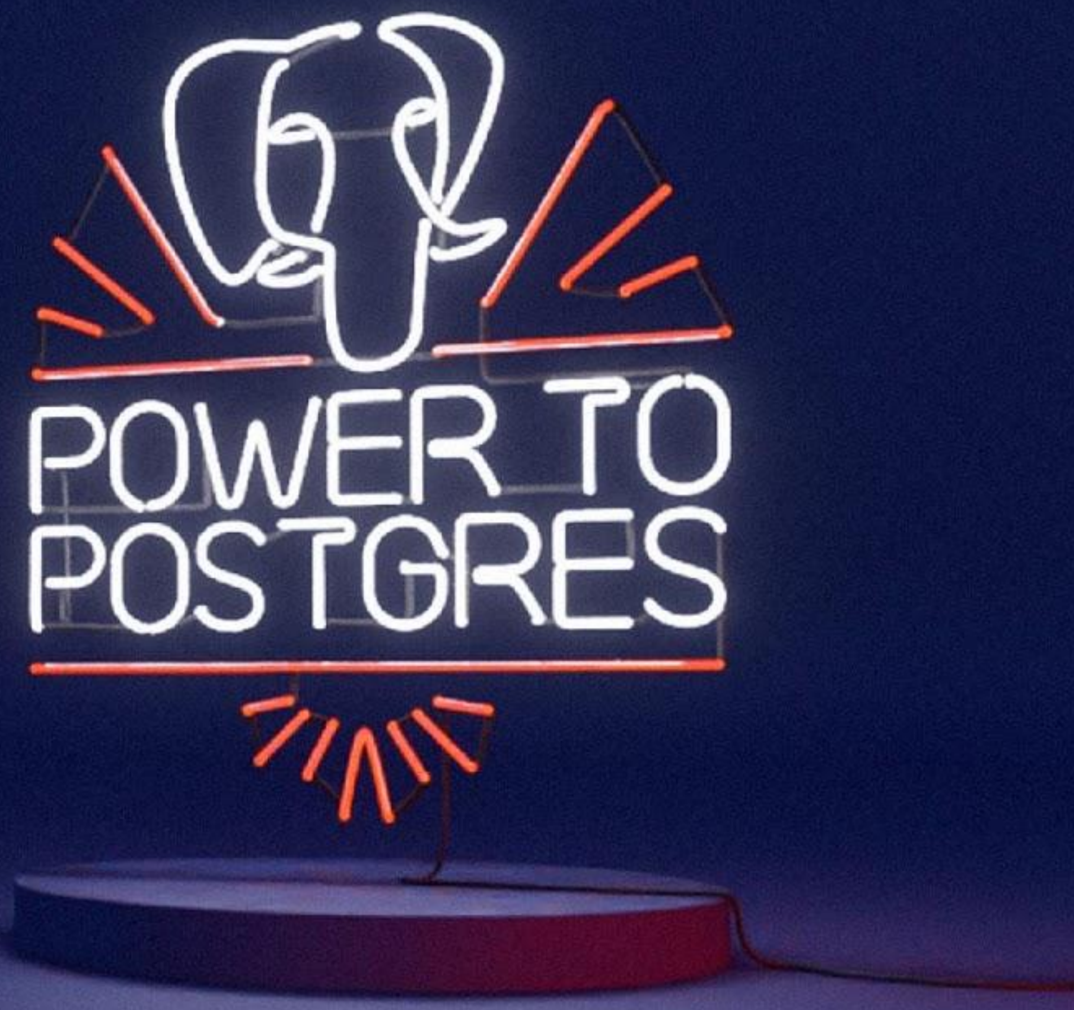
Advanced Server 14 Highlights

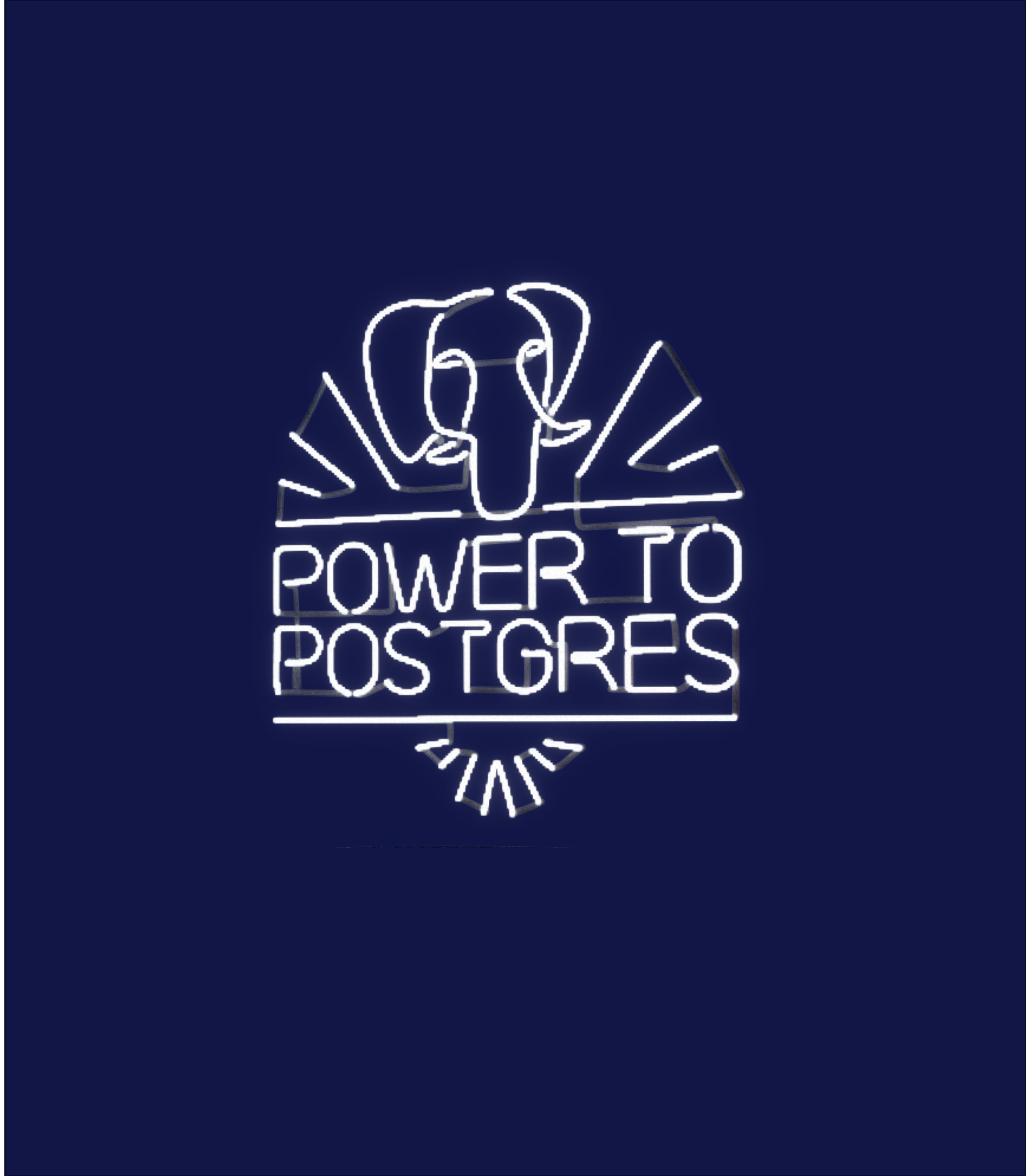
Robert Haas

October 27, 2021



Agenda

- Advanced Server 14 Features
- Full Support for Postgres-BDR
- PostgreSQL 14 Features



Advanced Server 14 Features

Overview

- **Object-Level Auditing**. Also, some other cool auditing improvements!
- **Object Creation and Last DDL Time**.
- **Full support for CONNECT BY**.
- **Subpartitioning Enhancements**. Subpartition templates, and more.
- **\h and Tab Completion for EDB Syntax Additions**.

Object-Level Auditing

- In existing versions, Advanced Server lets you choose the types of statements that you'd like to audit:

  ```
  logging_collector=on
  edb_audit=xml (or csv)
  edb_audit_statement='insert, update, delete'
  ```

- In Advanced Server 14, you can choose to audit select, insert, update, and delete operations on specific tables:

  ```
  alter table receivables set (edb_audit_group = 'finance');
  alter system set edb_audit_statement =
      'insert@finance, update@finance, delete@finance';
  ```

Object-Level Auditing (2)

- In Advanced Server 14, you can also choose to audit operations *except for* those on specific tables:

```
alter table sock_drawer set (edb_audit_group = 'boring');
alter system set edb_audit_statement =
    'insert-boring, update-boring, delete-boring';
```

- Inclusion (@) and exclusion (–) operators can be used together, e.g.:

```
edb_audit_statement = 'insert@finance, update-boring'
```

Audit Control by User and Database

- Since Advanced Server 10, auditing rules can be different by user and by database:

```
alter user sketchy set edb_audit_statement = 'all';
alter database sensitive set edb_audit_statement = 'all';
```

- The flexibility to set different rules for different users or databases can be combined with the new per-object auditing syntax:

```
alter system set edb_audit_statement =
'select@finance, insert@finance, update@finance, delete@finance';
alter user cfo set edb_audit_statement = 'delete@finance';
```

Audit Log Rotation

- There is now an optional feature, disabled by default, that can compress or expire audit log files after a certain period of time:

```
edb_audit_archiver = 'on'
edb_audit_archiver_compress_time_limit = '1d'
edb_audit_archiver_expire_time_limit = '7d'
```

- There are many new, related settings in `postgresql.conf` that can be used to get the exact compression and rotation behavior that you need.

Object Creation and Last DDL Time

- The system now automatically records the creation time and the last DDL time for every object in the system - tables, functions, schemas, everything!

- You can access this information via the `all_objects` view:

```
edb=# select created, last_ddl_time from all_objects
edb-# where object_name = 'FOO';
            created             |          last_ddl_time
---------------------------------+---------------------------------
 21-SEP-21 15:25:07.964559 -04:00 | 21-SEP-21 15:25:07.964559 -04:00
(1 row)
```

Full Support for CONNECT BY

- In previous releases of Advanced Server, `CONNECT BY` queries were supported only if they were of the form `CONNECT BY x = PRIOR y` or alternatively `CONNECT BY PRIOR x = y`.

- Beginning in Advanced Server 14, you can `CONNECT BY` any expression. For example, you can `CONNECT BY level <= 10`, or you can `CONNECT BY x = PRIOR y AND z = t`.

- `PRIOR` is now supported in the target list, so you can write queries like:

```
SELECT empno, ename, PRIOR ename AS mgr FROM emp
START WITH mgr IS NULL CONNECT BY mgr = PRIOR empno ORDER BY 1, 2;
```

Subpartition Templates + References

- ```
  CREATE TABLE example (col1 NUMBER, col2 NUMBER)
  PARTITION BY RANGE (col1) SUBPARTITION BY RANGE (col2)
  SUBPARTITION TEMPLATE (
      SUBPARTITION s1 VALUES LESS THAN (10),
      SUBPARTITION s2 VALUES LESS THAN (20)
  )
  (
      PARTITION p1 VALUES LESS THAN (100),
      PARTITION p2 VALUES LESS THAN (200)
  );
  ```

- `SELECT * FROM example SUBPARTITION (p1_s1);`
  ```

\h and Tab Completion for EDB Syntax Additions

```
edb=# \h create package
Command:     CREATE PACKAGE
Description: define a new package
Syntax:
CREATE [ OR REPLACE ] PACKAGE package_name
[ AUTHID { DEFINER | CURRENT_USER } ]
{ IS | AS }
[ PRAGMA RESTRICT_REFERENCES (name, { RNDS | RNPS | TRUST | WNDS | WNPS } [, ... ] ); ]
  [ declaration; ] [, ... ]
  [ {  PROCEDURE proc_name [ ( [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ... ] ) ]
     | FUNCTION func_name [ ( [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ... ] ) ]
       RETURN rettype [ DETERMINISTIC ];
     | TYPE type_name IS
        { [ [ TABLE | VARRAY(size) | VARYING ARRAY(size) ] OF data_type [ INDEX BY data_type ] ]
          | [ RECORD (variable_name data_type) ]
          | [ REF CURSOR [RETURN rettype] ]
        }
     | SUBTYPE type_name IS data_type [ NULL | NOT NULL ];
     | CURSOR cur_name [ ( [ argmode ] argname argtype ) ] IS query;
    }
  ] [ ... ]
  END [ package_name ];
```

Full Support for Postgres-BDR

Postgres-BDR & Advanced Server

- Postgres-BDR is EDB's Bi-Directional Replication product, allowing for multi-master logical replication with advanced conflict management.

- Postgres-BDR is a server extension. It can be used with PostgreSQL version 11 or higher, Advanced Server version 11 or higher, and with EDB Postgres Extended (formerly 2ndQPostgres).

- As a server extension, Postgres-BDR has limited ability to modify server behavior. However, there are some very desirable features that can only be unlocked with additional server functionality.

- The necessary server changes are included in all versions of EDB Postgres Extended, and in Advanced Server version 14.

- While this is not a BDR presentation, we'll briefly review the additional BDR functionality that is unlocked by the addition of these server capabilities to Advanced Server 14.

Postgres-BDR Features Unlocked (1 of 2)

- **Commit At Most Once**. If your application loses the connection to the database server while COMMIT is in progress, it might be hard to know whether or not the `COMMIT` was processed. With this feature, your application can find out what happened, even if your new database connection is to a different node than your previous connection.

- **Eager Replication**. This is an optional feature to avoid replication conflicts. Every transaction is applied on all nodes simultaneously, and commits only if no replication conflicts are detected. This feature does reduce performance, but provides very strong consistency guarantees.

- **Hold Back Freezing**. BDR tries to ensure that transactions touching the same data are committed in the same order as they were on the origin nodes. However, if `VACUUM` freezes the rows too soon, commit timestamp data may be lost, leading to incorrect ordering in some cases where the replication conflict involves at least one `DELETE`. This infrastructure avoids that problem.

Postgres-BDR Features Unlocked (2 of 2)

- **Decoding Worker**. In PostgreSQL and in previous versions of Advanced Server, a single process decodes the write-ahead log (WAL) to identify logical changes and also sends those changes to the remote node. With this enhancement, there is a separate process which only performs decoding, and the decoded changes can then be sent to every node in the cluster. In large clusters, this can reduce CPU and memory consumption significantly.

- **Timestamp-based Snapshots**. This feature provide a globally consistent view of the cluster as it exists as of some moment in time, either now or in the recent past. The snapshot includes all transactions that committed on any node prior to the chosen timestamp, and nothing that committed afterwards.

PostgreSQL 14 Features

EPAS 14 is based on PostgreSQL 14

- All new features added to PostgreSQL 14 are also present in EPAS 14.
- The complete feature list is quite long, and going through every change in detail would take quite some time!
- In this presentation, I’ll just tell you about a few of the best features of PostgreSQL 14 that will also be available in EPAS 14.

Developer Features

- **JSON Subscripting**. Queries like `SELECT ('{ "postgres": { "release": 14 }}'::jsonb)['postgres']['release']` now work.
- **Multi-Ranges**. PostgreSQL has supported range types since version 9.2. PostgreSQL 14 adds multi-ranges, which are can store multiple ranges in a single value. For example, something like `SELECT '{[3,7), [8,9)}'::int4multirange;`
- **`ALTER TABLE ... DETACH PARTITION ... CONCURRENTLY`**.
- **libpq Pipelining**. If you use libpq to connect to the database, you can modify your application to send several queries at once, instead of waiting for the results of one query before sending the next.

Performance Improvements

- **Snapshot Scalability**. To support MVCC, PostgreSQL takes “snapshots” frequently. This can scale poorly when the number of active connections is very large, especially on systems with many CPU cores. It v14 it will scale better.

- **Bottom-Up btree Index Deletion**. When PostgreSQL doesn’t clean up old row versions fast enough, tables and indexes become “bloated.” In v14, there is a new opportunistic cleanup strategy for btree indexes. When it applies, it can be very effective in preventing index bloat. It helps mostly with non-HOT updates where the same rows are updated repeatedly.

- **Partition Pruning for UPDATE and DELETE**. Partition pruning is now effective for UPDATE and DELETE queries, whereas in previous releases it mostly only worked for SELECT queries.

- **Logical Replication**. There are various improvements to logical replication, including the ability to stream transactions to subscribers while they are still in progress, which can improve latency for large transactions.

Thank you!

Any questions?