**PostgreSQL**

PostgreSQL 9.6 <https://postgrespro.ru/docs/postgresql/9.6/preface>

Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом. Часть 1 <https://habr.com/ru/post/282764/>

PostgreSQL — это объектно-реляционная система управления базами данных (ОРСУБД, ORDBMS), основанная на POSTGRES, Version 4.2 — программе, разработанной на факультете компьютерных наук Калифорнийского университета в Беркли. В POSTGRES появилось множество новшеств, которые были реализованы в некоторых коммерческих СУБД гораздо позднее.

PostgreSQL — СУБД с открытым исходным кодом, основой которого был код, написанный в Беркли. Она поддерживает большую часть стандарта SQL и предлагает множество современных функций:

* сложные запросы
* внешние ключи
* триггеры
* изменяемые представления
* транзакционная целостность
* многоверсионность

Кроме того, пользователи могут всячески расширять возможности PostgreSQL, например создавая свои

* типы данных
* функции
* операторы
* агрегатные функции
* методы индексирования
* процедурные языки

А благодаря свободной лицензии, PostgreSQL разрешается бесплатно использовать, изменять и распространять всем и для любых целей — личных, коммерческих или учебных.

PostgreSQL не просто реляционная, а объектно-реляционная СУБД. Это даёт ему некоторые преимущества над другими SQL базами данных с открытым исходным кодом, такими как MySQL, MariaDB и Firebird.

Фундаментальная характеристика объектно-реляционной базы данных — это поддержка пользовательских объектов и их поведения, включая типы данных, функции, операции, домены и индексы. Это делает PostgreSQL невероятно гибким и надежным. Среди прочего, он умеет создавать, хранить и извлекать сложные структуры данных.

* **Отличительные характеристики PostgreSQL**

Существует обширный список типов данных, которые поддерживает Постгрес. Кроме числовых, с плавающей точкой, текстовых, булевых и других ожидаемых типов данных (а также множества их вариаций), PostgreSQL так же имеет поддержку uuid, денежного, перечисляемого, геометрического, бинарного типов, сетевых адресов, битовых строк, текстового поиска, xml, json, массивов, композитных типов и диапазонов, а также некоторых внутренних типов для идентификации объектов и местоположения логов. MySQL, MariaDB и Firebird тоже имеют некоторые из этих типов данных, но только PostgreSQL поддерживает их все.

*PostgreSQL имеет поддержку таких типов и структур данных как:*

* *Сетевые адреса*

PostgreSQL обеспечивает хранение разных типов сетевых адресов. Тип данных CIDR следует соглашению для сетевых адресов IPv4 и IPv6. Вот несколько примеров:

* 192.168.100.128/25
* 10.1.2.3/32
* 2001:4f8:3:ba:2e0:81ff:fe22:d1f1/128
* ::ffff:1.2.3.0/128

Также для хранения сетевых адресов доступен тип данных INET, используемый для IPv4 и IPv6 хостов, где подсети являются необязательными. Тип данных MACADDR может использоваться для хранения MAC-адресов для идентификации оборудования, таких как 08-00-2b-01-02-03.

У MySQL и MariaDB тоже есть INET функции для конвертации сетевых адресов, но они не предоставляют типы данных для внутреннего хранения сетевых адресов. У Firebird тоже нет типов для хранения сетевых адресов.

* *Многомерные массивы*

Поскольку PostgreSQL — это объектно-реляционная база данных, массивы значений могут храниться для большинства существующих типов данных. Сделать это можно путём добавления квадратных скобок к спецификации типа данных для столбца или с помощью выражения ARRAY. Размер массива может быть задан, но это необязательно.

MySQL, MariaDB, и Firebird так не умеют. Чтобы хранить такие массивы значений в традиционных реляционных базах данных, придется использовать обходной путь и создавать отдельную таблицу со строками для каждого из значений массива.

* *Геометрические данные*

Геоданные быстро становятся основным требованием для многих приложений. PostgreSQL уже давно поддерживает множество геометрических типов данных, таких как точки, линии, круги и многоугольники. Один из этих типов – PATH, он состоит из множества последовательно расположенных точек и может быть открытым (начальная и конечная точки не связаны) или закрытым (начальная и конечная точки связаны).

* *Поддержка JSON*

Поддержка JSON в PostgreSQL позволяет перейти к хранению schema-less данных в SQL базе данных. Это может быть полезно, когда структура данных требует определённой гибкости: например, если в процессе разработки структура всё ещё меняется или неизвестно, какие поля будет содержать объект данных.

Тип данных JSON обеспечивает проверку корректности JSON, который позволяет использовать специализированные JSON операторы и функции, встроенные в PostgreSQL для выполнения запросов и манипулирования данными. Также доступен тип JSONB — двоичная разновидность формата JSON, у которой пробелы удаляются, сортировка объектов не сохраняется, вместо этого они хранятся наиболее оптимальным образом, и сохраняется только последнее значение для ключей-дубликатов. JSONB обычно является предпочтительным форматом, поскольку требует меньше места для объектов, может быть проиндексирован и обрабатывается быстрее, так как не требует повторного синтаксического анализа.

В MySQL 5.7.8 и MariaDB 10.0.1 была добавлена поддержка встроенных объектов JSON. Но, хотя существует множество функций и операторов для JSON, которые теперь доступны в этих базах данных, они не индексируются так, как JSONB в PostgreSQL. Firebird пока что не присоединился к тренду и поддерживает объекты JSON только в виде текста.

* *Создание нового типа*

Если вдруг так случится, что обширного списка типов данных PostgreSQL вам окажется недостаточно, вы можете использовать команду CREATE TYPE, чтобы создать новые типы данных, такие как составной, перечисляемый, диапазон и базовый.

Поскольку они не являются объектно-реляционными, MySQL, MariaDB и Firebird не предоставляют такую мощную функциональность.

* **Обоснование выбора PostgreSQL.**

У PostgreSQL множество возможностей. Созданный с использованием объектно-реляционной модели, он поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных. Он обеспечивает расширенную ёмкость данных и заслужил доверие бережным отношением к целостности данных.