



# Новые «глазные капли» или как сделать статистику Postgres более детализированной, не повышая DST

Артем Бугаенко, «Тантор Лабс»



# Репозиторий BootCamp

[https://github.com/TantorLabs/meetups/tree/main/2025-04-10\\_Ekb/](https://github.com/TantorLabs/meetups/tree/main/2025-04-10_Ekb/)

Ссылка на репозиторий, где находятся материалы к выступлению и презентация



QR-Код



# План выступления

## Введение в тему

Что такое Default Statistics Target?

Механизм работы DST

На что он влияет

## Другие подходы

Сравнение механизма DST с аналогами

в других СУБД

Плюсы и минусы разных подходов

## Проблема, будем знакомы

В каком случае стоит повышать значение DST?

Когда это не поможет?

## Statistics Sample Multiplier

Новый параметр для разделения функционала DST

## Как тебя внедрять?

Руководство по добавлению нового параметра SSM в кодовую базу Postgres 16

## Подводим итоги

Анализ результатов

Разбор плюсов и минусов подхода

Инструкция по применению

# Мой путь

- › Спб ГУАП
- › Константа
- › Контур НИИРС
- › SK Hynix
- › Tantor Labs



@BUGGAI

@BUGGAI



# Default Statistics Target

- › `default_statistics_target` – определяет, **сколько** статистической **информации** **собирается** для каждого столбца таблицы при **анализе**.
- › Чем *выше* **DST**, тем больше **объем** статистики, сохраняемой для каждого столбца.
- › В частности он определяет:
  - Количество наиболее частых значений (Most Common Values, **MCV**) и их **частот**.
  - Количество «корзин» (**bins**) **гистограммы распределения** значений столбца.
  - Скалярные метрики: оценка числа различных значений ( Number Distinct Values, **NDV**), средняя **длина значения** и т.п.



# Результат сбора статистики

ANALYZE

pg\_stats

column_name	avg_width	n_distinct	correlation
_active	1	1	1
_fld101573rref	17	1	1
_fld101574rref	17	2	0.9945171
_fld112586	3	4734	0.95673287
_fld2488	3	1	1
_fld50091rref	17	6	0.9975408
_fld50092rref	17	144	0.032824297
_fld50093rref	17	1	1
_fld50094rref	17	360	0.013280741
_fld50095_rrref	17	1018	0.21460111
_fld50095_rtref	5	11	0.40714076
_fld50095_type	2	2	0.85697055
_fld50096	3	19384	0.94330037
_fld50097	3	14812	0.979717
_fld50098	5	66720	-0.033403505
_fld50099	3	14008	0.7812441
_fld50100	3	10170	0.59694105
_fld50101	3	19517	0.9435138
_fld50102rref	17	51	0.8292725
_fld50103rref	17	4086	0.9853446
_fld50104rref	17	1	1



**MCV**

НАИБОЛЕЕ ЧАСТО  
ВСТРЕЧАЮЩИЕСЯ  
ЗНАЧЕНИЯ



**ГИСТОГРАММА**  
РАСПРЕДЕЛЕНИЯ



**СКАЛЯРНЫЕ**  
ПОКАЗАТЕЛИ

# Связь с планировщиком

- › Опираясь на список наиболее частых значений (MCV), планировщик оценивает селективность **равенств**.
- › Гистограмма используется для оценок селективности **диапазонных условий** и **неравенств**.
- › Оценки числа различных значений влияют на селективность, кардинальность и, следовательно, на выбор алгоритмов **соединения** и **агрегирования**.
- › `SELECT ... FROM ... WHERE ... = 'mcv_v'`
- › `SELECT ... FROM ... WHERE ... BETWEEN value1 AND value2`
- › `SELECT ... FROM ... JOIN ... ON ... = ...`

\*Селективность (selectivity) — это **доля строк** таблицы, удовлетворяющих конкретному условию.

\*Кардинальность (cardinality) — это **оценка количества** строк, возвращаемых оператором или запросом.

# Сбор статистики в других СУБД

Для сравнения подходов рассмотрим:

- › Microsoft SQL Server
- › MySQL
- › Oracle



# Microsoft SQL Server

Оптимизатор **SQL Server** автоматически собирает статистику при создании индексов и при достаточном изменении данных.

- › Гистограмма **максимум 200 «шагов»** (аналог корзины).
- › Полу автоматическая настройка размера выборки.
- › Есть аналог MCV.

# MySQL

MySQL оценивает селективность выражений:

- › На основе индексов – количество сэмплируемых данных контролируется опцией **STATS\_SAMPLE\_PAGES**.
- › На основе гистограмм (`ANALYZE TABLE ... UPDATE HISTOGRAM WITH ... BUCKETS`)
  - Количество сэмплируемых данных контролируется опцией **histogram\_generation\_max\_mem\_size**
  - Максимум корзин – **1024**.
  - По **умолчанию не имеет** детальной статистики распределения для неиндексированных колонок.

Автоматическое обновление статистики (опция **STATS\_AUTO\_RECALC** )

# Oracle

В **Oracle** нет прямого глобального аналога **DST**, но есть гибкие настройки сбора статистики через пакет **DBMS\_STATS**.

- › **Максимальное** число корзин гистограммы по умолчанию – **254**.
- › **Динамически** выбирает **объём выборки** и **детализацию** статистики.

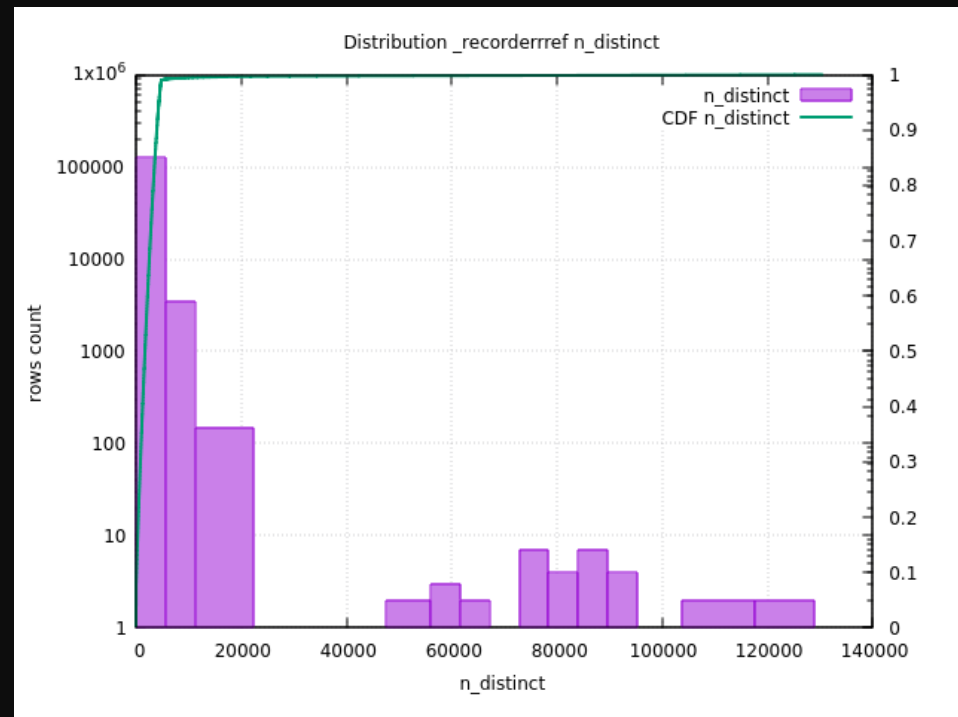
# Резюме по аналогам

- › SQL Server: : фиксированная детализация (200 бинов) + **автоматическая** выборка.
- › MySQL: **настраиваемое** число страниц индекса для анализа + автоматический подход.
- › Oracle: **автоматический** подход (AUTO\_SAMPLE\_SIZE) с возможностью **настройки**.
- › PostgreSQL: **300 \* DST** строк для анализа.

# Проблематика наглядно

Когда приходится повышать **DST**:

- › Столбец с *нестандартным* распределением:
  - Очень большой **разброс частот**
  - Длинный «хвост» редких значений
  
- › Столбец с **большим числом** *уникальных* значений:
  - Десятки - сотни миллионов строк
  - Сотни тысяч - миллионы **уникальных значений**



Пример проблемного распределения

# Проблематика наглядно

## › Текст запроса

```
1  explain analyze SELECT
2  T1._Period
3  FROM _AccumRg50090 T1
4  WHERE ((T1._Fld2488 = CAST(0 AS NUMERIC))) AND ((T1._RecorderTRef = '\\000\\000\\004\\262'::bytea
5  AND T1._RecorderRRef = '\\211\\302\\030f\\332\\261R\\333\\021\\356\\336\\374\\035\\2265\\252'::bytea))
6  ORDER BY (T1._Period) LIMIT 1;
```

## › Недостаточная точность

```
Limit (cost=1059.44..1589.05 rows=1 width=8) (actual time=3444.646..3444.647 rows=1 loops=1)
-> Index Only Scan using _accumrg50090_1 on _accumrg50090 t1
    (cost=0.23..1583002.88 rows=2989 width=8) (actual time=3444.644..3444.644 rows=1 loops=1)
        Index Cond: ((_fld2488 = '0'::numeric) AND (_recordertref = '\\x000004b2'::bytea)
            AND (_recorderrrref = '\\x89c21866dab152db11eedefc1d9635aa'::bytea))
        Heap Fetches: 0
Planning Time: 3.630 ms
Execution Time: 3444.673 ms
```

## Достаточная точность

```
Limit (cost=2507.95..2507.95 rows=1 width=8) (actual time=0.620..0.621 rows=1 loops=1)
-> Sort (cost=2507.94..2509.30 rows=1356 width=8) (actual time=0.619..0.619 rows=1 loops=1)
    Sort Key: _period
    Sort Method: quicksort Memory: 25kB
-> Index Scan using _accumrg50090_2 on _accumrg50090 t1
    (cost=0.23..2494.38 rows=1356 width=8) (actual time=0.598..0.598 rows=1 loops=1)
        Index Cond: ((_fld2488 = '0'::numeric) AND (_recordertref = '\\x000004b2'::bytea)
            AND (_recorderrrref = '\\x89c21866dab152db11eedefc1d9635aa'::bytea))
Planning Time: 1.882 ms
Execution Time: 0.643 ms
```

## › Ускорение выполнения запроса в ~ 5 300 раз



# Повышаем DST

- › + Повышение размера выборки
- › + Увеличение количества MCV
- › + Повышение точности показателей
- › - Увеличение времени ANALYZE
- › - Повышение размера хранимой статистики
- › - Снижение скорости работы планировщика

# Расширяя горизонты. Statistics Sample Multiplier

- › Новый колоночный атрибут Statistic Sample Multiplier (**SSM**)

Когда он пригодится:

- › Нет возможности повысить DST из-за **накладных затрат** планировщика и размера статистики
- › DST **максимальный**, а точность статистики всё ещё **недостаточна**

# Основная идея

## › Разделение функционала DST

- Базовый функционал DST (без изменений)
- Параметр SSM вносит возможность увеличить объём выборки.
- Диапазон значений SSM от 1 до 10 000 ( “-1” — значение по умолчанию)

## › Внесение изменений в Analyze.c, где target\_rows – количество строк для выборки.

• До:  
$$\text{target\_rows} = 300 \times \text{DST}$$

• После:  
$$\text{target\_rows} = 300 \times \text{DST} \times \text{SSM}$$

# План внедрения

## Создать переменную

/src/include/catalog/pg\_attribute.h

## Добавить лексему

/src/backend/parser/gram.y

/src/include/parser/kwlist.h

## Установка переменной

/src/include/nodes/parsenodes.h

/src/backend/commands/tablecmds.c

## Документирование Sgml

/doc/src/sgml/ref/alter\_table.sgml

## Учесть зависимости

/src/backend/catalog/index.c

/src/backend/bootstrap/bootstrap.c

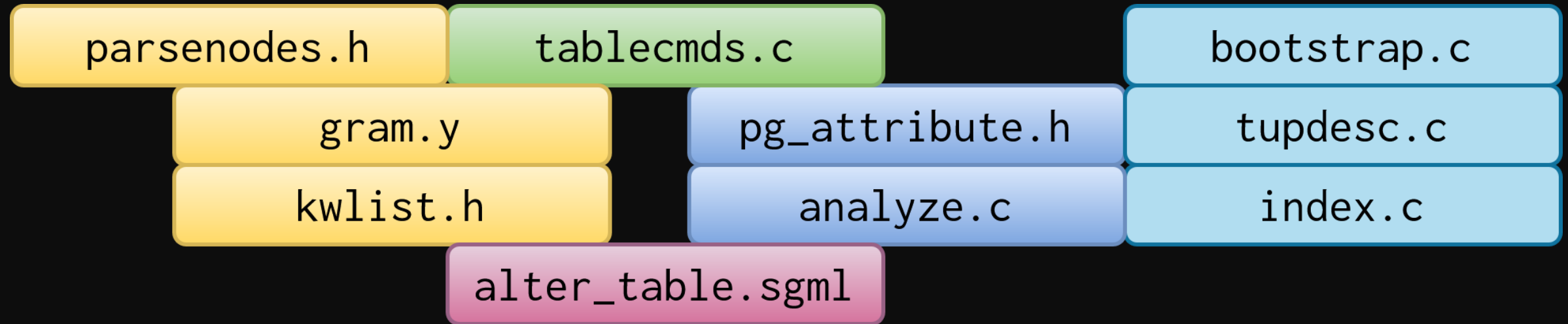
/src/backend/access/common/tupdesc.c

pg\_dump / pg\_restore

## Основной функционал

/src/backend/commands/analyze.c

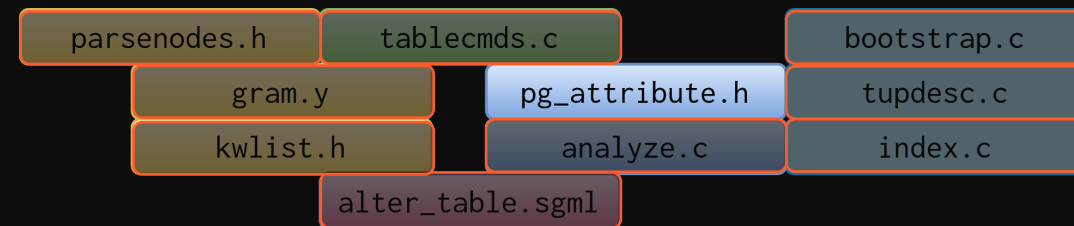
# Схема зависимостей колоночного атрибута



- › Поддержка в синтаксисе SQL
- › Цепочка Data Definition Language-обработки (DDL)
- › Изменения, связанные с вводимой переменной
- › Зависимости
- › Документация

# Создадим переменную (pg\_attribute.h)

› Файл: `src/include/catalog/pg_attribute.h`



› Добавлено новое поле:

```
170 → int16 → attstattarget BKI_DEFAULT(-1);
171 →
172 → /* statistics sample multiplier */
173 → int16 → attstatssamplemultiplier BKI_DEFAULT(-1);
174 →
175 → /* attribute's collation, if any */
176 → Oid → attcollation BKI_LOOKUP_OPT(pg_collation);
```

› Важно:

Поместить новый атрибут перед  
`Oid attcollation BKI_LOOKUP_OPT(pg_collation);`  
 Его позиция используется для вычисления размера  
 структуры `FormData_pg_attribute`

› Для **безопасного перехода** на новую версию  
 системного каталога стоит использовать механизм  
`pg_dump/pg_restore`



# Объявим лексему для парсера (kwlist.h)



- › Нужно добавить ключевое слово **STATMULTIPLIER** и задать ему соответствующие категории.

```

408 PG_KEYWORD("statement", STATEMENT, UNRESERVED_KEYWORD, BARE_LABEL)
409 PG_KEYWORD("statistics", STATISTICS, UNRESERVED_KEYWORD, BARE_LABEL)
410 PG_KEYWORD("statmultiplier", STATMULTIPLIER, UNRESERVED_KEYWORD, BARE_LABEL)
411 PG_KEYWORD("stdin", STDIN, UNRESERVED_KEYWORD, BARE_LABEL)
412 PG_KEYWORD("stdout", STDOUT, UNRESERVED_KEYWORD, BARE_LABEL)
  
```

- › "statmultiplier" — ключевое слово, добавленное в грамматику.
- › **STATMULTIPLIER** — его внутренний токен.
- › UNRESERVED\_KEYWORD — Парсер распознаёт его как **специальное** слово только в контексте, где оно **ожидается**.
- › BARE\_LABEL — можно использовать, как label (Например имена переменных в PL/pgSQL).

# Создадим лексему Gram.y

## › Добавим ключевое слово **STATMULTIPLIER**:

- в список токенов ключевых слов
- в категории **bare\_label\_keyword**: и **unreserved\_keyword**:

## › Создадим лексические правила:

- ALTER TABLE <name> ALTER [COLUMN] <col**name**> SET STATMULTIPLIER <SignedIconst>  
ALTER opt\_column **ColId** SET **STATMULTIPLIER** SignedIconst

- Где **SQL** превращается в абстрактное синтаксическое дерево (**AST**)

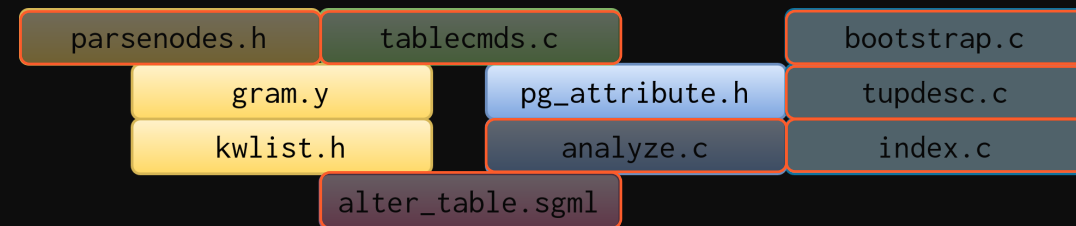


# Создадим лексему Gram.y

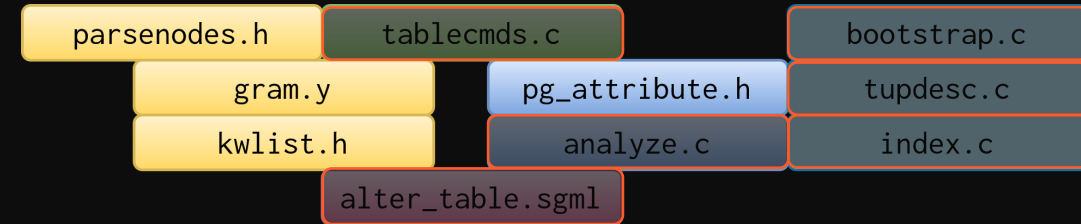
```
> | ALTER opt_column ColId SET STATMULTIPLIER SignedIconst
{
    AlterTableCmd *n = makeNode(AlterTableCmd);
    n->subtype = AT_SetStatisticsSampleMultiplierColumn;
    n->name = $3;
    n->def = (Node *) makeInteger($6);
    $$ = (Node *) n;
}
```

> Так запрос: `ALTER TABLE users ALTER COLUMN age SET STATMULTIPLIER 5;`  
будет преобразован в AST:

```
AlterTableStmt
├─ relation = RangeVar("users")
└─ cmds = List
    └─ AlterTableCmd
        ├─ subtype = AT_SetStatisticsSampleMultiplierColumn
        ├─ name = "age"
        └─ def = Integer(5)
```



# Создадим подтип в Parsenodes.h



› Обновим `typedef enum AlterTableType`

```

2181 → AT_SetStatistics, → → → /*alter.column.set.statistics*/
2182 → AT_SetStatisticsSampleMultiplierColumn, → /*alter.column.set.statistics.sample.multiplier*/
2183 → AT_SetOptions, → → → /*alter.column.set.(options)*/

```

# DDL — команда Tablecmds.c



- › 1) **Объявим** функцию  
`static ObjectAddress ATExecSetStatisticsSampleMultiplierColumn(...)`
- › 2) Подключим в `AlterTableGetLockLevel(...)`, подготовку обработки `ATPrepCmd(...)` и **общий обработчик** `ATExecCmd(...)`
- › 3) Подключим в **вывод описания** при выполнении команды `alter_table_type_to_string(...)`
- › 4) **Реализация** функции  
`ATExecSetStatisticsSampleMultiplierColumn(...)`

# Реализация функции в Tablecmds.c

## › Входные параметры:

- rel — отношение (таблица)
- colName — имя столбца
- newValue — AST-узел со значением множителя
- lockmode — режим блокировки

## › Извлечение значения

## › Проверка допустимого диапазона

## › Открытие системного каталога pg\_attribute

## › Поиск записи столбца

- › Внесение этих изменений позволяет передать новое значение в переменную `attstatssamplemultiplier`, объявленную в `pg_attribute.h` ранее.



## › Проверки валидности

## › Обновление значения

## › Вызов хука пост-изменения

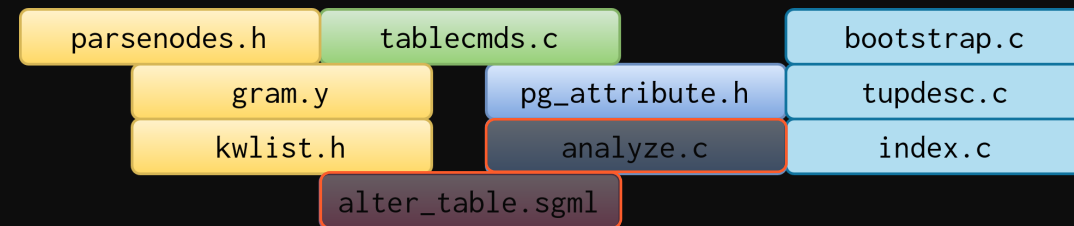
## › Возвращение адреса



# ЗАВИСИМОСТИ

› Добавим инициализацию\* в файлах

- `bootstrap.c` в функции `DefineAttr()`
- `tupdesc.c` в функциях `TupleDescInitEntry()`  
`TupleDescInitBuiltinEntry()`
- `index.c` в функции `ConstructTupleDescriptor()`

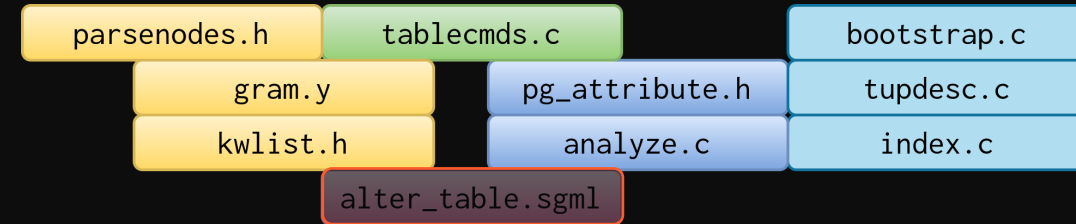


Изменения `pg_attribute` также должны быть учтены в:

- `pg_dump.c` – добавить SQL-генерацию для `statmultiplier`
- `psql/describe.c` – отобразить в `\d+`

\* Инициализация нового поля значением по умолчанию – `-1` означает "не задано", как и в реализации `attstattarget(DST)`.

# Основной функционал Analyze.c



- › В функции `std_typanalyze()`
- › Если значение SSM не установлено, считаем `SSM = 1`
- › Внедряем **SSM** во все ветки расчёта `stats->minrows`

Задание на лабораторную:

- › Дополнить код патча защитой от переполнения `stats->minrows`.
- › Дополнить код патча тестом нового функционала.

# Документация Alter\_table.sgml



- › Незадокументированный код – недописанный код
- › Добавим описание для нового параметра в SGML




# Результаты:

- › Выполнение тестового запроса при **DST = 2 500**, SSM = 1 (не установлен)

```
16 SET default_statistics_target = 2500;
17 ALTER TABLE _AccumRg50090 ALTER COLUMN _RecorderTRef SET STATMULTIPLIER 1;
18 analyze _AccumRg50090;
20 explain analyze SELECT
21   T1._Period
22 FROM _AccumRg50090 T1
23 WHERE ((T1._Fld2488 = CAST(0 AS NUMERIC))) AND ((T1._RecorderTRef = '\\000\\000\\004\\262'::bytea
24 AND T1._RecorderRRef = '\\211\\302\\030f\\332\\261R\\333\\021\\356\\336\\374\\035\\2265\\252'::bytea))
25 ORDER BY (T1._Period) LIMIT 1 ;
```

Data Output Messages Graph Visualiser X Notifications

SQL

QUERY PLAN		
	text	
1	Limit (cost=2638.77..2638.78 rows=1 width=8) (actual time=0.017..0.018 rows=1 loops=1)	
2	-> Sort (cost=2638.77..2640.20 rows=1428 width=8) (actual time=0.016..0.017 rows=1 loops=1)	
3	Sort Key: _period	
4	Sort Method: quicksort Memory: 25kB	
5	-> Index Scan using _accumrg50090_2 on _accumrg50090 t1 (cost=0.23..2624.49 rows=1428 width=8) (actual time=0.011..0.012 rows=1 loops=1)	
6	Index Cond: ((_fld2488 = '0'::numeric) AND (_recordertref = '\x000004b2'::bytea) AND (_recorderrref = '\x89c21866dab152db11eedefc1d9635a...	
7	Planning Time: 0.443 ms	
8	Execution Time: 0.031 ms	

ANALYZE

Query returned successfully in 1 min 20 secs.

	total_size text	table_size text	indexes_size text	toast_size text
1	52 GB	23 GB	29 GB	0 bytes

- › Время выполнения запроса

Planning Time: 0.443 ms

Execution Time: 0.031 ms

- › Время выполнения  
ANALYZE: 1 m 20 sec



# Инструкция по применению:

## › SSM стоит применять

- Статистика недостаточно точная при максимальном значении DST (Вычисленное значение `n_distinct` в разы отличается от реального)
- Высокий DST создает неприемлимый объем статистики

## › Как проверить

- Сравнить показатели `n_distinct` в `pg_stats` с реально вычисленными (запросом вида):  

```
SELECT COUNT(DISTINCT column_name) AS unique_count  
FROM table_name;
```
- Проверить, что при повышении DST, `n_distinct` продолжает монотонно расти



# Инструкция по применению:

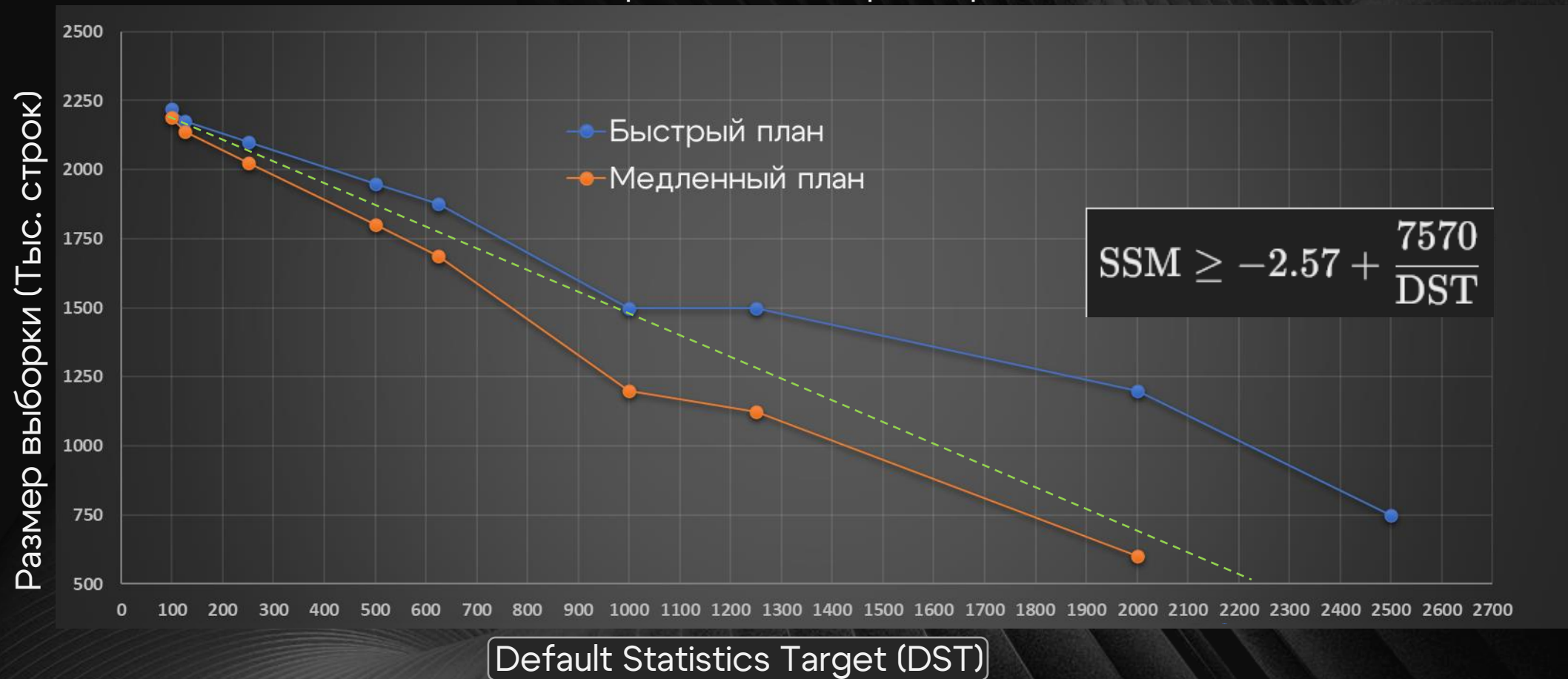
## › Порядок действий

- **Повысить DST** до достижения:
  - › Достаточной точности статистики
  - › Изменения планов проблемных запросов
- Если необходимая точность не достигнута, **повысить SSM** для исследуемой колонки таблицы
- Экспериментально подобрать соотношение SSM к DST
  - › Снизить значение DST повышая SSM для сохранения детализации статистики
  - › Учитывать, что для сохранения точности статистики, что при увеличении **пропорции SSM/DST**, необходимо **увеличивать** размер **выборки**



# Эксперименты:

Результаты тестирования:  
Зависимость выбора плана от параметров DST и SSM





# Результаты:

- › Выполнение тестового запроса при **DST = 125**, SSM = **58**

```
16 SET default_statistics_target = 125;
17 ALTER TABLE _AccumRg50090 ALTER COLUMN _RecorderTRef SET STATMULTIPLIER 58;
18 analyze _AccumRg50090;
19
20 explain analyze SELECT
21   T1._Period
22 FROM _AccumRg50090 T1
23 WHERE ((T1._Fld2488 = CAST(0 AS NUMERIC))) AND ((T1._RecorderTRef = '\\000\\000\\004\\262'::bytea
24 AND T1._RecorderRRef = '\\211\\302\\030f\\332\\261R\\333\\021\\356\\336\\374\\035\\2265\\252'::bytea))
25 ORDER BY (T1._Period) LIMIT 1 ;
--
```

Data Output Messages Notifications



	QUERY PLAN	
	text	
1	Limit (cost=2932.79..2932.79 rows=1 width=8) (actual time=0.017..0.017 rows=1 loops=1)	
2	-> Sort (cost=2932.79..2934.37 rows=1587 width=8) (actual time=0.016..0.016 rows=1 loops=1)	
3	Sort Key: _period	
4	Sort Method: quicksort Memory: 25kB	
5	-> Index Scan using _accumrg50090_2 on _accumrg50090 t1 (cost=0.23..2916.92 rows=1587 width=8) (actual time=0.011..0.012 rows=1 loops=1)	
6	Index Cond: ((_fld2488 = '0'::numeric) AND (_recordertref = '\\x000004b2'::bytea) AND (_recorderref = '\\x89c21866dab152db11eedefc1d9635a...	
7	Planning Time: 0.361 ms	
8	Execution Time: 0.028 ms	

ANALYZE

Query returned successfully in 4 min 16 secs.

	total_size text	table_size text	indexes_size text	toast_size text
1	52 GB	23 GB	29 GB	0 bytes

- › Время выполнения запроса
- › Было:

Planning Time: 0.443 ms

Execution Time: 0.031 ms

- › Стало:

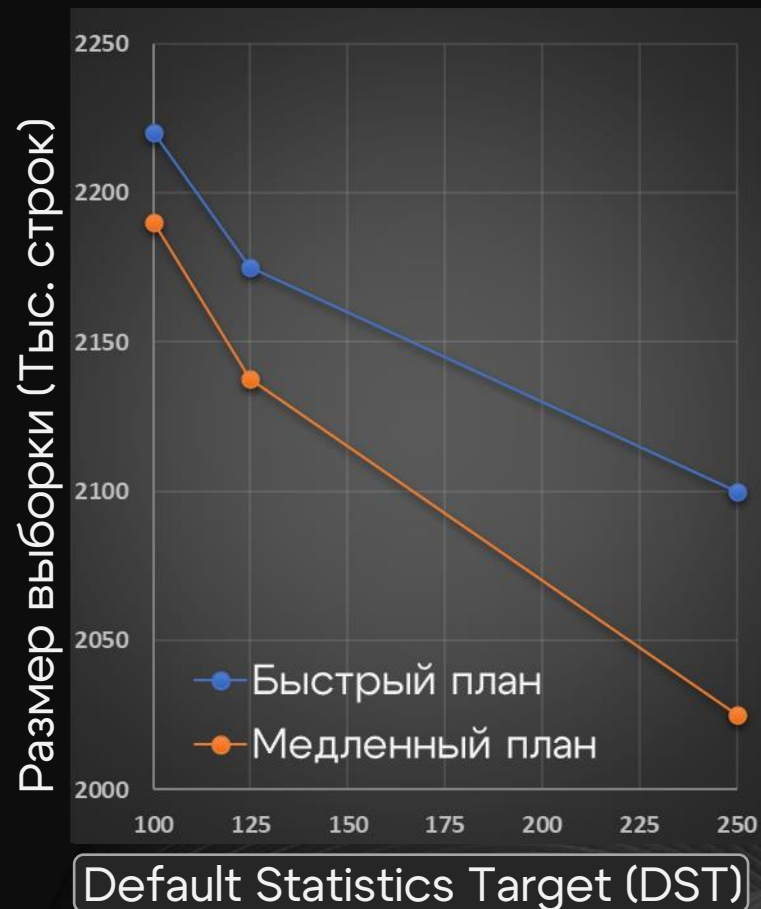
Planning Time: 0.361 ms

Execution Time: 0.028 ms

- › Время выполнения  
ANALYZE: **4 m 16 sec**
- › Выросло в **3.2** раза

# Итоги:

Результаты тестирования:



- › При **DST** = 2500 и **SSM** = 1  
количество строк для анализа: **750** тысяч
- › При **DST** = 100 и **SSM** = 74  
количество строк для анализа: **2 220** тысяч
- › Размер необходимой выборки для ANALYZE в **2.96** раз больше
- › Объем хранимой статистики в **25** раз меньше по задействованным колонкам



Спасибо  
за внимание!



[www.tantorlabs.ru](http://www.tantorlabs.ru)