NoSQL

Репликация  – это процесс синхронизации данных на нескольких серверах. Данный механизм уменьшает затраты ресурсов и увеличивает доступность данных, копии которых хранятся на разных серверах. Репликация защищает базу данных (далее – БД) от потери единственного сервера и позволяет сохранить данные в случае технической неисправности на одном из серверов.

**Преимущества репликации:**

* **Защищает данные**
* **Повышенная доступность данных**
* **Резервирование данных**
* **Возможность изъятия одного сервера для технического обслуживания**
* **Не оказывает** **влияния на работу самого приложения**

**Репликация в MongoDB**

В MongoDB репликация достигается путём  использования набора копий (**replica set**). Это группа экземпляров **mongod**, который хранят одинаковые наборы данных. В копии один узел – это ключевой узел, который получает все операции записи. Все остальные узлы – вторичны, принимают операции из первого, таким образом, храня такие же записи, как и первичный узел. Набор копий может иметь только один первичный узел.

Резюмируя, мы можем отметить следующие моменты:

* Набор копий – это группа из двух или более узлов (обычно, минимальное количество обязательных узлов – три)
* Все данные копируются из первичного узла во вторичные
* В наборе копий только один узел ключевой, а все остальные – вторичные
* Во время отключения первичного узла (обслуживание, поломка и т.д.) устанавливается новый первичный узел.
* После возврата первичного узла в работу, он становится вторичным

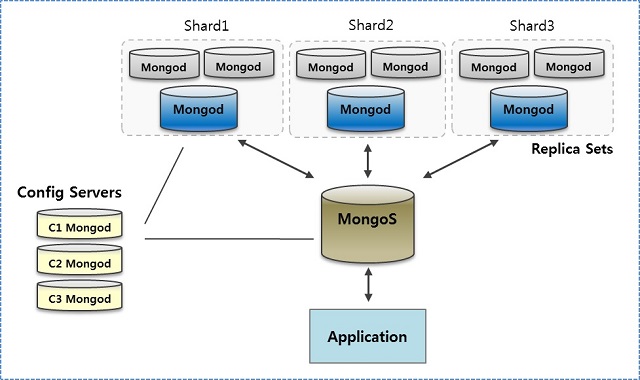
**Свойства наборы копий (Replica Set)**

* Любой узел может быть первичным
* Все операции записи идут в первичный узел
* Кластер состоит из N узлов
* Автоматическая отказоустойчивость
* Автоматическое восстановление данных
* Автоматический выбор первичного ключа

Шардинг – это процесс хранения документов на нескольких серверах и это способ, которым MongoDB справляется с большими данными. С ростом количества данных, один сервер не может хранить все данных, ни записывать их, ни давать к ним доступ. Шардинг решает проблему путём горизонтального масштабирования. Благодаря данному механизму мы можем подключать дополнительные сервера для хранения, записи и чтения данных.

**Преимущества шардинга**

* При репликации все записи идут в главный узел.
* Обработка запросов ложится на первичный узел.
* Один набор копий (replica set) имеет ограничение в 12 узлов.
* При работе с большим количеством данных, одна машина не может хранить все данные.
* Вертикальное масштабирование слишком дорогое



* **Конфигурационный сервер**Сервер, который хранит метаданные кластера. Эти данные содержат информацию о маппинге данных кластера на шарды (shards).
* **Шарды (Shards)**Используются для хранения данных. Обеспечивает высокую производительность. В промышленной разработке, каждый шард представляет собой отдельную **replica set**.

Преимущества перед SQL

* Отсутствие схемы  
  Данная БД основана на коллекциях различных документов. Количество полей, содержание и размер этих документов может отличаться. Т.е. различные сущности не должны быть идентичны по структуре.
* Крайне понятная структура каждого объекта.
* Легко масштабируется
* Для хранения используемых в данный момент данных используется внутренняя память, что позволяет получать более быстрый доступ.
* Данные хранятся в виде JSON документов
* MongoDB поддерживает динамические запросы документов (document-based query)
* Отсутствие сложных JOIN запросов
* Нет необходимости маппинга объектов приложения в объекты БД

**База данных**

База данных представленна в виде физического хранилища коллекций. Каждая БД имеет свой собственный набор файлов в файловой системе. Обычно, один MongoDB сервер имеет несколько БД.

**Коллекция**

Коллекция – это группа документов MongoDB. Является эквивалентом простой таблицы в реляционной базе данных. Коллекция помещена внутри одной БД. Документ в коллекции моет иметь различные поля. Чаще всего, все документы в коллекции созданы для одной, либо относящихся друг ко другу целей.

**Документ**

Документ – это набор пар “ключ – значение”. Документ имеет динамическую схему. Это означает, что документ в одной и той же коллекции не обязан иметь один одинаковый набор полей или структуру, а общие поля в коллекции могут иметь различные типы данных.

Ниже приведена небольшая сравнительная таблица реляционных БД и MongoDB:

|  |  |
| --- | --- |
| Реляционная БД | MongoDB |
| База данных | База данных |
| Таблица | Коллекция |
| Ряд | Документ |
| Колонка | Поле |
| Объединенние таблиц | Встроенные документы (embedded) |
| Первичный ключ (primary key) | Первичный ключ (primary key). По умолчанию MongoDB генерирует Default key\_id |

Типы данных в NoSQL:

* **Integer**Используется для хранения целочисленных значений. В зависимости от сервера может быть как 32-битным, так и 64-битным.
* **Double**Используется для хранения значений с плавающей точкой
* **Boolean**Используется для хранения логических (true / false) значений.
* **String**Используется для хранения символьных строк. В MongoDB используется кодировка UTF-8.
* **Arrays**Данный тип данных используется для хранения массивов значений по одному ключу.
* **Object**Используется для встроенных документов.
* **Symbol**Используется также, как и **String**, но, обычно, резервируется для языков, которые используют специальные символы.
* **Null**Используется для хранения значения Null.
* **Timestamp**Используется для хранения даты и времени
* **Min / Max**Используется для сравнения значений с наибольшим и наименьшим BSON (Binary JSON) элементом.
* **Object ID**Используется для хранения ID документа.
* **Regular Expression**Используется для хранения  регулярных выражений.
* **Code**Используется для хранения JavaScript кода в документе.
* **Binary data**Данный тип данных позволяет хранить бинарные данные.
* **Date**Используется для хранения текущей даты или времени в UNIX формате.