PGConf.Сибирь 2023

PostgresPro

Как мы тестировали Adaptive Query Optimization на разных конфигурациях 1С

Василий Пучков

ООО «ЛУКОЙЛ-Технологии»

PostgresPro

О докладчике

30 лет работы с базами данных

9 лет работы в Газпромнефть

8 лет работы с PostgreSQL

Полгода в ЛУКОЙЛ-Технологии

Что такое AQO

Adaptive Query Optimization

Адаптивная оптимизация запросов

Оптимизация запросов

Этапы обработки запроса

Разбор

Переписывание

Планирование (Оптимизация)

Выполнение

Оптимизация запросов

Данные для планировщика

Синтаксис и семантика

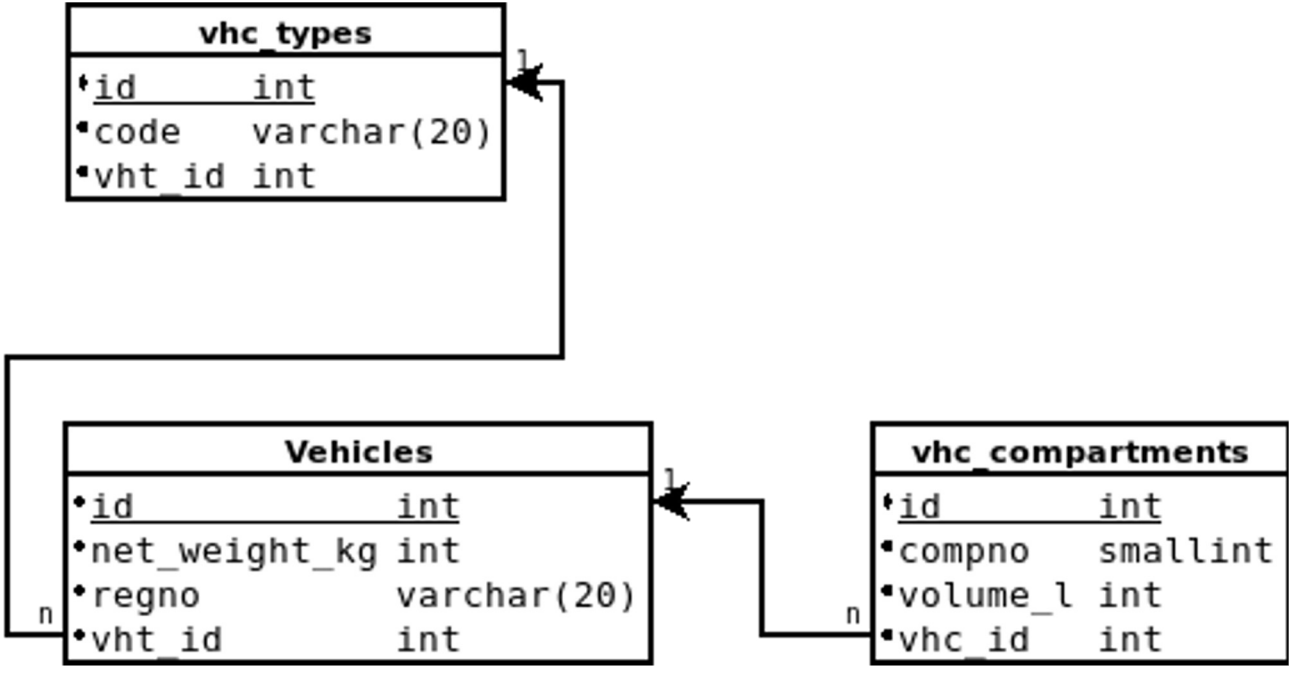
Метаданные

Статистика данных

Статистика выполнения

PostgresPro

Пример



Синтаксис и семантика

```
SELECT vhc.regno
FROM vehicles vhc
JOIN vhc_types vht
ON vhc.vht_id = vht.id
WHERE vht.code =
'semitrailer';
```

```
SELECT vhc.regno
FROM vehicles vhc
WHERE vhc.vht_id IN (
  SELECT vht.id
  FROM vhc_types vht
  WHERE vht.code =
'semitrailer' );
```

```
WITH vht AS MATERIALIZED
(
SELECT id
FROM vhc_types
WHERE code =
'semitrailer'
)
SELECT vhc.regno
FROM vehicles vhc
JOIN vht
ON vhc.vht_id = vht.id;
```

Метаданные

```
SELECT
  vhc.id
FROM
  vehicles vhc
WHERE
  vhc.regno = 'CT0001 54 RUS';
```

```
SELECT
  vhc.id
FROM
  vehicles vhc
WHERE
  vhc.net_weight_kg = 5000;
```

Статистика данных

```
SELECT
  vhc.regno,
  vht.code
FROM
  vehicles vhc
JOIN
  vhc_types vht
ON
  vhc.vht_id = vht.id;
```

```
SELECT
  vhc.regno,
  vcc.compno
FROM
  vehicles vhc
JOIN
  vhc_compartments vcc
ON
  vcc.vcc_id = vhc.id;
```

Статистика выполнения

```
SELECT
  vcc.vhc_id,
  max(vcc.compno)
FROM
  vhc_compartments vcc
WHERE
  vcc.vhc_id = 1
GROUP BY
  vcc.vhc_id;
```

```
SELECT
  max(vcc.compno)
FROM
  vhc_compartments vcc
WHERE
  vcc.vhc_id = 1;
```

PostgresPro

Что такое AQO

Анализиатор статистики выполнения запросов с использованием машинного обучения

PostgresPro

Планирование тестирования

Базы 1С

PostgresPro

Планирование тестирования

Базы 1С

Не слишком хорошо работающие

PostgresPro

Планирование тестирования

Базы 1С

Не слишком хорошо работающие

Но и не слишком плохо

PostgresPro

Планирование тестирования

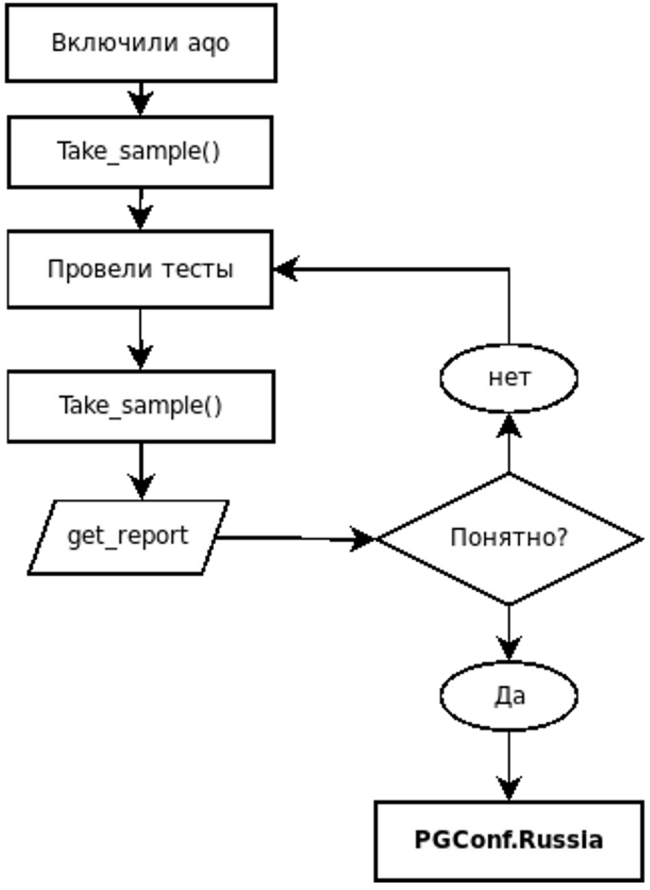
Базы 1С

Не слишком хорошо работающие

Но и не слишком плохо

Возможность проведения воспроизводимых тестов с контролируемым результатом

Процесс


Включили aqo
Take_sample()
Провели тесты
Take_sample()
get_report
Понятно?
нет
Да
PGConf.Russia

Полученные результаты

```
WITH
          qstna AS (
         SELECT
           qs.query_hash,
           avg(etime ) AS avg_e_time
         FROM aqo_query_stat qs,
         LATERAL UNNEST(
          qs.execution_time_without_a
qo )  etime
GROUP BY qs.query_hash ) ,
                    qsta AS (
         SELECT
                    qs.query_hash,
                    avg(etime ) AS
avg_e_time
             FROM aqo_query_stat qs,
                     LATERAL
UNNEST(
qs.execution_time_with_aqo )  etime
          GROUP BY qs.query_hash )
```

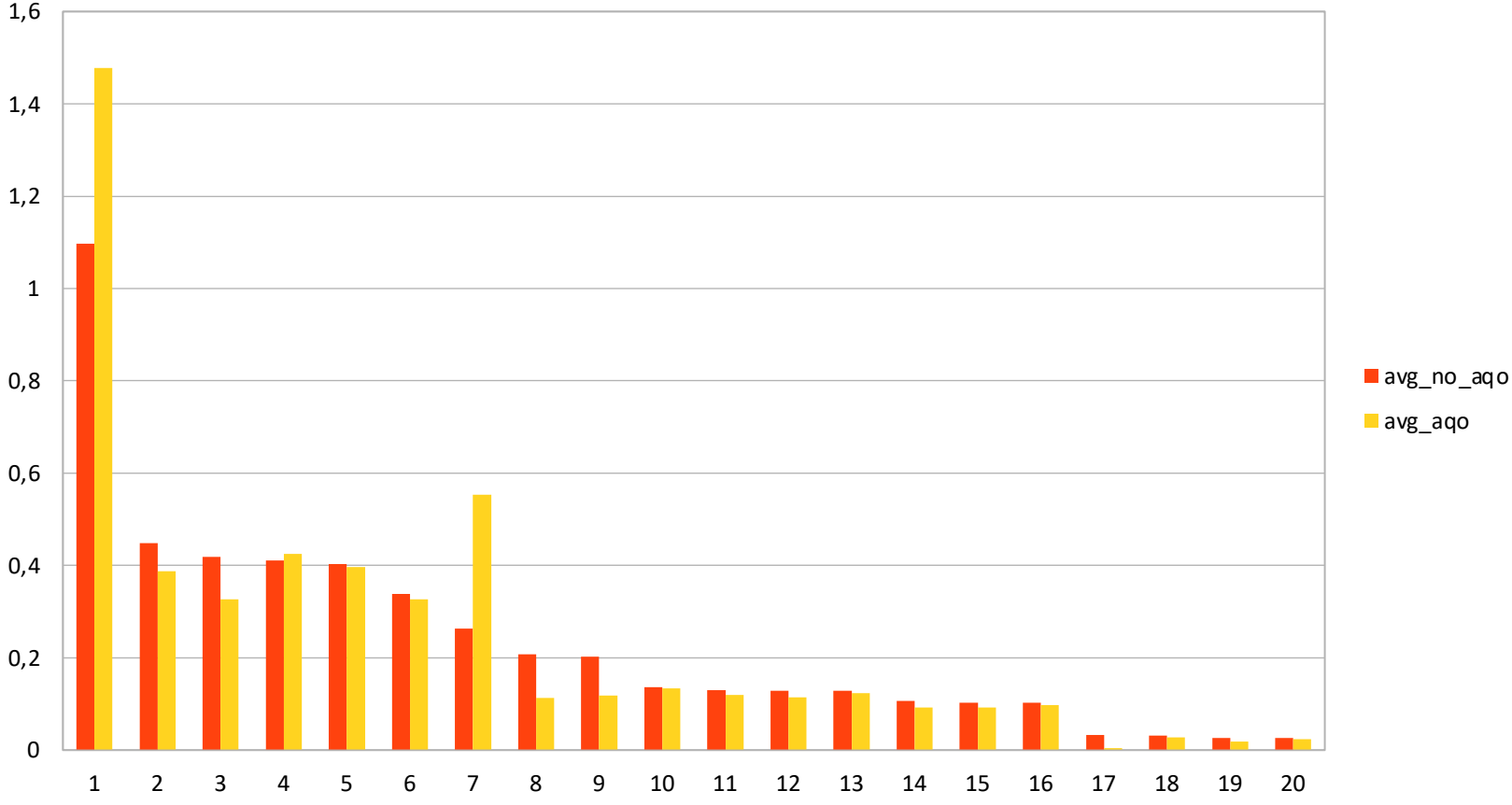
```
SELECT
    qstna.avg_e_time
as avg_no_aqo,

qsta.avg_e_time as
avg_aqo,

qs.executions_with_aq
o,

qs.executions_without
_aqo
FROM
```

Полученные результаты



PostgresPro

Полученные результаты

SQL query statistics

Top SQL by execution time

I	Exec (s)	%Total	Rows	Execution times (ms)				Executions
				Mean	Min	Max	StdErr	
1	178.30	25.14	98	3638.720	3321.001	4868.049	369.323	49
2	223.92	24.36	124	3611.681	3319.265	4868.049	343.646	62
1	66.21	9.33	1625	1018.663	0.841	22642.013	4627.281	65
2	108.92	11.85	2425	1122.937	0.841	22642.013	4813.848	97
1	61.37	8.65	912756	4720.527	2442.895	5862.534	722.214	13
2	61.37	6.68	912756	4720.527	2442.895	5862.534	722.214	13
1	57.11	8.05	912743	4393.016	3985.835	5262.416	295.205	13
2	57.11	6.21	912743	4393.016	3985.835	5262.416	295.205	13
1	55.33	7.80	1046734	4256.340	4119.707	4632.493	142.878	13
2	55.33	6.02	1046734	4256.340	4119.707	4632.493	142.878	13
1	20.69	2.92	280848	5173.541	4787.676	6307.406	654.655	4
2	20.69	2.25	280848	5173.541	4787.676	6307.406	654.655	4

PostgresPro

Выводы

- В дефолтной конфигурации AQO даёт небольшой, но ощутимый эффект за счёт ускорения самых массовых запросов
- Принудительно включать AQO имеет смысл только на запросах, для которых обнаружено несоответствие ожиданий оптимизатора реальной выборке

PostgresPro

Полученные результаты

```
SELECT aq.*
FROM aqo_queries aq
JOIN aqo_query_texts aqt on aq.query_hash = aqt.query_hash
WHERE
         REPLACE( aqt.query_text, chr(10), ' ' ) LIKE
         'SELECT T1._Fld6235RRef, T1._Fld7904 FROM _InfoRg6233 T1 LEFT OUTER JOIN
_Document9127 T2%';

query_hash | learn_aqo | use_aqo | fspace_hash | auto_tuning
-----------+-----------+---------+-------------+-------------
 323327177 | f         | f       |   323327177 | t

(1 row)
```

PostgresPro

Полученные результаты

```
UPDATE aqo_queries SET use_aqo=true, learn_aqo=true, auto_tuning=false
WHERE query_hash IN (
  SELECT query_hash
  FROM aqo_query_texts
  WHERE
         REPLACE( query_text, chr(10), ' ' ) LIKE
         'SELECT T1._Fld6235RRef, T1._Fld7904 FROM _InfoRg6233 T1 LEFT OUTER JOIN
_Document9127 T2%' );

SELECT aq.*
FROM aqo_queries aq
JOIN aqo_query_texts aqt on aq.query_hash = aqt.query_hash
WHERE
         REPLACE( aqt.query_text, chr(10), ' ' ) LIKE
         'SELECT T1._Fld6235RRef, T1._Fld7904 FROM _InfoRg6233 T1 LEFT OUTER JOIN
_Document9127 T2%';

query_hash | learn_aqo | use_aqo | fspace_hash | auto_tuning
-----------+-----------+---------+-------------+-------------
 323327177 | t         | t       |   323327177 | f

(1 row)
```

PostgresPro

Но нет...

I	Exec (s)	%Total	Rows	Execution times (ms)				Executions
				Mean	Min	Max	StdErr	
1	223.92	24.36	124	3611.681	3319.265	4868.049	343.646	62
2	23.20	8.62	12	3866.592	3603.244	4701.840	378.466	6
1	108.92	11.85	2425	1122.937	0.841	22642.013	4813.848	97
2	66.93	24.88	1375	1216.969	0.873	22442.367	5061.661	55

PostgresPro

Спасибо!