Авторы: Егор Рогов, Павел Лузанов

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Темы

Соединение слиянием

Использование памяти при сортировке

Запросы с сортировкой

Группировка с помощью сортировки

Соединение слиянием

Сортировка

Слияние

Вычислительная сложность

Параллельное соединение

Сортировка

```
SELECT a.title, s.name
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

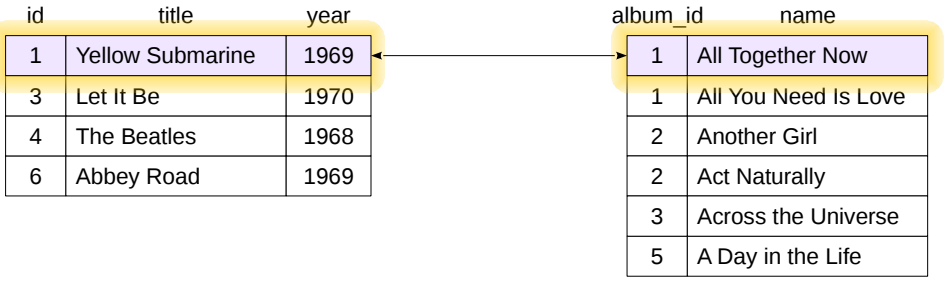
либо сортировка в оперативной памяти (*work_mem*),
либо получение отсортированных данных индексным сканированием

Третий, и последний, способ соединения — соединение слиянием.

Подготовительным этапом для него служит сортировка обоих наборов строк. Сортировка — довольно дорогая операция: $N \log N$, если сортировка выполняется в памяти. Но иногда этого этапа удается избежать, если можно сразу получить отсортированные наборы строк (например, за счет индексного доступа к таблице).

Слияние

```
SELECT a.title, s.name
FROM albums a JOIN songs s ON a.id = s.a_id;
```



id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

результат соединения автоматически отсортирован

Само слияние устроено просто. Сначала берем первые строки обоих наборов и сравниваем. В данном случае мы сразу нашли соответствие и можем вернуть первую строку результата: («Yellow Submarine», «All Together Now»).

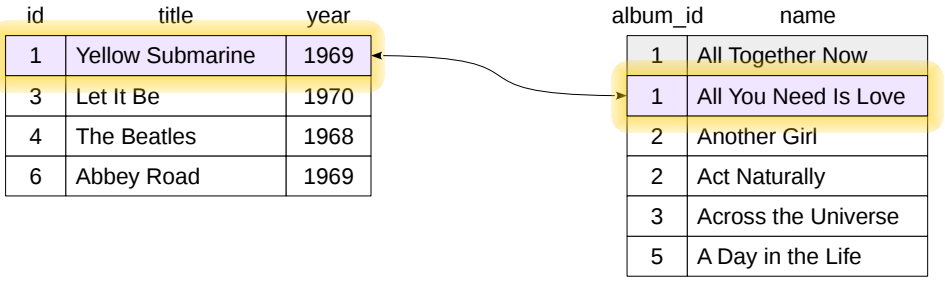
Общий алгоритм таков: читаем следующую строку того набора, для которого значение поля, по которому происходит соединение, меньше (один набор «догоняет» другой). Если же значения одинаковы, как в нашем примере, то читаем следующую строку второго набора.

(На самом деле алгоритм сложнее — что, если и в первом наборе строк может быть несколько одинаковых значений? Но не будем загромождать картину деталями. Псевдокод алгоритма можно посмотреть в файле src/backend/executor/nodeMergejoin.c.)

Важно, что алгоритм слияния возвращает результат соединения в отсортированном виде. В частности, полученный набор строк можно использовать для следующего соединения слиянием без дополнтельной сортировки.

Слияние

```
SELECT a.title, s.name
FROM albums a JOIN songs s ON a.id = s.a_id;
```



id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

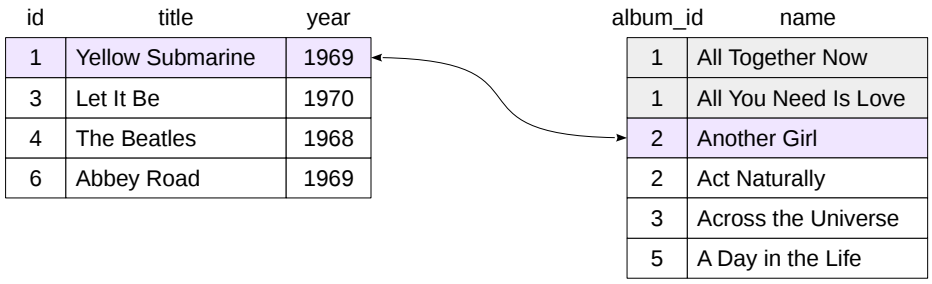
album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

Вновь соответствие: («Yellow Submarine», «All You Need Is Love»).

Снова читаем следующую строку второго набора.

Слияние

```
SELECT a.title, s.name
FROM albums a JOIN songs s ON a.id = s.a_id;
```



id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

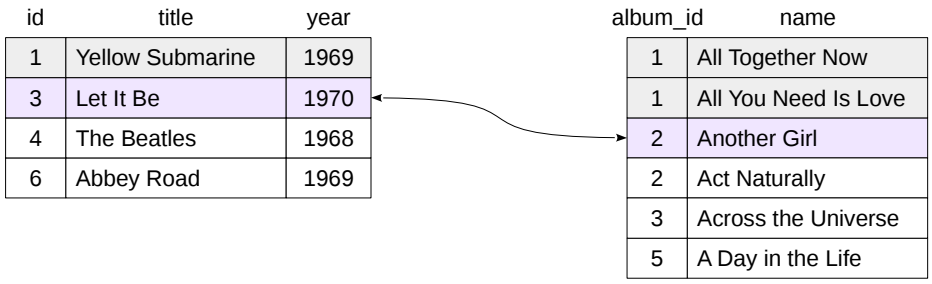
album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

В данном случае соответствия нет.

Поскольку 1 < 2, читаем следующую строку первого набора.

Слияние

```
SELECT a.title, s.name
FROM albums a JOIN songs s ON a.id = s.a_id;
```



id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

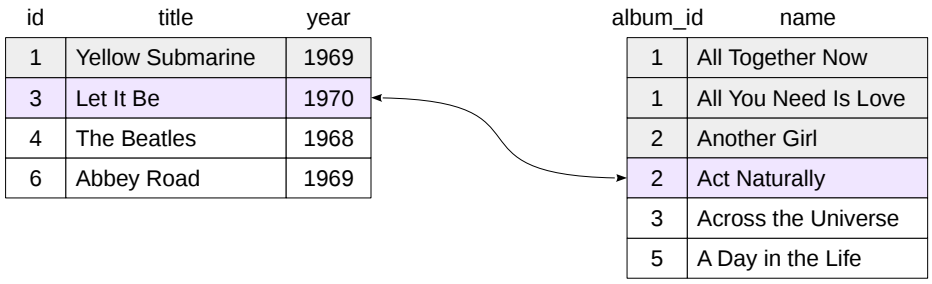
album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

Соответствия нет.

3 > 2, поэтому читаем следующую строку второго набора.

Слияние

```
SELECT a.title, s.name
FROM albums a JOIN songs s ON a.id = s.a_id;
```



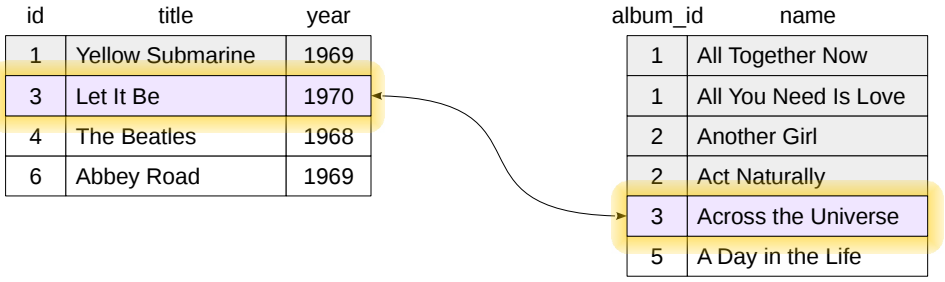
id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

Снова нет соответствия, снова 3 > 2, снова читаем строку второго набора.

Слияние

```
SELECT a.title, s.name
FROM albums a JOIN songs s ON a.id = s.a_id;
```



id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

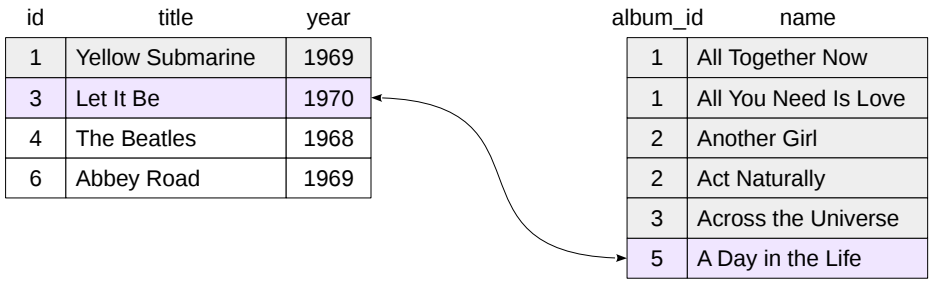
album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

Есть соответствие: («Let It Be», «Across the Universe»).

3 = 3, читаем следующую строку второго набора.

Слияние

```
SELECT a.title, s.name
FROM albums a JOIN songs s ON a.id = s.a_id;
```



id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

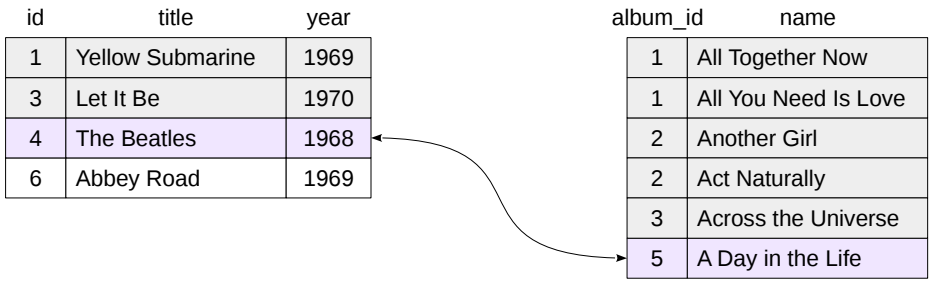
album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

Соответствия нет.

3 < 5, читаем строку первого набора.

Слияние

```
SELECT a.title, s.name
FROM albums a JOIN songs s ON a.id = s.a_id;
```



id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

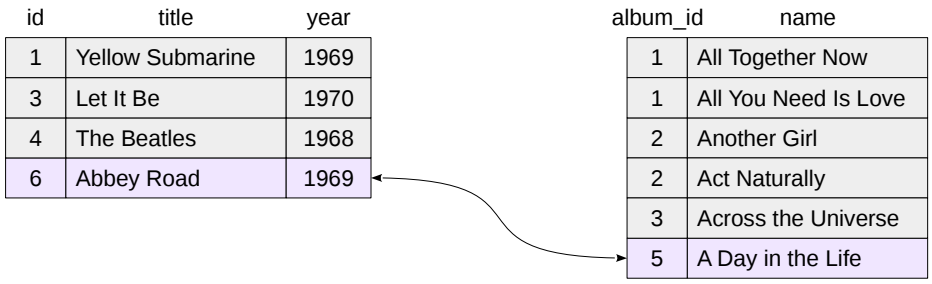
album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

Соответствия нет.

4 < 5, читаем строку первого набора.

Слияние

```
SELECT a.title, s.name
FROM albums a JOIN songs s ON a.id = s.a_id;
```



id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

И последний шаг: снова нет соответствия.

На этом соединение слиянием окончено.

Вычислительная сложность

$\sim N + M,$

где N и M — число строк в первом и втором наборах данных, если не требуется сортировка

$\sim N \log N + M \log M,$

если сортировка нужна

Возможные начальные затраты на сортировку

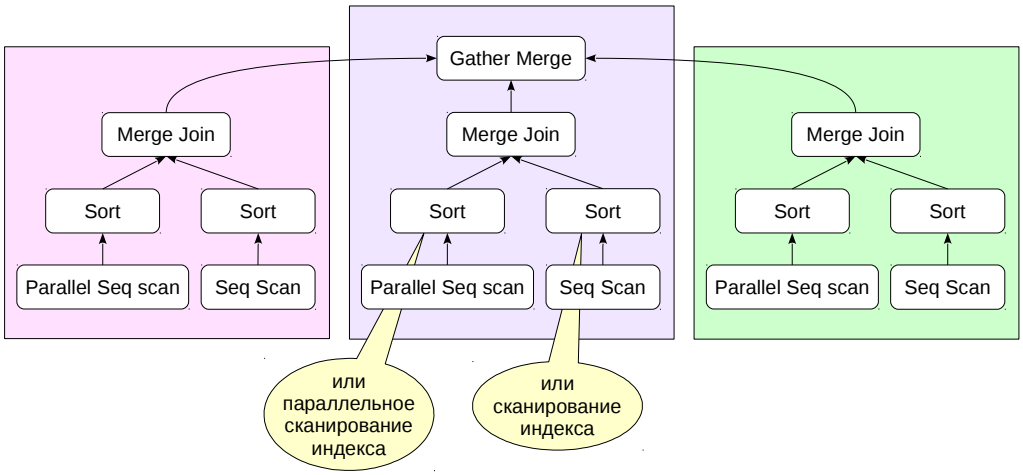
Эффективно для большого числа строк

В случае, когда не требуется сортировать данные, общая сложность соединения слиянием пропорциональна сумме числа строк в обоих наборах данных. Но, в отличие от соединения хешированием, здесь не требуются накладные расходы на построение хеш-таблицы.

Поэтому соединение слиянием может успешно применяться как в OLTP-, так и в OLAP-запросах.

Однако если сортировка требуется, то стоимость становится пропорциональной произведению числа строк на логарифм числа строк, и на больших объемах данных это может работать хуже соединения хешированием.

В параллельном режиме



один набор строк читается параллельно,
другой — последовательно каждым рабочим процессом

Соединение слиянием может выполняться в «почти параллельном» режиме начиная с PostgreSQL версии 10.

Так же, как и при соединении хешированием, сканирование одного набора строк выполняется рабочими процессами параллельно, но другой набор строк каждый рабочий процесс читает полностью самостоятельно.

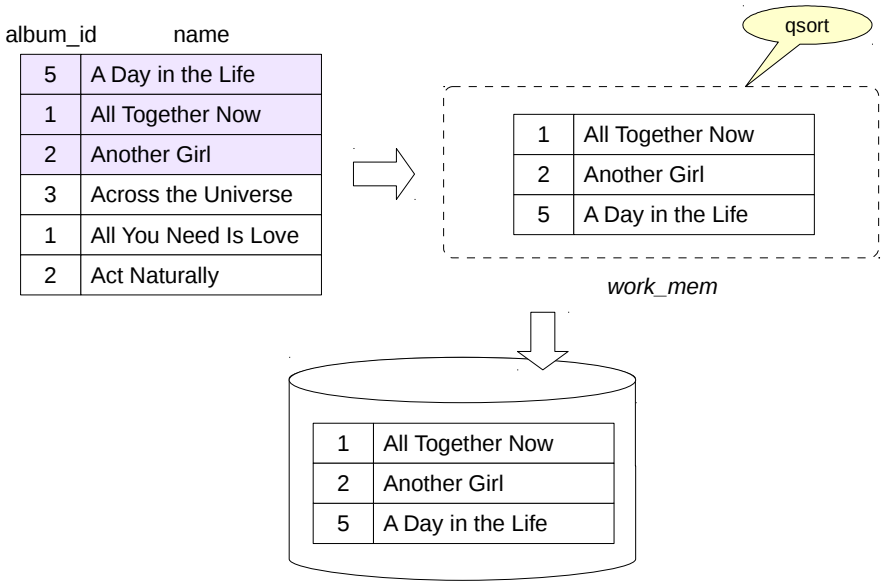
Поскольку каждый рабочий процесс выдает свою часть данных в отсортированном виде, то и объединенные данные имеет смысл выстроить в общем порядке. Для этого вместо узла Gather используется узел Gather Merge, сохраняющий порядок сортировки.

Использование памяти

Использование памяти при сортировке

Построение индекса (B-дерева)

Внешняя сортировка

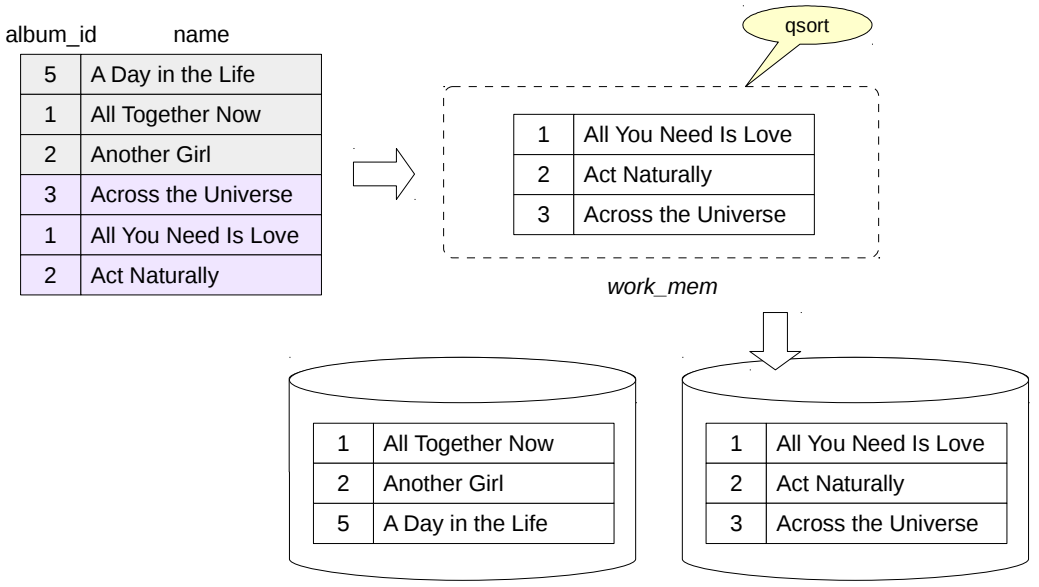


В идеальном случае набор строк, подлежащий сортировке, целиком помещается в память, ограниченную параметром *work_mem.* В этом случае все строки просто сортируются (используется алгоритм быстрой сортировки qsort) и возвращается результат.

Если набор строк велик, он не поместится в память целиком. В таком случае используется алгоритм внешней сортировки.

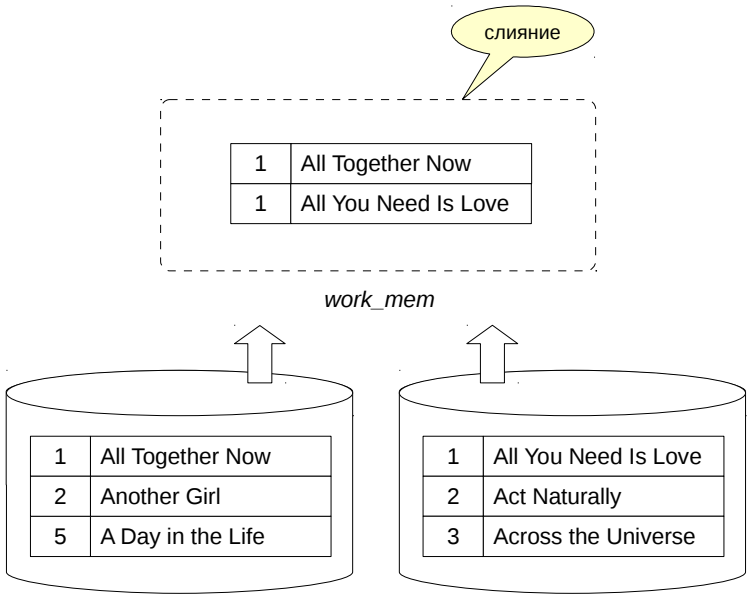
Набор строк читается в память, пока есть возможность, затем сортируется и записывается во временный файл.

Внешняя сортировка



Эта процедура повторяется столько раз, сколько необходимо, чтобы записать все данные в файлы, каждый из которых по отдельности отсортирован.

Внешняя сортировка



Далее несколько файлов сливаются аналогично тому, как работает соединение слиянием. Основное отличие состоит в том, что сливаться могут более двух файлов одновременно.

Для слияния не требуется много места в памяти. Достаточно разместить по одной строке из каждого файла (как в примере на слайде). Среди этих строк выбирается минимальная (максимальная) и возвращается как часть результата, а на ее место читается новая строка из того же файла.

На практике строки читаются не по одной, а порциями, чтобы оптимизировать ввод-вывод.

Если оперативной памяти недостаточно, чтобы слить сразу все файлы, процесс повторяется многократно: сливаются по несколько файлов за раз и результаты записываются в новые временные файлы, затем происходит слияние этих новых файлов и так далее.

На самом деле, конечно, сортировка устроена сложней. С деталями реализации можно познакомиться в файле src/backend/utils/sort/tuplesort.c.

История развития способов сортировки в PostgreSQL хорошо описана в презентации Грегори Старка «Sorting Through The Ages».

Создание индекса

Используется сортировка

сначала все строки сортируются

затем строки собираются в листовые индексные страницы

ссылки на них собираются в страницы следующего уровня

и так далее, пока не дойдем до корня

Ограничение

maintenance_work_mem, так как операция не частая

для уникальных индексов требуется дополнительно *work_mem*

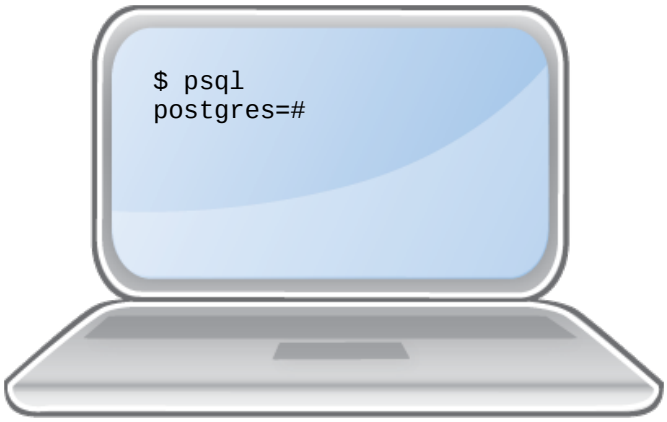
Индекс (речь идет про B-дерево) можно строить, добавляя последовательно в пустой индекс по одной строке из таблицы. Но такой способ крайне неэффективен.

Поэтому при создания индекса используется сортировка: все строки таблицы сортируются и раскладываются по листовым индексным страницам. Затем достраиваются верхние уровни дерева, состоящие из ссылок на элементы страниц нижележащего уровня, до тех пор, пока на очередном уровне не получится одна страница — она и будет корнем дерева.

Сортировка устроена точно так же, как рассматривалось выше. Однако размер памяти ограничен не *work_mem,* а *maintenance_work_mem,* поскольку операция создания индекса не слишком частая и имеет смысл выделить для нее больше памяти.

Для создания уникальных индексов требуется дополнительная память размером *work_mem.* Она нужна для хранения неактуальных версий строк, чтобы исключить их из проверки на уникальность.

Демонстрация



Итоги

Соединение слиянием может потребовать подготовки

- надо отсортировать наборы строк
- или получить их заранее отсортированными

Эффективно для больших выборок

- хорошо, если наборы уже отсортированы
- хорошо, если нужен отсортированный результат

Не зависит от порядка соединения

Поддерживает только эквисоединения

- другие не реализованы, но принципиальных ограничений нет

Чтобы начать соединение слиянием, оба набора строк должны быть отсортированы. Хорошо, если удается получить данные уже в нужном порядке; если нет — требуется выполнить сортировку.

Само слияние выполняется очень эффективно даже для больших наборов строк. В качестве приятного бонуса результирующая выборка тоже упорядочена, поэтому такой способ соединения привлекателен, если вышестоящим узлам плана также требуется сортировка (например, запрос с фразой ORDER BY или еще одна сортировка слиянием).

В настоящее время соединение слиянием поддерживает только эквисоединения, соединения по операциям «больше» или «меньше» не реализованы (однако патч уже предложен).

Итак, в распоряжении планировщика есть три способа соединения: вложенный цикл, хеширование и слияние (не считая различных модификаций). Есть ситуации, в которых каждый из способов оказывается более эффективным, чем остальные. Это позволяет планировщику выбрать именно тот способ, который — как предполагается — лучше подойдет в каждом конкретном случае. А точность предположений напрямую зависит от имеющейся статистики.

Практика

1. Создайте индекс по столбцам passenger_name и passenger id таблицы билетов (tickets).
   Потребовался ли временный файл для выполнения этой операции?
2. Проверьте план выполнения запроса из демонстрации, показывающего все места в салонах в порядке кодов самолетов, но оформленного в виде курсора.
   Уменьшите значение параметра *cursor_tuple_fraction* в десять раз. Как при этом изменился план выполнения?