PostgreSQL インサイド

バイナリデータを pg_dump でバックアップしたときのエラーを解決したい

技術を知る

- [ ] 導入／環境設定
- [ ] 移行
- [ ] 性能
- [ ] チューニング
- [x] バックアップ／リカバリー
- [ ] 冗長化／負荷分散
- [ ] 監視
- [ ] データ連携
- [ ] 災害対策
- [x] 豆知識

実現方法

PostgreSQL では、バイナリデータを「バイナリ列データ型」としてテーブル内で扱う場合、型名は「bytea」を使用します。 バイナリデータを含むテーブルを pg_dump コマンドで論理バックアップしたところ、以下のようなエラーメッセージが出力されることがあります。

```
pg_dump: error: Dumping the contents of table "xxxxx" failed: PQgetResult() failed.
pg_dump: error: Error message from server: ERROR:  invalid memory alloc request size xxxxxxxxx
```

日本語環境では以下のようなメッセージが出力されることがあります。

```
pg_dump: error: テーブル"xxxxx"の内容のダンプに失敗: PQgetResult()が失敗しました。
pg_dump: error: サーバのエラーメッセージ: ERROR:  invalid memory alloc request size xxxxxxxxx
```

pg_dump コマンドでは、バイナリ形式のデータを文字として出力します。そのため、バイナリ列データ型のデータが文字に変換され、データ量が増大し、文字として扱えるデータの最大長（約 1 ギガバイト）を超えたことが原因です。

このような場合、SQL コマンドの COPY を使ってバイナリ形式で論理バックアップを実施してください。

実行例

ここでは、約 537 メガバイトのバイナリデータを含むテーブルをバックアップする例で説明します。データベース名は mydb とします。なお、この記事は、PostgreSQL 15.0 をベースに作成しています。initdb 時の符号化方式（エンコーディング）は「UTF8」を指定しています。

前提

1. テーブル：goods1 を作成します。列は 3 つ作成し、バイナリデータを格納する列：data のデータ型は「bytea」と定義します。

```
mydb=# CREATE TABLE goods1 (id serial not null primary key, name text, data bytea not null);
CREATE TABLE
```

2. 動画ファイルのバイナリデータ：sample.mp4（約 537 メガバイト）を pg_read_binary_file 関数で bytea 型に変換し、テーブルに格納します。

```
mydb=# INSERT INTO goods1(name,data) VALUES ('DEMO1',pg_read_binary_file('/home/data/sample.mp4'));
INSERT 0 1
```

3. pg_dump コマンドでテーブル：goods1 をバックアップすると、エラーが発生しました。

```
$ pg_dump -t goods1 mydb -c > goods1_dump.data
pg_dump: error: Dumping the contents of table "goods1" failed: PQgetResult() failed.
pg_dump: error: Error message from server: ERROR:  invalid memory alloc request size 1126352709
pg_dump: error: The command was: COPY public.goods1 (id, name, data) TO stdout;
```

エラーメッセージから、約 537 メガバイトのバイナリデータが約 2 倍となる「1126352709 バイト（約 1074 メガバイト＝約 1.05 ギガバイト)」で処理されたことがわかります。

対処例

そこで、SQL コマンドの COPY を使って、バイナリデータをバイナリ形式のままで論理バックアップを実施します。

1. COPY でテーブル：goods1 のデータをバックアップします。データ形式の FORMAT パラメーターには「binary」（デフォルトは text）を設定します。

```
mydb=# COPY public.goods1  TO  '/home/data/goods1_copy.data' (FORMAT binary);
COPY 1      <-- 1 件のコピー（バックアップ）が正常終了
```

2. バックアップデータ：goods1_copy.data が生成されたかを確認します。

```
$ ls -l
-rw-r--r--. 1 fsepuser fsepuser 563176397 5 月 22 11:16 goods1_copy.data
```

なお、論理バックアップの方法を変更（pg_dump コマンドから SQL コマンドの COPY に変更）することにより、取得したバックアップデータを使ったリカバリー方法も変更する必要があります。詳細については「論理バックアップの方法」を参照してください。

ポイント

バイナリデータを論理バックアップする上で、参考になる情報を示します。

バイナリデータの論理バックアップ

バイナリデータは、文字列に変換せずにそのままの形式で論理バックアップを実施することをお薦めします。1 バイトのバイナリデータは文字コードに変換されると、コード系によって 1~4 バイトに変動しますので、注意が必要です。
参考までに、約 300 メガバイトの動画ファイルのバイナリデータ：FEP_demo2.mp4 を pg_dump コマンドと COPY のそれぞれでバックアップしたデータ量を以下に示します。
pg_dump コマンドでバックアップしたデータが COPY でバックアップしたデータに比べて、約 2 倍のデータ量になっていることがわかります。

```
$ ls -l
-rw-r--r--. 1 fsepuser fsepuser  307359797 5 月 22 11:44 goods2_copy.data   <-- COPY でバックアップ
-rw-rw-r--. 1 fsepuser fsepuser  614721497 5 月 22 11:45 goods2_dump.data   <-- pg_dump でバックアップ（注）
```

- 注)　お薦めのバックアップ方法ではありませんが、処理中に 1 ギガバイトを超えなかったため、エラーになりませんでした。

論理バックアップの方法

論理バックアップは、PostgreSQL を起動した状態でデータベースに格納されているデータをファイルに書き出すバックアップ方法です。論理バックアップは、バックアップを実行した時点へのリカバリーのみが可能です。論理バックアップを実施する方法として以下の 3 つがあります。

pg_dumpall コマンド

データベースクラスタ全体の内容を SQL の形でバックアップします。取得したスクリプト形式(SQL コマンドが書き込まれた平文ファイル)のバックアップデータは、psql コマンドを使ってバックアップを取得した時点へリカバリーします。

pg_dump コマンド

指定したデータベースの内容を SQL の形でバックアップします。取得したスクリプト形式のバックアップデータは、psql コマンドを使ってバックアップを取得した時点へリカバリーします。取得したアーカイブファイル形式のバックアップデータは、pg_restore コマンドを使ってバックアップを取得した時点へリカバリーします。

SQL コマンドの COPY

指定したテーブルのデータを独自のテキスト形式、バイナリ形式、または CSV 形式でバックアップします。取得したバックアップデータは、COPY を使ってバックアップを取得した時点へリカバリーします。

論理バックアップの詳細およびリストアについては、以下の記事を参照してください。

- PostgreSQL のバックアップとリカバリー

参考

PostgreSQL 15 文書

- Documentation（PostgreSQL オフィシャル）
  https://www.postgresql.org/docs/
  - Chapter 8. Data Types
    - 8.4. Binary Data Types
  - SQL Commands
    - COPY
- PostgreSQL 日本語ドキュメント（日本 PostgreSQL ユーザ会）
  https://www.postgresql.jp/document/
  - 第 8 章 データ型
    - 8.4. バイナリ列データ型
  - SQL コマンド
    - COPY

2023 年 6 月 9 日