PostgreSQL 19 への展望

- 開発中機能の紹介 -

2026-02-28　OSC Tokyo/Spring 2026

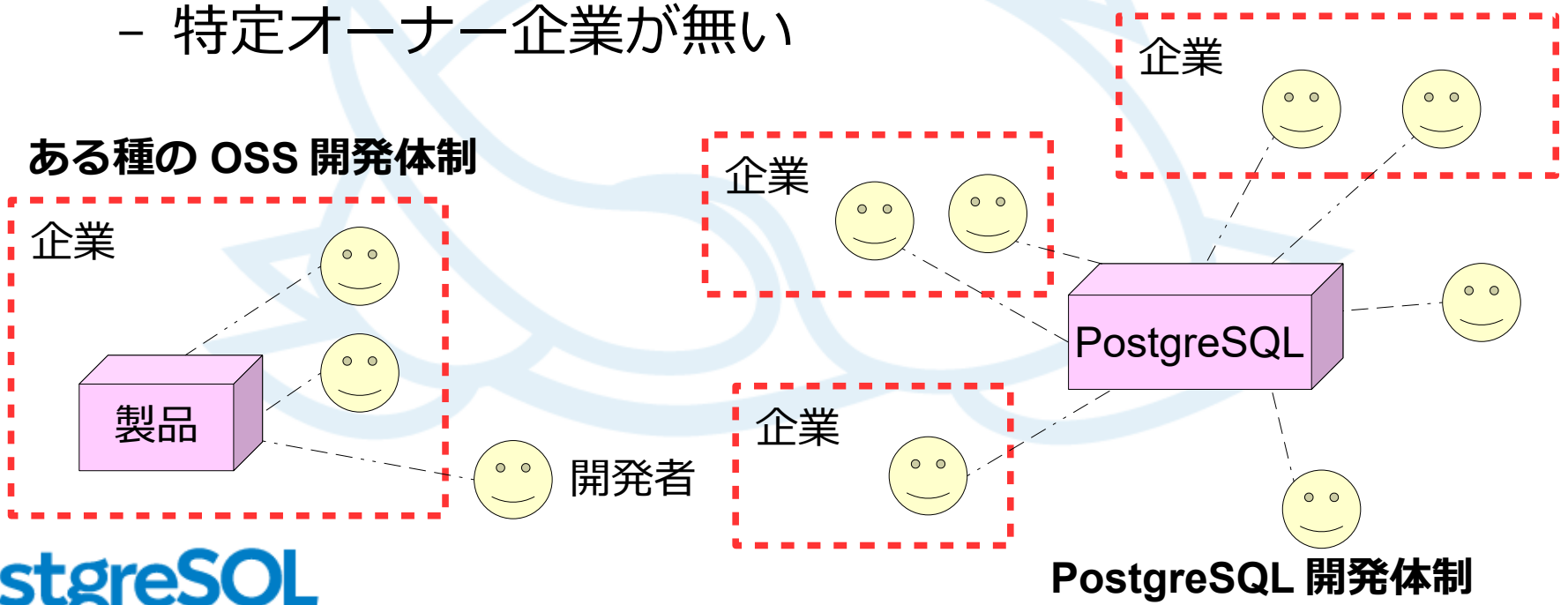
日本 PostgreSQL ユーザ会
理事 高塚 遥

日本 PostgreSQL ユーザ会 (JPUG) とは

- PostgreSQL の普及を促進し、技術情報を公開し、ユーザ相互の情報交換を支援する
  - メーリングリストや Slack スペースの運営
  - 勉強会、カンファレンスの主催や開催支援
  - マニュアル翻訳
  - PostgreSQL グローバル開発グループ（ PGDG ）の日本における advocacy 代表窓口

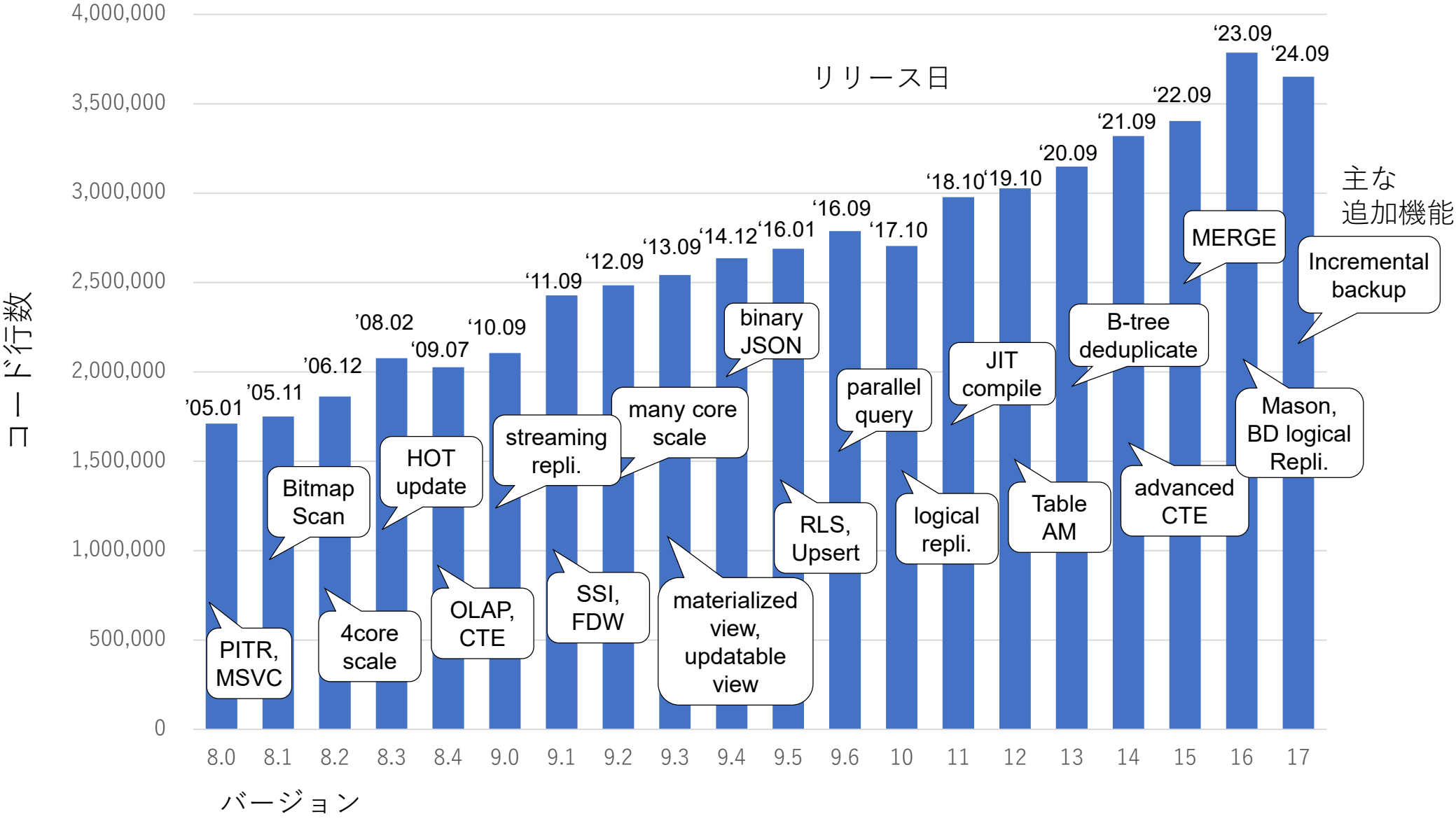
PostgreSQL とは

- 多機能、高性能、かつオープンソースの
  リレーショナルデータベース管理システム ソフトウェア
  - INGRES('70), POSTGRES('80) 由来の長い歴史
  - BSD タイプのライセンス
  - 特定オーナー企業が無い



PostgreSQL
http://www.postgresql.jp

PostgreSQL これまでのリリース



PostgreSQL19 開発状況

Commitfests

	Details	When Open	When In Progress
Open:	PG19-Final	Now – 2026-02-28	2026-03-01 – 2026-03-31
In Progress:	None in progress		
Previous:	PG19-4	Not open anymore	2026-01-01 – 2026-01-31

- 何回かの Commitfests で機能追加パッチをコミット
- 毎年春に仕様凍結（Final Commitfests で）
- 夏～秋に beta 版、rc 版をリリース
- 秋～冬に正式版リリース

主な候補新機能 (1)

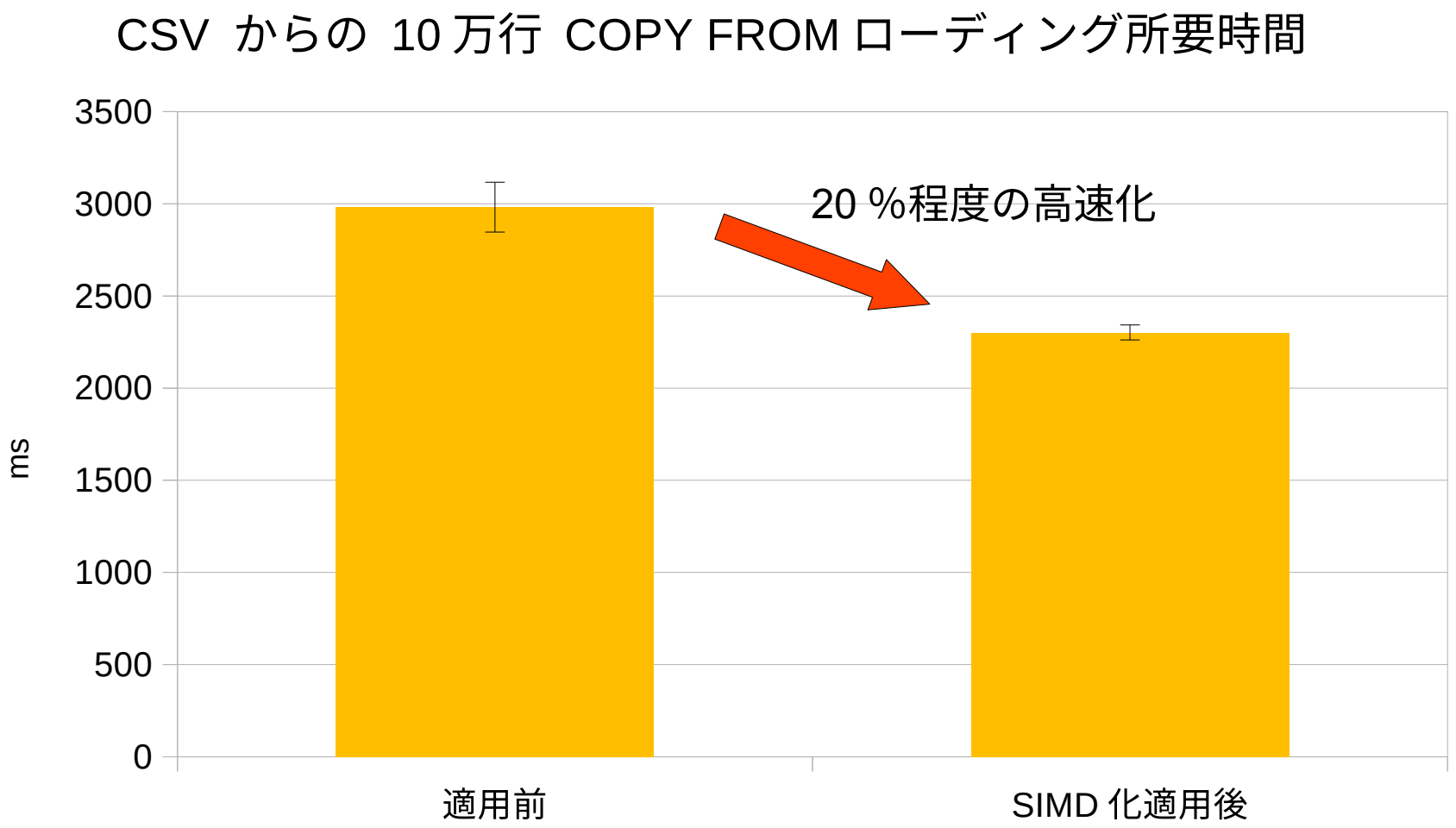
- Property Graph Queries (SQL/PGQ)
- 行パターン認識
- Incremental Vew Maintenance (リアルタイム更新マテビュー)
- マテリアライズドビューの部分リフレッシュ
- LET コマンド – セッション変数の実現
- パーティションの分割・マージ
- REPACK コマンド – pg_repack 的な機能
- pg_plan_advice – pg_hint_plan 的な機能
- CREATE SUBSCRIPTION ... SERVER による指定
- SQL:2023 JSON シンプルアクセッサ
- ON CONFLICT DO SELECT

主な候補新機能 (2)

- タプルの基数ソート
- 8 バイト TOAST 値
- shared_buffers オンライン変更
- wal_level=logical のオンライン変更
- ステートメントログでパラメータ数上限設定
- SQL:2023 JSON シンプルアクセッサ
- フルページに限らない WAL 圧縮
- COPY FROM の SIMD 化
- バッファページ =OS ページ =NUMA の観測
- シーケンスの Access Method

他にも多数 : https://commitfest.postgresql.org/58/

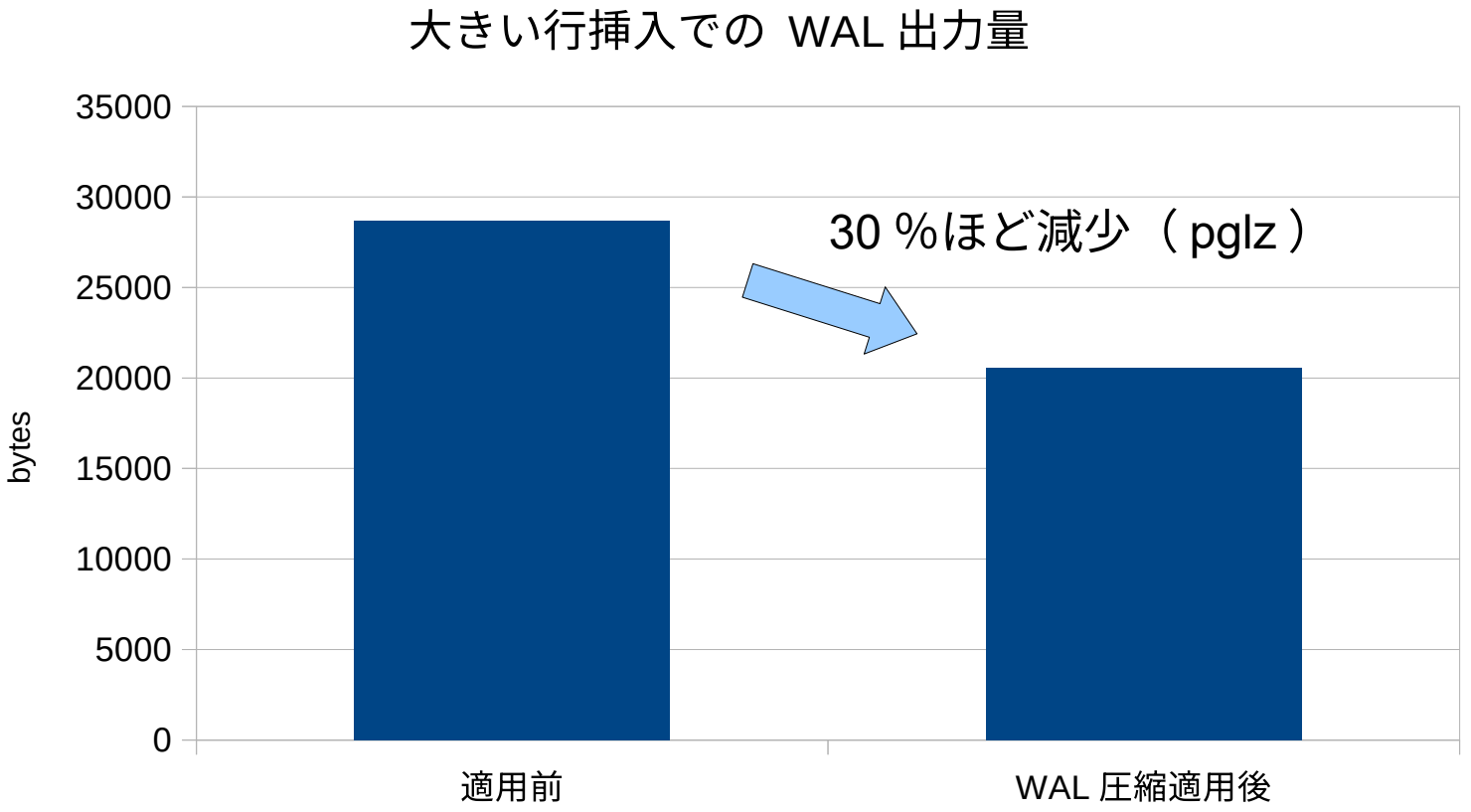
COPY FROM の SIMD 化（性能向上）



- 数値、テキスト、タイムスタンプ型で計 30 列のテーブルで一行あたり 300 バイト程度 × 10 万行

WAL 圧縮（フルページに限らない）

- フルページ書き込み（＝バッファページをそのまま書き出す場合）については、以前から WAL 圧縮が行なわれていた
- それ以外でも設定した閾値より大きい WAL レコードを pglz 、lz4 、zstd で圧縮できる



圧縮余地のある
1 行 3500 バイト
× 10 行を挿入する
COPY FROM
を実行したときの
WAL 出力量

マテビューの部分リフレッシュ

```
db=# CREATE TABLE tscore (id int PRIMARY KEY,
       teamid int, score int, retire date);
db=# INSERT INTO tscore SELECT g, g/10, .. 中略..
       FROM generate_series(1, 1000000) g;
db=# CREATE MATERIALIZED VIEW vscore AS
       SELECT teamid, avg(score) FROM tscore
         WHERE retire IS NULL GROUP BY teamid;
db=# CREATE UNIQUE INDEX ON vscore (teamid);

db=# UPDATE tscore SET retire = CURRENT_DATE
       WHERE id = 11 AND teamid = 1;
db=# REFRESH MATERIALIZED VIEW vscore WHERE teamid = 1;
Time: 23.071 ms
db=# REFRESH MATERIALIZED VIEW vscore;
Time: 1271.332 ms
```

100 万行テーブルに対する集約と選択のマテビュー

部分リフレッシュで負荷を軽減

IVM: リアルタイム差分更新マテビュー

- Incremental View Maintenance

同じ 100 万行テーブルに MV 作成

```
db=# CREATE INCREMENTAL MATERIALIZED VIEW ivscore
       AS SELECT teamid, avg(score) FROM tscore
       WHERE retire IS NULL GROUP BY teamid;
NOTICE: created index "ivscore_index" on materialized view "ivscore"
```

```
db=# UPDATE tscore SET score = 0 WHERE teamid = 15;
UPDATE 10
Time: 202.622 ms

db=# UPDATE tscore SET retire = CURRENT_DATE
       WHERE id = 23 AND teamid = 2;
UPDATE 1
Time: 26.698 ms
```

元テーブルのデータ更新で MV も自動で部分更新

利用可能な SELECT 構文 / 関数に制限が数多くある：
外部結合、サブクエリ、 HAVING 、 WINDOW 、
count, sum, avg, min, max 以外の集約、など

CREATE VARIABLE / LET コマンド

- トランザクション制御外のセッション変数を実現
  - COMMIT/ABORT は影響しない
  - 商用 DB 製品の PL/SQL パッケージ変数の移植に

```
db=# CREATE TEMP VARIABLE v1 AS integer;
db=# LET v1 = 123;
db=# SELECT VARIABLE(v1);
 v1
-----
 123
(1 row)
```

REPACK コマンド

- 空き領域回収コマンド
- CLUSTER 、VACUUM FULL コマンドを統合
- CONCURRENTLY オプションで排他ロック（ ACCESS EXCLUSIVE ロック）取得範囲を最小化
- pg_repack 、pg_squeeze と似た実現方式
  - クリーンなコピーを作って、切り替え
  - 再構築中の変更差分はロジカルデコーディングで採取

```
db=# REPACK CONCURRENTLY tbl1 USING INDEX tbl1_idx1;
```

グラフ問い合わせ SQL/PGQ

- 表からグラフにマッピングして、グラフとして問合せ

```
db=# CREATE TABLE usr (id int PRIMARY KEY, name text);
db=# CREATE TABLE follows (id1 int, id2 int,
       PRIMARY KEY(id1, id2));

     《・・・適当にデータを充填・・・》

db=# CREATE PROPERTY GRAPH g1
       VERTEX TABLES (usr KEY (id) PROPERTIES (id, name))
       EDGE TABLES (follows
         SOURCE KEY (id1) REFERENCES usr(id)
         DESTINATION KEY (id2) REFERENCES usr(id));
```

グラフ問い合わせ SQL/PGQ

```
db=# SELECT * FROM GRAPH_TABLE (g1 MATCH
       (u1 IS usr)-[f1 IS follows]->(u2 IS usr)
                  -[f2 IS follows]->(u3 IS usr)
      WHERE u1.id = u3.id COLUMNS (u1.id, u2.id));
 uid  | uid
------+------
 1484 | 9103
 9103 | 1484
(2 rows)
```

相互フォローのユーザの
組を調べる問い合わせ

行パターン認識

- 時系列データに対して、値の変化のパターンを問い合わせ
  - 「下がって上がって下がる」
  - 「 10 回連続上昇」
  - 「 1000 円以下を起点として、さらに下がる」

dt	price
2025-01-01	1000
2025-01-02	1002
2025-01-03	992
2025-01-04	987
2025-01-05	997
2025-01-06	995
2025-01-07	1000
2025-01-08	1005
2025-01-09	996
2025-01-10	1000
2025-01-11	998
:	

```
SELECT dt, price,
  first_value(price) OVER w,
  max(price) OVER w,
  count(price) OVER w
FROM tprice WINDOW w AS (
  ORDER BY dt
  ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
  AFTER MATCH SKIP PAST LAST ROW
  INITIAL
  PATTERN (LOWPRICE UP+ DOWN+)
  DEFINE
    UP AS price > PREV(price),
    DOWN AS price < PREV(price),
    LOWPRICE AS price <= 1000,
);
```

「UP{10,}」
10 回連続上昇

「LOWPRICE DOWN+」
1000 以下からの下降

```
     dt      | price | first_value | max  | count
-------------+-------+-------------+------+-------
 2025-01-01 |  1000 |        1000 | 1002 |     4
 2025-01-02 |  1002 |             |      |     0
 2025-01-03 |   992 |             |      |     0
 2025-01-04 |   987 |             |      |     0
 2025-01-05 |   997 |             |      |     0
                          :
```

pg_plan_advice

```
db=# EXPLAIN (COSTS OFF, PLAN_ADVICE)
       SELECT * FROM tfact f
         JOIN tdim d ON f.dim_id = d.id;
             QUERY PLAN
-----------------------------------
 Hash Join
   Hash Cond: (f.dim_id = d.id)
   ->  Seq Scan on tfact f
   ->  Hash
         ->  Seq Scan on tdim d
 Generated Plan Advice:
   JOIN_ORDER(f d)
   HASH_JOIN(d)
   SEQ_SCAN(f d)
   NO_GATHER(f d)
```

「ヒント付与を助言」
（自動収集も可）
＋
「ヒント付与機能」

実行計画の
ヒント候補を
出力

実行計画の
ヒントを指定

```
db=# SET pg_plan_advice.advice = 'JOIN_ORDER(f d)';
```

PostgreSQL
http://www.postgresql.jp

オンライン設定変更

- wal_level　　（WAL に書き出す情報量）
  - ″logical″ レベル書き出しの有無の変更がオンラインで
    - 論理レプリケーションスロットが無ければ出力を自動縮減
- shared_buffers （共有バッファサイズ）
  - 本番運用後の設定チューニングに

従来はサービス再起動が必要であったもの

モニタリング機能の追加

- 共有バッファページ、OS ページ、NUMA ノード 対応付け
- 他バックエンドプロセスのメモリ使用を観測
- 累積的なロック統計　pg_stat_lock
- バックエンド毎の諸統計（ロック、コミット、CLUD ）

Committed なもの

2/27 17:00 JST 確認

- ON CONFLICT .. DO SELECT
- wal_level=logical のオンライン変更
- パーティションの分割・マージ

→ 本日紹介した大きい追加機能はいずれも未定

例年の傾向からの類推では PG19 に入るのは半分くらい