# 電力自由化を陰で支えるPostgreSQL

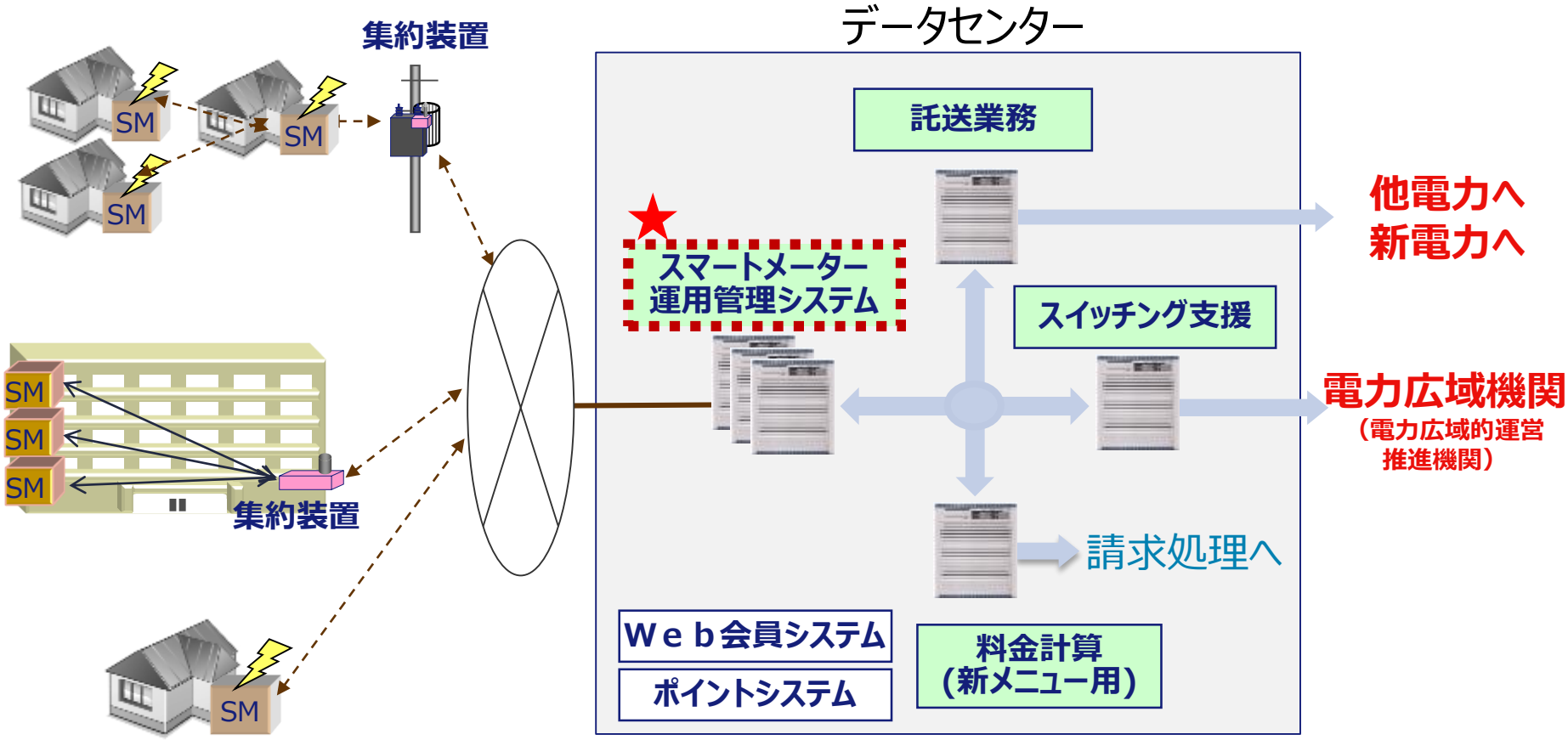
2016年12月2日
株式会社NTTデータ
システム技術本部

PGCONF.ASIA 発表資料

NTT DATA

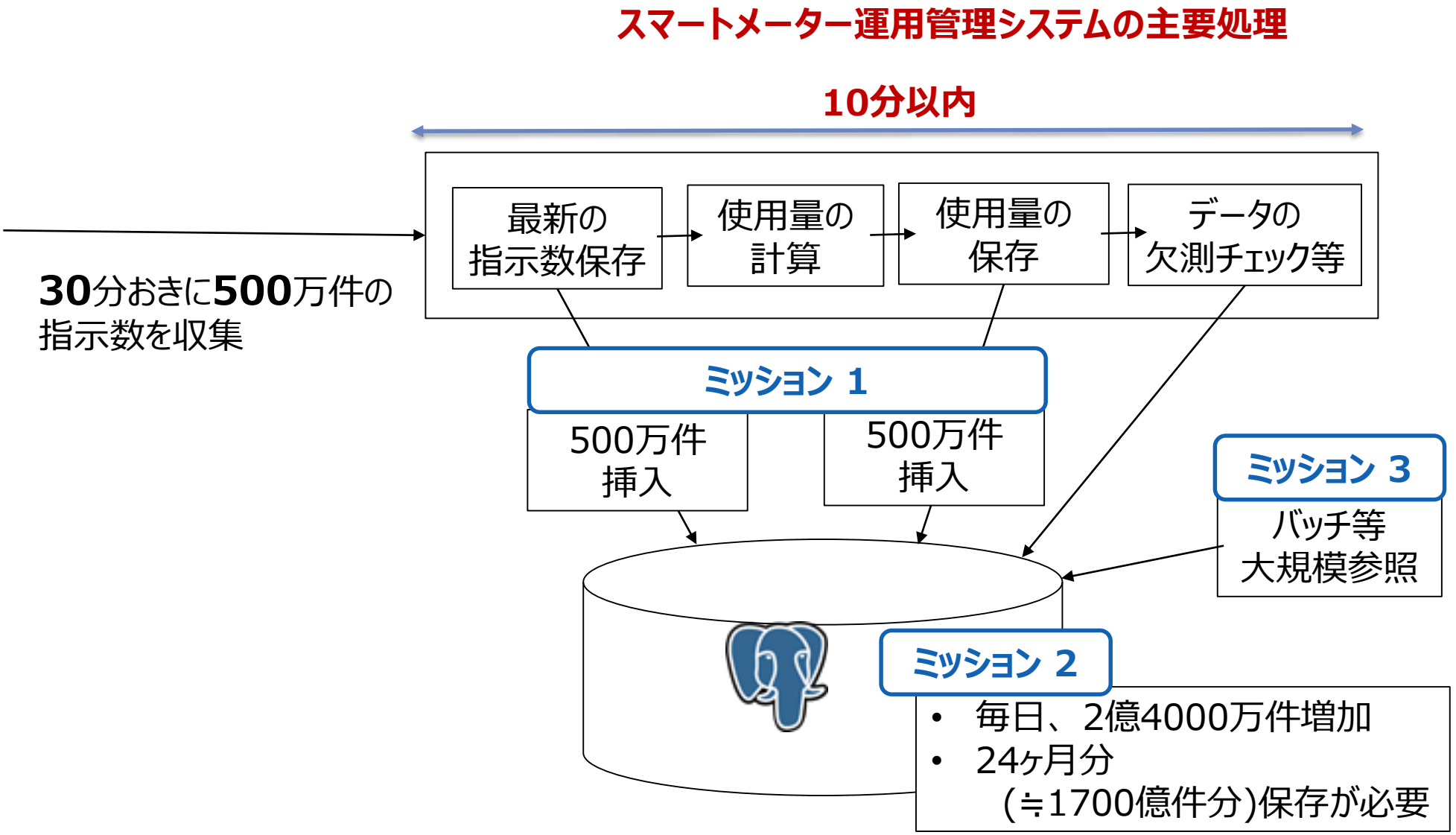
# 社会インフラへPostgreSQLを適用する道のり

# スマートメーター運用管理システムの位置づけ




SM:スマートメーター

# システムの主要処理とミッション



# ミッション

1. 1000万件データを10分以内にロードせよ！

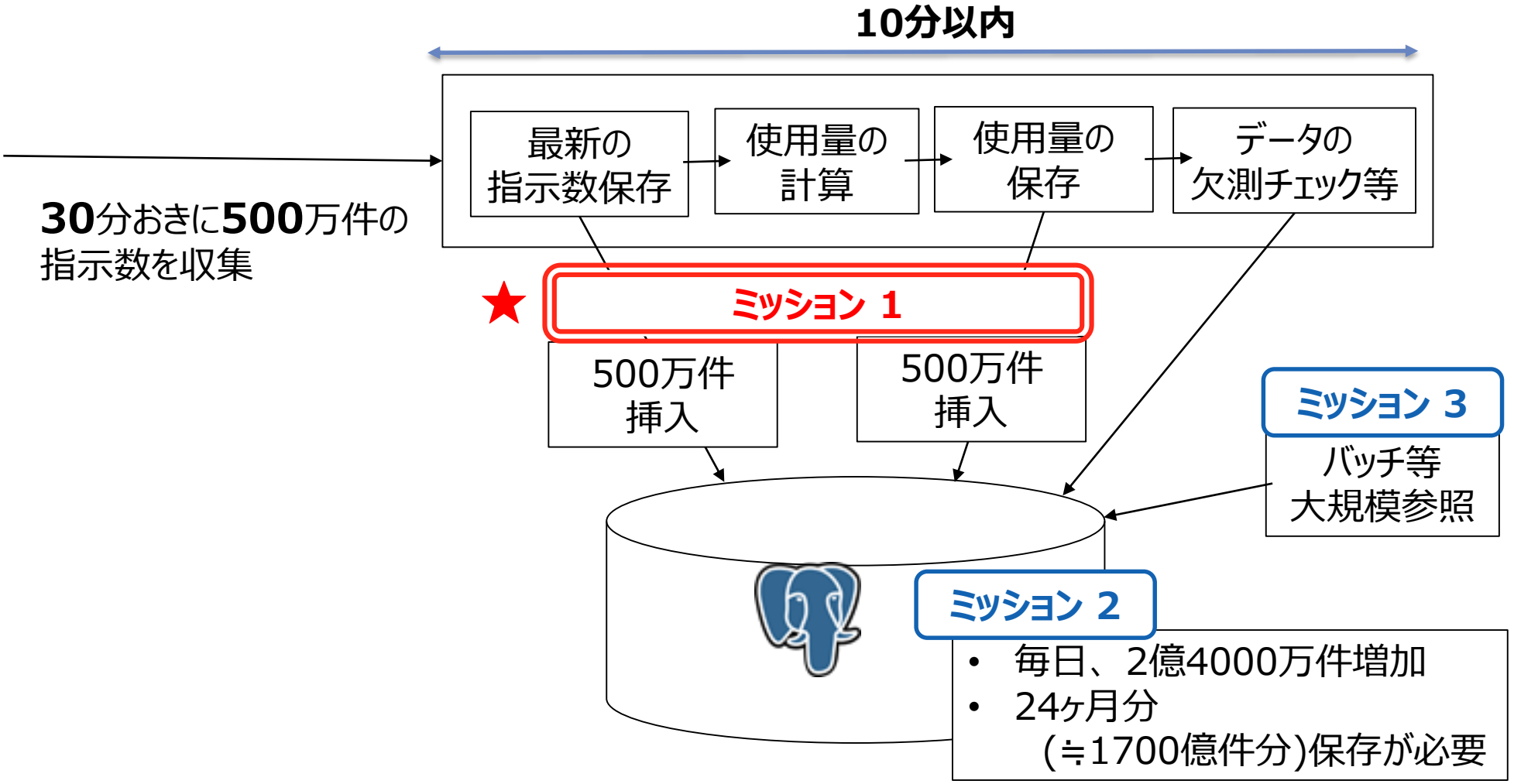
2. 24ヶ月分の大量データを長期保管せよ！

3. 大規模参照の性能を安定化せよ！

# (1)1000万件データを10分以内にロードせよ！

NTT DATA

## スマートメーター運用管理システムの主要処理



# 格納モデルの選択

NTT DATA

データは「**機器ID**」、「**日時**」、「**指示数**」を含む

例）機器ID：1は、8/1　1:00 の指示数は500

方法①：UPDATEモデル

機器ID、**日付**ごとに行を持ち、該当する時間のカラムの値を**UPDATE**

| 機器ID | 日付 | 0:00 | 0:30 | 1:00 | 1:30 | … |
|---|---|---|---|---|---|---|
| 1 | 8/1 | 100 | 300 | 500 | | |
| 2 | 8/1 | 200 | 400 | | | |

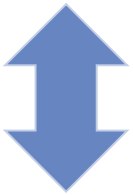
PostgreSQLは追記型アーキテクチャを採用しているため、
高頻度の更新処理は性能がでにくい。

# 格納モデルの選択

NTT DATA

## 方法①：UPDATEモデル

| 機器ID | 日付 | 0:00 | 0:30 | 1:00 | 1:30 | … |
|---|---|---|---|---|---|---|
| 1 | 8/1 | 100 | 300 | 500 | | |
| 2 | 8/1 | 200 | 400 | | | |



## 方法②：INSERTモデル

機器ID、**日時**ごとに、レコードを**INSERT**

| 機器ID | 日付時刻 | 値 |
|---|---|---|
| 1 | 8/1 0:00 | 100 |
| 1 | 8/1 0:30 | 300 |
| 1 | 8/1 1:00 | 500 |
| … | … | … |

○ レコードの追加(INSERT) が速い。
× テーブルサイズが大きくなる。

# 格納モデルの選択

## 方法①：UPDATEモデル

| 機器ID | 日付 | 0:00 | 0:30 | 1:00 | 1:30 | … |
|---|---|---|---|---|---|---|
| 1 | 8/1 | 100 | 300 | 500 | | |
| 2 | 8/1 | 200 | 400 | | | |

格納性能重視で採用！

## 方法②：INSERTモデル

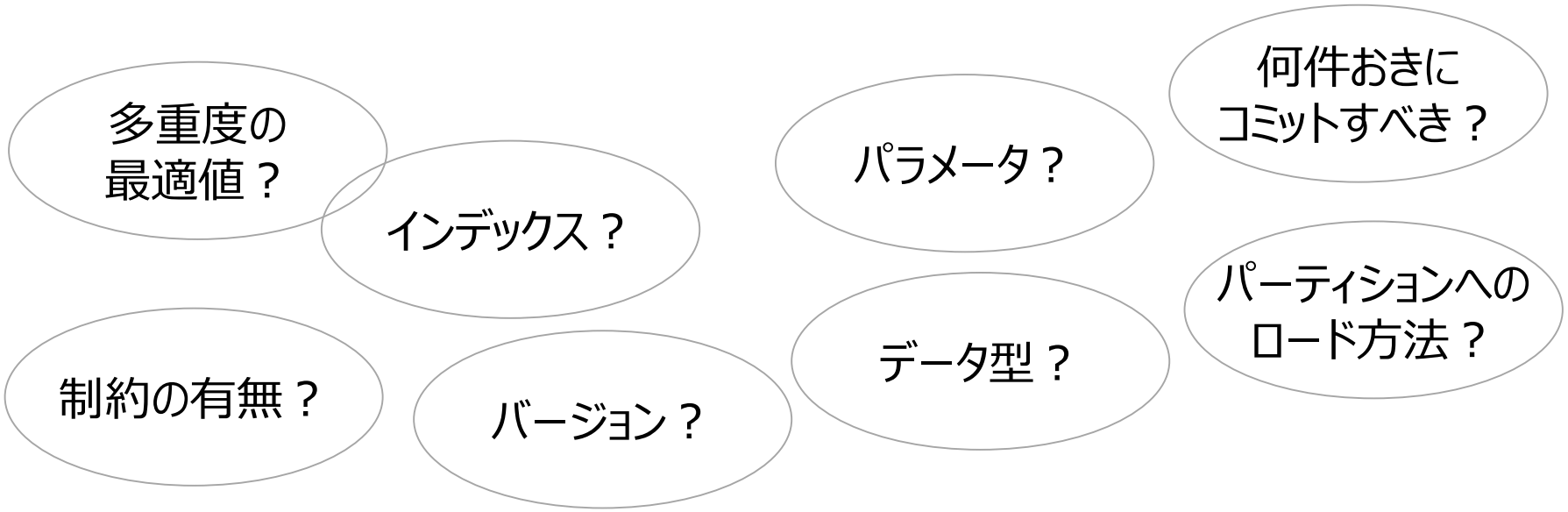
機器ID、**日時**ごとに、レコードを**INSERT**

| 機器ID | 日付時刻 | 値 |
|---|---|---|
| 1 | 8/1 0:00 | 100 |
| 1 | 8/1 0:30 | 300 |
| 1 | 8/1 1:00 | 500 |
| … | … | … |

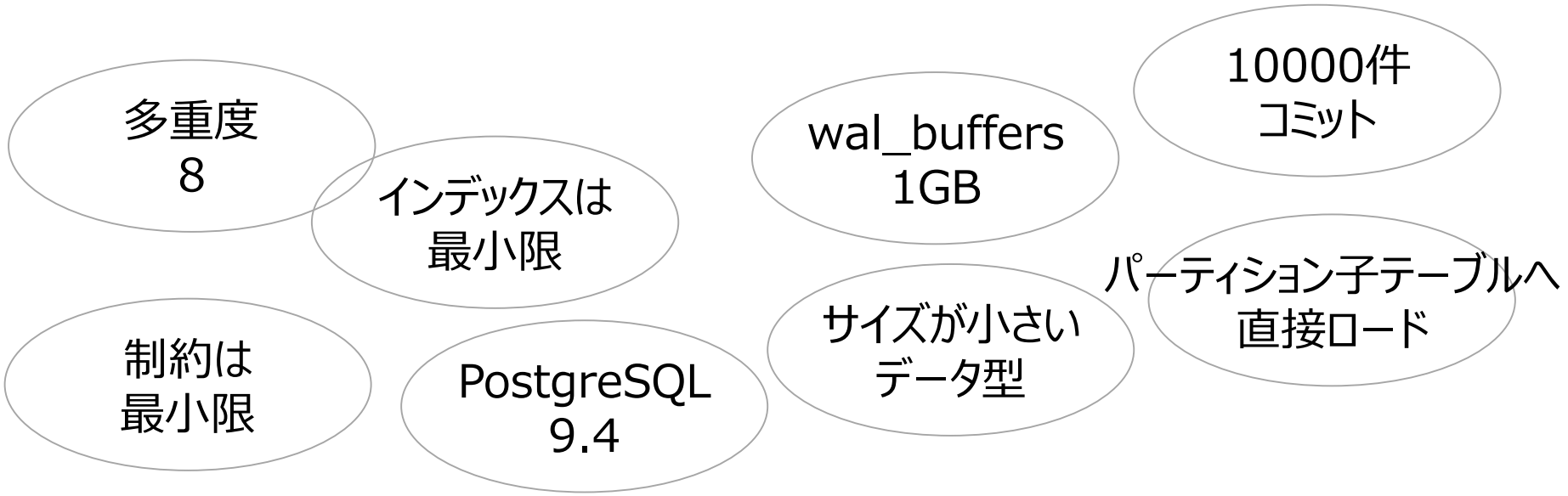
○ レコードの追加(INSERT) が速い。
× テーブルサイズが大きくなる。

NTT DATA

# 性能ファクターを洗い出して事前検証

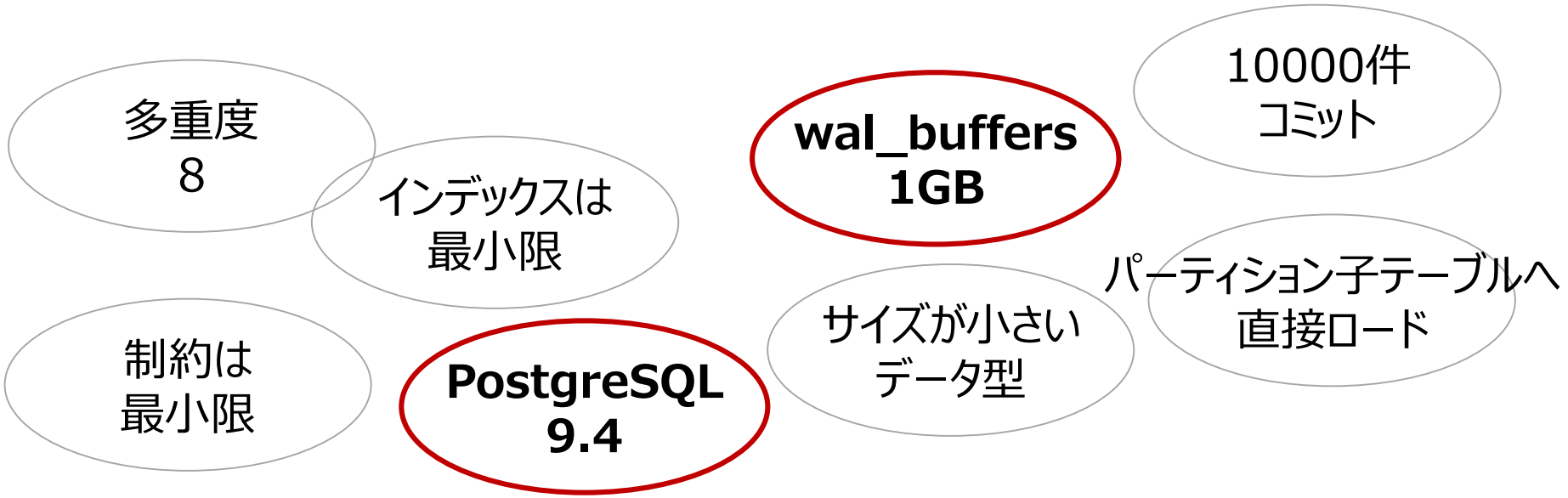


# 高速ロードにおける性能ファクター




**DB設計へのフィードバック**
**チューニング時の切り札**

# 特に性能改善に寄与した性能ファクター



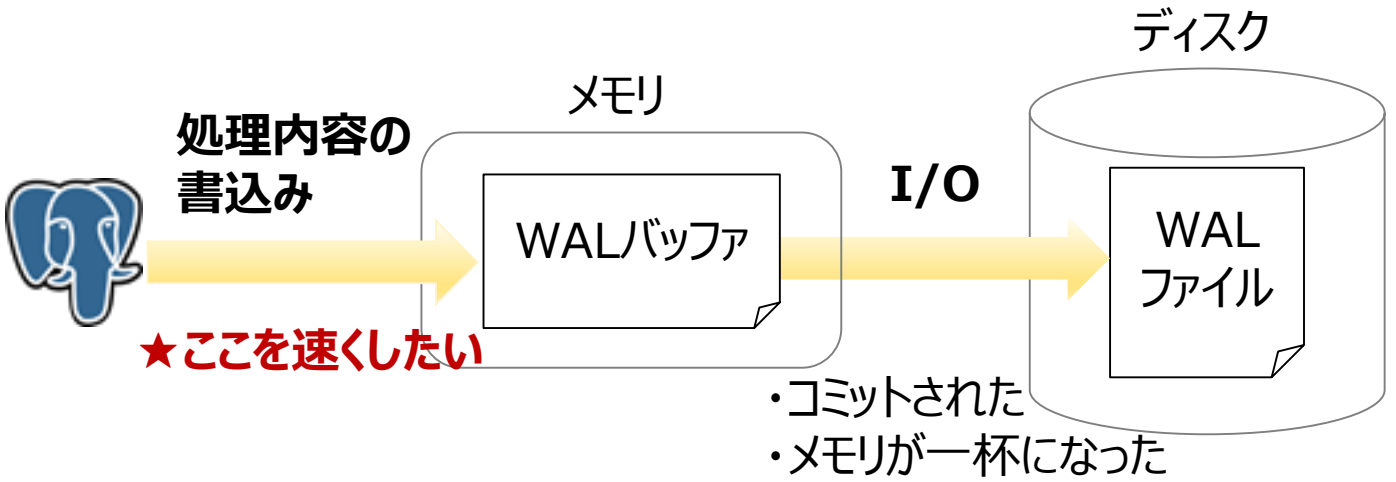
# ボトルネックはWAL処理だった！

perf（Linux性能解析ツール）

```
19.83% postgres postgres [.] XLogInsert ★
 6.45% postgres postgres [.] LWLockRelease
 4.41% postgres postgres [.] PinBuffer
 3.03% postgres postgres [.] LWLockAcquire
```

**WALがネック！**

**WAL処理**

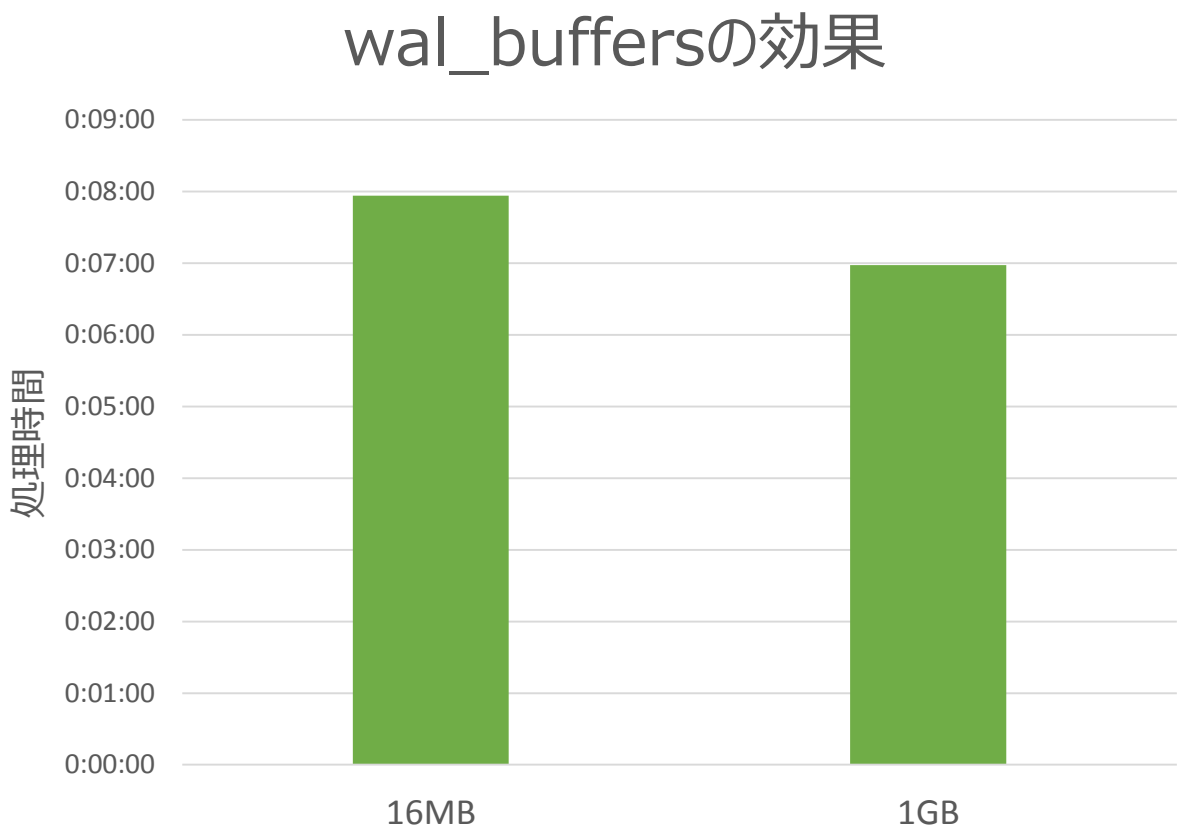


# まずは･･･wal_buffersパラメータ

「デフォルト設定の-1で選択される自動チューニングによると、**ほとんどの場合**妥当な結果が得られます。」

PostgreSQLドキュメントより

# wal_buffers検証結果




※格納時間のみ
（参照時間等除く）

# PostgreSQLバージョン

NTT DATA

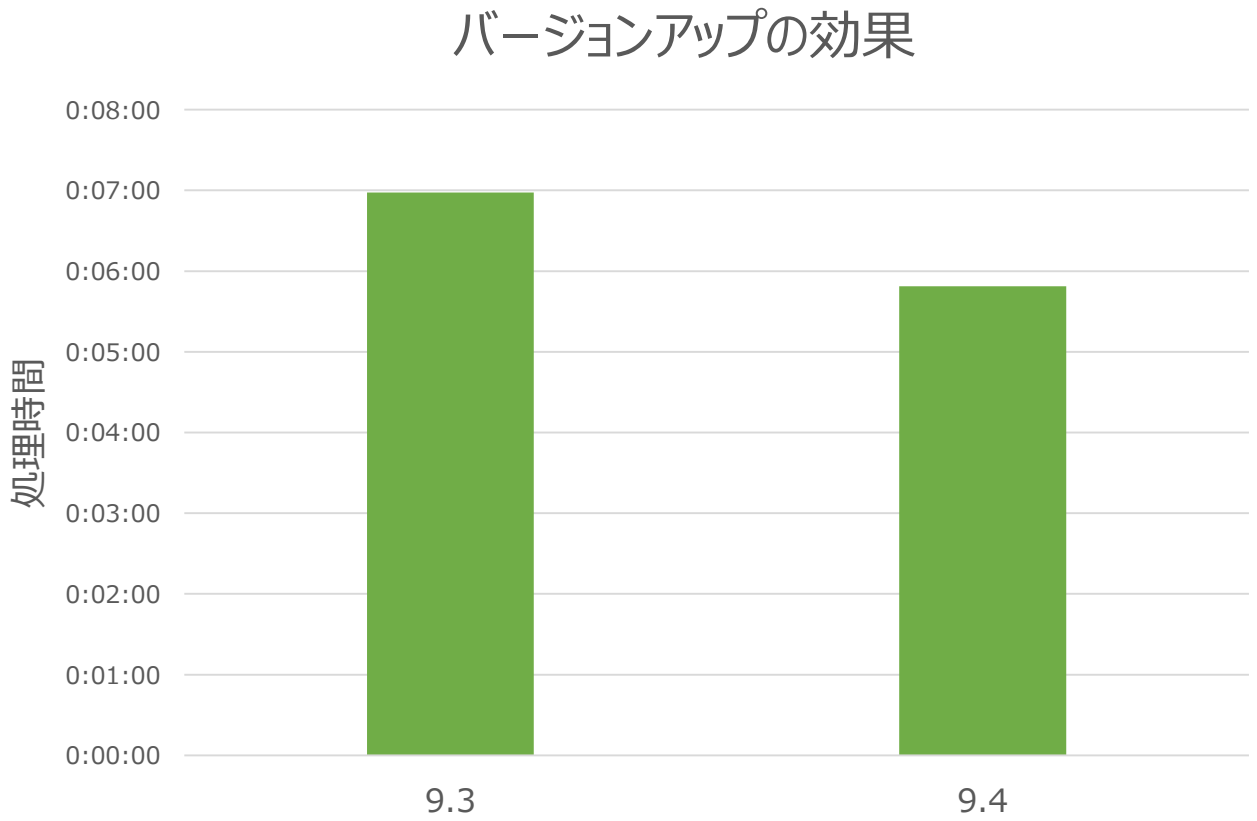
・WAL性能向上
・JSONB
・GINインデックス性能向上
・ロジカル・デコーディング

**9.3**

**9.4**
2014年12月リリース

# バージョンアップによる高速化

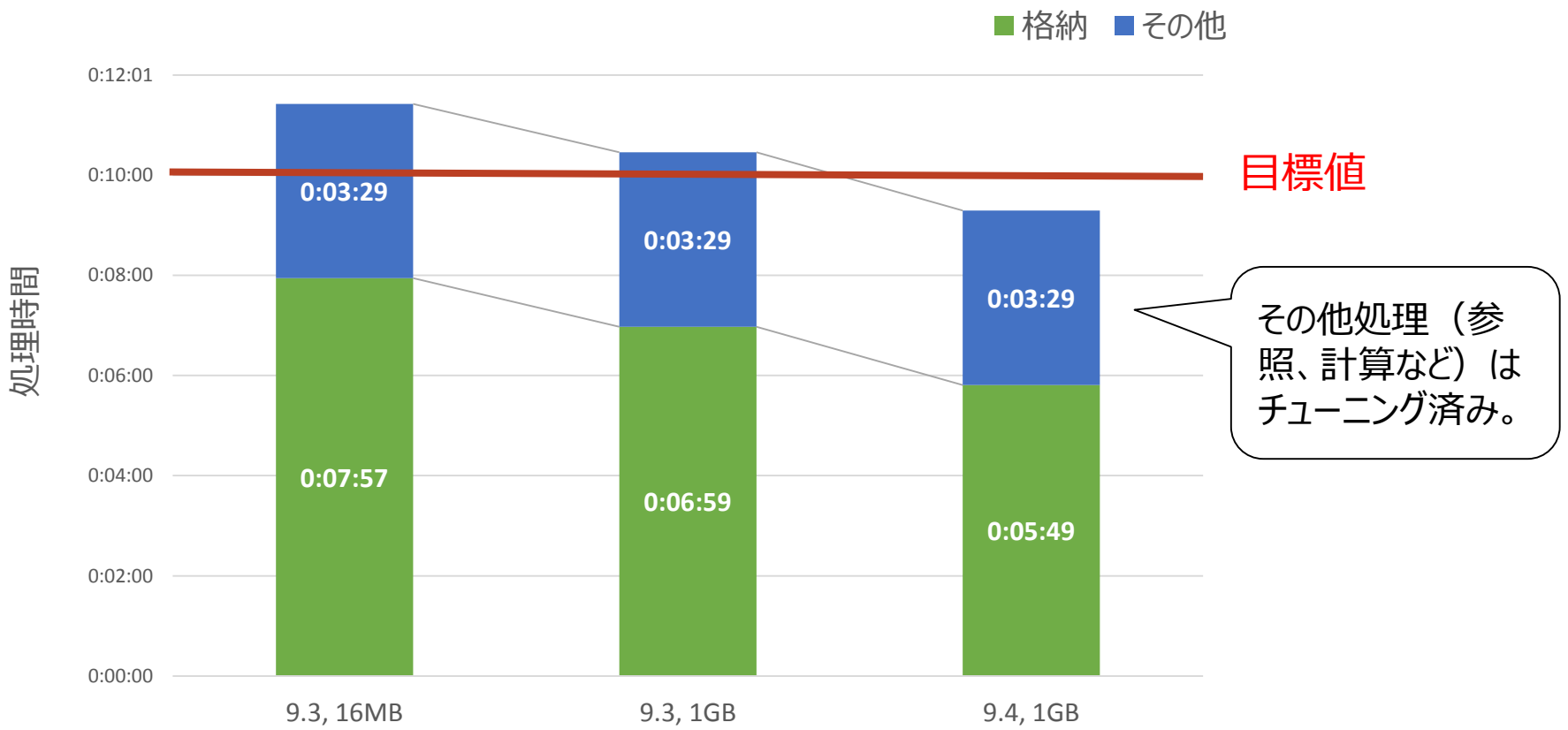
- 当初PostgreSQL9.3を採用予定から**9.4**に変更




※格納時間のみ
（参照時間等除く）

# 改善の効果

## 目標達成！！

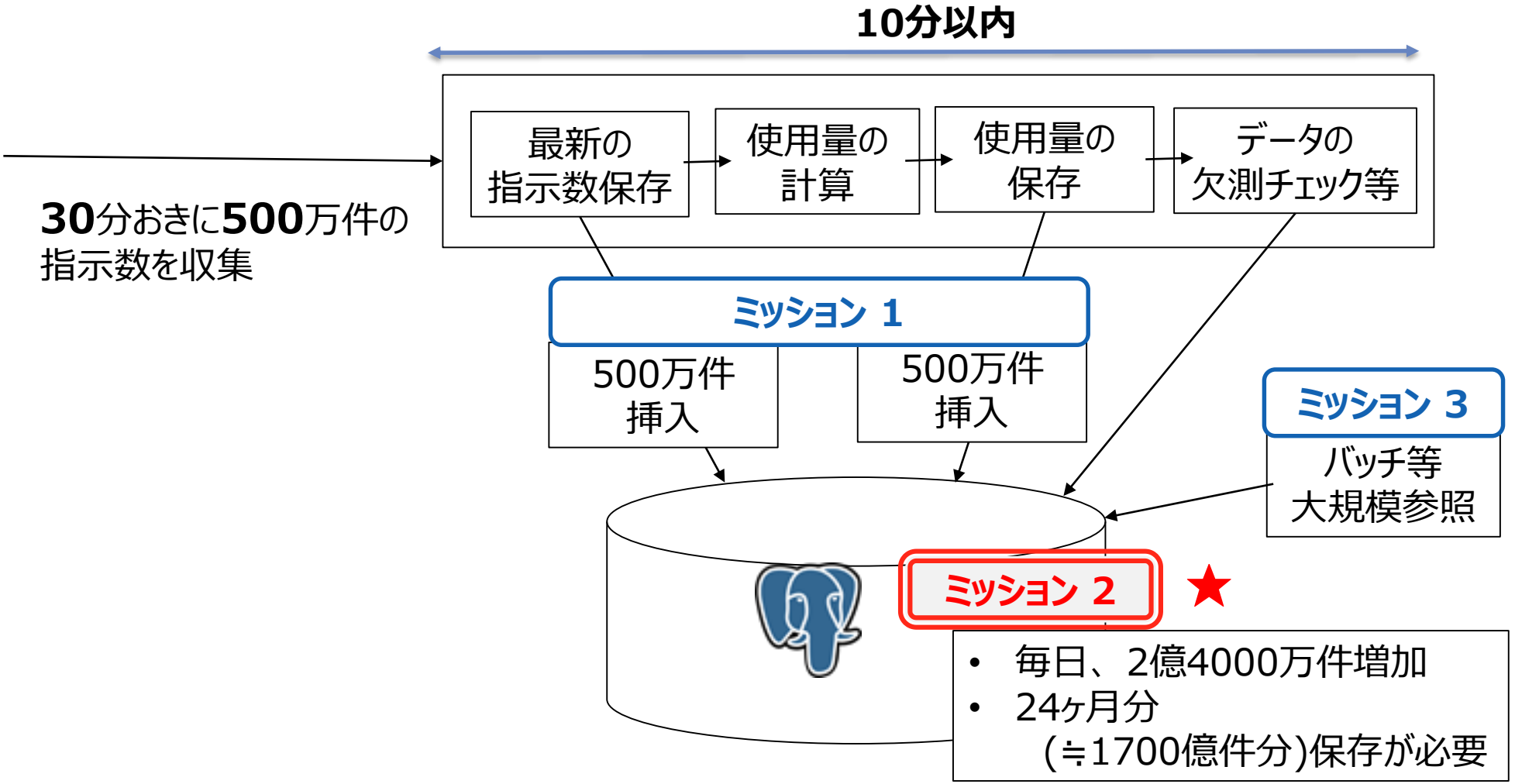



※終局時ピークを想定した試験

# (2) 24ヶ月分の大量データを長期保管せよ！

NTT DATA

## スマートメーター運用管理システムの主要処理



108TB

# データ型を選んでサイズを小さく

NTT DATA

- 整数
  - ✓ 範囲、精度をカバーできる**最も小さいデータ型**を使う！

| 型 | 精度 | サイズ |
|---|---|---|
| SMALLINT | 4桁(約±3.2万) | 2 byte |
| INTEGER | 9桁(約±21億) | 4 byte |
| BIGINT | 18桁(約±922京) | 8 byte |
| NUMERIC | 1000桁 | 3 or 6 or 8 + ceiling(桁数 / 4) * 2 |

- 真偽値
  - ✓ CHAR(1)ではなく、**フラグ**を使う！

| 型 | 表現可能なデータ | サイズ |
|---|---|---|
| CHAR(1) | 文字列長1の文字列 | 5 byte |
| BOOLEAN | 真または偽の状態 | 1 byte |

# カラムの順番を変えてデータサイズを小さく

NTT DATA

- アライメント（配置境界）
  - PostgreSQLはアライメントをまたがってカラムを格納しない

8 byteの場合

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| column_1(4byte) | | | | ***PADDING*** | | | |
| column_2(8byte) | | | | | | | |

72 → 60

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| column_1 | | | | ***PADDING*** | | | |
| column_2 | | | | | | | |
| column_3 | | | | column_4 | | *PADDING* | |
| column_5 | | | | | | | |
| column_6 | ********PADDING********* | | | | | | |
| column_7 | | | | | | | |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| column_2 | | | | | | | |
| column_5 | | | | | | | |
| column_7 | | | | | | | |
| column_1 | | | | column_3 | | | |
| column_4 | | column_6 | | | | | |

| Column | Type |
|---|---|
| column_1 | integer |
| column_2 | timestamp without time zone |
| column_3 | integer |
| column_4 | smallint |
| column_5 | timestamp without time zone |
| column_6 | smallint |
| column_7 | timestamp without time zone |

1行あたり12byte
節約すると…
一日あたり**2.8**GB！

# 格納モデルの変換

挿入時は性能重視で**INSERTモデル**を選択
- サイズが大きいので、長期保管には向いていない

アクセス頻度が低くなる期間からモデルを変換

| 項番 | 対象データ | 参照頻度 | 更新頻度 | 方針 | 方式 |
|---|---|---|---|---|---|
| 1 | 当日<br>～65日分 | 高 | 高 | **更新性能重視** | INSERTモデル |
| 2 | 66日<br>～24ヶ月分 | 低 | 低 | **参照容易なレベルで<br>ストレージ量削減** | UPDATEモデル |

# 格納モデル変換

## INSERTモデル

| 機器ID | 日付時刻 | 値 |
|---|---|---|
| 1 | 8/1 0:00 | 100 |
| 2 | 8/1 0:00 | 100 |
| 1 | 8/1 0:30 | 300 |
| 2 | 8/1 0:30 | 200 |
| 1 | 8/1 1:00 | 500 |
| 2 | 8/1 1:00 | 300 |
| … | … | … |
| 1 | 8/1 22:30 | 1000 |
| 2 | 8/1 22:30 | 800 |
| 1 | 8/1 23:00 | 1100 |
| 2 | 8/1 23:00 | 900 |
| 1 | 8/1 23:30 | 1200 |
| 2 | 8/1 23:30 | 1000 |

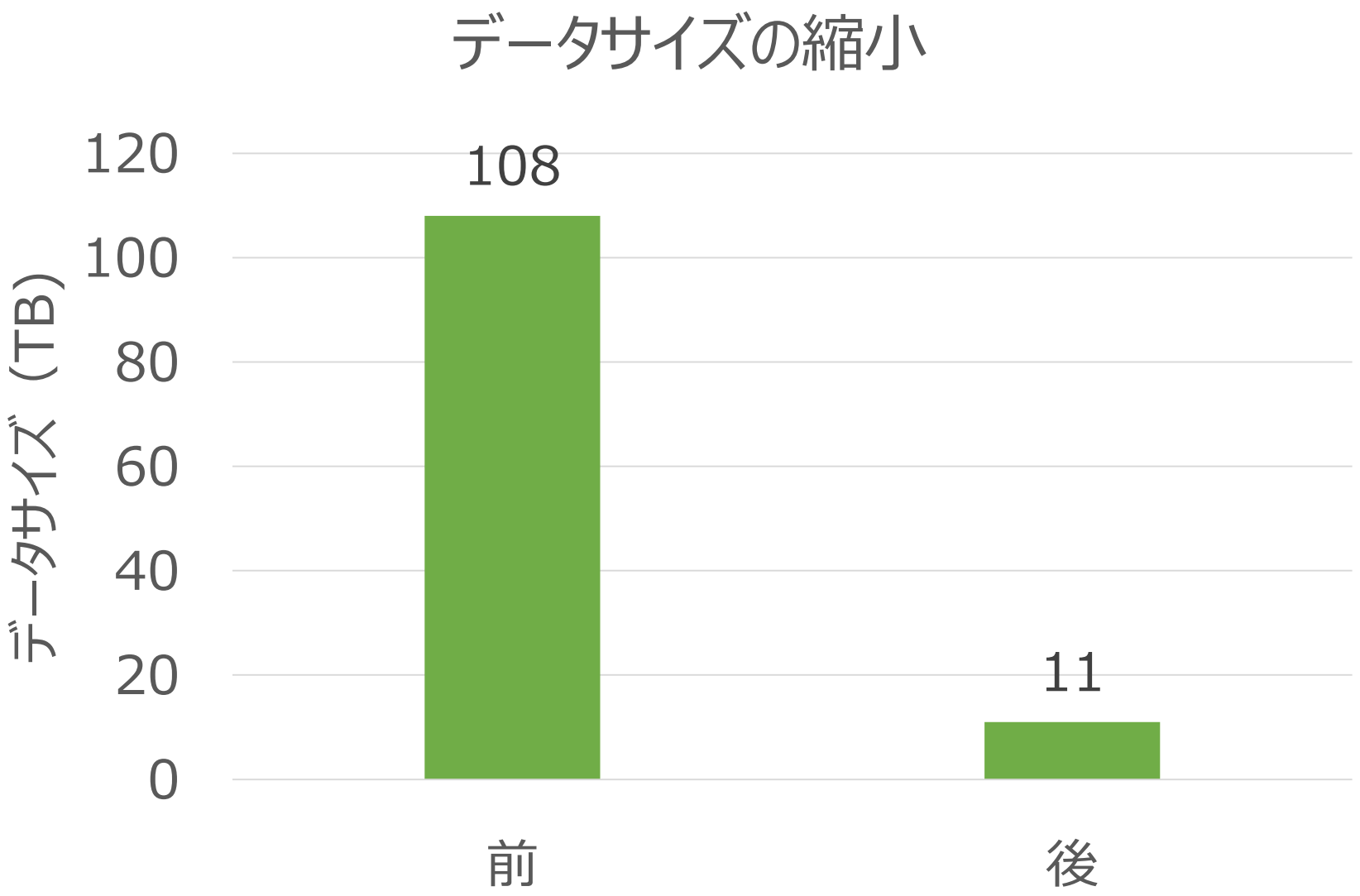
## UPDATEモデル

| 機器ID | 日付 | 0:00 | 0:30 | 1:00 | … | 22:30 | 23:00 | 23:30 |
|---|---|---|---|---|---|---|---|---|
| 1 | 8/1 | 100 | 300 | 500 | … | 1000 | 1100 | 1200 |
| 2 | 8/1 | 100 | 200 | 300 | … | 800 | 900 | 1000 |

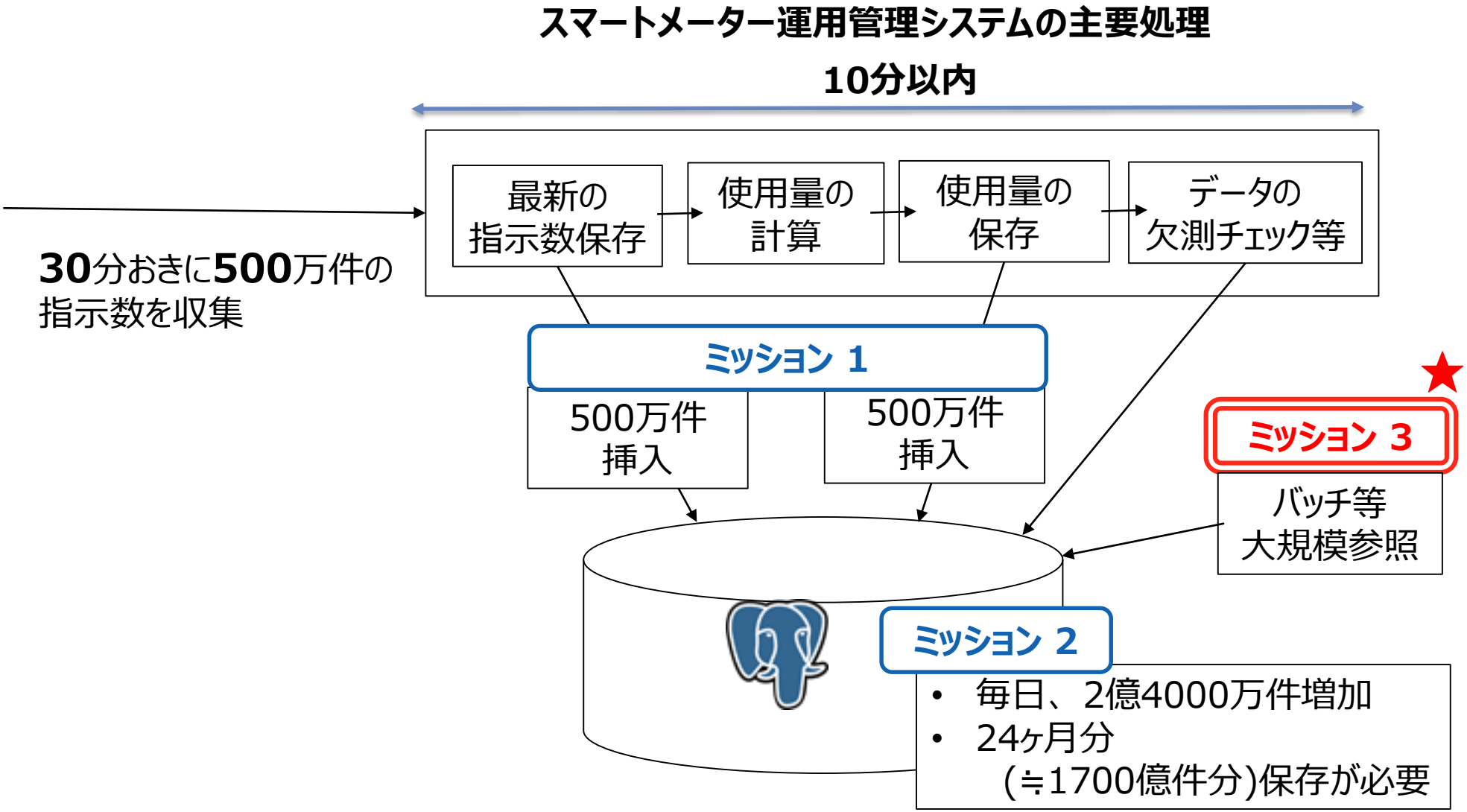
機器ID、**日時**ごとに1レコード から
機器ID、**日付**ごとに1レコード　に変換

毎レコードに存在する機器ID,日付の情報を重複して持たない。
1日の件数：　**2億4000万件→500万件**
サイズ：　**22GB→3GB**

# 改善の効果



# (3) 大規模参照の性能を安定化せよ！



# 大規模参照の性能を安定化せよ！

本案件では、「**安定して性能要件を満たすこと**」が大切

- 突然の実行計画の変化による性能劣化は避けたい

**安定稼動**

**pg_hint_plan**

**実行計画制御**
**特定のプランを選ばせる**

**pg_dbms_stats**

**統計情報固定化**
**プランを変更させない**

# pg_hint_planとpg_dbms_statsを使用する前に

NTT DATA

PostgreSQLのプランナは**ほとんどの場合は正しい実行計画を選ぶ**

実行計画を固定すると、むしろ性能問題を引き起こすかもしれない

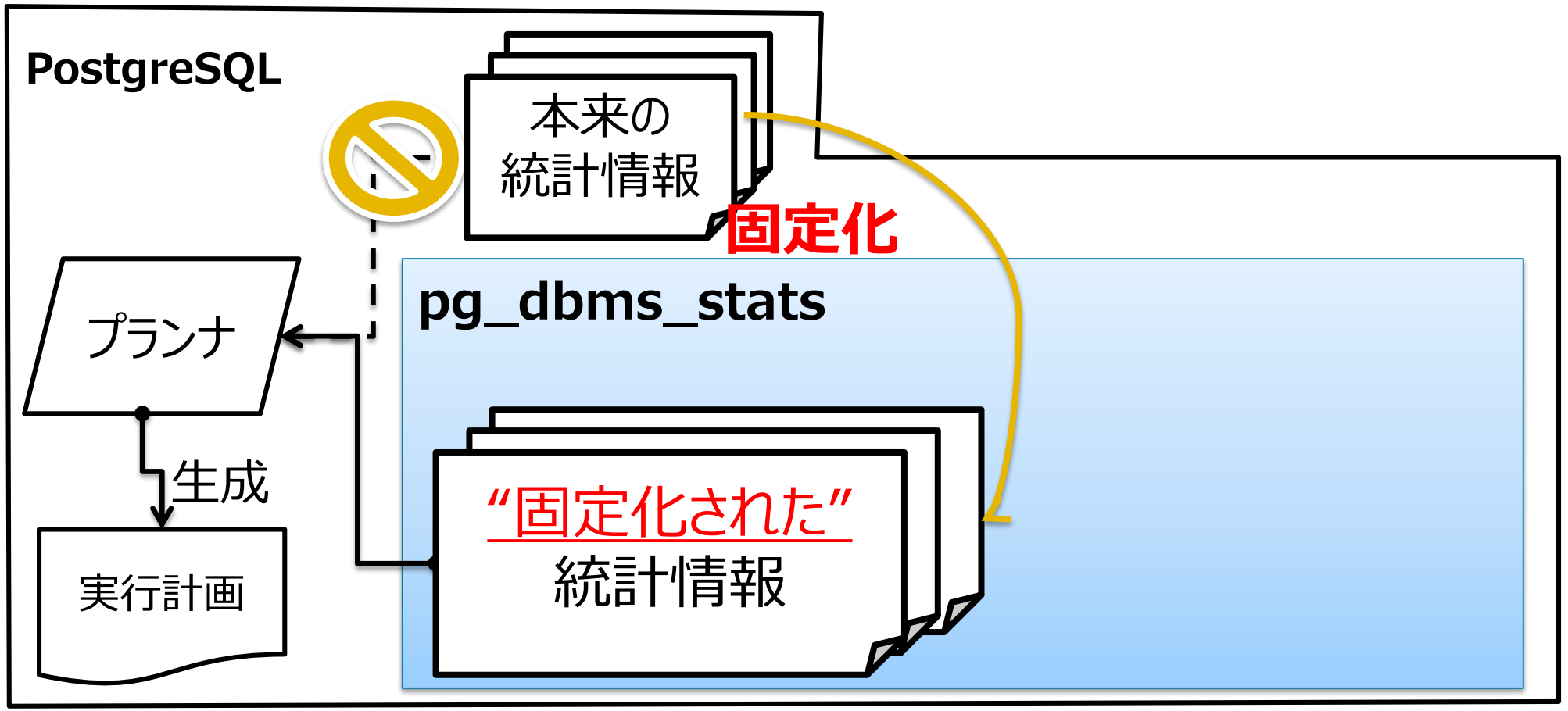
- ある時点で最適な実行計画でも、データの内容/サイズの変化により、**遅い実行計画になりうる**

それでも、プランナが間違えるリスクを極力ゼロにしなければならない。

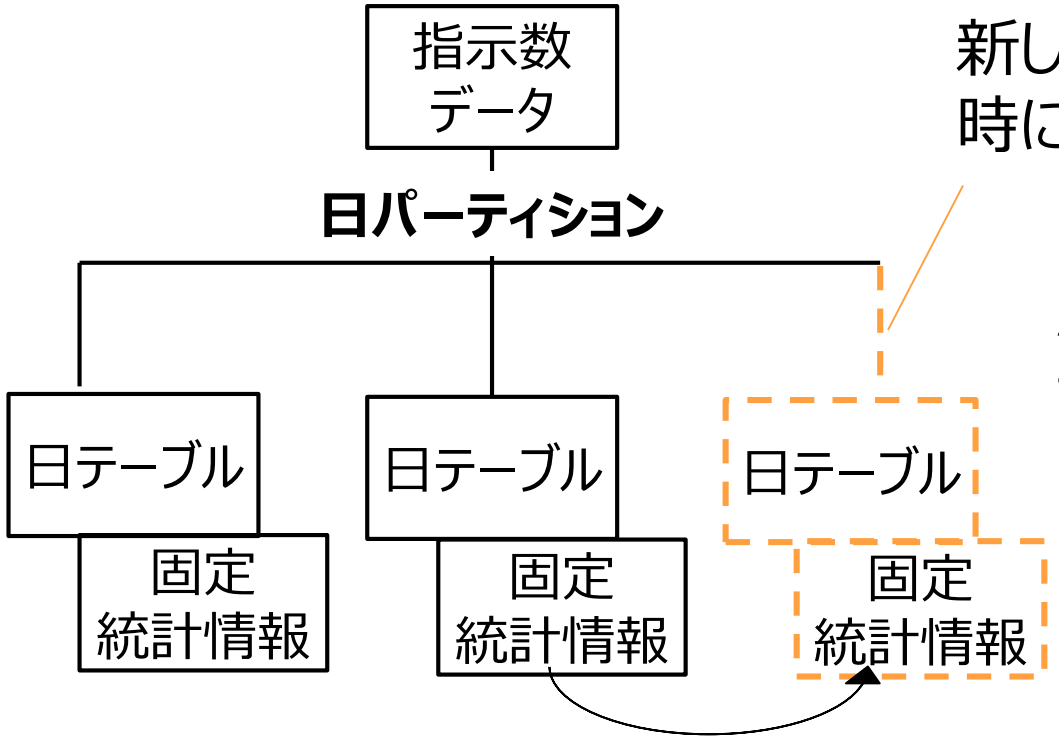
- バッチ処理のあと直後のANALYZEが終わるまで
- テーブルの結合数が非常に多いケース
- …

**→デメリットを理解したうえで採用**

# pg_dbms_statsによる固定化



# 本プロジェクトのpg_dbms_stats使いこなし




新しいパーティション子テーブルの追加と同時に準備された固定統計情報をセット

ANALYZEをしなくても、
テーブル作成直後から
**すぐに最適なプランが得られる**

コピー

**一部の統計情報はテーブルによって異なる**

- テーブルOID、テーブル名
- パーティションキー、時刻

# テーブルによって変わるべき統計情報の付け替え

| 子テーブルによって格納される値が大きく違うカラム |
| --- |
| パーティションキー |
| 時刻 |

×

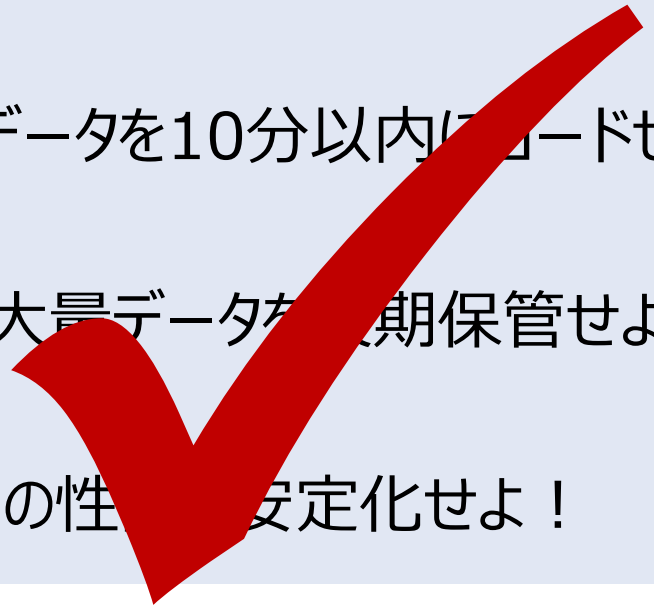
| 格納された値を元にした統計情報 |
| --- |
| 最頻値・頻度 |
| ヒストグラム |

- 値は想定可能なので、**ダミーデータ**を作成。
- ダミーデータを**ANALYZEした結果**をセット

たとえば、”8/1 0:00” “8/1 0:30” “8/1 1:00” など。
1日で48種類。分布は一様になる。

# COMPLETE

1. 1000万件データを10分以内にロードせよ！
2. 24ヶ月分の大量データを長期保管せよ！
3. 大規模参照の性能を安定化せよ！

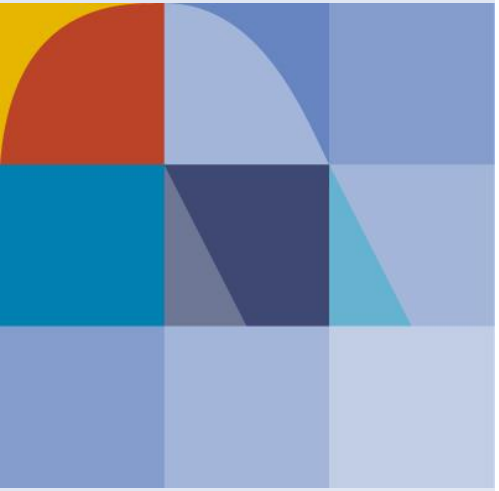


**2016年、PostgreSQL20周年**

PostgreSQLはついに大規模社会インフラに採用されるまでとなりました。

しかし巨大なサイズ、厳しい性能要件を持った高難易度のシステムには、
PostgreSQLのノウハウに加え、業務特性も活かした工夫が必要です。

システムの成功のために、徹底した事前検証とノウハウで
PostgreSQLのポテンシャルを引き出しましょう！



NTT DATA

Global IT Innovator