САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Безопасность систем баз данных

Лекция 8
Секционирование. Сегментирование

Агафонов Антон Александрович
д.т.н., профессор кафедры ГИиИБ

Самара

План лекции

- Секционирование
  - Типы секционирования
  - Секционирование в MySQL
  - Секционирование в PostgreSQL
- Сегментирование
  - Сегментирование в MongoDB

Основные понятия

Репликация - набор технологий копирования и распространения данных и объектов баз данных между базами данных и последующей синхронизации баз данных для поддержания их согласованности.

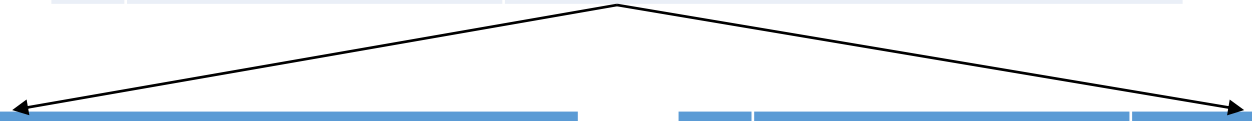
Секционирование (partitioning) — разделение хранимых объектов баз данных (таких как таблиц, индексов, материализованных представлений) на отдельные части с раздельными параметрами физического хранения. Используется в целях повышения управляемости, производительности и доступности для больших баз данных.

Сегментирование (sharding) — подход, предполагающий разделение баз данных, отдельных её объектов или индексов поисковых систем на независимые сегменты, каждый из которых **управляется отдельным экземпляром сервера базы данных**, размещаемым, как правило, на отдельном вычислительном узле.

Секционирование

Секционирование данных – разбиение одной большой логической таблицы на несколько меньших физических секций. Секционирование позволяет распределять части табличных данных (секции / разделы) по файловой системе на основе набора определяемых пользователем правил.

Id	title	description
1	ACADEMY DINOSAUR	A Epic Drama of a Feminist …
2	ACE GOLDFINGER	A Astounding Epistle of a Database…
3	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack…
4	AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee …
5	AFRICAN EGG	A Fast-Paced Documentary of a Pastry …
6	AGENT TRUMAN	A Intrepid Panorama of a Robot …



Id	title	description
1	ACADEMY DINOSAUR	A Epic Drama of a Feminist …
2	ACE GOLDFINGER	A Astounding Epistle of a Database…
3	ADAPTATION HOLES	A Astounding Reflection of a Lumberjack…

Id	title	description
4	AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee …
5	AFRICAN EGG	A Fast-Paced Documentary of a Pastry …
6	AGENT TRUMAN	A Intrepid Panorama of a Robot …

Секционирование

- Секционирование позволяет хранить в одной таблице больше данных, чем может храниться на одном диске или разделе файловой системы.
- В определённых ситуациях секционирование значительно увеличивает быстродействие, особенно когда большой процент часто запрашиваемых строк таблицы относится к одной или лишь нескольким секциям. Секционирование по сути заменяет верхние уровни деревьев индексов, что увеличивает вероятность нахождения наиболее востребованных частей индексов в памяти.
- Массовую загрузку и удаление данных можно осуществлять, добавляя и удаляя секции, если такой вариант использования был предусмотрен при проектировании секций. Удаление отдельной секции выполняется гораздо быстрее, чем аналогичная массовая операция. И наоборот, процесс добавления новых данных в некоторых случаях можно значительно облегчить, добавив один или несколько новых разделов для хранения именно этих данных.
- Редко используемые данные можно перенести на более дешёвые и медленные носители.

Типы секционирования

В различных СУБД возможности реализации несколько различаются. Можно выделить следующие типы секционирования:

- по интервалам (range partitioning),
- по списку значений (list partitioning),
- по хешу (hash partitioning),
- по ключам (key partitioning).

В некоторых СУБД (включая MySQL и PostgreSQL) поддерживается вложенное секционирование, при котором сами секции также разбиваются на секции.

MySQL. Секционирование по интервалам

При данном типе секционирования строки таблицы ставятся в соответствие разделам на основе интервала, в который попадает значение выражения секционирования (partitioning expression) строки.

- Интервалы значений должны быть смежными, но не должны перекрывать друг друга.
- Оператор VALUES LESS THAN будет использоваться для определения таких диапазонов в порядке от наименьшего.
- Строки, для которых выражение секционирования возвращает значение NULL, попадут в первый раздел.
- Секционирования данного типа может выполняться по нескольким столбцам таблицы.

MySQL. Секционирование по интервалам

```
CREATE TABLE userslogs (
    username VARCHAR(20) NOT NULL,
    logdata BLOB NOT NULL,
    created DATETIME NOT NULL,
    PRIMARY KEY (username, created)
)
PARTITION BY RANGE ( YEAR(created) ) (
    PARTITION p1 VALUES LESS THAN (2014),
    PARTITION p2 VALUES LESS THAN (2015),
    PARTITION p3 VALUES LESS THAN (2016),
    PARTITION p4 VALUES LESS THAN MAXVALUE
);
```

MySQL. Секционирование по списку значений

Секционирование по списку значений похоже на секционирование по интервалам, за исключением того, что раздел выбирается на основе соответствия значения выражения секционирования одному из наборов дискретных значений.

- Для секционирования по списку значений нет аналога выражения MAXVALUE, которое позволяет покрыть все возможные значения выражения секционирования. Соответственно, необходимо учесть все возможные значения выражения секционирования.
- Значения NULL обрабатываются как любые другие значения.
- Секционирования данного типа может выполняться по нескольким столбцам таблицы.

MySQL. Секционирование по списку значений

```
CREATE TABLE serverlogs (
    serverid INT NOT NULL,
    logdata BLOB NOT NULL,
    created DATETIME NOT NULL
)
PARTITION BY LIST ( serverid ) (
    PARTITION server_east VALUES IN (1, 43, 65, 73),
    PARTITION server_west VALUES IN (534, 6422, 196, 22)
);
```

MySQL. Секционирование по хешу

Разбиение по хешу используется, прежде всего, для обеспечения равномерного распределения данных между заданным числом секций.

- При секционировании по хешу необходимо указать выражение для секционирования и количество секций, на которые должна быть разделена таблица.
- Выражение секционирования вычисляется каждый раз, когда строка вставляется или обновляется.
- Сложные выражения могут привести к проблемам с производительностью, особенно при выполнении операций, которые затрагивают большое количество строк одновременно.

MySQL. Секционирование по хешу

```
CREATE TABLE serverlogs_hash (
    serverid INT NOT NULL,
    logdata BLOB NOT NULL,
    created DATETIME NOT NULL
)
PARTITION BY HASH ( YEAR(created) )
PARTITIONS 10;
```

MySQL. Секционирование по ключам

Разбиение по ключу используется для секционирования по ключевому полю.

```
CREATE TABLE serverlogs_key (
    serverid INT NOT NULL,
    logdata BLOB NOT NULL,
    created DATETIME NOT NULL,
    UNIQUE KEY (serverid)
)
PARTITION BY KEY ()
PARTITIONS 10;
```

MySQL. Отсечение секций

Отсечение секций (partition pruning) — это приём оптимизации запросов, который ускоряет работу с секционированными таблицами. Оптимизация основана на относительно простой концепции, которую можно описать как «не сканировать разделы, в которых не может быть строк, удовлетворяющих условию запроса».

Оптимизатор может выполнять отсечение всякий раз, когда условие WHERE может быть сведено к одному из следующих двух случаев:

- `partition_column = constant`
- `partition_column IN (constant1, constant2, ..., constantN)`

Функции над `partition_column` (например, `YEAR(partition_column)`) в запросе не используются.

Операторы SELECT, DELETE и UPDATE поддерживают отсечение секций. Однако, выбор выражений секционирования, поддерживающих отсечение секции, ограничен.

```
CREATE TABLE userslogs (
    ...
)
PARTITION BY RANGE ( YEAR(created) ) (
    PARTITION p1 VALUES LESS THAN (2014), PARTITION p2 VALUES LESS THAN (2015),
    PARTITION p3 VALUES LESS THAN (2016), PARTITION p4 VALUES LESS THAN MAXVALUE
);
```

```
EXPLAIN SELECT COUNT(*) FROM userslogs
WHERE created = '2012-10-01';
```

	Id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
	1	SIMPLE	artist	p1	index	PRIMARY	PRIMARY	87	NULL	19446	10.00	Using where Using index

```
EXPLAIN SELECT COUNT(*) FROM userslogs
WHERE YEAR(created) < '2013';
```

	Id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
	1	SIMPLE	artist	p1,p2,p3,p4	index	NULL	PRIMARY	87	NULL	49146	100.00	Using where Using index

MySQL. Особенности секционирования

- **Внешние ключи**. Секционированные таблицы не поддерживают внешние ключи. Таким образом, нельзя добавить внешний ключ в секционированную таблицу. И наоборот, если у таблицы есть внешний ключ, ее нельзя секционировать. Кроме того, в несекционированной таблице не может быть внешнего ключа, указывающего на столбец секционированной таблицы.
- **Индексы**. Секционирование применяется ко всем данным и индексам в таблице; нельзя секционировать только данные, а не индексы, или наоборот. Также нельзя секционировать только часть таблицы. Секционированные таблицы не поддерживают полнотекстовые индексы.
- **Связь с первичными и уникальными ключами таблицы**. Все столбцы, используемые в выражении секционирования таблицы, должны быть частью каждого уникального ключа данной таблицы. Сюда также относится первичный ключ таблицы, поскольку он по определению является уникальным ключом.

PostgreSQL поддерживает два типа секционирования:

- декларативное секционирование;
- секционирование с использованием наследования.

При **декларативном** секционировании декларируется, что некоторая таблица разделяется на секции. Сама секционированная таблица является «виртуальной» и как таковая не хранится. Хранилище используется её *секциями*, которые являются обычными таблицами, связанными с секционированной.

Сами секции могут представлять собой секционированные таблицы, таким образом реализуется *вложенное секционирование*. Хотя все секции должны иметь те же столбцы, что и секционированная родительская таблица, в каждой секции независимо от других могут быть определены свои индексы, ограничения и значения по умолчанию.

Преобразовать обычную таблицу в секционированную и наоборот нельзя. Однако в секционированную таблицу можно добавить в качестве секции существующую обычную или секционированную таблицу, а также можно удалить секцию из секционированной таблицы и превратить её в отдельную таблицу.

Возможности **секционирования с использованием наследования**:

- При декларативном секционировании все секции должны иметь в точности тот же набор столбцов, что и секционируемая таблица, тогда как обычное наследование таблиц допускает наличие в дочерних таблицах дополнительных столбцов, отсутствующих в родительской таблице.
- Механизм наследования таблиц поддерживает множественное наследование.
- С декларативным секционированием поддерживается только разбиение по спискам, по диапазонам и по хешу, тогда как с наследованием таблиц данные можно разделять по любому критерию, выбранному пользователем.

Но при секционировании с использованием наследования для добавления записей в нужные секции необходимо вручную создавать триггеры.

1. Создание таблицы как секционированной таблицы с предложением PARTITION BY, указав метод разбиения (в нашем случае RANGE) и список столбцов, которые будут образовывать ключ разбиения.

```
CREATE TABLE userslogs (
    username VARCHAR(20) NOT NULL,
    logdata TEXT NOT NULL,
    created DATE NOT NULL
)
PARTITION BY RANGE (created);
```

2. Создание секций. В определении каждой секции должны задаваться границы, соответствующие методу и ключу разбиения родительской таблицы. Указание границ, при котором множество значений новой секции пересекается со множеством значений в одной или нескольких существующих секциях, будет ошибочным.

```
CREATE TABLE userslogs_y2013 PARTITION OF userslogs
       FOR VALUES FROM (MINVALUE) TO ('2014-01-01');

CREATE TABLE userslogs_y2014 PARTITION OF userslogs
       FOR VALUES FROM ('2014-01-01') TO ('2015-01-01');

CREATE TABLE userslogs_y2015 PARTITION OF userslogs
       FOR VALUES FROM ('2015-01-01') TO ('2016-01-01');

CREATE TABLE userslogs_y2016 PARTITION OF userslogs
       FOR VALUES FROM ('2016-01-01') TO (MAXVALUE);
```

3. Создание в секционируемой таблице индекса по ключевому столбцу (или столбцам), а также любые другие индексов, которые могут понадобиться. При этом автоматически будет создан соответствующий индекс в каждой секции, и все секции, которые будут создаваться или присоединяться позднее, тоже будут содержать такой индекс. Индексы или ограничения уникальности, созданные в секционированной таблице, являются «виртуальными», как и сама секционированная таблица: фактически данные находятся в дочерних индексах отдельных таблиц-секций.

```
CREATE INDEX ON userslogs (created);
```

4. Проверка, что параметр конфигурации `enable_partition_pruning` не выключен в `postgresql.conf`. Иначе запросы не будут оптимизироваться должным образом.

Сегментирование

Сегментирование (sharding) — подход, предполагающий разделение баз данных, отдельных её объектов или индексов поисковых систем на независимые сегменты, каждый из которых **управляется отдельным экземпляром сервера базы данных**, размещаемым, как правило, на отдельном вычислительном узле.

Сегментирование позволяет:

- Хранить больше данных и обрабатывать бо*л*ьшую нагрузку без необходимости наличия мощных серверов
- Повысить отказоустойчивость
- Создавать георасп*р*еделенные системы

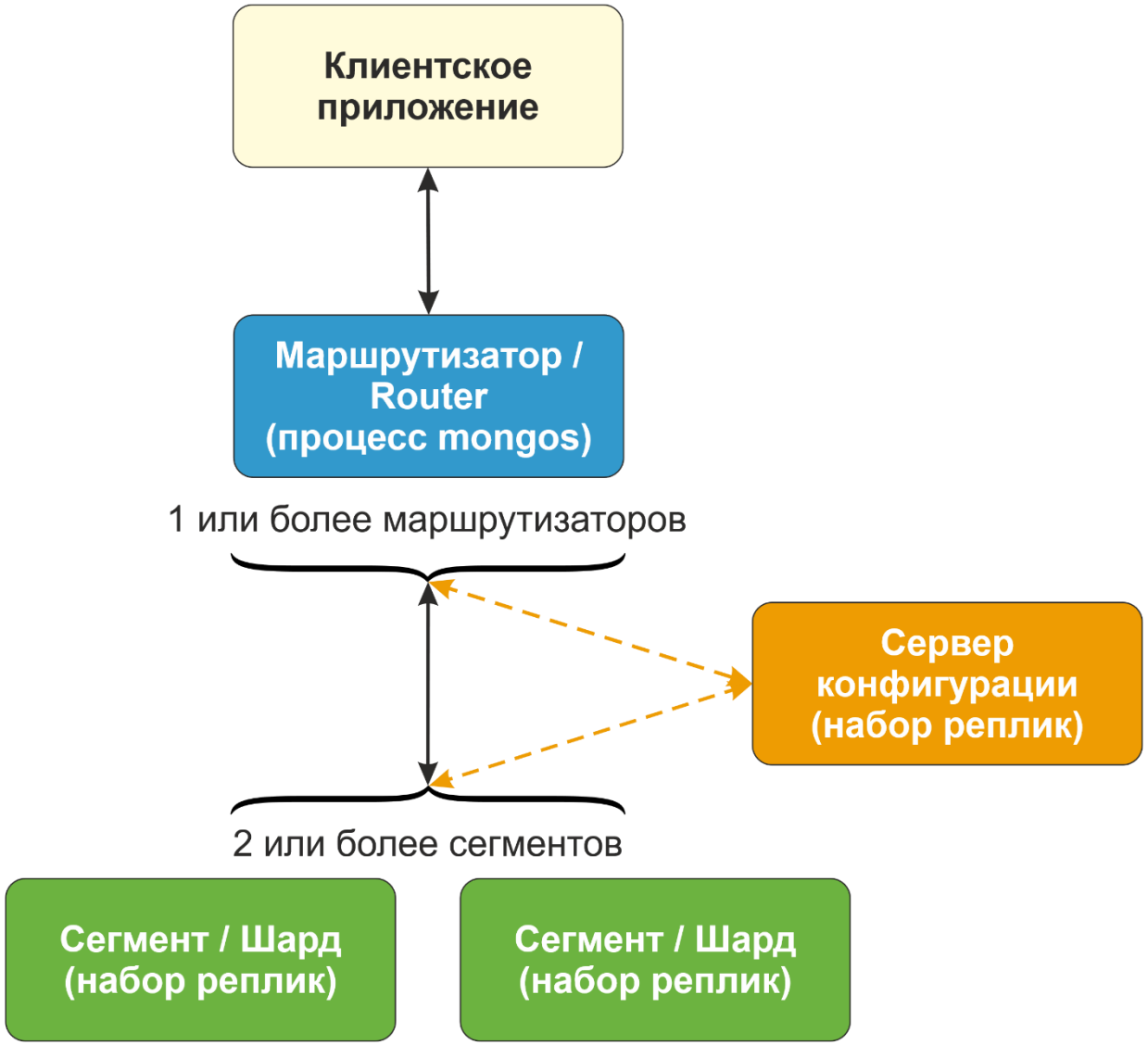
Вручную сегментирование можно осуществить практически с помощью любой СУБД. При таком подходе приложение поддерживает соединения с несколькими различными серверами баз данных, каждый из которых является полностью независимым. Приложение управляет хранением различных данных на разных серверах и выполняет запросы к соответствующему серверу, чтобы получить данные. Однако, такой подход трудно поддерживать при добавлении или удалении узлов из кластера или при изменении распределения данных по сегментам.

Сегментирование

Существует несколько СУБД, в которые встроено автоматическое сегментирование, большинство из них – это NoSQL решения: MongoDB, Apache Cassandra, Elasticsearch.

Автоматическое сегментирование в реляционных СУБД поддерживается хуже и в основном требует использование либо дополнительного ПО, либо обладает существенными недостатками.

В частность, большинство гибридных решений для Postgres основаны на технологии FDW (foreign data wrapper). FDW— это функциональность сервера Postgres, которая позволяет ему рассматривать данные на удаленном сервере как часть большого набора и интерпретировать этот разделенный на несколько серверов набор данных как единое целое. У этой технологии есть свои проблемы, но самая большая из них — доступ к данным через FDW чрезвычайно медленный.


Клиентское приложение
Маршрутизатор / Router (процесс mongos)
1 или более маршрутизаторов
Сервер конфигурации (набор реплик)
2 или более сегментов
Сегмент / Шард (набор реплик)
Сегмент / Шард (набор реплик)

MongoDB. Компоненты кластера

Сегментированный кластер MongoDB состоит из следующих компонентов:

- Сегмент – каждый сегмент содержит подмножество данных коллекции. Каждый сегмент можно развернуть как набор реплик.
- Процесс mongos – mongos действует как маршрутизатор запросов, предоставляя интерфейс для взаимодействия клиентского приложения с сегментированным кластером.
- Серверы конфигурации – серверы конфигурации хранят метаданные и параметры конфигурации для кластера.

MongoDB. Ключ сегментирования

Горизонтальное масштабирование подразумевает распределение набора данных и нагрузки по нескольким узлам СУБД. Для этого используется так называемый **ключ сегментирования**: сущности, связанные одинаковым значением ключа, группируются в набор данных по заданному ключу. Этот набор данных хранится в пределах одного физического сегмента, что существенно облегчает обработку данных.

MongoDB использует ключ сегментирования для распределения документов коллекции по сегментам. Ключ состоит одного или нескольких полей документа.

- Начиная с версии MongoDB 4.4, в документах в сегментированных коллекциях могут отсутствовать поля ключа сегментирования. Отсутствующие поля ключа обрабатываются как имеющие значения NULL при распределении документов по сегментам, но не при маршрутизации запросов.
- Начиная с версии MongoDB 5.0, можно выполнить повторное сегментирование коллекции, изменив ключ сегментирования.
- Начиная с версии MongoDB 4.2, можно обновить значение ключа сегментирования документа, если поле ключа сегментирования не является неизменяемым полем `_id`.

Индекс ключа сегментирования

Чтобы сегментировать заполненную коллекцию, она должна иметь индекс, начинающийся с ключа сегментирования. При сегментировании пустой коллекции MongoDB создает поддерживающий индекс, если в коллекции еще нет соответствующего индекса для указанного ключа сегмента.

Стратегия сегментирования

Выбор ключа сегментирования влияет на производительность, эффективность и масштабируемость сегментированного кластера. Кластер с наилучшим аппаратным обеспечением и инфраструктурой может оказаться узким местом из-за выбранного ключа сегментирования.

MongoDB поддерживает две стратегии сегментирования:

- Ranged — разделение данных на непрерывные диапазоны;
- Hashed — разделение данных на основе хеш-функции.

Чанки / Фрагменты / «Куски» данных (Chunks)

MongoDB разделяет сегментированные данные на чанки, являющиеся единицами, которые MongoDB использует для перемещения данных.

Каждый процесс mongos всегда должен знать, где найти документ по его ключ сегментирования. Теоретически MongoDB может отслеживать, в каком сегменте находится каждый документ, но в случае с коллекциями с миллионами или миллиардами документов это становится затратно, поэтому MongoDB группирует документы в чанки по некоторому диапазону ключа сегментирования.

Чанки всегда хранятся в одном сегменте, что позволяет MongoDB хранить небольшую таблицу связей чанк-сегмент.

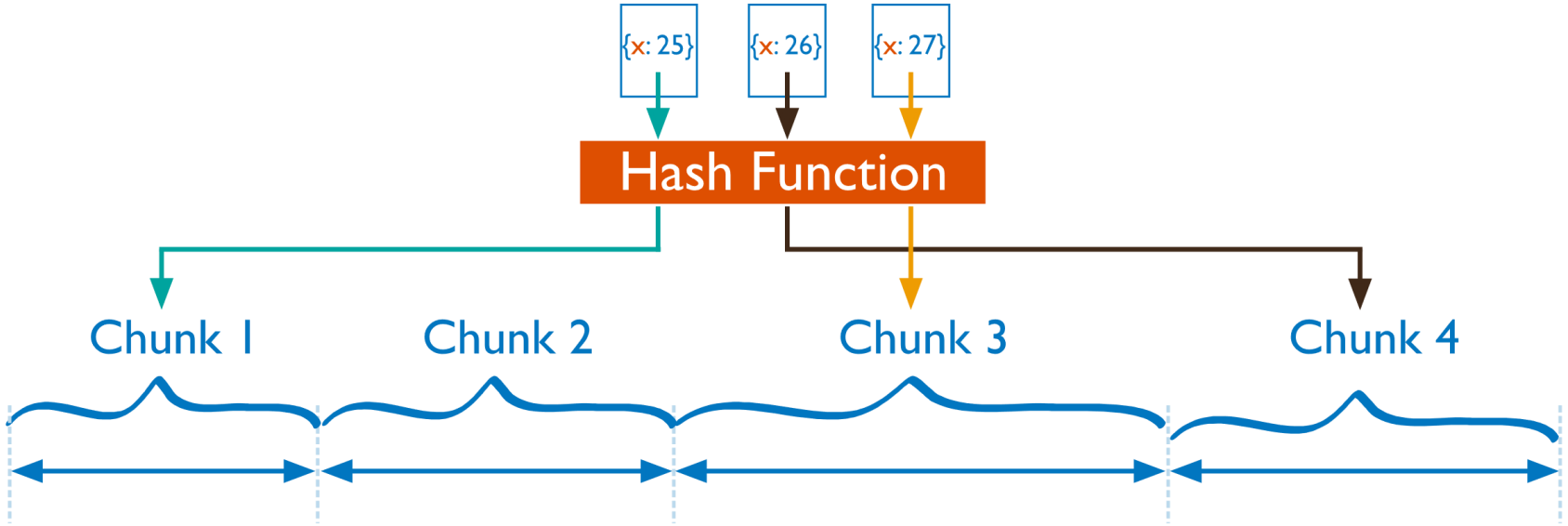
Балансировка и равномерное распределение чанков

Чтобы добиться равномерного распределения фрагментов по всем сегментам в кластере, в фоновом режиме запускается балансировщик для переноса чанков между сегментами.

Сегментирование по хешу основывается на вычислении хеша значения поля, входящего в ключ сегментирования. Каждому чанку назначается диапазон хеш-значений ключа сегментирования.

Хотя диапазон ключей сегментирования может быть «близким», их хешированные значения вряд ли будут находиться в одном и том же чанке. Следствия:

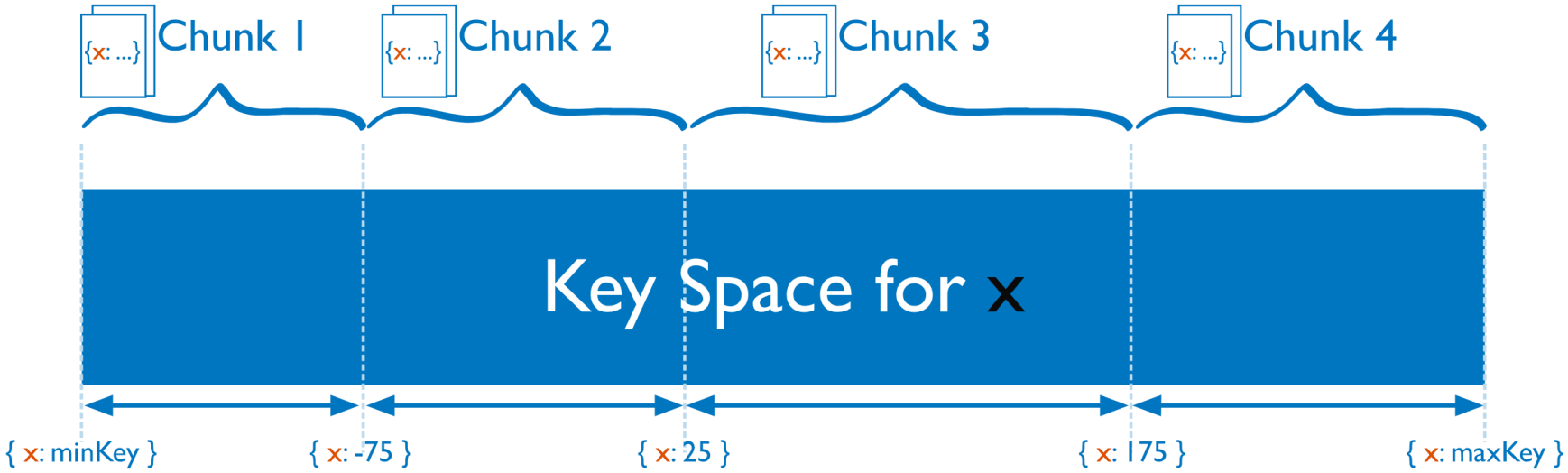
- ▲ более равномерное распределение данных;
- ▼ запросы по диапазону значений с меньшей вероятностью будут нацелены на один сегмент, что приводит к большему количеству широковещательных операций в масштабе кластера.



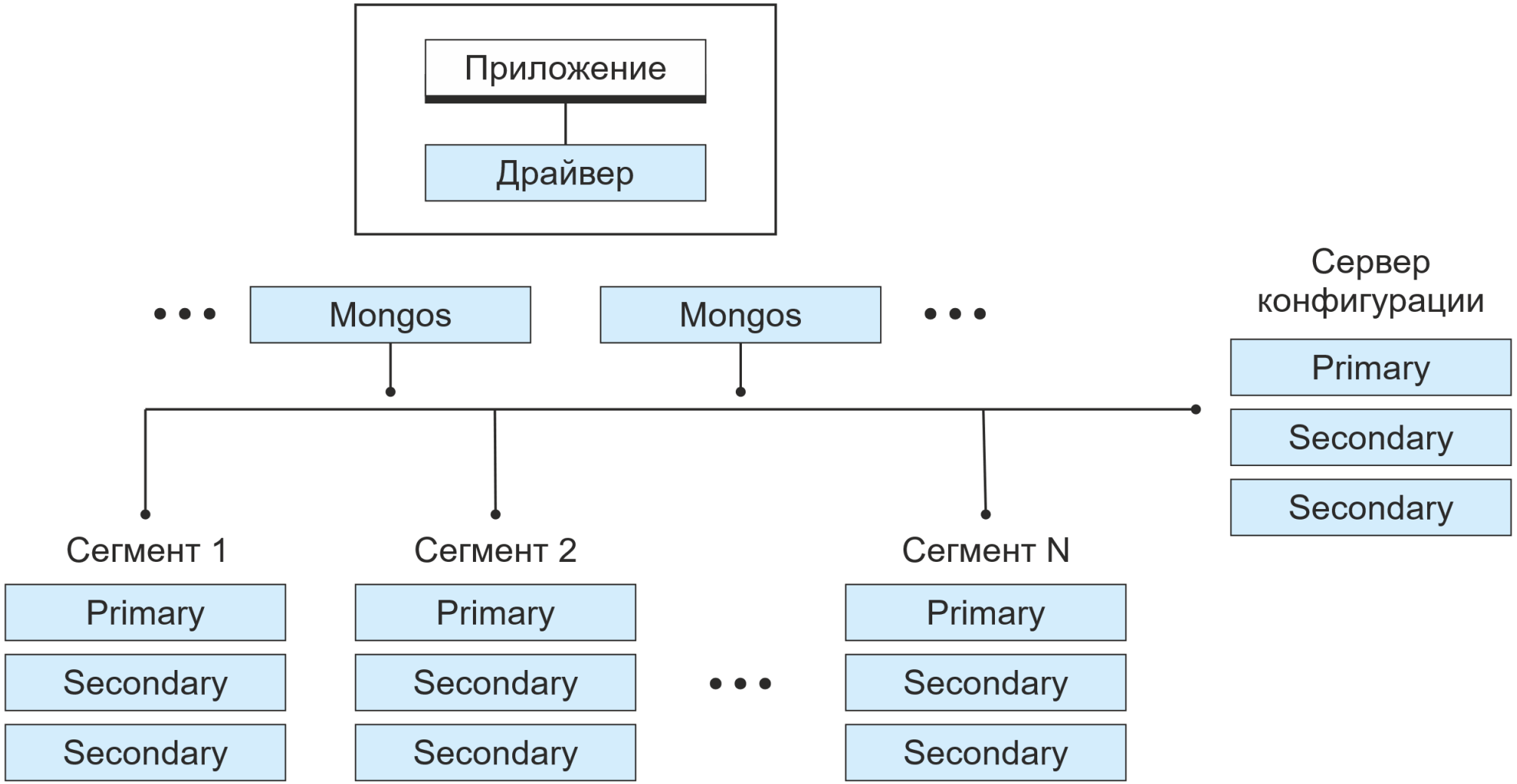
Сегментирование по диапазонам включает разделение данных на диапазоны на основе значений ключа сегментирования. Каждому чанку назначается диапазон на основе значений ключа сегментирования.

Диапазон ключей сегментирования, значения которых «близки», с большей вероятностью будет находиться в одном и том же чанке. Следствия:

- ▲ запросы по ключу сегментирования могут направляться к конкретному сегменту/сегментам;
- ▼ неравномерное распределение данных может привести к снижению эффективности.



MongoDB. Архитектура кластера в рабочей среде



1. Создание каталогов для сервера конфигурации и инициализация серверов

```
storage:
     dbPath: E:\mongo\cfg1\
net:
     port: 27018
     bindIp: 127.0.0.1  #localhost,<имя хоста | ip-адрес(а)>
sharding:
     clusterRole: configsvr
replication:
     replSetName: cfg_replica_set  #<replica set name>
```

```
$> mongod --config <path-to-config-file>
```

```
$> mongod --config E:\mongo\cfg1.cfg
$> mongod --config E:\mongo\cfg2.cfg
$> mongod --config E:\mongo\cfg3.cfg
```

2. Настройка набора реплик для конфигурационного сервера

```
$> mongosh --host <hostname> --port <port>

$> mongosh --host 127.0.0.1 --port 27018

rs.initiate({
    _id: "cfg_replica_set",
    configsvr: true,
    members: [
        { _id: 0, host: "localhost:27018" },
        { _id: 1, host: "localhost:27019" },
        { _id: 2, host: "localhost:27020" }
    ]
})

{ ok: 1, lastCommittedOpTime: Timestamp({ t: 1669921118, i: 1 }) }
```

3. Создание каталогов для сервера сегмента и инициализация серверов

```
storage:
     dbPath: E:\mongo\shard1_rs1\
net:
     port: 27021
     bindIp: 127.0.0.1  #localhost,<имя хоста | ip-адрес(а)>
sharding:
     clusterRole: shardsvr
replication:
     replSetName: shard1_replica_set  #<replica set name>
```

```
$> mongod --config <path-to-config-file>
```

```
$> mongod --config E:\mongo\shard1_rs1.cfg
$> mongod --config E:\mongo\shard1_rs2.cfg
...
$> mongod --config E:\mongo\shard2_rs3.cfg
```

4. Настройка набора реплик для **сегмента 1**

```
$> mongosh --host 127.0.0.1 --port 27021

rs.initiate({
    _id: "shard1_replica_set",
    members: [
        { _id: 0, host: "localhost:27021" },
        { _id: 1, host: "localhost:27022" },
        { _id: 2, host: "localhost:27023" }
    ]
})

{ ok: 1 }
```

5. Настройка набора реплик для **сегмента 2**

…

6. Создание процесса-маршрутизатора mongos

```
net:
     port: 27027
     bindIp: 127.0.0.1  #localhost,<имя хоста | ip-адрес(а)>
sharding:
     configDB: cfg_replica_set/127.0.0.1:27018,127.0.0.1:27019
```

```
$> mongos --config E:\mongo\mongos.cfg
```

```
$> mongosh --host 127.0.0.1 --port 27027
```

```
sh.addShard("shard1_replica_set/localhost:27021,localhost:27022,localhost:27023")
sh.addShard("shard2_replica_set/localhost:27024,localhost:27025,localhost:27026")
```

```
sh.shardCollection("<database>.<collection>", { <shard key field> : type } )
```

```
sh.shardCollection("airbnb.listings", { _id : "hashed" } )
```

Просмотр статусы сегментирования

```
sh.status()
```

```
shardingVersion {
  _id: 1,  minCompatibleVersion: 5,
  currentVersion: 6, clusterId: ObjectId("6388f96a6b4116d884baeb28")
}
---
shards
[ { _id: 'shard1_replica_set',
host: 'shard1_replica_set/localhost:27021,localhost:27022,localhost:27023', ... }, ... ]
---
active mongoses
[ { '6.0.3': 1 } ]
---
balancer
{ 'Currently enabled': 'yes', 'Failed balancer rounds in last 5 attempts': 0, ... }
---
databases
[ { database: { _id: 'airbnb', primary: 'shard1_replica_set', ... },
    collections: { 'airbnb.listings': { shardKey: { _id: 'hashed' }, ... } } }, ... ]
```

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

Агафонов А.А.
д.т.н., профессор кафедры ГИиИБ