OSS-DB

OSS-DB Silver
技術解説無料セミナー
2023/10/15 開催

主題	S2 運用管理 （52%）
副題	S2.4 バックアップ方法 【重要度：7】

本日の講師

SRA OSS LLC
陳 寧韡（チン ネイイ）

LPI-JAPAN

講師紹介

#OSS-DB

- 陳 寧華（チン ネイイ）
- SRA OSS LLC /データベース 技術グループ 所属
- PostgreSQL の技術サポート、トレーニング講師

- SRA OSS LLC (https://www.sraoss.co.jp/)
- OSS のコンサルティング、サポートサービス
- PostgreSQL のトレーニング
- PowerGres の開発 & 販売

OSS-DBとは

#OSS-DB

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

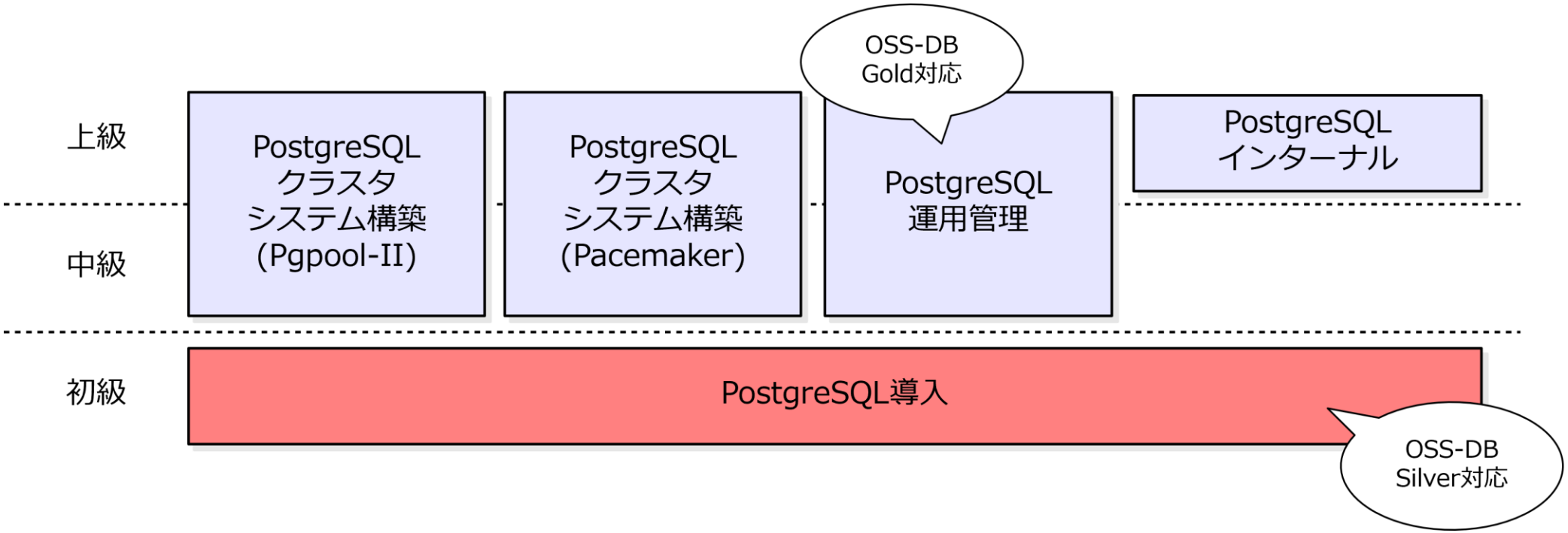
  PostgreSQLについての基本的な知識を持ち、データベースの運用管理が行えるエンジニアとしての証明ができます

- ✓対象のバージョンはPostgreSQL 14

プロモーション 1/2

#OSS-DB

- **SRA OSS では幅広いコースを用意しています**



- **OSS-DB 認定校**
- **OSS-DB Silver/Gold Ver.3.0 に対応しています**

プロモーション 2/2

#OSS-DB

- 直近のスケジュール
(https://www.sraoss.co.jp/prod_serv/training/pgsql/schedule.php)

- 会場とオンラインの開催

« 前月　　次月 »

日程	コース名	場所	価格	
2023 年 10 月 16 日 ～ 17 日 締切 : 2023 年 9 月 27 日	PostgreSQL導入トレーニング	オンライン	88,000円 （税抜価格 80,000円）	» 締切 » 詳細
2023 年 10 月 18 日 ～ 19 日 締切 : 2023 年 9 月 29 日	PostgreSQL運用管理トレーニング	オンライン	165,000円 （税抜価格 150,000円）	» 締切 » 詳細
2023 年 11 月 13 日 ～ 14 日 締切 : 2023 年 10 月 25 日	PostgreSQL導入トレーニング	東京	88,000円 （税抜価格 80,000円）	» 申込 » 詳細
2023 年 11 月 15 日 ～ 16 日 締切 : 2023 年 10 月 27 日	PostgreSQL運用管理トレーニング	東京	165,000円 （税抜価格 150,000円）	» 満席 » 詳細
2023 年 11 月 17 日 締切 : 2023 年 10 月 31 日	PacemakerによるPostgreSQLクラスタ構築トレーニング	東京	85,800円 （税抜価格 78,000円）	» 申込 » 詳細
2023 年 11 月 22 日 締切 : 2023 年 11 月 6 日	PostgreSQLインターナル講座	オンライン	55,000円 （税抜価格 50,000円）	» 申込 » 詳細

« 前月　　次月 »

今回のテーマ: バックアップの方法

S2 運用管理（52%）

S2.4 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
  非排他的低レベルバックアップ
  postgresql.confに関する以下の項目
  - Archive Recovery
  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
  pg_start_backup()
  pg_stop_backup()
  backup_label
  tablespace_map
  PITR
  postgresql.conf
  recovery.signal
  COPY
  ¥copy

#OSS-DB

V3.0 と V2.0 の違い

- 「Ver.2.0とVer.3.0の比較（PDF資料：790KB）」より抜粋
  https://oss-db.jp/outline/gold

#OSS-DB

S2.4 バックアップ方法

変更: 下記を「主要な知識範囲」から「重要な用語、コマンド、パラメータなど」に移動

- pg_start_backup()
- pg_stop_backup()

追加

- 非排他的低レベルバックアップ
- postgresql.confに関する以下の項目
  - Archive Recovery
- backup_label
- tablespace_map
- postgresql.conf
- recovery.signal

削除

- recovery.conf

環境情報

- 仮想マシン ossdb-01 を用意
  - CentOS 7.9

```
[postgres@ossdb-01 ~]$ cat /etc/redhat-release
CentOS Linux release 7.9.2009 (Core)
```

  - ossdb-01 の ip アドレスは

```
[postgres@ossdb-01 ~]$ hostname --ip
127.0.0.1 192.168.19.16
```

#OSS-DB

環境情報

PostgreSQL パッケージインストール

#OSS-DB

```
[postgres@ossdb-01 ~]$ yum list installed | grep postgres
postgresql14.x86_64                  14.9-2PGDG.rhel7           @pgdg14
postgresql14-libs.x86_64             14.9-2PGDG.rhel7           @pgdg14
postgresql14-server.x86_64           14.9-2PGDG.rhel7           @pgdg14
```

環境変数設定

```
[postgres@ossdb-01 ~]$ cat .bash_profile
[ -f /etc/profile ] && source /etc/profile
PGDATA=/var/lib/pgsql/14/data
export PGDATA
PGHOME=/usr/pgsql-14
export PATH=$PGHOME/bin:$PATH
export MANPATH=$PGHOME/share/man:$MANPATH
```

事前準備

#OSS-DB

- バックアップ用のデータを作成
  - 0.initdb でデータベースクラスタ初期化
  - ↓
  - 1.サーバを起動し、バックアップ用のデータベースを作成
  - ↓
  - 2.pgbench でバックアップ対象データベースにデータを投入

事前準備 0. initdb でデータベースクラスタ初期化

initdb の実行

#OSS-DB

```
[postgres@ossdb-01 ~]$ initdb --no-locale --encoding=UTF8
データベースシステム内のファイルの所有者はユーザー"postgres"となります。
このユーザーをサーバープロセスの所有者とする必要があります。
……

成功しました。以下のようにしてデータベースサーバーを起動することができます:

    pg_ctl -D /var/lib/pgsql/14/data -l ログファイル start
```

- **--no-locale**

ロケールを利用しない

- **--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

事前準備 1.バックアップ用のデータベースを作成

#OSS-DB

クラスタを起動

```
[postgres@ossdb-01 ~]$ pg_ctl start
サーバーの起動完了を待っています....
サーバー起動完了
```

- **pg_ctl**
  PostgreSQL を起動・停止・リロード・ステータス確認するためのコマンド
- **-D** 操作対象クラスタを指定する (デフォルトの対象が **$PGDATA**)

データベースを作成

```
[postgres@ossdb-01 ~]$ createdb ossdb
[postgres@ossdb-01 ~]$ psql -l
                                  データベース一覧
   名前    | 所有者 | エンコーディング | 照合順序 | Ctype(変換演算子) |     アクセス権限
-----------+----------+------------------+----------+-------------------+-----------------------
 ossdb     | postgres | UTF8             | C        | C                 |
 postgres  | postgres | UTF8             | C        | C                 |
 template0 | postgres | UTF8             | C        | C                 | =c/postgres          +
           |          |                  |          |                   | postgres=CTc/postgres
 template1 | postgres | UTF8             | C        | C                 | =c/postgres          +
           |          |                  |          |                   | postgres=CTc/postgres
(4 行)
```

事前準備 2.pgbench でバックアップ対象データベースにデータを投入

#OSS-DB

pgbench

- **PostgreSQL 本体に同梱されているベンチマークツール**

```
[postgres@ossdb-01 ~]$ pgbench -i -s 10 ossdb
```

- **-i / --initialize**

ベンチマークテーブルの初期化

- **-s NUM / --scale=NUM**

生成される行数の倍率
(デフォルトの倍率1では、pgbench_accountsテーブルに100000行が生成される)

PostgreSQL のバックアップ戦略

#OSS-DB

- **論理バックアップ**
  - pg_dump、pg_dumpall
- **物理バックアップ**
  - **オフライン物理バックアップ**
    クラスタファイルの物理コピー
  - **オンライン物理バックアップ**
    pg_basebackup、低レベルAPIによるバックアップ
- **PITR (Point-In-Time Recovery)**

論理バックアップ(1/2)

#OSS-DB

pg_dump

- **データベース単位のバックアップ**
- **実行できるのは所有者かスーパユーザ**
- **選べるファイル形式**
  - **(デフォルト) テキスト形式** (-Fp)
    SQL文をプレーンテキスト形式で出力
  - **バイナリ形式**
    tar 形式(-Ft)、カスタム形式(-Fc)、ディレクトリ形式(-Fd)

```
[postgres@ossdb-01 ~]$ pg_dump -f ossdb.sql ossdb
[postgres@ossdb-01 ~]$ pg_dump -Fd -j 2 -f ossdb.d ossdb
```

- **リストア方法**
  - **テキスト形式の場合**
    psql コマンドを利用する
  - **バイナリ形式の場合**
    pg_restore コマンドを利用する

```
[postgres@ossdb-01 ~]$ createdb re_ossdb
[postgres@ossdb-01 ~]$ psql -f ossdb.sql re_ossdb
[postgres@ossdb-01 ~]$ createdb re_ossdb_bi
[postgres@ossdb-01 ~]$ pg_restore -d re_ossdb_bi -j 2 ossdb.d
```

論理バックアップ(2/2)

#OSS-DB

pg_dumpall

- 全てのデータベースのバックアップ
- テキスト形式のみ
- 実行ユーザはスーパユーザ

```
[postgres@ossdb-01 ~]$ pg_dump -f db_all.sql
```

リストア

- 既存クラスタがある場合は退避 → 新しいクラスタを作成
  → 設定ファイルのリストア → 起動して psql コマンドでリストア

```
[postgres@ossdb-01 ~]$ pg_ctl stop
[postgres@ossdb-01 ~]$ mv $PGDATA $PGDATA.bck1
[postgres@ossdb-01 ~]$ initdb --no-locale --encoding=UTF8
[postgres@ossdb-01 ~]$ cp $PGDATA.bck1/postgresql.conf $PGDATA/
[postgres@ossdb-01 ~]$ pg_ctl start
[postgres@ossdb-01 ~]$ psql -f db_all.sql postgres
```

物理バックアップ (1/3)

- オフラインバックアップ(コールドバックアップ)
  - サーバを停止して OS コマンドで $PGDATA を物理バックアップ

#OSS-DB

  - PostgreSQL を停止

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

  - OS 付属コマンド(tar,cp,rsync…) で $PGDATA をバックアップ

```
[postgres@ossdb-01 ~]$ cp –ra $PGDATA  /bckups/$PGDATA.bck
```

  - PostgreSQL を起動

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

物理バックアップ (2/3)

pg_basebackup

#OSS-DB

- サーバを停止せずにに物理バックアップを取得する
- レプリケーション権限を持つユーザで実行

```
postgres@ossdb-01 ~]$ pg_ctl status
pg_ctl: サーバーが動作中です(PID: 5788)   ← 稼働中であることを確認
/usr/pgsql-14/bin/postgres
[postgres@ossdb-01 ~]$ pg_basebackup -D $PGDATA.bck2 -P
513268/513268 kB (100%), 1/1 テーブル空間
```

- **-D**　　バックアップ先のディレクトリ
- **-P**　　進捗状況を表示
- **-h,-p**　接続先ホスト、ポートを指定

物理バックアップ (3/3)

#OSS-DB

- **低レベル API を使ったバックアップ**
  - **pg_start_backup / pg_stop_backup 関数を利用**
  - **バックアップモード中は $PGDATA の物理コピーが可能**

  - **排他的と非排他的の違い**

	排他的バックアップ	非排他的バックアップ
バックアップが同時実行可能か	×	○
pg_stop_backup() の実行セッション	pg_start_backup() と別セッションでも OK	pg_start_backup()と同じセッションでないといけない
backup_label と tablespace_map	pg_start_backup() 実行後、$PGDATA 配下に作成され、pg_stop_backup() 実行後に削除される	pg_stop_backup() 実行後に返された結果を基にユーザ自身で作成する

物理バックアップ (3/3)

#OSS-DB

排他的バックアップ

- **排他モードでバックアップモードを開始**
  - pg_start_backup() の第三引数が true (デフォルト)
  - 第一引数はバックアップの識別名

```
ossdb=# SELECT pg_start_backup('exclusive');
 pg_start_backup
-----------------
 0/21000028
```

- **$PGDATA の物理コピー**

- **バックアップモードの終了**

```
ossdb=# SELECT pg_stop_backup();
pg_stop_backup
---------------
 0/21000138
```

物理バックアップ (3/3)

非排他的バックアップ

#OSS-DB

- **非排他モードでバックアップモードを開始**
  - pg_start_backup() の第三引数に false を指定

```
ossdb=# SELECT pg_start_backup('non-exclusive', false, false);
```

- **$PGDATA の物理コピー**
- **バックアップモードの終了**
  - pg_stop_backup() の第一引数に false を指定

backup_label

tablespace_map
(テーブルスペースを
利用している時にだけ必要)

```
ossdb=# SELECT * FROM pg_stop_backup(false);
NOTICE:  WAL archiving is not enabled; you must ensure that all required WAL segments are copied through other means to complete the backup
    lsn     |                                                                                                  labelfile                                                                                                  | spcmapfile
------------+-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+------------
 0/1D000138 | START WAL LOCATION: 0/1D000028 (file 000000010000000000000001D) +|
            | CHECKPOINT LOCATION: 0/1D000060                                 +|
            | BACKUP METHOD: streamed                                         +|
            | BACKUP FROM: primary                                            +|
            | START TIME: 2023-10-03 09:02:30 JST                             +|
            | LABEL: non-exclusive                                            +|
            | START TIMELINE: 1                                               +|
            |                                                                  |
```

PITR(1/10)

#OSS-DB

- Point In Time Recovery
  - ベースバックアップとアーカイブログを両方使う
  - データベースを任意の時点までリストアできる

- ベースバックアップ
  - データベースクラスタの物理コピー

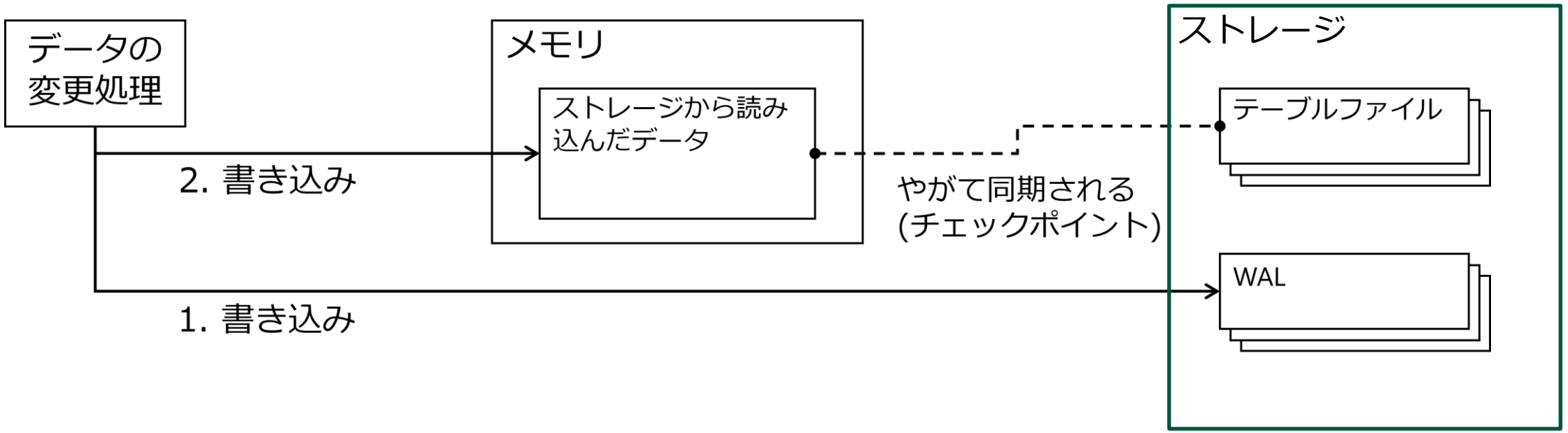
- アーカイブログ
  - WAL の物理コピー

PITR(2/10)

PostgreSQL のデータの書き込み

#OSS-DB

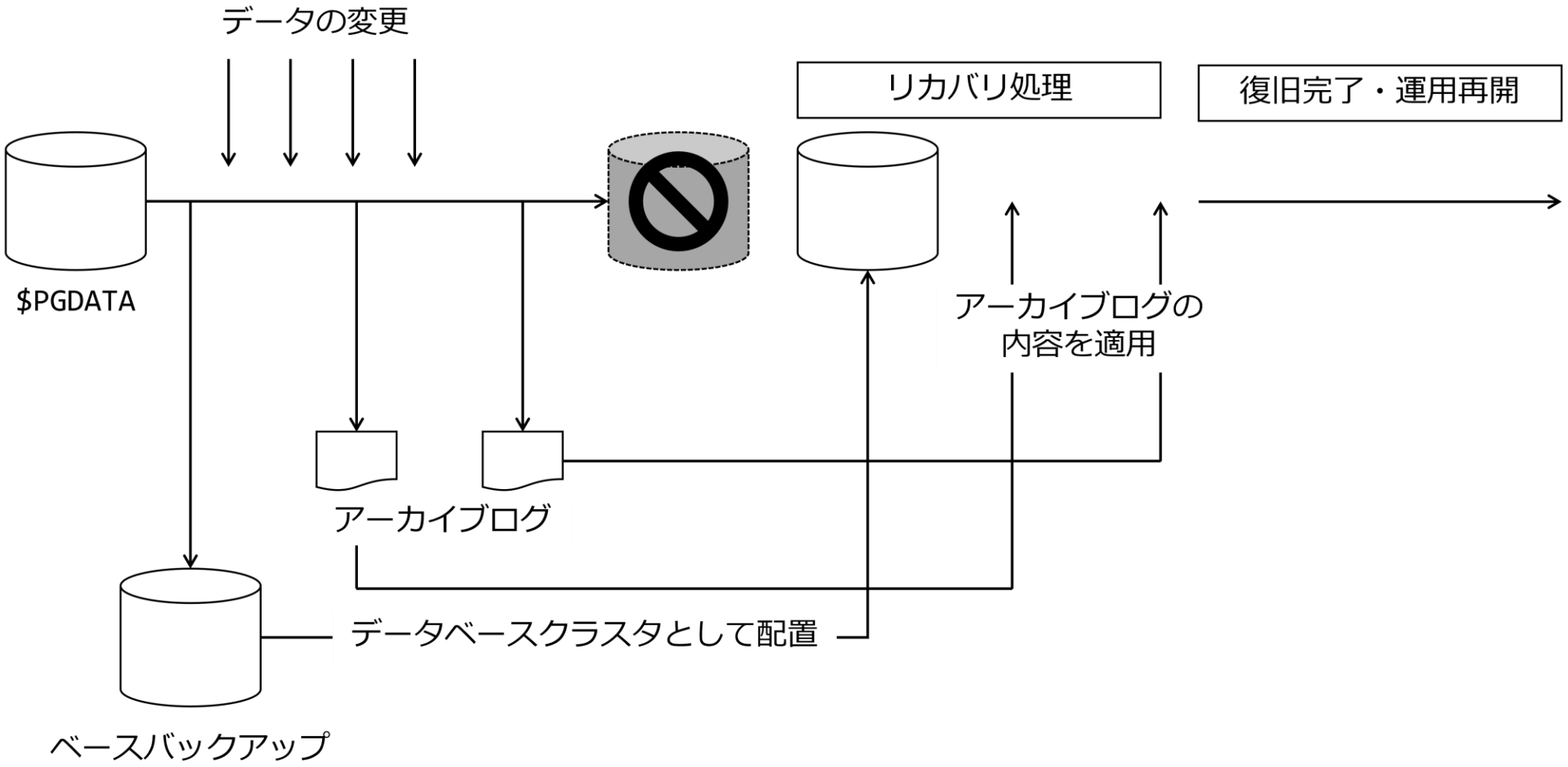
- メモリを介する（速度）
- データの変更内容をWALとメモリ両方に書き込む (保全性)
- ディスクに書き込むタイミング=チェックポイント
  - 不要になった WAL は削除される



PITR(3/10)

PITR の流れ

#OSS-DB



PITR(4/10)

事前準備

#OSS-DB

- ベースバックアップとアーカイブ WAL の格納領域を用意

```
[postgres@ossdb-01 ~]$ mkdir /backups/base
[postgres@ossdb-01 ~]$ mkdir /backups/arc
```

- WAL アーカイブの設定
  - postgresql.conf の編集

```
[postgres@ossdb-01 ~]$ vi $PGDATA/postgresql.conf
# - Archiving -

archive_mode = on               # enables archiving; off, on, or always
                                # (change requires restart)
archive_command = 'cp "%p" "/backups/arc/%f"'# command to use to archive a logfile segment
```

  - postgresql.conf の反映

アーカイブ対象WALのパス　　アーカイブWALファイル名

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

PITR(5/10)

#OSS-DB

1. ベースバックアップの取得

```
[postgres@ossdb-01 ~]$ pg_basebackup -D /backups/base/001 -Xn -P
NOTICE:  WAL archiving is not enabled; you must ensure that all required WAL segments are copied through other means to complete the backup
516884/516884 kB (100%), 1/1 テーブル空間
```

- **-D**　バックアップ先ディレクトリを指定
- **-P**　進行状態を表示
- **-Xn**　WALファイルを含めない (アーカイブするように設定済みのため)

PITR(6/10)

更新データを生成

#OSS-DB

- ossdb データベースにデータ書き込み

```
[postgres@ossdb-01 ~]$ pgbench -c 30 -t 50 ossdb
```

  - **-c** NUM　クライアントの同時接続数
  - **-t** NUM　トランザクション数

- 履歴テーブルから最新のデータ行を確認しておく

```
ossdb=# SELECT * FROM pgbench_history ORDER BY mtime LIMIT 1;
 tid | bid |  aid   | delta |           mtime            | filler
-----+-----+--------+-------+----------------------------+--------
  43 |   7 | 512580 | -2986 | 2023-10-03 11:54:00.258502 |
(1 行)
```

PITR(7/10)

#OSS-DB

障害発生

- データベースクラスタが壊れたことを想定

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

  - -m MODE 停止モードを指定する
    - i (immediate) 全ての接続を即座に切断して強制終了
    - f (fast) 接続中のトランザクションをロールバックしてからサーバを停止(デフォルト)
    - s (smart) 新しい接続は禁止で、接続中のクライアントの切断を待ってからサーバ停止

- 壊れたクラスタを別名に変更して退避

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

PITR(8/10)

- リカバリ

#OSS-DB

  - ベースバックアップファイルの展開

```
[postgres@ossdb-01 ~]$ cp -r /backups/base/001 $PGDATA
```

  - postgresql.conf の編集

```
[postgres@ossdb-01 ~] $ vi $PGDATA/postgresql.conf
# - Archive Recovery -
# These are only used in recovery mode.

restore_command = 'cp "/backups/arc/%f" "%p"'          # command to use to restore an archived
logfile segment
```

  - recovery.signal の作成

```
[postgres@ossdb-01 ~] $ touch $PGDATA/recovery.signal
```

※ 12 より前は、recovery.conf にリカバリ用のパラメータを記載していた

PITR(9/10)

#OSS-DB

- リストア
- 未アーカイブの WAL をコピー

  - 未アーカイブの WAL の存在を確認

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

  - リカバリ先データベースクラスタ内のWAL領域へ手動コピー

```
[postgres@ossdb-01 ~]$ cp $PGDATA.crash/pg_wal $PGDATA/pg_wal/
```

PITR(10/10)

- **リストア**
- **リカバリ実行**

#OSS-DB

```
[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所要時間が長くなる

- **リカバリ完了**
  - recovery.signal が削除される
  - ログに「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

```
ossdb=# SELECT * FROM pgbench_history ORDER BY mtime LIMIT 1;
 tid | bid |  aid   | delta |           mtime            | filler
-----+-----+--------+-------+----------------------------+--------
  43 |   7 | 512580 | -2986 | 2023-10-03 11:54:00.258502 |
(1 行)
```

まとめ

バックアップ方法まとめ

#OSS-DB

戦略	バックアップ対象	方法	データの復旧地点
論理バックアップ	PostgreSQL のデータ	pg_dump, pg_dumpall	バックアップの取得時点
物理バックアップ (オフライン)	データベースクラスタ内のすべてのディレクトリ・ファイル	OS のコマンド (tar,cp, …)	
物理バックアップ (オンライン)		pg_basebackup, pg_start_backup() と pg_stop_backup()	
PITR	上記+WALアーカイブログ	上記 + archive_mode と archive_command の設定	ベースバックアップ取得時点からアーカイブが適用できる範囲の任意の地点