PostgreSQL 恢复利器

PDU(PostgreSQL Data Unloader)

第一版

	产品和技术咨询	商务咨询
微信/WeChat	ZhangChen-PDU	xifenfei88
邮箱/E-mail	1109315180@qq.com	dba@xifenfei.com
电话/Tel	+86-15251853831	+86-17813235971

2025.4

第一章 PDU 软件介绍

什么是 PDU

PDU 全称为 PostgreSQL DataUnloader，是用于 PostgreSQL 数据库紧急恢复的软件，在各种原因造成的数据库不能打开或数据删除后没有备份时，使用 PDU 抢救数据，最大限度地减少数据丢失。

现实中总会有很多的意外，数据被意外删除、硬件问题、软件 BUG 等原因导致数据库损坏，在没有备份的情况下，PDU 能够通过直接访问 PostgreSQL 数据库数据文件恢复出所有完好的数据，以避免所有数据丢失造成的损失。即使在有备份的情况下，也可能由于只有逻辑备份，备份的数据与当前库中数据存在时间差，因此也必须恢复库中的数据，此时 PDU 就可以派上用场。

PDU 的功能特点

1. 无需在 PostgreSQL 数据库启动的情况下，PDU 直接读取数据库文件进行数据解析。在由于硬件问题或人为误操作引起数据损坏时，PostgreSQL 数据库软件为遵行数据完整性和一致性原则，往往不能打开数据库对数据库进行访问，而 PDU 可以绕过 PostgreSQL 数据库软件，直接从 PostgreSQL 数据库文件解析数据，从而最大恢复数据，减少因为数据丢失而引起的损失。

2. 多版本 PostgreSQL 支持，支持的 PostgreSQL 数据库版本包括 10,11,12,13,14,15,16,17.

3. 支持类似于 psql 的命令，比如:\l、\dt、\dt+、\dn

4. 支持各种类型的表，包括普通的 HEAP 表，聚簇（CLUSTER）表，各种类型分区表

5. 支持单个表 oid 文件恢复

6. 支持单个数据库目录恢复

7. 恢复数据类型可以选择.sql(insert 方式)和.csv(copy 方式)

8. 支持数据类型：smallserial、smallint、int、tinyint、oid、xid、serial、bigint、bigserial、float4、float8、numeric、time、timetz、date、timestamp、timestamptz、real、bool、uuid、macaddr、name、char、charn、varchar、bpchar、text、json、xml、clob、blob、bytea、jsonb、_int2、_int4、_int8、_text、_varchar、_float4、_float8、_timestamp

9. 支持数据库字符集 utf8 和 gbk

10. 支持 oid 文件有坏块的情况下对好的 block 中数据进行抢救性恢复

11. 支持多平台（windows 和 linux）

12. 支持 delete 表恢复

13. 支持 openGauss（open 高斯）、KingbaseES（电科金仓）、Vastbase（海量数据库）

PDU 其他特点

除了上述功能上的特点之外，PDU 还具有如下特点：

数据恢复速度快、占用内存少

PDU 由在所有主流开发语言中速度最快、内存使用最有效率的 C 语言所写，恢复速度快。同时 PDU 的一些功能能够并行执行，以提高恢复速度；PDU 支持命令文件，通过使用不同的命令文件多并发执行 PDU 提高数据恢复速度。这样在进行数据恢复时，能够大大减少恢复所需要的时间，从而减少因为数据丢失和损坏引起的业务停止时间。

使用简单

使用 PDU 恢复数据非常简单，只需要简单的配置加上两三条命令即可恢复数据。这样在数据损坏或丢失时，不需要经过专门的培训和长时间的学习，就能使用 PDU 进行恢复。

运行稳定

PDU 能够自动检测并跳过数据库中的坏块，避免了坏块解析时引起的异常，保持数据恢复的稳定不出错。

目前不支持的功能

PDU 目前暂时不支持下列特性或数据类型的恢复：

1. 使用 PostgreSQL TDE 加密的数据。
2. 用户自定义的数据类型。
3. PostGIS 插件数据类型。

由于这几类数据用得极少，基本不影响数据的恢复，如果有特殊需求，可以进行定制化开发，同时功能也在进一步完善中。

第二章 安装 PDU 和使用 PDU

下载 PDU 软件

访问 https://www.pgdul.com 下载最新的 PDU 软件，版本号中带有 Free 表示社区版。
正式版需要授权之后才能使用
社区版可以直接使用，但是每个表最多恢复记录数不超过 8888 条，delete 记录恢复最多不超过 888 条，用于验证软件恢复功能。

创建目录和上传 PDU 软件

上传 pdu 软件到空间比较大的目录（恢复文件恢复存储在该目录下）

解压 PDU 软件

使用 unzip 命令进行解压
unzip pdu_版本_数据库类别.zip 或者 pdu_版本_数据库类别_Free.zip

PDU 软件授权

在软件没有授权的情况下，正式版软件无法启动成功

```
[root@localhost pdu]# ./pdu12
在PDU软件同级目录下没有发现license.pdu授权文件
请将下列字符提供给PDU开发者，用于生成license文件：

n9VFo6eKmgqV1xup9dOSC5qZTaqmhMkHmdNSqqXRnx6b1h716YnOCsrIRqei1p8FmtBMq/SF

License校验失败
```

需要把上述字符（机器码）提供给 PDU 开发者，生成授权文件，然后才能正常进入软件，授权文件为 license.pdu

使用 PDU 软件

启动 pdu 软件

```
./pdu12(对应 pg 数据库版本号)
[root@localhost pdu]# ./pdu12

╔═══════════════════════════════════════════════════════════════╗
║  Copyright 2024-2025 ZhangChen. All rights reserved           ║
║  PDU: PostgreSQL Data Unloader                                ║
║  Version 2.0.0 (2025-04-24)                                   ║
╚═══════════════════════════════════════════════════════════════╝
```

```
Current DB Supported Version:
──────────────────────────────────────────────

• PostgreSQL 12

╔═══════════════════════════════════════════════════════════════════════════════════════╗
║            PROFESSIONAL EDITION          ║                                            ║
╠═══════════════════════════════════════════════════════════════════════════════════════╣
║ • Licensed to:                           ║                                            ║
║ • Full functionality                     ║                                            ║
║ • No limitations                         ║                                            ║
╚═══════════════════════════════════════════════════════════════════════════════════════╝


Contact Me:
──────────────────────────────────

• WeChat Public: ZhangChen-PDU
• Email: 1109315180@qq.com
• Tel: 15251853831 or 17813235971

[!] 重要提示:
──────────────────────

使用前请设置 ulimit -n 为足够大的值
否则可能导致数据导出失败
```

退出 pdu

exit;或者\q;

```
pg_db.public-# exit;
[root@localhost pdu]#
****或者****
PDU.public=# \q;
[root@localhost pdu]#
```

帮助命令

只要输入的非 pdu 支持的命令,即会提示帮助命令,以下是 pdu 支持的所有命令集

```
PDU 数据拯救工具 | 命令帮助
*************************************************************************************************
*************
**基础操作**
b;                                      | 初始化数据库元信息
<exit;>|<\q;>                           | 退出工具
                                 -------.--------

**数据库切换**
use <db>;                               | 指定目标数据库 (例: use logs;)
```

```
set <schema>;                          | 指定操作模式 (例: set recovery;)
                              -------.--------

**元数据展示**
\l;                                    | 列出所有数据库
\dn;                                   | 显示当前数据库模式
\dt;                                   | 列出当前模式下的表
\d+ <table>;                           | 查看表结构详情 (例: \d+ users;)
\d <table>;                            | 查看表列类型 (例: \d users;)
                              -------.--------

**数据导出**
unload tab <table>;                    | 导出表数据 → <表名>.csv (例: unload tab orders;)
unload sch <schema>;                   | 导出指定模式的所有数据 (例: unload sch public;)
unload ddl;                            | 生成当前模式下的 DDL 语句文件
unload copy;                           | 生成已导出的 CSV 文件的 COPY 语句文件
                              -------.--------

**误删数据恢复**
scan t1;                               | 扫描被误删的表
restore del <Tx Number>;               | 通过 事务号 恢复被误删的数据
restore del all;                       | 通过 时间区间 恢复被误删的数据
                              -------.--------

**自定义数据文件/数据目录恢复**
add <filenode> <tablename> <attibutes>; | 将特定表信息手动添加到 restore 库中
例              如              :              <add              12345              t1
varchar,varchar,timestamp,varchar,numeric,varchar,varchar,varchar,numeric;>
scan manual;                           | 从 manual 路径下的 ddl 脚本中初始化元数据
restore db <dbname> <DB Path>;           | 初始化特定的数据库目录 (例: restore db xmandb
/home/postgres/data/base/290113;)
                              -------.--------

**参数设置**
param startwal 000000010000000800000008; | 设置 scan 扫描的起始 WAL 文件 ,如果未设置则默认是归
档目录的第一个文件
param endwal 000000010000000800000009;   | 设置 scan 扫描的结束 WAL 文件 ,如果未设置则默认是归
档目录的最后一个文件
param resmode tx|time;                    | 设置 restore 恢复的模式 ,选择按照 事务号/TX 或 时
间区间/TIME 进行恢复
param starttime|endtime <TIME>;             | 设置 scan 扫描的起始时间或结束时间(例: param
starttime|endtime 2025-01-01 00:00:00)
param exmode csv|sql;                  | 设置导出模式为 CSV 文件或 SQL 文件, 默认 CSV 文件
param encoding utf8|gbk;               | 设置导出数据的字符集为 utf 或 gbk, 默认 utf8
```

```
reset <param name>                          | 重置某个参数
show;                                       | 查看当前所有参数
*****************************************************************************************************
*************

语法规则
◈ 指令后缀必须带 `;`
```

初始化字典

b;

```
PDU.public=# b;
开始初始化...
 -pg_database:</data/pg/12/data/global/1262>

数据库:postgres
      -pg_schema:</data/pg/12/data/base/13458/2615>
      -pg_class:</data/pg/12/data/base/13458/1259> 共 86 行
      -pg_attribute:</data/pg/12/data/base/13458/1249> 共 2913 行
      模式:
        -->public 0 张表

数据库:pg_db
      -pg_schema:</data/pg/12/data/base/16384/2615>
      -pg_class:</data/pg/12/data/base/16384/1259> 共 239 行
      -pg_attribute:</data/pg/12/data/base/16384/1249> 共 5471 行
      模式:
        -->public 87 张表
```

显示所有数据库

\l;

```
PDU.public-# \l;
|------------------|
|     数据库名     |
|------------------|
|    postgres      |
|    pg_db         |
|------------------|
```

进入某个数据库

use <database>;

```
PDU.public=# use pg_db;
|--------------------------------------|
|          模式            |  表数量   |
|--------------------------------------|
```

```
|    public                |  87        |
|------------------------------------------|
```

显示某个数据库下面的 schema 列表

\dn;

```
pg_db.public=# \dn;
|------------------------------------------|
|          模式            |  表数量     |
|------------------------------------------|
|    public                |  87        |
|------------------------------------------|


     1 rows selected
```

进入某个 schema

set <schema>;

```
pg_db.public=# set public;
|-----------------------------------------------|
|              表名                |  表大小     |
|-----------------------------------------------|
|    event                         |  516.51 MB |
|    device                        |  150.59 MB |
|    ts_kv_latest                  |  124.70 MB |
|    device_credentials            |  122.71 MB |
|    attribute_kv                  |  94.54 MB  |
|    ts_kv_2023_03                 |  63.39 MB  |
|    device_batch_relation         |  43.54 MB  |
|    audit_log                     |  42.32 MB  |
|    scada                         |  5.15 MB   |
|    ts_kv_2023_02                 |  3.91 MB   |
|    widget_type                   |  1.74 MB   |
|    device_group_device           |  1.62 MB   |
|    ts_kv_2023_01                 |  1.37 MB   |
|    car_gps                       |  928.00 KB |
|    rule_node                     |  672.00 KB |
|    device_shadow                 |  512.00 KB |
|    mechanism_model               |  504.00 KB |
|    relation                      |  440.00 KB |
|    ts_kv_2023_04                 |  312.00 KB |
|    thing_model_release           |  280.00 KB |
|-----------------------------------------------|


        仅显示表大小排名前 20 的表名
```

显示某个表创建语句

\d+ <table>;

```
pg_db.public=# \d+ device;
-------------------------------------------------------------------
|                            建表语句                              |
-------------------------------------------------------------------
   CREATE TABLE device (
        id                      uuid,
        created_time            int8,
        additional_info         varchar,
        customer_id             uuid,
        device_profile_id       uuid,
        device_data             jsonb,
        type                    varchar(255),
        name                    varchar(255),
        label                   varchar(255),
        search_text             varchar(255),
        tenant_id               uuid,
        firmware_id             uuid,
        software_id             uuid,
        external_id             uuid,
        device_no               varchar,
        sn_code                 varchar,
        node_type               varchar(40),
        auth_type               varchar,
        product_key             varchar(255),
        device_secret           varchar(255)
   );
-------------------------------------------------------------------
|                                                                 |
-------------------------------------------------------------------
```

显示某个 schema 下面所有表名

\dt;

```
pg_db.public-# \dt;
|-------------------------------------------------|
|              表名                 |  表大小     |
|-------------------------------------------------|
|    event                          |  516.51 MB |
|    device                         |  150.59 MB |
|    ts_kv_latest                   |  124.70 MB |
|    device_credentials             |  122.71 MB |
|    attribute_kv                   |  94.54 MB  |
|    ts_kv_2023_03                  |  63.39 MB  |
```

```
|    device_batch_relation            |  43.54 MB  |
|    audit_log                        |  42.32 MB  |
|    scada                            |  5.15 MB   |
|    ts_kv_2023_02                    |  3.91 MB   |
|    widget_type                      |  1.74 MB   |
|    device_group_device              |  1.62 MB   |
******篇幅原因，省略部分表（实际软件不会省略）******
|    scada_model                      |  24.00 KB  |
|    ts_kv_dictionary                 |  16.00 KB  |
|    tenant_profile_relation          |  16.00 KB  |
|    device_bind_device               |  16.00 KB  |
|    tb_schema_settings               |  8.00 KB   |
|    admin_settings                   |  8.00 KB   |
|    entity_alarm                     |  8.00 KB   |
|    customer                         |  8.00 KB   |
******篇幅原因，省略部分表（实际软件不会省略）******
|    tb_user                          |  0         |
|    map_data                         |  0         |
|-------------------------------------------------|

       共计 87 张表
```

恢复单个表数据

unload tab <table>;

```
pg_db.public=# unload tab device;
正在解析表 <device>. 已解析数据页: 19275, 已解析数据: 434827 条
表名<device>-</data/pg/12/data/base/16384/19358> 解析完成, 19275 个数据页 ,共计 434827 条数据. 成功 434827 条; 失败 0 条  |  COPY 文件路径为:<pg_db/public/device.csv>
```

恢复整个 schema 数据

unload sch <schema>;

```
pg_db.public=# unload sch public;
正在解析表 <tb_schema_settings>. 已解析数据页: 1, 已解析数据: 1 条
表名<tb_schema_settings>-</data/pg/12/data/base/16384/19178> 解析完成, 1 个数据页 ,共计 1 条数据. 成功 1 条; 失败 0 条  |  COPY 文件路径为:<pg_db/public/tb_schema_settings.csv>

正在解析表 <admin_settings>. 已解析数据页: 1, 已解析数据: 6 条
表名<admin_settings>-</data/pg/12/data/base/16384/19183> 解析完成, 1 个数据页 ,共计 6 条数据. 成功 6 条; 失败 0 条  |  COPY 文件路径为:<pg_db/public/admin_settings.csv>

正在解析表 <alarm>. 已解析数据页: 3, 已解析数据: 51 条
表名<alarm>-</data/pg/12/data/base/16384/19191> 解析完成, 3 个数据页 ,共计 51 条数据. 成功 51 条; 失败 0 条  |  COPY 文件路径为:<pg_db/public/alarm.csv>
```

正在解析表 <entity_alarm>. 已解析数据页: 1, 已解析数据: 26 条
表名<entity_alarm>-</data/pg/12/data/base/16384/19199> 解析完成, 1 个数据页 ,共计 26 条数据. 成功 26 条; 失败 0 条 | COPY 文件路径为:<pg_db/public/entity_alarm.csv>

正在解析表 <audit_log>. 已解析数据页: 3192, 已解析数据: 18757 条
表名<audit_log>-</data/pg/12/data/base/16384/19221> 解析完成, 3192 个数据页 ,共计 18757 条数据. 成功 18757 条; 失败 0 条 | COPY 文件路径为:<pg_db/public/audit_log.csv>

正在解析表 <attribute_kv>. 已解析数据页: 12101, 已解析数据: 901378 条
表名<attribute_kv>-</data/pg/12/data/base/16384/19229> 解析完成, 12101 个数据页 ,共计 901378 条数据. 成功 901378 条; 失败 0 条 | COPY 文件路径为:<pg_db/public/attribute_kv.csv>

正在解析表 <component_descriptor>. 已解析数据页: 16, 已解析数据: 60 条
表名<component_descriptor>-</data/pg/12/data/base/16384/19237> 解析完成, 16 个数据页 ,共计 60 条数据. 成功 60 条; 失败 0 条 | COPY 文件路径为:<pg_db/public/component_descriptor.csv>

.....

正在解析表 <oauth2_client_registration_template>. 已解析数据页: 1, 已解析数据: 4 条
表名<oauth2_client_registration_template>-</data/pg/12/data/base/16384/19558> 解析完成, 1 个数据页 ,共计 4 条数据. 成功 4 条; 失败 0 条 | COPY 文件路径为:<pg_db/public/oauth2_client_registration_template.csv>

正在解析表 <oauth2_client_registration_info>. 已解析数据页: 0, 已解析数据: 0 条
表名<oauth2_client_registration_info>-</data/pg/12/data/base/16384/19568> 解析完成, 0 个数据页 ,共计 0 条数据. 成功 0 条; 失败 0 条 | COPY 文件路径为:<pg_db/public/oauth2_client_registration_info.csv>

正在解析表 <oauth2_client_registration>. 已解析数据页: 0, 已解析数据: 0 条
表名<oauth2_client_registration>-</data/pg/12/data/base/16384/19576> 解析完成, 0 个数据页 ,共计 0 条数据. 成功 0 条; 失败 0 条 | COPY 文件路径为:<pg_db/public/oauth2_client_registration.csv>

正在解析表 <api_usage_state>. 已解析数据页: 19, 已解析数据: 103 条
|-块号 103 空页面或页面已损坏 ,已跳过
正在解析表 <api_usage_state>. 已解析数据页: 20, 已解析数据: 103 条
|-块号 104 空页面或页面已损坏 ,已跳过
正在解析表 <api_usage_state>. 已解析数据页: 22, 已解析数据: 104 条
表名<api_usage_state>-</data/pg/12/data/base/16384/19581> 解析完成, 22 个数据页 ,共计 104 条数据. 成功 104 条; 失败 0 条 | COPY 文件路径为:<pg_db/public/api_usage_state.csv>

.....

正在解析表 <thing_model_release>. 已解析数据页: 31, 已解析数据: 199 条

```
表名<thing_model_release>-</data/pg/12/data/base/16384/30088> 解析完成, 31 个数据页 ,共计 199 条数据. 成功 199 条; 失败 0 条 | COPY 文件路径为:<pg_db/public/thing_model_release.csv>

正在解析表 <thing_model_function_block>. 已解析数据页: 1, 已解析数据: 43 条
表名<thing_model_function_block>-</data/pg/12/data/base/16384/30101> 解析完成, 1 个数据页 ,共计 43 条数据. 成功 43 条; 失败 0 条 | COPY 文件路径为:<pg_db/public/thing_model_function_block.csv>

正在解析表 <car_gps>. 已解析数据页: 116, 已解析数据: 11165 条
表名<car_gps>-</data/pg/12/data/base/16384/31505> 解析完成, 116 个数据页 ,共计 11165 条数据. 成功 11165 条; 失败 0 条 | COPY 文件路径为:<pg_db/public/car_gps.csv>



模式<public>共 87 张表。成功：69, 无数据：18, 失败 0
日志路径:log/pg_db_unload_schema_public_err.txt + log/pg_db_unload_schema_public_succ.txt

COPY 命令导出完成, 文件路径: pg_db/COPY/public_copy.sql ,共找到 69 个 csv 文件

DDL 导出完成. 文件路径: pg_db/DDL/public_ddl.sql, 共计 87 张表
```

恢复结果导入数据库

创建新库

```
postgres=# create database pg_restore;
CREATE DATABASE
```

创建库/表

```
postgres=# \c pg_restore;
You are now connected to database "pg_restore" as user "postgres".
pg_restore=# \i /data/tools/pdu/pg_db/DDL/public_ddl.sql
psql:/data/tools/pdu/pg_db/DDL/public_ddl.sql:1: ERROR:  schema "public" already exists
SET
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
.....
CREATE TABLE
CREATE TABLE
pg_restore=#
```

1. 导入数据

```
pg_restore=# \i /data/tools/pdu/pg_db/COPY/public_copy.sql
SET
COPY 1
COPY 6
COPY 51
COPY 26
COPY 1116074
psql:/data/tools/pdu/pg_db/COPY/public_copy.sql:17:  ERROR:  missing  data  for  column
"category"
CONTEXT:  COPY device_profile, line 1: "4873da20-cdff-11ed-a7df-37683e842ddf    1680073183426
车联网  DEFAULT \N      DEFAULT DISABLED        {"alarms": ..."
COPY 434827
.....
COPY 43
COPY 11165
pg_restore=#
```

2. 补充说明

如果使用 copy 方式导入数据，由于长度或者分隔符等原因导致个别表/个别行报错,可以使用 sql 方式导出,然后对异常对象进行单独导入数据

报错信息

```
psql:/data/tools/pdu/pg_db/COPY/public_copy.sql:17:  ERROR:  missing  data  for  column
"category"
CONTEXT:  COPY device_profile, line 1: "4873da20-cdff-11ed-a7df-37683e842ddf    1680073183426
车联网  DEFAULT \N      DEFAULT DISABLED        {"alarms": ..."
```

sql 方式恢复

```
PDU.public=# param exmode sql;
-----------------------------------------------------------------
|              参数                |              当前值          |
-----------------------------------------------------------------
|    startwal                      |                              |
|    endwal                        |                              |
|    starttime                     |                              |
|    endtime                       |                              |
|    resmode(Data Restore Mode)    |              TX              |
|    exmode(Data Export Mode)      |              SQL             |
|    encoding                      |              UTF8            |
-----------------------------------------------------------------

PDU.public=# use pg_db;
|---------------------------------------|
|          模式             |  表数量    |
|---------------------------------------|
```

```
|    public                 |  87        |
|-------------------------------------|
pg_db.public=# show;
-----------------------------------------------------------------
|              参数                |              当前值           |
-----------------------------------------------------------------
|    startwal                      |                              |
|    endwal                        |                              |
|    starttime                     |                              |
|    endtime                       |                              |
|    resmode(Data Restore Mode)    |              TX              |
|    exmode(Data Export Mode)      |              SQL             |
|    encoding                      |              UTF8            |
-----------------------------------------------------------------
pg_db.public=# unload tab device_profile;
正在解析表 <device_profile>. 已解析数据页: 15, 已解析数据: 141 条
表名<device_profile>-</data/pg/12/data/base/16384/29803> 解析完成, 15 个数据页 ,共计 141 条数据. 成功 141 条; 失败 0 条  |  导出的文件路径为:<pg_db/public/device_profile.sql>
```

导入恢复结果

```
postgres=# \c pg_restore;
You are now connected to database "pg_restore" as user "postgres".
pg_restore=# truncate table device_profile;
TRUNCATE TABLE
pg_restore=# \i /data/tools/pdu/pg_db/public/device_profile.sql
INSERT 0 1
INSERT 0 1
.....
INSERT 0 1
INSERT 0 1
pg_restore=# select count(1) from device_profile;
 count
-------
   141
(1 row)
```

第三章 使用 PDU 恢复

PDU 数据恢复快速上手

b;
use <database>(database,首先必须选择需要恢复的数据库);
set <schema>(schema,一般默认是 public);

unload tab<table>(恢复特定表);
unload sch <schema>(恢复特定模式下面所有表);

PDU 恢复数据的几种场景

单个 oid 文件恢复

1. 把需要恢复的文件放到 restore/datafile 目录中
2. 在 pdu 中执行 use restore;
3. 根据表的创建语句获取列的类型（也可以通过 scan manual;获取列类型等信息）
4. 执行 add <filenode> <tablename> <attibutes>;（例如：add 12345 t_pdu varchar,varchar,timestamp,varchar,numeric,varchar,varchar,varchar,numeric;）
5. 执行 unload tab t_pdu;

删除表数据恢复

1. 修改 pdu.ini 中的 ARCHIVE_DEST 参数，指向需要扫描的 arch 或者 wal 目录
2. 启动 pdu,设置日志扫描起始位置：param startwal 和 param endwal
3. 进入到需要恢复的表的 schema 中(use 和 set 命令);
4. scan 表名(需要恢复删除表名字)
5. 根据扫描结果，如果是多个事务，建议设置 param starttime|endtime <TIME>;重新扫描，然后进行 restore del all;
6. 如果只要恢复特定事务记录，可以直接使用 restore del <Tx Number>;进行恢复

恢复特定数据库目录

使用 restore db <dbname> <DB Path>;命令加载目录，然后和正常字典方式恢复

第四章 配置文件说明

配置文件说明(pdu.ini 文件)

pdu.ini 文件中有两个参数 PGDATA 和 ARCHIVE_DEST
PGDATA 指定为需要恢复的库的 PGDATA 路径（一般可以和 pg 环境中的$PGDATA 一致）
ARCHIVE_DEST 在需要基于 wal/arch 恢复表删除记录的时候需要，指定为 wal/arch 所在目录

第五章 目录和文件说明

pdu 所在目录文件情况

```
[root@localhost pdu]# ls -ltra
total 15072
drwxrwxrwx 4 root root      71 Apr 10 18:49 ..
-rw-r--r-- 1 root root     103 Apr 14 19:52 license.pdu
-rwxr-xr-x 1 root root 1925576 Apr 23 17:46 pdu10
-rwxr-xr-x 1 root root 1925576 Apr 23 17:46 pdu11
-rwxr-xr-x 1 root root 1921480 Apr 23 17:46 pdu12
-rwxr-xr-x 1 root root 1921480 Apr 23 17:46 pdu13
-rwxr-xr-x 1 root root 1921424 Apr 23 17:46 pdu14
-rwxr-xr-x 1 root root 1925896 Apr 23 17:46 pdu16
-rwxr-xr-x 1 root root 1925896 Apr 23 17:46 pdu15
-rwxr-xr-x 1 root root 1925896 Apr 23 17:46 pdu17
-rw-r--r-- 1 root root      63 Apr 24 20:58 pdu.ini
drwxr-xr-x 6 root root      60 Apr 24 21:10 restore
drwxr-xr-x 4 root root      32 Apr 24 21:10 postgres
-rw-r--r-- 1 root root      59 Apr 24 21:10 pg_database.txt
drwxr-xr-x 7 root root      69 Apr 24 21:12 pg_db
drwxr-xr-x 2 root root     192 Apr 24 21:24 log
drwxr-xr-x 7 root root     237 Apr 24 21:37 .
```

license.pdu 文件为 pdu 企业版授权 liscense
pdu10-17 文件为 pdu 主程序执行文件
restore 目录为恢复孤立的 oid 文件表数据时，存放 oid 文件位置
pdu.ini 文件为 pdu 程序配置文件
postgres 和 pg_db 目录为当前字典加载出来的数据库名字而创建的命令
pg_database.txt 文件为当前指定的 PGDATA 的 pg 数据库目录中包含的数据库列表文件
log 目录为恢复日志文件存储目录

数据库名目录文件情况

```
[root@localhost pdu]# cd pg_db
[root@localhost pg_db]# ls -ltr
total 4
drwxr-xr-x 2 root root    6 Apr 24 21:10 manual
drwxr-xr-x 2 root root  110 Apr 24 21:10 meta
drwxr-xr-x 2 root root   28 Apr 24 21:12 DDL
drwxr-xr-x 2 root root   29 Apr 24 21:12 COPY
```

```
drwxr-xr-x 2 root root 4096 Apr 24 21:24 public
```

manual 目录用来存放 pdu 执行“scan manual;”命令生成 pdu 恢复的表的字典信息的目录

meta 目录为数据库恢复的字典文件存储目录

```
[root@localhost pg_db]# ls -l meta
total 216
-rw-r--r-- 1 root root 158864 Apr 24 21:10 pg_attr.txt
-rw-r--r-- 1 root root   8441 Apr 24 21:10 pg_class.txt
-rw-r--r-- 1 root root    101 Apr 24 21:10 pg_schema.txt
-rw-r--r-- 1 root root  14078 Apr 24 21:10 pg_type.txt
-rw-r--r-- 1 root root  26343 Apr 24 21:10 public_tables.txt
```

DDL 目录为恢复过程中生成的 ddl 文件（创建表语句）所在目录

```
[root@localhost pg_db]# ls -l DDL
total 24
-rw-r--r-- 1 root root 20994 Apr 24 21:12 public_ddl.sql

[root@localhost pg_db]# head -10 DDL/public_ddl.sql
CREATE SCHEMA public;
set search_path to public;
CREATE TABLE tb_schema_settings(
        "schema_version" int8
);

CREATE TABLE admin_settings(
        "id" uuid,
        "tenant_id" uuid,
        "created_time" int8,
```

COPY 目录为生成 copy cvs 文件的命令

```
[root@localhost pg_db]# ls -ltr COPY/
total 8
-rw-r--r-- 1 root root 5183 Apr 24 21:12 public_copy.sql

[root@localhost pg_db]# head -5 COPY/public_copy.sql
set search_path to public;
COPY tb_schema_settings FROM '/data/tools/pdu/pg_db/public/tb_schema_settings.csv';
COPY admin_settings FROM '/data/tools/pdu/pg_db/public/admin_settings.csv';
COPY alarm FROM '/data/tools/pdu/pg_db/public/alarm.csv';
COPY entity_alarm FROM '/data/tools/pdu/pg_db/public/entity_alarm.csv';
```

public 为恢复的 schema 中表数据存储位置（根据 exmode 设置为 csv 和 sql 而显示不同）

```
[root@localhost pg_db]# ls -l public/|head -5
total 2296188
-rw-r--r-- 1 root root       2139 Apr 24 21:11 admin_settings.csv
-rw-r--r-- 1 root root        745 Apr 24 21:12 alarm_config.csv
```

```
-rw-r--r-- 1 root root        957 Apr 24 21:12 alarm_contact.csv
-rw-r--r-- 1 root root      14311 Apr 24 21:11 alarm.csv

[root@localhost pg_db]# ls -l public/*.sql
-rw-r--r-- 1 root root 81795 Apr 24 21:24 public/device_profile.sql
```