**PostgreSQL** — свободная объектно-реляционная система управления базами данных (СУБД). PostgreSQL базируется на языке SQL.

**Основные возможности**:

**Функции**

Функции являются **блоками кода**, исполняемыми на сервере, а не на клиенте БД. Хотя они могут писаться на чистом SQL, реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки SQL и требует использования некоторых **языковых расширений**.

**Триггеры**

Триггеры определяются как функции, **инициируемые DML-операциями**. Например, операция INSERT может запускать триггер, проверяющий добавленную запись на соответствия определённым условиям. При написании функций для триггеров могут использоваться различные **языки программирования** (см. выше).

Триггеры ассоциируются с таблицами. Множественные триггеры выполняются в **алфавитном порядке**.

**Правила и представления**

Механизм правил (англ. rules) представляет собой механизм создания пользовательских обработчиков не только DML-операций, но и операции выборки. Основное **отличие от механизма триггеров** заключается в том, что правила срабатывают на этапе разбора запроса, до выбора оптимального плана выполнения и самого процесса выполнения. Правила позволяют **переопределять поведение системы** при выполнении SQL-операции к таблице. Хорошим примером является реализация механизма представлений (англ. views): при создании представления создается правило, которое определяет, что вместо выполнения операции выборки к представлению система должна выполнять операцию выборки к базовой таблице/таблицам с учётом условий выборки, лежащих в основе определения представления. Для создания представлений, поддерживающих операции обновления, правила для операций вставки, изменения и удаления строк должны быть определены пользователем.

**Индексы**

В PostgreSQL имеется поддержка индексов следующих **типов**: B-дерево, хеш, GiST, GIN, BRIN, Bloom. При необходимости можно создавать новые типы индексов. Индексы в PostgreSQL обладают следующими свойствами:

* возможен просмотр индекса **не только в прямом, но и в обратном порядке** — создание отдельного индекса для работы конструкции ORDER BY ... DESC не нужно;
* возможно создание индекса **над несколькими столбцами таблицы**, в том числе над столбцами различных типов данных;
* индексы могут быть **функциональными**, то есть строиться не на базе набора значений некоего столбца/столбцов, а на базе набора значений функции от набора значений;
* индексы могут быть **частичными**, то есть строиться только по части таблицы (по некоторой её проекции); в некоторых случаях это помогает создавать намного более компактные индексы или достигать улучшения производительности за счёт использования разных типов индексов для разных (например, с точки зрения частоты обновления) частей таблицы;
* **планировщик запросов** может использовать несколько индексов одновременно для выполнения сложных запросов.

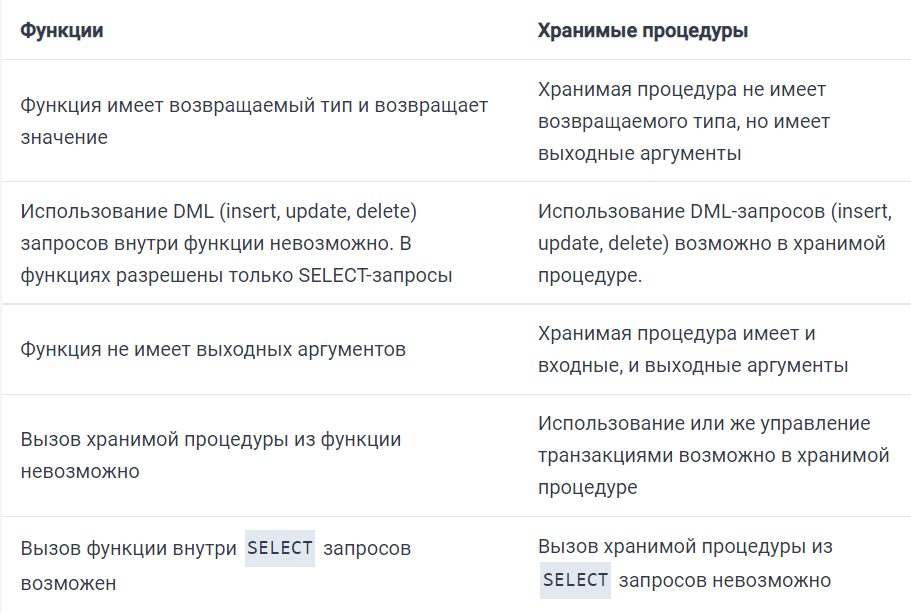
**Многоверсионность**

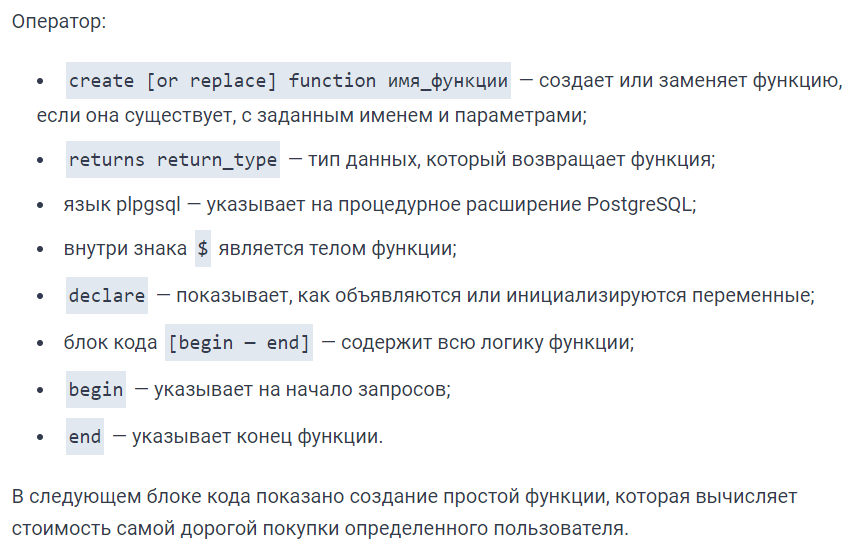
PostgreSQL поддерживает одновременную модификацию БД несколькими пользователями с помощью механизма **Multiversion Concurrency Control** (MVCC).

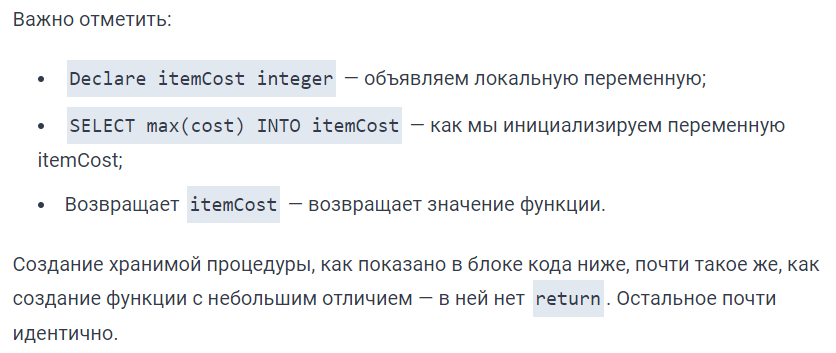
**Наследование и партицирование**

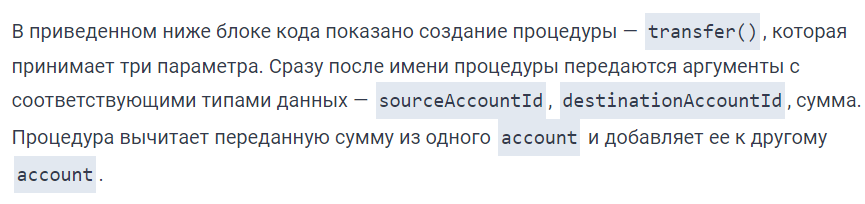
Таблицы могут **наследовать** характеристики и наборы полей от других таблиц (родительских). При этом данные, добавленные в порождённую таблицу, автоматически будут **участвовать** (если это не указано отдельно) в запросах к родительской таблице.

В PostgreSQL 10 был добавлен **механизм партицирования** таблиц. Партицирование предназначено для разделения одной таблицы на несколько, так называемые партиции.









<https://itnan.ru/post.php?c=1&p=325642>

<http://www.sai.msu.su/~megera/postgres/talks/fts_pgsql_intro.html>

<https://eax.me/postgresql-full-text-search/#:~:text=Полнотекстовый%20поиск%20(Full-Text%20Search%2C%20FTS),других%20РСУБД%2C%20очень%20даже%20неплохой>

Идея нового поиска состояла в том, чтобы затратить время на обработку документа **один раз** и сохранить время при поиске, использовать специальные программы-словари для **нормализации слов**, чтобы не заботиться, например, о **формах слов**, учитывать информацию о важности различных атрибутов документа и **положения слова** из запроса в документе для ранжирования найденных документов. Для этого, требовалось создать новые типы данных, соответствующие документу и запросу, и полнотекстовый оператор для сравнения документа и запроса, который возвращает TRUE, если запрос удовлетворяет запросу, и в противном случае - FALSE.

Таким образом, были созданы новые типы данных - **tsvector**, который является **хранилищем для лексем** из документа, оптимизированного для поиска, и **tsquery** - **для запроса** с поддержкой логических операций, полнотекстовый оператор "**две собаки**" @@ и **индексная поддержка** для него с использованием [GiST] и [GIN]. tsvector помимо самих лексем может хранить информацию о **положении лексемы** в документе и **ее весе** (важности), которая потом может использоваться для вычисления ранжирующей информации.

Кроме этого, были реализованы **вспомогательные функции**:

* **to\_tsvector** для преобразования документа в tsvector,
* **to\_tsquery** - для получения tsquery

Для разбиения документа на **токены** используется **парсер**, который выдает токен и его тип. Каждому **типу токена** ставится в соответствие **набор словарей**, которые будут стараться **распознать и "нормализовать"** его. Порядок словарей фиксирован и важен, так как именно в этом порядке токен будет попадать на вход словарю, до тех пор, пока он не опознается одним из них. Если токен **не распознался** ни одним из словарей, или словарь опознал его как **стоп-слово**, то этот токен **не индексируется**. Таким образом, можно сказать, что для каждого типа токена существует правило обработки токена, которое описывает схему попадания токена в полнотекстовый индекс.

Все это, конечно, здорово. Вот только доставать из базы мы хотим не константы, а **конкретные поля**, и для этого, наверное, нужно использовать какие-то **индексы**. Для полнотекстового поиска PostgreSQL предлагает два индекса на выбор:

**GIN** — быстро ищет, но не слишком быстро обновляется. Отлично работает, если вы сравнительно редко меняете данные, по которым ищите;

**GiST** — ищет медленнее GIN, зато очень быстро обновляется. Может лучше подходить для поиска по очень часто обновляемым данным;