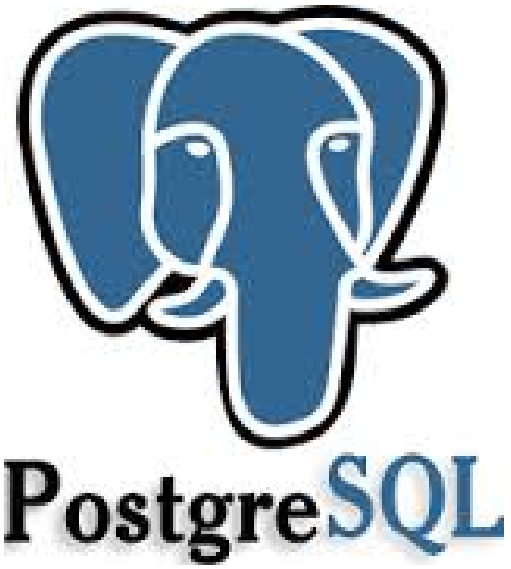
PostgreSQL

Ein paar bescheidene Tipps...



Stefan Tzeggai, Tübingen

Programme / Tools

- *deb http://apt.postgresql.org/pub/repos/apt/ xenial-pgdg main*
  https://wiki.postgresql.org/wiki/Apt
- Clients: PgAdmin3 + psql
- PostGIS - weil ich ja Geograph bin ;-)
- PGXN PostgreSQL Extension Network
- *apt install postgresql-9.6-postgis-2.3 postgresql-9.6-postgis-2.3-scripts postgis pgadmin3 pgxnclient postgresql-server-dev-9.6*

Use PgAdmin3 easily

```
sudo vim /etc/postgresql/9.6/main/pg_hba.conf
```

```
#TYPE  DATABASE    USER        ADDRESS                 METHOD
host   tuebix      all         127.0.0.1/32            trust
```

but PgAdmin3 sucks...

Create example DB & data

…

…

…

Materialized Views

- Problem: Die Anwendung will Statistiken anzeigen, die zu lange dauern um ON-DEMAND berechnet zu werden
  => GUI hangs
- Schmerzhafte Lösung: Ergebnis in der Anwendung cachen
- Problem: 2+ Anwendungen berechnen die gleichen Statistiken mehrmals => CPU-Verschwendung

- Lösung: Ergebnisse als MATERIALIZED VIEWs so oft wie möglich/nötig berechnen und diese minimal veralteten Zahlen in der/den Anwendungen anzeigen.

MATERIALIZED VIEW Unterstützung in PG:

- Auf MAT.VIEWs können INDEXE erstellt werden! (Auf VIEWs nicht)
- REFRESH CONCURRENTLY wird unterstützt, wenn ein UNIQUE Index auf dem View existiert
- Man kann per SQL checken, welche MAT.VIEWs keinen UNIQUE INDEX haben
- DROP CASCADE wird unterstützt

...show in SQL...

refreshMatView.sh skript called via CRON

```
echo Milliseconds until TIMEOUT: $1
echo Name of materialized view to be refreshed concurrently: $2

LOCKFILE=/var/lock/$2.lock
COMMENT=`date +"%Y-%m-%d %H:%M"`
/usr/bin/flock -n $LOCKFILE -c "/usr/bin/psql -c \"
  begin; set statement_timeout to $1; set work_mem to '12GB'; commit; begin;
  REFRESH MATERIALIZED VIEW CONCURRENTLY $2;
  COMMENT ON MATERIALIZED VIEW $2 IS '$COMMENT last time
     CONCURRENTLY refreshed';
  commit;\";" > /dev/null
if [ $? -ne 0 ]; then
    echo "Looks like a job for lock $2 is already running until success or timeout after $1 ms"
fi
```

Zusammenfassung: MATERIALIZED VIEWS + CRON

- Zentrale Stelle um Multi-Core-Server auszulasten
- Keine doppelten Ausführungen (wegen lock-file)
- Weniger Codezeilen als in Java/Anwendung
- MATVIEWs können als API fungieren

- Problem: SQL-Code ist nicht Teil der Versionsverwaltung GIT/SVN etc., sondern liegt im DB-Backup.

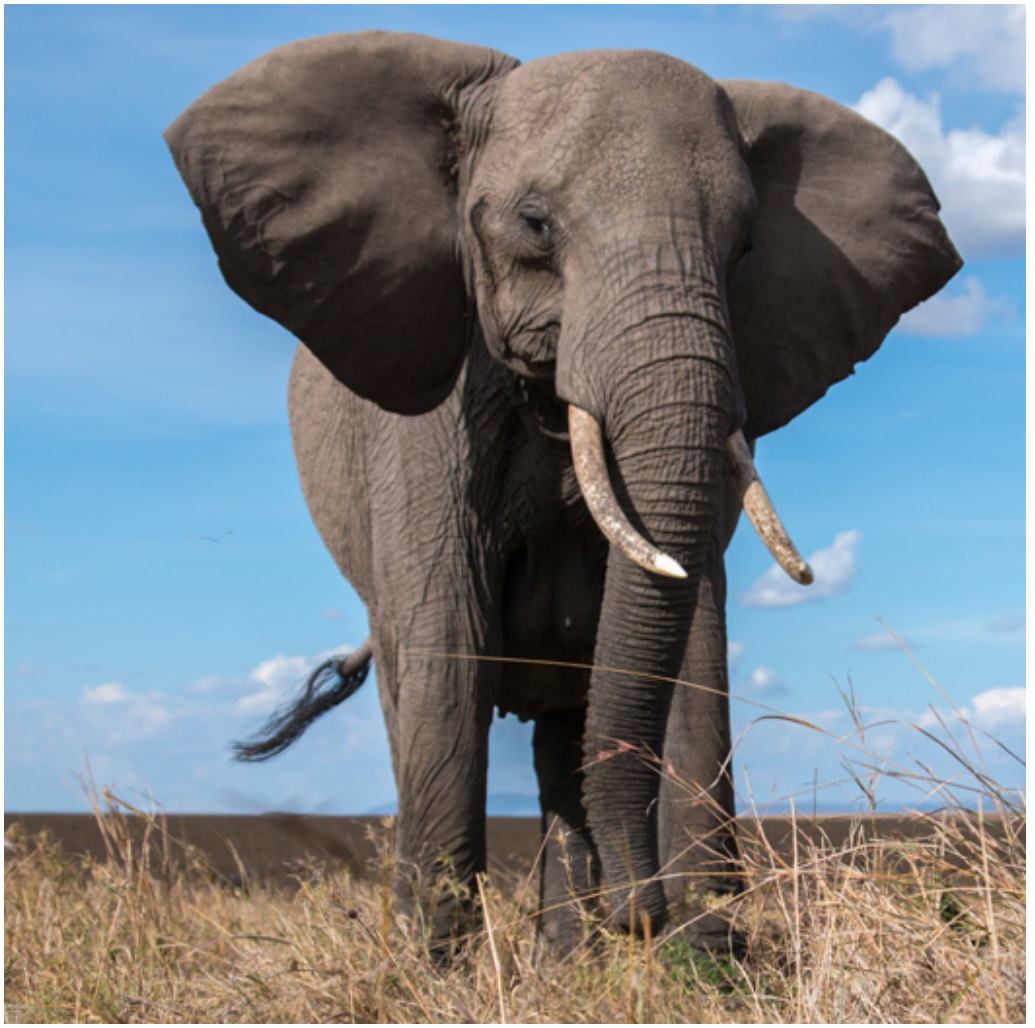
PostgreSQLs Vorteile

(my 5¢)

- OpenSource („a liberal Open Source license, similar to the BSD or MIT“)
- Sehr gute Dokumentation
- IMHO: Gesundes Ökosystem von Firmen die PG unterstützen – Wird nicht wie MySQL plötzlich gekauft
- >20 Jahre alt und stetig aktiv entwickelt.
- Tolle neue Features in den letzten und nächsten Versionen

  MAT.VIEWS, REPLICATION/STANDBY/FAILOVER, JSON=NOSQL, AGGREGATION:CUBE,GROUPING SETS, SYNTAX (like FILTER), PERCENTILES/QUANTILES, UPSERT, BRIN-INDEX, PARALLEL QUERY EXECUTION
- Features zielen auf Enterprise – Alternative zu Oracle

Danke!



Wenn noch Zeit ist:

Bitte Eure Fragen!

- UPSERT
- BRIN-Index
- INHERITANCE / Partitionierung
- NOSQL mit PG, JSON / JSONB

UPSERT

- INSERT … ON CONFLICT … DO …
  https://www.postgresql.org/docs/current/static/sql-insert.html#SQL-ON-CONFLICT
  I

- INSERT INTO distributors (did, dname) VALUES (7, 'Redline GmbH')
  ON CONFLICT (did) DO NOTHING;

- INSERT INTO distributors (did, dname)
  VALUES (5, 'Gizmo Transglobal'), (6, 'Associated Computing, Inc')
  ON CONFLICT (did) DO UPDATE SET dname = EXCLUDED.dname;

Thanks!

Stefan Tzeggai

stefan@tzeggai.name