Количество строк 120000

"Простые" запросы

1. ```
   explain select "idContent", val_boolean from "contentAttrValues" where val_boolean;
   ```
   ```
   Seq Scan on "contentAttrValues"  (cost=0.00..2153.00 rows=15328 width=5)Filter: val_boolean
   ```
2. ```
   explain select "idContent" from "contentAttrValues" where idfield = 10;
   ```
   ```
   Seq Scan on "contentAttrValues"  (cost=0.00..2453.00 rows=10348 width=4)Filter: (idfield = 10)
   ```
3. ```
   explain select "idContent" from "contentAttrValues" where val_date = '2020-12-24'
   ```
   ```
   Seq Scan on "contentAttrValues"  (cost=0.00..2453.00 rows=2948 width=4)Filter: (val_date = '2020-12-24'::date)
   ```

"Сложные" запросы

1. ```
   explain select c2."name", cav.val_text from "contentAttrValues" cav
      join "content" c2 on c2.id = cav."idContent"
      where c2."name" = 'movie2';
   ```
   Nested Loop (cost=0.70..36.30 rows=12 width=60)
   -> Index Scan using content_un on content c2 (cost=0.41..8.43 rows=1 width=17)
   Index Cond: ((name)::text = 'movie2'::text)
   -> Index Scan using contentattrvalues_un on "contentAttrValues" cav (cost=0.29..27.75 rows=12 width=55)
   Index Cond: ("idContent" = c2.id)
2. ```
   explain select c2."name" as content_name, cf."name" as award, ca."name" as type from "contentAttrValues" cav
            join "content" c2 on c2.id = cav."idContent"
            join "contentFields" cf ON cf.id  = cav.idfield
            join "contentAttr" ca ON ca.id  = cf."idAttr"
            where ca.id = 2;
   ```
   Nested Loop (cost=300.16..3006.72 rows=10000 width=1041)
   -> Seq Scan on "contentAttr" ca (cost=0.00..1.05 rows=1 width=520)
   Filter: (id = 2)
   -> Hash Join (cost=300.16..2905.67 rows=10000 width=529)
   Hash Cond: (cav."idContent" = c2.id)
   -> Hash Join (cost=1.16..2580.41 rows=10000 width=524)
   Hash Cond: (cav.idfield = cf.id)
   -> Seq Scan on "contentAttrValues" cav (cost=0.00..2153.00 rows=120000 width=8
   -> Hash (cost=1.15..1.15 rows=1 width=524)
   -> Seq Scan on "contentFields" cf (cost=0.00..1.15 rows=1 width=524)
   Filter: ("idAttr" = 2)
   -> Hash (cost=174.00..174.00 rows=10000 width=17)
   -> Seq Scan on content c2 (cost=0.00..174.00 rows=10000 width=17)
3. ```
   explain select c2."name", cf."name", cav.val_date from "contentAttrValues" cav
            join "content" c2 on c2.id = cav."idContent"
            join "contentFields" cf ON cf.id  = cav.idfield
            join "contentAttr" ca ON ca.id  = cf."idAttr"
            where ca.id = 4 and cav.val_date < '2021-01-01' order by cav.val_date ASC;
   ```
   Sort (cost=3012.32..3018.12 rows=2322 width=529)
   Sort Key: cav.val_date

```
 ->  Nested Loop  (cost=300.16..2882.50 rows=2322 width=529)
       ->  Seq Scan on "contentAttr" ca  (cost=0.00..1.05 rows=1 width=4)
             Filter: (id = 4)
       ->  Hash Join  (cost=300.16..2858.23 rows=2322 width=533)
             Hash Cond: (cav."idContent" = c2.id)
             ->  Hash Join  (cost=1.16..2553.14 rows=2322 width=528)
                   Hash Cond: (cav.idfield = cf.id)
                   ->  Seq Scan on "contentAttrValues" cav  (cost=0.00..2453.00 rows=27864
width=12)
                         Filter: (val_date < '2021-01-01'::date)
                   ->  Hash  (cost=1.15..1.15 rows=1 width=524)
                         ->  Seq Scan on "contentFields" cf  (cost=0.00..1.15 rows=1 width=524)
                               Filter: ("idAttr" = 4)
             ->  Hash  (cost=174.00..174.00 rows=10000 width=17)
                   ->  Seq Scan on content c2  (cost=0.00..174.00 rows=10000 width=17)
```

Количество строк 1200000

"Простые" запросы

1. ```
   explain select "idContent", val_boolean from "contentAttrValues" where val_boolean;
   ```
   Seq Scan on "contentAttrValues" (cost=0.00..21524.00 rows=149920 width=5) Filter: val_boolean
2. ```
   explain select "idContent" from "contentAttrValues" where idfield = 10;
   ```
   Seq Scan on "contentAttrValues" (cost=0.00..24524.00 rows=99800 width=4) Filter: (idfield = 10)
3. ```
   explain select "idContent" from "contentAttrValues" where val_date = '2020-12-24'
   ```
   Gather (cost=1000.00..19690.00 rows=29160 width=4)
   Workers Planned: 2
   -> Parallel Seq Scan on "contentAttrValues" (cost=0.00..15774.00 rows=12150 width=4) Filter: (val_date = '2020-12-24'::date)

"Сложные" запросы

1. ```
   explain select c2."name", cav.val_text from "contentAttrValues" cav
            join "content" c2 on c2.id = cav."idContent"
      where c2."name" = 'movie2';
   ```
   Nested Loop (cost=0.84..36.44 rows=12 width=60)
   -> Index Scan using content_un on content c2 (cost=0.42..8.44 rows=1 width=18)
   Index Cond: ((name)::text = 'movie2'::text)
   -> Index Scan using contentattrvalues_un on "contentAttrValues" cav (cost=0.43..27.89rows=12 width=54)
   Index Cond: ("idContent" = c2.id)
2. ```
   explain select c2."name" as content_name, cf."name" as award, ca."name" as type from "contentAttrValues" cav
                  join "content" c2 on c2.id = cav."idContent"
                  join "contentFields" cf ON cf.id  = cav.idfield
                  join "contentAttr" ca ON ca.id  = cf."idAttr"
            where ca.id = 2;
   ```
   Nested Loop (cost=3573.19..39008.27 rows=300000 width=64)
   -> Seq Scan on "contentAttr" ca (cost=0.00..1.05 rows=1 width=24)
   Filter: (id = 2)
   -> Hash Join (cost=3573.19..36007.22 rows=300000 width=48)
   Hash Cond: (cav."idContent" = c2.id)
   -> Hash Join (cost=1.19..25787.69 rows=300000 width=42)
   Hash Cond: (cav.idfield = cf.id)
   -> Seq Scan on "contentAttrValues" cav (cost=0.00..21524.00 rows=1200000 width=8)
   -> Hash (cost=1.15..1.15 rows=3 width=42)
   -> Seq Scan on "contentFields" cf (cost=0.00..1.15 rows=3 width=42)
   Filter: ("idAttr" = 2)
   -> Hash (cost=1736.00..1736.00 rows=100000 width=18)
   -> Seq Scan on content c2 (cost=0.00..1736.00 rows=100000 width=18)
3. ```
   explain select c2."name", cf."name", cav.val_date from "contentAttrValues" cav
                  join "content" c2 on c2.id = cav."idContent"
                  join "contentFields" cf ON cf.id  = cav.idfield
                  join "contentAttr" ca ON ca.id  = cf."idAttr"
            where ca.id = 4 and cav.val_date < '2021-01-01' order by cav.val_date ASC;
   ```
   Gather Merge (cost=24380.12..31119.96 rows=57766 width=48)
   Workers Planned: 2
   -> Sort (cost=23380.10..23452.30 rows=28883 width=48)
   Sort Key: cav.val_date
   -> Hash Join (cost=3574.27..21240.16 rows=28883 width=48)
   Hash Cond: (cav."idContent" = c2.id)
   -> Hash Join (cost=2.27..16498.35 rows=28883 width=42)
   Hash Cond: (cav.idfield = cf.id)

```
      ->  Parallel Seq Scan on "contentAttrValues" cav  (cost=0.00..15774.00 rows=115533 width=12)
            Filter: (val_date < '2021-01-01'::date)
      ->  Hash  (cost=2.23..2.23 rows=3 width=38)
            ->  Nested Loop  (cost=0.00..2.23 rows=3 width=38)
                  ->  Seq Scan on "contentAttr" ca  (cost=0.00..1.05 rows=1 width=4)
                        Filter: (id = 4)
                  ->  Seq Scan on "contentFields" cf  (cost=0.00..1.15 rows=3 width=42)
                        Filter: ("idAttr" = 4)
->  Hash  (cost=1736.00..1736.00 rows=100000 width=18)
      ->  Seq Scan on content c2  (cost=0.00..1736.00 rows=100000 width=18)
```

Оптимизация для “простых” запросов.

Первый “простой” запрос был оптимизирован примерно в два раза, путем добавления частичного индекса на поле val_boolean.

```
create index "i_cav_val_boolean_only_true" on "contentAttrValues" using btree (val_boolean) where  val_boolean;
```

```
Bitmap Heap Scan on "contentAttrValues"  (cost=2443.47..13466.67 rows=149920 width=5)
  Recheck Cond: val_boolean
  ->  Bitmap Index Scan on i_cav_val_boolean_only_true  (cost=0.00..2405.99 rows=149920 width=0)
```

Второй “простой” запрос был оптимизорован также примерно в два раза, путем добавления частично индекса на поле idfield

```
create index "i_cav_idfield_10" on "contentAttrValues" using btree ("idfield") where idfield=10;
```

```
Bitmap Heap Scan on "contentAttrValues"  (cost=1628.24..12399.74 rows=99800 width=4)
  Recheck Cond: (idfield = 10)
  ->  Bitmap Index Scan on i_cav_idfield_10  (cost=0.00..1603.29 rows=99800 width=0)
```

Третий “простой” запрос был оптимизорован также примерно в два раза, путем добавления частично индекса на поле val_date

```
create index "i_cav_val_date_2020_12_24" on "contentAttrValues" using btree ("val_date") where val_date = '2020-12-24'
```

```
Bitmap Heap Scan on "contentAttrValues"  (cost=481.38..10369.88 rows=29160 width=4)
  Recheck Cond: (val_date = '2020-12-24'::date)
  ->  Bitmap Index Scan on i_cav_val_date_2020_12_24  (cost=0.00..474.09 rows=29160 width=0)
```

Дополнение:

Удалил частичный индекс `"i_cav_val_date_2020_12_24”` и добавил индекс на все поле val_date.

```
CREATE INDEX i_cav_val_date ON public."contentAttrValues" USING btree (val_date)
```

```
Bitmap Heap Scan on "contentAttrValues"  (cost=550.42..10438.92 rows=29160 width=4)
  Recheck Cond: (val_date = '2020-12-24'::date)
  ->  Bitmap Index Scan on i_cav_val_date  (cost=0.00..543.13 rows=29160 width=0)
        Index Cond: (val_date = '2020-12-24'::date)
```

Таким образом частиный индекс для поля val_date добавляет минимальный прирост, на выборе из 1200000 записей и поэтому он не нужен.

Сделал аналогично и для других индексов `"i_cav_val_boolean_only_true"` `"i_cav_idfield_10”.`

```
create index "i_cav_idfield" on "contentAttrValues" using btree ("idfield");
create index "i_cav_val_date" on "contentAttrValues" using btree ("val_date");
```

Вывод: Добавление частичного индекса добавляет производительность, но является не эффективным по сравнению с общим индексом на выборке 1200000, ведь частиный индекс будет использоваться только на специальном запросе, для конкретного значения.

Оптимизация для “сложных” запросов.

В данных тестах уже используются индексы из “простых” запросов.

Первый “сложный” запрос был оптимизирован, путем добавления индекса в таблицу content на поле name, а также индекс был добавлен на поле idContent в таблицу contentAttrValues. Добавление данных индексов оптимизировала в два раза.

```
CREATE INDEX "i_cav_idContent" ON public."contentAttrValues" USING btree ("idContent")
CREATE INDEX i_c_name ON public.content USING btree (name)
```

```
Nested Loop  (cost=0.84..17.19 rows=12 width=60)
  ->  Index Scan using i_c_name on content c2  (cost=0.42..8.44 rows=1 width=18)
        Index Cond: ((name)::text = 'movie2'::text)
  ->  Index Scan using "i_cav_idContent" on "contentAttrValues" cav  (cost=0.43..8.64 rows=12 width=54)
        Index Cond: ("idContent" = c2.id)
```

Вторый “сложный” запрос не смог оптимизировать индексами. Добавлял HASH индексы, а также добавил индекс `CREATE INDEX i_ca_id2 ON public."contentAttr" USING btree (id) WHERE (id = 2),` **но он имеет размер такой же как и первичный индекс для таблицы contentAttr, поэтому он не дал прироста. Переписал запрос таким образом**

```
explain select c2."name" as content_name, cf."name" as award, ca."name" as type from "contentAttrValues" cav
join "content" c2 on c2.id = cav."idContent"
join "contentFields" cf ON cf.id  = cav.idfield
join "contentAttr" ca ON ca.id  = cf."idAttr"
where cav.idfield in (select cf.id from "contentFields" as cf, "contentAttr"as ca where ca.id = cf."idAttr" and ca.id = 2);
```

```
Hash Join  (cost=5083.82..30136.12 rows=300000 width=64)
  Hash Cond: (cav."idContent" = c2.id)
  ->  Nested Loop  (cost=1511.82..18744.59 rows=300000 width=58)
        ->  Hash Join  (cost=3.39..4.61 rows=3 width=62)
              Hash Cond: (cf."idAttr" = ca.id)
              ->  Hash Join  (cost=2.30..3.50 rows=3 width=46)
                    Hash Cond: (cf.id = cf_1.id)
                    ->  Seq Scan on "contentFields" cf  (cost=0.00..1.12 rows=12 width=42)
                    ->  Hash  (cost=2.27..2.27 rows=3 width=4)
                          ->  HashAggregate  (cost=2.24..2.27 rows=3 width=4)
                                Group Key: cf_1.id
                                ->  Nested Loop  (cost=0.00..2.23 rows=3 width=4)
                                      ->  Seq Scan on "contentAttr" ca_1  (cost=0.00..1.05 rows=1 width=4)
                                            Filter: (id = 2)
                                      ->  Seq Scan on "contentFields" cf_1  (cost=0.00..1.15 rows=3 width=8)
                                            Filter: ("idAttr" = 2)
              ->  Hash  (cost=1.04..1.04 rows=4 width=24)
                    ->  Seq Scan on "contentAttr" ca  (cost=0.00..1.04 rows=4 width=24)
        ->  Bitmap Heap Scan on "contentAttrValues" cav  (cost=1508.43..5246.66 rows=100000 width=8)
              Recheck Cond: (idfield = cf.id)
              ->  Bitmap Index Scan on i_cav_idfield  (cost=0.00..1483.43 rows=100000 width=0)
                    Index Cond: (idfield = cf.id)
  ->  Hash  (cost=1736.00..1736.00 rows=100000 width=18)
        ->  Seq Scan on content c2  (cost=0.00..1736.00 rows=100000 width=18)
```

Analyze показал время выполнения в два раза меньше чем для предыдущего запроса.

Третий “сложный” запрос был оптимизирован, путем добавления частичного индекса в таблицу contentAttrValues на поле val_date со значением.
```
CREATE INDEX i_cav_val_date_less_2021 ON public."contentAttrValues" USING btree (val_date) WHERE (val_date < '2021-01-01'::date)
```
это дало прирост.

```
Gather Merge  (cost=23787.08..30318.07 rows=55976 width=48)
  Workers Planned: 2
  ->  Sort  (cost=22787.05..22857.02 rows=27988 width=48)
        Sort Key: cav.val_date
        ->  Hash Join  (cost=7945.25..20719.79 rows=27988 width=48)
              Hash Cond: (cav."idContent" = c2.id)
              ->  Hash Join  (cost=4373.25..15996.32 rows=27988 width=42)
                    Hash Cond: (cav.idfield = cf.id)
                    ->  Parallel Bitmap Heap Scan on "contentAttrValues" cav  (cost=4370.99..15294.36 rows=111950 width=12)
                          Recheck Cond: (val_date < '2021-01-01'::date)
                          ->  Bitmap Index Scan on i_cav_val_date_less_2021  (cost=0.00..4303.82 rows=268679 width=0)
                    ->  Hash  (cost=2.23..2.23 rows=3 width=38)
                          ->  Nested Loop  (cost=0.00..2.23 rows=3 width=38)
                                ->  Seq Scan on "contentAttr" ca  (cost=0.00..1.05 rows=1 width=4)
                                      Filter: (id = 4)
                                ->  Seq Scan on "contentFields" cf  (cost=0.00..1.15 rows=3 width=42)
                                      Filter: ("idAttr" = 4)
              ->  Hash  (cost=1736.00..1736.00 rows=100000 width=18)
                    ->  Seq Scan on content c2  (cost=0.00..1736.00 rows=100000 width=18)
```

Вывод:
Для сложных запросов не всегда помогают индексы, нужно рассматривать сам запрос и возможно его перестроить на основе команд exlain analyze.