POSTGRESQL is an open-source, full-featured relational database. This presentation gives an overview of the Postgres 12 release.

https://momjian.us/presentations

Creative Commons Attribution License
1. Partitioning improvements
2. Btree improvements
3. Multi-column most-common-value statistics
4. Inline many CTE queries
5. Prepared plan control
6. Just-in-time Compilation
7. Checksum Control
8. REINDEX CONCURRENTLY

Full item list at https://www.postgresql.org/docs/12/index.html
1. Partitioning Improvements

- Thousands of partitions now efficiently processed
- Partitioned tables can now be referenced as foreign keys
- Improve `COPY` into partitioned tables
- Partition bounds can now be expressions
- New partition introspection SQL functions
2. Btree Improvements

- Reduce multi-column index size by using space more efficiently
- Improve performance of indexes with many duplicates
- Allow VACUUM to more efficiently remove tuples from indexes with many duplicates
- Reduce locking requirements during index updates
3. Multi-Column Most-Common-Value Statistics

Allow most-common-value statistics for multiple columns; previously only a single correlation value was recorded for multiple columns.

```
CREATE STATISTICS stts3 (mcv) ON city, state FROM zipcodes;
```

```
ANALYZE zipcodes;
```

```
SELECT m.* FROM pg_statistic_ext,
pg_mcv_list_items(stxmcv) m WHERE stxname = 'stts3';
```
Many common table expressions (CTE) can now be inlined:

-- PG 11
EXPLAIN WITH t(x) AS (SELECT 1) SELECT * FROM t;
QUERY PLAN

CTE Scan on t  (cost=0.01..0.03 rows=1 width=4)
  CTE t
    ->  Result  (cost=0.00..0.01 rows=1 width=4)

-- PG 12
EXPLAIN WITH t(x) AS (SELECT 1) SELECT * FROM t;
QUERY PLAN

Result  (cost=0.00..0.01 rows=1 width=4)
5. Prepared Plan Control

Prepared statements usually use generic/prepared plans after six executions:

```sql
PREPARE p (INTEGER) AS
SELECT relname FROM pg_class WHERE oid = $1;

EXPLAIN EXECUTE p (1);
```

```
QUERY PLAN
-------------------------------------------------
Index Scan using pg_class_oid_index on pg_class …
  Index Cond: (oid = '1'::oid)
```

```
EXPLAIN EXECUTE p (1);
```

```
QUERY PLAN
-------------------------------------------------
Index Scan using pg_class_oid_index on pg_class …
  Index Cond: (oid = '1'::oid)
```
PREPARED PLAN CONTROL

EXPLAIN EXECUTE p (1);

QUERY PLAN

---------------------------------------------
Index Scan using pg_class_oid_index on pg_class ...
  Index Cond: (oid = '1'::oid)

EXPLAIN EXECUTE p (1);

QUERY PLAN

---------------------------------------------
Index Scan using pg_class_oid_index on pg_class ...
  Index Cond: (oid = '1'::oid)
EXPLAIN EXECUTE p (1);

QUERY PLAN

-----------------------------------------------
Index Scan using pg_class_oid_index on pg_class ...
  Index Cond: (oid = '1'::oid)

EXPLAIN EXECUTE p (1);

QUERY PLAN

-----------------------------------------------
Index Scan using pg_class_oid_index on pg_class ...
  Index Cond: (oid = ($1)::oid)
**plan_cache_mode**

`plan_cache_mode` allows users to force always-custom or always-generic plans:

```sql
DEALLOCATE p;

PREPARE p (INTEGER) AS SELECT relname FROM pg_class WHERE oid = $1;

SET plan_cache_mode = force_generic_plan;

EXPLAIN EXECUTE p (1);
```

```
QUERY PLAN
--------------------------------------------------
Index Scan using pg_class_oid_index on pg_class ...
  Index Cond: (oid = ($1)::oid)
```
6. Just-in-Time Compilation

- Enable JIT by default
- Useful for data warehouse queries
7. Checksums Control

- Allow a cluster’s checksum mode to be changed while it is offline
- `pg_checksums --enable --progress`
- Online change control planned
Like `CREATE INDEX CONCURRENTLY`, this allows `REINDEX` with minimal locking, specifically, just before completion:

```sql
CREATE TABLE test (x INTEGER);

INSERT INTO test SELECT generate_series(1, 1000);

CREATE INDEX i_test ON test (x);

REINDEX INDEX CONCURRENTLY i_test;
```
Conclusion

https://momjian.us/presentations

https://www.flickr.com/photos/thomasletholsen/