Язык SQL

Лекция 6
Индексы

Е. П. Моргунов
Сибирский государственный университет науки и технологий
имени академика М. Ф. Решетнева
г. Красноярск
Институт информатики и телекоммуникаций
emorgunov@mail.ru

Компания Postgres Professional
г. Москва

Версия от 18.10.2018

Подготовка к работе (1)

На вашем компьютере уже должна быть развернута база данных demo.

- Войдите в систему как пользователь postgres:

```
su - postgres
```

- Должен быть запущен сервер баз данных PostgreSQL:

```
pg_ctl  start  -D /usr/local/pgsql/data
```

- Для проверки запуска сервера выполните команду

```
pg_ctl  status  -D /usr/local/pgsql/data
```

или

```
ps -ax | grep postgres | grep -v grep
```

- Если у вас база данных demo была модифицирована, то для ее восстановления выполните команду

```
psql -f demo_small.sql -U postgres
```
(для ОС Debian)

```
psql -f demo_small.sql
```
(для ОС Xubuntu)

Подготовка к работе (2)

- Запустите утилиту psql и подключитесь к базе данных demo

```
psql -d demo -U postgres
```
(для ОС Debian)

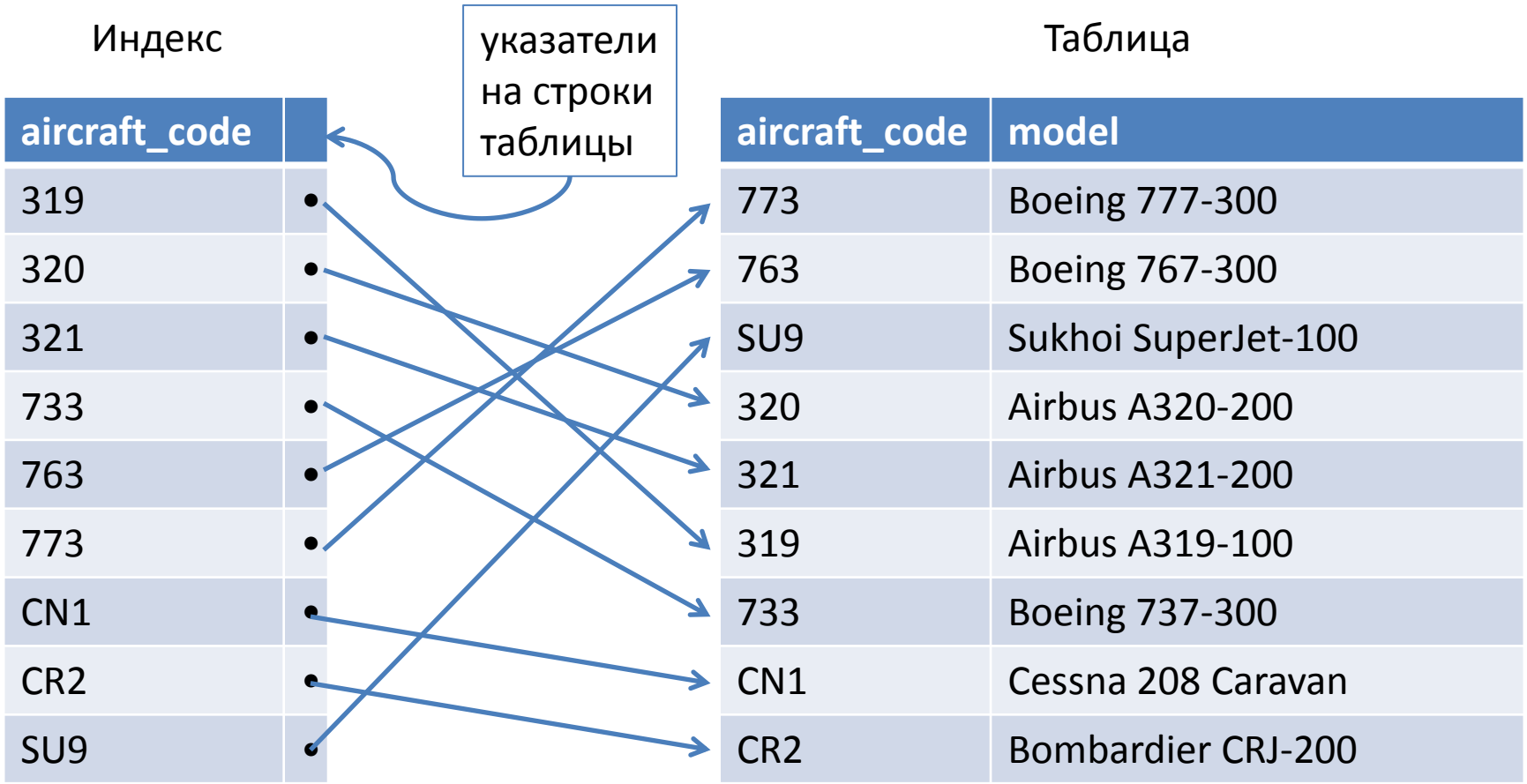
```
psql -d demo
```
(для ОС Xubuntu)

- Назначьте схему bookings в качестве текущей

```
demo=# set search_path = bookings;
```

6.1. Общая информация

Принципиальная схема индекса



- Записи в индексе *упорядочены*, а записи в таблице *не упорядочены*.
- При создании индекса значения атрибутов, участвующих в нем, могут комбинироваться и преобразовываться различными способами.

Как используется индекс при поиске строк

- Специальная подсистема СУБД (планировщик) проверяет, *имеется ли для этой таблицы индекс*, созданный на основе тех же столбцов, что указаны, например, в условии предложения WHERE.
- Если такой индекс существует, то планировщик оценивает *целесообразность* его использования в данном конкретном случае.
- Если его использование целесообразно, то *сначала* выполняется поиск необходимых значений *в индексе*, а затем, если такие значения в нем найдены, производится обращение к таблице с использованием указателей, которые хранятся в записях индекса.
- Таким образом, *полный перебор* строк в таблице может быть заменен поиском *в упорядоченном индексе* и переходом к строке таблицы по прямому указателю (ссылке).
- ВАЖНО! Индексы требуют некоторых накладных расходов на их создание и поддержание в актуальном состоянии при выполнении обновлений данных в таблицах.

Создание индекса

```
CREATE INDEX имя_индекса ON имя_таблицы ( имя_столбца, ...);
```

может быть более одного столбца

можно не указывать

Пример. Создадим индекс для таблицы «Аэропорты» (airports) по столбцу airport_name.

```
CREATE INDEX ON airports ( airport_name );
CREATE INDEX
```

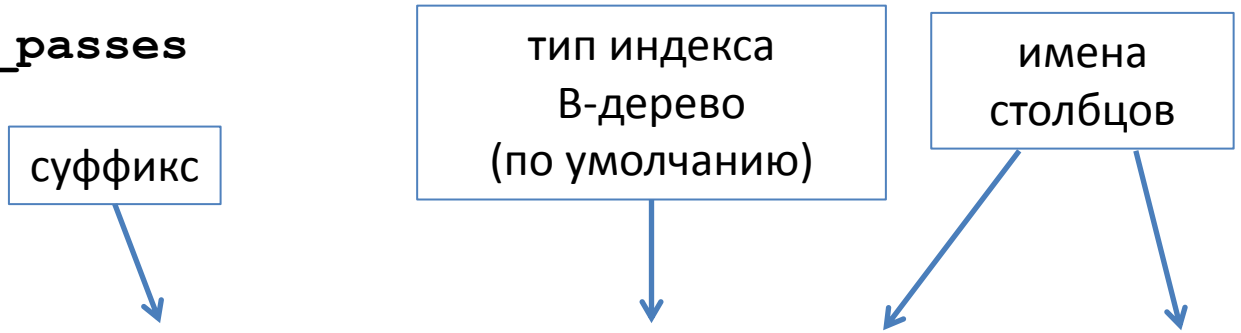
```
\d airports
...
Индексы:
...
"airports_airport_name_idx" btree (airport_name)
...
```

суффикс

- Автоматически создаются индексы для ограничений PRIMARY KEY и UNIQUE.
- Индекс позволяет выполнять проверку на дублирование очень быстро.

Как увидеть созданные индексы?

\d boarding_passes




```
...
Индексы:
"boarding_passes_pkey" PRIMARY KEY, btree (ticket_no, flight_id)
"boarding_passes_flight_id_boarding_no_key" UNIQUE CONSTRAINT,
btree (flight_id, boarding_no)
"boarding_passes_flight_id_seat_no_key" UNIQUE CONSTRAINT, btree
(flight_id, seat_no)
...
```

вид ограничения,
для которого создан индекс

Список всех индексов в текущей базе данных:
\di или **\di+**

- Обратите внимание, что эти индексы созданы не по *одному* столбцу, а по *двум* столбцам.

Проведем эксперимент: выборка без индекса

Включим в утилите psql секундомер:
\timing on

Выключить можно будет так:
\timing off

```
SELECT count( * ) FROM tickets
WHERE passenger_name = 'IVAN IVANOV';
 count
-------
   200
(1 строка)
Время: 373,232 мс
```

Эти значения времени нужно рассматривать лишь как качественные ориентиры.

Проведем эксперимент: индекс создан

```
CREATE INDEX passenger_name
ON tickets ( passenger_name );
CREATE INDEX
Время: 4023,408 мс

\d tickets
...
Индексы:
...
"passenger_name" btree (passenger_name)

SELECT count( * ) FROM tickets
WHERE passenger_name = 'IVAN IVANOV';
 count
-------
   200
(1 строка)
Время: 17,660 мс
```

имя индекса

время стало на порядок меньше

Удаление индекса

Для удаления индекса используется команда:

```
DROP INDEX имя_индекса;
```

Давайте удалим созданный нами индекс для таблицы tickets:

```
DROP INDEX passenger_name;
DROP INDEX
```

Ряд замечаний

- Когда индекс уже создан, о его поддержании в актуальном состоянии заботится СУБД.
- Конечно, следует учитывать, что это требует от СУБД затрат ресурсов и времени.
- Индекс, созданный по столбцу, участвующему в соединении двух таблиц, может позволить ускорить процесс выборки записей из таблиц.
- При выборке записей в отсортированном порядке индекс также может помочь, если сортировка выполняется по тем столбцам, по которым индекс создан.

6.2. Индексы и сортировка

Селективность индекса

- Если в SQL-запросе есть предложение ORDER BY, то индекс может позволить избежать этапа сортировки выбранных строк.
- Однако если SQL-запрос просматривает значительную часть таблицы, то *явная сортировка выбранных строк может оказаться быстрее*, чем использование индекса.
- Создавая индексы с целью ускорения доступа к данным, нужно учитывать предполагаемую долю строк таблицы (**селективность**), выбираемых при выполнении типичных запросов, в которых создаваемый индекс будет использоваться.
- Если эта *доля велика* (т. е. *селективность — низкая*), тогда наличие индекса может не дать ожидаемого эффекта.
- Индексы более полезны, когда из таблицы выбирается лишь небольшая доля строк, т. е. при высокой селективности выборки.

Эксперимент с LIMIT и ORDER BY

Если для таблицы «Билеты» (tickets) еще не создан индекс по столбцу book_ref, то создайте его:

```
CREATE INDEX tickets_book_ref_test_key
ON tickets ( book_ref );
CREATE INDEX
SELECT * FROM tickets ORDER BY book_ref LIMIT 5;
...
Время: 0,442 мс
```

Удалите этот индекс и повторите запрос. Время его выполнения увеличится, вероятно, на два порядка. Почему?

Порядок сортировки и значения NULL

- При создании индексов может использоваться не только возрастающий порядок значений в индексируемом столбце, но также и убывающий.
- По умолчанию порядок возрастающий (ASC), при этом значения NULL, которые также могут присутствовать в индексируемых столбцах, идут последними.

```
CREATE INDEX имя_индекса ON имя_таблицы
( имя_столбца NULLS FIRST, ... );
```

значения NULL идут первыми

```
CREATE INDEX имя_индекса ON имя_таблицы
( имя_столбца DESC NULLS LAST, ... );
```

убывающий порядок

значения NULL идут последними

Для разных столбцов можно указать разные порядки сортировки.

6.3. Уникальные индексы

Создание уникальных индексов

Индексы могут также использоваться для обеспечения уникальности значений атрибутов в строках таблицы. В таком случае создается уникальный индекс.

```
CREATE UNIQUE INDEX имя_индекса
ON имя_таблицы ( имя_столбца, ...);
```

```
CREATE UNIQUE INDEX aircrafts_unique_model_key
ON aircrafts ( model );
```

ПРИМЕЧАНИЕ. Мы могли при создании таблицы задать ограничение уникальности для столбца model, и тогда уникальный индекс был бы создан автоматически.

ВАЖНО! В уникальных индексах допускается наличие значений NULL, поскольку они считаются не совпадающими ни с какими другими значениями, в том числе и друг с другом.

Если уникальный индекс создан по нескольким атрибутам, то совпадающими считаются лишь те комбинации значений атрибутов в двух строках, в которых совпадают *значения всех соответственных атрибутов*.

6.4. Индексы на основе выражений

Использование функции при создании индекса

В команде создания индекса можно использовать не только имена столбцов, но также функции и скалярные выражения, построенные на основе столбцов таблицы.

Пример. Запретим значения столбца model в таблице aircrafts, отличающиеся только регистром символов.

```
CREATE UNIQUE INDEX aircrafts_unique_model_key
ON aircrafts ( lower( model ) );

INSERT INTO aircrafts
VALUES ( '123', 'Cessna 208 CARAVAN', 1300);
ОШИБКА: повторяющееся значение ключа нарушает ограничение уникальности,  "aircrafts_unique_model_key"
ПОДРОБНОСТИ: Ключ "(lower(model))=(cessna 208 caravan)" уже существует.
```

перевод в нижний регистр

такое значение хранится в индексе

Индекс строится уже на основе преобразованных значений, поэтому при поиске строки в таблице искомое значение *сначала переводится в нижний регистр*, а затем осуществляется поиск в индексе.

Ряд замечаний

- Индексы на основе выражений требуют *больше ресурсов* для их создания и поддержания при вставке и обновлении строк в таблице.
- Зато при выполнении выборок, построенных на основе сложных выражений, работа происходит с *меньшими накладными расходами*, поскольку в индексе хранятся *уже вычисленные значения* этих выражений, пусть даже самых сложных.
- Поэтому такие индексы целесообразно использовать тогда, когда *выборки производятся многократно*, и время, затраченное на создание и поддержание индекса, компенсируется (окупается) при выполнении выборок из таблицы.

6.5. Частичные индексы

Правдоподобная ситуация

Частичный индекс формируется не для всех строк таблицы, а лишь для их *подмножества*. Это достигается с помощью использования условного выражения, называемого **предикатом индекса**. Предикат вводится с помощью предложения WHERE.

Пример. Представим, что руководство компании интересуют бронирования на сумму свыше одного миллиона рублей.

```
SELECT * FROM bookings
WHERE total_amount > 1000000
ORDER BY book_date DESC;
```

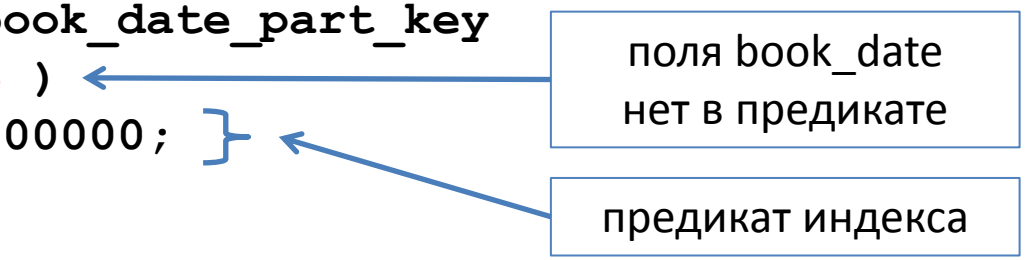
предикат индекса

убывающий порядок

```
 book_ref |         book_date          | total_amount
----------+----------------------------+--------------
 D7E9AA   | 2016-10-06 09:29:00+08 |  1062800.00
 EF479E   | 2016-09-30 19:58:00+08 |  1035100.00
 3AC131   | 2016-09-28 05:06:00+08 |  1087100.00
 3B54BB   | 2016-09-02 21:08:00+08 |  1204500.00
 65A6EA   | 2016-08-31 10:28:00+08 |  1065600.00
(5 строк)
Время: 90,996 мс
```

Поможет ли частичный индекс?

```
CREATE INDEX bookings_book_date_part_key
ON bookings ( book_date )
WHERE total_amount > 1000000;
CREATE INDEX
```



- Хотя сортировка строк производится по датам бронирования в *убывающем* порядке, т. е. от более поздних дат к более ранним, тем не менее, включать ключевое слово *DESC* в индексное выражение, когда индекс создается только по одному столбцу, нет необходимости.
- Это объясняется тем, что PostgreSQL умеет совершать обход индекса как по *возрастанию*, так и по *убыванию* с одинаковой эффективностью.
- Обратите внимание, что индексируемый столбец book_date не участвует в формировании предиката индекса — в предикате используется столбец total_amount. Это вполне допустимая ситуация.

Повторим вышеприведенный запрос. Теперь он выдаст результат за время, *на порядок меньшее*, чем без использования частичного индекса.

Ряд замечаний

- Для того чтобы СУБД использовала частичный индекс, необходимо чтобы условие, записанное в запросе в предложении WHERE, соответствовало предикату индекса.
- Это означает, что либо условие должно быть точно таким же, как использованное в предикате частичного индекса при его создании, либо условие запроса должно математически сводиться к предикату индекса, а система должна суметь это понять.

Пример. В таком запросе ранее созданный индекс будет использоваться:

```
SELECT * FROM bookings WHERE total_amount > 1100000 ...
```

А в таком не будет:

подмножество строк

```
SELECT * FROM bookings WHERE total_amount > 900000 ...
```

- В большинстве случаев преимущества частичных индексов по сравнению с обычными индексами будут *минимальными*. Однако размер частичного индекса будет *меньше*, чем размер обычного. Для получения заметного полезного эффекта от их применения необходим опыт и понимание того, как работают индексы в PostgreSQL.

Литература

1. Лузанов, П. PostgreSQL для начинающих / П. Лузанов, Е. Рогов, И. Лёвшин ; Postgres Professional. – М., 2017. – 146 с.
2. Моргунов, Е. П. Язык SQL. Базовый курс : учеб.-практ. пособие. / Е. П. Моргунов ; под ред. Е. В. Рогова, П. В. Лузанова ; Postgres Professional. – М., 2017. – 257 с.
3. PostgreSQL [Электронный ресурс] : официальный сайт / The PostgreSQL Global Development Group. – http://www.postgresql.org.
4. Postgres Professional [Электронный ресурс] : российский производитель СУБД Postgres Pro : официальный сайт / Postgres Professional. – http://postgrespro.ru.

Задание

Для выполнения практических заданий необходимо использовать книгу:

Моргунов, Е. П. Язык SQL. Базовый курс : учеб.-практ. пособие / Под ред. Е. В. Рогова, П. В. Лузанова ; Postgres Professional. – М., 2017. – 257 с. https://postgrespro.ru/education/books/sqlprimer

1. Изучить материал главы 8. Запросы к базе данных выполнять с помощью утилиты psql, описанной в главе 2, параграф 2.2.