Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

ТЕХНОЛОГИИ БИЗНЕС-АНАЛИТИКИ

Методические указания к выполнению лабораторных работ
и организации самостоятельной работы для студентов направления
«Бизнес-информатика» (уровень бакалавриата)

2022

Николаенко Валентин Сергеевич
Технологии бизнес-аналитики: Методические указания к выполнению лабораторных работ и организации самостоятельной работы для студентов направления «Бизнес-информатика» (уровень бакалавриата) / В.С. Николаенко. – Томск, 2022. – 259 с.

СОДЕРЖАНИЕ

Введение

В 2011 году вышел фильм «Moneyball» с Брэдом Питтом в главной роли. В российском прокате фильм известен под названием «Человек, который изменил все».

Это история о спортивном менеджере, который пытается добиться больших высот с заурядной бейсбольной командой. Билл Бин (герой Питта) разочаровывается в классическом подходе к подбору игроков – на основе наблюдения скаутов и их экспертных оценок. Так, при обсуждении кандидатур Бин слышит от коллег аргументы в духе: «Он не нравится моей жене» – и прочие непрофессиональные суждения. В команде полностью отсутствует системный подход.

Однажды в офисе у знакомого Бин встречает молодого специалиста по статистике и анализу данных. Вместе они начинают анализировать большие объемы информации о технических действиях сотен игроков. По результатам статистического анализа Бин собирает команду из спортсменов, которые хороши только в одном или двух игровых действиях: броске, приеме, перемещениях между базами и др. Как менеджер команды Билли Бин использует игроков лишь в наиболее выгодных для них ситуациях. Таким образом, он становится первым кто отказывается от субъективного поиска игроков, опирающегося на личный опыт и впечатления. Взяв в союзники статистику и аналитику, менеджер за значительно меньшие деньги формирует бейсбольный коллектив. Впоследствии такой подход перенимают и другие команды.

Фильм «Moneyball» является ярким примером который показывает *суть аналитики* – помочь в принятии решений на основе данных. Данные используемые аналитиком в работе, могут представлять собой все что угодно: статистические таблицы, карты бизнес-процессов, техническую документацию и др.

Соответственно, задача аналитика состоит в том, чтобы собрать необходимые данные, проанализировать их и представить руководителю для принятия взвешенного и обоснованного управленческого решения.

Анализ – это логический прием, с помощью которого субъект мысленно разбивает предметы и явления, выделяя их отдельные части и свойства. Обратный анализу логический прием называется *синтез*.

На основании данных определений можно заключить, что аналитик стремится разложить на составные части сложный предмет или явление, чтобы понять как они устроены. Например, если аналитик работает в организации, занимающейся разработкой программ для ЭВМ или

автоматизацией деятельности, то объектом его анализа будут бизнес-процессы заказчика.

Рассмотрим наиболее распространенные виды аналитики.

Бизнес-анализ. Согласно Business Analysis Body of Knowledge (BABOK) под *бизнес-анализом* понимается деятельность, которая позволяет внедрять изменения в организацию путем определения потребностей заинтересованных сторон.

С позиции бизнес-анализа у организации есть два состояния: текущее состояние и целевое состояние.

Текущее состояние – это положение дел, которое по тем или иным причинам не устраивает руководство организации. Проблемы могут касаться длительности выполнения процессов, высоких заработных плат, неясных зон ответственности и др.

Целевое состояние – это состояние, при котором организация проанализировала свои слабые места и оптимизировала бизнес-процессы: повысила их прозрачность и эффективность.

Роль бизнес-аналитика заключается в оказании помощи идентификации проблемных зон организации в ее текущем состоянии и предложении вариантов перехода в целевое состояние.

Системный анализ. Если бизнес-аналитик отвечает на вопрос «что нужно сделать?», то системный аналитик определяет, «как сделать то, что нужно заказчику». Задача системного аналитика – предложить заказчику варианты реализации требований, полученных от бизнес-аналитика.

Рассмотрим работу системного аналитика в ключе сферы информационных технологий.

Системный аналитик, как правило, является экспертом, который разбирается в работе конкретного программного продукта или информационной системы. Он знает основные технологические процессы системы, принципы хранения информации в ней, нюансы интеграции с другими системами и др.

Системный аналитик формирует *техническое задание*, в котором на более детальном техническом уровне, нежели бизнес-аналитик, описывает требования к будущей программе для ЭВМ. Если бизнес-аналитик говорит о том, что должно произойти при нажатии на определенную кнопку, то системный аналитик фиксирует, за счет чего достигается необходимый результат: как происходит обращение к серверу, как обрабатывает ответ, как хранится информация в базе данных и др.

Продуктовая аналитика. Продуктовая аналитика посвящена разработке и развитию какого-либо продукта [1].

При работе с программными продуктами главным инструментом продуктового аналитика становится A/B-тестирование. *A/B-тестирование* – это маркетинговый метод, использующийся для оценки и управления эффективностью веб-страницы. Суть метода заключается в том, что создается веб-страница A, копируется и в копии (страница B) меняется какой-либо параметр, например, заголовок. Затем половине пользователей показывается страница A, а другой половине – страница B и оценивается какая из двух страниц имеет большую конверсию.

Рассмотрим работу A/B-тестирования на примере. Допустим у заказчика есть интернет-магазин и он хочет добавить к нему пару новых функций – красивую кнопку оформления заказа и 3D-обзор товара. Для удобства назовем их «Функция 1» и «Функция 2». Текущее состояние интернет-магазина без новых функций будет служить контрольным показателем. Подрядчик создает несколько версий интернет-магазина: только с «Функция 1», только с «Функция 2» и обеими «Функциями» сразу.

Для того, чтобы понять как новые изменения и их комбинации повлияли на работу магазина по сравнению с контрольным показателем подрядчик демонстрирует новые версии интернет-магазина фокус-группе пользователей.

Во время работы с фокус-группой подрядчик устанавливает, что смена цвета кнопки с синего на красный приводит к оттоку пользователей, а смена на зеленый цвет повышает кликабельность.

Финансовый анализ. Финансовый анализ связан с изучением финансово-хозяйственной деятельности организации. Финансовый аналитик постоянно сталкивается в своей работе с:

- планами развития организации;
- разработкой и контролем ключевых показателей эффективности деятельности;
- управлением запасами;
- ассортиментной и ценовой политикой;
- подготовкой отчетов для контролирующих органов;
- формированием бюджетов организации и др.

Аналитика данных (Business Intelligence, BI). Аналитик данных решает следующие задачи:

- Построение прозрачных моделей на основе больших массивов данных. Например, прогнозов оттока клиентов на основе анализа данных об их активности с момента появления в клиентской базе.
- Разработка механизмов персональных рекомендаций на основе анализа больших объемов данных.

- Выявление скрытых аномалий и закономерностей в данных.

Для решения подобных задач аналитику данных нужны глубокие знания в области математики, статистики и эксплуатации аналитических информационных систем поддержки принятия решений (OnLine Analytical Processing, OLAP) [2].

Глава 1. ВВЕДЕНИЕ В АНАЛИТИКУ ДАННЫХ

1.1. Развитие аналитики данных

По одной из классификаций анализ данных проистекает из задач прикладной математики (рисунок 1.1). Кроме анализа данных, в данной классификации выделяют еще два направления: вычислительная математика и идентификация моделей. Так как исторически они возникли первыми, их еще называют ***классическими подходами прикладной математики***.

Рисунок 1.1 – Классификация задач прикладной математики

Вычислительная математика решает задачу вычисления одних характеристик изучаемого объекта или явления по известным значениям других его характеристик. При этом модель объекта считается известной, а зависимости между характеристиками описываются аналитическим выражением в виде уравнения или системы уравнений. Проблемы, возникающие при решении таких задач, связаны, в основном, с большими объемами вычислений и с защитой от погрешностей, накапливающихся из-за округления чисел.

Задача ***идентификации модели*** формулируется по-другому. Известен набор переменных, влияющих на целевую характеристику, известен также общий вид зависимости между характеристиками, но коэффициенты, показатели степени и другие параметры модели неизвестны, и, чтобы их определить, используются протоколы наблюдений, отражающие значения одних характеристик при разных значениях других.

Делается серия предположений о значениях неизвестных параметров модели, и эти предположения проверяются на протоколах. В результате выбираются такие значения параметров, при которых модель с заданной точностью позволяет по одним (входным) характеристикам определять другие (выходные или целевые) характеристики. К таким задачам принято относить дедуктивные процедуры математической статистики: корреляционный и

регрессионный анализы, факторный анализ, численные методы оптимизации и др.

Рассмотрим термины «модель» и «моделирование» подробнее.

Слово ***модель*** *(лат.: modelium)* означает «меру», «способ», «сходство с какой-то вещью». Построение моделей – универсальный способ изучения окружающего мира, позволяющий обнаруживать зависимости, прогнозировать, разбивать на группы и решать множество других задач.

Модель – это объект или описание объекта, системы для замещения (при определенных условиях, предположениях, гипотезах) одной системы (то есть оригинала) другой системой для лучшего изучения оригинала или воспроизведения каких-либо его свойств.

Моделирование – универсальный метод получения, описания и использования знаний. Применяется в любой профессиональной деятельности. В прикладной математике имеют дело, как правило, с математическими моделями.

Основная цель моделирования в том, что модель должна достаточно хорошо отображать функционирование моделируемой системы.

Рисунок 1.2 – Процесс анализа данных в вычислительной математике

На рисунке 1.2 изображен процесс анализа в вычислительной математике, так называемый ***классический подход***. Для решения задач в данном подходе выбираются готовые математические модели с известными параметрами. Модели проверяются на основе имеющихся данных. Расчеты осуществляются с помощью специализированного программного обеспечения.

Сразу видны минусы подхода. Риск сделать ошибку в выборе модели и ее параметров остается вне поля внимания. За адекватность модели и ее параметров изучаемому явлению исследователь не отвечает. Например, если в рассматриваемом явлении переменная **Возраст** влияет на выходную переменную, и эта связь носит логарифмический характер, то log(Возраст) должен присутствовать в модели.

Конечно же, исследователь, решая современные задачи анализа данных, практически никогда не знает ни точный вид «истинной» модели, ни характер связей между переменными, ни исчерпывающий перечень самих переменных.

Здесь становится актуально высказывание известного статистика Дж. Бокса *«...все модели неправильные, но некоторые из них полезные»*.

Задачи ***идентификации моделей*** требуют от исследования ответственности за правильный выбор параметров модели (сама модель считается неизвестной). Наличие этого рискованного шага в процессе решения задачи также лишает результат строкой математической чистоты.

На практике часто возникают задачи, для решения которых нет готовых математических моделей, поэтому в середине XX века разрабатываются новые подходы к анализу данных. В частности, в 1962 году Джон Тьюки (John Tukey) предложил концепцию ***разведочного анализа данных*** (англ.: Exploratory Data Analysis, EDA). Дж. Тьюки был убежден, что можно многое узнать из данных, просто визуализируя их. Этот первичный этап анализа он и назвал ***разведочным,*** а важнейшим его элементом определил широкое использование визуального представления многомерных данных. Для этого данные представляются в виде графиков, схем, условных рисунков, таблиц, особенностью которых является наглядность – возможность увидеть признаки каких-либо закономерностей. Так, *ящичная диаграмма* (англ.: box plot) и *диаграмма рассеяния* (англ.: scatterplot) получили широкое распространение в статистике именно после работ Дж. Тьюки. Например, ящичная диаграмма наглядно показывает распределение данных, медиану, выбросы, центральные квартили и диапазон распределения.

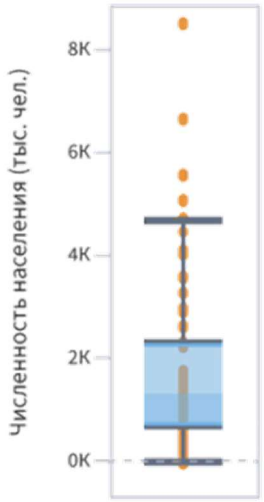


Рисунок 1.3 – Пример ящичной диаграммы

Всего в своей концепции Дж. Тьюки выделил три этапа анализа данных: разведочный; подтверждающий (конфирматорный); итоговый.

Разведочный анализ – это синтез детерминированных, стохастических и эвристических подходов к анализу выборочных наблюдений. На этом этапе

цель исследователя – выявить внутренние вероятностные и геометрические закономерности в данных для формирования и верификации тех или иных рабочих гипотез о связях между переменными, когда отсутствуют априорные представления о природе этих связей.

На следующем этапе ***подтверждающего анализа*** ставится задача проверки соответствия сформулированных гипотез полученным эмпирическим данным. Вычисляются итоговые статистические оценки моделей и определяются их погрешности.

На ***итоговом анализе данных*** проводится экспертный анализ результатов и их обобщение. В случае необходимости, на всех этапах исследования возможны итерационные уточнения и обобщения.

Можно смело утверждать, что концепция разведочного анализа Дж. Тьюки воплотилась позже в таких подходах аналитики данных как Обнаружение знаний в базах данных (KDD), Data Mining , Big Date и др.

Что касается самого процесса анализа данных, то по Дж. Тьюки он выглядит по-другому: вместо последовательности действий *Модель-Анализ-Данные* идет *Данные-Анализ (разведочный)-Модель-Анализ (подтверждающий)* (рисунок 1.4).

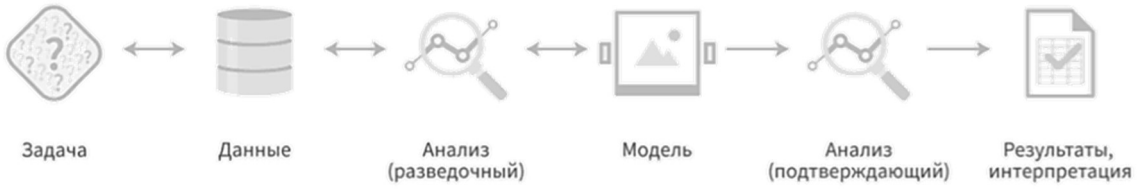


Рисунок 1.4 – Процесс анализа данных согласно концепции Дж. Тьюки

Отправной точной в процессе анализа согласно концепции Дж. Тьюки служат данные, характеризующие исследуемый объект или явление. Модель «следует» за данными, а не наоборот, как в классическом подходе. Двунаправленные стрелки показывают, что анализ данных носит циклический характер: на первых итерациях выдвинутые гипотезы могут потребовать дополнительных экспериментальных данных или наблюдений, уточнений. Это существенно облегчает подбор способов более глубокой обработки данных на этапах подтверждающего анализа.

Если результаты разведочного анализа говорят в пользу некоторой модели, то ее правильность можно проверить, применив к новым (то есть вновь измеренным) данным явления или объекта. В вычислительной математике и математической статистике такие действия невозможны; именно по этой причине процедуры разделения выборок на обучающие и тестовые

множества и валидации на них моделей пришли в аналитику данных из подходов, заложенных Дж. Тьюки.

Благодаря концепции Дж. Тьюки был создан современный анализ данных, где решается задача анализа явлений, для которых еще нет математических моделей. Есть только наборы экспериментальных данных входы-выходы и даже только входы, представленные в виде массивов или таблиц.

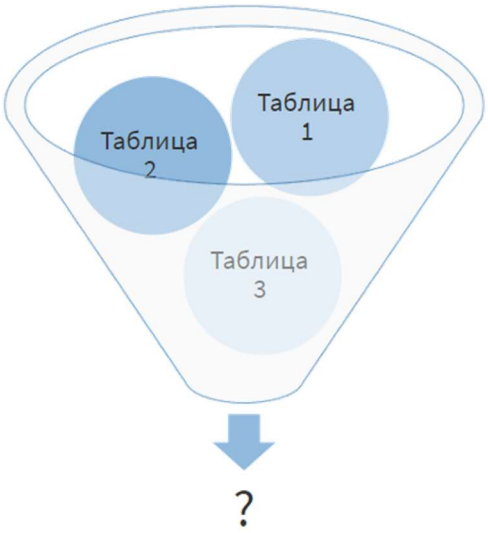


Рисунок 1.5 – Современное представление анализа данных

Конструирование моделей и определение параметров этих моделей является основным предметом внимания современного анализа данных. Исследователи отвечают за привнесение эвристических гипотез о возможных формах (моделях) зависимостей, параметрах предполагаемых законов распределений и так далее. Наряду с дедуктивным аппаратом при решении этих задач используются индуктивные методы, реализованные в методах машинного обучения.

Рисунок 1.6 – Процесс интеграции, подготовки, очистка и анализа данных

Однако концепция «моделей от данных» требует тщательного подхода к качеству самих исходных данных, поскольку ошибочные, зашумленные данные могут привести к моделям и выводам, не имеющим никакого отношения к действительности. Поэтому в анализе данных важную роль играют интеграция, подготовка и очистка данных.

1.2. Аналитика данных как процесс

Современная аналитика данных опирается на четыре составляющие: эксперт, гипотеза, аналитик и руководитель проекта. Дадим им определения.

Эксперт – это специалист в предметной области, профессионал, который за годы обучения и практической деятельности научился эффективно решать задачи, относящиеся к конкретной предметной области.

Эксперт является ключевой фигурой в процессе анализа. По-настоящему эффективные аналитические решения можно получить не на основе одних лишь компьютерных программ, а в результате сочетания лучшего из того, что может человек и компьютер. Эксперт выдвигает гипотезы (предположения) и для проверки их достоверности либо просматривает некие выборки различными способами, либо строит те или иные модели.

Гипотезой в аналитике данных часто выступает предположение о влиянии какого-либо фактора или группы факторов на результат. К примеру, при построении прогноза продаж допускается предположение, что на величину будущих продаж существенно влияют продажи за предыдущие периоды и остатки на складе. При оценке кредитоспособности потенциального заемщика выдвигается гипотеза, что на кредитоспособность

влияют социально-экономические характеристики клиента: возраст, образование, семейное положение и др.

Аналитик – это специалист в области анализа и моделирования. Аналитик на достаточном уровне владеет какими-либо инструментальными и программными средствами аналитики данных, например методами *машинного обучения* и *Data Mining*.

Аналитик играет роль «мостика» между экспертами, то есть является связующим звеном между специалистами разных уровней и областей. Он собирает у экспертов различные гипотезы, выдвигает требования к данным, проверяет гипотезы и вместе с экспертами анализирует полученные результаты. Аналитик должен обладать системными знаниями, так как помимо задач анализа на его плечи часто ложатся технические вопросы, связанные с базами данных, интеграцией с источниками данных, тестированием и производительностью.

Для технических задач, связанных с обеспечением надежной инфраструктуры для данных, в последнее десятилетние появилась отдельная профессия: *инженер по данным* или *дата-инженер*.

Главным лицом в процессе анализа данных считается аналитик, т. к. он тесно сотрудничает со всеми участниками проекта. В крупных проектах участвуют, как правило, несколько экспертов, аналитиков и руководитель проекта.

В обязанности ***руководителя проекта*** входят функции координации действий всех участников проекта, решение спорных вопросов, планирование и контроль сроков проекта.

Современная аналитика данных делит методы решения задач на две основные группы:

- извлечение и визуализация данных;
- построение и использование моделей.

На рисунке 1.7 приведена общая схема такого анализа.

Рисунок 1.7 – Классификация методов решения задач в аналитике данных

Стоит отметить, что группе задач «Извлечение и визуализация данных» по сути соответствует этап разведочного анализа данных согласно концепции Дж. Тьюки, а группа «Построение и использование моделей» – это этап подтверждающего анализа.

Чтобы получить новые знания об исследуемом объекте или явлении, не обязательно строить сложные модели. Часто достаточно «посмотреть» на данные в нужном виде, чтобы сделать определенные выводы или выдвинуть предположение о характере зависимостей в системе, получить ответ на интересующий вопрос. Это помогает сделать ***визуализация***.

В случае визуализации аналитик некоторым образом формулирует запрос к информационной системе, извлекает нужную информацию из различных источников и просматривает полученные результаты. На их основе он делает выводы, которые и являются результатом анализа.

Существует множество способов визуализации данных. Несомненными достоинствами визуализации являются относительная простота создания и введения в эксплуатацию подобных систем и возможность их применения практически в любой сфере деятельности. Кроме того, в этом случае по максимуму используются знания эксперта в предметной области и его способность принимать во внимание многие трудно формализуемые факторы, влияющие на бизнес.

Построение моделей – это универсальный способ изучения окружающего мира, позволяющий обнаруживать зависимости, прогнозировать, разбивать на группы и решать множество других важных

задач. Но самое главное: полученные таким образом знания можно тиражировать.

Тиражирование знаний – это совокупность инструментальных средств для создания моделей, которые обеспечивают конечным пользователям возможность использовать результаты моделирования для принятия решений, без необходимости понимания методик, при помощи которых эти результаты получены.

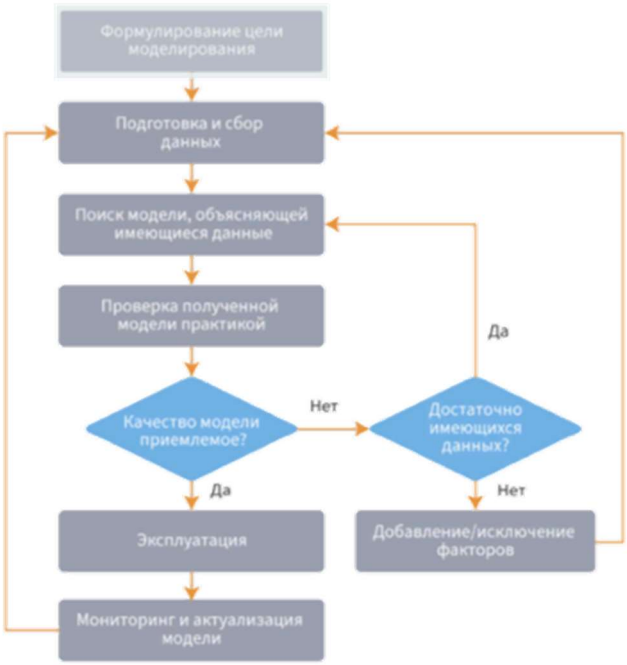


Рисунок 1.8 – Схема процесса построения моделей

Рассмотрим подробнее элементы схемы процесса построения моделей (рисунок 1.8).

Формулирование цели и моделирование. При построении модели следует отталкиваться от задачи, которую можно рассматривать как получение ответа на интересующий заказчика вопрос.

Например, в розничной торговле к таким вопросам относятся следующие:

- Сколько товара будет продано в следующем периоде?
- Какие клиенты откликаются на акции?
- Какие товары продаются или заказываются вместе?
- Как оптимизировать товарные остатки на складах?

В этом случае можно говорить о создании модели прогнозирования продаж, вероятности отклика, рекомендаций и так далее.

Подготовка и сбор данных. Аналитика данных опирается на использование подготовленных и систематизированных данных. Как правило, данные операции являются отдельными трудоемкими задачами.

Поиск модели. После сбора и систематизации данных переходят к поиску модели, которая объясняла бы имеющиеся данные, позволила бы добиться эмпирически обоснованных ответов на интересующие вопросы. В аналитике данных предпочтение отдается самообучающимся алгоритмам, машинному обучению, методам Data Mining, а в ряде случаев бывает достаточно статистических методов.

Проверка модели. Если построенная модель показывает приемлемые результаты на практике (например, в опытной эксплуатации), ее запускают в промышленную эксплуатацию. Например, внедряют *скоринговую оценку* в принятие решений о выдаче кредита или займа (см. §1.6 Прикладные задачи аналитики данных). Важно, чтобы модель улучшала существующий бизнес-процесс компании. Как правило, опытная эксплуатация делается в режиме АВ-теста: одна часть объектов обрабатываются по «старой» схеме, а вторая – на основе модели. Результаты сравниваются, в том числе с применением методов статистической обработки данных эксперимента.

Если качество модели неудовлетворительное, то процесс построения модели повторяется, как это было показано на рисунке 1.8. Кроме того, любые модели подвержены «старению» и нуждаются в актуализации, поэтому можно сказать, что аналитика данных – это *непрерывный процесс*.

Стоит отметить, что рассмотренные подходы могут комбинироваться. Например, визуализация данных наводит аналитика на некоторые идеи, которые он пробует проверить при помощи различных моделей, а к полученным результатам снова применяются методы визуализации. Механизмы визуализации и построения моделей дополняют друг друга.

Рассматривая аналитику данных как процесс, нельзя не упомянуть межотраслевой стандарт **CRISP-DM** (англ.: **CR**oss **I**ndustry **S**tandard **P**rocess for **D**ata **M**ining). По сути это популярная методология ведения проектов в аналитике данных, особенно если в них используются многофакторные модели, построенные на принципах машинного обучения.

CRISP-DM был разработан в конце 1996 года четырьмя организациями: ISL (поглощена SPSS Inc.), NCR Corporation, Daimler-Benz и OHRA.

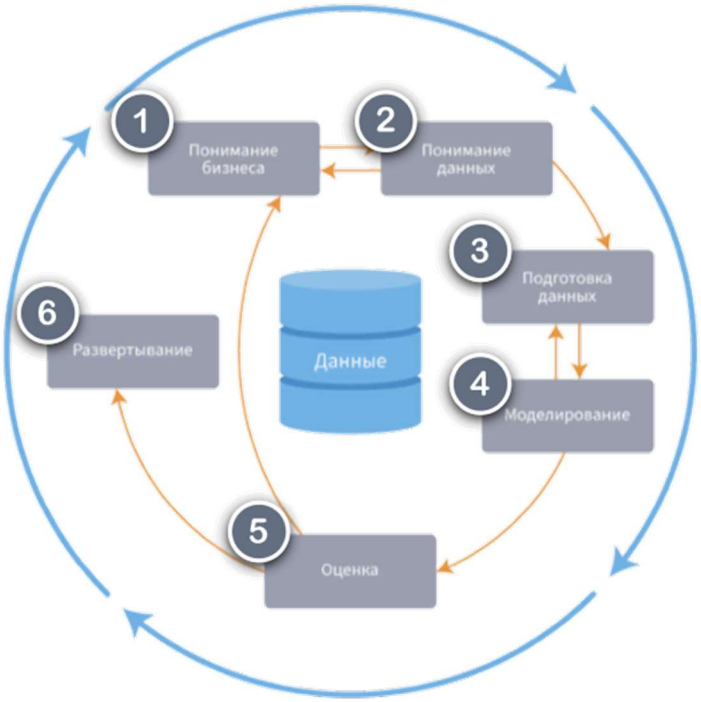


Рисунок 1.9 – Модель жизненного цикла проекта по аналитике данных CRISP-DM

На рисунке 1.9 приведена модель жизненного цикла проекта по аналитике данных CRISP-DM. Сам проект состоит из шести этапов, а последовательность этапов не является строгой. Стрелки указывают наиболее важные и частые зависимости между этапами.

Внешний круг на рисунке 1.9 указывает на цикличность процесса аналитики данных, который продолжается и после развертывания проекта. Рассмотрим этапы CRISP-DM более подробно.

Понимание бизнеса. Первый этап посвящен определению целей проекта и требований к результату с точки зрения бизнеса. Далее необходимо сформулировать их на языке аналитики данных, а также разработать предварительный план проекта.

Понимание данных. Этап начинается с первоначального сбора данных, визуализации и разведочного анализа, выявления проблем с качеством данных. Цель этапа – понять структуру данных, обнаружить интересные подмножества для формирования и последующей проверки гипотез.

Подготовка данных. Этап ставит целью получить итоговый набор данных, которые будут использоваться при моделировании, из исходных первичных источников.

Процедуры подготовки данных могут выполняться много раз. Они включают в себя:

- отбор таблиц, записей и атрибутов;

- очистку данных;
- получение производных данных;
- объединение данных;
- перевод данных в нужный формат.

Моделирование. На этом этапе идет выбор методов и алгоритмов моделирования, строятся модели, а их параметры настраиваются на оптимальные значения. Как правило, для решения любой задачи анализа данных существует несколько подходов. Некоторые подходы накладывают особые требования на представление данных. Таким образом, часто бывает нужен возврат на шаг назад к фазе подготовки данных.

Оценка. На этом этапе проекта уже построена модель и получены количественные оценки ее качества. Перед тем, как внедрять эту модель, необходимо убедиться, что основная бизнес-цель проекта достигнута. Возможно, придется какие-то вопросы рассмотреть более детально. В конце этапа принимается решение по использованию результатов анализа данных на практике.

Развертывание. В зависимости от требований этот этап может быть *простым*, например, составление финального отчета, или *сложным*, например, встраивание модели в бизнес-процесс. Обычно развертывание – это забота клиента. Но важно дать понять клиенту, что ему нужно сделать для того, чтобы начать использовать полученные модели:

- запланировать развертывание;
- запланировать поддержку проекта;
- подготовить документацию;
- провести аудит проекта.

Основными преимуществами методологии CRISP-DM являются:

- Модель процесса CRISP-DM универсальна и подходит для внедрения проектов по аналитике в любых отраслях.
- Нет привязки к конкретным программных продуктам или инструментам.
- Методология близка по духу к технологии извлечения знаний из баз данных – *Knowledge Discovery in Databases (KDD)*.

1.3. Структурированные данные

Данные, описывающие реальные объекты, процессы и явления, могут быть представлены в различных формах, измерены в различных шкалах и иметь определенный тип и вид.

По степени структурированности выделяют следующие формы представления данных:

- *Структурированные данные*. Структурированными называются данные, определенным образом упорядоченные и организованные с целью обеспечения возможности применения к ним некоторых действий (например, визуального или компьютерного анализа). Это основная форма представления сведений в базах данных. Структурированные данные отражают отдельные факты предметной области.
- *Неструктурированные данные.* К неструктурированным относятся данные, произвольные по форме, включающие тексты и графику, мультимедиа (видео, речь, аудио). Эта форма представления данных широко используется, например, в Интернете, а сами данные предоставляются пользователю в виде отклика поисковыми системами. Неструктурированные данные непригодны для обработки напрямую методами аналитики данных и подвергаются специальным приемам структуризации. Например, в анализе текстов при структурировании из исходного текста может быть сформирована таблица с частотами встречаемости слов, и уже такой набор данных будет обрабатываться методами, пригодными для структурированных данных.
- *Слабоструктурированные данные* – это данные, для которых определены некоторые правила и форматы, но в самом общем виде. Например, строка с адресом, строка в прайс-листе, ФИО и др. В отличие от неструктурированных, такие данные с меньшими усилиями преобразуются к структурированной форме, однако без процедуры преобразования они тоже непригодны для анализа. На рисунке 1.10 приведен пример стандартизации строки с адресом.

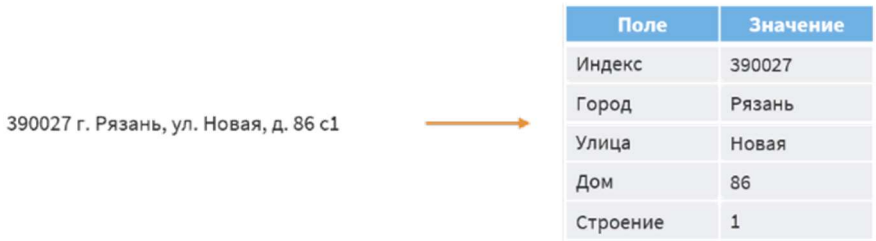


Рисунок 1.10 – Пример структурированных данных

Организация того или иного вида хранения данных (структурированных или неструктурированных) связана с обеспечением доступа к ним. Под *доступом* понимается возможность выделения отдельного элемента данных (или множества элементов) среди других по каким-либо признакам с целью выполнения некоторых действий над ним.

Одной из самых распространенных моделей хранения структурированных данных является **таблица**. В ней все данные упорядочиваются в двумерную структуру, состоящую из столбцов и строк.

В ячейках таблицы содержатся элементы данных: символы, числа, логические значения. Подавляющее большинство методов аналитики данных работает только со структурированными данными, представленными в табличном виде.

Поля структурированных данных принято делить на четыре типа:

I. **Числовой**, который бывает целым (количество товара, код товара и др.) и вещественным (цена, скидка и др.). ***Числовые переменные*** определяют числовые величины, измеряемые на некоторой интервальной шкале (относительной шкале) или шкале отношений (абсолютной шкале). Такие величины, измеренные на интервальной шкале, могут сравниваться, упорядочиваться, складываться, вычитаться. С ними можно выполнять все обычные операции над числами, такие, как вычисление среднего и оценку изменчивости. По характеру варьирования переменные делятся на дискретные и непрерывные.

Дискретные данные являются значениями признака, общее число которых конечно или бесконечно, но может быть подсчитано при помощи натуральных чисел от одного до бесконечности. С дискретными данными не могут быть произведены никакие арифметические действия, либо они не имеют смысла.

Дискретными данными являются все данные строкового и логического типа. Дискретными могут быть и числовые данные. Например, показатель *«Код товара»*, принимающий значение целого типа, является дискретным, так как операции сложения, вычитания, умножения над этим показателем не имеют смысла.

Непрерывные данные – это данные, которые могут принимать любые значения в некотором интервале. Над непрерывными данными можно производить арифметические операции: сложение, вычитание, умножение и деление.

Примерами непрерывных данных являются: возраст, рост, любые стоимостные показатели, количественные оценки (ВВП страны, количество товара, объем отгрузки, вес отгрузки, прибыль и так далее).

II. **Символьный** или **строковый** (фамилия, наименование, адрес, пол, образование, и др.).

Данные, представляющие собой значения ***категорийных (символьных) переменных***, измеряются по номинальной, либо по порядковой шкале (рисунок 1.11). Номинальные переменные могут принимать значения,

измеренные на шкале наименований, состоящей из наименований категорий, которые никак естественным образом не упорядочиваются. Никаких соотношений, кроме равенства или неравенства, между такими значениями нет. Эти данные не могут упорядочиваться, с ними не могут быть произведены никакие арифметические действия. Как правило, по умолчанию алгоритмы аналитики данных считают, что категорийные (символьные) данные будут обрабатываться как номинальные.

Переменная	Категории
Наличие машины	Да Нет
Кредитная история	Положительная Отрицательная Нет данных
Провайдер	Мегафон МТС Билайн

Рисунок 1.11 – Пример категорийных (символьных) данных

Переменные, измеренные на шкале порядка (их еще называют ***ординальными переменными***), имеют упорядоченные категории. К таким переменным можно отнести различные балльные или экспертные оценки с очевидным упорядочением значений. Балльная оценка успеваемости (неудовлетворительно, удовлетворительно, хорошо, отлично) является типичным примером порядковой величины. Измерения в ординальной шкале содержат информацию только о порядке следования величин, но не позволяют количественно выразить, насколько или во сколько раз одно значение больше или меньше другого. Для таких данных применимы только операции сравнения и ранжирования: равно, не равно, больше, меньше; арифметические действия не могут быть произведены.

Переменная	Упорядоченные категории
Образование	Среднее Среднее специальное Высшее Ученая степень
Оценка продукции	Плохо Удовлетворительно Хорошо Очень хорошо
Дисконтная карта	Бронзовая Серебряная Золотая

Рисунок 1.12 – Пример ординальных данных

III. **Логический** (Да/Нет, Ложь/Истина, 0/1).

IV. **Дата/Время**.

Показатели данных измеряются по определенной шкале. Выделяют четыре основных типа измерительных шкал:

I. шкала наименований;

II. шкала порядка;

III. интервальная шкала;

IV. шкала отношений.

По отношению к аналитической задаче наборы данных могут быть упорядоченными и неупорядоченными.

В ***упорядоченном наборе*** данных каждому столбцу соответствует один признак, а в каждую строку заносятся упорядоченные по какому-либо признаку события с интервалом периода между строками. Часто таким признаком выступает время. На рисунке 1.13 приведены примеры упорядоченных наборов данных – временной ряд (слева, упорядочен по дате) и ряд показаний датчика зонда (справа, упорядочен по глубине скважины).

Дата	Количество	Сумма
01.02.2017	4	283,31
01.02.2017	1	72,48
01.02.2017	1	173,32
02.02.2017	6	294,84
02.02.2017	2	405,76
02.02.2017	12	303,13
02.02.2017	1	210,50
03.02.2017	6	512,16
03.02.2017	3	156,96

Глубина	BK	DS
887,9	8,85	0,218
888,1	9,627	0,216
888,3	14,584	0,217
888,5	21,647	0,215
888,7	17,172	0,216
888,9	6,118	0,215
889,1	2,886	0,217
889,3	2,506	0,219

Рисунок 1.13 – Пример упорядоченных данных

В ***неупорядоченном наборе*** каждому столбцу соответствует признак, а в каждую строку заносится пример (ситуация, прецедент), соответственно, упорядоченность строк не требуется. Пример такого набора данных приведен на рисунке 1.14.

Номер	Банк	Город	Филиалы	Собственные активы
2	Внешторгбанк	Москва	32	23236327
3	Газпромбанк	Москва	27	9255041
4	Альфа-Банк	Москва	17	12446938
5	ОАО «ПСБ»	Санкт-Петербург	44	1275859
6	Банк Москвы	Москва	34	3335734
7	АКБ «ДИБ»	Москва	0	2616993

Рисунок 1.14 – Пример неупорядоченных данных

Особо выделяют транзакционные данные. Под ***транзакцией*** подразумеваются несколько объектов или действий, являющихся логически связанной единицей (рисунок 1.15). Этот способ представления используется алгоритмами анализа покупок (чеков) в супермаркетах.

Рисунок 1.15 – Пример транзакционных данных

Аналитика данных базируется на различных алгоритмах извлечения закономерностей из данных, результатом работы которых являются модели и знания. Таких алгоритмов довольно много, но они не способны гарантировать качественное решение. Никакой метод сам по себе не даст хорошего результата, так как критически важным является качество исходных данных. Чаще всего именно оно становится причиной неудачи проектов по аналитике данных. Несмотря на то что существуют специальные методы аудита и повышения качества данных, понимание и соблюдение принципов сбора и подготовки данных значительно облегчит построение моделей и позволит получить хорошие результаты.

Среди аналитиков и руководителей проекта распространено мнение, подкрепленное практикой, что до 80% процесса анализа данных – это время, потраченное на их подготовку.

Данные, которые накапливают организации в своих информационных источниках (так называемые ***бизнес-данные***), имеют свои особенности:

1. *Бизнес-данные редко накапливаются специально для решения задач аналитики.* Организации собирают данные для коммерческих целей: ведения учета, проведения финансового анализа, составления отчетности, принятия решений и др. Этим бизнес-данные отличаются от экспериментальных данных, которые собираются для исследовательских целей. Основными потребителями бизнес-данных обычно являются лица, принимающие решения.

2. *Бизнес-данные, как правило, содержат выбросы, ошибки, противоречия и пропуски.* Это следствие того, что организации не собирают данные с целью их анализа. В них появляются ошибки различной природы, что снижает качество данных.

3. *С точки зрения анализа объемы хранимых данных очень велики*. Современные базы данных содержат гигабайты и терабайты информации. Для ресурсоемких алгоритмов аналитики данных таблицу объемом 100 тыс. записей можно считать большой, поэтому при построении моделей это нужно

учитывать, например, использовать масштабируемые алгоритмы, способные работать на больших наборах данных.

Отмеченные особенности бизнес-данных влияют как на сам процесс анализа, так и на подготовку и систематизацию данных.

При сборе данных следует придерживаться следующих принципов:

1. *Абстрагироваться от существующих информационных систем и имеющихся в наличии данных*. Большие объемы накопленных данных совершенно не говорят о том, что их достаточно для анализа в конкретной организации. Необходимо отталкиваться от задачи и подбирать данные для ее решения, а не брать имеющуюся информацию. К примеру, при построении моделей прогноза продаж опрос экспертов показал, что на спрос очень влияет цветовая характеристика товара. Анализ имеющихся данных продемонстрировал, что информация о цвете товарной позиции отсутствует в учетной системе. Значит, нужно каким-то образом добавить эти данные, иначе не стоит рассчитывать на хороший результат использования моделей.

2. *Описать все факторы, потенциально влияющие на анализируемый процесс/объект.* Основным инструментом здесь становится опрос экспертов и людей, непосредственно владеющих проблемной ситуацией. Необходимо максимально использовать знания экспертов о предметной области и, полагаясь на здравый смысл, постараться собрать и систематизировать максимум возможных предположений и гипотез.

3. *Экспертно оценить значимость каждого фактора*. Эта оценка не является окончательной, она будет отправной точкой. В процессе анализа вполне может выясниться, что фактор, который эксперты посчитали очень важным, таковым не является, и наоборот, незначимый, с их точки зрения, фактор может оказывать значительное влияние на результат.

4. *Определить способ представления информации*. Будет ли это число, дата, да/нет, категория (то есть тип данных). Определить способ представления, то есть формализовать некоторые данные просто. Например, объем продаж в рублях – это определенное число. Но довольно часто бывает непонятно, как представить фактор. Чаще всего такие проблемы возникают с качественными характеристиками. Например, на объемы продаж влияет качество товара. Качество – сложное понятие, но если этот показатель действительно важен, то нужно придумать способ его измерения. Скажем, качество можно определять по количеству брака на тысячу единиц продукции либо оценивать экспертно, разбив на несколько категорий: отлично / хорошо / удовлетворительно / плохо.

5. *Собрать легко доступные факторы.* Они содержатся в первую очередь в источниках структурированной информации: учетных системах, базах данных и др.

6. *Собрать наиболее значимые факторы*. Такие факторы определяются экспертами. Вполне возможно, что без них не удастся достичь высоких результатов.

7. *Оценить сложность и стоимость сбора средних и наименее важных по значимости факторов*. Некоторые данные легкодоступны, их можно извлечь из существующих информационных систем. Но есть информация, которую непросто собрать, например, сведения о конкурентах, поэтому необходимо оценить, во что обойдется сбор данных. Сбор данных не является самоцелью. Если информацию получить легко, то, естественно, ее нужно собрать. Если сложно, то необходимо соизмерить затраты на ее сбор и систематизацию с ожидаемыми результатами.

К методам сбора необходимых для анализа данных являются:

1. *Получение из косвенных источников информации*. О многих показателях можно судить по косвенным признакам, и этим нужно воспользоваться. Например, можно оценить реальное финансовое положение жителей определенного региона следующим образом. В большинстве случаев имеется несколько товаров, предназначенных для выполнения одной и той же функции, но отличающихся по цене: товары для бедных, средних и богатых. Если получить отчет о продажах товара в интересующем регионе и проанализировать пропорции, в которых продаются товары для бедных, средних и богатых, то можно предположить, что чем больше доля дорогих изделий из одной товарной группы, тем более состоятельны в среднем жители данного региона.

2. *Использование открытых источников.* Большое количество данных присутствует в таких открытых источниках, как статистические сборники, отчеты корпораций, опубликованные результаты маркетинговых исследований, социальные сети и прочее.

3. *Приобретение данных у специализированных организаций*. На рынке работает множество организаций, которые профессионально занимаются сбором данных и предоставлением результатов клиентам для последующего анализа. Собираемая информация обычно предоставляется в виде различных таблиц и сводок, которые с успехом можно применять при анализе. Стоимость получения подобной информации чаще всего относительно невысока.

4. *Проведение собственных мероприятий по сбору данных*. Этот вариант сбора данных может быть достаточно дорогостоящим, но в любом случае он существует.

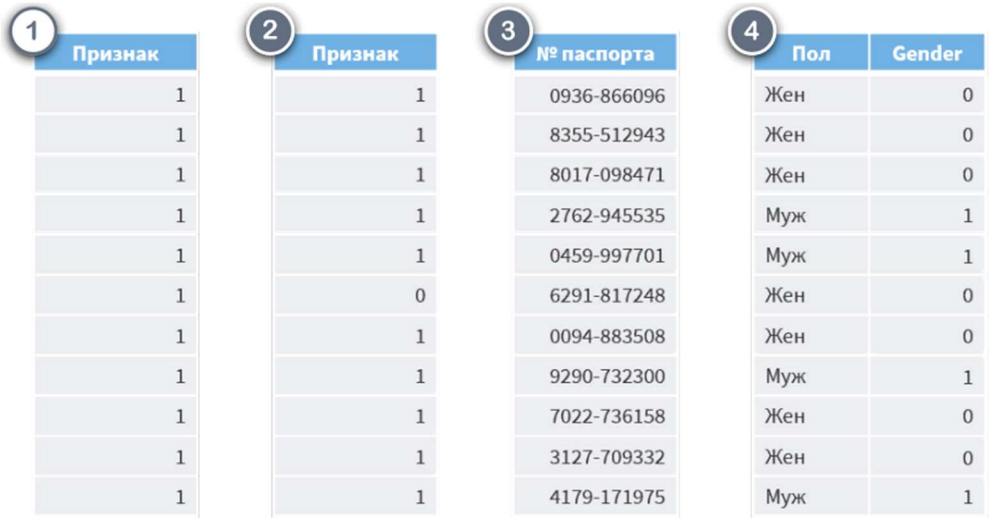
5. *Ввод данных вручную*. Данные вводятся по различного рода экспертным оценкам сотрудниками организации. Такой метод является наиболее трудоемким.

Методы сбора информации существенно отличаются по стоимости и необходимому времени, поэтому следует соизмерять затраты с результатами. Возможно, от сбора некоторых данных придется отказаться, но факторы, которые эксперты оценили как наиболее значимые, нужно собрать обязательно несмотря на стоимость этих работ, либо вообще не проводить анализ.

Данные должны быть собраны в таблицы базы данных, в текстовые файлы с разделителями, в файлы MS Excel, то есть должны быть представлены в структурированном виде. Кроме того, необходимо унифицировать представление данных: один и тот же объект должен везде описываться одинаково.

Одной из распространенных ошибок при сборе данных из структурированных источников является стремление взять для анализа как можно больше признаков, описывающих объекты. Между тем предварительная оценка данных, которая проводится при помощи разведочного анализа данных, существенно помогает в определении информативности признаков. Среди неинформативных признаков можно выделить четыре типа (рисунок 1.16).

1. Признаки, содержащие только одно значение.
2. Признаки, содержащие в основном только одно значение.
3. Признаки с уникальными значениями.
4. Признаки, между которыми имеет место причинно-следственная связь, – в этом случае для анализа можно взять только один из них.



1

Признак
1
1
1
1
1
1
1
1
1
1
1

2

Признак
1
1
1
1
1
0
1
1
1
1
1

3

№ паспорта
0936-866096
8355-512943
8017-098471
2762-945535
0459-997701
6291-817248
0094-883508
9290-732300
7022-736158
3127-709332
4179-171975

4

Пол	Gender
Жен	0
Жен	0
Жен	0
Муж	1
Муж	1
Жен	0
Жен	0
Муж	1
Жен	0
Жен	0
Муж	1

Рисунок 1.16 – Примеры неинформативных признаков

Существуют определенные начальные требования к минимальным объемам данных для моделирования. В зависимости от представления данных и решаемой задачи эти требования различны.

Для *временных рядов*, которые относятся к упорядоченным данным, требования следующие:

- Если для бизнес-процесса (например, продажи) характерна сезонность/цикличность, то необходимо иметь данные хотя бы за один полный сезон/цикл с возможностью варьирования интервалов (понедельное, помесячное и так далее).
- Максимальный горизонт прогнозирования зависит от объема данных: данные за 1,5 года – а прогноз возможен максимум на 1 месяц; данные за 2-3 года – на 2 месяца.

Для *неупорядоченных данных* требования следующие:

- Количество примеров (прецедентов) должно быть минимум на один порядок больше количества факторов (столбцов).
- Желательно, чтобы данные покрывали как можно больше ситуаций реального процесса.

Анализ *транзакций* целесообразно производить на большом объеме данных, иначе могут быть выявлены статистически необоснованные шаблоны поведения. Алгоритмы поиска таких шаблонов способны быстро перерабатывать огромные массивы данных. Примерное соотношение между количеством объектов и объемом данных следующее:

- 300-500 объектов – от 10 тыс. транзакций;
- 500-1000 объектов – более 300 тыс. транзакций.

1.4. Технологии аналитики данных

Концепция анализа данных, предложенная Дж. Тьюки, стала только отправной точкой в развитии аналитики данных. Появление персональных компьютеров и доступность специализированного программного обеспечения для визуализации и статистического анализа, безусловно, открыло массу новых возможностей широкому кругу исследователей, что потребовало пересмотреть существующее подходы к анализу данных. Причинами пересмотра существующих подходов стали:

1. Значительные объемы данных. К концу 80-х годов объем накопленной информации удваивался каждые 20 месяцев.
2. Требование принятия решений в режиме реального времени или близком к реальному.
3. Разнородный характер данных, измеренных в различных шкалах, необходимость их обработки разнотипными средствами и сведение результата в некоторой единой инструментальной среде.
4. Осознание потребности в создании средств автоматического выделения и анализа скрытых зависимостей.
5. Сложный характер объекта управления (взаимодействие большого множества разнородных процессов и подсистем), ограничивающий использование как традиционных моделей и методов, так и экспертных систем.

Перед исследователями встают новые задачи, характеризующиеся следующими особенностями:

- Объект исследования характеризуется большими объемами данных, требуется анализ в ограниченное время.
- Гарантий того, что данные хорошего качества, нет, требуется проводить их аудит, очистку и обогащение.
- Формальная модель объекта отсутствует (нет полного и непротиворечивого аналитического, описания).
- Необходимо уметь выделять параметры, определяющие поведение объекта исследования в тех или иных ситуациях.
- Необходимо уметь обобщать имеющуюся информацию, выделяя неявно представленные зависимости (то есть те эмпирические правила, которые позволяют предсказывать поведение модели объекта в новых обстоятельствах).

Ответом на эти вызовы стало появление технологий хранилищ и витрин данных (англ.: Data Warehousing), аналитической отчетности, извлечения

знаний из баз данных и майнинга данных (англ.: Knowledge Discovery in Databases и Data Maning).

Понимание того, что аналитика данных требует качественной визуализации, чтобы обнаружить закономерности в данных и выдвинуть гипотезы, а сделанные выводы и рекомендации необходимо презентовать заинтересованным лицам, привело к бурному развитию и росту инструментов визуализации.

Распространение получили ***дашборды*** или ***аналитические панели***: ключевые показатели эффективности, тренды, зависимости и другие метрики в понятном компактном виде, расположенные рядом друг с другом. Кроме наглядной визуализации данных, основные цели дашбордов связаны с отслеживанием того или иного показателя во времени или оценкой относительно других показателей.

Стоит отметить, что персональные компьютеры и информационные системы оказали значительное влияние на развитие методов анализа данных. Например, в 90-е годы получили широкое распространение такие понятия, как *обнаружение знаний в базах данных* (англ.: Knowledge Discovery in Databases) и *интеллектуальный анализ данных* (англ.: Data Maning). Сегодня на базе этих технологий и связанных с ними программных продуктов создается большинство прикладных аналитических решений.

Рассмотрим подробнее эти технологии и связанные с ними термины.

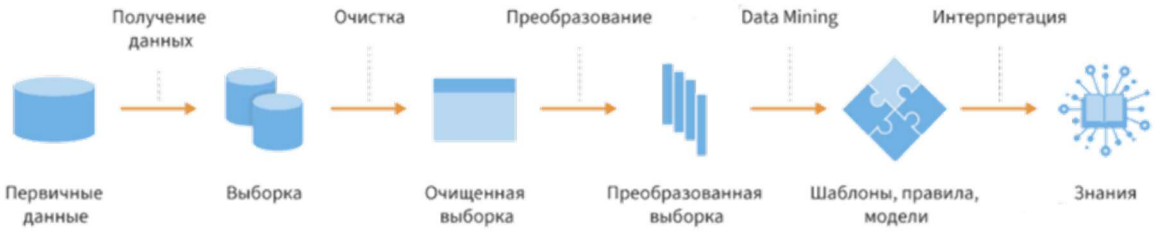


Рисунок 1.17 – Схема обнаружения знаний в базах данных

Knowledge Discovery in Databases (KDD). Несмотря на разнообразие задач аналитики данных, почти все задачи, в которых требуется моделирование, могут решаться по единой схеме. Эта схема, зародившаяся в 1989 г. и актуальная до сих пор, получила название ***Knowledge Discovery in Databases (KDD)*** – обнаружение знаний в базах данных. Она описывает не конкретный алгоритм или математический аппарат, а последовательность действий, которую необходимо выполнить для обнаружения знания в данных.

Последовательность действий не зависит от предметной области; это набор атомарных операций, комбинируя которые, можно получить наилучший

возможный результат. KDD включает в себя этапы подготовки данных, выбора информативных признаков, очистки, построения моделей, постобработки и интерпретации полученных результатов. «Ядром» этого процесса являются методы Data Maning.

Обнаружение знаний в базах данных – это получение из данных знаний в виде зависимостей, правил, моделей. По задумке авторов термина, KDD рассматривался как часть более общего подхода Knowledge Extraction, который изучает извлечение знаний не только из структурированных данных, но и из слабоструктурированных и неструктурированных.

Рассмотрим каждый этап, выполняемый в рамках KDD подробнее:

1. *Получение данных*. Первым шагом в процессе обнаружения знаний является формирование выборки с данными. Здесь требуется активное участие экспертов для выдвижения гипотез и отбора признаков, влияющих на анализируемый процесс. Желательно, чтобы данные были уже собраны и интегрированы. Крайне необходимы удобные механизмы подготовки выборок: запросы, фильтры и др. Чаще всего в качестве источника рекомендуется использовать специализированное хранилище, витрину данных или оперативный склад, в которых консолидируются бизнес-данные.
2. *Очистка данных.* Реальные данные для анализа редко бывают хорошего качества. Необходимость в предварительной обработке данных возникает независимо от того, какие технологии и алгоритмы используются. Более того, эта задача может представлять самостоятельную ценность. К задачам очистки данных относятся: заполнение пропусков, подавление выбросов, сглаживание, исключение дубликатов и противоречий и прочее.
3. *Преобразование данных*. Этот шаг необходим для тех методов, при использовании которых исходные данные должны быть представлены в каком-то определенном виде. Дело в том, что различные алгоритмы анализа требуют специальным образом подготовленные выборки. Например, для прогнозирования необходимо преобразовать временной ряд при помощи скользящего окна или вычислить агрегаты. К задачам преобразования данных относятся: приведение типов, выделение временных интервалов, квантование, сортировка, группировка, расчет производных столбцов и прочие.
4. *Data Maning*. На этом этапе строятся модели. Нередко процесс обнаружения знаний KDD отождествляют с Data Maning. Однако правильнее считать Data Maning шагом процесса KDD.
5. *Интерпретация.* В случае, когда извлеченные зависимости и шаблоны непрозрачны для пользователя, должны существовать методы постобработки, позволяющие привести их к интерпретируемому виду. Для оценки качества полученной модели нужно использовать как формальные

методы, так и знания аналитика. Именно аналитик может сказать, насколько применима полученная модель к реальным данным. Построенные модели являются, по сути, формализованными знаниями, полученными из фактографических данных, а следовательно, в таком виде знания можно распространять (тиражировать).

Data Maning. Считается, что термин Data Maning был введен Г. Пятецким-Шапиро в 1989 г. и оригинальное определение звучит так: Data Maning – это обнаружение в «сырых» данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности.

Английское словосочетание Data Maning не получило устоявшегося перевода на русский язык. В литературе используются следующие варианты перевода: раскопка данных, добыча полезных данных (сродни добыче полезных ископаемых), интеллектуальный анализ данных, глубинный анализ данных, просев информации. Некоторые специалисты считают неудачными большинство вариантов перевода (добыча данных – разве добывают данные, а не знания?; интеллектуальный анализ – а что тогда «неинтеллектуальный» анализ?) и оперируют прямым англоязычным термином.

В Data Maning принято считать, что найденное знание должно обладать следующими свойствами:

- Знание отражает результат исследования системы (познание объективной реальности).
- Знание выражено определенным, понятным человеку образом (использует общепринятые символы, понятия, естественные знаки).
- Знание компактно (по форме, описанию), что делает его доступным для понимания, интерпретации и дальнейшего использования.

Закономерности в виде логических правил ***если-то*** представляют собой хороший пример выполнимости второго свойства: они легко интерпретируются человеком и удобны для дальнейшей компьютерной обработки. Но так бывает не всегда. Например, весовые коэффициенты нейронной сети понятны машине, но не аналитику. В этом случае ему через косвенные признаки придется удовлетвориться тем фактом, что эти модели объективно правильны: переменные действительно информативны, объекты распознаются правильно, прогнозы сбываются.

Чтобы сделать сложные результаты прозрачными, инструменты Data Maning используют широкий набор вспомогательных средств в виде специализированных визуализаторов и метрик качества моделей.

Data Maning имеет дело с закономерностями нестатистического характера ***(паттернами)***. Они позволяют делать заключения не в среднем по некоторому множеству объектов, а для каждого изучаемого объекта в отдельности. Если удастся такие паттерны обнаружить, понять возможную причину возникновения паттерна и найти применение этому знанию – можно говорить о том, что Data Maning работает.

Data Maning – это не один метод, а совокупность большого числа различных методов обнаружения паттернов. Для них существует несколько условных классификаций. Мы будем говорить о *двух* типах моделей и *пяти* базовых классах задач (рисунок 1.18).

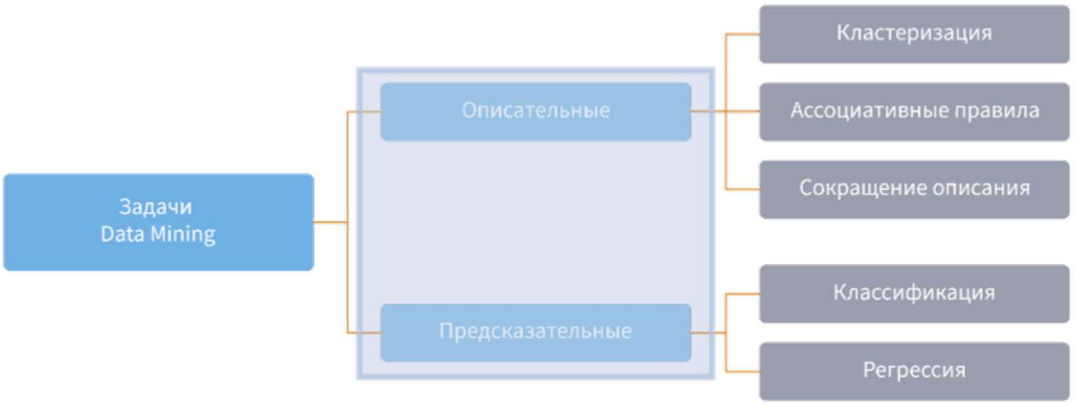


Рисунок 1.18 – Классификация методов Data Maning

В современной аналитике данных принято выделять два класса моделей Data Maning: ***описательные (дескриптивные)***, которые необходимы для лучшего понимания исследуемой системы, выявления структуры и взаимосвязей в данных, и ***предсказательные***, необходимые для понимания новых сведений о системе, прогнозирования событий или значений на основе набора входных переменных.

Описательная аналитика ближе к сложной визуализации и разведочному анализу данных в том плане, что результат моделирования – компактное описание множества объектов в виде кластеров, правил, шаблонов поведения, групп. Описательные модели пытаются ответить на такие вопросы, как:

- Какова структура клиентской базы?
- Какой профиль идеального клиента?
- Какие есть взаимосвязи между характеристиками клиентов?

Основным недостатком описательных моделей является их относительная простота, не позволяющая эффективно решать задачи предсказания новых состояний объектов. Это делает *предсказательное*

моделирование, и оно предполагает использование более сложных математических алгоритмов, повышенных требований к данным и квалификации аналитика. Предсказательные модели отвечают на следующие вопросы:

- Откликнется ли клиент на данную маркетинговую кампанию?
- Какой размер прибыли будет в следующем месяце?
- Какие из потенциальных клиентов вероятно совершат приобретение услуги в следующем месяце?
- Какой прогнозируемый спрос на товар на следующий период планирования? и так далее.

Далее рассмотрим задачи Data Maning:

I. **Ассоциативные правила.** ***Поиск ассоциативных правил*** (англ: association rules) – задача выявления зависимостей между связанными событиями, указывающих, что из события X следует событие Y. Такие зависимости и называются *ассоциативными правилами*. Впервые эта задача была предложена для нахождения типичных шаблонов покупок, совершаемых в супермаркетах, поэтому иногда ее еще называют *анализом потребительской корзины* (англ.: market basket analysis).

Цели такого анализа:

- оптимизировать размещение товаров в торговом зале;
- сформировать персональные рекомендации;
- спланировать рекламные кампании (промо-акции);
- подобрать товары для кросс-продаж (cross-sales) и дополнительных продаж (up-sales).

Если события можно упорядочить по времени наступления, то говорят о последовательных шаблонах – ассоциативных правилах, в которых важен порядок следования событий.

II. **Кластеризация** (англ: clustering) – это группировка объектов (наблюдений, событий) на основе данных (свойств), описывающих сущность объектов. Объекты внутри кластера должны быть «похожими» друг на друга и отличаться от объектов, вошедших в другие кластеры. Чем больше «похожи» объекты внутри кластера и чем больше отличий между кластерами, тем точнее кластеризация.

Результатом кластеризации являются группы похожих объектов – кластеры или сегменты. Но кластеризация указывает только на схожесть объектов, и не более того. Для объяснения образовавшихся кластеров необходима их интерпретация. Она производится с использованием статистических показателей, так называемых профилей кластеров.

Целью кластеризации является желание оперировать не каждым объектом в отдельности, а их подгруппами.

Так, при достаточно большом количестве клиентов становится трудно подходить к каждому индивидуально. После кластеризации можно узнать, какие сегменты наиболее активны, какие приносят наибольшую прибыль, выделить характерные для них признаки и повысить эффективность работы с клиентами благодаря учету их персональных предпочтений.

III. **Задача сокращения описания** (англ.: summarization task) – это обобщение данных. Набор данных «суммируется», в результате чего получается меньший набор, который дает агрегированную информацию о данных. Например, покупки, сделанные клиентом, можно суммировать в виде общих продуктов, общих расходов, использованных предложений и так далее. Такая обобщенная информация высокого уровня может быть полезна для отдела продаж или работы с клиентами для подробного анализа поведения клиентов и покупателя. Данные можно суммировать на разных уровнях абстракции и под разными углами.

Задача сокращения описания стала актуальной в связи с развитием Интернета и резким увеличением объема информационных ресурсов. Там говорят о задаче *автоматического реферирования документа* (англ: automatic summarization task). Суть ее в том, чтобы сократить текстовый документ до краткого резюме, в котором сохранены наиболее важные моменты исходного документа. В частности, такой задачей является автоматическое составление аннотаций статей для выдачи их в результатах поисковых систем.

Сегодня сфера применения задачи реферирования расширилась и охватывает все основные типы информации (изображения, звук, видео). Среди ее приложений – тематический поиск контента, контекстный поиск, идентификация материалов, нарушающих интересы правообладателей и законодательные ограничения и так далее.

IV. **Задача классификации**. Предсказательное моделирование разбивается на два этапа. На первом этапе на основе набора данных с известными результатами строится модель. На втором этапе она используется для предсказания событий на новых наборах данных. К предсказательному моделированию относят задачи *классификации* и *регрессии*.

Классификация – отнесение входного объекта (события, наблюдения) к одному из заранее известных классов. По количеству классов, на которые разбивается входная выборка, следует различать бинарную и полиномиальную классификации. В первом случае в выборке выделяются только два класса (точнее, один класс и все остальное). Бинарная классификация покрывает большое количество бизнес-задач, связанных с

выбором из двух альтернатив, одна из которых интерпретируется как событие, а другая – как не-событие. Примеры таких задач:

- заемщик банка: допустит просрочку и не допустит просрочку;
- клиент телекоммуникационной компании: уйдет в течение года и не уйдет в течение года;
- фильтр электронной почты: спам, не спам.

При полиномиальной классификации выборка размечена на три и более класса. Примерами таких задач являются: распознавание образов, определение класса заемщика, и тому подобное. Несмотря на сходство формулировок задач бинарной и полиномиальной классификации, методы и инструменты их решения сильно различаются.

V. **Задача регрессии** – это установление зависимости непрерывной выходной переменной от входных переменных. К задаче регрессии сводится, в частности, прогнозирование временного ряда на основе исторических данных. Примеры бизнес-постановок для задач регрессии:

- оценка стоимости недвижимости;
- прогнозирование объема продаж;
- прогноз цен на оптовом рынке электроэнергии;
- какие банкоматы какими суммами инкассировать.

Нельзя не упомянуть о важной задаче, которую сложно однозначно отнести однозначно к классификации или регрессии. Это задача вероятностного оценивания наступления события. В отличие от обычной задачи регрессии, здесь не производится предсказание значения числовой переменной исходя из выборки исходных значений. Вместо этого, значением функции является вероятность того, что данное исходное значение принадлежит к определенному классу. Эту вероятность можно использовать для принятия решений, например, вероятность вернуть долг умножить на сумму долга и получить ожидаемый денежный поток. А можно задать порог вероятности, при превышении которого считать, что событие наступит. Распространенные постановки задач: вероятность выхода на просроченную задолженность, вероятность повторной покупки, поломки оборудования и др.

Одно из важнейших назначений методов Data Maning состоит в наглядном представлении результатов вычислений для интерпретации, что позволяет использовать инструментарий Data Maning людьми, не имеющими специальной математической подготовки. В то же время применение статистических методов анализа данных требует хорошего владения теорией вероятностей и математической статистикой.

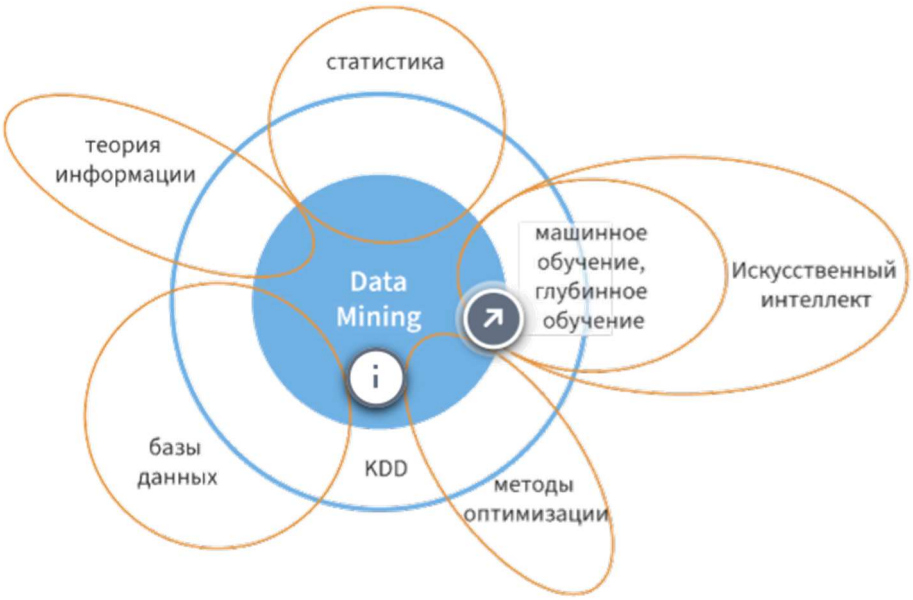


Рисунок 1.19 – Методы и алгоритмы Data Maning

Для решения вышеперечисленных задач используются различные методы и алгоритмы Data Maning. Методы Data Maning лежат на стыке информатики, баз данных, статистики, искусственного интеллекта, теории информации, алгоритмизации.

В общем случае непринципиально, каким именно алгоритмом будет решаться задача, главное – иметь метод решения для каждого класса задач. На сегодняшний день наибольшее распространение в Data Maning получили методы машинного обучения: деревья решений, случайные леса, нейронные сети, машины опорных векторов и другие.

Машинное обучение (англ.: Machine Learning) – область научного знания, объектом исследования которой являются методы построения алгоритмов, способных обучаться на данных, то есть конструировать из данных функцию (формулу), которая может быть использована для прогноза.

В задачах машинного обучения часто применяются методы теории вероятностей, линейной алгебры, статистики, оптимизации и многих других дисциплин. Но, в отличие от традиционных подходов прикладной математики, работа алгоритма машинного обучения далеко не всегда предусматривает объяснение происходящего, а поиск решения – итерационная процедура, оптимизирующая заданную функцию.

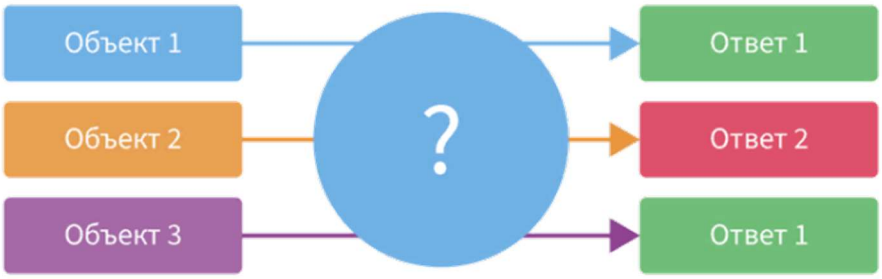


Рисунок 1.20 – Множество объектов и множество возможных ответов

Общая постановка задачи обучения следующая. Имеется множество объектов (ситуаций) и множество возможных ответов (откликов, реакций, событий). Между ответами и объектами существует некоторая зависимость, но она неизвестна. Известна только конечная совокупность прецедентов – пар вида **объект-ответ** и называемая ***обучающей выборкой***. На основе этих данных требуется обнаружить зависимость, то есть построить модель, способную для любого объекта выдать достаточно точный ответ. Чтобы измерить точность ответов, вводится критерий качества на независимом (тестовом) множестве.

1.5. Прикладные задачи аналитики данных

Аналитика данных может использоваться в любой организации для управления продажами, разработки маркетинговых стратегий, формирования корпоративной отчетности, повышения эффективности бизнеса. В данном списке задач акцент, конечно же, сделан на коммерческие организации. Однако это не означает, что аналитика данных не применяется в государственном управлении, медицине и промышленности. Сфер, где анализируются данные, очень много. Наша цель – изучить наиболее часто встречающиеся постановки задач аналитики данных в бизнесе. Мы подробно остановимся на трех отраслях:

- розничные банки и финансовые организации;
- ритейл, включая электронную коммерцию, e-grocery;
- телекоммуникации и провайдинг.

Возникает вопрос: для чего аналитику нужно хорошо знать постановки прикладных задач? Во-первых, определимся с термином ***прикладная задача***. Это таким образом сформулированная постановка задачи, смысл которой понимает менеджмент среднего и высшего звена. А проблемы руководство зачастую формулирует нечетко: «уменьшить отток клиентов», «увеличить выручку от партнерских программ», «удержать покупателей». Именно в такой формулировке задача попадает к аналитику данных, то есть этот человек

должен быть знаком с предметной областью, чтобы понимать то, что же на самом деле нужно; как модели Data Mining помогут повысить эффективность работы компании; какой экономический эффект они принесут при внедрении в эксплуатацию.

В противном случае между аналитиком и руководством возникнет проблема непонимания, что может «поставить крест» на аналитическом проекте. Руководитель, как правило, не мыслит в терминах моделей и обучающих выборок.

Задачи аналитики данных в банках и финансовых организациях. Задачи можно разделить на три группы:

1. *Традиционный BI*. Интеграция и визуализация данных, отчетность, аналитические панели.
2. *Моделирование*. Описательные и предсказательные модели Data Mining, машинное обучение.
3. *Принятие решений*. Процессы принятия и исполнения решений, основанных на простых правилах, так и на сложной логике с использованием моделей машинного обучения.

Формирование различных видов отчетности – важная задача для всех банков. В российских банках регулярно используется отчетность для ЦБ РФ, отчетность по МФСО (международный стандарт финансовой отчетности), отчетность по различным *кредитным рискам* и аналитические KPI – сколько клиентов обслужено, какие подразделения работали хуже, какие лучше и так далее.

Рисунок 1.21 – Схема комплексного формирования банковской отчетности

Один из способов для решения проблемы комплексного формирования банковской отчетности – это использование интегрированных средств, встроенных в инструменты бизнес-аналитики (рисунок 1.21). Способ оптимален в случаях, когда информационный ландшафт банка насыщен, присутствуют много учетных систем и проблемой является качество данных и так называемая «единая версия правды», на основе которой формируется

отчетность (второй способ – это использование специализированных модулей в составе ***автоматизированных банковских систем***, АБС).

Для формирования консолидированной отчетности применяются традиционные BI-инструменты, которые решают задачи построения централизованного хранилища и локальных витрин данных, аудита, очистки и предобработки данных, визуализации OLAP-кубами и современными графическими средствами.

Рисунок 1.22 – Пример кредитного скроллинга

Технологии кредитного скоринга занимают одно из центральных мест в управлении розничными рисками современного банка. Это метод оценки рисков и управления ими на основе прогноза, с какой вероятностью конкретный заемщик может просрочить платежи по кредиту. Считается, что чем выше скоринговый балл заемщика, тем ниже вероятность возникновения у него просроченной задолженности в будущем.

Существует несколько видов кредитного скоринга:

1. *Аппликационный скоринг*. В основе кредитного скоринга лежит предположение, что люди со схожими социальными показателями ведут себя одинаково. Если некоторым социально-экономическим характеристикам клиента (пол, возраст, длительность проживания в данной местности и так далее) присвоить определенные веса, то для каждого нового клиента можно, на основе его анкеты, рассчитать скоринговый балл. Он будет использован для принятия решения о выдаче кредита. Это так называемый аппликационный или анкетный скоринг.

Изменяя скоринговый балл, в аппликационном скоринге можно управлять процессами выдачи кредитов и качеством кредитного портфеля путем поиска компромисса между объемами выдач и рисковой составляющей (долей просроченных кредитов). Чем выше пороговый скоринговый балл, тем меньше уровень одобрений и одновременно «чище» входной поток клиентов

с точки зрения их платежной дисциплины (при условии, что качество скоринговой модели хорошее).

2. *Поведенческий скоринг*. От англ. behavioral scroing – оценка вероятности возврата уже выданных кредитов. Осуществляется в пределах кредитного периода с целью выявления риска дефолта и принятию мер по снижению этих рисков.

3. *Коллекторский скоринг*. От англ. collection scoring – оценка возможности полного или частичного возврата кредита заемщиком при нарушении им сроков погашения задолженности.

4. *Мошеннический скоринг*. От англ. fraud scoring – выявление и предотвращение мошеннических действий со стороны потенциальных и уже существующих клиентов. Широко используется в розничном банковском бизнесе при анализе карточных транзакций. Помимо традиционного способа борьбы с мошенниками – проверки по «черным» спискам, данный вид скоринга призван на основе моделей Data Minig выявлять лица, с которыми связан повышенный риск совершения противозаконных действий в отношении банка.

Результатом моделирования (на основе прошлых статистических данных о выданных кредитах и фактах просрочек, платежей, мошеннических действий) становится скоринговая модель, которая затем трансформируется в *скоринговую карту* (рисунок 1.23). Скоринговая модель дает *вероятностную оценку* (от 0 до 1) наступления интересующего события по каждому кредиту.

Характеристика	Атрибут	Балл
Возраст	до 25	63
	от 24 до 29	76
	от 29 до 40	79
	от 41 до 51	85
	свыше 52	48
Образование	Среднее	75
	Среднее специальное	81
	Высшее	93
Состоит в браке	Да	84
	Нет	70

→ 76 + 93 + 70 = 239

Рисунок 1.23 – Пример скоринговой карты

Скоринговая модель состоит из набора характеристик, которые принимают определенное множество уникальных значений, называемых ***атрибутами***.

Каждому атрибуту присваивается числовой балл (например, у характеристики Возраст значение от 24 до 29 является атрибутом), так, чтобы «плохим» клиентам давались низкие баллы, а «хорошим» – высокие.

Клиентам присваиваются баллы в соответствии с таблицей, и результат суммируется. Обычно *итоговый скоринговый балл* – это целое положительное число в диапазоне от 0 до 999.

Для построения скоринговых карт используются алгоритмы предсказательного моделирования Data Mining: логистическая регрессия, деревья классификационных правил, нейронные сети.

Для банков важным вопросом является оптимизация стоимости владения сетями самообслуживания. Одной из больших статей расходов являются *затраты на инкассацию*. Для оптимизации надо уметь моделировать процесс движения наличности в банкомате (англ.: Cash Management), что даст возможность прогнозировать дневной расход, необходимый для обеспечения банкоматных операций.

Для того чтобы производить оптимальное планирование инкассации, необходимо для каждого банкомата определить характерные для этого банкомата особенности: среднесуточный расход, количество дней с нулевым расходом и другие.

Аналитика данных в данном случае помогает проанализировать потоки наличности в банкоматах и спрогнозировать средства, оптимальные для инкассации в заданный период, для чего используются принципы логистики и управления запасами и модели временных рядов снятий наличности. После чего становится возможным автоматизировать процедуры регулярного планирования инкассаций по каждому банкомату с минимальной вовлеченностью персонала. Кроме того, решение задачи дает ряд важных преимуществ:

- *Повышение лояльности клиентов* за счет увеличения периода доступности денежных средств.
- *Минимизация простоев точки* за счет своевременного пополнения банкомата.
- *Рост операционной эффективности* за счет сокращения издержек на инкассацию и логистику.

Кредитные конвейеры (или *кредитные фабрики*) необходимы банкам и финансовым организациям, которые работают на рынке массовой выдачи кредитов для частных лиц. Автоматизация и унификация процесса в рамках работы кредитного конвейера позволяют банку снизить стоимость обслуживания, уменьшить операционные риски, увеличить скорость, а значит, и количество выданных кредитов.

По сути, кредитный конвейер – это автоматизированная процедура обработки заявки (рисунок 1.24).

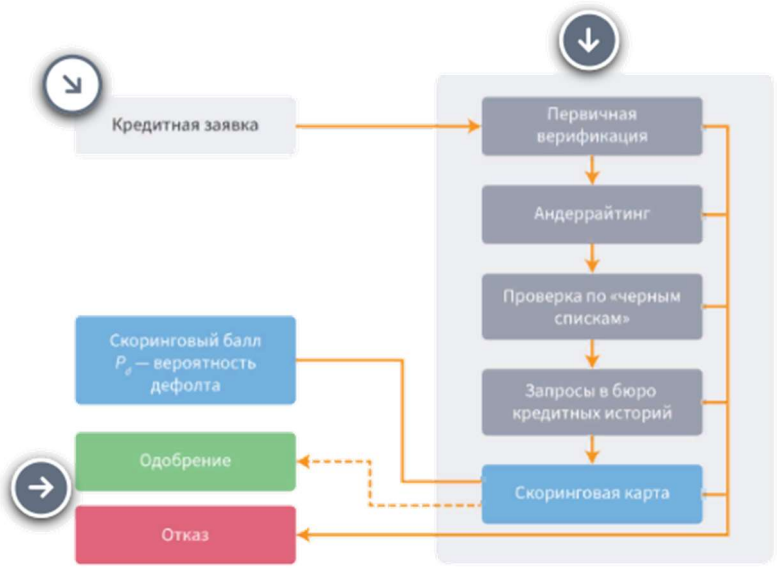


Рисунок 1.24 – Пример кредитного конвейера

Часто скоринговая оценка применяется вместе с другими этапами проверки клиента (последовательно или параллельно): соответствие набору минимальных требований к заемщику («стоп-факторы»), андеррайтинг (набор правил вида **если-то**), поиск заявителя в «черных» списках и запросы в сторонние сервисы информации (бюро кредитных историй, Национальный Хантер, Пенсионный фонд и другие).

Некоторые аналитические платформы кроме непосредственного функционала для моделирования (в данном случае построения скоринговых карт) имеют возможность реализации логики принятия решений. Это делает практически «бесшовной» интеграцию конвейера с аналитической составляющей бизнес-процесса принятия решений по клиенту, в частности, кредитным скорингом.

Далее рассмотрим сегментацию клиентом банка. Разбив клиентов на сегменты, банки могут предложить им персонализированные услуги, попытаться перевести в более прибыльный для банка сегмент.

Большим плюсом для банков является то, что каждый клиент всегда четко идентифицирован (в отличие от ритейла, где даже сам факт использования дисконтной карты не всегда гарантирует, что ее применял владелец).

Для целей сегментации используются алгоритмы Data Mining, решающие задачу кластеризации: сети Кохонена, k-means и другие. Они

позволяют в приемлемые сроки разбить миллионные клиентские базы на несколько сегментов.

Большие преимущества достигаются одновременным применением различных вариантов сегментации, полученных как традиционными маркетинговыми инструментами (например, использование стандартных потребительских профилей), так и методами Data Mining.

Задачи аналитики данных в ритейле. Задачи можно разделить на три группы:

1. *Традиционный BI*. Аналитическая отчетность.
2. *Моделирование*. Сегментация покупателей. Отчет ритейлера о каждой индивидуальной операции по продаже содержит информацию о времени, месте и сумме покупки, количестве и ассортименте приобретенных товаров. Эти данные позволяют получать довольно полное представление о поведении клиентов, постоянно бывающих в торговых точках.
3. *Принятие решений*. После сегментации каждый клиент относится к некоторому сегменту. Эта операция может производиться регулярно, что позволит розничным организациям оперативно фиксировать изменения в потребительском поведении и планировать маркетинговые кампании под каждый сегмент.

Подход со скоринговыми моделями, при помощи которых банки оценивают заемщиков, успешно применяется и в маркетинге. Маркетологи говорят о моделировании предрасположенности клиента совершить повторную покупку, купить товар определенного бренда, отказаться от ранее сделанного заказа, откликнуться на рекламное предложение, уйти в отток.

Предположим, что крупный ритейлер, занимающийся торговлей по каталогам, желает повысить продажи среди существующих клиентов. Компания не может сделать рассылку каталогов всем клиентам – расходы на нее могут превысить прибыль от возможных продаж. Требуется выделить группу клиентов, которые с наибольшей вероятностью сделают заказ, и им выслать новые каталоги.

Посредством анкет можно получить некоторую общую информацию о клиентах (пол, возраст, место работы и так далее) и информацию о том, что и как часто они покупают. Далее выбирается небольшая группа клиентов (обычно 1-2 тыс. человек), им делается пробная рассылка и фиксируется факт отклика. Теперь требуется качественная скоринговая модель, оценивающая вероятность отклика. Применив модель ко всей клиентской базе, мы сможем отобрать наиболее перспективных респондентов для большой рассылки.

Несмотря на то, что для моделирования предрасположенности используются аналогичные методики, что и для кредитного скоринга,

стоимость и скорость получения выборок для моделирования существенно ниже, чем в кредитовании – затраты на пробные рассылки на несколько порядков ниже, чем потери от неблагонадежных заемщиков (так как для сбора статистики в кредитном скоринге требуются не только «хорошие» заемщики, но и «плохие»). Однако в ритейле, если речь о крупных компаниях, аналитики данных могут строить десятки и сотни моделей предрасположенности в год, поэтому здесь очень важна автоматизация этого процесса и непрерывный контроль жизненного цикла моделей.

Следующее направление – поиск ***шаблонов покупок***, когда выявляются товары (услуги), которые клиенты стремятся приобрести в комплексе (или наоборот, только по отдельности), что важно для решения ряда задач (см. рисунок). Формирование рекомендаций особенно актуально для интернет-магазинов. Такой анализ может быть эффективно выполнен описательными моделями Data Mining, такими, как ассоциативные правила и последовательные шаблоны. Более сложные алгоритмы используют ***матричную факторизацию***. Для их генерации требуются только транзакции – когда, что и в каком количестве куплено.

Далее выявленные закономерности используются при разработке подсистемы подбора товаров, или, более широко, подсистемы персонализации (адаптацию продукта, услуги или контента для нужд пользователя, в зависимости от его особенностей, личных предпочтений или предварительной информации, которую он сообщил, называют ***персонализацией***). Можно сказать, что сегментация также решает задачу персонализации, но на более обобщенном уровне.

Результаты моделирования предрасположенности необходимо использовать для персональных коммуникаций с клиентами. Подобно кредитному конвейеру, логику принятия решений или ее часть можно выстроить в аналитической платформе.

Аналитика данных помогает определить критерии сегментации и провести отбор целевой аудитории (то есть сформировать список участников маркетинговой кампании). Причем можно гибко управлять процессом сегментации и создавать списки различного размера в зависимости от особенностей кампании, ограничений по ресурсам и бюджету, используя как пользовательские правила, так и результаты расчетов моделями. Полученные списки и вероятности откликов можно выгружать во внешние информационные системы или публиковать как аналитический веб-сервис.

Еще одной прикладной задачей в ритейле является *оптимизация запасов*. Поддержание оптимального уровня запаса продукции является актуальной задачей любой торговой, производственной или логистической

компании. Уровень ниже оптимального приведет к дефициту и последующим потерям продаж, а слишком высокий уровень съедает часть рентабельности организации.

Решение данной проблемы относится к разряду сложных с большой долей неопределенности. В оптимизации запасов аналитика данных активно используется для решения следующих задач:

1. Консолидация, подготовка и очистка исторических данных по продажам.
2. Прогнозирование спроса на основе исторических данных множеством способов с автоматическим выбором лучшей модели.
3. Расчет оптимального страхового запаса с поправкой на колебания спроса, сроков поставки, финансовых, транспортных, складских и множества других ограничений, позволяющего повысить оборачиваемость средств и увеличить вероятность наличия необходимых товаров.
4. Автоматическое формирование автозаказа и календаря закупок, автоматизирующую рутинную работу специалистов, участвующих в цепочке поставок.

Задачами аналитики данных в телекоме являются аналитическая отчетность, скоринг откликов, сегментация абонентов и так далее. В качестве задачи особой важности отметим *управление уходом клиентов* и *моделирование их жизненного цикла*. Отток клиентов повсеместно считается самой большой опасностью для провайдеров телекоммуникационных услуг.

Сегодня все больше организаций смещают акцент в работе от активного привлечения новых клиентов в сторону удержания существующих клиентов, а значит – эффективного выявления клиентов, склонных к оттоку, и предотвращения их ухода применением превентивных мероприятий. Необходимыми составляющими для решения этих задач служат, прежде всего:

- единая витрина данных, содержащая полную информацию по клиенту;
- разнообразная отчетность по оттоку и модели Data Maning, позволяющие сегментировать абонентов и прогнозировать их уход.

Для получения вероятностных оценок ухода клиентов (то есть отказа от пользования услугами) на конкретном горизонте будущего времени используются модели на принципах скоринговых карт (рисунок 1.25). Особенностью моделирования является тщательная подготовка данных (по клиенту требуется рассчитать большое количество агрегатов) и необходимость проверки множества гипотез (особенность в том, что между решением уйти и уходом проходит небольшой временной лаг, обычно 1-3 месяца, поэтому важно заранее выявить клиентов, склонных к оттоку).

Рисунок 1.25 – Пример управления оттоком клиентов

Еще одним направлением является построение ***кривых выживания*** – кривых вероятностей того, что «усредненный» клиент пробудет с компанией **X** дней. В отличие от скоринговых моделей, они описывают процесс оттока клиентов в течение длительного времени и применяются для прогнозирования месячных оттоков в целом по клиентской базе или ее сегментам, выявления ключевых точек жизненного цикла клиентов, понимания того, почему некоторые клиенты сотрудничают с компанией дольше других.

1.6. Инструменты аналитики данных

Среди основных причин популярности технологий аналитики данных можно отметить:

1. Развитие технологий автоматизированной обработки информации создало основу для учета сколь угодно большого количества факторов и достаточного объема данных.
2. Возникла острая нехватка высококвалифицированных специалистов в области анализа данных. Поэтому потребовались технологии обработки и анализа, доступные для специалистов любого профиля за счет применения методов машинного обучения.
3. Возникла объективная потребность в тиражировании знаний. Полученные в процессе KDD и Data Mining результаты являются формализованным описанием некоего процесса, а следовательно, поддаются автоматической обработке и повторному использованию на новых данных.
4. На рынке появились программные продукты, поддерживающие технологии аналитики данных – от настольных до корпоративных версий.
5. В настоящее время развитие технологий аналитики данных идет в сторону больших данных (англ.: Big Data), интернета вещей (англ.: Internet of Things), мобильной и облачной аналитики, а также low-code подхода. Но все это развитие невозможно без соответствующих программных инструментов.

На рынке программных средств для аналитики данных можно выделить некоторые стандарты де-факто и предложить следующую классификацию представленную на рисунке 1.26.

Рисунок 1.26 – Классификация программных средств для аналитики данных

Рассмотрим классификацию программных средств для аналитики данных подробнее.

Инструменты традиционного BI. Эти инструменты хорошо решают задачу визуализации данных, но с остальными задачами аналитики данных, такими как реализация разветвленной логики расчетов и построение моделей, справляются с трудом или вообще не справляются.

Настольные пакеты и библиотеки. Для индивидуальной работы востребованы *пакеты и библиотеки с алгоритмами Data Mining*. Они существуют двух видов: как получившие развитие статистические и математические пакеты прикладных программ, в которые добавились новые алгоритмы, и приложения нового образца, изначально создававшиеся как инструменты Data Mining визуальным проектированием логики обработки данных.

Такие пакеты ориентированы в основном на решение разовых задач. Отличительными особенностями являются:

- слабая интеграция с промышленными источниками данных (базы данных, веб-сервисы);
- конвейерная (поточная) обработка новых данных затруднительна или реализуется встроенными языками программирования и требует высокой квалификации;
- на локальных рабочих станциях обработка больших объемов данных затруднена.

Плюсом статистических пакетов является их широкая распространенность. Настольные Data Mining пакеты могут быть ориентированы на решение всех классов задач Data Mining или какого-либо одного, например, кластеризации или классификации.

В последнее десятилетие распространение получили специализированные языки программирования с библиотеками для решения задач аналитики данных, например, Python. Они предоставляют богатые возможности в плане алгоритмов, что важно для решения исследовательских задач и разработки различных прототипов решений.

Аналитические платформы. Аналитические платформы изначально ориентированы на комплексную аналитику данных и предназначены для создания и эксплуатации прикладных корпоративных решений в различных областях (рисунок 1.27).

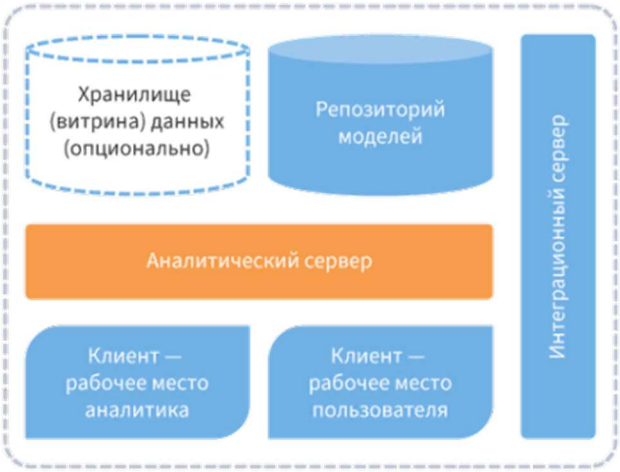


Рисунок 1.27 – Схема аналитической платформы

Аналитическая платформа – специализированный программный инструмент (или набор инструментов), который содержит в себе все средства для извлечения закономерностей из «сырых» данных: средства интеграции данных и управления метаданными, извлечения и преобразования данных, методы Data Mining и алгоритмы машинного обучения, средства визуализации и распространения результатов среди пользователей, а также возможности «конвейерной» обработки новых данных, коллективную работу над моделями, включая управление их версиями и разграничение прав доступа.

Еще один важный компонент аналитической платформы – *интеграционный сервер*. Он предоставляет специальный механизм обмена данными со сторонними приложениями, реализует так называемую сервис-ориентированную архитектуру (англ.: Service-oriented architecture, SOA). Обмен данными осуществляется посредством веб-запросов. На основе этой

технологии строятся большинство сервисов, предоставляющих аналитику данных как услугу (например, продажа прогнозов финансовых котировок, полученных моделью машинного обучения).

Реализация компонентов аналитической платформы может быть «привязана» к определенной ***системе управления базами данных (СУБД)***. Инструменты аналитики данных как бы встраиваются в СУБД.

Отличительные особенности подхода:

- высокая производительность;
- алгоритмы анализа данных по максимуму используют преимущества СУБД;
- жесткая привязка всех технологий анализа к одной СУБД;
- сложность в создании прикладных решений, поскольку работа с СУБД ориентирована на программистов и администраторов баз данных.

Отметим, что к облачной аналитике относится и понятия ***модель как услуга*** и ***данные как услуга***: использование результатов моделирования (модели разработаны поставщиком облачной услуги) путем получения прогнозов, оценок, вероятностей и т.п. для заданных входных воздействий. Хороший пример – покупка скорингового балла заемщика в бюро кредитных историй (балл рассчитывается моделью или каскадом моделей, которые построены на данных, накопленных в бюро кредитных историй).

Основное различие между low-code и no-code платформами состоит в том, что первые в некоторых случаях требуют написания программного кода, тогда как вторые обходятся вообще без программирования.

Применительно к аналитике данных в малокодовых системах используется принцип визуальной разработки, то есть визуальный интерфейс. Он включает в себя функции drag-and-drop и несложную логику настройки последовательностей шагов по обработке данных, которые делают процесс разработки простым и понятным по сравнению с традиционной разработкой путем написания кода.

Шаги представляют собой набор атомарных по отношению к данным операций, каждую из которых можно представить отдельным узлом. Примеры таких операций: выборка данных, фильтрация, сортировка, добавление нового столбца, построение модели и др. Такой способ представления очень близок к рассуждениям и действиям аналитика, которые он так или иначе проделывает в своей голове.

Рисунок 1.28 – Пример применения принципа визуальной разработки на базе Loginom

Набор узлов – поток данных – образует графическую диаграмму (рисунок 1.28). Чаще всего это граф. Иконка, соответствующая какому-либо узлу, несет в себе дополнительную семантику. Это помогает аналитику различать узлы по функциям и определять их активность. Также графовые структуры позволяют использовать принципы структурного проектирования: проектирование «сверху вниз», декомпозиция, абстрагирование и другие.

Low-code помогает организациям преодолевать ресурсные ограничения за счет повышения производительности труда и привлечения к разработке бизнес-пользователей и экспертов. Этому способствует низкий порог входа для пользователя и сокращение времени от идеи до реализации благодаря визуальной разработке.

1.7. Большие данные

Предпосылками к появлению Big Data являются [3]:

1. Технологии аналитики данных развиваются и совершенствуются уже несколько десятилетий, и за это время стали основным инструментом поиска новых знаний в массивах данных, необходимых для принятия эффективных управленческих решений. Аналитика данных стала привычным элементом информационного ландшафта организаций.

2. Параллельно с математическим аппаратом аналитики данных развивались и средства доставки, хранения, интеграции данных, что было обусловлено непрерывным возрастанием их объема, территориальной распределенности, сложности, требованиям по более адекватному описанию окружающего мира и событий в нем. Постепенно становилось очевидным, что потребности общества в области обработки «сырых» данных с целью их преобразования в полезные знания уже не могли удовлетворяться развитием

существующих подходов и методов, а требовали переосмысления с учетом новых реалий.

3. Назрел конфликт в терминологии. Обилие терминов и их трактовок, так или иначе связанных с аналитикой данных, способно запутать даже опытного исследователя.

4. Современные тренды в развитии аналитики данных накладывают ряд ограничений на использование приложений Data Mining. Основным из этих ограничений является то, что технологии Data Mining ориентированы, прежде всего, на обработку структурированных данных. Между тем, все больший интерес представляют собой данные, поступающие в режиме реального времени из социальных медиа, видео и фото регистраторов, электронной почты и других распределенных источников, расположенных во внешнем окружении. Основным свойством таких источников является наличие нарастающего высокоскоростного потока данных с неопределенной структурой.

За последние несколько лет человечество произвело информации больше, чем за всю историю своего существования и рост продолжается экспоненциально. Так, согласно прогнозам консалтинговой компании IDC, к 2025-му году объем данных в мире достигнет 173 Збайт (1 Збайт = триллион Гб), и 30% сгенерированных данных будут использоваться в режиме реального времени.

Это во многом было обусловлено быстрым ростом количества вычислительных средств, приложений и пользователей, участвующих в формировании глобальных потоков данных. Современная эпоха – эпоха мобильной связи, мобильного Интернета, социальных сетей, блогов, Интернета вещей привела к появлению миллиардов пользователей и миллионов приложений.

Необходимость обработки качественно новых объемов структурированных и неструктурированных данных показала, что традиционные подходы к их хранению и анализу стали неэффективными, а, следовательно, необходимы новые технологии. Аналитики рассуждают следующим образом: Мы не знаем, нужна ли нам информация, а если нужна, то какая, до тех пор, пока не проанализируем ее. Стоимость хранения информации настолько снизилась, что появилась возможность собирать все больше данных и анализировать их, руководствуясь принципом мы не знаем, чего мы не знаем. Например, может быть обнаружено, что площадь того или иного цвета на обложке журнала влияет на вероятность его продаж в определенном периоде.

Данные обстоятельства обозначили проблему, связанную с построением новой вычислительной инфраструктуры, которая была бы эффективной и не очень дорогой. Ключом к построению такой инфраструктуры и стал комплекс технологий, известный в настоящее время как ***Большие данные*** – ***Big Data***.

Стоит отметить, что в литературе нет единого мнения о том, что собой представляют Большие данные. Некоторые авторы рассматривают Big Data как Data Mining с кардинально увеличенными возможностями в плане объемов хранимых и обрабатываемых данных, а также скорости доступа к ним. Другие авторы рассматривают Big Data как небольшую составляющую Data Mining А третьи вообще не упоминают Data Mining в контексте Big Data.

Для того, чтобы дать определение Big Data с позиции специалиста по аналитике приведем примеры источников, которые порождают большие данные. В частности:

- *Торговые сети*. Крупные розничные торговые сети регистрируют ежечасно миллионы клиентских транзакций, которые пересылаются в хранилища данных, объем которых может составлять несколько петабайт.
- *Мобильные устройства*. Более 5 миллиардов людей по всему миру говорят, обмениваются сообщениями и производят поиск в Интернет с помощью мобильных устройств.
- *Автоматические регистраторы.* Тысячи автоматических регистраторов по всему миру непрерывно фиксируют погодные условия, и передают метеорологические данные в центры их обработки.
- *Социальные сети.* Пользователи социальных сетей ежеминутно отправляют десятки миллионов сообщений.

Таким образом, с точки зрения аналитики данных ***Big Data*** можно определить как технологию в области аппаратного и программного обеспечения, которая интегрирует, организует, управляет и анализирует данные, характеризующиеся четырьмя характеристиками: объемом, разнообразием, изменчивостью и скоростью. В частности:

- *Объем*. Поскольку в англоязычном варианте эти характеристики обозначаются, соответственно Volume, Variety, Variability и Velocity, то их часто называют ***четыре V***. Таким образом, если до сих пор характеристикой данных, определяющей организацию их обработки, был объем, то Big Data предполагает использование трех дополнительных параметров.
- *Разнообразие*. Отражает тот факт, что в отличие от технологий Data Mining, ориентированных на анализ данных, структурированных в виде таблиц, технологии Big Data должны позволять обрабатывать неструктурированные данные самых различных форматов, в том числе, текст, аудио и видео.

- *Изменчивость*. Возможность проводить обработку данных, которые могут непрерывно изменяться, что отличается от концепции хранилищ данных, используемых в бизнес-аналитике с базовым принципом неизменчивости данных.
- *Скорость*. Указывает на то, что требуется анализировать данные, которые не являются заранее консолидированными в некотором неизменчивом источнике, а представлены непрерывным потоком, поступающим по телекоммуникационным каналам.

У компаний, которым необходимо хранить и анализировать непрерывно возрастающие объемы данных, есть возможность выбора двух направлений развития вычислительной инфраструктуры:

1. *Вертикальное масштабирование*. Первый вариант – приобрести более мощный компьютер с большим количеством процессоров, объемом оперативной памяти, дискового пространства и так далее. Это называется *масштабированием по вертикали*, то есть добавление ресурсов на единственный вычислительный узел.
2. *Горизонтальное масштабирование.* Горизонтальное масштабирование базируется на добавлении дополнительных вычислительных узлов, то есть предоставляет возможность добавлять в систему дополнительные компьютеры и распределять работу между ними. Горизонтальное масштабирование позволяет построить высоконадежное решение с обеспечением должной степени резервирования на базе недорогих стандартных компьютеров. Сотни и даже тысячи маломощных компьютеров, объединенных в кластер, могут обеспечивать вычислительную мощность суперкомпьютеров.

Для решения задач аналитической обработки массивов данных, которые по своей локализации, размерам и структуре соответствуют Большим данным, используются технологии распределенных вычислений: вычислительная нагрузка распределяется между некоторым количеством (чем больше, тем лучше) компьютеров-клиентов, которые работают под управлением некоторого управляющего центрального компьютера. Последний распределяет «задания» между клиентскими машинами, получает результаты обработки и формирует из них общий результат. Современные реализации распределенных вычислительных систем позволяют эффективно обрабатывать терабайты, петабайты и даже экзабайты данных.

Для того чтобы построить инфраструктуру для больших данных, понадобятся две подсистемы: большое число компьютеров (узлов) относительно небольшой мощности, объединенных в вычислительную сеть

(кластер), а также программное обеспечение, способное распределять вычислительные ресурсы сети при решении сложных задач.

Кратко рассмотрим программные инструменты, которые в литературе наиболее часто упоминают как основу создания информационной инфраструктуры для Big Data: MapReduce, Hadoop и NoSQL.

MapReduce – модель распределенных вычислений, разработанная компанией Google, которая используется для параллельных вычислений над очень большими (несколько петабайт) массивами данных в распределенных вычислительных сетях. Компьютеры в таких сетях делятся на узлы, которые непосредственно производят вычисления, и главные узлы, которые получают задачу, разделяют ее на части и распределяют ее между рабочими узлами для предварительной обработки. Данный шаг называется ***map***.

После того, как мастер-узел получает от остальных машин сообщение о том, что обработка данных ими закончена (то есть шаг map завершен), он выдает команду на переход к шагу ***reduce*** (свертка), в процессе которого формируется результат, возвращаемый на мастер узел для формирования итогового решения.

При этом MapReduce это не какая-то конкретная программа, а метод организации распределенных вычислений, который может быть реализован с помощью программы, написанной на каком-то, наиболее удобном в конкретном случае языке (например, в реализации MapReduce в Google используется язык C++).

Hadoop. Проект фонда Apache Software Foundation, свободно распространяемый набор утилит, библиотек и программный каркас для разработки и выполнения распределенных программ, работающих на кластерах из сотен и тысяч узлов. Используется для реализации поисковых и контекстных механизмов многих высоконагруженных веб-сайтов. Разработан на основе модели распределенных вычислений MapReduce. Считается одной из основополагающих технологий Big Data.

NoSQL. Группа подходов, которые для хранения и обработки данных используют параллельные распределенные системы интернет-приложений (например, поисковые системы), но при этом отказываются от традиционных реляционных систем управления базами данных с доступом к данным с помощью языка SQL. Отсюда и термин – не SQL.

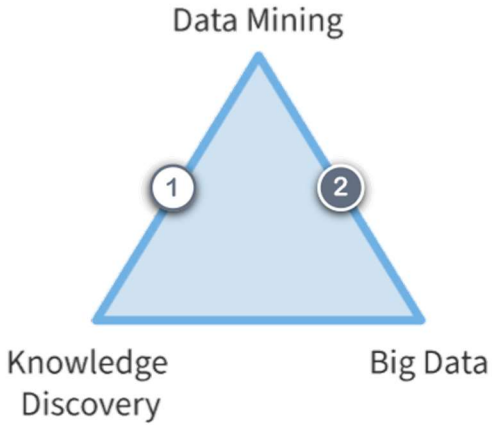


Рисунок 1.29 – Роль и место Big Data в аналитике данных

Технологии Knowledge Discovery и Data Mining решают задачи поддержки принятия решений на основе обнаруженных зависимостей и закономерностей в данных, описывающих бизнес-процессы компании. При этом предполагается, что чем больше данных будет задействовано, тем лучше будут полученные решения. Именно поэтому появление Больших данных очень быстро привело к появлению Большой аналитики или аналитики Больших данных.

Для создания моделей Data Mining необходимы структурированные данные, и далеко не всегда огромное число обучающих примеров, которое способно предоставить Big Data, способствует улучшению качества модели. Тогда возникает естественный вопрос: как соотносятся Big Data, оперирующие петабайтами данных неопределенной структуры, и относительно небольшие наборы выборок для построения предсказательных моделей, которые должны быть жестко структурированы? Роль Big Data сточки зрения предсказательной аналитики заключается в том, чтобы помочь «зачерпнуть» из стремительного и бурного потока данных образцы, анализ которых поможет описать закономерности всего потока с целью получения знаний о связанных с ним бизнес-процессах.

Таким образом, задача Big Data – управление огромными потоками данных из различных распределенных источников (Интернета, мобильных приложений, биржевой информации, аудио и видеорегистраторов, данных и так далее), проведение их описательного анализа и формирование наборов данных для построения моделей Data Mining.

Итак, Big Data правильнее рассматривать как технологию подготовки данных сверхбольшого, непрерывно возрастающего объема, расположенных в распределенных файловых системах и готовых к анализу методами Data Mining и бизнес-аналитики.

В этой связи возникает вопрос: существуют ли какие-либо особенности анализа данных уровня Big Data относительно обычных данных, то есть можно ли использовать к ним термин Большая аналитика? Таких особенностей две.

Во-первых, при использовании технологий Big Data в распоряжении исследователя оказывается намного больше данных, причем как структурированных, так и не структурированных. Поэтому для анализа необходимо использовать приложения, «умеющие» работать не только с табличными данными.

Во-вторых, при работе с данными уровня бизнес-аналитики, исследователь в большинстве случаев имеет представление о характере, природе и происхождении используемых данных, что очень важно при интерпретации результатов их исследования. В случае Больших данных такие представления, как правило, отсутствуют.

Рассмотрим еще один термин ***«наука о данных»*** или ***Data Science***. Впервые он прозвучал более 50 лет назад, но массово это понятие вошло в лексикон специалистов в области информационных технологий сравнительно недавно. Само понятие за это время значительно эволюционировало.

Бурное развитие информационных технологий привело к тому, что данные становились предметом все более пристального внимания исследователей в различных областях как потенциальный источник ценных знаний, использование которых в бизнесе обещает получение конкурентных преимуществ. Данные стали называть «новой нефтью». Параллельно развивалась и обогащалась наука о данных.

1966 год. Впервые термин «наука о данных» был введен профессором Копенгагенского университета Питером Науром в 1966 году. В основе концепции П. Наура лежит представление о данных, как о сырье, из которого могут быть сделаны те или иные полезные продукты для использования в других областях знаний и наук.

Следовательно, можно описать «жизненный цикл» данных с момента их появления и до момента практического внедрения разработанных на их основе продуктов. В этом контексте наука о данных представляет собой дисциплину, которая изучает этот жизненный цикл. Хотя П. Наура и считают «пионером» в области науки о данных, многие исследователи рассматривают ее истоки в разведочном анализе Дж. Тьюки.

2001 год. В 2001 году У. С. Кливленд опубликовал статью, в которой предложил рассматривать науку о данных как самостоятельную дисциплину «в контексте информатики и интеллектуального анализа данных». В частности он отметил, что «имеет место ограниченность знаний специалистов в области информационных технологий относительно подходов и методов организации

поиска знаний, с другой стороны, статистики недостаточно хорошо знают информационные технологии». Таким образом, по его мнению, «наука о данных должна связывать статистику и достижения в области компьютерной обработки данных».

2002 год. В 2002 году при Международном совете по науке (англ. International Council for Science, ICSU) начал издаваться журнал Наука о данных, который рассматривал проблемы описания информационных систем и их приложений. В нем Data Science определили как «дисциплину, объединяющую в себе различные направления статистики, Data Mining, машинное обучение и применение баз данных для решения сложных задач, связанных с обработкой данных».

2005 год. В 2005 году Национальный научный фонд США определил исследователей в области науки о данных как специалистов в области «информационных и компьютерных технологий, баз данных и программного обеспечения, а также программистов, экспертов по различным дисциплинам, библиотекарей, архивистов и других работников, которые участвуют в процессе сбора и обработки цифровых данных».

Значимыми вехами в развитии и становлении науки о данных стали работы под руководством Г. Пятецкого-Шапиро, в которых были заложены основы методик Knowledge Discovery и Data Mining. И, наконец, последним кирпичиком в фундамент науки о данных в современном ее понимании, явилось появление технологий Big Data.

Таким образом, в широком смысле, наука о данных решает практические задачи компьютерной обработки данных с целью получения полезных знаний. Зародившись как академическая дисциплина, наука о данных последовательно включала в себя как традиционные статистические подходы и разведочный анализ, так и технологии Knowledge Discovery, Data Mining и бизнес-аналитику. Важным направлением развития науки о данных в последние годы является обработка экстремально больших наборов данных Big Data.

Следует отметить, что общеупотребительным понятие науки о данных, как дисциплины, интегрирующей все направления использования данных для обнаружения в них полезных знаний (вместо набора различных терминов, употреблявшихся ранее), стало примерно в 2010 году. Поэтому часто науку о данных считают наукой, которой еще нет, которая находится на стадии становления и формирования. А в последнее время вместо науки о данных все чаще и чаще мы видим употребление термина аналитика данных.

В настоящее время в среде ученых и специалистов ведутся активные дебаты – правильно ли называть Data Science наукой. Мнения на этот счет разнятся.

Некоторые предлагают считать науку о данных частью статистики. Другие считают, что наука о данных – просто красивое выражение, не имеющего реального содержания. Третьи полностью идентифицируют науку о данных как Big Data.

Специалисты в области статистики склонны больше видеть в Data Science науку, а специалисты в области компьютерных технологий – данные.

Слово *наука* подразумевает знания, полученные путем систематических исследований. В первом приближении, это совокупность систематических действий, с помощью которых из данных извлекаются знания в форме проверяемых выводов, заключений и предсказаний. Можно сказать, что статистика решает те же самые задачи.

Основными отличиями Data Science от статистики являются:

- *Качество данных.* Статистические методы ориентированы на работу с качественными, структурированными данными. Если данные являются «сырыми», содержат пропуски, дубликаты, выбросы, то результаты статистического анализа могут оказаться смещенными. Поэтому необходима определенная предобработка и очистка данных.
- *Неструктурированные данные.* Все больше возрастает объем и роль неструктурированных и слабоструктурированных данных (видео, аудио, изображения, речь, текст, тэги и так далее), непосредственное применение статистических методов к которым невозможно.
- *Источники данных.* Еще одной проблемой является множественность и разнообразие источников данных, что требует их интеграции. Таким образом, применения в современных условиях для поиска знаний в данных только статистических методов недостаточно.

Что касается сравнения науки о данных с Big Data, то можно сказать, что для того, чтобы получить ценные и полезные знания на основе небольшого объема данных, необязательно привлекать Большие данные. Хорошему специалисту может оказаться достаточным для решения задачи нескольких сотен наблюдений, а другому и терабайт не поможет.

Таким образом, под общим названием Data Science в настоящее время существует множество различных, слабо систематизированных подходов, методов и технологий для анализа данных различного объема с целью поиска знаний. Однако наукой это можно называть весьма условно, поскольку для науки должны быть разработаны строгие предметы и методы исследований, что применительно к Data Science, вообще говоря, в настоящее время отсутствует. Поэтому Data Science правильнее считать междисциплинарным направлением информационных технологий, включающим все аспекты работы с данными с целью извлечения из них полезных знаний.

Глава 2. ОСНОВЫ РАБОТЫ В LOGINOM

2.1. Аналитические информационные системы поддержки принятия решений

В современном мире успех организации на рынке напрямую зависит от того, как быстро руководство может распознать изменения динамики рынка и насколько своевременно может отреагировать на них с целью увеличения прибыли. Менеджеры организации должны отслеживать тенденции рынка, идентифицировать конкурентов, контролировать ресурсы, проводить оценку рисков и др. Информация является необходимым производственным ресурсом для принятия эффективных управленческих решений.

Business Intelligence (BI) – это методы и инструменты для поиска, анализа, моделирования и доставки информации, необходимой для принятия управленческих решений. Технологии BI обрабатывают большие объемы данных, чтобы найти стратегические возможности для бизнеса.

Рождение BI датируется 1958 годом, когда американский ученый Ханс Петер Лун (1896-1964) опубликовал в IBM System Journal статью «A Business Intelligence System». В ней он представил бизнес как набор различных видов деятельности в науке, технологиях, коммерции, индустрии и даже в законодательной сфере, а обеспечивающие его системы – системами, поддерживающими разумную деятельность (intelligence system).

Словом «intelligence» Лун обозначал способность устанавливать взаимосвязь между представлениями отдельных фактов и действиями в интересах решения поставленных задач и достижения намеченных целей. В 1989 году аналитик из Gartner Ховард Дреснер дал BI расширительную трактовку, предложив использовать BI в качестве общего термина для различных технологий, предназначенных для поддержки принятия решений.

Поддержка BI рассматривается как совокупность различных технологий. Среди них по-прежнему остается и классический инструмент – электронные таблицы, а также генераторы отчетов, технологии OLAP (OnLine Analytical Processing – оперативная аналитическая обработка данных), технологии разработки данных и текстов и др.

Современные инструменты BI – это программное обеспечение, которое позволяет бизнес-пользователям видеть и использовать большое количество сложных данных. Знания, основанные на данных, (data-based knowledge) получаются из данных с использованием инструментов business intelligence и процесса создания и ведения хранилища данных (data warehousing).

Аспекты проблемы анализа данных и необходимые для их разрешения функции нашли выражение в соответствующих программных продуктах. Например, имеются комплексные информационно-аналитические системы, выполняющие в той или иной степени функции в соответствии с рассмотренными аспектами. Также на рынке представлены программные продукты и целевые программные системы, выполняющие в увеличенном объеме, расширенном составе и повышенной сложности какие-либо функции, например, оперативного или интеллектуального анализа.

В целом сложился рынок инструментальных средств создания и поддержки OLAP-систем, информационных хранилищ (DWH), СППР (DSS), интеллектуального анализа Data Mining, который получил обобщенное название – *Business intelligence (BI)*.

Условно эволюцию программных инструментов в области технологий анализа данных можно развить на три этапа: статистические пакеты, инструменты Dada Mining и low-code аналитические платформы.

Статистические пакеты. До появления аналитических платформ анализ данных осуществлялся в основном в статистических пакетах. Их использование требовало наличия хорошей математической подготовки, а большинство реализованных алгоритмов не позволяло эффективно обрабатывать большие объемы информации. Для автоматизации рутинных операций приходилось использовать встроенные языки программирования и скрипты.

Инструменты Dada Mining. В конце 80-х гг. произошел стремительный рост объемов информации, которая накапливается на машинных носителях, и возросли потребности бизнеса по применению анализа данных. Ответом этому стало появление новых парадигм в анализе: хранилища данных, машинное обучение, Dada Mining, Knowledge Discovery in Databases, Big Data, Deep Learning. Это позволило популяризировать анализ данных и решить большое число бизнес-задач с большим экономическим эффектом, что было недоступно статистическим пакетам.

Однако подавляющая часть ПО в области Dada Mining была настольной, а промышленные системы стоили очень дорого, и на этом рынке присутствовали несколько игроков.

Low-code аналитические платформы. Наконец, в начале 2000-х на рынке анализа данных стали доминировать специализированные программные системы – ***аналитические платформы***, которые полностью автоматизируют все этапы анализа от получения и интеграции данных до эксплуатации моделей и интерпретации результатов и базируются на (low-code принципах – языки программирования используются только в сложных

ситуациях, а все остальные процессы обработки данных настраиваются в визуальном редакторе.

Параллельно с этим большое развитие получили новые языки программирования для анализа данных: Python, R. Библиотеки алгоритмов машинного обучения, которые в них реализованы, позволяют обращаться к ним из low-code платформ, благодаря чему можно с высокой скоростью проектировать прикладные решения любой сложности. Это позволило популяризовать аналитику данных.

Многообразие представленных на рынке решений, от мощных платформ до простых систем аналитики и отчетности, позволяет выбрать решение, доступное любой организации. Развитие средств визуального представления данных, мобильных и облачных технологий сделали BI-инструменты массовыми всего за последние несколько лет.

Крупнейшие поставщики предоставляют всевозможные решения для реализации аналитических систем, такие как SAP Business Objects (разработчик – организация SAP AG), Oracle OLAP (разработчик – Oracle Corporation), IBM Smart Analytics System, Statistical Analysis System, Microsoft, Prognoz Platform (разработчик – организация «Прогноз»), АП Loginom (разработчик – организация ООО «Аналитические технологии» Loginom Company) и др.

SAP AG занимается разработкой и внедрением автоматизированных систем управления такими внутренними процессами предприятия, как: бухгалтерский учет, торговля, производство, финансы, управление персоналом, управление складами и др. Приложения обычно можно адаптировать под правовой контекст определенной страны. Аналитические приложения SAP работают с разнообразными источниками данных и ИТ-средами, в них предустановлены инструменты управления данными для той или иной отрасли или сектора.

Oracle (Oracle Corporation) – американская организация, крупнейший в мире разработчик программного обеспечения для организаций. Oracle Business Intelligence Suite – открытое, основанное на стандартах программное обеспечение, предоставляет единую, интегрированную инфраструктуру для бизнес-анализа, включающую комплексный набор продуктов для обработки запросов и проведения анализа, формирования корпоративных отчетов, доступа к средствам анализа с мобильных устройств, использования информационных панелей и порталов, интеграции с Microsoft Office и Excel, управления процессами бизнес-анализа, рассылки уведомлений в реальном времени, мониторинга бизнес-деятельности и множество других возможностей.

IBM Smart Analytics System предоставляет гибкий набор функциональных возможностей, включая бизнес-анализ, аналитическую отчетность, оценочные таблицы, инструментальные панели, извлечение информации из данных, сервисы Cubing Services (обеспечивающие многомерную визуализацию данных), текстовый анализ, управление хранилищем данных, а также серверную платформу и среду хранения данных.

SAS Institute Inc. (Statistical Analysis System) – американская частная организация, разработчик технологического программного обеспечения и приложений класса Business Intelligence, Data Quality и Business Analytics.

Microsoft поставляет BI-инструменты в составе трех групп продуктов – *Excel, SharePoint* и *SQL Server*. В каждом из них имеются свои функции аналитики и коллективной работы.

Prognoz Platform – платформа бизнес-аналитики для создания информационных систем и применения в качестве самостоятельного решения. Prognoz Platform позволяет разрабатывать приложения (в том числе мобильные) для оперативного анализа данных, а также моделирования бизнес-процессов и прогнозирования.

Loginom – это low-code аналитическая платформа для создания законченных прикладных решений в области анализа данных. Реализованные в Loginom технологии позволяют на базе единой архитектуры пройти все этапы построения прикладного решения: от интеграции данных до построения моделей и визуализации полученных результатов.

Сегодня Loginom – это яркий представитель как настольной, так и корпоративной платформы аналитики данных последнего поколения.

Платформа Loginom может быть инсталлирована и запущена в одном из двух режимов: локальном, как автономный продукт, и распределенном, когда клиентское приложение взаимодействует с удаленным сервером, что важно при обработке больших объемов данных и коллективном использовании.

В локальном режиме запускается обычное настольное приложение Loginom Desktop.

В распределенном режиме Loginom открывается в браузере и называется Loginom Studio.

2.2. Начало работы в Loginom

Окно выбора действий (домашняя страница) открывается сразу после входа в Loginom со стартовой страницы загрузки Desktop. Нажимаем кнопку **Создать черновик** (рисунок 2.1).

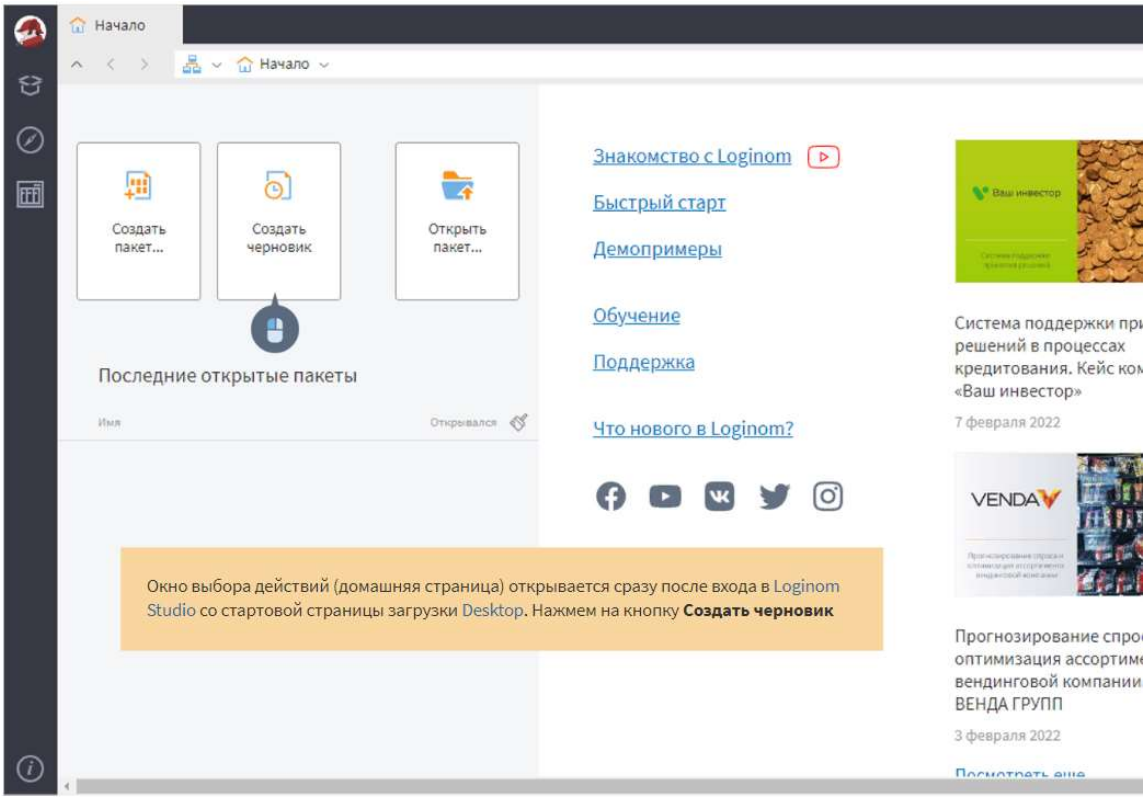


Рисунок 2.1 – Создание черновика в Loginom

Нажимаем **Пакеты**. В результате создается новый пакет с именем **Package1**. Этот пакет еще не сохранен (рисунок 2.2).

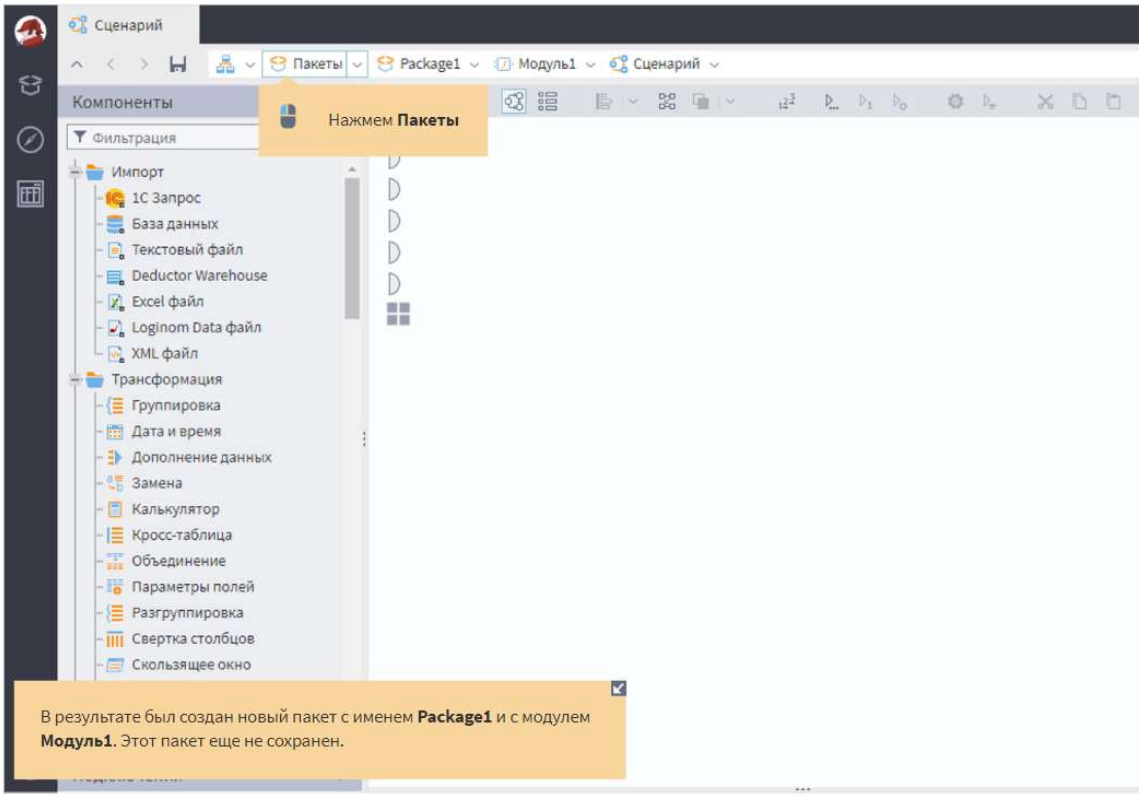


Рисунок 2.2 – Создание Пакета

Переходим к списку открытых данных. Клиентское приложение состоит из нескольких рабочих зон. Слева расположено боковое выдвижное меню с кнопками: **Меню**, **Пакеты**, **Навигация**, **Файлы**, **Процессы**.

На панели меню можно увидеть пользователя, под которым открыта текущая сессия Loginom, команду **Начало** для перехода на домашнюю страницу, команду вызова **Справки**, пункт **Задать вопрос** (открывает сервис Вопрос & Ответ), пункт **О программе** (открывает страницу с информацией о версии Loginom, редакции и др.) и команду **Выход** (закрывает все пакеты).

Нажатие на кнопку **Пакеты** откроет список команд для работы с пакетами. ***Пакет*** – это важное понятие платформы Loginom. Представляет собой контейнер для составных частей процесса обработки данных, файл с расширением *.lgp.

Выберем пакет, чтобы стали доступны команды сохранения (рисунок 2.3).

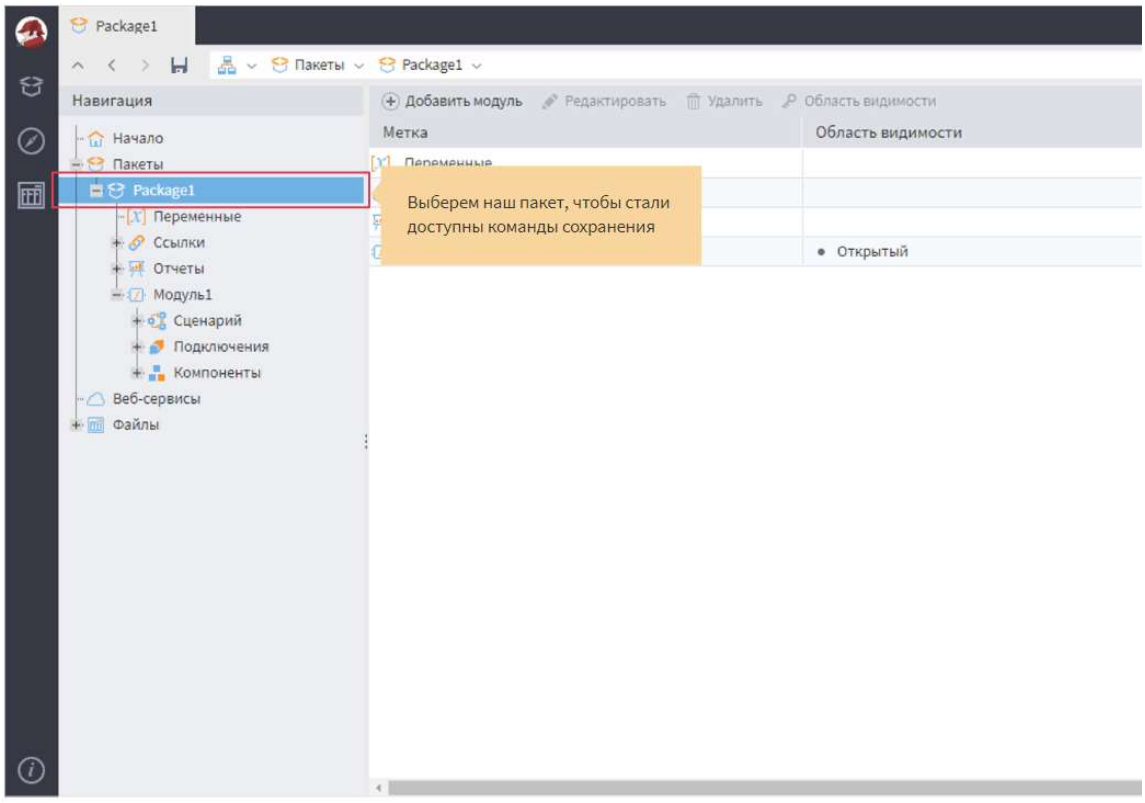


Рисунок 2.3 – Созданный Пакет

Активный пакет можно сохранить под текущим именем (команда меню **Сохранить**) или под другим именем (**Сохранить как…**), а также **Закрыть**. Кроме того, с помощью соответствующих команд одновременно сохранить или закрыть все открытие пакеты.

Все действия, которые можно проводить с пакетами, также доступны на панели инструментов (рисунок 2.4).

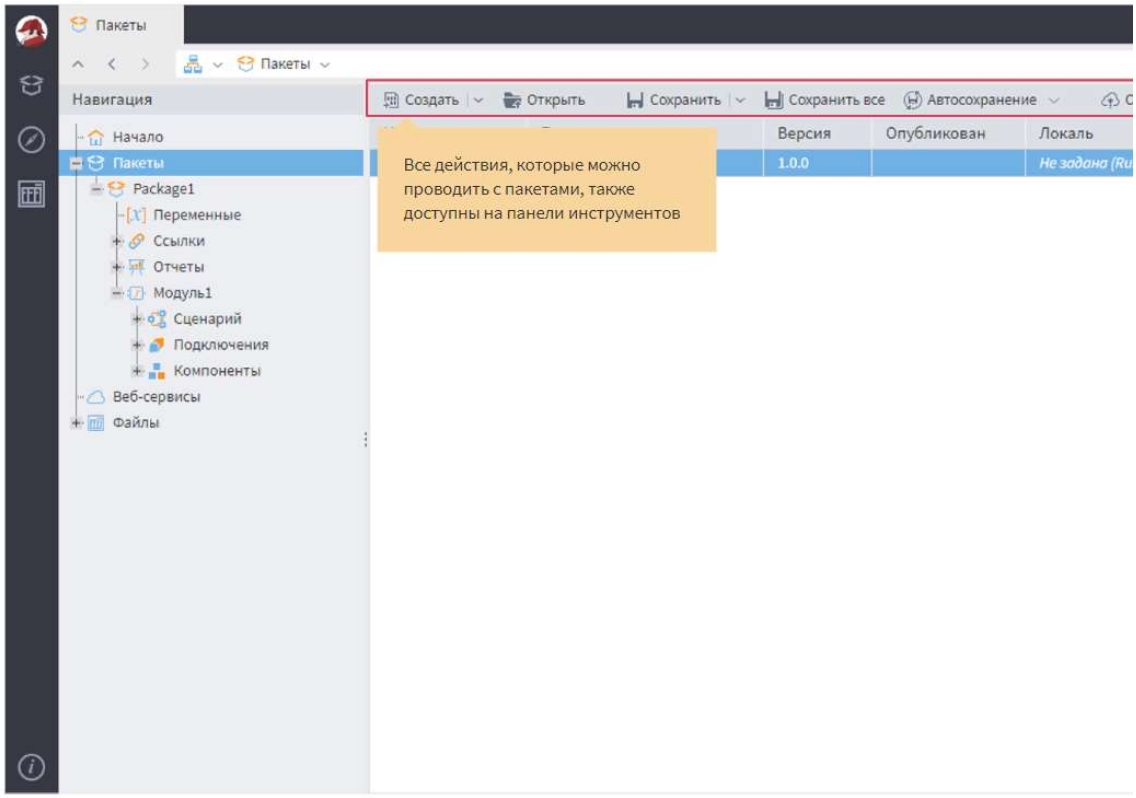


Рисунок 2.4 – Панель инструментов

Нажатие на кнопку **Навигация** откроет древовидную структуру объектов. Это так называемое ***дерево пакетов***. Верхним узлом являются имена пакетов, по умолчанию Package1, Package2 и др.

Слева от значков некоторых объектов есть значок +, который указывает на наличие иерархии. Щелкнув на нем левой кнопкой мыши, можно развернуть ветвь дочерних объектов. Повторный щелчок позволит свернуть ветвь.

Каждый пакет состоит из модулей, отчетов, ссылок и переменных.

Модули. Модуль включает в себя:

- *Сценарий* – последовательность шагов по обработке данных;
- *Подключения* – настроенные источники и приемники данных;
- *Компоненты* – производные компоненты, доступные в текущем модуле.

Каждый пакет содержит хотя бы один модуль.

Отчеты. Отчеты представляют собой визуализаторы, настроенные для набора данных и добавленные в группу отчетов.

Ссылки. Ссылки применяются для подключения других пакетов с целью использования их элементов в текущем пакете. Доступность элементов определяется специальными модификаторами доступа.

Переменные. Существует несколько категорий пользователей Loginom. Лицу, принимающему решения, – пользователю Loginom Viewer – доступен только просмотр готовых отчетов, подготовленных аналитиком. Иногда для отчета требуется задать некоторые параметры, например, диапазон дат, которые необходимо включить в отчет. Эти параметры аналитик создает в качестве переменных при настройке логики обработки данных, а пользователь Viewer может задать их значения, перейдя в **Переменные**.

Вкладки расположены в специальной верхней зоне. Кроме них, там находится адресная строка с кнопками для навигации по дереву пакетов **Наверх**, **Назад**, **Вперед**. Они позволяют сделать один шаг в прямом или обратном направлении, либо подняться на самый верхний уровень к списку пакетов. Там же расположена кнопка сохранения текущего пакета.

Чтобы открыть содержимое объекта, значок которого отображается в области просмотра, дважды щелкнем на нем левой кнопкой мыши (рисунок 2.5).

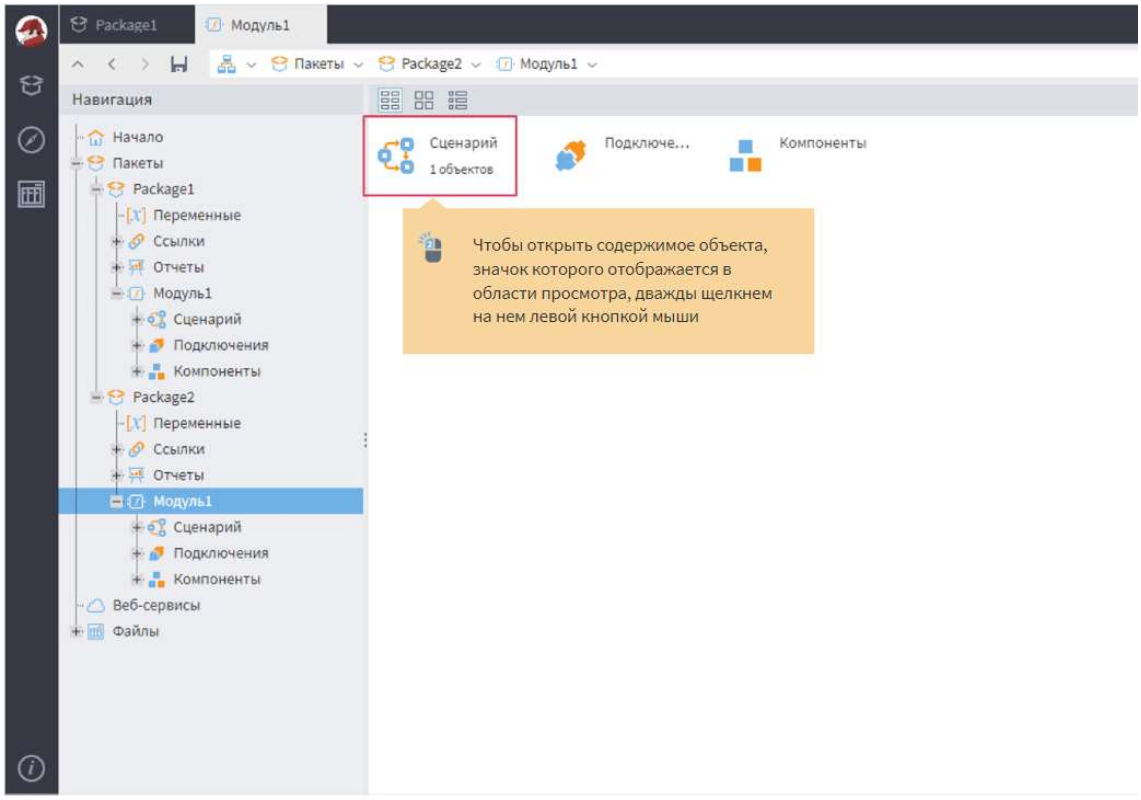


Рисунок 2.5 – Открытие содержимого объекта

Сценарий по умолчанию пустой, а слева место панели навигации заняла слайд-панель с тремя вкладками **Компоненты**, **Производные компоненты**, **Подключения**.

Сценарий – главная составная часть любого модуля, представляет собой последовательность шагов по обработке данных. Эти шаги задаются узлами из компонентов: стандартными (из списка на панели **Компоненты**) или производными. В первом приближении под ***производными компонентами*** будем понимать пользовательские компоненты, созданные на основе стандартных. При использовании баз и других промышленных источников данных могут также понадобиться узлы подключений.

Стандартные компоненты объединены в логические группы: **Импорт**, **Трансформация**, **Data Mining** и другие.

Проектирование сценария ведется в специальной зоне, или ***области построения сценария***. Компонент при добавлении в область построения превращается в ***узел***.

Добавить узел в сценарий можно двумя способами. Первый – через **Drag & Drop**: достаточно перетащить мышью нужный компонент в область построения (рисунок 2.6). Второй способ – вызвать контекстное меню для нужно компонента и нажать команду **Добавить узел в сценарий**.

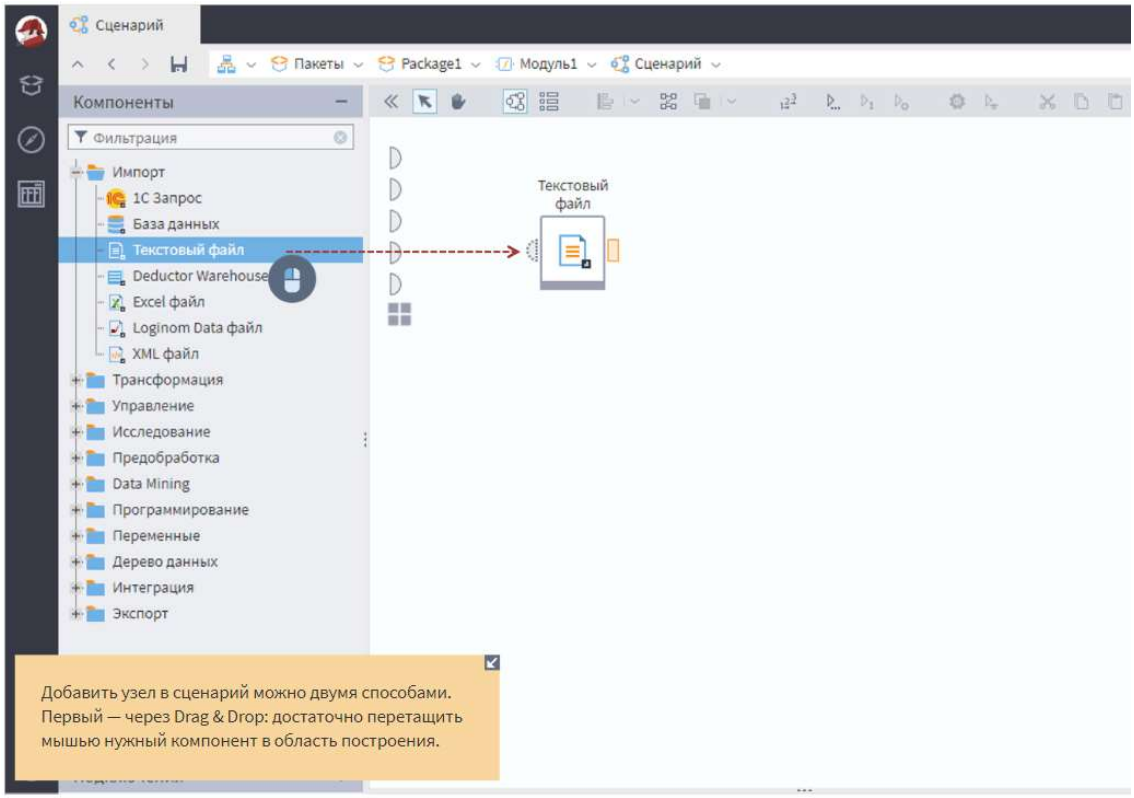


Рисунок 2.6 – Создание узла в области построения сценария

Каждый узел в сценарии может работать с несколькими видами объектов, основными из которых являются два: набор данных и переменные. Наличие тех или иных объектов у узла опционально и зависит от конкретного компонента.

Передача этих объектов между компонентами осуществляется посредством ***портов***, и для каждого вида объекта имеется свое графическое обозначение порта. Так, для набора данных это ***прямоугольник***.

Несмотря на то, что групп компонентов на слайд-панели много, существует только три типа действий в сценарии: *импорт*, *обработка* и *экспорт*. Так, компоненты импорта только «отдают» набор данных, поэтому прямоугольники (выходные порты) у них ***справа***.

Наоборот, компоненты экспорта только «принимают» набор данных, поэтому прямоугольники (входные порты) у них ***слева***, и продолжить сценарий после них невозможно.

Наконец, компоненты обработки имеют как входные, так и выходные порты, то есть прямоугольники имеются на каждой из сторон: слева и справа. Исключение составляют несколько компонентов группы **Управление**: порты появляются только после настройки таких узлов.

Любой узел, с которым мы хотим производить какие-либо действия, сначала нужно выделить, щелкнув по нему мышкой.

Его границы станут голубого цвета, а на месте иконки появятся четыре кнопки. Например, первая кнопка – ***Выполнить узел***. Это перевода узла в активное состояние, то есть выполнение расчетов, которые в нем заложены.

Последовательность обработки данных требует связывания компонентов между собой. Делается это снова через механизм **Drag & Drop**: удерживая нажатой левую кнопку мыши на выходном порте, подведите ее и «наложите» поверх нужного входного порта другого компонента (рисунок 2.7)

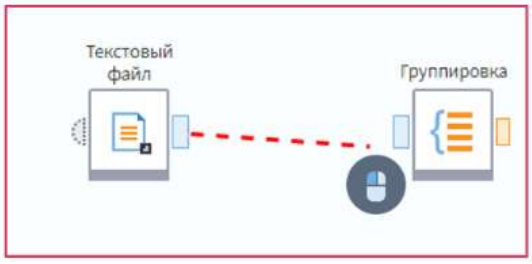


Рисунок 2.7 – Установление связи между узлами

Удалить связь легко, для этого нужно ее выделить и нажать соответствующую команду. Выделенная связь отмечается зеленой

пунктирной линией. Этой же командой можно удалить узел, предварительно выделив его.

После запуска узла на обработку он может принять одно из двух состояний: успешное или неуспешное. На рисунке 2.8 показано неуспешное состояние – контуры узла стали красными. Узнать причину этого можно, нажав на нижнюю область с иконкой значка «!». Появится модальное окно с сообщением о причине события.

Рисунок 2.8 – Неуспешное состояние узла

Если узел отработал успешно, его контур станет зеленым. Контуры портов также станут зелеными, что означает, что они активированы, и в них есть какие-то данные (рисунок 2.9).

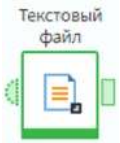


Рисунок 2.9 – Успешное состояние узла

В таком состоянии кнопка **Выполнить узел** заменяется на **Деактивировать узел**. Это перевод узла в неактивное состояние с освобождением занятых им системных ресурсов. При деактивации данные становятся недоступны.

Когда обработка в узле занимает продолжительное время, его внутренняя часть превращается в индикатор прогресса выполнения. На время обработки узел блокируется: его нельзя будет открыть для перенастройки. Об этом говорит иконка с «замком».

Теперь кратко остановимся на вопросе использования переменных. В Loginom переменные играют важную роль при создании динамических сценариев. В каждом узле можно использовать переменные. Но начнем мы знакомство сними с набора из пяти портов, которые всегда присутствуют в любом модуле – ***Переменные сценария*** (рисунок 2.10).

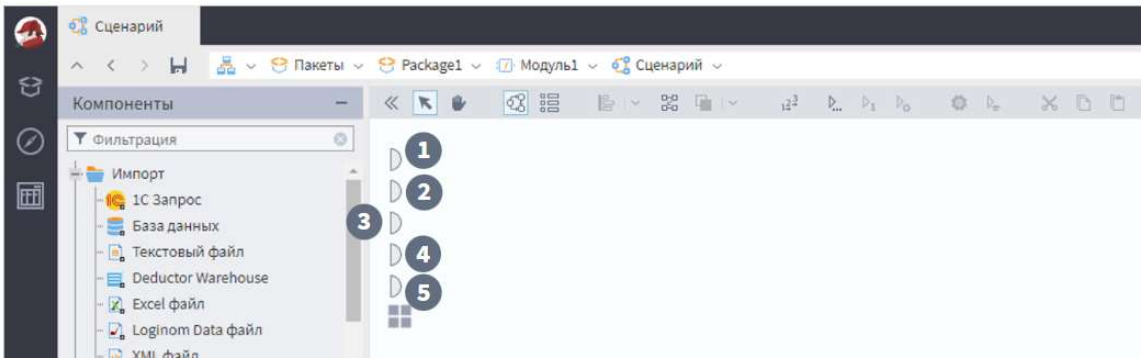


Рисунок 2.10 – Переменные сценария

Переменная – объект, который может содержать только одно значение для вычислений. В этом их основное отличие от набора данных. В остальном, они точно так же могут передаваться из узла в узел и использоваться внутри.

Порты, которые взаимодействуют с наборами переменных, графически отображаются полукругом (рисунок 2.11).

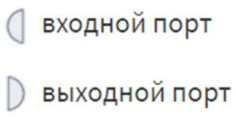


Рисунок 2.11 – Входной и выходной порты

Выходной порт для переменных может иметь линию связи только с входным портом для переменных (рисунок 2.12).

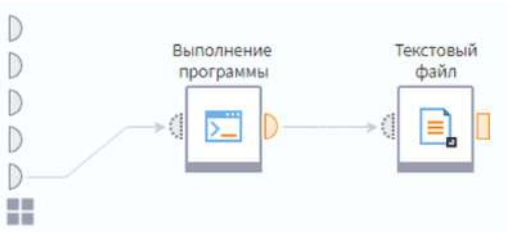


Рисунок 2.12 – Пример связи выходного порта

2.3. Компонент и узел в Loginom

Компонент – элемент Loginom, предназначенный для обработки данных и содержащий определенную логику. Компоненты располагаются на слайд-панели в левой части программного окна (рисунок 2.13).

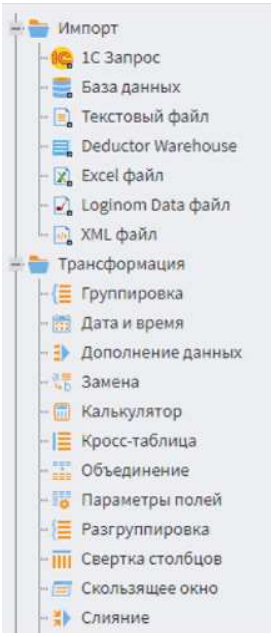


Рисунок 2.13 – Компоненты

Стандартные компоненты включены в состав Loginom и располагаются на вкладке **Компоненты**. Производные компоненты отражаются на соответствующей вкладке. Они создаются пользователями из экземпляров стандартных компонентов и содержат определенную последовательность обработки данных. Все производные компоненты сгруппированы по пакетам.

Экземпляр компонента, который помещен в область построения сценария, называется ***узлом***. Узел имеет входные и/или выходные порты, настройки по умолчанию, а также интерфейс изменения настроек в виде пошагового мастера.

С узлом можно проводить различные действия: настраивать, переименовывать, удалять и др.

Последовательность узлов со связями образует сценарий обработки данных.

Графический узел представляется пиктограммой с указанием метки, входных и выходных портов. Входными и выходными портами узла могут быть подключения, наборы данных, переменных и дерево данных. Порт каждого вида обозначается соответствующей пиктограммой (рисунок 2.14).

Рисунок 2.14 – Пример входных портов узла экземпляра компонента «База данных»

Выходные порты типа подключение могут быть только у узлов подключений. Для большинства узлов основные порты – это наборы данных и переменные. Количество входных и выходных портов варьируется и определяется функционалом узла (рисунок 2.15).

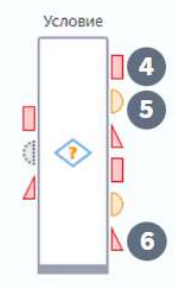


Рисунок 2.15 – Пример выходных портов узла экземпляра компонента «Условие»

Порт может находиться в одном из двух состояний: ***сконфигурирован*** / ***не сконфигурирован***.

Сконфигурированным порт может быть:

- по умолчанию (настройки будут взяты по умолчанию);
- если подключена связь – тогда настройки передаст узел, от которого идет связь.

Сконфигурированный порт может быть ***активным*** (данные есть; отображается зеленым цветом) или ***неактивным*** (данных нет; отображается серым цветом).

Несконфигурированный порт обводится красной рамкой (рисунок 2.16).

Рисунок 2.16 – Примеры сконфигурированного и несконфигурированного порта

Существует также промежуточное, состояние, когда порт **частично сконфигурирован**. При добавлении узла в область построения его выходные порты наборов данных всегда частично сконфигурированы.

Если для выполнения узла необходимо подключить связь к порту, такой порт называется **обязательным** и отображается сплошной линией. **Необязательные** порты отображаются пунктиром.

Количество портов узла может быть фиксированным или динамическим. Добавить порт можно с помощью кнопки или приема **Drag & Drop**.

Когда на входе узла нет обязательных портов, но имеется несколько необязательных, должен быть сконфигурирован хотя бы один из них.

Любой порт можно переименовать с помощью команды **Редактирование метки порта…**

При наличии у узла нескольких портов полезно изменять их метки в соответствии с наборами данных, которые ожидаются на входе. Например, **Транзакции** или **Каталог товаров**.

Узел можно ***переобучать***. Эта функция используется для узлов, основанных на обучении (нейронные сети, деревья решений и прочее). Переобучение особенно необходимо в тех случаях, когда набор данных изменился, и требуется перестроить модель на новых данных.

Если необходимо получить идентичный узел с такими же настройками и свойствами, его можно ***клонировать*** или ***копировать***. При клонировании связи, идущие к узлу, сохраняются, при копировании – нет.

Для клонирования следует выделить нужный узел и выбрать команду **Клонировать узел**.

Для удобства работы предусмотрен ряд действий с группой узлов, находящихся в области построения. Эти действия помогут сократить время на модификацию сценария, структурировать его и сделать более наглядным.

Для каждого узла можно настроить ***модификаторы доступа***, которые определяют область его видимости. Благодаря этому становится возможным многократное использование узла, например, через ссылку на него в сценарии.

Настройка модификатора доступа проводится через меню **Другие действия**.

Для оперативного доступа к данным выходных портов без настройки визуализаторов предусмотрена функция ***быстрого просмотра***: нажатием правой кнопкой мыши на порте и выбором соответствующего пункта контекстного меню, либо двойным нажатием левой кнопкой мыши (рисунок 2.17).

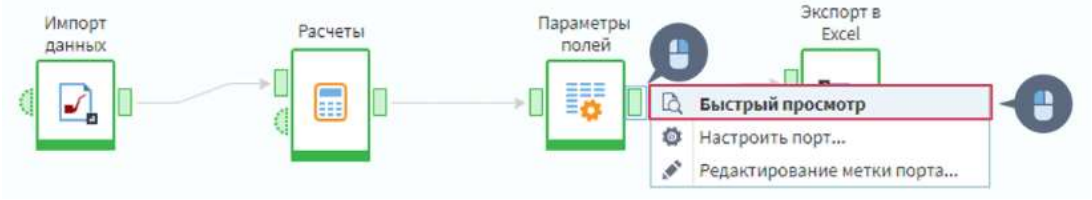


Рисунок 2.17 – Функция быстрого просмотра

2.4. Первый сценарий в Loginom

На рисунке 2.18 представлен пример сценария. По сути, сценарий – набор узлов, связанных между собой и выстроенных в определенной последовательности, которая позволяет решить поставленную задачу. Каждый узел отвечает за конкретный шаг алгоритма решения задачи.

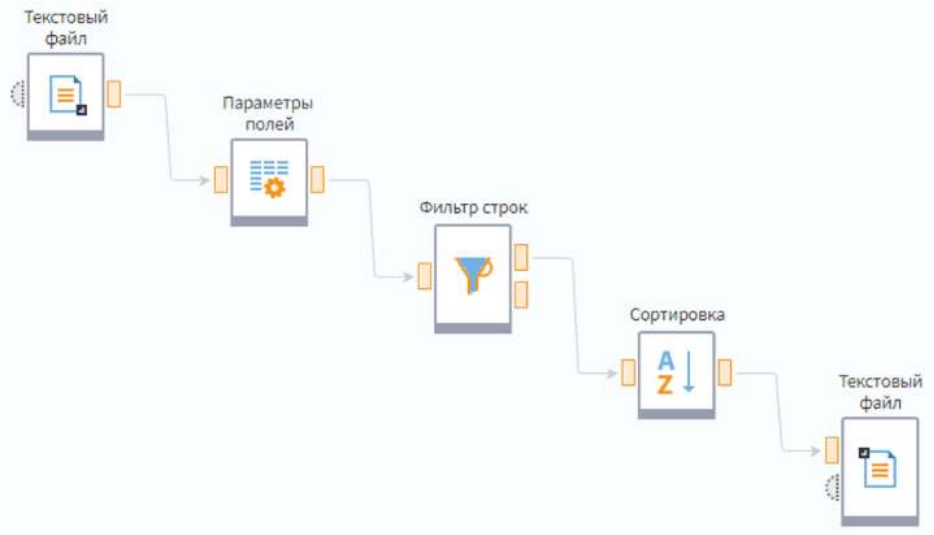


Рисунок 2.17 – Пример сценария

Первым идет узел **импорта**. В сценарии их может быть несколько, и они могут быть разных видов – в зависимости от формата данных, который необходимо импортировать.

Основная часть сценария – группа узлов обработки данных. Логически правильный набор и последовательность узлов отвечают за корректное решение задачи.

Последняя часть – узел экспорта. Он отвечает за сохранение результатов обработки во внешний источник.

В готовом сценарии не всегда присутствуют все три типа действий. Узла импорта может не быть, если в расчетах используются только переменные. Узел экспорта отсутствует, когда нужны только отчеты, которые удобнее просматривать непосредственно в приложении. Могут отсутствовать даже узлы обработки – в случаях, когда необходимо только изменить формат данных.

Создадим новый пакет и рассмотрим все указанные этапы подробнее путем создания небольшого сценария. Итак, первое – это импорт данных.

Существует 7 стандартных компонентов для создания узлов импорта (рисунок 2.19).

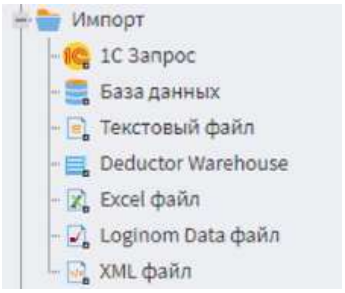


Рисунок 2.19 – Компоненты для создания узлов импорта

Компоненты импорта можно условно поделить на две группы: компоненты, для которых требуется предварительно настроить подключение; компоненты, для которых оно не требуется.

К первой группе относятся: 1С Запрос, База данных, Deductor Warehouse, XML файл. Во вторую группу входят: Текстовый файл, Excel файл, Loginom Data файл.

1С Запрос. Компонент предназначен для импорта из базы данных 1С.

База данных. Этот компонент позволяет импортировать данные из баз данных различных типов.

Deductor Warehouse. Это собственный формат Хранилища данных, специально разработанный для использования с Loginom и Deductor. Через данный компонент осуществляется подключение к Data Warehouse, которое было спроектировано на предыдущем поколении аналитической платформы – Deductor (на базе Firebird, MS SQL, Oracle). Важно: в аналитической платформе Loginom на текущий момент возможно лишь импортировать данные из хранилища данных Deductor Warehouse. Полная поддержка этого формата есть в аналитической платформе Deductor.

XML файл. Предназначен для импорта xml-файлов.

Текстовый файл. Компонент предназначен для импорта файлов в txt или csv-формате, строки и столбцы данных в которых разделены однотипными символами-разделителями.

Excel файл. Позволяет импортировать файлы в xls или xlsx формате.

Loginom Data файл. Компонент для импорта файлов в собственном формате (расширение файла *.lgd).

Это формат, разработанный для работы с Loginom. Его использование обеспечивает максимальную скорость импорта и экспорта данных. Скорость обмена информацией в общем случае ограничивается только скоростью работы дисковой подсистемы и гораздо меньше зависит от объема данных, чем при работе с прочими форматами. Благодаря встроенным алгоритмам сжатия размер файла в lgd -формате также будет меньше, чем тот же набор данных, хранимый, например, в формате csv.

LGD-файл уже содержит в себе всю необходимую информацию о параметрах полей набора данных (имя поля, метка, тип данных и так далее), что позволяет выполнять экспорт и импорт данных с минимальными настройками.

Создадим узел импорта путем добавления в область построения компонента **Текстовый файл** и перейдем в его настройку (рисунок 2.20).

Рисунок 2.20 – Создание узла сценария Текстовый файл

В параметре **Имя файла** задается путь к файлу, который необходимо импортировать.

Компонент **Фильтр строк** предназначен для выделения из набора данных записей, которые соответствуют определенным условиям. Условия задаются при настройке узла. Фильтр строк позволяет разделить набор данных на две части: записи, которые удовлетворяют заданным условиям, и остальные записи.

Компонент **Сортировка** позволяет отсортировать набор данных по возрастанию или убыванию значений. Сортировка возможна по одному или нескольким полям.

Добавим в область построения узел **Фильтр строк** и создадим связь с узлом импорта (рисунок 2.21).

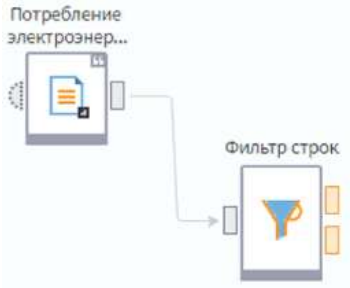


Рисунок 2.21 – Установления связи между узлом Фильтром строк и узлом импорта

Рассмотрим несколько условий фильтрации.

<, <=, >, >=, = <>. Условию будут соответствовать записи, значения которых в поле, участвующем в фильтре, выбранным образом соотносятся с содержимым поля Значение для сравнения.

Пустой. Отбираются записи, для которых в поле содержатся пустые значения.

Не пустой. Отбираются записи, для которых в поле содержатся любое значение, кроме пустого.

В интервале (вне интервала). Условию соответствуют записи, значения которых лежат в указанном диапазоне (вне указанного диапазона). Используется для числовых полей и полей типа **Дата/время**.

В списке (вне списка). Отбираются записи, которые содержатся в выбранном списке (вне выбранного списка).

Содержит (не содержит). Условие для строковых полей. Отбираются записи, значения которых в данном поле содержат (не содержат) указанную последовательность символов.

Отдельно необходимо обговорить ситуацию, когда в фильтруемом поле присутствуют значения <null>.

При выборе для такого поля условий <> (не равно) или **не содержит** некое значение, в выходной набор **Соответствуют условию** не попадут записи с пустыми значениями в данном поле. Пример представлен на рисунке 2.22.

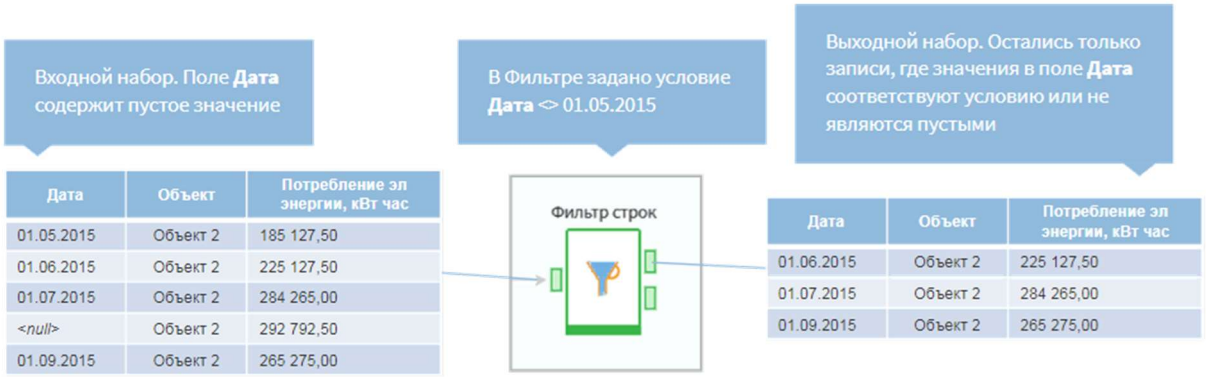


Рисунок 2.22 – Пример работы Фильтра строк, когда в фильтруемом поле присутствуют значения <null>

Вернемся к сценарию. Добавим в область построения компонент **Сортировка**. Он позволяет отсортировать записи входного набора данных последовательно по нескольким полям.

Свяжем узел **Сортировка** с узлом фильтра и перейдем в настройку узла (рисунок 2.23).

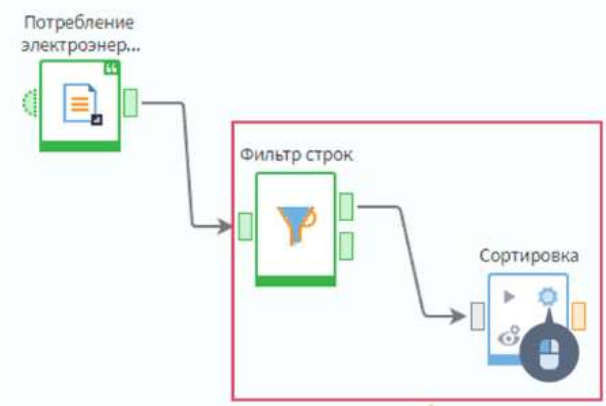


Рисунок 2.23 – Установления связи между узлом Сортировка и узлом Фильтром строк

Создадим узел экспорта в Loginom Data файл в нашем сценарии. По умолчанию при экспорте выгружаются все поля. Если нам нужно экспортировать только часть полей, настроить необходимые можно с помощью настройки входного порта.

2.5. Компонент Калькулятор

Калькулятор – компонент Loginom с наиболее широкой сферой применения. Он предназначен для добавления в наборы данных новых полей, значения которых вычисляются по формулам, заданным при настройке узла.

Компонент содержит ряд встроенных функций для проведения различных расчетов, а также позволяет использовать в формулах значения переменных и полей, в том числе полей, вычисленных ранее в этом же узле.

В **Калькуляторе** используется механизм ***ленивых вычислений***. Это означает, что расчет производится только в момент, когда необходимо отобразить значение выражения или использовать его в другом узле сценария. Прейдем в настройку узла и посмотрим, каким образом создаются новые поля набора данных.

Основная настройка узла осуществляется на первом шаге. Окно настроек состоит из нескольких областей (рисунок 2.24). Рассмотрим их подробнее.

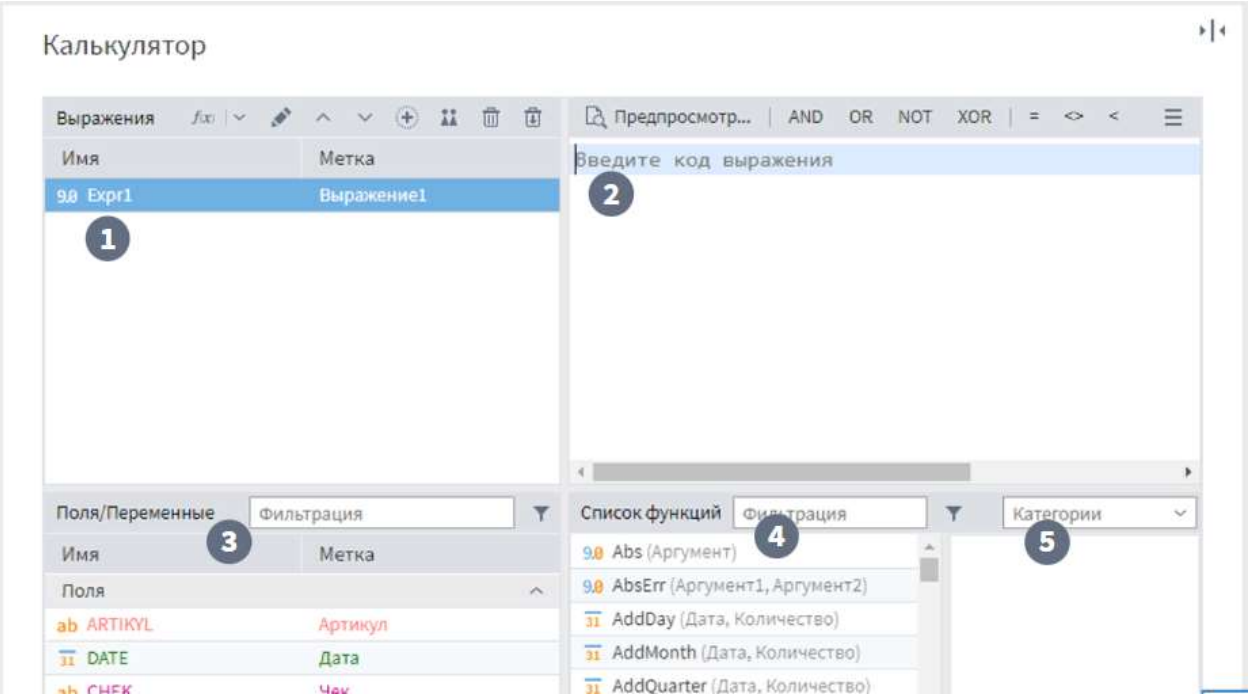


Рисунок 2.24 – Настройка узла компонента Калькулятор

В списке выражений по умолчанию содержится одно новое пустое выражение вещественного типа. Для его редактирования дважды щелкнем по нему левой кнопкой мыши.

Раскрывается область редактирования параметров выражения.

В формуле выражения возможно использовать ссылки на ранее созданные выражения при условии, что они находятся выше по списку. При установке флага **Промежуточное выражение** может участвовать в расчетах, но оно не будет передано в выходной набор данных.

Включение флага **Кэшировать** позволяет сохранить рассчитанное однажды значение выражения для его использования последующими узлами и визуализаторами без выполнения повторных вычислений. Это обеспечивает увеличение скорости работы этого и последующих узлов. Рекомендуется включать его при использовании функций, вызываемых каждый раз при необходимости получения значения выражения, например, Random(), Today() и другие.

Правила составления выражений соответствуют общепринятым в математике. Формула в целом или аргумент функции могут содержать:

- ссылки на переменные и поля набора данных в виде их имен, в том числе на поля, созданные в этом узле (поле, на которое ссылаются, должно стоять в списке выражений до поля, в котором присутствует ссылка);
- скобки, определяющие порядок выполнения операций;
- знаки математических операций и отношений;

- логические операции (and, or, not, xor) и значения (true или 1, false или 0);
- имена функций;
- целые и вещественные числа;
- строковые выражения в кавычках: «строковое выражение»;
- однострочные и многострочные комментарии.

Все функции списка сгруппированы по категориям. Для того чтобы быстрее найти нужную функцию, можно использовать фильтрацию по первым буквам или выбрать соответствующую категорию.

Имя функции в области списка выражений появляется вместе со скобками, куда вводятся аргумент или аргументы. Аргументы, которые необходимы для той или иной функции, можно увидеть в области описания функций. Аргумент может содержать те же символы, что и формула в целом. Если у функции несколько аргументов, они отделяются запятыми.

С помощью узла **Фильтр строк** невозможно отфильтровать строки по значению поля, равного бесконечности. Заменить бесконечность на другое значение при помощи узла **Замена** также не получится, по крайней мере, без использования регулярных выражений.

Кроме того, для дальнейшей обработки обычно удобнее, чтобы вместо бесконечности при делении на ноль в поле также стояло значение ноль или пустое значение. Поэтому, если есть вероятность того, что в поле-делителе окажутся нулевые значения, рекомендуется при делении использовать функцию IFF().

Рассмотрим пример. Пусть нам нужно поделить поле Amount на поле Cnt, в поле Cnt часть значений равны нулю. Чтобы в новом поле вместо бесконечности были пустые значения, нам необходимо использовать выражение: IFF(Cnt=0,null,Amount/Cnt).

2.6. Лабораторная работа №1 «Первый сценарий в Loginom»

Для выполнения задания скачайте файл Товары.txt, который содержит следующие поля (рисунок 2.25):

- Артикул – артикул товара;
- Группа товара – наименование группы товара;
- Дата продажи – дата последней продажи товара.

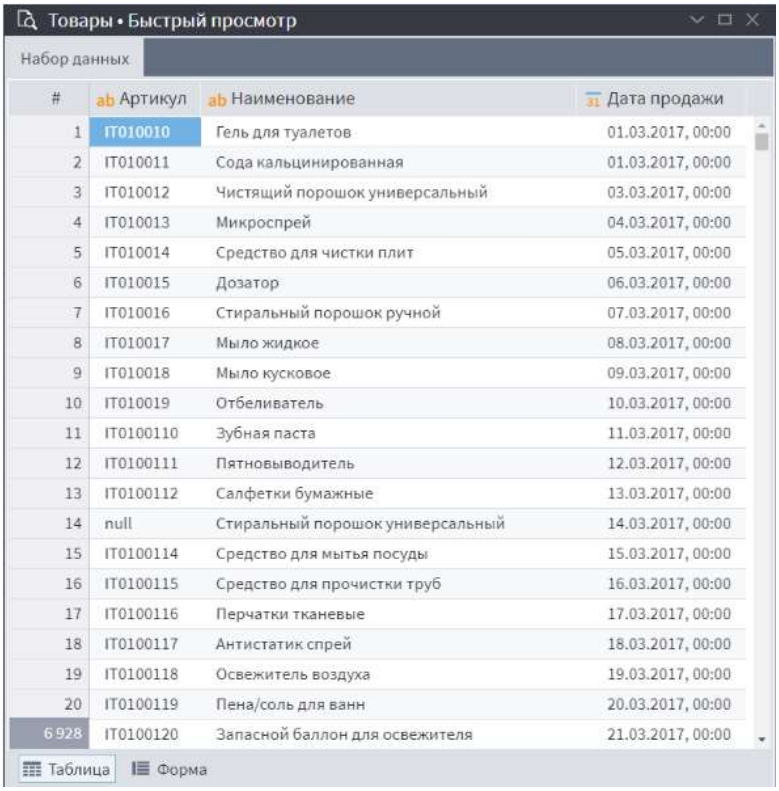


Рисунок 2.25 – Поля файла Товары.txt

Задание:

1. Создайте пакет в Loginom и импортируйте файл **Товары.txt**.
2. Исключите из набора записи, в которых для товара отсутствует **артикул**.
3. С помощью функций компонента **Калькулятор** рассчитайте, сколько месяцев прошло от даты последней продажи каждого товара до 01.04.2018.
4. Отсортируйте набор данных по количеству месяцев по убыванию.
5. Добавьте в набор поле логического типа **Вывод из продажи** и установите значение **true** для товаров, у которых от даты последней продажи до 01.04.2018 прошло более 10 месяцев.

Ответьте на вопросы:

1. У какого количества товаров отсутствует артикул?
2. Сколько записей осталось в наборе после исключения товаров с пустым артикулом?
3. Какой Артикул товара оказался первым после сортировки?
4. Сколько месяцев прошло с даты последней продажи товара с артикулом IT010016864?

2.7. Настройка портов. Автосинхронизация

Тип, ***вид*** и ***назначение*** поля, а также понятие автоматической синхронизации – важнейшие понятия Loginom. В случае, если разработанные вами сценарии или их части будут применяться для обработки разных наборов данных, понимание этих тем и эффективное использование автосинхронизации сэкономит много времени.

Для экспериментов понадобятся два текстовых файла: **1 показатель** и **Несколько показателей**. Импортируем их в сценарий. Эти два файла отличаются только тем, что во втором добавлены столбцы с показателями 2-5 (рисунок 2.26).

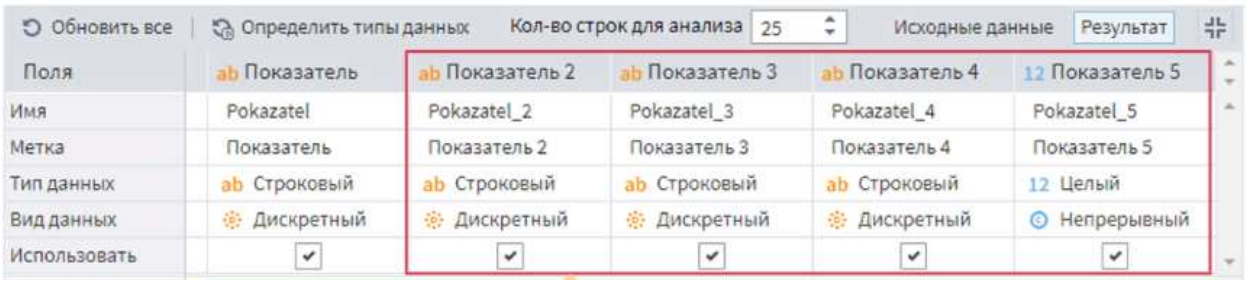


Рисунок 2.26 – Сравнение двух текстовых файлом

На первом этапе импорта все настройки разделителей устанавливаются из системного файла. Показатели вещественного типа определились неверно (рисунок 2.27). Попробуем воспользоваться алгоритмом автоматического, определения параметров импорта из импортируемого файла. Нажмем кнопку **«Определить автоматически»**.

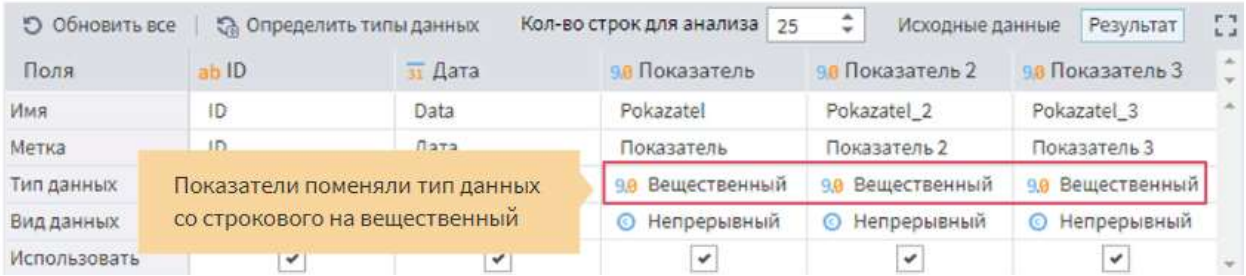


Рисунок 2.27 – Показатели вещественного типа

В Loginom существуют поля следующих **типов**:

- ***логический*** – данные в поле могут принимать только три значения: ***истина***, ***ложь*** и ***пусто***;
- ***дата/время*** – поле содержит данные типа дата/время;
- ***вещественный*** – числа с плавающей точкой;

- ***целый*** – целые числа;
- ***строковый*** – строки символов;
- ***переменный*** – в поле могут содержаться все вышеперечисленные типы данных одновременно.

В Loginom можно изменить **вид** данных поля (рисунок 2.28).

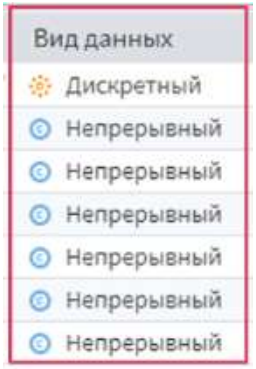


Рисунок 2.28 – Вид данных поля

Непрерывный – значения в столбце могут принимать любое значение в рамках своего типа. ***Дискретный*** – данные в столбце могут принимать ограниченное число значений. В контексте нашей задачи все поля непрерывные, кроме ID.

На рисунке 2.29 представлена страница мастера настройки соответствия между столбцами (или **мэппинг полей**). Мэппинг полей будет часто встречаться в мастерах узлов. Цель настройки – проставить соответствия между наборами входных и выходных полей.

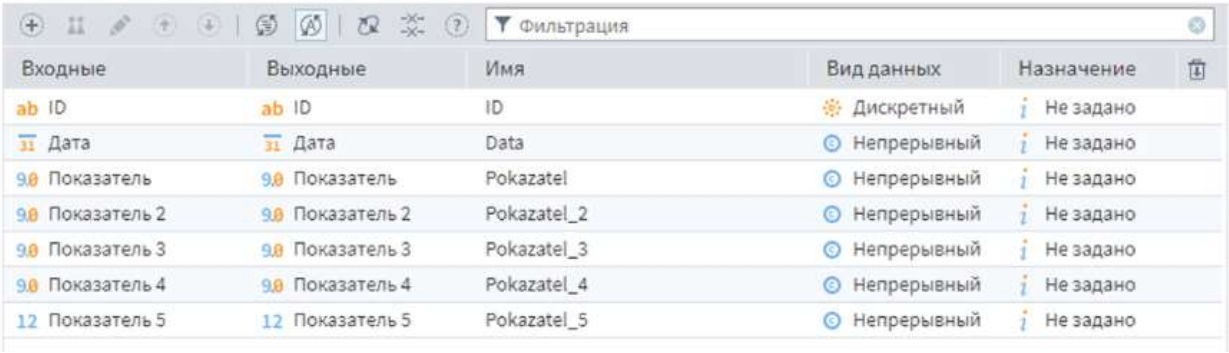


Рисунок 2.29 – Страница мэппинга полей

Рассмотрим пример, когда происходит мэппинг. Предположим, что есть узел, у которого имеется один входной и один выходной порты с набором данных. К его входному порту идет связь от другого узла. Тогда набор данных пройдет два этапа мэппинга полей: на входе и на выходе (рисунок 2.30). При этом на каждом этапе мэппинга количество операций сопоставления равняется

количеству портов. В каждой операции мэппинга полей левый список столбцов называется входным, а правый – выходным.

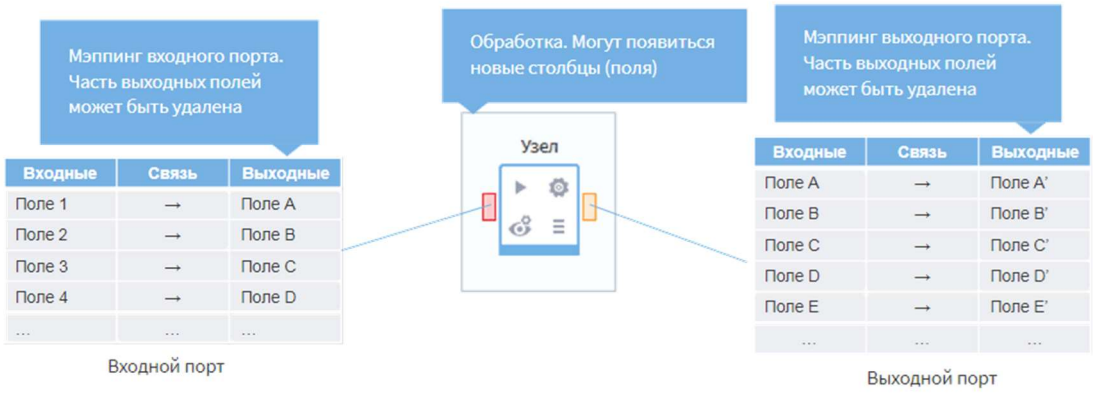


Рисунок 2.30 – Пример мэппинга полей

Иногда удобнее переключиться в режим представления связей. Для этого необходимо нажать кнопку **Связи**.

Одной командой можно удалить все имеющиеся связи входных и выходных полей операций мэппинга нажав на кнопку «Удалить все связи». Обратной операцией является автоматическое связывание.

Алгоритм связывания пытается связать выходные поля с входными, у которых нет связи, сначала по паре признаков **Тип данных-Имя поля**, затем – по паре **Тип данных-Метка** (рисунок 2.31).

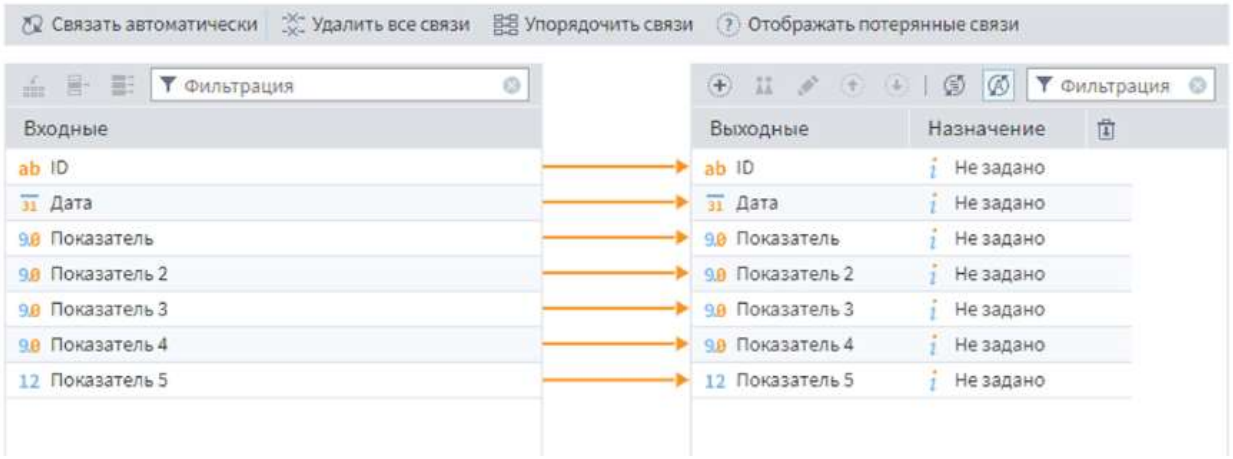


Рисунок 2.31 – Пример работы алгоритма связывания

Мэппинг считается выполненным успешно, если у каждого выходного поля имеется связь с каким-либо входным полем. В противном случае при попытке перейти на следующий шаг появится красная рамка и предупреждение об ошибке. Однако, мастер настройки можно завершить, сохранив настройки.

Красная индикация выходного порта будет говорить о том, что мэппинг полей неуспешен, выходной порт не сконфигурирован, следовательно, выполнить узел не удастся.

Вернемся в настройку соответствия полей. Вместо назначения связи мы можем удалить выходное поле, и оно не сможет участвовать в дальнейшей обработке. Один из способов восстановить удаленное поле – использовать команду **Создать выходной...** контекстного меню, вызываемого щелчком правой кнопки мыши на нужном входном поле (рисунок 2.32).

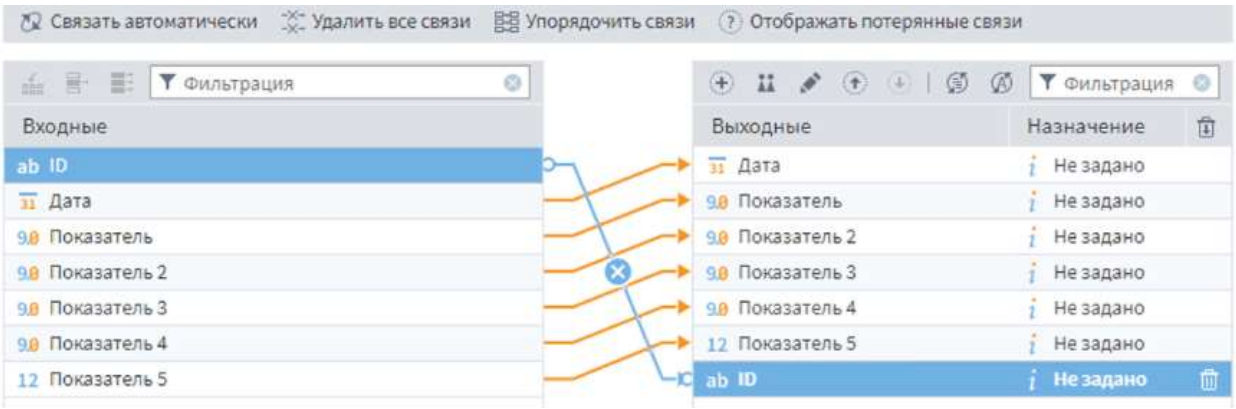


Рисунок 2.32 – Пример пересечений связей

Сразу продемонстрируем полезную команду **Упорядочить связи** – поля списков пересортировываются для устранения пересечений связей.

Второй способ восстановить удаленное выходное поле – через команду **Добавить столбец**. Но данный способ требует больше действий: понадобится вручную ввести нужное имя, метку и тип данных столбца, после чего установить связь.

Кружок в середине порта свидетельствует о том что у порта отключена автосинхронизация (рисунок 2.33).

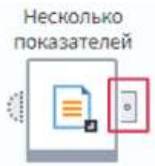


Рисунок 2.33 – Пример отключения автосинхронизации

Начнем эксперименты с набором данных **1 показатель**. Экспортируем его в файл формата **lgd**. Для этого активируем узел и добавим в область построения еще один – экспорт в **Loginom Data File**. Соединим выходной и входной порты.

Обратите внимание: входной порт сам сконфигурировался, то есть мэппинг полей порта прошел успешно (иначе порт был бы красного цвета). Это произошло потому, что отработала операция синхронизации полей.

Синхронизация входных и выходных полей – это однократный запуск специальной последовательности действий с целью мэппинга входных и выходных полей без участия пользователя. Алгоритм следующий:

1. Попытка автоматического связывания входных и выходных полей.
2. Добавление полей в список выходных на основе списка входных и их автоматическое связывание.
3. Обновление списка выходных полей.

При самом первом мэппинге, когда список полей входного порта пустой, выполняется только пункт 2.

Синхронизация запускается каждый раз, когда:

- проводится соединение связью портов двух узлов, а список полей входного порта пустой;
- вызывается команда **Синхронизация** в мастере настройки порта;
- открывается страница мастера настройки соответствия столбцов, и в порте включена автосинхронизация;
- узел активируется, и в порте включена автосинхронизация.

При переходе в мастер настройки узла **Loginom Data File** можно увидеть, что отработала операция синхронизации, в результате которой появилось три выходных поля, идентичных входным, и проставились связи между ними (рисунок 2.34).

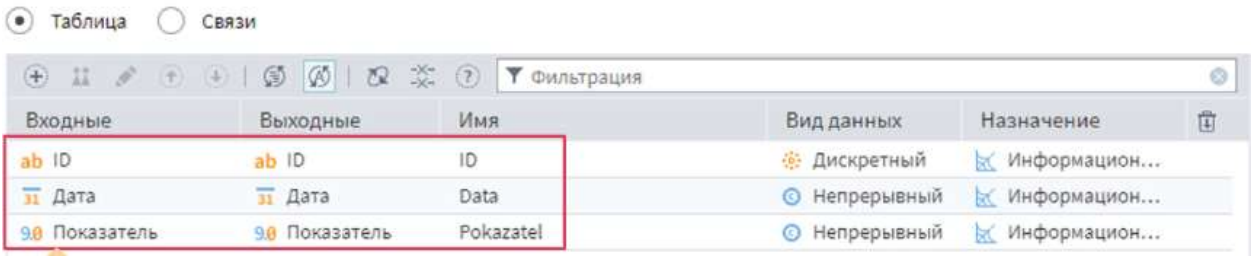


Рисунок 2.34 – Настройка соответствия между столбцами в узле Loginom Data File

Кроме того, мы наблюдаем, что включена кнопка **Автоматическая синхронизация**. Значит, если набор полей, который «поступит» во входной порт, изменится, список выходных полей порта будет синхронизирован автоматически безучастия пользователя.

Посмотрим, что получится, если изменится структура импортируемого файла. Создадим связь порта узла экспорта с выходным портом файла **Несколько показателей**.

Так как у нас включена автосинхронизация входного порта, мэппинг полей отработал, и порт успешно сконфигурировался. Убедимся в этом, зайдя в настройку порта.

Видим, что список выходных полей порта обновился и расширился – в нем есть все новые поля набора данных **Несколько показателей**. Таким образом, при изменении структуры входного набора данных ничего перенастраивать непришлось – нам помогла автосинхронизация полей.

Выключим автоматическую синхронизацию, сохраним изменения и вернем связь с наборам данных **1 показатель**.

При попытке активировать узел будет выдана ошибка «Порт не настроен». Зайдем в настройку порта узла экспорта.

После отключения автосинхронизации список полей на входе стал фиксированным: узлу требуются четыре поля, которые были в «старом» наборе данных, но которых нет в новом. Это так называемые «потерянные связи» (рисунок 2.35).

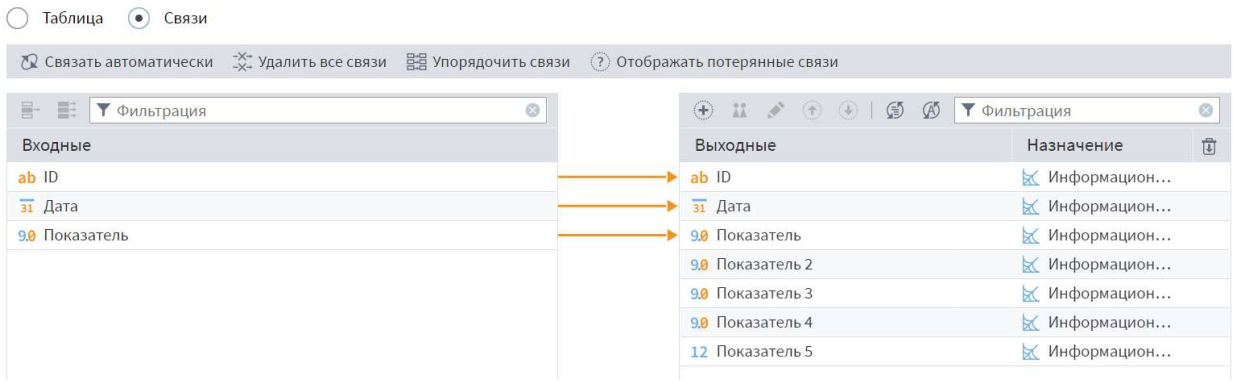


Рисунок 2.35 – Пример потерянных связей

Отобразить потерянные связи можно нажав на кнопку **Отображать потерянные связи**.

Чтобы разрешить конфликт мэппинга и корректно настроить порт, придется удалить «лишние» поля. Для этого нажмем на кнопку **Автоматическая синхронизация**.

2.8. Переменные и параметризация

Переменные используются в Loginom с целью создания динамических сценариев.

Переменная – это объект, который может содержать только одно значение. Она имеет те же параметры, что и поле набора данных: Имя, Метку, Тип данных и Значение. При открытии пакета в Loginom в области построения сценария имеется пять портов переменных – **Переменные сценария** (рисунок 2.36):

- ***Переменные*** – порт содержит все переменные сценария с портов 2-5.
- ***Переменные системы*** – переменные окружения операционной системы, считываются из ее настроек.
- ***Переменные сессии*** – содержат сведения о текущей сессии, их значения существуют, пока открыт пакет.
- ***Переменные пакета*** – содержат информацию о текущем пакете – имя, идентификатор, расположение и так далее.
- ***Переменные пользователя*** – переменные, созданные пользователем.

Рисунок 2.36 – Переменные сценария

Соблюдать уникальность имен переменных обязательно только в пределах одного порта. Если в двух разных портах имеются переменные с одним и тем же именем, избегать конфликтов помогает **приоритет**. Самый высокий приоритет имеют **Переменные пользователя**. Далее приоритет уменьшается в порядке следования портов снизу вверх.

Рассмотрим пример. Активируем порты и посмотрим, какие переменные находятся в порте **Переменные пакета** (рисунок 2.37)

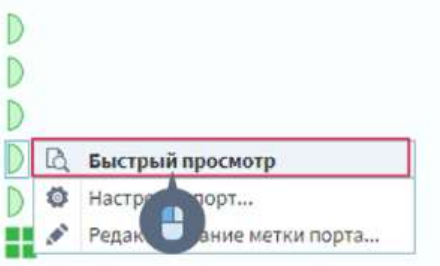


Рисунок 2.37 – Быстрый просмотр порта Переменные пакета

Данный порт содержит информацию о пакете: его имя, версию, идентификатор и так далее. Используем в нашем примере переменную **PackageName**. Обратим внимание на ее значение. Именно оно на данный момент указано для этой переменной в порте **Переменные** (рисунок 2.38).

Переменные | Переменные системы | Переменные сессии | Переменные пакета | Переменные пользователя

№	Имя	Метка	Значение
1	ab PackageName	PackageName	Package1
2	ab PackageVersion	PackageVersion	1.0.0
3	ab PackageGuid	PackageGuid	{3C9DDD4B-C5F8-458C-9204-8A4AF31F3B0B}
4	ab PackageFileName	PackageFileName	
5	ab PackageFilePath	PackageFilePath	
6	ab DerivedPackageName	DerivedPackageName	Package1
7	ab DerivedPackageVersion	DerivedPackageVersion	1.0.0
8	ab DerivedPackageGuid	DerivedPackageGuid	{3C9DDD4B-C5F8-458C-9204-8A4AF31F3B0B}
9	ab DerivedPackageFileName	DerivedPackageFileName	
10	ab DerivedPackageFilePath	DerivedPackageFilePath	

Рисунок 2.38 – Переменная PackageName

Создадим переменную пользователя **PackageName**. Для этого можно щелкнуть на порте **Переменные пользователя** левой кнопкой мыши и нажать **Настроить порт** или щелкнуть на **Переменные сценария** и нажать кнопку **Настройка**.

Создадим переменную и заполним параметры, как показано на рисунке 2.39.

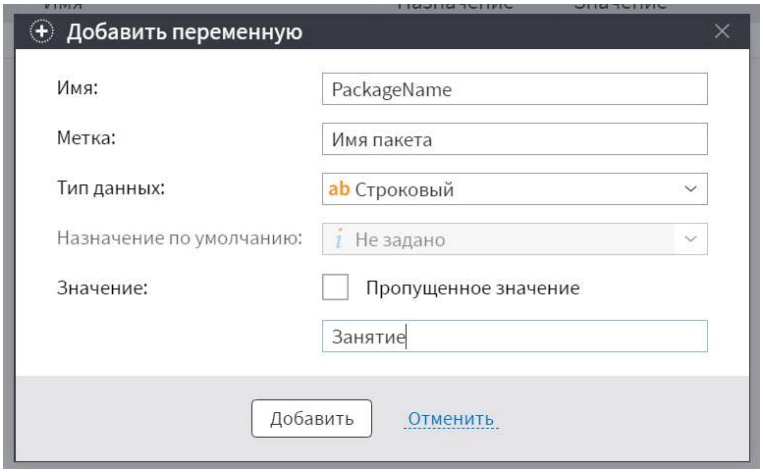


Рисунок 2.39 – Создание переменной PackageName

Активируем порты и откроем Быстрый просмотр порта **Переменные**.

38	ab	windir	windir	C:\WINDOWS
39	ab	ProductVersion	ProductVersion	6.5.3
40	ab	ProductEdition	ProductEdition	Community
41	ab	SessionGuid	SessionGuid	{9E5344AC-5897-41FA-9040-D34C90A929FE}
42	ab	RequestId	RequestId	*<null>*
43	ab	PackageName	Имя пакета	Занятие
44	ab	PackageVersion	PackageVersion	1.0.0
45	ab	PackageGuid	PackageGuid	{3C9DDD4B-C5F8-458C-9204-8A4AF31F3B0B}
46	ab	PackageFileName	PackageFileName	
47	ab	PackageFilePath	PackageFilePath	

Рисунок 2.39 – Быстрый просмотр порта Переменные

Теперь для переменной PackageName здесь отображаются метка и значение, которые мы указали в переменных пользователя. Однако ее значение не изменилось в порте **Переменные пакета**. Для того, чтобы в этом убедиться можно перейти по вкладке **Переменные пакета**.

Подача переменных на порт **Входные переменные** позволяет работать с ними, как с полями набора данных: обрабатывать их и использовать для расчетов. Такой порт есть у компонента **Калькулятор** и специальных компонентов для работы с переменными.

Для передачи переменных в узел сценария используются входные порты узлов. Переменные можно добавить следующими способами:

- передать с одного из портов переменных сценария;
- создать непосредственно на входном порте нужного узла;
- передать с выходного порта переменных любого другого узла.

Кроме входных переменных также в Loginom существуют и **управляющие переменные**. Они содержат значения параметров настроек мастера какого-либо узла, что позволяет более гибко реализовать логику обработки данных.

Например, можно организовать ограниченный перебор какого-либо параметра алгоритма по критерию оптимальности, к примеру, ошибки прогноза.

Если хотя бы для одной из управляющих переменных задано значение, то в мастере настройки узла возле параметра, значение которого можно задавать с помощью переменных, появляется переключатель:

-  значение параметра задается вручную;
-  в значение параметра задается переменной. В этом случае в поле параметра необходимо выбрать переменную, содержащую его значение.

Порт управляющих переменных имеется у каждого узла. У узлов импорта и экспорта он отображается всегда. Для прочих компонентов порт по умолчанию скрыт, и его необходимо отобразить с помощью соответствующей команды контекстного меню (рисунок 2.40).

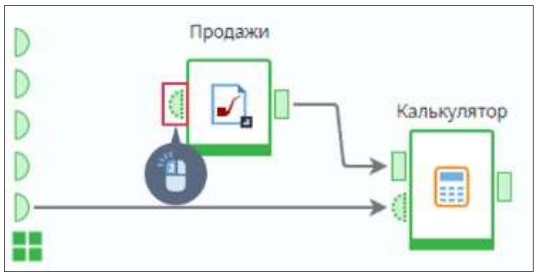


Рисунок 2.40 – Пример управляющей переменной

Существует пять специальных компонентов, предназначенных для работы с переменными:

- **Замена (переменные).** Заменяет значения выбранных переменных на значения из таблицы замен.
- **Калькулятор (переменные)**. Позволяет создать новые переменные в соответствии с введенными формулами. Компонент **Калькулятор (переменные)** поддерживает тот же набор функций и те же правила составления выражений, что и компонент **Калькулятор** раздела **Трансформация**.
- **Переменные в таблицу**. Преобразует набор переменных в таблицу.
- **Соединение (переменные)**. Объединяет два или более набора переменных в один. Может использоваться для объединения переменных с двух разных ветвей узла **Условие**.
- **Таблица в переменные**. Преобразует табличные данные в переменные. При преобразовании данные агрегируются выбранным способом.

2.9. Лабораторная работа №2 «Переменные»

Импортируем файл **Продажи.lgd**, в котором содержатся суммы последних покупок клиентов магазина бытовой техники и перейдем в настройку переменных сценария.

Создадим переменную **Скидка** с параметрами, как показано на рисунке 2.41.

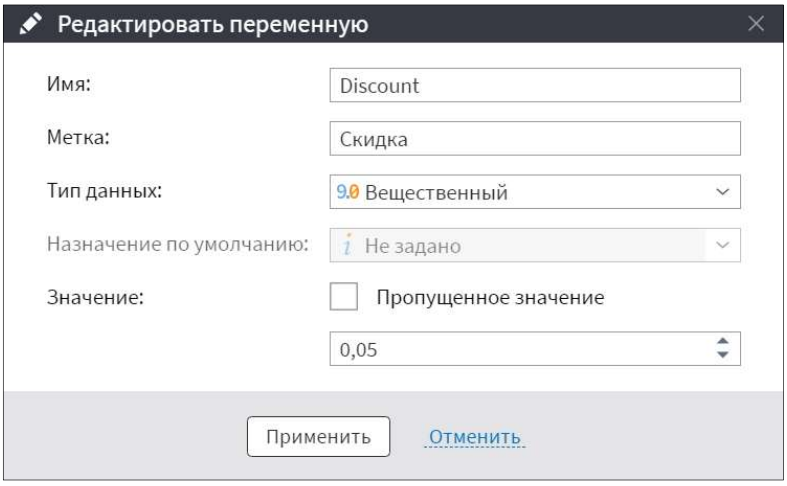


Рисунок 2.41 – Параметры переменной Скидка

Подадим набор данных на порт данных узла **Калькулятор**, а переменные пользователя – на порт переменных и откроем настройки этого порта (рисунок 2.42).

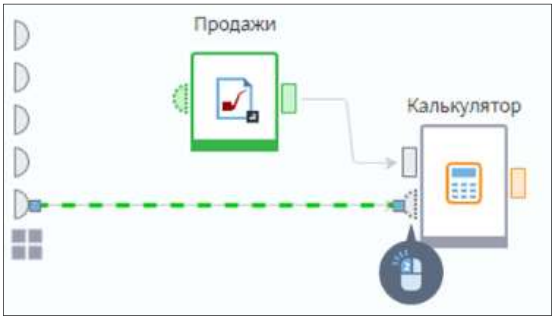


Рисунок 2.42 – Пример установление связей с узлом Калькулятор

Перейдем к настройке узла **Калькулятор** и зададим параметры выражения **DiscountSum** и **Сумма скидки**. Поданные на вход переменные отобразились в соответствующей области в отдельном разделе **Переменные**.

В область кода выражения запишем формулу **IFF(Amonut>1000,Amount*Discount,0)**, обозначающую:

- если значение в поле Сумма больше 1000, то сумма скидки равна произведению суммы покупки на величину скидки,
- иначе сумма скидки равна 0.

Отметит, что входная переменная используется в формуле в явном виде также, как и поля набора данных. Перед тем как завершить работу с формулой, проверим выполнения условия в **Предпросмотре**.

Далее добавил узел **Фильтр строк** и нажмем в его контекстном меню команду **Показать порт управляющих переменных** (рисунок 2.43).

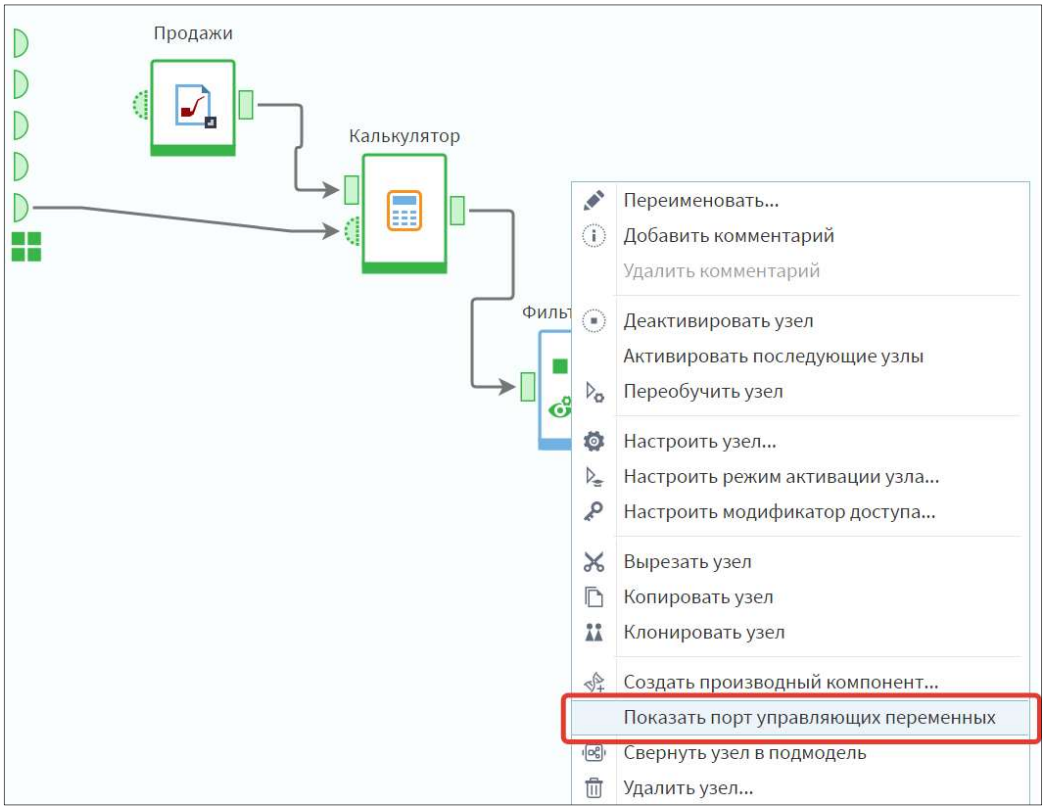


Рисунок 2.43 – Вызов команды Показать порт управляющих переменных

Порт отобразится под портом входного набора данных узла. Зайдем в настройку порта переменных (рисунок 2.44).

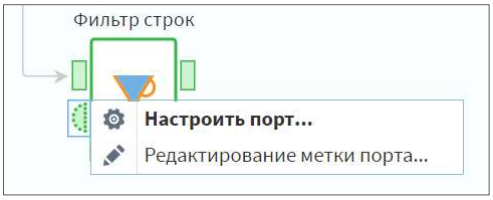


Рисунок 2.44 – Вызов команды Настроить порт у порта управляющих переменных

Создадим управляющую переменную с параметрами, которые представлены на рисунке 2.45. Согласно данным параметрам, нас интересуют клиенты с суммой скидки от 1 000 рублей.

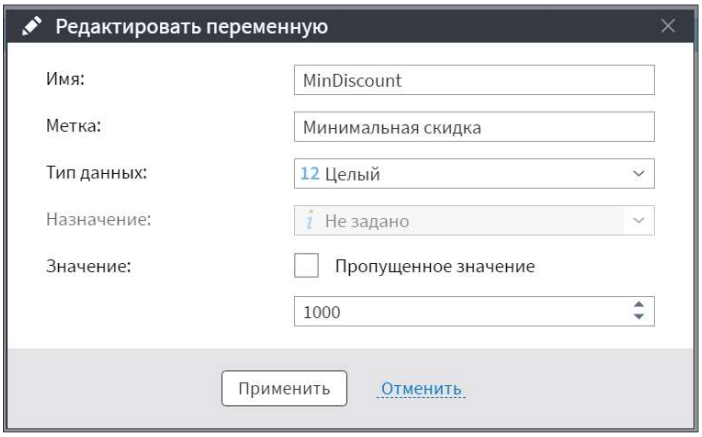


Рисунок 2.45 – Параметры управляющей переменной

Перейдем в настройку узла **Фильтр строк**. После задания параметров **Поле** и **Условие** у нас появился переключатель возле поля **Значение для сравнения** (рисунок 2.46). Нажнем его.

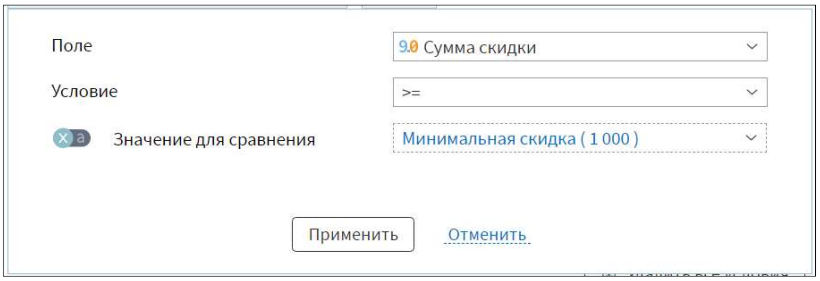


Рисунок 2.46 – Переключатель возле поля Значение для сравнения

После сохранения настроек и запуска обработки на входе **Соответствуют условию** оказались только клиенты с соответствующей суммой.

Далее добавим узел **Таблица в переменные** в сценарий и перейдем в настройку (рисунок 2.47).

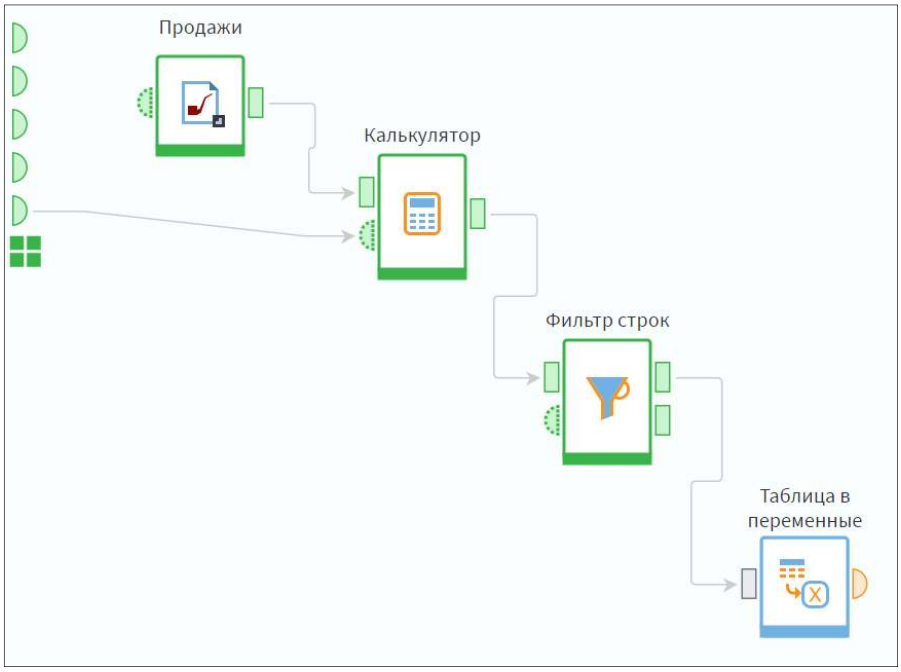


Рисунок 2.47 – Добавление узла **Таблица в переменные**

Выберем поле **Сумма скидки** и нажнем **Переместить в Переменные**.

Выбранное поле переместилось в список **Переменные** с вариантом агрегации **Сумма**, который используется по умолчанию для числовых и вещественных полей. Дважды щелкнем по полю левой кнопкой мыши для редактирования варианта агрегации. Снимаем галочку с варианта **Сумма**, выбираем **Количество** и **Максимум** и нажимаем кнопку **Применить**.

Мы получили две переменные по количеству агрегированных значений поля (их имена и метки изменили в выходном порте узла). Теперь их можно передать на вход переменных любого узла (рисунок 2.48).

Выходные переменные

№	Имя	Метка	Значение
1	12 DiscountSum_Count	Сумма скидки|Количество	10
2	9.0 DiscountSum_Max	Сумма скидки|Максимум	2 076,60

Рисунок 2.48 – Переменные Количество и Максимум

Далее создадим в области построения узел **Соединение (Переменные)** (рисунок 2.49). Он предназначен для объединения двух или более наборов переменных в один. У данного компонента динамическое количество портов. По умолчанию узел содержит два входных порта: **Входные переменные** и **Добавляемые переменные**. При необходимости можно добавить еще порты с помощью кнопки **Добавить еще один порт**.

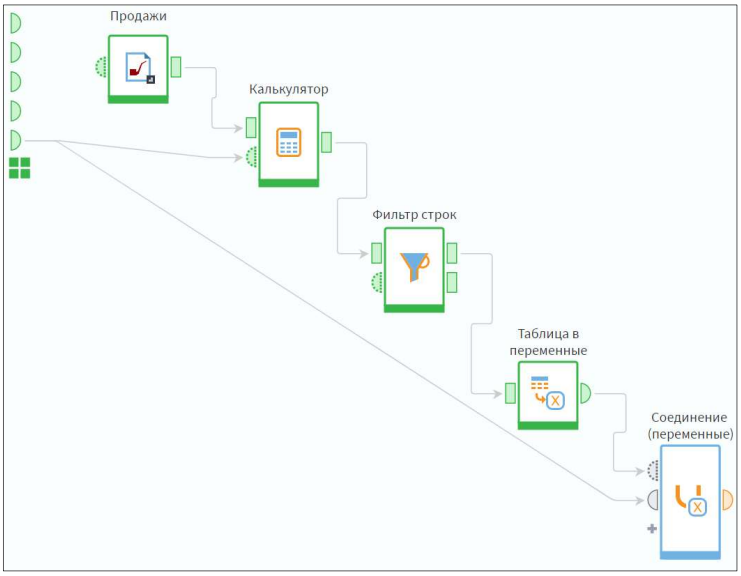


Рисунок 2.49 – Добавление узла Соединение (Переменные)

Создадим связи между портами, как показано на рисунке 2.50. В процессе обработки набор переменных, поданный на порт **Входные переменные**, будет дополнен записями с порта **Добавляемые переменные**. Перейдем в настройку узла и устанавливаем соответствие между переменными.

В результате обработки получаем набор, где содержатся все созданные нами переменные.

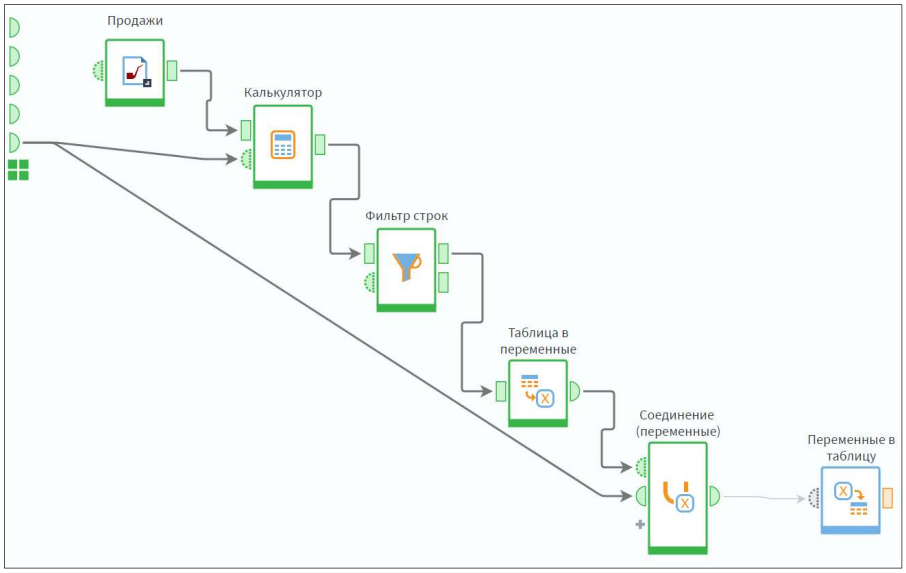


Рисунок 2.50 – Добавление узла Переменные в таблицу

Далее добавим компонент **Переменные в таблицу** для того, чтобы перевести созданные нами переменные в структурированный вид (рисунок 2.50).

2.10. Компоненты Условия и Замена

Компонент **Условие** предназначен для ветвления логики обработки данных в сценарии. В зависимости от заданных условий будет выполняться та или иная ветвь сценария, а прочие ветви – блокироваться. Использование компонента позволяет создавать гибкие сценарии обработки данных.

В общем случае условие используется, когда необходимо обработать набор данных (часть набора данных) или переменных одним из возможных способов.

Рассмотрим три кейса использования:

Метод расчета. Частным случаем является ситуация, когда необходимо провести расчеты одним из нескольких методов в зависимости от заданного параметра. В этом случае данные выходных портов условия передаются в соответствующие ветви сценария. Если в дальнейшем, независимо от метода, данные нужно передать на один порт, можно воспользоваться компонентом **Объединение**.

- **Опциональный этап**. Другим частным случаем является необязательность некоторого этапа обработки данных: в зависимости от условий он может выполняться, либо не выполняться.
- **Параметры**. Узел **Условие** используется и в работе с переменными. Например, можно изменить значение некоторых переменных из исходного набора и сформировать новый набор, соответствующий заданным условиям.

Ненастроенный узел **Условие** не имеет входных и выходных портов. Чтобы задать условия ветвления набора данных или переменных, необходимо настроить узел.

Входные порты создаются с помощью команды **Добавить**. В условиях, определяющих ветвление, могут использоваться и переменные, и поля таблиц, и дерево данных.

После завершения настройки у узла появились заданные нами выходные порты: **Таблица** и **Переменные** (рисунок 2.51).

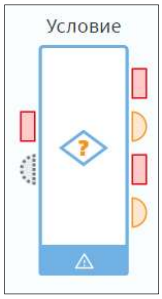


Рисунок 2.51 – Пример настроенного узла Условие

Так как мы отметили флажок **Передавать на выход** данные входных портов, а в списке ветвей у нас была только одна ветвь, у узла появилось два набора выходных портов: первый набор соответствует выполнению условия **Ветви 1**, второй – невыполнению ни одного из заданных условий (это ветвь Иначе).

Таким образом, количество наборов выходных портов всегда будет равняться количеству ветвей плюс один.

Компонент **Замена** используется для замены данных в исходном наборе с помощью таблиц замен. Таблицы замен представляют собой справочники, содержащие пары значений: те значения, которые необходимо заменить в исходных данных, и соответствующие им новые значения.

Эти таблицы могут быть подготовлены заранее и подаваться на вход узла, либо формироваться вручную внутри него.

2.11. Лабораторная работа №3 «Условие и Замена»

Рассмотрим использование компонента **Условие** на конкретном примере. Пусть есть набор исходных данных вида **Идентификатор-Сумма, руб.** В зависимости от значения входной переменной необходимо провести обработку данных одним из способов:

1. Вывести 13% от первоначальных сумм;
2. Вывести все суммы не менее 20 000 руб.;
3. Вывести все суммы менее 17 000 руб.

Добавим узел **Условие** и настроим его. На вход узла Условие подаем набор исходных данных и переменную, которая будет определять выполнение той или иной ветви сценария (рисунок 2.52).

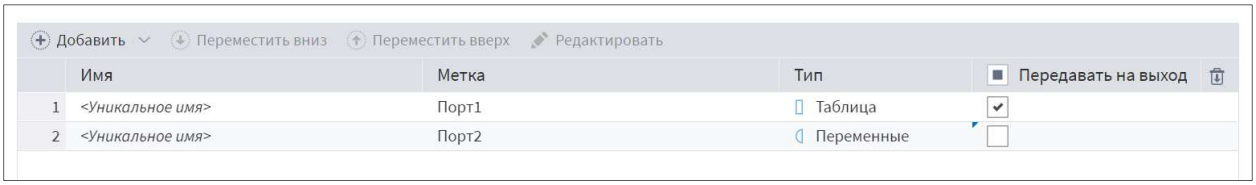


Рисунок 2.52 – Настройка узла Условие

Добавим в список все ветви нашего сценария (рисунок 2.53). Условия для каждой ветви настроем позже.

Рисунок 2.53 – Настройка ветвей сценария узла Условие

Поскольку мы передаем на выход таблицу исходных данных, у узла появилось три выходных порта, соответствующих заданным ветвям, и четвертый порт, соответствующий ветви **Иначе** (рисунок 2.54).

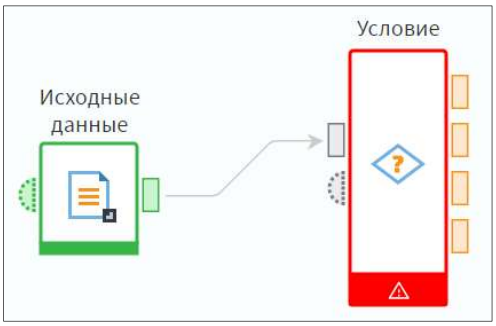


Рисунок 2.54 – Связь между узлами Исходные данные и Условие

Зайдем в настройки порта переменных. Добавим переменную с пропущенным значением как показано на рисунке 2.55.

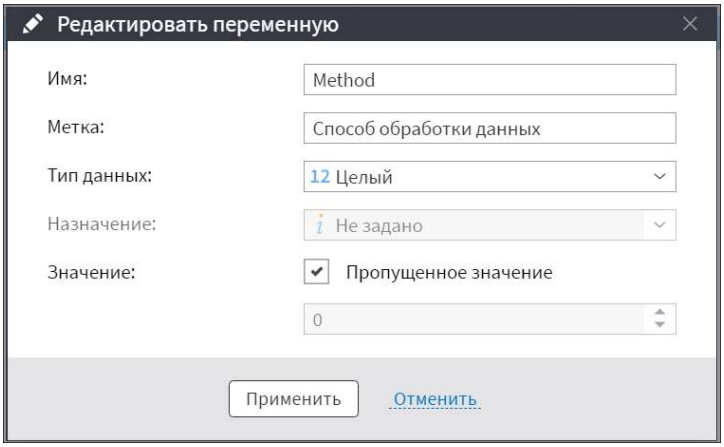


Рисунок 2.55 – Параметры переменной

Далее зададим условие для первой ветви. Она выполняется, когда значение переменной **Способ обработки данных** равно **1** (рисунок 2.56).

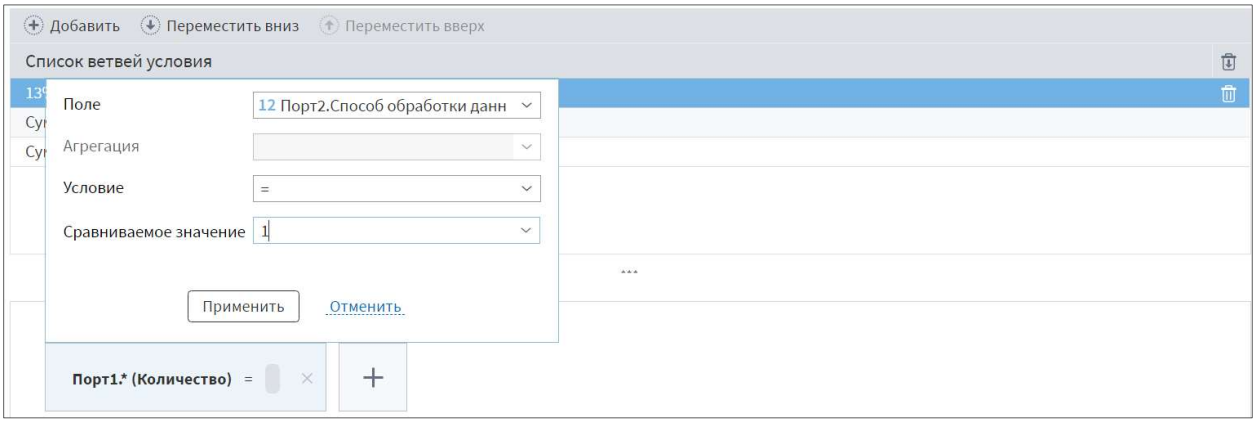


Рисунок 2.56 – Настройка первой ветви в узле Условие

Аналогичным образом зададим условие для оставшихся ветвей (рисунок 2.57). Поставим галочку **Режим отладки**.

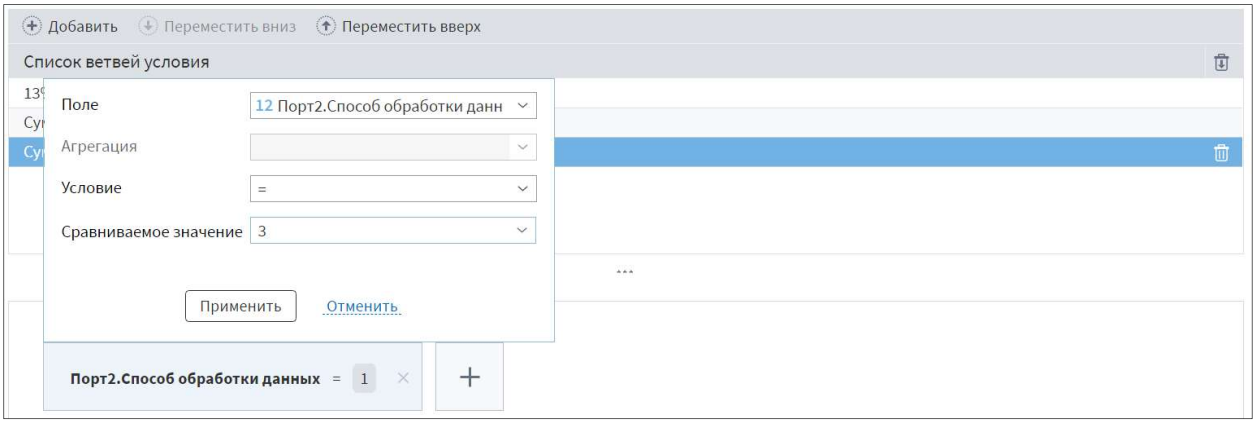


Рисунок 2.57 – Настройка оставшихся ветвей в узле Условие

Продолжим сценарий и добавим ветви обработки данных (рисунок 2.58).

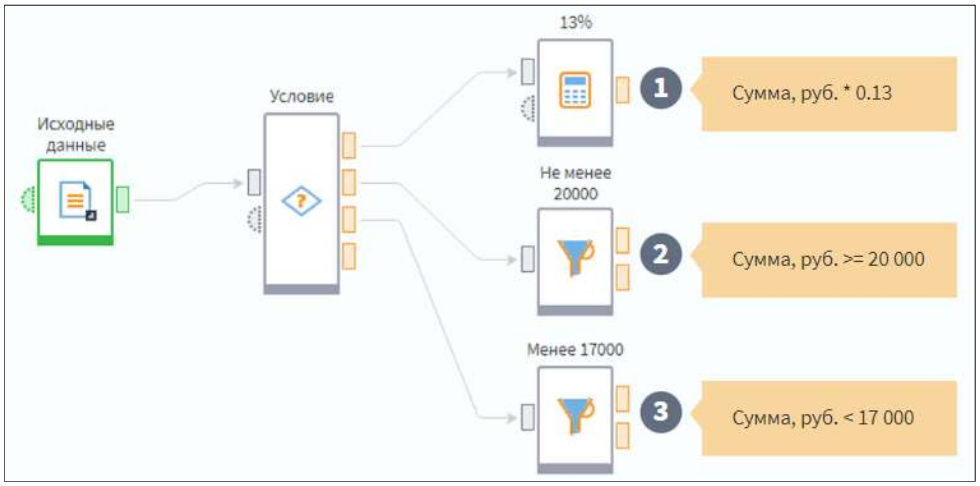


Рисунок 2.58 – Пример добавление ветвей обработки данных

Зададим переменной **Способ обработки данных** значение **1** и посмотрим, как работает сценарий. В соответствии с заданным условием, обработка данных проводится по первой ветви: рассчитались 13% от первоначальных сумм. Другие ветви не выполняются. Просмотрим полученный набор данных.

В поле **Сумма*0,13** выведены 13% от первоначальных сумм (рисунок 2.59).

#	9.0 Сумма*0,13	12 Идентификатор	12 Сумма, руб.
1	2 340,00	1	18 000
2	2 535,00	2	19 500
3	2 600,00	3	20 000
4	3 120,00	4	24 000
5	3 757,00	5	28 900
6	1 950,00	6	15 000
7	4 550,00	7	35 000
8	2 730,00	8	21 000
9	2 912,00	9	22 400
10	5 850,00	10	45 000
11	2 366,00	11	18 200
12	2 678,00	12	20 600
13	2 795,00	13	21 500
14	3 250,00	14	25 000
15	4 290,00	15	33 000
16	4 251,00	16	32 700

Рисунок 2.59 – Полученные данные на выходном порте узла Калькулятор

Если изменить значение переменной **Способ обработки данных** на **3**, будет выполняться третья ветвь сценария, а в полученный набор данных, соответствующих условию, войдут только суммы менее 17 000 руб. (рисунок 2.60).

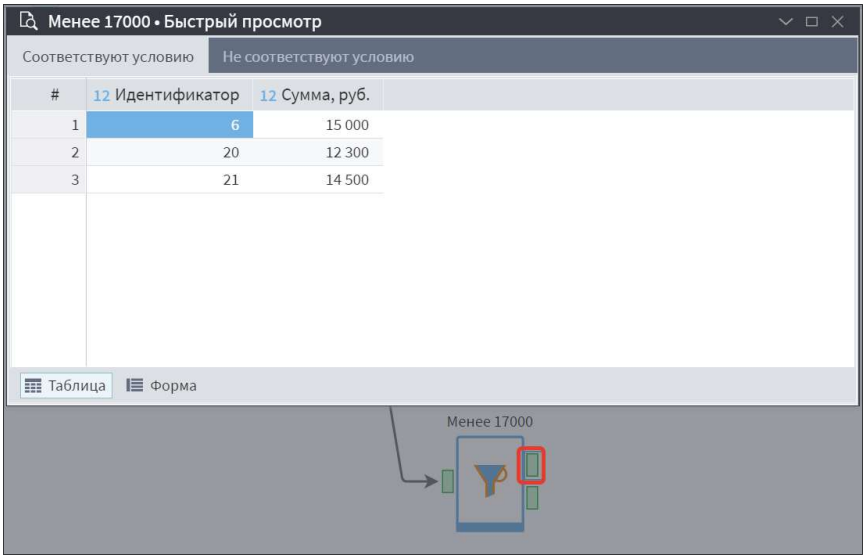


Рисунок 2.60 – Выходной порт узла Фильтр строк

Далее рассмотрим работу компонента **Замена**. Сначала рассмотрим случай, когда используется внешняя таблица замен, подаваемая на вход узла, где необходимо заменить в исходном наборе данных кратные названия месяцев на полные (рисунок 2.61).

Исходные данные • Быстрый просмотр

Набор данных

#	ab Идентификатор	ab Месяц регистрац…
1	01	Окт
2	02	Июнь
3	03	Ноябрь
4	04	Авг
5	05	Янв
6	06	Июль
7	07	Сен
8	08	Дек
9	09	Февраль
10	10	Март
11	11	Октябрь
12	12	Апр
13	13	Июн
14	14	Май
15	15	Апрель
16	16	Июл
17	17	Декабрь
18	18	Фев
19	19	Мар
20	20	Сентябрь

Рисунок 2.61 – Пример сходного набора данных

Внешняя таблица замен содержит пары значений: **Кратное название месяца-Полное название месяца** (рисунок 2.62).

Таблица замен • Быстрый просмотр

Набор данных

#	ab Краткое назван…	ab Полное назван…
1	Янв	Январь
2	Фев	Февраль
3	Мар	Март
4	Апр	Апрель
5	Май	Май
6	Июн	Июнь
7	Июл	Июль
8	Авг	Август
9	Сен	Сентябрь
10	Окт	Октябрь
11	Ноя	Ноябрь
12	Дек	Декабрь

Рисунок 2.62 – Пример данных внешней таблицы замен

Подключим наборы данных к соответствующим входным портам узла **Замена** и настроем его (рисунок 2.63).

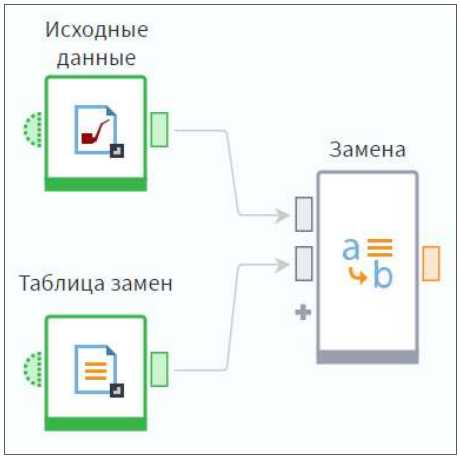


Рисунок 2.63 – Создание узла Замена

Для поля **Месяц регистрации** выберем способ замены **Таблица замен** (рисунок 2.64).

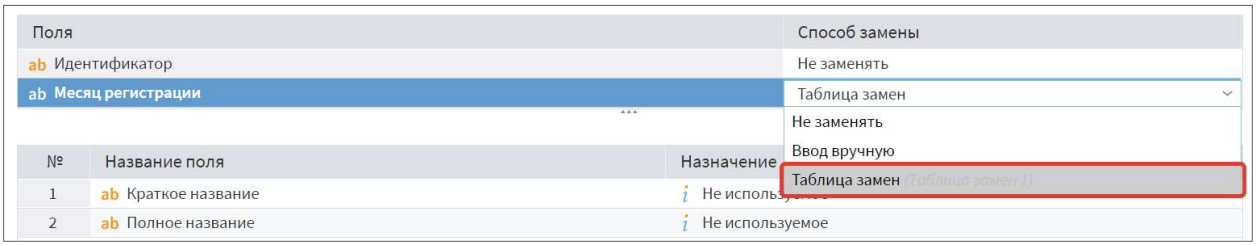


Рисунок 2.64 – Способ замены Таблица замен

Для каждого поля исходных данных можно настроить **Способ замены**:

- **Не заменять**;
- **Ввод вручную** – в этом случае формируем внутреннюю таблицу замен;
- **Таблица замен** – используем внешнюю таблицу замен.

Поля внешней таблицы могут принимать следующие значения:

- **Не используемое**;
- **Значение** – это заменяемые значения;
- **Замена** – это новые значения;
- **Информационное**.

Поле **Краткое название** установим как, заменяемые значения, поле **Полное название** – как новые значения (рисунок 2.65).

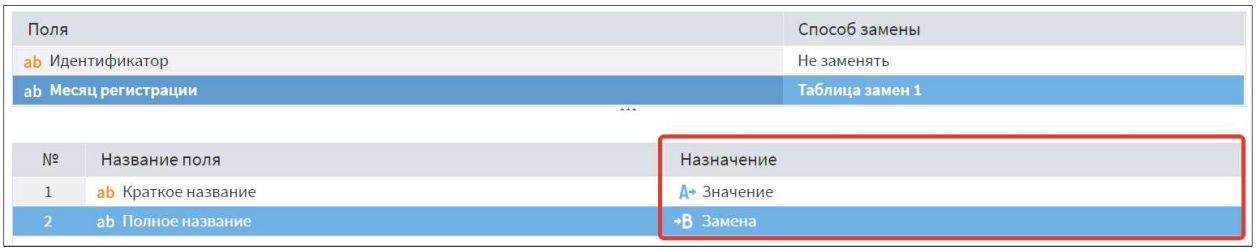


Рисунок 2.65 – Настройка полей в узле Замена

На втором шаге мастера по умолчанию используется вариант соответствия столбцов **Замена**: столбец с исходными значениями будет заменен на столбец с новыми значениями.

Замена • Быстрый просмотр

Выходной набор данных

#	ab Идентификатор	ab Месяц регистрации Заме…	0/1 Месяц регистрации Замен…
1	01	Октябрь	true
2	02	Июнь	false
3	03	Ноябрь	false
4	04	Август	true
5	05	Январь	true
6	06	Июль	false
7	07	Сентябрь	true
8	08	Декабрь	true
9	09	Февраль	false
10	10	Март	false
11	11	Октябрь	false
12	12	Апрель	true
13	13	Июнь	true
14	14	Май	true
15	15	Апрель	false
16	16	Июль	true
17	17	Декабрь	false
18	18	Февраль	true
19	19	Март	true
20	20	Сентябрь	false

Рисунок 2.66 – Исходные данных на выходном порте узла Замена

В исходном наборе данных краткие названия месяцев замены на полные. Исходные значения не отображаются (рисунок 2.66). Для каждой записи было добавлено логическое поле: была ли осуществлена замена.

2.12. Подмодели

Подмодель – это компонент, который может содержать в себе последовательность узлов, реализующих заданную логику обработки данных.

Фрагменты сценария, представляющие определенный этап обработки данных, можно «свернуть» в подмодель, что позволит:

- упростить структуру сценария;
- многократно использовать этот фрагмент, в том числе в ссылках и циклах;
- создавать производные компоненты.

Как и другие компоненты Loginom, подмодель может иметь входные и/или выходные порты: табличные наборы данных, переменные или дерево данных. Количество портов не ограничено и настраивается пользователем.

В зависимости от функционала, у подмодели может не быть входных или выходных портов. Функционал подмодели определяется ее внутренней структурой.

Если подмодель не имеет входных портов, при ее деактивации внутренние узлы останутся активными. Чтобы избежать этого, необходимо использовать настройку порядка выполнения узлов.

Откроем файл **lg511_6_04.lgp**. Для того чтобы увидеть внутреннюю структуру подмодели, нажмем войти.

На рисунке 2.67 отображена внутренняя структура подмодели. В ее состав могут входить любые узлы Loginom, как стандартные, так и созданные на основе производных компонентов, а также другие подмодели. Уровни вложенности подмоделей не ограничены.

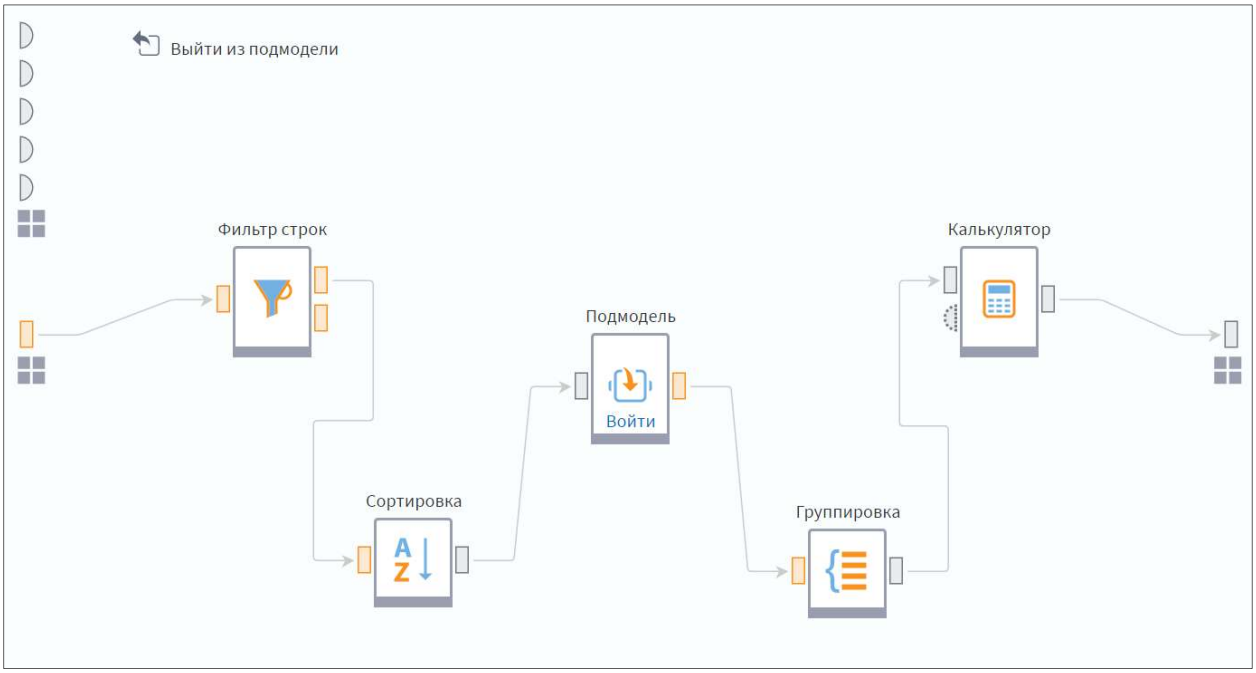


Рисунок 2.67 – Пример подмодели

Входной порт подмодели, доступен для быстрого просмотра данных, настройки и редактирования. Входной порт подмодели, доступен для быстрого настройки и редактирования

Для входного и выходного узлов подмодели доступны действия выполнения и остановки, настройки, добавления визуализаторов и прочее.

Выход из подмодели осуществляется с помощью кнопки **Выход из подмодели**.

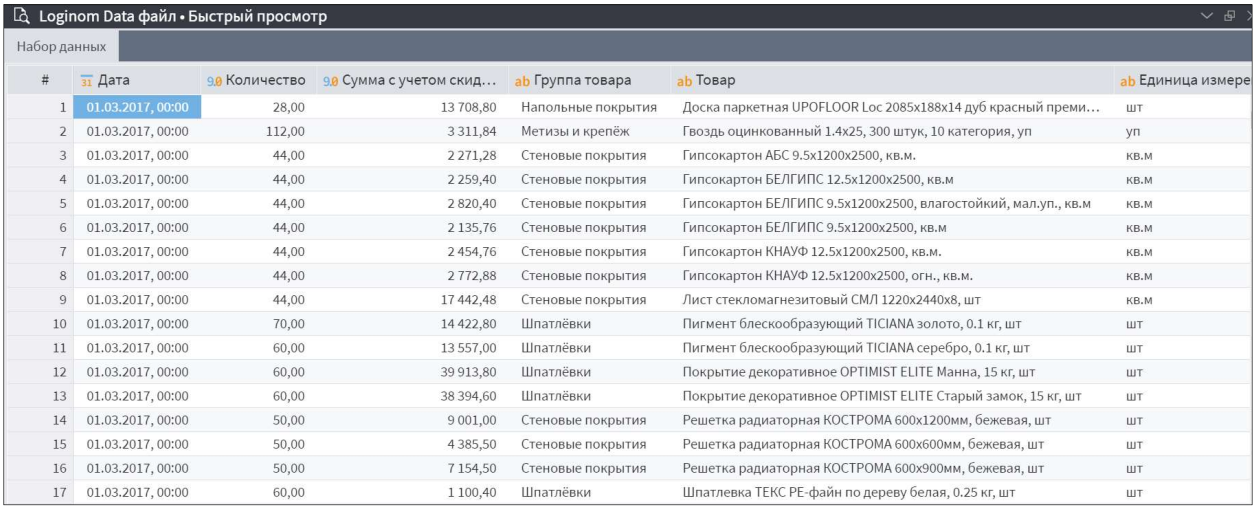
Возможные действия с подмоделью аналогичны действиям с прочими узлами (команды доступны и в контекстном меню). Дополнительной командой является **разворачивание подмодели** – восстановление заключенного в подмодель фрагмента сценария.

Существует два способа создания подмоделей:

- «сворачивание» в подмодель фрагмента существующего сценария;
- создание пустой подмодели и ее наполнение.

2.13. Лабораторная работа №4 «Подмодели»

Пусть дан набор данных о продажах товаров **sale.lgd** (рисунок 2.68).



Loginom Data файл • Быстрый просмотр

Набор данных

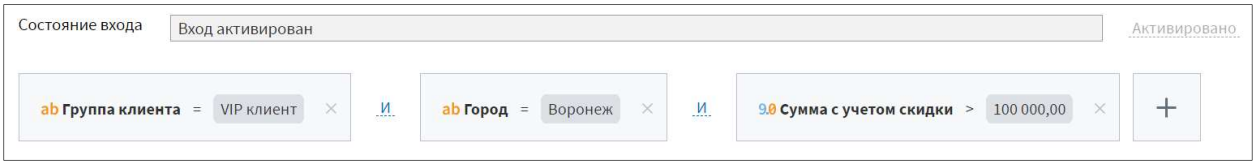
#	Дата	Количество	Сумма с учетом скид…	Группа товара	Товар	Единица измере
1	01.03.2017, 00:00	28,00	13 708,80	Напольные покрытия	Доска паркетная UPOFLOOR Loc 2085x188x14 дуб красный преми…	шт
2	01.03.2017, 00:00	112,00	3 311,84	Метизы и крепёж	Гвоздь оцинкованный 1.4x25, 300 штук, 10 категория, уп	уп
3	01.03.2017, 00:00	44,00	2 271,28	Стеновые покрытия	Гипсокартон АБС 9.5x1200x2500, кв.м.	кв.м
4	01.03.2017, 00:00	44,00	2 259,40	Стеновые покрытия	Гипсокартон БЕЛГИПС 12.5x1200x2500, кв.м	кв.м
5	01.03.2017, 00:00	44,00	2 820,40	Стеновые покрытия	Гипсокартон БЕЛГИПС 9.5x1200x2500, влагостойкий, мал.уп., кв.м	кв.м
6	01.03.2017, 00:00	44,00	2 135,76	Стеновые покрытия	Гипсокартон БЕЛГИПС 9.5x1200x2500, кв.м	кв.м
7	01.03.2017, 00:00	44,00	2 454,76	Стеновые покрытия	Гипсокартон КНАУФ 12.5x1200x2500, кв.м.	кв.м
8	01.03.2017, 00:00	44,00	2 772,88	Стеновые покрытия	Гипсокартон КНАУФ 12.5x1200x2500, огн., кв.м.	кв.м
9	01.03.2017, 00:00	44,00	17 442,48	Стеновые покрытия	Лист стекломагнезитовый СМЛ 1220x2440x8, шт	кв.м
10	01.03.2017, 00:00	70,00	14 422,80	Шпатлёвки	Пигмент блескообразующий TICIANA золото, 0.1 кг, шт	шт
11	01.03.2017, 00:00	60,00	13 557,00	Шпатлёвки	Пигмент блескообразующий TICIANA серебро, 0.1 кг, шт	шт
12	01.03.2017, 00:00	60,00	39 913,80	Шпатлёвки	Покрытие декоративное OPTIMIST ELITE Манна, 15 кг, шт	шт
13	01.03.2017, 00:00	60,00	38 394,60	Шпатлёвки	Покрытие декоративное OPTIMIST ELITE Старый замок, 15 кг, шт	шт
14	01.03.2017, 00:00	50,00	9 001,00	Стеновые покрытия	Решетка радиаторная КОСТРОМА 600x1200мм, бежевая, шт	шт
15	01.03.2017, 00:00	50,00	4 385,50	Стеновые покрытия	Решетка радиаторная КОСТРОМА 600x600мм, бежевая, шт	шт
16	01.03.2017, 00:00	50,00	7 154,50	Стеновые покрытия	Решетка радиаторная КОСТРОМА 600x900мм, бежевая, шт	шт
17	01.03.2017, 00:00	60,00	1 100,40	Шпатлёвки	Шпатлевка ТЕКС РЕ-файн по дереву белая, 0.25 кг, шт	шт

Рисунок 2.68 – Пример набора данных о продажах

Необходимо:

1. Выделить продажи VIP клиентам из города Воронеж на сумму более 100 000 рублей. Определить количество таких продаж, дату последней продажи и число дней, прошедших с этой даты.
2. Выделить продажи обычным клиентам из города Москва на сумму менее 100 рублей. Определить количество таких продаж, дату последней продажи и число дней, прошедших с этой даты.

Начнем решение первой задачи. С помощью узла **Фильтр строк** из исходного набора данных получим только записи о покупках VIP клиентов в городе Воронеж на сумму более 100 тыс. руб. Для этого в настройках фильтра установим три условия со связкой **И** (рисунок 2.69).



Состояние входа: Вход активирован — Активировано

ab Группа клиента = VIP клиент × И ab Город = Воронеж × И 9.0 Сумма с учетом скидки > 100 000,00 ×

Рисунок 2.69 – Условия в настройке узла Фильтр строк

Отсортируем полученные данные по дате в порядке возрастания с помощью узла **Сортировка**. Далее добавим определение количества записей через узел **Калькулятор** (рисунок 2.70).

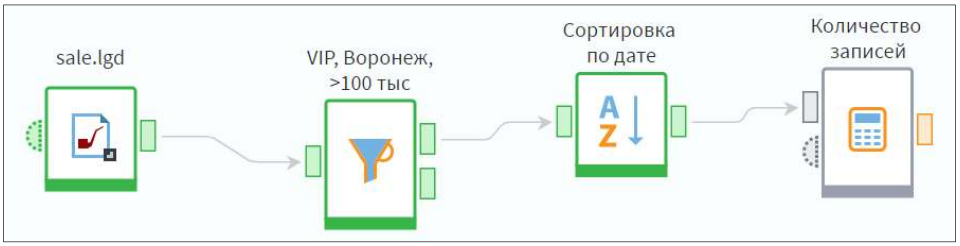


Рисунок 2.70 – Добавление узла Калькулятор в создаваемый сценарий

Создадим в настройках узла **Калькулятор** поле Количество записей (строк) с помощью функции **RowCount**. Далее преобразуем табличные значения в переменные с помощью компонента **Таблица в переменные**. Возьмем первое значение количества записей и максимальную дату (рисунок 2.71).

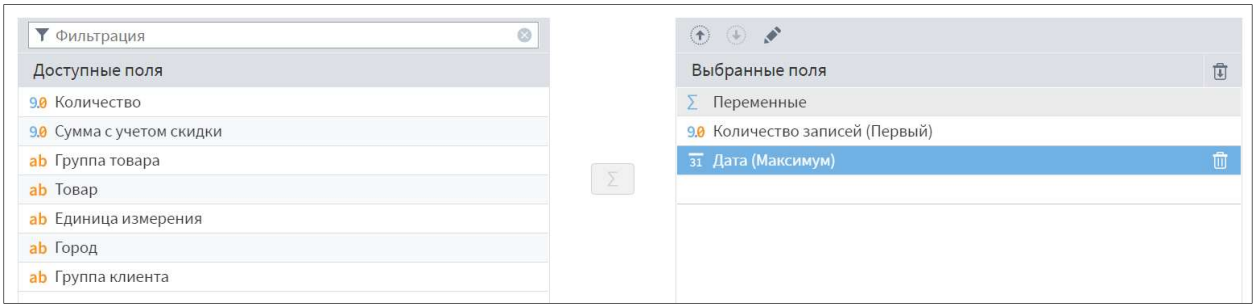


Рисунок 2.71 – Настройка узла Таблица в переменные

На выходе узла **Таблица в переменные** получены необходимые переменные.

Количество дней между текущей датой и датой последней покупки рассчитаем в **Калькуляторе переменных** (рисунок 2.72).

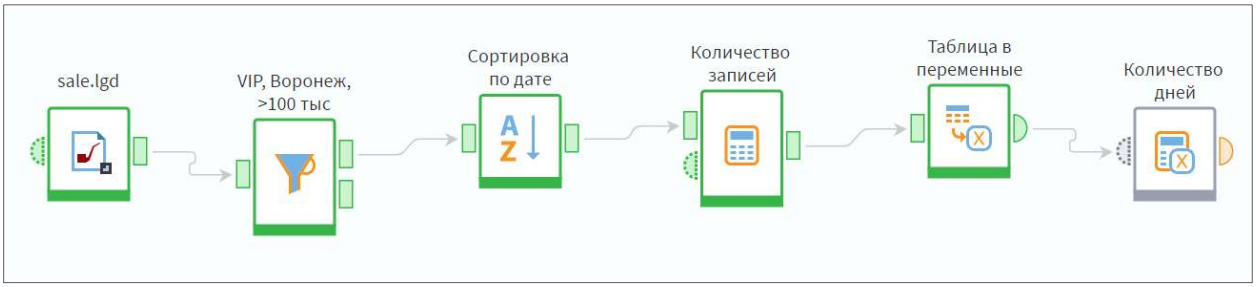


Рисунок 2.72 – Добавление узла компонента Калькулятор (переменные)

Добавим переменную **Количество дней**, рассчитанную с помощью функции **DaysBetween** (количество дней между датами) и **Today** (текущая дата).

Узлы сортировки и расчетов объединим в подмодель: так мы сможем повторно использовать их, не прибегая к копированию, а также упростим структуру сценария.

При сворачивании фрагмента сценария в подмодель формирование ее портов, определение их типа и настройка происходят автоматически, если к портам подключены связи. В нашем случае необходимо добавить и настроить выходные порты подмодели вручную (рисунок 2.73).

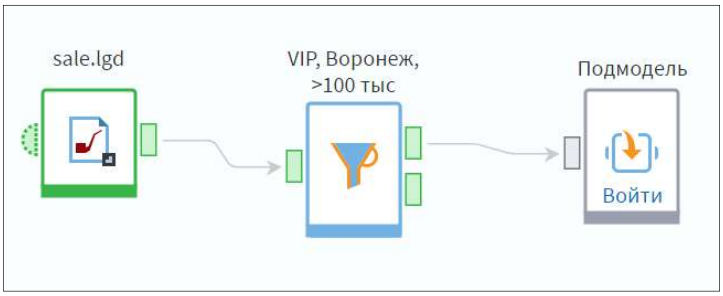


Рисунок 2.73 – Пример сворачивания узлов в подмодель

Входными и выходными портами подмодели могут быть **Переменные**, **Таблица** и **Дерево данных**. На первом шаге мастера настройки подмодели можно добавить произвольное количество портов и отсортировать их в нужном порядке. Если у подмодели несколько портов одного типа, желательно давать им осмысленные метки, чтобы было понятно, какой набор данных ожидается на входе/выходе.

Созданный порт необходимо связать с источником данных во внутренней структуре подмодели. Войдем в подмодель и увидим внутри выбранные нами узлы. Связь между входным портом подмодели и выходным портом **Сортировка по дате** установилась автоматически. Установим связь между выходными портами заключительного узла и подмодели. Соответствие переменных устанавливается автоматически.

Созданную подмодель мы сможем использовать в качестве базового узла для экземпляра компонента **Выполнение узла**.

Далее для решения второй задачи отфильтруем требуемый набор данных (рисунок 2.74).

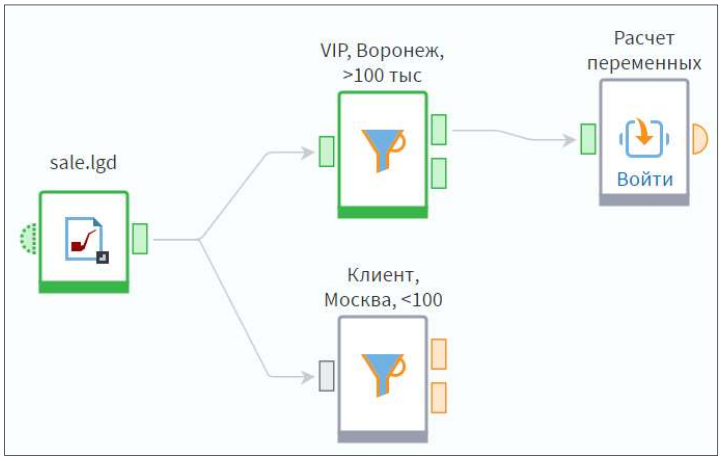


Рисунок 2.74 – Пример создаваемого сценария

Воспользуемся компонентом **Выполнение узла**, в его настройках выберем узел **Расчет переменных** (рисунок 2.75).

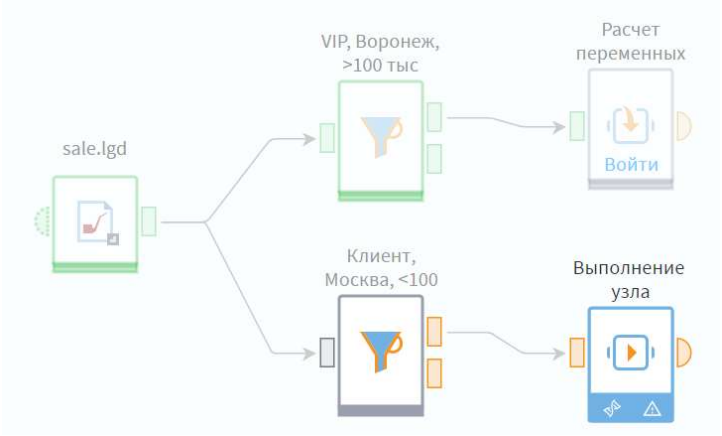


Рисунок 2.76 – Создание узла Выполнение узла

На входе получим нужный набор переменных, рассчитанный по новым данным.

Подмодели представляют собой инструмент иерархической декомпозиции, позволяют создавать многоуровневые структурированные сценарии. Чтобы максимально упростить структуру сценария на верхнем уровне, мы можем объединить в подмодель узлы, решающие одну из поставленных нами задач (рисунок 2.77).

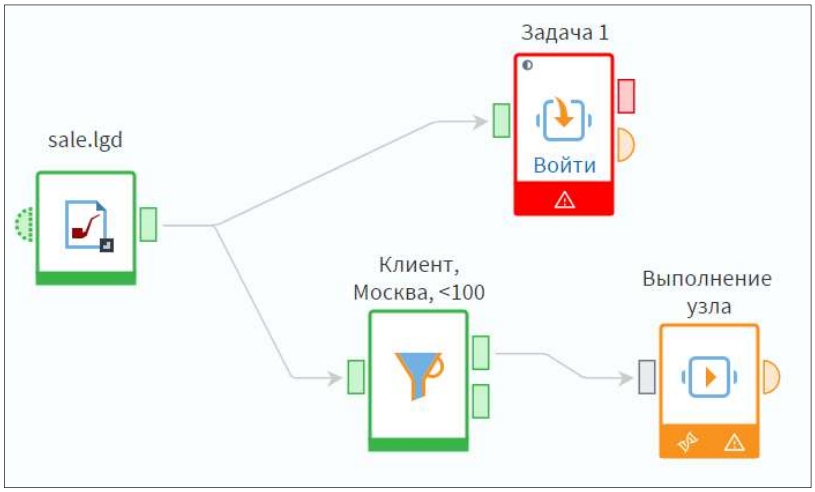


Рисунок 2.77 – Создание подмодели Задача 1

Войдем в подмодель, что подключить связи к выходным портам. На вход подадим табличный набор отфильтрованных данных и рассчитанные переменные (рисунок 2.78).

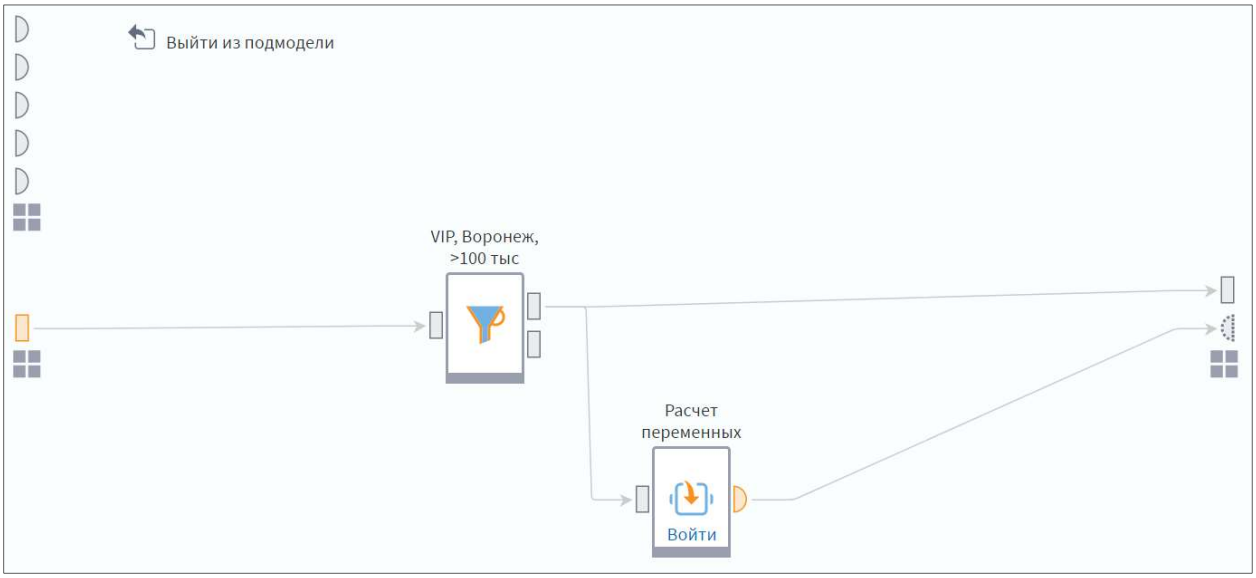


Рисунок 2.78 – Установление связей с выходными портами подмодели

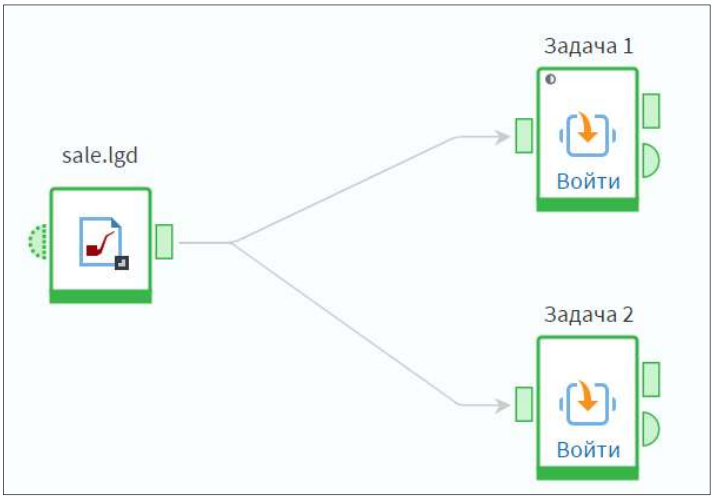


Рисунок 2.79 – Итоговый вид подмоделей

На выходных портах подмоделей получаем нужные нам данные и переменные (рисунок 2.79).

2.14. Компоненты Узел-ссылка и Выполнение узла

Узел-ссылка и **Выполнение узла** – компоненты, которые позволяют значительно облегчить работу аналитика. Их можно настраивать на другие узлы сценария с тем, чтобы повторить логику обработки. Отличие между компонентами в том, что **узел-ссылка** работает с данными исходного узла, а узел выполнения позволяет работать с новыми данными. Исходный узел в терминологии Loginom называется **базовым**, им может быть:

- любой из узлов текущего модуля;
- любой из доступных узлов других модулей (в том числе из других пакетов). В этом случае модификатор доступа узла должен быть настроен как **внутренний**, **открытый** или **опубликованный**.

Важно: невозможно выполнить узлы **Узел-ссылка**, **Цикл**, **Выполнение узла**.

Узел-ссылка – это компонент, который позволяет использовать данные другого узла сценария. Это полезно, в частности, в следующих случаях:

- одни и те же исходные данные необходимо использовать в нескольких сценариях в рамках решения какой-либо задачи;
- данные, рассчитанные с помощью одного сценария, либо части сценария, необходимо использовать в другом;
- нужно сделать сценарий более читабельным.

Компонент **Выполнение узла** позволяет выполнить настроенную ранее логику обработки на новых данных. Его использование позволяет избежать чрезмерного повторения узлов и ветвей сценария, оптимизирует сценарий.

2.15. Лабораторная работа №5 «Узел ссылка и Выполнение узла»

Пусть есть набор данных **data_2014.lgd** год, содержащий информацию о потреблении электроэнергии на различных объектах в течение календарного года. И аналогичный набор данных за 2015 год **data_2015.lgd** (рисунок 2.80).

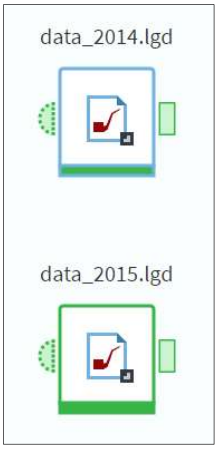


Рисунок 2.80 – Набор данных за 2014 и 2015 год

Перед нами стоит задача подсчитать общее потребление электроэнергии по месяцам для каждого набора. При этом мы точно знаем, что в дальнейшем эти данные понадобятся нам для решения еще нескольких задач.

Мы создали в пакете модуль **Импорт данных** и импортировали наши наборы с тем, чтобы можно было использовать их в сценариях других модулей

Для того чтобы это сделать у узлов должны быть настроены модификаторы доступа не ниже **внутреннего**. Зайдем в меню **Другие действия** и выберем команду **Настроить модификатор доступа …** (рисунок 2.81).

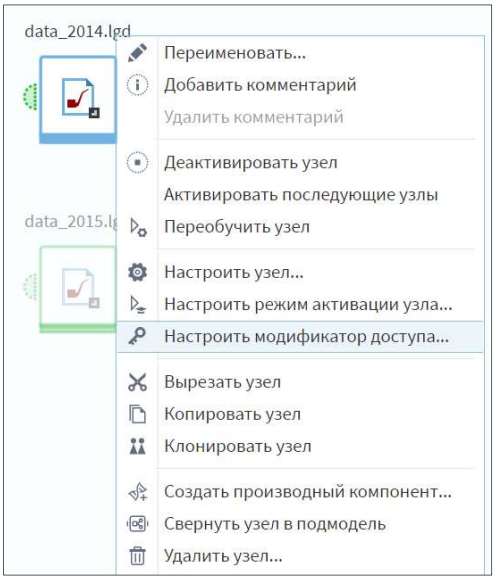


Рисунок 2.81 – Команда Настроить модификатор доступа …

Установим нужный модификатор и применим настройку (рисунок 2.82).

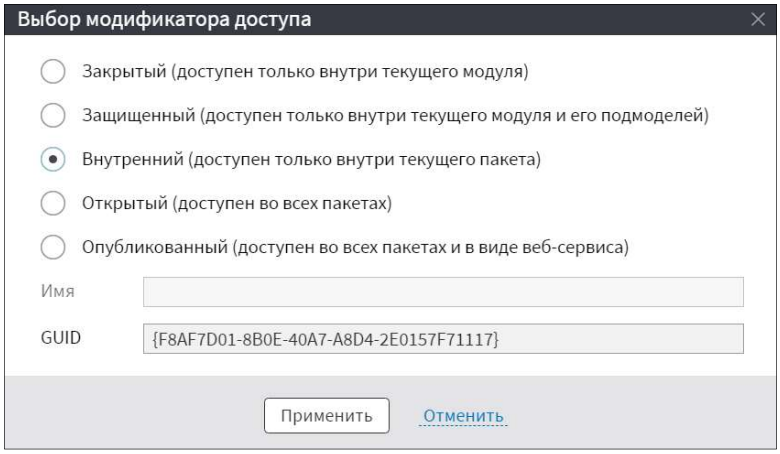


Рисунок 2.82 – Выбор модификатора доступа

Таким же образом настроим модификатор для другого узла и перейдем в другой модуль пакета, который заранее создали для решения нашей задачи.

Откроем сценарий модуля. В группе **Управление** найдем компонент **Узел-ссылка**. Добавим его в область построения сценария и настроим.

В настройках узла присутствует дерево со всеми доступными пакетами и их узлами. На данный момент нам доступны только узлы текущего пакета. Как воспользоваться узлами из другого пакета, мы поговорим в следующем занятии (рисунок 2.83). Развернув дерево, видим наши узлы. Если бы мы не настроили модификатор доступа заранее, список узлов был бы пуст. Выберем первый узел импорта.

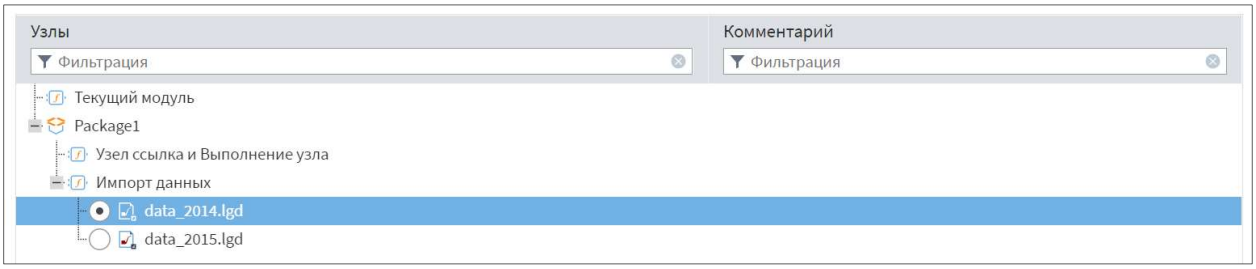


Рисунок 2.83 – Настройка Узел-ссылка

На втором шаге зададим узлу метку **Импорт: data_2014.lgd** и сохраним настройки. У узла-ссылки появился значок исходного узла, а также его выходной порт. Обратите внимание: так как узел-ссылка позволяет использовать только данные другого узла, входных портов у него не бывает.

Если мы выполним узел и откроем быстрый просмотр, убедимся, что получили нужные данные. Таким образом, мы можем использовать их в других модулях или в разных частях сценария.

С помощью **узла-ссылки** мы можем получить данные любого узла. Кроме того, он позволяет повысить читабельность сценариев: можно перенести часть сценария на другую «строку», тем самым повышая его наглядность и удобство работы с ним.

Посмотрим, как это работает на другом примере (рисунок 2.84).

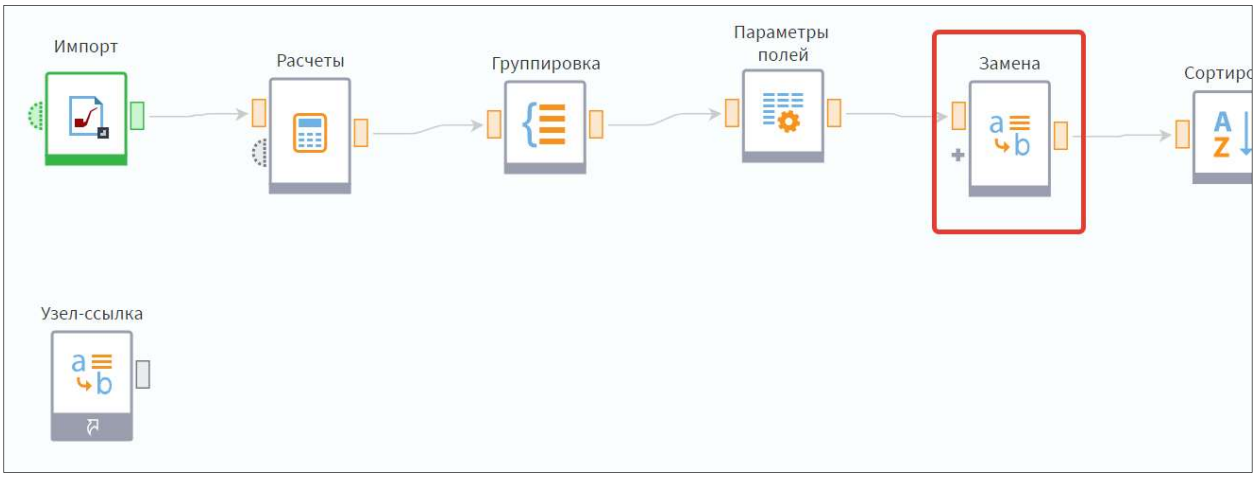


Рисунок 2.84 – Пример рассматриваемого сценария

Перенесем часть сценария, следующую за узлом **Замена**, на новую «строку». Для этого добавим и настроим узел-ссылку. Данные на выходе узла-ссылки соответствуют данным на выходе узла замены.

Теперь можно перенести часть сценария, которая не помещалась на экран, связав ее с выходным портом узла-ссылки. Сценарий стал более компактным и удобным (рисунок 2.85).

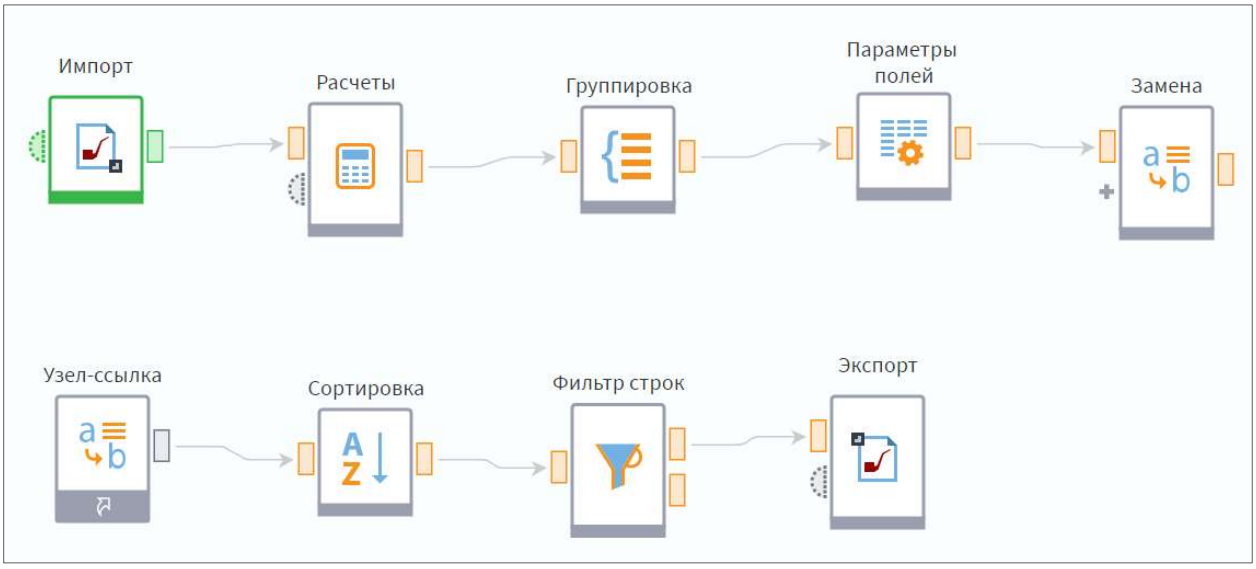


Рисунок 2.85 – Пример рассматриваемого сценария с узлом Узел-ссылка

Нажав на пиктограмму со стрелкой, можно посмотреть источник конкретной ссылки или исходные узлы имеющихся в сценарии в сценарии узлов ссылок (рисунок 2.86).

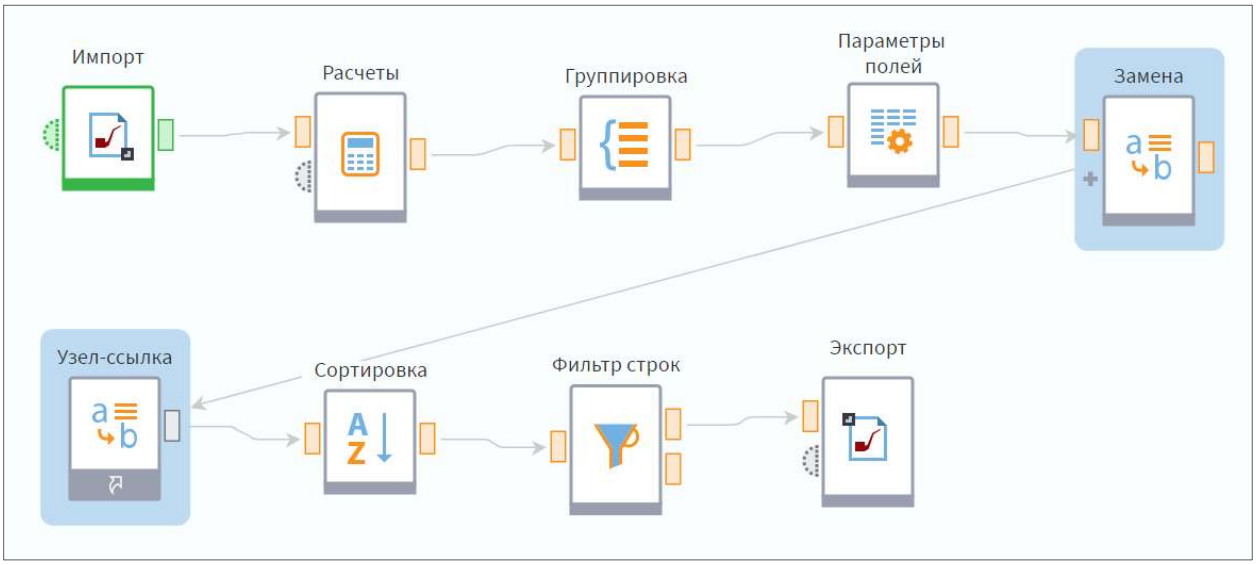


Рисунок 2.86 – Стрелка направлена от источника (или базового узла) к ссылке

Узел-ссылку можно использовать также для ветвления сценария. Особенно удобно это при наличии большого количества связей между узлами.

Продолжим пример с потреблением электроэнергии. Напомним, что задача состоит в том, чтобы подсчитать для каждого набора потребление по месяцам. Добавим в сценарий ссылку на второй набор.

Теперь подсчитаем общее потребление электроэнергии по месяцам 2014 года. Для этого группируем данные по дате, указав для поля **Потребление электроэнергии** вариант агрегации **Сумма** (рисунок 2.87).

Выходной набор данных

#	Дата	Потребление эл.энергии, кВт ч|Сумма
1	01.01.2014, 00:00	28 764,00
2	01.02.2014, 00:00	27 387,00
3	01.03.2014, 00:00	27 171,00
4	01.04.2014, 00:00	151 970,77
5	01.06.2014, 00:00	134 309,03
6	01.07.2014, 00:00	144 942,30
7	01.08.2014, 00:00	136 287,23
8	01.09.2014, 00:00	149 105,48
9	01.10.2014, 00:00	173 734,65
10	01.11.2014, 00:00	173 346,75
11	01.12.2014, 00:00	240 997,43
12	01.05.2014, 00:00	111 480,00

Рисунок 2.87 – Группируем данные по дате

Для того чтобы таким же образом обработать данные за 2015 год, добавим в наш сценарий узел выполнения. Он также относится к группе **Управление**.

В настройках видим такое же дерево, как и в узле-ссылке. В качестве базового можно выбрать только один из доступных узлов. Если необходимо выполнить последовательность из нескольких узлов, то сначала следует объединить их в подмодель, а затем указать ее в настройках **Выполнения узла**. Выберем в качестве базового узел **Группировка: Дата**.

В нижней части окна настройки присутствует флаг **Сохранять конфигурацию выбранного узла** (рисунок 2.88). Когда он включен, при выполнении узла сохраняется его конфигурация на случай, если она отличается от конфигурации исходного узла. Эта опция актуальна, когда при выполнении узла происходят какие-то изменения, например, узел переобучается. Конфигурация входных портов сохраняется всегда и от данного флага не зависит. По умолчанию флаг выключен. В данном примере в нем нет необходимости, поэтому оставим без изменений.

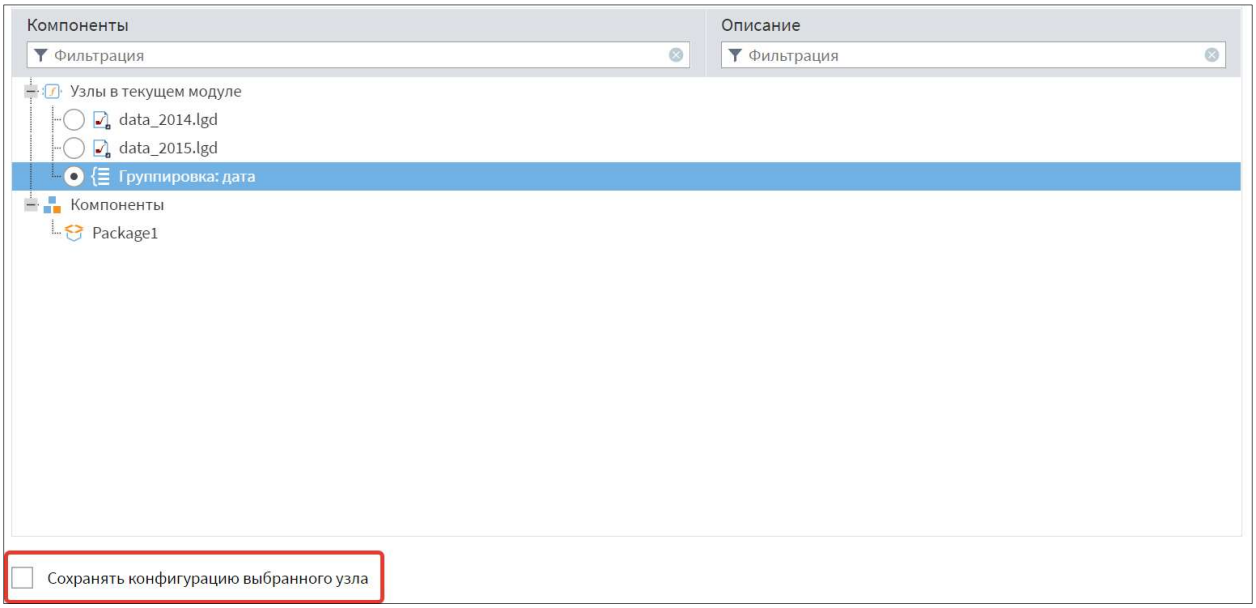


Рисунок 2.88 – Флаг Сохранять конфигурацию выбранного узла

У узла появились порты, аналогичные портам базового узла, причем не только выходные, но и входные: узел позволяет обработать новые данные. Также сразу отображается порт управляющих переменных. Переименуем узел, подадим на входной порт данные за 2015 год и выполним узел (рисунок 2.89).

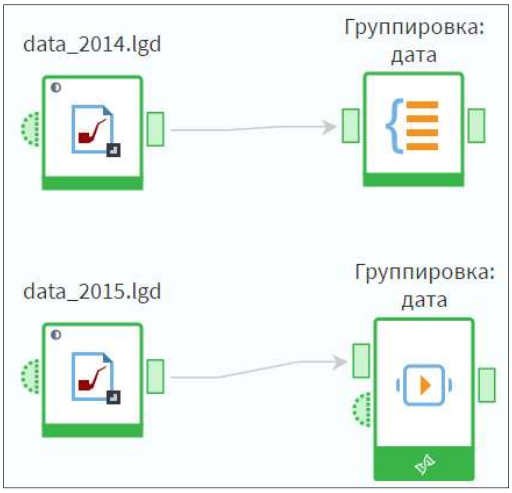


Рисунок 2.89 – Запуск узла Выполнение узла

Настроенный функционал узла Группировка: Дата выполняется на новых данных.

Внесем изменения в базовый узел и рассчитаем среднее потребление.

После того как в базовый узел внесли изменения, все узлы выполненные на основе него подсветились желтым цветом. После запуска узла Выполнение узла в результате получен набор данных совпадающий по структуре с выхода базового узла.

Далее создадим новый модуль в котором попробуем использовать ранее сделанные наработки. С помощью контекстного меню перейдем к настройкам производного компонента. В связи с тем, что компонент не виден в других модулях до настроем его с помощью команды **Настроить…** (рисунок 2.90).

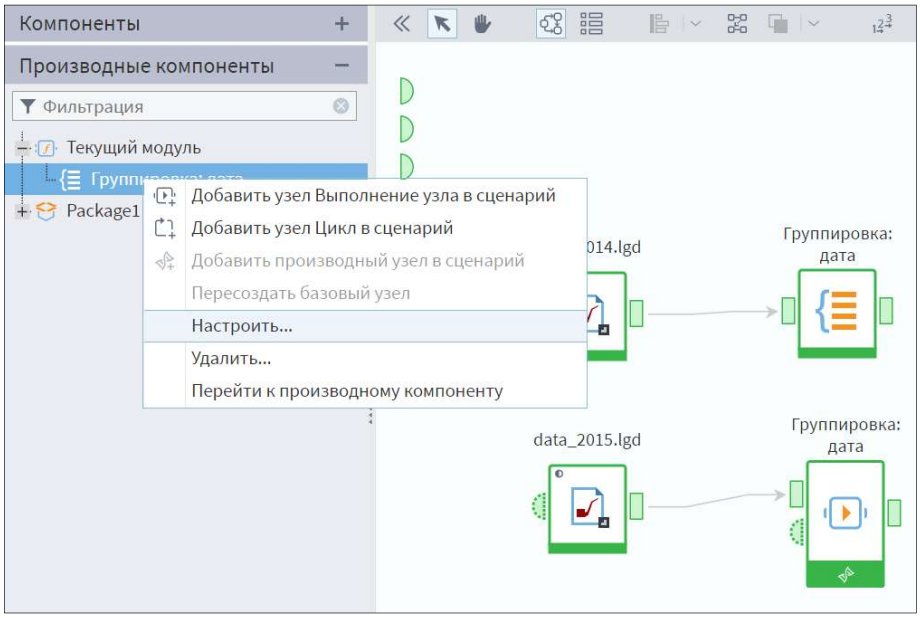


Рисунок 2.90 – Настройка производного компонента

Изменим область видимости на **Внутренний**, после этого он должен быть доступен в других модулях. Теперь наш компонент появился в списке и его можно добавить в сценарий.

Добавим набор данных **data_2015.lgd** через узел ссылку и перенесем производный компонент **Группировка: дата**. Таким образом, мы получили результат обработки данных с помощью компонента, созданного в другом модуле пакета.

2.16. Внешние компоненты и библиотеки

В Loginom предусмотрена возможность повторного использования настроенных узлов или фрагментов сценария. Рассмотрим их подробнее:

- **Копирование фрагмента.** Данный способ позволяет создать полностью независимую копию узла/узлов. Никакой связи между исходным узлом и копией не формируется, при необходимости внесения изменений их придется вносить в **каждую** копию. По сути это то же самое, что создать нужные узлы заново с такими же настройками. Таким образом, единственное преимущество копирования – ускорение разработки сценария. Поэтому использовать этот способ при реализации повторной логики не рекомендуется.

- **Использование стандартного компонента Выполнение узла.** Данный способ позволяет использовать имеющиеся узлы повторно для обработки новых данных, но исключает возможность увидеть внутреннюю структуру/настройки узлов или вносить в них изменения. При этом существует связь с базовым узлом: при внесении изменений в его настройки, узел выполнения отнаследует эти изменения.
- **Создание производного компонента.** Производный компонент и созданный на базе него узел также имеют связь с базовым узлом, но, в отличие от узла выполнения, в его логику обработки можно вносить изменения. Последние два способа подразумевают не только работу с созданными фрагментами в рамках одного пакета, но также создание и подключение внешней библиотеки компонентов. Остановимся на этих способах подробнее.

Внешняя библиотека компонентов – это пакет с набором спроектированных подмоделей, каждая из которых выполняет какую-либо законченную обработку. Все подмодели библиотеки, ставшие производными компонентами, по умолчанию имеют область видимости **Внутренний**. Для того, чтобы компонент стал доступен в любом пакете, который будет его использовать, область видимости нужно вручную повысить до **Открытый**.

Библиотеки компонентов позволяют сократить время на разработку сценариев, многократно использовать логику обработки, ранее заложенную в компоненты. Механизм наследования обеспечивает актуальность версий компонентов. В библиотеку можно объединять компоненты для решения задач отдельной предметной области.

Внешние библиотеки можно разделить на закрытие и открытие:

- **Закрытая библиотека** представляет собой зашифрованный пакет, компоненты которого можно использовать в другом пакете только одним способом – с помощью выполнения узла. Сам пакет библиотеки невозможно открыть в Loginom, можно использовать только ссылку на него. Соответственно, в компоненты библиотеки нельзя войти и увидеть, как они устроены внутри.
- **Открытые библиотеки.** Пакет открытой библиотеки можно открыть в Loginom точно также, как и любой другой пакет. На основе каждого из открытых узлов может быть создан производный компонент, что позволяет использовать возможности объектно-ориентированного моделирования. В этом случае можно увидеть, а, при желании, модифицировать внутреннее устройство компонентов как в пакете библиотеки, так и при подключении ссылки на него. Существует и промежуточный вариант, когда пакет библиотеки зашифрован, но ее компоненты (или часть компонентов) доступны как производные.

Рассмотрим на примере способы использования внешних компонентов. Библиотека расположена на **GitHub**. Для скачивания библиотеки перейдите, пожалуйста, по ссылке: https://github.com/loginom/loginom-silver-kit – и нажмите кнопку **Download ZIP**. Там же можно познакомиться с документацией к библиотеке. После скачивания распакуйте архив с библиотекой на свой компьютер, либо в файловое хранилище.

Для подключения библиотек к текущему пакету необходимо перейти в раздел **Ссылки** и выбрать команду **Добавить** в контекстном меню разделав дереве пакетов, в контекстном меню области ссылок, либо воспользоваться одноименной кнопкой на панели инструментов (рисунок 2.91). Далее следует указать путь к файлу пакета библиотеку и нажать **Добавить**.

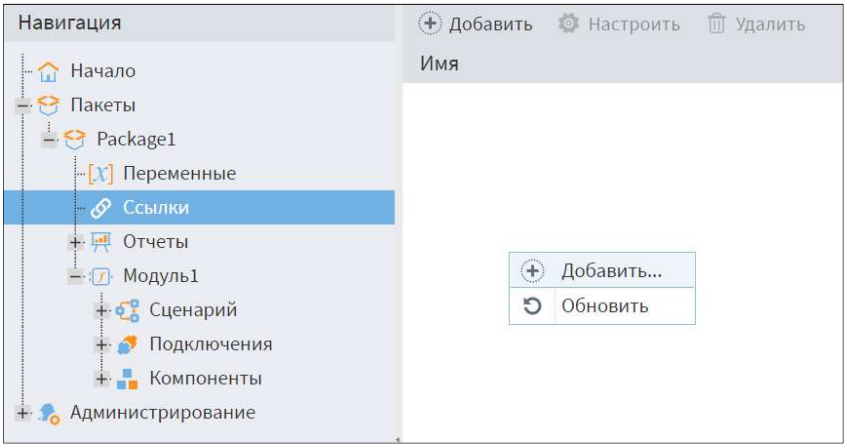


Рисунок 2.91 – Подключение библиотеки к текущему пакету

Перейдем в сценарий. Во вкладке производные компоненты появился подключенный пакет и список компонентов, которые можно из него использовать в текущем пакете. Все производные компоненты сгруппированы по пакетам, в которых они созданы.

После нажатия правой кнопкой на компоненте библиотеки во всплывающем меню становятся доступны три варианта добавления узла по выбранному компоненту: через узел **Выполнения узла**, **Цикл** и **производный узел**.

Выберем вариант **Добавить узел Генератор календаря**. Добавлять узлы на выполнение можно также и простым перетаскиванием в область построения.

Узел выполнения настроился на внешний компонент: у него появились такие же входы и выходы и таже логика обработки. При этому нас нет доступа к внутреннему устройству компонента, он для нас как «черный ящик». Посмотрим порт входных переменных.

На входах внешних компонентов заданы все обязательные поля и переменные, узел выполнения отнаследовал эту структуру. Рассматриваемый

компонент предназначен для генерации списка дат между двумя заданными датами, поэтому здесь мы имеем только переменные – первую и последнюю даты требуемого календаря (рисунок 2.92).

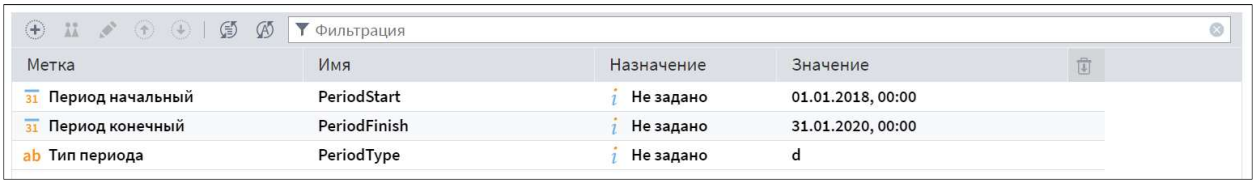


Рисунок 2.92 – Порт входных переменных узла Генератор календаря

Пусть мы хотим получить календарь за 2022 год. Зададим соответствующие знание переменным и сохраним настройки. Выполним узел. Узел успешно отработал и мы получили нужный нам список дат (рисунок 2.93). Аналогичным образом можно использовать компоненты библиотек любого типа: открытых или зашифрованных. При внесении разработчиком изменений в библиотеку достаточно заменить пакет библиотеки на пакет новой версии – и узлы выполнения будут использовать новую версию компонентов благодаря их связи с базовыми узлами

Набор данных

#	Календарь
1	01.01.2022, 00:00
2	02.01.2022, 00:00
3	03.01.2022, 00:00
4	04.01.2022, 00:00
5	05.01.2022, 00:00
6	06.01.2022, 00:00
7	07.01.2022, 00:00
8	08.01.2022, 00:00

Рисунок 2.93 – Список дат за 2022 год

Следующий способ использования внешних компонентов позволяет не только использовать, но и модифицировать компоненты библиотеки. Он доступен для библиотек, в которых на базе подмоделей созданы **производные компоненты**.

Выберем вариант **Добавить производный узел в сценарий**. Также добавить производный узел можно перетаскиванием с зажатым Ctrl.

Мы добавили в область построения узел с вкладки производных компонентов. У него такие же входы, выходы и метка, каки у производного компонента. Это, по сути, второй способ использования заложенной внутрь логики обработки.

После задания тех же значений переменным и выполнения узла получили точно такой же набор данных. Посмотрим, что произойдет, если мы нажмем **Войти** (рисунок 2.94).

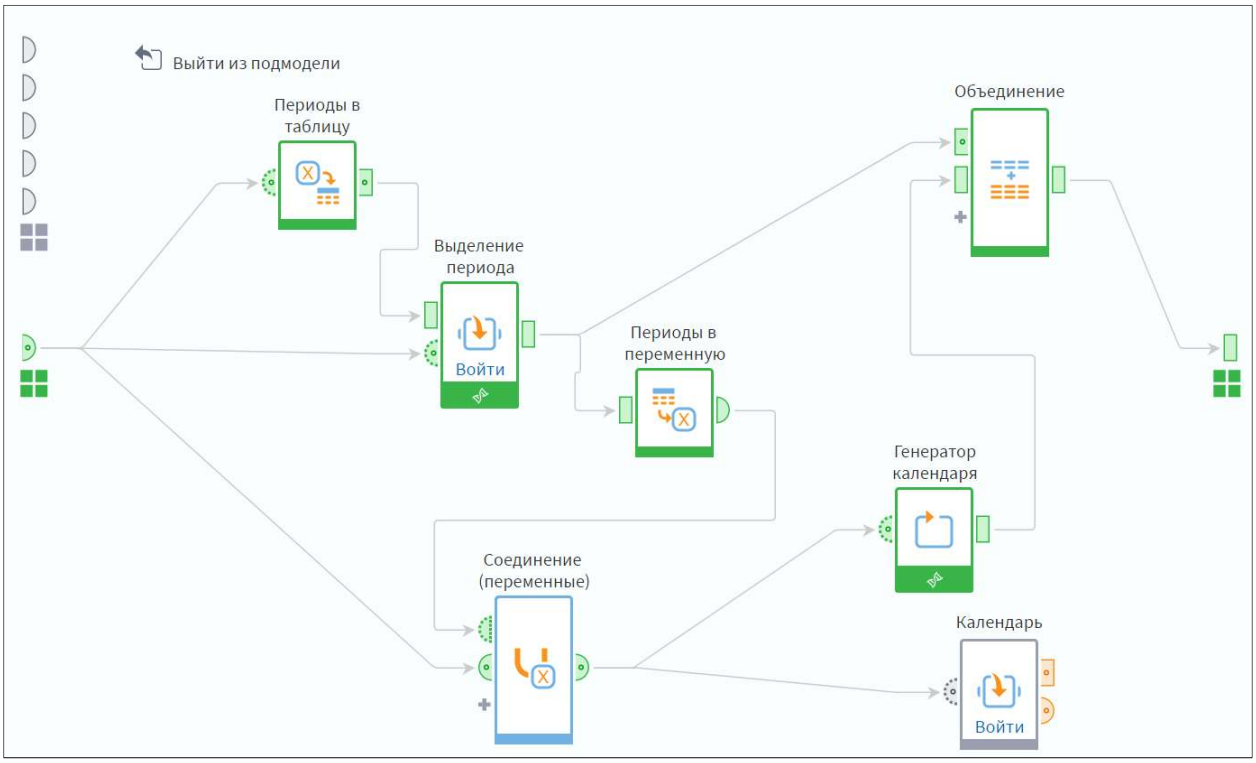


Рисунок 2.94 – Исследование узла Генератор календаря

Использовать производные компоненты можно также в виде цикла. Выберем вариант **Добавить узел Цикл в сценарий**. Также добавить узел Цикл можно перетаскиванием с зажатием Shift.

Чтобы использовать узел как цикл нужно перейти в настройки и задать необходимое количество итераций.

2.17. Лабораторная работа №6 «Закрепление материалы в части основ работы в Loginom»

Задание:

1. Для выполнения задания скачайте файл **Вероятности визитов.txt**, который имеет следующие поля:

- **Клиент.Код** – уникальный идентификатор клиента;

- **Отклик** – поле принимает значения **True** или **False**;
- **Вероятность визита** – поле принимает значения в диапазоне от **0** до **1**;
- **Признак тестового множества** – поле принимает значения **True** или **False**;
- **Тип клиента** – поле принимает значения **Тип 1** или **Тип 2**.

2. Создайте в Loginom новый пакет и импортируйте описанный набор данных.

3. Рассчитайте количество клиентов, у которых в идентификаторе **Клиент.Код** присутствуют два и более нулей (рисунок 2.95).

14	24 554	false	0,06	false	Тип 2
15	24 880	false	0,00	false	Тип 1
16	29 173	false	0,01	false	Тип 2
17	29 440	true	0,05	false	Тип 1
18	30 600	false	0,05	false	Тип 1
19	42 439	true	0,36	false	Тип 1
20	45 799	false	0,04	false	Тип 1
21	54 496	false	0,17	false	Тип 2
22	57 002	false	0,01	false	Тип 2
23	59 862	false	0,04	false	Тип 2
24	63 002	false	0,02	false	Тип 1
25	75 205	false	0,00	false	Тип 1
26	75 469	true	0,01	false	Тип 1
27	84 573	false	0,04	false	Тип 1
28	89 796	true	0,03	false	Тип 2
29	91 588	true	0,17	false	Тип 2
30	95 036	false	0,10	false	Тип 2
31	95 982	true	0,39	false	Тип 2
32	102 385	false	0,01	false	Тип 2

Рисунок 2.95 – Пример идентификаторе Клиент.Код присутствуют два и более нулей

Обратите внимание: для решения данной задачи не требуется создавать сложный сценарий, достаточно внимательно изучить функции узла Калькулятор.

4. Подключите к вашему пакету библиотеку **Loginom Silver Kit**.

5. Отберите всех клиентов, которые входят в тестовое множество.

6. Далее используйте внешний компонент **AUC** из библиотеки **Loginom Silver Kit**. Вам потребуются следующие настройки в мэппинге полей:

- Отклик – Событие;
- Вероятность визита – Оценка.

После настройки входного порта достаточно запустить узел на выполнение.

Ответьте на вопросы:

1. Сколько клиентов входит в тестовое множество?
2. У каких клиентов из тестового множества вероятность визита максимальна?
3. У каких клиентов из тестового множества вероятность визита минимальна?
4. Чему равны значения различных показателей на выходе узла **AUC**?

Глава 3. ПОДГОТОВКА ДАННЫХ

3.1. Группировка и преобразование даты

Важнейшим процессом в системах хранения данных является извлечение данных из различных источников, их преобразование к единому формату и модели данных, очистка от дубликатов, противоречий и других факторов, которые могут помешать анализу, а также загрузка в единый интегрированный источник.

Для реализации данного процесса в систему включается специальный комплекс аппаратно-программных средств, называемых **ETL** (англ.: Extract, Transform, Load – извлечение, преобразование, загрузка) [5].

Рассмотрим преобразование данных подробнее. **Преобразование данных** (которое еще называют **трансформацией**) зависит от задач, алгоритмов и целей анализа. Таким образом, для разных задач потребуются различные методы преобразования.

Преобразование данных – очень широкое понятие, которое не имеет четко очерченных границ. Данный термин иногда распространяют на любые манипуляции с данными независимо от их целей и методов.

Однако в контексте аналитики преобразование данных имеет вполне конкретные цели и задачи, а также использует достаточно стабильный набор методов.

Преобразование данных не ставит целью изменить информационное содержание данных. Его задача – **представить информацию в таком виде, чтобы она могла быть использована наиболее эффективно с точки зрения решаемых задач аналитики данных**.

Преобразование данных в том или ином виде выполняется во многих компонентах информационных систем:

- В процессе переноса и загрузки данных в интегрированный источник или области временного хранения (ETL).
- Непосредственно при подготовке данных к анализу в бизнес-приложении (SKD).

Такая распределенность процесса преобразования обусловлена тем, что на каждом этапе он преследует различные цели.

Операции преобразования могут проводиться для обеспечения технической и логической совместимости данных, их подготовки к извлечению, переносу в хранилище данных и так далее.

Например, адреса часто вводят одной строкой. В то же время для анализа могут представлять интерес отдельные компоненты адреса, которые имеют

как **текстовый формат** (улица, город), так и **числовой** (номер дома, офиса). С помощью трансформации можно распределить соответствующие элементы по отдельным полям и преобразовать их в нужный формат.

Существует две основные цели преобразования данных на этапе процесса ETL:

- Приведение их в соответствие с моделью данных, используемой в консолидированном источнике.
- Осуществление корректной консолидации данных и загрузка в консолидированный источник.

В связи с этим возникает вопрос. Если преобразованию данных так много внимания уделяется на этапе интеграции данных, то зачем включать средства преобразования в аналитическое приложение? Ведь это, несомненно, усложняет и делает более дорогим его разработку.

Ответ прост. Не все данные поступают в бизнес-приложение из систем, где они прошли предварительную подготовку. Но главная причина заключается в том, что трансформация данных в этих системах в большей степени носит технический характер и слабо связана с возможными методами, алгоритмами и целями анализа.

Прежде чем приступить к подробному рассмотрению методов, алгоритмов и целей отдельных направлений преобразования данных, проведем краткий обзор базовых операций преобразования, которыми оснащается большинство аналитических платформ и ETL-инструментов:

- **Параметры полей.** Позволяет изменять имена, типы, метки и назначения полей исходной выборки данных. Например, если поле, содержащее числовую информацию, в источнике данных по какой-либо причине имеет строковый тип, значения этого поля не могут обрабатываться как числа. Чтобы работа с числовыми данными этого поля стала возможной, их следует преобразовать к числовому типу.
- **Квантование.** Позволяет разбить диапазон возможных значений числового признака на заданное количество интервалов и присвоить номера интервалов или иные метки попавшим в них значениям.
- **Фильтр строк.** Оставляет только те записи, которые удовлетворяют заданным условиям.
- **Сортировка.** Позволяет изменить порядок следования записей исходной выборки данных в соответствии с алгоритмом, определенным пользователем. В некоторых случаях сортировка дает возможность упростить визуальный анализ выборки, оперативно определить наибольшие и наименьшие значения признаков и так далее.

- **Обогащение данных.** Включает в себя несколько операций, позволяющих дополнять выработку недостающей информации из других выборок, если исходная информация содержит недостаточно данных для анализа (слияние, дополнение данных, объединение, соединение).

Операция **слияния**, в частности, позволяет объединить две таблицы по одноименным полям, **дополнение** данных – использовать одноименные поля для дополнения одной таблицы полями из других, которые отсутствуют в первой.

При **объединении** к записям исходной выборки добавляются все записи другой, а в случае операции **соединения** добавляются все выбранные поля.

- **Табличная подстановка значений**. Позволяет производить замену значений в исходной выборке данных на основе так называемой таблицы подстановки. Таблица подстановки содержит пары **исходное значение-новое значение**. Каждое значение выборки данных проверяется на соответствие исходному значению таблицы подстановки, и если такое соответствие найдено, то значение выборки изменяется на соответствующее новое значение из таблицы подстановки. Это очень удобный способ для автоматической корректировки значений.
- **Группировка**. Очень часто информация, интересующая аналитика, в таблице оказывается «разбавлена» посторонними данными, разобщена, разбросана по отдельным полям и записям. Используя группировку, можно обобщить нужную информацию, объединить ее в минимально необходимое количество полей и значений.
- **Вычисляемые значения**. Иногда для анализа требуется информация, которая отсутствует в явном виде в исходных данных, но может быть получена на основе вычислений над имеющимися значениями.

Например, если известны цена и количество товара, то сумма может быть рассчитана как их произведение. Для этих целей в аналитическое приложение включается своего рода калькулятор, который позволяет выполнять над данными исходной выборки различные вычисления.

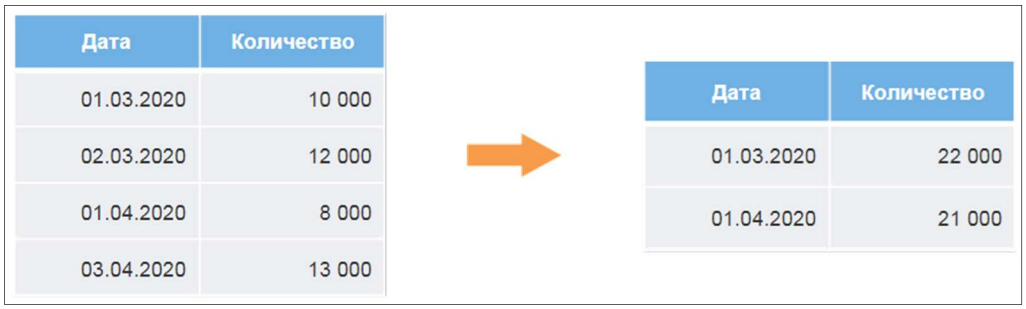
Поскольку анализируемые данные могут быть различных типов (**строковый, числовой, дата/время, логический**), то механизм расчетов должен поддерживать работу не только с числовыми данными, но и сданными других типов, например, выделять подстроку, выполнять логические операции и так далее.

- **Преобразование упорядоченных данных.** Позволяет оптимизировать представление таких данных с целью обеспечения дальнейшего анализа, например, решения задачи прогнозирования временного ряда или группировки по временному периоду.

- **Транспонирование.** Служит для вращения набора данных, когда строки необходимо сделать столбцами и наоборот.

Рассмотрим в качестве примера такую ETL-операцию, как **группировка**.

Иногда высокая степень детализации данных может мешать анализу данных. Ежедневные показатели некоторого бизнес-процесса, например, продаж, подвержены колебаниям под действием различных случайных факторов, поэтому если взять значения продаж по отдельности за каждый день, вряд ли они дадут надежную оценку, характеризующую развитие процесса. Но если сгруппировать ежедневные показатели за неделю или месяц (рисунок 3.1), то оценка будет более обоснованной.



Дата	Количество
01.03.2020	10 000
02.03.2020	12 000
01.04.2020	8 000
03.04.2020	13 000

Дата	Количество
01.03.2020	22 000
01.04.2020	21 000

Рисунок 3.1 – Пример группировки данных

Таким образом, группировка данных – полезный инструмент, применение которого в процессе подготовки данных к анализу позволяет более эффективно использовать содержащуюся в данных информацию.

Перед тем, как приступать к группировке, необходимо определить в исходной выборке данных поля групп и показателей.

Группы (или измерения) – это данные, характеризующие исследуемый процесс качественно, а **показатели** (их также называют фактами) – количественно (рисунок 3.2).

Рисунок 3.2 – Пример группы и показателей

Например, процесс продаж описывается **качественными** и **количественными** данными.

Качественные данные – наименования товаров и клиентов. Очевидно, что просто указать эти параметры недостаточно. С каждым проданным товаром или клиентом связан набор числовых показателей: цены, количество, суммы, скидки, наценки и так далее.

Каждое наименование товара или клиента, которое содержится в поле группы, называется **значением группы** (рисунок 3.3). Суть группировки заключается в том, что все записи, содержащие одноименные значения группы, по которой проводится группировка, объединяются в одну, а соответствующие показатели агрегируются.

Рисунок 3.3 – Значение группы

Рассмотрим для примера три варианта группировки.

В качестве исходной таблицы рассмотрим данные представленные на рисунке 3.4. Группировку будем проводить по трем полям: **Дата**, **Клиент** и **Товар**.

Дата	Клиент	Товар	Цена	Количество	Сумма
01.03.2016	ООО «Полигон»	Цемент	150	20	3 000
01.03.2016	ЗАО «Монтажник»	Керамзит	100	60	6 000
01.03.2016	ООО «Тандем»	Кирпич	1 500	5	7 500
02.03.2016	ООО «Шплинт»	Плиты	1 100	10	11 000
02.03.2016	ЗАО «Монтажник»	Блоки	900	20	18 000
02.03.2016	ООО «Полигон»	Кирпич	1 500	10	15 000
03.03.2016	ООО «Агрострой»	Плиты	1 100	8	8 800
03.03.2016	ЗАО «Монтажник»	Керамзит	100	30	3 000
03.03.2016	ООО «Тандем»	Блоки	900	10	9 000
03.03.2016	ООО «Полигон»	Плиты	1 100	30	33 000
04.03.2016	ООО «Шплинт»	Керамзит	100	100	10 000
04.03.2016	ООО «Тандем»	Кирпич	1 500	10	15 000
04.03.2016	ЗАО «Монтажник»	Цемент	150	20	3 000
04.03.2016	ООО «Полигон»	Блоки	900	15	13 500
05.03.2016	ООО «Агрострой»	Плиты	1 100	20	22 000
05.03.2016	ООО «Шплинт»	Керамзит	100	50	5 000
05.03.2016	ЗАО «Монтажник»	Цемент	150	40	6 000
05.03.2016	ООО «Тандем»	Блоки	900	15	13 500
06.03.2016	ООО «Полигон»	Кирпич	1 500	6	9 000
06.03.2016	ООО «Агрострой»	Керамзит	100	70	7 000
06.03.2016	ЗАО «Монтажник»	Цемент	150	30	4 500

Рисунок 3.4 – Исходная таблица

Группировка по полю Дата. Если в качестве группы выбирается **Дата**, то все записи, которые содержат одинаковые даты, будут объединены в одну.

В нашем примере для показателя **Цена** была выбрана функция агрегации с вычислением среднего значения, для показателей **Количество** и **Сумма** – с вычислением суммы.

Таким образом, данный вариант группировки позволяет получить информацию о том, какое количество единиц товара, по какой средней цене и на какую сумму было продано в пределах каждой даты (рисунок 3.5).

Дата	Цена	Количество	Сумма
01.03.2016	583,33	85	16 500
02.03.2016	1 166,67	40	44 000
03.03.2016	800,00	78	53 800
04.03.2016	662,50	145	41 500
05.03.2016	562,50	125	46 500
06.03.2016	583,33	106	20 500

Рисунок 3.5 – Пример группировки по полю Дата

Группировка по ролю Клиент. Группировка по полю **Клиент** позволяет получить те же агрегированные показатели, что и в предыдущем примере, но уже не для каждой даты, а для каждого клиента. Таким образом мы получаем среднюю цену товаров, которые приобретал клиент, общее количество приобретенных товаров и общую сумму всех покупок (рисунок 3.6).

Клиент	Цена	Количество	Сумма
ЗАО «Монтажник»	258,33	200	40 500
ООО «Агрострой»	766,67	98	37 800
ООО «Полигон»	1030,00	81	73 500
ООО «Тандем»	1200,00	40	45 000
ООО «Шплинт»	433,33	160	26 000

Рисунок 3.6 – Пример группировки по полю Клиент

Группировка по полю Товар. Группировка по полю **Товар** позволяет получить информацию о том, по какой цене, в каком количестве и на какую сумму был продан каждый из товаров (рисунок 3.7).

Товар	Цена	Количество	Сумма
Блоки	900	60	54 000
Керамзит	100	310	31 000
Кирпич	1500	31	46 500
Плиты	1100	68	74 800
Цемент	150	110	16 500

Рисунок 3.7 – Пример группировки по полю Товар

Полученная в результате группировки информация может использоваться для различных задач:

- Распределение сумм по продажам и датам – анализ динами продаж, выявление тенденций и др.
- Группировка по клиенту – оптимизация работы с клиентами, например, предоставление скидок наиболее активным и др.

- Группировка по товару – определение наиболее и наименее продаваемых товаров, оценка вклада конкретного товара в общий объем продаж и др.

При группировки могут быть использованы различные функции агрегации. Рассмотрим их подробнее:

- **Сумма**. Вычисляется сумма агрегируемых значений показателей.
- **Среднее**. Вычисляется среднее агрегируемых значений.
- **Количество**. Число агрегируемых знаний показателей для каждой комбинации групп.
- **Максимум, минимум**. Максимальное или минимальное из агрегируемых знаний.
- **Медиана** – агрегируемые значения сортируются в порядке возрастания, и из полученного набора выбирается центральное (то есть такое значение, что все значения слева от него будут меньше, а справа – больше).

Медиана – это порядковая статистика, которая используется, как альтернатива среднего значения, устойчивая к аномальным значениям данных. Если аномальное значение попадет в число усредняемых, то это может существенно сместить полученную оценку, в то время как медиана в большинстве случаев дает более устойчивую оценку.

Для строковых значений в качестве функций агрегации могут использоваться только **максимум, минимум, количество, первый, последний**.

При этом **максимум (минимум)** нескольких строковых значений рассчитывается посимвольным сравнением.

1. Сначала сравниваются два первых символа строк.
2. Если их коды одинаковы, сравниваются вторые символы и др.
3. Как только в строках появляется первый несовпадающий символ, функция агрегации принимает значение строки, код символа в которой оказался больше (меньше).

За счет объединения значений группировка позволяет оптимизировать представление анализируемых данных с точки зрения эффективности анализа и интерпретируемости его результатов.

Кроме того, группировка дает возможность снизить количество наблюдений, которые необходимо обработать в процессе анализа, а, значит, уменьшить время и вычислительные затраты на его выполнение.

Рассмотрим возможные ETL-операции с датой и временем.

Обычно каждый элемент временного ряда представляет собой соответствие **дата-значение**, а в случае многомерного ряда – **дата-значение 1**, **значение 2** ... (рисунок 3.8).

Дата	Значение 1	Значение 2	Значение N
01.01.2020	10	25,6	...
01.02.2020	20	245,25	...
...	...	...	...
01.12.2020	560	54,34	...

Рисунок 3.8 – Пример операции с датой и временем

Все методы обработки, рассмотренные ранее, были направлены на преобразование значений, сами же даты обычно оставались без изменений. Однако, как показывает практика, при решении различных задач анализа может быть полезным и преобразование даты.

Чаще всего даты в исходных данных представлены в определенном формате, обычно **ДД.ММ.ГГ**, то есть день, месяц и год, при этом на отображение каждого элемента даты отводятся две цифры.

Возможны некоторые вариации: указывается полностью год (02.10.2014) или(и) месяц (12 сентября 2014) и т.д.

С точки зрения анализа наибольший интерес представляют не детализированные данные за каждый день, а агрегированные по неделям, месяцам, кварталам или годам. В этой связи может оказаться полезным извлекать из даты дополнительную информацию о временных интервалах, на которых проводится анализ (рисунок 3.9).

Дата	Дата (Неделя)	Дата (Месяц)	Дата (Квартал)	Дата (Год)
24.04.2020	17	04 Апрель	2	2020
12.08.2020	33	08 Август	3	2020
30.12.2020	53	12 Декабрь	4	2020

Рисунок 3.9 – Пример временного интервала

Так, если базовый интервал, по которому осуществляется анализ деятельности предприятия, – неделя, то в некоторых случаях может

представлять интерес отображение даты не на каждый день, а на первый или последний день недели.

Иными словами, все данные за неделю определяются по первому или последнему ее дню (рисунок 3.10).

Дата	Кол-во
02.01.2017	250
03.01.2017	230
04.01.2017	345
05.01.2017	215
06.01.2017	312
07.01.2017	124
08.01.2017	321
09.01.2017	234
10.01.2017	243
11.01.2017	312
12.01.2017	321
13.01.2017	267
14.01.2017	351
15.01.2017	216
16.01.2017	187
17.01.2017	179
18.01.2017	261
19.01.2017	305
20.01.2017	156

Дата	Дата (Год + Неделя, Первый день)	Кол-во
02.01.2017	02.01.2017	250
03.01.2017	02.01.2017	230
04.01.2017	02.01.2017	345
05.01.2017	02.01.2017	215
06.01.2017	02.01.2017	312
07.01.2017	02.01.2017	124
08.01.2017	02.01.2017	321
09.01.2017	09.01.2017	234
10.01.2017	09.01.2017	243
11.01.2017	09.01.2017	312
12.01.2017	09.01.2017	321
13.01.2017	09.01.2017	267
14.01.2017	09.01.2017	351
15.01.2017	09.01.2017	216
16.01.2017	16.01.2017	187
17.01.2017	16.01.2017	179
18.01.2017	16.01.2017	261
19.01.2017	16.01.2017	305
20.01.2017	16.01.2017	156

Рисунок 3.10 – Пример преобразования данных даты и времени

Аналогично строится преобразование **Год + Квартал**, когда все даты в пределах квартала отображаются его первым/последним числом, или **Год + Месяц**, когда даты в пределах месяца отображаются его первым/последним числом.

Также представляет интерес извлечение из даты номеров временных интервалов: недель, месяцев и кварталов – как в числовом, так и в текстовом виде. Тогда значения этих временных интервалов сами могут использоваться в качестве исходных данных для дальнейшего анализа.

Приведение даты к различным форматам представления позволяет упростить визуальный анализ исходных данных, которые зависят от времени, а также использовать значения времени в качестве исходных данных для решения разнообразных аналитических задач.

Следует отметить, что после преобразования даты часто являются уже не значениями типа Дата, а обычными строковыми и числовыми значениями.

Поэтому их больше нельзя обрабатывать как даты, но можно использовать в моделях как обычные строковые или числовые переменные.

В некоторых случаях могут быть интересны данные, полученные для различных интервалов времени в пределах одной даты, то есть представленные в масштабе часов, минут и секунд.

Обычный формат для представления времени – **ЧЧ:ММ:СС**, то есть для отображения каждого элемента времени отводится по два знакоместа, а символом-разделителем служит двоеточие.

Как правило, для трансформации времени используется извлечение часов, минут и секунд в отдельные поля. Результаты такого преобразования представлены в таблице (рисунки 3.11).

Время	Время (часы)	Время (минуты)	Время (секунды)
12:10:39	12	10	39
13:15:20	13	15	20
16:20:44	16	20	44
18:09:56	18	09	56
19:15:30	19	15	30
21:22:50	21	22	50
23:12:40	23	12	40

Рисунок 3.11 - Пример трансформации времени на часы, минуты и секунды

3.2. Лабораторная работа №7 «Группировка и преобразование даты»

Загрузим в Loginom набор данных **sales.lgd**. В файле **sales.lgd** хранятся данные о продажах строительных товаров: их названия и товарные группы, дата продажи, количество проданного и единица измерения, сумма с учетом скидки, а также информация о городе продажи и типе клиента-покупателя. Всего в наборе 98 471 запись.

Нам необходимо получить набор данных с продажами каждого товара по дням. Поскольку один и тот же товар мог быть продан в один день несколько раз в разных городах, нам потребуется провести группировку. Для этого создадим новый пакет и импортируем данный набор в Loginom.

Компонент **Группировка** позволяет объединять записи по выбранным полям в группы, агрегируя данные в полях, которые выбраны в качестве показателей, с помощью различных статистических функций. Для каждой группы возвращается одна строка.

Создадим в области построения узел **Группировка**. Свяжем го с узлом импорт и перейдем в настройку (рисунок 3.12).

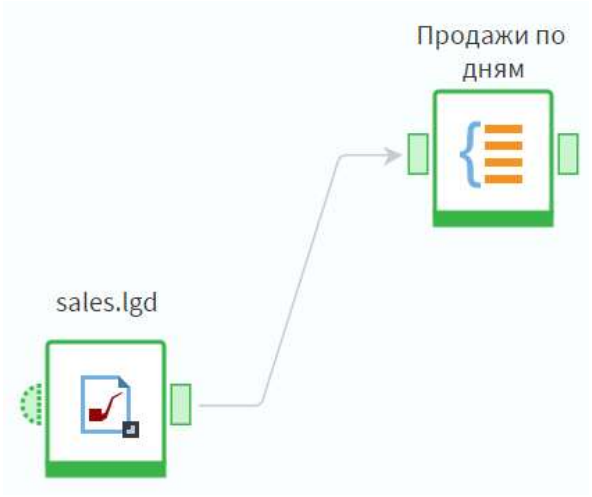


Рисунок 3.12 – Связь между узлами sales.lgd и Продажи по дням

Окно настройки узла группировки состоит из двух основных областей. По умолчанию входные поля попадают в область **Доступные поля** в левой части окна. После соответствующих настроек поля попадают в область **Выбранные поля**, где распределяются по спискам **Группа** и **Показатели**.

Мы хотим получить на выходе суммарные продажи по дням и товарам, значит, в списке групп нам нужны поля **Дата** и **Товар**. Выделим их и нажмем кнопку **Переместить** в **Группу**.

Добавить поля в соответствующий список области выбранных полей можно также перетаскиванием, с помощью контекстного меню или горячих клавиш:

- Alt + G – в список **Группа**;
- Alt + S – в список **Показатели**.

Выбранные поля отобразились в списке **Группа**.

Аналогичным образом добавляем поля **Количество** и **Сумма** с учетом скидки. Для них автоматически установился вариант агрегации **Сумма**, используемый по умолчанию для числовых и вещественных полей (рисунок 3.13). При необходимости изменить вариант агрегации для поля необходимо дважды щелкнуть по нему левой кнопкой мыши.

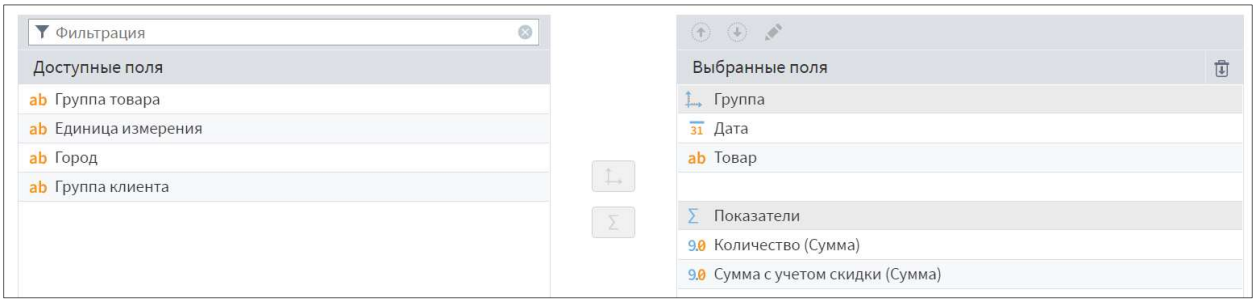


Рисунок 3.13 – Настройка узла Группировка

Рассмотрим еще два параметра настроек узла. Флаг **Кэшировать** значения групп включен по умолчанию и позволяет сохранить полученные после группировки значения групп. Это может существенно ускорить последующую обработку, особенно при подаче большого набора данных, однако необходимо учитывать, что чем больше набор, тем больше памяти займут сохраненные значения.

При установке флага **Сортировать результирующие данные** записи на выходе: будут отсортированы по полям в списке **Группы** с учетом их расположения в списке.

Оставим настройки без изменений и перейдем к просмотру результатов обработки.

Мы получили набор данных по продажам товаров по дням. Необходимо обратить внимание, что при группировке мы потеряли информацию о значениях показателей в размене исключенных групп.

Перейдем в настройку входного узла **Продажи по дням**. После того, как узел настроен, назначение полей во входном порте автоматически меняется в соответствии с заданными настройками (рисунок 3.14). Чтобы изменить этот параметр необходимо дважды щелкнуть на нужном поле левой кнопкой мыши.

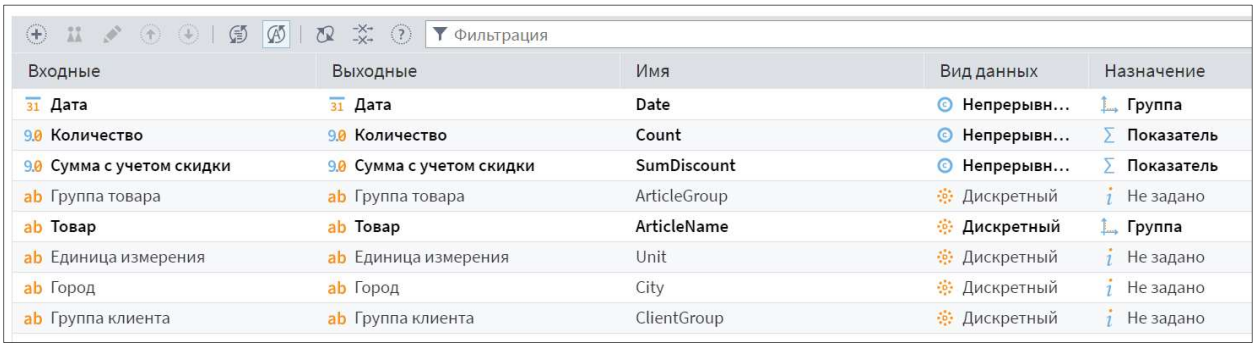


Рисунок 3.14 – Настройка входного порта узла Продажи по дням

Обратите внимание, что при включенной автосинхронизации работать с узлом группировки нужно с осторожностью – если имена полей в новых

входных данных не совпадут с заданными на входе узла, новые поля, даже если у них настроено назначение, не свяжутся с имеющимися, а добавятся к ним.

Рассмотрим пример. Если в новом наборе поле **Дата** будет иметь имя **SaleDate** вместо **Data**, в таблице на входном порте будет два поля: входное поле с именем **SaleData** будет связано с выходным полем с именем **SaleData**, а для поля **Date** входного поля в таблице не будет, что приведет к ошибке.

Добавим в сценарий еще один узел группировки и переименуем оба узла, как показано на рисунке 3.15.

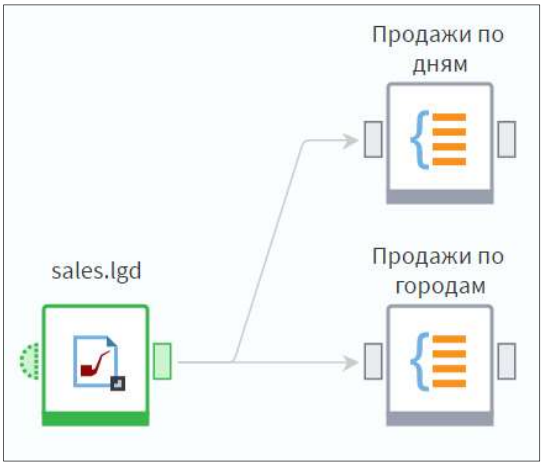


Рисунок 3.15 – Пример установления связей между узлами

В новом узле настроим группировку продаж по сумме и количеству по городам. Для этого нам понадобится указать в списке групп только одно поле – **Город**, а показатели останутся такими же.

На выходе мы получили общие продажи по каждому из 73 городов.

Иногда требуется получить список всех значений какого-либо поля. В этом случае можно провести группировку без показателей. Обратите внимание: если выбрать несколько полей, то сохранится иерархия, которая присутствовала в исходных данных. Проведем группировку по полю **Группа клиентов** (рисунок 3.16).

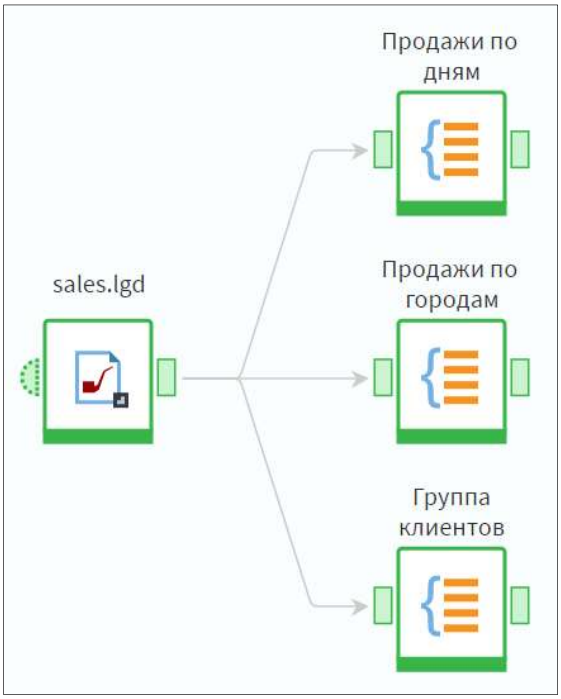


Рисунок 3.16 – Пример установления связей между узлами

Мы получили список всех существующих в нашем наборе групп клиентов. Можно также использовать группировку без измерений, тогда мы получим агрегированное значение по выбранному полю. Отметим, что для этой операции рекомендуется использовать компонент **Таблица в переменные**.

Теперь добавим в наш сценарий узел **Дата и время** и подадим на его вход данные с узла группировки **Продажи по дням**. По сути эти данные – временные ряды продаж товаров (рисунок 3.16).

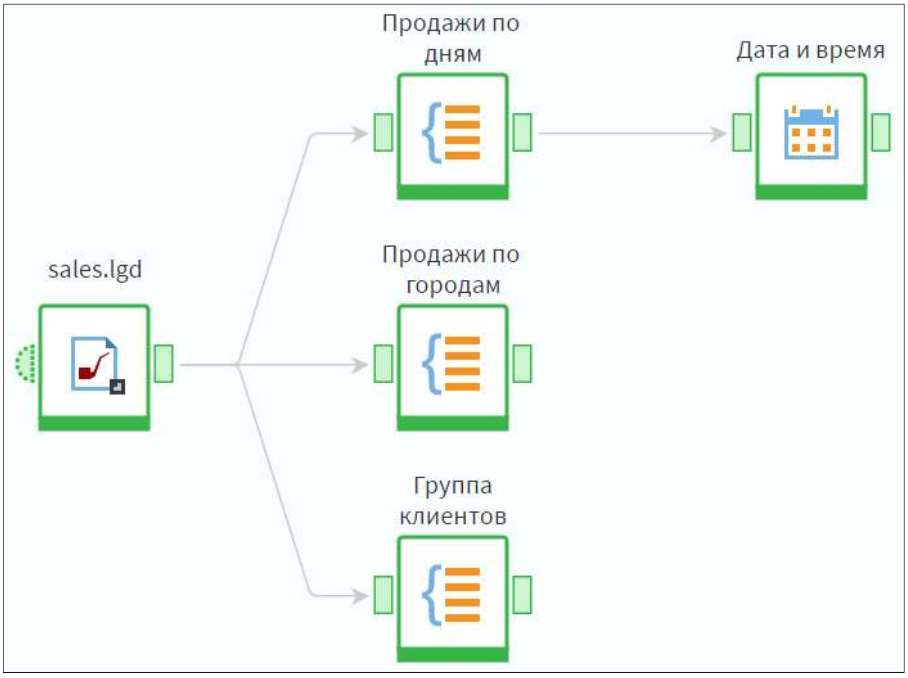


Рисунок 3.16 – Пример установления связей между узлами Продажи по дням и Дата и время

Пусть нам необходимо получить продажи по месяцам года. Очевидно, что для этого нужно сделать группировку, но поля с месяцем продажи в нашем наборе нет. Однако его легко получить из поля **Дата** с помощью узла **Дата и Время**.

Данный узел позволяет провести следующее преобразование: на основе поля типа **Дата/Время** формируется одно или несколько дополнительных полей, в которых указывается, к какому заданному интервалу времени (год, квартал и так далее) принадлежит запись данных. Перейдем в настройку узла **Дата и время**.

Перед нами окно задания параметров преобразования. Слева расположена область **Поле**, в нее попадают только те поля входного набора данных, которые имеют тип **Дата/Время**. Справа находится область **Разбиение**, где можно выбрать формат и тип данных, которые мы хотим получить на выходе. По умолчанию ничего не выбрано.

Нам необходимо поле с месяцем года, поэтому выберем разбиение **Год+Месяц** с типом **Дата/Время** (выберем оба варианта: **Дата начала и Дата конца**). Дополнительно выделим из поля **Дата** номер месяца и неделю продажи с типами **Число** и **Строка** соответственно (рисунок 3.17).

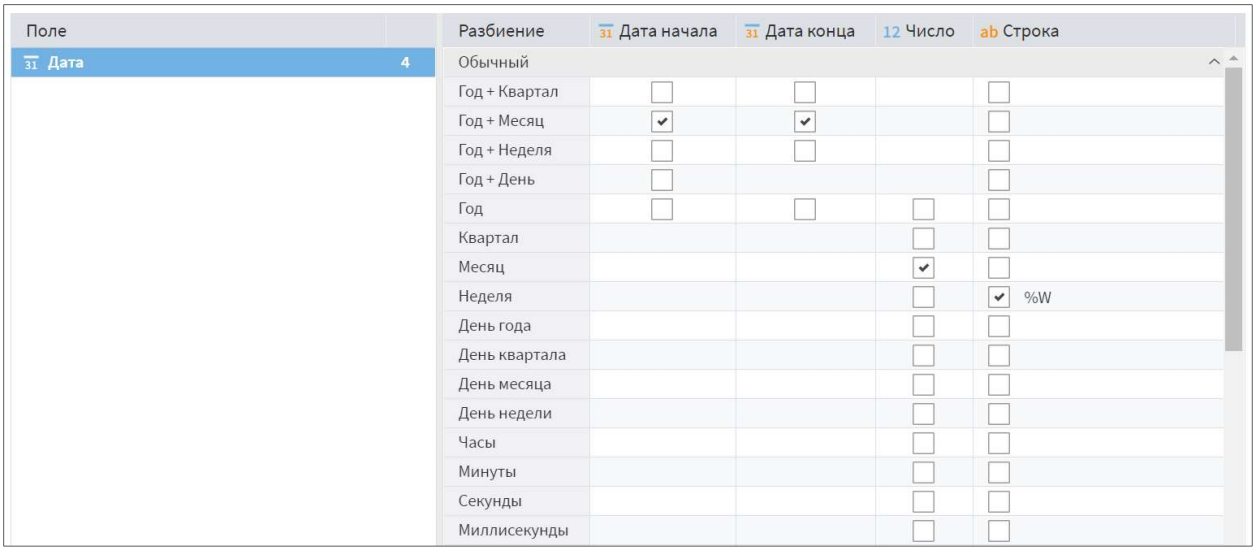


Рисунок 3.17 – Настройка узла Дата и время

При выборе типа Строка в выпадающем списке справа от установленного флага можно выбрать формат представления даты в строковом виде.

Формат представления даты в строковом виде задается с помощью специальных маркеров. Рассмотрим маркеры, доступные для представления недели:

- %W – номер недели в году;
- %e – номер недели в квартале;
- %w – номер недели в месяце.

При задании пользовательского формата можно использовать сочетание произвольного текста и этих маркеров. Например, для даты 01.01.2018 можно задать значение: **произвольный текст %W, произвольный текст**. В результате которого на выходе мы получим: **произвольный текст 1, произвольный текст**.

В наш набор данных добавились четыре новых поля, которые характеризуют поле **Дата**. Обратите внимание, что поля **Дата (Год+Месяц, Последний день)** и **Дата (Год+Месяц, Первый день)** обозначают один и тот же период, только в одном случае указывается дата конца этого периода, а в другом – дата начала (рисунок 3.18).

Выходной набор данных

#	Дата	Дата (Год + Месяц, Последний день)	Дата (Год + Месяц, Первый де…	Дата (Месяц)	Дата (Неделя)	Товар
1	01.03.2016, 00:00	31.03.2016, 00:00	01.03.2016, 00:00	3	10	Доска паркетная UPOFLOOR Loc 2085x18
2	01.03.2016, 00:00	31.03.2016, 00:00	01.03.2016, 00:00	3	10	Гвоздь оцинкованный 1.4x25, 300 штук,
3	01.03.2016, 00:00	31.03.2016, 00:00	01.03.2016, 00:00	3	10	Гипсокартон АБС 9.5x1200x2500, кв.м.
4	01.03.2016, 00:00	31.03.2016, 00:00	01.03.2016, 00:00	3	10	Гипсокартон БЕЛГИПС 12.5x1200x2500, к
5	01.03.2016, 00:00	31.03.2016, 00:00	01.03.2016, 00:00	3	10	Гипсокартон БЕЛГИПС 9.5x1200x2500, вл
6	01.03.2016, 00:00	31.03.2016, 00:00	01.03.2016, 00:00	3	10	Гипсокартон БЕЛГИПС 9.5x1200x2500, кв
7	01.03.2016, 00:00	31.03.2016, 00:00	01.03.2016, 00:00	3	10	Гипсокартон КНАУФ 12.5x1200x2500, кв.м
8	01.03.2016, 00:00	31.03.2016, 00:00	01.03.2016, 00:00	3	10	Гипсокартон КНАУФ 12.5x1200x2500, огн
9	01.03.2016, 00:00	31.03.2016, 00:00	01.03.2016, 00:00	3	10	Лист стекломагнезитовый СМЛ 1220x24
10	01.03.2016, 00:00	31.03.2016, 00:00	01.03.2016, 00:00	3	10	Пигмент блескообразующий TIGIANA ae

Рисунок 3.18 – Быстрый просмотр активированного выходного порта Дата и время

Теперь мы можем провести группировку по месяцам (рисунок 3.19).

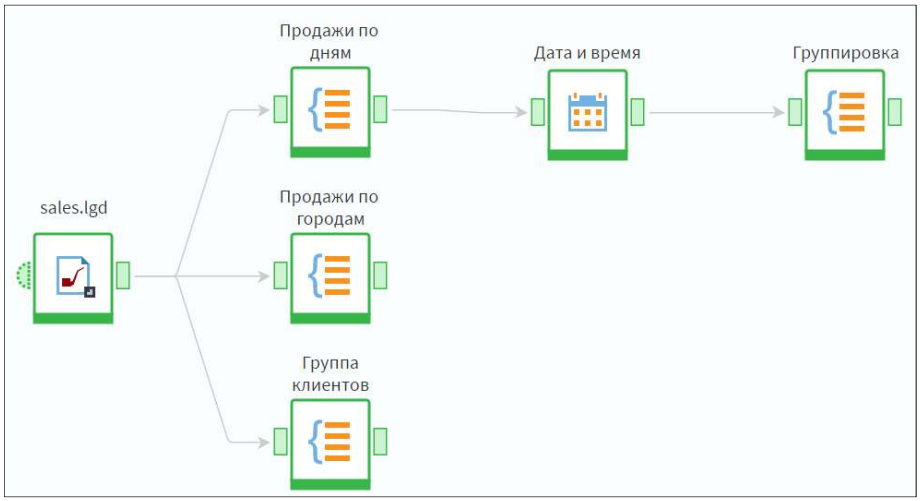


Рисунок 3.19 – Пример установления связей между узлами

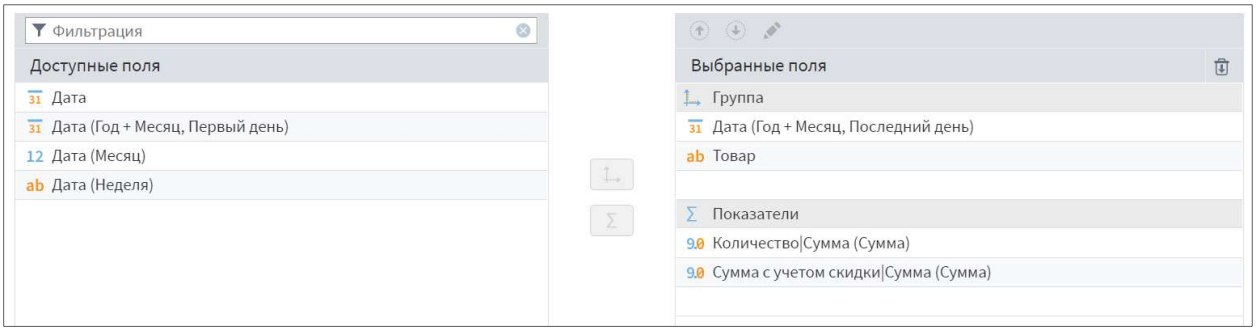


Рисунок 3.20 – Пример настройки узла Группировка

В результате количество записей сильно сократилось (до 8 410), так как мы получили продажи по месяцам.

Компонент **Дата и время** позволяет также получить интервал времени, к которому принадлежит запись данных, в формате, определенном международным стандартом **ISO 8601** (рисунок 3.20).

Новый период (год, квартал, месяц) в данном формате всегда начинается с понедельника и не всегда совпадает с календарным. Первой неделей года считается та, которая содержит первый четверг в году по Григорианскому календарю. Аналогичным образом определяется первая неделя квартала и месяца.

3.3. Обогащение данных

В практике анализа данных достаточно часто возникает ситуация, когда нужные данные приходится собирать из нескольких таблиц. Такая необходимость возникает в следующих случаях:

- данные, которые требуется для анализа, «разбросаны» по нескольким таблицам;
- данные в исходной таблице несут недостаточно информации для анализа, и требуется дополнить их данными из других источников.

Например, для решения задачи прогнозирования во множестве используются один или два выходных признака, тогда как для надежной классификации требуется не менее трех или четырех. В этом случае недостающие признаки можно взять из другой таблицы.

Таким образом, если для решения одной аналитической задачи информативность множества данных может быть достаточна или даже избыточна, то для другой она окажется недостаточной и потребует

обогащения. Данная процедура включает в себя несколько операций. Рассмотрим их.

Слияние данных. При необходимости объединить две таблицы в одну выполняется процедура **слияния** (англ.: merge).

Таблица, к которой в процессе слияния добавляются данные из другой, называется **главной**, или основной; вторую таблицу, данные из которой добавляются к главной, называют **присоединяемой**.

Главная и присоединяемая таблицы должны иметь одно или несколько одинаковых полей, на основе которых будет проводиться связывание этих таблиц, это – **ключевые поля**.

Остальные поля, уникальные для каждой из таблиц, могут быть присоединены к результирующему набору после слияния.

Внутреннее соединение. Существует несколько операций слияния, которые применяются в зависимости от того, какие данные и в каком виде должны быть объединены в результирующей таблице. Рассмотрим их.

Первая операция – **внутреннее соединение**. Оно позволяет получить в результирующем наборе только те записи, для которых значения ключевых полей совпадают.

То есть в таблице, полученной в результате внутреннего соединения, останутся только записи, которые содержат одинаковые значения в заданном поле (или полях).

Рассмотрим пример внутреннего соединения (рисунок 3.21). Пусть имеется исходная выборка данных, в которой содержатся сведения о поставляемых товарах: наименования и поставщики (таблица 1), – а также выборка данных, содержащая историю продаж этих товаров (таблица 2).

1 — Главная таблица

Поставщик	Товар
ООО «Невод»	Рыба свежая
ЗАО «Молкомбинат»	Сыр
ООО «Птичник»	Куры
ЗАО «Овощевод»	Капуста
ОАО «Хладокомбинат»	Мороженое
ЗАО «Маслосырбаза»	Молоко
ОАО «Горпищекомбинат»	Кетчуп

2 — Присоединяемая таблица

Дата	Поставщик	Товар	Кол-во
02.04.2017	ЗАО «Молкомбинат»	Сыр	150
08.04.2017	ООО «Птичник»	Куры	200
11.04.2017	ЗАО «Маслосырбаза»	Молоко	170
15.04.2017	ООО «Птичник»	Куры	100
22.04.2017	ЗАО «Молкомбинат»	Сыр	80
27.04.2017	ОАО «Хладокомбинат»	Мороженое	250
27.04.2017	ООО «Невод»	Рыба свежая	160

Рисунок 3.21 – Пример внутреннего соединения

Используем поля **Поставщик** и **Товар** в качестве ключевых и проведем операцию внутреннего соединения.

Результирующая таблица (таблица 3) содержит только те записи, для которых значения в ключевых полях одинаковые. При этом ЗАО «Овощевод» и ОАО «Горпищекомбинат», которые присутствуют в общем списке поставщиков, в результирующую таблицу внесены не были, поскольку в таблице истории продаж записи с их участием отсутствуют (рисунок 3.22).

Поставщик	Товар	Дата	Кол-во
ООО «Невод»	Рыба свежая	27.04.2017	160
ЗАО «Молкомбинат»	Сыр	02.04.2017	150
ЗАО «Молкомбинат»	Сыр	22.04.2017	80
ООО «Птичник»	Куры	08.04.2017	200
ООО «Птичник»	Куры	15.04.2017	100
ОАО «Хладокомбинат»	Мороженое	27.04.2017	250
ЗАО «Маслосырбаза»	Молоко	11.04.2017	170

Рисунок 3.22 – Пример результата внутреннего соединения

Таким образом, внутреннее соединение позволило из всех поставщиков отобрать только тех, закупка у которых проводилась за наблюдаемый период времени.

При типах операций **левого** и **правого соединения** все записи одной таблицы дополняются значениями из другой, если значения этих записей по ключевым полям совпадают.

Например, если таблицы связываются по полю **Товар**, и существуют записи, где значения данного поля в обеих таблицах идентичны, то эти записи будут дополнены значениями, которые отсутствуют в одной таблице, но присутствуют в другой.

Фактически этот механизм позволяет добавлять поля из одной таблицы в другую, но не по всем записям, а только по тем, значения которых в ключевом поле совпадают для обеих таблиц (рисунок 3.23).

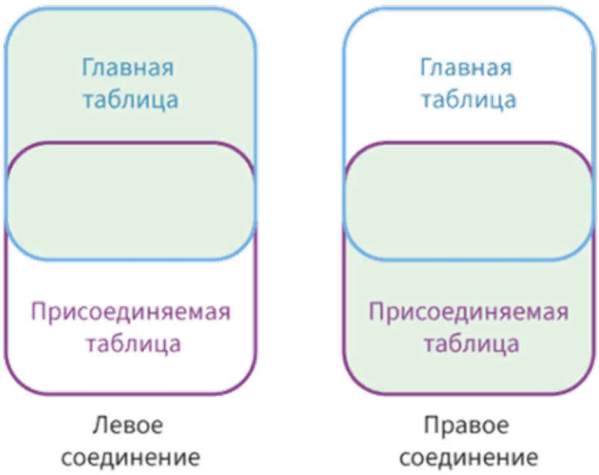


Рисунок 3.23 – Пример левого и правого соединения. При левом соединении записи главной таблицы дополняется значениями из присоединяемой, а при правом – наоборот

Рассмотрим пример **левого соединения**. Пусть к таблице 1, где содержится только информация о товарах и поставщиках, требуется добавить поля, которые отражают историю продаж (**Дата** и **Количество**). Для этого в качестве ключевого поля будет использоваться поле **Товар** (рисунок 3.24).

1 — Главная таблица

Поставщик	Товар
ООО «Невод»	Рыба свежая
ЗАО «Молкомбинат»	Сыр
ООО «Птичник»	Куры
ЗАО «Овощевод»	Капуста
ОАО «Хладокомбинат»	Мороженое
ЗАО «Маслосырбаза»	Молоко
ОАО «Горпищекомбинат»	Кетчуп

2 — Присоединяемая таблица

Дата	Поставщик	Товар	Кол-во
02.04.2017	ЗАО «Молкомбинат»	Сыр	150
08.04.2017	ООО «Птичник»	Куры	200
11.04.2017	ЗАО «Маслосырбаза»	Молоко	170
15.04.2017	ООО «Птичник»	Куры	100
22.04.2017	ЗАО «Молкомбинат»	Сыр	80
27.04.2017	ОАО «Хладокомбинат»	Мороженое	250
27.04.2017	ООО «Невод»	Рыба свежая	160

Рисунок 3.24 – Пример левого соединения

При операции левого соединения получим следующее (таблица 3). Как мы видим, для товаров **Капуста** и **Кетчуп** отсутствуют значения в полях **Дата** и **Количество**, так как в присоединяемой таблице не было данных по этим товарам (рисунок 3.25).

Поставщик	Товар	Дата	Кол-во
ООО «Невод»	Рыба свежая	27.04.2017	160
ЗАО «Молкомбинат»	Сыр	02.04.2017	150
ЗАО «Молкомбинат»	Сыр	22.04.2017	80
ООО «Птичник»	Куры	08.04.2017	200
ООО «Птичник»	Куры	15.04.2017	100
ЗАО «Овощевод»	Капуста		
ОАО «Хладокомбинат»	Мороженое	27.04.2017	250
ЗАО «Маслосырбаза»	Молоко	11.04.2017	170
ОАО «Горпищекомбинат»	Кетчуп		

Рисунок 3.25 – Пример результата левого соединения

Рассмотрим пример **правого соединения**, когда имеется информация о наименовании товаров и истории продаж, а о поставщике информация отсутствует. Получить ее можно через поле связи **Товар**, поскольку в одной таблице каждый товар связан со своим поставщиком, а в таблице с историей продаж – с датой и количеством. Главной будет таблица с историей продаж, а присоединяемой – с информацией о поставщике (рисунок 3.26).

1 — Главная таблица

Дата	Товар	Кол-во
02.04.2017	Сыр	150
08.04.2017	Куры	200
11.04.2017	Молоко	170
15.04.2017	Куры	100
22.04.2017	Сыр	80
27.04.2017	Мороженое	250
27.04.2017	Рыба свежая	160

2 — Присоединяемая таблица

Поставщик	Товар
ООО «Невод»	Рыба свежая
ЗАО «Молкомбинат»	Сыр
ООО «Птичник»	Куры
ЗАО «Овощевод»	Капуста
ОАО «Хладокомбинат»	Мороженое
ЗАО «Маслосырбаза»	Молоко
ОАО «Горпищекомбинат»	Кетчуп

Рисунок 3.26 – Пример правого соединения

Применив **правое соединение**, получим следующий результат (таблица 3). Как мы видим, для поставщиков ЗАО «Овощевод» и ОАО «Горпищекомбинат» отсутствуют значения полей Дата, Товар и Количество, так как в главной таблице не было записей о продажах товаров этих поставщиков (рисунок 3.27).

Дата	Товар	Кол-во	Поставщик
27.04.2017	Рыба свежая	160	ООО «Невод»
02.04.2017	Сыр	150	ЗАО «Молкомбинат»
22.04.2017	Сыр	80	ЗАО «Молкомбинат»
08.04.2017	Куры	200	ООО «Птичник»
15.04.2017	Куры	100	ООО «Птичник»
			ЗАО «Овощевод»
27.04.2017	Мороженое	250	ОАО «Хладокомбинат»
11.04.2017	Молоко	170	ЗАО «Маслосырбаза»
			ОАО «Горпищекомбинат»

Рисунок 3.27 – Пример результата правого соединения

Еще один тип операции слияния – **полное соединение**. В результирующий набор включаются все строки и поля как главной, так и присоединяемой таблиц. При этом, если в некоторой записи значения ключевого поля для обеих таблиц совпадают, то все поля этой записи заполняются соответствующими значениями.

Если совпадение по ключевому полю отсутствует, то остальные поля такой записи будут заполнены пустыми значениями. Фактически это означает, что для значений главной таблицы отсутствуют соответствующие значения в присоединяемой (рисунок 3.28).

Рисунок 3.28 – Пример полного соединения

Рассмотрим полное соединение для главной и присоединяемой таблиц из предыдущего примера (рисунок 3.29).

1 — Главная таблица

Дата	Товар	Кол-во
02.04.2017	Сыр	150
08.04.2017	Куры	200
11.04.2017	Молоко	170
15.04.2017	Куры	100
22.04.2017	Сыр	80
27.04.2017	Мороженое	250
27.04.2017	Рыба свежая	160

2 — Присоединяемая таблица

Поставщик	Товар
ООО «Невод»	Рыба свежая
ЗАО «Молкомбинат»	Сыр
ООО «Птичник»	Куры
ЗАО «Овощевод»	Капуста
ОАО «Хладокомбинат»	Мороженое
ЗАО «Маслосырбаза»	Молоко
ОАО «Горпищекомбинат»	Кетчуп

Рисунок 3.29 – Пример полного соединения

Применение **полного соединения** даст результат, указанный в таблице 3: если информация по определенному товару и поставщику присутствует в обеих таблицах, то она полностью объединяется.

Однако для товаров **Капуста** и **Кетчуп** отсутствует информация о продажах, а, значит, о дате и количестве. Поэтому соответствующие ячейки остаются пустыми.

Полное соединение позволяет полностью связать информацию таблиц (рисунок 3.30).

Дата	Товар	Кол-во	Поставщик
02.04.2017	Сыр	150	ЗАО «Молкомбинат»
08.04.2017	Куры	200	ООО «Птичник»
11.04.2017	Молоко	170	ЗАО «Маслосырбаза»
15.04.2017	Куры	100	ООО «Птичник»
22.04.2017	Сыр	80	ЗАО «Молкомбинат»
27.04.2017	Мороженое	250	ОАО «Хладокомбинат»
27.04.2017	Рыба свежая	160	ООО «Невод»
			ЗАО «Овощевод»
			ОАО «Горпищекомбинат»

Рисунок 3.30 – Пример результата полного соединения

Последняя операция – **разность**. При ее использовании в результате мы получаем только строки, которые присутствовали в главной таблице, но отсутствовали в присоединяемой.

Таким образом, из основной таблицы как бы вычитаются записи, найденные в присоединяемой (рисунок 3.31).

Рисунок 3.31 – Пример разности

Для демонстрации примера операции **разности** необходимо поменять главную и присоединяемую таблицы из прошлого примера местами: главной будет таблица со списком поставщиков, а присоединяемой – с историей продаж (рисунок 3.32).

1 — Главная таблица

Поставщик	Товар
ООО «Невод»	Рыба свежая
ЗАО «Молкомбинат»	Сыр
ООО «Птичник»	Куры
ЗАО «Овощевод»	Капуста
ОАО «Хладокомбинат»	Мороженое
ЗАО «Маслосырбаза»	Молоко
ОАО «Горпищекомбинат»	Кетчуп

2 — Присоединяемая таблица

Дата	Товар	Кол-во
02.04.2017	Сыр	150
08.04.2017	Куры	200
11.04.2017	Молоко	170
15.04.2017	Куры	100
22.04.2017	Сыр	80
27.04.2017	Мороженое	250
27.04.2017	Рыба свежая	160

Рисунок 3.32 – Пример разности

В результате слияния по полю **Товар** мы получили список поставщиков, товары которых не продавались (рисунок 3.33).

Поставщик	Товар
ЗАО «Овощевод»	Капуста
ОАО «Горпищекомбинат»	Кетчуп

Рисунок 3.33 – Пример результата разности

Соединение. Теперь рассмотрим такой способ обогащения данных, как **Соединение**.

Данная операция позволяет дополнить один набор данных – главную таблицу – полями из присоединяемых наборов.

Ключевые поля здесь отсутствуют, и соединение происходит построчно: каждая запись главной таблицы соединяется с записями дополнительных таблиц с таким же порядковым номером.

В отличие от слияния, количество присоединяемых таблиц не ограничивается двумя (рисунок 3.34).

Рисунок 3.34 – Пример соединения

Рассмотрим пример **соединения**. Возьмем уже известную нам таблицу с поставщиками и список дат, в которые проводились продажи, с количеством проданного товара. Пусть мы уверены, что данные о поставщиках отсортированы в соответствии с этими датами (рисунок 3.35).

1 — Главная таблица

Поставщик	Товар
ЗАО «Молкомбинат»	Сыр
ООО «Птичник»	Куры
ЗАО «Маслосырбаза»	Молоко
ОАО «Хладокомбинат»	Мороженое
ООО «Невод»	Рыба свежая
ЗАО «Овощевод»	Капуста
ОАО «Горпищекомбинат»	Кетчуп

2 — Присоединяемая таблица

Дата	Кол-во
02.04.2017	150
11.04.2017	170
15.04.2017	100
27.04.2017	250
27.04.2017	160

Рисунок 3.35 – Пример соединения

Тогда, используя операцию **соединения**, мы получим следующий результат. Недостающие записи в присоединяемых полях остались пустыми (рисунок 3.36).

Поставщик	Товар	Дата	Кол-во
ЗАО «Молкомбинат»	Сыр	02.04.2017	150
ООО «Птичник»	Куры	11.04.2017	170
ЗАО «Маслосырбаза»	Молоко	15.04.2017	100
ОАО «Хладокомбинат»	Мороженое	27.04.2017	250
ООО «Невод»	Рыба свежая	27.04.2017	160
ЗАО «Овощевод»	Капуста		
ОАО «Горпищекомбинат»	Кетчуп		

Рисунок 3.36 – Пример результата соединения

Дополнение данных – способ обогащения, очень похожий на операцию левого соединения, но позволяющий дополнять главную таблицу полями из неограниченного количества источников.

Для связи главной и присоединяемых таблиц, каки в слиянии, используются ключевые поля. При этом разные присоединяемые таблицы могут быть связаны с главной по разным полям.

Данный способ обычно применяется при необходимости обогащения набора данных полями из различных справочников (рисунок 3.37).

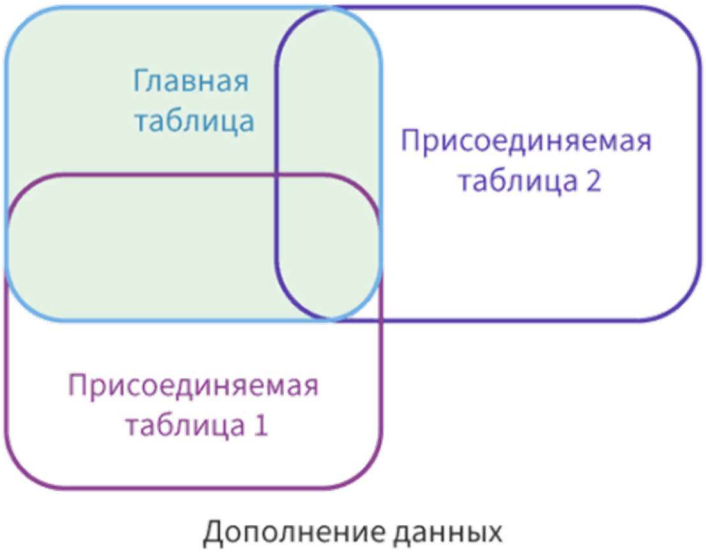


Рисунок 3.37 – Пример дополнения данных

Для примера дополнения данных снова возьмем нашу таблицу с историей продаж и используем ее в качестве главной таблицы (рисунок 3.38). Пусть нам необходимо дополнять ее информацией о поставщиках, а также о товарных группах, к которым относятся проданные товары. В качестве ключевого о обоих случаях используем поле **Товар**.

1 — Главная таблица

Дата	Товар	Кол-во
02.04.2017	Сыр	150
08.04.2017	Куры	200
11.04.2017	Молоко	170
15.04.2017	Куры	100
22.04.2017	Сыр	80
27.04.2017	Мороженое	250
27.04.2017	Рыба свежая	160

2 — Присоединяемая таблица 1

Поставщик	Товар
ООО «Невод»	Рыба свежая
ЗАО «Молкомбинат»	Сыр
ООО «Птичник»	Куры
ЗАО «Овощевод»	Капуста
ОАО «Хладокомбинат»	Мороженое
ЗАО «Маслосырбаза»	Молоко
ОАО «Горпищекомбинат»	Кетчуп

3 — Присоединяемая таблица 2

Группа товара	Товар
Молочные товары	Сыр
Молочные товары	Мороженое
Молочные товары	Молоко
Птица	Куры
Рыба	Рыба свежая
Овощи	Капуста
Соусы	Кетчуп

Рисунок 3.38 – Пример дополнения данных

Тогда результирующая таблица будет содержать информацию о поставщике и группе товаров по каждой его продаже (рисунок 3.39).

Дата	Товар	Кол-во	Поставщик	Группа товара
02.04.2017	Сыр	150	ЗАО «Молкомбинат»	Молочные товары
08.04.2017	Куры	200	ООО «Птичник»	Птица
11.04.2017	Молоко	170	ЗАО «Маслосырбаза»	Молочные товары
15.04.2017	Куры	100	ООО «Птичник»	Птица
22.04.2017	Сыр	80	ЗАО «Молкомбинат»	Молочные товары
27.04.2017	Мороженое	250	ОАО «Хладокомбинат»	Молочные товары
27.04.2017	Рыба свежая	160	ООО «Невод»	Рыба

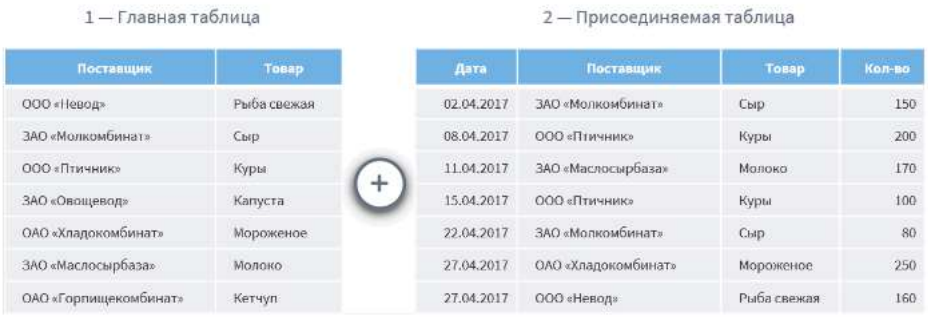
Рисунок 3.39 – Пример результата дополнения данных

Объединение. Последний способ обогащения данных – объединение. Он применяется в тех случаях, когда к строкам главной таблицы требуется добавить все строки присоединяемой, при этом, в отличие от ранее рассмотренных способов, строки добавляются снизу (рисунок 3.40).

Рисунок 3.40 – Пример объединения

Рассмотрим пример **объединения**. Снова возьмем выборку данных, в которой содержатся сведения о поставляемых товарах (таблица 1), а также выборку, содержащую историю их продаж (таблица 2).

Необходимо объединить информацию так, чтобы таблицы содержали два одинаковых поля: **Поставщик** и **Товар**, – а также поля **Дата** и **Количество**, которые находятся только во второй таблице (рисунок 3.41).



Поставщик	Товар
ООО «Невод»	Рыба свежая
ЗАО «Молкомбинат»	Сыр
ООО «Птичник»	Куры
ЗАО «Овощевод»	Капуста
ОАО «Хладокомбинат»	Мороженое
ЗАО «Маслосырбаза»	Молоко
ОАО «Горпищекомбинат»	Кетчуп

Дата	Поставщик	Товар	Кол-во
02.04.2017	ЗАО «Молкомбинат»	Сыр	150
08.04.2017	ООО «Птичник»	Куры	200
11.04.2017	ЗАО «Маслосырбаза»	Молоко	170
15.04.2017	ООО «Птичник»	Куры	100
22.04.2017	ЗАО «Молкомбинат»	Сыр	80
27.04.2017	ОАО «Хладокомбинат»	Мороженое	250
27.04.2017	ООО «Невод»	Рыба свежая	160

Рисунок 3.41 – Пример объединения

Результатом объединения по полям Товар и Поставщик будет таблица, представленная на рисунке 3.42.

Поставщик	Товар	Дата	Кол-во
ООО «Невод»	Рыба свежая		
ЗАО «Молкомбинат»	Сыр		
ООО «Птичник»	Куры		
ЗАО «Овощевод»	Капуста		
ОАО «Хладокомбинат»	Мороженое		
ЗАО «Маслосырбаза»	Молоко		
ОАО «Горпищекомбинат»	Кетчуп		
ЗАО «Молкомбинат»	Сыр	02.04.2017	150
ООО «Птичник»	Куры	08.04.2017	200
ЗАО «Маслосырбаза»	Молоко	11.04.2017	170
ООО «Птичник»	Куры	15.04.2017	100
ЗАО «Молкомбинат»	Сыр	22.04.2017	80
ОАО «Хладокомбинат»	Мороженое	27.04.2017	250
ООО «Невод»	Рыба свежая	27.04.2017	160

Рисунок 3.42 – Пример результата объединения

3.4. Лабораторная работа №8 «Обогащение данных»

Компонент **Слияние** позволяет дополнить основной набор данных полями из присоединяемого набора. Слияние осуществляется с помощью связей, которые создаются между ключевыми полями входных наборов.

В компоненте доступны 5 операций слияния:

- Внутреннее соединение;
- Левое соединение;
- Правое соединение;

- Полное соединение;
- Разность.

Рассмотрим указанные операции. Для этого нам понадобятся наборы данных **sales.lgd** и **plan.lgd**.

Создадим в Loginom новый пакет. Каждый компонент для удобства будем рассматривать в отдельном модуле. Откроем сценарий первого модуля и импортируем указанные наборы.

Файл sales.lgd уже знаком нам по занятию предыдущего блока, в нем содержится информация о продажах строительных товаров за некоторый период времени в разных городах.

Сгруппируем продажи по городам, похожую операцию мы проводили в предыдущем занятии.

Для узла группировки необходимо сделать следующие настройки: поле **Город** добавить в **Группы**, поле **Сумма с учетом скидки** – в показатели с вариантом агрегации **Сумма** (рисунок 3.43).

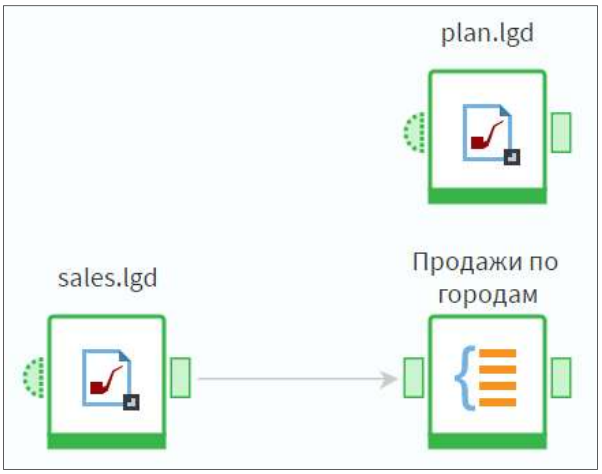


Рисунок 3.43 – Установление связей между узлами

Мы получили набор из 73-х городов, в которых произошли продажи.

Теперь импортируем файл **plan.lgd**, содержащий информацию о плане продаж (суммы) по городам.

Пусть нам необходимо для дальнейшего анализа иметь в одном наборе данных суммы прошлых продаж и план продаж. Отметим, что план содержит только 63 города, то есть их меньше, чем городов, по которым были продажи.

Перенесем в область построения компонент **Слияние**. Он имеет два входных порта: **Главная таблица** и **Присоединяемая таблица** – и только один выходной порт. Результат обработки будет зависеть от выбранного типа операции слияния.

Подадим на порт **Главная таблица** данные с узла группировки **Продажи по городам**, а на порт **Присоединяемая таблица** данные набора **plan.lgd** и перейдем в настройку узла (рисунок 3.44).

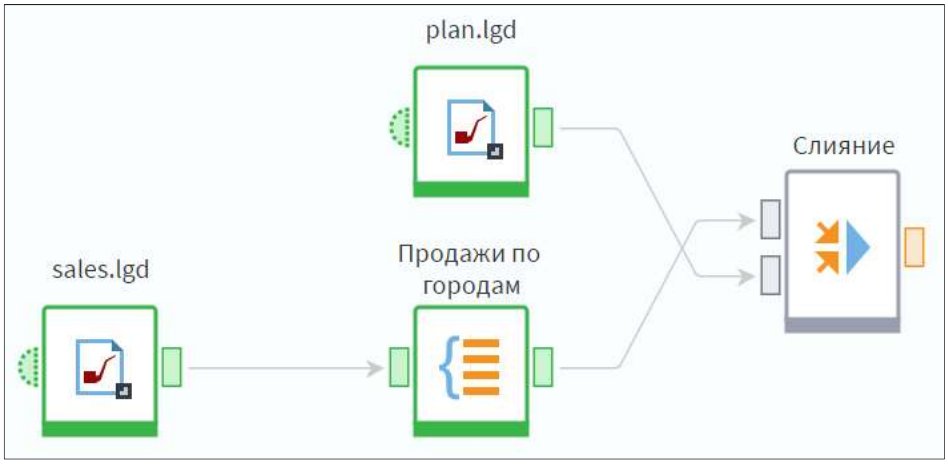


Рисунок 3.44 – Установление связей с узлом Слияние

Для начала необходимо определиться с типом операций слияния. По умолчанию указан тип **Левое соединение**. Раскроем список операций.

Мы уже знаем, что всего вариантов слияния пять. Разберем, что мы получим на выходе, если выбрать самый первый – **Внутреннее соединение**.

Основную часть окна настройки занимает область сопоставления полей. Она содержит два списка: **Столбцы основного набора данных** и **Столбцы присоединяемого набора данных**. Здесь создается связь между наборами: каким полям из главной таблицы соответствуют поля присоединяемой.

Для связи необходимо с помощью механизма **Drag&Drop** перетащить поле из одного списка на поле, с которым его нужно связать, из другого списка. Связывание допускается только для полей с одинаковыми типами данных.

Создадим связь, как показано на рисунке 4.45, и перейдем к просмотру результата.

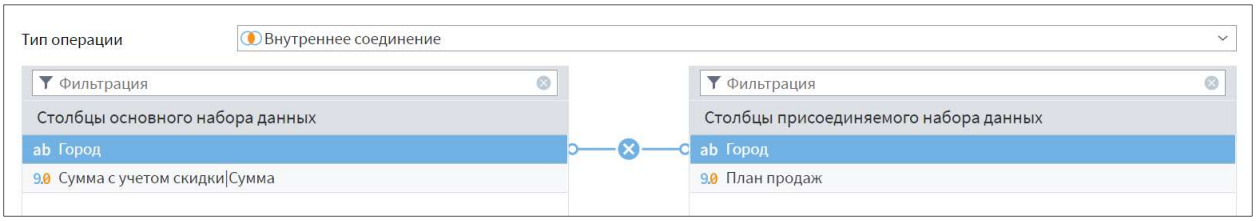


Рисунок 4.45 – Настройка внутреннего соединения

На выходе мы получили набор данных с новым полем **План продаж**. В наборе содержатся только те записи, которые присутствовали в обоих выходных наборах, это 62 города.

Добавим еще один узел **Слияние** и создадим такие же связи с его портами (рисунок 4.46).

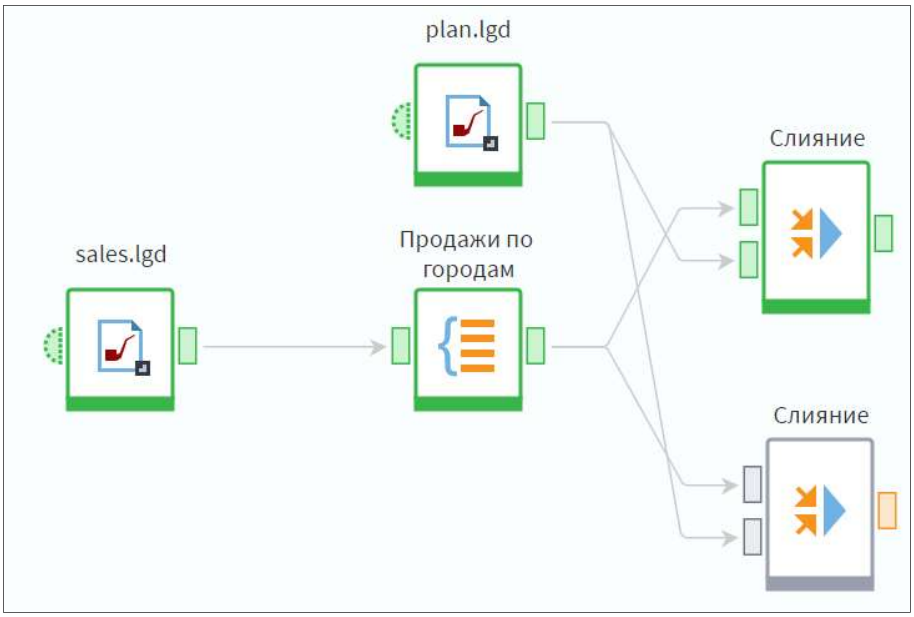


Рисунок 4.46 – Пример установление связи с новым узлом Слияние

В настройках оставим по умолчанию – **Левое соединение**, - связи между полями будут такие же, как и в предыдущем узле.

Дополнительно установим флаг **Добавлять присоединяемые ключевые поля**. Это позволит на выходе узла видеть оба ключевых поля **Город**: и из главной таблицы, и из присоединяемой.

Если флаг не устанавливать, на выходе будет только поле главной таблицы, как в предыдущем узле.

В результате мы получили 73 записи по всем городам, которые содержались в наборе с продажами.

Если для города не нашлось записи в файле **plan.lgd**, для него указаны пустые значения (null) в полях **Город** и **План продаж** из присоединяемого набора (рисунок 4.47).

Слияние • Быстрый просмотр

Выходной набор данных

#	ab Город	9.0 Сумма с учетом скидки|Сумма	9.0 План прод…	ab Город
1	Балашиха	9 823 339,29	14 719 099,00	Балашиха
2	Барнаул	16 145 202,69	17 749 851,00	Барнаул
3	Великие Луки	14 877 644,50		<null>
4	Екатеринбург	1 882 384,41	2 094 496,00	Екатеринбург
5	Зеленоград	26 587 113,84	36 583 861,00	Зеленоград
6	Иваново	30 147 324,08	32 823 203,00	Иваново
7	Иркутск	572 372,28	767 634,00	Иркутск
8	Москва	476 055 694,18	599 309 701,00	Москва
9	Нальчик	2 733 682,58	3 567 268,00	Нальчик
10	Нижний Новгород	202 129 396,56	251 400 516,00	Нижний Новгород
11	Новороссийск	4 669 672,13	5 565 961,00	Новороссийск
12	Пермь	19 259 598,13	23 457 887,00	Пермь
13	Петрозаводск	6 893 788,11	9 227 014,00	Петрозаводск
14	Самара	8 051 614,33	8 413 562,00	Самара
15	Сергиев Посад	10 933 750,80		<null>
16	Тольятти	33 823 757,01	45 307 079,00	Тольятти
17	Уссурийск	1 435 470,84	1 753 330,00	Уссурийск
18	Уфа	20 805 622,08	21 074 959,00	Уфа
19	Владимир	71 025 229,83	106 313 094,00	Владимир

Рисунок 4.47 – Результат слияния данных

Если выбрать **Правое соединение**, мы получим 63 записи, как в наборе **plan.lgd** (здесь и далее флаг Добавлять присоединяемые ключевые поля устанавливать не будем).

Однако в записях из этого набора, для которых город отсутствует в таблице продаж, поля **Город** и **Сумма** с учетом скидки будут пусты. Это связано с тем, что главной в слиянии является таблица по продажам, и значения этих полей берутся из нее. Поэтому при проведении слияния важно правильно определить главную таблицу.

Теперь рассмотрим **Полное соединение**. Оно позволяет соединить каждую строку одной таблицы с каждой строкой второй таблицы, давая в результате все возможные сочетания строк. Это может понадобиться, например, в случае необходимости сравнить каждую строку одного набора с каждой строкой другого с помощью расстояния Левенштейна между строками.

В полном соединении допускается не создавать ни одной связи. Тогда на выходе мы получим все поля из обеих таблиц независимо от того, установлен ли флаг, – так как ключевые поля не выбраны. Перейдем к просмотру результата.

Мы получили 4 599 записей. Каждая запись таблицы с продажами повторяется 63 раза, так как к ней добавляется каждая запись из набора с планом продаж.

Последняя операция, которая доступна в узле, – **Разность**. При ее использовании мы получаем на выходе только записи главной таблицы, для которых не нашлось соответствий по ключевым полям в присоединяемой таблице.

В нашем случае таких записей 11. Для этих городов план продаж не был сформирован.

Перейдем к рассмотрению следующего компонента – **Соединение**. С его помощью один набор данных можно дополнить полями из других наборов, либо переменными.

В отличие от слияния, здесь нет ключевых полей. К каждой записи главного набора присоединяются записи из дополнительных наборов с тем же порядковым номером, то есть первая запись соединяется с первой, вторая со второй и так далее.

С помощью компонента **Соединение** возможно, к примеру, добавить в набор данных поле с позициями объектов в рейтинге.

Например, в ежемесячном рейтинге по продажам сети магазинов достаточно будет отсортировать магазины по убыванию суммы продаж, после чего можно добавить поле с позициями (рисунок 4.48).

Магазин	Сумма
Магазин 4	69 821
Магазин 1	55 071
Магазин 2	42 800
Магазин 3	31 212

Позиция
1 место
2 место
3 место
4 место

Магазин	Сумма	Позиция
Магазин 4	69 821	1 место
Магазин 1	55 071	2 место
Магазин 2	42 800	3 место
Магазин 3	31 212	4 место

Рисунок 4.48 – Пример соединения данных

Еще один, наиболее распространенный случай использования **Соединения**, – необходимость добавить в набор данных поле с каким-либо определенным значением. Это значение может содержаться как в таблице, так и в переменных (рисунок 4.49).

Магазин	Сумма
Магазин 1	55 071
Магазин 2	42 800
Магазин 3	31 212
Магазин 4	69 821

Общая сумма
198 904

Магазин	Сумма	Общая сумма
Магазин 1	55 071	198 904
Магазин 2	42 800	198 904
Магазин 3	31 212	198 904
Магазин 4	69 821	198 904

Рисунок 4.49 – Пример соединения данных с полем определенного значения

Рассмотрим данную ситуацию на примере. Для этого снова используем набор данных **sales.lgd**.

В новом модуле импортируем указанный набор и сгруппируем продажи по городам. Пусть мы хотим добавить к получившемуся набору столбец, содержащий общую сумму продаж по всем городам. Добавим в сценарий узел **Таблица в переменные**, чтобы получить эту сумму (рисунок 4.50).

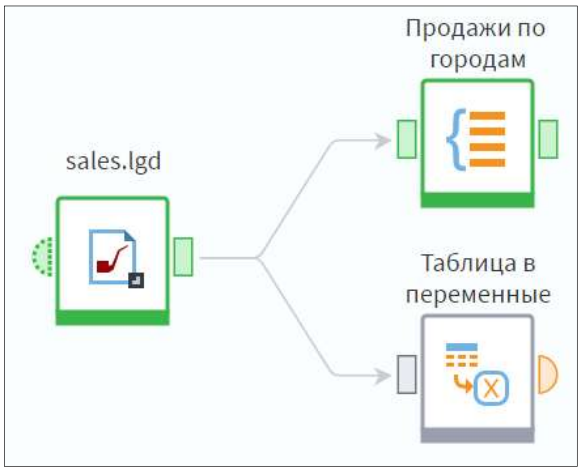


Рисунок 4.50 – Пример установление связей между узлами

В узле выполним настройку как показано на рисунке 4.51.

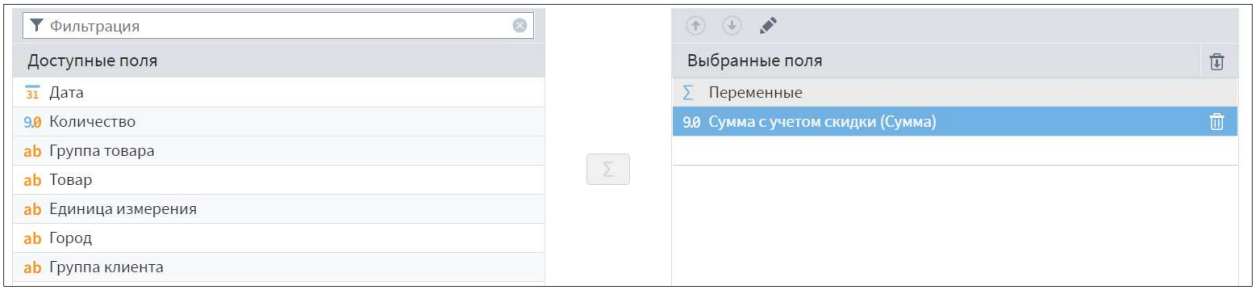


Рисунок 4.51 – Настройка узла Таблица в переменные

На выходе получим переменную, которое содержит исходное значение.

Теперь добавим в область построения узел **Соединение**. По умолчанию он имеет всего один порт – **Главная таблица**. Дополнительные порты нам

необходимо добавить самостоятельно. Это связано с тем, что они могут быть двух типов: **Набор данных** и **Переменные**.

Нажмем кнопку **Добавить еще один порт**. Выберем вариант **Переменные**.

Теперь мы можем подать на порты соответствующие данные. Можно добавить любое количество дополнительных портов. Кроме указанного способа это можно сделать, протянув к кнопке **Добавить еще один порт** связь от выходного порта другого узла (рисунок 4.52).

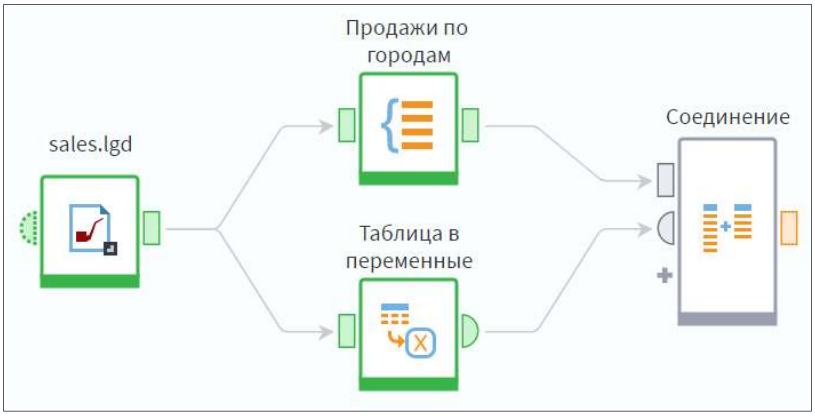


Рисунок 4.52 – Добавление в сценарий узла Соединение

Перейдем в настройку узла Соединение. В поданных на вход узла наборах данных может быть разное количество строк. Данный параметр определяет, каким образом меньший набор будет дополняться до больше:

- **Не дополнять** – меньший набор будет дополнен ‘до большего пустыми значениями;
- **Повторять набор данных** – значения в полях меньшего набора будут повторяться: после последней строки поля снова будет идти первая, затем вторая и так далее;
- **Дополнять последней строчкой** – значение из последней записи добавляемого поля меньшего набора заполнит все недостающие строки.

Следующий параметр позволяет выбрать, какому из входных наборов будет соответствовать количество строк:

- **Максимальному набору** – значение по умолчанию;
- **Минимальному набору**;
- **Определяется набором** – при выборе данного варианта станет активна серая область ниже, где можно будет выбрать любой из поданных на вход узла наборов данных.

Мы ходим добавить в главную таблицу столбец со значением переменной. Выберем параметры, как показано на рисунке 4.53, и перейдем к просмотру результата.

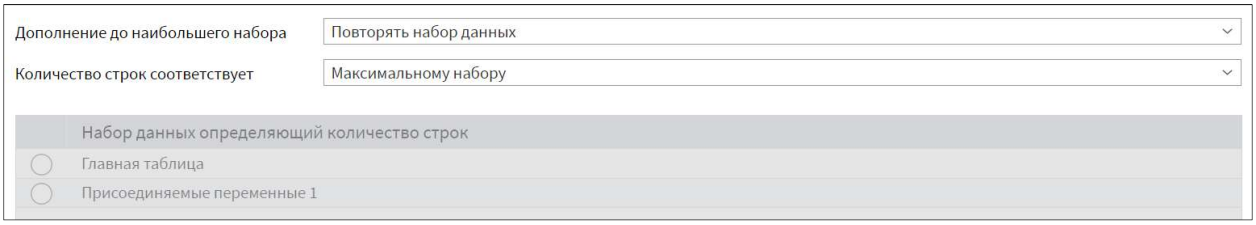


Рисунок 4.53 – Настройка узла Соединение

К нашему наборе данных добавилось еще одно поле – **Общая сумма**, – которое содержит значение одноименной переменной.

Далее рассмотрим компонент **Дополнение данных**. Компонент позволяет обогатить данные путем присоединения к главной таблице полей из других таблиц по принципу левого соединения.

Как и в слиянии, здесь используются связи по ключевым полям, но число присоединяемых таблиц не ограничено.

Рассмотрим пример обогащения набора данных из нескольких справочников. Нам понадобятся наборы **sales_november.lgd**, **goods.lgd**, **suppliers.lgd**, **warehouses.lgd**.

Импортируем перечисленные наборы. Набор **sales_november.lgd** содержит данные о продажах строительных товаров за ноябрь с указанием даты продажи и количества проданного товара. Пусть мы хотим добавить к этим данным информацию о группах товаров, их остатках на складе и поставщиках.

В наборе **goods.lgd** хранится полный список товаров с указанием групп, к которым они относятся.

Набор **suppliers.lgd** содержит список товаров с указанием поставщика.

Последний набор – **warehouses.lgd** – содержит информацию о складе, на котором находится товары, и остатках на 1 ноября 2016 год.

Добавим в область построения узел **Дополнение данных**. По умолчанию он имеет два входных порта. Для того, чтобы добавить дополнительные порты, можно нажать кнопку **Добавить еще один порт** или протянуть к этой кнопке связь от выходного порта другого узла.

Подадим на входы все наши наборы данных, для удобства переименуем порты и перейдем в настройку узла (рисунок 4.54).

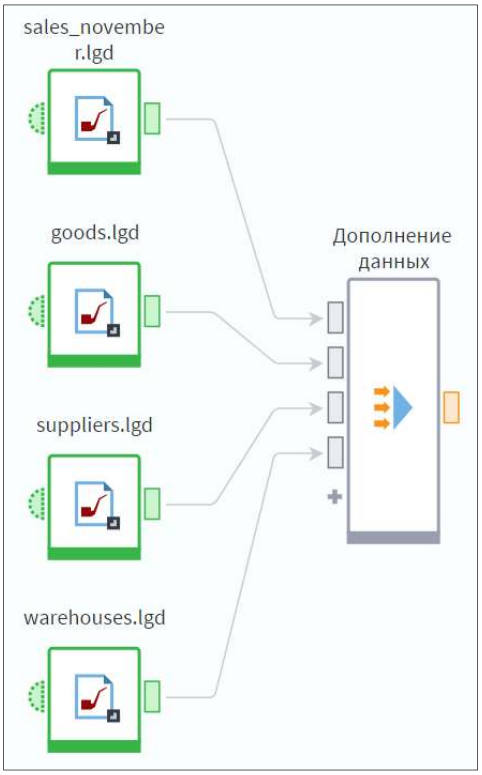


Рисунок 4.54 – Установление связей с узлом Дополнение данных

Основную часть окна занимает **Область настройки ключевых полей**. Здесь необходимо напротив ключевого поля главной таблицы выставить флаг в столбце каждой присоединяемой таблицы. Поставим флаг для таблицы **Товары** напротив поля **Товар** главной таблицы.

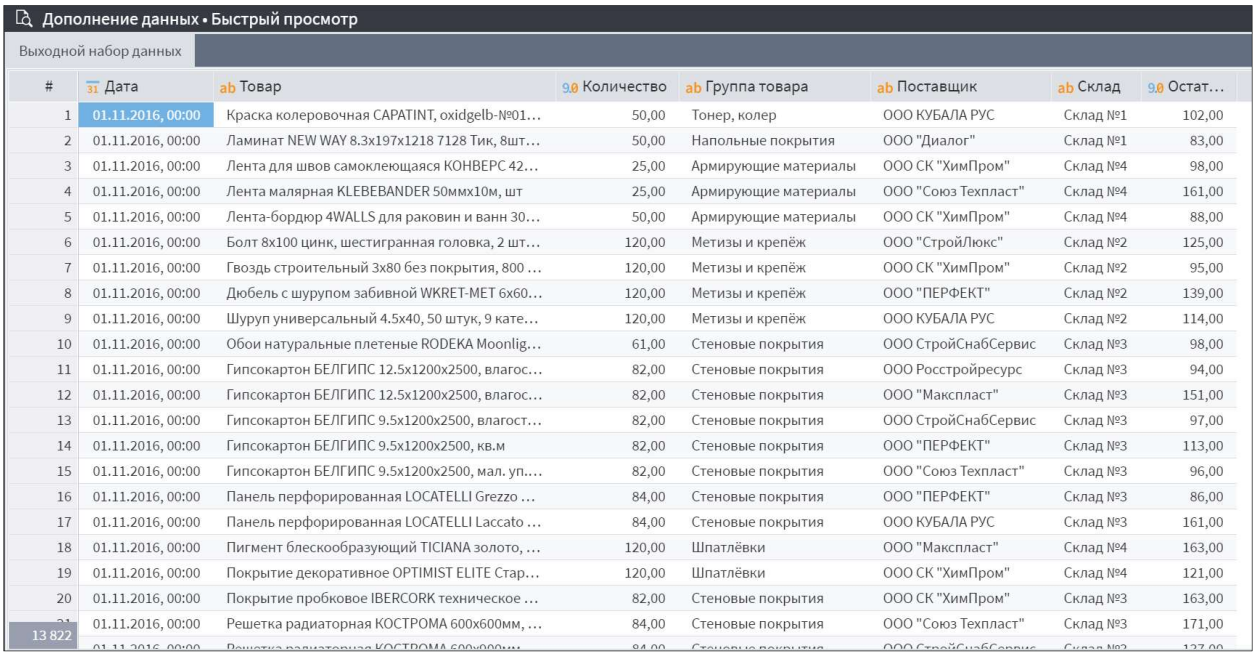
Стал активен список доступных полей набора данных **Товары**. Доступные поля с таким же типом данных. Автоматически установилась связь по полю **Товар**.

Значок фильтрации включен по умолчанию и означает, что в списке доступны только те поля, которые еще не связаны с ключевыми полями главной таблицы. Если выключить фильтрацию, в списке будут отображаться все поля, совместимые по типу данных.

В нижней части окна доступен флаг **Использовать префиксы**. Если его включить, именам и меткам полей, взятых из присоединяемых таблиц, на выходе будут добавлены префиксы, указанные, соответственно, в параметрах **Префикс имени** и **Префикс метки**.

Поставим соответствия для остальных наборов и посмотрим на результат.

На выходе получили все тот же список продаж за ноябрь 2016, к которому добавилась информация из выбранных полей. Таким образом можно с помощью одного узла легко собрать в один набор информацию из разных таблиц (рисунок 4.55).



#	Дата	Товар	Количество	Группа товара	Поставщик	Склад	Остат…
1	01.11.2016, 00:00	Краска колеровочная CAPATINT, oxidgelb-№01…	50,00	Тонер, колер	ООО КУБАЛА РУС	Склад №1	102,00
2	01.11.2016, 00:00	Ламинат NEW WAY 8.3x197x1218 7128 Тик, 8шт…	50,00	Напольные покрытия	ООО "Диалог"	Склад №1	83,00
3	01.11.2016, 00:00	Лента для швов самоклеющаяся КОНВЕРС 42…	25,00	Армирующие материалы	ООО СК "ХимПром"	Склад №4	98,00
4	01.11.2016, 00:00	Лента малярная KLEBEBANDER 50ммх10м, шт	25,00	Армирующие материалы	ООО "Союз Техпласт"	Склад №4	161,00
5	01.11.2016, 00:00	Лента-бордюр 4WALLS для раковин и ванн 30…	50,00	Армирующие материалы	ООО СК "ХимПром"	Склад №4	88,00
6	01.11.2016, 00:00	Болт 8x100 цинк, шестигранная головка, 2 шт…	120,00	Метизы и крепёж	ООО "СтройЛюкс"	Склад №2	125,00
7	01.11.2016, 00:00	Гвоздь строительный 3x80 без покрытия, 800 …	120,00	Метизы и крепёж	ООО СК "ХимПром"	Склад №2	95,00
8	01.11.2016, 00:00	Дюбель с шурупом забивной WKRET-MET 6x60…	120,00	Метизы и крепёж	ООО "ПЕРФЕКТ"	Склад №2	139,00
9	01.11.2016, 00:00	Шуруп универсальный 4.5x40, 50 штук, 9 кате…	120,00	Метизы и крепёж	ООО КУБАЛА РУС	Склад №2	114,00
10	01.11.2016, 00:00	Обои натуральные плетеные RODEKA Moonlig…	61,00	Стеновые покрытия	ООО СтройСнабСервис	Склад №3	98,00
11	01.11.2016, 00:00	Гипсокартон БЕЛГИПС 12.5x1200x2500, влагос…	82,00	Стеновые покрытия	ООО Росстройресурс	Склад №3	94,00
12	01.11.2016, 00:00	Гипсокартон БЕЛГИПС 12.5x1200x2500, влагос…	82,00	Стеновые покрытия	ООО "Макспласт"	Склад №3	151,00
13	01.11.2016, 00:00	Гипсокартон БЕЛГИПС 9.5x1200x2500, влагост…	82,00	Стеновые покрытия	ООО СтройСнабСервис	Склад №3	97,00
14	01.11.2016, 00:00	Гипсокартон БЕЛГИПС 9.5x1200x2500, кв.м	82,00	Стеновые покрытия	ООО "ПЕРФЕКТ"	Склад №3	113,00
15	01.11.2016, 00:00	Гипсокартон БЕЛГИПС 9.5x1200x2500, мал. уп.…	82,00	Стеновые покрытия	ООО "Союз Техпласт"	Склад №3	96,00
16	01.11.2016, 00:00	Панель перфорированная LOCATELLI Grezzo …	84,00	Стеновые покрытия	ООО "ПЕРФЕКТ"	Склад №3	86,00
17	01.11.2016, 00:00	Панель перфорированная LOCATELLI Laccato …	84,00	Стеновые покрытия	ООО КУБАЛА РУС	Склад №3	161,00
18	01.11.2016, 00:00	Пигмент блескообразующий TICIANA золото, …	120,00	Шпатлёвки	ООО "Макспласт"	Склад №4	163,00
19	01.11.2016, 00:00	Покрытие декоративное OPTIMIST ELITE Стар…	120,00	Шпатлёвки	ООО СК "ХимПром"	Склад №4	121,00
20	01.11.2016, 00:00	Покрытие пробковое IBERCORK техническое …	82,00	Стеновые покрытия	ООО СК "ХимПром"	Склад №3	163,00
13 822	01.11.2016, 00:00	Решетка радиаторная КОСТРОМА 600x600мм, …	84,00	Стеновые покрытия	ООО "Союз Техпласт"	Склад №3	171,00

Рисунок 4.55 – Результат Дополнения данных

Рассмотрим компонент **Объединение**. С помощью трех рассмотренных выше компонентов можно дополнять набор данных полями из других наборов.

Компонент **Объединение** позволяет дополнить набор, поданный на порт **Главная таблица**, записями из присоединяемых наборов. Для рассмотрения работы компонента нам понадобятся наборы данных **sales.lgd** и **sales_december.lgd**.

Импортируем указанные наборы. Набор **sales_december.lgd** содержит такие же данные по продажам, что и **sales.lgd**, но за декабрь 2016-го года. Задача состоит в том, чтобы добавить к данным по продажам за несколько месяцев декабрьские продажи.

Создадим узел **Объединение** и подадим на его порты наши наборы, как показано на рисунке 4.56. При необходимости узла можно добавить любое количество портов.

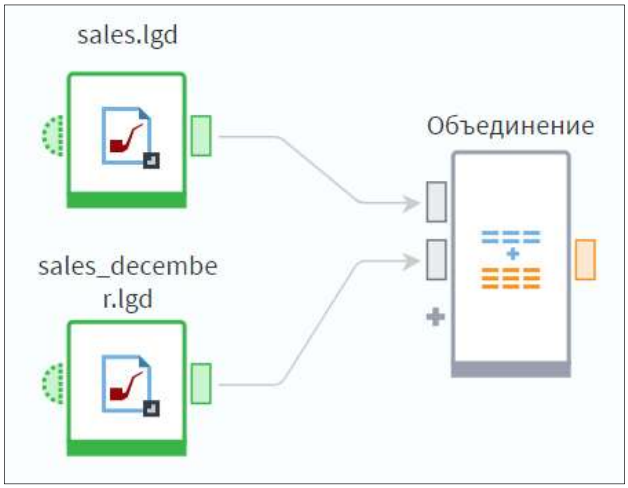


Рисунок 4.56 – Установление связей с узлом Объединение

Если в присоединяемой таблице есть поля, для которых в главной отсутствует (или не поставлено} соответствие, они добавятся в выходной набор в качестве отдельных полей. Для них можно использовать префиксы.

Настройка узла похожа на настройку **Дополнения данных**. Здесь также необходимо проставить флаги напротив полей главной таблицы, которым соответствуют поля присоединяемых. Однако в данном случае мы дополняем основной набор записями из присоединяемого, поэтому соответствия необходимо проставить для всех возможных полей.

В нашем случае все поля присоединяемой таблицы аналогичны полям главной. При проставлении соответствий необходимо помнить, что сопоставить можно только поля с одинаковым типом данных.

На выходе мы получаем набор данных с такими же полями, как и в главной таблице, но с большим количеством строк: 111 150.

3.5. Квантование и скользящее окно

В основе **квантования** лежит процедура состоящая из двух шагов.

Диапазон значений, в пределах которого изменяется некоторая числовая величина (признак, показатель и так далее), разбивается на некоторое количество интервалов, каждому из которых присваивается определенный номер.

Эти интервалы называются **интервалами квантования**, а присвоенные им номера – **уровнями квантования**.

Каждое значение заменяется номером или меткой интервала квантования, в который попало данное значение.

Представим процесс квантования графически (рисунок 4.57). Пусть наблюдаемый на рисунке ряд значений представляет собой суммы выданных

кредитов. При этом минимальная сумма составляет **10 000 руб.**, а максимальная – **65 000 руб.**

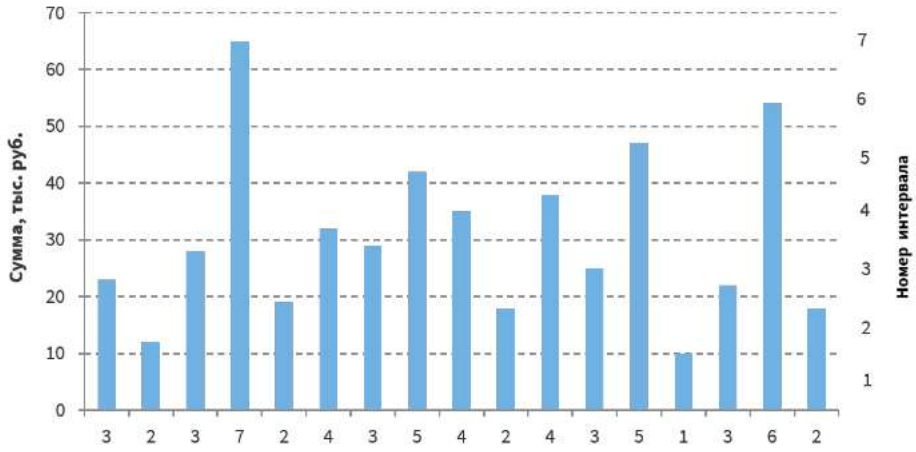


Рисунок 4.57 – Пример квантования

Если диапазон значений ряда от **0** до **70 000 руб.**, то его можно разбить на **7** равных интервалов, взятых через **10 000**, по которым и будет проводиться квантование.

Для этого каждому интервалу будет присвоен порядковый номер, после чего все наблюдаемые значения будут заменены номерами интервалов квантования, в которые они попали.

То есть вместо значения **23**, которое принадлежит третьему интервалу квантования, в результирующем наборе данных будет **3**, вместо значения **35** будет **4** и так далее.

Исходное значение	23	12	28	65	19	32	29	42	35	18	38	25	47	10	22	54	18
Квантованное значение	3	2	3	7	2	4	3	5	4	2	4	3	5	1	3	6	2

Рисунок 4.58 – Результат квантования

Итоговый результат преобразования представлен на рисунке 4.58.

При квантовании необходимо определить, какую из границ интервала следует включить в этот интервал. Поскольку нижняя граница диапазона всегда принадлежит нижнему интервалу, то и для других интервалов можно условиться о включении нижней границы.

Единственным исключением является самый верхний интервал, который включает в себя как верхнюю, так и нижнюю границы.

Рассмотрим подробнее где и как обычно используется квантование.

1. Квантование широко используется во всех областях, где возникает необходимость в обработке, передаче и хранения данных.

2. Квантование – неотъемлемая часть процесса преобразования **аналоговых** (то есть непрерывных по времени и амплитуде) сигналов в **цифровые** (то есть дискретные по времени и квантованные по амплитуде).

3. Квантование позволяет представлять и хранить данные в более компактном и защищенном от искажений виде.

4. Процесс дискретизации заключается в представлении непрерывной функции в виде набора отдельных значений, взятых в определенные моменты времени – **отсчеты**.

5. В результате квантования значения отсчетов преобразуется в номера интервалов квантования, в которые эти значения попали.

Принцип преобразования аналоговых данных в цифровые представлен на рисунке 4.59.

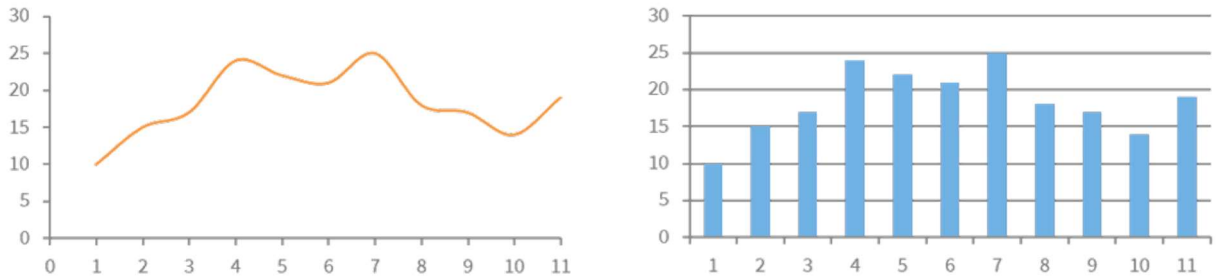


Рисунок 4.59 – Пример принципа преобразования аналоговых данных в цифровые данные

В бизнес-аналитике квантование способствует достижению следующих целей:

- изменяется вид данных (из непрерывных они могут быть преобразованы в дискретные);
- сокращается размерность данных (уменьшается число разнообразных значений признака).

Например, если для анализа клиентов банка, получающих кредит, интерес представляют не отдельные клиенты и суммы кредитов, а группы, объединяющие клиентов по интервалам сумм, то в результате квантования можно получить более удобный для анализа ряд данных.

Уменьшение разнообразия значений признаков в некоторых случаях позволяет сделать работу моделей более эффективной. Действительно, если с точки зрения анализа нет разницы между суммами кредита **15** и **17 тыс.**, то нет смысла рассматривать эти величины отдельно.

В некоторых случаях представляет интерес использование в качестве результатов квантования не номеров интервалов, а других значений, связанных с них:

- **Нижняя граница интервалов**. Вместо значения, которое попало в интервал, устанавливается значение нижней границы.
- **Верхняя граница интервалов**. Вместо значения, которое попало в интервал, устанавливается значение верхней границы.
- **Среднее арифметическое интервала**. Вместо значения, которое попало в интервал, устанавливается его срединное значение. Границы и середину интервала удобно применять в тех случаях, когда квантованный ряд значений должен сохранять количественное выражение исходных данных. Однако в этом случае результирующее поле по-прежнему останется непрерывным и не сможет быть использовано в качестве выходного в классификационной модели. Преимуществом использования данного метода будет сокращение разнообразия значений признака.
- **Метка интервала**. Пользователь может задать произвольное значение, которое обозначат интервал, например, наименование категории, к которой будет относится объект классификации.

Использование меток интервалов дает возможность сделать результаты квантования более наглядными и сразу определить метки классов, если целью квантования является разбиение признака по категориям. Так, если цель квантования поля **Сумма кредита** – разделить всех клиентов на категории в зависимости от взятой ими суммы, то можно использовать соответствующие метки, как это демонстрирует таблица (рисунок 4.59).

Срок кредита	Возраст	Пол	Образование	Сумма кредита	Категория клиента
6	37	Жен	Специальное	7 000	Категория 1
6	38	Муж	Среднее	7 500	Категория 1
12	60	Муж	Высшее	14 500	Категория 2
6	28	Муж	Специальное	15 000	Категория 2
12	59	Жен	Специальное	32 000	Категория 4
6	25	Жен	Специальное	11 500	Категория 1
6	57	Муж	Специальное	5 000	Категория 1
30	29	Муж	Высшее	61 500	Категория 7
12	37	Муж	Специальное	13 500	Категория 2
18	36	Муж	Специальное	25 000	Категория 3
24	68	Муж	Высшее	25 500	Категория 3
6	20	Жен	Высшее	9 500	Категория 1

Рисунок 4.59 – Пример результата квантования

На практике этот набор данных можно использовать как обучающую выборку для построения классификационной модели, где в качестве целевого поля будет использоваться **Категория клиента**.

В квантовании важно правильно выбрать **число интервалов**.

Так как в результате квантования осуществляется переход от точных данных к некоторой интервальной оценке, неизбежна потеря информации. Фактически ряд значений, полученных в результате квантования, просто выражает отношения между исходными значениями признака.

То, что два значения расположены в двух соседних интервалах квантования, не позволяет точно определить, насколько одно из них больше или меньше другого. Можно сказать только, что они не различаются больше, чем на две **ширины интервала**.

Ширина интервала представляет собой разницу между верхней и нижней границами интервала. Следовательно, **чем больше интервалов используется при квантовании, тем точнее представление исходных значений данных**. При уменьшении ширины интервала в пределе мы получим исходный набор значений. Увеличение интервала, напротив, огрубляет описание данных и в пределе дает один интервал для всего диапазона значений, которые меняются на метку интервала (например, 0).

Иными словами, меняя число интервалов квантования, можно перейти от точного воспроизведения исходных значений данных к полной потере информации об изменчивости значений признака.

На практике выбрать количество интервалов квантования можно исходя из следующих соображений:

1. Если квантование выполняется для преобразования непрерывных данных в дискретные, то число интервалов будет определяться числом уникальных значений (меток, категорий), которые используются при решении задачи анализа.

2. Необходимо учитывать требуемую точность описания данных. Например, может быть поставлено условие, что количество интервалов квантования должно быть таким, чтобы ширина интервала не превышала **10%** от полного диапазона изменения исходных значений.

3. Иногда может потребоваться проведение экспериментов, чтобы определить лучшие параметры квантования с сточки зрения решения конкретной задачи анализа.

Кроме выбора числа интервалов, при выполнении операции квантования требуется выбрать ее метод.

Выбор метода квантования зависит от характера данных. Различают два основных метода квантования:

- **Равномерное (однородное) квантование.** При равномерном квантовании диапазон изменения значений признака разделяется на интервалы одинаковой ширины. Количество интервалов в таком случае может задаваться явно, тогда ширина интервала рассчитывается как отношение разницы между верхней и нижней границами диапазона к заданному количеству, либо неявно, с помощью задания ширины интервала.

Во втором случае количество рассчитывается как отношение разницы между верхней и нижней границами диапазона к заданной ширине.

На рисунке 4.60 представлен пример равномерного квантования, где диапазон значений поделен на 7 интервалов, ширина каждого из которых составляет 10 тыс. руб.

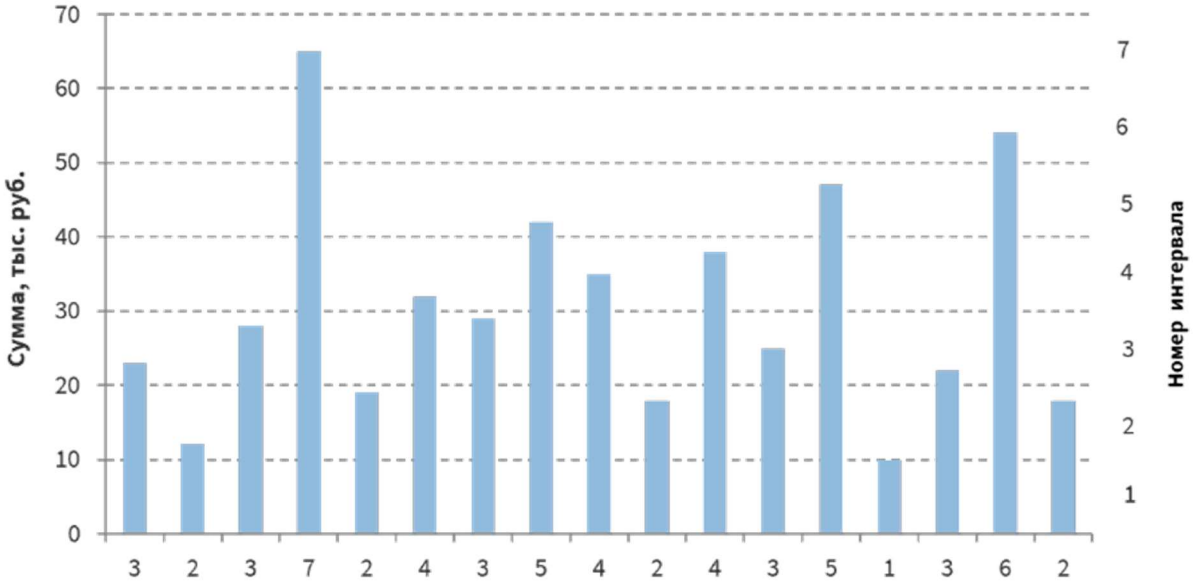


Рисунок 4.60 – Пример равномерного квантования

- **Неравномерное (неоднородное) квантование**. При неравномерном квантовании ширина интервалов может быть различной. Здесь также есть несколько способов распределения значений признака. Например, при **плиточном** квантовании ширина интервалов выбирается таким образом, чтобы в каждый из них попало примерно одинаковое количество значений.

Еще один способ неравномерного квантования – **коэффициенты СКО** (среднеквадратического отклонения).

При его использовании формируются интервалы с шириной, кратной среднеквадратическому отклонению значений признака (рисунок 4.61).

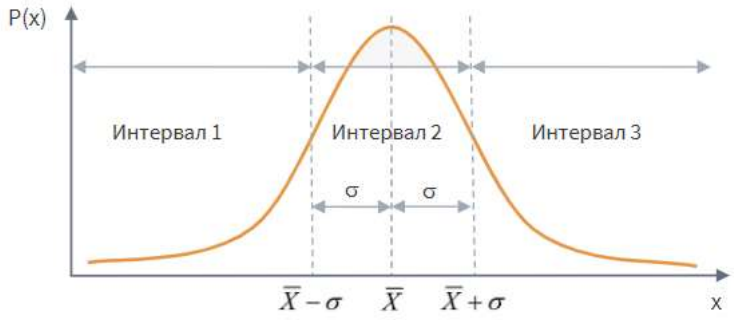


Рисунок 4.61 – График среднеквадратичного отклонения

Границы интервалов будут рассчитываться на основе вычисленных математического ожидания $\overline{X}$ и среднеквадратического отклонения σ, например, если используется одно среднеквадратическое отклонение, то формируется три интервала:

$$x < \bar{X} - \sigma, \bar{X} - \sigma \leq x \leq \bar{X} + \sigma, x > \bar{X} + \sigma \quad (4.1)$$

Среднеквадратичное отклонение – показатель рассеивания значений случайной величины относительно ее математического ожидания. Если количество всех значений признаков – *n*.

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - X)^2} \quad (4.2)$$

Математическое ожидание дискретной случайной величины – это сумма парных произведений всех возможных ее значений на соответствующие вероятности:

$$\bar{X} = x_1 p_1 + x_2 p_2 + \cdots + x_n p_n = \sum_{i=1}^{n} x_i p_i \quad (4.3)$$

Равномерное квантование используется, если данные равномерно распределены по всему диапазону их изменения, то есть в результате квантования не будет интервалов, в которых значения почти отсутствуют или заполнены очень плотно.

В противном случае лучшие результаты даст неравномерное квантование. Рассмотрим это на примере.

Пусть дан набор значений {2, 4, 3, 1, 4, 22, 24, 23, 21, 24} диапазон изменения его значений составит **23**.

Разделим диапазон значений на 5 равных интервалов квантования, каждый из которых будет содержать 5 значений.

Интервал с номером 0 будет содержать значения от 0 до 4, с номером 1 – от5 до9, с номером 2 – от 10 до 14 и так далее.

На рисунке 4.62 видно, что значения в указанном диапазоне распределены неравномерно: в нулевом интервале квантования расположено 5 значений, а остальные значения – в диапазоне 20-24, который соответствует интервалу квантования с номером 4.

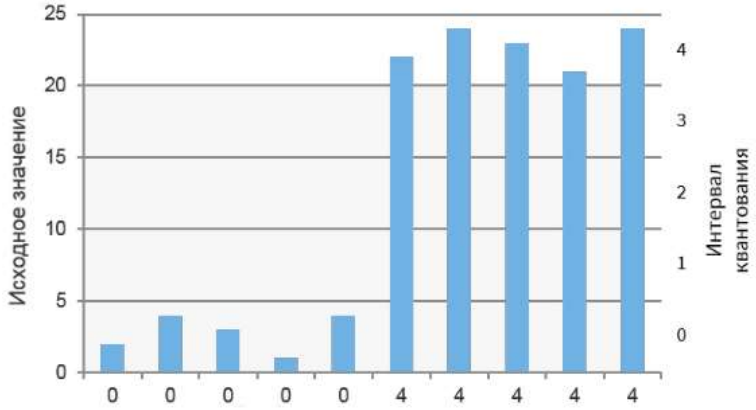


Рисунок 4.62 – Пример неравномерного квантования

При этом в диапазон 5-20, которому соответствуют интервалы квантования с номерами 1, 2 и 3, не попало ни одного значения.

Следовательно, квантованные значения также распределяются неравномерно: интервалы 0 и 4 будут заполнены очень плотно, в то время как интервалы 1, 2 и 3 окажутся пустыми.

Если квантование проводится для преобразования непрерывных данных в набор категорий, это приведет к тому, что все объекты выборки окажутся отнесенными всего к двум категориям – 0 и 4 (рисунок 4. 63).

Исходное значение	Квантованное значение
2	0
4	0
3	0
1	0
4	0
22	4
24	4
23	4
21	4
24	4

Рисунок 4.63 – Пример квантования

Преодолеть данную проблему позволяет неравномерное квантование, когда используются интервалы разной ширины так, чтобы в каждый из них попало примерно одинаковое количество значений.

Мы уже знаем, что такое квантование называется **плиточным**.

Если его применить к приведенному примеру приведенному на рисунке 4.64, то получим следующие результаты.

Рисунок 4.64 – Пример квантования

Как видно из правой таблицы, заполнение интервалов квантования получилось более равномерным, чем при обычном равноинтервальном квантовании.

Следующая тема – это операции по **преобразованию упорядоченных данных**.

Многие аналитические задачи, например, анализ продаж, связаны с обработкой данных, которые зависят от времени. Такие данные называют **упорядоченными**, или **временными рядами**.

В процессе обработки временных рядов требуется специальная подготовка данных, чтобы оптимизировать их представление для всех возможных интервалов даты и времени. Это необходимо для решения определенных аналитических задач, в частности:

- прогнозирование;
- классификация состояний объектов;
- выявление закономерностей, объясняющих динамику бизнес-процессов.

Временной ряд состоит из последовательности наблюдений за состоянием параметров (признаков) исследуемых объектов или процессов. Если наблюдения содержат один признак, то ряд является одномерным, а если два или более – многомерным.

Поскольку значения временного ряда определены только в фиксированные моменты времени, так называемые отсчеты, последовательность его значений может быть представлена в следующем виде:

$$X = \{x_1, x_2, \dots x_n\} \qquad (4.4)$$

где x_n – последнее значение рассматриваемой временной последовательности.

При этом отсчеты времени полагаются равноотстоящими друг от друга (рисунок 4.65).

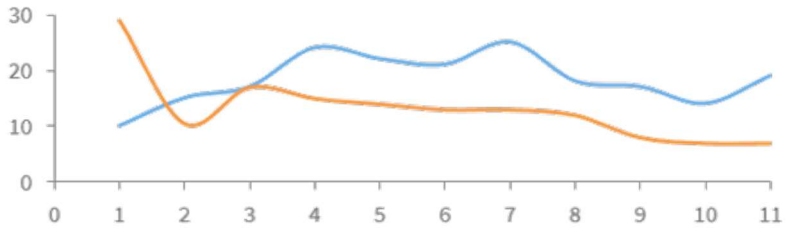


Рисунок 4.65 – Пример преобразования упорядоченных данных

Целью преобразований временных рядов является не изменение их содержания, а представление информации таким образом, чтобы обеспечивалась максимальная эффективность решения определенной задачи анализа.

Можно выделить два основных типа преобразования, которые наиболее часто используются при подготовке временных рядов к анализу:

- **Преобразование даты и времени**. Преобразование даты и времени заключается в приведении даты и времени к виду, наиболее удобному для визуального анализа и обработки временного ряда. При этом результаты преобразования даты не обязательно являются значениями типа **Дата/Время** и могут обрабатываться как обычные числа и строки.
- **Скользящее окно**. Скользящее окно применяется при решении задач прогнозирования и классификации состояний бизнес-объектов, чтобы преобразовывать последовательность значений ряда в таблицу, которую можно использовать для построения моделей или какой-либо другой обработки. С первым типом мы уже познакомились в первой лекции, поэтому далее подробно рассмотрим только **скользящее окно**.

Скользящее окно широко применяется при обработке временных рядов, например, чтобы построить модель прогноза временного ряда, или когда требуется при обработке набора данных сравнивать соседние значения – лучше, чтобы они располагались в столбцах, а не в строках.

Целью прогнозирования значений временного ряда является предсказание значения x_{n+1}, на основе предыдущих значений признака. Решение задачи прогнозирования возможно только в том случае, если значения временного ряда связаны между собой (рисунок 4.66).

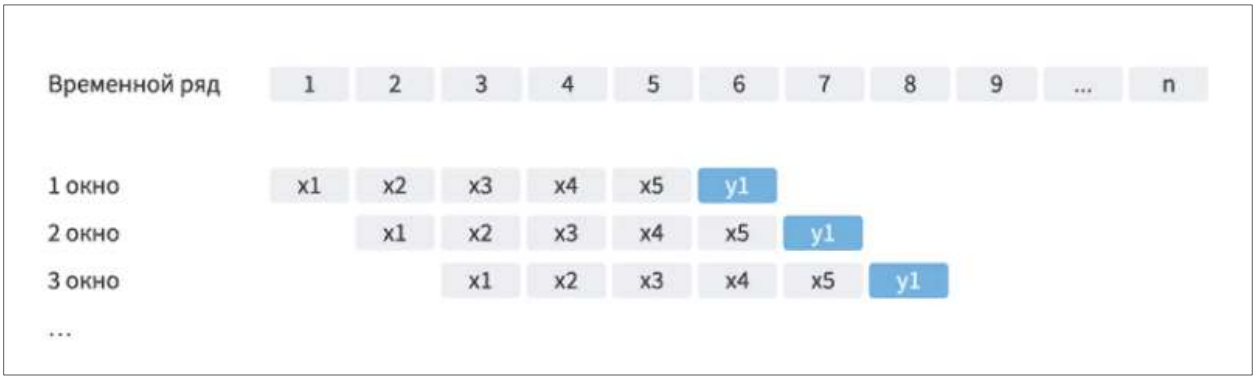


Рисунок 4.66 – Пример скользящего окна

Например, пусть при регистрации заявок, поступивших от клиентов, среди прочего фиксируются дата подачи заявки и адрес клиента. Из базы данных системы регистрации заявок можно извлечь временной ряд с последовательностью номеров квартир, указанных в адресах.

Очевидно, что пытаться предсказать номер квартиры следующего клиента на основе знания номеров квартир клиентов, чьи заявки были зарегистрированы ранее, бессмысленно.

Скользящее окно оперирует следующей терминологией анализа и прогнозирования временных рядов:

- **Интервал прогноза.** Временной интервал, на котором будет осуществляться прогнозирование (день, неделя, месяц, квартал, год).
- **Горизонт прогноза.** На какое количество интервалов (дней, недель и др.) вперед мы ходим получить прогноз.
- **Глубина истории.** Количество значений интервалов прогноза в прошлом, которое мы будем использовать для предсказания значений интервалов в будущем.

Пусть имеется ряд данных, содержащий 11 наблюдений:

$$X = \{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}. \quad (4.5)$$

Если глубину истории задать равной **5**, то с помощью скользящего окна можно преобразовать исходный ряд данных в табличную форму, состоящую из 6 записей, которая может быть использована для дальнейших вычислений (рисунок 4.67).

№	x_{n-5}	x_{n-4}	x_{n-3}	x_{n-2}	x_{n-1}	x_n	x_{n+1}	x_{n+2}
1	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7
2	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
3	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
4	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
5	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
6	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}

Рисунок 4.67 – Ряд данных в табличной форме

Таким образом, основная задача скользящего окна – преобразование ряда данных в таблицу, где каждая запись представляет собой наблюдение, сформированное из некоторого интервала ряда.

Рассмотрим еще пример. Пусть имеется ряд наблюдений, который отражает количество клиентов, обслуженных за месяц (таблица 1 на рисунке 4.68).

Дата начала месяца	Число клиентов
01.01.2017	1 540
01.02.2017	960
01.03.2017	1 150
01.04.2014	1 230
01.05.2017	1 056
01.06.2017	995

Дата начала месяца	x_{n-2}	x_{n-1}	x_n	x_{n+1}
				1 540
01.01.2017			1 540	960
01.02.2017		1 540	960	1 150
01.03.2017	1 540	960	1 150	1 230
01.04.2014	960	1 150	1 230	1 056
01.05.2017	1 150	1 230	1 056	995
01.06.2017	1 230	1 056	995	
	1 056	995		
	995			

Дата начала месяца	x_{n-2}	x_{n-1}	x_n	x_{n+1}
01.03.2017	1 540	960	1 150	1 230
01.04.2014	960	1 150	1 230	1 056
01.05.2017	1 150	1 230	1 056	995

Рисунок 4.68 – Ряд данных отражающий количество клиентов, обслуженных за месяц

Задача состоит в том, чтобы построить модель прогноза числа клиентов на будущий месяц. При этом глубина погружения задается равной 2, горизонт прогнозирования – 1, то есть на основе двух предыдущих месяцев прогнозируется следующий.

Выборка, полученная в результате обработки данных скользящим окном, будет содержать в начале и в конце неполные записи, количество которых будет равно глубине погружения (таблица 2 на рисунке 4.68).

Неполные записи можно исключить из рассмотрения. Если это сделать, то результирующая выборка будет иметь вид представленный на рисунке 4.68 – таблица 3.

3.6. Лабораторная работа №9 «Квантование»

Начнем с компонента **Квантование**. Он позволяет разбить диапазон значений выбранного поля на конечное число интервалов. Чаще всего он используется для преобразования непрерывных данных в дискретные.

Для рассмотрения работы компонента нам понадобятся наборы данных **regions_population.lgd** и **bounds.lgd**.

Создадим новый пакет и импортируем в сценарий набор данных **regions_population.lgd**. В наборе содержится информация о численности населения по регионам.

Добавим в область построения узел **Квантование** из раздела **Предобработка**. Интервалы квантования можно задать двумя способами: настроить непосредственно в узле, либо подать на его вход дополнительную таблицу специальной структуры. Для начала рассмотрим настройку интервалов в узле.

Основные настройки проводятся в области настройки параметров квантования. В ней отображаются только поля с теми типами данных, к которым применима процедура квантования: **целый**, **вещественный** и **дата/время**. Кроме того, квантуемое поле должно содержать хотя бы 2 уникальных значения.

Для начала необходимо выбрать для нужного поля метод квантования. Раскроем список доступных методов.

Всего метода четыре. Выберем самый первый – **Ширина**. Он позволяет задать ширину интервалов – количество значений, которое содержится в каждом. Количество интервалов при этом рассчитывается автоматически как отношение разности верхней и нижней границ всего диапазона значений поля к заданной ширине.

Раскроем область настроек для выбранного метода (рисунок 4.69). Ширина указывается в одноименном параметре и по умолчанию не задана. Кроме того мы можем задать нижнюю и верхнюю границы диапазона значений поля, к которому будем применено квантование, а также снять/установить флаг **Округлять границы**.

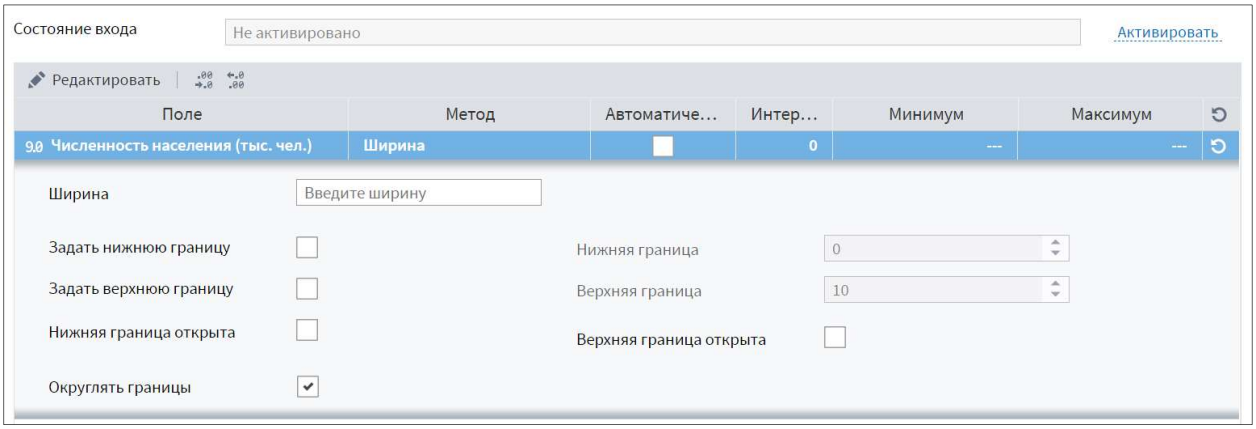


Рисунок 4.69 – Настройки компонента квантование. Метод Ширина

Соответствующие флаги позволяют сделать нижнюю и/или верхнюю границы всего диапазона значений открытыми. Тогда при подаче нового набора данных, в котором есть значения меньше нижней или больше верхней границы, они попадут, соответственно, в первый или последний интервал. В противном случае эти значения не попадут ни в один из интервалов. Для пересчета границ при подаче нового набора данных понадобится переобучить узел.

Возможна автоматическая настройка параметров квантования для выбранного метода. Установим флаг автоматической настройки. Область настройки параметров стала неактивной.

В параметрах «Интервалов», «Минимум» и «Максимум» указывается, на сколько интервалов будут разбиты значения поля, а также максимальное и минимальное из этих значений. Для того, чтобы увидеть значения параметров необходимо нажать кнопку **Рассчитать интервалы**.

Расчет интервалов возможен только в том случае, когда вход узла активирован. Так как у нас в поле **Состояние входа** стоит значение **Не активировано**, на предложено активировать входные порты. Нажмем **Да**.

При автоматическом расчете мы получили **7** интервалов, ширина каждого из которых равна **1 435,5**. В нижней части окна расположена область отображения результатов квантования, где можно увидеть номер интервала, его границы, тип границ, метку и объем значений, попавших в интервал.

Теперь выберем метод **Количество**. В этом случае мы указываем, на сколько интервалов хотим разделить значения, а ширина этих интервалов рассчитывается, как отношение разницы верхней и нижней границ к заданному количеству.

Если отключить флаг **Автоматически**, для настройки станут доступны параметры, аналогичные параметрам предыдущего метода, только вместо ширины будет задаваться количество интервалов. По умолчанию оно равно **5**.

При изменении настроек интервалы автоматически не пересчитываются, для пересчета нужно снова нажать кнопку **Рассчитать интервалы**. Не будем менять настройки и нажмем ее.

Интервалы в области отображения были пересчитаны в соответствии с новыми настройками.

Еще один метод – **Коэффициенты СКО**. В его настройках необходимо задать количество среднеквадратических отклонений от среднего, исходя из этого будет рассчитано количество интервалов:

- **±1 СКО** – три интервала;
- **±2 СКО** – пять интервалов;
- **±3 СКО** – семь интервалов.

При использовании этого метода в случае нормального распределения признака 68% всех наблюдений попадают в интервал с границами, равными ±1 среднеквадратичное отклонение от математического ожидания, 95% – в интервал ±2 СКО и 99% – в интервал ±3 СКО.

Мы могли убедиться, что в рассмотренных методах значения по интервалам распределяются неравномерно. В частности, при текущих настройках почти все они сосредоточены в среднем. Еще один метод – **Плитка** – позволяет разделить значения таким образом, что каждый интервал имеет примерно одинаковый объем.

В области настройки метода **Плитка** также задается количество интервалов, а равномерное распределение значений по ним происходит автоматически.

При установке флага **Из сумм значений** равномерное распределение будет проводится исходя из сумм значений, а не исходя из количества.

В поле **Совпадающие наблюдения** можно задать способ обработки совпадающих наблюдений, когда есть значения, совпадающие с выбранной границей:

- **Добавлять в следующий** – все совпадающие с границей значения будут перенесены в следующий интервал;
- **Сохранять в текущем** – значения сохранятся в текущем интервале, что при большом количестве совпадений может привести к созданию меньшего количества интервалов;
- **Назначать случайно** – граница и совпадающие с ней наблюдения включаются либо в текущий, либо в следующий интервал случайным образом, но по возможности в равных количествах;
- **Оставить как есть** – совпадающие значения будут распределены относительно выбранной границы, то есть могут попасть как в текущий, таки в следующий интервал, их количество в интервалах не регулируется;

- **Одинаковые плитки** – после разбиения границы корректируются таким образом, чтобы интервалы были примерно равны по объему.

Оставим настройки по умолчанию и снова нажмем **Рассчитать**. С помощью поля Объем области отображения результатов мы можем убедиться, что теперь значения распределены по интервалам равномерно.

В области отображения можно перенастроить границы интервалов, а также изменить метки. Рассмотрим эти настройки подробнее. Для изменения границ любого интервала вручную достаточно щелкнуть левой кнопкой мыши в соответствующей ячейке таблицы. Значение станет доступным для редактирования.

Кнопкой **Инвертировать тип** можно одновременно изменить тип включения границ на противоположный для всех интервалов: если в интервал изначально включается нижняя граница, после нажатия кнопки будет включаться верхняя и наоборот.

Кнопка **Рассчитать гистограмму** позволяет пересчитать значение в поле **Объем** после изменения границ.

В поле Тип можно изменить способ включения границ в интервал. Обычно во все интервалы включается нижняя граница, а в последний – еще и верхняя.

Метки интервалов тоже возможно изменить в ручную, щелкнув мышкой в соответствующей ячейке. Есть возможность изменить метки сразу для всех интервалов с помощью шаблона. Шаблон можно выбрать из списка или написать самостоятельно, используя специальные обозначения – **ключи**. В нашем случае в метке указан диапазон значений интервала.

Рассмотрим основные ключи, которые можно использовать в шаблоне:

- **%MIN, %MAX, %AVE**. В метку будет записано соответственно минимальное, максимальное или среднее арифметическое значение интервала. Для них возможно указать дополнительные параметры форматирования, например, **%MIN[format][round]**, где:
    - **[format]** – позволяет задать формат отображения числа, в частности, в параметре можно указать количество отображаемых разрядов, например, **[5]** – отображать **5** разрядов;
    - **[round]** – позволяет задать значение, до которого будет округлено значение интервала. Задается в квадратных скобках с буквой **r**, например, **[r5]** означает округлять до **5**.
- **%N**. Номер интервала.
- **%FN, %FD**. Имя и метка квантуемого поля соотвественно. Для них можно указать дополнительный параметр форматирования:

- **[sub]** – позволяет задать количество или диапазон символов строки для отображения, например, **[3]** – будут отображаться первые **3** символа имени/метки поля, **[2-7]** – будут отображены символы со **2**-го по **7**-ой.

- **%OP, %CP**. Открывающая и закрывающая скобки соответственно, используется для указания границ интервала: если граница включена в интервал, будет указана квадратная скобка, в противном случае – круглая.

Список ключей и примеры шаблонов можно также посмотреть, нажав кнопку **Образец**. В случае изменения шаблона для просмотра результата необходимо нажать кнопку **Применить шаблон метки**. Оставим настройки без изменений и перейдем к просмотру результатов.

Узел имеет два выходных порта: **Выходной набор данных** и **Диапазоны для квантования**. Рассмотрим данные с первого порта.

К полям исходного набора добавились поля, которые содержат информацию о полученных интервалах для каждого значения квантуемого поля. Метка каждого из новых полей выходного набора по умолчанию составляется по шаблону: <Метка квантуемого поля> <Собственная метка поля>.

Собственная метка поля поясняет, какая информация в нем содержится. Рассмотрим собственные метки выходных полей:

- **Идентификатор интервала**. Содержит имя квантуемого поля.
- **Номер интервала**. Порядковый номер интервала, в который попало значение признака (начинается с 0).
- **Метка**. Заданная шаблоном метка интервала.
- **Нижняя/Верхняя граница**. Значение нижней/верхней границы интервала.
- **Нижний/Верхний предел интервала включительно.** Входит ли в интервал нижняя/верхняя граница: **true** – входит; **false** – не входит.
- **Нижняя/Верхняя граница диапазонов открыта.** Задается тип границы интервала: **true** – открыта; **false** – закрыта.
- **Значение вне диапазонов.** Информирует, входит ли значение признака в весь диапазон квантования. Может содержать одно их трех значений:
  - **-1** – значение признака меньше наименьшей границы диапазона;
  - **0** – значение входит в один из интервалов;
  - **1** – оно больше верхней границы диапазона.

На втором выходе содержится информация о выходных диапазонах квантования – список интервалов и их характеристики.

Рассмотрим поля таблицы, которые отсутствуют в выходном наборе данных:

- **Тип данных границ диапазонов**. Указывается числовой код, обозначающий, к какому типу данных относится значение границы. Актуально при использовании внешних границ. Используется следующие коды:
  - **0** – неопределенный;
  - **1** – логический;
  - **2** – дата/время;
  - **3** – действительный;
  - **4** – целый;
  - **5** – строковый;
  - **6** – переменный.
- **Квота нижней/верхней границы**. Количество попавших в интервал значений квантуемого поля, которые равны значению нижней/верхней границы. Если квота не задействована, в поле будет знаний **null**.
- **Квота интервала**. Поле используется, если верхняя и нижняя границы равны, и нет необходимости считать их квоты по отдельности. В остальных случаях значение в поле **null**.
- **Отклонение от нижней/верхней границы из диапазона.** При использовании внешних интервалов квантования в этом поле можно задать допустимое отклонение от граничного значения, чтобы два близких значения, одно из которых оказалось больше граничного, могли попасть в один интервал. Если данный параметр не задан, в поле записываются граничные значения. Перейдем к рассмотрению использования внешних границ диапазонов.

Для задания внешних интервалов необходимо для начала создать набор данных информацией об этих интервалах. Пусть у нас есть фиксированные интервалы квантования и метки для них, при этом нижняя граница интервала должна входить в каждый из них, а верхняя граница всего диапазона должна быть открыта. Импортируем файл **bounds.lgd**, где содержится эта информация.

Обратите внимание, что при использовании внешних интервалов информация о том, входит ли граница в интервал, задается путем указания значения 1 или 0 в поле квоты соответствующей границы.

Создадим еще один узел квантования и перезимуем его, как показано на рисунке 4.70. Для подачи на вход таблицы с внешними границами необходимо добавить у узла еще один порт.

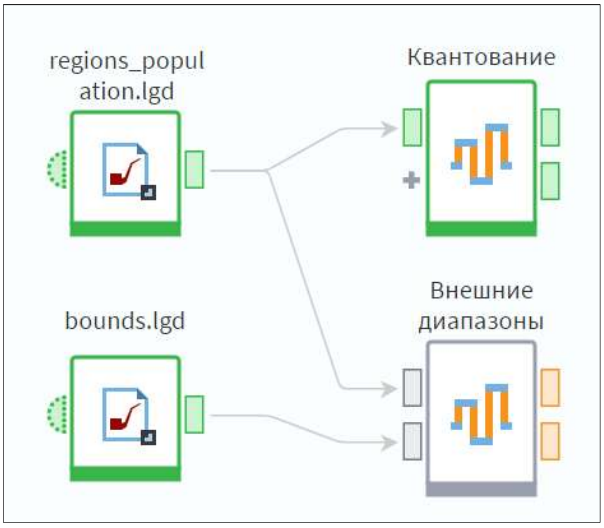


Рисунок 4.70 – Установление связей между узлами

Данный порт называется Внешние диапазоны квантования и предназначен специально для подачи на него данных об интервалов. Подадим данные на входы узла, как показано на рисунке 4.70, и перейдем в его настройку.

Первый шаг настройки изменился. Теперь это **Настройка внешних** интервалов. Справа находится область **Назначения полей**, где содержится полный список полей, которые можно подать на вход. Он соответствует списку полей таблицы на выходе узла **Диапазоны для квантования**.

Обратите внимание, что для некоторых полей типы данных заданы жестко. Кроме того, для поля **Тип данных границ диапазонов** есть список допустимых значений, который мы рассмотрели ранее. При формировании набора с внешними границами необходимо соблюдать эти ограничения (рисунок 4.71).

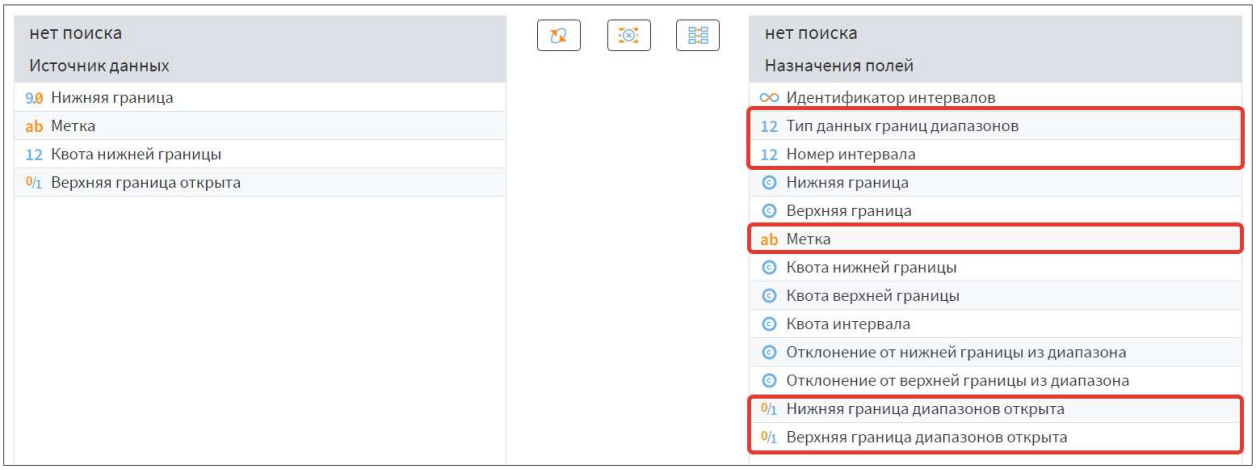


Рисунок 4.71 – Настройка внешних интервалов

Поле **Идентификатор интервалов** может быть любого типа.

Поля **Квота нижней/верхней границы** – только целого или вещественного типа.

Остальные могут быть одного из трех типов данных: **Целый**, **Вещественный** или **Дата/Время**.

Теперь рассмотрим обязательные и необязательные поля, а также их значения. Нижние границы на вход. подавать обязательно. Если мы квантуем только одно поле, этого будет достаточно для обработки. Если полей несколько, необходимо также подать **Идентификатор интервалов**, с помощью которого будет определено, какие границы к какому полю относятся.

Остальные поля не являются обязательными, для них значения будут проставлены следующим образом:

- **Тип данных границ диапазонов** – определится автоматически в зависимости от значений границ;
- **Номер интервала** – проставится автоматически, начиная с 0;
- **Верхняя граница** – значения определятся на основе значений нижних границ;
- **Метка** – останется пустой, ее можно настроить на следующем шаге с помощью шаблона;
- **Квота верхней границы** – будет содержать **0**, при использовании внешних интервалов это означает, что граница не входит в интервал;
- **Квота нижней границы** – значение **1**, граница войдет в интервал;
- **Квота интервала** – значение **null**;
- **Отклонение от нижней/верхней границы из диапазона** – будут содержать значения границ;
- **Нижняя граница диапазонов открыта** – значение **false**;

- **Верхняя граница диапазонов открыта** – значение **true**.

Слева находится область **Источник данных**, где расположены поля набора, поданного на вход для внешних диапазонов. С помощью перетаскивания свяжем входные поля с соответствующими полями из правого списка, как показано на рисунке 4.72.

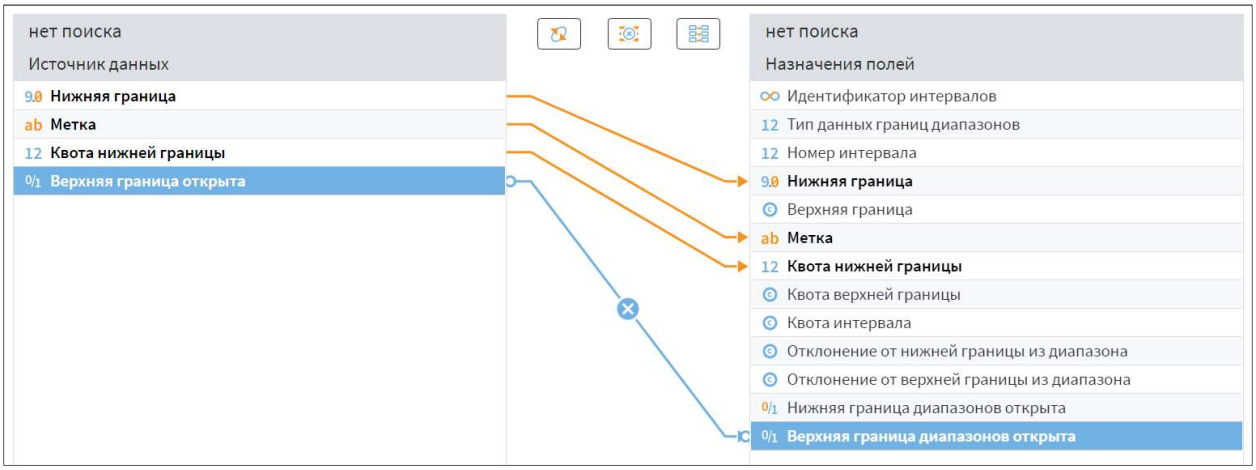


Рисунок 4.72 – Настройка внешних интервалов

Обратите внимание, что для полей **Нижняя граница** и **Квота нижней границы** тип данных изменился в соответствии с типом связанного поля. Перейдем на следующий шаг настройки.

При необходимости мы по-прежнему можем выбрать любой другой метод квантования для нашего поля, а также для других полей в случаях квантования нескольких.

В доступных методах квантования появился метод **Внешние диапазоны**. Выберем его.

В настройках метода задается параметр **Идентификатор**. В нашем случае идентификатор не был задан, так как мы квантуем всего одно поле. В случае, когда полей несколько, для каждого в данном параметре необходимо будет указать соответствующий идентификатор интервала. Чтобы доступные идентификаторы появились в списке, необходимо активировать входные порты.

После нажатия кнопки **Рассчитать интервалы** можно увидеть, что номера интервалов проставились автоматически, а остальные значения были взяты из поданного набора, либо рассчитаны на их основе. При необходимости здесь можно изменить настройки также, каки при использовании любого другого метода квантования.

Оставим настройки без изменений и перейдем к просмотру результатов.

На первом выходе мы получаем набор данных, квантованный по заданным нами внешним интервалам. Обратите внимание, что в качестве идентификатора интервалов автоматически проставилось имя квантуемого поля.

На втором выходе будет таблица заданных нами диапазонов

При работе с квантованием необходимо учитывать, что это – так называемый **компонент с обучением**. В данном случае это означает, что после первичного расчета границ интервалов на каком-либо наборе данных, эти границы фиксируются и сохраняются в узле. Поэтому если далее мы подадим на вход настроенного узла другой набор, границы не пересчитаются в соответствии с новыми значениями признака. Вместо этого новые значения распределятся по рассчитанным ранее интервалам. Если же в новом наборе окажутся значения, которые ни в один интервал не попадают, их можно будет отследить по полю **Значение вне диапазонов**.

Для того, чтобы границы интервалов квантования пересчитались на новых данных, узел **Квантование** необходимо будет переобучить. Для этого в контекстном меню узла существует специальная команда.

Подробнее об обучении и переобучении узлов можно прочитать в справке по Loginom.

3.7. Лабораторная работа №10 «Скользящее окно»

Скользящее окно широко применяется в задачах прогнозирования. С его помощью можно выделить непрерывный отрезок данных – окно – которое может перемещаться по всему набору данных, позволяя в одну запись поместить для некоего объекта его текущее значение, а слева и справа от него – значения, смещенные от текущего в прошлое и будущее соответственно.

Для рассмотрения работы компонент нам понадобятся наборы данных **sales.lgd** и **calendar.lgd**.

Добавим в пакет новый модуль и импортируем в сценарий файл **sales.lgd** с продажами строительных товаров, уже знакомый нам по занятиям предыдущих блоков.

Пусть нам необходимо получить таблицу, где запись для каждого товара с информацией по количеству проданного за текущий месяц будет дополнена значениями за два прошлых и один будущий месяц.

Для решения задачи нам потребуется предобработка данных. Для начала выделим из поля **Дата** месяц продажи, выбрав разбитие **Год+Месяц** с типом **Дата/Время**. Далее проведем группировку данных, где в качестве групп

укажем поля **Дата (Год+Месяц)** и **Товар**, а показателем будет **Количество** с вариантом агрегации **Сумма** (рисунок 4.73).

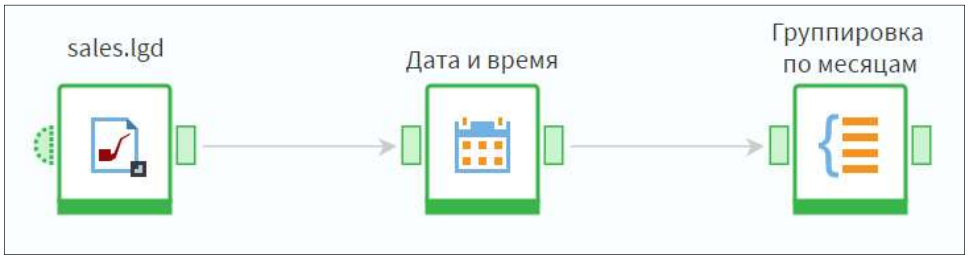


Рисунок 4.73 – Пример построения сценария

Необходимо отметить, что в некоторые месяцы продаж определенных товаров не было. История продаж начинается с марта 2016 года, и для выбранного товара, в частности, отсутствуют данные за несколько месяцев. Отсутствующие месяцы: необходимо добавить в набор, заполнив для них поле **Количество** нулями.

Возникшую проблему можно решить с помощью специального справочника – календаря, – где для каждого товара сформирован полный ряд дат.

В этом нам может помочь компонент **Генератор календаря** библиотеки Loginom Silver Kit. Перейдем в ссылки.

Выберем соответствующий компонент из списка производных компонентов библиотеки и с помощью команды **Добавить узел Выполнения узла** в сценарий вставим его в рабочую область построения сценария. Перейдем в порт переменных узлов.

Напомним, история продажи у нас начинается с марта 2016 года и заканчивается декабрем. Укажем соответствующие значения переменным, тип периода – месяц. На выходе получаем требуемый список месяцев.

Нам необходимо, чтобы по каждому товару был полный список месяцев, поэтому нам также нужно получить список товаров. Сделаем это с помощью узла группировки, в настройках которого укажем поле **Товар** в качестве группы.

Теперь воспользуемся узлом слияния, в настройках которого укажем тип операции **полное соединение**, а связи по ключевым полям создавать не будем (рисунок 4.74).

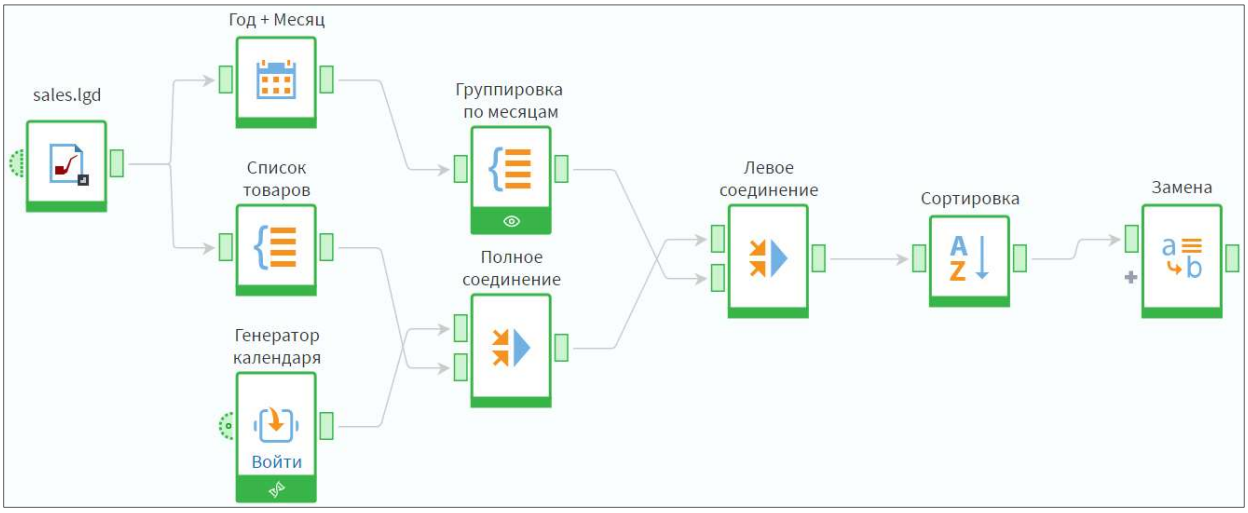


Рисунок 4.74 – Пример установления связей

Таким образом мы получили полный список периодов по каждому товару.

Добавим в сценарий еще одно **слияние** и подадим данные на его порты, как показано на рисунке 4.74. В настройках выберем операцию **Левое соединение** и создадим связи: **Календарь –Дата (Год+Месяц)** и **Товар–Товар** – и переименуем узел в соответствии с выбранной операцией.

Отсортируем данные по возрастанию по полям **Товар** и **Календарь**.

В результате мы получили пустые записи по тем месяцам, в которых не было продаж. Теперь нам необходимо заметить пустые значения нулями. Для этого добавим в сценарий узел **Замена** и зададим в настройках замену пустых значений для поля **Количество** на **0**.

Для того чтобы более наглядно продемонстрировать работу компонента **Скользящее окно** с помощью узла **Фильтр строк** выделим записи для какого-либо одного товара.

Свернем в подмодель узлы **Список товаров**, **Генератор календаря** и **Полное соединение**, чтобы сценарий стал компактнее.

Как мы уже знаем компонент Скользящее окно преобразует исходную структуру данных в новую добавляя поля со смещенными значениями записей. Добавим в сценарий соответствующий узел и перейдем в настройку (рисунок 4.75).

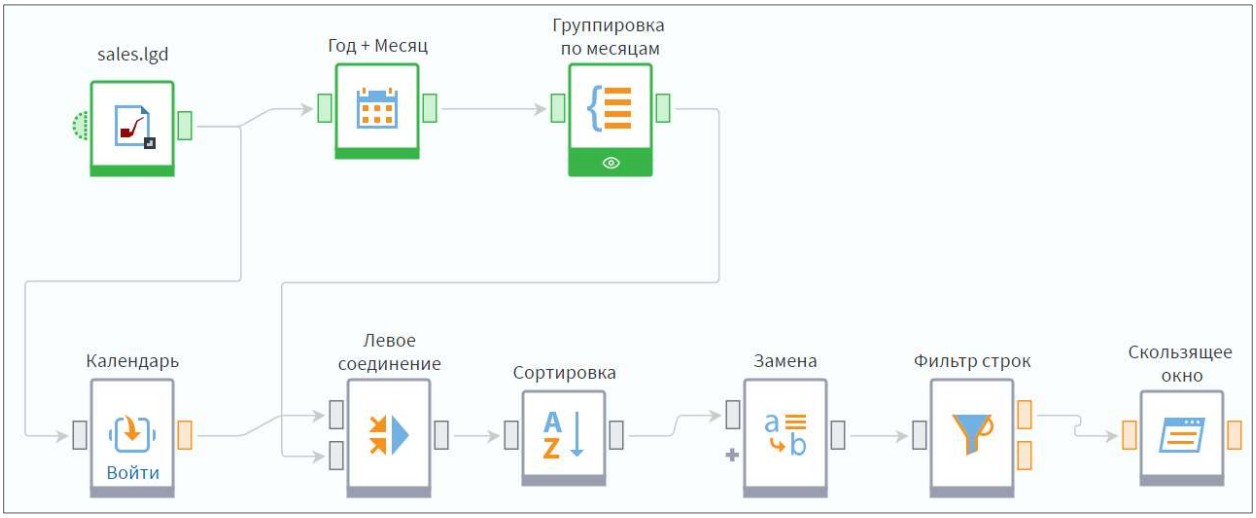


Рисунок 4.75 – Пример добавления узла Скользящее окно в сценарий

Для каждого столбца входных данных доступны два основных параметра настройки:

- **Глубина истории** – количество значений из предыдущих записей, для которых создаются новые поля в выходном наборе;
- **Горизонт прогноза** – количество значений из будущих записей, для которых создаются новые поля в выходном наборе.

Нам необходимо добавить запись для каждого товара информацию о количестве проданного в два предыдущий и один будущий месяц. Зададим параметры как показано на рисунке 4.76 и нажмем кнопку **Применить**.

Столбец	Глубина истории	Горизонт прогноза
ab Товар	Не выбрана	Не выбран
Календарь	Не выбрана	Не выбран
Количество	2	1

Рисунок 4.76 – Настройка узла Скользящее окно

Еще один параметр позволяет задать способ обработки неполных записей:

- **Удалять все неполные записи** – записи, где в полях для смещенных значений содержится хотя бы одна пустая ячейка, не попадут в выходной набор, значение по умолчанию;
- **Удалять добавленные неполные записи** – на выход не попадут записи, полностью добавленные при обработке и не существовавшие во входном наборе;
- **Оставлять неполные записи** – все записи будут сохранены и попадут на выход узла.

Оставим способ обработки без изменения и перейдем к просмотру результатов.

На выходе мы получили три новых поля: **Количество[-2]**, **Количество[-1]** и **Количество[+1]**. Их количество равно сумме параметров **Глубина истории** и **Горизонт прогнозирования**.

Метки новых полей составляются по правилу: <Метка исходного поля>[±< Число>].

Знак ± показывает направление смещения – назад/вперед, – а <Число> указывает на количество периодов (строк) смещения.

Если в параметре **Способ обработки неполных записей** выбрать значение **Оставлять неполные записи**, в выходном наборе появляются дополнительные строки, которые не всегда требуются в сценарии.

При варианте **Удалять добавленные неполные записи** выходной набор выглядит так, как представлено на рисунке 4.77. Отметим, что компонент **Скользящее окно** может потребоваться не только в задачах подготовки временных рядов для прогнозирования, но и при создании каких-либо нетривиальных пользовательских отчетов, например, для вычисления времени между событиями в строках.

Скользящее окно • Быстрый просмотр

Выходной набор данных

#	Календарь	Товар	Количество[-2]	Количество[-1]	Количество	Количество[+1]
1	01.03.2016, 00:00	Балка де…			27,00	0,00
2	01.04.2016, 00:00	Балка де…		27,00	0,00	0,00
3	01.05.2016, 00:00	Балка де…	27,00	0,00	0,00	0,00
4	01.06.2016, 00:00	Балка де…	0,00	0,00	0,00	474,00
5	01.07.2016, 00:00	Балка де…	0,00	0,00	474,00	123,00
6	01.08.2016, 00:00	Балка де…	0,00	474,00	123,00	38,00
7	01.09.2016, 00:00	Балка де…	474,00	123,00	38,00	0,00
8	01.10.2016, 00:00	Балка де…	123,00	38,00	0,00	184,00
9	01.11.2016, 00:00	Балка де…	38,00	0,00	184,00	0,00
10	01.12.2016, 00:00	Балка де…	0,00	184,00	0,00	

Таблица Форма

Рисунок 4.77 – Вариант представления данных при выборе Удалять добавленные неполные записи

Вернем настройку **Удалять все неполные записи** и подадим на **Скользящее окно** данные непосредственно с узла замены.

Так как узел просто обращается к данным предыдущих и последующих строк, теперь для каждого следующего товара в первых двух строках берутся

данные по предыдущему товару, а в последней – по следующему. Такие записи нужно отфильтровать. В этом нам поможет компонент **Калькулятор**.

Добавим в сценарий соответствующий узел и настроем его.

Нам требуется логическое поле, в котором будет стоять значение **true**, если запись нужно исключить из набора. Поставим для этого поля флаг **Кэшировать**, т.к. нам понадобится функция **Data()**.

Нам точно известно, что нужно убрать две первые и одну последнюю записи для каждого товара. Выражение, которое позволит это сделать, показано на рисунке 4.78.

```
If(RowNum()=0 or RowNum()=1 or RowNum()=RowCount()-1,false,
    If(ArticleName<>Data("ArticleName",RowNum()-1) or
ArticleName<>Data("ArticleName",RowNum()-2) or
ArticleName<>Data("ArticleName",RowNum()+1),true,
    false))
```

Рисунок 4.78 – Формула в узле Калькулятор

Флаг проставлен для записей, которые необходимо исключить из набора. Теперь достаточно отфильтровать их, и мы получим корректное скользящее окно по каждому товару.

3.8. Транспонирование данных

Большинство алгоритмов аналитики данных могут применяться только к структурированным данным, то есть данным, представленным в виде таблиц, в которых каждый столбец представляет собой некоторый атрибут или признак, а каждая запись – наблюдение, описывающее состояние анализируемого объекта или процесса.

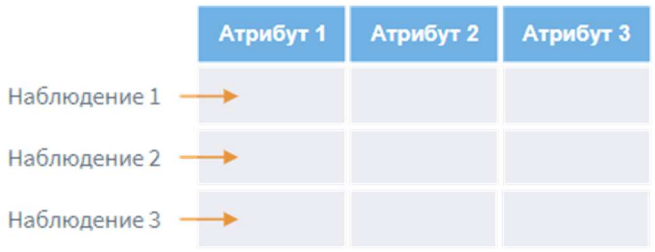


Рисунок 4.79 – Пример структурированных данных

Однако даже если данные являются структурированными, это еще не гарантирует, что структура таблицы соответствует требованиям той или иной задачи анализа.

Действительно, для одного и того же набора данных можно построить множество табличных представлений. При этом некоторые представления способны привести к частичной и даже полной утрате смысла данных.

В лучшем случае это может привести к невозможности построить на основе данных модель или применить к ним готовую модель. В худшем можно получить некорректные результаты анализа, сделать по ним неправильные выводы и заключения.

Иллюстрацией к такой ситуации может служить построение графика (рисунок 4.50). Для получения корректного графического представления данных значения независимой переменной (например, **Дата продажи**) откладываются по горизонтальной оси, а зависимой (например, **Сумма продажи**) – по вертикальной. Если сделать наоборот, то разобраться в таком графике будет очень непросто.

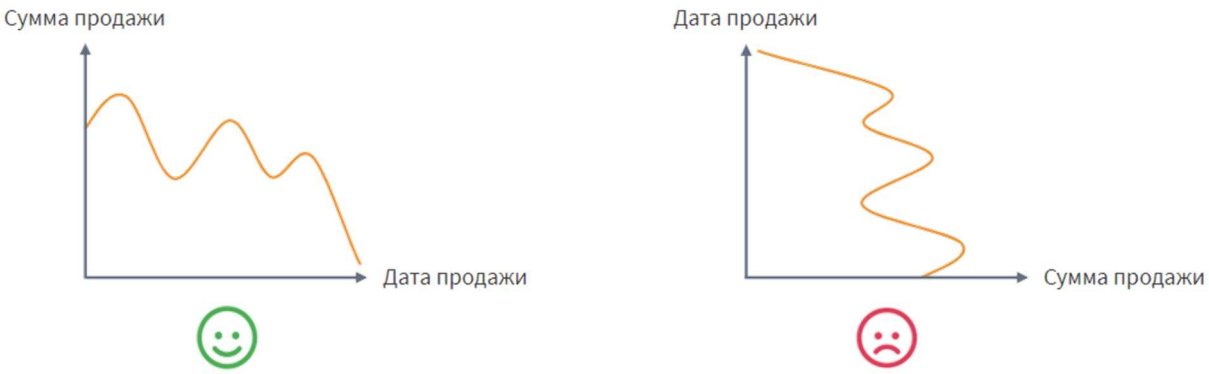


Рисунок 4.50 – Пример корректного и некорректного представления данных

Ситуации, когда структура таблицы, в которой содержатся анализируемые данные, не соответствует решаемой задаче, чаще всего возникают тогда, когда данные поступают с рабочих мест отдельных пользователей, которые строят таблицу, как умеют, или как им удобно.

Приведем пример. Пусть имеется плоская таблица, по которой требуется построить модель для прогнозирования временного ряда, которую пользователь заполнил в виде, представленном на рисунке 4.51.

Дата	01.03.17	02.03.17	03.03.17	04.03.17	05.03.17	06.03.17
Товар						
Количество						
Сумма						
Остаток						

Рисунок 4.51 – Пример некорректных данных

Использование такой таблицы в качестве источника данных для алгоритма прогнозирования крайне неудобно, поскольку даты, товары и количественные показатели продаж должны образовывать столбцы (поля), а наблюдения – строки (записи).

В данном же случае наблюдение – это информация о товаре, его количестве, сумме и остатке на определенную дату. При этом наблюдения образуют столбцы, что противоречит самой идее структуризации.

Кроме того, поля источника данных должны быть типизированы, то есть содержать данные только одного типа. В представленной таблице 4.51 поля содержат значения различных типов: строковые – для наименований товаров, и числовые – для суммы, количества и остатков, то есть тип данных каждого столбца – переменный.

Такая ситуация в большинстве случаев приведет к ошибке несоответствия типов данных при попытке применить к данным из таблицы какую-либо обработку.

Ситуацию можно легко исправить, если применить к таблице операцию, которая преобразует строки таблицы в столбцы, а столбцы – в строки. В результате мы получим те же данные, но другой структуры (рисунок 4.52).

Дата	Товар	Количество	Сумма	Остаток
01.03.2017				
02.03.2017				
03.03.2017				
04.03.2017				
05.03.2017				
06.03.2017				

Рисунок 4.52 – Пример корректных данных

Транспонирование – это термин из теории матриц, который обозначает операцию, преобразующую столбцы матрицы в строки, а строки – в столбцы.

При работе с таблицами, содержащими анализируемые данные, этот термин имеет более широкий смысл. Такие таблицы могут иметь сложную структуру, десятки полей измерений и фактов. В результате возникает противоречие между «плоской» структурой таблицы и «многомерным» характером содержащихся в ней данных.

Традиционно в бизнес-аналитике для выбора наиболее удобного представления многомерных данных используются **OLAP-кубы**, но OLAP-куб – это всего лишь средство визуализации многомерных данных (рисунок 4.53).

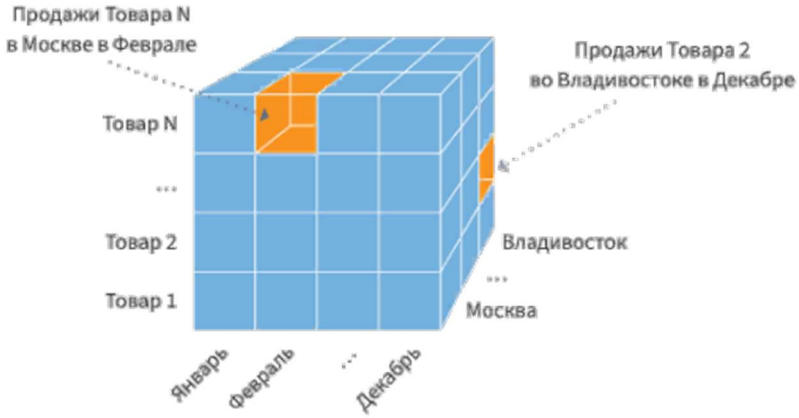


Рисунок 4.53 – Пример OLAP-куба

Манипулируя OLAP -кубом для выбора оптимального представления, мы не изменяем саму структуру источника данных. Можно сказать, что с помощью OLAP -куба пользователь управляет представлением данных, а с помощью транспонирования – структурой источника данных.

Транспонирование на этапе ETL используется для оптимизации структуры источника данных сточки зрения определенной задачи. С помощью транспонирования можно не только менять местами строки и столбцы таблицы, но и проводить более сложные манипуляции с ее структурой.

В качестве источников данных часто выступают файлы баз данных самых различных форматов, поэтому, получая источник данных, аналитик заранее не знает, в каком виде эти данные будут загружены в информационную систему компании.

Например, возможна ситуация, когда данные из первичного источника представлены структурой, как это демонстрирует таблица 1 (рисунок 4.54). Каждое наблюдение (клиент) оказывается размещенным в нескольких строках, а атрибуты не образуют поля.

С помощью транспонирования можно «развернуть» таблицу 2, как показано на рисунке 4.54.

Таблица 1

ФИО клиента	Атрибут	Значение атрибута
Иванов И.И.	Возраст	36
Иванов И.И.	Доход	18 000
Иванов И.И.	Стаж работы	15
Сидоров П.В.	Возраст	45
Сидоров П.В.	Доход	24 000
Сидоров П.В.	Стаж работы	21

Таблица 2

ФИО клиента	Возраст	Доход	Стаж работы
Иванов И.И.	36	18 000	15
Сидоров П.В.	45	24 000	21

Рисунок 4.54 – Пример «разворачивания» таблицы

Еще одной проблемой, с которой часто приходится сталкиваться аналитикам при работе с источниками данных, являются нарушения в их структуре.

Если в витринах, хранилищах и базах данных регулярность структуры данных поддерживается автоматически, то в файлах отдельных пользователей структура таблиц не является жестко заданной.

Особенно характерна данная ситуация для файлов электронных таблиц, где в случае сложной, иерархической структуры данных появляются заголовки, общие для нескольких полей.

Типичный пример представлен на рисунке 4.55, где в таблице присутствуют три измерения: **Дата**, **Товар** и **Группа товара**. При этом измерение **Товар** является иерархически подчиненным измерению **Группа товара**. Часто встречаются еще более сложные структуры, например, товары могут группироваться по городам, где они продавались, фирмам-поставщикам и клиентам и так далее.

Дата	Группа 1						Группа 2			
	Товар A		Товар B		Товар C		Товар D		Товар E	
	Кол-во	Сумма	Кол-во	Сумма	Кол-во	Сумма	Кол-во	Сумма	Кол-во	Сумма
01.03.17										
02.03.17										
03.03.17										

Рисунок 4.55 – Пример нарушения структуры данных

Операция **обратного транспонирования** позволяет избавиться от структурных нарушений в таблице. В результате обратного транспонирования таблицы может быть получена структура, показанная на рисунке 4.56.

Дата	Группа товара	Товар	Кол-во	Сумма
01.03.17	Группа 1	Товар A		
01.03.17	Группа 1	Товар B		
01.03.17	Группа 1	Товар C		
01.03.17	Группа 2	Товар D		
01.03.17	Группа 2	Товар E		
02.03.17	Группа 1	Товар A		
02.03.17	Группа 1	Товар B		
02.03.17	Группа 1	Товар C		
02.03.17	Группа 2	Товар D		
02.03.17	Группа 2	Товар E		
03.03.17	Группа 1	Товар A		
03.03.17	Группа 1	Товар B		
03.03.17	Группа 1	Товар C		

Рисунок 4.56 – Результат применения обратного транспонирования

В каждой записи новой таблицы содержится наблюдение по каждой дате, группе товара и отдельному товару, структура данных является полностью регулярной, а столбцы – типизированы. Такая таблица вполне отвечает требованиям, предъявляемым источникам данных, хотя и выглядит более громоздкой и избыточной.

Для выполнения операции транспонирования пользователь должен определить, какие именно измерения и факты исходной таблицы должны войти в транспонированную таблицу, а также, какие измерения должны отображаться в столбцах, а какие – в строках.

При этом для выполнения операции транспонирования должно быть выбрано хотя бы одно поле фактов, поскольку именно факты являются «связующим звеном» измерений.

Иными словами, какие бы изменения в таблице не происходили в результате ее транспонирования, факты должны быть жестко связаны со своими измерениями.

3.9. Лабораторная работа №11 «Расчет скорингового балла»

Для анализа рассмотрим два набора данных: **Карта.lgd** и **Клиенты.lgd**.

Карта.lgd – набора данных, который описывает бальную скорринговую карту, которая состоит из набора пар **Характеристика-Атрибут** и соответствующих им **баллов**.

Клиенты.lgd –список клиентов с известными характеристиками и атрибутами: **Клиент.Код**, **Возраст**, **Дети**, **Жилье**, **Кредитная история**, **Пол**, **Семейное положение**, **Стаж в отрасли**, **Стаж на последнем месте**.

Задание: Требуется разработать сценарий, который позволяет рассчитать итоговый скоринговый балл клиентов, подавших заявки на кредит. Для расчета итогового балла каждому атрибуту клиента присваиваются баллы в соответствии со скоринговой картой, и результат суммируется. К полученной сумме прибавляется начальный (стартовый) балл, называемый **Константа**.

Выходным результатом работы сценария должен быть набор данных из двух полей: **Клиент. Код** и **Итоговый балл**.

Начнем решение нашей задачи. Создадим новый пакет и переименуем первый модуль в **Кейс 1**. Расчет скорингового балла. Далее перейдем в сценарий и импортируем входные наборы данных.

Нам необходимо для каждого значения характеристик клиента начислить балл в соответствии со скоринговой картой. Значит, мы должны объединить карту со сведениями по клиенту. Но в наборе с клиентами каждая характеристика является отдельным полем, и выполнять слияние с таким набором – очень трудоемкая работа, которая сопряжена с серьезным редактированием в случае добавления или удаления характеристики.

Есть более оптимальное решение – для начала привести сведения по клиенту к набору с тремя полями: **Клиент.Код, Характеристика, Атрибут**. Для этого добавим в сценарий узел **Свертка столбцов.**

В настройках узла все поля, которые характеризуют клиента, перенесем в группу Транспонируемые. Поле, идентифицирующее клиента, – в **Информационные**.

Выполним узел и перейдем в настройки выходного порта, чтобы исключить из набора ненужные поля.

Поля **Метки** и **Значения** переименуем, как показано на слайде. Поля **Имена** и **Типы** данных удалим из выходного набора (рисунок 3.57).

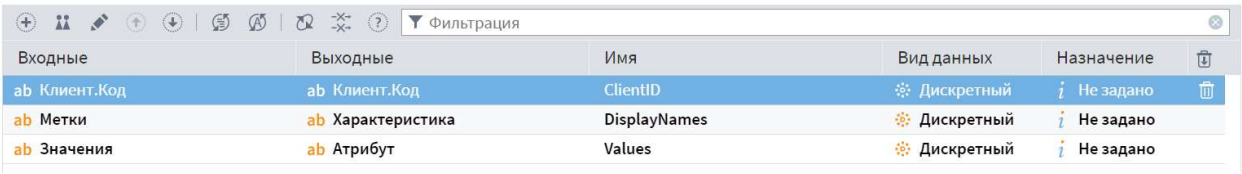


Рисунок 3.57 – Пример настройки узла Свертка столбцов

Получили набор данных, который характеризует клиентов, в новом формате. Теперь мы легко можем начислить клиентам баллы по характеристикам.

Добавим в сценарий узел **Слияние** и назовем его **Начисление баллов**. На порт **Главная таблица** подадим набор со сведениями о клиентах, на порт **Присоединяемая таблица** – скоринговую карту (рисунок 3.58).

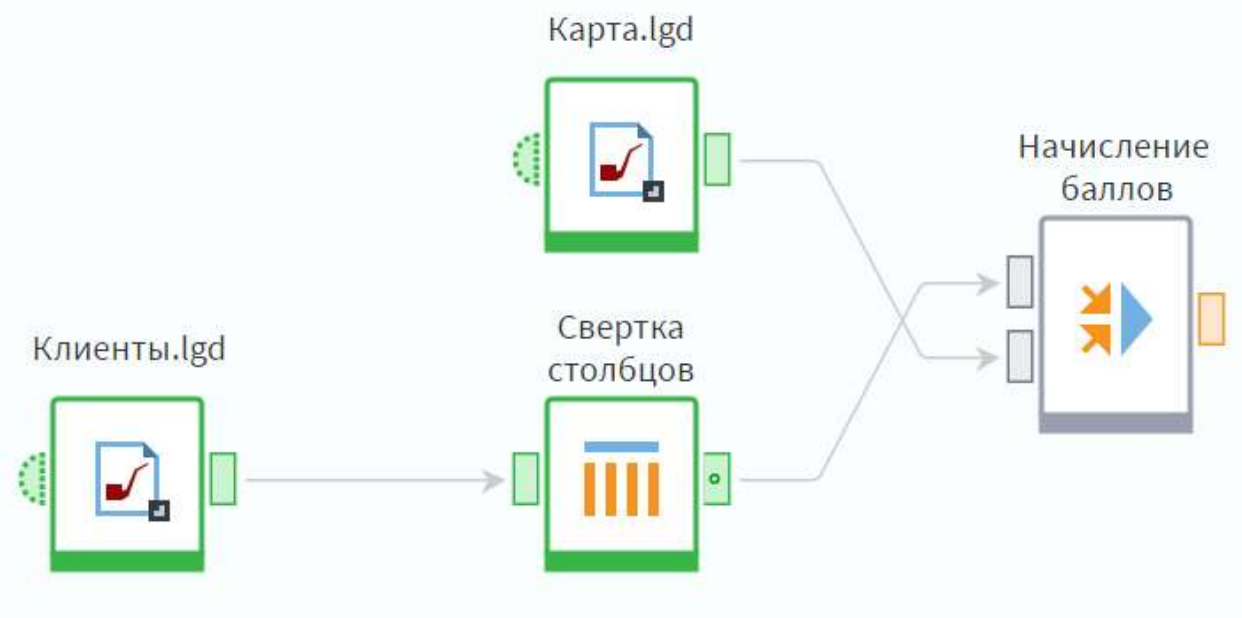


Рисунок 3.58 – Пример построения сценария

В настройках узла оставим тип операции **Левое соединение** и свяжем наборы по полям **Характеристика** и **Атрибут**.

В набор данных добавилось поле **Балл**, где содержатся баллы, соответствующие значениям атрибутов той или иной характеристики каждого клиента.

Теперь необходимо просуммировать количество баллов, набранное каждым клиентом. Для этого воспользуемся узлом **Группировка**.

В настройках узла поле **Балл** добавляем в **Показатели** с вариантом агрегации **Сумма**. Группировать будем по полю **Клиент.Код**. Поля **Характеристика** и **Атрибут** нам больше не понадобятся.

На выходе получаем набор данных с общим количеством баллов, которые набрали клиенты по анкетам. Для получения итогового балла

осталось прибавить к полученной сумме начальный балл (константу), который также содержится в наборе **Карта**.

Прежде, чем перейти к итоговым расчетам, посмотрим на наш сценарий. Получившиеся три узла, по сути, представляют собой один этап – расчет общего балла по анкете каждого клиента. Выделим эти узлы и нажмем кнопку **Свернуть узлы в подмодель**.

Полученную подмодель назовем **Расчет балла по анкетам** и перейдем в ее настройку, чтобы добавить выходной порт.

Добавим выход типа **Таблица** и назовем его **Анкетный балл**. Входные порты также переименуем, задав метки, как показано на рисунке 3.59.

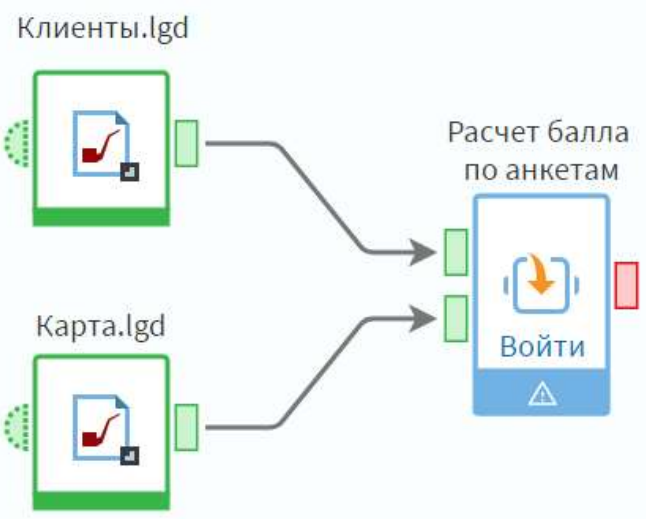


Рисунок 3.59 – Пример настройки Подмодели

Войдем в подмодель.

Подадим на созданный выход данные с узла **Сумма баллов по клиенту** и вернемся в основной сценарий.

Запустим обработку, чтобы убедиться, что узлы в подмодели срабатывают корректно.

Теперь перейдем к расчету итогового балла. Очевидно, что в целом это – еще один, последний этап нашего сценария. Создадим для него подмодель, которую назовем **Расчет итогового балла**.

В настройках подмодели создадим два входа и один выход, как показано на рисунке 3.60.

Рисунок 3.60 – Пример настройки Подмодели «Расчет итогового балла»

Подадим на входы соответствующие наборы данных и войдем в подмодель (рисунок 3.61). Настроим выходной порт.

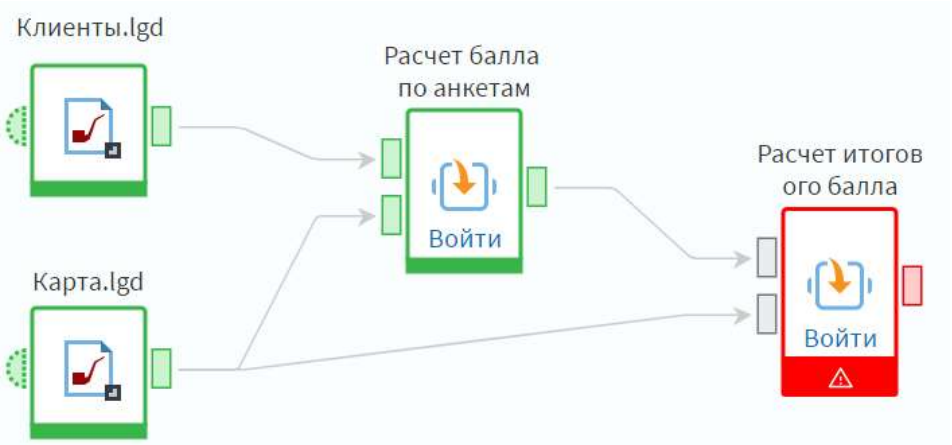


Рисунок 3.61 – Пример построения сценария

Добавим два столбца: **Клиент.Код** и **Итоговый балл** строкового и вещественного типов соответственно – и отключим в порте автосинхронизацию (рисунок 3.62).

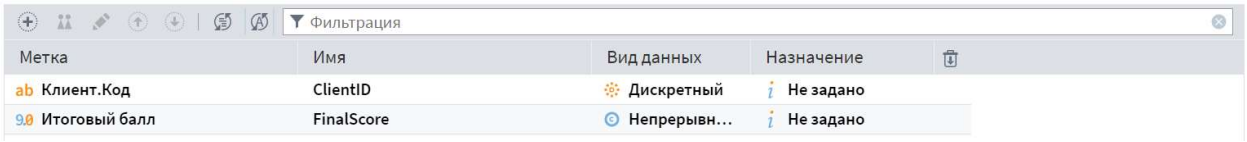


Рисунок 3.62 – Пример настройки выходных портов

Теперь создадим узел **Фильтр строк**, назовем его **Выделение константы** и подадим на вход данные с порта **Скоринговая карта**.

В настройках фильтра зададим условия: **Характеристика = <Константа>**. Запустив обработку, на выходе **Соответствуют условию** получаем значение константы. Константа представляет собой единственное значение, поэтому сделаем из нее переменную. Добавим в узел Таблица в переменные и переименуем как показано на рисунке 3.63.

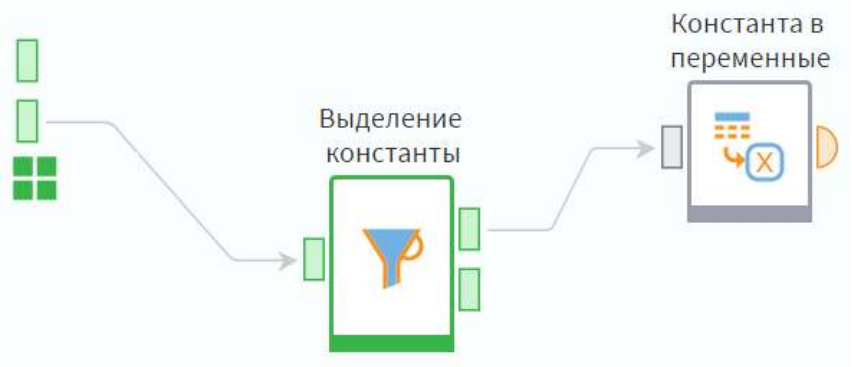


Рисунок 3.63 – Добавление узла Таблица в переменные

В настройке узла добавим в переменные поле **Балл**. Так как в поле содержится всего одно значение, вариант агрегации можно не менять.

В настройке выходного порта узла зададим переменной имя **InitialScore** и метку **Начальный балл**.

Далее добавим узел **Калькулятор** и назовем его **Итоговый балл**. На порты необходимо подавит данные, как показано на рисунке 3.64.

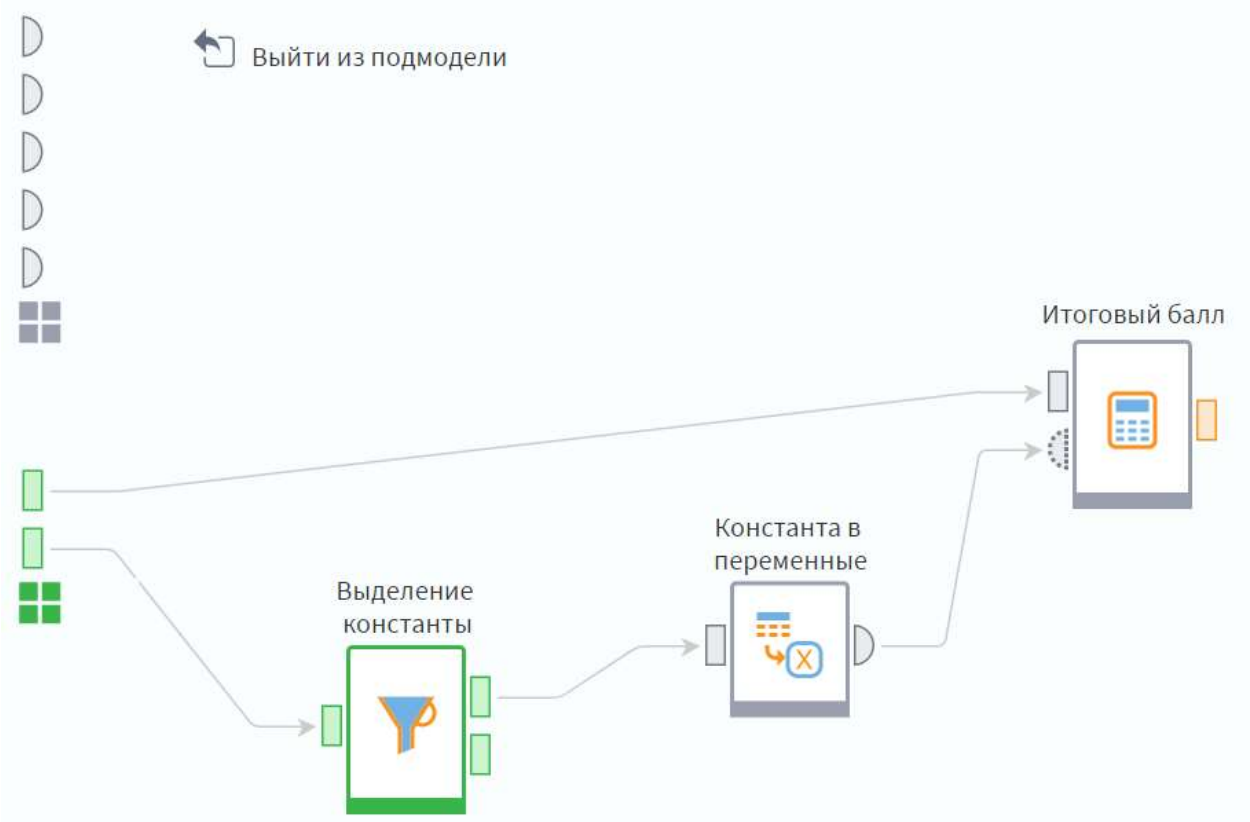


Рисунок 3.64 – Добавление в сценарий узел Калькулятор

Зададим форму расчета итогового балла **Score+InitialScore**. Данные с выходного порта калькулятора передадим на выход нашей подмодели и выйдем из нее. Запустим обработку и откроем быстрый просмотр. В

результате мы получили набор данных, где для каждого клиента рассчитан итоговый балл на основе скоринговой карты.

3.10. Лабораторная работа №12 «Анализ динамики структуры чека»

В данном практическом кейсе мы преобразуем данные по продажам товаров клиентам в разрезе чеков за некоторый период. Результат данного преобразования демонстрирует динамику структуры чека, что позволяет судить об успешности бонусной программы и оценивать лояльность клиентов.

Для выполнения анализа предоставлен набор данных **Чеки.lgd**. Он содержит следующие поля:

- **Дата** – дата и время продажи товара;
- **Чек** – уникальный код, подтверждающий покупку;
- **Карта** – идентификационный код дисконтной карты клиента;
- **Код товара** – код проданного товара в чеке;
- **Количество** – количество штук проданного товара;
- **Сумма** – стоимость товара.

Задание: Нам необходимо преобразовать предоставленные данные и рассчитать дополнительные показатели таким образом, чтобы на выходе получить следующую информацию:

- изменение доли чеков по годам с различным количеством позиций в общем количестве чеков;
- изменение доли объемов продаж и количества в разрезе количества позиций в чеке по годам (рисунок 3.65).

Количество позиций в чеке	2015			...		
	Доля чека в общем объеме продаж	Доля чека в общем количестве продаж	Доля чека в общем количестве чеков	Доля чека в общем объеме продаж	Доля чека в общем количестве продаж	Доля чека в общем количестве чеков
1 позиция						
от 2 до 3						
от 3 до 5						
от 5						

Рисунок 3.65 – Пример задания

Импортируем входные данные из файла **Чеки.lgd**. Согласно постановке задачи, анализ будет выполняться в разрезе года. Поэтому для начала нужно выделить год из поля **Дата**. Воспользуемся компонентом **Дата и время**.

В настройках узла выберем тип разбиение **Год** с числовым типом данных.

Перейдем в настройку выходного порта и изменим для поля **Дата (Год)** вид данных на **Дискретный**. Это необходимо, чтобы при построении отчета мы могли использовать данное поле в качестве колонки в кросс-таблице. Проверим результат преобразования.

В наборе данных появилось поле **Дата (Год)**. Теперь можно перейти к расчету количества позиций в чеке.

Добавим в сценарий узел группировки и переименуем его соответствующим образом.

В настройках узла укажем в качестве групп поля **Дата (Год)** и **Чек**. Поле **Код товара** добавим в **Показатели** с вариантом агрегации **Кол-во уникальных** – это позволит посчитать количество позиций в каждом чеке, при этом если одна и та же позиция записана несколько раз, она будет считаться только один. Также показателями назначим поля **Количество** и **Сумма** с вариантом агрегации **Сумма**. Поля **Дата** и **Карта** нам больше не понадобятся.

Получили необходимую информацию о количестве позиций.

Нас интересуют доли чеков в разрезе числа позиций, поэтому далее определим, сколько чеков имеет одинаковое количество позиций. Снова воспользуемся группировкой (рисунок 3.66).

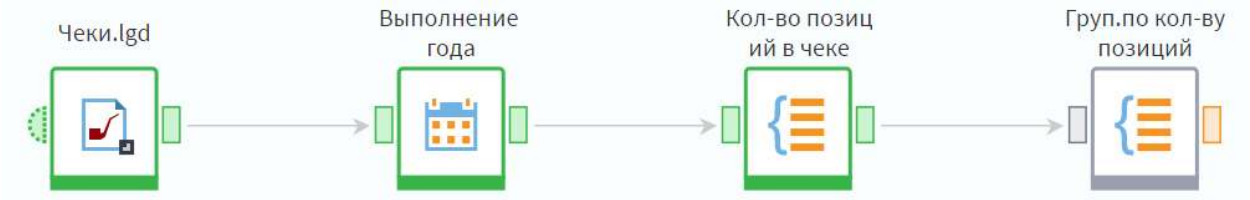


Рисунок 3.66 – Пример построения сценария

На этот раз в качестве групп выступают поля **Дата (Год)** и **Код товара|Количество** уникальных, а показателями будут **Количество|Сумма** и **Сумма|Сумма**, снова с вариантом агрегации **Сумма**.

Поле **Чек** также необходимо добавить в показатели, чтобы получить расчет количества чеков, для него укажем вариант агрегации **Кол-во уникальных**.

В данном случае мы могли выбрать вариант агрегации **Количество** для чеков, так как ранее провели группировку таким образом, что номера чеков в наборе повторяться не могут. Но в целом в данных по продажам номера чеков часто дублируются, как в нашем исходном наборе: когда в чеке несколько товаров, и коды товаров присутствуют в данных, идентификатор чека будет одинаковым для каждого товара в этом чеке.

Аналогична ситуация для позиций в чеке: один и тот же товар может быть включен в чек несколько раз при покупке клиентом нескольких одинаковых товаров.

Таким образом, при выборе между этими вариантами агрегации нужно обращать внимание на то, какой результат вам нужно получить: количество записей по группе или количество уникальных записей по группе.

Перейдем в настройки выходного порта узла группировки и изменим имена и метки полей, как показано на рисунке 3.67. После этого необходимо отключить автосинхронизацию.

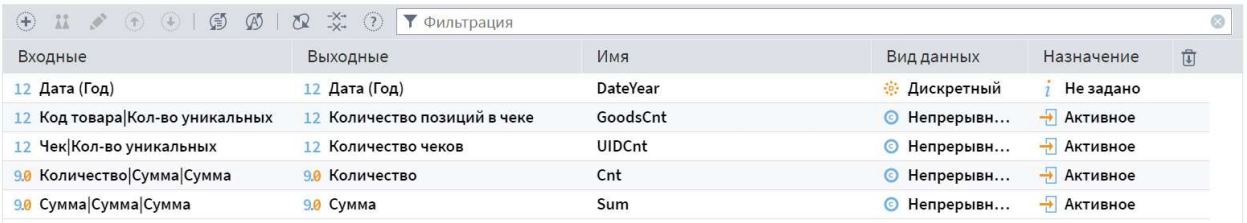


Рисунок 3.67 – Пример настройки выходного порта Груп.по кол-ву позиций

В результате мы получили набор данных, где отражено ‘изменение структуры чека по абсолютным показателям.

Но анализ данных затруднен, так как в наборе есть единичные чеки с большим количеством позиций (например, строки 5 и 8). Такие группы необходимо, объединить в одну. Для этого добавим в сценарий узел **Квантование** (рисунок 3.68).

Рисунок 3.68 – Пример построение сценария

Для поля **Количество позиций в чеке** выберем метод квантования **Плитка**, укажем 4 интервала, а в поле **Совпадающие наблюдения** выберем **Одинаковые плитки**.

Нам нужно будет скорректировать границы и метки интервалов вручную, поэтому нажмем кнопку **Рассчитать интервалы**.

Настроим границы: установим флаг **Верхняя граница открыта**, после чего в нижней части окна выставим для всех интервалов такой тип границ, чтобы нижняя граница входила в интервал, а верхняя – нет. Далее скорректируем границы таким образом, чтобы значения распределились по 4 группам:

- **1 позиция**;
- **от 2 до 3**;
- **от 3 до 5**;
- **от 5**.

Соответствующим образом изменим метки интервалов (рисунок 3.69).

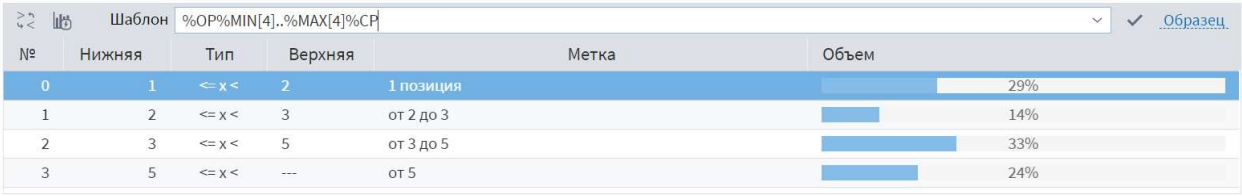


Рисунок 3.69 – Настройка узла Квантование

На следующем шаге поместим поле **Количество позиций в чеке Метка** после поля **Количество позиций в чеке**, чтобы легче было проверить правильность назначения группы, и переименуем его в **Количество позиций в чеке (Метка)**.

Мы видим, что всем значения присвоена соответствующая группа. Теперь проведем группировку количества чеков по группам позиций.

Из новых полей, полученных после квантования, нам понадобится только **Количество позиций в чеке (Метка)**. Используем его в качестве группы совместно с полем **Дата (Год)**, а также добавим показатели, как показано на рисунке 3.70.

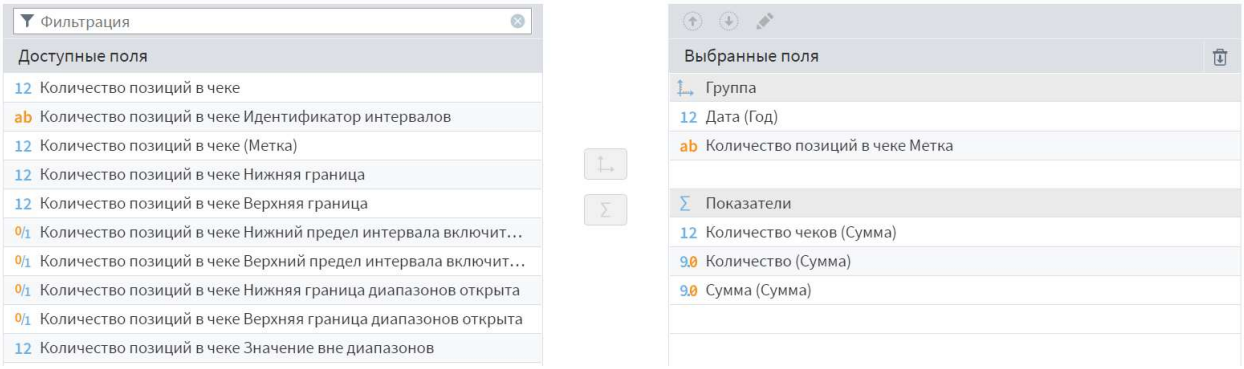


Рисунок 3.70 – Пример настройки узла Группы позиций

Получили распределение количества чеков по годам и по группам позиций. Для упрощения структуры сценария свернем получившиеся узлы в подмодель. Назовем ее **Чеки по кол-ву позиций** и добавим табличный выходной порт, на который подадим данные с узла **Группы позиций**.

Следующие несколько узлов позволяют рассчитывать доли чеков для финального отчета.

Для начала рассчитываем по средствам узла **Группировка (Общие суммы по годам)** общие суммы количества чеков, количества товаров и суммы продаж по годам с помощью узла группировки. Параметры настройки:

- **Дата (Год)** – группа;
- **Количество чеков** – показатель, вариант агрегации **Сумма**;
- **Количество** – показатель, вариант агрегации **Сумма**;
- **Сумма** – показатель, вариант агрегации **Сумма**.

С помощью узла **Слияние (+общие суммы)** выполняем **Левое соединение** по полю **Дата (Год)**, чтобы добавить полученные поля с общими суммами к нашему набору данных.

По средствам узла **Калькулятор** проведем расчет долей (рисунок 3.71).

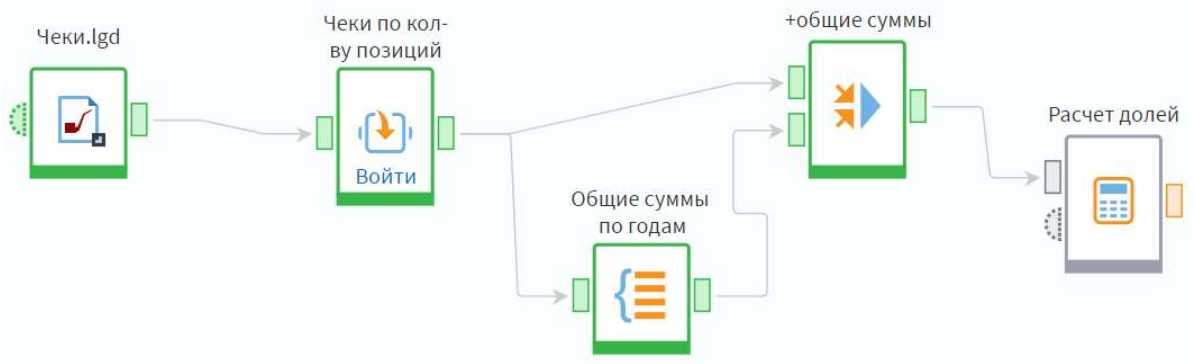


Рисунок 3.71 – Пример построения сценария

В настройке узла Расчет долей введем формулу расчета **Round(Sum/SumTotal*100,2)**.

3.11. Лабораторная работа №13 «ABC-XYZ анализ»

В данном практическом кейсе мы рассмотрим популярные методы классификации ресурсов организации: ассортимента, клиентов, поставщиков и т.д.: **ABC** и **XYZ анализ** и совместный анализ. Для решения задачи мы воспользуемся готовыми компонентами из библиотеки **Loginom Silver Kit**.

В качестве входного набора данных используется информация по продажам строительных товаров (**sales.lgd**). В наборе данных содержатся следующие поля:

- Товар;
- Дата;
- Сумма с учетом скидки.

ABC-анализ. В основе ABC-анализа лежит принцип Парето: «20% усилий дают 80% результата». Результатом ABC-анализа является разделение объектов в зависимости от их вклада в общий итог на три группы:

- **A** – наиболее ценные,
- **B** – промежуточные,
- **C** – наименее ценные.

Такой способ категорирования подсказывает, что нужно вести пристальный контроль за категорией **A**, слабее отслеживать состояние объектов в классе **B** и меньше всего заботиться о классе **C**.

XYZ-анализ. XYZ-анализ позволяет классифицировать объекты в зависимости от характера потребления и точности прогнозирования его изменения. В результате мы также получаем разделение на три группы:

- **X** – ресурсы со стабильной величиной потребления и высокой точностью прогноза,
- **Y** – ресурсы с известными тенденциями потребления (например, сезонными колебаниями) и средними возможностями прогнозирования,
- **Z** – ресурсы с нерегулярным потреблением, какие-либо тенденции отсутствуют, точность прогнозирования невысокая.

Результаты ABC и XYZ анализа можно совместить, получив разделение на 9 групп, которые будут характеризовать ресурсы по обоим критериям. Таким образом, группа **AX** будет содержать самые важные ресурсы: наиболее ценные и стабильно потребляемые, а группа **CZ** – наименее ценные с нерегулярным потреблением (рисунок 3.72).

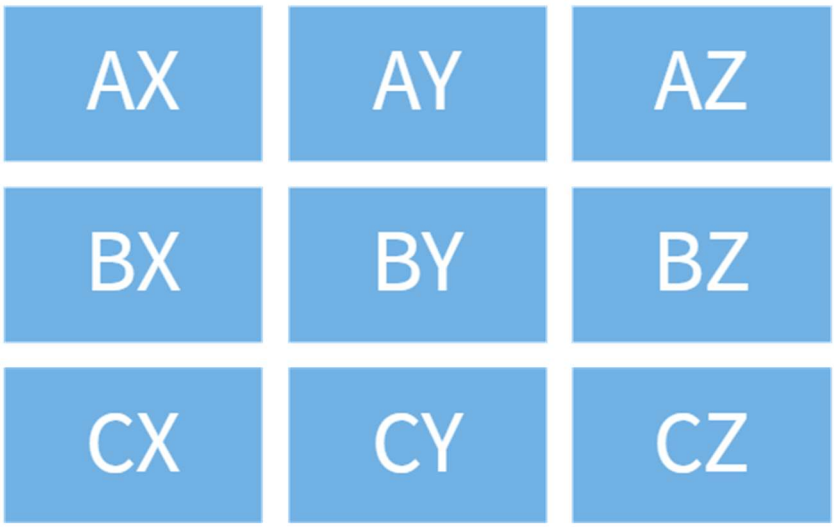


Рисунок 3.72 – Совместный анализ

Задание: Необходимо создать сценарий, который классифицирует товары по объему продаж на основе ABC-анализа и на основе XYZ-анализа. Далее необходимо совместить результаты анализа, разбив товары на 9 групп. После провести группировку, подсчитав, какое количество товаров попало в ту или иную совмещенную группу.

На выходе сценария мы должны получить два набора данных: группы, к которым относится каждый товар, и количественное распределение товаров по совмещенным группам (рисунок 3.73).

Товар	Группа ABC	Группа XYZ	Совместная группа

Совместная группа	Количество товаров

Рисунок 3.73 – Постановка задания

В нашем сценарии мы будем использовать уже готовые компоненты ABC-анализа и XYZ-анализа из библиотеки **Loginom Silver Kit.**

Импортируем файл **sales.lgd** с историей продаж наших товаров. Данные детализированы до дней.

Библиотека **Loginom Silver Kit** является открытой, все ее компоненты доступны и через выполнение узла, и как производные компоненты.

Добавим производный компонент ABC-анализ в область построения сценария с помощью перетаскивания. Передадим на вход наши данные и перейдем в настройку порта.

В общем случае для ABC-анализа необходимо всего два поля: **Объект** и **Показатель**. Таким образом, данный компонент можно использовать для анализа любых сущностей: клиентов, товаров, поставщиков и т.д. – по любым показателям, например, сумма, количество, маржа. Главное – правильно указать соответствия полей. У нас в качестве объекта анализа выступает товар, а в качестве показателя – сумма его продаж. Проставим соответствие и сохраним настройки.

Теперь посмотрим порт переменных.

Здесь задаются границы групп **A** и **B**. Границы групп – это значения, по. результатам сравнения с которыми дол ей накопительного итога по показателю, объект попадает в ту или иную группу. По умолчанию они равны 80% и 95%.

Еще один распространенный вариант – когда в группу **A** попадают объекты, которые внесли вклад в результат 50%. Установим границы, как показано на рисунке 3.74, и сохраним настройки.

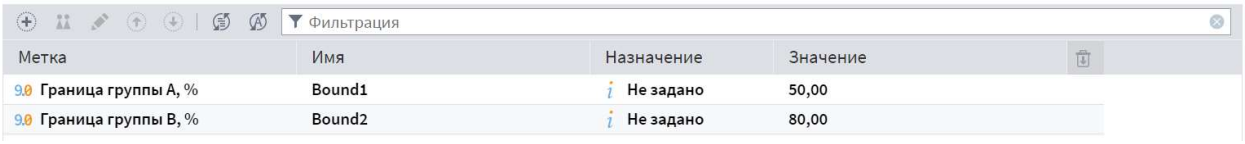


Рисунок 3.74 – Пример настройки узла ABC-анализ (открытый узел)

Мы готовы к проведению ABC-анализа наших клиентов. Выполним узел. Посмотрим какие данные мы получили в выходном порте. Всего в нашем наборе оказалось 1 166 товаров, для каждого из них подсчитан вклад в общую сумму продаж и определена группа.

Теперь нам нужен компонент **XYZ-анализ**. Добавим соответствующий производный компонент в область построения сценария. Настроим входной порт.

На вход требуются такие же поля, как и для ABC-анализа. В нашем случае мы проставим такие же соответствия, но в целом можно проводить ABC-анализ по одному показателю, а XYZ – по другому.

В переменных тоже задаются границы групп. Стандартные значения – 10% для групп **X** и 25% для группы **Y**. Не будем менять их и сохраним настройки. Выполним узел. Посмотрим выходные данные. Здесь кроме группы присутствуют среднее значение и коэффициент вариации по каждому клиенту.

Итак, мы получили результаты ABC и ХУ7 анализа, теперь нам нужно свести эти результаты в один набор данных. Воспользуемся узлом слияния (рисунок 3.74). Подадим на его входы наши данные и настроим порты.

Рисунок 3.74 – Пример построение сценария

В итоговом наборе нам не понадобятся значения показателя и вклада, нас интересует только принадлежность товара группе. Выделим ненужные поля и удалим их с помощью. клавиши **Delete** на клавиатуре. Кроме того переименуем остальные поля: **Объект** снова назовем **Товар**, а **Группа** переименуем в **Группа ABC**.

Аналогичным образом поступим на втором порте.

Далее настроим узел «Слияние». Создадим связь по полю **Товар**, оставив тип операции по умолчанию. Теперь в наборе для каждого товара указана его группа по каждой из классификаций. Осталось получить совмещенные группы и подсчитать количество товаров, которое попало в ту или иную группу.

Далее воспользуемся компонентом **Калькулятор**. В настройках узла создадим строковое поле Совместная группа, в котором запишем выражение, как показано на рисунке 3.75.

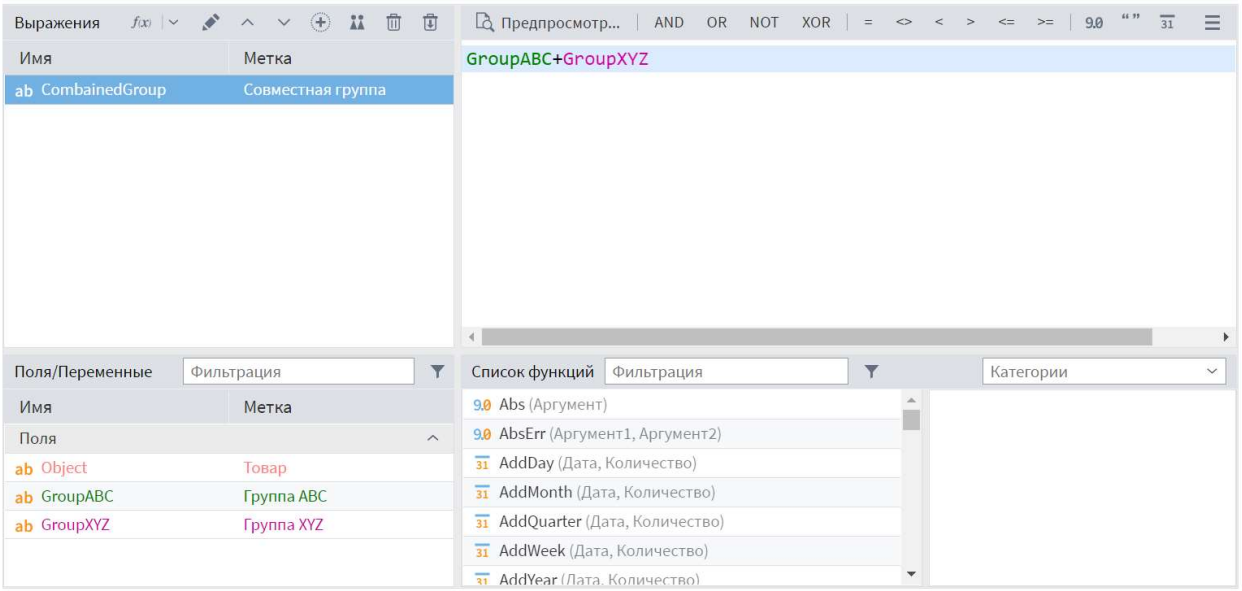


Рисунок 3.75 – Пример настройки узла Калькулятор

На выходе получаем первый из нужным нам наборов. Далее воспользуемся группировкой. В настройках узла поле **Совместная группа** добавим в группы, а поле **Товар** – в показатели с агрегацией **Количество**. Узел назовем **Статистика по группам**. Сохраним изменения и выполним узел.

На выходе получим всего 7 групп, среди наших клиентов не оказалось тех, кто не входит в группы **AX** и **BX**.

Итак, мы провели совмещенный ABC-XYZ анализ и подсчитали статистики по группам. Если данный анализ проводится периодически, имеет смысл свернуть выделенные узлы в подмодель и настроить ей модификатор доступа. Тогда ее можно будет использовать через выполнение узла, что позволит легко сравнивать, результаты анализа в разные периоды. Создадим такую подмодель.

Перейдем в настройку подмодели для добавления портов.

На данный момент порты переменных, в которых задаются границы групп, оказались внутри. Если границы групп меняться не будут, можно оставить все как есть, но лучше предусмотреть возможность задать их снаружи.

Кроме того, как мы видели, на выходе узла группировки могут оказаться не все группы. Рекомендуется использовать полный список групп, чтобы было проще отслеживать переходы между группами во времени. Так как групп немного, их значения можно задать в переменных.

Добавим на входе два порта переменных, а на выходе – два табличных порта и зададим им метки, как показано на рисунке 3.76. Подмодель назовем ABC-XYZ анализ.

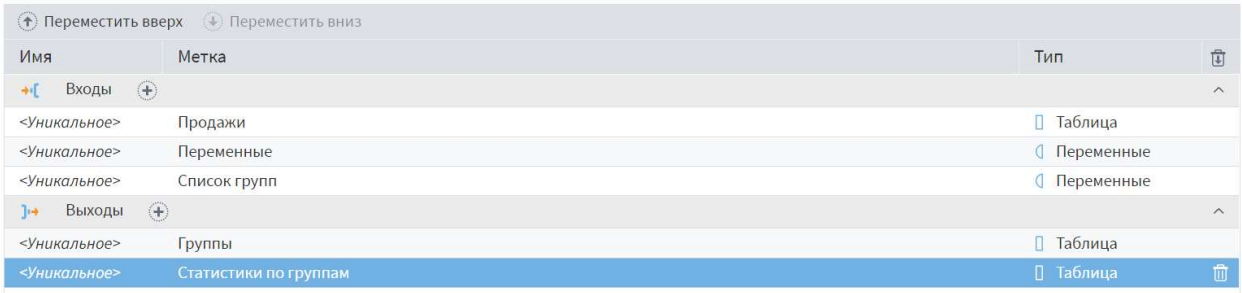


Рисунок 3.76 – Пример настройки узла Подмодель (ABC-XYZ анализ)

Настроим порты.

На табличном входе удалим ненужное поле **Дата**. Остальные поля оставим. Так как мы решаем конкретную задачу, не будем задавать полям абстрактные метки.

На входе Переменные создадим вещественные переменные для указания границ указания групп (рисунок 3.77).

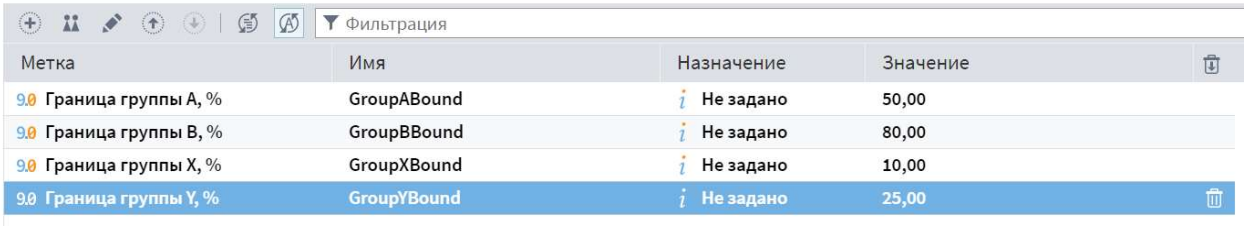


Рисунок 3.77 – Пример настройка входного порта Переменные узла Подмодель (ABC-XYZ анализ)

На последнем входе – Список групп – создадим девять переменных, в имена, метки и значения которых запишем наименования групп (рисунок 3.78).

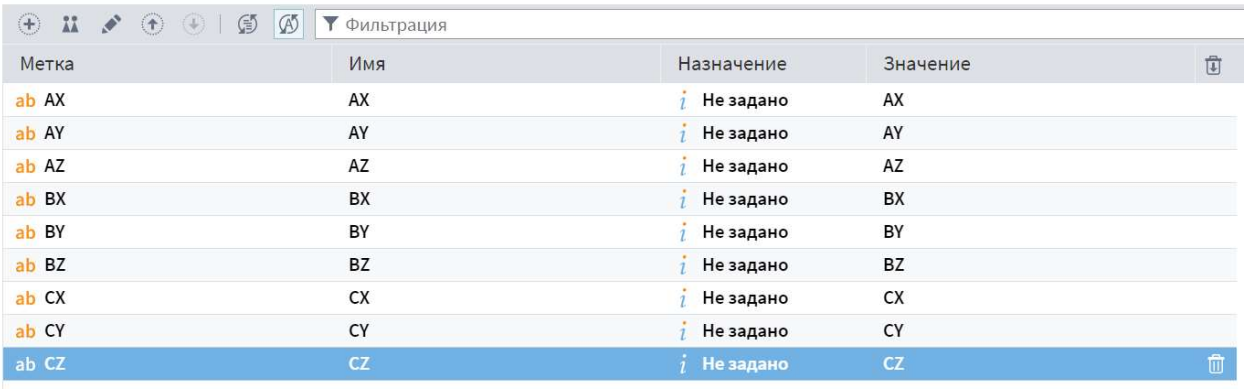


Рисунок 3.78 – Пример настройка входного порта Список групп узла Подмодель (ABC-XYZ анализ)

Войдем в подмодель. С первого порта переменных подадим данные на порты с границами групп и проверим, что соответствия установились корректно.

Далее позаботимся о том, чтобы получить полный список групп. Добавим в сценарий узел **Переменные в таблицу** (рисунок 3.79).

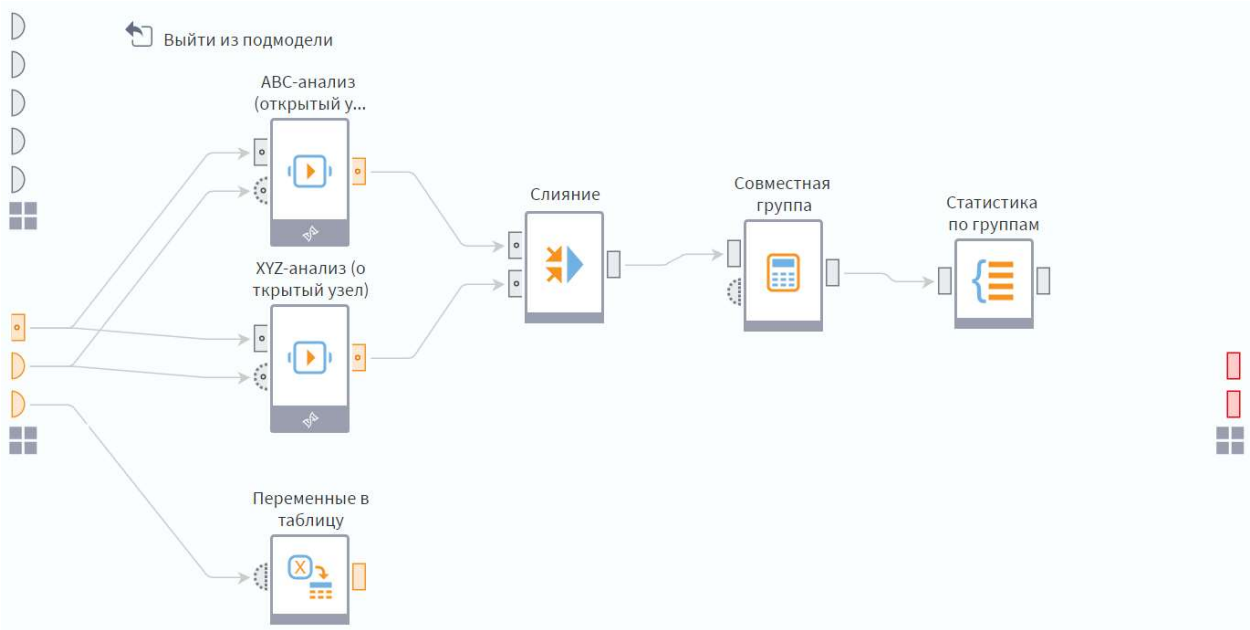


Рисунок 3.79 – Пример построения сценария

В настройках узла Переменные в таблицу переменные запишем в строки. На выходе получаем набор данных со списком групп.

Далее снова воспользуемся слиянием, на входы которого подадим список групп и статистики по группам. Настроем его первый порт. В поле **Метка** переименуем в **Совместная группа**, а остальные поля удалим – они нам не понадобятся.

В настройках самого узла выберем левое соединение по полю Совместная группа. Узлы слияний переименуем, чтобы точно знать, за какие действия они отвечают (рисунок 3.80).

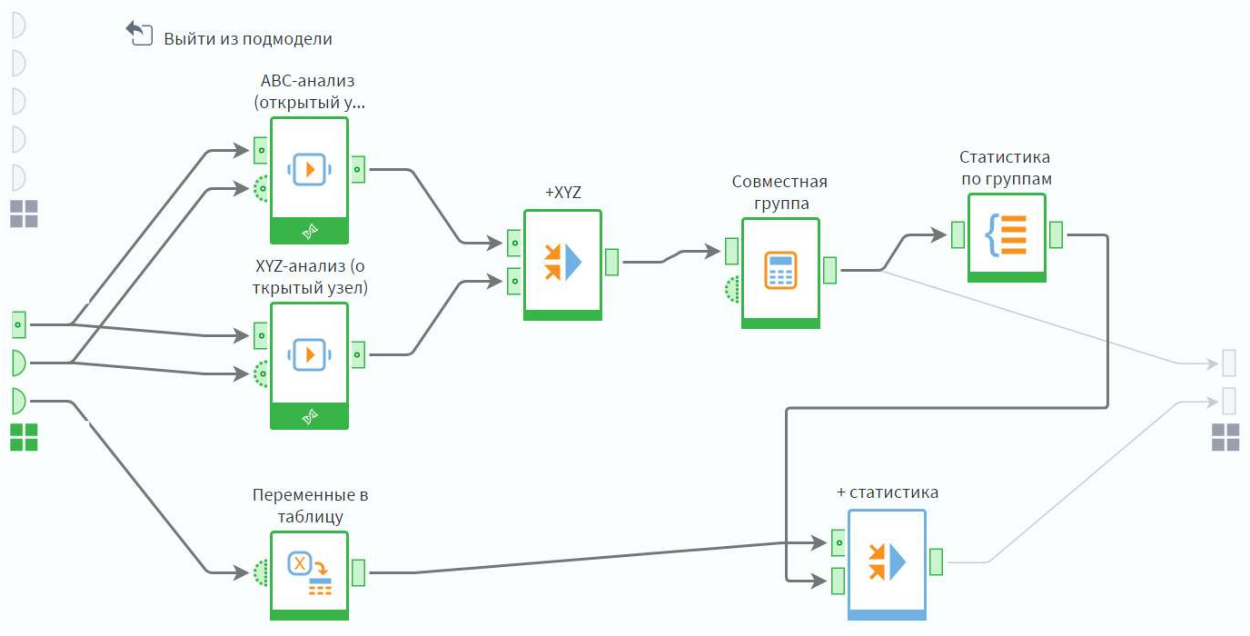


Рисунок 3.80 – Пример построения сценария

Теперь на выходе узла у нас полный список групп. Можно оставить значения для групп, в которых нет клиентов, пустыми, а можно заполнить их нулями с помощью компонента **Замена**. Мы оставим все как есть.

Подадим выходные наборы данных на соответствующие выходы подмодели. На втором выходе поле **Товар|Количество** переименуем в **Количество товаров**.

В результате на первом выходе подмодели получили группы по каждому товару, а на втором – количество товаров в ABC-XYZ-группах. Можно настроить на данную подмодель узел выполнения и сравнивать результаты анализа за разные периоды, таким образом отслеживания перемещения товаров между группами.

3.12. Лабораторная работа №14 «Оценка товарного портфеля»

В качестве входных данных для анализа предоставлены два набора: **Ассортимент товара.lgd** и **Продажи по чекам.lgd**.

Ассортимент товара.lgd – утвержденный к закупке и продаже список товаров, разделенный по товарным группам и подгруппам. Набор данных содержит поля: Артикул; Наименование товара; Товарная группа; Подгруппа.

Продажи по чекам.lgd – исторические данные по продажам. Набор данных содержит поля: Дата; Чек; Артикул; Количество и Цена.

Данные относятся к сети магазинов детских товаров.

Задание: Вам необходимо разработать сценарий, на основании расчетов которого можно будет принять решение о выводе товара из ассортимента

предприятия. Для принятия решения необходима следующая информация по каждой товарной позиции:

- дата первой продажи и дата последней продажи;
- доля в объеме продаж подгруппы товара, рассчитанная по выручке, полученной с продаж;
- число чеков по артикулу – количество чеков, в которых встречался данный артикул. Полученное значение должно быть указано в интервалах: до 10; от 10 до 50; от 50 до 100; от 100 до 200; от 200 до 300; свыше 300.

Результат работы сценария – набор данных, содержащий такую информацию.

Обратите внимание: при ответе на первый вопрос теста по данной задаче слушатели часто ошибаются, так как не все учитывают в сценарии. В тесте ошибки нет, будьте внимательнее при разработке сценария.

3.13. Лабораторная работа №15 «Матрица перехода»

Дан набор данных **Классификация клиентов.lgd**, в нем представлена информация по динамике изменения класса предпочтений постоянных клиентов с течением времени к предлагаемым типам услуг. Набор данных состоит из следующих полей:

- **Код Клиента** – идентификатор клиента в учетной системе предоставления услуг;
- **Квартал** – временной период, в котором определялся класс клиентов;
- **Класс** – присвоенный класс предпочтений клиента.

Задание: Необходимо создать сценарий, формирующий матрицу перехода, в которой будут сравниваться предпочтения клиентов в первом году (это полные 12 месяцев 2016 года) с последующим (полные 12 месяцев 2017 года). Для этого строится кросс-таблица, в столбцах которой откладываются классы предпочтений за предыдущий период, а в строках — за последующий. Внутри таблицы выводится процент клиентов, изменивших в течение времени свои предпочтения. Внешний вид кросс-таблицы представлен на рисунке 3.81.

Период 2 (последующий)	Период 1 (предыдущий)			
	активные приверженцы	частичные приверженцы	непостоянные приверженцы	без предпочтений
активные приверженцы				
частичные приверженцы				
непостоянные приверженцы				
без предпочтений				

Рисунок 3.81 – Внешний вид кросс-таблицы

Рассмотрим пример интерпретации значений матрицы перехода при пересечении столбца «без предпочтений» и строки «активные приверженцы» – это оранжевая ячейка (рисунок 3.82). В указанной ячейке будет отражен процент клиентов, которые не имели в периоде 1 (2016 год) никаких явных предпочтений по оказываемым услугам, но четко определились в течение года со своим выбором и стали активными приверженцами в периоде 2 (2017 год).

Период 2 (последующий)	Период 1 (предыдущий)			
	активные приверженцы	частичные приверженцы	непостоянные приверженцы	без предпочтений
активные приверженцы				
частичные приверженцы				
непостоянные приверженцы				
без предпочтений				

Рисунок 3.82 – Интерпретация значений матрицы перехода при пересечении столбца «без предпочтений» и строки «активные приверженцы»

Рассмотрим пример заполнения матрицы перехода. Предположим, что нашими услугами на протяжении двух лет постоянно пользуется **100** клиентов. Из них в периоде **1** активных приверженцев было **10** и частичных приверженцев – **90**. В периоде **2** из **10** активных приверженцев **2** перешли в частичные, а из частичных приверженцев **18** клиентов стали активными. Заполним матрицу перехода (рисунок 3.83).

Период 2 (последующий)	Период 1 (предыдущий)	
	активные приверженцы	частичные приверженцы
активные приверженцы	8/10*100% = 80%	18/90*100%=20%
частичные приверженцы	20%	80%

Рисунок 3.83 – Матрица перехода

Основные требования к создаваемой матрице перехода:

1. Предыдущий период – первый год обслуживания клиентов; последующий период – следующий год.
2. В предоставленных данных каждый клиент классифицируется раз в квартал. В зависимости от покупаемых услуг он может попадать в разные классы, но для построения матрицы перехода ему необходимо присвоить наименее категоричный класс предпочтений за каждый период (год). Классы упорядочиваются следующим образом по категоричности: 1 – активные приверженцы (самый категоричный); 2 – частичные приверженцы; 3 – непостоянные приверженцы; 4 – без предпочтений (наименее категоричный).
3. Внешне создаваемая матрица перехода должна иметь вид такой же, как отображаемая выше кросс-таблица. Значения ее строк должны идти в том же порядке.

3.14. Лабораторная работа №16 «Массовый расчет агрегатов»

Дан набор данных **Биллинг.lgd**, в нем представлены предварительно агрегированные месячные данные о потреблении телекоммуникационных услуг (за три месяца). После полей **КЛИЕНТ_КОД** и **МЕСЯЦ** следуют 30 различных поведенческих показателей: количество отправленных **sms**, длительность вызовов, объем GPRS-трафика и т.п. Всего в выборке около 320 000 записей по 106,5 тысячам клиентов.

Задание: В процессе построения аналитической отчетности, а также моделей описательной и предсказательной аналитики требуется иметь агрегированное представление о портрете клиента на определенный момент времени. С этой целью его «периодические срезы» (дневные, месячные, квартальные и т.п.) дополнительно агрегируются. Варианты агрегации могут быть разными: от простых (сумма, среднее...) до сложных. Чаще всего используется вариант **Среднее** – средняя продолжительность разговоров

абонента, среднее число позиций в чеке, среднее число дней между просрочками.

Требуется для каждого клиента по каждому из 30 поведенческих показателей рассчитать:

- Минимум;
- Максимум;
- Среднее;
- Сумма.

В данном случае задача простая, и ее можно решить несколькими способами.

В случае с расчетом более сложных агрегатов стандартных вариантов узла **Группировка** будет недостаточно, и придется производить вычисления в узле **Калькулятор**. При разработке сценария обратите на это внимание.

Глава 4. ВИЗУАЛИЗАЦИЯ ДАННЫХ

4.1. Введение в визуализацию

Одной из важнейших составляющих аналитики данных является **визуализация** – представление данных в виде, который обеспечивает наиболее эффективную работу пользователя [4].

Способ визуализации должен максимально полно отражать поведение данных, содержащуюся в них информацию, тенденции, закономерности и так далее.

При этом выбор способа визуализации зависит от характера исследуемых данных и от задачи анализа, а также от предпочтений пользователя.

Многие связывают визуализацию только с интерпретацией, оценкой качества и достоверности результатов анализа. Однако это в корне неверно.

На практике в процессе анализа данных пользователь непрерывно работает с различными визуализаторами.

На разных этапах аналитического процесса визуализация используется для достижения следующих целей и решения следующих задач (рисунок 4.1):

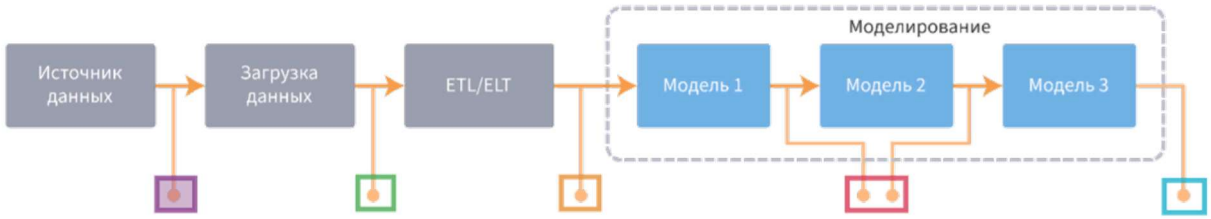


Рисунок 4.1 – Этапы аналитического процесса

- Визуализация источника: идентификация; оценка качества; выдвижение гипотез.
- Визуализация загруженной выборки: проверка результатов загрузки; оценка качества данных.
- Визуализация данных предобработки: проверка результатов предобработки; оценка готовности данных к анализу.
- Визуализация промежуточных результатов: проверка корректности моделей; контроль правильности результатов.
- Визуализация результатов анализа: интерпретация результатов; оценка достоверности результатов.

В источнике требуется визуально оценить: характер, тип и поведение данных; динамический диапазон значений; степень гладкости; наличие факторов, снижающих качество данных, таких как шумы, аномальные и пропущенные значения.

Визуальный анализ позволяет: увидеть, соответствуют ли данные ожидаемым; оценить степень пригодности данных к анализу; выдвинуть гипотезы о закономерностях процессов, описываемых данными; определить, какие виды очистки и предобработки необходимо применить к данным.

Кроме того, визуализация источников данных позволяет определить метод загрузки данных в аналитический контур и параметры, которые при этом должны быть использованы.

Например, для корректной загрузки данных из текстового файла с разделителями необходимо правильно определить символ-разделитель, используемый формат даты и времени, расположение заголовков столбцов и так далее.

Для визуализации источников данных можно использовать приложения, в которых они были созданы (текстовые редакторы, СУБД, электронные таблицы и так далее). Кроме того, аналитические платформы содержат собственные средства предварительного просмотра источников данных.

После загрузки данных из первичного в централизованный источник (хранилище или витрина данных, оперативный склад данных, MDM-система и так далее) работа с выборкой также начинается с визуального анализа. Однако теперь его цели, задачи и методы будут другими (рисунок 4.2).

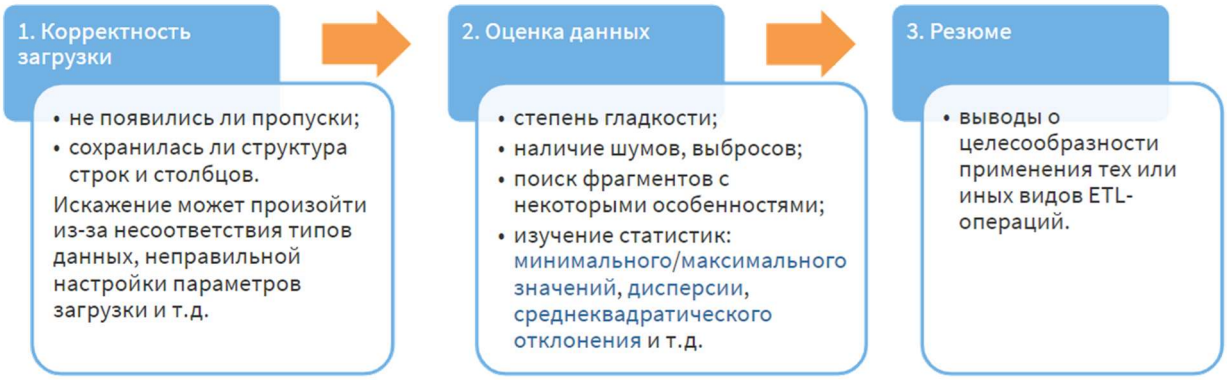


Рисунок 4.2 – Визуализация данных, закруженных в аналитический центр

Сложные аналитические процедуры являются многошаговыми, т.е. в процессе анализа к данным последовательно применяется несколько алгоритмов или моделей. Например, сначала данные подвергаются предобработке с целью сглаживания и кодирования, затем к результирующей выборке применяется та или иная модель.

При этом выборка, формируемая на выходе каждого алгоритма или модели, может подаваться на вход следующего этапа обработки. Очевидно, что если данные, поступившие с предыдущего этапа, окажутся некорректными, то дальнейшая обработка теряет смысл. Поэтому очень важно предусмотреть визуализацию промежуточных результатов анализа с целью проверки корректности используемых моделей и алгоритмов (рисунок 4.3).

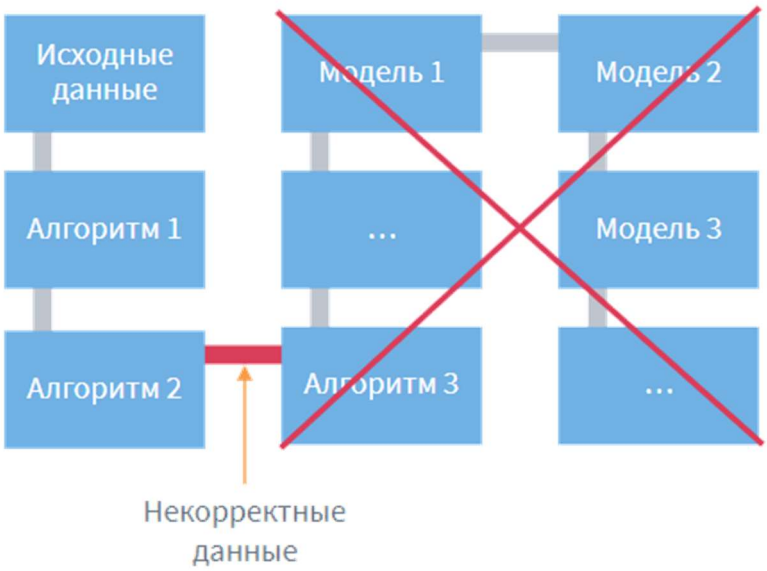


Рисунок 4.3 – Визуализация данных в процессе аналитической обработки

После получения конечных результатов аналитической обработки на первый план выходит задача их интерпретации и оценки достоверности. И здесь не обойтись без визуализации.

Следует заметить, что, даже если в процессе анализа были получены достоверные и ценные результаты, неудачный выбор визуализации не позволит их интерпретировать, увидеть в них зависимости и закономерности (рисунок 4.4).

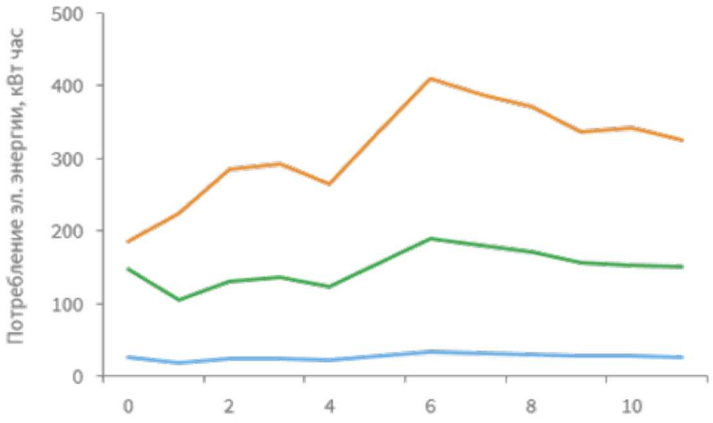


Рисунок 4.4 – Визуализация результатов анализа

В настоящее время в аналитике данных и Data Science используется несколько десятков основных методов визуализации. Выбор метода определяется особенностями и характером данных, спецификой решаемой задачи и, наконец, предпочтениями пользователя. Рассмотрим методы визуализации, приняв за основу следующую классификацию (рисунок 4.5).

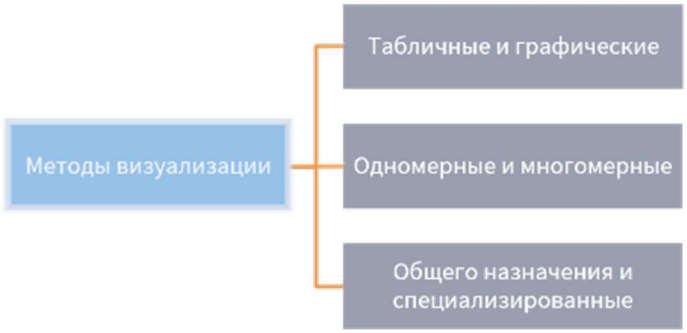


Рисунок 4.5 – Методы визуализации

Методы визуализации:

- **Табличные и графические**. Как правило, таблицы применяются в том случае, когда пользователю необходимо работать с отдельными значениями данных, вносить изменения, контролировать форматы данных, пропуски, противоречия и так далее. Графические методы позволяют лучше увидеть общий характер данных — закономерности, тенденции, периодические изменения. Кроме того, графические методы более эффективно сопоставляют данные: достаточно построить графики двух исследуемых процессов на одной системе координат, чтобы оценить степень их сходства и различия.
- **Одномерные и многомерные.** Одномерные визуализаторы представляют информацию только об одном измерении данных, в то время как многомерные — о двух или более. Если график показывает зависимость суммы продаж от даты, то он будет одномерным, поскольку на нем будет отображаться только одно измерение — Дата, значениям которого будет соответствовать факт Цена. Если же информация о продажах приводится по датам и наименованиям товаров, то появляется еще одно измерение — Товар, и тогда для корректного представления данных используется многомерный визуализатор. Популярные многомерные визуализаторы: OLAP-куб, тепловая карта и др.
- **Общего назначения и специализированные**. Методы визуализации общего назначения не связаны с каким-либо определенным

видом задач анализа или типом данных и могут использоваться на любом этапе аналитического процесса. Это своего рода типовые визуализаторы: графики и диаграммы, графы, гистограммы и их разновидности, статистические характеристики и др. В то же время существует ряд задач, специфика которых требует применения специализированных визуализаторов. Например, карты Кохонена специально разработаны для визуализации результатов кластеризации, матрицы классификации используются в основном для проверки состоятельности классификационных моделей, а с помощью диаграмм рассеяния оценивается корректность работы регрессионных моделей.

При изучении различных видов визуализации удобнее рассматривать их не по отдельности, а в контексте задач, для которых они наиболее часто применяются. Можно выделить следующие группы методов визуализации:

- **Общего назначения.** Применяются для решения типовых задач анализа данных: визуальной оценки качества и характера данных, распределения значений признаков, статистических характеристик и так далее. В них можно выделить два подвида — простые и сложные. К последним, в частности, относится OLAP-анализ — комплекс методов для визуализации многомерных данных.
- **Оценка качества моделей**. Позволяет оценивать различные характеристики моделей, такие как точность, эффективность, достоверность результатов, интерпретируемость, устойчивость и так далее.
- **Интерпретация результатов анализа.** Служат для представления конечных результатов анализа в виде, наиболее удобном с точки зрения их интерпретации пользователем.

Подсистемы визуализации данных содержатся не только в специализированных аналитических платформах, но и практически во всех программных средствах, которые связаны с обработкой данных, – от офисных приложений до систем компьютерной математики. Однако в аналитических платформах визуализации данных уделяется особое внимание, поскольку она является одной из составляющих аналитического процесса, без которой невозможно эффективно решать поставленные задачи.

Наилучших результатов можно добиться, если считать визуализацию не отдельной подсистемой, а такой же частью аналитического процесса, как, например, подготовка данных, аудит и профайлинг, моделирование. Даже если для построения качественной модели данных недостаточно, визуализация позволяет выдвигать гипотезы, делать выводы на основе экспертных оценок, разрабатывать способы повышения информативности данных.

4.2. Визуализаторы общего назначения: простые и сложные

Можно выделить набор средств визуализации, которые очень часто используются в бизнес-аналитике. Такие средства визуализации называются визуализаторами **общего назначения** [6]. Начнем с простых визуализаторов. К ним относятся следующие:

- таблицы;
- графики;
- диаграммы;
- гистограммы.

Кратко рассмотрим каждый из перечисленных визуализаторов [7].

Таблицы. С такой формой представления, как таблица, мы уже знакомы. Это простой и удобный способ отображения однотипной информации, кроме того, структурированные данные в первичных и вторичных источниках выгружаются именно в табличной форме. Например, в таблице справа приведена история продаж (рисунок 4.6).

Чек	Дата	Товар	Цена	Кол-во
1	01.05.2020	1235	258	1
1	01.05.2020	5648	325	2
2	01.05.2020	2485	924	1
3	01.05.2020	1878	157	1
3	01.05.2020	9759	654	1

Рисунок 4.6 – Таблицы

У таблицы как визуализатора существует множество вариантов представления. На рисунке изображена так называемая кросс-таблица с подсвечивающимися ячейками (англ.:highlight table). В ней каждому значению признака (продажи) соответствует один из оттенков в заранее выбранной цветовой гамме.

Такое представление помогает оперативно увидеть низкие и высокие зоны значений (продаж). Приставка кросс- означает, что таблица содержит по одному измерению в строках и столбцах, в данном случае это Месяц и Объект (рисунок 4.7).

Объект	Январь	Февраль	Март	Апрель	Май	Июнь	Июль	Август	Сентябрь	Октябрь	Ноябрь	Декабрь
Объект 1	146 753	141 794	140 665	123 484	74 282	73 461	68 610	64 914	94 294	118 492	124 655	131 911
Объект 2	528 413	521 445	347 460	565 013	416 415	313 500	392 340	388 313	414 870	500 303	526 193	580 080
Объект 3	679 508	639 218	630 953	576 158	440 803	380 115	508 695	516 188	534 945	658 290	718 545	707 025

Рисунок 4.7 – Таблицы

Графики. Графики представляют собой линии, отображающие зависимость между несколькими переменными в некоторой системе координат. Линия на графике состоит из множества точек, положение каждой из которых определяется значениями зависимой и независимой переменной (переменных).

Зачастую именно с построения графика и начинается работа с данными. С помощью графиков оценивается ряд характеристик данных, которые определяют их готовность к дальнейшей обработке.

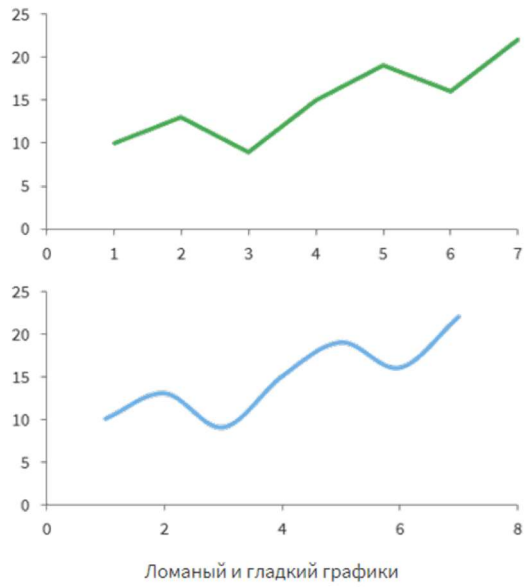
Особенно они полезны при анализе временных рядов. Иногда одного взгляда на график достаточно, чтобы выявить наличие тренда, сезонной компоненты, оценить степень влияния случайной составляющей на исследуемый процесс (рисунок 4.8).

Рисунок 4.8 – Графики

Чтобы построить график, достаточно задать таблично значения зависимой и независимой переменной, отметить соответствующие точки на координатной плоскости и соединить их линиями. Линии, соединяющие узлы графика, могут быть прямыми или сглаженными.

На рисунке представлены примеры ломаного (вверху) и сглаженного графика (внизу).

Во многих случаях гладкие графики удобнее для визуального восприятия и корректнее отображают реальные бизнес-процессы, которые также чаще всего изменяются плавно. Иногда точки, по которым строится график, вообще не соединяют, в этом случае график называется точечным (рисунок 4.9).



Ломаный и гладкий графики

Рисунок 4.9 – Графики

Если на графике требуется представить несколько рядов данных, то в одной системе координат строится несколько линий, но для этого необходимо, чтобы все отображаемые на графике ряды имели одинаковые единицы измерения и могли быть представлены в одном и том же масштабе.

Например, если нужно сравнить ежемесячные продажи за три года, то можно воспользоваться графиком на рисунке 4.10.

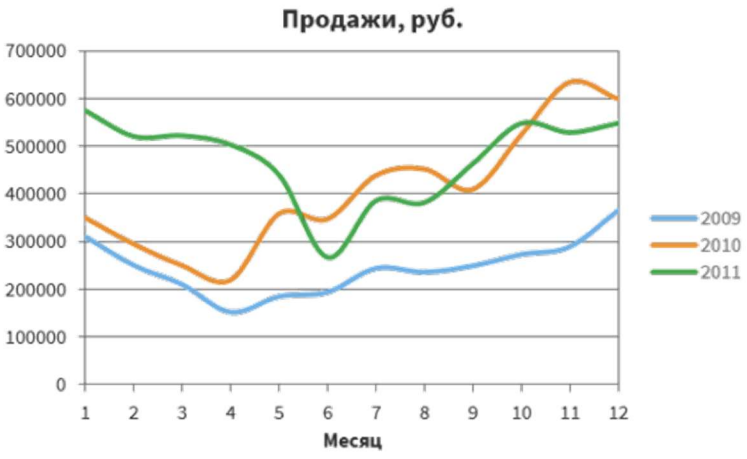


График ежемесячных продаж за три года

Рисунок 4.10 – Графики

Если значения в рядах данных лежат в различных диапазонах и отличаются на несколько порядков, то отображение этих рядов на одном графике может вызвать определенные затруднения.

Допустим, нужно сравнить динамику продаж нескольких товаров, но цены на различные товары даже в одной группе могут различаться в десятки и сотни раз.

Так, дорогие агрегаты для автомобиля (коробка передач, двигатель и др.) продаются **единично**, а мелкие детали ценой в несколько рублей (гайки, болты и др.) – **сотнями**.

Диаграммы. С помощью графика удобнее всего отображать непрерывные (числовые) величины, поскольку можно получить достаточное число точек, чтобы его построить.

Если же речь идет о категориальных (дискретных) значениях, то более подходящим средством визуализации является диаграмма. Принципиального различия между понятиями график и диаграмма нет.

Просто под графиком традиционно понимают представление зависимостей в виде линий, тогда как в диаграмме значения отображаются с помощью самых разнообразных объектов и фигур.

Как правило, в диаграммах, по горизонтальной оси X откладываются категории, а по вертикальной оси У — значения.

Самые простые и часто используемые диаграммы — столбчатые. В них значение каждой категории представляется в виде столбика, высота которого пропорциональна соответствующему значению (1).

Разновидностью столбчатой диаграммы является линейчатая диаграмма, которая отличается от столбчатой тем, что ось категорий откладывается вертикально, а ось значений — горизонтально (2) (рисунок 4.11).

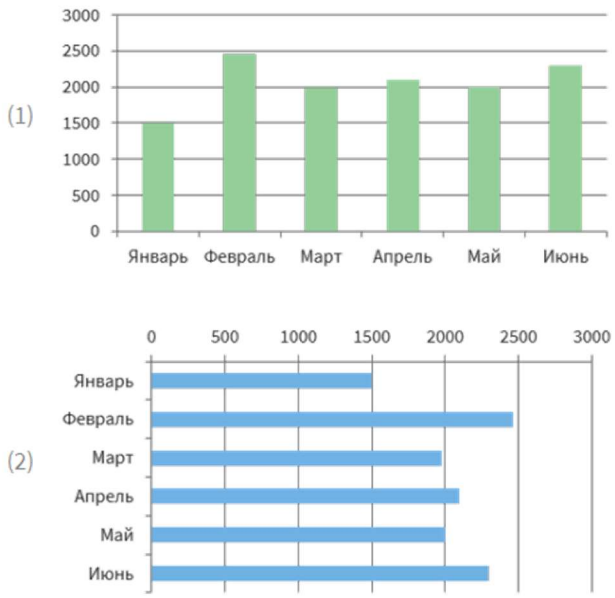


Рисунок 4.11 – Диаграммы

Еще одним распространенным видом диаграмм является круговая диаграмма. Ее очень удобно использовать, если нужно показать долю, которую вносит то или иное значение в общий результат. Эта доля может быть выражена как в абсолютных единицах, так и в процентах (например, процент от выручки, который обеспечил товар А). Такие примеры круговых диаграмм, показывающие распределение структуры месячных затрат на виды мобильной связи, представлены на рисунках 4.12.

Рисунок 4.12 – Круговые диаграммы

Гистограммы. Гистограмма показывает распределение набора данных внутри выборки (например, количество заемщиков банка в нескольких возрастных группах) в виде столбиков.

Гистограммы широко используются в статистике для определения наиболее вероятных значений, которые может приобретать некоторая величина, а также для выявления законов распределения, которым подчиняется случайная величина.

Гистограмма строится следующим образом. Пусть исследуемой величиной являются ежедневные продажи торговой точки в течение месяца. При этом минимальное наблюдаемое значение составило 10 тыс., а максимальное – 100 тыс. (рисунок 4.13).

Дата	Сумма
01.05.2020	10 000
02.05.2020	33 000
03.05.2020	51 000
04.05.2020	25 000
05.05.2020	48 000
06.05.2020	62 000
07.05.2020	73 000
08.05.2020	100 000
09.05.2020	77 000
…	…

Рисунок 4.13 – Гистограммы

Разобьем диапазон изменения величины на 9 поддиапазонов по 10 тыс. и подсчитаем, сколько раз значение продаж попадает в тот или иной поддиапазон. Результаты сведем в таблицу. Как видим, продажи на сумму от 10 до 20 тыс. наблюдались только один раз, от 30 до 40 тыс. – два раза итак далее. На основе полученной таблицы строим гистограмму (рисунок 4.14).

Диапазон в тыс.	10 — 20	20 — 30	30 — 40	40 — 50	50 — 60	60 — 70	70 — 80	80 — 90	90— 100
Частота	1	1	2	6	10	6	3	1	1

Рисунок 4.14 – Гистограммы

По горизонтальной оси гистограммы откладываются значения продаж, а по вертикальной — количество или частота наблюдений, значения которых

попали в заданный диапазон (поэтому иногда гистограмму называют частотным полигоном).

Гистограмма на рисунке показывает, что наибольшее число наблюдений попало в диапазон 50-60 тыс. Таким образом, значения из данного диапазона можно рассматривать как наиболее вероятные. Эту информацию используют для восстановления пропущенных значений при очистке данных, для планирования денежных поступлений, закупок и так далее.

Что касается крайних элементов гистограммы, то они представляют редкие события — экстремально высокие или экстремально низкие значения продаж. Видно, что в диапазон 10-20 тыс. попало всего одно значение, следовательно, вероятность такого события мала, и его не стоит включать в рассмотрение. Экстремально низкие продажи могут быть вызваны исключительной ситуацией, например, погодными условиями, отключением электроэнергии и т. п. (рисунок 4.15).

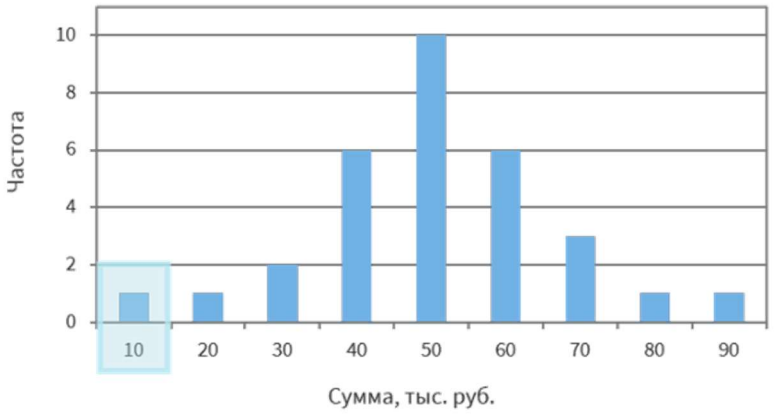


Рисунок 4.15 – Диаграмма

Иногда используют нормированную гистограмму, что позволяет оперировать не значениями наблюдений, а их вероятностями. Для этого каждый элемент гистограммы делится на количество наблюдений, то есть в нашем случае на 31 (число дней в месяце). Обратимся к рисунку.

Теперь высота столбца определяется не количеством наблюдений, попавших в соответствующий диапазон, а вероятностью попадания в него.

На рисунке видно, что вероятность попадания значения в диапазон 50-60 тыс. составляет примерно 0,32, или 32%.

Соответственно, вероятность появления значений в диапазоне 10-20 тыс. не превышает 0,03, или 3 %.

В нормированной гистограмме сумма значений всех ее элементов должна быть равна 1, поскольку сумма вероятностей всех возможных событий (попадания значения в какой-либо диапазон) есть 1.

Обычно при построении гистограммы аналитик имеет возможность задать число поддиапазонов, на которое будет разбиваться исходный диапазон изменения величины (фактически это число столбцов гистограммы). Здесь существуют различные рекомендации, например: число поддиапазонов не должно быть меньше, чем $\log_2 N$, где N — число наблюдений.

На практике можно руководствоваться следующим эмпирическим правилом. Количество столбцов в гистограмме должно быть таким, чтобы в ней не образовывались провалы, резкие выбросы или множественные пики. Она должна быть достаточно гладкой, чтобы по ней можно было определить характер распределения наблюдаемой величины. Часто хороших результатов удается добиться при использовании 10-15 столбцов.

Далее рассмотрим сложные визуализаторы общего назначения. Обычно **сложные визуализаторы общего назначения** – это многомерные визуализаторы, которые позволяют представить информацию из нескольких измерений за счет использования различных цветов, форм, размеров и расположения объектов анализа. В последнее десятилетие получили широкое распространение следующие визуализаторы (рисунок 4.16). Рассмотрение их начнем с OLAP-анализа.

Рисунок 4.16 – Сложные визуализаторы общего назначения

Большинство реальных бизнес-процессов являются сложными, поскольку в них участвует много объектов, которые находятся в самых разнообразных отношениях, и с каждым из которых может быть связано несколько числовых характеристик.

Процесс анализа данных требует сравнения и сопоставления этих характеристик (например, продаж по различным городам или группам товаров), вычисления дополнительных показателей и характеристик и так далее.

Поэтому при визуализации данных часто встает вопрос: как представить сложные данные в таком виде, чтобы человек мог их осмыслить и интерпретировать?

Предположим, требуется визуализировать с помощью таблицы продажи нескольких наименований товаров по нескольким датам и городам (рисунок 4.17).

Даже такой простой фрагмент демонстрирует сложности при визуализации бизнес-процессов. Если продажи велись одновременно по множеству наименований товаров в нескольких городах, то таблица, содержащая соответствующие наблюдения, неизбежно окажется избыточной: в ней много раз будут повторяться одни и те же данные. А оперативно получить нужную информацию из такой таблицы будет весьма проблематично из-за огромного количества наблюдений.

Дата	Товар	Город	Цена, руб.	Количество	Сумма, руб.
01.02.15	Пылесос	Михайлов	6 500	10	65 000
01.02.15	Ст. машина	Михайлов	16 700	5	83 500
01.02.15	Утюг	Михайлов	4 200	3	12 600
01.02.15	Пылесос	Касимов	7 500	2	15 000
01.02.15	Ст. машина	Касимов	16 700	4	66 800
01.02.15	Пылесос	Касимов	6 500	5	32 500
02.02.15	Пылесос	Михайлов	7 500	4	30 000
02.02.15	Утюг	Михайлов	4 200	10	42 000
02.02.15	Телевизор	Михайлов	78 000	2	156 000

Рисунок 4.17 – Проблема визуализации сложных данных

Возможным выходом из ситуации является декомпозиция сложной таблицы на множество более простых, где содержится информация о продажах отдельных товаров и групп товаров по городам, интервалам дат, поставщикам и так далее, но такой подход позволяет лишь отчасти решить проблему.

Все эти трудности, возникающие при обработке больших и сложных массивов данных, создали предпосылки для появления метода визуализации многомерных табличных данных — OLAP-анализа.

В основе OLAP лежит многомерное представление данных, которые могут быть разделены на количественные и качественные. Принцип многомерного хранения данных подробно рассматривается в курсах, посвященным технологиям хранилищ и витрин данных.

Качественные данные (Измерения). Качественные данные представляют собой значения, выраженные в категориальной форме. Обычно

это наименования товаров, групп товаров, организаций, названия городов, ФИО сотрудников и так далее.

В рамках многомерной модели данные, качественно описывающие исследуемый бизнес-процесс, называются измерениями (или группами).

Измерениями могут быть: Товар, Город, Клиент, Организация, Дата и др.

Количественные данные (Факты). С каждым объектом связаны признаки, количественно описывающие его. Для товара это может быть цена, количество или сумма; для города, в котором расположено торговое представительство, — расстояние до него и количество жителей; для сотрудника — заработная плата и стаж работы.

Данные, количественно описывающие процесс или объект, называются фактами (или показателями).

Примеры фактов: Количество, Сумма, Возраст, Доход, Торговая наценка и др.

С каждым качественным значением анализируемого бизнес-процесса связаны один или несколько количественных показателей. Другими словами, с каждым измерением связаны один или несколько фактов (рисунок 4.18).

Измерения несут смысловую нагрузку, а факты — количественную. Чтобы достоверно отделить измерения от фактов, достаточно сопоставить значение с вопросом.

Измерения позволяют ответить на вопросы: Что? (товар), Кто? (клиент), Когда? (дата) и Где? (город).

Факты отвечают на единственный вопрос: Сколько?

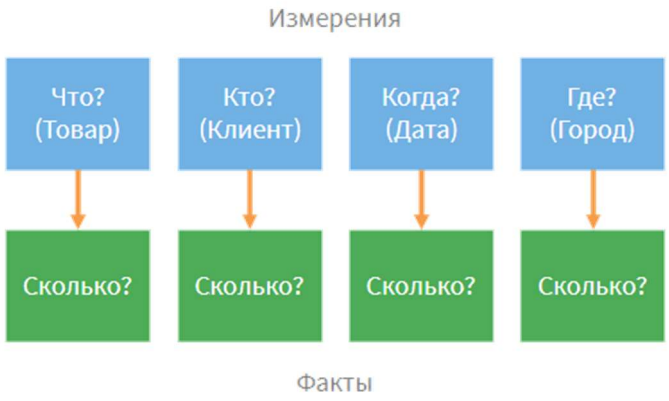


Рисунок 4.18 – OLAP-анализ

Чтобы построить OLAP-куб, пользователь должен указать системе следующие параметры:

- какие измерения и факты включать в куб;

- варианты агрегации значений фактов.

Визуализация OLAP -куба проводится с помощью специального вида таблиц, которые строятся на основе срезов OLAP-куба, содержащих необходимую пользователю информацию. Срезы, в свою очередь, являются результатом выполнения соответствующего запроса к источнику данных.

Как правило, в процессе построения срезов пользователь с помощью мыши и клавиатуры манипулирует заголовками измерений, добиваясь наиболее информативного представления данных в кубе. В зависимости от положения заголовков измерений в таблице автоматически формируется запрос к базе или хранилищу данных. Запрос извлекает данные из базы или хранилища, после чего OLAP -ядро системы визуализирует их.

На рисунке показан пример визуализации данных с помощью OLAP-куба, где представлена информация о количестве и суммах продаж по аптекам в разрезе года и месяца. Эти сведения могут оказаться полезными при определении наиболее загруженных аптек. Так, в кубе для каждого месяца и года перечислены все продажи, которые были сделаны в пределах этого периода, с указанием количества проданных единиц товара и их общей суммы. Визуальный анализ позволяет увидеть, что больше всего продаж пришлось на Аптека 1.

Для более детального анализа представляют интерес продажи по дням. Чтобы преобразовать куб в соответствующий вид, достаточно добавить еще одно измерение День месяца, как показано на рисунке 4.19.

Отдел.Наименование | Факты

Дата (Год + Месяц) | Дата (Год + День) | Группа.Наименование

		Аптека 1			Аптека 2			Итого:		
		Сумма	Количество	Цена	Сумма	Количество	Цена	Сумма	Количество	Цена
01.01.2017		9 942,17	202	49,22				9 942,17	202	49,22
01.02.2017		33 809,16	623	54,27				33 809,16	623	54,27
01.03.2017		32 241,17	534	60,38				32 241,17	534	60,38
01.04.2017	01.04.2017	1 021,61	18	56,76	523,93	8	65,49	1 545,54	26	59,44
	02.04.2017	1 680,54	27	62,24	847,26	6	141,21	2 527,80	33	76,60
	03.04.2017	895,69	24	37,32	1 424,13	18	79,12	2 319,82	42	55,23
	04.04.2017	1 028,58	14	73,47	741,55	8	92,69	1 770,13	22	80,46
	05.04.2017	2 498,81	30	83,29	582,39	10	58,24	3 081,20	40	77,03
	06.04.2017	2 195,76	22	99,81	1 202,16	23	52,27	3 397,92	45	75,51
	07.04.2017	679,92	10	67,99	497,48	10	49,75	1 177,40	20	58,87
	08.04.2017	579,44	15	38,63	1 331,57	15	88,77	1 911,01	30	63,70
	09.04.2017	1 708,28	28	61,01	936,47	11	85,13	2 644,75	39	67,81
	10.04.2017	1 074,86	18	59,71	1 221,23	13	93,94	2 296,09	31	74,07
	Итого:	13 363,49	206	64,87	9 308,17	122	76,30	22 671,66	328	69,12
01.05.2017		22 377,64	449	49,84	10 759,29	224	48,03	33 136,93	673	49,24
01.06.2017		21 364,05	425	50,27	8 160,47	186	43,87	29 524,52	611	48,32
01.07.2017		13 536,43	373	36,29	8 158,10	164	49,74	21 694,53	537	40,40
01.08.2017		14 324,56	312	45,91	10 764,90	227	47,42	25 089,46	539	46,55
01.09.2017		23 436,44	453	51,74	15 008,15	278	53,99	38 444,59	731	52,59
01.10.2017		31 328,32	536	58,45	21 777,81	361	60,33	53 106,13	897	59,20
01.11.2017		33 413,86	588	56,83	16 416,76	281	58,42	49 830,62	869	57,34
01.12.2017		32 596,49	591	55,15	21 365,74	350	61,04	53 962,23	941	57,35
Итого:		281 733,78	5 292	53,24	121 719,39	2 193	55,50	403 453,17	7 485	53,90

Рисунок 4.19 – Построение OLAP-куб

Таким образом, OLAP-куб можно использовать не только как метод визуализации, но и как средство оперативного формирования отчетов и представления информации в нужном разрезе (так называемая аналитическая отчетность).

Наибольший интерес OLAP-куб представляет с точки зрения визуального анализа данных, поиска особенностей и закономерностей в данных. При этом существуют два подхода.

Аналитик может задаться несколькими вопросами, например:

- Какой товар самый популярный?
- Где продажи были наилучшими и с чем это связано? и так далее.

Затем с помощью манипуляций заголовками измерений выбирается такое представление куба, которое позволяет ответить на поставленные вопросы.

Аналитик последовательно перебирает возможные варианты представления данных, которые обеспечивает куб, и сих помощью сопоставляет данные, выявляет закономерности и связи между элементами данных, выдвигает гипотезы и так далее.

Умелое использование OLAP-анализа и работа с кубом порой позволяют получить результаты даже в тех случаях, когда методы Data Mining оказываются малоэффективными (например, из-за недостатка или низкого качества данных).

Для того чтобы сделать процесс извлечения информации из куба более гибким и эффективным, в OLAP-системах обычно предусматривается набор операций с измерениями, которые позволяют получить максимум возможных представлений данных. К таким операциям относят следующие:

- **Замена и добавление изменений**. Любые измерения куба можно заменить другими, которые присутствуют в исходном множестве данных. Например, если ранее нас интересовала информация о продажах в разрезе каждого отдельного клиента, а позже — в разрезе каждого города, где проводились продажи, то можно заменить измерение Клиент измерением Город. При необходимости в куб могут встраиваться новые измерения.
- **Удаление измерений**. Если в процессе работы с кубом какое-либо измерение утратило информативность, то его можно исключить из куба. Оставлять в кубе лишние измерения нежелательно: чем выше размерность куба, тем сложнее для понимания и осмысления представленная в нем информация. Кроме того, чем больше измерений в кубе, тем выше временные и вычислительные затраты на его обработку.
- **Скрытие измерений**. Пользователь может выбрать в кубе несколько наиболее информативных представлений и работать с ними, но

часто возникает ситуация, когда какое-либо измерение для одного из представлений является лишним, то есть не несет полезной информации, а только усложняет визуальное восприятие, в то время как для других представлений оно нужно. Временно удалять, а затем встраивать это измерение в куб непрактично, поскольку перестройка куба при большом объеме данных может занять много времени. Поэтому предусматривается возможность временного скрытия измерений без удаления их из куба. Тогда при необходимости можно вновь отобразить скрытое измерение без перестройки всего куба.

- **Измерение порядка следования измерений**. При работе с несколькими измерениями в кубе закладывается возможность выбрать порядок их отображения.
- **Отбор значений измерений**. Часто необходимость в отображении всех возможных значений измерения (например, дат, товаров ит.д.) отсутствует. Поэтому, чтобы не загромождать куб ненужными данными, лишние значения измерений могут быть временно скрыты, а если понадобится — отображены снова. При этом выбирать отображаемые значения измерений можно или непосредственного из списка, или с помощью фильтрации по какому-либо условию.
- **Транспонирование**. Если при работе с кубом выяснится, что значения измерений, которые отображаются в столбцах, удобнее отображать в строках, или наоборот, то соответствующее преобразование можно легко произвести с помощью операции транспонирования. Пример транспонирования представлен на рисунке: вверху расположено исходное представление куба, а внизу — результат применения к нему операции транспонирования (рисунок 4.20).

Отдел	Месяц					
	10 октября		10 ноября		10 декабря	
	Кол-во	Сумма	Кол-во	Сумма	Кол-во	Сумма
Аптека 1	187	11714,1	186	10308,2	192	9856,4
Аптека 2	141	8734,1	81	4984,4	116	7360,3
Аптека 3	137	10490,6	204	14347,7	216	12580,3
Итого:	465	30938,9	471	29640,2	524	29797

Месяц	Отдел					
	Аптека 1		Аптека 2		Аптека 3	
	Кол-во	Сумма	Кол-во	Сумма	Кол-во	Сумма
10 октября	187,0	11714,1	141,0	8734,1	137,0	10490,6
10 ноября	186,0	10308,2	81,0	4984,4	204,0	14347,7
10 декабря	192,0	9856,4	116,0	7360,3	216,0	12580,3
Итого:	565,0	31878,7	338,0	21078,8	557,0	37418,6

Рисунок 4.20 – Транспонирование

При работе с OLAP-кубами широко применяется еще одна операция, называемая детализацией (англ.: drill down — проникновение, более детальное исследование).

Ее необходимость вызвана тем, что в большинстве случаев значения в кубе являются агрегированными, например, в пределах некоторой даты или интервала дат. В то же время пользователя могут интересовать и атомарные (то есть детализированные) значения, на основе которых были получены агрегированные.

Операция детализации заключается в отображении набора записей выборки данных, в результате агрегирования которых было получено соответствующее значение OLAP-куба.

На рисунке 4.21 представлен фрагмент выборки данных по кредитованию.

Код клиента	Дата	Сумма кредита	Цель кредитования
120	08.01.2015	250 000	Иное
121	08.01.2015	190 000	Покупка авто
122	09.01.2015	63 000	Оплата образования
123	09.01.2015	22 500	Оплата услуг
124	09.01.2015	48 500	Покупка товара
125	09.01.2015	45 500	Оплата образования
126	09.01.2015	32 000	Оплата услуг
127	09.01.2015	82 500	Оплата услуг
128	09.01.2015	46 500	Оплата услуг
129	09.01.2015	13 500	Оплата услуг
130	09.01.2015	31 500	Оплата образования
131	09.01.2015	56 000	Иное
132	09.01.2015	128 000	Покупка авто
133	09.01.2015	13 500	Покупка товара
134	09.01.2015	49 500	Оплата образования
135	10.01.2015	18 000	Покупка товара
136	10.01.2015	55 000	Турпоездка

Рисунок 4.21 – Детализация

На рисунке 4.22 представлен фрагмент куба. Данные о кредитах агрегированы по датам: в кубе мы видим не суммы отдельных кредитов, а значения, полученные их суммированием по отдельным целям кредитования в пределах одной даты. Другими словами, из куба можно почерпнуть только информацию о том, на какую общую сумму было выдано кредитов на ту или иную цель в пределах определенной даты, а суммы, выданные отдельным клиентам, остаются неизвестными.

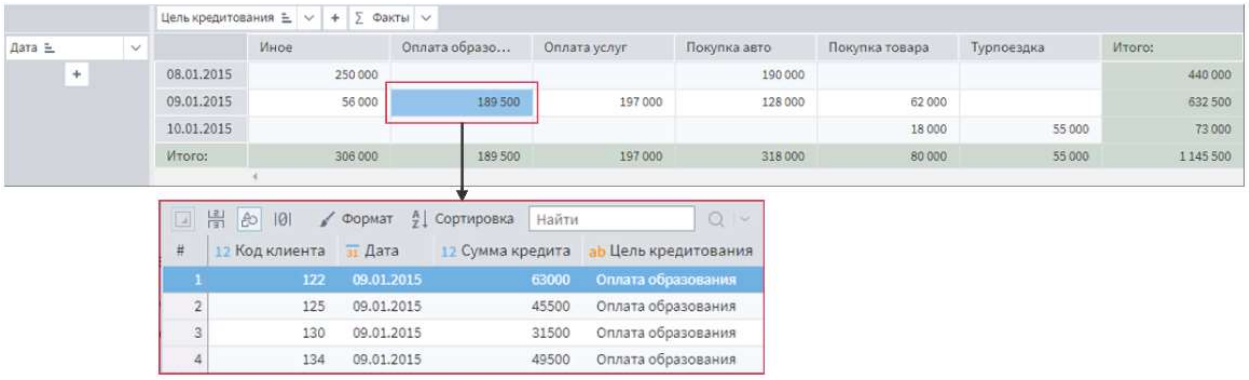


Рисунок 4.22 – Детализация

Предположим, аналитика заинтересовали суммы кредитов, выданных 09.01.2015 на оплату образования, которые составили 189 500 руб. Для более

детального исследования может понадобиться информация о сумме каждого кредита в отдельности. Получить такую информацию поможет операция детализации.

Географические и тепловые карты. Карты (англ.: maps) — позволяют наглядно представить данные, связанные с географическим расположением исследуемых объектов и процессов. В бизнес-аналитике это информация, связанная с уровнем потребления в регионах, характером спроса и предложения по различным видам товаров, сведения о продажах, осуществляемых региональными дилерами, о логистических и транспортных потоках и так далее.

Рассмотрим рисунок 4.23. В таблице содержатся данные об отгрузках продукции, произведенных компанией мелким оптовикам из различных субъектов Центрального федерального округа (ЦФО).

№ п/п	Область	Продажи, млн. руб.	№ п/п	Область	Продажи, млн. руб.
1	Белгородская	162	10	г. Москва	664
2	Брянская	61	11	Московская	328
3	Владимирская	54	12	Орловская	54
4	Воронежская	154	13	Рязанская	151
5	Ивановская	209	14	Смоленская	137
6	Калужская	117	15	Тамбовская	29
7	Костромская	90	16	Тверская	178
8	Курская	57	17	Тульская	186
9	Липецкая	50	18	Ярославская	120

Рисунок 4.23 – Географические карты

Имея карту ЦФО, можно в соответствии с палитрой раскрасить области и получить сравнительный отчет о продажах по регионам.

Видно, что максимальные значения продаж приходятся на регионы, обозначенные 10 и 11 (Москва и Московская область), а минимальные — на 15 (Тамбовская область). При этом может использоваться многомерное представление данных (рисунок 4.24).

Например, если требуется показать на карте не только суммы продаж, но и прибыль, то последнюю можно отобразить рельефным выделением соответствующей области на карте. Также на карты можно наносить значки, форма, цвет, размер и взаимное положение которых соответствуют свойствам исследуемых объектов.

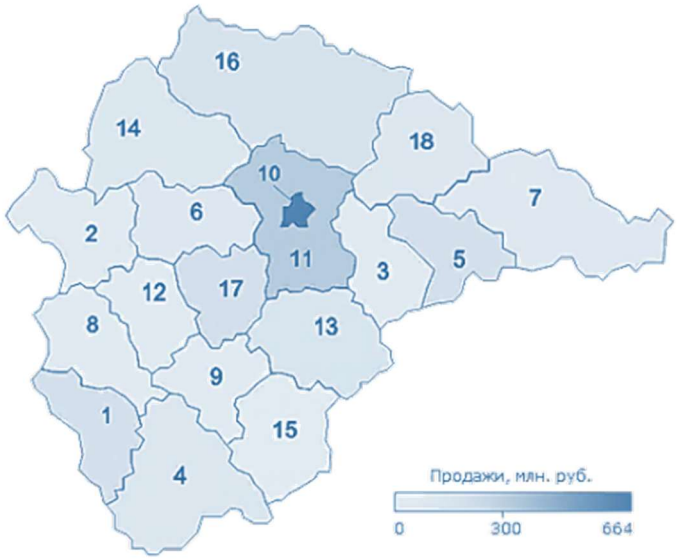


Рисунок 4.24 – Географические карты

Карты не обязательно должны быть связаны с географией. В бизнес-аналитике почти всегда приходится иметь дело с объектами, которые описываются двумя признаками и более. То есть выборки являются многомерными, и представление подобных данных на плоских визуализаторах (графиках, диаграммах) не всегда удобно и корректно отображает результаты.

В данном случае имеет смысл использовать двумерные тепловые карты (англ.: heat maps), где каждому значению признака соответствует один из оттенков в заранее выбранной цветовой гамме.

На рисунке 4.25 можно наблюдать пример тепловой карты, построенной по двум измерениям: Топливо и Город. Цвет каждой прямоугольной ячейки карты формируется на основе цены топлива: чем темнее оттенок, тем она выше.

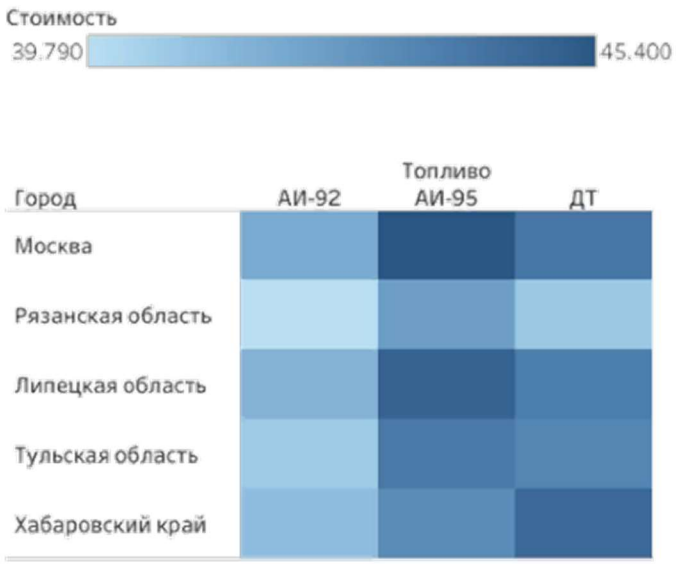


Рисунок 4.25 – Тепловые карты

Плоское дерево. К семейству карт можно отнести и метод плоское дерево (англ.: treemapping).

Он является очень эффективным при изображении численных атрибутов элементов (размер, стоимость, значение), организованных в большие иерархии.

На рисунке 4.26 за размер прямоугольников отвечает Среднегодовая численность занятых в экономике, тыс. чел., а за цвет — численность населения, тыс. чел. по субъектам и федеральным округам РФ.

Таким образом, близкие по значению этого атрибута регионы располагаются на плоском дереве рядом.

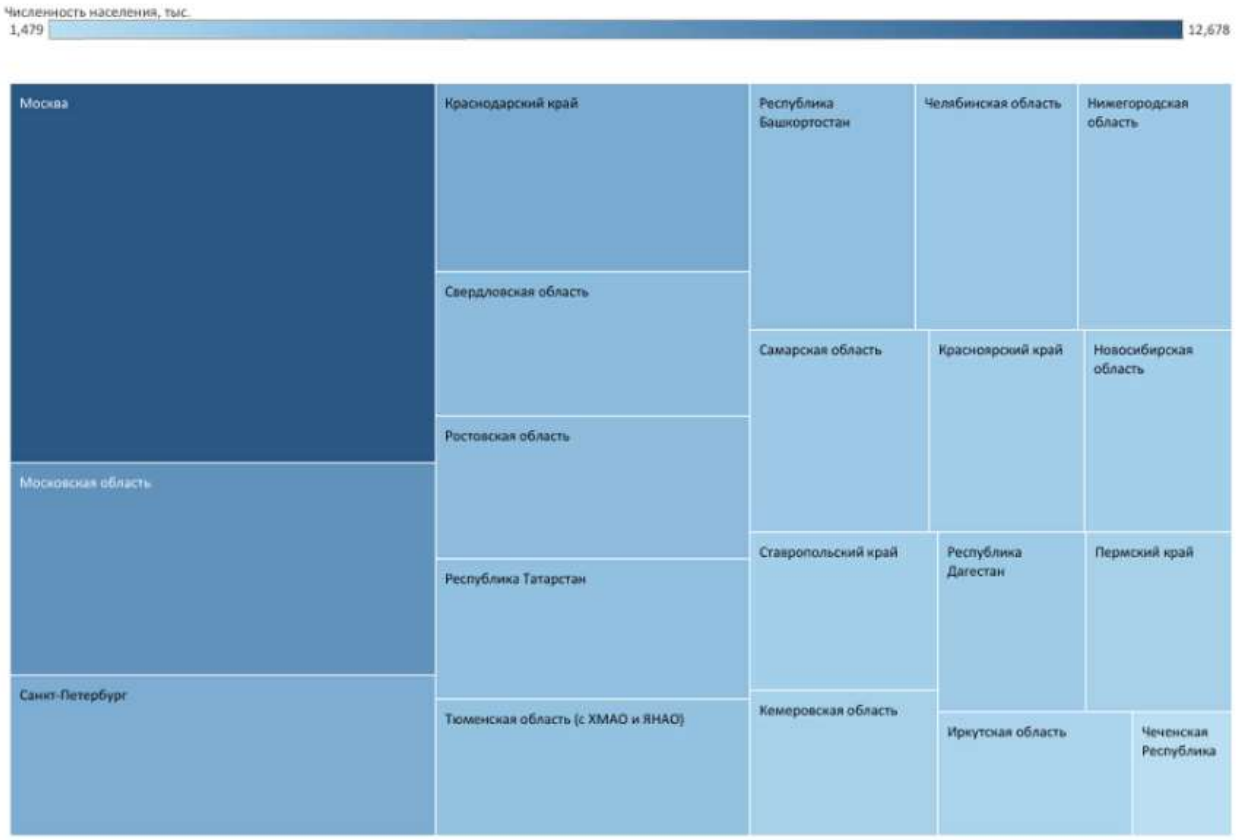


Рисунок 4.26 – Плоское дерево

Диаграмма связей. Нередко требуется исследовать характер и степень взаимной зависимости между различными объектами. Для анализа можно использовать визуализацию связей, когда объекты представляются в виде значков, а связи между ними — в виде линий, соединяющих соответствующие значки. При этом сила связи, то есть степень взаимной зависимости объектов, может показываться различными способами.

Чаще всего для этого используют:

- толщину линии: чем сильнее связь между объектами, тем толще соединяющая их линия (1);
- цвет линии, при этом выбираются оттенки определенного спектра (2).

Допускается одновременно использовать и толщину, и цвет, но такие диаграммы более сложны для восприятия. Кроме того, можно по-разному располагать объекты на плоскости.

Звезда. В первом случае показываются связи между одним выбранным объектом, который помещен в центре, и всеми остальными (рисунок 4.27).

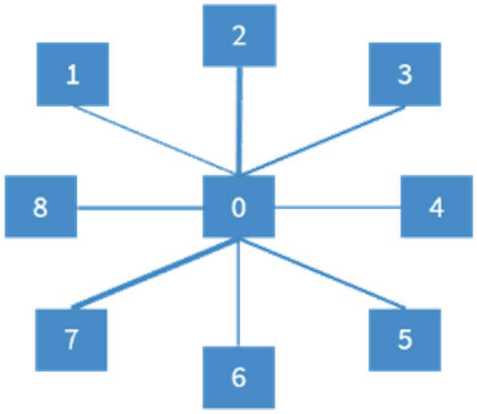


Рисунок 4.27 – Звезда

Сеть. Во втором случае визуализируются все попарные связи между объектами (рисунок 4.28).

Если число объектов велико, связей становится еще больше, поэтому анализируют только наиболее сильные или наиболее слабые, накладывая фильтр на значения силы связи.

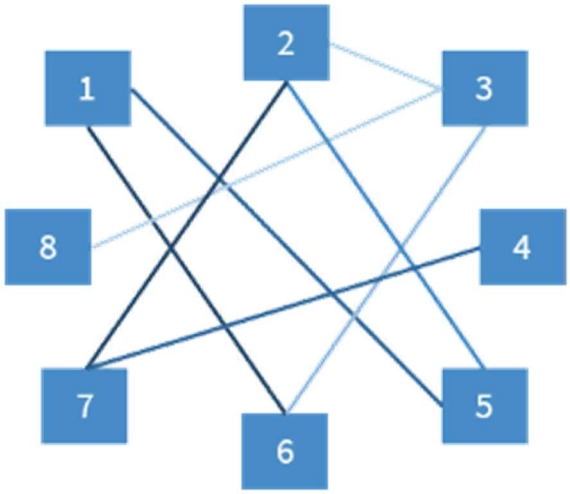


Рисунок 4.28 – Сеть

Облако данных. Облако данных (англ.: Data Cloud) — это визуализатор, в котором используется другой цвет и/или размер шрифта для обозначения числовых данных.

Родителем этого визуализатора можно считать облако тегов (англ.: Tag cloud), в котором сравниваются ключевые слова или фразы (значения), которые содержатся внутри фрагмента текста (набора данных), при этом каждому из них задается свой размер шрифта.

На рисунке 4.29 приведен пример облака данных регионов РФ, в котором за цвет (в заранее выбранной шкале) отвечает число рожденных на 1000 чел., %, а за размер шрифта — численность населения.

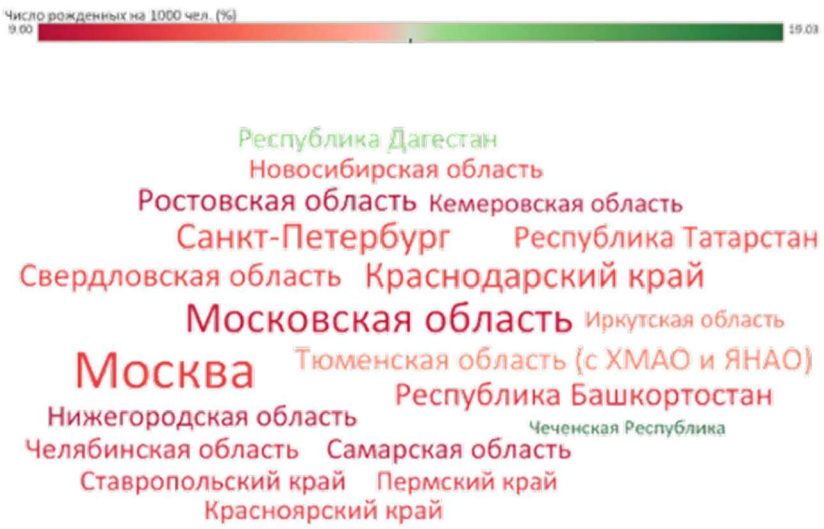


Рисунок 4.29 – Облако данных

Пузырьковая диаграмма. Пузырьковая диаграмма (англ.: Bubble Chart) — это смесь графика и диаграммы, когда по двум осям расставлен набор точек, соответствующий значениям. При этом сами точки не соединены и имеют различную величину и/или цвет, которые задаются дополнительными показателями.

На рисунке 4.30 изображен пример такой диаграммы, на которой за размер точки отвечает показатель Численность населения занятых в экономике (тыс. чел.).

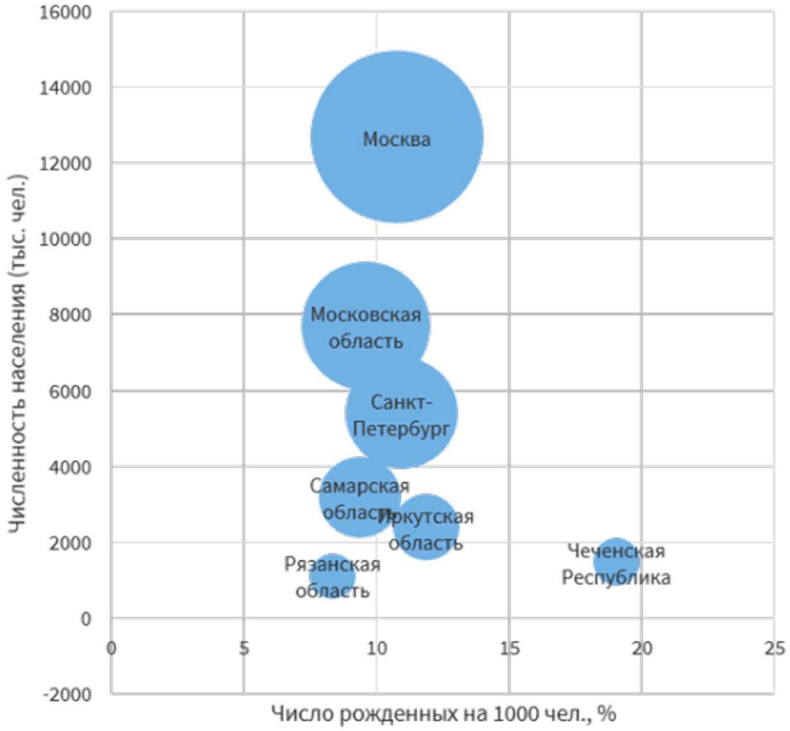


Рисунок 4.30 – Пузырьковая диаграмма

Диаграмма рассеяния. Диаграмма рассеяния (англ.: Scatterplot) показывает распределение ограниченного набора точек, которые соответствуют значениям по осям (рисунок 4.31).

Например, связь между двумя показателями по всем регионам РФ: Число рожденных на 1000 человек, % и Численность населения (тыс. чел.). Видно, что высокая рождаемость более характерна для субъектов с небольшой численностью населения.

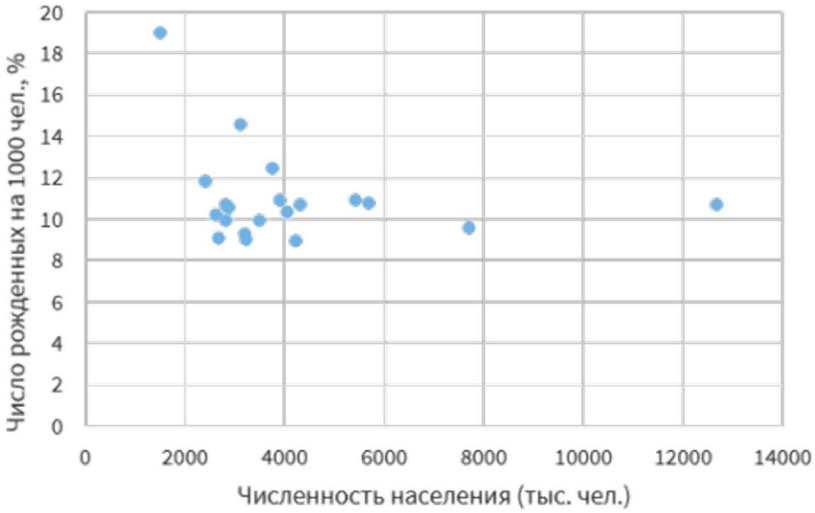


Рисунок 4.31 – Диаграмма рассеяния

4.3. Лабораторная работа № 17 «Визуализация данных»

Импортируем набор данных **farma_sales.lgd**. Он включает в себя информацию по продажам лекарственных препаратов за некоторый период. Для того чтобы исследовать входной набор, воспользуемся визуализатором.

Добавим визуализатор **Статистика**.

С его помощью можно просмотреть различные статистические показатели по каждому полю набора данных. В верхней части окна визуализатора отображается общее количество записей в наборе.

В столбце **Уникальные** мы можем видеть, что в нашем наборе: присутствуют продажи по 3 отделам, всего в продажах участвовало 270 товаров из 8 групп.

Для показателя **Сумма** можно посмотреть, на какие минимальную, максимальную и среднюю суммы произошли продажи.

По умолчанию отображается 8 показателей: гистограмма значений поля, диаграмма размаха, минимальное, максимальное и среднее значения, стандартное отклонение, количество пропусков и уникальных значений.

Гистограмма – наиболее универсальный показатель. Она отображает распределение значений по некоторым интервалам для полей непрерывного вида и распределение по уникальным значениям для дискретного вида данных

При наведении курсора на столбец гистограммы можно увидеть количество значений, соответствующих данному интервалу или уникальному значению. Например, по полю **Категория ABC** мы можем сказать, что товары из группы **A** были проданы 3480 раз.

Гистограмма не отображается, если в поле большое количество уникальных значений, как в поле **Товар.Наименование**, где их 270. В этом случае мы можем рассмотреть ее более детально, нажав кнопку **Гистограмма** в правом верхнем углу. Обратите внимание, что уникальные значения рассчитываются только для полей с дискретным видом данных. Здесь отображается уже полный список интервалов/уникальных значений с количеством и процентом значений, относящихся к каждому интервалу.

Для полей непрерывного вида также отображаются минимум и максимум – границы первого и последнего интервалов, – и число интервалов. Все эти значения можно задать вручную, и распределение значений по интервалам изменится в соответствии с новыми данными.

В целом доступность показателя для определенного поля зависит от типа или вида данных этого поля. Например, **Диаграмма размаха** доступна только для полей непрерывного вида.

Нажмем кнопку **Настройка показателей** на панели инструментов визуализатора (рисунок 4.32).

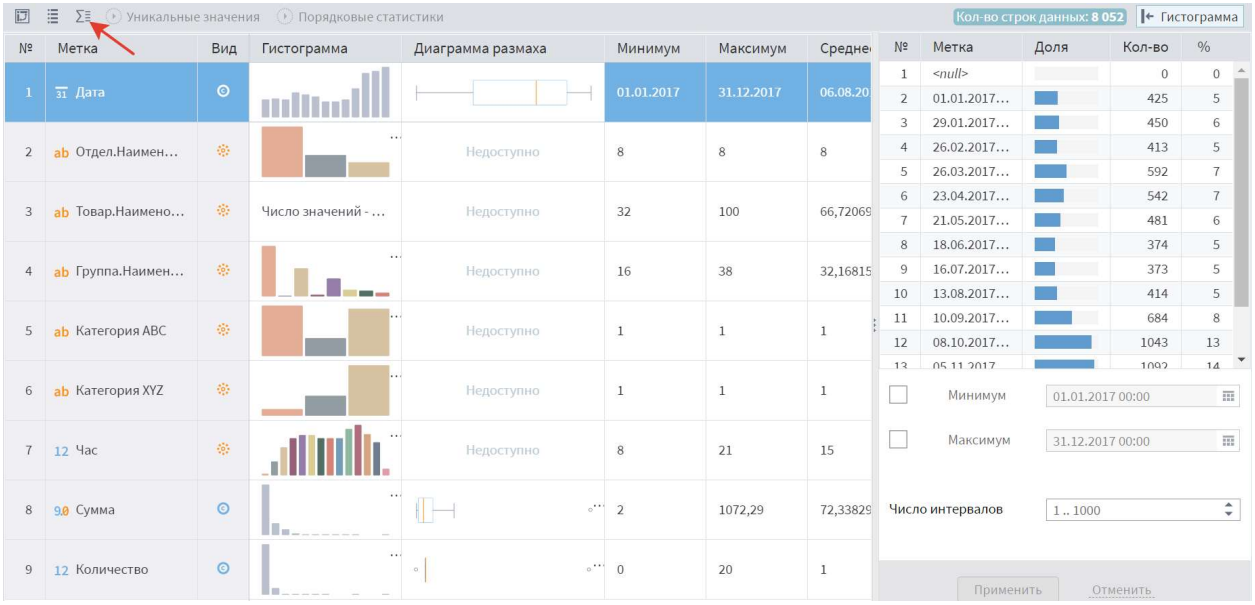


Рисунок 4.32 – Настройка показателей на панели инструментов визуализатора

Перед нами полный список доступных статистических показателей. Большинство из них рассчитываются для полей вещественного, числового типов, а также типа дата/время.

Оставим только показатели **Гистограмма**, **Минимум**, **Максимум**, **Среднее** и **Уникальные** и применим изменения.

Кнопка **Настройка полей** позволяет отключить отображение полей, которые нас не интересуют. Теперь отображаются только выбранные нами показатели.

Кроме того, есть возможность поменять местами поля и показатели. Для этого нажмем кнопку **Транспонировать**. Теперь наименования полей отображаются в столбцах, а показатели – в строках. Таким образом, визуализатор статистика позволяет быстро исследовать набор данных и изучить его основные характеристики.

При большом наборе данных расчет статистик может занимать продолжительное время. В этом случае по умолчанию при открытии визуализатора по ряду показателей значения полей будут пусты. Нужно выбрать, что вы хотите отобразить: уникальные значения, либо порядковые статистики, – и нажать соответствующую кнопку на панели инструментов или команду контекстного меню.

Не забывайте: все изменения, которые сделаны в визуализаторе, влияют только на **отображение данных**. Для преобразования самих данных нужно

использовать соответствующие узлы Loginom, например, для транспонирования используется **Кросс-таблица**.

Диаграмма – один из наиболее активно используемых визуализаторов. Она предназначена для визуального отображения зависимости значений одного поля от другого в виде двумерного графика. По горизонтальной оси откладываются значения независимого столбца, а по вертикальной – соответствующие им значения зависимого.

В Loginom доступны различные типы диаграмм, но наиболее часто используются:

- **Линии**;
- **Столбчатая**.

Этот визуализатор доступен для любого набора данных.

Пусть мы хотим получить график количества проданных товаров в подразделении **Аптека 1** по определенной товарной группе. Добавим компонент **Фильтр строк**, подадим на него данные и перейдем в настройки (рисунок 4.33).

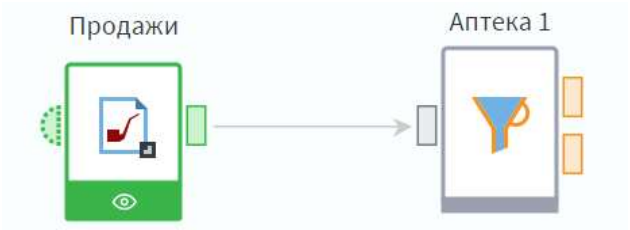


Рисунок 4.33 – Добавление в сценарий узла Фильтр строк

Добавим фильтр по нужному подразделению, в качестве группы товаров выберем **Иммуномодуляторы**. Далее добавим в сценарий узел **Группировка** (рисунок 4.34).

Рисунок 4.34 – Добавление в сценарий узла Группировка

Откроем настройку узла **Группировка**. Сгруппируем данные как показано на рисунке 4.35.

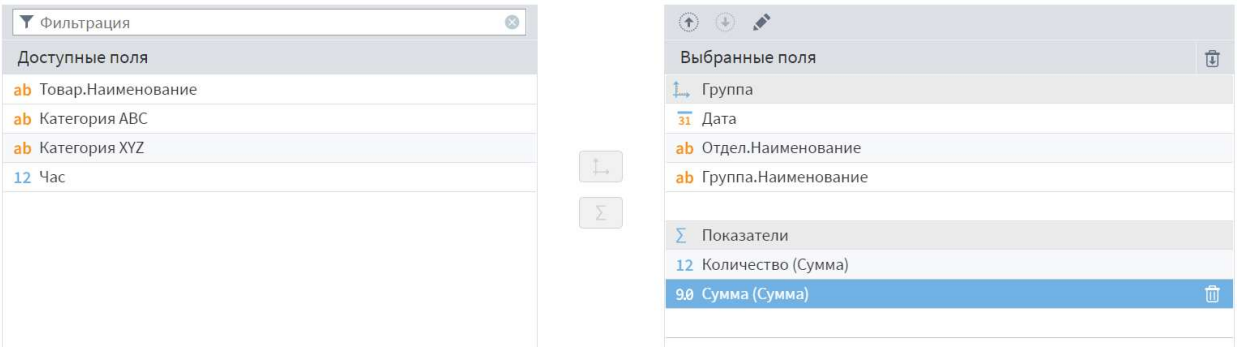


Рисунок 4.35 – Пример настройки узла Группировка

Для того чтобы график был легко читаемым и имел однозначную интерпретацию, перед построением диаграммы необходимо провести подготовку данных: одному значению независимого поля ставится в соответствие только одно значение зависимого.

Получили набор данных по ежедневным продажам товарной группы **Иммуномодуляторы** в **Аптека 1**.

Перейдем непосредственно к настройке визуализатора. Добавим визуализатор **Диаграмма** из группы. **Табличное представление** в область списка выходных портов узла и нажмем кнопку **Войти**. Перед нами окно построения диаграммы. Оно состоит из двух частей: **Область списка полей** и **Область построения диаграммы**. Для начала необходимо перетащить нужное поле в область построения. Она также состоит из двух частей: верхняя отвечает за ось **Y**, нижняя – за ось **X**. Возьмем поле **Количество** и перетащим его в верхнюю часть.

Нам открылось окно **Добавить серию** (то есть ряд значений, на основе которых строится диаграмма), где можно произвести первичную настройку диаграммы. Все настраиваемые параметры впоследствии можно изменить. Поэтому оставим настройки без изменения и нажмем кнопку **Добавить**.

Теперь необходимо задать настройки оси X. Мы хотим видеть, динамику изменения количества проданного товара по дням. Нажмем возле поля **Дата** кнопку **Использовать как поле абсцисс**. Также ось X можно задать:

- выделив поле и нажав комбинацию клавиш **Alt+X** на клавиатуре;
- перетащив нужное поле в нижнюю часть области построения;
- с помощью кнопки **Настройки осей...** на панели инструментов.

Перед нами итоговый график. Чтобы расширить область построения, скроем область списка полей. Это можно, сделать с помощью контекстного меню области построения или любым из способов (рисунок 4.36).

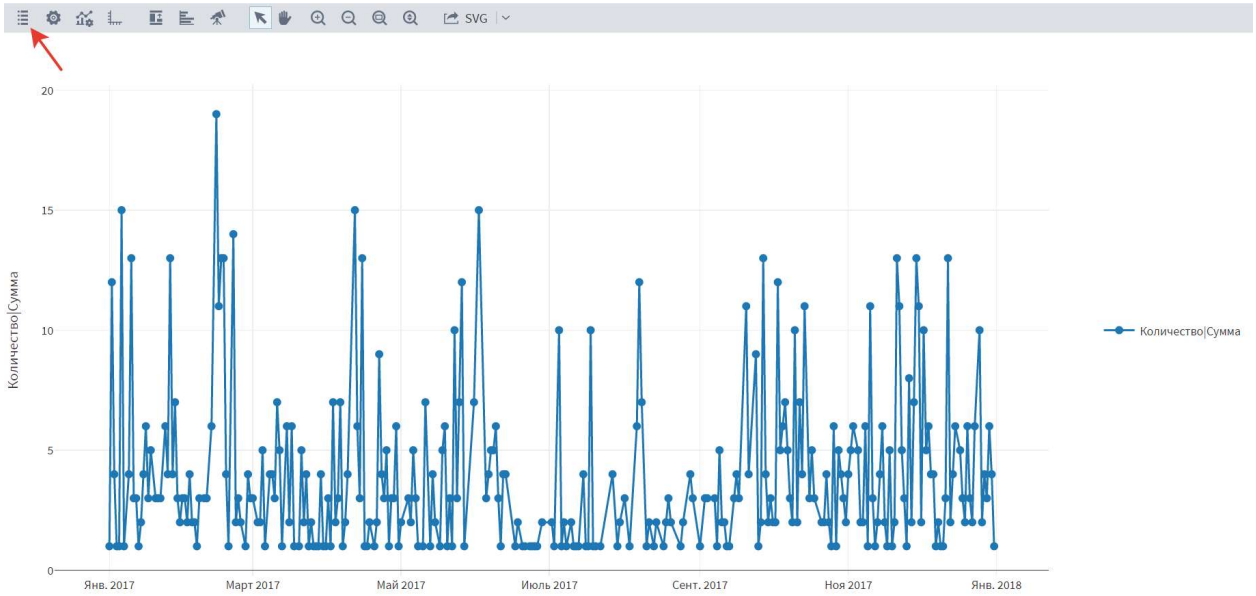


Рисунок 4.36 – Настройка итогового графика

Над областью построения расположена панель инструментов, где можно изменить любые настройки диаграммы.

Общие настройки отвечают в основном за внешний вид диаграммы и области построения. Их можно также открыть двойным щелчком левой кнопки мыши в области построения. Часть параметров из общих настроек вынесена также на панель инструментов.

Настройки осей позволяют задать параметры соответствующим осям: отображение сетки, тип оси, заголовок, минимальное и максимальное значения по осям, выравнивание.

Настройки нижней оси позволяют также задать поле оси X и поле меток, а настройки левой/правой осей – отображение динамики по оси по значению или в процентах.

При работе с диаграммой предусмотрена также возможность увеличения масштаба просмотра всей диаграммы или ее произвольной области. Для этого нужно, удерживая левую кнопку мыши нажатой, выделить ту область диаграммы, которую нужно просмотреть детально, при этом двигая мышь слева направо. Отменить масштабирование можно обратным действием.

Рассмотрим более детально продажи за последний месяц (рисунок 4.37).

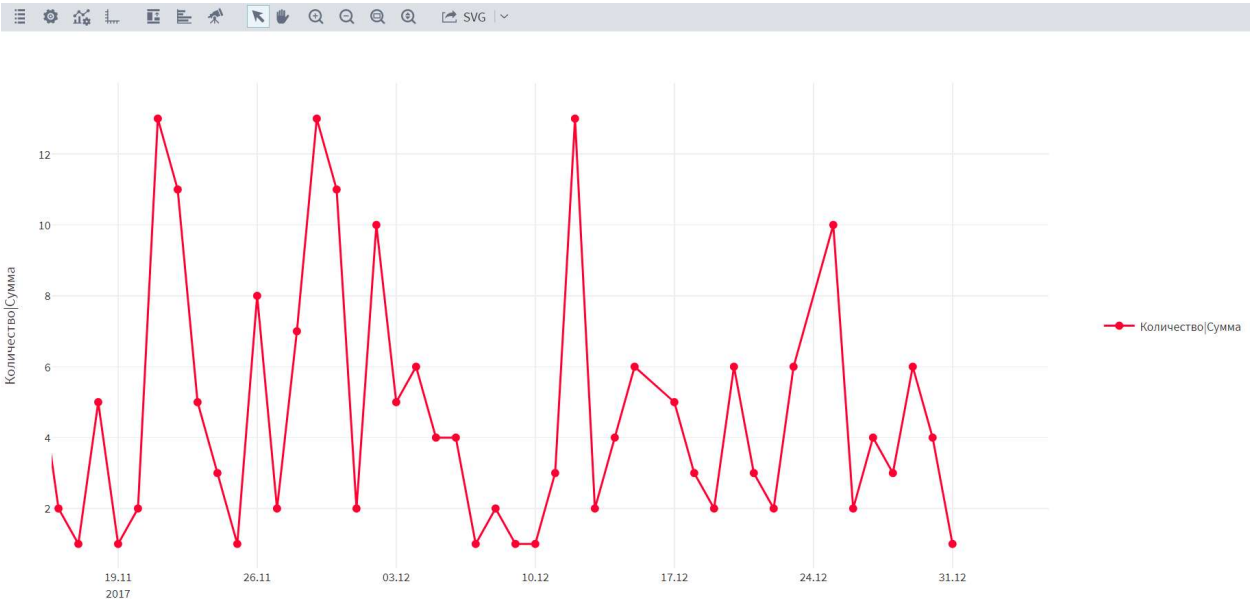


Рисунок 4.37 – График продаж за последний месяц

Перед нами график продаж за последний месяц. Нажав кнопку **Режим перетаскивания** на панели инструментов можно. перемещать диаграмму по экрану, делая доступными для просмотра различные ее части.

Кроме того, доступны кнопки на панели инструментов, позволяющие приблизить и отдалить диаграмму, а также полностью сбросить приближение. Нажмем кнопку **Сбросить масштаб**.

Мы вернулись к исходному виду диаграммы. Также осуществлять навигацию помогает кнопка **Навигатор**, она позволяет детализировать по оси X какой-либо участок диаграммы.

Пусть теперь мы хотим узнать, какое подразделение лидирует по продажам. Переименуем узел группировки в **Диаграмма (Линии)** и добавим новый узел группировки. Назовем его **Диаграмма (Столбчатая)** и перейдем в настройки (рисунок 4.38).

Рисунок 4.38 – Добавление в сценарий узел Группировка

Повторим настройки, как показано на рисунке 4.39.

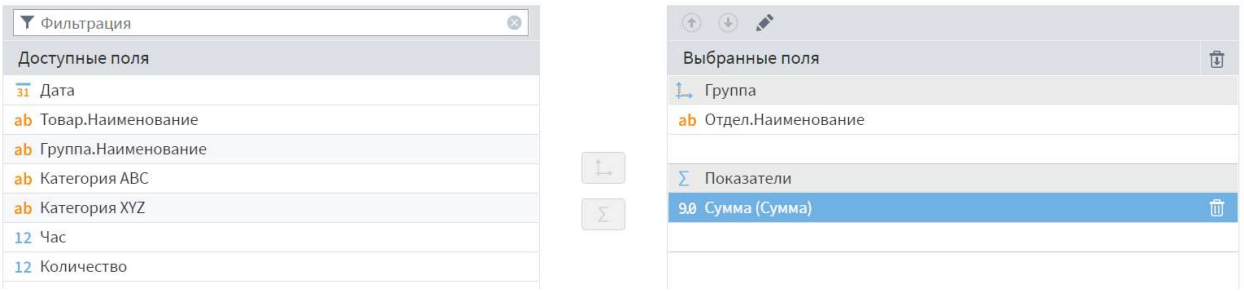


Рисунок 4.39 – Пример настройки узла Группировка

Получили общие суммы продаж по каждой аптеке за весь период (год). Для визуального анализа этих данных удобно использовать столбчатую диаграмму.

Добавим визуализатор **Диаграмма** и перетащим в верхнюю часть его области построения поле **Сумма**. В настройках серии изменим тип. Выберем вариант **Столбчатая**.

Нажатием на кнопку **Использовать как поле меток** у поля **Отдел.Наименование** добавили соответствующие подписи для столбцов диаграммы.

Легко увидеть, что **Аптека 1** является лидером по продажам за год (рисунок 4.40).

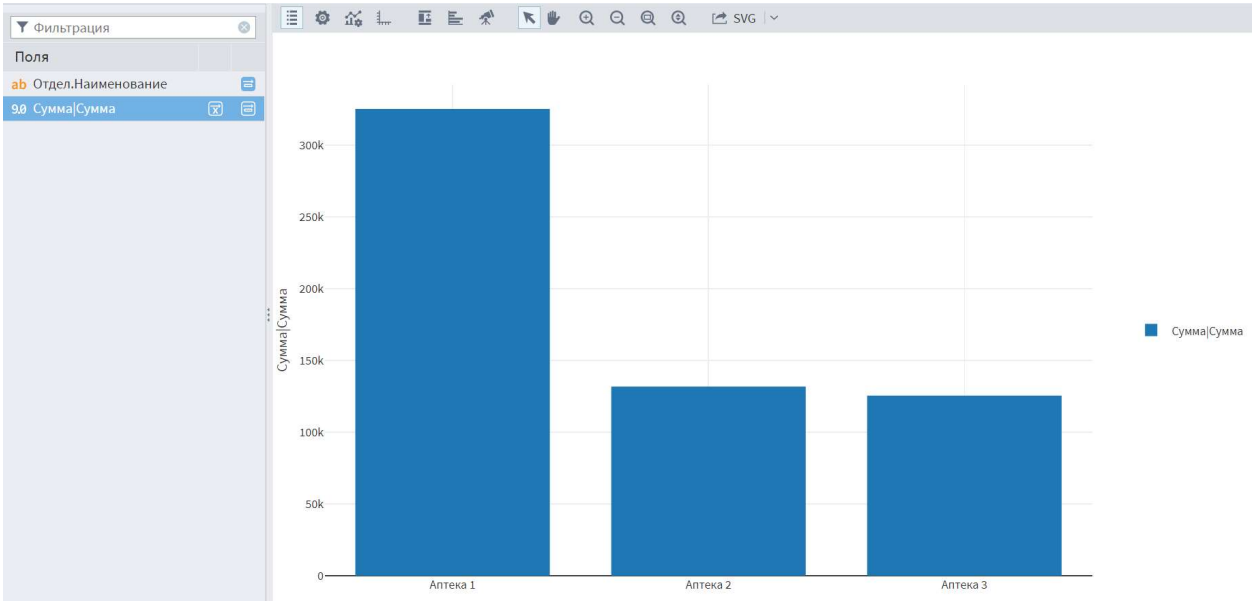


Рисунок 4.40 – Столбчатая диаграмма

С помощью контекстного меню можно также добавить значение суммы для каждой аптеки в качестве метки.

Рекомендуемая литература

1. Сабинин О.Ю., Шейкина Е.С. АВТОМАТИЗАЦИЯ ПОСТРОЕНИЯ ОТЧЕТОВ В СФЕРЕ СЕРТИФИКАЦИИ СИСТЕМ МЕНЕДЖМЕНТА С ПОМОЩЬЮ ORACLE BUSINESS INTELLIGENCE // Theoretical & Applied Science, 2017. – № 3 (47). – С. 121-127.
2. Доброжинская А.А. РОЛЬ BUSINESS INTELLIGENCE В БИЗНЕС-АНАЛИЗЕ // Развитие аналитического инструментария стратегического управления бизнесом. Материалы научно-исследовательской работы преподавателей и студентов Финансового университета при Правительстве Российской Федерации. Под редакцией М.М. Басовой, 2020. – С. 29-33.
3. Шаль А.В., Боковая К.А. ИСПОЛЬЗОВАНИЕ ИНСТРУМЕНТОВ BIG DATA И BUSINESS INTELLIGENCE В ЭКОНОМИЧЕСКОМ АНАЛИЗЕ // ЦИФРОВАЯ ЭКОСИСТЕМА ЭКОНОМИКИ. сборник статей по итогам IX международной научно-практической онлайн конференции. Ростов-на-Дону, 2022. – С. 247-250.
4. Литвина А.А. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ОБУЧЕНИЮ ШКОЛЬНИКОВ РАБОТЕ С ПЛАТФОРМАМИ BUSINESS INTELLIGENCE // Открытая наука 2021. Сборник материалов научной конференции с международным участием. Москва, 2021. – С. 338-342.
5. Дзюба А.Г., Жилина Е.В. РАЗРАБОТКА АНАЛИТИЧЕСКИХ СИСТЕМ НА ПРИМЕРЕ СРЕДЫ ORACLE BUSINESS INTELLIGENCE SUITE ENTERPRISE EDITION (OBIEE) // Новые направления научной мысли. сборник научных статей Национальной (Всероссийской) научно-практической конференции. Ростов-на-Дону, 2021. – С. 572-575.
6. Магамедова Л.Р. СУЩНОСТЬ СИСТЕМ BUSINESS INTELLIGENCE И ИХ ПРИМЕНЕНИЯ В УПРАВЛЕНИИ ЧЕЛОВЕЧЕСКИМИ РЕСУРСАМИ ПРЕДПРИЯТИЙ // ЦИФРОВОЙ КОНТЕНТ СОЦИАЛЬНОГО И ЭКОСИСТЕМНОГО РАЗВИТИЯ ЭКОНОМИКИ. СБОРНИК ТРУДОВ МЕЖДУНАРОДНОЙ НАУЧНО-ПРАКТИЧЕСКОЙ КОНФЕРЕНЦИИ. Симферополь, 2022. – С. 410-412.
7. Киселева В.В., Турыгина В.Ф. ПРИМЕНЕНИЕ СИСТЕМ BUSINESS INTELLIGENCE В ЦЕЛЯХ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ УПРАВЛЕНЧЕСКИХ РЕШЕНИЙ // Актуальные вопросы образования и науки. Сборник научных трудов по материалам международной научно-практической конференции, 2018. – С. 32-35.