



Управление расширениями



Расширения в PostgreSQL

Создание расширений

Обновление на новую версию

Заложена в архитектуре системы

Возможности

функции и языки программирования

типы данных и операторы

индексы и методы доступа

обертки сторонних данных (FDW)

Расширение → contrib → ядро

Расширение — упаковка связанных объектов БД

Нельзя удалить отдельный объект,
только расширение целиком

pg_dump не выгружает отдельные объекты

Инструменты для обновления расширений

Информация

список расширений

`pg_extenstion`

`\dx`

состав расширения

`pg_depend where deptype = 'e'`

`\dx+ имя`

Команды управления расширениями

CREATE EXTENSION

ALTER EXTENSION

DROP EXTENSION

Файлы расширений

управляющие файлы

файлы SQL

Управляющие файлы

Имя основного файла: *имя.control*
Расположение: SHAREDIR/extension

```
# Некоторые параметры

directory = 'extension' # каталог для файлов SQL
default_values = 1.0 # версия по умолчанию
relocatable = true # возможность изменить схему
superuser = true # создает только суперпользователь
encoding = UTF8 # кодировка файлов SQL
#requires = '' # должны быть установлены

comment = 'Use only ASCII characters'
```

pg_available_extensions

Создание расширения

```
CREATE EXTENSION имя [VERSION 'версия'];
```

имя.control

```
...  
default_version = 1.0  
...
```

если версия
не указана

имя--версия.sql

```
\echo Use "CREATE EXTENSION имя" to load this file. \quit  
comment on extension имя is 'Описание на русском';  
-- Создание объектов расширения  
...
```

pg_available_extension_versions

Транзакционность

```
CREATE EXTENSION ИМЯ;
```

```
BEGIN;
```

имя--версия.sql

```
create function ...  
create view ...  
create table ...
```

```
BEGIN;  
VACUUM;  
COMMIT;
```

нетранзакционные
команды запрещены

```
COMMIT;
```

Выбор схемы

```
CREATE EXTENSION имя [SCHEMA схема];
```

имя.control

```
...  
relocatable = false  
schema = схема
```

имя--версия.sql

```
SET LOCAL search_path TO @extschema@ ;  
... select @extschema@ .function_name()...
```

макроподстановка

Использование таблиц

```
CREATE EXTENSION ИМЯ;
```

имя--версия.sql

```
create table table_name ...  
insert into table_name ...  
  
select pg_extension_config_dump  
       ('table_name::regclass, 'WHERE ...');
```

ВЫВОД утилиты `pg_dump`

```
CREATE EXTENSION IF NOT EXISTS ИМЯ WITH SCHEMA схема;  
  
COPY table_name (...) FROM stdin;  
...  
\.
```

pg_extension (extconfig, extcondition)

Управляющие файлы

Дополнительные файлы: *имя--версия.control*

Расположение: SHAREDIR/extension

имя.control

```
...  
#requires = ''                должны быть установлены
```

имя--1.1.control

```
...  
requires = 'postgres_fdw'
```

Обновление расширения

1) `CREATE EXTENSION ИМЯ;`

ИМЯ.control

```
default_version = 1.1  
...
```

ИМЯ--1.1.sql

```
\echo Use "CREATE EXTENSION ИМЯ" to load this file. \quit  
-- Создание всех объектов версии 1.1
```

2) `ALTER EXTENSION ИМЯ UPDATE TO '1.1';`

ИМЯ--1.0--1.1.sql

```
\echo Use "ALTER EXTENSION ИМЯ" to load this file. \quit  
-- Команды обновления из версии 1.0 в 1.1
```

Обновление расширения

```
ALTER EXTENSION ИМЯ UPDATE TO '1.2';
```

текущая версия 1.0

имя--1.0--1.1.sql

```
\echo Use "ALTER EXTENSION ИМЯ" to load this file. \quit  
-- Команды обновления из версии 1.0 в 1.1
```

имя--1.1--1.2.sql

```
\echo Use "ALTER EXTENSION ИМЯ" to load this file. \quit  
-- Команды обновления из версии 1.1 в 1.2
```

`pg_extension_update_paths ('ИМЯ')`

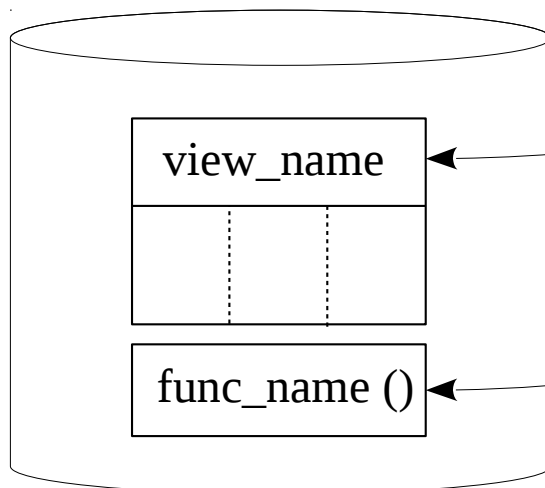
Создание расширения

Особый случай: включение существующих объектов

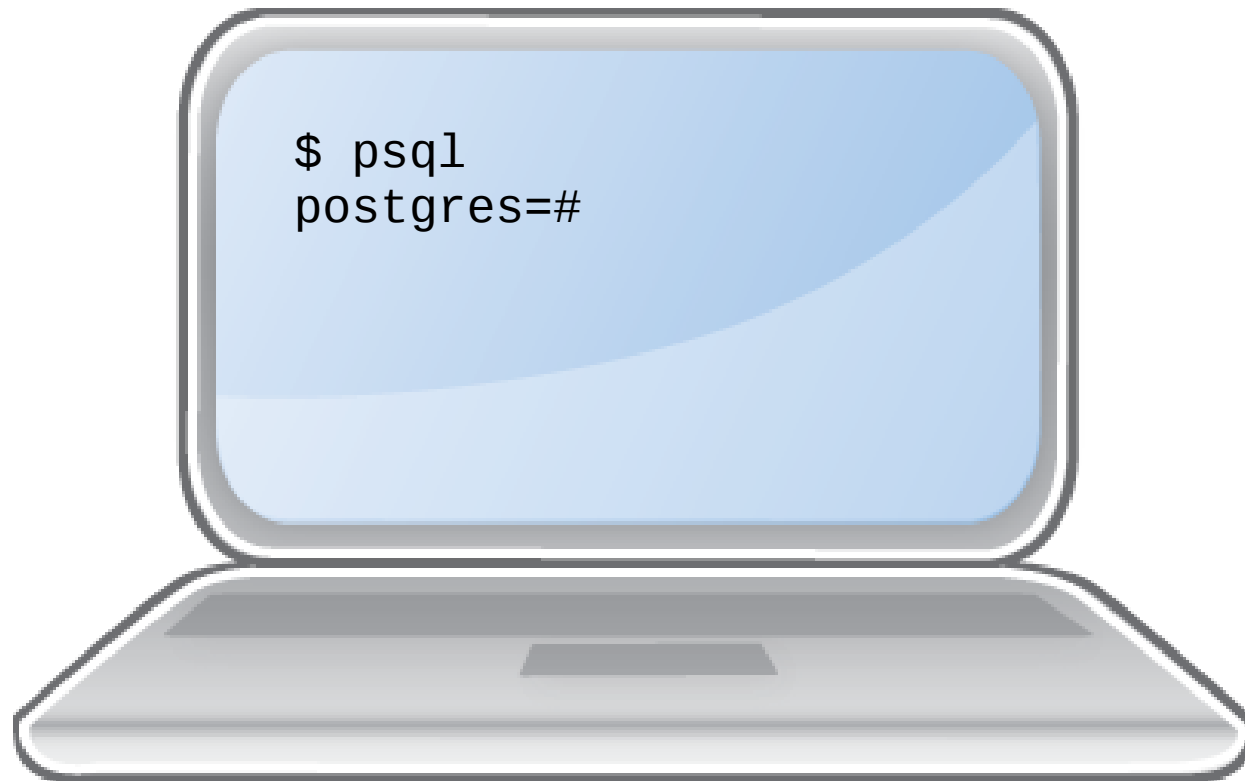
```
CREATE EXTENSION ИМЯ FROM unpackaged;
```

ИМЯ--unpackaged--1.0.sql

```
alter extension ИМЯ ADD function func_name();  
alter extension ИМЯ ADD view view_name;  
  
-- Создание других объектов
```



Демонстрация



Расширяемость заложена в архитектуре PostgreSQL

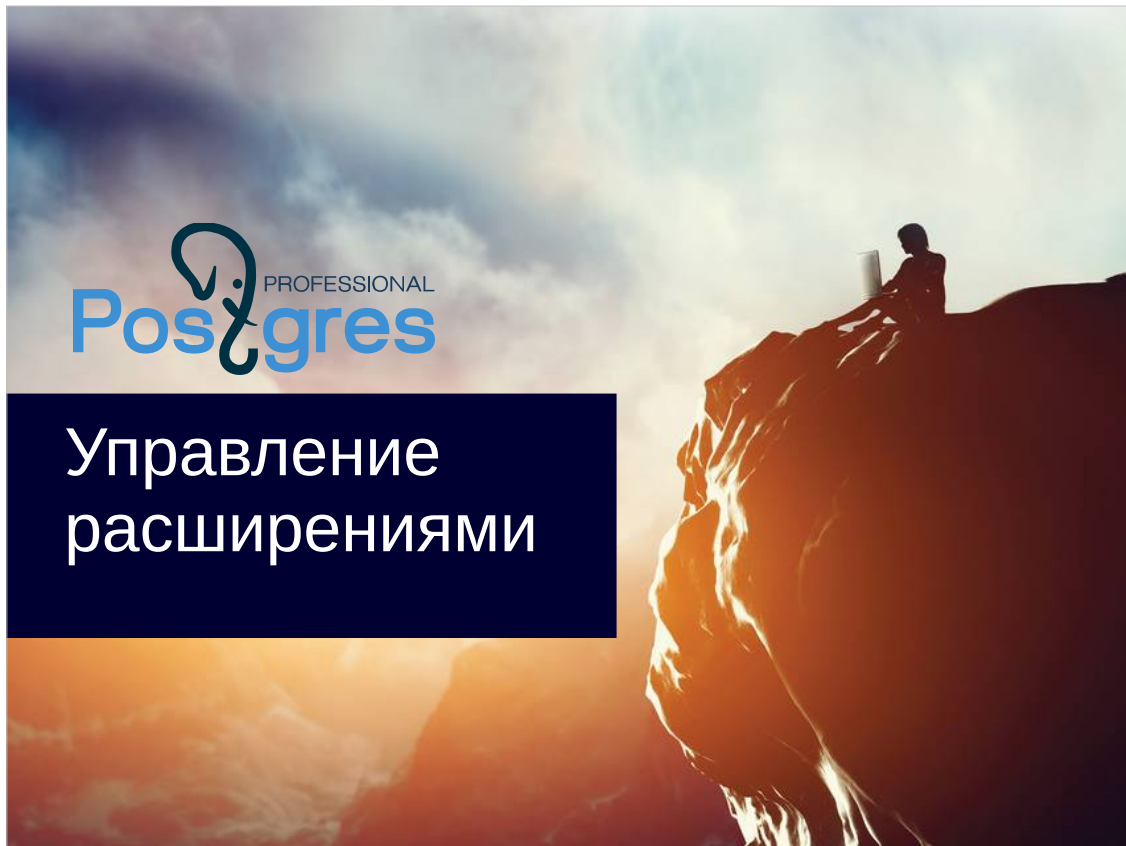
Расширения — упаковка связанных объектов БД

Управляющие файлы содержат важные свойства расширений

Код создания объектов БД записывается в файлах SQL

Механизм расширений позволяет поддерживать и управлять версиями расширений

1. Установить в БД db24 расширение uom версии 1.1.
2. Добавить футы и дюймы в таблицу.
Проверить, сколько дюймов в одном аршине.
3. Обновить расширение uom до версии 1.2.
4. Проверить, что pg_dump выгружает из таблицы футы и дюймы.



Авторские права

Курс «Администрирование PostgreSQL 9.4. Расширенный курс» разработан в компании Postgres Professional (2015 год).

Авторы: Егор Рогов, Павел Лузанов

Использование материалов курса

Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

Обратная связь

Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Отказ от ответственности

Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или косвенным, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Расширения в PostgreSQL

Создание расширений

Обновление на новую версию

Заложена в архитектуре системы

Возможности

- функции и языки программирования
- типы данных и операторы
- индексы и методы доступа
- обертки сторонних данных (FDW)

Расширение → contrib → ядро

PostgreSQL спроектирован с расчетом на расширяемость.

Можно подключать языки программирования и разрабатывать на них функции. Определять новые типы данных и операторы для работы с ними. Создавать новые методы доступа для типов данных. Разрабатывать обертки сторонних данных для подключения к внешним источникам.

И для этого не обязательно изменять основной код сервера PostgreSQL. Можно разработать расширение, которое будет подключаться к существующему серверу.

Если такое расширение станет популярным, то разработчики PostgreSQL могут принять решение о включение его в дистрибутив в составе модулей contrib. А со временем и включить в ядро PostgreSQL.

Расширение — упаковка связанных объектов БД

Нельзя удалить отдельный объект,
только расширение целиком

pg_dump не выгружает отдельные объекты

Инструменты для обновления расширений

Информация

список расширений

pg_extenstion

\dx

состав расширения

pg_depend where deptype = 'e'

\dx+ имя

Дополнительная функциональность, как правило, включает в себя несколько объектов. Например, для разработки нового типа данных, кроме самого типа, потребуются функции и операторы для работы с типом. А возможно и новые методы доступа.

Расширения PostgreSQL позволяют упаковать связанные объекты вместе. Это облегчает управление связанными объектами в БД:

- Объекты расширения нельзя удалить по отдельности.
- Утилита pg_dump не будет выгружать отдельные объекты. Вместо этого она будет выгружать команду CREATE EXTENSION.
- Специальные инструменты облегчают переход на новые версии расширений

Подробнее об упаковке объектов БД в расширения:

<http://www.postgresql.org/docs/9.5/interactive/extend-extensions.html>

Команды управления расширениями

CREATE EXTENSION

ALTER EXTENSION

DROP EXTENSION

Файлы расширений

управляющие файлы

файлы SQL

Для создания и изменения расширений используются команды SQL CREATE EXTENSION и ALTER EXTENSION.

Для своей работы этим командам требуются специальные файлы расширений. К таким файлам относятся управляющие файлы, содержащие важную информацию о свойствах расширения. А также файлы SQL, предназначенные для создания объектов расширения.

Расширения написанные на языке C содержат дополнительные файлы. В этом курсе они не рассматриваются.

Удалить все объекты расширения можно командой DROP EXTENSION. Не требуется разрабатывать отдельный скрипт "uninstall".

Имя основного файла: `имя.control`
Расположение: `SHAREDIR/extension`

```
# Некоторые параметры

directory = 'extension' # каталог для файлов SQL
default_values = 1.0    # версия по умолчанию
relocatable = true     # возможность изменить схему
superuser = true       # создает только суперпользователь
encoding = UTF8        # кодировка файлов SQL
#requires = ''         # должны быть установлены

comment = 'Use only ASCII characters'
```

`pg_available_extensions`

При выполнении команды `CREATE EXTENSION`, механизм расширений проверяет наличие управляющего файла в директории `SHAREDIR/extension`.

Расположение `SHAREDIR` можно посмотреть командой:
`pg_config --sharedir`

Имя управляющего файла состоит из имени расширения, к которому добавляется `".control"`.

Управляющий файл имеет формат конфигурационных файлов PostgreSQL и состоит из пар `key = value`. PostgreSQL не определяет кодировку символов файла, поэтому следует использовать только символы ASCII.

Наличие управляющего файла говорит PostgreSQL о возможности создания расширения. Список доступных для установки расширений находится в таблице системного каталога `pg_available_extensions`.

Создание расширения

```
CREATE EXTENSION имя [VERSION 'версия'];
```

имя.control

```
...  
default_version = 1.0  
...
```

если версия
не указана

имя--версия.sql

```
\echo Use "CREATE EXTENSION имя" to load this file. \quit  
comment on extension имя is 'Описание на русском';  
-- Создание объектов расширения  
...
```

pg_available_extension_versions

7

Помимо управляющего файла, при создании расширения требуется еще файл SQL с командами на создание объектов БД. Имя файла SQL зависит от устанавливаемой версии расширения.

Версию расширения можно явно указать во фразе VERSION команды CREATE EXTENSION. Если этого не сделать, то будет использоваться версия из параметра default_version управляющего файла.

Имя файла SQL строится по шаблону *имя--версия*.sql

, где *версия* не обязательно должна состоять из цифр. Она может включать и другие символы (кроме: "--", а также "-" в начале или конце).

Список доступных для установки версий расширений находится в таблице системного каталога pg_available_extension_versions.

Строки файла SQL, начинающиеся на "\echo" считаются механизмом расширений комментариями. Поэтому обычно в начале файла SQL добавляют такую строку с предупреждением и завершают её командой "\quit". Это делается для того, чтобы такой файл не был случайно выполнен из psql.

В файле SQL можно использовать кириллицу, предварительно указав кодировку символов в параметре encoding управляющего файла. Например, можно задать русскоязычный комментарий к расширению в команде COMMENT ON EXTENSION.

```
CREATE EXTENSION ИМЯ;
```

```
BEGIN; ИМЯ--версия.sql
```

```
create function ...  
create view ...  
create table ...
```

```
BEGIN;  
VACUUM;  
COMMIT;
```

нетранзакционные
команды запрещены

```
COMMIT;
```

Файл SQL неявно выполняется в рамках одной транзакции.

Поэтому в нем можно использовать любые команды SQL, кроме команд управления транзакциями (BEGIN, COMMIT и т. д.) и команд, которые не могут выполняться внутри блока транзакции (например, VACUUM).

```
CREATE EXTENSION имя [SCHEMA схема];
```

имя.control

```
...  
relocatable = false  
schema = схема
```

имя--версия.sql

```
SET LOCAL search_path TO @extschema@;  
... select @extschema@.function_name()...
```

макроподстановка

Расширение не принадлежит ни одной схеме. Но все объекты расширения должны создаваться в какой-то схеме. Более того, расширение может создавать схемы.

Хотя это и не обязательно, но обычно все объекты расширения размещают в одной схеме.

Для указания схемы размещения объектов есть несколько возможностей. Схему можно явно указать в команде CREATE EXTENSION. Схему можно задать параметром расширения `schema`.

В противном случае будет использоваться первая схема из параметра `search_path`.

Выбранная в результате схема явно устанавливается в `search_path` в начале выполнения файла SQL. Внутри самого файла можно обращаться к этой схеме используя макроподстановку `@extschema@`.

Если параметр `relocatable` установить в `true` (по умолчанию `false`), то расширение после установки можно переносить в другую схему командой: `ALTER EXTENSION ... SET SCHEMA ...`

```
CREATE EXTENSION ИМЯ;
```

ИМЯ--версия.sql

```
create table table_name ...  
insert into table_name ...  
  
select pg_extension_config_dump  
       ('table_name::regclass, 'WHERE ...');
```

вывод утилиты `pg_dump`

```
CREATE EXTENSION IF NOT EXISTS ИМЯ WITH SCHEMA схема;  
COPY table_name (...) FROM stdin;  
...  
\.
```

`pg_extension (extconfig, extcondition)`

10

В расширение можно включать и таблицы.

Если не предпринимать специальных действий, то строки таблиц, вставленные после установки расширения, не будут выгружаться утилитой `pg_dump`.

Если же требуется включить содержимое таблиц в выгрузку `pg_dump`, то в файле SQL расширения нужно вызвать для каждой таблицы функцию `pg_extension_config_dump()`.

Функция имеет два параметра. Первый это OID таблицы, второй (необязательный) - фраза `WHERE`, которую `pg_dump` будет применять к таблице при выгрузке.

Эта же функция используется и для последовательностей, которые могут быть связаны с таблицей. В вывод `pg_dump` будет записываться вызов функции `setval`, устанавливающий последнее полученное из последовательности значение.

Функцию `pg_extension_config_dump()` можно вызывать только из файлов SQL расширения.

Дополнительные файлы: *имя--версия.control*

Расположение: SHAREDIR/extension

имя.control

```
...  
#requires = ''           должны быть установлены
```

имя--1.1.control

```
...  
requires = 'postgres_fdw'
```

Помимо основного управляющего файла расширения, могут быть еще и дополнительные файлы для каждой версии расширения.

Формат имени таких файлов следующий: *имя—версия.control*

При установке расширения, считывается и дополнительный файл этой версии (если он есть). Параметры дополнительного файла имеют предпочтения перед основным.

В приведенном примере, при установке версии расширения 1.1 будет проверяться наличие установленного расширения `postgres_fdw`, хотя в предыдущей версии этого не требовалось.

Параметры `directory` и `default_version` нельзя задавать в дополнительных управляющих файлах.

1) CREATE EXTENSION *ИМЯ*;

ИМЯ.control

```
default_version = 1.1  
...
```

ИМЯ--1.1.sql

```
\echo Use "CREATE EXTENSION ИМЯ" to load this file. \quit  
-- Создание всех объектов версии 1.1
```

2) ALTER EXTENSION *ИМЯ* UPDATE TO '1.1';

ИМЯ--1.0--1.1.sql

```
\echo Use "ALTER EXTENSION ИМЯ" to load this file. \quit  
-- Команды обновления из версии 1.0 в 1.1
```

12

При выходе новой версии расширения возможны две ситуации:

1) Расширение еще не установлено. Поэтому потребуется обновленный управляющий файл (как минимум параметр `default_values` нужно установить в новую версию). А также файл SQL для новой версии, включающий создание всех объектов расширения.

2) Для тех, у кого установлена предыдущая версия расширения, можно подготовить файл SQL на обновление. Формат имени файла обновления следующий: *старая_версия--новая_версия.sql*. Внутри такого файла должны быть только команды на обновление со старой версии на новую.

Само обновление выполняется командой `ALTER EXTENSION ИМЯ UPDATE TO новая_версия;`

Обновление расширения

```
ALTER EXTENSION ИМЯ UPDATE TO '1.2';
```

текущая версия 1.0

имя--1.0--1.1.sql

```
\echo Use "ALTER EXTENSION имя" to load this file. \quit  
-- Команды обновления из версии 1.0 в 1.1
```

имя--1.1--1.2.sql

```
\echo Use "ALTER EXTENSION имя" to load this file. \quit  
-- Команды обновления из версии 1.1 в 1.2
```

`pg_extension_update_paths('имя')`

Выполнять обновление версий можно только при наличии файлов обновления.

Команда ALTER EXTENSION может выполнять последовательности файлов обновления для получения запрошенной версии.

Например, если мы выполняем обновлении версии 1.0 на версию 1.2, то механизм расширений последовательно выполнит файлы обновления *имя*--1.0--1.1.sql и *имя*--1.1--1.2.sql.

Также отметим, что можно создавать файлы SQL для обновления на более старую версию расширения, например *имя*--1.2--1.0.sql.

Функция `pg_extension_update_paths('имя')` показывает возможные пути обновления переданного параметром расширения.

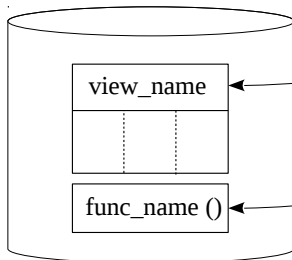
Создание расширения

Особый случай: включение существующих объектов

```
CREATE EXTENSION имя FROM unpackaged;
```

имя--unpackaged--1.0.sql

```
alter extension имя ADD function func_name();  
alter extension имя ADD view view_name;  
  
-- Создание других объектов
```



14

В PostgreSQL есть возможность включить в расширение уже существующие в БД объекты. Для этого используется команда `ALTER EXTENSION имя ADD object ...`

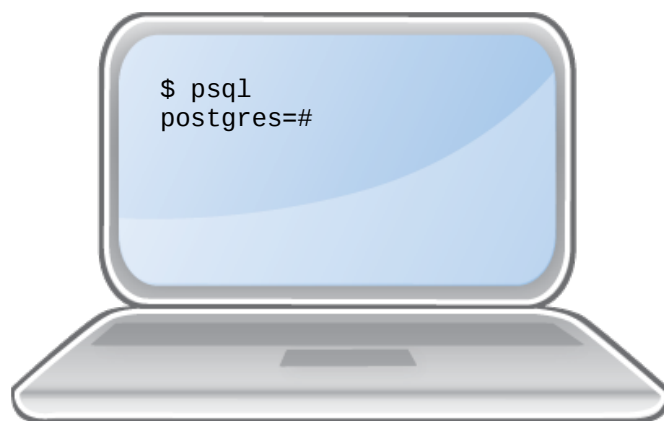
Отвязать объекты от расширения (без удаления объектов): `ALTER EXTENSION имя DROP object ...`

Эти команды не обязательно вызывать из файлов SQL расширения. Но их можно поместить в файл SQL с именем *имя--unpackaged--1.0.sql* и вызвать команду `CREATE EXTENSION имя FROM unpackaged;`

В результате из существующих объектов будет создано расширение с версией 1.0.

В общем случае, для использования с фразой FROM, имя файла SQL должно отвечать шаблону *имя--старая_версия--новая_версия.sql*.

В качестве *старой_версии* широко используется "unpackaged", но это не является обязательным, вместо него можно использовать любое допустимое имя версии.



Расширяемость заложена в архитектуре PostgreSQL

Расширения — упаковка связанных объектов БД

Управляющие файлы содержат важные свойства расширений

Код создания объектов БД записывается в файлах SQL

Механизм расширений позволяет поддерживать и управлять версиями расширений

1. Установить в БД db24 расширение uom версии 1.1.
2. Добавить футы и дюймы в таблицу.
Проверить, сколько дюймов в одном аршине.
3. Обновить расширение uom до версии 1.2.
4. Проверить, что pg_dump выгружает из таблицы футы и дюймы.

При добавлении записей в таблицу UOM учесть следующее:

1 фут = 0.3048 м

1 дюйм = 0.0254 м

Значение столбца predefined должно быть false.