

OSS-DB Silver 技術解説無料セミナー

2020/07/19 開催

主題	運用管理（出題範囲 52 %）
副題	バックアップ方法 【重要度：7】

本日の講師



SRA OSS, Inc. 日本支社
正野 裕大

LPI-JAPAN

©SRAOSS, Inc. Japan / LPI-Japan / EDUCO all rights reserved.



#OSS-DB

■正野 裕大

- SRA OSS, Inc. 日本支社所属
- PostgreSQL の技術サポート・コンサルティング
- PostgreSQL トレーニング講師

■ SRA OSS, Inc. 日本支社 (<https://www.sraoss.co.jp/>)

- PostgreSQL などの OSS プロダクトの技術サポート・コンサルティング
- PowerGres 製品の製造、販売
- PostgreSQL トレーニング



■OSS-DBとは

オープンソースのデータベースソフトウェア「PostgreSQL」を扱うことができる技術力の認定です。様々な分野でPostgreSQLの利用拡大が進む中でOSS-DBの認定を持つことは、自分のキャリアのアピールにもつながります。

✓ OSS-DB Goldは設計やコンサルティングができる技術力の証明

PostgreSQLについての深い知識を持ち、データベースの設計や開発のほか、パフォーマンスチューニングやトラブルシューティングまで行えることが証明できます

✓ OSS-DB Silverは導入や運用ができる技術力の証明

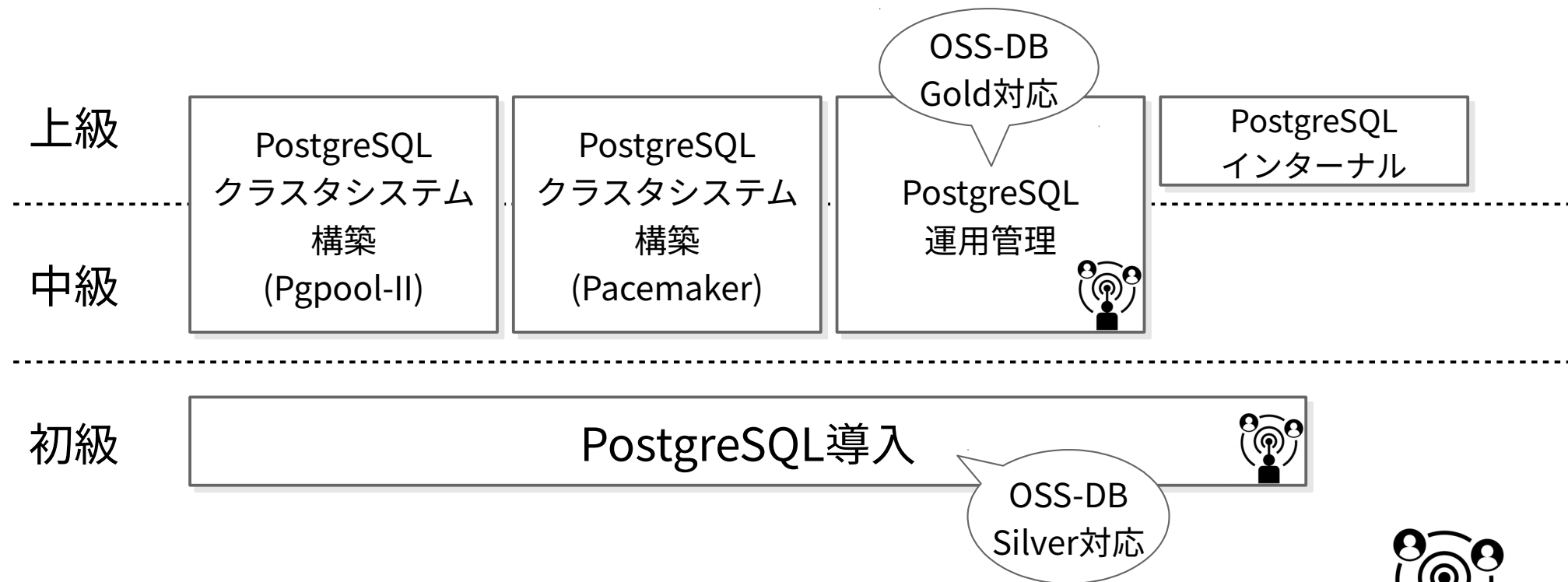
PostgreSQLについての基本的な知識を持ち、データベースの運用管理が行えるエンジニアとしての証明ができます

✓ 対象のバージョンはPostgreSQL 11



#OSS-DB

■ SRA OSS, Inc. 日本支社は PostgreSQLトレーニングコースを各種用意しています



オンライン対応

- OSS-DB Exam認定校第1号ならではの充実した内容
- 受講者のレベルに合わせたコースでOSS-DB試験にも対応した内容



#OSS-DB

■直近のトレーニング開催スケジュール

直近の開催スケジュール	
2020 年 8 月 17 日 ~ 18 日 (東京) 締切: 2020 年 7 月 29 日 PostgreSQL導入トレーニング	» 申込
2020 年 8 月 21 日 (東京) 締切: 2020 年 8 月 4 日 Pgpool-IIによるPostgreSQLクラスタ構築トレーニング	» 申込
2020 年 9 月 7 日 ~ 8 日 (東京) 締切: 2020 年 8 月 20 日 PostgreSQL導入トレーニング	» 申込
2020 年 9 月 9 日 ~ 10 日 (東京) 締切: 2020 年 8 月 24 日 PostgreSQL運用管理トレーニング	» 申込
2020 年 9 月 11 日 (東京) 締切: 2020 年 8 月 26 日 Pacemaker/DRBDによるPostgreSQLクラスタ構築トレーニング	» 申込
2020 年 9 月 14 日 ~ 15 日 (オンライン) 締切: 2020 年 8 月 27 日 PostgreSQL導入トレーニング	» 申込

■詳しくは

https://www.sraoss.co.jp/prod_serv/training/pgsql.php

PostgreSQLのバックアップ方法



#OSS-DB

運用管理 (52%)

pg_settings

バックアップ方法 【重要度：7】

- 説明：
PostgreSQLのバックアップ方法に関する理解を問う
- 主要な知識範囲：
各種バックアップコマンドの使い方
ファイルシステムレベルのバックアップとリストア
ポイントインタイムリカバリ(PITR)の概念と手順
トランザクションログ(WAL)とWALアーカイブ
pg_start_backup() / pg_stop_backup()
COPY文(SQL)、¥copyコマンド(psql)の使い方
- 重要な用語、コマンド、パラメータなど：
pg_dump
pg_dumpall
pg_restore
psql
pg_basebackup
PITR
recovery.conf
COPY
¥copy

<https://oss-db.jp/outline/silver>

PostgreSQLのバックアップ方法



#OSS-DB

運用管理 (52%)

pg_settings

バックアップ方法 【重要度：7】

- 説明：
PostgreSQLのバックアップ方法に関する理解を問う
- 主要な知識範囲：
 - 各種バックアップコマンドの使い方
 - ファイルシステムレベルのバックアップとリストア
 - ポイントインタイムリカバリ(PITR)の概念と手順
 - トランザクションログ(WAL)とWALアーカイブ
 - pg_start_backup() / pg_stop_backup()
 - COPY文(SQL)、¥copyコマンド(psql)の使い方
- 重要な用語、コマンド、パラメータなど：

pg_dump
pg_dumpall
pg_restore
psql
pg_basebackup
PITR
recovery.conf
COPY
¥copy

■ 仮想マシンossdb-01を用意

 #OSS-DB

■ CentOS 7.8

```
[postgres@ossdb-01 ~]$ cat /etc/redhat-release
CentOS Linux release 7.8.2003 (Core)
```

■ ossdb-01 = 192.168.1.101

```
[postgres@ossdb-01 ~]$ hostname --ip
127.0.0.1 192.168.1.101
```


■ PostgreSQL



#OSS-DB

- 両マシンとも開発コミュニティ公式のパッケージからPostgreSQL 11をインストールして環境変数(*)を調整済み

```
[postgres@ossdb-01 ~]$ yum list installed | grep postgresql11
postgresql11.x86_64                11.8-1PGDG.rhel7      @pgdg11
postgresql11-contrib.x86_64       11.8-1PGDG.rhel7      @pgdg11
postgresql11-devel.x86_64          11.8-1PGDG.rhel7      @pgdg11
postgresql11-docs.x86_64           11.8-1PGDG.rhel7      @pgdg11
postgresql11-libs.x86_64           11.8-1PGDG.rhel7      @pgdg11
postgresql11-llvmjit.x86_64        11.8-1PGDG.rhel7      @pgdg11
postgresql11-odbc.x86_64            12.01.0000-1PGDG.rhel7 @pgdg11
postgresql11-plperl.x86_64         11.8-1PGDG.rhel7      @pgdg11
postgresql11-plpython.x86_64       11.8-1PGDG.rhel7      @pgdg11
postgresql11-plpython3.x86_64      11.8-1PGDG.rhel7      @pgdg11
postgresql11-pltcl.x86_64          11.8-1PGDG.rhel7      @pgdg11
postgresql11-server.x86_64         11.8-1PGDG.rhel7      @pgdg11
postgresql11-tcl.x86_64             2.4.0-2.rhel7.1       @pgdg11
postgresql11-test.x86_64           11.8-1PGDG.rhel7      @pgdg11
```

```
[postgres@ossdb-01 ~]$ echo $PATH
/usr/pgsql-11/bin:/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin
[postgres@ossdb-01 ~]$ echo $MANPATH
/usr/pgsql-11/share/man:
[postgres@ossdb-01 ~]$ echo $PGDATA
/var/lib/pgsql/11/data
```

(*) 環境変数: コマンドやアプリケーションの実行時に参照される変数のこと。



■バックアップするデータの作製

■initdbでデータベースクラスタの初期化



■最小限の設定ファイル調整



■PostgreSQLを起動してバックアップ対象データベースを作製



■pgbenchでバックアップ対象データベースにデータを作製

■ initdbの実行

- データベースクラスタ(*)を初期化するPostgreSQLコマンド

```
[postgres@ossdb-01 ~]$ initdb --no-locale --encoding=UTF8
```

- --no-locale

ローケールを使用しない（慣習的にローケールを使用しないことが推奨されている）

- --encoding=ENCODING

データベースのデフォルト文字エンコーディングを指定

- データベースクラスタパスの指定方法

\$PGDATAで指定するか--pgdata=《データベースクラスタパス》で指定

(*) データベースクラスタ: ストレージに記録されるPostgreSQLのデータ一式を格納するディレクトリ。



■ \$PGDATA/postgresql.conf の編集

■ PostgreSQLの設定ファイル

```
@@ -56,7 +56,7 @@
```

```
# - Connection Settings -
```

```
-#listen_addresses = 'localhost'          # what IP address(es) to listen on;
+listen_addresses = '*'                    # what IP address(es) to listen on;
                                           # comma-separated list of addresses;
                                           # defaults to 'localhost'; use '*' for all
                                           # (change requires restart)
```

■ listen_addresses

PostgreSQLが接続を待ち受けるIPアドレス

「*」は全てのIPインターフェイスで待ち受け



■ pg_ctlで起動

```
[postgres@ossdb-01 ~]$ pg_ctl start
```

- PostgreSQLを起動・停止・ステータス確認するPostgreSQLコマンド
操作対象となるデータベースクラスターをセットで指定する
指定方法は\$PGDATAか--pgdata=DATADIR

■ データベース(*)benchmarkを作製

```
[postgres@ossdb-01 ~]$ createdb benchmark
```

(*) データベース: テーブルやインデックスを格納する領域。initdb直後はtemplate0, template1, postgresの3つのデータベースが存在している。

ただし、それらはメンテナンス用データベースなので、アプリのデータを入れてはいけない。

■ pgbench



- PostgreSQLに同梱されているベンチマークツール
- TPC-B(*)というベンチマークシナリオを実行して性能測定する

```
[postgres@ossdb-01 ~]$ pgbench --initialize --scale=30 benchmark
```

- --initialize
ベンチマークテーブルの初期化
- --scale=NUM
ベンチマークテーブルの規模
--scale=1でデータベースサイズは15MB程度になる
- 第一引数
pgbenchの対象データベース

(*) TPC-B: トランザクション処理性能評議会が定めたシナリオ。銀行口座、銀行支店、銀行窓口担当者などの業務をモデル化している。

TPC-Bは1995/06/06に廃止されていて、他にも様々なシナリオがあったり、サードパーティー製のベンチマークツールがあるが、PostgreSQLの簡易的なベンチマークは専らpgbenchが用いられる。



方略	バックアップ対象	方法
論理バックアップ	PostgreSQLのデータ	PostgreSQLコマンド
物理バックアップ	PostgreSQLのデータを構成する全てのディレクトリ・ファイル	OSコマンド or PostgreSQLコマンド
PITR (Point In Time Recovery)	物理バックアップに加えてWALファイル(*1)	物理バックアップ and WALアーカイブ(*2)

(*1) WALファイル: データの保全性と書き込み速度を担保するファイル。詳細は後述。

(*2) WALアーカイブ: WALファイルをバックアップするPostgreSQLのしくみ。詳細は後述。

方略	バックアップ対象	方法
論理バックアップ	PostgreSQLのデータ	PostgreSQLコマンド
物理バックアップ	PostgreSQLのデータを構成する全てのディレクトリ・ファイル	OSコマンド or PostgreSQLコマンド
PITR (Point In Time Recovery)	物理バックアップに加えてWALファイル(*1)	物理バックアップ and WALアーカイブ(*2)

(*1) WALファイル: データの保全性と書き込み速度を担保するファイル。詳細は後述。

(*2) WALアーカイブ: WALファイルをバックアップするPostgreSQLのしくみ。詳細は後述。

■ PostgreSQLのデータをPostgreSQLコマンドでバックアップ



#OSS-DB

■ pg_dumpコマンド

指定のデータベースをバックアップ

■ pg_dumpallコマンド

データベース全体のバックアップ

■ pg_dumpしたバックアップのリストア方法

■ pg_dumpallしたバックアップのリストア方法



■ pg_dump

- データベース単位のバックアップを行なうPostgreSQLコマンド
- 実行時に使うデータベースユーザはデータベースの所有者かスーパーユーザ

```
[postgres@ossdb-01 ~]$ pg_dump --file=benchmark.sql benchmark
```

- --file=FILENAME
バックアップの出力先
- 第一引数
バックアップ対象データベース

■ 出力形式を選択可能

- --format=p テキスト形式（PostgreSQLデータがSQL文のテキストとして出力される）※デフォルト
- --format=c カスタム形式（PostgreSQL独自の圧縮形式）
- --format=t tar形式
- --format=d ディレクトリ形式（テーブル単位のカスタム形式 / --jobs=NUMで並列バックアップ可能）



■ pg_dumpall

- データベース全体のバックアップを行なうPostgreSQLコマンド
- 実行時に使うデータベースユーザはスーパーユーザのみ

```
[postgres@ossdb-01 ~]$ pg_dumpall --file=db_all.sql
```

- --file=FILENAME
バックアップの出力先

■ 出力形式

- テキスト形式（PostgreSQLのデータがSQL文のテキストとして出力される）のみ

■テキスト形式

■psqlコマンドを使う

```
[postgres@ossdb-01 ~]$ createdb benchmark02-t
[postgres@ossdb-01 ~]$ psql --file=benchmark.sql benchmark02-t
```

- file=FILENAME リストアに使うバックアップファイルを指定
- 第一引数 リストア先のデータベースを指定

■それ以外の形式

■pg_restoreコマンドを使う

```
[postgres@ossdb-01 ~]$ createdb benchmark02-c
[postgres@ossdb-01 ~]$ pg_restore --dbname=benchmark02-c benchmark.dump
```

- dbname=DBNAME リストア先のデータベースを指定
- 第一引数 リストアに使うバックアップファイル、ディレクトリを指定
- ディレクトリ形式はリストア時に--jobos=NUMで並列リストア可能



■ データベースクラスタを初期化してリストアする

- 既存のPostgreSQLを停止

```
[postgres@ossdb-01 ~]$ pg_ctl stop
```

- 既存のデータベースクラスタは待避（または削除）

```
[postgres@ossdb-01 ~]$ mv $PGDATA $PGDATA.back01
```

- データベースクラスタの初期化と設定ファイルのリストア(*)

```
[postgres@ossdb-01 ~]$ initdb --no-locale --encoding=UTF8
[postgres@ossdb-01 ~]$ cp $PGDATA.bak01/postgresql.conf $PGDATA
```

- 起動してpsqlコマンドでリストア

```
[postgres@ossdb-01 ~]$ pg_ctl start
[postgres@ossdb-01 ~]$ psql --file=db_all.sql postgres
```

(*) 論理バックアップは「PostgreSQLデータ」のバックアップです。設定ファイルはバックアップされないため、別途自分でバックアップ、リストアする必要があります。

方略	バックアップ対象	方法
論理バックアップ	PostgreSQLのデータ	PostgreSQLコマンド
物理バックアップ	PostgreSQLのデータを構成する全てのディレクトリ・ファイル	OSコマンド or PostgreSQLコマンド
PITR (Point In Time Recovery)	物理バックアップに加えてWALファイル(*1)	物理バックアップ and WALアーカイブ(*2)

(*1) WALファイル: データの保全性と書き込み速度を担保するファイル。詳細は後述。

(*2) WALアーカイブ: WALファイルをバックアップするPostgreSQLのしくみ。詳細は後述。

■ コールドバックアップ

PostgreSQLを停止してOSコマンドで\$PGDATAを物理バックアップ



#OSS-DB

■ オンラインバックアップ

PostgreSQLを稼働したままPostgreSQLコマンド / OSコマンドで物理バックアップ

■ pg_basebackupコマンド

■ 低レベルAPIを使う方法

■ 物理バックアップのリストア方法



■ コールドバックアップ

PostgreSQLを停止してOSコマンドで\$PGDATAを物理バックアップ

■ PostgreSQLを停止

```
[postgres@ossdb-01 ~]$ pg_ctl stop
```

■ 任意のOSコマンド (cp, tar, rsync ...) で\$PGDATAをバックアップ

```
[postgres@ossdb-01 ~]$ cp -a $PGDATA 《バックアップ領域》
```

■ PostgreSQLを起動

```
[postgres@ossdb-01 ~]$ pg_ctl start
```

■ コールドバックアップは必ずPostgreSQLを停止して行なう

- PostgreSQLの稼働中にOSコマンドで取得したバックアップは中身が不正な状態なのでリストアできない

■ pg_basebackup

PostgreSQLを起動したまま\$PGDATAの物理コピーが取得できるPostgreSQLコマンド
オンラインバックアップ



#OSS-DB

- デフォルトの設定でローカルのPostgreSQLに対して実行可能

```
[postgres@ossdb-01 ~]$ pg_ctl status
pg_ctl: server is running (PID: 3332)
/usr/pgsql-11/bin/postgres
[postgres@ossdb-01 ~]$ pg_basebackup --pgdata=$PGDATA.bak02
```

} PostgreSQLは起動中

- --pgdata=DIRECTORY
物理バックアップの出力先



■ pg_start_backup関数とpg_stop_backup関数

- PostgreSQLをバックアップモードを開始・終了するSQL関数
- バックアップモード中はPostgreSQLを起動したまま\$PGDATAの物理コピー取得可能

```
postgres=# SELECT pg_start_backup('label', true);
postgres=# -- この間なら$PGDATAをOSコマンドで物理コピーOK
postgres=# SELECT pg_stop_backup();
```

■ pg_start_backup(label text, fast boolean)

- label text 一意な識別子
- fast boolean チェックポイント(*)の高速化（その分I/O負荷は高まる）

- pg_basebackupコマンド以前の古い手法だが、クラウド環境や仮想環境では使われやすい
→物理コピーを仮想化基盤のスナップショットに任せられる

(*) チェックポイント: メモリ上のPostgreSQLのデータをデータファイルに反映させること。詳細は後述。

■物理バックアップデータは単体で起動できる



#OSS-DB

■\$PGDATAに配置してpg_ctl startで起動

```
[postgres@ossdb-01 ~]$ pg_ctl stop
[postgres@ossdb-01 ~]$ rm -rf $PGDATA
[postgres@ossdb-01 ~]$ cp -a $PGDATA.bak02 $PGDATA
[postgres@ossdb-01 ~]$ pg_ctl start
```

■pg_ctl --pgdata=... startと起動するデータベースクラスタを明示的に指定して起動

```
[postgres@ossdb-01 ~]$ pg_ctl --pgdata=$PGDATA.bak02 start
[postgres@ossdb-01 ~]$ pg_ctl --pgdata=$PGDATA.bak02 stop
```

方略	バックアップ対象	方法
論理バックアップ	PostgreSQLのデータ	PostgreSQLコマンド
物理バックアップ	PostgreSQLのデータを構成する全てのディレクトリ・ファイル	OSコマンド or PostgreSQLコマンド
PITR (Point In Time Recovery)	物理バックアップに加えてWAL ファイル(*1)	物理バックアップ and WALアーカイブ(*2)

(*1) WALファイル: データの保全性と書き込み速度を担保するファイル。詳細は後述。

(*2) WALアーカイブ: WALファイルをバックアップするPostgreSQLのしくみ。詳細は後述。

■ PITRとは



#OSS-DB

- 物理バックアップに加えてWALのバックアップ（WALアーカイブ）も行なう

■ WALとは

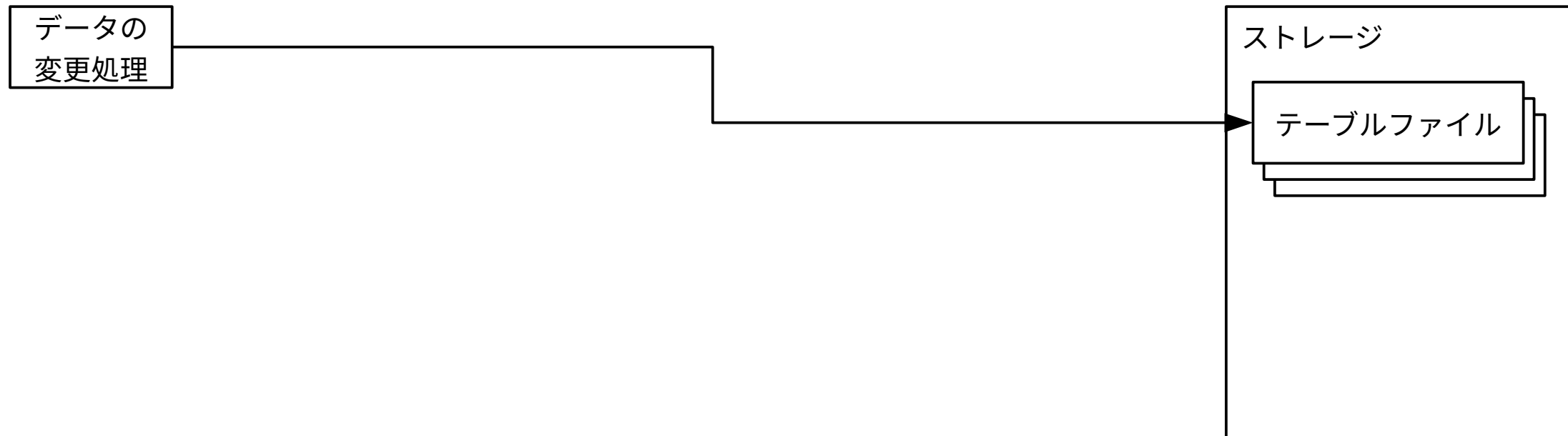
- PITRの理解にはWALの理解が必要なので先に解説

■前提: PostgreSQLデータのデータ書き込みについて



#OSS-DB

- ストレージに直接書き込むのは高コストなのでやりたくない

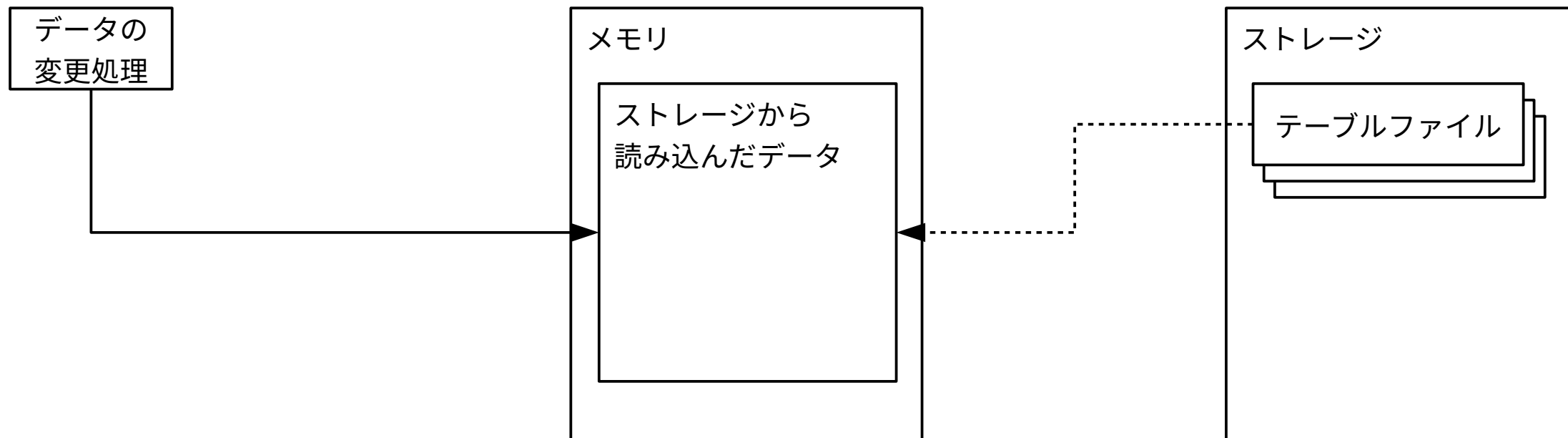


■前提: PostgreSQLデータのデータ書き込みについて



#OSS-DB

- ストレージに直接書き込むのは高コストなのでやりたくない
- そこで: メモリを介する

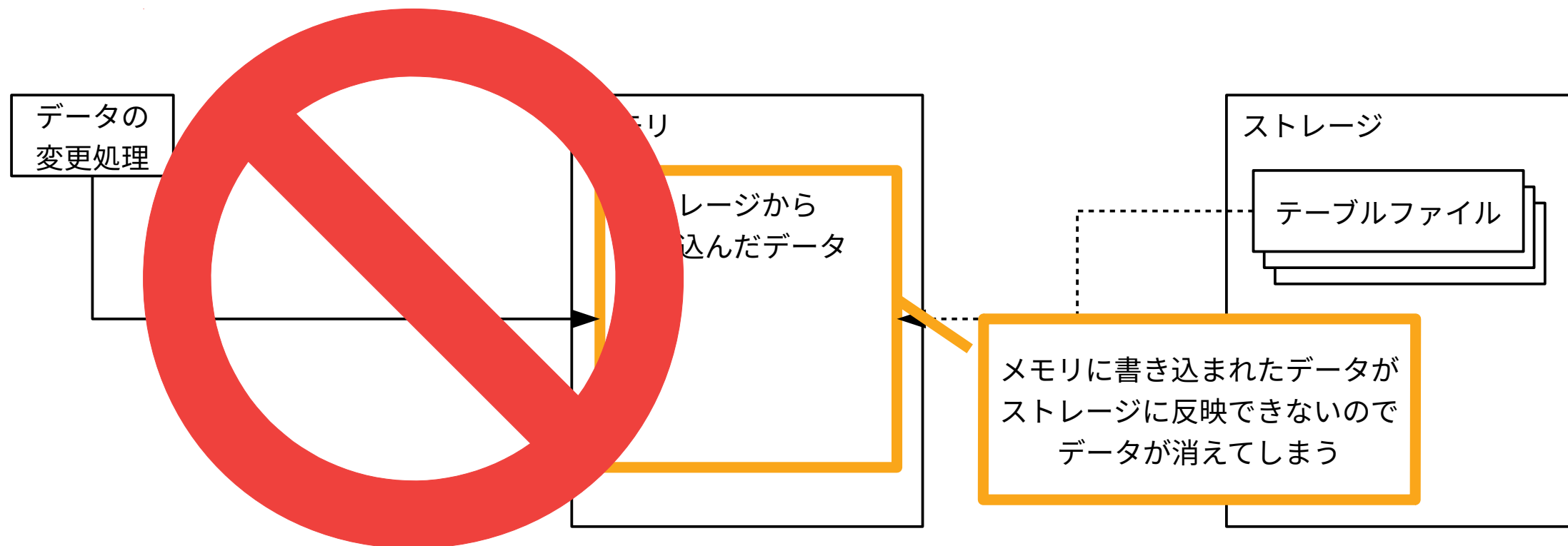


■前提: PostgreSQLデータのデータ書き込みについて



#OSS-DB

- ストレージに直接書き込むのは高コストなのでやりたくない
- そこで: メモリを介する **しかし: メモリにデータを書き込むだけではPostgreSQLがクラッシュするとデータが消えてしまう**

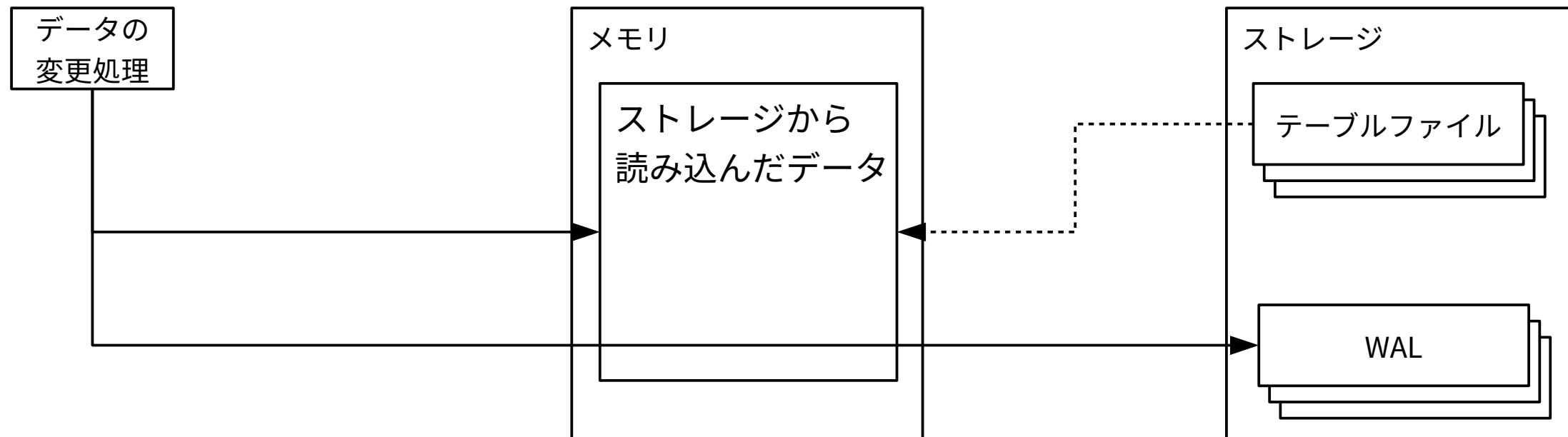


■前提: PostgreSQLデータのデータ書き込みについて



#OSS-DB

- ストレージに直接書き込むのは高コストなのでやりたくない
- そこで: メモリを介する しかし: メモリにデータを書き込むだけではPostgreSQLがクラッシュするとデータが消えてしまう
- そこで: データの変更内容をWALとメモリに書き込む(*)



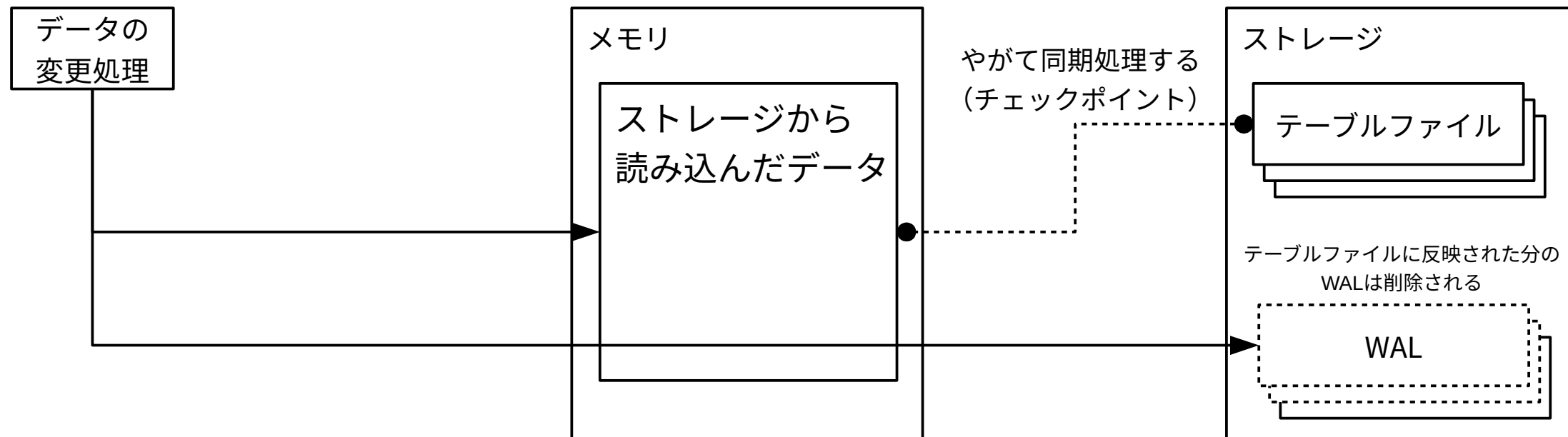
(*) シーケンシャルに追記するのでテーブルファイルに書き込むよりずっと高速です。

■前提: PostgreSQLデータのデータ書き込みについて



#OSS-DB

- ストレージに直接書き込むのは高コストなのでやりたくない
- そこで: メモリを介する しかし: メモリにデータを書き込むだけではPostgreSQLがクラッシュするとデータが消えてしまう
- そこで: データの変更内容をWALとメモリに書き込む
- メモリとテーブルファイルが同期（チェックポイント）したら不要になったWALは削除される

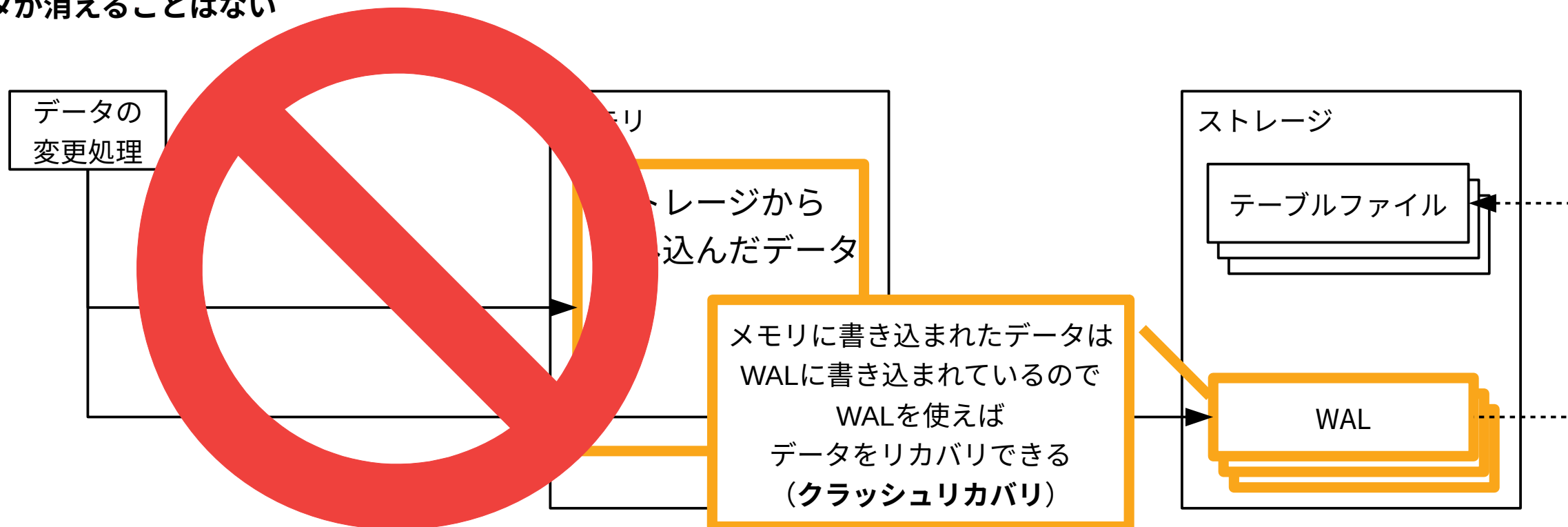


■前提: PostgreSQLデータのデータ書き込みについて



#OSS-DB

- ストレージに直接書き込むのは高コストなのでやりたくない
- そこで: メモリを介する しかし: メモリにデータを書き込むだけではPostgreSQLがクラッシュするとデータが消えてしまう
- そこで: データの変更内容をWALとメモリに書き込む
- メモリとテーブルファイルが同期 (チェックポイント) したら不要になったWALは削除される
- これなら: チェックポイント前にPostgreSQLがクラッシュしてもデータの変更内容はストレージに記録されているのでデータが消えることはない



■ PITRとは



#OSS-DB

- 物理バックアップに加えてWALのバックアップ（WALアーカイブ）も行なう

■ WALとは

- PITRの理解にはWALの理解が必要なので先に解説

■ PITRとは



#OSS-DB

- 物理バックアップに加えてWALのバックアップ（WALアーカイブ）も行なう

■ WALとは

- データの保全性と書き込み速度を担保する仕組み
- 不要になると削除される
- PostgreSQLがダウンしてもWALを使ってデータのリカバリができる（クラッシュリカバリ）

■ PITRとは



#OSS-DB

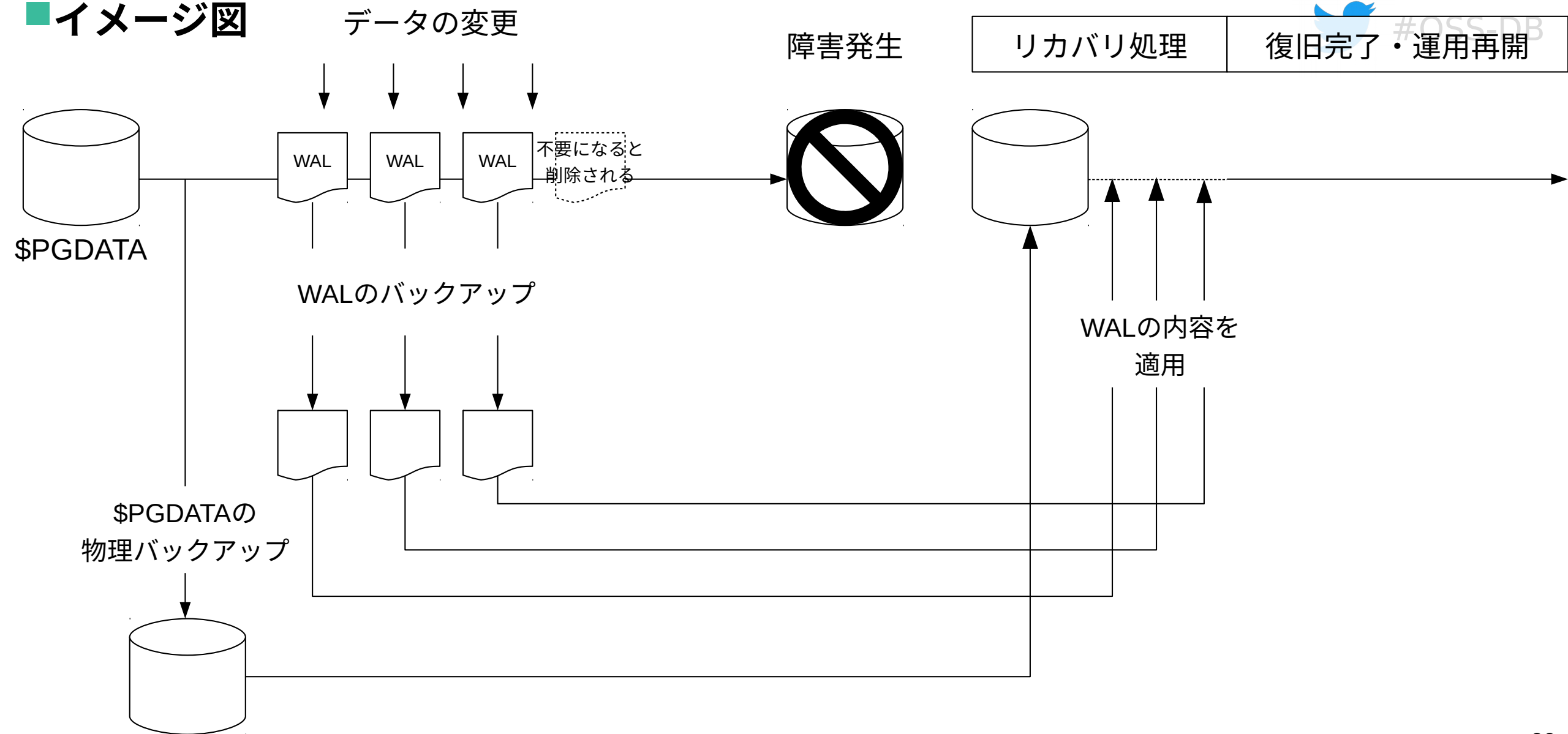
- 物理バックアップに加えてWALのバックアップ（WALアーカイブ）も行なう
- 物理バックアップにバックアップしたWALを適用してデータのリカバリができる
- バックアップしたWALが適用できる限り任意の時点にデータをリカバリできる

■ WALとは

- データの保全性と書き込み速度を担保する仕組み
- 不要になると削除される
- PostgreSQLがダウンしてもWALを使ってデータのリカバリができる（クラッシュリカバリ）

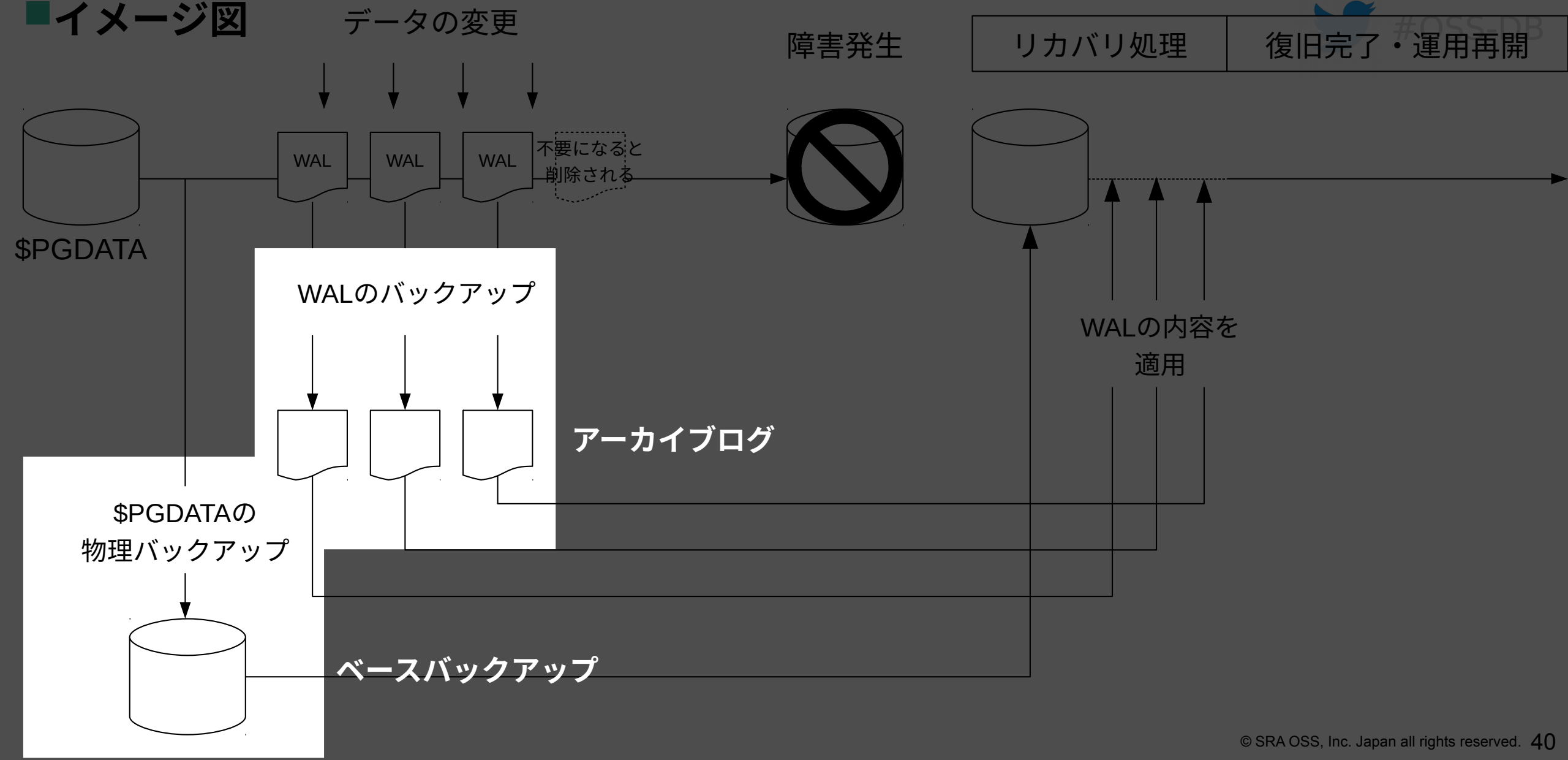
PITR (Point In Time Recovery)

イメージ図



PITR (Point In Time Recovery)

イメージ図



■ ベースバックアップとアーカイブログの格納領域を作製



```
[postgres@ossdb-01 ~]$ mkdir $HOME/11/backups/base # ベースバックアップ
[postgres@ossdb-01 ~]$ mkdir $HOME/11/backups/arc # アーカイブログ
```

■ WALアーカイブの設定（次のページへ続く）

■ postgresql.confの編集と適用

```
@@ -218,9 +218,9 @@ min_wal_size = 80MB

# - Archiving -

-#archive_mode = off                # enables archiving; off, on, or always
+archive_mode = on                  # enables archiving; off, on, or always
                                     # (change requires restart)
-#archive_command = '               # command to use to archive a logfile segment
+archive_command = 'cp "%p" "/var/lib/pgsql/11/backups/arc/%f"' # command ...
                                     # e.g. 'test ! -f /mnt/server/archivedir/%f && cp ...
```



■ WALアーカイブの設定（前のページの続き）

■ archive_mode

on にすると不要になって削除されるWALをアーカイブ領域に転送するようになる

■ archive_command

archive_mode = on のとき、実行するOSにコマンド
→ アーカイブ領域への転送方法はDB管理者が決める

■ %p: アーカイブ対象のWALファイルのパス名

■ %f: アーカイブ対象のWALファイル名

■ postgresql.confの反映

```
[postgres@ossdb-01 ~]$ pg_ctl restart
```

■ 設定ファイルの変更反映は**restart**か**reload**

■ 変更したパラメータによって決まる



■ ベースバックアップの取得

ベースバックアップ = \$PGDATAの物理バックアップ

■ pg_basebackupでOK

```
[postgres@ossdb-01 ~]$ pg_basebackup --pgdata=$HOME/11/backups/base/001 ↵
--wal-method=none
```

■ --wal-method=none

物理バックアップ対象にWALを含めない

(WALはWALアーカイブしているので含める必要がない)



■ benchmarkデータベースにデータ書き込み

pgbenchでベンチマークを実行してデータ書き込みを発生させる

```
[postgres@ossdb-01 ~]$ pgbench --client=10 --time=30 benchmark
```

■ --client=NUM

ベンチマーククライアントの同時接続数

■ --time=NUM

ベンチマーク実行時間 (秒)

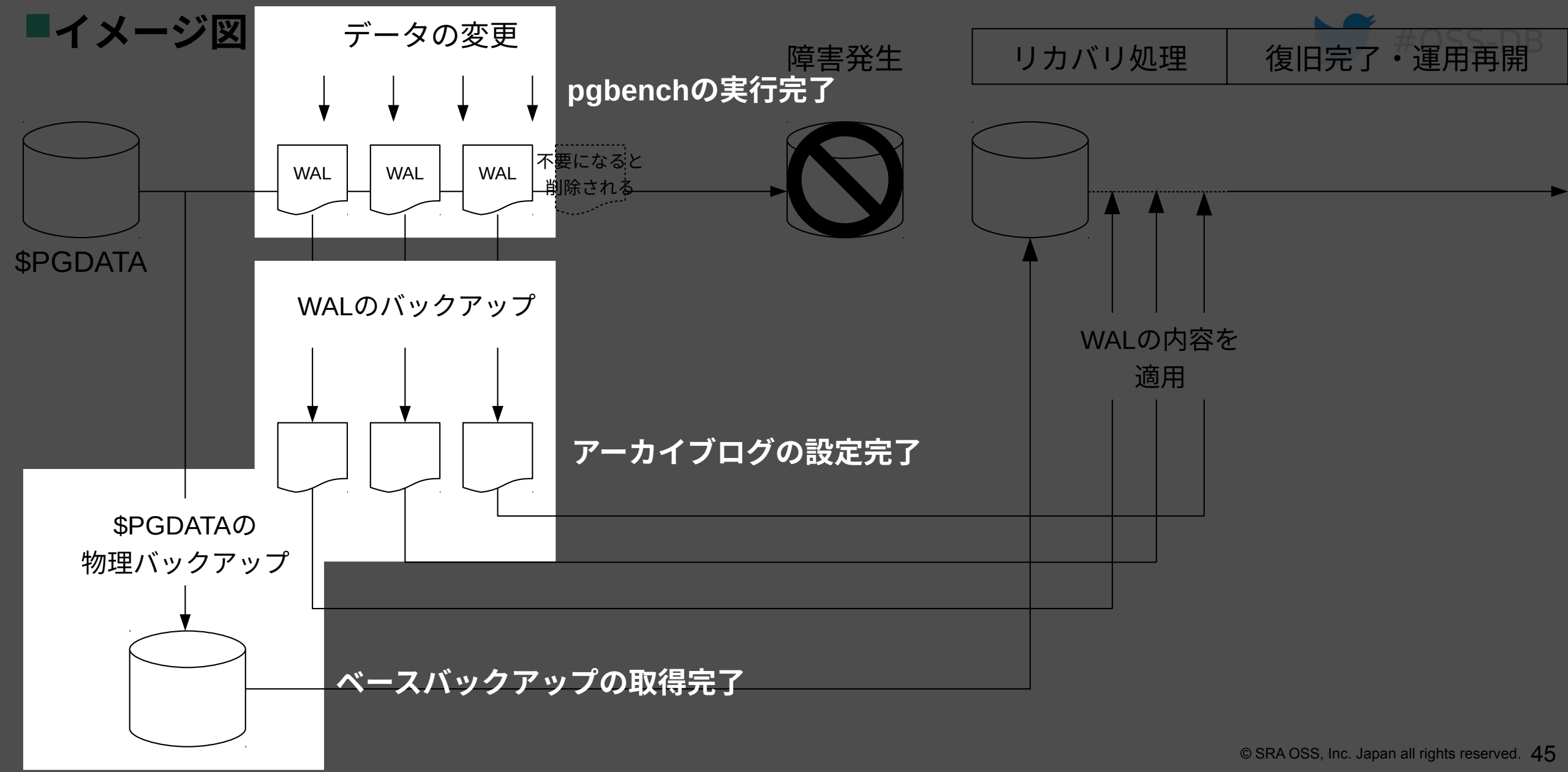
■ 履歴テーブルから最新のデータ行を確認しておく(*)

```
benchmark=# SELECT * FROM pgbench_history ORDER BY mtime LIMIT 1;
 tid | bid | aid  | delta |          mtime          | filler
-----+-----+-----+-----+-----+-----
  28 |   5 | 917249 | -1794 | 2020-07-13 11:29:37.957632 |
(1 row)
```

(*) 履歴テーブルのタイムスタンプを、昇順でソート(ORDER BY)して、先頭の1行だけを表示(LIMIT 1)しています。

PITR (Point In Time Recovery)

イメージ図





■ ベンチマークを実行したデータベースクラスタを強制終了

```
[postgres@ossdb-01 ~]$ pg_ctl stop --mode=immediate
```

■ --mode=MODE

シャットダウンモードの指定

immediateは強制終了（PostgreSQLがクラッシュするので検証作業以外で使わない）

■ 強制終了したデータベースクラスタを待避

```
[postgres@ossdb-01 ~]$ mv $PGDATA $PGDATA.crash
```



■ ベースバックアップを\$PGDATAに配置してログを削除(*)

```
[postgres@ossdb-01 ~]$ cp -a $HOME/11/backups/base/001 $PGDATA
[postgres@ossdb-01 ~]$ rm -rf $PGDATA/pg_log/*
```

■ リカバリ設定ファイル(recovery.conf)作製

```
[postgres@ossdb-01 ~]$ vi $PGDATA/recovery.conf
[postgres@ossdb-01 ~]$ cat $PGDATA/recovery.conf
restore_command = 'cp "/var/lib/pgsql/11/backups/arc/%f" "%p"'
```

■ recover.conf

- \$PGDATA/recovery.confがあるとPostgreSQLはリカバリモードで起動
 - **restore_command** アーカイブログを適用するOSコマンドを指定
 - **recovery_target_time** 指定時刻でリカバリを終了

(*) これからリカバリして稼働させるデータベースクラスタに、ベースバックアップ取得時点までのログファイルを残しておく意味はありません。



■ 未アーカイブのWALを\$PGDATA/walに配置

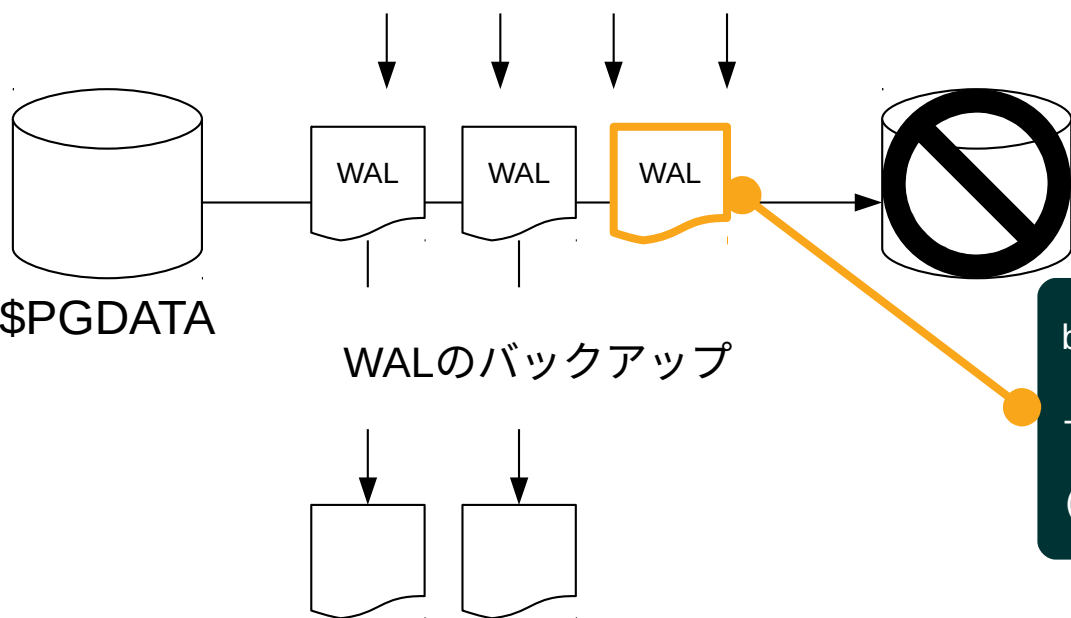
```
[postgres@ossdb-01 ~]$ ls $PGDATA.crash/pg_wal
[postgres@ossdb-01 ~]$ ls $HOME/11/backups/arc
[postgres@ossdb-01 ~]$ cp $PGDATA.crash/pg_wal/* $PGDATA/pg_wal/
```

■ なぜやるのか？

■ WALがアーカイブされないまま障害が発生する場合もある

データの変更

障害発生



```
benchmark=# SELECT * FROM pgbench_history ORDER BY mtime LIMIT 1;
tid | bid | aid  | delta | mtime                | filler
-----+-----+-----+-----+-----+-----
 28 |   5 | 917249 | -1794 | 2020-07-13 11:29:37.957632 |
(1 row)
```




■ リカバリ実行

```
[postgres@ossdb-01 ~]$ pg_ctl start
```

- リカバリ中はデータベースに接続できない

→ リカバリするアーカイブログが多いほどデータベースに接続できない時間は長くなる

■ リカバリ完了

- recovery.conf → recovery.done にリネームされる
- ログファイルに「LOG: archive recovery complete」が出力される
- 障害発生直前の最新データまで復旧できている

```
benchmark=# SELECT * FROM pgbench_history ORDER BY mtime LIMIT 1;
 tid | bid | aid  | delta | mtime                | filler
-----+-----+-----+-----+-----+-----
  28 |   5 | 917249 | -1794 | 2020-07-13 11:29:37.957632 |
(1 row)
```

方略	バックアップ対象	方法	データの復旧時点
論理バックアップ	PostgreSQLのデータ	pg_dump, pg_dumpall	
物理バックアップ (コールド)	PostgreSQLのデータを 構成する全てのディレク トリ・ファイル	OSコマンド	バックアップ取得時点
物理バックアップ (オンライン)		pg_basebackup, pg_start_backup() と pg_stopbackup()	
PITR (Point In Time Recovery)	物理バックアップと WALファイル	物理バックアップ と WALアーカイブ	ベースバックアップ取 得時点からアーカイブ ログが適用できる範囲 の任意の地点