



OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

Онлайн-образование

Не забыть включить запись!





Меня хорошо видно && слышно?

Ставьте ☐, если все хорошо
Напишите в чат, если есть проблемы





PostgreSQL. Работа с индексами, оптимизация запросов

Курочкин Константин
«Medindex»

Правила вебинара



Активно участвуем



Задаем вопрос в чат



Off-topic обсуждаем в telegram



Вопросы вижу в чате, могу ответить не сразу

Маршрут вебинара

Рассмотрим понятие индекса и explain



Рассмотрим различные варианты создания индекса



Попробуем выполнить запросы с индексами и без



Порефлекслируем 😊

Цели вебинара | После занятия вы сможете

1

Создать простой индекс, составной индекс, индекс на часть таблицы или индекс по функции

2

Пользоваться командой EXPLAIN

3

Строить индекс там, где это необходимо

Смысл | Зачем вам это уметь

1

Для написания запросов с применением индексов

2

Для правильного выбора полей для индекса

3

Для того, чтобы ускорить ваши запросы

Слайд с заданием

1 Создать простой индекс на таблицу и написать запрос, который применит этот индекс для фильтрации или сортировки данных

2 Создать составной индекс на таблицу и написать запрос, который применит этот индекс для фильтрации или сортировки данных



Оглавление

Предисловие автора	5
Введение	7
Глава 1. Введение в базы данных и SQL	13
1.1. Что такое базы данных и зачем они нужны	13
1.2. Основные понятия реляционной модели	15
1.3. Что такое язык SQL	18
1.4. Описание предметной области и учебной базы данных	19
Контрольные вопросы и задания	23
Глава 2. Создание рабочей среды	25
2.1. Установка СУБД	25
2.2. Программа psql — интерактивный терминал PostgreSQL	26
2.3. Развертывание учебной базы данных	27
Контрольные вопросы и задания	29

Об индексах



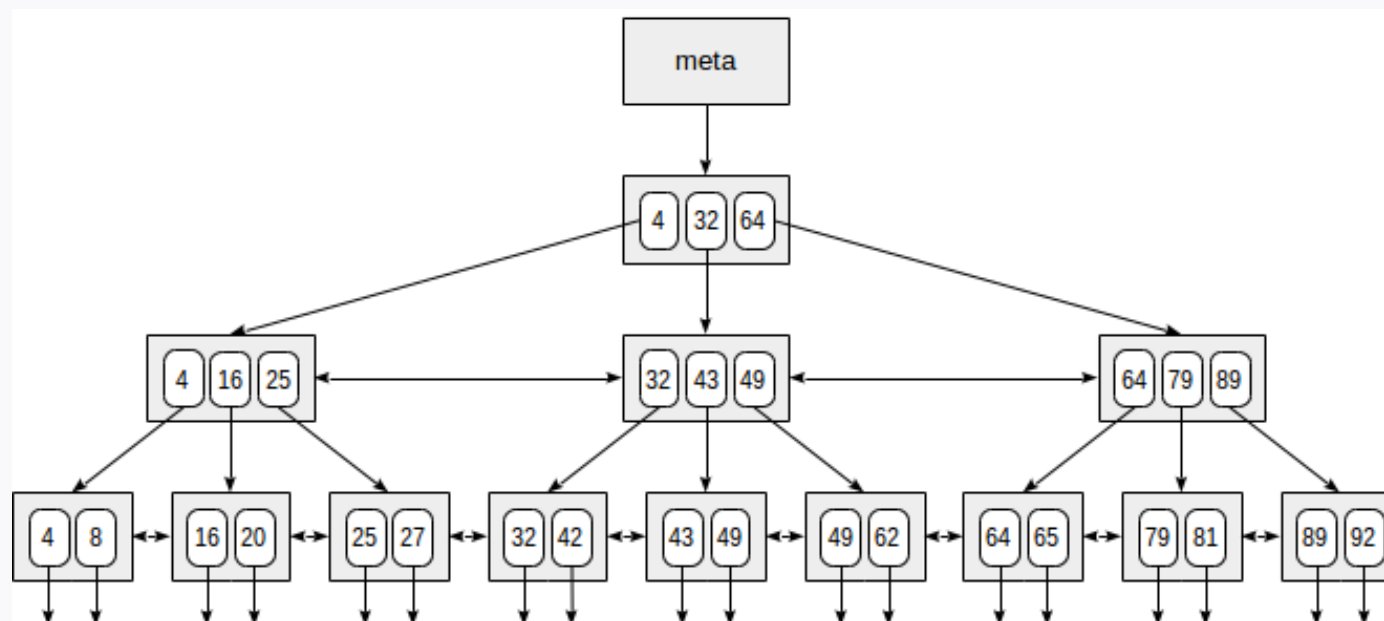
```
CREATE [ UNIQUE ] INDEX [ CONCURRENTLY ] [ [ IF NOT EXISTS ] имя ] ON [ONLY] имя_таблицы [ USING метод ]  
    ( { имя_столбца | ( выражение ) } [ COLLATE правило_сортировки ] [ класс_операторов ] [ ASC | DESC ]  
[ NULLS { FIRST | LAST } ] [, ...] )  
    [ INCLUDE ( имя столбца [,...] ) ] для Btree и Gist  
    [ NULLS [ NOT ] DISTINCT ]  
    [ WITH ( параметр_хранения = значение [, ... ] ) ]  
    [ TABLESPACE табл_пространство ]  
    [ WHERE предикат ]
```

Методы: btree, hash, gist, spgist, gin, brin

Про классы операторов - <https://postgrespro.ru/docs/postgrespro/15/indexes-opclass>

WITH fillfactor = 10..100

Btree

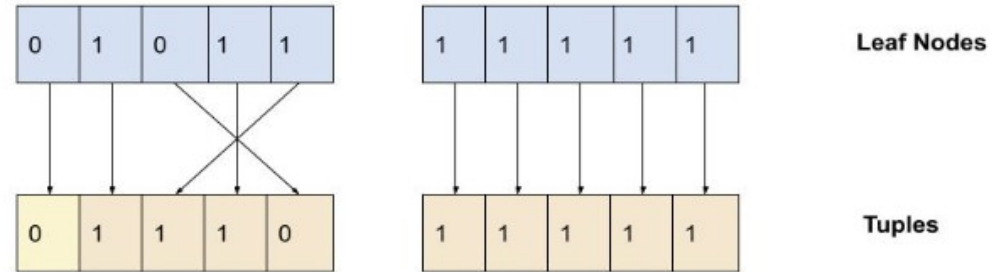


В самом начале файла находится метастраница, которая ссылается на корень индекса. Ниже корня расположены внутренние узлы; самый нижний ряд — листовые страницы. Стрелочки вниз символизируют ссылки из листовых узлов на строки таблицы (TID-ы).

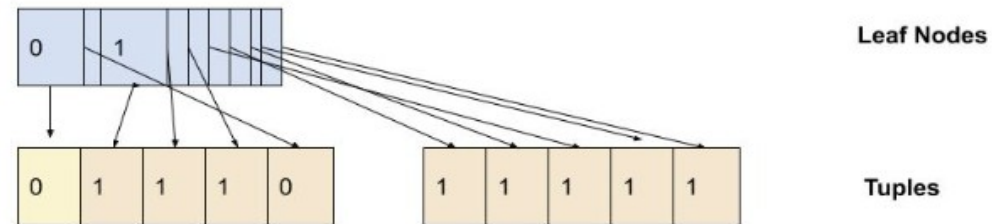
Btree



Without Deduplication



With Deduplication



Заметка начиная с версии PostgreSQL 13 - Btree может весить меньше! (и по умолчанию он это и делает)
<https://www.cybertec-postgresql.com/en/b-tree-index-deduplication/>

Gin, Gist

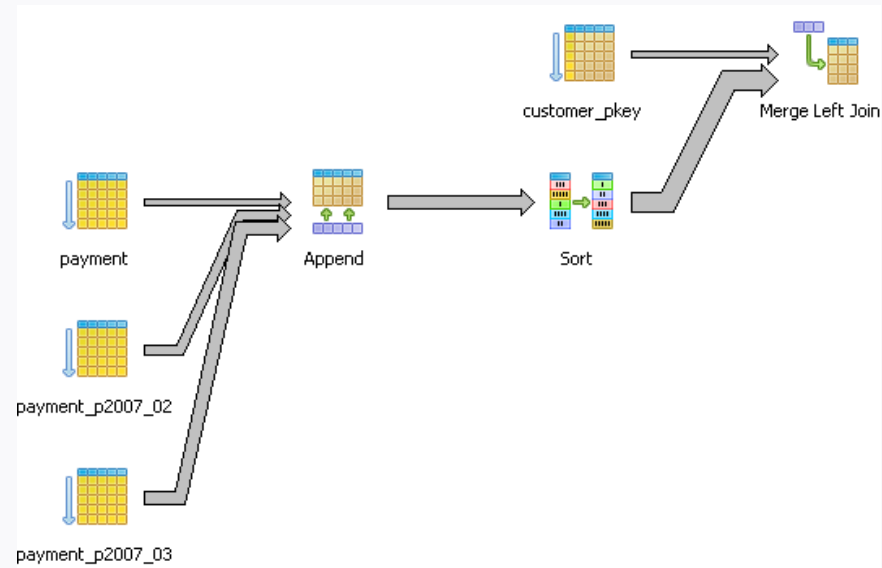


- Используются для работы с текстом (более предпочтителен Gin) и геоданными (Gist)
- Тип столбца должен быть tsvector или tsquery (в случае GIST)
- Gin похож на алфавитный указатель
- На время создания Gin индекса влияет параметр maintenance_work_mem

- <https://postgrespro.ru/docs/postgrespro/12/textsearch-indexes>
- <https://habr.com/ru/company/postgrespro/blog/333878/>
- <https://habr.com/ru/company/postgrespro/blog/340978/>

А	О
Автоформат, 8, 9	Объединение документов, 45
В	Оглавление, 21
Всплывающие подсказки, 10	Организатор стандартных блоков, 13
Вставка	Отображение
буквицы, 38	сносок, 26
видеоклипов, 41	П
маркированный список, 29	Панель инструментов
многоуровневый список, 29	колоннотитулы, 11
нумерации строк, 23	Параметры
нумерованный список, 28	Word, 8, 9
оплавления, 22	автозамены, 8, 9
раздела, 19	Подгонка страниц, 33
разрыва страницы, 18	Предварительный просмотр, 33
рисунка, 39	Р
связей, 39	Расстановка переносов, 33
символов, 24	С
сносок, 25	Сноски
специальная, 7	обычные и концевые, 25
флеш-объекта, 44	формат, 27
Выравнивание	Стили
текста, 1	выделить все вхождения, 4, 5
Г	копирование, импортирование, 5
Гиперссылка, 9	Т
Границы	
рисунка, 36	
страниц, 9	

06 Explain



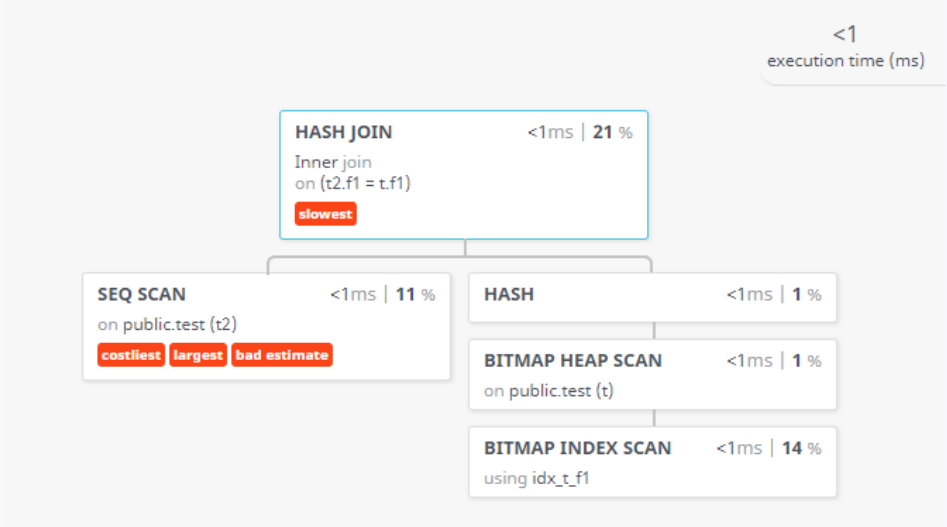
QUERY PLAN
► Sort (cost=169.51..172.01 rows=1000 width=87)
Sort Key: f.title
-> Hash Join (cost=41.93..119.68 rows=1000 width=87)
Hash Cond: (f.film_id = fc.film_id)
-> Seq Scan on film f (cost=0.00..64.00 rows=1000 width=19)
-> Hash (cost=29.43..29.43 rows=1000 width=70)
-> Hash Join (cost=1.36..29.43 rows=1000 width=70)
Hash Cond: (fc.category_id = c.category_id)
-> Seq Scan on film_category fc (cost=0.00..16.00 rows=1000 width=4)
-> Hash (cost=1.16..1.16 rows=16 width=72)
-> Seq Scan on category c (cost=0.00..1.16 rows=16 width=72)

<https://postgrespro.ru/docs/postgrespro/14/sql-explain>

06 Explain



<https://tatiyants.com/pev/#/>



<https://explain.tensor.ru/>

Explain PostgreSQL Новый план Архив Нормализовать запрос

Explain / Explain Analyze Copy&Paste or Drag'n'Drop ↓

для получения информации о распределении данных в памяти и времени работы с диском используйте
SET track_io_timing = TRUE; EXPLAIN (ANALYZE, BUFFERS) { SELECT | INSERT | UPDATE | DELETE } ...

QUERY PLAN

```
-----
Limit  (cost=0.28..2.49 rows=1 width=359) (actual time=0.008..0.009 rows=1 loops=1)
  Buffers: shared hit=3
  -> Index Scan using pg_class_oid_index on pg_class  (cost=0.28..2.49 rows=1 width=359) (actual time=0.006..0.006 rows=1 loops=1)
        Index Cond: (oid = '1259'::oid)
        Buffers: shared hit=3
Planning:
  Buffers: shared hit=148 read=13
I/O Timings: read=22.473
Planning Time: 23.242 ms
Execution Time: 0.060 ms
(10 rows)
```

<https://explain.depesz.com/>

HTML	SOURCE	TEXT	STATS	
#	exclusive	inclusive	rows x	rows loops node
1.	0.021	0.048	↓ 0.0	0 1 → Hash Join (cost=85.86..430.9 rows=3,256 width=32) (actual time=0.047..0.048 rows=0 loops=1) Buffers: shared hit=6
2.	0.011	0.011	↑ 11,424.0	1 1 → Seq Scan on test t2 (cost=0..198.24 rows=11,424 width=32) (actual time=0.011..0.011 rows=1 loops=1) Buffers: shared hit=1
3.	0.001	0.016	↓ 0.0	0 1 → Hash (cost=85.14..85.14 rows=57 width=32) (actual time=0.015..0.016 rows=0 loops=1) Buffers: shared hit=2
4.	0.001	0.015	↓ 0.0	0 1 → Bitmap Heap Scan on test t (cost=4.73..85.14 rows=57 width=32) (actual time=0.015..0.015 rows=0 loops=1) Filter: (t.f1 ~~ 'c'::text) Buffers: shared hit=2
5.	0.014	0.014	↓ 0.0	0 1 → Bitmap Index Scan on idx_t_f1 (cost=0..4.71 rows=57 width=0) (actual time=0.014..0.014 rows=0 loops=1) Index Cond: (t.f1 = 'c'::text) Buffers: shared hit=2



Любой
программист

Тупит запрос — делаем
индекс

Простой, уникальный и составной индексы



```
create index uk_test_id on test(id);
```

```
create unique index uk_test_col2 on test(col2);
```

```
create index uk_test_id_col2 on test(id, col2);
```



А в чём разница между unique constraint и unique index?



<https://www.db-fiddle.com/f/97z9hNo3ZqFCXUvN9DNAV5/0>

<https://www.db-fiddle.com/f/wiVaeAC2LRwxSPxatfxUH/1>

<https://www.db-fiddle.com/f/vRGwNsXjSRGmRyD94462n6/1>



Индекс на функцию и частичный индекс



```
create index uk_test_name on test(lower(name));
```

Можно создавать индексы на свои функции, но они должны быть IMMUTABLE

```
create index uk_test_id on test(id) where id < 100;
```




<https://www.db-fiddle.com/f/iCsikL3HGn7Wcp3FzxEfNf/1>

<https://www.db-fiddle.com/f/m3MQH8J7XrZMYJEm8zyMnc/1>



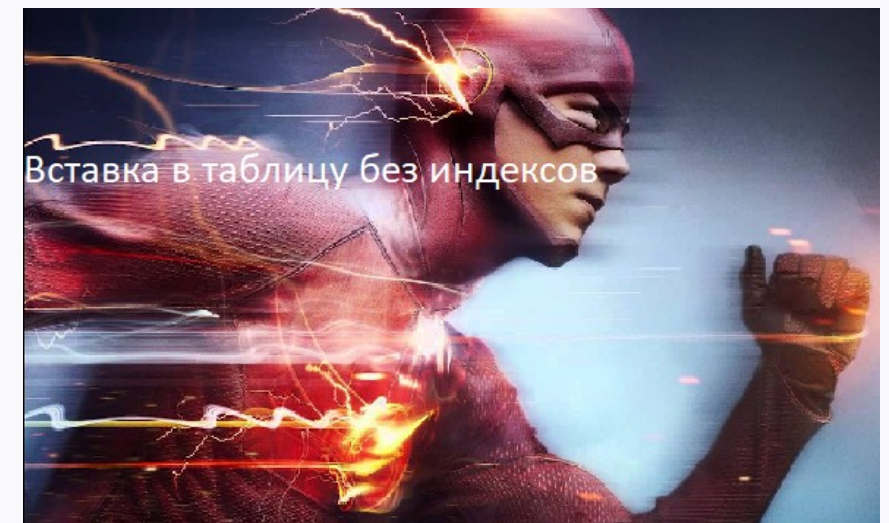
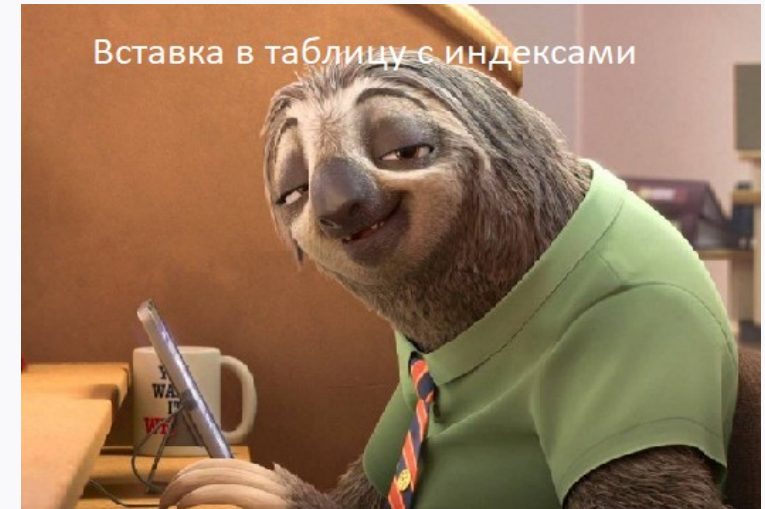


А давайте создадим индексы



на все колонки!

Недостатки индексов







- Попробуем создать индексы и посмотреть их размеры, сравнить с таблицей
- Посмотрим на вставку в таблицу без индексов и с ними
- Рассмотрим различные вариации запросов, оценим применение индексов

Подведение итогов

Рассмотрели понятие индекса



Рассмотрели примеры создания индексов



Попрактиковались



Рефлексия

Слайд с заданием

1 Создать простой индекс на таблицу и написать запрос, который применит этот индекс для фильтрации или сортировки данных

2 Создать составной индекс на таблицу и написать запрос, который применит этот индекс для фильтрации или сортировки данных


Рефлексия




Отметьте самый не раскрытый, по вашему мнению пункт



Какой индекс из рассказанных вы встречаете чаще всего?

The background of the image is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer. On the left side of this layer, there is a network diagram consisting of white dots connected by thin white lines, forming a web-like structure. The text is centered on the right side of the blue overlay.

Заполните, пожалуйста,
опрос о занятии по ссылке в чате
<https://otus.ru/polls/62016/>



До новых встреч!
Приходите на следующие занятия

Курочкин Константин
«Medindex»