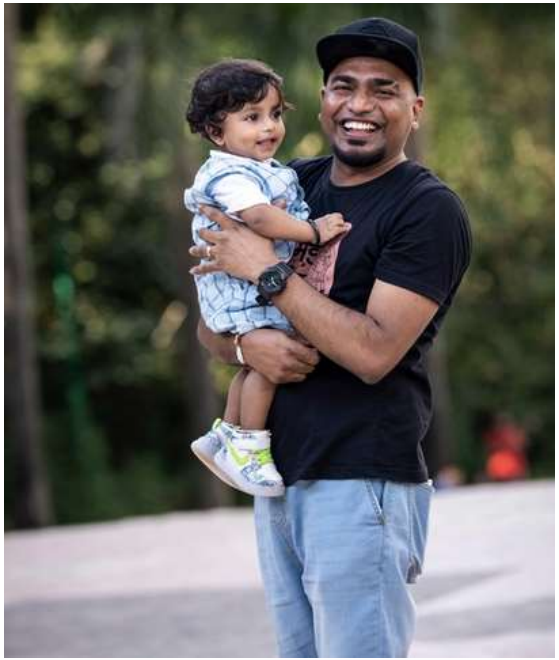
Must-Know Postgres Extensions for DBAs and Developers During Migration

Deepak Mahto
DataCloudGaze Consulting

About Me

I am Deepak Mahto, and I like to call myself a Database Guy.

- Founder of DataCloudGaze Consulting.
- I have 15+ years of database experience, with more than 7 years in cloud and migrations.
- I have published 150+ technical blogs on databases.
- I live in Mumbai and have a 3-year-old child.
- Loves to explore street food.

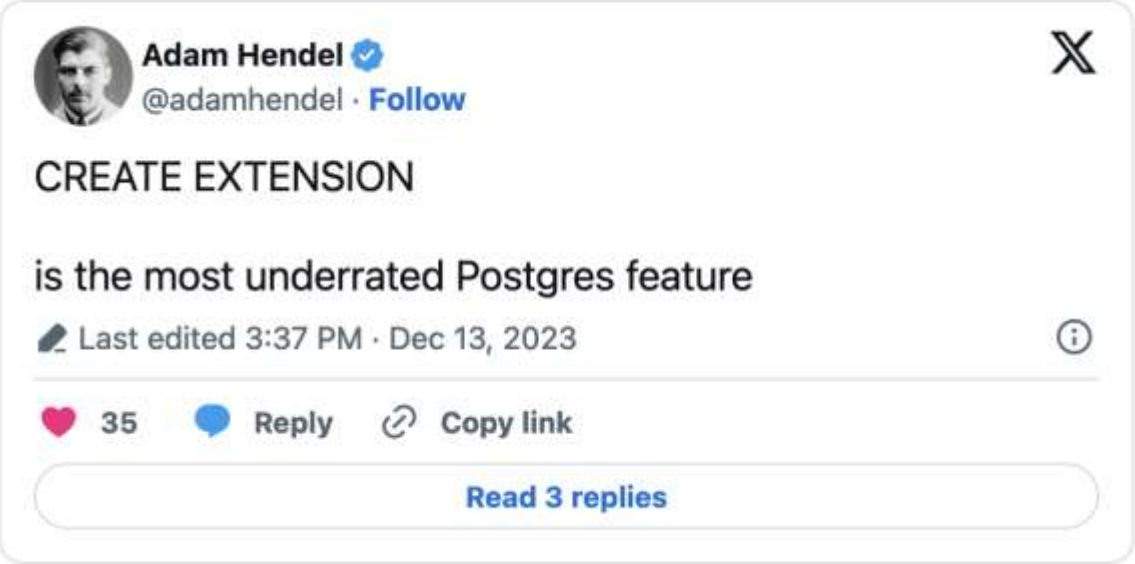


Agenda

- What are Postgres Extensions and Their Importance
- Some Fun Facts About Extensions
- List of Extensions to Aid in Migration from Heterogeneous Sources Like Oracle and MSSQL

What is extension?

PostgreSQL's extensibility allows for seamless integration of extensions, making them function like built-in features, thus enhancing its capabilities and flexibility.



Extensive versatility of Extension's

ExtensionCategory	ExtensionCount	Extension List
Analytics	14	hydra_columnar/pg_timeseries/timescaledb
AnalyticsConnectors	1	pg_tier
Auditing / Logging	7	auto_explain/pgaudit
Change Data Capture	6	pg_ivm/pglogical/wal2json
Connectors	27	dblink/oracle_fdw/mysql_fdw/postgres_fdw
Data / Transformations	52	citext/postgis/postgresql_anonymizer
Debugging	2	plpgsql_check
Index / Table Optimizations	15	hypopg/pg_repack/pg_squeeze/pgtt
Machine Learning	3	pg_embedding-pgvector-postgresml
Machine LearningOrchestration	1	vectorize
Metrics	17	pg_proctab/pg_stat_kcache/pg_stat_statements
MetricsTooling / Admin	1	pgtelemetry
Orchestration	8	pg_cron/pg_dbms_job/pg_partman
Procedural Languages	16	plpgsql/plproxy/plrust/plv8/pljava
Query Optimizations	6	pg_hint_plan/pg_stat_monitor
Search	11	fuzzystrmatch/pg_trgm
Security	11	passwordcheck/pgcrypto
Tooling / Admin	13	pgtap/adminpack/citus

Reference - https://cloud.tembo.io

Create Extension in Postgres

CREATE EXTENSION loads and manages new extensions, requiring appropriate privileges.

```
extensionrepo=# \help create extension
Command:     CREATE EXTENSION
Description: install an extension
Syntax:
CREATE EXTENSION [ IF NOT EXISTS ] extension_name
    [ WITH ] [ SCHEMA schema_name ]
             [ VERSION version ]
             [ CASCADE ]
```

URL: https://www.postgresql.org/docs/16/sql-createextension.html

Postgres fun fact - default extension.

PL/pgSQL is a loadable procedural language for the PostgreSQL database system that is default extension created with each new database.

```
postgres=# load 'plpgsql';
LOAD
postgres=# \dx plpgsql
                 List of installed extensions
  Name   | Version |   Schema   |         Description
---------+---------+------------+------------------------------
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(1 row)

postgres=# do language plpgsql 'begin end';
DO
```

More supported Procedural Language as extensions

- PL/R - *PostgreSQL support for R as a procedural language (PL)*
- PL/V8 - *A procedural language in JavaScript powered by V8*
- PL/Tcl - *Tcl procedural language for PostgreSQL.*
- PL/Perl - *The Perl procedural language for PostgreSQL.*
- PL/Rust - *Procedural language in the Rust programming.*
- PL/Python - *Untrusted procedural language for PostgreSQL.*

More..

pg_stat_statements

Track statistics of SQL planning and execution

```
WITH statements AS (
SELECT * FROM pg_stat_statements pss
where pss.dbid in (SELECT oid from pg_database where datname=current_database())
and query like '%pgbench%'
)
SELECT calls,
        mean_exec_time,
        query
FROM statements
ORDER BY calls DESC
LIMIT 3;
-[ RECORD 1 ]--+-----------------------------------------------------
calls          | 225
mean_exec_time | 0.3473672000000002
query          | SELECT abalance FROM pgbench_accounts WHERE aid = $1
-[ RECORD 2 ]--+-----------------------------------------------------
calls          | 1
mean_exec_time | 0.0208
query          | select count(*) from pgbench_branches
```

pg_stat_statements - toplevel to track nested calls.

```
extension=# CREATE OR REPLACE FUNCTION public.sampleproceduralblock(id integer) RETURNS boolean
extension-#    LANGUAGE plpgsql
extension-#    AS $$
extension$# DECLARE
extension$# cnt bigint ;
extension$# BEGIN
extension$#     select count(1) into cnt from pgbench_accounts where bid = id;
extension$#     if cnt > 0 then
extension$#        return true;
extension$#     else
extension$#        return false;
extension$#     end if;
extension$# END $$;
CREATE FUNCTION
Time: 214.587 ms
extension=# explain analyze select public.sampleproceduralblock(col1) from generate_series(1,25) col1;
```

Problematic SQL!

pg_stat_statements - toplevel to track nested calls.

**pg_stat_statements.track = 'all' (only for non prod or specific session)*

```
select query , total_exec_time , toplevel  from pg_stat_statements where
query like 'select%sampleproceduralblock%' and toplevel;
-[ RECORD 1 ]---+-------------------------------------------------------------------
--------
query           | select public.sampleproceduralblock(col1) from generate_series($1,
$2) col1
total_exec_time | 11203.602380999999
toplevel        | t

Time: 235.243 ms
select query , total_exec_time , toplevel  from pg_stat_statements where
not toplevel;
-[ RECORD 1 ]---+-------------------------------------------------------------
query           | select count($2)          from pgbench_accounts where bid = id
total_exec_time | 11200.898884
toplevel        | f
```

Identify problematic sql within procedural block.

pg_hint_plan - Influence sql performance

Makes it possible to tweak PostgreSQL execution plans using so-called "hints" in SQL comments

```
/*+SeqScan(t2)*/
EXPLAIN (COSTS false) SELECT * FROM t1, t2 WHERE t1.id = t2.id;
                QUERY PLAN
-----------------------------------
 Merge Join
   Merge Cond: (t1.id = t2.id)
   ->  Index Scan using t1_pkey on t1
   ->  Sort
         Sort Key: t2.id
         ->  Seq Scan on t2
(6 rows)
```

Influence access path and Join method

hypopg - Invisible Indexes

PostgreSQL extension adding support for hypothetical indexes

```
SELECT * FROM hypopg_create_index('CREATE INDEX ON hypo_sample (id)') ;
 indexrelid |          indexname
------------+-----------------------------
      13543 | <13543>btree_hypo_sample_id
(1 row)

Time: 242.272 ms
EXPLAIN (COSTS OFF) SELECT val FROM hypo_sample WHERE id = 1;
                          QUERY PLAN
-----------------------------------------------------------------
 Index Scan using "<13543>btree_hypo_sample_id" on hypo_sample
   Index Cond: (id = 1)
(2 rows)
```

Validate and test indexes before actually creating it.

orafce - Oracle’s compatibility functions and packages

Emulate a subset of functions and packages from the Oracle RDBMS.

```
\df add_months
                                    List of functions
 Schema |    Name    |      Result data type       |        Argument data types        | Type
--------+------------+-----------------------------+-----------------------------------+------
 oracle | add_months | date                        | day date, value integer           | func
 oracle | add_months | timestamp without time zone | timestamp with time zone, integer | func
(2 rows)

SELECT NVL(1,2) AS NVL  , INSTR('POSTGRESQL ON MYDBOPS', 'SQL', 1) AS INSTR , DECODE(3, 1, 100, 2, 200,0) AS DECODE ;
 nvl | instr | decode
-----+-------+--------
   1 |     8 |      0
(1 row)

SELECT ADD_MONTHS(CLOCK_TIMESTAMP(),1) AS ADD_MONTHS ;
     add_months
---------------------
 2024-07-07 15:00:38
(1 row)

SELECT DBMS_RANDOM.RANDOM() AS RANDOM ;
   random
------------
 1873518366
(1 row)

SELECT COUNT(1) AS OUTPUT FROM DBA_SEGMENTS;
 output
--------
    288
```

oracle_fdw - Foreign Data Wrapper for Oracle

Facilitate seamless data migration and integration between Oracle databases and PostgreSQL.

```
postgres=> create extension oracle_fdw;
CREATE EXTENSION
postgres=> CREATE SERVER oradb1 FOREIGN DATA WRAPPER oracle_fdw
          OPTIONS (dbserver '//          :1521/  pdb1' );
CREATE SERVER
postgres=> CREATE USER MAPPING FOR postgres SERVER oradb1
          OPTIONS (user '       ', password '        ');
CREATE USER MAPPING
postgres=> IMPORT FOREIGN SCHEMA "SYS" LIMIT TO (DUAL)
    FROM SERVER oradb1 INTO public;
IMPORT FOREIGN SCHEMA
postgres=> select * from dual;
 dummy
-------
 X
(1 row)
```

pgtt - Temporary Table across schema

Use Oracle-style Global Temporary Tables and the others RDBMS.

```
postgres=> \dx pgtt
                                  List of installed extensions
 Name | Version |   Schema    |                          Description
------+---------+-------------+-----------------------------------------------------------------
 pgtt | 3.0.0   | pgtt_schema | Extension to add Global Temporary Tables feature to PostgreSQL
(1 row)

postgres=> load 'pgtt';
LOAD
postgres=> CREATE /*GLOBAL*/ TEMPORARY TABLE TMP1(COL1 INTEGER);
CREATE TABLE
postgres=> insert into TMP1 select 1;
INSERT 0 1
postgres=>
```

Session 1

```
postgres=> load 'pgtt';
LOAD
postgres=> table tmp1;
 col1
------
(0 rows)

postgres=> insert into TMP1 select 2;
INSERT 0 1
postgres=> table tmp1;
 col1
------
    2
(1 row)
```

Session 2

pgaudit - Compliance and Regulatory

provides detailed session and/or object audit logging via the standard PostgreSQL logging facility.

```
                    List of installed extensions
  Name   | Version | Schema |           Description
---------+---------+--------+--------------------------------
 pgaudit | 16.0    | public | provides auditing functionality
(1 row)

extension=> \dconfig pgaudit.*
      List of configuration parameters
           Parameter             |   Value
--------------------------------+------------
 pgaudit.log                    | write, ddl
 pgaudit.log_catalog            | on
 pgaudit.log_client             | off
 pgaudit.log_level              | log
 pgaudit.log_parameter          | off
 pgaudit.log_parameter_max_size | 0
 pgaudit.log_relation           | off
 pgaudit.log_rows               | off
 pgaudit.log_statement          | on
 pgaudit.log_statement_once     | off
 pgaudit.role                   |
```

pgaudit

Configuring custom logging based on compliance requirement.

```
extension=> alter user test1 set pgaudit.log = 'ddl';
ALTER ROLE
extension=> alter user test2 set pgaudit.log = 'read,write';
ALTER ROLE
extension=> \drds test1|test2
             List of settings
 Role  | Database |        Settings
-------+----------+------------------------
 test1 |          | pgaudit.log=ddl
 test2 |          | pgaudit.log=read,write
(2 rows)
```

```
DROP TABLE
CREATE TABLE
INSERT 0 1
```

```
ension:

t logged>

ension:

 id int,
>

ension:
o
user1',
```

Table Partitioning Maintenance – pg_partm

```
partitionsample=>
```

PL/pgSQL Conversion - Challenge

PL/pgSQL functions aren't syntax-checked until executed.

```
postgres=# CREATE OR REPLACE FUNCTION FUNC_DEMO1()
postgres-# RETURNS void
postgres-# LANGUAGE plpgsql
postgres-# AS
postgres-# $$
postgres$# DECLARE VAR1 INTEGER;
postgres$# BEGIN
postgres$# SELECT 1 INTO VAR1 WHERE COL1 = 1;
postgres$# END;
postgres$# $$;
CREATE FUNCTION
```

```
postgres=# SELECT func_demo1();
ERROR:  column "col1" does not exist
LINE 1: SELECT 1      WHERE COL1 =
1

QUERY:  SELECT 1
                WHERE COL1 = 1
CONTEXT:  PL/pgSQL function
func_demo1() line 4 at SQL statement
```

plpgsql_check extension to rescue.

Extension serves as a comprehensive linter for plpgsql in Postgres

- Utilizes the internal PostgreSQL parser/evaluator to display runtime errors.
- Parses SQL inside routines to identify errors not typically found during "CREATE PROCEDURE/FUNCTION" commands.
- The plpgsql_check extension detects issues in PL/pgSQL code: undefined/unused variables, type mismatches, control flow errors, incorrect function calls, trigger problems, and SQL statement errors.

Some more essential extensions

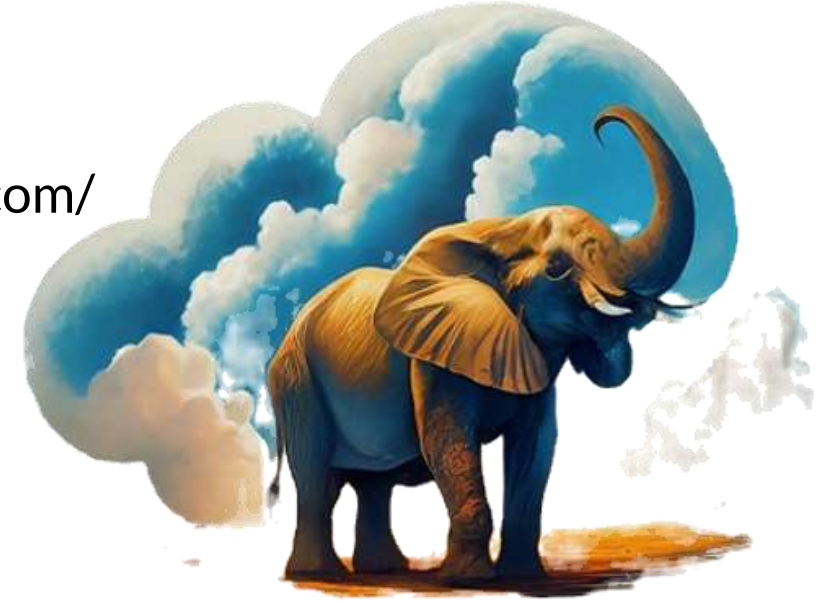
- pg_repack - lets you remove bloat from tables and indexes as online.
- pglogical - PostgreSQL Logical Replication - Change Data Capture.
- hydra_columnar - Analytics- Columnar storage for Postgres
- auto_explain - logging execution plans of slow statements automatically.
- postgresql_anonymizer - Anonymization & Data Masking.
- passwordcheck* - Checks and rejects weak passwords.

Thank you!

https://www.linkedin.com/in/mahtodeepak/

https://x.com/mahtodeepak05

https://databaserookies.wordpress.com/



https://www.datacloudgaze.com/