Добрый день. И добро пожаловать в блок - как работать с базой данных в go, в котором мы разберемся: что такое база данных, какие бывают базы данных и как работать с ними в go на практике. Погнали. Итак, если мы говорим про, что такое база данных, то база данных это просто каким-то определенным структурированным образом коллекция данных, которые хранятся в памяти компьютера. Причем, они могут храниться в оперативной памяти, могут храниться на жестких дисках. Короче, как-то мы организовываем эти данные, как-то их храним. В принципе, вот она и есть база данных. В ней как в таковой, в этом определении нет ничего сложного.

Другое дело, когда мы говорим про систему управления базами данных. Система управления базами данных это уже конкретный софт, конкретная реализация, с которой мы работаем, которая, как раз, отвечает за хранение данных, за их отдачу, за их надежность и прочее. Давайте посмотрим, какие у нее задачи есть и что определяет система управления базами данных. Первая задача — это надежное хранение данных, то есть вы можете хранить данные в блокноте, просто записав их в столбик, строчку, как вам удобно, но, во-первых, это не очень надежно потому, что ваш жесткий диск может посыпаться, а никакого восстановления данных у вас нет. Соответственно, эти данные вы, скорее всего, потеряете. При этом, у вас один символ хранится в одном конкретном месте на жестком диске, то есть, если что-то там не получится прочитать, то этот символ будет потерян. База данных, естественно, делает это несколько надежнее, включая, в том числе, и избыточность хранения данных и реплицирование, так называемое. То есть данные дуплицируются в нескольких местах и, если у вас посыпался один жесткий диск - ничего страшного, второй жесткий диск может, соответственно, содержать полную копию и за это ответственность берет на себя база данных. Система управления базами данных. И вам не нужно об этом думать.

Плюс к этому, конечно, система управления базами данных должна обеспечивать доступ к данным, должна позволять нам записывать данные, считывать данные, обеспечивать саму базу данных, саму логику хранение данных. То есть вы можете в некоторых системах управления вы можете задать какую-то логику хранения данных, то есть какие-то внутренние логические элементы, за которые база будет нести ответственность, в каких-то вы не можете, но в целом за вот обеспечение структуры и, так называемой, консистентности, то есть целостности данных отвечает именно система управления базами данных. И, естественно, большинство систем управления базами данных, с которыми вы будете работать, подавляющее большинство предназначено для работы внутри сети, ну или, по крайней мере, обеспечение соединения с каким-то клиентом.

Итак, какие типы баз данных бывают? Соответственно, если вы только начинаете работу с базами данных, то, скорее всего, вы встречаете, первое, что вы встречаете это реляционные базы данных общего назначения. В чем их смысл? У вас данные хранятся в табличках, есть колонки, каждая колонка имеет свой тип данных и есть столбцы. Каждый столбец — это совокупность колонок. Они могут быть пустыми, не пустыми. Уже зависит от внутренней структуры базы данных, как правило, помечаются какими-то идентификаторами, и, собственно говоря, в таком виде и хранятся. Каждая колонка может указывать на определенное поле из другой таблицы и создавать, так называемые, связи. Поэтому они называются реляционными, от слова relation - связь. Используется очень много где. Как правило, когда проект только начинается и у него нет никаких специальных требований, то берут именно реляционные базы данных для этого проекта. Это такой обычный стандартный выбор и, как правило, берут либо PostgreSQL, либо MySQL, Oracle, если вы кодите на android, у вас тем есть, соответственно, файловая база данных это SQLite. Ну, то есть, какие-то такого рода базы данных.

Для, соответственно, второй тип баз данных, который можно встретить очень часто это документа-ориентированные базы данных. Базы данных без схем, то есть у вас есть коллекции, так называемые, это какие-то логические единицы хранения данных. И вы в эти коллекции засовываете данные в произвольном формате. У вас в одной коллекции необязательно лежат данные с одинаковым набором колонок и, так же, строго говоря, они не должны быть одинаковые в плане структуры, там может быть что угодно реально. Это используется, когда вам нужно хранить какие-то данные, у которых высока флуктуация, то есть у них могут различаться списки столбцов и вам нужно просто их сваливать в кучу для того, чтобы проанализировать потом. Это могут быть логи, это может быть какая-то первичная система заказов для какого-нибудь ресторана, это может быть какая-то очередь задач для системы и, соответственно, выполнения этих задач ну и так далее. Яркими представителями являются MongoDB. Это, наверное, самая популярная schemaless база данных, а также CouchDB.

Соответственно, третий тип баз данных, которые существуют это база данных ключ-значение. То есть эти базы данных вам позволяют просто положить, соответственно, объект по ключу и достать его по ключу. Список объектов, который можно положить по ключу большой, можно класть простые типы, можно класть какие-то json, соответственно. Все это, как правило, предназначено для того, чтобы работать очень быстро потому, что, в отличие от традиционных баз данных, нет задач, например, получить по списку все данные, есть задача получить данные по конкретному ключу. Также, как и в (НРЗ 05:55) в go. И поэтому, как правило, эти базы данных хранятся в памяти и используются для кеширования данных, для хранения...для какого-то быстрого доставания, убирания данных. То есть, если вы, например, делает кеширующий слой или у вас система должна хранить небольшой объем данных по каким-то конкретным адресам, и вы не будете доставать эти данные списками это прекрасный выбор.

Соответственно, самой популярной является Redis, кто работает с Kubernetes, те точно знают (НРЗ 06:27), ну и, соответственно, базы данных memcache так же относятся к этому типу. Четвертый тип баз данных по популярности, я постараюсь их здесь отсортировать, если вы их встречали в другом порядке, ничего такого. Это просто моя небольшая статистика. По моему опыту. Соответственно, это база данных Wide-column, так называемая, то есть они похожи на реляционные базы данных, то есть там тоже есть таблицы с колонками и с какими-то строками, но, при этом, сами таблицы хранятся другим образом для того, чтобы оптимизировать запросы именно по одной таблице большому объему данных и какой-то агрегации внутри этих таблиц, то есть объединение каких-то данных по полям, например, если у вас хранится зарплата, посчитать всю зарплату этого отдела за какой-то месяц. И данные там хранятся без связи, как правило, с другим таблицами. Они хранятся в самой таблице и, если вам нужно что-то еще хранить (НРЗ 07:26) вы добавляете новую колонку. Но это все оптимизированно, обычно используется в системах BI, либо в системах анализа данных, когда у вас много данных, которые надо как-то проагрегировать.

Это не, как правило, не реалтаймовые базы данных. Ну, Cassandra вполне себе реалтаймовая, но тем не менее. И требует определенного подхода к организации этих данных. Ну и, соответственно, да, одна из самых популярных баз данных такого типа является ClickHouse, база данных от яндекса. И полная противоположность ClickHouse, предназначенного для аналитики, база данных реалтайм для огромного объема данных - база данных Cassandra. Следующий тип баз данных, с которым программистам и специалистам DevOps часто приходится сталкиваться это базы данных поисковые. То есть в чем смысл? Они ориентированны на то, что вы в них загружаете какие-то большие блоки текста и они по этим блокам текста ищут определенные слова и включения. Соответственно, у них существует развитый механизм построения индексов, то

есть хранение, запись данных в эти базы, обычно, не быстрая и нужно писать блоками, но, зато, поиск по ним очень быстрый и очень удобный, соответственно, они часто используются, например, для хранения логов или какой-то там большой информации, по которой надо искать. Самая известная из них это Elasticsearch, бывает, еще существует Splunk и база данных Solr.

Следующий тип баз данных, с которым, ну я лично работаю очень мало, но знаю, что это довольно популярный формат, это базы данных графовые. То есть как в них хранятся данные? В них данные хранятся в виде, так называемого, связанного графа, то есть вот есть, например, очень удобно себе представлять социальную сеть. Есть вы, у вас есть друг, вы с ним связаны. У вас есть, соответственно, взаимная дружба. У вашего друга существуют еще друзья и у их друзей есть еще друзья и получается, так называемая, сеть, как в том фильме про Цукерберга. И эту сеть очень сложно представить средствами, например, реляционных баз данных. Потому что у вас получаются таблицы, в которых есть связь к другому типу, потом есть связь с другой таблицей, у другой таблицы есть связь на эту таблицу. В общем, они начинают обмениваться ссылками, разобраться в этом сложно, поэтому существуют специальные базы данных, которые именно ориентированы на то, чтобы хранить графоподобные структуры. Это базы данных Neo4j, virtuoso и OrientDB, которая, кстати, так же является document-oriented, то есть там вы можете хранить еще и связку информации к этому графу, ну как и в любых других графовых базах данных. Там не просто хранится сущности таблицы, у сущности есть еще какие-то атрибуты.

Следующий тип баз данных, он стоит ниже, но он так же очень широко используется, как видите, да, этот список он несколько абстрактный, база данных Time series. То есть базы данных, которые хранят все документы в привязке к времени их записи или их происхождения, или к какому-то времени, которые вы указываете, то есть они специально оптимизированы для работы именно со

временем. Как правило, в них очень легко извлекать временные отрезки, сравнивать временные интервалы, очищать временные интервалы, ну, в общем, все, что связано с временем и анализом данных во времени. Это все базы данных Time series. Существует расширение timescale для PostgreSQL, которые делают PostgreSQL более-менее time oriented. Существует, соответственно, Prometheus это система мониторинга и сбора метрик, для того, чтобы вы могли как-то построить статистику по времени, вы тоже с ней, наверняка, столкнетесь, база данных Influx, ну и, в принципе, довольно широко используется.

И последний тип баз данных, который я не смог никуда отнести потому, что существует еще много различных типов баз данных это базы данных, так называемого, специального назначения. Ничего общего у них друг с другом нет. У всех этих баз данные, которые я объединил это, как правило, базы данных или работы с геоданными, или работы с какими-то специальными математическими данными, хранение каких-то там функций определенным образом, базы данных GeoMesa, QLDB, Spacetime можете на досуге с ними разобраться. В обычной программистской жизни, в обычной web-ориентированной компании вы, как правило, их не встретите. Либо, если у вас компания занимается какими-то специальными вещами, вы уже про них, итак, знаете.

Ну и перед тем, как мы перейдем, непосредственно, к практике, хотелось бы обратить внимание на три пункта. Во-первых, разные базы данных имеют разные способы забирать данные, естественно, что то, что подходит для реляционной базы данных, будет очень плохо подходить для базы данных ключ-значение. Соответственно, запросы тоже разные. Мы посмотрим на разницу в запросах, попробуем найти общие какие-то вещи, попробуем найти какие-то различия. Второе, это коммуникация. Естественно, разные базы данных, системы управления базами данных написаны разными людьми, используют разные протоколы для коммуникации, это следует как из первого пункта, так и из того, что

они еще и по-разному устроены внутри. И третье, это каждая база данных, это вытекает из того, что я говорил в предыдущем слайде, каждая база данных имеет, естественно, свою концепцию хранения, которую вы должны иметь ввиду, когда с ней работаете. Со всем этим вместе давайте перейдем к практике.