Занятие 7. Базы данных. PostgreSQL. Redis

База данных - это набор данных, организованных каким-либо образом. СУБД (Система Управления Базами Данных) - это средство для манипуляций данными в базе и самими базами.

Основными задачами баз данных можно назвать: поиск данных, манипуляции с данными, разграничение прав доступа к данным, возможность масштабирования, обеспечение целостности и надежности хранения данных, организация одновременной работы многих пользователей, организация безопасного хранения и доступа к данным. Внутри каждой базы данных и ее СУБД устанавливаются правила о типах и виде хранимых данных и их свойствах, одновременной и параллельной работы нескольких пользователей, набору действий с данными.

В общем виде можно считать, что данные в базе хранятся как файл или несколько файлов на дисках. По способу доступа к базам и самим данным СУБД можно разделить на несколько групп:

- Файл-серверные, когда сами файлы данных располагаются на сервере, а софт СУБД располагается на клиентских устройствах (MS Access, dBase)
- Клиент-серверные, когда сами файлы и СУБД располагаются на сервере, а клиентские устройства обращаются к СУБД (Oracle, DB2, Informix, MSSQL, PostgreSQL, MySQL)
- Встраиваемые, когда СУБД и данные поставляются как часть какого-либо продукта, не требуют установки или администрирования, обычно используется для локального хранения данных, не рассчитана на использование по сети и часто СУБД реализуется в виде подключаемой библиотеки (SQLite, BerkleyDB)

В зависимости от типа хранимых данными и принципами работы с ними базы можно разделить на:

- Реляционные (табличные)
- Нереляционные иерархические (деревья, например, файловая система)
- Нереляционные сетевые (построенные на связях, в отличие от деревьев узел в сетевых базах данных может иметь более чем одного предка)
- Объектно-ориентированные и объектно-реляционные (данные моделируются в виде объектов, их атрибутов и методов, используются для обработки данных, имеющих сложную структуру, в объектно-реляционных используются объекты, классы, наследования, реализованные в структуре баз данных и языке запросов)

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ

Реляционные базы данных (построенные на реляционной модели данных) еще называют табличными, потому что все данные в них можно представить в виде таблиц с разными данными, которые могут быть связаны между собой. Термин «реляционный» означает, что теория основана на математическом понятии отношения (relation), в качестве неформального синонима термину отношение часто встречается слово таблица, хотя это слово больше описывает именно визуальное представление отношения. В реляционной модели данных есть несколько важных обстоятельств: модель является логической, то есть все связи являются не физическими структурами данных, а логическими (абстрактными), все данные представлены одним и тем же способом и явным заданием значений, существует возможность реализовать декларативное описание ограничений целостности данных.

Связи в реляционных базах данных показывают, как одни данные могут зависеть от других. В связи может участвовать одна запись, а может сразу много. Самый простой вид связи - один к одному - одной записи из одной таблицы соответствует одна запись из другой. Один ко многим - когда одной записи из одной таблицы соответствует несколько записей из другой таблицы. Многие ко многим - хранение связей в отдельной таблице, поскольку одной записи из первой таблицы может соответствовать несколько записей из второй таблицы, но при этом запись из второй таблицы может соответствовать нескольким записям из первой.

Для управления реляционными базами данных и самими данными используется язык структурированных запросов - SQL (Structured Query Language). Он считается языком программирования, однако не Тьюринг-полным. Изначально SQL был основным инструментом работы пользователя с базой данных и позволял выполнять небольшой набор операций, в основном манипуляции данными или изменения структур хранения данных. Со временем он усложнился, в него добавились новые конструкции, функции, возможность управления новыми объектами - индексами, триггерами, процедурами. К сожалению в языке SQL существует проблема совместимости языка для разных СУБД, поскольку производители программного обеспечения, использующего SQL, начали расширять язык под свои задачи и во многих СУБД используются свои диалекты языка.

Для реляционных баз существует понятие нормализации - разбиения данных на таблицы или больших таблиц на несколько, чтобы данные стали более структурированы. Нормализация используется для упрощения системы, уменьшения количества ошибок, устранения избыточности данных, соответственно и уменьшения самого размера базы, защиту данных и устранение несогласованных зависимостей. Несогласованные зависимости устраняются за счет использования уникальных первичных ключей в таблицах (primary key) и ссылках на первичные ключи из других таблиц (foreign key), это же может служить и защитой от случайного удаления данных. При росте количества данных в таблицах для ускорения поиска и выборок данных из таблиц используют индексы - это своего рода оглавление, составленное из части данных, по которым чаще всего идет поиск. При росте количества одновременных пользователей и их работы с одними и теми же данными для предотвращения перезаписи данных используются механизмы блокировки записи и транзакций, когда при выполнении нескольких последовательных операций сбой на любой из них откатывает действия всех.

NOSQL БАЗЫ ДАННЫХ

NoSQL (Not Only SQL) - это подход к реализации масштабируемого хранилища с гибкой моделью данных, такие базы оптимизированы для приложений, которые должны быстро и с минимальной задержкой обрабатывать большой объем данных с разной структурой. Обычно NoSQL базы данных делят на 4 типа:

- *Ключ-Значение (Key-Value)* - наиболее простой вариант хранилища данных, использующий ключ для доступа к данным в рамках большой хеш-таблицы. Часто применяются для создания кешей.
- *Документно-ориентированое хранилище* - в нем данные, представленные парами ключ-значение сжимаются в виде структурированного документа, например, JSON, XML. Часто применяются для создания каталогов, хранения профилей, систем управления контентом.
- *Колоночное хранилище* - в нем данные хранятся в виде разреженной матрицы, где столбцы и строки используются как ключи. К колоночным хранилищам часто относятся базы типа «семейство столбцов», сами значения хранятся в столбцах, представленных в виде отдельных файлов, благодаря такой организации хранилища возможно хранить большое количество данных в сжатом виде, наличие временных меток позволяет использовать такие СУБД для счётчиков или обработчиков событий.
- Графовое хранилище представляет собой сетевую базу данных, которая использует узлы и рёбра для отображения и хранения данных. Поскольку рёбра графа являются хранимыми, его обход не требует дополнительных вычислений. Обычно грабовые базы данных поддерживают специализированные языки запросов. Обычно их используют для социальных сетей, маршрутов и дорожных карт.

По сравнению с реляционными SQL-базами, нереляционные NoSQL-базы обладают некоторыми преимуществами: линейная масштабируемость, гибкость операций с данными, возможность работы с разными представлениями информации, высокая доступность, производительность, широкие функциональные возможности, в том числе собственные SQL-подобные языки, REST-full интерфейсы, API, работа со сложными типами данных - map, list и т.д. Однако из преимущества иногда могут оказаться и недостатками: ограничения встроенных языков запросов, сложность поддержки требований к транзакциям и из организации, сильная привязка приложения/софта к конкретной СУБД из-за специфики языка и гибкой модели данных.

POSTGRESQL

PostgreSQL - это свободная объектно-реляционная СУБД, базируется на языке SQL. Она была создана на основе некоммерческой СУБД Postgres, разработанной Майклом Стоунбрейкером (руководителем проекта СУБД Ingres) и его студентами в университете в Беркли, позже сам Стоунбрейкер занялся разработкой своей коммерческой СУБД, а его студенты разработали новую версию Postgres - Postgres95 и заменили в ней язык запросов от «наследия» Ingres на SQL.

Фундаментальная характеристика объектно-реляционной базы данных - это поддержка пользовательских объектов и их поведения, включая типы данных, функции, операции, домены и индексы. PostgreSQL умеет создавать, хранить и извлекать сложные структуры данных. Существует обширный список типов данных, которые он поддерживает, в отличие от других подобных СУБД PostgreSQL поддерживает большее количество типов (кроме числовых, с плавающей точкой, текстовых, булевых и других ожидаемых типов данных, существует поддержка uuid, денежного, перечисляемого, геометрического, бинарного типов, сетевых адресов, битовых строк, текстового поиска, xml, json, массивов, композитных типов и диапазонов и т.п.). Так же СУБД поддерживает возможность создания пользовательских типов данных.

PostgreSQL считается решением для сложных операций с большими объемами данных: максимальный объем базы данных не ограничен, максимальный размер таблицы 32ТБ, максимальный размер строки 1.6ТБ и т.д. PostgreSQL предотвращает повреждение данных и сохраняет их целостность на транзакционном уровне. Поддерживает одновременный параллельный доступ множества клиентов как на чтение, так и на запись, устраняя необходимость блокировки данных.

Сетевое взаимодействие с PostgreSQL осуществляется на транспорте TCP, стандартный порт 5432, однако он может быть изменен. Для работы с СУБД требуется специальный клиентский софт (psql) или библиотеки (в случае java - jdbc driver).

В наших проектах PostgreSQL используется в виде множества различных инстансов, по которым логически распределены и сгруппированы данные по их типам и операциям с ними, объемы баз варьируются от 1 до 200ГБ. Несмотря на то, что PostgreSQL - объектно-реляционная СУБД, используется она именно реляционное представление, поскольку оно позволяет с минимальными изменениям портировать работу с данными на другие реляционные СУБД.

REDIS

Redis (Remote Dictionary Server) - это NoSQL база данных и СУБД, работающая со структурами типа ключ-значение. Все данные в Redis хранятся в памяти, а не на дисках, однако существует возможность создания снимков базы или журналирования для хранения данных в файловой системе в виде дампа, а так же поддерживает механизмы устаревания данных. Поскольку Redis не нуждается в доступе к диску, это исключает задержки, связанные с поиском, и обеспечивает доступ к данным за микросекунды, а размер БД ограничивается объемом доступной оперативной памяти. В Redis отсутствует поддержка языка SQL, однако существует возможность написания скриптов на Lua, которые так же будут выполняться в памяти. Из-за отсутствия четких разграничений прав доступа к данным внутри СУБД Redis не может использоваться в качестве основной базы, требовательной к безопасности, но прекрасно подходит для создания кешей, реализации событийного механизма, механизмов

подписки, хранения токенов сессий, быстрых поисковых механизмов за счет Lua-скриптинга и встроенных методов поиска по ключам и т.п.

Все данные Redis хранит в виде словаря, в котором ключи связаны со своими значениями. Одно из ключевых отличий Redis от других хранилищ данных заключается в том, что значения этих ключей не ограничиваются строками. Поддерживаются следующие абстрактные типы данных: строки, списки, множества, хеш-таблицы, упорядоченные множества. Тип данных значения определяет, какие операции (команды) доступны для него; поддерживаются такие высокоуровневые операции, как объединение и разность наборов, сортировка наборов.

Redis поддерживает механизмы односторонней репликации master -> slave, данные с любого сервера Redis могут реплицироваться произвольное количество раз. Репликация может быть использована в качестве механизма защиты данных от потери, а так же для увеличения производительности (горизонтальное масштабирование).

Сетевое взаимодействие с Redis осуществляется на транспорте TCP, стандартный порт 6379, однако он может быть изменен. Для работы с СУБД требуется специальный клиентский софт (redis-cli) или библиотеки (в случае java - jedis driver), однако так же возможна работа в режиме отладки и при помощи утилиты telnet.

В наших проектах Redis используется в основном в качестве распределенного быстрого кеша актуальных или часто-запрашиваемых данных, для хранения временных пользовательских токенов, генерации и выдачи номеров последовательностей, а так же хранилища быстроустаревающих или временных данных.