МСА:

Факторный анализ

Многомерное шкалирование

Дискриминантный анализ

Кластерный анализ

Коэф корреляции от -1 до 1, показывает силу связи между переменными (прямая или обратная), если нет зависимой переменной, функционально зависящей от всех других, то делаем множественную корреляцию

ФА - процедура, с помощью которой хорошо коррелируемые между собой переменные объединяют в меньшее количество независимых друг от друга величин.

Факторный анализ классифицирует признаки, описывающие наблюдения.

Factor - скрытая переменная, конструируемая так, чтобы можно было объяснить корреляцию между набором имеющихся переменных.

Целью ФА является сокращение числа переменных на основе их классификации и определения структуры взаимосвязей меж них.

У нас появляются возможности анализировать данные по факторам, что дает сокращение числа исходных взаимосвязей

Смысл ФА - измеряемые эмпирические показатели считаются следствием других, более глубинных, скрытых от прямого измерения характеристик.

Например: расчет прибыли от продаж компании

Прибыль зависит от объема, ассортимента продукции, себестоимости продукции и цены реализации.

С помощью факторки можно выяснить, что влияет на прибыль больше всего и выработать пути максимизации прибыли.

Условия ФА:

1. Все признаки должны быть количественными переменными (интервал\метрика)

2. Число наблюдений > числа переменных в 2+ раз

3. Выборка однородна

4. Исходные переменные распределены симметрично

5. Номинальные переменные должны быть переведены в дихотомические ( не больше 2 категорий)

6. Ненулевая корреляция переменных, но и не близка к 1.

Разновидности:

1. Анализ главных компонент, подходит больше, если основная цель - сократить число переменных.

2. Анализ главных факторов, больше подходит для поиска структуры переменных, опред. иерархии.

Методы: невзвеш\обобщенный МНК, максимум правдоподобия, факторизация главной оси, альфа факторизация, анализ образов

Этапы:

1. Вычисление корр матрицы для переменных, уч. в анализе

2. Извлечение факторов

3. Выбор факторов и вращение для создания упрощенной структуры.

4. Интерпретация факторов

Извлечение факторов начинается с подсчета суммарного разброса значений всех переменных в анализе. Логическая интерпретация суммарного разброса - строго опред. математическая величина, которую непросто найти.

1. Извлечение факторов заключается в выборе взаимодействующих перменных, чья взаимная корреляция обуславливает долю общей дисперсии. Они образуют первый фактор

2. Затем этот первый фактор исключается и снова выбираются переменные, чье взаимодействие даёт большую долю оставшейся общей дисперсии. Они образуют второй фактор

3. Процедура извлечения факторов продолжается, пока не кончится общая дисперсия

4. Напротив каждого фактора появляется число, пропорциональное доле общей дисперсии приход. на фактор.

Метод главных компонент обусловлен тем, что основная дисперсия лежит в первых нескольких факторах (компонентах), и можно для анализа брать не все, а только их. При всем при этом они независимы и, например, 1 и 2 - ортогональны.

Выбор факторов.

1. Если факторов столько же, сколько и переменных - смысла факторного анализа нет

2. Нужно выбрать, что оставить, с помощью:

а) формальных критериев, если собственные значения превышают единицу - фактор оставляем

б) критерий каменистой осыпи Кеттелла - поиск точки, где убывание замедляется наиболее сильно

3. Вывод о числе факторов принимается после их интерпретации, исходя из этого можно понять, что они выбираются не однократно.

После интерпретации факторов вычисляются их значения. ФА имеет свою ценность, но если нам нужно уменьшить количество переменных для многомерного анализа, то стоит вычислить значения каждого фактора для респондента. Это линейная комбинация исходных переменных. Значение для f-iтого фактора вычисляется по формуле:

Fi = bi1X1 + bi2X2 + ... + bikXk

Коэффициент взаимосвязи между некоторым признаком (Х) и общим фактором (F), выражающий меру влияния фактора на признак – факторная нагрузка данного признака по данному общему фактору.

Матрица, состоящая из факторных нагрузок и имеющая число столбцов, равная числу общих факторов, и число строк равное числу исходных признаков, называется факторной матрицей. Факторная матрица фиксирует степень линейной связи каждого признака с каждым общим фактором.

Величина факторной нагрузки не превышает по модулю 1, а знак говорит о положительной/отрицательной связи признака с фактором. Значение факторной нагрузки, близкое к нулю – фактор практически не влияет на данный признак.

Можно вычислить вклады факторов в общую дисперсию всех признаков (путем вычисления суммы квадратов факторных нагрузок для каждого фактора по всем признакам). Чем выше доля этого вклада в общей дисперсии, тем более значимым является данный фактор.

Оценки качества модели ФА:

Критерий сферичности Бартлетта — показатель, с помощью которого

проверяют, отличаются ли корреляции от 0 Если r близко к нулю, то

выбранная переменная не взаимосвязана с другими. Значимость меньше

0,05 указывает, на то что проведение факторного анализа приемлемо.

Корреляционная матрица — матрица, включающая в себя все возможные

коэффициенты корреляций r между анализируемыми переменными.

КМО — мера адекватности выборки Кайзера—Мейера—Олкина - величина, используемая для оценки применимости факторного анализа.

Значения от 0,5 до 1 говорят об адекватности факторного анализа, значения

до 0,5 указывают на то, что факторный анализ неприменим к выборке.

Графическое изображение критерия “каменистой осыпи” - график собственных значений факторов, расположенных в порядке убывания, используется для определения достаточного числа факторов.

Если распределение не нормальное, связь переменных нелинейная, выборка небольшая, то используем:

Многомерное шкалирование (MultiDimentional Scaling - MDS)

На основе сходства (любых дистанций) между наблюдениями позволяет расположить их в пространстве нескольких новых факторов так, чтобы факторы объясняли как можно больше изменчивости.

Но если данные более-менее удовлетворяют требованиям факторного анализа, лучше проводить его, так как:

1 Факторный анализ - гораздо более мощная процедура, намного лучше оценивает связи исходных переменных

2 Результаты гораздо проще интерпретировать: в многомерном шкалировании очень трудно объяснить, что же значат полученные факторы

Номинальная шкала (Nominal) ― шкала наименований, которая состоит из значений признаков, не упорядоченных по степени возрастания или убывания. Пример: национальность, профессия, семейное положение, пол и т.д.

Порядковая шкала (Ordinal) ― градации располагаются в определенном порядке относительно возрастания либо убывания интенсивности свойства. Пример: переменная «Курение» со значениями (1 = некурящий; 2 = изредка курящий; 3 = интенсивно курящий; 4 = очень интенсивно курящий ). Переменная сортирована в порядке значимости снизу вверх: умеренный курильщик курит больше, нежели некурящий, а сильно курящий — больше, чем умеренный курильщик и т.д., поэтому порядковая шкала.

Интервальные шкалы (Interval) ― основаны на процедурах, обеспечивающих равные или примерно равные расстояния между градациями переменной. В данном случае сравниваются не значения переменных, а расстояния между значениями. Пример: температура, измеренная в градусах Цельсия. Можно не только сказать, что температура 30 градусов выше, чем 20 градусов, но и то, что увеличение температуры с 10 до 30 градусов.

Шкалы отношений (Метрические) ― соответствуют всем требованиям, предъявляемым к шкалам более низких классов. Пример: возраст. Если Максу 30 лет, а Сергею 60, можно сказать, что Сергей вдвое старше Макса.

Цель МШ:

Уменьшить число исходных переменных с минимальными потерями информации. Новые переменные получаются на основе матрицы дистанций между объектами так, чтобы дистанции в пространстве новых переменных изменились как можно меньше. То, что было «похожим», должно остаться похожим; большие исходные различия должны остаться большими. Многомерное шкалирование (МШ) позволяет представить большой массив данных о различии объектов в графическом виде. Если две точки на изображении удалены друг от друга, то между соответствующими объектами имеется значительное расхождение, если же две точки на изображении расположены близко друг к другу, то между соответствующими объектами наблюдается сходство.

Идея многомерного шкалирования заключается в исследовании близости и различия, величин, обратных друг другу; алгоритм МШ ищет такое числовое признаковое пространство и такое расположение рассматриваемых объектов в пространстве, что структура расстояний между ними отвечает структуре исходной матрицы близостей.

Шкала - ось теоретического пространства, которая является носителем значений обобщенного признака(фактора). Для комплексного восприятия сложной реальности требуется многомерное пространство шкал.

Функция стресса - критерий различий матрицы близостей и матрицы расстояний. От 0 до 1, 0 - полное сходство, 1 - сходства нет.

Матрица евклидовых расстояний между объектами в этих координатах оказалась близка к матрице расстояний между объектами, определенной из каких-либо соображений ранее.

Евклидово расстояние - дистанция в прост переменных, как в геометрии (корень из суммы (y1j - y2j)^2) от 1 до p.

Квадрат евклидова расстояния увеличивает вес больших разностей

Манхэттенское расстояние увеличивает вес больших разниц, берется суммой abs(y1j - y2j)

Этапы МШ:

В ходе анализа решаются следующие вопросы:

1. Поиск и интерпретация переменных общих факторов

2. Сжатие исходного массива данных

3. Визуализация геометрической конфигурации наблюдаемых объектов в координатном пространстве латентных признаков.

Функция стресса

Если элементы матрицы различий можно считать числами, полученными по интервальной шкале, то можно рассмотреть разницу omega(ij) (элемент матрицы различий) и d(ij). Функция стресса строится с принципом наименьших квадратов

S = summ(1<=i, j<=n; (d(ij) - omega(ij))^2)

Чем ближе каждое раастояние dij к различию, тем еньше значений этой функции.

Точное решение МШ соответствует случаю, когда S = 0.

Этап 1. Подготовительный

Трансформируем данные, если нелинейные связи и стандартизируем, если переменные измерены в принципиально разных шкалах.

Получаем матрицу n\*n дистанций между объектами

Определяем сколько новых переменных мы будем получать

Если матрица различий имеет евклидовы дистанции, и данные центрированы по всем переменным так, что среднее = 0, взаиморасположение будет таким же, как и в ФА в пространстве новых компонент. => classical scaling - обобщенный вариант РСА.

2 этап. Получение новых переменных.

Из матрицы дистанций получаем новые переменные. Первая - описание максимума различий между объектами

Объекты помещаются в пространство k новых переменных(первых, лучших) и многократно поворачиваются там, приближая дистанции между объектами в новом пространстве(d-hat) к реальным различия между объектами(d).

т.е. это итеративная процедура, новые переменные получаются не одним действием из матриц, а они постепенно подгоняются к данным.

Примечание про проблему сходимости (convergence problem).

Этап 3. Оценка качества модели.

Считается простой показатель stress, показывающий насколько модель описывает реальные различия между объектами, чем он меньше - тем он лучше. >0,3 недопустимо.

Картинка, показ. качество модели - диаграмма Шепарда.

Если стресс большой, повторяем процедуру с увеличением числа измерений - новых переменных. Стресс оценивается на основе residuals в регрессионной модели (дистанции = реальные различия).

Диаграмма Шепарда позволяет сказать, насколько сильно искажены расстояния на карте сходства. Это точечный график. По горизонтальной оси исходные расстояния R(ij); по вертикальной евклидовы d(ij). Каждая точка соотвествует некоторой паре объектов ij внутри D.

Этап 4. Интерпретация реузльтатов.

Недостаток метода - отсутствие готового способа выяснить вклад переменных в новые и их корреляцию с ними.

Все потому что новые переменные не линейная комбинация исходных, и получены они из дистанций.

Допустимо проанализировать обычные корреляции исходных переменных с новыми измерениями.

Теперь можно получить для каждого объекта значения новых переменных и исп. их в дальнейшем анализе. (сравнение групп, к примеру)

Требования к выборкам:

1. В качестве исходных данных можно взять любую матрицу дистанций, ограничений нет.

2. Для интерпретации новых переменных желательно, чтобы связь переменных была линейной (диагностика - осмотр скаттерплотов и возможно трансформация)

3. Желательно исключить аутлаеры, если они меняют результат

4. Если переменных измерены в разных шкалах, их нужно стандартизировать.

5. Следует исключить переменных, сильно коррелирующие с другими. MDS - аналог метода главных компонент, не имеющий ограничений РСА, но его возможности меньше. Лучше им пользоваться в крайнем случае

Дискриминантный анализ

Это раздел многомерного статистического анализа, который позволяет предсказать принадлежность объектов к двум или более непересекающимся группам.

Исходные данные - множество объектов, разеделнных на группы так, что каждый объект может быть отнесен только в одну группу. Для каждого объекта данные меняются по ряду количественных переменных. Такие переменные - дискриминантные или предикторы.

Задачами ДА является определение:

1. Решающих правил, позволяющих по значениям дискриминантных переменных отнести каждый объект к одной из известных групп

2. "веса" каждого предиктора для разделения объектов на группы

Ядром ДА является построение дискриминантной функции: d = b1x1 + b2x2 + .... + bnxn + a

b1 - bn и а - коэффициенты которые и предстоит оценить с помощью ДА, в то время как xn - значения переменных, соотв случаям.

Необходим определить такие коэффициенты, чтобы по значениям ДФ можно было с максимальной четкость провести разделение по группам. Затем на основе этого же правила отнести респондента или их множество к свойственной им группе.

Примером использования ДА является работа HR специалиста.

ДА делится на два вида:

анализ для двух групп и множественный

(two-group discriminant analysis - зависимая переменная имеет две категории; multiple discriminant analysis - три и больше категорий)

Целью ДА является разделение совокупности объектов на несколько более мелких групп

Процесс классификации в ДА - изначально заданы количество и состав классов и основная задача заключается в определении того, насколько точно можно предсказать принадлежность объектов к классам при помощи данного набора предикторов.

Предсказание значение зависимой функции

Определение, какие переменные лучше всего подходят для предикта.

ДА основан на составлении уравнения регрессии, использующего номинальную зависимую переменную

Уравнение регрессии составляется на основе тех объектов, о которых известна групповая принадлежность, что позволяет максимально точно подобрать его коэффициенты.

После того, как уравнение регрессии получено, его можно использовать для группировки интересующих нас объектов в целях прогнозирования их принадлежности к какому-либо классу.

Условия ДА:

1. Зависимая переменная должна быть категориальной, а предикторы интервальными

2. Необходимо наличие тестовой или проверочной ( validation sample) выборки и соответствие ее анализируемой выборке (analysis sample).

Этапы:

1. Разделение выборки на две части

2. Выбор переменных - предикторов

3. Вычисление параметров дискриминантной функции

4. Интерпретация результатов

Разделение выборки:

1. Анализируемая - часть общей выборки, которую используют для вычисления дискриминантной функции.

2. Проверочная - часть общей выборки, которую используют для проверки результатов расчета на основании анализируемой выборки.

Выбор переменных в ДА:

Если число переменных достаточно велико, например несколько сотен, то не представляется возможным применить дискриминантный анализ ко всем переменных одновременно.

Поэтому:

На начальном этапе дискр. анализа для предикторов формируется корр матрица

Помимо корр матрицы можно также вычислить ковариационные матрицы для отдельных групп, для всей выборки либо общую внутригрупповую ковариационную матрицу.

Кроме того:

Нередко исследователи применяют серию t-критериев между двумя группами для каждой переменной либо однофакторный дисперсионный анализ, если число групп оказывается больше двух.

Поскольку целью дискриминантного анализа является составление наилучшего уравнения регрессии, дополнительный анализ исходных данных никогда не является лишним.

Вычисление функции.

Метод принудительного включения (direct method): дискриминантную функцию вычисляют при одновременном введении всех предикторов. В этом случае учитывается каждая независимая переменная.

Пошаговый ДА:

1. Предикторы вводятся последовательно, в зависимости от их способности различать группы.

2.Основан на минимизации коэф Уилкса(лямбда) после включения в уравнение регрессии каждого нового предиктора.

3. Лямбда Уилкса - критерий используемый при проведении теста и показывающий, значимо ли различаются между собой средние значения дискриминантной функции в исследуемых группах, оставляем предикторы только если значимость < 0,005

Этот метод лучше применять в ситуации, когда исследователь хочет отобрать подмножество предикторов для включения их в дискр функцию

Интерпретация результатов

Целью ДА является составление уравнения регрессии с использованием выборки, для которой известны значения и предикторов, и критерия. Это уравнение позволяет по известным значениям предикторов определить неизвестные значения критерия для другой выборки.

Дискриминантную функцию рассчитывают для объектов, изначально разделенных на группы.

Подставляя в классификационные функции значения переменных для объекта, можно для него посчитать их значения (classification score) и отнести в ту или иную группу - предсказать, к какой группе он относится и оценить точность предсказания.

Кластерный анализ

Предназначен для разбиения исходных данных на поддающиеся интерпретации группы, таким образом, чтобы элементы входящие в одну группу были максимально схожи, а элементы из разных групп были максимально отличными друг от друга.

КА - группа методов, используемых для классификации объектов или событий в относительно гомогенные группы, которые называют кластерами

Цель - объединить эти объекты в группы, внутри которых они будут более сходными, чем объекты из разных групп.

Результат - получение дендрограммы (иерархического дерева) - только картинка, никакой проверки гипотез и тд

В ФА группируются столбцы, т.е. цель - анализ структуры множества признаков и выявление обобщенных факторов

В КА - группируются строки, т.е. цель - анализ структуры множества объектов.

КА выполняет кластеризацию объектов

Каждый объект - точка в пространстве признаков

Задача кластерного анализа - выделение сгущений точек, разбиение совокупности на однородные подмножества объектов (сегментация).

Кластерный анализ бывает:

1. Иерархический ( при маленьком кол-ве наблюдений), когда объекты по одному присоединяются к кластеру и он становится больше и больше

2. Анализ к-средних (при большом кол-ве наблюдений),

когда объекты присоединяются по одному, но кластеры перегруппировываются в процессе анализа

Иерархический КА:

Каждое наблюдение образовывает сначала свой отдельный кластер.

На первом шаге анализа два соседних кластера объединяются в один.

Этот процесс продолжается, пока не останутся только два кластера

Расстояние между кластерами является средним значением всех расстояний между всеми возможными парами точек из обоих кластеров.

Этапы:

Формулировка проблемы, выбор способа измерения расстояния, выбор метода кластеризации, принятие решения о кол-ве кластеров, интерпретация и профилирование кластеров, оценка достоверности кластеризации.

Расстояния:

Самой распространенной мерой для определения расстояния между двумя точками на плоскости, образованной координатными осями х и у, является Евклидова мера.

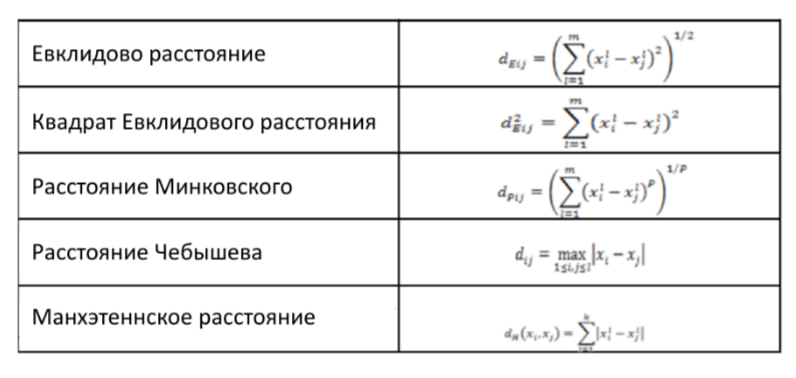
Благодаря возведению в квадрат лучше учитываются большие разности. Эта мера должна всегда использоваться при построении кластеров центройдным, медианным методом или методом Варда.

d(X,Y) = summ(1,m; (Xi-Yi)^2)

Расстояние Чебышева - вычисление расстояние как максимума абсолютного значения разности между элементами. Используется при определении двух объектов как различные, если они отличаются по какому-то одному измерению.

Расстояние Минковского - равно корню в степени из суммы абсолютных разностей пар значений взятых в степени. Степень разности значений можно выбрать в пределах от 1 до 4. Если эту степень взять равной 2, то получим евклидово расстояние.

Расстояние Манхэттена или Хемминга - определяется суммой абсолютных разностей пар значений.



КА Кластеризация

Метод кластеризации - это способ вычисления расстояний между кластерами.

Существуют следующие основные методы кластеризации:

Межгрупповая связь(Between-group linkage) - дистанция между кластерами, которая равна среднему значению всех расстояний между всеми возможными парами точек из обоих кластеров. Необходимая для расчета дистанции информация находится на основании всхе теоретических возможных пар наблюдений

Внутригрупповая связь (Within-groups linkage) - дистанция между двумя кластерами, рассчитывается на основании всех возможных пар наблюдений, принадлежащих обоим кластерам, причем учитываются также и пары наблюдений, образующиеся внутри кластеров

Ближайший сосед (Nearest neighbor) - дистанция между двумя кластерами определяется как расстояние между парой наблюдений, расположенных друг к другу ближе всего, причем каждое наблюдение берется из своего кластера.

Самый дальний сосед (Furthest neighbor) - дистанция между двумя кластерами определяется как расстояние между парой наблюдений, расположенных друг к другу дальше всего, причем каждое наблюдение берется из своего кластера.

Центроидная кластеризация (Centroid clustering) - в обоих кластерах рассчитываются средние значения переменных, относящихся к ним наблюдений. Затем расстояние между кластерами рассчитывается как дистанция между двумя усредненными наблюдениями

Медианная кластеризация(Median clustering) - тот же центроидный, но центр вычисляется как среднее всех объектов.

Метод Варда(Ward's method) - сначала в обоих кластерах для всех имеющихся наблюдений производится расчет средних значений отдельных переменных. Затем вычисляются квадраты евклидовых расстояний от отдельных наблюдений каждого кластера до этого кластерного значения. Эти дистанции суммируются, потом в один новый кластер объединяются те кластеры, которые дают наименьший прирост общей суммы дистанций.

КА Стандартизация

Согласно формуле евклидовой меры, переменная, имеющая большие значения, практически полностью доминирует над переменной с малыми.

Для решения это проблемы исп. z-преобразования (стандартизация) значений переменных, которое приводит значения всех преобразованных переменных к единому диапазону значений

Кластеры

1. Необходимо руководствоваться практическим и теоретическими соображениями. Исходя из цели исследования, например, может быть задано нужное количество кластеров.

2. В иерархической кластеризации в качестве критерия исп. расстояния. Необходимо смотреть на коэффициент в протоколе объединения (расстояние между двумя кластерами, опред. на основании выбранной дистанционной меры с учетом предусмотренного преобразования значений).

Когда мера расстояния между кластерами увеличивается скачкообразно, процесс создания новых кластеров нужно остановить. Иначе будут объединены кластеры, наход. на большом расстоянии друг от друга.

Оптимальным считается кол-во кластеров, равное разности кол-ва наблюдений и кол-ва шагов, после которого коэфф увеличивается скачкообразно.

3. Размеры кластеров должны быть значимыми.

Оценка качества

1. Необходимо выполнять кластерный анализ одних и тех же данных, но с использованием разных способов измерения расстояния.

2. Сравнить результаты, полученные различными способами расстояния, чтобы определить насколько совпадают полученные результаты.

3. Разбить данные на две равные части случайным образом. Выполнить кластерный анализ отдельно для каждой половины. Сравнить кластерные центроиды двух подвыборок.

4. Случайным образом удалить некоторые переменные. Выполнить кластерный анализ по сокращенному набору переменных. Сравнить результаты с полученными на основе полного набора переменных.

КА метод K-средних

Кластеризация К-средних - метод разделения множества данных на К различных непересекающихся кластеров. Для выполнения кластеризации сначала нужно определить желаемое число кластеров К, затем алгоритм К-средних будет относить каждое наблюдение в точности к одному и К кластеров.

Следующий алгоритм находит локальный минимум задачи - достаточно хорошее решение.

1. Произвольным образом присваиваем номер от 1 до К каждому из наблюдений. Это послужит начальным разделением на классы.

2. повторяем пока разбиение на классы не перестанет меняться:

а) Для каждого из К кластеров вычисляем кластерный центроид. Центроид К-го кластера - вектор средних р характеристик для наблюдения в к-м кластере.

б) относим каждое наблюдение к тому кластеру, чей центроид ближайший ( где ближайший определяется с использованием евклидова расстояния).

Сначала определяется центр кластера, а затем группируют все объекты в пределах заданного от центра порогового значения.

Недостатки:

1. Чувствительность к выбросам

2. Необходимо заранее задавать кол-во кластеров, а не как в иерархическом анализе, получать это в качестве результата

Проблему с выбором числа кластеров можно преодолеть проведением иерархического анализа со случайно отобранной выборкой наблюдений и, таким образом, определить оптимальное кол-во кластеров.

Достоинства:

1. простота использования

2. в кач-ве метрики используется евклидово расстояние

3. возможность наглядной интерпретации кластеров с использованием графика "средних значений в кластерах"

Требования к выборкам

1. В кач-ве исходных данных можно взять любую матрицу дистанций - никаких ограничений и требований к выборке

2. Нет необходимости исключать аутлаеры

3. Перменные, измеренные в разных шкалах, должны быть стандартизированы.

Архитектура хранения и загрузки данных.

Оперативная обработка транзакций (OLTP) — это управление данными о транзакциях с помощью компьютерных систем. Системы OLTP записывают операции обмена данными в организации, выполняющиеся каждый день, и поддерживают запрашивание этих данных, чтобы на их основе делать выводы.

Оперативная аналитическая обработка (OLAP) — это технология, которая упорядочивает большие коммерческие базы данных и поддерживает сложный анализ. Ее можно использовать для выполнения сложных аналитических запросов без негативного воздействия на системы транзакций.

Базы данных, в которых компании хранят свои транзакции и записи, называются базами данных оперативной обработки транзакций (OLTP).

Хранилище данных

Data Warehouse (DWH) — хранилище, предназначенное для сбора и аналитической обработки исторических данных организации. Анализ помогает руководителям видеть цельную картину бизнеса и принимать решения, как развивать отдельные направления или бизнес в целом.

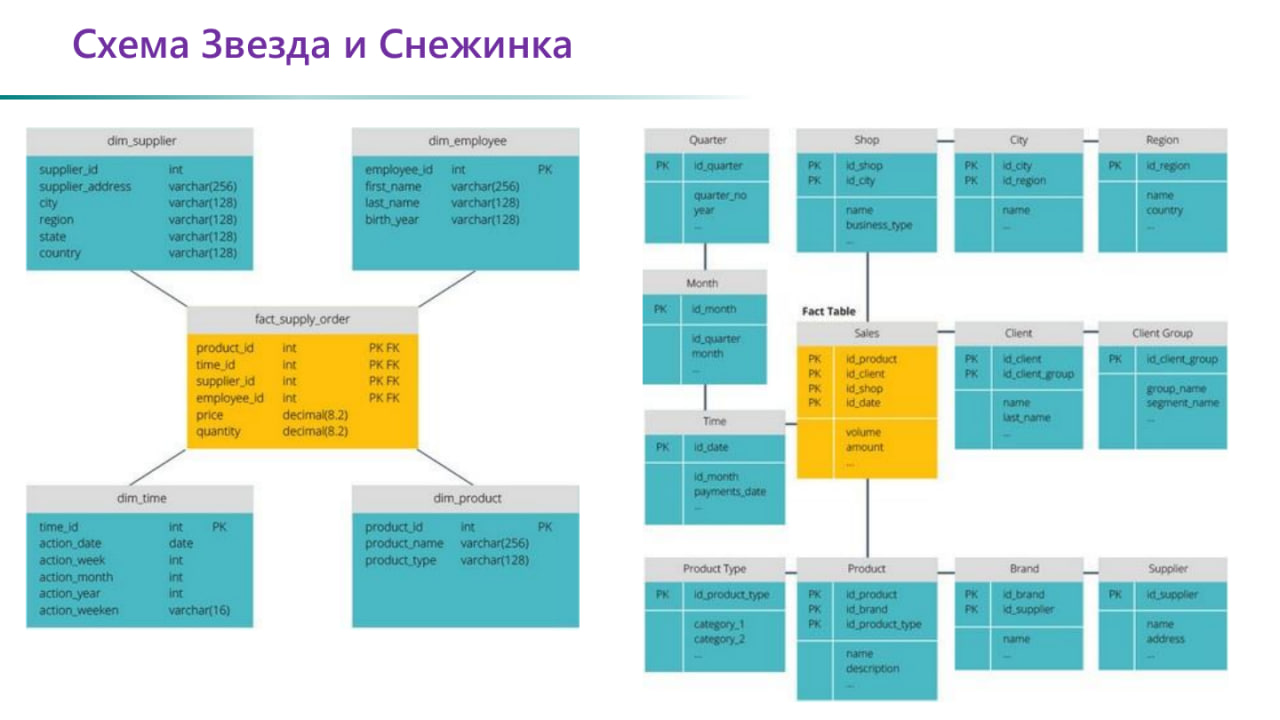
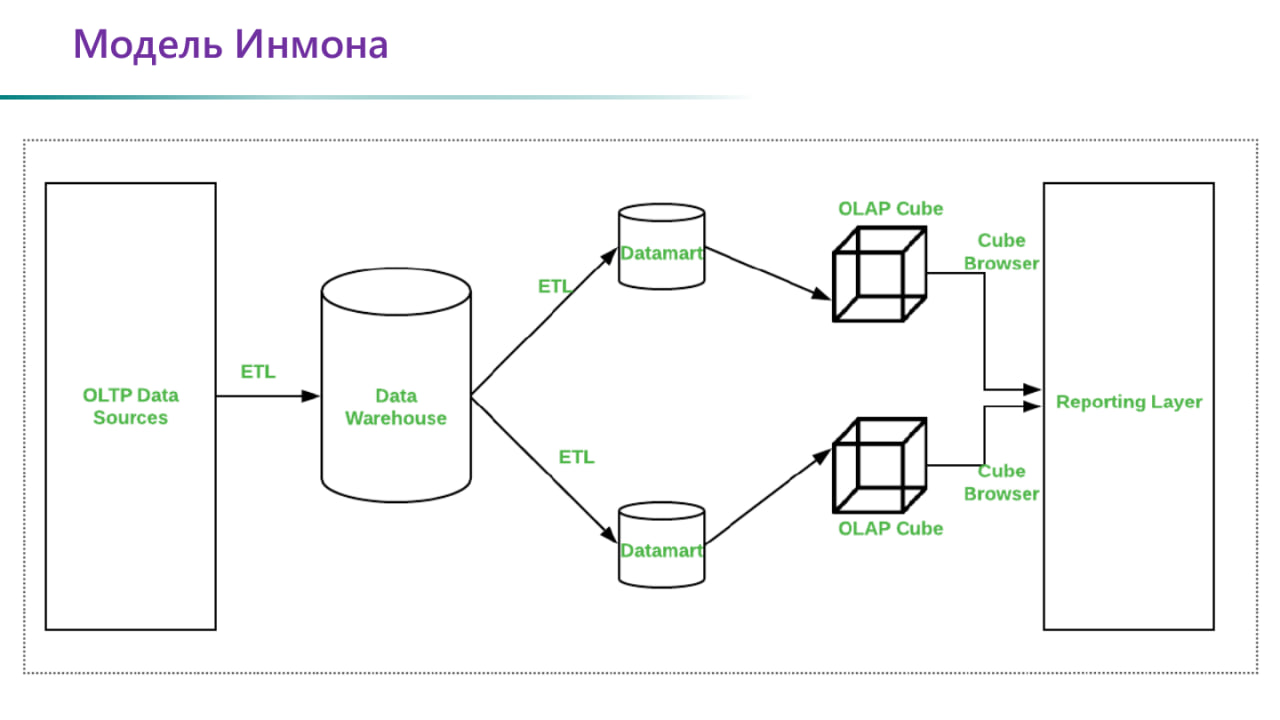
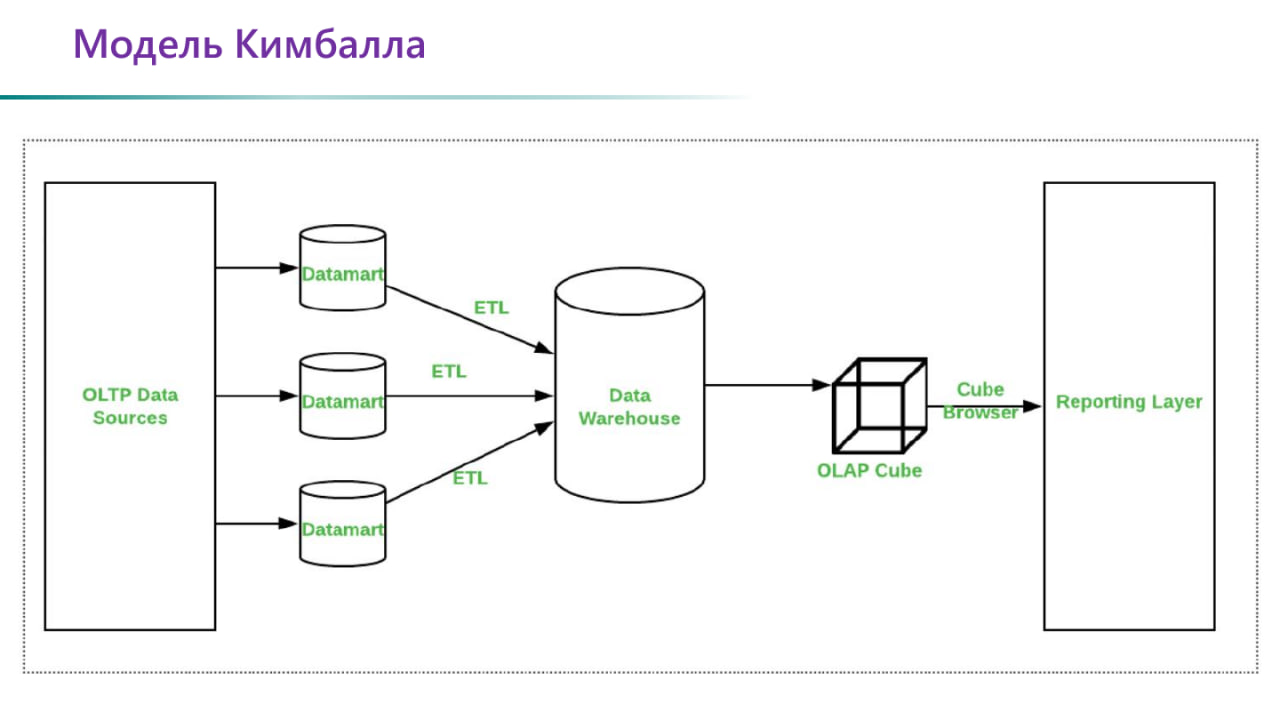
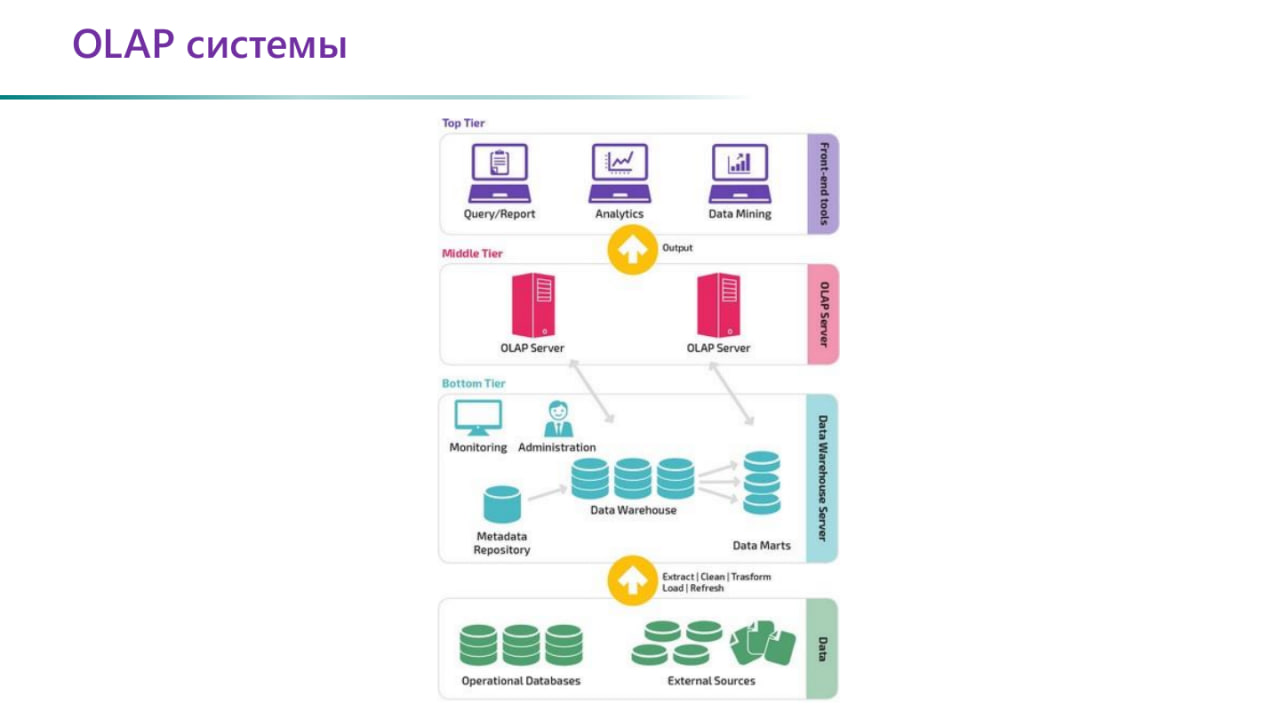
В DWH данные из всех СУБД предприятия аккумулируют и очищают, формируя их единый источник.

Enterprise Data Warehouse (EDW) — это разновидность централизованного корпоративного репозитория, хранящего все исторические бизнес-данные корпорации и управляющего ими. Информация обычно поступает от различных систем, например, из ERP, CRM, физических записей и других неструктурированных файлов.

Корпоративное хранилище данных (КХД) – это огромная система, в которой собрана информация из различных источников компании. Данные, собранные в хранилище, проходят различные модификации, чтобы вся информация была унифицированной и обрастала связями между собой. Такой подход даёт возможность быстрого доступа к оперативной и исторической информации, что в свою очередь позволяет пользователю принимать различные управленческие решения, основываясь на анализе данных.

Data Lake — это репозиторий для хранения, который может вмещать большой объем данных в собственном необработанном формате. Хранилища Data Lake оптимизированы для масштабирования до нескольких терабайт и даже петабайт данных. Данные обычно поступают из нескольких разнородных источников и могут быть структурированными, частично структурированными и неструктурированными.

Data Lake House - это архитектура данных, которая объединяет озеро данных и хранилище данных в единое целое. Хранилища данных обеспечивают машинное обучение, бизнес-аналитику и прогнозную аналитику, позволяя организациям использовать недорогое и гибкое хранилище для всех типов данных — структурированных, неструктурированных и полуструктурированных - при одновременном предоставлении структур данных и функций управления данными.



Архитектура слоев Хранилищ данных

Собрать

RAW (Цель - сохранить информацию с источника, Задачи - Собрать данные с источника as-is, Преобразовать в устойчивый к изменениям формат)

-

Стандартизировать

ODS (Цель - хранить операционные данные источника, Задачи - Сформировать набор сущностей источника и разложить данные по сущностям, Предоставить стандартный интерфейс доступа к данным вне зависимости от особенностей)

-

Сохранить

DDS(Цель - накапливать данные о сущностях доменной модели, Задачи - хранить детальную историю изменений, Консолидировать данные между источниками, Предоставлять стандартный интерфейс доступа к сущностям)

-

Предоставить

CDM(Цель - предоставлять витрины данных для анализа, Задачи - формировать данные в контексте бизнес-потребностей, оптимизировать доступ на чтение)

-

Проанализировать

REP(Цель - хранить отчетные срезы, Задачи - формировать данные в контексте бизнес-потребностей, готовить агрегированные отчеты)

Сервис репликации данных состоит из RAW. Он забирает инкрименты и снепшоты с источников различных типов и преобразовывает данные в устойчивый к изменениям формат

Data Lake состоит из RAW + ODS

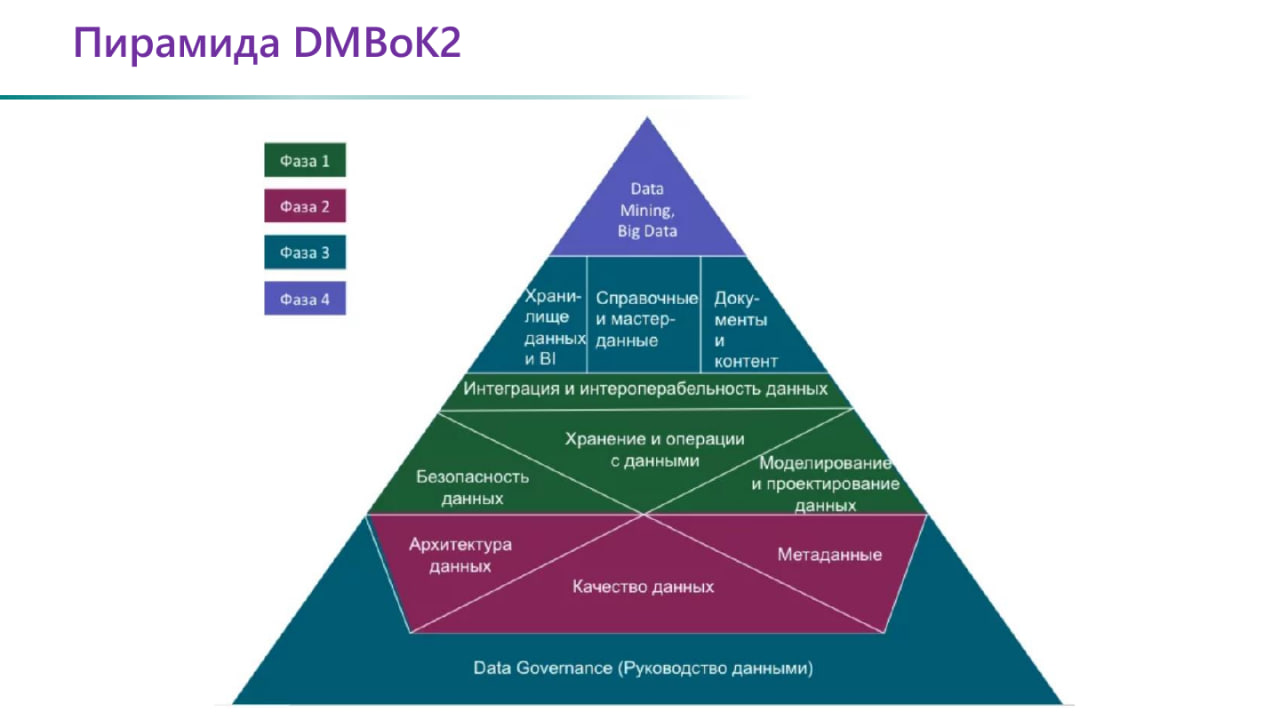
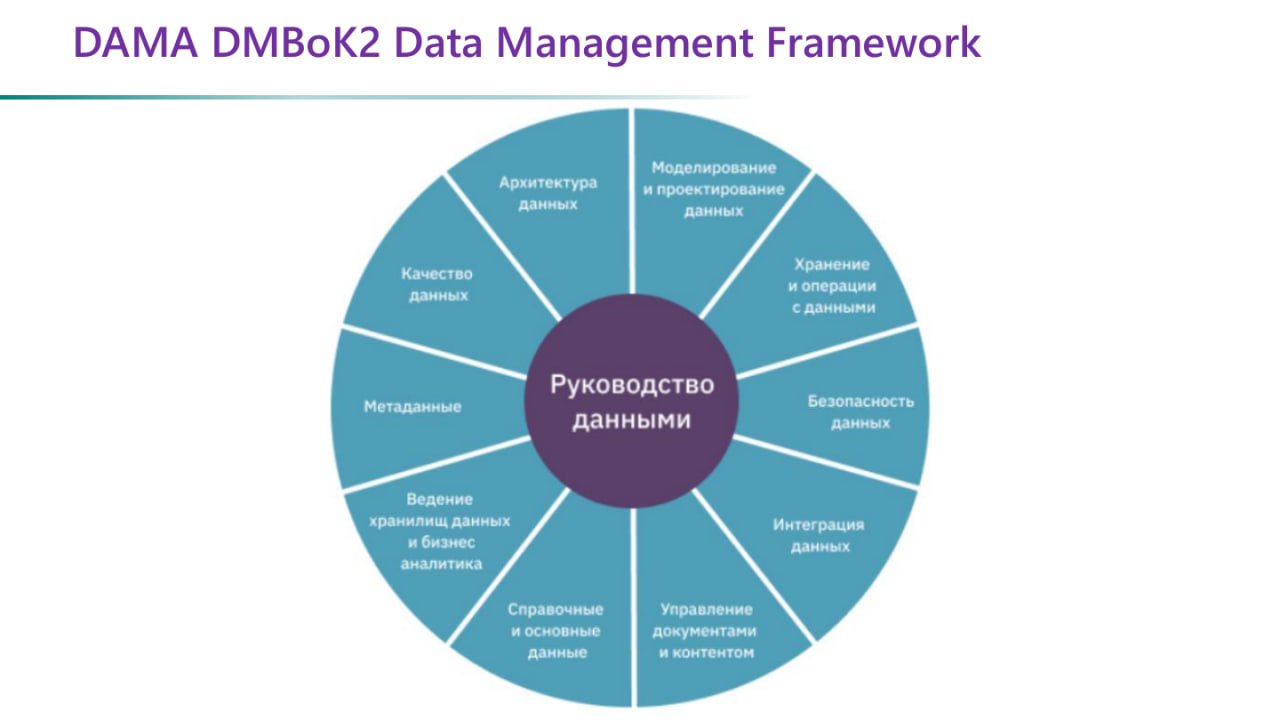
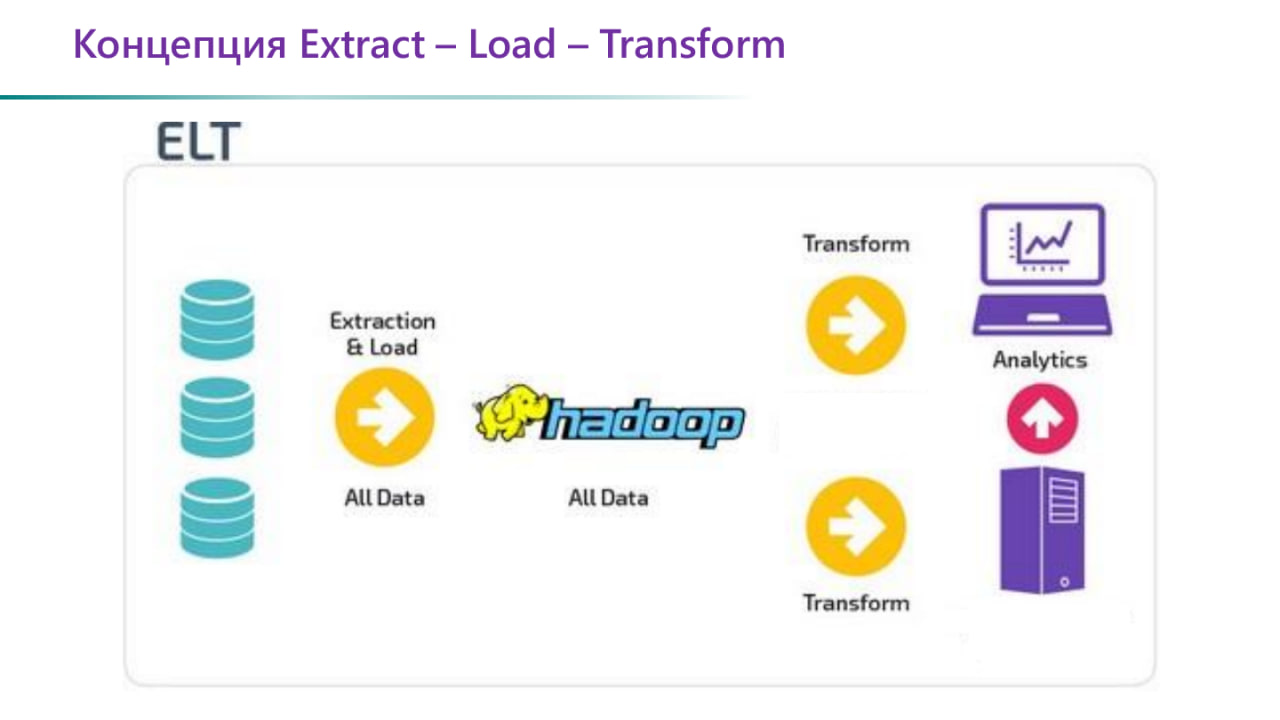
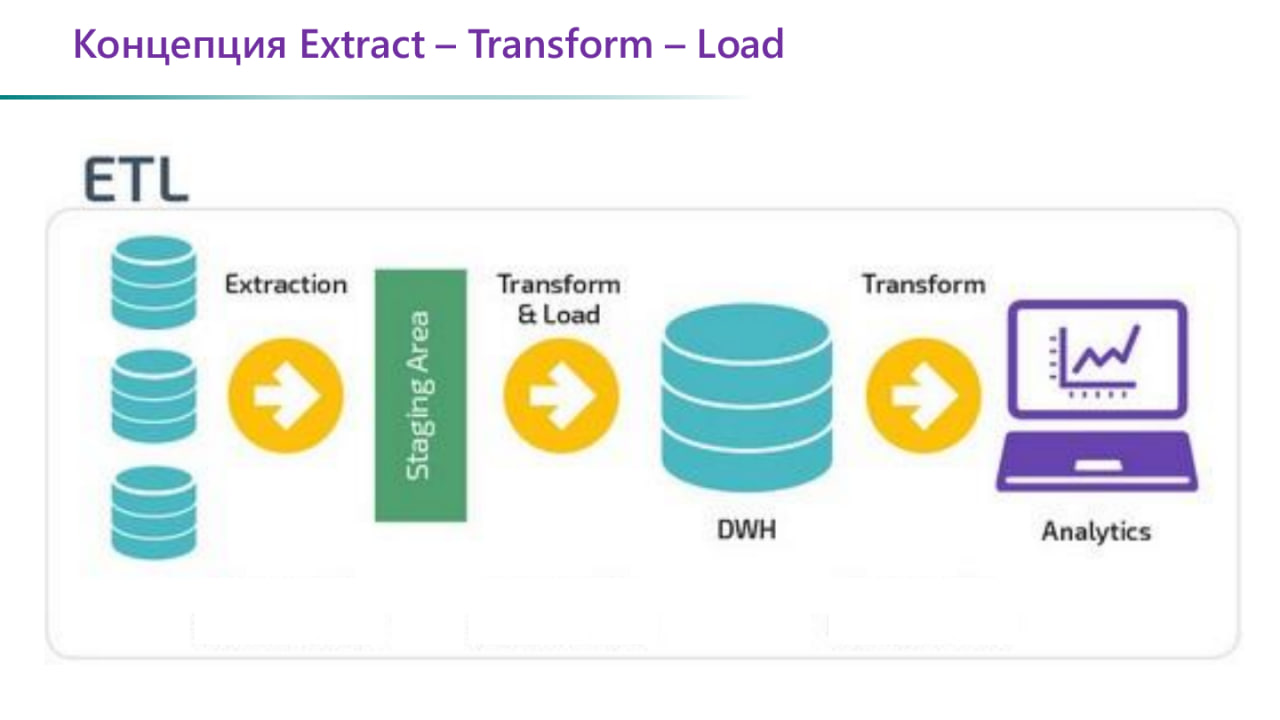
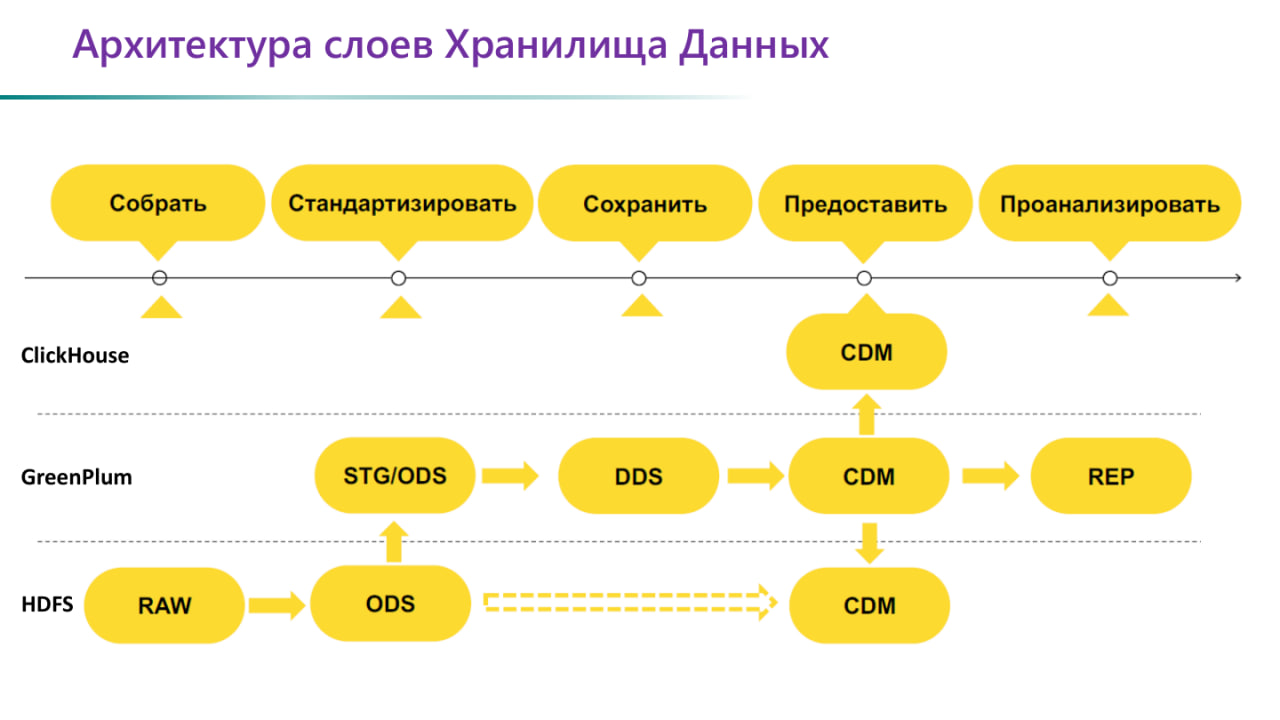
Занимается полуструктурированием данны, каркас - MapReduce, аналоги экосистемы hadoop

ODS + DDS + CDM = Data warehouse

Это различные ad-hoc запросы, много join-ов, малое время отклика

CDM + REP - системы анализа и визуализации

Кубы данных, отчеты и дашборды



Data Vault и Anchor Modeling  
Обе методологии:  
1. Повышают градус нормализации выше ЗНФ  
2. Вводят свои типы таблиц и накладывают жесткие ограничения на их использование  
3. При использовании создают оверсток таблиц (>9000)  
  
Обещают:  
1. Уменьшить постоянное дублированние данных от изменения атрибутного состава  
2. Избавить от деструктивных изменений, заниматься именно расширением модели  
3. Позволить дорабатывать хранилище легко и быстро  
  
DV вводит и регламентирует основные типы таблиц: Hub, Link, Satellite.  
  
Хаб хранит сущности  
Линк обеспечивает транзакционную интеграцию между хабами  
Саттелайт предоставляет контекст первичного ключа хаба (атрибуты, описания)  
Бридж упрощает соединение через несколько связей  
Поинт ин тайм упрощает получение информации из саттелитов одной сущности с разной частотой обновления  
  
  
HUB  
  
Хабы – отдельные таблицы, содержащие как минимум уникальный список бизнес ключей  
  
Атрибуты включают:  
Ключ бизнес-сущности из внешней системы  
Суррогатный ключ  
Временная отметка даты загрузки  
Код источника данных  
  
LINK  
  
Линки представляют отношения или транзакцию меж двух и более компонент бизнеса  
  
Атрибуты:  
Суррогатный ключ  
Ключ хабов от 1го до Nного  
Временная отметка даты загрузки  
Код источника данных  
  
SATTELITE  
Является контекстной информацией ключа хаба. Обычно с историзмом по SCD 2  
  
Атрибуты:  
Первичный ключ саттелита  
первичный ключ хаба или связи  
даты действия записи SCD2  
Временная отметка даты загрузки  
Код источника данных  
  
Anchor Modeling – технология моделирования гибкой базы данных, подходящая для информации, которая со временем изменяется как по структуре, так и по содержанию.  
  
В методике моделирования используются четыре типа таблиц: якорь, атрибут, связь и узел – каждый из которых отражает различные аспекты моделируемого домена. Полученные модели могут быть переведены в физические проекты баз данных с использованием формализованных правил. Когда такой перевод сделан, таблицы в реляционной базе данных будут в основном в шестой нормальной форме.

Anchor

Существительное, объект реального мира.

Anchor таблица должна хранить

Суррогатный ключ

Временную отметку даты загрузки

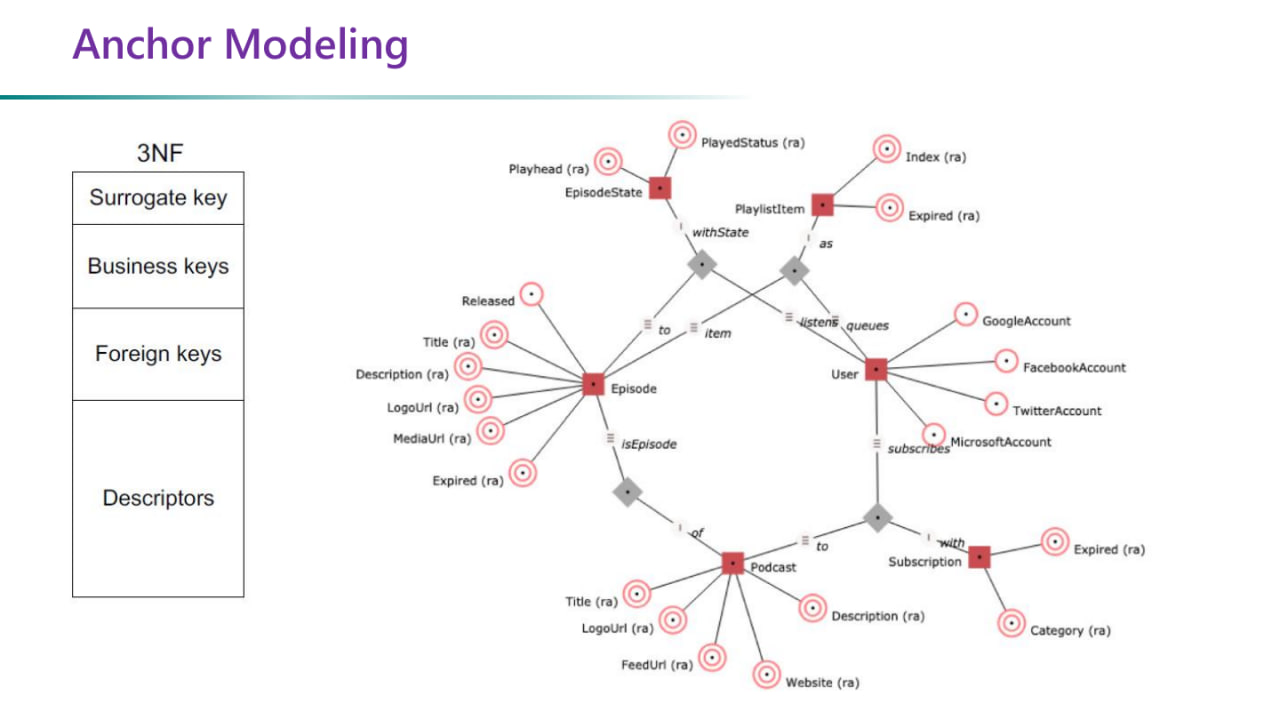
Tie  
Это таблица для хранения связей между объектами

У связи нет атрибутов

Attribute

Это таблица для хранения свойства, атрибута объекта, при этом на каждое свойство ровно одна таблица

Каждая Attribute таблица содержит:  
  
Суррогатный ключ

Временную отметку даты загрузки  
Непосредственное значение

DV vs AM

DV

1. На каждую сущность создается хаб таблица с бизнес ключом и суррогатным ключом
2. Атрибуты группируются в таблицы-сателлиты по принципам совместности изменения и\или источника\использования
3. Связи только через отдельные таблицы, на них можно повесить саттелайт
4. Есть специальные ПиТ и Бридж таблицы

АМ

1. На каждую таблицу создается анкор-таблица с суррогатным ключом
2. Один атрибут – одна таблица
3. Связи только через отдельные таблицы, нет атрибутов
4. Есть специальная таблица кнот – стат справочник



Доп.  
Шардирование — это принцип проектирования базы данных, при котором части одной таблицы размещаются на разных шардах. Шард — узел кластера, который может состоять из одной или нескольких реплик. Реплики — это серверы, на которых дублируются данные в рамках шарда. Запрос на чтение или запись в шард может быть отправлен на любую его реплику, выделенного мастера нет.

Batching – анализ полной совокупности собранных данных  
SQL – запросы к базе

MapReduce – запрос к данным хадуп

+ Можно сделать сложную логику

+ Легко пересчитывать

-Существенные временные задержки

-Объем анализируемых данных ограничен местом на дисках

Streaming – живая агрегация на потоке

RealTime – счетчики на потоке (NRT)

+ агрегация на лету и очень быстро

+ объем обрабатываемых данных не ограничен дисками

- тяжело реализовать сложную логику

- очень тяжело исправить ошибку задним числом



Интернет вещей (Internet of Things)

Термин впервые использовался в 1999 году британской новатор в области технологий по имени Кевин Эштон для описания системы в которой предметы физического мира могут подключаться к интернету с помощью датчиков  
  
Эштон создал этот термин для того, чтобы проиллюстрировать потенциальные возможности подключения меток радиочастотной индетификации, используемых в корпоративных цепочках поставок, для подсчета и отслеживания товаров без необходимости вмешательства со стороны человека.  
Сегодня термин интернет вещей широко используется для описания сценариев в которых подключение к интернету и вычислительные функции распространяются на ряд объектов, устройств, датчиков и других предметов повседневной жизни.

Условия для использования ИОТ  
  
1. Рабочие объекты – специализированная производственная среда

2. Транспортные средства – системы внутри движущихся транспортных средств

3. Города – городская среда

4. Открытые пространства – за пределами городской среды и других условий

Create Read Update Delete CRUD

В мире реляционных баз данных общеизвестны ACID-транзакции, некоторое время являвшиеся эталоном. Гарантии ACID обеспечивают безопасную среду для обработки данных:

•атомарность (Atomic) - все операции в транзакции либо успешны, либо для всех них выполняется откат;

•согласованность (Consistent) - по окончании транзакции база данных является структурно согласованной;

•изолированность (Isolated) - транзакции не мешают друг другу. Спорные ситуации разрешаются базой данных так, что транзакции выполняются последовательно;

•долговечность (Durable) - результаты применения транзакции не должны теряться, даже при сбоях.

Эти свойства означают, что сразу по завершении транзакции ее данные согласуются (так называемая согласованность записи) и записываются на диск (или на диски, или в разные области памяти). Это отличная абстракция для разработчиков приложений, но требует сложных блокировок, которые могут вызвать логическую недоступность, и, как правило, считается излишне тяжеловесным шаблоном в подавляющем большинстве прикладных областей.

NoSQL = Not Only SQL Термин «NoSQL» впервые был использован в 1998 году для описания реляционной базы данных, не использовавшей SQL

Основные особенности:

• Отказ от реляционной модели и языка SQL

• Использование распределенной архитектуры

• Отсутствие полноценной поддержки ACID-транзакций

Преимущества:

• Большая производительность

• Хорошая масштабируемость при возрастающих нагрузках и огромных объемах данных

Для многих прикладных областей ACID-транзакции являются излишне пессимистичными. В мире NoSQL ACID-транзакции вышли из моды, поскольку хранилища смягчили требования к немедленной согласованности, актуальности данных и их точности, чтобы получить другие преимущества, такие как масштабируемость и гибкость. Вместо ACID возник другой популярный подход BASE, описывающий принципы более оптимистичной стратегии хранения:

•обычно доступно (Basic Availability) - хранилище доступно большую часть времени;

•гибкое состояние (Soft-state) - хранилища не обязаны соблюдать очередность записей, и разные реплики не должны постоянно согласовываться;

•отложенная согласованность (Eventual Consistency) - хранилища достигают согласованности с некоторой задержкой по времени (например, позже, во время чтения).

Принципы BASE значительно слабее гарантий ACID, и между ними нет прямого соответствия. Значения в BASE-хранилище доступны (потому что это является основой масштабирования), но это не предлагает гарантированной согласованности реплик при записи. BASE-хранилища обеспечивают менее строгий контроль: данные будут согласованы позднее, вероятнее всего, во время чтения, или всегда будут согласованы, но только для определенных фиксаций, обработанных последними.

Schema-less данные – в NoSQL БД структура данных заранее не регламентируется. Вместо строго структурированных таблиц имеем, например, набор разнородных документов.

Плюсы: структура каждого документа может быть уникальной (он сам себя описывает).

Минусы: поддержка структуры данных переносится с сервера в логику приложений.

Причины появления – распределенность систем:

• Потребность в горизонтальной масштабируемости БД (путем добавления новых узлов).

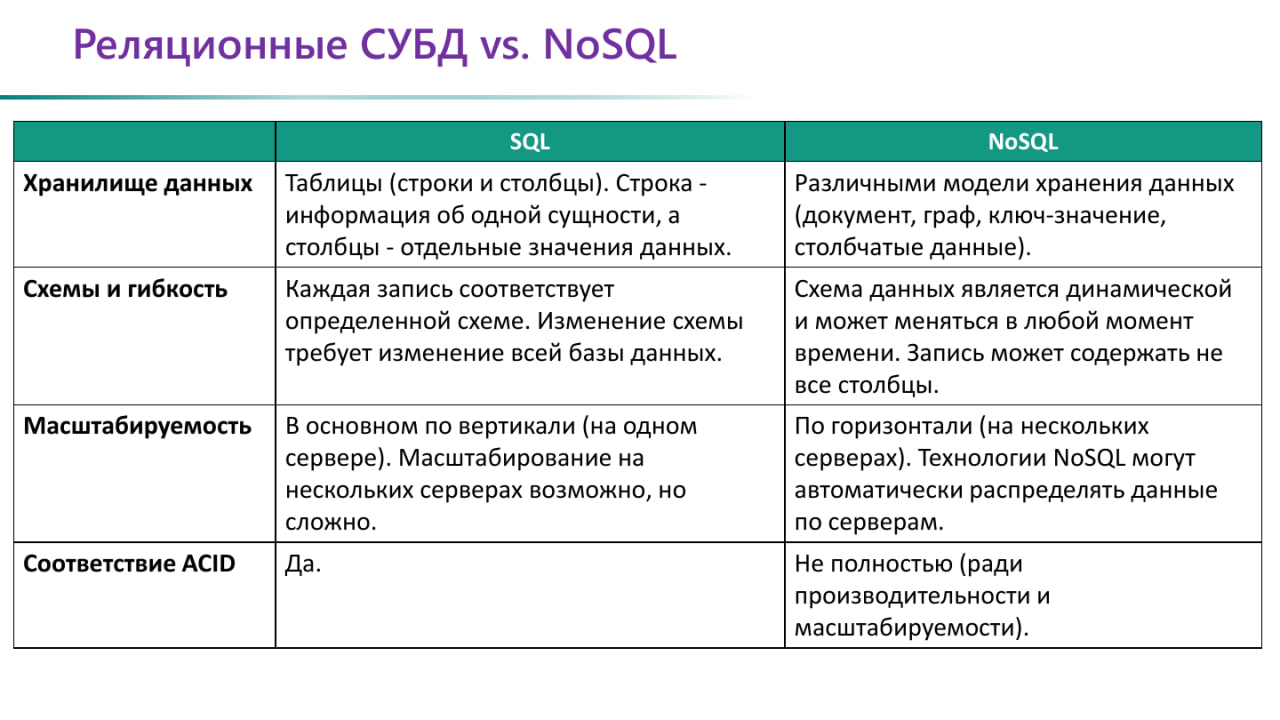
• Сложность эффективной реализации транзакций в распределенной среде.

MapReduce

Механизм параллельной обработки больших данных на кластерах:

Map: Главный узел (master node) делит задачу на части, распределяя их по остальным машинам (worker nodes).

Reduce: Master node получает предварительные результаты, и формирует из них конечный результат.



Системы «ключ-значение»

• CRUD-операции (Create-Read-Update-Delete)

• Хорошая горизонтальная масштабируемость

• Простота и производительность

• Плохо подходят для операций, отличных от CRUD

Документ-ориентированные системы

• Единица хранения – документ, т.е. объект, обладающий произвольным набором атрибутов (полей), который может быть представлен, например, в JSON

• Поддерживают поиск по полям документов, индексы, часто допускаются вложенные документы и массивы.

Системы типа Google BigTable

• Данные хранятся в виде строк. Для доступа к данным три ключа: ключ строки (по нему отсортированы строки в базе), ключ столбца, временная метка.

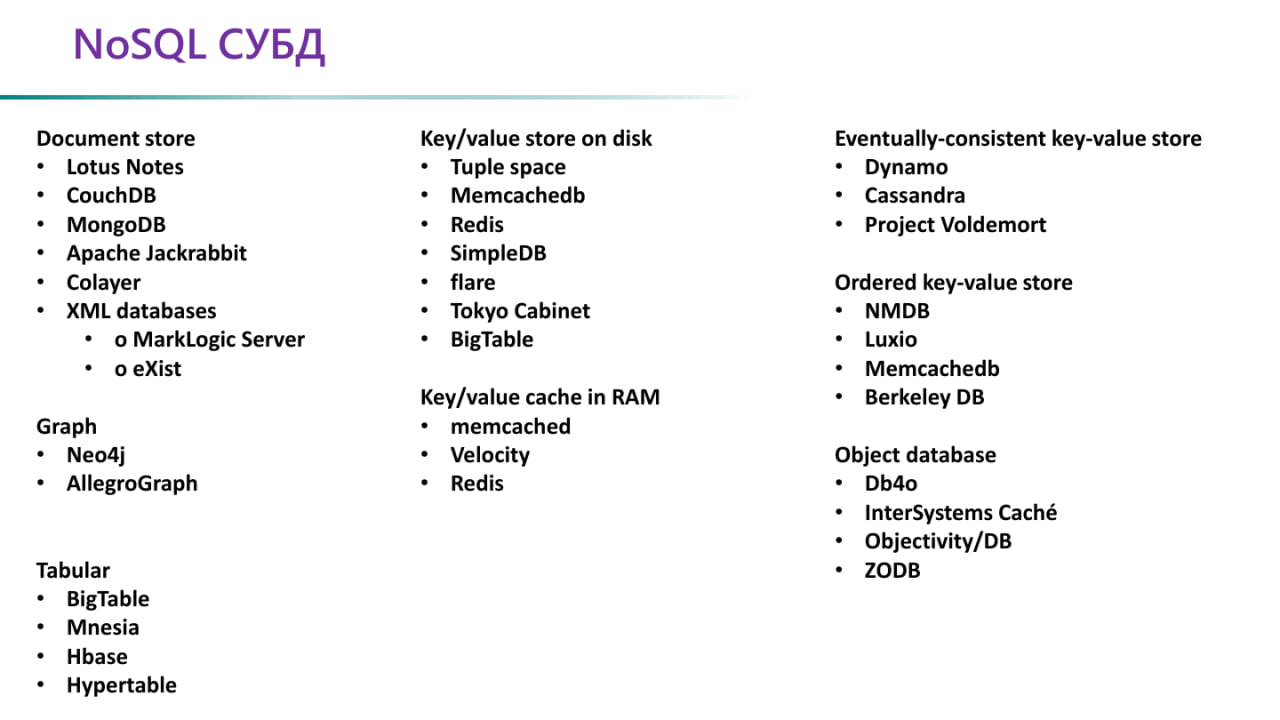
• Связанная информация хранится в одном месте.

Графовые системы:

• Состоит из узлов и связей между ними.

• Как с узлами, так и со связями можно ассоциировать свойства (пары ключ- значение), в которых хранятся данные.

• Быстрый просмотр узлов и связей для поиска нужных данных.



Hadoop - это программная среда с открытым исходным кодом, используемая для распределенного хранения и обработки наборов данных больших данных с использованием модели программирования MapReduce.

Он состоит из компьютерных кластеров.

Все модули в Hadoop разработаны с фундаментальным предположением, что аппаратные сбои являются обычным явлением и должны автоматически обрабатываться платформой.

Для чего нужна хадуп?  
Гибкость для хранения и обработки данных любого типа, будь то структурированные, полуструктурированные или неструктурированные. Он не ограничен одной схемой.

Преимущество при обработке данных сложного характера. Его масштабируемая архитектура распределяет рабочие нагрузки по многим узлам.

Еще одним дополнительным преимуществом является то, что его гибкая файловая система устраняет узкие места ETL.

Экономно масштабируется, может использоваться на обычном оборудовании. Помимо этого его природа с открытым исходным кодом защищает от vendor lock.

Концепции хадуп:

1. Масштабируемость для больших объемов данных.
2. Экономическая эффективность  
   Использование узлов потребительского класса  
   Стандартное оборудования для сетей  
   Автоматическое восстановление после сбоя  
   Легко использовать

Стандартное дешевое оборудование будет сбоить => Ставим защиту от сбоев

Стандартное межсоединение = низкая пропускная способность => Переносим вычисления в данные

Программирование распределенных систем – тяжело => Даем среду только с мап и редюс.

Автоматическое распараллеливание

Отказоустойчивость

Инструменты мониторинга состояния

Понятная абстракция для программистов

Программы MapReduce пишут на JAVA или на любом другом сценарном языке

Hadoop написан на Java

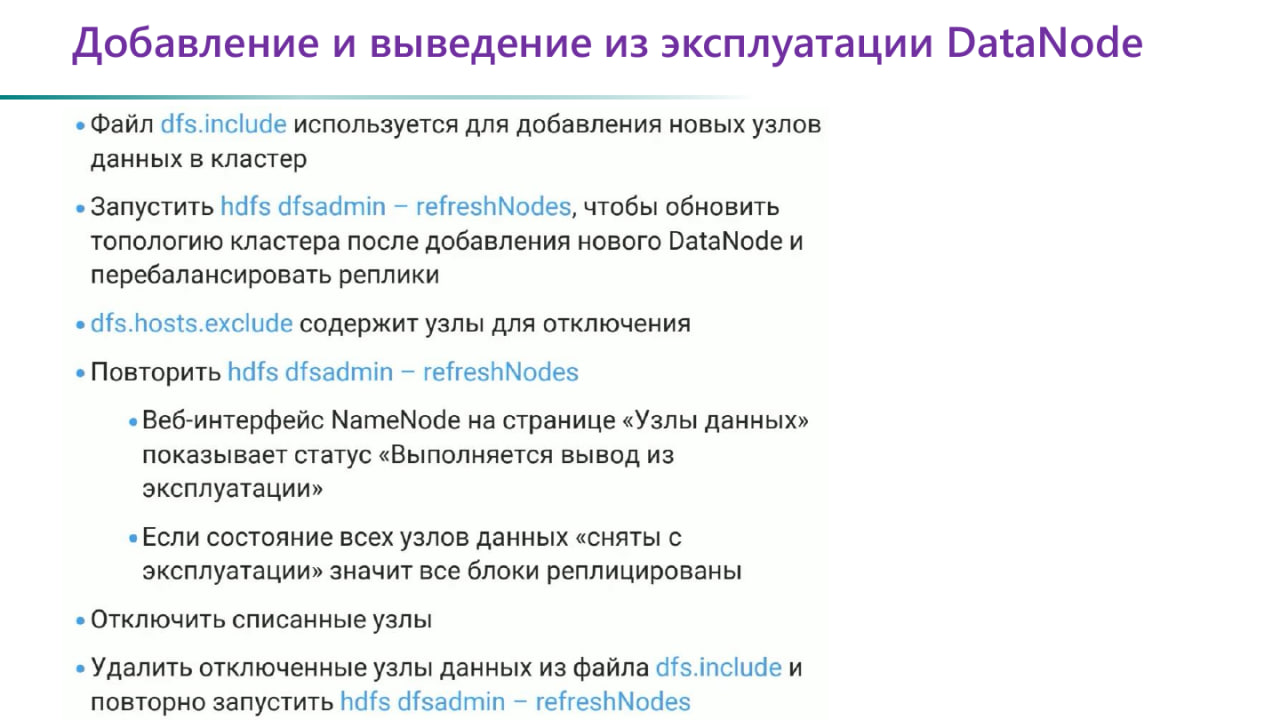
Архитектура хадуп:  
Hadoop работает по системе master-slave. Есть главный узел и есть подчиненные узлы.

Главный узел управляет, поддерживает и контролирует подчиненных, тогда как подчиненные выполняют вычислительные задачи.

В архитектуре Hadoop Master должен развертываться на хорошем серверном оборудовании, а не на обычном оборудовании.

NameNode: сердце файловой системы HDFS, он поддерживает и управляет метаданными файловой системы. Например; какие блоки составляют файл и на каких DataNode эти блоки хранятся.

DataNode: здесь HDFS хранит фактические данные, обычно этих узлов довольно много.

Yes Another Resource Negotiator

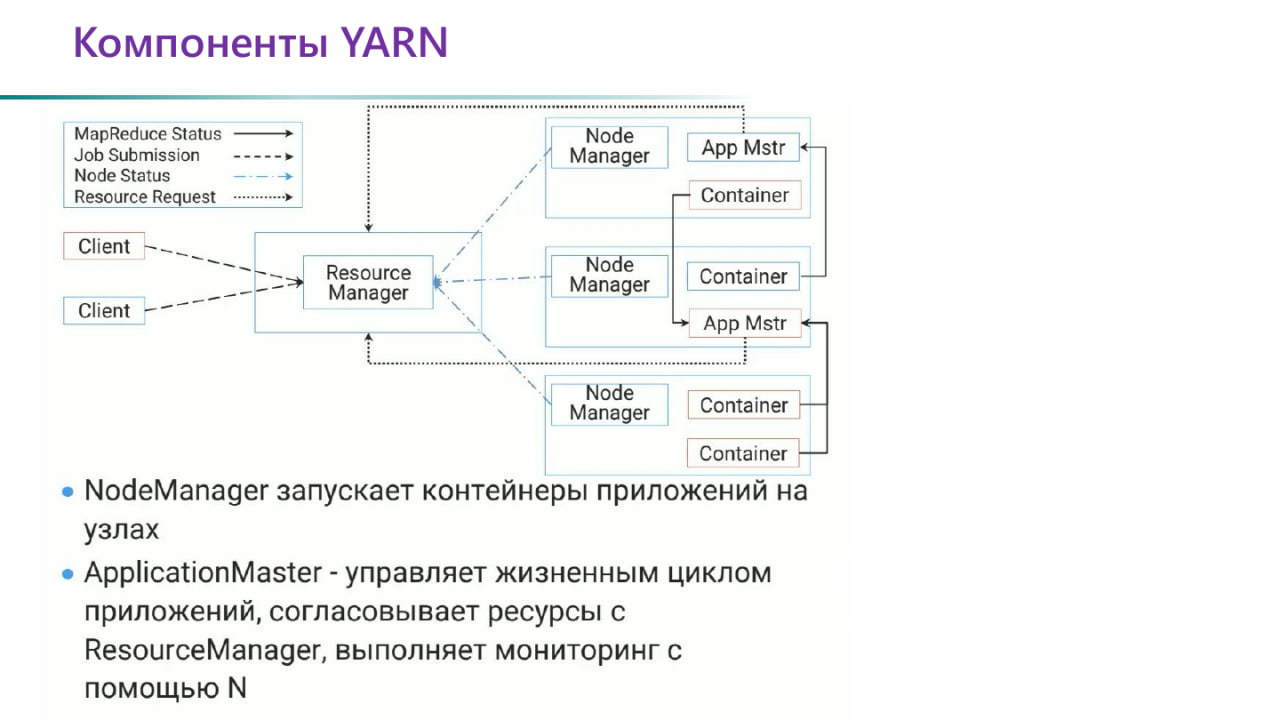
Технология управления ресурсами с открытым исходным кодом, которая развернута в кластере Hadoop.

YARN стремится эффективно распределять ресурсы между различными приложениями.

Он запускает два процесса, которые выполняют две разные задачи: отслеживание заданий и мониторинг прогресса.

Эти два демона называются resource manager и application master / node manager соответственно.

Диспетчер ресурсов распределяет ресурсы между различными приложениями, а мастер приложений контролирует выполнение процессов.



По умолчанию Hadoop использует планировщик FIFO, но в промышленных задачах чаще используют fair (Cloudera CDH) или capacity (Hortonworks HDP) планировщики.

Для чего нужна YARN:

Гибкость - включает другие специализированные модели обработки данных, помимо MapReduce (пакетные), такие как интерактивная и потоковая. Благодаря этой функции YARN, другие приложения также могут запускаться вместе с программами Map Reduce в Hadoop.

Эффективность. Поскольку многие приложения работают в одном кластере, эффективность Hadoop увеличивается без особого влияния на качество обслуживания.

Shared - обеспечивает стабильную, надежную, безопасную основу и разделяемые операционные сервисы для нескольких рабочих нагрузок. Дополнительные модели программирования, такие как обработка графов и итеративное моделирование, теперь возможны для обработки данных с помощью YARN.

Hadoop MapReduce

Простота - задания MapReduce легко выполнять. Приложения могут быть написаны на любом языке, например, Java, C ++ и Python.

Масштабируемость - MapReduce может обрабатывать петабайты данных.

Скорость - с помощью параллельной обработки, решение задач, на которые обычно уходят дни, выполняется за часы или минуты с помощью MapReduce.

Отказоустойчивость - MapReduce устойчива к сбоям. Если одна копия данных недоступна, на другой машине будет копия данных, которую можно использовать для решения подзадачи.

Задача Map

Обрабатывать блоки входного файла

Произвести пары ключ-значение

Задача Reduce

Взять пары ключ-значение, сортировка по ключам

Объединить

Вывести окончательные результаты

Детали MapReduce:

Все запуски функции map работают независимо и могут работать параллельно, в том числе на разных машинах кластера.

Все запуски функции reduce работают независимо и могут работать параллельно, в том числе на разных машинах кластера.

Shuffle внутри себя представляет параллельную сортировку, поэтому также может работать на разных машинах кластера. Пункты 1-3 позволяют выполнить принцип горизонтальной масштабируемости.

Функция map, как правило, применяется на той же машине, на которой хранятся данные – это позволяет снизить передачу данных по сети (принцип локальности данных).

Форматы файлов ХДФС:  
Parquet,

JSON,

CSV

… поддерживаются "из коробки", а также есть множество плагинов для различных других форматов данных.

HDFS - это распределенная файловая система, отказоустойчивая, масштабируемая и чрезвычайно легко расширяемая.

HDFS является основным распределенным хранилищем для приложений Hadoop.

HDFS предоставляет интерфейсы для приложений, чтобы перемещаться ближе к данным.

HDFS разработана, чтобы «просто работать», однако практические знания помогают в диагностике и улучшениях.

Основная система хранения Hadoop.

Файловая система на основе Java.

Отказоустойчивость - данные дублируются на нескольких узлах данных для защиты от сбоев компьютера. По умолчанию коэффициент репликации равен 3 (каждый блок хранится на трех машинах);

Масштабируемость - передача данных происходит непосредственно с узлами данных, поэтому ваша емкость чтения / записи достаточно хорошо масштабируется с количеством узлов данных;

Объём - нужно больше места на диске? Просто добавьте больше узлов данных и перебалансируйте;

Промышленный стандарт - Другие распределенные приложения построены на основе HDFS (HBase, Map-Reduce)

HDFS предназначена для обработки больших наборов данных с семантикой однократная запись-чтение, она не предназначена для доступа с низкой задержкой.

Каждый файл, записанный в HDFS, разбивается на блоки данных

Каждый блок хранится на одном или нескольких узлах

Каждая копия блока называется репликой

Политика размещения блоков

Первая реплика размещается на локальном узле

Вторая реплика находится в другой стойке

Третья реплика находится в той же стойке, что и вторая реплика

Вывод:  
Экосистема Hadoop огромна, и содержит элементы для реализации практически любых задач, связанных с данными;

Hadoop ориентирован на доставку запросов к данным, а не наоборот;

У Hadoop высокое время отклика, поэтому он не подходит для прямого взаимодействия с пользователями.

Быстрая, выразительная кластерная вычислительная система, совместимая с Apache Hadoop

Работает с любой системой хранения, поддерживаемой Hadoop (HDFS, S3, Avro,…)

Повышает эффективность благодаря:

Примитивам вычислений в памяти

Графикам вычислений

Улучшает удобство использования благодаря:

Богатым API в Java, Scala, Python

Интерактивной оболочке

Работайте с распределенными коллекциями, как с локальными

Концепция: устойчивые распределенные наборы данных (RDD)

Неизменные коллекции объектов, распределенных по кластеру

Построены через параллельные преобразования (map, filter, …)

Автоматически перестраивается при сбое

Управляемое сохранение (например, кеширование в оперативной памяти)

Transformations (e.g. map, filter, groupBy, join)

Операции, которые строят RDD из других RDD

Actions (e.g. count, collect, save)

Возвращают результат или записывают его в хранилище

Автономные программы могут быть написаны на любом из трех языков (Java, Scala, Python), но консоль - это только Python или Scala

Python подходит для обоих вариантов

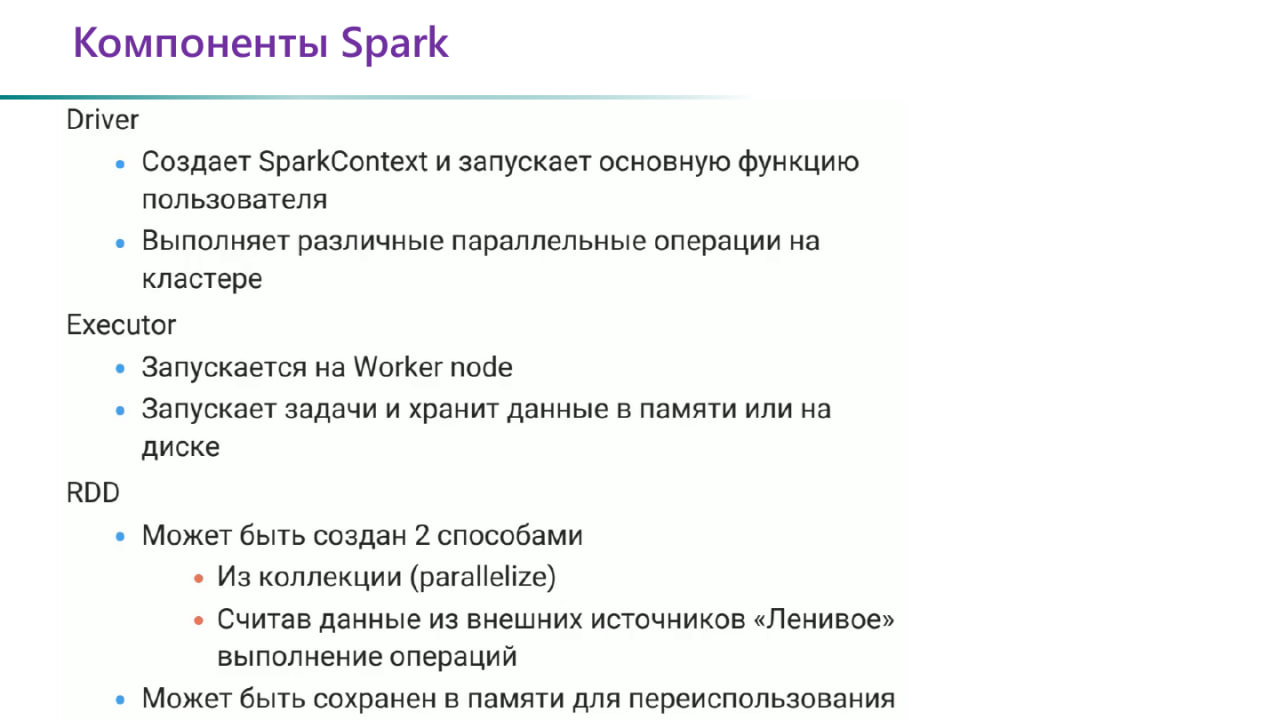
Производительность: Java / Scala будет быстрее (статически типизированной), но Python лучше для вычислений с NumPy

Главная точка входа в функциональность Spark

Создан в Spark Shell как переменная sc

В автономных программах вы создаете свои собственные контексты





Трансформация Мар с функцией UCASE

Возвращает новый распределенный набор данных, сформированный в результате применения функции трансформации к каждому элементу входных данных.

Трансформация Filter  
Возвращает новый набор данных, сформированный из тех элементов источника, для которых критерий отбор возвращает значение true.

Трансформация Union

Возвращает новый набор данных, содержащий объединение элементов в исходном наборе данных и аргументе.

Трансформация Distinct

Возвращает новый набор данных, который содержит только уникальные элементы.

Трансформация Join

Операция объединения совмещает элементы на основе ключа, значение – кортеж с двумя полями исходного значения.

Трансформация Reparation

Произвольно переставляет данные в RDD, чтобы изменить количество разделов в большую или меньшую сторону.

Job

Создается в RDD в результате наступления Action. Это рабочий элемент верхнего уровня, разбитый на стадии и переданный в DAGSheduler.

Стадия (Стейдж)

Набор параллельных задач, которые вычисляют частичные результаты функции

По одной стадии на партицию РДД  
Конвейеризируется в запускается одним независимым исполнителем.

Задача (Task)

Вычисление над партицией данных для job stage

Состояния задачи:

1. Лаунчинг
2. Раннинг
3. Фишид
4. Фейлед
5. Киллед
6. Лост

Предоставляет способ использовать непрерывные потоки данных  
Масштабируемый, высокопроизводительный, отказоустойчивый  
Построен на основе Spark RDD

API похож на Core API

Поддерживает много входов

Интегрируется с другими модулями Spark

Микропакетная архитектура

МА Спарк Стриминг

Постоянно получает входные потоки данных  
Делит данные на небольшие, детерменированные пакетные задания

Пакеты обрабатываются Спарк Энджайн

DStream = Discretized Stream

Dstream – это серия пакетов

Каждый пакет представлен как РДД  
Каждый РДД содержит данные за определенный интервал

Операции DStream приводят к преобразованию RDD

Возможность прямого доступа к API RDD

Объединение потоковой и пакетной обработки

Основные метрики

Интервал между пакетами  
 Новые пакеты создаются через равные промежутки времени

В конце каждого интервала создается новый RDD  
 Обычно от 500 мс до нескольких секунд

Интервал между блоками

Входные данные разбиваются на блоки

Один блок создается за интервал между блоками

По умолчанию 200 мс, конфигурируемый

Hive

Входит в экосистему Hadoop

HiveQL (SQL-подобный язык запросов)

Разработан Фейсбук в 2007 году

Предназначен для противников МапРедюс

Инфраструктура хранилища данных, построенная на основе Хадуп для обобщения данных, построения запросов и анализа

ЕТЛ

Структурирование данных  
Доступ к различным хранилищам  
Выполнением запросов через приложения Ярн

Преимущества технологии:  
 Привычный СКЛ  
 Расширяемость – типы, функции, форматы, скрипты  
 Масштабируемость

Driver – управляет жизненным циклом выражения HiveQL. Содержит дескриптор и статистику сеанса

Query Compiler – компилирует HiveQL в ориентированный ациклический граф map/reduce-задач

Execution Engine – исполняет Execution Plan, созданный комплиятором. Execution Plan представляет собой DAG.

Внутренняя таблица – основной тип таблиц в Hive

Создает каталоги данных в HDFS

Можно создавать таблицы и разделы

Удаление внутренней таблицы, удаляет также данные и метаданные

Это тип таблицы по умолчанию

Внешняя таблица – пользователь предоставляет LOCATION файла, из которого создается таблица

Данные должны быть в формате совместимом с Hive

При удалении таблицы EXTERNAL данные в таблице НЕ удаляются из файловой системы

Impala — это популярный движок MPP с открытым исходным кодом и широким спектром возможностей в Cloudera Distribution Hadoop (CDH ) и CDP. Impala заслужила доверие рынка благодаря low-latency highly interactive SQL-запросам. Impala не только поддерживает HDFS с Parquet, ORC, JSON, Avro, и текстовые форматы, но также имеет встроенную поддержку Kudu, Microsoft Azure Data Lake Storage (ADLS) и Amazon Simple Storage Service (S3). Impala обладает высоким уровнем безопасности и может поддерживать тысячи пользователей с кластерами из сотен узлов на петабайтных датасетах.

Отличительными особенностями Apache Drill считаются следующие:  динамическое вычисление схемы данных на лету, в процессе исполнения SQL-запроса. Не требуется определять схему данных или спецификацию формата хранения заранее. Поскольку некоторые форматы (Parquet, JSON, AVRO) и NoSQL базы данных уже содержат описание формата внутри себя, Drill вычисляет это динамически, начиная обработку данных в процессе записи и определяя схему во время обработки SQL-запроса. При этом учтена возможность изменения схемы данных – многие команды могут быть переконфигурированы.

отсутствие централизованной системы хранения метаданных, например, Hive Metastore, в отличие от Cloudera Impala и Apache Hive. Для Drill метаданные доставляется от плагинов, передающих запрос хранилищу. При этом возможно делать SQL-запросы к нескольким разным узлам, собирая таким образом информацию из разных источников (баз данных или файлов).

поддержка множества нереляционных баз данных и файловых систем – не только Apache Hadoop (Hbase, HDFS), но и MongoDB, Amazon S3, Azure Blob Storage, Google Cloud Storage, а также локальные файлы. Один запрос может включать данные из нескольких хранилищ. В частности, можно объединить информацию о пользователях в MongoDB с каталогом журнала событий из Hadoop.

С архитектурной точки зрения Drill предоставляет собой гибкую иерархическую модель столбчатых структур. Реляционные данные в Drill рассматриваются как особый (упрощенный) случай сложных данных. Таким образом, Дрилл позволяет получить доступ к вложенным атрибутам данных, как к столбцам обычной реляционной таблицы, предоставляя интуитивно понятные расширения, что значительно упрощает работу с информационными массивами.

Phoenix (Феникс) – это проект верхнего уровня фонда Apache Software Foundation (c 2014 года), механизм параллельной реляционной базы данных с открытым исходным кодом, который поддерживает обработку транзакций в реальном времени (OLTP, Online Transaction Processing) в Hadoop с использованием NoSQL-СУБД HBase в качестве резервного хранилища. В отличие от Apache Hive, Феникс компилирует SQL-запросы в собственные API-интерфейсы NoSQL, не используя MapReduce, что позволяет создавать быстрые Big Data приложения с низкой временной задержкой (low latency), работающие с нереляционными хранилищами данных.

Apache Presto – это механизм выполнения распределенных параллельных запросов, оптимизированный для малой задержки и интерактивного анализа запросов. Presto легко выполняет запросы и масштабируется без простоев даже от гигабайтов до петабайтов.

Presto содержит следующие функции:

Простая и расширяемая архитектура.

Простая абстракция и синтаксис.

Connectors – Presto поддерживает коннекторы для предоставления метаданных и данных для запросов.

Координатор – позволяет избежать ненужных задержек ввода / вывода.

Пользовательские функции. Аналитики могут создавать пользовательские функции для удобной миграции.

Колоночная обработка данных.

Масштабируемость.

Это распределенная колоночная timeseries OLAP-система. В ней нет привычных для мира SQL понятий таблица (вместо неё datasource) и строка (вместо неё event). Druid основан на нескольких допущениях (ограничениях) о данных:

в каждой строке данных есть timestamp, который монотонно растёт (в пределах окна в 10 минут по умолчанию).

данные не меняются, Insert only (операции Update нет).

Это позволяет нарезать данные на так называемые сегменты по времени. Сегмент — это минимальная неделимая и неизменная «партиция» одной таблицы за определенный промежуток времени. Все операции с данными, все запросы выполняются посегментно.

Каждый сегмент самодостаточен: помимо основной таблицы, записанной в колоночном виде, в нем также содержатся справочники и индексы, необходимые для выполнения запросов. Таким образом, сегмент — это небольшая колоночная read-only БД.

В свою очередь, отсюда возникает «распределенность»: возможность поделить большой объем данных на небольшие сегменты, чтобы выполнять вычисления параллельно (как на одной машине, так и на многих сразу).

Apache Druid

Высокая производительность.

Если нужно провести Update, то перегружаем всю таблицу, если нужно часто – то нужен ли Druid?

Отказоустойчивость, способность пережить потерю дата-центра.

Легкая масштабируемость.

Удобство модификации, так как написана на Java.

SQL – ON – HADOOP решения:  
  
EDW поддерживающая BI дэшборды - Hive LLAP.

Ad-hoc, self-service и исследовательское хранилище данных - Impala.

Data Engineering - Spark SQL. Высокий параллелизм в HDFS - Hive on Tez.

OLAP с timeseries – Druid.

OLTP с низким временем задержки и высоким параллелизмом - Phoenix.

Ещё раз про отличие Phoenix, Cloudera Impala, Apache Hive и Drill: при том, что все эти продукты можно условно отнести к стеку SQL-on-Hadoop, Apache Phoenix предназначен специально для HBase, тогда как Hive и Impala могут работать в том числе и другими распределенными файловыми системами, помимо HDFS, например, Amazon S3, а Drill вообще позиционируется как средство для работы с любыми файловыми хранилищами и базами данных. Кроме того, если Hive, Impala и Drill могут рассматриваться еще и в качестве ETL-инструментов, то Phoenix больше предназначен для построения корпоративных хранилищ данных и реализации BI-задач.

Hadoop ecosystem

Apache Kafka - быстрая масштабируемая долговечная и отказоустойчивая система обмена сообщениями типа издатель-подписчик

Кафка часто используется вместо традиционных брокеров сообщений, таких как ДЖИМИЭС

Кафка работает в сочетании с системами обработки, как спарк

Кафка обеспечивает передачу потоков сообщений для анализа с малой задержкой

Характеристики

Масштабируемость без простоев

Долговечность (охрана сообщений на диске и внутрикластерная репликация)

Надежность – репликация данных, поддержка нескольких потребителей и автобаланс потребителей в случае сбоя

Производительность – выоская пропускная способность как для публикации, так и для подписки. Дисковые структуры обеспечивают постоянную прозводительность даже при наличии большого количества терабайтов хранимых сообщений

Архитектура

Топик – раздел потоков сообщений на категории

Продюсер – процессы, которые публикуют сообщения в топик

Консьюмер – процессы, которые подписываются на топик и обрабатывают ленту опубликованных сообщений  
Брокер – сервер кластера

Топики и партиции кафки:  
топики состоят из партиций  
Пиртиция – постоянно растущая последовательность сообщений

Каждое сообщение хранится только в одной партиции

Репликация

Одна ведущая и ноль или более реплик для партиции

Лидер обрабатывает все запросы на чтение и запись для партиции в то время как ведомые партиции пассивно копируют лидера

Чтение-запись  
Продюсерс пишут в конец топика

Сообщения являются неизменяемыми парами ключ-значение

Каждое сообщение получает свой уникальный послед. Идентификатор

У каждого читателя есть свой указатель последнего прочитанного сообщения

Читатели могут менять свои указатели программно

Acknowledgemenet – Подтверждение записи

Лог компакшн – удаляет сообщения, если есть более свежие с тем же первичным ключом

ZooKeeper – это сервис, используемый кластером (группой узлов) для координации между собой и поддержки общих данных с помощью надежных методов синхронизации на разных хостах. ZooKeeper сам по себе является распределенным приложением, предоставляющим сервисы для написания распределенного приложения.

Клиент  
Клиенты, один из узлов распределенного кластера приложений, получают доступ к информации с сервера. В течение определенного промежутка времени каждый клиент отправляет сообщение на сервер, чтобы сервер знал, что клиент жив. Точно так же сервер отправляет подтверждение, когда клиент соединяется. Если нет ответа от подключенного сервера, клиент автоматически перенаправляет сообщение на другой сервер.  
  
Сервер  
Сервер, один из узлов ансамбля ZooKeeper, предоставляет все услуги клиентам. Дает подтверждение клиенту, чтобы сообщить, что сервер жив.

Ансамбль  
Группа серверов ZooKeeper. Минимальное количество узлов, необходимое для формирования ансамбля, составляет 3

Лидер  
Узел сервера, который выполняет автоматическое восстановление в случае сбоя любого из подключенных узлов. Лидеры избираются при запуске сервиса.

Последователь  
Серверный узел, который следует инструкции сервера-лидера.

Апачи Скуп

Структурированные данные обычно хранятся в РСУБД

Использование мап-редьюс для прямого доступа к данным в субд является нетривиальным, подверженным ошибкам и опасным для РСУБД  
данные после ЕТЛ хранятся ХДФС, по разным причинам их необходимо перемещать во внешнее хранилище.

Скуп – инструмент командной строки который принимает набор параметров и генерирует мап-задачи для выполнения передачи данных.

Импортирует отдельную таблицу из СУБД в ХДФС

Экспорт набора файлов из ХДФС обратно в РСУБД

Поддерживается набор утилит: eval, sqoop-job, sqoop-merge и тд

Апачи Флюми

Перенос данных с тысяч внешних серверов в хадуп практически в реальном времени

Поточный подход

Разделение производителей и потребителей

Масштабируемость

Настраиваемость

Надежность

Технологии и методы анализа Big Data

Статистический анализ

Суть метода заключается в сборе данных, их изучении на основе конкретных параметров и получении результата, выраженного, как правило, в процентах. У этого метода есть слабое звено — неточность данных в маленьких выборках. Поэтому для получения максимально точных результатов необходимо собирать большой объем исходных данных.

Некоторые маркетинговые методы исследования, например, А/В тестирование относятся к статистической аналитике. A/B тестирование чаще всего используют для увеличения конверсии, а само тестирование складывается из сравнения двух групп: контрольной — не подвергающейся изменениям, и второй группы, на которую оказывали какое-либо влияние (например, ей показывали другой формат рекламы). Такое тестирование позволяет понять, что улучшает целевые показатели.

Для получения точных статистических результатов используют разные методы. Вот некоторые из них:

Простой подсчет процентного соотношения.

Вычисление средних значений данных, иногда распределенных по группам.

Корреляционный анализ, который помогает выявить взаимосвязи и понять, как изменение одних данных повлияет на другие.

Метод динамических рядов, который оценивает интенсивность и частоту изменений данных с течением времени.

Метод смешения и интеграции данных  
Работа с big data часто связана со сбором разнородных данных из разных источников. Чтобы работать с этими данными, их нужно собрать воедино. Просто загрузить их в одну базу нельзя — разные источники могут выдавать данные в разных форматах и с разными параметрами, которые не будут иметь взаимного соотношения. Тут и поможет смешение и интеграция данных — процесс приведения разнородной информации к единому виду основанный на подходе ETL.

Чтобы использовать данные из разных источников, используют следующие методы:

Приводят данные к единому формату: распознают текст с фотографий, конвертируют документы, переводят текст в цифры.

Дополняют данные. Если есть два источника данных об одном объекте, информацию от первого источника дополняют данными от второго, чтобы получить более полную картину.

Отсеивают избыточные данные: из лишней информации отфильтровывают и удаляют ту, которая недоступна для анализа.

Смешение и интеграция данных нужны, если есть несколько разных источников данных, и нужно анализировать эти данные в комплексе.

В качестве примера метода интеграции и смешения данных можно рассмотреть: магазин, который ведет торговлю в нескольких каналах — оффлайн-продажи, маркетплейс и соцсети. Чтобы получить полную информацию о продажах и спросе, надо собрать множество данных: кассовые чеки, товарные остатки на складе, интернет-заказы, заказы через маркетплейс и соцсети. Все эти данные поступают из разных мест и обычно имеют разный формат. Чтобы работать с ними, их нужно привести к единому виду для дальнейшего анализа.

Машинное обучение и Нейросети

Нейронные сети работают по единому алгоритму — получают на входе данные, «прогоняют» их через сеть своих нейронов и на выходе выдают результат, например, относят входные данные к определенной группе.

Чтобы нейросеть работала, ее нужно сначала обучить. Возьмем пример.

Представим, что нужно научить нейросеть различать мужчин и женщин по фото. Для этого понадобится:

Построить нейросеть — запрограммировать искусственные нейроны воспринимать входные данные и создавать связи.

Передать нейросети очищенную выборку — базу лиц, однозначно отмеченных как женские или мужские. Так нейросеть поймет, по каким критериям отличать лица, то есть научится это делать.

Протестировать нейросеть — передать ей новую очищенную выборку, но не сообщать, какие лица мужские, а какие женские. Это поможет понять, как часто нейросеть ошибается, и приемлем ли для вас такой уровень ошибок.

После обучения и тестов можно использовать нейросеть для обработки big data.

Data Mining

Метод подразумевает обнаружение определенных закономерностей в сырых данных с помощью интеллектуального анализа.

Data mining решает несколько основных задач:

Классификация— распределение данных по заранее известным классам.

Кластеризация — распределение данных на группы по степени похожести друг на друга. Например, составление разных портретов покупателей на основе их поведения в магазине.

Ассоциация — поиск повторяющихся образцов данных. Например, одинаковых наборов продуктов в чеках покупателей.

Регрессионный анализ — нахождение важных факторов, влияющих на какой-либо заданный параметр.

Анализ отклонений— выявление нетипичных данных, резко отличающихся от обычных.

Везде, где из больших данных нужно извлекать какие-то тенденции и закономерности. Решение большинства задач компании, связанных с данными, сводится к той или иной задаче data mining или их комбинации. Например, оценить риски можно с помощью регрессионного анализа, сегментировать покупателей с помощью кластеризации, предсказать спрос по выявлению ассоциаций в данных и так далее.

Стратегия краудсорсинга

В некоторых ситуациях, когда нет экономической выгоды в разработке системы искусственного интеллекта, для выполнения разовых работ привлекают большое количество людей. Для этих целей существует краудсорсинг — привлечение к решению какой-либо проблемы большой группы людей.

Примером может быть сбор и обработка данных социологического опроса. Такая информация может находиться в неоцифрованном виде, в ней могут быть допущены ошибки и сокращения. Такой формат будет понятен человеку, и он сможет организовать данные в тот вид, который будет читаем алгоритмами программ.

Или предположим, у вас есть большой объем сырых данных. Например, записи о продажах магазинов, где товары часто записаны с ошибками и сокращениями. К примеру, дрель Dexter с аккумулятором на 10 мАч записана как «Дрель Декстр 10 мАч», «Дрель Dexter 10», «Дрель Dexter акк 10» и еще десятком других способов. Вы находите группу людей, которые готовы за деньги вручную просматривать таблицы и приводить такие наименования к одной форме. Яркий пример – Яндекс Толока.

Краудсорсинг хорош, если задача разовая и для ее решения нет смысла разрабатывать сложную систему искусственного интеллекта. Если анализировать большие данные нужно регулярно, система, основанная на Data Mining или машинном обучении, скорее всего, обойдется дешевле краудсорсинга. Кроме того, машины лучше справятся со сложным анализом, основанном на математических методах, например, со статистикой или имитационным моделированием.

Метод предиктивной аналитики

Часто нужно не просто анализировать и классифицировать исторические данные, а делать на их основе прогнозы о будущем. Например, по продажам за прошлые 10 лет предположить, какими они будут в следующем году.

Задача предиктивной аналитики — выделить несколько параметров, которые влияют на данные. Например, мы хотим понять, продолжит ли крупный клиент сотрудничество с компанией.   
Для этого изучаем базу прошлых клиентов и смотрим, какие параметры или факторы клиентов повлияли на их поведение. Это может быть объем покупок, дата последней сделки или даже неочевидные факторы вроде длительности общения с менеджерами. После этого с помощью математических функций или нейросетей строим модель, которая сможет определять вероятность отказа для каждого клиента и предупреждать об этом заранее.

Предиктивная аналитика нужна чтобы строить прогнозы. Одними из первых ее начали использовать трейдеры, чтобы предсказывать колебания курсов на бирже. Сейчас такую аналитику используют в различных сферах, чтобы предсказывать:

продажи и поведение клиентов в маркетинге;  время доставки грузов в логистике;

мошенничество в банковской и страховой сферах;

рост компании и финансовые показатели в любых сферах.

предрасположенность к каким-либо заболеваниям пациентов в медицине.

Технология имитационного моделирования

Имитационное моделирование отличается от методики прогнозирования тем, что берутся в учет факторы, чье влияние на результат затруднительно отследить в реальных условиях. Т.е. выстраиваются модели будущего с учетом гипотетических, а не реальных данных.

Метод имитационных моделей применяют для анализа влияния разных обстоятельств на итоговый показатель. Например, в сфере продаж таким образом исследуют воздействие изменения цены, наличия предложений со скидками, количества продавцов и прочих условий. Различные вариации изменений помогают определить наиболее эффективную модель маркетинговой стратегии для внедрения в практику.

Важно помнить, что даже в масштабной модели часто бывают учтены не все факторы. Для такого рода моделирования необходимо использовать большое число возможных факторов, чтобы снизить риски недостоверности результатов.

Визуализация аналитических данных

Для удобства оценки результатов анализа применяют визуализацию данных. Данный метод используется везде, где с данными должны работать люди. Основной плюс визуализации в том, что такой формат данных воспринимается лучше, чем текстовый, ведь до 90 % всей информации человек усваивает с помощью зрения.

Обычно визуализация — это конечный этап, демонстрация результатов анализа, проведенного другими способами. Например, вы построили имитационную модель и выводите результат ее работы в виде графика, который показывает колебание продаж в зависимости от изменений цены. Или сравнили продажи в разных регионах и визуализировали эти данные на карте, раскрасив регионы в разные цвета.

Результат анализа визуализируют в виде графиков, 3D-моделей, диаграмм и т. д. Инструментарий такого метода представлен платформами: Orange, Microsoft (Excel, Power BI), Microstrategy, Qlik, Tableau, SAP, Oracle.

Метод визуализации аналитических данных позволяет быстро воспринять и сравнить, например, уровни продаж в разных регионах, или оценить зависимость объемов продаж от снижения/увеличения стоимости товара.

ЯП для бигдаты

Архитектура Фон Неймана

В 1946 году американский учёный Герман Голдстайн опубликовал доклад математика Джона фон Неймана «Предварительное рассмотрение логической конструкции электронно-вычислительного устройства». Автор доклада изложил принципы, ставшие основой архитектуры всех последующих поколений ЭВМ:

Память компьютера состоит из ячеек, каждой из которых присвоен номер — адрес. Любая из ячеек доступна компьютеру в любой момент времени, и он может обратиться к её содержимому по адресу.

Память компьютера используется как для хранения данных, так и для хранения программы. Команды и числа кодируются в двоичной системе счисления. Над командами можно производить те же операции, что и над числами. На этом принципе основана трансляция — перевод текста программы с языка высокого уровня на язык конкретной ЭВМ.

Компьютером управляет программа, которая состоит из команд, хранящихся в последовательных ячейках памяти. Каждая команда выполняет операцию из набора операций, доступных компьютеру.

Компьютер выполняет команды последовательно — от первой к последней, согласно порядку следования в программе. Этот порядок может быть изменён с помощью специальных команд в зависимости от результатов вычислений.

Инновации за пределами Закона Мура

Процессоры стали многоядерными и многопоточными.

Появилась масса сопроцессоров, которые невероятно эффективно решают отдельные задачи: обработка фотографий, кодирование видео, нейронные движки, облачные вычисления.

Перенос вычислений на видеокарты.

Оптимизация софта.

Производителям железа пришлось ежегодно совершенствовать свою продукцию, а не только CPU.

Дальше произойдет отказ от текущей концепции центрального процессора, основанной на архитектуре Фон Неймана и произойдет переход на асинхронные нейроморфные процессоры.

R

Идеальный вариант для первоначальных целей R был создан для работы со статистикой в 1997 году в качестве альтернативы SAS и MATLAB. Он позволяет собирать и очищать данные, работать с таблицами, проводить статистические тесты, различные виды анализа и составлять графические отчеты. R подойдет для специалистов, знакомых с теорией вероятности, статистическими методами и математическим анализом, поэтому на первый взгляд он может показаться сложным из-за интуитивно непонятного синтаксиса.

На практике R используют:

для научных исследований в разных сферах;

машинного обучения и нейросетей;

маркетинговых исследований.

Преимущества:

Отличный набор высококачественных предметно-ориентированных пакетов с открытым исходным кодом. R имеет в своем распоряжении пакеты практически для любого количественного и статистического применения. Сюда входят нейронные сети, нелинейная регрессия, построение сложных диаграмм, графиков и многое-многое другое.

Вместе с базовой установкой в довесок предоставляется возможность установки обширных встроенных функций и методов. Кроме того, R прекрасно обрабатывает данные матричной алгебры.

Возможность визуализации данных является немаловажным преимуществом наряду с возможностью использования различных библиотек (ggplot2).

Недостатки:

Низкая производительность. R не является быстрым ЯП.

Специфичность. R прекрасно подходит для статистических исследований, но он не так хорош, когда дело доходит до программирования для общих целей.

Другие особенности. R имеет несколько необычных особенностей, которые могут сбить с толку программистов, привыкших работать с другими ЯП: индексирование начинается с 1, использование нескольких операторов присваивания, нетрадиционные структуры данных.

Python

Удобен во всех отношениях В 1991 году был представлен язык программирования Python. С тех пор этот он стал чрезвычайно популярным языком программирования общего назначения и широко используется в сообществе специалистов по данным.

Python является хорошим вариантом для целей Data Science, и это утверждение справедливо как для начального, так и для продвинутого уровней работы в данной области. Это один из самых универсальных и популярных языков программирования, который полюбился и специалистам Big Data. Библиотеки!

В отличие от R, Python является традиционным объектно-ориентированным языком программирования, поэтому большинству разработчиков комфортно с ним работать.

Python поддерживается фреймворками обработки больших данных, но в то же время он не входит в высшую лигу в этой сфере. Например, новые функции Spark почти всегда выходят в первую очередь для Scala / Java, и только потом уже для PySpark.

Преимущества:

Python – это очень популярный, широко используемый язык программирования общего назначения. Он имеет обширный набор специально разработанных модулей и широко используется разработчиками. Многие онлайн-сервисы предоставляют API для Python.

Python очень прост в изучении. Низкий порог вхождения делает его идеальным первым языком для тех, кто занимается программированием.

Отлично применим как для задач обработки данных, так и в области машинного обучения.

Недостатки:

Python – это динамически типизированный язык, а это значит, что следует быть аккуратным при работе с ним. Ошибки несоответствия типов могут время от времени случаться и приводить к ошибкам.

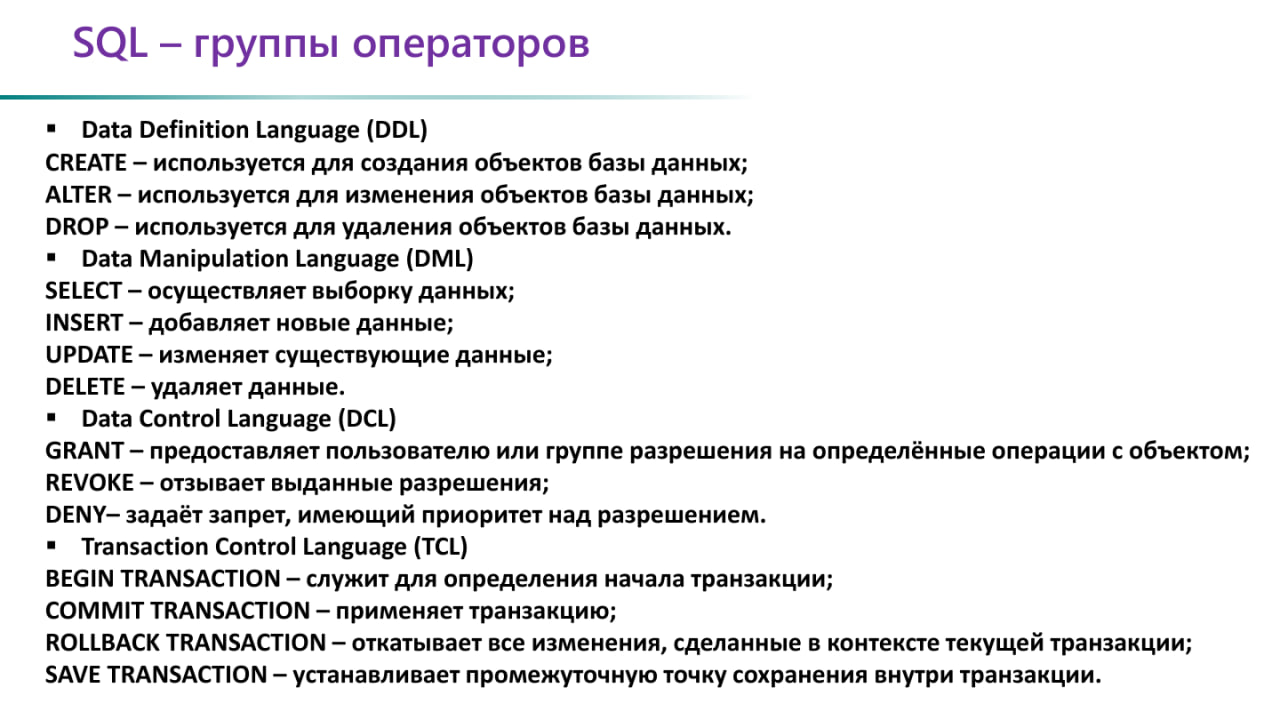
Для целей статистического анализа и анализа данных обширный набор пакетов языка R имеет преимущество перед Python. Кроме того, существуют более быстрые и безопасные альтернативы Python среди языков программирования.

SQL

Эффективность, доказанная временем

SQL («язык структурированных запросов») появился в 1974 году и с тех пор претерпел множество видоизменений, но основные его принципы остаются неизменными.

SQL более полезен в качестве языка для обработки данных, чем в качестве передового аналитического инструмента. Тем не менее, так много процессов в области data science зависит от ETL, а долговечность и эффективность SQL лишний раз свидетельствуют о том, что такой язык программирования должен знать каждый специалист по данным (data scientist).



Преимущества

Очень эффективен при работе с запросами, обновлениями, а также при обработке реляционных баз данных.

Декларативный синтаксис делает SQL очень читаемым языком. Нет никакой неопределенности в том, что запрос «SELECT name FROM users WHERE age > 18» должен делать.

SQL очень часто используется в различных приложениях, поэтому зная его можно решить достаточно широкий круг задач.

Недостатки:

Синтаксис SQL может показаться достаточно сложным для тех, кто привык к императивному программированию.

Существует множество различных вариаций и диалектов SQL, таких как PostgreSQL, SQLite, PLSQL. Все они достаточно разные и придется вникать в тонкости каждого при работе.

Java

Претендент на звание лучшего языка для работы в области науки о данных (универсальность)

Это практически универсальный язык, который может использоваться во многих областях. Big Data — не исключение. Это не профильная область применения для данного языка. Такие компании как LinkedIn, Facebook и Twitter давно используют элементы Java в обработке данных. Да, у этого языка нет таких возможностей статистического моделирования, какие есть у Python и R, но если прототипирование не является важной задачей — Java может стать отличным решением.

На Java написано множество Big Data инструментов с открытым кодом (например, большая часть экосистемы Hadoop), поэтому разработчики могут на их основе создавать собственные продукты для управления данными.

Преимущества:

Универсальность. Многие современные системы и приложения разработаны с помощью языка Java. Огромным преимуществом такого языка программирования является способность интегрировать методы Data Science непосредственно в существующую кодовую базу.

Строгая типизация. Обеспечение типобезопасности есть в Java, и в случае разработки критически важных приложений для работы с большими данными эта особенность как никогда важна.

Java – это высокопроизводительный, скомпилированный язык общего назначения. Это делает его пригодным для написания эффективного производственного кода ETL, а также алгоритмов машинного обучения с использованием вычислительных средств.

Недостатки:

«Многословность» языка Java делает его не лучшим вариантом для проведения специальных анализов и разработки более специализированных статистических приложений.

Java не имеет большого количества библиотек для передовых статистических методов по сравнению с некоторыми предметно-ориентированными языками, например R.

Scala

Идеальный вариант для работы с большими данными

Функционирующий на JVM язык программирования Scala был разработан в 2004 году. Это язык с несколькими парадигмами, позволяющий использовать как объектно-ориентированные, так и функциональные подходы.

Если необходимо использовать кластерные вычисления для работы с большими данными, то пара Scala + Spark – это идеальное решение. Однако, если приложение не имеет ничего общего с большими объемами данных, работа с которыми может оправдать добавление всех составляющих Scala, то, скорее всего, добиться большей производительности можно, используя другие языки, такие как R или Python.

На Scala написан фреймворк Apache Spark, важный для машинного обучения и анализа больших данных. Этот фреймворк входит в экосистему Hadoop и позволяет параллельно обрабатывать неструктурированные данные в реальном времени. Он легко взаимодействует с кодом на Java и библиотеками этого языка.

Преимущества:

Используя Scala и Spark становится возможным работать с высокопроизводительными кластерными вычислениями. Scala – это идеальный выбор для работы с большими объемами данных.

Мультипарадигматический. На Scala доступны как объектно-ориентированные, так и функциональные парадигмы программирования.

Scala компилируется в байт-код Java и работает на JVM. Это позволяет ему взаимодействовать с языком Java, делая Scala очень мощным языком общего назначения. Кроме того, он также хорошо подходит для работы в области Data Science.

Недостатки:

Синтаксис и система типов Scala являются достаточно сложными.

Julia

Язык, который себя еще проявит

Выпущенный чуть более 5 лет назад, Julia произвела впечатление на мир вычислительных методов. Язык добился такой популярности благодаря тому, что несколько крупных организаций почти сразу начали использовать его для своих целей.

Главная проблема языка Julia – это его молодость. Поскольку Julia был создан лишь недавно, он пока что не может конкурировать со своими основными конкурентами: Python и R. И тем не менее он быстро заполучил любовь многих специалистов Big Data. Язык отличается простотой и широкими возможностями масштабирования. В этом он напоминает Python. А вот по эффективности Julia вполне может сравниться с R. Поскольку язык новый, у него есть небольшие недоработки. Также пока нет качественных библиотек и организованной поддержки от программистского сообщества.

Преимущества:

Julia – это высокопроизводительный язык. Также он является достаточно простым и предусматривает возможности динамической типизации, и сценариев интерпретируемого языка, такого как Python.

Julia предназначен для проведения численного анализа, он также может рассматриваться в качестве языка программирования общего назначения.

Читаемость. Многие программисты, работающие с данным языком, считают, что такая особенность является его наибольшим преимуществом.

Недостатки:

Незрелость. Поскольку Julia является достаточно новым языком, некоторые разработчики сталкиваются с нестабильностью во время работы с его пакетами. Тем не менее, базовые средства языка считаются стабильными.

Еще одним признаком незрелости языка является ограниченное количество пакетов программ, а также небольшое число поклонников среди разработчиков. В отличие от устоявшихся R и Python, язык программирования Julia не располагает большим количеством пакетов программ.

MATLAB

Лучший вариант для целей, требующий значительных математических расчетов

MATLAB – это признанный язык для численных расчетов, используемый как в научных целях, так и в индустрии. Он был разработан и лицензирован MathWorks, основной целью которой являлось коммерциализация программного обеспечения.

Благодаря своему широкому использованию в различных количественных вычислениях как для научных целей, так и для целей индустрии, MATLAB стал достойным вариантом для применения в области науки о данных. Он придётся как нельзя кстати, если для ежедневных целей необходима интенсивная, продвинутая математическая функциональность, собственно, для чего MATLAB и был разработан.

Это не всегда эффективный инструмент с не слишком доступной средой. Однако при грамотном подходе он дарит почти безграничное количество возможностей в сфере обработки и моделирования данных. MATLAB можно успешно использовать в науке, в анализе имеющейся и прогнозируемой информации.

Преимущества:

MATLAB, предназначенный для численных вычислений, хорошо подходит для использования количественного анализа со сложными математическими требованиями, такими как обработка сигналов, преобразования Фурье, матричная алгебра и обработка изображений.

Визуализация данных. MATLAB имеет ряд встроенных возможностей построения графиков и диаграмм.

MATLAB используется в педагогической среде.

Недостатки:

Платная лицензия. Вне зависимости от выбранного вами варианта (для научных, личных целей или целей компании) вам придется заплатить за дорогостоящую лицензию.

MATLAB – это не лучший язык программирования для общего назначения.

Go

Язык программирования Go создан компанией Google для работы с большими данными, поэтому сейчас он используется в большинстве продуктов компании:

для работы с искусственным интеллектом;

работы с базами данных;

веб-разработки (особенно для backend).

Go активно разрабатывается, внедряется, постепенно завоевывает позиции в рейтинге популярности, и уж абсолютно точно он не хуже подавляющего большинства своих конкурентов. Он прост, удобен, перспективен, и специально создан для Big Data.

Go Преимущества

Развитая инфраструктура.

Встроенная защита.

Встроенная среда тестирования и автоматическая документация.

Недостатки

Нет универсальности.

Небольшая коллекция библиотек.

C++

Не лучший выбор для повседневных задач, но если дело касается производительности...

C++ — язык общего назначения. Это значит, что с его помощью можно решить задачу из любой области программирования. Чаще всего на нем пишут операционные системы, крупные игры и такие пакеты программ, как MS Office или Adobe. В Big Data он тоже используется в основном для создания инструментов обработки данных, а не для непосредственной работы с ними. Например, MapReduce, который сейчас входит в экосистему Hadoop, изначально был написан как раз на C++.

C++ быстрее, чем многие конкуренты (Go, R или Python). Особенно это востребовано в машинном обучении, где нужно быстро обрабатывать терабайты данных. Это единственный язык, на котором данные размером более 1 Гб могут быть обработаны за секунду.

Но при этом он действительно сложный в изучении. В 2009 году компания Google создала простой и понятный язык Go, который справлялся бы с задачами C++, высокой нагрузкой и большими объемами данных.