今、改めて学ぶVACUUM

PostgreSQL Conference Japan 2018
2018年11月22日

SRA OSS, Inc. 日本支社
佐藤 友章
sato@sraoss.co.jp

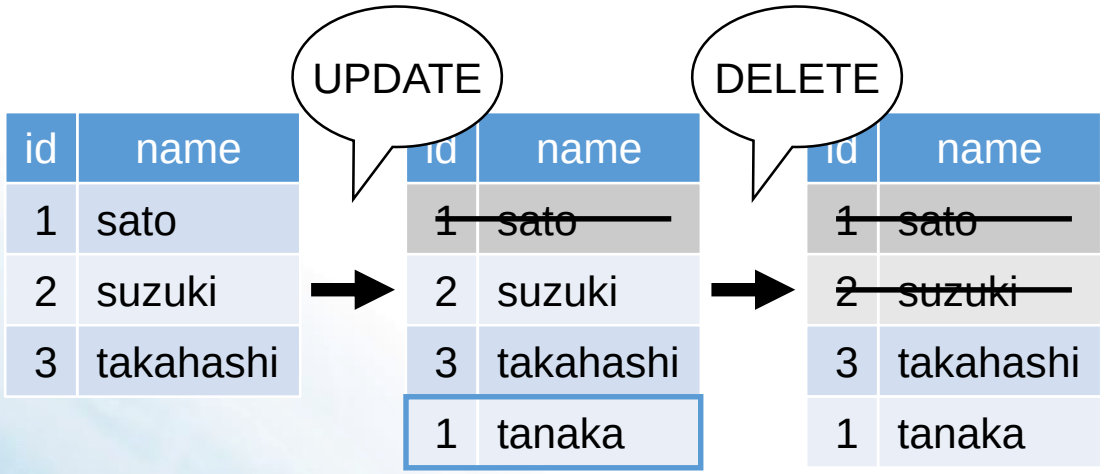
目次

- VACUUMの概要
- VACUUMの実行
- VACUUM実行状況の確認
- VACUUMに関するパラメータの設定

VACUUMの概要

VACUUMとは

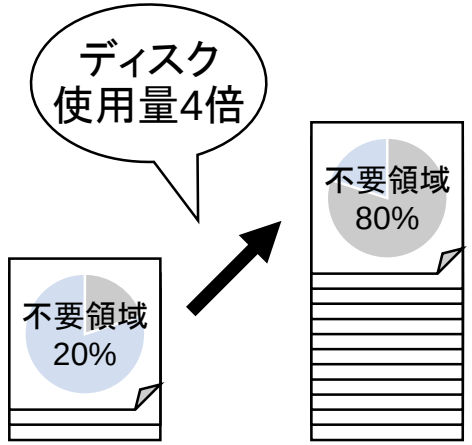
- UPDATE、DELETEで発生した不要領域を回収する処理
- なぜ不要領域が発生するのか？



  - UPDATEでは、更新前のデータに削除フラグをつけて、更新後のデータを新たに登録
  - DELETEでは、削除対象のデータに削除フラグをつける

削除フラグのついた領域＝不要領域

- 不要領域が増えると、どうなるのか？
  - ファイルサイズが大きくなり、ディスク使用量が増える
  - メモリにキャッシュされにくくなり、ディスクアクセスが増え、性能が下がる

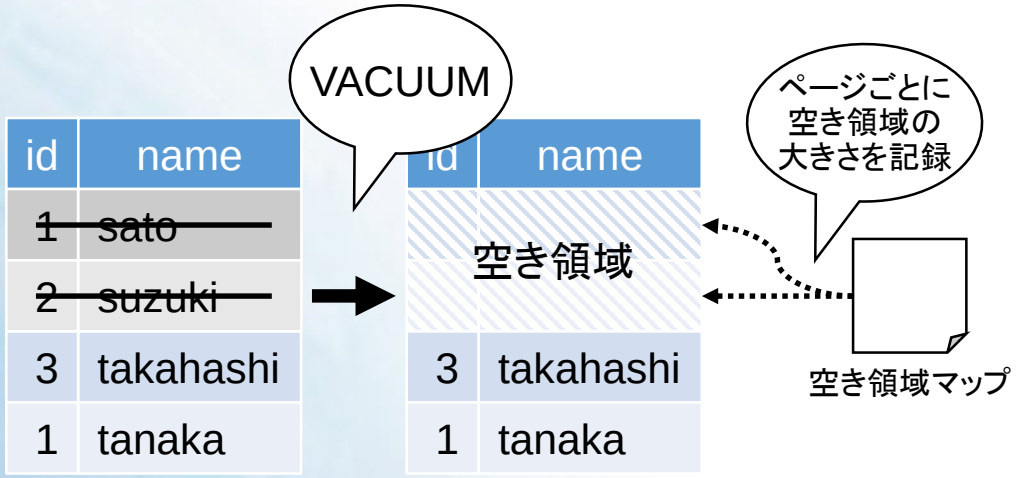


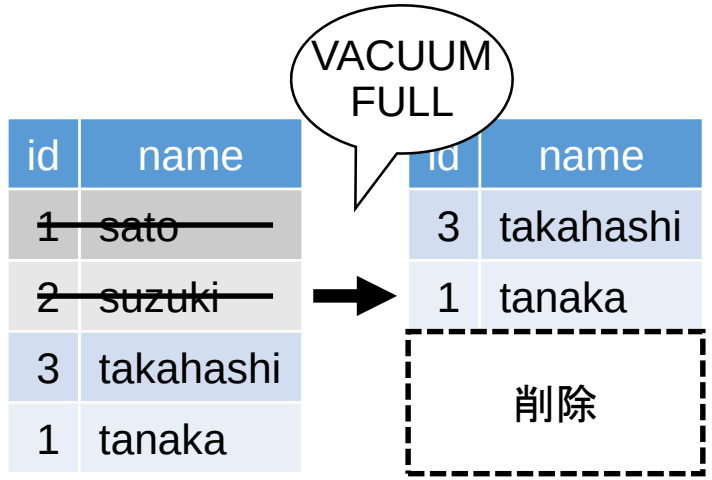
1GBのデータを格納するのに、
不要領域の割合が20%だと
1.25GB必要なのに対して、
80%だと5GB必要になる

不要領域が増えないように
定期的なVACUUMの実行が必要

VACUUMとVACUUM FULL

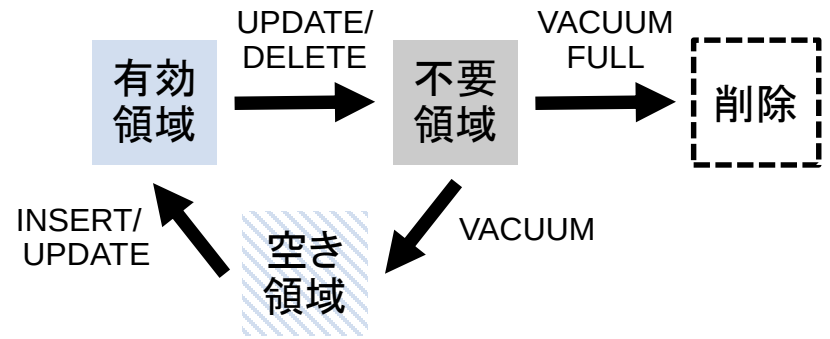
- VACUUMは（FULLでない）VACUUMとVACUUM FULLの2種類
- VACUUM
  - 不要領域を空き領域マップ（FSM）に登録して、再利用可能にする処理
  - ファイルサイズは基本的に小さくならない
  - INSERT、UPDATE、DELETEと並行して実行できる
- VACUUM FULL
  - 不要領域を削除する処理
  - ファイルサイズが小さくなる
  - SELECTとも並行して実行できない
  - 対象テーブルと同程度の空き容量が必要
  - インデックスも再構築される



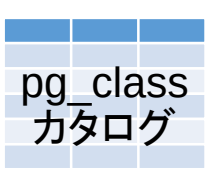


VACUUMを適切に実行しておけば、基本的にVACUUM FULLは不要

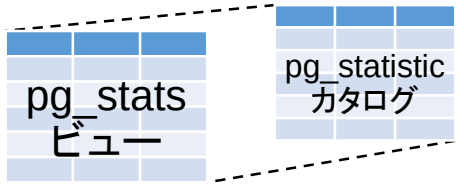
VACUUMの役割



ディスク使用量の増加防止/削減

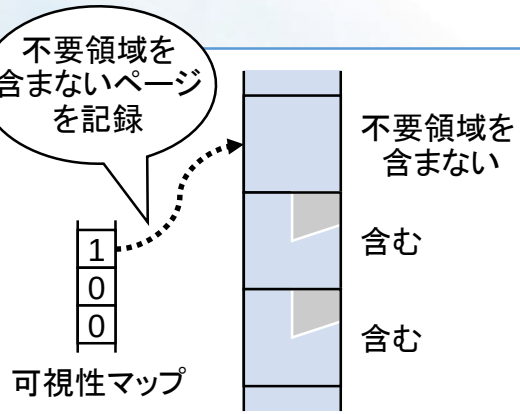


- relpages、reltuplesにテーブルのページ数、行数を格納
- ANALYZE、CREATE INDEXなど、一部のDDLでも更新



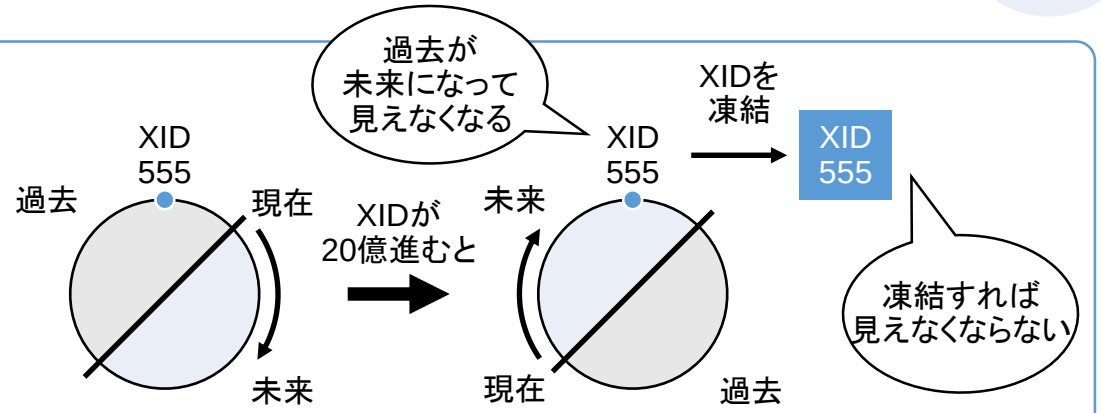
- テーブルの内容に関する統計情報を格納
- ANALYZEオプションをつけて実行した場合に更新

プランナ用統計情報の更新



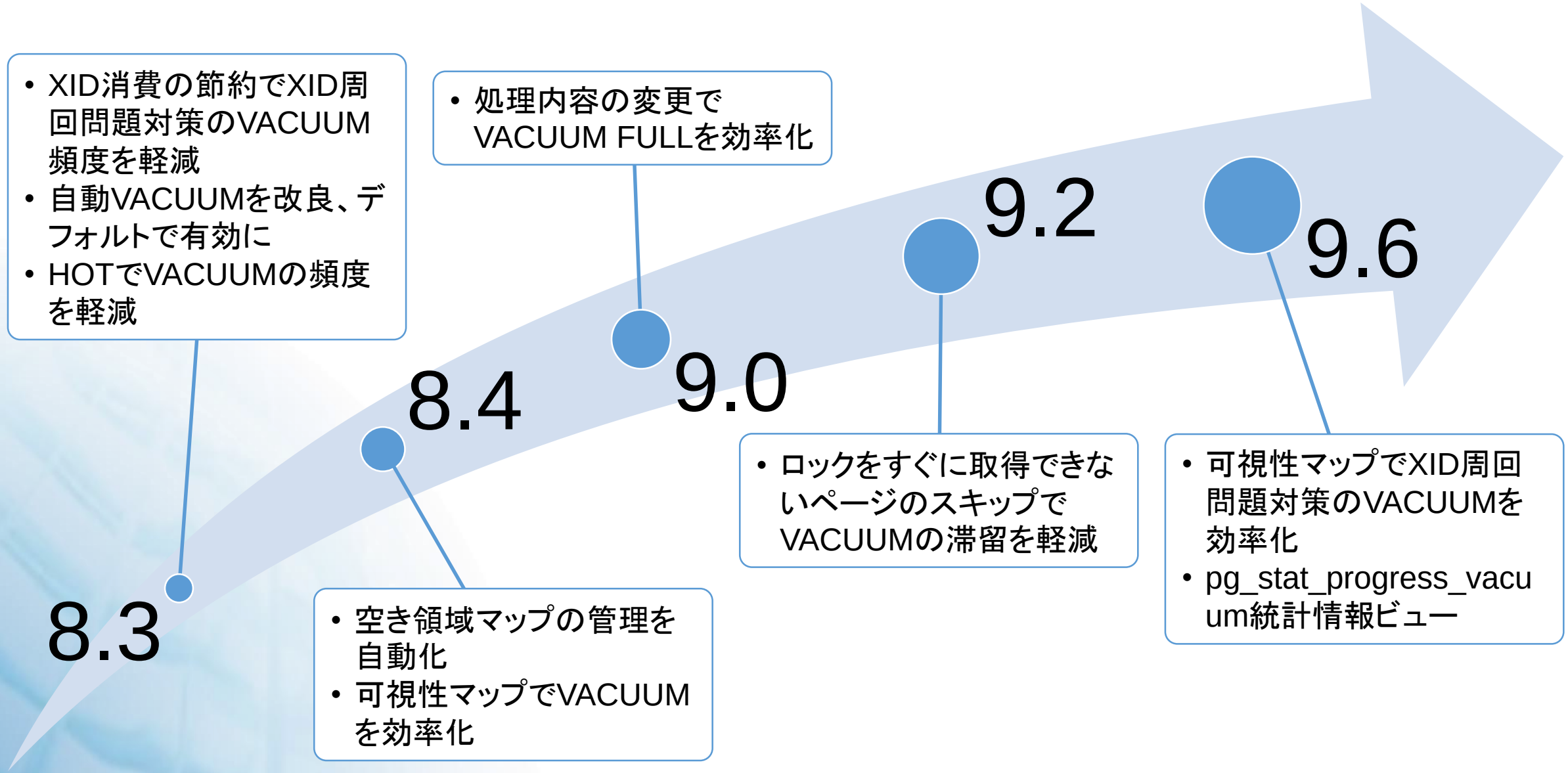
- VACUUM時に不要領域を含まないページをスキップできる
- インデックスオンリースキャンが使える

可視性マップ（VM）の更新



トランザクションID周回問題の防止

VACUUMの改善点



VACUUMの実行

VACUUMの実行（1）手動VACUUM

- ユーザが任意に手動でVACUUMを実行
- テーブルまたはデータベースの所有者で実行
- SQLのVACUUMまたはコマンドのvacuumdbを使用

SQL `VACUUM [ オプション ] [ テーブル名 ]`

コマンド `vacuumdb [ オプション ] [ -t | --table テーブル名 ] [ データベース名 ]`

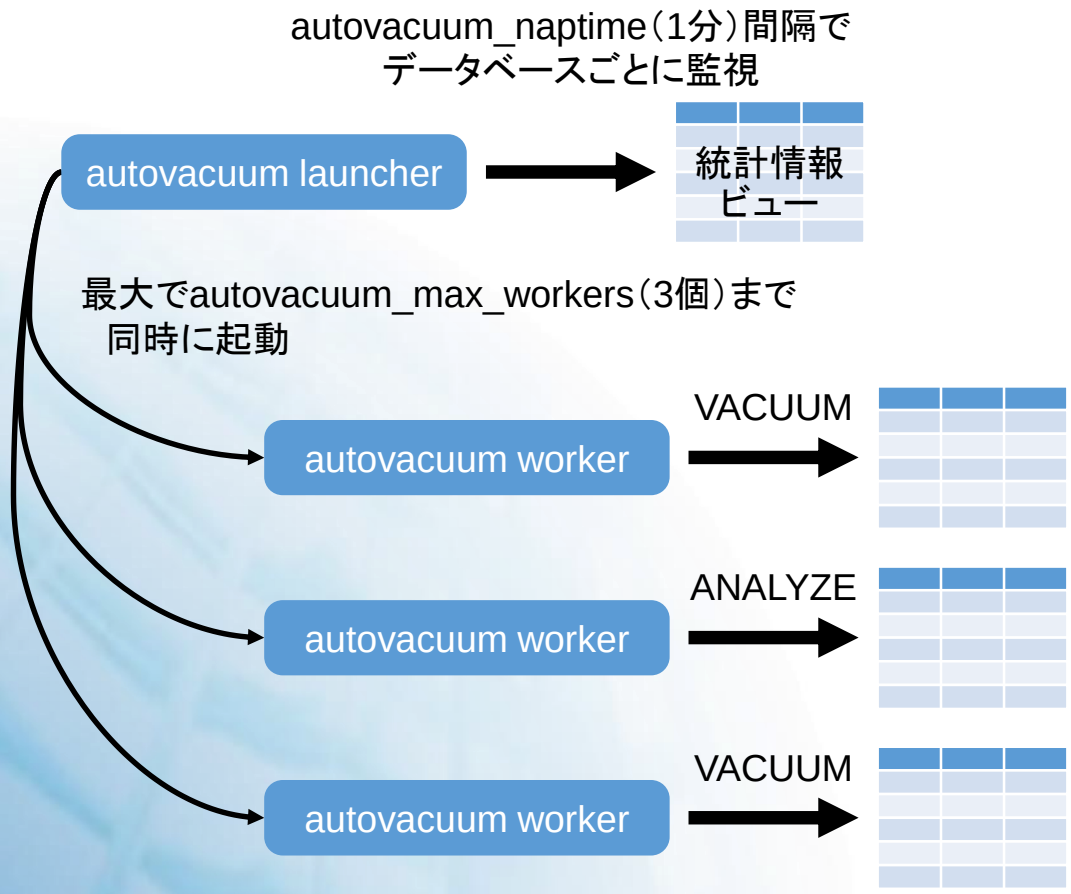
できることは基本的に同じ

- おもなオプション

VACUUM	vacuumdb	説明
FULL	-f、--full	不要領域を削除（VACUUM FULL）
FREEZE	-F、--freeze	XIDを凍結（XID周回問題対策のVACUUM）
VERBOSE	-v、--verbose	詳細情報を出力
ANALYZE	-z、--analyze	プランナ用統計情報を更新（ANALYZE）
テーブル名	-t、--table テーブル名	指定したテーブルを対象に
（該当なし）	-a、--all	すべてのデータベースを対象に

VACUUMの実行（2）自動VACUUM

- 不要領域の発生量を監視して、自動的にVACUUM、ANALYZEを実行
- autovacuumパラメータで有効にするかを指定（デフォルトで有効）



VACUUMの実行条件

不要行数pg_stat_all_tables.n_dead_tup
> autovacuum_vacuum_threshold（50行）
+ autovacuum_vacuum_scale_factor（20%）× 有効行数pg_class.reltuples

ANALYZEの実行条件

ANALYZE実行後の変更行数pg_stat_all_tables.n_mod_since_analyze
> autovacuum_analyze_threshold（50行）
+ autovacuum_analyze_scale_factor（10%）× 有効行数pg_class.reltuples

XID周回問題対策のVACUUMの実行条件

凍結後のトランザクション数age(pg_class.relfrozenxid)
> autovacuum_freeze_max_age（2億トランザクション）

**VACUUMはテーブルの20%が不要領域になったら
ANALYZEは10%が変更されたら実行される**

VACUUM実行のキャンセル

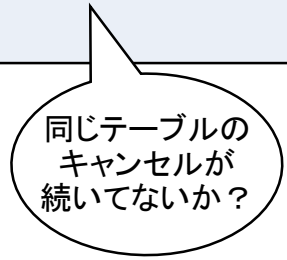
- SQL実行をキャンセルする場合と同じ
  - pg_stat_activity統計情報ビューか、psコマンドでプロセスIDを調べる

```
=# SELECT * FROM pg_stat_activity;
-[ RECORD 3 ]----+-----------------------------------------------
datid            | 13878                                          データベースのOID
datname          | postgres                                       スキーマ名
pid              | 1462                                           プロセスID
（省略）
query            | autovacuum: VACUUM ANALYZE public.pgbench_accounts  実行中のSQL
backend_type     | autovacuum worker                              バックエンドの種類（10以降）
```

  - pg_cancel_backend関数か、SIGINTシグナルを送って、キャンセル
  - キャンセルできない場合は、pg_terminate_backend関数か、SIGTERMシグナルを送って、プロセスを停止
- 競合するロックが要求されると、自動的にキャンセルされる

```
ERROR:  canceling autovacuum task
CONTEXT:  automatic vacuum of table "postgres.public.pgbench_accounts"
```

- キャンセルしても動作に影響はないが、途中の作業は基本的に失われる
  - 11以降ではVACUUM実行中に空き領域マップが更新される



VACUUM実行状況の確認

自動VACUUMの実行記録（1）

- log_autovacuum_min_durationパラメータ
  - 指定した時間以上かかった自動VACUUMの実行をログに記録
  - デフォルトは-1（無効）、0ですべて記録
- ログの出力例
  - VACUUMの場合（11以降ではXID周回問題対策だと「automatic aggressive vacuum」と表示）

```
LOG:  automatic vacuum of table "postgres.public.pgbench_accounts": index scans: 1
    pages: 0 removed, 2186 remain, 0 skipped due to pins, 0 skipped frozen
    tuples: 33333 removed, 100000 remain, 0 are dead but not yet removable, oldest xmin: 668
    buffer usage: 6607 hits, 0 misses, 0 dirtied
    avg read rate: 0.000 MB/s, avg write rate: 0.000 MB/s
    system usage: CPU: user: 0.18 s, system: 0.02 s, elapsed: 0.85 s
```

  - ANALYZEの場合

```
LOG:  automatic analyze of table "postgres.pg_catalog.pg_attribute" system usage: C
0.00 s, system: 0.00 s, elapsed: 0.12 s
```

同じテーブルのスキップが続いてないか？

  - VACUUM/ANALYZEがスキップされた場合（1つ目は10以降、2つ目は11以降）

```
LOG:  skipping vacuum of "postgres.public.pgbench_accounts" --- lock not available
```

```
LOG:  skipping vacuum of "postgres.public.pgbench_accounts" --- relation no longer exists
```

自動VACUUMの実行記録(2)

- ログ出力の読み方

```
LOG:  automatic vacuum of table "postgres.public.pgbench_accounts": index scans: 1
        pages: 0 removed, 2186 remain, 0 skipped due to pins, 0 skipped frozen
        tuples: 33333 removed, 100000 remain, 0 are dead but not yet removable, oldest xmin: 668
        buffer usage: 6607 hits, 0 misses, 0 dirtied
        avg read rate: 0.000 MB/s, avg write rate: 0.000 MB/s
        system usage: CPU: user: 0.18 s, system: 0.02 s, elapsed: 0.85 s
```

  - index scans:インデックススキャンの回数
  - pages:削除、残り、ピン(ページのロック)待ちでスキップ(9.5以降)、凍結済みでスキップ(9.6以降)されたページ数
  - tuples:削除、残り、不要だがまだ削除できない行数
  - oldest xmin:最も古い挿入トランザクションのXID(10以降)
  - buffer usage:バッファの使用状況(メモリ、ディスク読み取り、ディスク書き込みのページ数)
  - avg read/write rate:平均ディスク読み取り・書き込み速度
  - system usage:システムの使用状況(ユーザ、システムのCPU時間、経過時間)

- インデックススキャンが多い=ワークメモリが不足してないか?
- 削除できない不要行が増えてないか?
- 自動VACUUMの実行時間、実行頻度、実行時間帯は問題ないか?

テーブルのVACUUM実行状況

- pg_stat_{all,sys,user}_tables統計情報ビュー

```
=# SELECT * FROM pg_stat_user_tables;
-[ RECORD 2 ]-------+------------------------------
relid               | 16522                          テーブルのOID
schemaname          | public                         テーブルの属するスキーマ名
relname             | pgbench_accounts               テーブル名
(省略)
n_live_tup          | 66666                          有効行数
n_dead_tup          | 0                              不要行数
n_mod_since_analyze | 0                              ANALYZE実行後の変更行数(9.4以降)
last_vacuum         | 2018-10-26 09:54:40.840178+09  手動によるVACUUMの最終実行日時
last_autovacuum     | 2018-10-26 09:56:17.085096+09  自動VACUUMによるVACUUMの最終実行日時
last_analyze        | 2018-10-26 09:54:40.881142+09  手動によるANALYZEの最終実行日時
last_autoanalyze    | 2018-10-26 09:56:17.411067+09  自動VACUUMによるANALYZEの最終実行日時
vacuum_count        | 1                              手動によるVACUUMの実行回数
autovacuum_count    | 2                              自動VACUUMによるVACUUMの実行回数
analyze_count       | 1                              手動によるANALYZEの実行回数
autoanalyze_count   | 2                              自動VACUUMによるANALYZEの実行回数
```

- 不要行が増えてないか？
- VACUUM/ANALYZEはちゃんと実行されてるか？

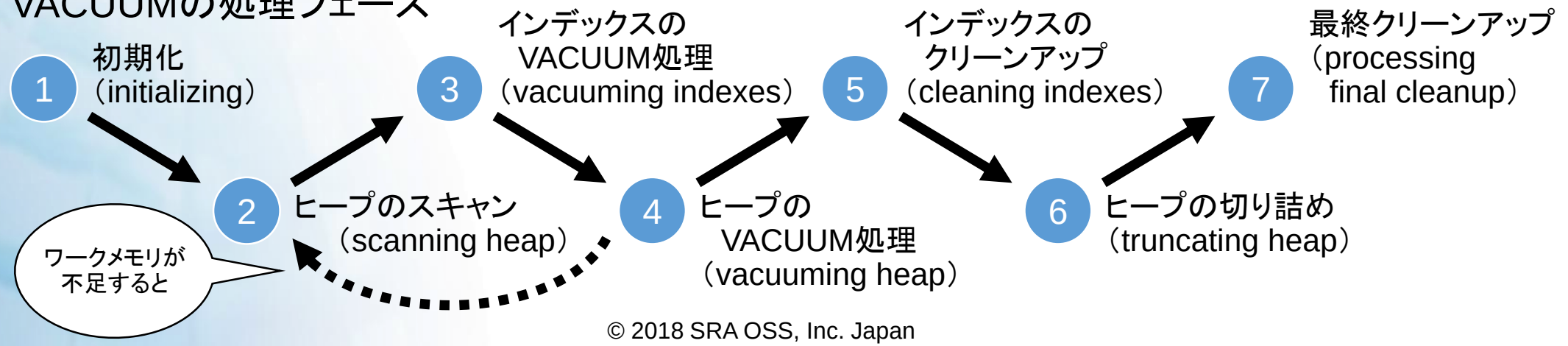
VACUUMの進捗状況

- pg_stat_progress_vacuum統計情報ビュー（9.4以降）

```
=# SELECT * FROM pg_stat_progress_vacuum;
-[ RECORD 1 ]------+---------------
pid                | 31570
datid              | 13284
datname            | postgres
relid              | 16566
phase              | vacuuming heap
heap_blks_total    | 69946
heap_blks_scanned  | 69946
heap_blks_vacuumed | 347
index_vacuum_count | 1
max_dead_tuples    | 11184810
num_dead_tuples    | 2133333
```

pid	プロセスID
datid	データベースのOID
datname	データベース名
relid	テーブルのOID
phase	VACUUMの処理フェーズ
heap_blks_total	ヒープ（テーブルのデータ部分）ブロック数
heap_blks_scanned	スキャン済みヒープブロック数
heap_blks_vacuumed	VACUUM処理済みヒープブロック数
index_vacuum_count	インデックスに対するVACUUM処理の実行回数
max_dead_tuples	回収可能な不要行数
num_dead_tuples	最後の回収済み不要行数

- VACUUMの処理フェーズ



VACUUMに関するパラメータの設定

VACUUM用ワークメモリの設定

- VACUUMで使うメモリを増やして、実行効率を上げる

パラメータ名	デフォルト値	説明
maintenance_work_mem	16MB（9.3以前） 64MB（9.4以降）	VACUUM、CREATE INDEXなどのメンテナンス作業で使うメモリの最大容量
autovacuum_work_mem（9.4以降）	-1	自動VACUUMで使うメモリの最大容量。手動VACUUMには影響しない。-1はmaintenance_work_memと同じ

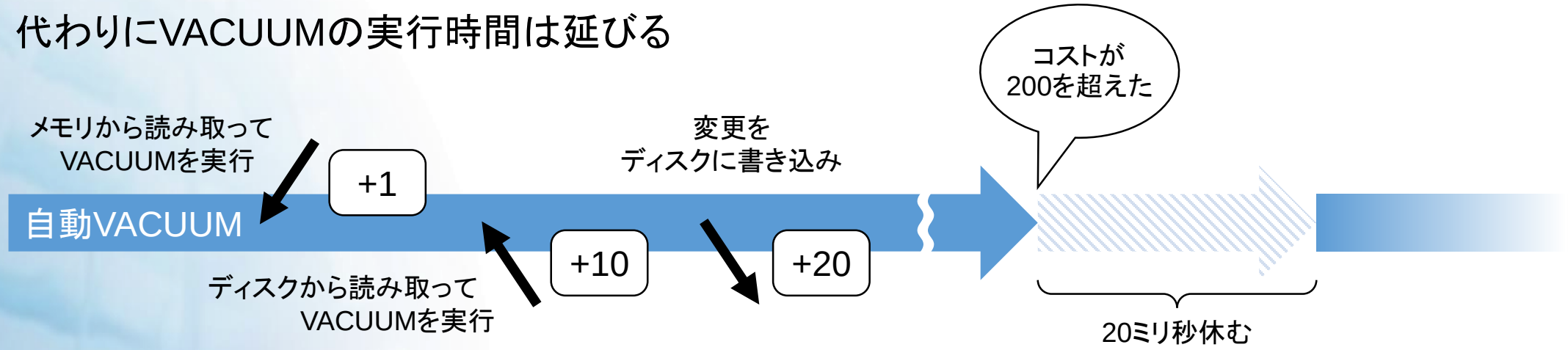
  - ワークメモリが不足すると、余計な処理が発生してしまう
  - 必要以上に増やしても効果はない
    - 不要行を1行回収するのに必要なメモリは6バイト、デフォルトの64MBでも約1000万行回収できる
    - 自動VACUUMの実行ログでindex scansが1より大きければ増やす

VACUUM遅延実行の設定

- VACUUMを休みながら実行して、ディスクI/Oの負荷を下げる

パラメータ名	デフォルト値	説明
vacuum_cost_delay	0	手動VACUUMの実行を休む時間。0は無効
vacuum_cost_page_hit	1	メモリから読み取ったページにVACUUMを実行するコスト
vacuum_cost_page_miss	10	ディスクから読み取ったページにVACUUMを実行するコスト
vacuum_cost_page_dirty	20	VACUUMを実行して変更のあったページをディスクに書き込むコスト
vacuum_cost_limit	200	手動VACUUMのコスト上限
autovacuum_vacuum_cost_delay	20ms	自動VACUUMの実行を休む時間。-1はvacuum_cost_delayと同じ
autovacuum_vacuum_cost_limit	-1	自動VACUUMのコスト上限。-1はvacuum_cost_limitと同じ

- 代わりにVACUUMの実行時間は延びる



自動VACUUMの設定

- 自動VACUUMの動作を調整

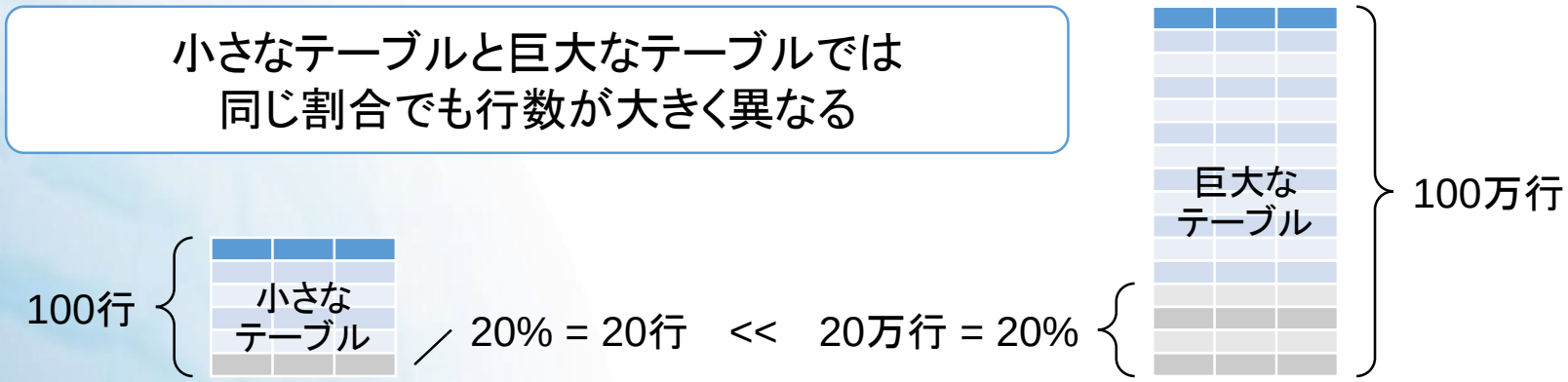
パラメータ名	デフォルト値	説明
autovacuum	on	自動VACUUMを有効にするか？
log_autovacuum_min_duration	-1	指定した時間以上かかった実行をログに記録。-1は無効
autovacuum_max_workers	3	同時に起動可能なautovacuum workerの最大数。要再起動
autovacuum_naptime	1min	データベースごとの監視間隔
autovacuum_vacuum_threshold	50	VACUUMの実行条件となる不要行数
autovacuum_analyze_threshold	50	ANALYZEの実行条件となる変更行数
autovacuum_vacuum_scale_factor	0.2	VACUUMの実行条件となる不要行数の割合
autovacuum_analyze_scale_factor	0.1	ANALYZEの実行条件となる変更行数の割合
autovacuum_freeze_max_age	200000000	XID周回問題対策VACUUMの実行条件となるトランザクション数。要再起動
autovacuum_multixact_freeze_max_age	400000000	XID周回問題対策VACUUMの実行条件となるマルチトランザクションにおけるトランザクション数。要再起動

テーブルごとの自動VACUUM設定

- 自動VACUUMの設定はテーブルごとにできる
  - 設定はALTER TABLE、確認は¥d+で行う
  - パラメータ名は基本的に同じだが、有効にするかはautovacuum_enabledで指定

```
=# ALTER TABLE pgbench_accounts
-#   SET (autovacuum_vacuum_scale_factor = 0.1, autovacuum_analyze_scale_factor = 0.05);
ALTER TABLE
=# ¥d+ pgbench_accounts
（省略）
オプション: fillfactor=100, autovacuum_vacuum_scale_factor=0.1,
autovacuum_analyze_scale_factor=0.05
```

小さなテーブルと巨大なテーブルでは
同じ割合でも行数が大きく異なる



巨大なテーブルではより小さなautovacuum_vacuum/analyze_scale_factorが適切

オープンソースとともに

SRA OSS, INC.