1. **TOAST**

Папка задания в репозитории:

<https://github.com/artemgur/AdvancedPostgreSQL/tree/master/PostgreSQL%20Task%201/TOAST>

Создание таблиц и задание стратегии TOAST:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/TOAST/TOASTCreateTable.sql>

Генерация строк таблиц. В результате столбец text содержит строку длиной 32\*100, > 3 кБ.

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/TOAST/TOASTInsert.sql>

**Измерение размера таблиц**

Размер таблиц определялся с помощью запроса с лекции (слайд 4), из которого для удобства были убраны ненужные в данный момент столбцы.

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/TOAST/OriginalTableSize.sql>

Размеры таблиц получились следюующими:

* toast\_plain — 400 кБ (50 страниц), так как длинные строки хранятся в изначальной таблице без сжатия.
* toast\_extended — 16 кБ (2 страницы). При стратегии EXTENDED длинные строки сжимаются. Далее строки сохраняются в TOAST-таблице, только если они превышают 2 кБ даже после сжатия. В данном случае после сжатия строки становятся меньше 2 кБ, и хранятся в изначальной таблице.
* toast\_external — 8 кБ (1 страница). Строки не сжимаются и, так как они превышают 2 кБ, хранятся в TOAST-таблице.
* toast\_main — 16 кБ (2 страницы). Строки сжимаются и хранятся в изначальной таблице. Строки могут быть сохранены в TOAST-таблице, если они не помещаются на странице после сжатия, но в данном случае этого не происходит.

Таким образом, в данном случае строки таблиц toast\_extended и toast\_main хранятся одинаково — в сжатом виде в изначальной таблице.

**Измерение размера TOAST-таблиц**

Для получения информации о размере TOAST-таблиц использовался значительно модифицированный запрос с лекции:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/TOAST/TOASTTableSize.sql>

Как сказано выше, единственная TOAST-таблица, содержащая данные — таблица, соответствующая toast\_external. Ее размер — 400 кБ (такой же, как у toast\_plain), а размер остальных TOAST-таблиц равен 0.

**Измерение времени выполнения SELECT**

Изменение времени выполнения SELECT измерялось с помощью pgbench. С помощью команды:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/TOAST/TOASTPgBench.sh>

запускался скрипт:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/TOAST/PgBenchSelect.sql>

Пример полученных результатов (в миллисекундах):

* toast\_plain — 0.658. Быстрее всего, так как все нужные данные находятся в изначальной таблице в несжатом виде.
* toast\_extended — 0.701. Все нужные данные хранятся в изначальной таблице, но строки нужно преобразовать из сжатого вида. Поэтому медленнее, чем toast\_plain.
* toast\_external — 1.121. Из-за того, что строки хранятся в TOAST-таблице (но в несжатом виде), работает медленно.
* toast\_main — 0.707. Работает с той же скоростью, что toast\_extended, так как в рассматриваемом случае данные в них хранятся одинаково (см. выше).

При этом, если бы строки были еще длиннее (или менее сжимаемыми) и занимали бы больше 2 кБ даже в сжатом состоянии, toast\_extended был бы, вероятно, самым медленным, так как строки нужно было бы не только получить из другой таблицы, но и преобразовать из сжатого состояния.

**Содержимое таблиц**

Содержимое изначальных таблиц (полученное запросом вида SELECT \* FROM toast\_plain):



TOAST-таблицы, соответствующие таблицам toast\_plain, toast\_extended и toast\_main, оказались пустыми.

Содержимое TOAST-таблицы, соответствующей toast\_external (полученное запросом вида SELECT \* FROM pg\_toast.pg\_toast\_60511):



1. **FillFactor**

Папка задания в репозитории:

<https://github.com/artemgur/AdvancedPostgreSQL/tree/master/PostgreSQL%20Task%201/FillFactor>

Создание таблиц:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/FillFactor/FillFactorCreateTable.sql>

Генерация строк таблиц:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/FillFactor/FillFactorInsert.sql>

**Измерение времени выполнения UPDATE**

Время выполнения UPDATE измерялось с помощью pgbench. С помощью команды:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/FillFactor/PgBenchUpdate.sh>

запускался скрипт:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/FillFactor/PgBenchUpdate.sql>

Пример полученных результатов (в миллисекундах):

* fillfactor50 — 33.207
* fillfactor75 — 37.053
* fillfactor90 — 37.515
* fillfactor100 — 38.133

Таким образом, чем меньше fill factor, тем быстрее происходит UPDATE. Это объясняется тем, что, чем меньше fill factor, тем больше свободного места в уже имеющихся страницах может быть использовано, и тем меньше новых страниц нужно создать.

**Измерение размера таблиц**

Для измерения страниц использовался следующий запрос (такой же, как использовался для измерения размера таблиц в задании про TOAST):

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/FillFactor/TableSize.sql>

Рассматривалась 2 сценария: с “чистой” таблицей — сразу после ее первоначального заполнения, и с “грязной” таблицей — после обновления строк таблицы.

В случае “чистых” таблиц были получены следующие результаты:

* fillfactor50 — 1824 кБ (228 страниц)
* fillfactor75 — 1216 кБ (152 страницы)
* fillfactor90 — 1016 кБ (127 страниц)
* fillfactor100 — 912 кБ (114 страниц)

В случае “грязных” таблиц были получены следующие результаты:

* fillfactor50 — 2968 кБ (371 страница)
* fillfactor75 — 2752 кБ (344 страниц)
* fillfactor90 — 2528 кБ (316 страниц)
* fillfactor100 — 2368 кБ (296 страниц)

Таким образом, чем больше fill factor, тем меньше страниц занимает таблица. Но после UPDATE разница в количестве страниц уменьшается.

**Измерение времени выполнения SELECT**

Время выполнения SELECT измерялось с помощью pgbench. Рассматривалась 2 сценария: с “чистой” таблицей:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/FillFactor/PgBenchSelectCleanTables.sh>

и с “грязной” таблицей (флагом pgbench был отключен предварительный VACUUM):

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/FillFactor/PgBenchSelectDirtyTables.sh>

В обоих случаях измерялось время выполнения скрипта:

<https://github.com/artemgur/AdvancedPostgreSQL/blob/master/PostgreSQL%20Task%201/FillFactor/PgBenchSelect.sql>

Пример полученных результатов для “чистой” таблицы (в миллисекундах):

* fillfactor50 — 1.413
* fillfactor75 — 1.365
* fillfactor90 — 1.355
* fillfactor100 — 1.346

Пример полученных результатов для “грязной” таблицы (в миллисекундах):

* fillfactor50 — 1.709
* fillfactor75 — 1.689
* fillfactor90 — 1.668
* fillfactor100 — 1.635

В случае “грязных” таблиц SELECT в среднем выполнялся дольше.

Таким образом, чем больше fill factor, тем быстрее выполняется SELECT. Это связано с тем, что, чем меньше fill factor, тем по большему числу страниц распределены строки таблицы. Поэтому, чем меньше fill factor, тем больше времени уходит, как минимум, на открытие и закрытие файлов.