Авторы: Алексей Береснев, Илья Баштанов, Павел Толмачев

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

Темы

Предназначение AQO

Принцип работы AQO

Режимы работы

Предназначение AQO

Исправление неверных оценок кардинальности,
когда штатные средства не помогают

AQO улучшает оценки в запросах с предикатами вида

атрибут = константа

атрибут > константа

Основная сфера применения — сложные аналитические запросы

Модуль AQO (Adaptive Query Optimization - Адаптивная Оптимизация Запросов) можно применить в тех случаях, когда стандартные инструменты улучшения оценок селективности и кардинальности (подробно описанные в курсе QPT) не срабатывают.

Модуль AQO работает с простыми условиями вида «атрибут = константа» или «атрибут > константа».

AQO улучшает оценку количества строк, что может способствовать выбору лучшего плана и, следовательно, ускорению запросов. Основная сфера применения AQO — сложные аналитические запросы с повторяющейся структурой, для которых EXPLAIN ANALYZE показывает неверную оценку кардинальности.

Пример запроса с неверной оценкой.

Начнем с установки расширения.

```
=> ALTER SYSTEM SET shared_preload_libraries = 'aqo';
```

```
ALTER SYSTEM
```

Необходима перезагрузка экземпляра.

```
student$ sudo systemctl restart postgrespro-ent-13.service
```

```
student$ psql demo
```

Добавим расширение AQO в базу данных:

```
=> CREATE EXTENSION aqo;
```

```
CREATE EXTENSION
```

В качестве примера мы будем использовать запрос с ошибкой вычисления кардинальности, вызванной коррелированными предикатами.

```
=> EXPLAIN SELECT *
FROM flights fl
  JOIN airports ap ON ( fl.departure_airport = ap.airport_code )
WHERE ap.airport_name = 'Внуково'
  AND fl.flight_no = 'PG0007';
```

```
                                                  QUERY PLAN
---------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=4.79..207.21 rows=1 width=162)
   Join Filter: (fl.departure_airport = ml.airport_code)
   ->  Seq Scan on airports_data ml  (cost=0.00..30.56 rows=1 width=145)
         Filter: ((airport_name ->> lang()) = 'Внуково'::text)
   ->  Bitmap Heap Scan on flights fl  (cost=4.79..175.34 rows=64 width=63)
         Recheck Cond: (flight_no = 'PG0007'::bpchar)
         ->  Bitmap Index Scan on flights_flight_no_scheduled_departure_key  (cost=0.00..4.77 rows=64 width=0)
               Index Cond: (flight_no = 'PG0007'::bpchar)
(8 rows)
```

Точное количество строк отличается от оценки:

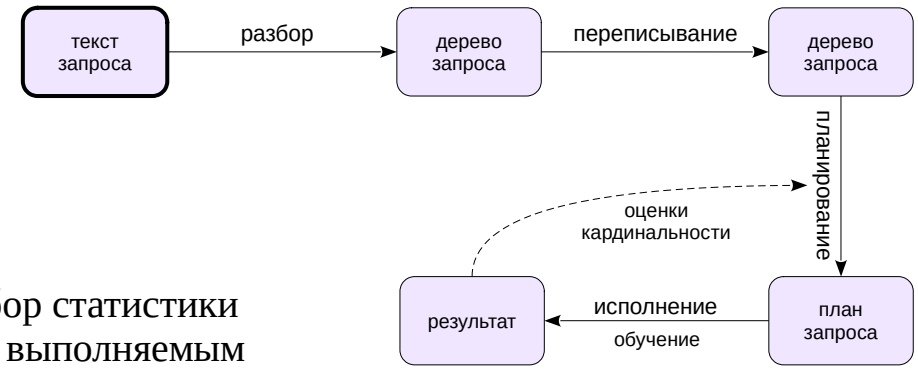
```
=> SELECT count(*)
FROM flights fl
  JOIN airports ap ON ( fl.departure_airport = ap.airport_code )
WHERE ap.airport_name = 'Внуково'
  AND fl.flight_no = 'PG0007';
```

```
 count
-------
    61
(1 row)
```

Причина неверной оценки в том, что рейс PG0007 всегда вылетает из Внуково. Но планировщику это не известно.

В курсе QPT продемонстрировано использование расширенной статистики по функциональным зависимостям, исправляющей неверную оценку кардинальности. В этой теме для исправления неверной оценки будет использован модуль AQO.

Предназначение AQO



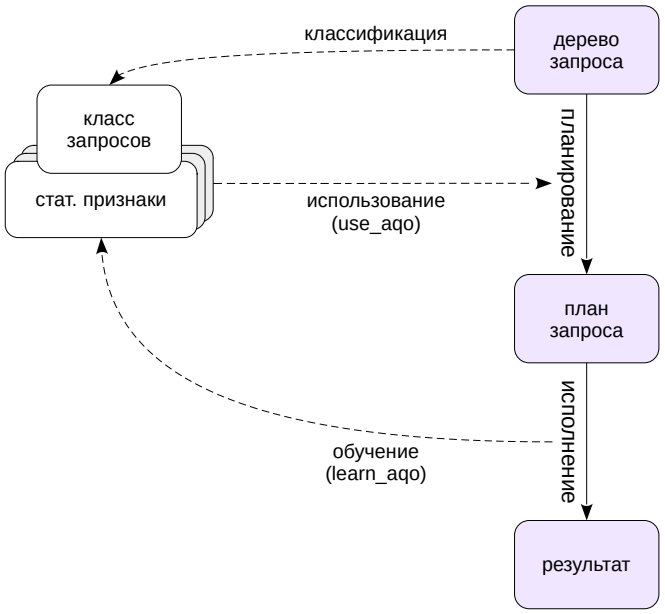
Сбор статистики по выполняемым запросам и улучшение оценок количества строк с помощью машинного обучения

Лучшие оценки — основа более оптимальных планов

Модуль AQO собирает статистику по выполненным запросам для обучения модели, с помощью которой AQO предоставляет уточненные оценки. Именно это концептуально отличает AQO от обычного планировщика, на работу которого никак не влияют ранее выполненные запросы.

Для оценки кардинальности используется метод машинного обучения на базе алгоритма поиска ближайших соседей k-NN вместо обычной статистики планировщика.

Принцип работы AQO



6

Выполняемый запрос должен быть отнесен к одному из классов, то есть *классифицирован*. В один класс попадают все запросы с одинаковыми деревьями разбора, отличающимися только константами.

При исполнении плана запроса AQO получает статистическую информацию (признаки, features) о реальной кардинальности в разных узлах плана. Эта информация сохраняется в привязке к соответствующему классу запроса — происходит *обучение* модели.

Данные обучения сохраняются в файлах и при запуске AQO загружаются в разделяемую память.

Если запросы некоторого класса уже выполнялись и по ним накоплена статистическая информация, модель *используется* для уточнения оценки планировщика.

Обучением и использованием можно управлять отдельно для каждого класса запросов. Например, сначала можно обучать модель для какого-то проблемного запроса, но не использовать ее оценки (которые могут поначалу оказаться неточными), а после успешного обучения — начать использовать, но перестать обучать.

Но если детализация до отдельных классов не требуется, AQO можно управлять, просто устанавливая один из режимов, которые будут рассмотрены ниже.

https://postgrespro.ru/docs/enterprise/13/aqo

Режимы работы

Режим learn

Режим controlled

Режим frozen

Режим intelligent

Режим forced

Режим disabled

Режим learn

Начальное обучение AQO на всех запросах

Обучение и использование оценок для всех классов

```
learn_aqo = true
use_aqo = true
```


В режиме learn для всех классов запросов выполняется как обучение, так и использование уточненных оценок. Статистическая информация сохраняется отдельно для каждого класса запросов.

Этот режим можно использовать для начального обучения AQO на всех запросах, но следует иметь в виду, что обычно этого не требуется — большая часть запросов планируется правильно и без AQO, а в некоторых случаях применение AQO может и ухудшить производительность.

Режим learn

Переключим режим работы AQO:

```
=> SET aqo.mode = 'learn';
```

```
SET
```

Для удобства включим параметры детализации и вывода хешей запросов:

```
=> SET aqo.show_hash = on;
```

```
SET
```

```
=> SET aqo.show_details = on;
```

```
SET
```

Уменьшим значение параметра aqo.join_threshold (значение по умолчанию — 3). AQO игнорирует запросы, содержащие меньшее количество соединений.

```
=> SET aqo.join_threshold = 1;
```

```
SET
```

Выполним показанный ранее запрос:

```
=> EXPLAIN ANALYZE SELECT *
FROM flights fl
  JOIN airports ap ON ( fl.departure_airport = ap.airport_code )
WHERE ap.airport_name = 'Внуково'
  AND fl.flight_no = 'PG0007';
```

```
                                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=4.79..207.21 rows=1 width=162) (actual time=0.062..0.229 rows=61 loops=1)
   AQO not used, fss=-461720685
   Join Filter: (fl.departure_airport = ml.airport_code)
   ->  Seq Scan on airports_data ml  (cost=0.00..30.56 rows=1 width=145) (actual time=0.040..0.105 rows=1 loops=1)
         AQO not used, fss=1184327055
         Filter: ((airport_name ->> lang()) = 'Внуково'::text)
         Rows Removed by Filter: 103
   ->  Bitmap Heap Scan on flights fl  (cost=4.79..175.34 rows=64 width=63) (actual time=0.017..0.022 rows=61 loops=1)
         AQO not used, fss=-1476077258
         Recheck Cond: (flight_no = 'PG0007'::bpchar)
         Heap Blocks: exact=2
         ->  Bitmap Index Scan on flights_flight_no_scheduled_departure_key  (cost=0.00..4.77 rows=64 width=0) (actual time=0.013..0.013 rows=61 loops=1)
               Index Cond: (flight_no = 'PG0007'::bpchar)
 Planning Time: 0.588 ms
 Execution Time: 0.320 ms
 Using aqo: true
 AQO mode: LEARN
 Query hash: 1598258635823758032
 JOINS: 1
(19 rows)
```

Для AQO этот запрос является новым, поэтому для планирования использовались обычные оценки. Но AQO уже получил данные для обучения модели:

```
=> SELECT * FROM aqo_query_texts WHERE query_text ~ 'ap.airport_name = ''Внуково' \gx
```

```
-[ RECORD 1 ]----------------------------------------------------------------
queryid    | 1598258635823758032
query_text | EXPLAIN ANALYZE SELECT *                                        +
           | FROM flights fl                                                 +
           |   JOIN airports ap ON ( fl.departure_airport = ap.airport_code )+
           | WHERE ap.airport_name = 'Внуково'                               +
           |   AND fl.flight_no = 'PG0007';
```

```
=> SELECT * FROM aqo_query_stat WHERE queryid = 1598258635823758032 \gx
```

```
-[ RECORD 1 ]-----------------+---------------------
queryid                       | 1598258635823758032
execution_time_with_aqo       | {0.000359746}
execution_time_without_aqo    | {}
planning_time_with_aqo        | {0.000277111}
planning_time_without_aqo     | {}
cardinality_error_with_aqo    | {1.3862943611198906}
cardinality_error_without_aqo | {}
executions_with_aqo           | 1
executions_without_aqo        | 0
```

Выполним запрос еще раз. Теперь уже будет использована уточненная оценка:

```
=> EXPLAIN ANALYZE SELECT *
FROM flights fl
  JOIN airports ap ON ( fl.departure_airport = ap.airport_code )
WHERE ap.airport_name = 'Внуково' AND fl.flight_no = 'PG0007';
```

```
                                                                         QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=4.79..237.47 rows=61 width=162) (actual time=0.066..0.228 rows=61 loops=1)
   AQO: rows=61, error=0%, fss=-461720685
   Join Filter: (fl.departure_airport = ml.airport_code)
   ->  Seq Scan on airports_data ml  (cost=0.00..30.56 rows=1 width=145) (actual time=0.041..0.106 rows=1 loops=1)
         AQO: rows=1, error=0%, fss=1184327055
         Filter: ((airport_name ->> lang()) = 'Внуково'::text)
         Rows Removed by Filter: 103
   ->  Bitmap Heap Scan on flights fl  (cost=4.79..175.34 rows=61 width=63) (actual time=0.016..0.021 rows=61 loops=1)
         AQO: rows=61, error=0%, fss=-1476077258
         Recheck Cond: (flight_no = 'PG0007'::bpchar)
         Heap Blocks: exact=2
         ->  Bitmap Index Scan on flights_flight_no_scheduled_departure_key  (cost=0.00..4.77 rows=64 width=0) (actual time=0.010..0.010 rows=61 loops=1)
               Index Cond: (flight_no = 'PG0007'::bpchar)
 Planning Time: 0.378 ms
 Execution Time: 0.307 ms
 Using aqo: true
 AQO mode: LEARN
 Query hash: 1598258635823758032
 JOINS: 1
(19 rows)
```

Оценка кардинальности исправилась.

В статистике по классу количество выполнений с AQO должно увеличиться.

```
=> SELECT * FROM aqo_query_stat WHERE queryid = 1598258635823758032 \gx
```

```
-[ RECORD 1 ]-----------------+-------------------------
queryid                       | 1598258635823758032
execution_time_with_aqo       | {0.000359746,0.000343014}
execution_time_without_aqo    | {}
planning_time_with_aqo        | {0.000277111,0.000248136}
planning_time_without_aqo     | {}
cardinality_error_with_aqo    | {1.3862943611198906,0}
cardinality_error_without_aqo | {}
executions_with_aqo           | 2
executions_without_aqo        | 0
```

Новая оценка будет применяться и к другим запросам того же класса, например, с другим номером рейса:

```
=> EXPLAIN ANALYZE SELECT *
FROM flights fl
  JOIN airports ap ON fl.departure_airport = ap.airport_code
WHERE ap.airport_name = 'Внуково'
  AND fl.flight_no = 'PG0025';
```

```
                                                                        QUERY PLAN
----------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=4.64..194.17 rows=61 width=162) (actual time=0.063..0.144 rows=9 loops=1)
   AQO: rows=61, error=85%, fss=-461720685
   Join Filter: (fl.departure_airport = ml.airport_code)
   ->  Seq Scan on airports_data ml  (cost=0.00..30.56 rows=1 width=145) (actual time=0.036..0.103 rows=1 loops=1)
         AQO: rows=1, error=0%, fss=1184327055
         Filter: ((airport_name ->> lang()) = 'Внуково'::text)
         Rows Removed by Filter: 103
   ->  Bitmap Heap Scan on flights fl  (cost=4.64..132.05 rows=61 width=63) (actual time=0.018..0.019 rows=9 loops=1)
         AQO: rows=61, error=85%, fss=-1476077258
         Recheck Cond: (flight_no = 'PG0025'::bpchar)
         Heap Blocks: exact=1
         ->  Bitmap Index Scan on flights_flight_no_scheduled_departure_key  (cost=0.00..4.62 rows=44 width=0) (actual time=0.016..0.016 rows=9 loops=1)
               Index Cond: (flight_no = 'PG0025'::bpchar)
 Planning Time: 0.369 ms
 Execution Time: 0.227 ms
 Using aqo: true
 AQO mode: LEARN
 Query hash: 1598258635823758032
 JOINS: 1
(19 rows)
```

Обратите внимание, что запрос классифицируется исходя из дерева разбора, поэтому AQO не чувствителен к таким изменениям текста, как регистр, переносы строк и т. п.

Режим controlled

Основной режим для ручного управления AQO (используется по умолчанию)

Обучение и использование оценок для известных классов

learn_aqo = true

use_aqo = true

Новые классы запросов игнорируются

learn_aqo = false

use_aqo = false

Если в режиме learn AQO обучается и используется для всех запросов, то в режиме controlled — который установлен по умолчанию — AQO продолжает обучаться на известных классах запросов и улучшать их оценки, но игнорирует новые классы запросов.

Этот режим используется по умолчанию и предназначен для ручного управления обучением и применением AQO на уровне отдельных классов запросов. Например, можно разрешить AQO только для некоторых классов запросов (имеющих значительные ошибки оценок кардинальности), а все остальные запросы будут обрабатываться обычным образом.

Режим controlled

Вернем AQO в режим по умолчанию controlled.

```
=> RESET aqo.mode;
```

```
RESET
```

Снова выполним наш запрос:

```
=> EXPLAIN ANALYZE SELECT *
FROM flights fl
  JOIN airports ap ON ( fl.departure_airport = ap.airport_code )
WHERE ap.airport_name = 'Внуково'
  AND fl.flight_no = 'PG0007';
```

```
                                                                         QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=4.79..237.47 rows=61 width=162) (actual time=0.064..0.232 rows=61 loops=1)
   AQO: rows=61, error=0%, fss=-461720685
   Join Filter: (fl.departure_airport = ml.airport_code)
   ->  Seq Scan on airports_data ml  (cost=0.00..30.56 rows=1 width=145) (actual time=0.044..0.111 rows=1 loops=1)
         AQO: rows=1, error=0%, fss=1184327055
         Filter: ((airport_name ->> lang()) = 'Внуково'::text)
         Rows Removed by Filter: 103
   ->  Bitmap Heap Scan on flights fl  (cost=4.79..175.34 rows=61 width=63) (actual time=0.015..0.020 rows=61 loops=1)
         AQO: rows=61, error=0%, fss=-1476077258
         Recheck Cond: (flight_no = 'PG0007'::bpchar)
         Heap Blocks: exact=2
         ->  Bitmap Index Scan on flights_flight_no_scheduled_departure_key  (cost=0.00..4.77 rows=64 width=0) (actual time=0.011..0.011 rows=61 loops=1)
               Index Cond: (flight_no = 'PG0007'::bpchar)
 Planning Time: 0.376 ms
 Execution Time: 0.342 ms
 Using aqo: true
 AQO mode: CONTROLLED
 Query hash: 1598258635823758032
 JOINS: 1
(19 rows)
```

AQO используется, так как запрос принадлежит уже известному классу запросов (Using aqo: true).

Свойства этого класса запросов:

```
=> SELECT queryid, learn_aqo, use_aqo
FROM aqo_queries
WHERE queryid = 1598258635823758032;
```

```
       queryid       | learn_aqo | use_aqo
---------------------+-----------+---------
 1598258635823758032 | t         | t
(1 row)
```

Выполним похожий запрос, в котором вместо поиска по равенству использован поиск по регулярному выражению (символ тильды):

```
=> EXPLAIN ANALYZE SELECT *
FROM flights fl
  JOIN airports ap ON ( fl.departure_airport = ap.airport_code )
WHERE ap.airport_name ~ 'Внуково'
  AND fl.flight_no = 'PG0007';
```

```
                                                                         QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=4.79..207.21 rows=1 width=162) (actual time=0.066..0.258 rows=61 loops=1)
   Join Filter: (fl.departure_airport = ml.airport_code)
   ->  Seq Scan on airports_data ml  (cost=0.00..30.56 rows=1 width=145) (actual time=0.041..0.136 rows=1 loops=1)
         Filter: ((airport_name ->> lang()) ~ 'Внуково'::text)
         Rows Removed by Filter: 103
   ->  Bitmap Heap Scan on flights fl  (cost=4.79..175.34 rows=64 width=63) (actual time=0.020..0.025 rows=61 loops=1)
         Recheck Cond: (flight_no = 'PG0007'::bpchar)
         Heap Blocks: exact=2
         ->  Bitmap Index Scan on flights_flight_no_scheduled_departure_key  (cost=0.00..4.77 rows=64 width=0) (actual time=0.013..0.013 rows=61 loops=1)
               Index Cond: (flight_no = 'PG0007'::bpchar)
 Planning Time: 0.272 ms
 Execution Time: 0.277 ms
(12 rows)
```

В режиме controlled новые запросы не классифицируются AQO (обратите внимание, что в узлах плана запроса отсутствуют строки, соответствующие AQO).

Запретим использование AQO для запроса с равенством:

```
=> SELECT aqo_disable_class(1598258635823758032);
```

```
 aqo_disable_class
-------------------

(1 row)
```

```
=> SELECT queryid, learn_aqo, use_aqo
FROM aqo_queries
WHERE queryid = 1598258635823758032;
```

```
       queryid       | learn_aqo | use_aqo
---------------------+-----------+---------
 1598258635823758032 | f         | f
(1 row)
```

Теперь AQO не будет собирать статистику и не будет использоваться для улучшения оценок этого класса запросов:

```
=> EXPLAIN ANALYZE SELECT *
FROM flights fl
  JOIN airports ap ON ( fl.departure_airport = ap.airport_code )
WHERE ap.airport_name = 'Внуково'
  AND fl.flight_no = 'PG0007';
```

```
                                                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=4.79..207.21 rows=1 width=162) (actual time=0.052..0.214 rows=61 loops=1)
   Join Filter: (fl.departure_airport = ml.airport_code)
   ->  Seq Scan on airports_data ml  (cost=0.00..30.56 rows=1 width=145) (actual time=0.035..0.099 rows=1 loops=1)
         Filter: ((airport_name ->> lang()) = 'Внуково'::text)
         Rows Removed by Filter: 103
   ->  Bitmap Heap Scan on flights fl  (cost=4.79..175.34 rows=64 width=63) (actual time=0.014..0.019 rows=61 loops=1)
         Recheck Cond: (flight_no = 'PG0007'::bpchar)
         Heap Blocks: exact=2
         ->  Bitmap Index Scan on flights_flight_no_scheduled_departure_key  (cost=0.00..4.77 rows=64 width=0) (actual time=0.010..0.010 rows=61 loops=1)
               Index Cond: (flight_no = 'PG0007'::bpchar)
 Planning Time: 0.253 ms
 Execution Time: 0.236 ms
(12 rows)
```

Режим frozen

Как controlled, но без дообучения

Использование имеющихся оценок для известных классов

learn_aqo = false

use_aqo = true

Новые классы запросов игнорируются

learn_aqo = false

use_aqo = false

В режиме frozen статистика, собранная aqo для запросов известных классов, используется, но новые данные не собираются.

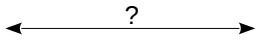
Этот режим позволяет снизить влияние AQO на время планирования и исполнения запросов, но при изменении данных точность планирования будет ухудшаться.

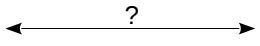
Режим intelligent

Полностью автоматический режим

Решение об обучении и использовании оценок принимается автоматически

auto_tuning = true

learn_aqo = true  false

use_aqo = false  true

В режиме intelligent выполняется автонастройка параметров обучения и использования – для всех запросов устанавливается свойство auto_tuning = true.

Все новые запросы получают свойства learn_aqo = true и use_aqo = false, то есть статистика по ним собирается, но не используется. Запросы выполняются несколько раз с использованием AQO и без него. Чем быстрее запрос выполняется с AQO, тем вероятнее, что автонастройка включит параметр use_aqo.

Если в режиме intelligent производительность запроса не увеличивается после 50 итераций, AQO прекращает сбор статистики и перестает управлять свойствами класса.

Этот режим позволяет использовать AQO в полностью автоматическом режиме, но точность принимаемых решений может оказаться неудовлетворительной, поэтому режим следует применять с осторожностью.

Режим forced

Общая статистика для всех запросов для экономии памяти

- запросы принудительно относятся к одному классу

Данные обучения взаимно несовместимы с данными других режимов

- learn_aqo = true
- use_aqo = true

В режиме forced не выполняется классификация запросов, а статистика собирается не по отдельным классам, а сразу по всем запросам.

Данные машинного обучения, собранные в любом другом режиме, неприменимы для режима forced и наоборот.

Режим forced может быть полезен для оптимизации нагрузки с множеством различных классов запросов, поскольку требует меньше памяти, чем остальные режимы, разделяющие статистику по классам. Однако это может приводить и к ухудшению оценок.

Режим disabled

Временное отключение AQO с сохранением собранной статистики

В режиме disabled отключается использование AQO для всех запросов, даже для запросов известных классов.

Собранная статистика и свойства, установленные для классов, сохраняются и могут быть использованы в дальнейшем.

Итоги

Адаптивная оптимизация запросов AQO — инструмент для улучшения планов методом машинного обучения

Используется для автоматического исправления неверных оценок кардинальности, которые сложно исправлять традиционными методами

AQO допускает тонкую настройку на уровне отдельных классов запросов, имеет несколько режимов для упрощения администрирования

Практика

1. В базе данных demo создайте индексы для таблицы flights по столбцу scheduled_departure и для ticket_flights по столбцу fare_conditions.

2. Отключите параллельное выполнение, увеличьте размер рабочей памяти до 256 Мбайт и выполните запрос для получения количества пассажиров, летавших бизнес-классом с 01.08.2016 и прибывших с опозданием не более часа.

3. Оптимизируйте запрос с помощью AQO.

2. Используйте следующий запрос:

```
SELECT t.ticket_no
FROM flights f
  JOIN ticket_flights tf ON f.flight_id = tf.flight_id
  JOIN tickets t ON tf.ticket_no = t.ticket_no
WHERE f.scheduled_departure > '2016-08-01'::timestamptz
  AND f.actual_arrival < f.scheduled_arrival + interval '1 hour'
  AND tf.fare_conditions = 'Business';
```

3. Сбросьте статистику AQO и исследуйте план запроса в режиме обучения (learn).

1. Индексы

```
=> \c demo
```

```
You are now connected to database "demo" as user "student".
```

Чтобы у планировщика PostgreSQL было больше вариантов соединения таблиц, создадим два индекса.

```
=> CREATE INDEX ON flights (scheduled_departure);
```

```
CREATE INDEX
```

```
=> CREATE INDEX ON ticket_flights (fare_conditions);
```

```
CREATE INDEX
```

2. Запрос с обычным планированием

Отключим параллельную обработку и увеличим work_mem, чтобы вся работа происходила в оперативной памяти.

```
=> ALTER SYSTEM SET max_parallel_workers_per_gather = 0;
```

```
ALTER SYSTEM
```

```
=> ALTER SYSTEM SET work_mem = '256MB';
```

```
ALTER SYSTEM
```

```
=> SELECT pg_reload_conf();
```

```
 pg_reload_conf
----------------
 t
(1 row)
```

Получим количество пассажиров, которые летели бизнес-классом начиная с 01.08.2016 и прибыли с опозданием не более часа.

```
=> EXPLAIN (analyze, buffers, timing off)
SELECT t.ticket_no
FROM flights f
  JOIN ticket_flights tf ON f.flight_id = tf.flight_id
  JOIN tickets t ON tf.ticket_no = t.ticket_no
WHERE f.scheduled_departure > '2016-08-01'::timestamptz
  AND f.actual_arrival < f.scheduled_arrival + interval '1 hour'
  AND tf.fare_conditions = 'Business';
```

```
                                                                                   QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=13205.46..24768.62 rows=36158 width=14) (actual rows=56111 loops=1)
   Hash Cond: (t.ticket_no = tf.ticket_no)
   Buffers: shared hit=495 read=13528
   ->  Seq Scan on tickets t  (cost=0.00..9826.33 rows=366733 width=14) (actual rows=366733 loops=1)
         Buffers: shared read=6159
   ->  Hash  (cost=12753.49..12753.49 rows=36158 width=14) (actual rows=56111 loops=1)
         Buckets: 65536  Batches: 1  Memory Usage: 3033kB
         Buffers: shared hit=492 read=7369
         ->  Hash Join  (cost=2324.74..12753.49 rows=36158 width=14) (actual rows=56111 loops=1)
               Hash Cond: (tf.flight_id = f.flight_id)
               Buffers: shared hit=492 read=7369
               ->  Bitmap Heap Scan on ticket_flights tf  (cost=1213.12..11357.08 rows=108477 width=18) (actual rows=107642 loops=1)
                     Recheck Cond: ((fare_conditions)::text = 'Business'::text)
                     Heap Blocks: exact=7369
                     Buffers: shared hit=95 read=7369
                     ->  Bitmap Index Scan on ticket_flights_fare_conditions_idx  (cost=0.00..1186.00 rows=108477 width=0) (actual rows=107642 loops=1)
                           Index Cond: ((fare_conditions)::text = 'Business'::text)
                           Buffers: shared read=95
               ->  Hash  (cost=973.62..973.62 rows=11040 width=4) (actual rows=15918 loops=1)
                     Buckets: 16384  Batches: 1  Memory Usage: 688kB
                     Buffers: shared hit=394
                     ->  Seq Scan on flights f  (cost=0.00..973.62 rows=11040 width=4) (actual rows=15918 loops=1)
                           Filter: ((scheduled_departure > '2016-08-01 00:00:00+03'::timestamp with time zone) AND (actual_arrival < (scheduled_arrival + '01:00:00'::interval)))
                           Rows Removed by Filter: 17203
                           Buffers: shared hit=394
 Planning:
   Buffers: shared hit=221 read=44 dirtied=1
 Planning Time: 0.794 ms
 Execution Time: 147.566 ms
(29 rows)
```

Имеется несоответствие по количествам планируемых и реально полученных строк.

---

3. AQO

Чтобы работал AQO, нужно загрузить библиотеку.

```
=> ALTER SYSTEM SET shared_preload_libraries = 'aqo';
```

```
ALTER SYSTEM
```

```
=> \q
```

```
student$ sudo systemctl restart postgrespro-ent-13.service
```

Подключим расширение.

```
student$ psql demo
```

```
=> CREATE EXTENSION aqo;
```

```
CREATE EXTENSION
```

Режим по умолчанию — controlled.

```
=> SHOW aqo.mode;
```

```
  aqo.mode
------------
 controlled
(1 row)
```

Сбросим предыдущую статистику AQO и включим режим learn.

```
=> SELECT aqo_reset();
```

```
 aqo_reset
-----------
         0
(1 row)
```

```
=> SET aqo.mode = 'learn';
```

```
SET
```

```
=> SET aqo.show_hash = on;
```

```
SET
```

```
=> SET aqo.show_details = on;
```

```
SET
```

Будем собирать статистику для всех соединений.

```
=> SET aqo.join_threshold = 1;
```

```
SET
```

Выполним запрос и исследуем его план.

```
=> EXPLAIN (analyze, buffers, timing off)
SELECT t.ticket_no
FROM flights f
  JOIN ticket_flights tf ON f.flight_id = tf.flight_id
  JOIN tickets t ON tf.ticket_no = t.ticket_no
WHERE f.scheduled_departure > '2016-08-01'::timestamptz
  AND f.actual_arrival < f.scheduled_arrival + interval '1 hour'
  AND tf.fare_conditions = 'Business';
```

```
                                                                                 QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=13205.46..24768.62 rows=36158 width=14) (actual rows=56111 loops=1)
   AQO not used, fss=-313062150
   Hash Cond: (t.ticket_no = tf.ticket_no)
   Buffers: shared hit=6 read=14017
   ->  Seq Scan on tickets t  (cost=0.00..9826.33 rows=366733 width=14) (actual rows=366733 loops=1)
         AQO not used, fss=1586366212
         Buffers: shared read=6159
   ->  Hash  (cost=12753.49..12753.49 rows=36158 width=14) (actual rows=56111 loops=1)
         Buckets: 65536  Batches: 1  Memory Usage: 3033kB
         Buffers: shared hit=3 read=7858
         ->  Hash Join  (cost=2324.74..12753.49 rows=36158 width=14) (actual rows=56111 loops=1)
               AQO not used, fss=505688925
               Hash Cond: (tf.flight_id = f.flight_id)
               Buffers: shared hit=3 read=7858
               ->  Bitmap Heap Scan on ticket_flights tf  (cost=1213.12..11357.08 rows=108477 width=18) (actual rows=107642 loops=1)
                     AQO not used, fss=115410731
                     Recheck Cond: ((fare_conditions)::text = 'Business'::text)
                     Heap Blocks: exact=7369
                     Buffers: shared read=7464
                     ->  Bitmap Index Scan on ticket_flights_fare_conditions_idx  (cost=0.00..1186.00 rows=108477 width=0) (actual rows=107642 loops=1)
                           Index Cond: ((fare_conditions)::text = 'Business'::text)
                           Buffers: shared read=95
               ->  Hash  (cost=973.62..973.62 rows=11040 width=4) (actual rows=15918 loops=1)
                     Buckets: 16384  Batches: 1  Memory Usage: 688kB
                     Buffers: shared read=394
                     ->  Seq Scan on flights f  (cost=0.00..973.62 rows=11040 width=4) (actual rows=15918 loops=1)
                           AQO not used, fss=-810760453
                           Filter: ((scheduled_departure > '2016-08-01 00:00:00+03'::timestamp with time zone) AND (actual_arrival < (scheduled_arrival + '01:00:00'::interval)))
                           Rows Removed by Filter: 17203
                           Buffers: shared read=394
 Planning:
   Buffers: shared hit=236 read=47
 Planning Time: 1.372 ms
 Execution Time: 150.060 ms
 Using aqo: true
 AQO mode: LEARN
 Query hash: -8407559008517015878
 JOINS: 2
(38 rows)
```

AQO не оптимизирует запрос, если встречает его впервые.

Повторим запрос.

```
=> EXPLAIN (analyze, buffers, timing off)
SELECT t.ticket_no
FROM flights f
  JOIN ticket_flights tf ON f.flight_id = tf.flight_id
  JOIN tickets t ON tf.ticket_no = t.ticket_no
WHERE f.scheduled_departure > '2016-08-01'::timestamptz
  AND f.actual_arrival < f.scheduled_arrival + interval '1 hour'
  AND tf.fare_conditions = 'Business';
```

```
                                                                                 QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=13513.45..25734.55 rows=56111 width=14) (actual rows=56111 loops=1)
   AQO: rows=56111, error=0%, fss=-313062150
   Hash Cond: (t.ticket_no = tf.ticket_no)
   Buffers: shared hit=14017
   ->  Seq Scan on tickets t  (cost=0.00..9826.33 rows=366733 width=14) (actual rows=366733 loops=1)
         AQO: rows=366733, error=0%, fss=1586366212
         Buffers: shared hit=6159
   ->  Hash  (cost=12812.06..12812.06 rows=56111 width=14) (actual rows=56111 loops=1)
         Buckets: 65536  Batches: 1  Memory Usage: 3033kB
         Buffers: shared hit=7858
         ->  Hash Join  (cost=2385.51..12812.06 rows=56111 width=14) (actual rows=56111 loops=1)
               AQO: rows=56111, error=0%, fss=505688925
               Hash Cond: (tf.flight_id = f.flight_id)
               Buffers: shared hit=7858
               ->  Bitmap Heap Scan on ticket_flights tf  (cost=1212.91..11356.88 rows=107642 width=18) (actual rows=107642 loops=1)
                     AQO: rows=107642, error=0%, fss=115410731
                     Recheck Cond: ((fare_conditions)::text = 'Business'::text)
                     Heap Blocks: exact=7369
                     Buffers: shared hit=7464
                     ->  Bitmap Index Scan on ticket_flights_fare_conditions_idx  (cost=0.00..1186.00 rows=108477 width=0) (actual rows=107642 loops=1)
                           Index Cond: ((fare_conditions)::text = 'Business'::text)
                           Buffers: shared hit=95
               ->  Hash  (cost=973.62..973.62 rows=15918 width=4) (actual rows=15918 loops=1)
                     Buckets: 16384  Batches: 1  Memory Usage: 688kB
                     Buffers: shared hit=394
                     ->  Seq Scan on flights f  (cost=0.00..973.62 rows=15918 width=4) (actual rows=15918 loops=1)
                           AQO: rows=15918, error=0%, fss=-810760453
                           Filter: ((scheduled_departure > '2016-08-01 00:00:00+03'::timestamp with time zone) AND (actual_arrival < (scheduled_arrival + '01:00:00'::interval)))
                           Rows Removed by Filter: 17203
                           Buffers: shared hit=394
 Planning:
   Buffers: shared hit=31
 Planning Time: 0.766 ms
 Execution Time: 81.563 ms
 Using aqo: true
 AQO mode: LEARN
 Query hash: -8407559008517015878
 JOINS: 2
(38 rows)
```

Теперь количества планируемых и реально полученных строк стали одинаковыми.