

# Arenadata™ Database

*Версия - v6.15.0-arenadata15*

**Рекомендации по работе с кластером ADB**

# Оглавление

|   |   |    |
|---|---|----|
| 1 | Модель данных                                       | 3  |
| 2 | Сравнение хранилищ типа “heap” и “append-optimized” | 4  |
| 3 | Хранение данных по столбцам и строкам               | 5  |
| 4 | Сжатие  | 6  |
| 5 | Распределение                                       | 7  |
| 6 | Управление памятью                                  | 8  |
| 7 | Секционирование                                     | 9  |
| 8 | Индексы   | 10 |
| 9 | Очереди ресурсов                                    | 11 |

Целью данного документа является предоставление краткого изложения наиболее важных аспектов в базе данных ADB.

---

**Important:** Контактная информация службы поддержки – e-mail: [info@arenadata.io](mailto:info@arenadata.io)

---

# Глава 1

## Модель данных

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

---

## Глава 2

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

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

Хранилище *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:** Нельзя выполнять однострочные операции *INSERT*, *UPDATE* или *DELETE* в таблицах типа *append optimized*

---

---

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

---

## Глава 3

# 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.

## Глава 4

# 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.

## Глава 5

# 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;
- Убедиться, что нет искажений данных.



## Глава 6

# 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 каждым 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:  
$$(\text{SWAP} + (\text{RAM} * \text{vm.overcommit\_ratio})) * 0.9 / \text{number\_segments\_per\_server}$$
- 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 соответствии с ежедневным потоком операций.

## Глава 7

# 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

## Глава 8

# Индексы

Как правило, при работе с **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 В-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.

## Глава 9

# 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.