УТВЕРЖДАЮ

Генеральный директор ООО «ППГ Разработка»

\_\_\_\_\_\_\_\_\_\_\_ В.В. Попов

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Отчет

по результатам аудита состояния БД ib\_rrko кластера

СУБД PostgreSQL

Версия №1 от 19.10.2022г.

на 18 листах

## **Оглавление**

[**Оглавление** 2](#_Toc1)

[1. **Цели и задачи аудита** 3](#_Toc2)

[2. **Анализ используемых программных и аппаратных средств** 4](#_Toc3)

[2.1. **Анализ используемых аппаратных средств** 4](#_Toc4)

[2.2. **Анализ используемых программных средств** 5](#_Toc5)

[3. **Общие выводы** 6](#_Toc6)

[4. **Рекомендации по оптимизации запросов** 7](#_Toc7)

[5. **Анализ индексов** 12](#_Toc8)

[6.  **Рекомендации по конфигурации параметров операционной системы** 13](#_Toc9)

[6.1. **Настройка HugePages** 13](#_Toc10)

[6.2. **Настройка Transparent Hugepage** 13](#_Toc11)

[7.  **Поиск разросшихся таблиц и индексов** 15](#_Toc12)

[8.  **Поиск large object** 17](#_Toc13)

[9.  **Конфигурация СУБД PostgreSQL** 18](#_Toc14)

[9.1. **Настройка фонового процесса записи** 18](#_Toc15)

[9.2. **Настройка процесса автоматической очистки** 18](#_Toc16)

[9.3. **Дополнительные параметры по настройке СУБД** 18](#_Toc17)

## **Цели и задачи аудита**

Целью проводимых работ является выдача рекомендаций по конфигурации операционной системы, СУБД Postgresql и ответы на вопросы заказчика.

## **Анализ используемых программных и аппаратных средств**

### **Анализ используемых аппаратных средств**

Ниже приведены характеристики сервера **p0dbop-pg5001lp.region.vtb.ru** с СУБД PostgreSQL, являющегося мастером от 01.09.2022:

|  |  |
| --- | --- |
| **Сервер** | **p0dbop-pg5001lp.region.vtb.ru** |
| Роль сервера | Мастер |
| Тип сервера | Виртуальный сервер |
| Операционная система | Red Hat Enterprise Linux Server release 7.9 (Maipo) 3.10.0-1160.42.2.el7.x86\_64 |
| CPU (ядро) | Кол-во сокетов: 2  Кол-во ядер на сокет: 24  Кол-во трэдов на ядро: 2  Идентификатор вендора: GenuineIntel  Наименование модели: Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz  NUMA Nodes: 2 |
| RAM | 503GB |
| Диск | Filesystem Type Size Used Avail Use% Mounted on  devtmpfs devtmpfs 252G 0 252G 0% **/**dev  tmpfs tmpfs 252G 1.4M 252G 1% **/**dev**/**shm  tmpfs tmpfs 252G 2.2G 250G 1% **/**run  tmpfs tmpfs 252G 0 252G 0% **/**sys**/**fs**/**cgroup  **/**dev**/**mapper**/**vg\_01**-**lv\_root xfs 10G 3.2G 6.9G 32% **/**  **/**dev**/**mapper**/**vg\_01**-**lv\_home xfs 5.0G 38M 5.0G 1% **/**home  **/**dev**/**mapper**/**vg\_p0dbop\_pg5001lp\_pgdata**-**lv\_pgdata xfs 4.0T 506G 3.5T 13% **/**pg\_data  **/**dev**/**mapper**/**vg\_01**-**lv\_pgaudit xfs 200G 564M 200G 1% **/**pg\_audit  **/**dev**/**mapper**/**vg\_01**-**lv\_tmp xfs 3.0G 34M 3.0G 2% **/**tmp  **/**dev**/**sda2 xfs 506M 183M 323M 37% **/**boot  **/**dev**/**mapper**/**vg\_01**-**lv\_etcd xfs 15G 403M 15G 3% **/**app**/**etcd  **/**dev**/**mapper**/**vg\_01**-**lv\_var xfs 8.0G 1.8G 6.2G 23% **/**var  **/**dev**/**mapper**/**vg\_p0dbop\_pg5001lp\_pgbackup**-**lv\_pgbackup xfs 4.0T 274G 3.7T 7% **/**pg\_backup  **/**dev**/**sda1 vfat 200M 9.9M 190M 5% **/**boot**/**efi  tmpfs tmpfs 51G 0 51G 0% **/**run**/**user**/**653  **/**dev**/**mapper**/**vg\_pgwal**-**lv\_pgwal xfs 2.0T 24G 2.0T 2% **/**pg\_walarchive  tmpfs tmpfs 51G 0 51G 0% **/**run**/**user**/**1376548563 |
| Директория данных СУБД | /pg\_data |

**Таблица 2.1.1. Характеристики сервера** **p0dbop-pg5001lp.region.vtb.ru**

### **Анализ используемых программных средств**

СУБД работает под управлением ПО patroni 2.1.2, в таблице ниже представлены используемая на сервере версия от 01.09.2022:

|  |  |
| --- | --- |
| Сервер | Версия |
| Мастер  (**p0dbop-pg5001lp.region.vtb.ru**) | PostgreSQL 11.14 on x86\_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44), 64-bit |

**Таблица 2.2.1. Используемая версия СУБД**

В настоящий момент СУБД обновлена до версии 13.

## **Общие выводы**

Выдан ряд рекомендаций по:

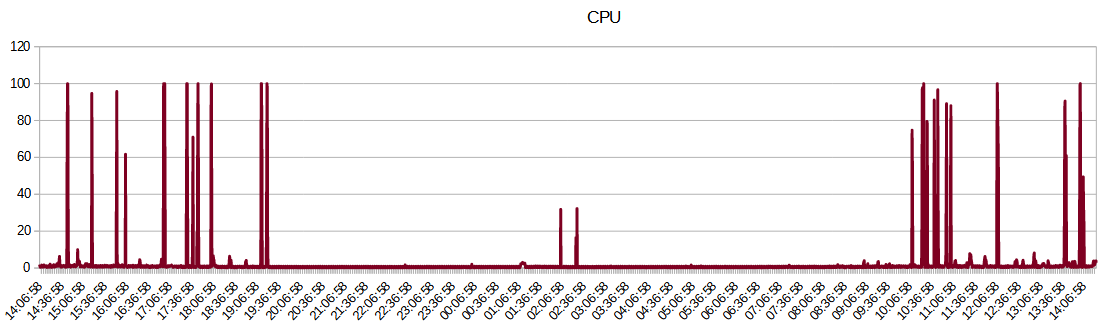
1. Оптимизации работы запросов.
2. Настройке параметров операционной системы.
3. Настройке параметров СУБД PostgreSQL.

Наиболее проблемные запросы относятся к таблице **rrko.rrko\_ru\_payment**, среднее время их выполнения от 772.632 мс. Нужно:

1. Разобраться с индексным составом этой таблицы, удалить дубликаты и неэффективные частичные индексы.
2. Определить, какое из полей client\_ext\_id или status\_base позволяет отфильтровать большее количество строк и при каких случаях.
3. По возможности заменить запросы с фильтрацией по полям payer\_account и payer\_bank\_bic так, чтобы они покрывались индексами, а не Filter узла Bitmap Heap Scan
4. Учесть, что работа с большими списками значений оператора IN улучшена в 14-ой версии СУБД PostgreSQL.
5. Нужны планы и параметры выполнения запросов для определения полей фильтрации большего числа строк. Это позволит создать новые индексы для эффективной фильтрации строк.

Рекомендуется провести повторный анализ после проведения работ согласно данным рекомендациям.

Также требуется сравнить графики нагрузки на CPU до и после оптимизации. Ниже приведён график потребления CPU за день до оптимизации:



**Рис 3.1. График потребления CPU до оптимизации**

## **Рекомендации по оптимизации запросов**

1. Обнаружен запрос удаления данных таблицы **rrko.rrko\_doc\_signer**, поиск строк осуществляется последовательным сканированием. Минимальное время выполнения 38.187 мс, максимальное время 98.080 мс, среднее время выполнения 42.181 мс, число выполнений в час 745

Текст запроса представлен в файле **rrko\_doc\_signer\_del.sql**.

Для ускорения доступа к данным таблицы **rrko.rrko\_doc\_signer** предлагается создать следующие индексы:

**CREATE** **INDEX** rds\_user\_id\_doc\_id\_doc\_type\_id\_ix

**ON** rrko**.**rrko\_doc\_signer**(user\_id,** doc\_id**,** doc\_type\_id**);**

**Листинг 4.1. Индекс для ускорения запроса rrko\_doc\_signer\_del**

1. Обнаружен запрос соединения следующих таблиц с использованием cross join:

* rrko\_pclib\_business\_object.
* rrko\_pclib\_business\_service.
* rrko\_pclib\_business\_operation

Текст запроса представлен в файле **rrko\_pclib\_business\_object\_service\_operation.sql**

cross join здесь не нужен, его нужно заменить на join. Также благодаря условиям фильтрации и соединения можно не использовать таблицу rrko\_pclib\_business\_object

Предлагаемый вариант запроса находится в файле

**sql/proposals/rrko\_pclib\_business\_object\_service\_operation.sql**

1. Обнаружен запрос удаления данных из таблицы rrko.rrko\_ru\_payment\_action\_entity, минимальное время выполнения запроса 0.004 мс, максимальное время выполнения 3172.451 мс, среднее время выполнения 7.322 мс. Текст запроса представлен в файле **sql/rrko\_ru\_payment\_action\_entity.sql**

В процессе поиска данных используются следующие поля:

* ru\_payment\_id
* point\_of\_view
* actual

При этом нет индекса с комбинацией этих полей, поэтому его предлагается создать:

**CREATE** **INDEX** rpae\_rpid\_pv\_actual\_ix

**ON** rrko**.**rrko\_ru\_payment\_action\_entity**(**ru\_payment\_id**,** point\_of\_view**,** actual**);**

**Листинг 4.2. Индекс для ускорения работы запроса rrko\_ru\_payment\_action\_entity**

1. Обнаружен запрос определения количества строк таблицы rrko.rrko\_ru\_payment. Минимальное время выполнения 1.613 мс, максимальное 4421.432 мс, среднее 772.632 мс**.** Текст запроса представлен в файле **sql/rrko\_ru\_payment\_cnt.sql**

Возможно, есть случаи, когда фильтрация происходит только по полю status\_base. В этом случае может пригодиться следующий индекс.

**CREATE** **INDEX** ru\_payment\_status\_base\_delete\_date\_idx

**ON** rrko**.**rrko\_ru\_payment**(**status\_base**,** delete\_date**);**

**Листинг 4.3. Индекс для поиска по полям status\_base и delete\_date**

При этом не ясна цель индекса из листинга 4.4, поскольку при delete\_date IS NULL нет смысла индексировать поле delete\_date.

"ru\_payment\_client\_ext\_id\_status\_base\_delete\_date\_idx"

btree **(**client\_ext\_id**,** status\_base**,** delete\_date**)**

**WHERE** delete\_date **IS** **NULL**

**Листинг 4.4. Индекс с NULL-значениями поля delete\_date**

Нужно определить параметры запросов, при которых время выполнения превышает 500 мс. Для этого можно использовать параметр **auto\_explain.log\_min\_duration** модуля **auto\_explain**. При этом **auto\_explain** должно содержаться в параметре **shared\_preload\_libraries**.

После этого определить поля, позволяющие отфильтровать большое количество строк. Если для этих полей не существует индекса, то его надо создать.

Также нужно учесть, что работа с большим списком значений по оператору IN улучшена в 14-ой версии СУБД PostgreSQL

1. Обнаружен запрос последовательного сканирования таблицы rrko\_ru\_payment по полям import\_date и create\_date. Минимальное время выполнения 34202.551 мс, максимальное время выполнения 51198.428 мс, среднее время выполнения 35661.757 мс, число выполнений в час 21.

Текст запроса представлен в файле **sql/rrko\_ru\_payment\_cnt\_monitor.sql**

Для ускорения доступа к данным предлагается создать следующий индекс:

**CREATE** **INDEX** rrp\_cr\_dt\_ix

**ON** rrko**.**rrko\_ru\_payment**(**create\_date**);**

**CREATE** **INDEX** rrp\_im\_dt\_ix

**ON** rrko**.**rrko\_ru\_payment**(**import\_date**);**

**Листинг 4.5. Индексы для ускорения поиска по полям create\_date и import\_date**

1. Обнаружены запросы определения количества строк таблицы rrko\_ru\_payment, при этом используются следующие поля:

* client\_ext\_id
* role\_template\_signature
* payer\_account
* payer\_bank\_bic
* role\_template\_signature
* visible\_for\_acceptor
* status\_base

Кроме того,есть поиск по таблицам **rrko\_doc\_sign** и **rrko\_doc\_signer**

Минимальное время выполнения 570.041 мс, максимальное время 1172.955 мс, среднее время 780.542 мс, кол-во выполнений в час 52.

Текст запроса представлен в файле **sql/rrko\_ru\_payment\_doc\_signer.sql**

Фактически, поиск по таблице rrko\_ru\_payment состоит из следующих веток:

1. Первая ветка:
   1. Проверка role\_template\_signature на null.
   2. Поиск значений client\_ext\_id в списке значений.
   3. Поиск строк по условиям ниже, соединённых между собой оператором or.

rupayment0\_**.**payer\_account **=** $38 **and**

rupayment0\_**.**payer\_bank\_bic **=** $39

**Листинг 4.6. Условия поиска по** **payer\_account** и **payer\_bank\_bic**

1. Вторая ветка:
   1. Проверка role\_template\_signature на null.
   2. Поиск строк по полю visible\_for\_acceptor.
   3. Поиск значений client\_ext\_id в списке значений.
   4. Поиск строк по условиям из листинга 4.6, соединённых между собой оператором or.
2. Третья ветка:
   1. Проверка role\_template\_signature на null.
   2. Поиск значений client\_ext\_id в списке значений.

После этого поиск происходит поиск строк на основе значений полей **status\_base** и **status\_extended**.

На основе этого можно сказать, что поиск по индексу будет только по полю ext\_id, при этом будет использовать метод Bitmap Index Scan. После этого найденные идентификаторы строк будут объединены в битовую карту, а затем каждая строка будет проверяться на удовлетворение значений фильтров при выполнении Bitmap heap Scan.

При проверке соответствия строк в rrko\_doc\_signer будут использоваться SubPlan из-за объединения двух условий exists оператором or.

Для определения направления оптимизации запроса нужно:

1. Определить параметры выполнения.
2. На основе пункта 1 определить план выполнения.
3. Определить поля, позволяющие отфильтровать наибольшее количество строк.
4. Скорректировать индексный состав (см раздел 5).
5. Учесть, что поиск по большому списку значений оператором IN улучшен в 14-ой версии СУБД PostgreSQL.
6. Учитывать, что при замене OR на UNION придётся использовать native query
7. Обнаружен запрос поиска данных по таблице rrko.rrko\_ru\_payment с последующим соединением таблицы rrko\_edoc\_ref. При этом в качестве фильтра используется поле sign\_bsk таблицы rrko\_ru\_payment.

Минимальное время выполнения 35068.031 мс, максимальное время выполнения 35745.709 мс, среднее время выполнения 35352.984 мс, кол-во вызовов 4.

Текст запроса представлен в файле

**sql/rrko\_ru\_payment\_edoc\_ref\_import\_session\_id.sql**

Здесь также нужны параметры выполнения запроса и план выполнения, поскольку с их помощью можно выявить комбинацию полей для наиболее эффективной фильтрации строк.

Также обнаружены следующие условия фильтрации:

**and** **(**rupayment0\_**.**import\_session\_id **is** **null** **or** rupayment0\_**.**channel **=** $32 **or** rupayment0\_**.**status\_base **<>** $33**)**

**and** **(**rupayment0\_**.**import\_session\_id **is** **not** **null** **or** rupayment0\_**.**status\_base **<>** $34**)**

**Листинг 4.6. Часть условий фильтрации строк таблицы rrko\_ru\_payment**

Скорее всего, их можно заменить на следующие условия:

rupayment0\_**.**import\_session\_id **is** **null** **and** rupayment0\_**.**status\_base **<>** $34 **or**

rupayment0\_**.**channel **=** $32 **or**

rupayment0\_**.**status\_base **<>** $33

**Листинг 4.7. Вариант замены условий из листинга 4.6.**

1. Обнаружен запрос расчёта количества строк таблицы rrko\_ru\_payment с использованием полей client\_ext\_id, payer\_account, payer\_bank\_bic, status\_base.

Минимальное время работы запроса 615.403 мс, максимальное время работы 2868.343 мс, среднее время выполнения 1490.985 мс, число выполнений в час 131.

Текст запроса представлен в файле

**sql/rrko\_ru\_payment\_payer\_account\_bank\_bic\_cnt.sql**.

Поиск по индексу будет только по полю ext\_id, при этом будет использовать метод Bitmap Index Scan. После этого найденные идентификаторы строк будут объединены в битовую карту, а затем каждая строка будет проверяться на удовлетворение значений фильтров при выполнении Bitmap Heap Scan.

Рекомендации аналогичны пункту 6.

1. Обнаружен запрос извлечения данных таблицы rrko\_ru\_payment, условия фильтрации сходи с пунктом 8. Минимальное время выполнения запроса 6615.333 мс, максимальное время выполнения 18845.809 мс, среднее время выполнения 14752 мс, число выполнений в час 131.

Текст запроса представлен в файле:

**sql/rrko\_ru\_payment\_payer\_bank\_bic\_account.sql**

Рекомендации аналогичны пункту 6.

1. Обнаружен запрос расчёта количества строк таблицы rrko\_ru\_payment с использованием полей sign\_bsk, branch\_ext\_id, status\_base, status\_extended. Текст запроса представлен в файле **sql/rrko\_ru\_payment\_sign\_bsk.sql**

Возможно, нужны индексы по полям **sign\_bsk, branch\_ext\_id, status\_base, status\_extended** и **status\_base, status\_extended** соответственно.

1. Обнаружены запросы поиска данных таблицы **rrko\_ru\_payment** по полям **client\_ext\_id** и **status\_base**. Минимальное время выполнения 7.007 мс, максимальное время 59514.798 мс, среднее время выполнения 28523.718 мс, число выполнений в час 339.

Текст запроса представлен в файле **sql/rrko\_ru\_payment\_simple.sql**

При большом offset запрос будет работать медленно, поскольку придётся извлечь большой набор данных, отфильтровать его, а затем пропустить большую его часть.

Рекомендации аналогичны пункту 6.

## **Анализ индексов**

В ходе анализа схемы данных обнаружены многочисленные дубликаты индексов, что замедляет операции изменения, добавления и удаления данных. Список представлен в **ib\_rrko\_dupl\_idx.csv**. Их нужно будет удалить в технологическое окно.

Также обнаружены следующие особенности:

1. В файле s**ql/rrko\_ru\_payment\_idx\_list.sql** перечислены частичные индексы. Однако их условия нужно проверить на эффективность. Для нужно выполнить запрос в файле **sql/rrko\_ru\_payment\_idx\_fcond\_stats.sql**. Если полученные количества строк будут близки к общему количеству строк таблицы, то такие индексы не позволят эффективно фильтровать строки.
2. В таблице **rrko.rrko\_balance\_snapshot** обнаружены следующие индексы:

"rrko\_account\_balance\_snapshot\_id\_blocked\_idx" **UNIQUE,** btree **(**id**)** **WHERE** is\_blocked **=** **true**

"rrko\_account\_balance\_snapshot\_id\_not\_blocked\_idx" **UNIQUE,** btree **(**id**)** **WHERE** **NOT** is\_blocked **=** **true**

**Листинг 5.1. Частичные индексы по первичному ключу**

Поскольку id является первичным ключом, то частичные индексы не позволят ускорить выполнение запроса.

Предлагается их удалить.

1. Для таблицы **rrko.rrko\_aclib\_client\_master** предлагается следующий индексный состав:

"rrko\_aclib\_client\_master\_id\_pk" **PRIMARY** **KEY,** btree **(**id**)**

"rrko\_aclib\_client\_id\_client\_master\_id\_ux" **UNIQUE,** btree **(**client\_id**,** client\_master\_id**)**

"rrko\_aclib\_client\_master\_id\_client\_id\_idx" **UNIQUE,** btree **(**client\_master\_id**,** client\_id**)**

**Листинг 5.2. Индексный состав таблицы rrko.rrko\_aclib\_client\_master**

## **Рекомендации по конфигурации параметров операционной системы**

### **Настройка HugePages**

Анализ значений файла /proc/meminfo от 01.09.2022 показал, что 1.1GB системы тратится на PageTables, т.е, на таблицы соответствия страниц виртуальной и физической памяти:

PageTables: 1161396 kB

**Листинг 6.1.1. Значение поля PageTables**

Для уменьшения потребления памяти рекомендуется активировать HugePages согласно следующей формуле:

num\_pages = 1.1 **\*** shared\_buffers\_in\_kb **/** hugepage\_size

**Листинг 6.1.2. Формула расчёта количества HugePages**

**shared\_buffers\_in\_kb** – объём памяти для буферного кэша СУБД PostgreSQL в килобайтах

**hugepage\_size** = 2048kB или 1GB. В данном случае, 2048kB

Таким образом, значение для num\_pages при **shared\_buffers = 125GB** равно 70400

Для активации HugePages после перезапуска системы необходимо сохранить вычисленное значение в файле **/etc/systcl.conf** как **vm.nr\_hugepages = num\_pages,** затем применить sysctl -p /ect/sysctl.conf.

Стоит отметить, что HugePages не выгружаются в swap

### **Настройка Transparent Hugepage**

Анализ значений файла /proc/meminfo от 01.09.2022 показал использование Transparent Hugepage.

AnonHugePages**:** 161792 kB

**Листинг 6.2.1. Значение поля AnonHugePages**

Для отключения THP нужно выполнить следующие команды пользователем root:

echo never **>** **/**sys**/**kernel**/**mm**/**transparent\_hugepage**/**enabled

echo never **>** **/**sys**/**kernel**/**mm**/**transparent\_hugepage**/**defrag

**Листинг 6.2.2. Команды отключения THP**

Также для отключения THP необходимо добавить **transparent\_hugepage=never** в конец опции GRUB\_CMDLINE\_LINUX в файл /etc/default/grub.

После этого от пользователя root нужно выполнить:

grub2-mkconfig -o /boot/grub2/grub.cfg

Это позволит сохранить настройку даже после перезагрузки сервера.

Далее нужно перезапустить СУБД PostgreSQL для последующего использования выделенных HugePages.

## **Поиск разросшихся таблиц и индексов**

Разрастание таблиц и индексов приводит к замедлению скорости поиска данных в них и, как следствие, к замедлению скорости всей системы.

Кроме того, избыточное пространство в индексах замедляет работу процесса автоматической очистки, поскольку последнему нужно пройти каждый блок каждого индекса таблицы.

Для поиска таких объектов использовались запросы **bloated\_tables.sql** и **bloated\_indexes.sql из директории sql**.

Ниже приведена статистика по проценту разрастания таблицы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Схема | Таблица | Размер | Избыточный размер | Процент избыточности |
| rup\_oc | rup\_oc\_ru\_payment\_snapshot | 12 GB | 5831 MB | 46.24 |
| rrko | rrko\_consumed\_message | 1305 MB | 463 MB | 35.47 |
| rrko | rrko\_batch\_doc\_sign | 119 MB | 70 MB | 58.68 |
| rrko | rrko\_revinfo | 969 MB | 531 MB | 54.81 |
| rrko | rrko\_user\_changed\_document | 302 MB | 158 MB | 52.28 |
| rup\_oc | rup\_oc\_batch\_notification\_sms\_outbox\_entity | 631 MB | 226 MB | 35.91 |

**Таблица 7.1. Список таблиц с наибольшим процентом избыточного места**

Ниже приведена статистика по проценту разрастания индексов:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Схема | Таблица | Индекс | Размер | Избыточный размер | Процент избыточности |
| rrko | rrko\_history\_entry | rrko\_pk\_history\_entry\_idx | 2460 MB | 2304 MB | 93.66 |
| rrko | rrko\_history\_entry | rrko\_entity\_id\_type\_id\_history\_entry\_idx | 2097 MB | 1879 MB | 89.61 |
| rrko | rrko\_history\_entry | rrko\_history\_entry\_entity\_id\_type\_id\_11\_idx | 1383 MB | 1229 MB | 88.86 |
| rrko | rrko\_ru\_payment\_action\_entity | rrko\_ru\_payment\_action\_entity\_action\_id\_idx | 5406 MB | 4613 MB | 85.34 |
| rrko | rrko\_ru\_payment\_action\_entity | rrko\_ru\_payment\_action\_entity\_pk | 5296 MB | 4504 MB | 85.04 |
| rrko | rrko\_ru\_payment\_action\_entity | rrko\_ru\_payment\_action\_entity\_id\_idx | 2335 MB | 1895 MB | 81.17 |
| rrko | rrko\_rupayment\_onchange\_request | rrko\_rupayment\_onchange\_request\_create\_date\_idx | 1210 MB | 700 MB | 57.89 |
| rrko | rrko\_cut\_off\_time\_data | cot\_entity\_id\_idx | 2710 MB | 943 MB | 34.79 |

**Таблица 7.2. Список индексов с наибольшим процентом избыточного места**

Рекомендуется перестроить указанные выше индексы в период минимальной нагрузки на систему, например, с помощью pg\_repack.

## **Поиск large object**

На пустой схеме определить некорректное использование large object невозможно, это нужно делать в период минимальной нагрузки следующим образом:

1. Найти поля типа text
2. Найти значения полей из шага 1, в которых хранятся только цифры.
3. Сверить значения из шага 2 с полем loid таблицы pg\_largeobject
4. Проверить по коду приложения использование LargeObjectManager для принятия решения о корректности использования типа large object.

Нужно помнить, что тип поля для large object должен быть oid или loid

## **Конфигурация СУБД PostgreSQL**

### **Настройка фонового процесса записи**

В таблице ниже приведены текущие и рекомендуемые настройки конфигурации фонового процесса записи от 01.09.2022 исходя из текущих серверных мощностей.

|  |  |  |
| --- | --- | --- |
| Параметр | Текущее значение | Рекомендуемое значение |
| bgwriter\_lru\_maxpages | 100 | 4000 |
| bgwriter\_lru\_multiplier | 2 | 4 |

**Таблица 9.1.1. Текущие и рекомендуемые настройки конфигурации фонового процесса записи**

### **Настройка процесса автоматической очистки**

В таблице ниже приведены текущие и рекомендуемые настройки конфигурации процесса автоматической очистки от 01.09.2022 исходя из текущих серверных мощностей.

|  |  |  |
| --- | --- | --- |
| Параметр | Текущее значение | Рекомендуемое значение |
| autovacuum\_naptime | 1min | 30s |
| autovacuum\_vacuum\_cost\_limit | -1 | 3200 |

**Таблица 9.2.1. Текущие и рекомендуемые настройки конфигурации процесса автоматической очистки**

### **Дополнительные параметры по настройке СУБД**

Ниже приведены значения параметров СУБД от 01.09.2022 и рекомендации по их изменению.

|  |  |  |
| --- | --- | --- |
| Параметр | Текущее значение | Рекомендуемое значение |
| default\_statistics\_target | 100 | 800 |
| join\_collapse\_limit | 8 | 20 |
| from\_collapse\_limit | 8 | 20 |

**Таблица 9.3.1.** **Текущие и рекомендуемые параметры работы планировщика**