

# DataSan. Инструкция по установке

---

Перфоманс Лаб Сервисез



## Содержание

Введение .....	3
Функциональные характеристики.....	3
О документации .....	3
Минимальные требования к аппаратному и программному обеспечению ..	5
Требования к серверам .....	5
Требования к сетям .....	5
Требования к рабочему месту .....	5
Установка DataSan для PostgreSQL.....	6
Поставка ПО.....	6
Компиляция.....	6
Настройка разрешений .....	8
Профилирование.....	8
Решение проблем для PostgreSQL .....	12
Проблемы при установке .....	12
Проблемы при профилировании .....	12

# Введение

DataSan – инструмент для деперсонализации конфиденциальной информации и персональных данных в БД с сохранением работоспособности системы.

## Функциональные характеристики

DataSan предоставляет пользователю следующие функциональные возможности:

- деперсонализация конфиденциальной информации и персональных данных в базах данных PostgreSQL с сохранением работоспособности системы и консистентности данных;
- защита данных за счет кодирования:
  - хэширование – основной метод кодирования, реализованный на основе метода хэширования MD5;
  - маппинг – метод кодирования, реализованный на основе сопоставления символов;
- добавление функций для обработки:
  - уникальных типов данных, например, ОГРН и ИНН;
  - динамических типов данных, например, транслитерация ФИО;
- добавление функций обработки данных в общий процесс деперсонализации базы данных;
- возобновление деперсонализации базы данных после прерывания процесса, например, при выключении электричества. Деперсонализация продолжится с данных, на которых процесс был прерван;
- многопоточность, которая позволяет использовать инструмент в многоядерных конфигурациях стенда для ускорения процесса деперсонализации;
- очистка вспомогательных объектов по окончании процесса деперсонализации данных;
- встраивание деперсонализации базы данных в процессы разработки.

## О документации

Инструкция по установке предназначена для инженеров нагрузочного и функционального тестирования. После прочтения руководства инженеры смогут установить инструмент в СУБД.

Чтобы эффективно использовать DataSan, рекомендуется ознакомиться с работой сервисов и программных продуктов:

- СУБД PostgreSQL;
- среда разработки DBeaver.

Если вам требуется помощь, [свяжитесь с нами](#).

# Минимальные требования к аппаратному и программному обеспечению

## Требования к серверам

Процессор	2 ядра с частотой 2,5 ГГц
Оперативная память	От 8 Гбайт
HDD для IIS и документов	От 50 Гбайт. Зависит от размера хранимых в системе документов
SSD для SQL	От 300 Гбайт
СУБД	PostgreSQL 7 и выше
Установленные расширения для СУБД PostgreSQL	PL/JAVA, pgAgent

## Требования к сетям

Скорость передачи данных	От 10 Мбит/с
--------------------------	--------------

## Требования к рабочему месту

Установленное ПО	DBeaver
------------------	---------

# Установка DataSan для PostgreSQL

Общий порядок установки DataSan для PostgreSQL включает следующие этапы:

1. Перед установкой ознакомьтесь с [типовыми требованиями к аппаратному и программному обеспечению](#).
2. [Получите экземпляр ПО](#).
3. [Скомпилируйте проект](#).
4. [Настройте разрешения для пользователя](#), под которым будет запускаться деперсонализация БД.
5. [Выполните профилирование БД](#).

Контакты технических специалистов, которые могут проконсультировать по процессу установки, настройки экземпляра ПО и его функционирования:

Ростислав Николаев	<a href="mailto:r.nikolaev@pflb.ru">r.nikolaev@pflb.ru</a>	+7 999 835-52-08
Петр Мелченко	<a href="mailto:p.melchenko@pflb.ru">p.melchenko@pflb.ru</a>	+7 985 571-72-60

## Поставка ПО

В рамках поставки клиент получает экземпляр ПО и документацию, которая подробно и понятно описывает эксплуатацию ПО, его настройку, устранение типичных неисправностей и частые вопросы, возникающие у пользователей в процессе эксплуатации.


Экземпляр ПО – 7Z-архив, который содержит файлы для компиляции в среде разработки:

- **1\_Create\_Java.sql** – алгоритмы обезличивания на языке Java;
- **2\_Create\_jproc.sql** – методы обезличивания на языке Java;
- **3\_Create\_Object.txt** – вспомогательные таблицы;
- **4\_Pflb\_Datasan.txt** – пакет с процедурами DataSan;
- **5.prep proce.txt** – процедуры для создания таблицы **PFLB\_VIEWCONTENT**;
- **6.prep code.txt** – алгоритм для выполнения процедур **PFLB\_VIEWCONTENT**;
- **7\_Body.txt** – движок DataSan.

## Компиляция

Чтобы скомпилировать проект:

1. Распакуйте 7Z-архив с экземпляром ПО.
2. Запустите среду разработки, например, DBeaver.

3. В контекстном меню БД выберите пункт **Connect**.
4. Укажите данные для подключения к БД и нажмите на кнопку **Connect**.
5. Последовательно скомпилируйте все файлы из 7Z-архива с экземпляром ПО:
  - A. Откройте файл для компиляции и скопируйте его содержимое.
  - B. В IDE в область **Worksheet** вставьте содержимое файла.
  - C. Выделите текст с помощью горячих клавиш **CTRL+A** и нажмите на кнопку .

После компиляции структура проекта выглядит следующим образом:

- Tables:
  - PFLB\_ACTIVE\_JOBS;
  - PFLB\_ACTIVE\_STATUS;
  - PFLB\_ALL\_TAB\_INDEXES;
  - PFLB\_ALL\_TRIGGERS;
  - PFLB\_CURRENT\_STATUS;
  - PFLB\_EVEN\_TABLES;
  - PFLB\_JOB\_PROCESSED\_QUERIES;
  - PFLB\_LOGS;
  - PFLB\_LT\_DEPERSONTABLES;
  - PFLB\_NOT\_PROCESSED\_JOBS;
  - PFLB\_PROCESSED\_QUERIES;
  - PFLB\_TABLE\_SIZE;
  - PFLB\_TABLE\_TRIGGERS;
  - PFLB\_TEMP\_TABLE\_INDEXES;
  - PFLB\_VIEWCONTENT;
- Packages:
  - PFLB\_DATASAN;
  - PFLB\_DATASAN\_PREP;
- Functions:
  - PFLB\_ENCODE\_HASH\_BDATE;
  - PFLB\_ENCODE\_HASH\_BINARY;
  - PFLB\_ENCODE\_HASH\_CARDNUMBER;
  - PFLB\_ENCODE\_HASH\_CDATE;
  - PFLB\_ENCODE\_HASH\_CHAR;
  - PFLB\_ENCODE\_HASH\_CHARDEL;
  - PFLB\_ENCODE\_HASH\_DATE;
  - PFLB\_ENCODE\_HASH\_INNUMBER;
  - PFLB\_ENCODE\_HASH\_IP;
  - PFLB\_ENCODE\_HASH\_NCHAR;
  - PFLB\_ENCODE\_HASH\_OGRN;

- PFLB\_ENCODE\_HASH\_PHONENUMBER;
- PFLB\_ENCODE\_HASH\_PPCHAR;
- PFLB\_ENCODE\_HASH\_SNILSNUMBER;
- Java:
  - BitConverter;
  - OraSQL.

## Настройка разрешений

Настройте разрешения для пользователя, под которым будет запускаться деперсонализация БД:

```
GRANT ALL on schema PFLBTEST to PFLBUSER;
GRANT ALL ON ALL SEQUENCES IN SCHEMA PFLBTEST TO PFLBUSER;
GRANT USAGE ON LANGUAGE java TO PFLBUSER;
```

## Профилирование

Профилирование – поиск колонок в таблицах в базе данных, содержащих персональные данные.

Чтобы выполнить профилирование БД:

1. Найдите таблицы и поля в БД, которые содержат персональные данные:
  - A. Получите список всех таблиц, соответствующих определенной схеме:

```
SELECT schemaname owner,relname table_name,n_live_tup num_rows
FROM pg_stat_user_tables
where schemaname='schema_name'
ORDER BY n_live_tup DESC;
```

Пример списка таблиц:

	ABC OWNER ▼	ABC TABLE_NAME ▼	123 NUM_ROWS ↓ ▼
1	USERFORDEPERS	DATASAN_EMPLOYEES	300 001
2	USERFORDEPERS	DATASAN_ACCOUNT	300 001
3	USERFORDEPERS	CLIENTS_BACKUP	300 001
4	USERFORDEPERS	DATASAN_CLIENTS	300 001
5	USERFORDEPERS	PFLB_LOGS	38
6	USERFORDEPERS	DATASAN_BLACKLIST	8
7	USERFORDEPERS	DATASAN_SUPPLIERS	7
8	USERFORDEPERS	DATASAN_DELIVERY	6
9	USERFORDEPERS	PFLB_VIEWCONTENT	5
10	USERFORDEPERS	DATASAN_PRODUCT	5

- B. Выберите таблицы с большим количеством строк. Рекомендуется



- выбирать таблицы, которые содержат более 10000 строк.
- C. Посмотрите содержимое каждой из найденных таблиц.
- D. Сохраните тип данных и названия полей, которые содержат персональные данные.
2. Соберите представления на основании найденных полей и выберите методы кодирования для каждого типа данных. Метод кодирования указывается в конце наименования представления:
- **char** – смешанные и неопределенные поля;
  - **chardel** – адреса и ФИО;
  - **nchar** – названия организации и ФИО;
  - **phonenumber** – номера телефонов;
  - **ppchar** – серия и номер паспорта;
  - **bdate** – дата рождения;
  - **inn\_umber** – ИНН;
  - **cdate** – даты в интервале за год до текущего дня;
  - **cardnumber** – номер банковской карты;
  - **ip** – IP-адреса.

Примеры:

- создание представления для поля **%E-MAIL%** и кодирование методом **char**:

```
create view PFLB_LT_Deperson_CommonEmail_CHAR
as
select C.*
from information_schema.columns c
inner join information_schema.tables t on c.table_name=t.table_name
where
    (C.COLUMN_NAME ILIKE '%EMAIL%' or C.COLUMN_NAME ILIKE '%E_MAIL%'
or C.COLUMN_NAME ILIKE '%E-MAIL%')
and C.DATA_TYPE = 'character varying';
```

- создание представления для поля **%INN%** и кодирование методом **inn\_umber**:

```
create view PFLB_LT_Deperson_Common_INNNUMBER
as
select C.*
from information_schema.columns c
inner join information_schema.tables t on c.table_name=t.table_name
where
    C.COLUMN_NAME ILIKE '%INN%'
and (C.DATA_TYPE = 'integer' or C.DATA_TYPE = 'numeric');
```

- создание представления для полей с названиями организаций и ФИО, их кодирование методом **nchar**:

```

create view PFLB_LT_Deperson_PhisFIO_NCHAR
as
select C.*
from information_schema.columns c
inner join information_schema.tables t on c.table_name=t.table_name
where
(c.table_name not like 'pflb%' and c.table_name not like
'viewcontent')
and
(
C.COLUMN_NAME ILIKE '%FIRST_NAME%'
or C.COLUMN_NAME ILIKE '%LAST_NAME%'
or C.COLUMN_NAME ILIKE '%SECOND_NAME%'
or C.COLUMN_NAME ILIKE '%MIDDLENAME%'
or C.COLUMN_NAME ILIKE '%PAYERNAME%'
or C.COLUMN_NAME ILIKE '%RECIEVERNAME%'
or C.COLUMN_NAME ILIKE '%CLIENTFULLNAME%'
or C.COLUMN_NAME ILIKE '%PERSONFULLNAME%'
or C.COLUMN_NAME ILIKE '%PAYERFULLNAME%'
or C.COLUMN_NAME ILIKE '%RECIEVERFULLNAME%'
or C.COLUMN_NAME ILIKE '%BUYERFULLNAME%'
or C.COLUMN_NAME ILIKE '%SELLERFULLNAME%'
or C.COLUMN_NAME ILIKE '%BENEFICAIARYFULLNAME%'
or C.COLUMN_NAME ILIKE '%BOSSFULLNAME%'
or C.COLUMN_NAME ILIKE '%KINFULLNAME%'
or C.COLUMN_NAME ILIKE '%NAME%'
)
and C.DATA_TYPE = 'character varying'
and C.TABLE_SCHEMA != 'information_schema';

```

3. Убедитесь, что поля в представлениях не повторяются, иначе в таблице для деперсонализации данных будут дублироваться строки.
4. Вставьте в промежуточную таблицу **PFLB\_LT\_DepersonTables** содержимое всех представлений. Например:

```

insert into PFLB_LT_DepersonTables values
('PFLB_LT_Deperson_CommonEmail_CHAR');
insert into PFLB_LT_DepersonTables values
('PFLB_LT_Deperson_Common_INNNUMBER');
insert into PFLB_LT_DepersonTables values
('PFLB_LT_Deperson_PhisFIO_NCHAR');

```

5. Запустите процедуру по наполнению основной таблицы для деперсонализации данных **PFLB\_VIEWCONTENT**:

```

CALL PFLB_DATASAN_PREP.PFLB_FILL_VIEWCONTENT('HASH');

```

Таблица **PFLB\_VIEWCONTENT** наполнится записями вида **имя\_таблицы + наименование\_колонки**, которые ранее были собраны в представлениях. Пример наполненной таблицы **PFLB\_VIEWCONTENT**:

	OWNER_NAME	TABLE_NAME	COLUMN_NAME	COLUMN_TYPE	ENCODE_METHOD	COLUMN_ENCODE_TYPE	EXAMPLE
1	TEST2_DB	DATASAN_CLIENTS	CLIENT_ADDRESS	VARCHAR	HASH	CHARDEL	554039, Тюменская область, город Н
2	TEST2_DB	DATASAN_CLIENTS	CLIENT_FULLNAME	VARCHAR	HASH	CHARDEL	Лихачев Владимир Христофорович
3	TEST2_DB	DATASAN_CLIENTS	CLIENT_PHONE_NUMBER	VARCHAR	HASH	PHONENUMBER	7(477)196-49-67
4	TEST2_DB	DATASAN_CLIENTS	CLIENT_PASSPORT	VARCHAR	HASH	PPCHAR	0123 456789
5	TEST2_DB	DATASAN_CLIENTS	CLIENT_BIRTHDAY	DATE	HASH	BDATE	01.01.2021

6. Проанализируйте каждую строку таблицы **PFLB\_VIEWCONTENT** и при необходимости скорректируйте ее:

- удалите все дубликаты, неправильно выбранные методы, пустые или системные таблицы. Например:

OWNER_NAME	TABLE_NAME	COLUMN_NAME	COLUMN_TYPE	ENCODE_METHOD	COLUMN_ENCODE_TYPE	EXAMPLE
TEST2_DB	DATASAN_CLIENTS	CLIENT_ADDRESS	VARCHAR	HASH	CHARDEL	554039, Тюменская область, город Н
TEST2_DB	DATASAN_CLIENTS	CLIENT_FULLNAME	VARCHAR	HASH	CHARDEL	Лихачев Владимир Христофорович
TEST2_DB	DATASAN_CLIENTS	CLIENT_PHONE_NUMBER	VARCHAR	HASH	PHONENUMBER	7(477)196-49-67
TEST2_DB	DATASAN_CLIENTS	CLIENT_PASSPORT	VARCHAR	HASH	PPCHAR	0123 456789
TEST2_DB	DATASAN_CLIENTS	CLIENT_BIRTHDAY	DATE	HASH	BDATE	01.01.2021
TEST2_DB	DATASAN_CLIENTS	CLIENT_BIRTHDAY	DATE	HASH	BDATE	01.01.2021

- удалите таблицы, которые не содержат персональные данные. Например, таблица **DATASAN\_LOGS** содержит колонку **CLIENT\_ADDRESS**, но в этом поле нет персональных данных:

6	USERFORDEPERS	DATASAN_LOGS	CLIENT_ADDRESS	VARCHAR	HASH	CHARDEL	Случайный комент
---	---------------	--------------	----------------	---------	------	---------	------------------

# Решение проблем для PostgreSQL

## Проблемы при установке

### Недостаточно разрешений

Отсутствуют необходимые разрешения у пользователя, под которым запускается деперсонализация БД.

#### Решение:

Запросите необходимые разрешения у администратора БД. Подробнее см. раздел [Настройка разрешений](#).

### Таблица или представление пользователя отсутствует

Не были скомпилированы некоторые таблицы, представления или процедуры.

#### Решение:

1. Найдите, какие таблицы, представления или процедуры не были скомпилированы из списка всех вспомогательных таблиц.
2. Скомпилируйте необходимые таблицы, представления или процедуры повторно.

Подробнее см. раздел [Компиляция](#).

## Проблемы при профилировании

### Неправильно выбран метод обезличивания данных

#### Решение:

Замените метод обезличивания через **update** в таблице

**PFLB\_VIEWCONTENT** на необходимый метод из списка методов.

Подробнее см. раздел [Профилирование](#).

### Дубликаты в таблице PFLB\_VIEWCONTENT

#### Решение:

1. Проверьте содержимое представлений и корректность списка колонок. Некоторые колонки дублируются или агрегируются неправильно из-за символов **%%** в представлении.
2. Явно укажите название колонки или укажите название с помощью символов **%%**, игнорируя символы до и после.
3. Удалите похожие колонки.

### В таблицу PFLB\_VIEWCONTENT попали колонки без

## **персональных данных**

### **Возможные причины и решения:**

- **PFLB\_VIEWCONTENT** – промежуточная таблица. На этапе профилирования в ней могли находиться персональные данные, но спустя время они были удалены. Рекомендуется оставить колонки.
- Колонки попали при создании представлений и таблицы **PFLB\_VIEWCONTENT**. Проверьте колонки вручную и при необходимости удалите их.

### **Нет метода для определенного вида персональных данных**

#### **Решение:**

Обратитесь в службу техподдержки ООО «Перфоманс Лаб Сервисез» по электронной почте [datasan@pflb.ru](mailto:datasan@pflb.ru).