# Arenadata™ Database

*Версия - v6.18.2_arenadata30*

## Руководство пользователя по работе с кластером ADB

*2021 Arenadata™*

# Оглавление

**1 Общие сведения** **3**

**2 Хранение информации и основные операции в Arenadata DB** **7**
2.1 Сравнение хранилищ типа “heap” и “append-optimized” . . . 7
2.2 Хранение данных по столбцам и строкам . . . 7
2.3 Сжатие . . . 7
2.4 Распределение . . . 8
2.5 Управление памятью . . . 8
2.6 Секционирование . . . 8
2.7 Индексы . . . 9
2.8 Очереди ресурсов . . . 9
2.9 COPY . . . 9

**3 Поддерживаемые сторонние клиентские приложения** **11**
3.1 Клиентские приложения . . . 11
3.2 Подключение с psql . . . 12
3.3 pgAdmin III для ADB . . . 12
3.4 Установка pgAdmin III . . . 12
3.5 Просмотр графического плана запроса . . . 12
3.6 Интерфейсы приложений баз данных . . . 13
3.7 Инструменты сторонних клиентов . . . 13

Руководство описывает основные операции для работы персонала в ADB.

Руководство может быть полезно администраторам, программистам, разработчикам и сотрудникам подразделений информационных технологий, осуществляющих сопровождение кластера.

**Important:** Контактная информация службы поддержки – e-mail: info@arenadata.io

# Глава 1

# Общие сведения

База данных **ADB** – это аналитическая база данных, использующая принцип MPP shared nothing. Эта модель существенно отличается от транзакционной базы данных, работающей по принципу SMP. **ADB** лучше всего работает с денормализованным схемным дизайном, подходящим для аналитической обработки MPP. Например, *Star* или *Snowflake*, с большими таблицами фактов и таблицами меньших размеров.

**Important:** Необходимо использовать те же типы данных для столбцов, которые используются в соединениях между таблицами

В базе данных **ADB** доступны следующие команды SQL:

ABORT

ALTER AGGREGATE

ALTER CONVERSION

ALTER DATABASE

ALTER DOMAIN

ALTER EXTENSION

ALTER EXTERNAL TABLE

ALTER FILESPACE

ALTER FUNCTION

ALTER GROUP

ALTER INDEX

ALTER LANGUAGE

ALTER OPERATOR

ALTER OPERATOR CLASS

ALTER OPERATOR FAMILY

ALTER PROTOCOL

ALTER RESOURCE GROUP

ALTER RESOURCE QUEUE

ALTER ROLE
ALTER SCHEMA
ALTER SEQUENCE
ALTER TABLE
ALTER TABLESPACE
ALTER TYPE
ALTER USER
ALTER VIEW
ANALYZE
BEGIN
CHECKPOINT
CLOSE
CLUSTER
COMMENT
COMMIT
COPY
CREATE AGGREGATE
CREATE CAST
CREATE CONVERSION
CREATE DATABASE
CREATE DOMAIN
CREATE EXTENSION
CREATE EXTERNAL TABLE
CREATE FUNCTION
CREATE GROUP
CREATE INDEX
CREATE LANGUAGE
CREATE OPERATOR
CREATE OPERATOR CLASS
CREATE OPERATOR FAMILY
CREATE PROTOCOL
CREATE RESOURCE GROUP
CREATE RESOURCE QUEUE
CREATE ROLE
CREATE RULE
CREATE SCHEMA

CREATE SEQUENCE
CREATE TABLE
CREATE TABLE AS
CREATE TABLESPACE
CREATE TYPE
CREATE USER
CREATE VIEW
DEALLOCATE
DECLARE
DELETE
DISCARD
DO
DROP AGGREGATE
DROP CAST
DROP CONVERSION
DROP DATABASE
DROP DOMAIN
DROP EXTENSION
DROP EXTERNAL TABLE
DROP FILESPACE
DROP FUNCTION
DROP GROUP
DROP INDEX
DROP LANGUAGE
DROP OPERATOR
DROP OPERATOR CLASS
DROP OPERATOR FAMILY
DROP OWNED
DROP PROTOCOL
DROP RESOURCE GROUP
DROP RESOURCE QUEUE
DROP ROLE
DROP RULE
DROP SCHEMA
DROP SEQUENCE
DROP TABLE

DROP TABLESPACE

DROP TYPE

DROP USER

DROP VIEW

END

EXECUTE

EXPLAIN

FETCH

GRANT

INSERT

LOAD

LOCK

MOVE

PREPARE

REASSIGN OWNED

REINDEX

RELEASE SAVEPOINT

RESET

REVOKE

ROLLBACK

ROLLBACK TO SAVEPOINT

SAVEPOINT

SELECT

SELECT INTO

SET

SET ROLE

SET SESSION AUTHORIZATION

SET TRANSACTION

SHOW

START TRANSACTION

TRUNCATE

UPDATE

VACUUM

VALUES

# Глава 2

# Хранение информации и основные операции в Arenadata DB

## 2.1 Сравнение хранилищ типа “heap” и “append-optimized”

Существуют два типа хранения данных в таблицах – *heap* и *append-optimized*.

Хранение типа *heap* следует использовать для таблиц и разделов, которые получают итеративные пакетные, однострочные, а также синхронные операции UPDATE, DELETE и INSERT.

Хранение *append-optimized* следует использовать для таблиц и разделов, которые редко обновляются после начальной загрузки и в которых последующие вставки выполняются только в больших пакетных операциях.

**Important:** Никогда не выполнять однострочные операции INSERT, UPDATE или DELETE в таблицах типа append-optimized

**Important:** Никогда не выполнять параллельные пакетные операции UPDATE или DELETE в append-optimized (это не касается одновременных операций вставки INSERT)

## 2.2 Хранение данных по столбцам и строкам

Построчное хранение необходимо использовать для рабочих нагрузок с итеративными транзакциями, в которых требуются обновления и частые вставки. Также его следует использовать для общих целей или в случае, если рабочие нагрузки носят смешанный характер.

Хранить данные по столбцам стоит в том случае, когда запросы необходимы для поиска узкого спектра данных. Также хранение по столбцам подходит при наличии отдельных столбцов, которые регулярно обновляются без изменения других столбцов в строке.

## 2.3 Сжатие

Для улучшения ввода-вывода в системе следует использовать сжатие для *append-optimized* таблиц. Параметры сжатия столбцов следует установить на уровне, на котором находятся данные. Рекомендуется балансировать более высокий уровень сжатия с временем и циклами ЦП, необходимыми для сжатия и распаковки данных.

## 2.4 Распределение

Для корректного распределения данных необходимо следовать рекомендациям:

- Задать значения столбца или выбрать случайное распределение для всех таблиц. Не использовать значение по умолчанию;
- Использовать один столбец, который будет равномерно распределять данные по всем сегментам;
- Не распределять на столбцы, которые будут использоваться при запросе WHERE;
- Не распределять на даты или временные метки;
- Никогда не распределять и не разделять таблицы в одном столбце;
- Достичь локальных объединений для значительного повышения производительности, распределив их в одном столбце для больших таблиц, которые обычно объединяются вместе;
- Убедиться, что данные равномерно распределены после начальной загрузки и после инкрементных нагрузок;
- В конечном итоге убедиться, что нет искажений данных.

## 2.5 Управление памятью

Для управления памятью необходимо следовать рекомендациям:

- Установить значение параметра `vm.overcommit_memory` на 2;
- Не настраивать ОС для использования огромных страниц;
- Использовать `gp_vmem_protect_limit`, чтобы установить максимальный размер памяти, которую экземпляр может выделить для работы, выполняющейся в каждом сегментном хосте;
- Не допускать, чтобы значение `gp_vmem_protect_limit` превышало значения физического RAM в системе;
- Установить правильное значение для `gp_vmem_protect_limit` следующим образом:

  `(SWAP + (RAM vm.overcommit_ratio)) 0.9 / number_segments_per_server`
- Использовать `statement_mem` для выделения памяти, используемой для запроса на сегмент db;
- Использовать очереди ресурсов для установки количества активных запросов (ACTIVE_STATEMENTS) и суммы памяти (MEMORY_LIMIT), которые могут использоваться запросами в очереди;
- Связать всех пользователей с очередью ресурсов. Не использовать значения, установленные по умолчанию;
- Установить PRIORITY в соответствии с реальными потребностями очереди для рабочей нагрузки;
- Убедиться, что распределение памяти очереди ресурсов не превышает `gp_vmem_protect_limit`;
- Обновить параметры очереди ресурсов в соответсвии с ежедневным потоком операций.

## 2.6 Секционирование

При секционировании необходимо следовать рекомендациям:

- Секционировать можно только большие таблицы;
- Использовать секционирование только в том случае, если обрезка разделов может быть выполнена на основе критериев запроса;
- Выбирать секционирование по диапазону вместо секционирования по списку;

- Разбить таблицу на основе предиката запроса;
- Не разделять и не распределять таблицы по одному столбцу;
- Не использовать секционирование по умолчанию;
- Не использовать многоуровневое секционирование; создавать меньше разделов с большим количеством данных в каждом разделе;
- Проверить, что запросы выборочно сканируют секционированные таблицы путем изучения плана запроса EXPLAIN;
- Не создавать слишком много разделов с хранилищем, ориентированным на хранение по столбцам, из-за общего количества физических файлов на каждом сегменте:

  ```
  физические файлы = сегменты * столбцы * разделы
  ```

## 2.7 Индексы

Как правило, при работе с базой данных **ADB** необходимости в использовании индексов нет. Однако, если это требуется, следует создать индекс в одном столбце таблицы (хранение данных по столбцам) для сквозного доступа для таблиц высокой кардинальности, требующих запросов с высокой избирательностью. И следовать рекомендациям:

- Не индексировать столбцы, которые часто обновляются;
- Всегда отбрасывать индексы перед загрузкой данных в таблицу. После загрузки повторно создавать индексы для таблицы;
- Создавать выборочные индексы B-дерева;
- Не создавать bitmap-индексы в столбцах, которые находятся в процессе обновления;
- Не использовать bitmap-индексы для уникальных столбцов и данных очень большой или маленькой кардинальности;
- Не использовать bitmap-индекс для транзакционных нагрузок;
- Как правило, не следует индексировать секционированные таблицы. Если все же индексы необходимы, столбцы индекса должны отличаться от столбцов секции.

## 2.8 Очереди ресурсов

Для управления рабочей нагрузкой в кластере необходимо использовать очереди ресурсов:

- Использовать параметр `ACTIVE_STATEMENTS` для ограничения количества активных запросов, которые члены конкретной очереди могут запускать одновременно;
- Использовать параметр `MEMORY_LIMIT` для управления общим объемом памяти, который может использоваться для запросов, запущенных в очереди;
- Не устанавливать все очереди в MEDIUM, так как это фактически ничего не делает для управления рабочей нагрузкой;
- Изменять очереди ресурсов в зависимости от рабочей нагрузки.

## 2.9 COPY

Самый низкоуровневый способ импортировать и экспортировать данные:

- Не создает паразитных блокировок в отличии от средств бекапирования;

- Работает как на мастере, так и на сегментах;
- Интегрируется со сторонним ПО;
- Текст и BINARY;
- Может экспортировать результат выполнения запроса, представлений и т.д.

# Глава 3

# Поддерживаемые сторонние клиентские приложения

Пользователи могут подключаться к базе данных **ADB** с помощью различных клиентских приложений:

- Ряд клиентских приложений ADB предоставляется с установщиком. Клиентское приложение *psql* предоставляет интерфейс командной строки; *pgAdmin III* для ADB – это расширенная версия популярного инструмента управления *pgAdmin III*. Начиная с версии *1.10.0*, *pgAdmin III* включает поддержку специфичных для ADB функций. Пакеты установки доступны для загрузки с сайта *pgAdmin*;
- Используя стандартные интерфейсы приложений баз данных, такие как *ODBC* и *JDBC*, пользователи могут создавать свои собственные клиентские приложения, взаимодействующие с базой данных ADB. Поскольку СУБД ADB основана на PostgreSQL, она использует стандартные драйверы баз данных *PostgreSQL*;
- Большинство сторонних клиентских инструментов, которые используют стандартные интерфейсы базы данных, такие как *ODBC* и *JDBC*, могут быть настроены для подключения к ADB.

## 3.1 Клиентские приложения

База данных **ADB** поставляется с несколькими клиентскими приложениями, расположенными в *$GPHOME/bin* главной хост-системы. В таблице перечислены наиболее часто используемые клиентские приложения.

При использовании клиентских приложений следует подключиться к базе данных с помощью инстанса мастера **ADB**. Необходимо знать имя целевой базы данных, имя хоста и номер порта мастера, а также имя пользователя базы данных для подключения. Эта информация может быть предоставлена в командной строке с использованием опций `-d`, `-h`, `-p` и `-U` соответственно. Если найдется аргумент, который не принадлежит ни одному из параметров, он интерпретируется как имя базы данных.

Все эти параметры имеют значения по умолчанию, которые используются, если опция не указана. По умолчанию хостом является локальный хост. Номер порта – `5432`. Имя пользователя – это имя пользователя ОС, как и имя базы данных по умолчанию. Следует обратить внимание, что имена пользователей ОС и имена пользователей **ADB** необязательно должны быть одинаковы. Если значения по умолчанию неверны, можно установить переменные *PGDATABASE*, *PGHOST*, *PGPORT* и *PGUSER* на соответствующие значения или использовать файл *psql~/.pgpass* для хранения часто используемых паролей.

## 3.2 Подключение с psql

В зависимости от используемых значений следующие примеры показывают, как получить доступ к базе данных через *psql*:

```
$ psql -d gpdatabase -h master_host -p 5432 -U gpadmin
$ psql gpdatabase
$ psql
```

Если пользовательская база данных еще не создана, можно получить доступ к системе, подключившись к базе данных *template1*. Например:

```
$ psql template1
```

После подключения к базе данных *psql* предоставляет приглашение с именем базы данных, к которой в настоящее время подключен *psql*, а затем строковой тип => (или = # для суперпользователя базы данных). Например:

```
gpdatabase =>'
```

В командной строке можно вводить команды SQL при этом она должна заканчиваться на знак ; для отправки на сервер и для выполнения. Например:

```
=> SELECT * FROM mytable;
```

## 3.3 pgAdmin III для ADB

Это GUI-клиент, поддерживающий базы данных **PostgreSQL** со всеми стандартными функциями *pgAdmin III*, добавляя поддержку функций, специфичных для **ADB**. *pgAdmin III* поддерживает следующие особенности **ADB**:

- Внешние таблицы;
- Таблицы с оптимизированным соединением (append-optimized), включая сжатые таблицы, оптимизированные для добавления;
- Разделение таблиц;
- Очереди ресурсов;
- Графический анализ ANPLYIN ANALYZE;
- Параметры конфигурации сервера ADB.

## 3.4 Установка pgAdmin III

Пакет установки для *pgAdmin III* для базы данных **ADB** доступен для загрузки с официального сайта *pgAdmin III*. Инструкции по установке включены в установочный пакет.

## 3.5 Просмотр графического плана запроса

Используя *pgAdmin III*, можно запустить запрос с помощью EXPLAIN, чтобы просмотреть детали плана запроса. Вывод содержит сведения об операциях, уникальных для обработки распределенных запросов **ADB** таких, как срезы плана и движения между сегментами. Можно просмотреть графическое изображение плана, а также текстовый вывод данных.

Действия для просмотра графического плана запроса:

1. Выделив нужную базу данных в браузере объектов в левой панели, выбрать инструмент *Tools* → *Query*.
2. Ввести запрос в SQL, перенести объекты в *GraphicalQuery Builder* или открыть файл.
3. Выбрать *Query* → *Explain* и проверить следующие параметры:

- *Verbose* – должно быть отключено для просмотра графического изображения плана запроса;
- *Analyze* – выбрать этот параметр для запуска запроса в дополнение к просмотру плана.

4. Завершить операцию, нажав *Explain* в верхней части панели или выбрав *Query* → *Explain*.

План запроса отображается в области вывода в нижней части экрана. Чтобы просмотреть графический вывод необходимо перейти на вкладку *Explain*.

## 3.6 Интерфейсы приложений баз данных

Есть возможность разработки собственных клиентских приложений, взаимодействующих с базой данных **ADB**. **PostgreSQL** предоставляет ряд драйверов баз данных для наиболее часто используемых API, которые также могут использоваться с **ADB**. Эти драйверы не упакованы базовым дистрибутивом **ADB**. Каждый драйвер является независимым проектом разработки **PostgreSQL** и должен быть загружен, установлен и настроен для подключения к базе данных **ADB**. Доступные драйверы представлены в таблице.

Таблица3.1.: Клиентские приложения

| API | Драйвер PostgreSQL | Ссылка для скачивания |
|---|---|---|
| ODBC | pgodbc | Доступно в комплекте подключения к базе данных ADB, который можно загрузить из Центра загрузки EMC |
| JDBC | pgjdbc | Доступно в комплекте подключения к базе данных ADB, который можно загрузить из Центра загрузки EMC |
| Perl DBI | pgperl | http://gborg.PostgreSQL.org/project/pgperl |
| Python DBI | pygresql | http://www.pygresql.org |

Общие инструкции по доступу к базе данных **ADB** с помощью API:

1. Загрузить платформу и соответствующий API из источника. Например, можно получить Java-комплект разработчика (JDK) и API JDBC от Sun.
2. Записать клиентское приложение в соответствии со спецификациями API. При программировании приложения необходимо следить за тем, чтобы не был включен какой-либо неподдерживаемый синтаксис SQL.
3. Загрузить соответствующий драйвер PostgreSQL и настроить подключение к мастер-инстансу базы данных ADB. ADB предоставляет пакет клиентских инструментов, содержащий поддерживаемые драйверы баз данных.

## 3.7 Инструменты сторонних клиентов

В большинстве сторонних средств ETL и бизнес-аналитики BI используются стандартные интерфейсы баз данных, такие как *ODBC* и *JDBC*. Их можно настроить для подключения к базе данных **ADB**. **ADB** работает со следующими инструментами:

- Business Objects;

- Microstrategy;
- Informatica Power Center;
- Microsoft SQL Server Integration Services (SSIS) and Reporting Services (SSRS);
- Ascential Datastage;
- SAS;
- Cognos.