1. Что такое Hadoop?

Проект фонда Apache Software Foundation, свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов. Используется для реализации поисковых и контекстных механизмов многих высоконагруженных веб-сайтов, в том числе, для Yahoo! и Facebook. Разработан на Java в рамках вычислительной парадигмы MapReduce, согласно которой приложение разделяется на большое количество одинаковых элементарных заданий, выполнимых на узлах кластера и естественным образом сводимых в конечный результат.

2. Что такое HDFS?

HDFS (Hadoop Distributed File System) — распределенная файловая система Hadoop для хранения файлов больших размеров с возможностью потокового доступа к информации, поблочно распределённой по узлам вычислительного кластера, который может состоять из произвольного аппаратного обеспечения. Hadoop Distributed File System, как и любая файловая система – это иерархия каталогов с вложенными в них подкаталогами и файлами.

3. Что такое YARN?

Apache Hadoop YARN - это система для планирования заданий и управления кластером. До получения официального названия YARN неофициально назывался MapReduce 2 или NextGen MapReduce. Будучи одним из основных компонентов Apache Hadoop, YARN отвечает за распределение системных ресурсов различным приложениям, работающим в кластере Hadoop, и за и планирование задач, выполняемых на разных узлах кластера.
Архитектура YARN
ResourceManager (RM) - менеджер ресурсов, задачей которого является распределение ресурсов, необходимых для работы приложений, и наблюдение за вычислительными узлами, на которых эти приложения выполняются.
ApplicationMaster (AM) – компонент, ответственный за планирование жизненного цикла, координацию и отслеживание статуса выполнения распределенного приложения. Каждое приложение имеет свой экземпляр ApplicationMaster.
NodeManager (NM) – агент, запущенный на вычислительном узле и отвечающий за отслеживание используемых вычислительных ресурсов (ЦП, RAM и т.д.), за управление логами и за отправку отчетов по используемым ресурсам планировщику менеджера ресурсов ResourceManager/Scheduler.

NodeManager управляет абстрактными контейнерами, которые представляют собой ресурсы на узле, доступные для конкретного приложения.

Контейнер (Container) - набор физических ресурсов, таких как ЦП, RAM, диски и др. в одной ноде.
Управление кластером

Кластер YARN вступает в работу с приходом запроса от клиента. ResourceManager выделяет необходимые ресурсы для контейнера и запускает ApplicationMaster для обслуживания указанного приложения. ApplicationMaster выделяет контейнеры для приложения в каждом узле и контролирует их работу до завершения работы приложения.
Планировщик FIFO - простой планировщик, который распределяет все ресурсы по задачам в порядке поступления на выполнение.
Fair-планировщик - обеспечивает все работающие приложения примерно равными ресурсами. Если используются очереди – то учитывается вес очереди для приоритета по ресурсам.

Capacity-планировщик - для каждой очереди конфигурируется минимальная и максимальная квота по ресурсам. Пользователям кластера выдаются права на очереди, в которых они могут запускать задачи на кластере.

4. Какие минусы или опасные места HDFS?

HDFS обладает рядом отличительных свойств:

- большой размер блока по сравнению с другими файловыми системами (>64MB), поскольку HDFS предназначена для хранения большого количества огромных (>10GB) файлов;

- ориентация на недорогие и, поэтому не самые надежные сервера – отказоустойчивость всего кластера обеспечивается за счет репликации данных;

- зеркалирование и репликация осуществляются на уровне кластера, а не на уровне узлов данных;

- репликация происходит в асинхронном режиме – информация распределяется по нескольким серверам прямо во время загрузки, поэтому выход из строя отдельных узлов данных не повлечет за собой полную пропажу данных;

- HDFS оптимизирована для потоковых считываний файлов, поэтому применять ее для нерегулярных и произвольных считываний нецелесообразно;

- клиенты могут считывать и писать файлы HDFS напрямую через программный интерфейс Java;

- файлы пишутся однократно, что исключает внесение в них любых произвольных изменений;

- принцип WORM (Write-once and read-many, один раз записать – много раз прочитать) полностью освобождает систему от блокировок типа «запись-чтение». Запись в файл в одно время доступен только одному процессу, что исключает конфликты множественной записи.

- HDFS оптимизирована под потоковую передачу данных;

- сжатие данных и рациональное использование дискового пространства позволило снизить нагрузку на каналы передачи данных, которые чаще всего являются узким местом в распределенных средах;

- самодиагностика – каждый узел данных через определенные интервалы времени отправляет диагностические сообщения узлу имен, который записывает логи операций над файлами в специальный журнал;

- все метаданные сервера имен хранятся в оперативной памяти.

В связи с особенностями архитектуры и принципом действия, для HDFS характерны следующие недостатки:

- сервер имен является центральной точкой всего кластера и его отказ повлечет сбой системы целиком;

- отсутствие полноценной репликации Secondary NameNode;

- отсутствие возможности дописывать или оставить открытым для записи файлы в HDFS, за счет чего в классическом дистрибутиве Apache Hadoop невозможно обновлять блоки уже записанных данных;

- отсутствие поддержки реляционных моделей данных;

- отсутствие инструментов для поддержки ссылочной целостности данных, что не гарантирует идентичность реплик. HDFS перекладывает проверку целостности данных на клиентов. При создании файла клиент рассчитывает контрольные суммы каждые 512 байт, которые в последующем сохраняются на сервере имен. При считывании файла клиент обращается к данным и контрольным суммам. В случае их несоответствия происходит обращение к другой реплике.

5. Что такое блок HDFS?

В HDFS отдельные файлы разбиваются на блоки фиксированного размера (128Мб). Эти блоки хранятся в кластере или в одной из машин с большим объемом для хранения данных. Отдельные машины в кластере называются узлами данных (DataNodes). Файл может состоять из нескольких блоков, и они необязательно будут храниться на одной и той же машине; целевые машины, каждая из которых хранит по блоку, выбираются случайно, по принципу «block-by-block». Хотя доступ к файлу может потребовать взаимодействия нескольких машин, но поддержка размера файла куда выше, чем у одиночной машины с DFS; отдельные файлы могут занять больше места, чем способен вместить один жесткий диск.

Если несколько машин должны использоваться в разделе файла, тогда файл может реплицироваться, пока не потеряна ни одна из машин. HDFS борется с этой проблемой путем репликации каждого блока соответственно числу машин (по умолчанию 3).

6. Для чего используется NameNode?

Управляющий узел, узел имен или сервер имен (NameNode) – отдельный, единственный в кластере, сервер с программным кодом для управления пространством имен файловой системы, хранящий дерево файлов, а также мета-данные файлов и каталогов. NameNode – обязательный компонент кластера HDFS, который отвечает за открытие и закрытие файлов, создание и удаление каталогов, управление доступом со стороны внешних клиентов и соответствие между файлами и блоками, дублированными (реплицированными) на узлах данных. Сервер имён раскрывает для всех желающих расположение блоков данных на машинах кластера.

Secondary NameNode — вторичный узел имен, отдельный сервер, единственный в кластере, который копирует образ HDFS и лог транзакций операций с файловыми блоками во временную папку, применяет изменения, накопленные в логе транзакций к образу HDFS, а также записывает его на узел NameNode и очищает лог транзакций. Secondary NameNode необходим для быстрого ручного восстанавления NameNode в случае его выхода из строя.

7. Для чего используется DataNode?

Узел или сервер данных (DataNode, Node) – один их множества серверов кластера с программным кодом, отвечающим за файловые операции и работу с блоками данных. DataNode – обязательный компонент кластера HDFS, который отвечает за запись и чтение данных, выполнение команд от узла NameNode по созданию, удалению и репликации блоков, а также периодическую отправку сообщения о состоянии (heartbeats) и обработку запросов на чтение и запись, поступающих от клиентов файловой системы HDFS. Стоит отметить, что данные проходят с остальных узлов кластера к клиенту мимо узла NameNode.

8. Что будет, если записать много маленьких файлов в HDFS?

Файлы запишутся в соответствии 1 файл - 1 блок.

Вред маленьких файлов

Метаданные небольших файлов занимают много памяти (небольшие файлы, метаданные не маленькие). Ограничьте возможность горизонтального расширения HDFS.

Если используется обработка MR, каждая карта будет обрабатывать блок HDFS. Если в данный момент слишком много файлов для чтения, это приведет к запуску большого количества задач карты.

Чтение и запись небольших файлов в HDFS также требует больше времени. Каждый раз вам необходимо получить информацию из NameNode и установить соединение с соответствующим DataNode.

9. Что будет, если несколько DataNode внезапно отключатся?

NameNode периодически получает отчет от каждого DataNode в кластере. Каждая DataNode отправляет сообщение через каждые 3 секунды NameNode. Отчет о

работоспособности — это просто информация о конкретном DataNode, который работает нормально или нет. Другими словами, мы можем сказать, что конкретный DataNode жив или нет.

Отчет о блоках определенного DataNode содержит информацию обо всех блоках, находящихся в соответствующем DataNode. Когда NameNode не получает сообщения в течение 10 минут (ByDefault) от определенного DataNode, то соответствующий DataNode считается потерянным. Поскольку блоки будут недостаточно реплицированы, NameNode начинает процесс репликации с одного работающего DataNode на другой работающий DataNode, получая всю информацию о блоке из отчета о блоке соответствующего DataNode. Данные для репликации передаются напрямую из одного DataNode в другой без передачи данных через NameNode.

10. Как проадпейдить несколько записей в большом файле на hdfs?

Принцип WORM (Write-once and read-many, один раз записать – много раз прочитать) не позволяет редактировать файлы в hdfs, поэтому проапдейтить несколько записей в файле не получится.

Кроме этого можно отредактировать блок, найдя местоположение блока каждого datanode в кластере. Но это будет проблематично. Также можно попробовать установить HUE. С HUE можно редактировать файлы в hdfs, используя web UI.

11. Почему задачи на YARN нестабильны?

Потому что все зависит какой планировщик работает и какие ресурсы, очереди выделяет для выполнения заданий.

Планировщик FIFO - простой планировщик, который распределяет все ресурсы по задачам в порядке поступления на выполнение.

Fair-планировщик - обеспечивает все работающие приложения примерно равными ресурсами. Если используются очереди – то учитывается вес очереди для приоритета по ресурсам.

Capacity-планировщик - для каждой очереди конфигурируется минимальная и максимальная квота по ресурсам. Пользователям кластера выдаются права на очереди, в которых они могут запускать задачи на кластере.

12. Что такое Hive?

Apache Hive - программное обеспечение, используемое хранилищами данных. Облегчает использование запросов и управление большими объемами данных, находящихся в распределенных хранилищах. Hive предоставляет механизм проектирования структур для этих данных и позволяет создавать запросы с использованием SQL -подобного языка, называемым HiveQL. В то же время этот язык позволяет программистам использовать их собственные запросы, когда неудобно или неэффективно использовать логику в HiveQL.

Возможности

● Работа с данными используя SQL-подобный язык запросов;

● Поддержка различных форматов хранения данных;

● Работа напрямую с HDFS и Apache HBase;

● Выполнение запросов через Apache Tez, Apache Spark или MapReduce.

HiveQL

Apache Hive поддерживает язык запросов Hive Query Language, который основан на языке SQL, но не имеет полной поддержки стандарта SQL-92. HiveQL имеет функции для работы с форматами XML и JSON, поддержку нескалярных типов данных, таких как массивы, структуры, ассоциативные массивы, поддерживает широкий набор агрегирующих функций, определяемые пользователем функции (User Defined Functions), блокировки.

13. Что хранит HiveMetastore?

Хранит метаданные для таблиц Hive — схему на чтение (schema-on-read), расположение, информацию о столбцах в таблице, типы данных, ACL и тд.

14. Чем отличается external table и managed table?

Когда удаляется внутренняя (managed) таблица, удаляются данные таблицы, а также удаляются метаданные таблицы.

Когда удаляется внешняя (external) таблица, удаляются только метаданные, данные остаются в целостности на hdfs. И при необходимости любой пользователь может загрузить заново данные из hdfs в hive.

15. Какие форматы умеет читать Hive?

.txt, .csv, .xml, .json

16. Чем отличается управление ресурсов в Hive и Impala?

Отличаются друг от друга по следующим параметрам:

- Режим обработки данных – Impala обрабатывает SQL-запросы на лету, реализуя интерактивные вычисления в режиме онлайн, тогда как Hive не походит для OLTP-задач, т.к. работает с пакетами данных не в реальном времени.

- Языки разработки – Хайв написан на Java, тогда как Импала – на C++. Это обусловливает более эффективное использование памяти в Impala, несмотря на то в Hive повторно используются экземпляры JVM (Java Virtual Machine), чтобы частично снизить накладные расходы при запуске виртуальной машины. Однако, Impala, разработанный на C ++, иногда может не работать с форматами, написанными на Java.

- Форматы данных – Impala работает с форматами LZO, Avro и Parquet, а Hive – с Plain Text и ORC. При этом обе рассматриваемые системы поддерживают форматы RCFIle и Sequence.

- Вычислительная модель – Impala основана на архитектуре массовой параллельной обработки (MPP, Massive Parallel Processing), благодаря чему реализуется распределенное многоуровневое обслуживание дерева для отправки SQL-запросов с последующей агрегацией результатов из его листьев. Также MPP позволяет Импала распараллеливать обработку данных, поддерживая интерактивные вычисления. В свою очередь, Hive используется технологию MapReduce, преобразуя SQL-запросы в задания Apache Spark или Hadoop.

- Задержка обработки данных (latency) – в связи с разными вычислительными моделями, рассматриваемые системы по-разному обрабатывают информацию. Cloudera Impala выполняет SQL-запросы в режиме реального времени. Для Apache Hive характерна высокая временная задержка и низкая скорость обработки данных. Impala способна работать в 6-69 раз быстрее Хайв с простыми SQL-запросами. Однако, со сложными запросами Hive справляется лучше благодаря LLAP.

- Пропускная способность Hive существенно выше, чем у Impala [6]. LLAP-функция (Live Long and Process), которая разрешает кэширование запросов в памяти, обеспечивает Hive хорошую производительность на низком уровне. LLAP включает долговременные системные службы (демоны), что позволяет напрямую взаимодействовать с узлами данных HDFS и заменяет тесно интегрированную DAG-структуру запросов (Directed acyclic graph) – графовую модель, активно используемую в Big Data вычислениях.

- Кодогенерация – Hive генерирует выражения запросов во время компиляции (compile time), тогда как Impala – во время выполнения (runtime). Для Хайв характерна проблема «холодного старта», когда при первом запуске приложения запросы выполняются медленно из-за необходимости установки подключения к источнику данных. В Импала отсутствуют подобные накладные расходы при запуске, т.к. необходимые системные службы (демоны) для обработки SQL-запросов запускаются во время загрузки (boot time), что ускоряет работу.

- Отказоустойчивость – Хайв является отказоустойчивой системой, которая сохраняет все промежуточные результаты. Это также положительно влияет на масштабируемость, однако приводит к снижению скорости обработки данных. В свою очередь, Impala нельзя назвать отказоустойчивой платформой.

- Безопасность – в Хайв отсутствуют инструменты обеспечения кибербезопасности, тогда как Impala поддерживает аутентификацию Kerberos.

- Основные пользователи – с учетом отказоустойчивости и высокой пропускной способности Хайв, эта система более адаптирована для промышленной эксплуатации в условиях высоких нагрузок, когда допустима некоторая временная задержка (latency). Поэтому она в большей степени востребована у инженеров больших данных (Data Engineer) в рамках построения сложных ETL-конвейеров. А быстрая и безопасная, но не

слишком надежная Impala лучше подходит для менее масштабных проектов и пользуется популярностью у аналитиков и ученых по данным (Data Analyst, Data Scientist).

17. Чем отличается колочный формат хранения данных от строчного?

В линейных форматах (AVRO, Sequence) строки данных одного типа хранятся вместе, образуя непрерывное хранилище. Даже если необходимо получить лишь некоторые значения из строки, все равно вся строка будет считана с диска в память. Линейный способ хранения данных обусловливает пониженную скорость операций чтения и выполнении избирательных запросов, а также больший расход дискового пространства. Линейные форматы, в отличие от колоночных, не являются строго типизированными: например, Apache AVRO хранит информация о типе каждого поля в разделе метаданных вместе со схемой, поэтому для чтения сериализованных данных информации не требуется предварительное знание схемы. Бинарный формат файлов последовательностей (Sequence File) для хранения Big Data в виде сериализованных пар ключ/значение в экосистеме Apache Hadoop также содержит метаданные в заголовке файла. Отметим, что формат Sequence File отлично обеспечивает параллелизм при выполнении задач MapReduce, т.к. разные порции одного файла могут быть распакованы и использованы независимо друг от друга. Тем не менее, степень сжатия информации у строковых форматов ниже, чем у столбцовых. Однако, именно линейно-ориентированные форматы лучше всего подходят для потоковой записи, т.к. в случае сбоя информация может быть восстановлена (повторно синхронизирована) с последней точки синхронизации.

В колоночно-ориентированных форматах (Parquet, RCFile, ORCFile) файл разрезается на несколько столбцов данных, которые хранятся вместе, но могут быть обработаны независимо друг от друга. Такой метод хранения информации позволяет пропускать ненужные столбцы при чтении данных, что существенно ускоряет чтение данных и отлично подходит в случае, когда необходим небольшой объем строк или выполняются избирательные запросы, как, например, в СУБД Apache Hive. Но такой формат чтения и записи занимает больше места в оперативной памяти, поскольку, чтобы получить столбец из нескольких строк, кэшируется каждая строка. Также колоночно-ориентированные форматы не используются в средах потоковой обработки (Apache Kafka, Flume), т.к. после сбоя записи текущий файл не сможет быть восстановлен из-за отсутствия точек синхронизации. Однако, колоночные файлы занимают меньше места на жестком диске вследствие более эффективного сжатия информации по столбцам.

18. Чем отличается parquet/orc от csv?

Сжатием, кодированием и метаданными.

19. Чем отличается Avro от json?

Сжатием, кодированием. У Avro метаданные могут храниться в json, что облегчает чтение и интерпретацию данных.

20. Чем отличается документориетированный формат данных от реляционного?

Реляционные базы данных обычно хранят данные в отдельных таблицах и один объект может быть распределен по нескольким таблицам. Документоориентированный формат хранит всю информацию для данного объекта в единственном экземпляре в формате документов (MongoDB), и каждый хранимый объект может отличаться от любого другого. Это устраняет необходимость объектно-реляционного сопоставления при загрузке данных в базу данных.

21. Чем отличается etl и elt?

ETL (Extract, Transform, Load) сначала извлекают данные из пула источников данных. Данные хранятся во временной промежуточной базе данных. Затем выполняются операции преобразования, чтобы структурировать и преобразовать данные в подходящую форму для целевой системы хранилища данных. Затем структурированные данные загружаются в хранилище и готовы к анализу.

В случае ELT (Extract, Load, Transform) данные сразу же загружаются после извлечения из исходных пулов данных. Промежуточная база данных отсутствует, что означает, что данные немедленно загружаются в единый централизованный репозиторий. Данные преобразуются в системе хранилища данных для использования с инструментами бизнес-аналитики и аналитики (Яндекс Метрика, Google Analytics).

22. Какие основные челенджы etl?

Форматы данных меняются с течением времени

Увеличение скорости передачи данных

Временные затраты на добавление новых источников данных

Затраты времени на исправление поврежденных источников данных

Запросы на новые функции, включая новые столбцы, размеры и производные

Очистка данных

Нормализация данных

Выбор правильных компонентов и их субкомпонентов может иметь независимые технические решения, которые будут влиять на масштабируемость, функциональность и нагрузку на обслуживание в будущем.

Масштабируемость

23. Какие инструменты etl вы знаете?

Apache Flume

Apache NiFi

Fluentd

StreamSets Data Collector

AWS Glue

Alooma

Stitch

Fivetran

Matillion

Talend Open Studio

Informatica PowerCenter

Blendo

IRI Voracity

Azure Data

Logstash

SAS

Pentaho

Etleap

Singer

Apache Camel

Actian DataConnect

Qlik

IBM infosphere Datastage

Oracle Data Integrator

24. Для чего нужны key-value СУБД?

Базы «ключ-значение» часто используют для кэширования данных и организации очередей.

Их достоинства — быстрый поиск и простое масштабирование.

Их недостаток — нельзя производить операции со значениями. Например — сортировать их или анализировать.

Базы «ключ-значение» применяют в поисковой системе Google, «Википедии», «Фейсбуке», интернет-магазине Amazon.

Одна из самых популярных — Redis. Её используют Uber, Slack, Stack Overflow, сайты гостиниц и туристические, социальная сеть Twitter.

25. Какие сложности стриминга в hdfs?

Файлы небольшого размера (<128Мб)

Множество источников данных

26. Какие минусы key-value хранилищ?

Их недостаток — нельзя производить операции со значениями. Например — сортировать их или анализировать.

Как правило, хранит данные в оперативной памяти, нужно будет позаботиться о регулярной выгрузке на жёсткий диск и о дальнейшем расширении оперативной памяти.

27. Из чего состоит хранилище данных?

Реализуется в виде следующих уровней:

- операционный слой первичных данных(Primary Data Layer или стейджинг), на котором выполняется загрузка информации из систем-источников в исходном качестве и сохранением полной истории изменений. Здесь происходит абстрагирование следующих слоев хранилища от физического устройства источников данных, способов их сбора и методов выделения изменений.

- ядро хранилища (Core Data Layer) – центральный компонент, который выполняет консолидацию данных из разных источников, приводя их к единым структурам и ключам. Именно здесь происходит основная работа с качеством данных и общие трансформации, чтобы абстрагировать потребителей от особенностей логического устройства источников данных и необходимости их взаимного сопоставления. Так решается задача обеспечения целостности и качества данных.

- аналитические витрины (Data Mart Layer), где данные преобразуются к структурам, удобным для анализа и использования в BI-дэшбордах или других системах-потребителях. Когда витрины берут данные из ядра, они называются регулярными. Если же для быстрого решения локальных задач не нужна консолидация данных, витрина может брать первичные данные из операционного слоя и называется соответственно операционной. Также бывают вторичные витрины, которые используются для представления результатов сложных расчетов и нетипичных трансформаций. Таким образом, витрины обеспечивают разные представления единых данных под конкретную бизнес-специфику.

- Наконец, сервисный слой (Service Layer) обеспечивает управление всеми вышеописанными уровнями. Он не содержит бизнес-данных, но оперирует метаданными и другими структурами для работы с качеством данных, позволяя выполнять сквозной аудит данных (data lineage), использовать общие подходы к выделению дельты изменений и управления загрузкой. Также здесь доступны средства мониторинга и диагностики ошибок, что ускоряет решение проблем.

28. Какие виды хранилищ данных вы знаете?

Иерархические

Сетевые

Объектно-ориентированные

Реляционные

Гибридные

NoSQL - формата ключ-значение, документоориентированные, колоночные, графовые.

29. Основные задачи Data governance?

Основные задачи Data Governance — увеличить отношение количества нужных данных к ненужным, повысить качество корпоративных данных и обеспечить эффективность их повторного использования. Кроме того, дисциплина включает в себя защиту «чувствительных» конфиденциальных данных, обеспечение доверия к ним, а также управление доступом к данным и их жизненным циклом.

Повышение качества

Низкое качество данных – это проблема не только для сотрудников отделов информационных технологий и аналитики, а бизнеса в целом. Хотя технология обеспечения качества данных внедряется и обслуживается IT-отделом, основную выгоду получают другие подразделения. Например, неверная разметка акционных продаж отделом маркетинга может привести к ошибкам при прогнозировании объёма товара на следующую акцию, поэтому важно внедрить в рабочий процесс контроль за качеством данных. Это позволит сократить время, которое тратится на то, чтобы привести данные в порядок: убрать дубликаты, привести к одному формату, заполнить пропуски, почистить шумы и т.д.

Обеспечение целостности и доступности

Ни одна система не будет эффективно работать, если у неё нет доступа к необходимой информации. Кроме того, нужно обеспечить работоспособность системы, которая предоставляет доступ пользователям к информации. Для этого рекомендуется использовать облачные хранилища и отказоустойчивые базы данных. Данные могут быть потеряны – случайно или намеренно стёрты с носителя, поэтому необходимо делать резервные копии.

Контроль

Внедрение процессов Data Governance позволяет получить полный контроль над данными. Можно контролировать, где и в каком формате они хранятся, обеспечивать версионность, поддерживать актуальность данных, определить правила доступа к данным и т.д.

Обеспечение согласованности

Если разные сотрудники будут работать с одними и теми же данными, но при этом не синхронизировать их между собой, это может привести к неверным результатам. Можно разместить всю информацию в одном общем хранилище, например, в базе данных или облаке. Это позволит сотрудникам работать с актуальной информацией, которая едина для всех.

Унификация

Накопление информации из множества источников приводит к получению данных в разрозненном виде, их нужно приводить к единому виду и формату. Задача усложняется, если речь идёт о тысячах строк. Каждый сотрудник может по-своему записать одно и то же значение. Например, «500 грамм» и «0.5 кг» воспринимаются человеком одинаково, но для машины это совершенно разные значения. И чтобы система понимала их правильно, необходимо привести все данные к одному формату, а это может занять много времени. Продуктивнее заранее определить политику работы с данными, стандарт, которому будут следовать все сотрудники. Таким образом, даже новый сотрудник или аналитик на стороне сможет понимать данные, и работа будет эффективнее.