Postgres Pro поддерживает несколько типов индексов: B-дерево, хеш, GiST, SP-GiST, GIN, BRIN

Более предпочтительными для текстового поиска являются индексы GIN. Будучи инвертированными индексами, они содержат записи для всех отдельных слов (лексем) с компактным списком мест их вхождений. При поиске нескольких слов можно найти первое, а затем воспользоваться индексом и исключить строки, в которых дополнительные слова отсутствуют. Индексы GIN хранят только слова (лексемы) из значений `tsvector`, и теряют информацию об их весах. Таким образом для выполнения запроса с весами потребуется перепроверить строки в таблице.

Индекс GiST допускает *неточности*, то есть он допускает ложные попадания и поэтому их нужно исключать дополнительно, сверяя результат с фактическими данными таблицы. (Postgres Pro делает это автоматически.) Индексы GiST являются неточными, так как все документы в них представляются сигнатурой фиксированной длины. Эта сигнатура создаётся в результате представления присутствия каждого слова как одного бита в строке из n-бит, а затем логического объединения этих битовых строк. Если двум словам будет соответствовать одна битовая позиция, попадание оказывается ложным. Если для всех слов оказались установлены соответствующие биты (в случае фактического или ложного попадания), для проверки правильности предположения о совпадении слов необходимо прочитать строку таблицы.

Секционированием данных называется разбиение одной большой логической таблицы на несколько меньших физических секций. Секционирование может принести

следующую пользу:

- В определённых ситуациях оно кардинально увеличивает быстродействие, особенно когда большой процент часто запрашиваемых строк таблицы относится к одной или лишь нескольким секциям. Секционирование по сути заменяет верхние уровни деревьев индексов, что увеличивает вероятность нахождения наиболее востребованных частей индексов в памяти.

- Когда в выборке или изменении данных задействована большая часть одной секции, производительность может возрасти, если будет выполняться последовательное сканирование этой секции, а не поиск по индексу, сопровождаемый произвольным чтением данных, разбросанных по всей таблице.

- Массовую загрузку и удаление данных можно осуществлять, добавляя и удаляя секции, если такой вариант использования был предусмотрен при проектировании секций. Удаление отдельной секции командой `DROP TABLE` и действие `ALTER TABLE DETACH PARTITION` выполняются гораздо быстрее, чем аналогичная массовая операция. Эти команды полностью исключают накладные расходы, связанные с выполнением `VACUUM` после массовой операции `DELETE`.

- Редко используемые данные можно перенести на более дешёвые и медленные носители.

Postgres Pro предлагает поддержку следующих видов секционирования:

Секционирование по диапазонам

Таблица секционируется по «диапазонам», определённым

по ключевому столбцу или набору столбцов, и не пересекающимся друг с другом. Например, можно секционировать данные по диапазонам дат или по диапазонам идентификаторов определённых бизнес-объектов. Границы каждого диапазона считаются включающими нижнее значение и исключающими верхнее. Например, если для первой секции задан диапазон значений от 1 до 10, а для второй — от 10 до 20, значение 10 относится ко второй секции, а не к первой.

Секционирование по списку

Таблица секционируется с помощью списка, явно указывающего, какие значения ключа должны относиться к каждой секции.

Секционирование по хешу

Таблица секционируется по определённым модулям и остаткам, которые указываются для каждой секции. Каждая секция содержит строки, для которых хеш-значение ключа разбиения, делённое на модуль, равняется заданному остатку.

Хранение вычисленного выражения индекса в отдельном столбце даёт ряд преимуществ. Во-первых, для использования индекса в запросах не нужно явно указывать имя конфигурации текстового поиска. Как показано в вышеприведённом примере, в этом случае запрос может зависеть от default_text_search_config. Во-вторых, поиск выполняется быстрее, так как для проверки соответствия данных индексу не нужно повторно выполнять to_tsvector. (Это актуально больше для индексов GiST, чем для GIN; см. Раздел 12.9.) С другой стороны, схему с индексом по выражению проще реализовать и она позволяет сэкономить место на диске, так как представление

tsvector не хранится явно.

Существует множество способов организации индексов:

1 В **плотных индексах** для каждого значения ключа имеется отдельная запись индекса, указывающая место размещения конкретной записи. **Неплотные** (разреженные) индексы строятся в предположении, что на каждой странице памяти хранятся записи, отсортированные по значениям индексируемого атрибута. Тогда для каждой страницы в индексе задаётся диапазон значений ключей хранимых в ней записей, и поиск записи осуществляется среди записей на указанной странице.

2 Для больших индексов актуальна проблема сжатия ключа. Наиболее распространенный метод сжатия основан на устранении избыточности хранимых данных. Последовательно идущие значения ключа обычно имеют одинаковые начальные части, поэтому в каждой записи индекса можно хранить не полное значение ключа, а лишь информацию, позволяющую восстановить его из известного предыдущего значения. Такой индекс называется **сжатым**.

3 Одноуровневый индекс представляет собой линейную совокупность значений одного или нескольких полей записи. На практике он используется редко. В развитых СУБД применяются более сложные методы организации индексов. Особенно эффективными являются **многоуровневые индексы** в виде сбалансированных деревьев (B-деревьев, balance trees).