Министерство науки и высшего образования Российской Федерации

Южно-Российский государственный политехнический университет (НПИ) имени М.И. Платова

С.Н. Широбокова

РАЗРАБОТКА КОРПОРАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ НА ПЛАТФОРМЕ «1С:ПРЕДПРИЯТИЕ»: КОНСПЕКТ ЛЕКЦИЙ

Учебное пособие

Новочеркасск
ЮРГПУ(НПИ)
2021

УДК 004.42 (075.8)
ББК 32.973-018.1я73
Ш64

Широбокова С.Н.

Ш64 Разработка корпоративных информационных систем на платформе «1С:Предприятие»: конспект лекций: учебное пособие / С.Н. Широбокова; Южно-Российский государственный политехнический университет (НПИ) имени М.И. Платова. – Новочеркасск: ЮРГПУ(НПИ), 2021.– 109с.

Учебное пособие содержит конспект лекций по дисциплине "Разработка корпоративных информационных систем на платформе «1С:Предприятие»". Рассмотрены вопросы архитектуры корпоративных информационных систем и бизнес-приложений на платформе «1С:Предприятие», походы к хранению информации, аспекты решения задач оперативного и бухгалтерского учета, задач сложных периодических расчетов.

Предназначено для студентов вузов, обучающихся по направлению подготовки «Прикладная информатика» (магистратура). Пособие может быть полезно студентам других компьютерных направлений подготовки, изучающим основы разработки корпоративных информационных систем на платформе «1С:Предприятие».

УДК 004.42 (075.8)
ББК 32.973-018.1я73

Содержание

ВВЕДЕНИЕ

Цели освоения дисциплины:

– формирование у обучающихся знаний о возможностях разработки на базе отечественной технологической платформы "1С:Предприятие", использование которой для разработки российского программного обеспечения соответствует политике импортозамещения, современных решений для обработки больших объемов учетной информации и цифровизации бизнеса: корпоративного сектора, отраслевых и специализированных решений для промышленности, предприятий топливно-энергетического комплекса, машиностроения, металлургии, химической промышленности, горнодобывающей промышленности, сельского хозяйства, транспорта и логистики, строительства и жилищно-коммунального хозяйства, энергетики, финансового сектора, здравоохранения, государственного и муниципального управления, оптовой и розничной торговли, сферы услуг и др.;

– формирование у обучающихся навыков проектирования и создания корпоративных информационных систем (КИС) на платформе «1С:Предприятие»;

– получение углубленных практических навыков конфигурирования прикладных объектов и программирования на встроенном языке «1С:Предприятие» при решении различных типов задач учета: оперативных, бухгалтерских и расчетных.

Задачи освоения дисциплины:

– формирование развёрнутого представления об архитектуре корпоративных информационных систем, построенных на платформе «1С:Предприятие», о подходе к описанию бизнес-приложения на основе модели метаданных, вариантах работы информационной системы (ИС) на платформе «1С:Предприятие», видах взаимодействия компонентов, видах клиентских приложений;

– ознакомление с парадигмой визуального проектирования и предметной ориентированности встроенного языка системы, подходами к хранению различной информации (иерархической, имеющей привязку ко времени, объектных и необъектных сущностей и т.д.), понятием управляемого приложения и декларативным описанием пользовательского интерфейса в приложениях на платформе «1С:Предприятие»;

– формирование навыков проектирования и реализации алгоритмов обработки больших объемов учетной информации в КИС на платформе «1С:Предпряитие».

– формирование теоретических знаний и навыков решения задач оперативного учета в корпоративных информационных системах;

– формирование теоретических знаний и навыков реализации в корпоративных информационных системах задач бухгалтерского учета, основ организации аналитического учета;

– формирование теоретических знаний основных понятий и навыков применения технологии реализации сложных периодических расчетов в КИС на платформе «1С:Педприятие».

Тема 1. Архитектура систем бизнес-аналитики. Архитектура и основные понятия платформы «1С:Предприятие»

Архитектура платформы «1С:Предприятие». Метаданные как способ описания бизнес-приложения: построение прикладного решения на основе модели. Классификация объектов конфигурации. Варианты работы системы "1С:Предприятие": файловый вариант, клиент-серверный вариант. Виды взаимодействия компонентов: прямое подключение, подключение через веб-сервер, мобильная платформа. Виды клиентских приложений: тонкий клиент, веб-клиент, мобильный клиент, приложение на мобильной платформе, мобильный клиент с автономным режимом.

Система программ «1С:Предприятие 8» включает в себя технологическую платформу и прикладные решения, разработанные на ее основе для автоматизации деятельности организаций и частных лиц. Сама платформа не является программным продуктом для использования конечными пользователями, которые обычно работают с одним из многих прикладных решений (конфигураций), разработанных на данной платформе [1]. Такой подход позволяет автоматизировать различные виды деятельности, используя единую технологическую платформу. Схема архитектуры решений на платформе «1С:Предприятие» представлена на рис. 1.1.

Рис. 1.1.
Схема архитектуры решений на платформе «1С:Предприятие»

Платформа «1С:Предприятие» включает две составные части: средства разработки и среда исполнения. Платформа обеспечивает работу конфигурации и позволяет вносить в нее изменения или создавать собственную конфигурацию. Технологии и механизмы платформы служат для создания, модификации и собственно функционирования конфигурации.

Но платформа «1С:Предприятие служит не только для настройки и исполнения типовых решений от фирмы «1С». Это еще и средство для создания новых прикладных решений, которые могут разрабатываться как на основе типовых конфигураций, так и «с нуля» – индивидуальные решения «под заказ».

Платформа «1С:Предприятие» является инструментом при реализации проектов разработки корпоративных информационных систем и автоматизации задач бизнес-аналитики. Это специализированное (предметно-ориентированное) средство разработки, ориентированное на задачи автоматизации бизнеса.

Конфигурация – прикладное решение, предназначенное для автоматизации определенной области человеческой деятельности. Является самостоятельной сущностью и может выступать в качестве отдельного программного продукта. Но ее создание, изменение и «исполнение» осуществляется с использованием платформы.

Внедрения, как правило, выполняются силами партнера и IT-специалистов заказчика с учетом особенностей деятельности предприятия и пожеланий заказчика.

Информационная база содержит: пользовательские ***данные*** + 2 конфигурации (***основная и конфигурация базы данных***) + *(необязательно) конфигурация поставщика*

В информационной базе, во-первых, содержаться пользовательские данные. Иными словами, это некоторые элементы справочников, который формирует оператор, документы и другие данные. Во-вторых, в информационной базе содержится, как минимум, две конфигурации. Это конфигурация основная, и конфигурация базы данных.

Для чего же нужны эти две конфигурации? Всё дело в том, что разработчик работает именно с основной конфигурацией. То есть, когда разработчик вносит какие-либо изменения в конфигураторе, все изменения делаются именно в основной конфигурации. А с конфигурацией общей базы данных работают пользователи, они обращаются к ней и вносят изменения в данные информационной базы. Для чего необходима такая схема? Дело в том, что разработчик при такой схеме взаимодействия может менять основную конфигурацию, вносить в неё какие-либо изменения, а параллельно могут осуществлять работу операторы со своей конфигурацией. В тот момент, когда настанет необходимость синхронизации двух конфигураций, можно попросить пользователей выйти из информационной системы, когда разработчики будут готовы сделать обновление, и выполнить обновление конфигурации новой базы данных до основной конфигурации.

Модель метаданных как способ описания бизнес-приложения. Когда создается корпоративная информационная система на платформе «1С:Предприятие» – в первую очередь строится его модель на основе метаданных, т.е. не только пишется код, как при создании приложения на других языках программирования. Хотя встроенный язык программирования в системе «1С», тоже есть и используется, но только там, где это действительно необходимо.

Рис. 1.2. Метаданные – способ описания бизнес-приложения

Метаданные (рис. 1.2) представляют собой структурированное декларативное описание бизнес-приложения и образуют иерархию объектов, которые формируют все составные части прикладной системы и определяют аспекты ее поведения. Платформа «1С:Предприятие» при работе бизнес-приложения «проигрывает» (интерпретирует) метаданные, при этом обеспечивая всю необходимую функциональность.

Метаданными описываются структуры данных, состав типов, связи между объектами, особенности их поведения и визуального представления, система разграничения прав доступа, пользовательский интерфейс и т. д. В метаданных,

фактически, сосредоточены сведения не только о том, «что хранить в базе данных», но и о том, «зачем» хранится та или иная информация, какова ее роль в системе и как связаны между собой информационные массивы. Использование языка программирования ограничено в основном решением тех задач, которые действительно требуют алгоритмического описания, например, расчета налогов, проверки корректности введенных данных и т.д.

Преимущества использования такого подхода:

– при описании метаданных широко используется ***визуальное редактирование***. Это позволяет свести существенную часть разработки к визуальному проектированию, не требующему кропотливого написания кода;

– описывая прикладное решение в терминах метаданных, разработчик «сообщает» платформе много очень полезной информации, которую та может эффективно использовать в самых различных целях. На основе метаданных система автоматически «выстраивает» большую часть механизмов и объектов, обеспечивающих функционирование прикладного решения. Например, описания метаданных платформе достаточно для того, чтобы автоматически сформировать пользовательский интерфейс системы, обеспечивающий ввод и редактирование взаимосвязанной информации. Другой пример — возможность построения даже конечным пользователем, не имеющим навыков программирования, достаточно сложных отчетов;

– идеология (Metadata Driven): *«Давайте не будет программировать все функции разрабатываемого решения. Расскажем платформе о составе, структуре, особенностях и взаимосвязи различных его частей, и пусть остальное она сделает сама» – тезис, кратко отражающий суть идеологии* использования метаданных [1]. В системе «1С:Предприятии» изначально заложена строгая ориентация на построение прикладного решения на основе определенной модели.

Эта идеология сегодня находит все большее применение во многих перспективных разработках. Подход является весьма перспективным и, как считают специалисты, оценке будет доминирующим в обозримом будущем в современных средствах разработки. Идеи построения бизнес-приложений на основе модели, например, нашли воплощение в архитектуре MDA (Model Driven Architecture) консорциума OMG.

Под моделью понимается вся идеология построения прикладного решения. Сюда относятся способы построения структур данных, типы связей между данными, принципы манипулирования данными, формы описания бизнес-логики, способы связи данных с интерфейсными объектами, разделение функциональности по уровням системы и многое другое.

Важно, что все бизнес-приложения неукоснительно следуют принятой модели и этим обеспечивается единообразие и предсказуемость их поведения. Фактически, разработчик, желающий отразить в прикладном решении специфику той или иной предметной области, имеет вполне определенный набор способов решения этой задачи средствами, заложенными в платформу. С одной стороны, такой подход ограничивает (вполне осмысленно) свободу разработчика, но с другой — защищает его от множества ошибок и позволяет в

сжатые сроки получать работоспособное решение, которое сможет в дальнейшем развиваться и поддерживаться как им самим, так и, при необходимости, другим специалистом.

Очевидным следствием этого подхода является изоляция разработчика бизнес-приложения от деталей технологий хранения информации, организации трехуровневой архитектуры и т. д. Например, как уже отмечалось выше, все прикладные решения, базирующиеся на «1С:Предприятии», без каких-либо изменений работают как с собственным файловым движком БД, так и с сервером БД. При этом необходимые структуры данных создаются и изменяются системой автоматически на основе описания метаданных, и разработчику не приходится вникать в детали форматов хранения конкретных СУБД. Манипулирование данными в приложении также описывается в высокоуровневой модели и автоматически исполняется с учетом особенностей используемого хранилища.

По нашей оценке, предоставление разработчику определенной модели, абстрагированной от конкретных используемых средств, будет доминирующим в обозримом будущем в современных средствах разработки.

Наличие единой модели ключевым образом сказывается и на простоте освоения системы. Вся разработка ведется в рамках одной сквозной системы понятий и в едином пространстве типов данных. У разработчика не возникает необходимости осваивать несколько моделей представления и тратить усилия на реализацию переходов между ними на разных уровнях.

Например, описание в метаданных тех или иных объектов (сущностей) сразу определяет и соответствующие типы встроенного языка программирования и необходимые для их хранения структуры БД. Все последующие манипуляции этими объектами, как в памяти, так и в БД выполняются единообразно, не требуя преодоления «барьеров» между различными нотациями, принятыми при работе с СУБД и с универсальными языками программирования.

Таким образом, на этапе создания конфигурации разработчик анализирует предметную область и требования пользователей, создает объекты конфигурации, настраивает связи между ними путем установки их свойств, визуально конструирует экранные формы и макеты отчетов, пишет программные модули в определенных точках конфигурации. В результате получается прикладное решение, призванное облегчить работу конечных пользователей.

Объекты конфигурации — это составные элементы, «детали», из которых складывается любое прикладное решение.

Они представляют собой проблемно-ориентированные объекты, поддерживаемые на уровне технологической платформы. По большому счету задача разработчика заключается в том, чтобы собрать из этих объектов, как из конструктора, необходимую структуру прикладного решения и затем описать специфические алгоритмы функционирования и взаимодействия этих объектов, отличающиеся от их типового поведения.

Состав объектов, поддерживаемых технологической платформой, является результатом анализа предметных областей использования «1С:Предприятия», и выделения и классификации используемых в этих областях бизнес-сущностей. В

результате этого анализа разработчик может оперировать такими объектами как справочники, документы, регистры сведений, планы счетов и пр. Разбивку объектов по видам можно увидеть в дереве конфигурации (они находятся на первом его уровне).

Разработчик оперирует метаданными — «данными о данных», или объектами конфигурации. Добавляя в структуру прикладного решения очередной объект конфигурации, разработчик, по сути, добавляет описание того, как будут размещаться соответствующие данные, и как они будут взаимодействовать с другими данными, хранящимися в информационной базе.

Состав объектов, которые может использовать разработчик, фиксирован и определен на уровне платформы. Разработчик не может создавать собственные виды объектов, он может оперировать только тем набором объектов, который имеется. Подобный подход к разработке прикладных решений позволяет, во-первых, стандартизировать процесс разработки, а во-вторых — обеспечить простую и быструю модификацию прикладных решений другими разработчиками или пользователями.

Рис. 1.3. Взаимосвязь объектов конфигурации

Три основные группы объектов конфигурации:

– *Общие объекты* – вспомогательные объекты конфигурации, с помощью которых осуществляется создание конфигурации, механизмов взаимодействия пользователей с учетными данными.

– *Прикладные объекты.* Их перечень можно увидеть на первом уровне дерева метаданных (исключая ветку «Общие»).

– *Подчиненные объекты.* В зависимости от вида объекта конфигурации (прикладного или общего) он может иметь различные подчиненные группы объектов. К ним относятся «Реквизиты», «Табличные», «Реквизиты табличных частей», «Формы», «Макеты», «Графы», «Измерения», «Ресурсы» и т.д.

В самом общем виде взаимосвязь всех прикладных объектов можно представить схемой на рис. 1.3 [1].

Блок "**Условно-постоянная информация**" содержит объекты, сохраняемые в базе данных и содержащие данные, меняющиеся **сравнительно редко**. Например, константа "Название организации", справочник "Сотрудники", перечисление "Тип клиента" и т.д. Можно сказать, что в этот блок данные вводятся один раз и используются много раз, в нескольких хозяйственных операциях, актах расчета.

Блок "**Документы**" включает, во-первых, документы, предназначенные для регистрации **событий и операций**, и, во-вторых, журналы, как средство их смысловой группировки. Например, документы "Приходная накладная", "Расходная накладная" и журнал "Складские документы". Документ характеризуется номером и датой. С помощью служебных объектов "Нумераторы" можно организовать "сквозную" нумерацию документов разных видов. Другой служебный объект "Последовательность" предназначен для поддержания правильности движений по регистрам, путем строгого порядка проведения документов.

Блок "**Регистры**" предназначен для хранения информации **о состояниях и количествах объектов** базы данных, например, регистр сведений "Состояние сотрудников", "Цены товаров", регистры накопления "Продажи", "Остатки товаров" и т.д. В регистрах, кроме фактических данных, могут храниться также плановые данные, например, плановый объем продаж, прогнозируемые курсы валют и т.д.

Блок "**Обработка и вывод информации**" включает обработки и отчеты, которые используют уже введенные в базу данные для их **обработки и представления пользователю** (печати). Обработки предназначены для выполнения действий и расчетов над имеющейся в базе информацией, например, обработка "Закрытие периода", а отчеты формируют различные печатные формы, например, отчет "Анализ продаж".

Существует два варианта работы платформы.

Файловый вариант работы (рис. 1.4) рассчитан на персональную работу одного пользователя или работу небольшого количества пользователей в локальной сети.

Рис. 1.4. Файловый вариант работы

В этом варианте все данные информационной базы (конфигурация, база данных, административная информация) располагаются в одном файле - файловой базе данных. Работу с этой базой данных осуществляет файловая СУБД, разработанная фирмой "1С" и являющаяся частью платформы.

Такой вариант работы обеспечивает легкость установки и эксплуатации системы. При этом для работы с информационной базой не требуются дополнительные программные средства, достаточно иметь операционную систему и системы «1С:Предприятие».

Файловый вариант работы обеспечивает целостность информационной базы и простое создание резервных копий.

Клиент-серверный вариант работы (рис. 1.5) предназначен для использования в рабочих группах или в масштабе предприятия. Он реализован на основе трехуровневой архитектуры «клиент-сервер».

Клиент-серверная архитектура разделяет всю работающую систему на три различные части, определенным образом взаимодействующие между собой:

– клиентское приложение,
– кластер серверов «1С:Предприятия»,

– сервер базы данных.

Программа, работающая у пользователя, (клиентское приложение) взаимодействует с кластером серверов «1С:Предприятия 8», а кластер, при необходимости, обращается к серверу баз данных.

При этом физически кластер серверов «1С:Предприятия 8» и сервер баз данных могут располагаться как на одном компьютере, так и на разных. Это позволяет администратору при необходимости распределять нагрузку между серверами. Использование кластера серверов «1С:Предприятия 8» позволяет сосредоточить на нем выполнение наиболее объемных операций по обработке данных. Например, при выполнении даже весьма сложных запросов программа, работающая у пользователя, будет получать только необходимую ей выборку, а вся промежуточная обработка будет выполняться на сервере. Обычно увеличить мощность кластера серверов гораздо проще, чем обновить весь парк клиентских машин.

Рис. 1.5. Клиент-серверный вариант работы

Кластер серверов «1С:Предприятия 8» – основной компонент платформы, обеспечивающий взаимодействие между пользователями и системой управления базами данных в клиент-серверном варианте работы. Наличие кластера позволяет обеспечить бесперебойную, отказоустойчивую, конкурентную работу большого количества пользователей с крупными информационными базами. Кластер серверов 1С:Предприятия 8 является логическим понятием и представляет собой совокупность рабочих процессов, обслуживающих один и тот же набор информационных баз.

Сервер баз данных – это система управления базами данных (СУБД).

Система управления базами данных (СУБД) – специализированная программа (чаще комплекс программ), предназначенная для организации и ведения базы данных.

Платформа поддерживает работу с пятью СУБД.

Одна из этих СУБД, – файловая СУБД, разработаная фирмой "1С" и являющаяся частью платформы. Она используется в файловом режиме.

В клиент-серверном режиме, требующем обеспечить работу многих пользователей, используются СУБД сторонних поставщиков. 1С работает со следующими СУБД:

- Microsoft SQL Server;
- PostgreSQL;
- IBM DB2;
- Oracle Database.

Клиентское приложение – это программа, работающая на компьютере пользователя и обеспечивающая интерактивное взаимодействие системы 1С:Предприятие 8 с пользователем, в отличие от других компонент системы (программ и рабочих процессов), предназначенных исключительно для программного взаимодействия с другими частями системы или с другими программными объектами [1].

Почему появилась необходимость разделения обычного клиентского приложения на несколько видов: «тонкий», «толстый» и web-клиент и др.? Виной этому развитие технологий вообще и интернета в частности. Очень часто стала появляться необходимость в работе с «1С: Предприятием» через сеть интернет, и это стало накладывать определенные ограничения, поскольку пропускная способность сети интернет гораздо уже, чем обычной локальной сети. Поэтому те технологии платформы, которые существовали при редакциях 8.0 и 8.1, стали неприменимы в новых реалиях. Как следствие, была разработана платформа 8.2,затем 8.3, в которых обычное приложение разделилось на несколько видов.

В текущей версии (8.3) клиентских приложений несколько:

- толстый клиент ;
- тонкий клиент;
- веб-клиент;
- мобильный клиент.

Их различия представлены в табл. 1.1 [1].

Таблица 1.1

Различия клиентских приложений

	Толстый клиент	Тонкий клиент	Веб-клиент	Мобильный клиент
Работа с конфигуратором, разработка	Да	Нет	Нет	Нет
Работа в локальной сети	Да	Да	Да	Нет
Работа через Интернет	Нет	Да	Да	Да
Необходимость предварительной установки	Да, большой дистрибутив	Да, маленький дистрибутив	Нет	Да
Работа на мобильных устройствах	Нет	Нет	iPad	Да

Толстый клиент позволяет реализовывать полные возможности «1С:Предприятия 8» в плане исполнения прикладного кода и единственный режим в котором можно запускать конфигуратор и вести разработку. Однако он не поддерживает работу с информационными базами через интернет, требует предварительной установки на компьютер пользователя и имеет довольно внушительный объем дистрибутива.

Тонкий клиент не позволяет разрабатывать и администрировать прикладные решения, однако может работать с информационными базами через интернет. Он также требует предварительной установки на компьютер пользователя, но имеет значительно меньший размер дистрибутива, чем толстый

клиент. «Тонким» клиент называется потому, что умеет исполнять ограниченный набор функциональности встроенного языка. В частности, на тонком клиенте недоступны все прикладные типы данных. Вместо этого тонкий клиент оперирует ограниченным набором типов встроенного языка, предназначенным лишь для отображения и изменения данных в памяти. Вся работа с базой данных, объектными данными, исполнение запросов – выполняется на стороне сервера. Тонкий клиент только получает готовые данные, подготовленные для отображения. Тонкий клиент обеспечивает работу только в пользовательском режиме «1С:Предприятие». Режим работы Конфигуратор тонким клиентом не поддерживается.

Веб-клиент не требует какой-либо предварительной установки на компьютер. В отличие от толстого и тонкого клиентов, он исполняется не в среде операционной системы компьютера, а в среде интернет-браузера (Microsoft Internet Explorer или Mozilla Firefox). Поэтому пользователю достаточно всего лишь запустить свой браузер, ввести адрес веб-сервера, на котором опубликована информационная база – и веб-клиент «сам приедет» к нему на компьютер и начнет исполняться.

Новая технология – Мобильный клиент.

До недавнего времени платформа «1С:Предприятие» предлагала единственную технологию, с помощью которой можно было работать с её приложениями, используя мобильные устройства. Это мобильная платформа. По своей архитектуре такие приложения очень похожи на файловый вариант работы системы 1С:Предприятие. На мобильном устройстве существует собственная база данных, «внутри» мобильного приложения существует как клиент, обеспечивающий взаимодействие с пользователем, так и сервер, обеспечивающий взаимодействие с базой данных.

Такие мобильные приложения могут взаимодействовать с «основным» приложением, установленным в офисе. Но это не онлайн взаимодействие, а периодический обмен данными с бэк-офисом (рис. 1.6).

Рис. 1.6. Мобильное приложение

Основная работа в мобильном приложении ведется в оффлайн-режиме. А при появлении интернет-соединения выполняется синхронизация данных. Мобильный клиент реализован с версии 8.3.12.64 мобильной платформы.

Мобильный клиент – это тонкий клиент для мобильных устройств, который обладает интерфейсом, аналогичным мобильной платформе (рис. 1.7).

Дистрибутив мобильного клиента содержит все необходимые исполняемые файлы, из которых разработчик может собрать приложение для мобильного устройства аналогично тому, как собираются мобильные приложения из мобильной платформы. Такое приложение, с одной стороны, может напрямую взаимодействовать с кластером серверов «1С:Предприятие» точно так же, как это делает тонкий клиент.

С другой стороны, мобильный клиент обеспечивает автоматическую трансформацию форм, декларативно описанных в конфигурации, в интерфейс, аналогичный интерфейсу мобильной платформы.

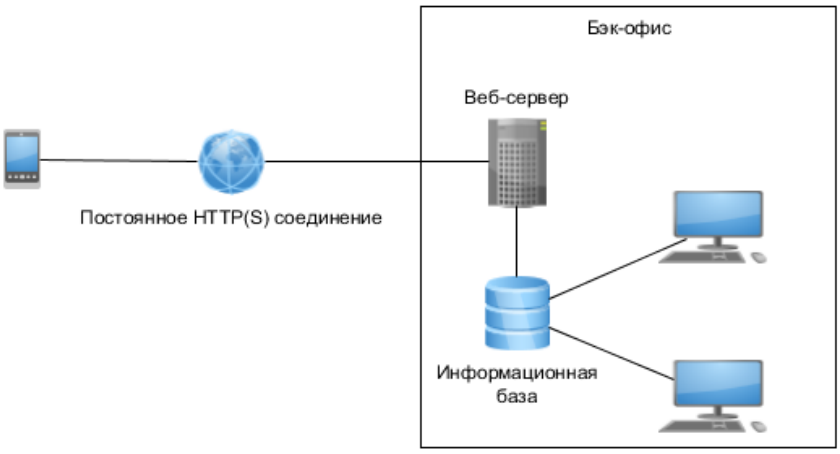


Рис. 1.7. Мобильный клиент

Формы, разработанные для настольной версии «1С:Предприятие», мобильный клиент автоматически компонует таким образом, чтобы обеспечить удобство работы с ними на маленьких экранах мобильных телефонов на приемлемом уровне.

Эта технология будет востребована в тех внедрениях, где обязательно требуется онлайн доступ в систему с мобильных устройств. Чтобы данные, введенные на мобильном устройстве, попадали непосредственно в «общую» базу данных, минуя промежуточные шаги синхронизации.

Конфигуратор – один из двух режимов работы системы. Функционирование системы подразделяют на два разделенных во времени процесса: настройку (конфигурирование) и исполнение, исходя из чего система «1С: Предприятие» имеет два основных режима запуска – «**Конфигуратор»** и «**1С:Предприятие»** (перечисленные ранее виды клиентских приложений работают во втором режиме – режиме исполнения). Первый из этих режимов предназначен для создания информационной базы, внесения изменений в ее конфигурацию, а также для выполнения административных функций.

В режиме «1С: Предприятие» пользователь запускает Конфигурацию на выполнение, как бы «проигрывая» файл информационной базы. При этом программная часть системы использует структуры, созданные на этапе конфигурирования, предоставляя пользователю возможность заполнить их конкретными значениями.

Если на этапе конфигурирования с помощью встроенного языка определены соответствующие алгоритмы обработки, то в режиме «1С: Предприятие» пользователь будет вызывать их работу, давая системе соответствующие команды.

Тема 2. Парадигма визуального проектирования и предметной ориентированности встроенного языка системы

Парадигма визуального проектирования и предметной ориентированности встроенного языка системы. Управляемое приложение. Декларативное описание пользовательского интерфейса. Клиентская и серверная части приложения. Директивы компиляции. Типы модулей и их назначение. Основные конструкции встроенного языка.

Существенная часть разработки бизнес-приложений на платформе «1С:Предприятие» ведется в концепции декларативного программирования, без написания кода. Широко используется визуальное редактирование, что позволяет свести объем собственно программирования к минимуму (парадигма разработки low-code, «минимальное программирование»). Для написания программного кода используется высокоуровневый предметно-ориентированный язык с понятийной моделью, максимально приближенной к задачам бизнеса (реализован подход Domain-Driven Design — предметно-ориентированное проектирование).

Исходный код прикладных решений на платформе «1С:Предприятие 8» открыт. Благодаря используемым парадигмам визуального проектирования и предметной ориентированности языка системы этот код является реально открытым — внедренцы и пользователи могут легко его прочитать, разобраться в бизнес-логике прикладных решений, поддерживать и развивать их, модифицировать и расширять функционал в соответствии с задачами организации. Реальная открытость прикладных решений «1С:Предприятия 8» позволяет легко передавать их для развития или на сопровождение от одних специалистов другим, например, от внедренческой фирмы в ИТ-службу организации-пользователя.

С версии платформы 8.2 в 1С стали использоваться новые принципы построения интерфейса и взаимодействия пользователя с базой данных. Новая технология получила название «**Управляемое приложение**». Наибольшей переработке подверглись механизмы построения форм и схема взаимодействий пользователя сервера 1С и базы данных. Обычный режим все еще поддерживается платформой, но со временем все пользователи 1С перейдут на управляемые формы. Для разработчика же это новый механизм со своими правилами, законами и условиями. Изменению подверглись многие области, но ключевыми среди опытных разработчиков 1С считаются следующие нововведения:

- Самостоятельное формирование структуры формы и размещение полей платформой. Если раньше разработчики описывали положение поля, указывая пиксели, то теперь есть возможность лишь указать вид группировки;
- Форма состоит из реквизитов, представляющих данные формы, и команд – выполняемых процедур и функций;
- Код формы выполняется на стороне и сервера, и клиента. Ведь сама по себе форма – это объект конфигурации, создаваемый на сервере и отображаемый на клиенте. Значит, объединяет в себе клиентскую и серверную часть;

- На клиентской стороне стали недоступны многие типы данных и теперь отсутствует возможность изменить данные в информационной базе;
- Для каждой процедуры или функции должна быть указана специальная настройка – директива компиляции. Она отвечает за место выполнения кода и может принимать следующие значения: &НаКлиенте; &НаСервере; &НаСервереБезКонтекста; &НаКлиентеНаСервере; &НаКлиентеНаСервереБезКонтекста. Рассмотрим их подробнее.

&НаКлиенте (&AtClient)

Директива определяет выполнение процедуры (функции) на клиенте. Используется на клиенте, доступны процедуры модуля и доступны данные форм.

&НаСервере (&AtServer)

Директива определяет выполнение процедуры (функции) на сервере. Выполняется на серверном приложении. В таких процедурах доступны данные формы, доступен серверный контекст формы и вызовы серверных процедур модуля. Данные формы передаются с клиента на сервер и обратно по окончанию вызова.

&НаСервереБезКонтекста (&AtServerNoContext)

Директива определяет выполнение процедуры (функции) на сервере вне контекста формы. В данном случае не будут доступен контекст формы и ее данные. Позволяет вызывать только внекотекстные процедуры и функции и не позволяет выполнять передачу данных между клиентом и сервером. Данный метод позволяет существенно снизить объем передоваемой информации.

&НаКлиентеНаСервереБезКонтекста (&AtClientAtServerNoContext)

Директива определяет выполнение процедуры (функции) на сервере и на клиенте, не имеющую доступа к данным формы, переменным. В данном методе имеется доступ к процедурам и функциям клиентских и серверных одновременно.

&НаКлиентеНаСервере (&AtClientAtServer)

Директива определяет выполнение процедуры (функции) на сервере и на клиенте. В данном методе имеется доступ к процедурам и функциям общих модулей – серверных, не глобальных и серверных и клиентских одновременно, не имеющую доступ к переменным.

В директивах 1С БезКонтекста недоступны реквизиты формы 1С 8.3 и экспортные переменные формы 1С, но доступен вызов процедур и функций 1С из серверных общих модулей 1С 8.3. По умолчанию если перед процедурой (функцией) ничего не указано, то применяется директива 1С &НаСервере. Перед одной процедурой (функцией) нельзя применять одновременно несколько директив компиляции. Также недопустимо наличие процедур (функций) с одинаковым именем, отличающихся только директивами компиляции.

Обзор встроенного языка системы.

Встроенный язык является важной частью технологической платформы «1С:Предприятие 8». Его наличие позволяет разработчику описывать собственные алгоритмы функционирования прикладного решения [1], он поддерживает концепцию настраиваемости системы. Встроенный язык «1С» является предварительно компилируемым предметно-ориентированным языком

высокого уровня. Встроенный язык имеет много общих черт с другими языками, такими как Pascal, Java Script, Basic, но не является прямым аналогом какого-либо из этих языков. С использованием объектной техники в языке организована работа со специализированными типами данных предметной области, определяемыми конфигурацией системы.

Как и в других языка программирования, поддерживаются конструкции, позволяющие определять переменные, процедуры, функции. Друг от друга операторы отделяются символом «;».

Создание новых классов программно в языке 1С 8.3 запрещено. Так как платформа 1С предприятие специализирована для задач учета — состав классов заранее предопределен: документы, справочники, регистры бухгалтерии, регистры накопления и т.д. На основании типовых классов можно в конфигураторе создать любое количество подклассов имеющих свои наборы. С помощью модулей менеджеров можно незначительно расширить функционал подкласса.

Хотя встроенный язык и не чувствителен к регистру, допускает двуязычное написание конструкций (Если, If), но все же рекомендуется писать код на языке типовых конфигураций, т.е. на русском.

Пример кода 1С:

```
Message("Hello, World!");
Сообщить("Привет, Мир!");
```

Наиболее значимые особенности встроенного языка:

– предварительная компиляция – перед исполнением модули, содержащие текст на встроенном языке, преобразуются во внутренний код;

– кэширование скомпилированных модулей в памяти;

– мягкая типизация – тип переменной определяется типом значения, которое она содержит, и может изменяться в процессе работы;

– отсутствие программного описания объектов конфигурации – разработчик может использовать либо встроенные в платформу объекты, либо объекты, созданные системой в результате визуального конструирования прикладного решения.

Прикладные решения в «1С:Предприятии 8» не кодируются целиком. Большая часть прикладного решения создается разработчиком путем визуального конструирования — создания новых объектов конфигурации, задания их свойств, форм представления, взаимосвязей и пр. Встроенный язык используется лишь для того, чтобы определить поведение объектов прикладного решения, отличное от типового, и создать собственные алгоритмы обработки данных. По этой причине модули, содержащие текст на встроенном языке, используются системой в конкретных, заранее известных ситуациях, которые могут возникнуть в процессе работы прикладного решения. Такие ситуации называются событиями. События могут быть связаны с функционированием объектов прикладного решения или с самим прикладным решением, как таковым.

Код всегда помещается в программные модули. В тех точках конфигурации, которые может потребоваться описание специфических

алгоритмов функционирования, конфигуратор предоставляет место размещения программных модулей. Алгоритмы следует оформлять в виде процедур или функций, которые будут вызваны самой системой в заранее предусмотренных ситуациях. Обзор некоторых основных аспектов встроенного языка 1С приведен в табл. 2.1.

Таблица 2.1

Базовые конструкции и аспекты встроенного языка 1С

Конструкция	**Описание**
1	**2**
Перем Адрес;	Явное объявление переменной. **Адрес** - имя переменной. Начинаться имя переменной должно с буквы, либо с символа подчеркивания. Имя может состоять из букв, цифр и символов подчеркивания.
А = 3;	Переменную во встроенном языке 1С явно можно не объявлять. Первое присвоение значения этой переменной инициирует ее создание системой.

// Чтобы написать
// комментарий -
// ставятся две косые
	Подсказки, пометки разработчика, которые помогают разобраться или вспомнить логику работы программного кода, прочая служебная информация может быть оформлена в виде комментариев. Каждая новая строка комментария начинется с символов //.

НашеЧисло = 3.7+13*4;	Переменной **НашеЧисло** присваивается числовое значение. С данными числового типа можно выполнять арифметические операции: сложение, вычитание, умножение и деление. В качестве разделителя целой и дробной части используется точка!
С = -0.765;	Числовые значения могут быть отрицательными.
ЭтоСтрока = "вуз";	Переменной **ЭтоСтрока** присваиваем строковое значение. Значение строкового типа пишется в кавычках.
ФИО="Петров" + " " + "Семен" + " " + "АЛександрович"; //результат: ФИО = "Петров Семен Александрович"	Сложение строк – конкатенация.

Продолжение табл. 2.1.

1	2
ДатаОтчета = '2021.03.30';	Переменной присваивается значение типа Дата, которое всегда записывается в одинарных кавычках.
ЧислоСекунд = '2021.01.15' – '2021.01.17'; // ЧислоСекунд = 86400	Над датами можно производить операцию вычитания, в результате рассчитывается разница между датами, выраженная в секундах. 1 сутки – это **86 400** секунд (60 сек * 60 мин * 24 ч).
НоваяДата ='2021.09.23'+ 86400; //НоваяДата = '2021.09.24'	К дате можно прибавлять и вычитать **число**. В результате к дате либо прибавится, либо отнимется число секунд.

Процедура ИмяПроцедуры (ИмяПараметра1, ИмяПараметра2,...)
// текст комментария
// тело процедуры
...
КонецПроцедуры
	За ключевым словом Процедура идет имя, затем указывают имена параметров, заключенных в круглые скобки. Между словами Процедура и КонецПроцедуры записывается тело процедуры. Имя должно начинаться с буквы или символа подчеркивания. Порядок описания (следования) процедур и функций между собой значения не имеет.

Функция ИмяФункции(Имя Параметра1, ...) // тело функции Возврат(Возвращаемое значение) ; КонецФункции	За ключевым словом Функция идет имя, затем указывают имена параметров, заключенных в круглые скобки. Далее идет тело функции. Функция должна возвращать результат в место ее вызова. Тело завершается ключевым словом КонецФункции.
Конструкции, реализующие ветвление, циклы.	
Если Оценка >= 3 Тогда Результат = "Экзамен сдан!"; Иначе Результат = "Еще надо подучить!"; КонецЕсли;	Простое условие. После слова КонецЕсли должна быть «;», это окончание конструкции оператора Если.
Результат = ?(Оценка >= 3, "Экзамен сдан!", "Еще надо подучить!");	Сокращенное Если. Краткая запись предыдущего простого условия.
Если (Доход > 100000) И (КодКатегории = 2) Тогда КонецЕсли;	Составное логическое выражение может содержать несколько простых выражений, связанных между собой И/ИЛИ.

Продолжение табл. 2.1.

1	2
Если Оценка >= 4 Тогда Результат = "Экзамен сдан, хорошие знания!"; ИначеЕсли Оценка = 3 Тогда Результат = "Сдан, но знания слабые"; Иначе Результат = "Знаний нет!"; КонецЕсли;	Множественное условие. Если первое условие не выполняется, то проверяется второе. Если ни одно из условий не выполняется то выполняется блок Иначе.
Пока Номер <= 20 Цикл КонецЦикла;	Это вариант простого цикла с заранее неизвестным числом повторений. После слова КонецЦикла должна быть «;», это окончание конструкции оператора Пока.
Для Номер = 1 По 20 Цикл КонецЦикла;	Простой цикл Для (цикл с известным числом повторений).
Для каждого СтрокаТаблицы Из Таблицы Цикл КонецЦикла;	Разновидность цикла для циклического обхода коллекций значений (универсальных коллекций значений (массивы, таблицы значений и т.п.) табличных частей справочников, документов и т.д.). При каждой итерации цикла возвращается новый элемент коллекции. Обход осуществляется до тех пор, пока не будут перебраны все элементы коллекции.
Пока <условие> Цикл Если <условие> Тогда Продолжить; КонецЕсли; КонецЦикла;	Если необходимо передать управление в начало цикла, то используется оператор Продолжить.
Пока <условие> Цикл Если <условие> Тогда Прервать; КонецЕсли; КонецЦикла;	Если необходимо произвести досрочный выход из цикла, то используется оператор Прервать. В этом случае управление передается на операторы после цикла.
Наим=Стр.Наименование; Наим=Спр["Наименование"];	Два подхода, которые используются во встроенном языке при работе с объектными сущностями (объектами, с набором свойств, методов) для обращения к свойству объекта.

Продолжение табл. 2.1.

1	2
Спр.Печать();	Вызов методов объектов производится «через точку».
Допускаются следующие конструкции: Док.Контрагент.ПолучитьОбъект().ПечатьКарточкиПоставщика();	
Использование универсальных коллекций значений:	
МойМассив = Новый Массив; МойМассив.Добавить(10); МойМассив.Добавить(«пятый»); или МойМассив = Новый Массив(1..5); МойМассив[1] = «пятый»;	Массив. Размерность можно указать или сразу при создании или сначала создать пустой массив и добавлять элементы уже позже. Могут храниться элементы разного типа.
СтруктураОтбора = Новый Структура; СтруктураОтбора.Вставить(«Склад», ВыбранныйСклад); СтруктураОтбора.Вставить(«Номенклатура», ВыбраннаяНоменклатура);	Структура. Это коллекция, состоящая из элементов, включающих пару «Ключ» и «Значение». Ключ у структуры всегда строковый.

НашеСоответствие = Новый Соответствие();
НашеСоответствие.Вставить(Номенклатура, Цена); // соответствие из 2 элементов
НашеСоответствие[Номенклатура] = ВыбраннаяНоменклатура;
	Соответствие. Это коллекция, состоящая из элементов, включающих пару «Ключ» и «Значение». Ключ у соответствия произвольный.

НашСписокЗначений = Новый СписокЗначений();	Список значений имеет 4 колонки: Пометка, Значение, Представление, Картинка. Позиция в списке задается индексом.
ТЗ = Новый ТаблицаЗначений();	Таблица значений. Хранит значения в табличном виде, может иметь любое количество колонок.
НашеДеревоЗначений = Новый ДеревоЗначений();	Дерево значений. Строки дерева значений могут иметь подчиненные строки и образовывать иерархические структуры.

Если визуальных средств разработки недостаточно разработчик создает исполняемый код на языке 1С в программных модулях. Алгоритмы на встроенном языке необходимы для того, чтобы определенным образом реагировать на действия системы или пользователя. Также в программных модулях мы можем описывать собственные методы (процедуры и функции).

Пример структуры программного модуля:

```
//*************** ОБЛАСТЬ ОБЪЯВЛЕНИЯ ПЕРЕМЕННЫХ *********************
Перем Фамилия Экспорт; //это глобальная переменная
Перем Имя, Отчество;     //это переменная модуля
Перем ФИО;       //это тоже переменная модуля и к ней можно обращаться
                 //из любой процедуры и функции нашего модуля

//************** ОБЛАСТЬ ОПИСАНИЯ ПРОЦЕДУР И ФУНКЦИЙ ***************
Процедура Процедура1()
   Перем Итог;    //Итог это локальная переменная (переменная процедуры)
Итог = Фамилия+" "+Имя+" "+Отчество;

КонецПроцедуры
Функция Функция1()
   // операторы функции
Возврат(Фамилия + " "+ Имя);
КонецФункции
//****************** ОСНОВНОЙ ТЕКСТ ПРОГРАММЫ **********************
Фамилия ="Иванов";
Имя = "Иван";
Отчество = "Иванович";
//*****************************************************************************
```

3 раздела типовой структуры программного модуля:

- область объявления переменных;
- область описания процедур и функций;
- основной текст программы.

В конкретном программном модуле любая из областей может отсутствовать.

Область объявления переменных размещается от начала текста модуля до первого оператора Процедура или оператора Функция или любого исполняемого оператора. В этом разделе могут находиться только операторы объявления переменных Перем.

Область описания процедур и функций размещается от первого оператора Процедура или оператора Функция до любого исполняемого оператора вне тела описания процедур или функций.

Область основной текст программы размещается от первого исполняемого оператора вне тела процедур или функций до конца модуля. В этом разделе могут находиться только исполняемые операторы. Область

основной текст программы исполняется в момент инициализации модуля. Обычно в разделе основной программы имеет смысл размещать операторы инициализации переменных какими-либо конкретными значениями, которые необходимо присвоить до первого вызова процедур или функций модуля.

Платформа «1С:Предприятие 8» сочетает в себе визуальные и языковые средства конфигурирования. Использование встроенного языка в системе имеет событийно-зависимую ориентацию, т.е. языковые модули используются в конкретных местах для отработки отдельных алгоритмов, настраиваемых в процессе конфигурирования. Программный код всегда помещается в модули. В Платформе существует достаточно большое количество видов модулей, каждый из которых имеет свое предназначение и особенности.

Программные модули располагаются в тех местах конфигурации, которые могут требовать описания специфических алгоритмов функционирования. Эти алгоритмы следует оформлять в виде процедур или функций, которые будут вызываться самой системой в заранее предусмотренных ситуациях (например, при открытии формы справочника, при нажатии кнопки в диалоговом окне, при изменении объекта и т.д.).

Каждый отдельный программный модуль воспринимается системой как единое целое, поэтому все процедуры и функции программного модуля выполняются в едином контексте.

Контекст выполнения модулей делится на клиентский и серверный. Кроме того, некоторые программные модули могут быть скомпилированы как на стороне клиента, так и на стороне сервера.

Виды модулей [1]:

1) Модуль приложения (управляемого или обычного):

- может содержать все 3 области;
- выполняется на стороне клиента;
- располагается в корневом разделе конфигурации.

В модуле приложения описываются процедуры (обработчики) событий, которые инициализируются при старте и окончании работы системы. Например, при начале работы приложения можно обновить какие-либо данные конфигурации (например, можно загружать курсы валют из Интернета), а при завершении работы – поинтересоваться, стоит ли вообще выходить из программы.

Кроме того, в данном модуле перехватываются события от внешнего оборудования, например, торгового или фискального. Стоит отметить, что модуль приложения выполняется только в случае интерактивного запуска приложения, то есть когда запускается окно программы. Этого не происходит, если приложение запускается в режиме com-соединения. В этом случае окно программы не создается.

В модуле приложения могут располагаться все 3 раздела – объявления переменных, описания процедур и функций, а так же основной текст программы. Модуль приложения компилируется на стороне клиента. Все переменные и методы программного модуля приложения, помеченные как экспортные, будут доступны в любом модуле конфигурации, работающем на стороне клиента. Однако, как бы ни было это заманчиво, не следует размещать здесь большое количество процедур и функций. Чем больше в данном модуле находится кода, тем длительнее время компиляции, а, следовательно, и время запуска приложения.

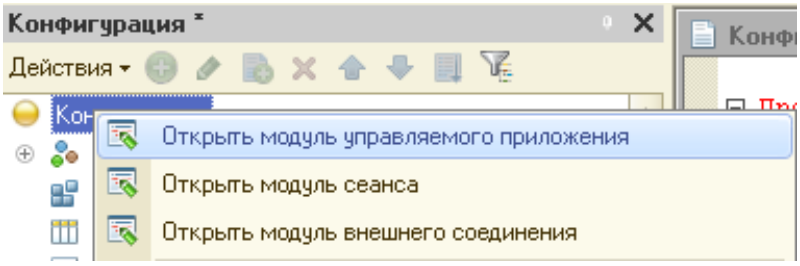


Рис. 2.1. Вызов модуля управляемого приложения

В платформе 1С 8 существует два различных модуля приложения: модуль Обычного приложения и модуль Управляемого приложения. Они срабатывают при запуске различных клиентов. Так, модуль Управляемого приложения срабатывает при запуске веб-клиента, тонкого клиента и толстого клиента в режиме управляемого приложения (рис. 2.1). А модуль обычного приложения срабатывает при запуске толстого клиента в режиме обычного приложения.

Если приложение работает и в режиме Управляемого, и в режиме Обычного приложения, то необходимо описывать процедуры-обработчики как для модуля Управляемого приложения, так и для модуля Обычного приложения.

Настройка режима запуска приложения задается в свойстве конфигурации "Основной режим запуска" (рис. 2.2).

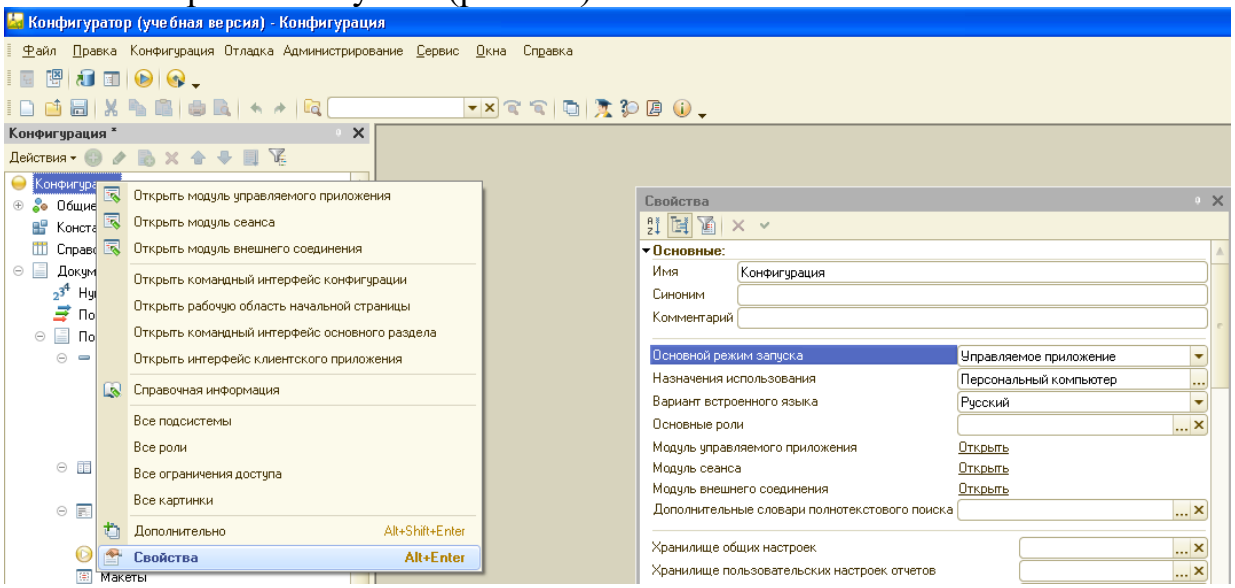


Рис. 2.2. Настройка режима запуска приложения

Как уже отмечалось выше, модуль приложения обрабатывает события запуска и завершения приложения (рис. 2.3). Для обработки каждого из этих событий в модуле приложения существует пара обработчиков Перед… и При… Отличия между заключается в следующем: при выполнении кода в обработчике Перед… действие еще не свершилось и мы можем отказаться от его выполнения. Для этого предназначен параметр Отказ. В обработчиках При.. действие уже свершилось, и отказаться от запуска приложения или выхода из него мы не можем.

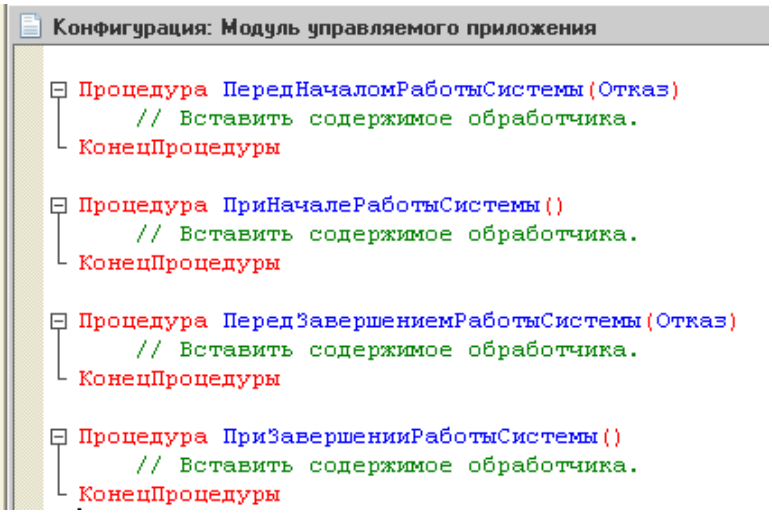


Рис. 2.3. Модуль управляемого приложения

2) Модуль внешнего соединения

- может содержать все 3 области;
- выполняется на стороне сервера;
- располагается в корневом разделе конфигурации.

Назначение модуля аналогично назначению модуля приложения. В нем идет обработка событий старта и завершения работы приложения. Модуль внешнего соединения срабатывает, когда запуск приложения происходит в режиме com-соединения. Сам процесс внешнего соединения – это процесс не интерактивный. В этом режиме происходит программная работа с информационной базой и не происходит открытия окна приложения, что накладывает определенные ограничения на использование методов, предназначенных для интерактивной работы. В этом режиме нельзя использовать вызовы диалоговых форм, предупреждений и сообщений пользователю и т.п. Они просто не будут выполняться.

Как и в модуле приложения, здесь доступны все три области: объявления переменных, описания процедур и функций, а так же основной текст программы. Главное отличие от модуля приложения заключается в том, что в режиме com-соединения вся работа с информационной базой происходит на стороне сервера, поэтому модуль внешнего соединения компилируется на стороне сервере. Соответственно в нем не доступны экспортные переменные и методы общих клиентских модулей.

3) Модуль сеанса (рис. 2.4).

- может содержать область описания процедур и функций;
- выполняется на стороне сервера;
- располагается в корневом разделе конфигурации.

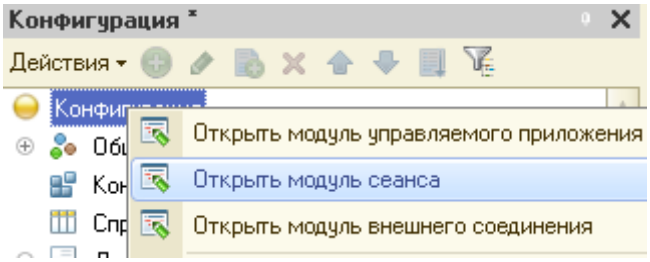


Рис. 2.4. Модуль сеанса

Это узкоспециализированный модуль, предназначенный исключительно для инициализации параметров сеанса. Почему для этого необходимо было делать собственный модуль? Его использование обусловлено тем, что само приложение может запускаться в различных режимах (что приводит к выполнению либо модуля управляемого, либо обычного приложения, либо модуля внешнего соединения), а инициализацию параметров сеанса необходимо производить вне зависимости от режима запуска. Чтобы не писать один и тот же программных код во всех трех указанных модулях, нам и потребовался дополнительный модуль, который выполняется вне зависимости от режима запуска приложения.

В модуле сеанса существует одно единственное событие «УстановкаПараметровСеанса», которое выполняется самым первым, даже раньше события модуля приложения ПередНачаломРаботыСистемы. В нем не доступны раздел объявления переменных и раздел основной программы. А так же нельзя объявлять экспортные методы. Модуль компилируется на стороне сервера.

4) Общие модули (рис. 2.5).

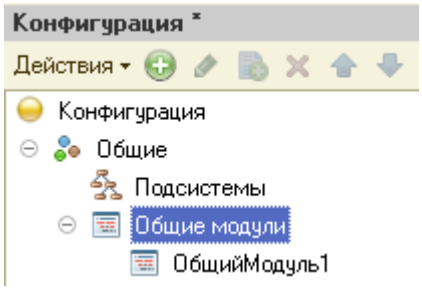


Рис. 2.5. Общие модули

– может содержать область описания процедур и функций;

– выполняется на стороне сервера или клиента (зависит от настроек модуля);

– располагается в ветке дерева объектов конфигурации «Общие» – «Общие модули».

Общие модули предназначены для описания некоторых общих алгоритмов, которые будут вызываться из других модулей конфигурации. Общий модуль не содержит областей объявления переменных и основного текста программы. В нем можно объявлять экспортные методы, доступность которых будет определяться настройками модуля (на какой стороне он выполняется: на стороне сервера или клиента). В связи с тем, что раздел описания переменных не доступен, определять глобальные переменные в общих модулях нельзя. Для этого можно использовать модуль приложения.

5) Модуль формы

– может содержать все 3 области;

– выполняется на стороне сервера и клиента.

Модуль формы предназначен для обработки действий пользователя с данной формой (обработка события нажатия кнопки, изменения реквизита формы и т.д.). Так же существуют события связанные непосредственно с самой

формой (например, ее открытие или закрытие). Модули управляемых и обычных форм различаются, прежде всего, тем, что модуль управляемой формы четко разделяется на контекст. Каждая процедура или функция должна иметь директиву компиляции. Если же директива компиляции не указана, то данная процедура или функция выполняется на стороне сервера. В обычной форме весь код исполняется на стороне клиента.

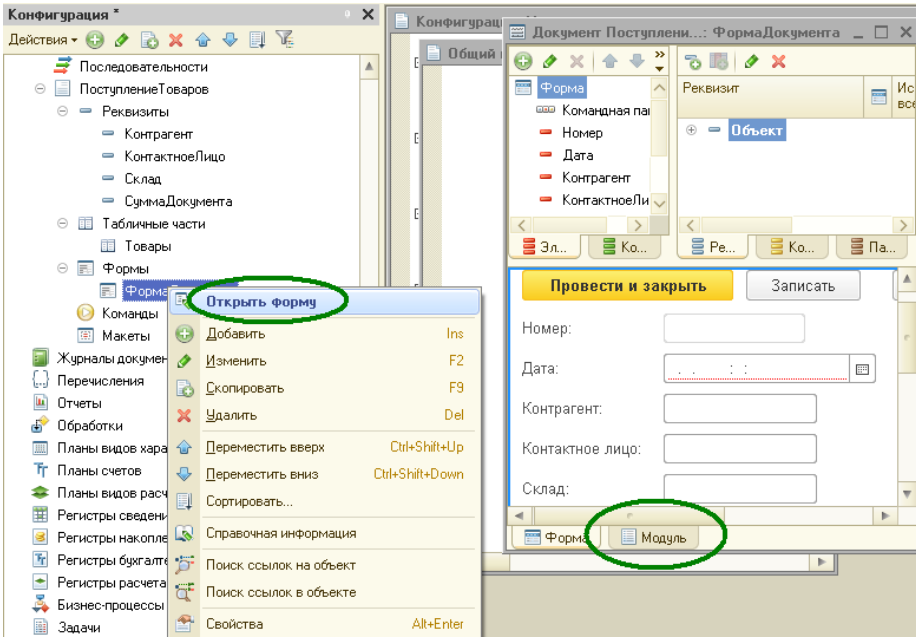


Рис. 2.6. Общие модули

6) Модуль объекта (рис. 2.7).

- может содержать все 3 области;
- выполняется на стороне сервера.

Данный модуль имеется у большинства объектов конфигурации и предназначен, в общем случае, для обработки событий, непосредственно связанных с объектом. Например, события записи и удаления объектов, проверка заполнения реквизитов объекта, проведение документа и т.д.

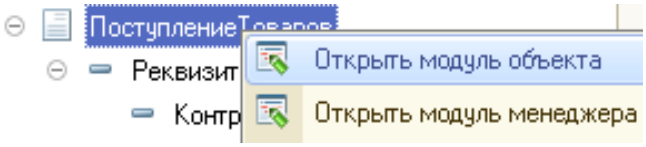


Рис. 2.7. Общие модули

Некоторые события модуля объекта дублируют события модуля формы. Например, события, связанные с записью. Однако следует понимать, что события модуля формы будут выполняться исключительно в конкретной форме объекта, то есть при открытии конкретной формы. А события модуля объекта будут вызываться в любом случае, даже в момент программной работы с объектом. Поэтому, если необходимо методы, связанные с объектом без привязки к конкретной форме объекта, то лучше использовать для этого модуль объекта.

7) Модуль менеджера объекта

- может содержать все 3 области;
- выполняется на стороне сервера.

Модуль менеджера объектов появился только начиная с версии 1С 8.2. Модуль менеджера существует у всех прикладных объектов и предназначен для управления этим объектом как объектом конфигурации. Модуль менеджера позволяет расширить функциональность объекта за счет введения (написания) процедур и функций, которые относятся не к конкретному экземпляру объекта базы данных, а к самому объекту конфигурации. Модуль менеджера объектов позволяет размещать общие процедуры и функции для данного объекта и обращаться к ним из вне, например, из обработки (конечно, если эта процедура или функция будет с ключевым словом Экспорт). Это дает упорядочивание процедур по объектам и хранения их в отдельных местах - Модулях менеджеров объектов. Примеры использования процедур и функций Модуля менеджеров объектов: первоначальное заполнение отдельных реквизитов справочника или документа по определенным условиям, проверка заполнения реквизитов справочника или документа по определенным условиям и т.д.

8) Модуль команды (рис. 2.8).

- может содержать раздел описания процедур и функций;
- выполняется на стороне клиента.

Команды – это объекты, подчиненные прикладным объектам или конфигурации в целом.

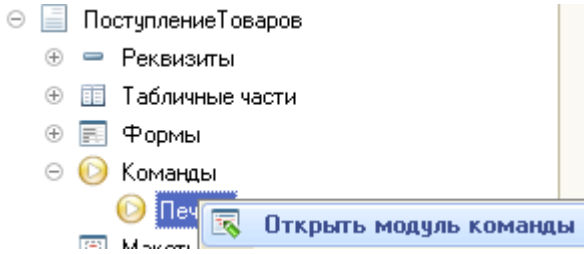


Рис. 2.8. Модуль команды

У каждой команды есть модуль команды, в котором можно описать предопределенную процедуру ОбработкаКоманды() для выполнения этой команды.

Необходимо помнить правила видимости экспортируемых переменных, процедур и функций различных модулей:

1) В общем модуле недоступны экспортируемые переменные, процедуры и функции модуля приложения (модуля внешнего соединения).

2) В модуле приложения (модуле внешнего соединения) доступны экспортируемые процедуры и функции общих модулей.

3) В общих модулях доступны экспортируемые процедуры и функции других общих модулей.

4) В модулях прикладных объектов и модулях форм доступны экспортируемые переменные, процедуры и функции модуля приложения (модуля внешнего соединения), а также экспортируемые процедуры и функции общих модулей.

5) Если у формы назначен основной реквизит, то контекст модуля формы содержит дополнительные свойства и методы, связанные с основным реквизитом. Например, в модуле формы элемента справочника Номенклатура доступны свойства и методы объекта СправочникОбъект.Номенклатура.

Проиллюстрируем применение первых четырех правил на следующей схеме (рис. 2.9) [1].

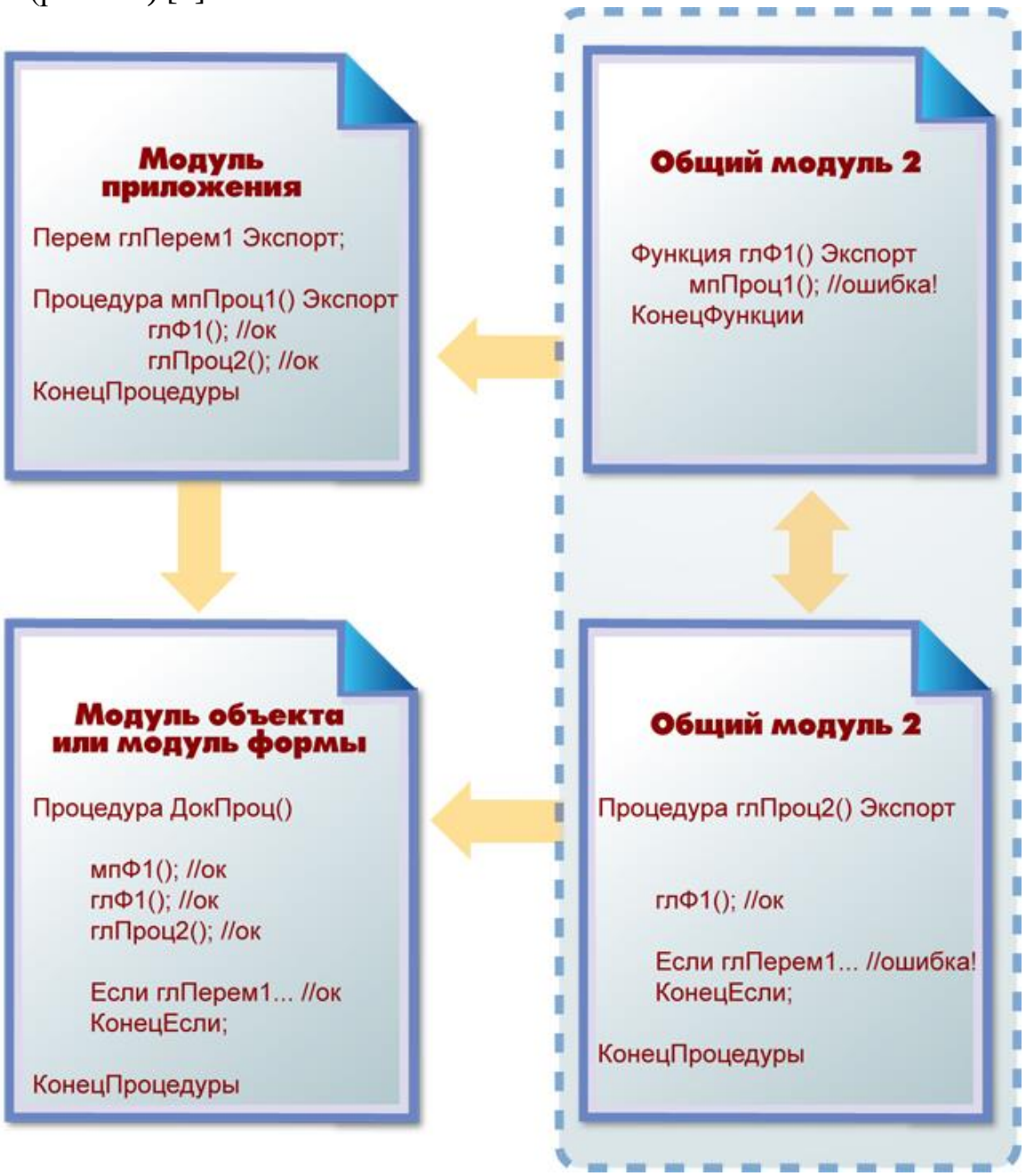


Рис. 2.9. Схема видимости экспортируемых переменных, процедур и функций различных модулей

Стрелки на схеме обозначают, что модуль *предоставляет* другим модулям возможность обращаться к своим экспортируемым переменным, процедурам и функциям. Напомним, что в общих модулях не может быть объявления переменных.

Тема 3. Подходы к хранению информации в приложениях на платформе «1С:Предприятие»

Подходы к хранению информации в приложениях на платформе «1С:Предприятие». Хранение информации объектных и необъектных сущностей; иерархической информации; информации, имеющей привязку ко времени; вспомогательной информации (в объекте «ХранилищеЗначений») и др.

Задачи хранения информации встают при создании практически любых решений в области автоматизации. Эти вопросы затрагивают и аспекты предназначения хранимой информации и разнообразные технологические вопросы, которые приходится решать для достижения оптимального соотношения таких показателей, как: объем, надежность, функциональность, быстродействие.

Возникают эти вопросы на всех этапах жизненного цикла информации: запись информации, хранение информации, получение информации, удаление информации. Наличие ряда противоречий при создании бизнес-приложений обуславливает сложность решения этих задач, в частности: необходимо обеспечить и удобство представления логики взаимодействия сущностей в информационной модели и условия для хранения больших объемов информации, обеспечивая повышенные требования к широкой функциональности и высокой производительности доступа к этим данным.

Рис. 3.1. иллюстрирует взаимную противоречивость таких требований: например, чем больше объем, тем в общем случае сложнее с обеспечением скорости доступа к информации.

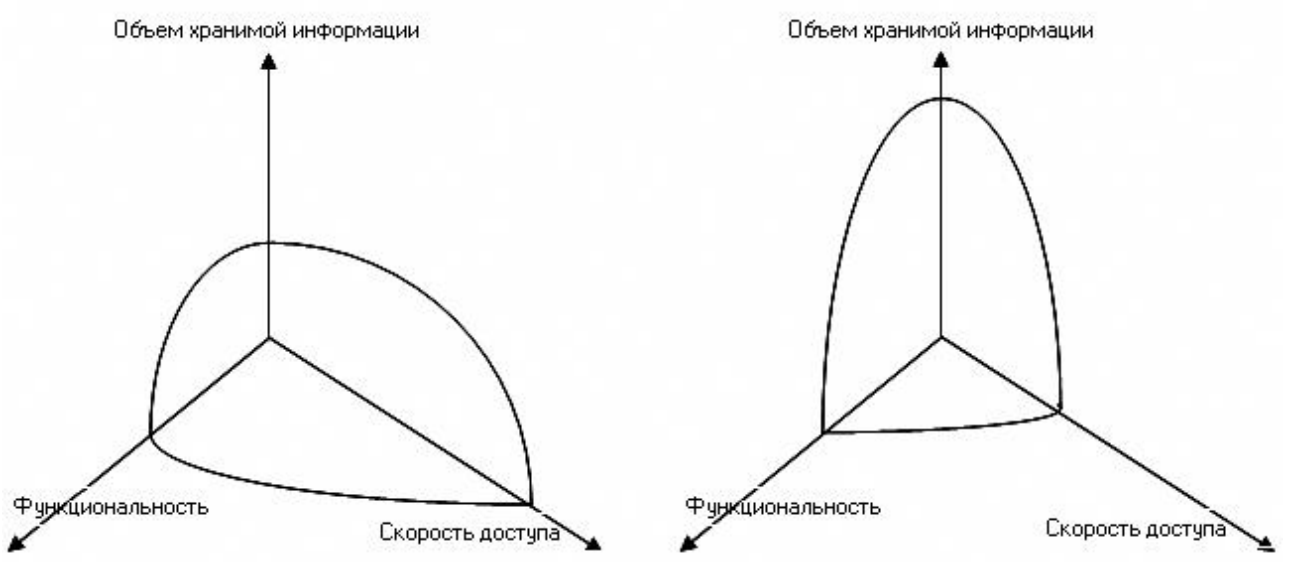


Рис. 3.1. Схема противоречий между требованиями к хранимой информации

Например, хранение информации о происходящих в жизни автоматизируемого предприятия событиях в виде набора неструктурированных текстов – идеальное решение с точки зрения простоты и скорости регистрации. Однако последующая обработка этой информации для составления каких бы то ни было аналитических отчетов в таком случае будет сопряжена с

колоссальными временными затратами. Потому что как для решения задачи поиска информации обо всех продажах предприятия, так и для поиска информации о продажах одному покупателю необходимо будет пересмотреть данные всех текстов.

И чем больше функциональных возможностей потребуется на этапе получения информации, тем дольше будут работать соответствующие обработки, зачастую по нескольку раз просматривая одну и ту же информацию, но уже с разными целями.

Решить проблему соотношения удобства представления и манипулирования объектами, отражающими прикладные сущности, с должной надежностью и эффективностью обработки больших объемов информации этих объектов в базе данных позволяет ***объектно-реляционная парадигма системы «1С:Предприятие».*** Схематично она представлена на рис. 3.2.

Объекты и процессы прикладной области отражаются в решении посредством объектов конфигурации. Таким образом, объекты конфигурации обеспечивают удобство манипулирования данными, характеризующими прикладные объекты и процессы [1].

Рис. 3.2. Схема представления и манипулирования данными в системе «1С:Предприятие»

В системе «1С:Предприятие» все возможные к применению в решениях прикладные объекты прототипированы. Каждый прототип отвечает за отражение в прикладном решении определенной совокупности объектов или процессов прикладной области, имеющих схожие поведенческие характеристики и схожую роль в общей картине решения. Примерами прототипов являются справочники, документы, регистры различных видов и так далее.

В рамках средств платформы для каждого прототипа уже предопределены:

– оптимальная для большинства задач структура хранения информации в реляционной базе данных;

– набор средств встроенного языка для манипулирования этой информацией;

– методы, свойства, события и типовые для решаемых задач операции;

– способы отображения и редактирования;
– средства регулирования прав доступа и т. д.

Хранение информации объектных и необъектных сущностей.

К объектным данным относятся данные справочников, документов, планов видов характеристик, планов счетов, планов видов расчета, бизнес-процессов, задач, планов обмена.

К необъектным данным относятся данные регистров сведений, регистров накопления, регистров бухгалтерии, регистров расчета, перерасчетов, последовательностей и констант.

При выборе прототипов объектов для хранения информации одним из типичных вопросов, возникающих при неочевидных случаях, является выбор между объектными и необъектными данными. Например, между регистром сведений и справочником.

Для принятия решения рекомендуется обращать внимание на природу данных предметной области. Если они обладают некоей «сутью», несмотря на смену значений отдельных свойств этого объекта, то предпочтение следует отдавать определению этих данных как объектных.

Например, сотрудник может сменить фамилию, имя, паспорт, цвет глаз и т.д., однако при этом он останется все тем же сотрудником.

Таким образом, типичный пример идентификации сотрудников предприятия – как данные объектные и подлежащие учету посредством справочника.

Однако следует учитывать, что выбор вида объекта конфигурации не должен производиться для каждой сущности отдельно. Необходимо анализировать наличие и остальных сущностей в комплексе всей прикладной задачи. Причем желательно не только на текущий момент, но и с учетом развития задачи в будущем. Например, как определились выше, состав сотрудников предприятия целесообразно хранить посредством объекта конфигурации Справочник (рис. 3.3).

Справочник "Сотрудники"			
Ссылка	Код	Наименование	...

Рис. 3.3. Таблица справочника «Сотрудники»

Взглянув шире на проблему, мы можем заметить тут наличие двух взаимосвязанных сущностей:

«Физические лица» как отражение реальных людей, с которыми предстоит иметь дело в рамках решения;

«Сотрудники подразделений предприятия», причем один и тот же человек может быть сотрудником нескольких подразделений предприятия, выполняя в них различные задачи (должности, круг обязанностей и проч.).

Причем вторая сущность как объект нигде не фигурирует, просто нужно помнить «кто есть кто». Тогда хранение сотрудников подразделений может быть реализовано посредством регистра сведений (рис. 3.4).

Информация о сотрудниках и их трудовых договорах: присмотревшись и тут можно также выделить две сущности:

– «Физическое лицо»;

– «Сотрудник – трудовой договор» как отражение юридических отношений данного лица с организацией, имеющее при этом объектную природу, поскольку впоследствии ссылка на трудовой договор будет фигурировать во многих документах и регламентированных отчетах.

Рис. 3.4. Связь таблицы справочника и таблицы регистра сведений

В этом случае для хранения данных следует использовать два справочника, один из которых (справочник ТрудовыеДоговораСотрудников) будет подчинен другому (справочник ФизическиеЛица), см. рис. 3.5.

Таким образом, выбор каждого из вариантов хранения информации определяется тем, какую сущность предметной области будет отражать тот или иной объект. При этом разработчикам рекомендуется также и имя объекта определять так, чтобы оно максимально отражало описываемую сущность. Это позволит впоследствии обеспечить правильное восприятие и использование объекта.

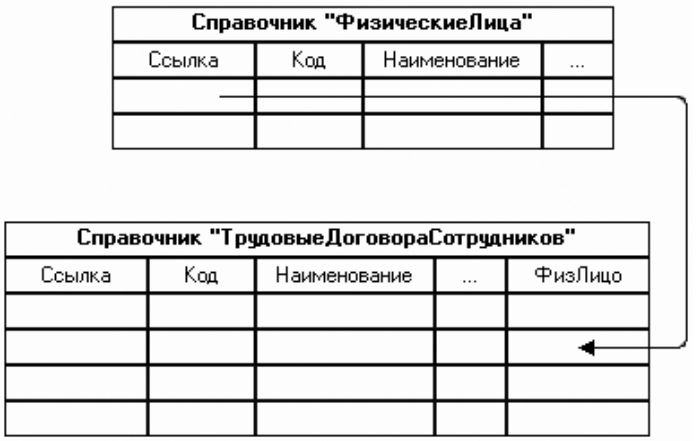


Рис. 3.5. Взаимосвязь справочника с подчиненным справочником

Хранение информации в самих объектах или в других объектах

Необходимо помнить, что любой объект – это единая сущность с точки зрения манипулирования данными. То есть при любых операциях с объектом (чтение, запись, модификация) происходит обращение к информации базы данных, касающейся всего объекта.

Например, справочник ДоговорыКонтрагентов имеет подчиненные объекты: два реквизита (ДатаПоставки и ВидДоговора) и табличную часть Спецификация. В свою очередь, в состав табличной части тоже входит ряд реквизитов.

Хранение этой информации в базе данных будет реализовано системой следующим образом (рис. 3.6) [2]:

В основной таблице справочника будет храниться информация в следующих полях:

- *Ссылка,*
- *Код,*
- *Наименование,*
- *Пометка удаления,*
- *Предопределенный,*
- *Родитель,*
- *Владелец,*
- *ЭтоГруппа,*
- *ДатаПоставки,*
- *ВидДоговора.*

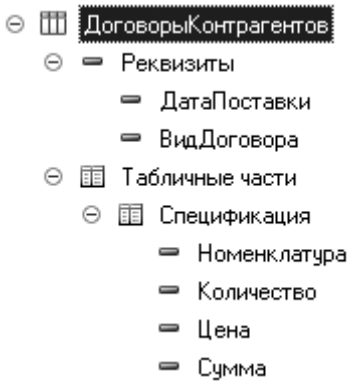


Рис. 3.6. Структура справочника «ДоговорыКонтрагентов»

Информация табличной части *Спецификация* – в отдельной таблице, содержащей следующие поля:

- *Ссылка* (значение этого поля равно значению поля *Ссылка* основной таблицы справочника)*;*
- *НомерСтроки;*
- *Номенклатура;*
- *Количество;*
- *Цена;*
- *Сумма.*

Любое обращение к объекту документа приведет к тому, что из базы данных будет считана информация, соответствующая данному объекту, из обеих таблиц. При этом не важно, как именно выполнялось обращение: посредством метода ПолучитьОбъект() из ссылки (листинг) или при открытии формы, основным реквизитом которой является объект договора.

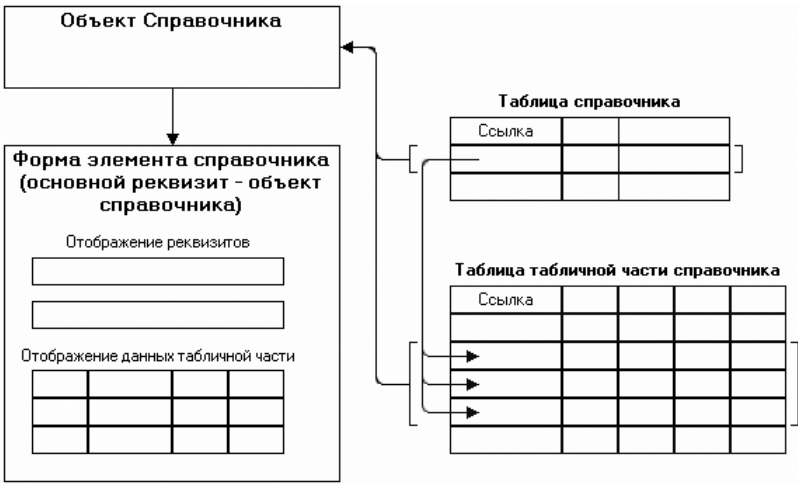


Рис. 3.7. Считывание из базы данных будет считана информация, соответствующая данному объекту, из обеих таблиц

В последнем случае, кстати, не важно, будут отображаться на форме данные табличной части элемента или нет (рис. 3.7).

Объект в любом случае считывается целиком, поскольку при открытии формы платформа также создает объект справочника, обеспечивая тем самым целостность изменений, вносимых в данные объекта как интерактивно, так и программно, в модуле формы (рис. 3.8).

Данную особенность нужно иметь в виду для сущностей, при работе с которыми часто необходим доступ именно к объектам, а не ссылкам (например, объекты, которые часто модифицируются пользователями). При чтении объекта или его модификации данные будут считываться и записываться по объекту целиком. Таким образом, чем меньше этих данных будет, тем быстрее будут выполняться операции, меньше будет продолжительность блокировок и т. д.

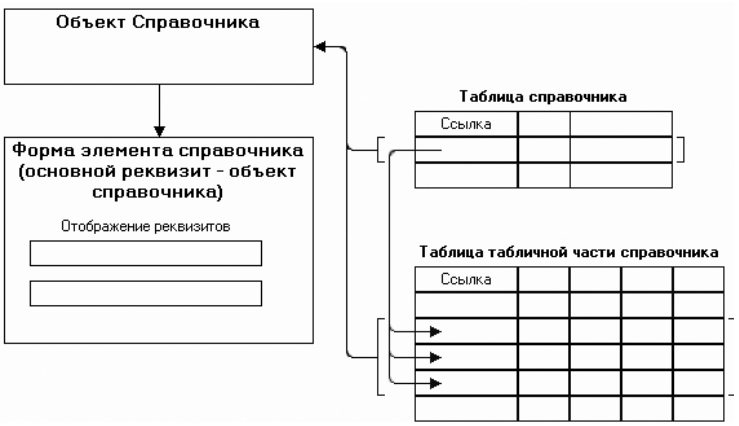


Рис. 3.8. Считывание объекта целиком при открытии формы

Из этого можно сделать вывод, что информацию, которую предполагается хранить в объектах, следует анализировать на предмет разделения на *активно используемую* и *неактивно используемую*. Активно используемую информацию можно хранить в реквизитах самого объекта или его табличных частей. Неактивно используемую информацию об объекте можно хранить не в самом объекте, а посредством других информационных структур (например, регистров сведений или подчиненных справочников). Однако при этом придется самостоятельно заботиться о поддержании целостности изменений при модификации данных объекта.

Для вышеприведенного примера, если информацию спецификаций можно считать «неактивно используемой», ее хранение можно организовать посредством, например, регистра сведений СпецификацииДоговоров или подчиненного справочника СпецификацииДоговоров (рис. 3.9).

Так же осторожно нужно подходить к использованию строковых реквизитов неограниченной длины. Крайне не рекомендуется использовать их для хранения информации, объем которой может быть достаточно большим или непрогнозируемым.

Также не рекомендуется хранить в реквизитах активно используемых объектов картинки и образы файлов (использование полей типа *ХранилищеЗначения*), если заранее известно, что размер их будет велик. Для сохранения подобной информации следует использовать альтернативные методы.

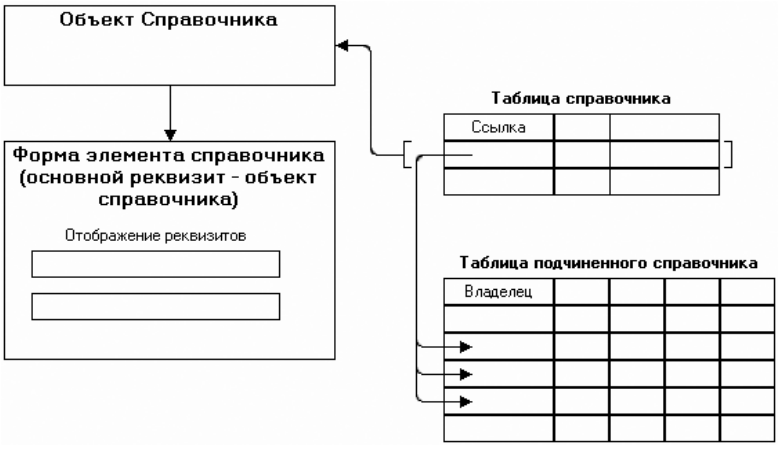


Рис. 3.9. Считывание из базы данных информации, соответствующей объекту, из обеих таблиц

Хранение иерархической информации.

Если у структуры четко прослеживаются отношения «один ко многим». ее принято считать иерархической.

Например, «один руководитель – много подчиненных» (рис. 3.10).

Рис. 3.10. Пример одноуровневой иерархии

Уровней иерархии может быть больше одного. «один руководитель, в подчинении которого находятся руководители подразделений, в подчинении которых находятся подчиненные» – пример трехуровневой иерархической структуры (рис. 3.11). Ну а в общем случае уровней может быть значительно больше [1].

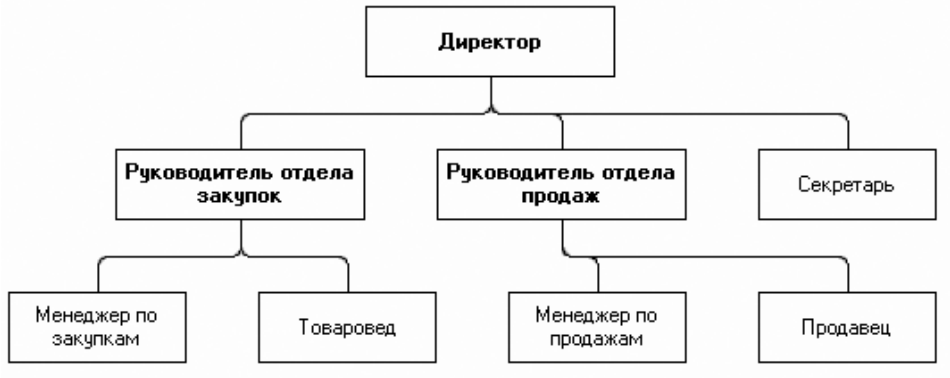


Рис. 3.11. Пример многоуровневой иерархии

Для «экономного» хранения информации об иерархичности хранимых данных достаточно в каждом нижестоящем элементе хранить ссылку на непосредственно стоящий над ним элемент.

Таким образом, информацию об иерархии можно хранить не только в виде схем, но и в виде таблиц. Например, табличный вариант представления информации о многоуровневой иерархии руководства приведен в табл.3.1.

Каждая строка таблицы содержит информацию по одной персоне. Для каждой строки таблицы упоминание некоей персоны в поле Руководитель означает, что данный сотрудник непосредственно подчинен именно этому руководителю (а в информации по этому руководителю будет указано, кому он, в свою очередь, подчинен). В строке, отражающей информацию по директору, поле Руководитель пустое. Это значит, что в данной схеме у директора нет руководителей. Еще говорят «директор находится на корневом уровне иерархии».

Данный прием применяется при хранении иерархической информации объектов прикладной области на уровне объектов базы данных системы «1С:Предприятие». То есть «нижестоящие» элементы должны помнить ссылку на «вышестоящие» элементы иерархии.

Таблица 3.1

Табличный способ хранения иерархической информации

Сотрудник	Руководитель
Директор	
Руководитель отдела закупок	Директор
Менеджер по закупкам	Руководитель отдела закупок
Товаровед	Руководитель отдела закупок
Руководитель отдела продаж	Директор
Менеджер по продажам	Руководитель отдела продаж
Продавец	Руководитель отдела продаж
Секретарь	Директор

При разработке бизнес-решений разработчику чаще всего приходится иметь дело с иерархическими структурами при организации хранения условно-постоянной информации (справочники, планы видов характеристик), реже – при работе с регистрацией событий.

В отношении хранения условно-постоянной информации могут возникать следующие ситуации:

– иерархия объектных данных одной сущности;

– хранение иерархии необъектных данных внутри объекта;

– иерархия объектных данных разных сущностей;

– хранение иерархии необъектных данных вне объекта.

Хранение иерархии данных одной сущности. Для этого используются ***иерархические справочники***. Чтобы справочник сделать иерархическим, необходимо на закладке *Иерархия* указать признак *Иерархический справочник* и выбрать вид иерархии. Дальнейшую работу возьмет на себя платформа. В состав основной таблицы справочника будет добавлено поле *Родитель* для хранения в записях иерархически подчиненных элементов ссылок на элементы, которым они непосредственно иерархически подчиняются.

Кроме того, при записи и модификации данных в справочнике Пользователи платформа будет отслеживать ограничение по количеству уровней иерархии, не позволяя реализовать попытки их превышения, выдавая соответствующие предупреждения.

Помимо этого, в расширения соответствующих форм и элементов форм будут включены средства, облегчающие решение задач отображения информации справочника в иерархическом виде.

Хранение иерархии необъектных данных внутри объекта. Практически все объекты системы «1С:Предприятие», предназначенные для хранения данных объектных сущностей (справочники, документы, планы видов характеристик, планы видов расчета и т. д.), имеют возможность хранить свою информацию не только в виде значений реквизитов, но и в составе подчиненных ***табличных частей***. Однако необходимо иметь в виду, что информация строк подчиненной табличной части не имеет своей объектной сущности, т.е. нельзя будет создать реквизиты в других объектах, ссылающиеся на строки табличной части. Для задач, когда ссылка на такие сведения смысла не имеет, хранение множества подчиненных данных, характеризующих данный объект, внутри самого объекта

может быть весьма удобным. Пример, справочник *ПоступлениеТоваров* и табличная часть *Состав*.

Например, при оформлении заказа покупателя необходимо дать возможность пользователю вводить не только информацию о том, какие номенклатурные позиции, в каком количестве и по какой цене желает приобрести покупатель, но и еще произвольное количество дополнительных пожеланий по любой номенклатурной позиции в свободной форме, с возможностью отметки, насколько эти пожелания действительно важны (табл. 3.2).

Таблица 3.2

Пример иерархической информации

Заказ	Номенклатура	Количество	Цена	Сумма	Пожелания
Заказ № 3 от 12.02.10	Лазерный принтер Canon LBP-810	2	200	400	Белого цвета
					Дополнительно упаковать в гофро-тару
	Телефон LG W7200	10	30	300	Без гарнитуры
					Доставку приурочить к 8 марта
					Цвет любой, только не черный

Реализация хранения этой информации может быть организована следующим образом. Для хранения информации о заказанных товарах с количественно-суммовыми характеристиками можно использовать табличную часть *Состав*. Но дополнительные пожелания будет неудобно хранить в рамках этой табличной части. Как показано в таблице выше, к одной номенклатурной позиции может предъявляться более одного требования и форма информации этих требований более «свободная», нежели указание цены, количества и суммы.

В этом случае для хранения информации о дополнительных пожеланиях покупателя, если эту информацию нужно хранить внутри объекта, можно использовать еще одну табличную часть *ДополнительныеТребования*. Реквизитами табличной части *ДополнительныеТребования* будут *Требование*, *Важно* и *Номенклатура* (рис. 3.12). Соответствие с информацией табличной части *Состав* будет организованно именно по номенклатурным позициям.

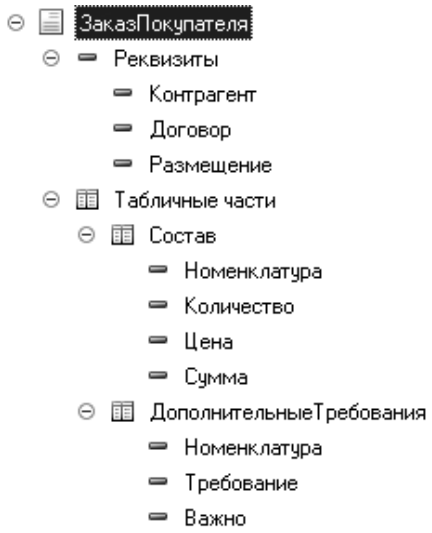


Рис. 3.12. Структура документа «ЗаказПокупателя»

Иерархия объектных данных разных сущностей. Используются ***подчиненные справочники***. Пример: справочник *Сотрудник* и подчиненный справочник *ТрудоваяДеятельность*.

Допустим, необходимо организовать хранение информации по предыдущей трудовой деятельности каждого сотрудника компании. Сами сотрудники признаны сущностями

объектного типа, и для хранения их информации в рамках решения создан справочник Сотрудники.

Как решить задачу? С одной стороны, данная информация непосредственно связана с сотрудником и могла бы быть реализована посредством включения табличной части ТрудоваяДеятельность в состав объектов справочника Сотрудники.

Но к выбору такого варианта нужно подходить осторожно. Подчиненные табличные части предназначены для учета только необъектных сущностей, то есть при таком решении в других объектах системы невозможно будет использовать поля или реквизиты, соответствующие понятию «место предыдущей работы конкретного сотрудника».

Если же необходимость в такой объектной сущности есть или может появиться, то решение видится в использовании подчиненного справочника ТрудоваяДеятельность. В его элементах можно хранить информацию о должности, месте работы, дате начала деятельности, дате окончания деятельности по каждому эпизоду предыдущей работы сотрудника.

Впоследствии эта сущность может быть использована, например, для организации структуры регистра накопления СтажРаботы или для заполнения реквизитов соответствующих документов.

Хранение иерархии необъектных данных вне объекта. Для хранения необъектных данных, соответствующих определенным комбинациям значений измерений, предназначены ***регистры сведений***.

В торговой компании покупатели закреплены за менеджерами, их обслуживающими. Сами менеджеры компании, в свою очередь, относятся к тому или иному департаменту. Необходимо иметь представление, кто за кем закреплен на каждый момент времени.

В данном случае вариант хранения информации в регистре сведений предпочтителен тем, что автоматически будет обеспечена уникальность значений измерений. То есть в регистре *ЗакреплениеПокупателей* невозможно будет создать две записи с одним и тем же покупателем (соответственно, невозможно одного покупателя закрепить за несколькими менеджерами). Аналогично в отношении регистра *ЗакреплениеМенеджеров*.

Кроме того, регистры сведений позволяют хранить не только статические варианты иерархических структур, но и изменяющиеся во времени. Если указанные регистры сведений сделать периодическими, то впоследствии можно будет легко хранить динамику картины отношений. То есть отражать в системе все изменения, касающиеся закреплений (например, менеджер со всей своей клиентской базой передан другому департаменту). И так же легко получать информацию о состоянии модели закреплений на любой момент времени.

Также для ситуаций, когда в готовое (и эксплуатируемое) решение приходится вводить новые сущности, добавление новых измерений к регистру сведений позволяет легко решать задачи усложнения «взаимоотношений» данных.

Например, начиная с какого-то момента, закрепление покупателей за менеджерами продолжает оставаться однозначным, но только в рамках

конкретных проектов. То есть в основном проекте *Торговля* за обслуживание некоего покупателя по-прежнему отвечает тот же менеджер, но в проекте *Информационное сопровождение* – совсем другой (да еще и из другого департамента).

С точки зрения классификации ситуации хранения данных такое изменение «революционно», отдельные его части можно даже рассматривать как переход от схемы «один ко многим» к схеме «многие ко многим». Однако сама реализация подобного «поворота событий» очень проста.

В состав регистра *ЗакреплениеПокупателей* достаточно добавить измерение *Проект*. И поскольку система обеспечивает в регистре сведений уникальность комбинаций значений измерений, задача уже фактически решена.

Остается лишь согласовать с пользователями, будет ли в регистре сведений *ЗакреплениеПокупателей* пустая ссылка на проект (для старых данных) с прикладной точки зрения восприниматься проектом основной деятельности предприятия – *Торговля*, либо необходимо произвести соответствующую обработку для заполнения этих данных.

Так же легко в состав регистра сведений может быть введена дополнительная информация, характеризующая конкретные комбинации значений измерений. Это реализуется посредством добавления ресурсов. Например, для регистра *ЗакреплениеПокупателей* могут быть введены такие ресурсы, как *Лояльность*, *ЧастотаКонтактов*, *СуммаКвотыПродаж* и так далее. Для регистра *ЗакреплениеМенеджеров* могут быть введены такие ресурсы, как *БазовыйОклад*, *ПроцентПремии*, *Эффективность* и тому подобное.

Хранение информации, имеющей привязку ко времени. Информацию, хранящуюся в привязке ко времени, чаще всего разделяют на содержащую объектные данные, необъектные данные.

В задачах хранения объектных данных, привязанных ко времени, речь обычно идет о регистрации неких событий и всего, что с ними связано, например, в хозяйственной жизни предприятия. Чаще всего для хранения такой информации выбираются объекты *Документ*. Функциональность объектов документов, поддерживаемая на уровне платформы «1С:Предприятие»: заполнение, запись, проведение, формирование движений по регистрам, расположение на оси времени, пометка на удаление, удаление.

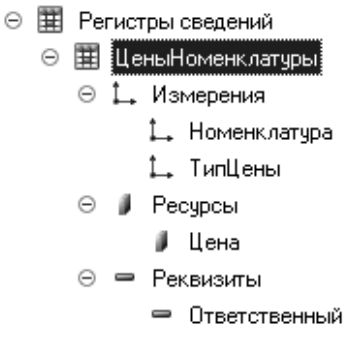


Рис. 3.13. Периодический регистр сведений

Для решения учетных и аналитических задач кроме учета событий важно, чтобы система в определенных ситуациях учитывала показатели, которые изменяются или при обработке вышеуказанных событий, или вследствие других действий пользователя. Использование периодических регистров сведений позволяет достаточно быстро создать решение, большую часть функциональности которого поддерживает сама платформа, например, получение информации об актуальных значениях цен на некоторый момент времени (рис. 3.13).

Оба варианта (и использование документов, и использование периодических регистров сведений) для хранения данных, имеющих привязку ко времени, не противоречат друг другу. Они могут сосуществовать в одном решении и обеспечивать каждый свою функциональность.

Хранение вспомогательной информации в объекте «ХранилищеЗначений».

При решении вопросов хранения информации желательно различать информацию, обеспечивающую основную бизнес-логику решения и другую, вспомогательную информацию. Данные основной бизнес-логики активно используются в учетных и аналитических механизмах. Для них важны такие вопросы, как индексирование, упорядочивание в запросах и выборках, суммирование, поиск и т. п. Данные же вспомогательной информации нужно просто хранить. Все возможное манипулирование этой информацией сводится к операциям записи и чтения.

Платформа системы «1С:Предприятие» предоставляет возможность хранения некоторой вспомогательной информации в базе данных посредством полей типа *ХранилищеЗначения*. Особенностью использования таких полей является то, что база данных «не обязана ничего знать» о природе хранимой там информации. Это могут быть картинки, файлы, таблицы, данные примитивных типов, да что угодно. База данных отвечает лишь за хранение этой информации, не обеспечивая больше никакой функциональности. При этом еще говорят, что тип значения *ХранилищеЗначения* позволяет хранить значения различных типов в сериализованном виде, то есть в виде, который позволяет записывать данные и восстанавливать их. Важной особенностью хранилища значений является возможность хранения данных в сжатом виде. Это позволяет существенно сократить размеры базы данных при необходимости хранения больших объемов информации.

С прикладной точки зрения в полях типа *ХранилищеЗначения* можно хранить, например, фотографии сотрудников и т.п.

При этом следует осмотрительно подходить к решению, что именно требуется хранить в базе данных. Большой объем хранимой вспомогательной информации может привести к тому, что административные операции (например, создание резервной копии базы данных) выполняются медленно из-за большого объема базы данных.

Особенно аккуратно нужно относиться к возможности использования полей типа *ХранилищеЗначения* в составе объектов (например, справочников, документов), активно использующихся при реализации основной бизнес-логики. Ведь данные объектов считываются целиком при обращении к ним. Поэтому, например, вопрос хранения тех же фотографий лучше решать не в составе справочника *Сотрудники*, а в отдельном подчиненном справочнике или регистре сведений. Тогда будет сохранена возможность идентификации соответствия изображений сотрудникам, к которым они относятся. И в то же время при использовании объектов справочника *Сотрудники* для решения других задач выполняемые операции не будут замедляться из-за необходимости считывания из базы данных больших объемов информации

Тема 4. Задачи оперативного учета и их реализация в корпоративных информационных системах на платформе «1С:Предприятие»

Задачи оперативного учета в корпоративных информационных системах. Виды регистров накопления: регистры остатков и регистры оборотов. Структура регистра накопления: измерения, ресурсы, реквизиты. Регистры сведений: структура регистра сведений. Периодические регистры сведений. Подчинение записей регистратору. Проектирование структуры регистров сведений.

Оперативный учет: задачи, решаемые регистрами накопления. Оперативный учет – учет, позволяющий максимально быстро получать информацию о значениях показателей, учитываемых в автоматизируемой системе. При автоматизации учета движения средств чаще приходится сталкиваться с ситуацией, когда в момент регистрации изменений показателя фиксируется не конечное итоговое значение показателя, а его приращение. А вот при получении данных из системы учета уже требуются накопленные (итоговые) значения показателей. Пример: количество товаров на складе, сумма в кассе. Такие показатели называют *показателями накопления.*

Регистры накопления — это прикладные объекты конфигурации. Они составляют основу механизма учета движения средств (финансов, товаров, материалов и т. д.), который позволяет автоматизировать такие направления, как складской учет, взаиморасчеты, планирование.

Регистр накопления образует многомерную систему измерений и позволяет «накапливать» числовые данные в разрезе нескольких измерений. Например, в таком регистре можно накапливать информацию об остатках товаров в разрезе номенклатуры и склада, или информацию об объемах продаж в разрезе номенклатуры и подразделения компании. Информация в регистре накопления хранится в виде записей, каждая из которых содержит значения измерений и соответствующие им значения ресурсов. Измерения регистра описывают разрезы, в которых хранится информация, а в ресурсах регистра накапливаются нужные числовые данные.

Информация о приращениях показателей (а они могут быть как положительными, так и отрицательными) вносится в регистр накопления только посредством движений, то есть посредством наборов записей регистра, подчиненных документу-регистратору. Этим обеспечивается обоснованность регистрации изменений показателей – регистрацией событий, приводящих к этим изменениям.

Показатели накопления имеют различный прикладной смысл. Различают следующие виды накапливаемых показателей [2]:

– показатели остатков,

– оборотные показатели.

Упрощенно можно сравнить регистр накопления с неким «черным ящиком», «на вход» которого подаются значения приращений, а «на выходе» можно получить накопленные значения приращений (рис. 4.1).

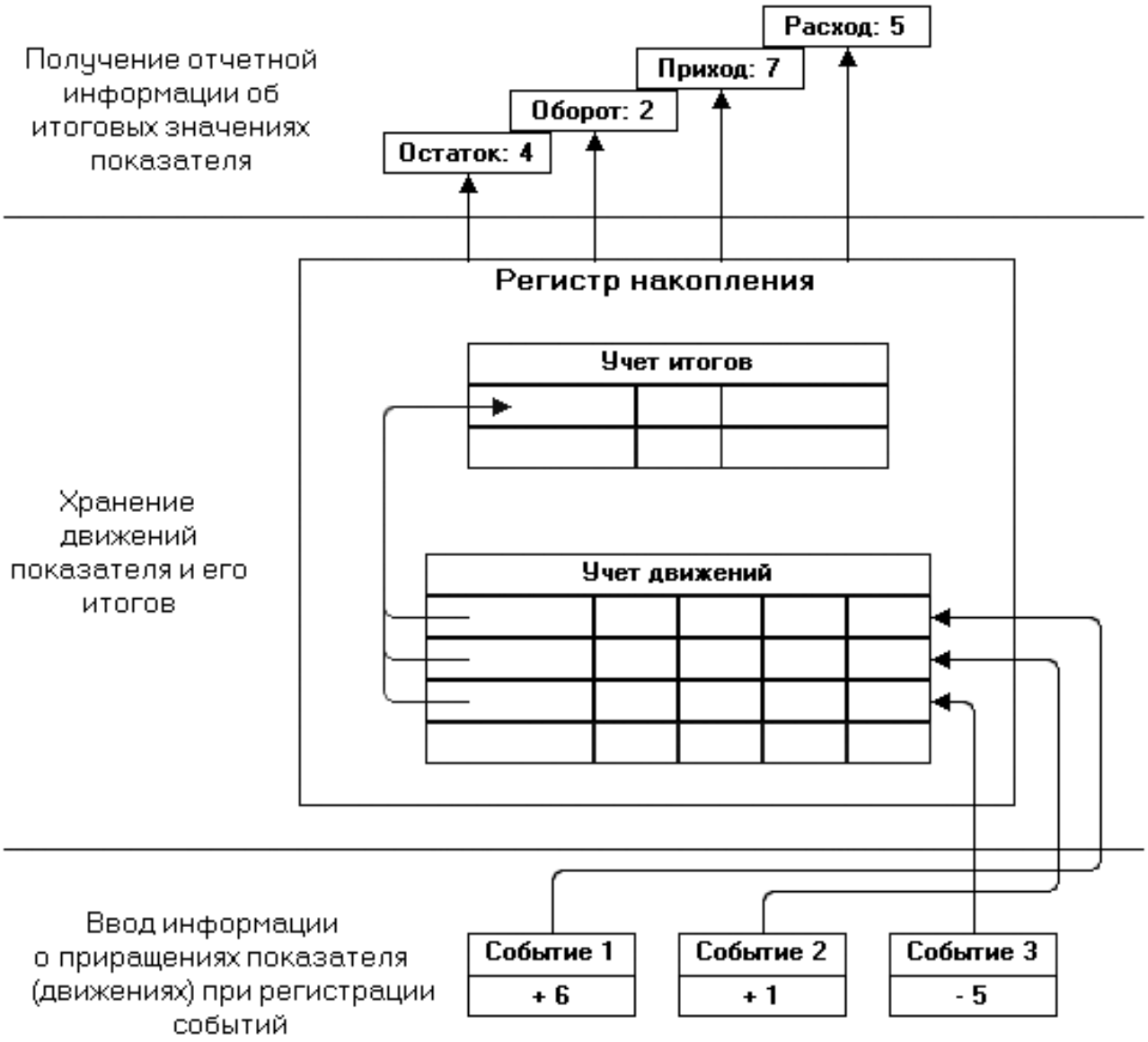


Рис. 4.1. Упрощенная схема регистра накопления [2]

Виды регистров накопления: *регистры остатков и регистры оборотов.*

Для гибкого и эффективного решения задач прикладной области система «1С:Предприятие» позволяет использовать два вида регистров накопления.

Регистры накопления остатков позволяют получать итоговые значения показателей остатков и, кроме того (суммируя приращения этих показателей за периоды), позволяют получать обороты. Например, при решении задачи учета товаров на складах может понадобиться как значение остатка товаров на момент времени, так и оборот поступлений или расходов товара за периоды времени.

Если же для некоторых сущностей накопление остатков смысла не имеет и требуется накапливать только обороты, тогда следует использовать *оборотные регистры накопления*. Например, учет оборотов продаж компании или запись данных о выручке предприятия.

При формировании структуры регистра накопления обязательно должен быть назначен регистратор, а также создан хотя бы один ресурс.

Структура регистра накопления: измерения, ресурсы, реквизиты.

В состав регистра накопления как объекта конфигурации входят измерения, ресурсы и реквизиты (рис. 4.2).

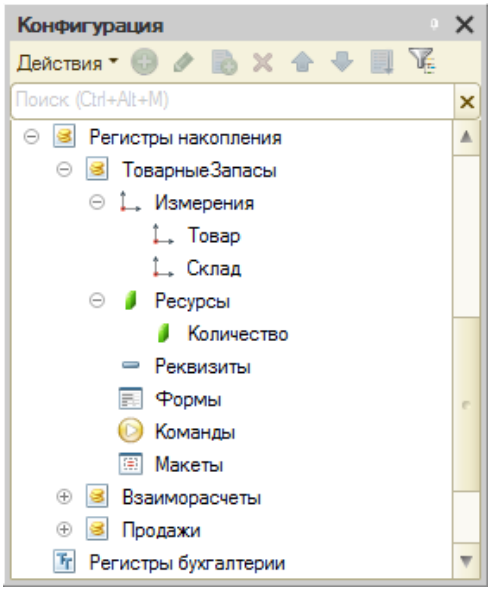


Рис. 4.2. Структура регистра накопления

Ресурсы используются для хранения информации как о приращениях, так и о самих значениях показателей. По сути, каждый ресурс хранит данные одного показателя [2].

В регистрах накопления разрезы учета реализуются с помощью измерений.

В состав регистра можно включить более одного измерения (рис. 4.3), поэтому в общем случае регистр накопления можно представить как *n*-мерную систему разрезов учета (измерений), в узлах которой хранятся совокупные данные ресурсов (итоги) [2].

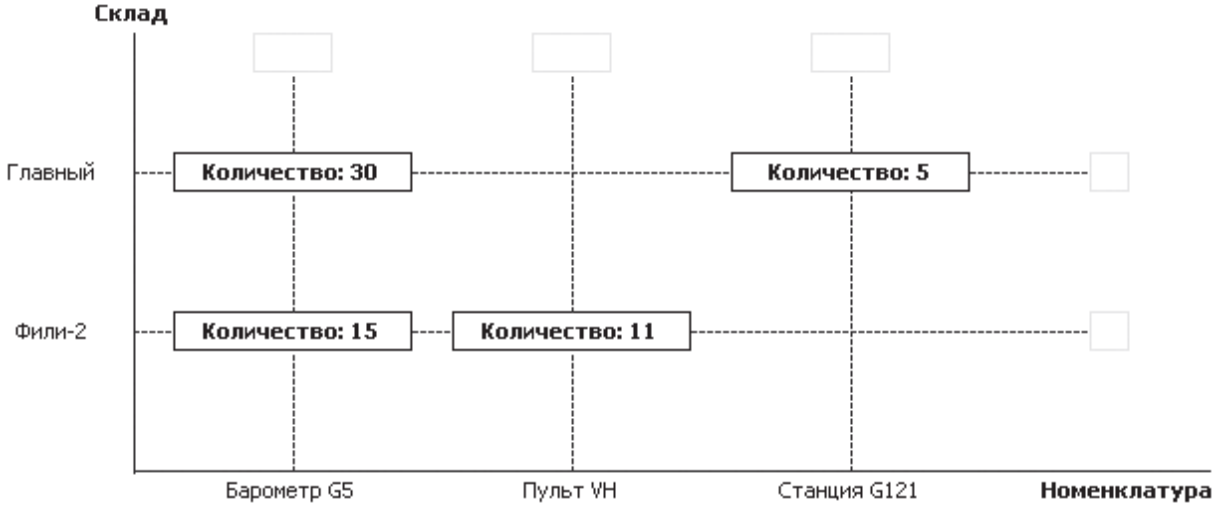


Рис. 4.3. Многомерная модель регистра накопления

Реквизиты – дополнительные характеристики движений, то есть первичных записей регистра. Реквизиты не влияют на итоги, хранимые в регистре. На вышеприведенной схеме, отражающей итоги, нет места реквизитам. Однако реквизиты могут быть использованы при решении задач анализа выполненных движений. То есть таких задач, в которых работа идет только с приращениями показателей, безотносительно их итоговых значений.

Обеспечения данных возможностей предусмотрена соответствующая структура таблиц базы данных и порядок работ с ними.

Данные каждого регистра накопления хранятся в базе данных в двух таблицах:

– таблица движений регистра накопления,

–таблица итогов регистра накопления.

Существует два вида регистров накопления, и каждый предназначен для ведения учета показателей своего вида (остатков или оборотов). И задачи учета показателей этих видов различны. Поэтому структура таблиц базы данных для регистров накопления разных видов тоже отличается. Она оптимизирована под решение соответствующих задач.

Для регистра накопления остатков состав колонок таблицы движений следующий [2]:

– *Период* – дата записи. Совместно с полями *Регистратор* и *НомерСтроки* определяет положение данной записи на временной оси;

– *Регистратор* – ссылка на документ, которому подчинена данная запись;

– *НомерСтроки* – уникальный номер данной записи в наборе записей регистра, подчиненных документу, указанному в поле *Регистратор*;

– *ВидДвижения* – значение системного перечисления *ВидДвиженияНакопления*, обозначающее направление приращения указанных в записи ресурсов на итоги по этим ресурсам (*Приход* или *Расход*);

– *Активность* – тип *Булево*. Содержит признак влияния записи на итоги регистра;

– <*Измерение*> – значение измерения. Количество таких полей равно количеству измерений, определенных в данных регистра как объекта конфигурации;

– <*Ресурс*> – значение ресурса. Количество таких полей равно количеству ресурсов, определенных в данных регистра как объекта конфигурации;

– <*Реквизит*> – значение реквизита. Количество таких полей равно количеству реквизитов, определенных в данных регистра как объекта конфигурации.

Таблица движений регистра хранит данные о выполненных по регистру движениях.

Таблица итогов регистра накопления остатков хранит текущие итоги на момент времени последнего движения (актуальные итоги). Дополнительно к этому могут храниться промежуточные итоги, если установлен период рассчитанных итогов.

Период рассчитанных итогов влияет только на производительность при получении итогов на некоторую промежуточную дату; если он установлен, итоги будут получены быстрее.

Структура таблицы итогов регистра накопления остатков следующая:

– *Период* – дата, на которую актуально состояние хранимого в таблице итога;

– <*Измерение*> – значение измерения – разреза учета хранимых итогов. Количество таких полей равно количеству измерений, определенных в данных регистра как объекта конфигурации;

– <*Ресурс*> – значение итога ресурса. Количество таких полей равно количеству ресурсов, определенных в данных регистра как объекта конфигурации;

– *Разделитель* – поле, позволяющее распараллелить обновление записей итогов. Добавляется в структуру таблицы итогов для регистров накопления, у которых установлено свойство *Разрешить разделение итогов*.

Структура таблиц оборотного регистра накопления, хранимых в базе данных, схожа. Отличие заключается только в том, что для оборотных регистров не существует понятия вид движения и понятия текущие итоги.

Состав колонок таблицы движений оборотного регистра накопления следующий:

– *Период* – дата записи. Совместно с полями *Регистратор* и *НомерСтроки* определяет положение данной записи на временной оси;

– *Регистратор* – ссылка на документ, которому подчинена данная запись;

– *НомерСтроки* – уникальный номер данной записи в наборе записей регистра, подчиненных документу, указанному в поле *Регистратор*;

– *Активность* – тип *Булево*. Содержит признак влияния записи на итоги регистра;

– <*Измерение*> – значение измерения. Количество таких полей равно количеству измерений, определенных в данных регистра как объекта конфигурации;

– <*Ресурс*> – значение ресурса. Количество таких полей равно количеству ресурсов, определенных в данных регистра как объекта конфигурации;

– <*Реквизит*> – значение реквизита. Количество таких полей равно количеству реквизитов, определенных в данных регистра как объекта конфигурации.

Структура таблицы итогов оборотного регистра накопления тоже схожа:

– *Период* – период (месяц), за который накоплен оборот итогов ресурсов;

– <*Измерение*> – значение измерения – разреза учета хранимых итогов. Количество таких полей равно количеству измерений, определенных в данных регистра как объекта конфигурации, у которых установлено свойство Использование в итогах;

– <*Ресурс*> – значение итога оборота ресурса. Количество таких полей равно количеству ресурсов, определенных в данных регистра как объекта конфигурации;

– *Разделитель* – поле, позволяющее распараллелить обновление записей итогов. Добавляется в структуру таблицы итогов для регистров накопления, у которых установлено свойство *Разрешить разделение итогов*.

Механизмы заполнения таблиц регистров накопления в БД. В таблицу движений регистра записи могут вводиться пользователем вручную, генерироваться в процессе выполнения обработок либо при проведении документов.

При формировании данных таблицы итогов важно обеспечить их непротиворечивость данным таблицы движений. Поэтому заполнение таблицы итогов осуществляется системой согласно активных записей таблицы движений, при расчете итогов. Т.е. пользователь или разработчик не могут непосредственно записывать данные в таблицу итогов регистра накопления, зато могут инициировать действия системы, производящие расчет и заполнение итогов.

Регистры накопления не поддерживают **независимого** формирования записей **без использования документа-регистратора.** Этим достигается *обоснованность* информации регистров – данными документов. Т.е. обоснованность информации объектов, осуществляющих учет показателей, данным объектов, осуществляющих первичную регистрацию событий, приводящих к изменению значений показателей. С другой стороны, к каждому регистратору может быть отнесено более одной записи движения. Поэтому для регистров накопления в таблице движений ключевыми являются поля Регистратор и НомерСтроки.

Для обеспечения манипулирования записями регистра при формировании или модифицировании движений используется объект встроенного языка системы РегистрНакопленияНаборЗаписей.<имя>. В отношении регистров накопления эти действия выполняются при использовании обязательного отбора по регистратору. Т.е. запись модифицированных данных в таблицу движений регистра накопления не может выполняться отдельно для каждого движения, а только «блоками», т.е. наборами записей, подчиненных одному регистратору.

В регистре накопления не могут существовать записи с пустым значением поля регистратор, то есть не подчиненные ни одному регистратору или «подчиненные пустому регистратору». Это противоречило бы принципу «обоснованности данных регистра накопления».

Соответственно, для заполнения или модификации данных таблицы движений есть две возможности:

– посредством создания набора записей регистра с обязательным отбором по регистратору;

– посредством использования свойства объекта документа Движения, представляющего собой коллекцию наборов записей, подчиненных данному регистратору.

Структура регистра сведений.

Объект 1С «*Регистры сведений*» – это прикладные объекты конфигурации, которые позволяют хранить в прикладном решении произвольные данные в разрезе нескольких измерений.

Для решения таких задач регистры сведений обладают функциональностью, которая включает в себя:

- уникальность записей в разрезах ключей записей,
- периодичность,
- подчинение записей регистратору.

Регистр сведений фактически представляет собой в общем случае многомерный массив данных, необходимый для реализации функции, которая может выдать нужную информацию по определенному набору аргументов. Аргументы функции называются **измерениями**, а результат функции – **ресурсами.** Количество используемых в составе регистра сведений измерений и ресурсов определяется задачами, которые решаются посредством этого регистра. Пример представлен на рис. 4.4.

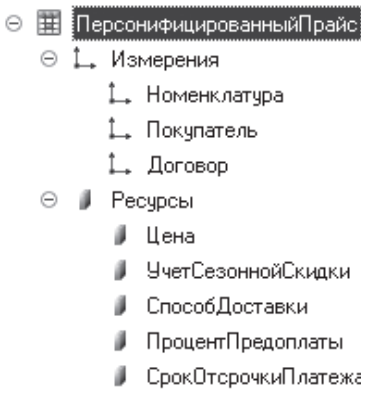


Рис. 4.4. Пример структуры регистра сведений

Регистры сведений обеспечивают уникальность хранимой информации для конкретных комбинаций значений разрезов ее хранения. То есть набор значений разрезов является уникальным ключом каждой записи регистра. регистр сведений не является простой таблицей для хранения произвольной информации. Требование уникальности записей по ключевым полям обязательно. И оно отслеживается системой автоматически, при любых попытках модификации данных или модификации состава регистра сведений [2]. Помимо измерений и ресурсов для регистра сведений может быть создан набор реквизитов. Реквизиты позволяют включать в записи регистров различную дополнительную информацию. Реквизиты не влияют на значения ресурсов регистра и могут использоваться для анализа записей регистра.

Внесение изменений в регистр сведений может выполняться: вручную; при помощи документов. На этапе проектирования разработчик для этого определяет режим записи в регистр сведений: *Независимый* или *Подчинение регистратору*.

Периодические регистры сведений.

Одной из возможностей регистра сведений является хранение данных не только в разрезе указанных измерений, но и в разрезе времени. Главное отличие периодического регистра сведений от обычного заключается в том, что в нем присутствует дополнительное системное измерение "Период", имеющее тип "дата". Это позволяет получать не только текущие сведения об объекте, но также на любой момент времени.

Данное свойство позволяет добавить к списку измерений регистра дополнительное измерение — «Период». Разработчик может указать минимальную периодичность, с которой записи будут заноситься в регистр.

Периодичность может принимать следующие значения [2]:

- «Непериодический»;
- «В пределах секунды»;
- «В пределах дня»;
- «В пределах месяца»;
- «В пределах квартала»;
- «В пределах года».

При выборе периодичности, отличной от варианта «Непериодический», система будет контролировать уникальность записей в пределах заданного промежутка времени. Если запись не уникальна, система 1С выдаст сообщение «Запись с такими ключевыми полями существует!» и не даст произвести запись в базу данных.

В результате установки периодичности в структуру хранения данных регистра будет добавлено поле *Период*. Причем оно будет включено в состав ключа записей регистра. По сути, еще один разрез хранения информации.

Таким образом, в регистре сведений можно хранить не просто статические данные, но и историю изменений этих данных.

При такой организации хранения информации появляются возможности:

– получения информации, актуальной на тот или иной временной момент;

– получения информации о значениях данных, вступающих в силу после некоего временного момента;

– получения картины динамики изменения данных за некоторый период;

– решения других прикладных задач, связанных с хранением информации в привязке ко времени.

Подчинение записей регистратору.

В ряде прикладных задач возникает необходимость обеспечения обоснования хранимой в регистрах сведений информации наличием документов, зарегистрировавших изменения этой информации.

Внесение изменений в регистр сведений может выполняться как вручную, так и при помощи документов. В случае, когда изменения в регистр сведений вносятся с помощью документов, к каждой записи регистра добавляется специальное поле, в котором хранится информация о регистраторе — документе, с которым связана эта запись. В процессе создания прикладного решения разработчик указывает, какой именно режим записи будет использоваться данным регистром сведений.

Использование значения свойства *Режима записи* равным **Подчинение регистратору** может потребоваться в случае, когда логика работы прикладного решения требует того, чтобы изменения, выполняемые в регистре сведений, были жестко связаны с документами, фиксирующими факты хозяйственной деятельности. Такая настройка приводит к включению в состав регистра полей *Регистратор, НомерСтроки* и *Активность*.

Далее требуется указать, какие именно документы могут быть регистраторами записей регистра на закладке *Регистраторы*. Поле Регистратор обеспечивает привязку к документу, но не всегда входит в разряд ключевых полей. Для того чтобы поле *Регистратор* стало ключевым, необходимо для такого регистра сведений (с режимом записи *Подчинение регистратору*) установить для свойства *Периодичность* значение *По позиции регистратора*. С прикладной точки зрения это означает возможность хранения в регистре сведений данных с привязкой к временной оси с дробностью до момента времени.

Проектирование структуры регистров сведений.

В состав регистра сведений как объекта конфигурации включаются: измерения, ресурсы, реквизиты.

Ресурсы хранят данные регистра, то есть ту информацию, ради хранения которой регистр, собственно, создавался и с которой работает функциональность регистра. *Измерения* используются для обеспечения разрезов хранения этой информации. Кроме того, для ситуаций, когда необходимо кроме данных хранить дополнительную информацию собственно о записях регистра, возможно использование *реквизитов* [2].

Для измерений может быть установлено свойство *Ведущее*. *Ведущее* – установка этого свойства имеет смысл для измерений, тип данных которых - ссылка на объект конфигурации. В этом случае считается, что запись регистра сведений имеет смысл, только пока существует этот объект. При удалении объекта записи по нему будут автоматически удалены из регистра.

Порядок расстановки измерений регистра сведений имеет важное значение. Измерения, к которым необходим быстрый доступ, следует располагать в начале списка измерений.

Последовательность расстановки измерений регистра сведений влияет на возможность применения методов встроенного языка, использующих позиционный доступ к измерениям.

Данные каждого регистра сведений хранятся в отдельной таблице базы данных. Каждая такая таблица имеет состав колонок, представленный в табл. 4.1.

Таблица 4.1

Состав колонок регистра сведений

Колонка	Описание
1	2
Период	дата записи, которая определяет положение данной записи на временной оси. Это поле существует только для периодических регистров
Регистратор	содержит ссылку на документ, которому подчинена данная запись. Это поле существует только для регистров с режимом записи *Подчинение регистратору*
НомерСтроки	уникальный номер данной записи в наборе записей регистра, подчиненных документу, указанному в поле *Регистратор*. Это поле существует только для регистров с режимом записи *Подчинение регистратору*
Активность	тип *Булево*. Содержит признак влияния записи на получение информации из регистра. Записи, для которых значение данного свойства установлено в *Ложь*, не будут учитываться при получении «первых» или «последних» записей регистра, а также при получении сведений на определенный момент времени. Существует только для регистров с режимом записи Подчинение регистратору
Ключ	короткий ключ записи регистра. Поле присутствует у непериодических регистров, имеющих хотя бы одно измерение
<Измерение>	содержит значение измерения. Количество таких полей равно количеству измерений, определенных для регистра как объекта конфигурации
<Ресурс>	значение ресурса. Количество таких полей равно количеству ресурсов, определенных для регистра как объекта конфигурации
<Реквизит>	значение реквизита. Количество таких полей равно количеству реквизитов, определенных для регистра как объекта конфигурации

Разработка структуры регистра заключается в создании наборов измерений, ресурсов и реквизитов.

Каждая запись в регистре сведений содержит информацию о том, что для данной комбинации ключевых полей установлены некоторые значения ресурсов. В том числе и в периодическом регистре сведений. То есть если запись добавляется или модифицируется, это выполняется для всех значений ресурсов. Нельзя отметить, что, начиная с такого-то момента, значение такого-то ресурса не меняется, а меняются только значения других ресурсов.

При проектировании регистра важно проанализировать, как реально меняются значения данных, которые будут храниться в регистре, и выработать определенную модель отражения этих изменений в базе данных. Важно, чтобы отражение предметной области в регистре правильно воспринималось пользователем.

Состав существующего регистра сведений всегда можно расширить. Одним из направлений расширения является ввод новых разрезов информации, то есть измерений.

При проектировании структуры регистра следует также учитывать такой фактор, как время получения данных, необходимых для работы механизмов решения. Например, при работе некоторого механизма важно получать данные объекта и данные, связанные с данным объектом. Очевидно, что получение этих связанных данных из нескольких ресурсов одного регистра (одной таблицы) будет происходить в общем случае быстрее, нежели получение этих данных из нескольких регистров (нескольких таблиц).

Следует также учитывать такой фактор, как время записи информации в базу данных. Запись в несколько регистров сведений в общем случае будет выполняться медленнее, нежели запись одной записи с несколькими ресурсами.

Но с другой стороны надо учитывать еще один фактор, так называемое «*Слабое звено*». Частота, с которой происходит модификация значений ресурсов, может сильно различаться. Например, некий Ресурс1 – нужно модифицировать очень часто. В этом случае если в составе регистра есть другие ресурсы, которые изменяются гораздо реже, то объем базы будет расти существеннее, чем если бы регистр содержал только Ресурс1. Особенно если какой-нибудь из ресурсов будет являться хранилищем значений, в котором будет храниться картинка или бинарные данные. Такие данные вообще целесообразнее выделять в отдельный регистр.

Рекомендации при проектировании структуры регистра [2].

– При записи регистра сведений записываются все ресурсы. Не нужно проектировать структуру таким образом, что один ресурс меняется каждый день, а второй два раза в год. Лучше вынести второй ресурс в отдельный регистр сведений.

– Получение данных из нескольких ресурсов одного регистра сведений выполняется быстрее, чем получение одного ресурса из нескольких регистров.

– Запись в несколько ресурсов одного регистра сведений выполняется быстрее, чем запись одного ресурса в несколько регистров.

Возможные способы записи движений по регистру накопления.

Всего можно классифицировать четыре различных способа формирования движений.

- *При проведении документа*, т.е. в модуле объекта «Документ» посредством процедуры – обработчика событий «ОбработкаПроведения». Наиболее распространенный способ формирования движений в регистр. Движения и текст процедуры «ОбработкаПроведения» чаще всего формируют с использованием конструктора движений.
- *Из объекта документа, но без проведения.* Любая процедура в модуле объекта документа может записать новые движения. Если это не процедура «ОбработкаПроведения» (и не вызывается из нее), тогда создается ситуация, при которой документ не проведен, но движения у него есть. Пример, процедура предварительного бронирования товара. В ситуации, когда при оформлении накладных параллельно работают несколько менеджеров, возможна ситуация конкуренции за ресурсы на складе. Пока один менеджер вводит большой документ с множеством позиций, другой менеджер, вводя документ с небольшим количеством позиций, может списать товар, нужный для первого документа. В этом случае функционал предварительного бронирования крайне полезен. Документ еще не проведен, т.к. не полностью введен, но позволяет наложить бронь на товары (выполнить проводки в соответствующий регистр бронирования из объекта документа, но не в процедуре ОбработкаПроведения).
- *Извне объекта документа.* Любой объект, содержащий процедуру работы с набором записей регистра может приписать любому возможному регистратору данного регистра любые движения.
- *Интерактивное внесение данных в регистр (ручная операция).* Достаточно редко, но бывают ситуации, когда логику установки или изменения значений показателей в регистре по каким-то причинам сложно, не нужно или невозможно описать заранее, на этапе разработки конфигурации. Но при этом необходимо дать возможность пользователю все же данные в регистре изменять. В таких ситуациях обычно применяют интерактивный способ внесения информации в регистры. Реализуется посредством специального документа (регистратора), которому будет запрещено делать движения программным способом. Но при этом предоставляются интерактивные возможности (на форме) непосредственного создания записей в регистре. Никаких реквизитов и табличных частей документ одержать не должен.

Тема 5. Реализация в корпоративных информационных системах задач бухгалтерского учета

Схема взаимодействия объектов системы "1С:Предприятие" при реализации в корпоративных информационных системах задач бухгалтерского учета. Основные объекты, участвующие в схеме. Подвиды и детализация учета в системе: синтетический учет, аналитический учет, консолидированный учет, количественный учет, валютный учет.

Бухгалтерский учет является сплошным, непрерывным, взаимосвязанным отражением всей хозяйственной деятельности предприятия (на всех участках), на основании документов в различных измерителях, где обобщающим является денежный измеритель. Методы, которые используются для его ведения: инвентаризация (проверка фактического наличия), документация (письменное свидетельство) и двойная запись на счетах.

Ведение бухгалтерского учета в системе «1С:Предприятие» обеспечивают объекты конфигурации Планы счетов и Регистры бухгалтерии. Эти объекты тесно взаимосвязаны между собой.

Схему взаимодействия объектов системы «1С:Предприятие» при реализации задачи бухгалтерского учета можно представить следующим образом.

Рассмотрим объекты, участвующие в схеме, и начнем это делать с плана счетов (рис. 5.1) [2]. Бухгалтерский учет ведется на счетах, каждый из которых предназначен для группировки однородных хозяйственных операций. Все счета, используемые для ведения учета на предприятии, объединяются в план счетов. Счета – это основной разрез учета. Все остальные разрезы (аналитический учет, измерения) являются вспомогательными и необязательными.

Для хранения бухгалтерских проводок по счетам используется объект конфигурации Регистр бухгалтерии. Самый простой регистр бухгалтерии, предназначенный для ведения бухгалтерского синтетического учета, может содержать всего один ресурс (для учета средств в валюте учета, в которой будет составляться вся отчетность и формироваться баланс) и более ничего. Такой регистр бухгалтерии позволит анализировать остатки и обороты по счетам, а также и обороты между счетами.

Для получения оборотов между счетами регистр должен поддерживать схему учета с поддержкой корреспонденций, при которой каждая запись обязательно включает два счета: дебетуемый и кредитуемый.

Дополнительными разрезами при ведении учета являются аналитические разрезы (аналитический учет) и измерения регистра бухгалтерии (с помощью которых возможно реализовать многовалютный учет, консолидированный учет и другие).

Аналитический учет описывается в системе «1С:Предприятие» с помощью понятий Вид субконто и Субконто. Под видом субконто подразумевается группа однородных объектов аналитического учета (список контрагентов, номенклатуры, статей и др.). Субконто – это один конкретный объект из этого

множества (объект аналитического учета, например: контрагент, карточка товара, статья затрат и др.).

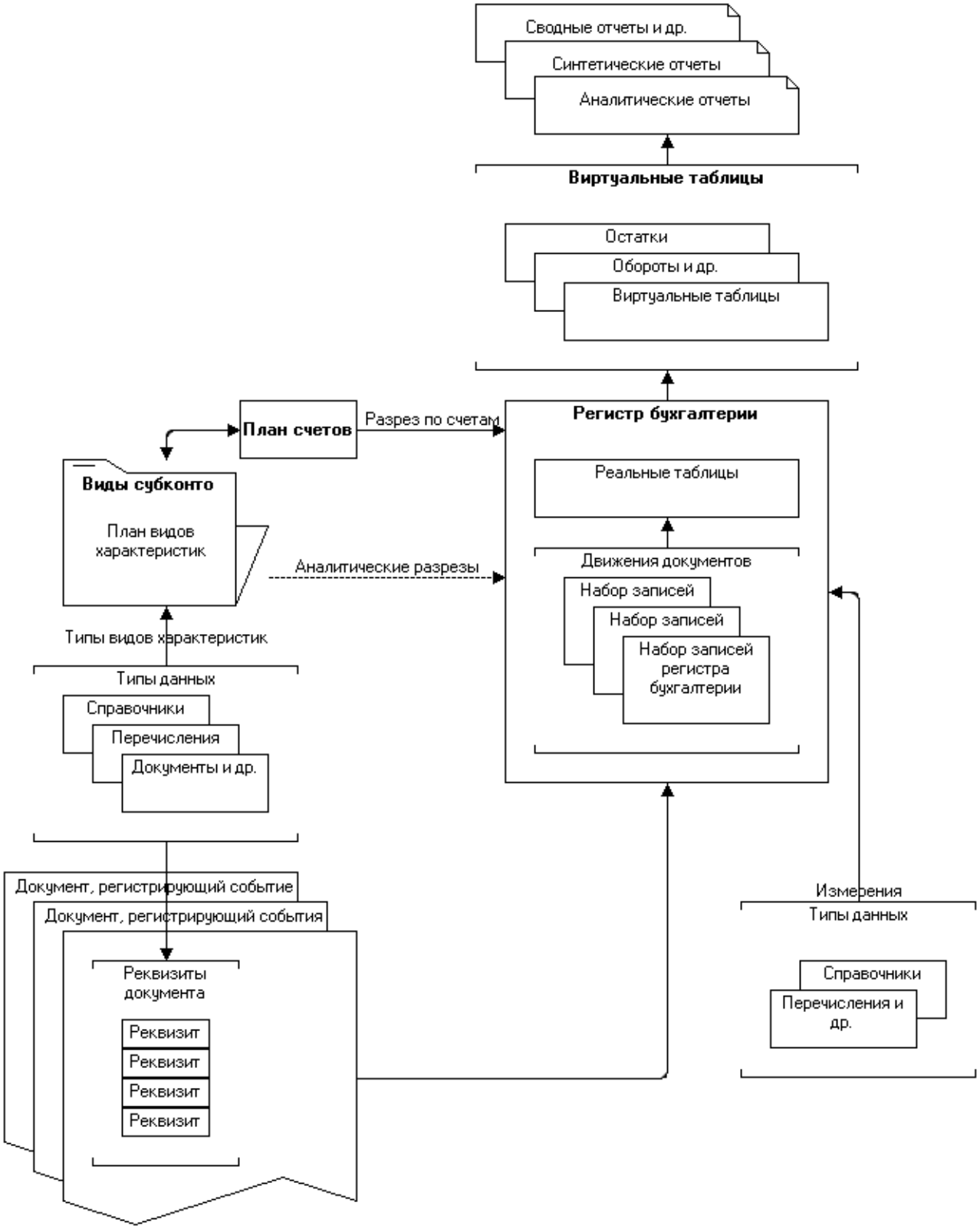


Рис. 5.1. Схема взаимодействия объектов

Для реализации аналитического учета используется «связка» объектов План видов характеристик, План счетов, Регистр бухгалтерии. План видов характеристик, использующийся для ведения аналитического учета на выбранном плане счетов, выбирается в соответствующем свойстве плана счетов.

Виды характеристик «привязываются» к счетам плана счетов, требующим ведения аналитического учета. Результат этих действий – изменяющиеся таблицы регистра бухгалтерии, которые позволят хранить остатки и обороты не только по счетам, но и по счетам в разрезе субконто.

Каждая запись (проводка) регистра бухгалтерии подчинена какому-то регистратору (документу) и, как правило, им же создана. При проектировании реквизитного состава документов, которые должны формировать записи в регистре бухгалтерии, как правило, используются ссылочные типы данных. Документ при проектировании задачи бухгалтерского учета – это, прежде всего, средство формирования движений в регистре бухгалтерии. А движения эти, как уже было сказано выше, содержат субконто и измерения, которые и заполняются обычно значениями реквизитов документа.

Каждый документ имеет свойство Движения, представляющее собой коллекцию наборов записей тех регистров, регистратором для которых он является.

Счет и план счетов. Каждый счет – это учетный регистр, представляющий собой таблицу с двусторонней записью. Левая сторона счета называется «дебет», правая – «кредит». Две стороны нужны счету для отражения операций увеличения и уменьшения на счете. На счете группируются однородные хозяйственные операции. Отрицательные числа используются в БУ только для операций сторнирования (исправления ошибок) – во всех остальных случаях – при поступлении средств в кассу, и при их списании из кассы, на счет заносится положительная цифра, меняется лишь сторона счета (дебет или кредит).

В современном БУ используется «двурядная система счетов», при которой состав и назначение счетов является следствием главной формы бухгалтерского учета – баланса.

В левой части баланса отражаются активы (средства) предприятия по их составу и размещению. Можно перечислить такие средства предприятия, как: деньги в кассе и на расчетном счете банка, товарно-материальные запасы, долги «кого-то» нашему предприятию. Все это наше имущество, т.е. активы.

В правой части баланса отражаются источники средств предприятия (т.е. откуда предприятие их получило и кому их должно). Средства, как известно, не могут взяться из «ниоткуда», и исчезнуть в «никуда», поэтому раз у нас какое-то имущество, должны быть и обязательства. Часть средств нам дали собственники (хозяева, инвесторы, учредители) бизнеса, и мы учитываем долг предприятия перед собственником на счете «Уставной капитал», другую часть средств мы получили в банке или взяли товары в долг у поставщика.

Средства не могли взяться из «ниоткуда» или исчезнуть в «никуда», поэтому сумма всего имущества (в денежном выражении) должна быть равна сумме обязательств – «баланс должен балансировать».

Счета, используемые для учета операций с имуществом, остатки которых находят свое отражение в активе баланса, называют «активными». Счета учета обязательств – «пассивными» (рис. 5.2) [2].

Активный счет	
Дебет	Кредит
Сальдо на начало	
Операции поступления средств на счет	Операции списания средств со счета
Дебетовый оборот	Кредитовый оборот
Сальдо на конец периода	

Пассивный счет	
Дебет	Кредит
	Сальдо на начало
Операции погашения задолженности	Операции увеличения задолженности
Дебетовый оборот	Кредитовый оборот
	Сальдо на конец периода

Рис. 5.2. Активный и пассивный счет

При отражении операций по активному счету увеличение средств на счете отображается по дебету счета, а списание по кредиту. Списать со счета больше, чем туда положили за все время, невозможно, поэтому остаток по счету (на начало и на конец периода) всегда дебетовый.

При отражении операции по пассивному счету, увеличение задолженности перед «кем-то» мы отражаем по кредиту счета, а ее погашение (когда мы отдаем долг) – по дебету. Соответственно, остаток по такому счету всегда будет кредитовый. Если вдруг мы отдали нашему контрагенту больше, чем были ему должны, то теперь не мы ему должны, а он нам. И такую задолженность в БУ принято отражать на другом счете (дебиторская задолженность актива баланса).

В актив баланса попадут дебетовые остатки активных счетов, а в пассив – кредитовые остатки пассивных.

Российские стандарты БУ предполагают наличие третьего вида счета – «активно-пассивного». Это, как правило, счета расчетов. Остаток на этих счетах зависит от совершенных операций и может быть как дебетовым, так и кредитовым. И включаться этот остаток может как в актив, так и в пассив.

Все счета, используемые для ведения учета на предприятии, объединяются в план счетов. Это справочник, список учетных регистров, применяемых для ведения учета.

Основанием записи по бухгалтерскому счету всегда является документ. Операция – это все записи, которые будут сделаны на основании одного документа. Любая запись в БУ затрагивает сразу 2 счета: дебет одного и кредит другого. Такая запись называется проводкой.

В самом простом варианте проводка содержит один дебетуемый счет и один кредитуемый. Именно таким образом и ведется учет в соответствии с РСБУ. При такой схеме учета, во-первых, соблюдается баланс двойной записи: оба счета изменились на одну и ту же сумму. А во-вторых, достигается возможность анализа оборотов между счетами: мы всегда можем узнать на какую сумму было получено товаров от поставщиков…

Западные стандарты учета допускают использование сложных проводок (один счет дебетуется, несколько кредитуется, или наоборот) и сборных проводок (несколько счетов дебетуется и несколько кредитуется). В этом случае каждая операция состоит из нескольких зависимых записей. Правило «двойной

записи» при этом не нарушается и баланс все равно «балансирует»: при вводе такой проводки необходимым условием ее записи будет являться равенство суммы всех дебетовых всем кредитовым записям одной операции.

Большая гибкость при ведении учета сложными и сборными проводками имеет и обратную сторону. При ведении учета сборными проводками теряется возможность анализа оборотов между счетами. Остается лишь возможность анализа остатков и оборотов по отдельно взятому счету.

Система «1С:Предприятие» позволяет реализовывать обе учетные схемы: с поддержкой корреспонденции – два счета в проводке, или без поддержки корреспонденции.

План счетов – объект, который позволяет описать состав счетов, используемый для ведения учета на предприятии, и их свойства.

В одной конфигурации может быть несколько планов счетов. Для целей регламентированного учета это не требуется, но встречаются задачи управленческого учета, требующие ведения учета по разным стандартам или ведения не только бухгалтерского учета, который фиксирует свершившиеся факты, но и бюджетного учета, который учитываем планируемые операции.

Основы организации аналитического учета. При автоматизации задач аналитического учета используются понятия Вид субконто и Субконто.

Каждый элемент вида субконто представляет собой объект аналитического учета и называется Субконто. Ни субконто, ни вид субконто не имеют объектного представления в системе и существуют лишь как понятия, реализуемые с помощью других объектов. Виды субконто, как правило, бывают ссылочного типа и содержат в себе ссылку на справочник, документ или перечисление, хотя возможны и другие варианты. Если видом субконто является справочник, то субконто – один элемент этого справочника. По сути субконто – подключаемые измерения для счетов плана счетов.

Использование видов субконто для организации аналитического учета можно представить в виде схемы (рис. 5.3).

Каждый счет на данной схеме можно рассматривать как учетный регистр, накапливающий остатки и обороты в денежном выражении. Виды субконто позволяют добавлять измерения в эти регистры и создают новые сущности. Так, например, один и тот же справочник Контрагенты может использоваться для организации аналитического учета как на счете Покупатели, так и на счете Поставщики. Возможность системы привязать к одному счету несколько самостоятельных видов субконто позволяет нам получить фасетное (параллельное) деление: например, на одном складе могут быть разные товары, один и тот же товар может быть на разных складах. Мы всегда можем получить остаток по складу, по товару и по товару на складе.

Необходимость учета всего множества хозяйственных операций предприятия исключает возможность задать однажды состав и структуру учетных регистров и не изменять их в процессе работы пользователя. В процессе работы возникают новые виды деятельности, новые хозяйственные операции и как результат новые учетные регистры, которые тоже должны попасть в баланс двойной записи. Причем возможность создавать эти новые учетные регистры

предоставлена не только программисту, но и пользователю программного продукта. Достигается эта возможность за счет добавления новых счетов в план счетов и определения для них аналитики путем выбора вида субконто из списка существующих в конфигурации или добавления новых видов субконто в план видов характеристик [2].

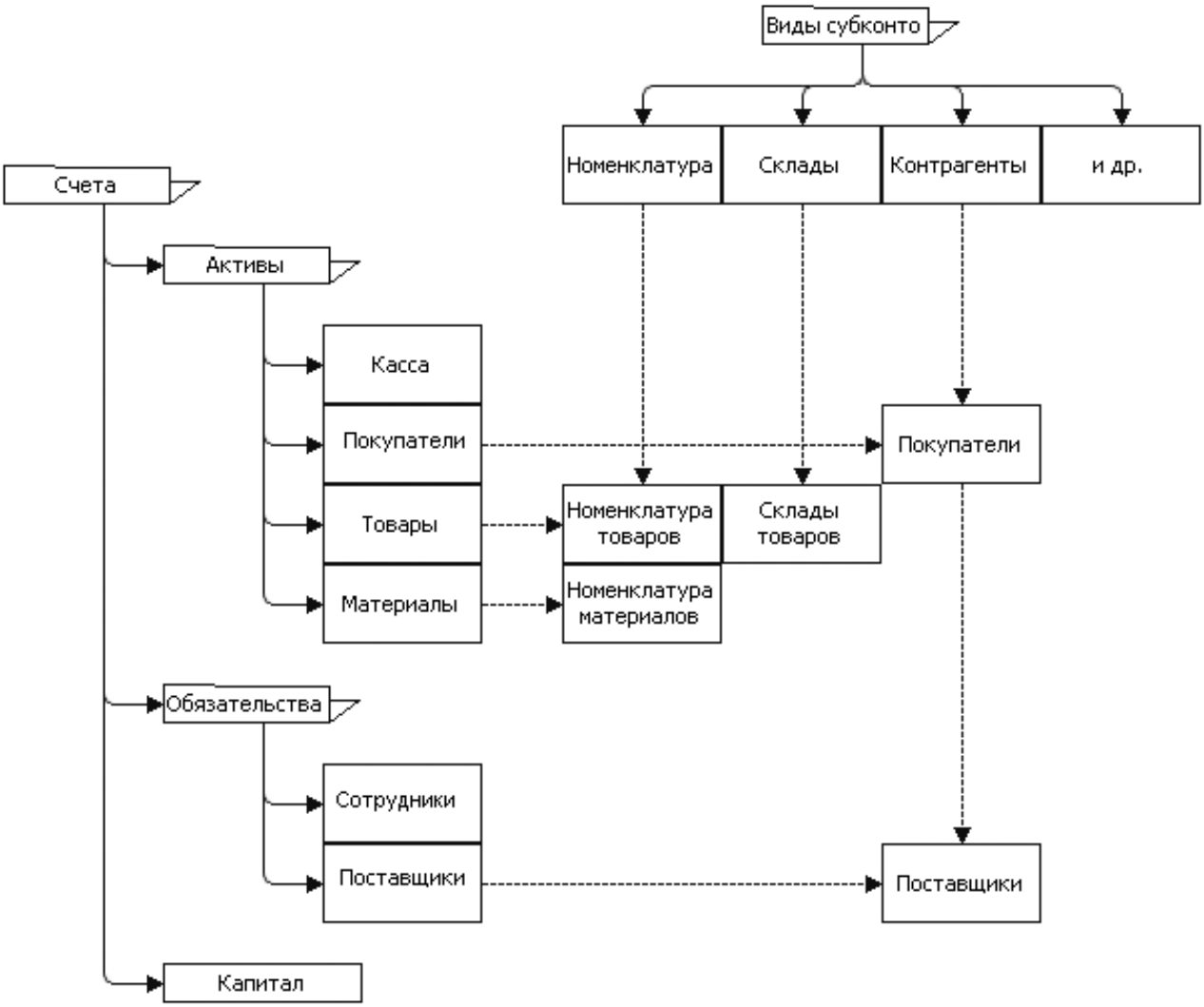


Рис. 5.3. Организация аналитического учета [2]

По сути регистр бухгалтерии – это регистра регистров, совокупность регистров. В целях контроля все учетные регистры (счета) связаны между собой правилом двойной записи, которое образуют замкнутую систему показателей: невозможно изменить значение показателя одного учетного регистра, не затронув другой, причем на ту же сумму.

Основные свойства регистра бухгалтерии.

План счетов, проводки по которым содержит регистр бухгалтерии. Как планов счетов, так и регистров бухгалтерии может быть несколько, но один регистр бухгалтерии содержит проводки только между счетами одного плана счетов. В то же время на одном плане счетов могут основываться два и более регистра бухгалтерии.

В случае если регистр поддерживает корреспонденцию, т.е. реализует привычную для России схему учета, каждая проводка включает два корреспондирующих счета (счет дебета и счет кредита) и ресурс Сумма (рис. 5.4). При такой архитектуре регистра пользователю будут доступны для анализа остатки и обороты по каждому счету и обороты между счетами.

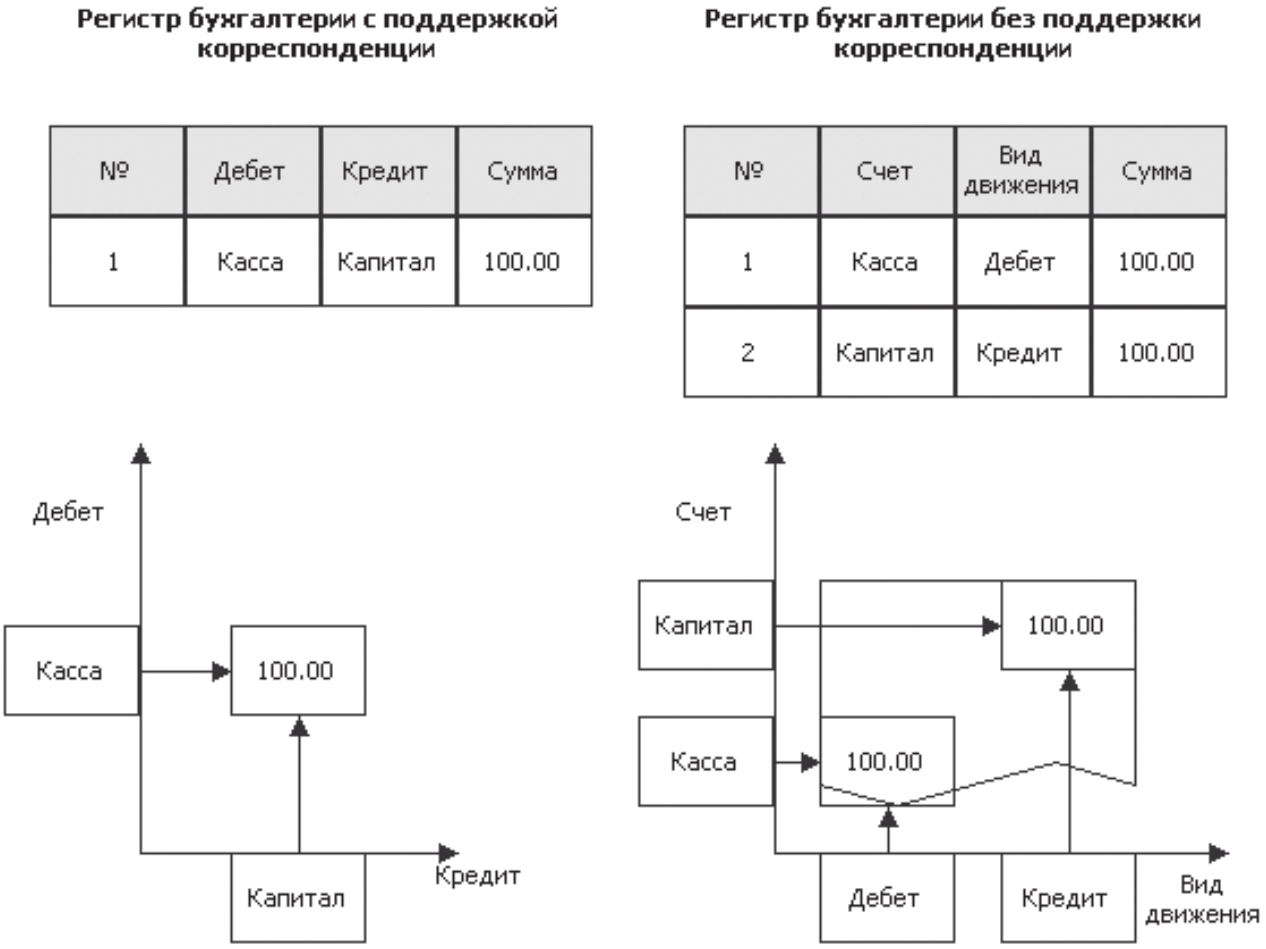


Рис. 5.4. Регистры бухгалтерии с поддержкой и без поддержки корреспонденции [2]

Учетная схема без поддержки корреспонденции может считаться более гибкой и в большей степени удовлетворяющей отражению экономической сути хозяйственных операций. Но отсутствие корреспонденции исключает возможность анализа оборотов между счетами.

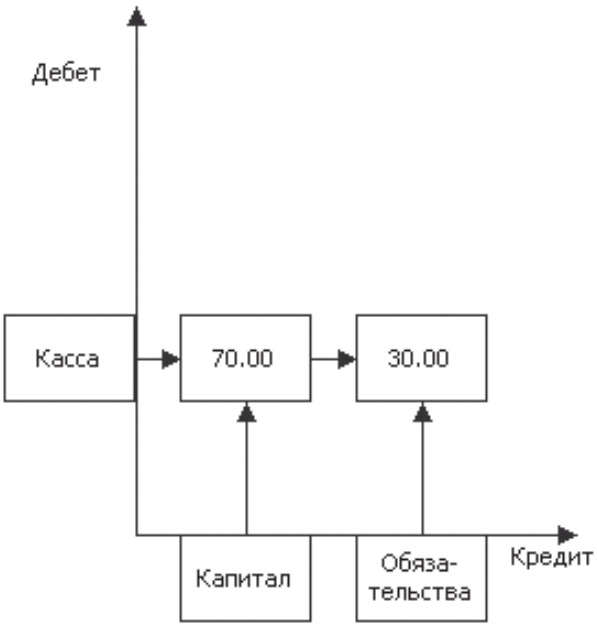
Однако отсутствие поддержки корреспонденции не означает отсутствие контроля двойной записи. Если корреспонденция поддерживается, контроль двойной записи осуществляется в рамках каждой проводки. Если корреспонденция не поддерживается, то контроль двойной записи осуществляется в рамках всего набора записей, принадлежащих одному регистратору (рис. 5.5).

Регистр бухгалтерии с поддержкой корреспонденции

№	Дебет	Кредит	Сумма
1	Касса	Капитал	70.00
2	Касса	Обяза-тельства	30.00

Регистр бухгалтерии без поддержки корреспонденции

№	Счет	Вид движения	Сумма
1	Касса	Дебет	100.00
2	Капитал	Кредит	70.00
3	Обяза-тельства	Кредит	30.00



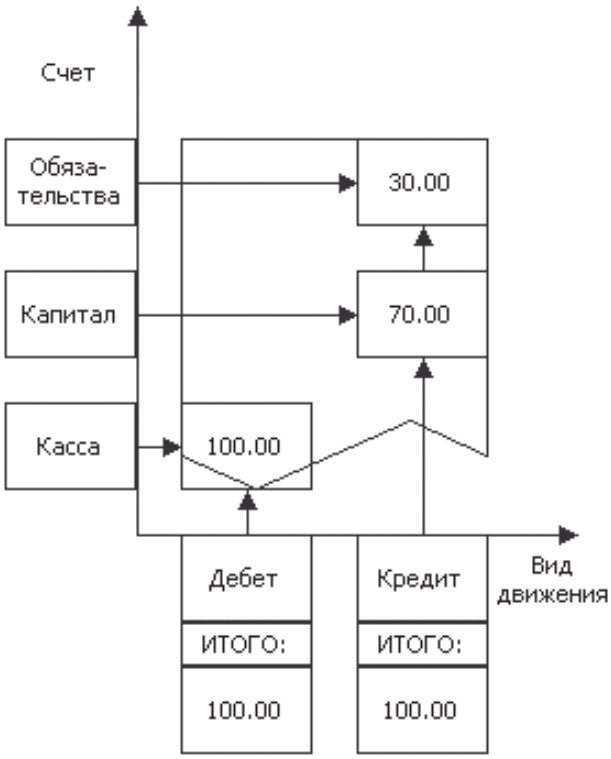


Рис. 5.5. Проводки в регистрах бухгалтерии с поддержкой и без поддержки корреспонденции

Запись движений в регистр бухгалтерии. Регистр бухгалтерии всегда подчинен регистратору. Движения в регистре можно сделать, только с указанием ссылки на документ-регистратор. Им может являться любой документ системы.

Документ, во-первых, требуется как «ссылка, которая будет прописана в одно из полей регистра бухгалтерии, а, во-вторых, документ обладает коллекцией наборов записей Движения, которая используется для записи движений в регистры. Коллекция Движения содержит в себе наборы записей регистров, по которым документ может делать движения (рис. 5.6). Коллекция содержит столько элементов, для скольких регистров документ является регистратором [2].

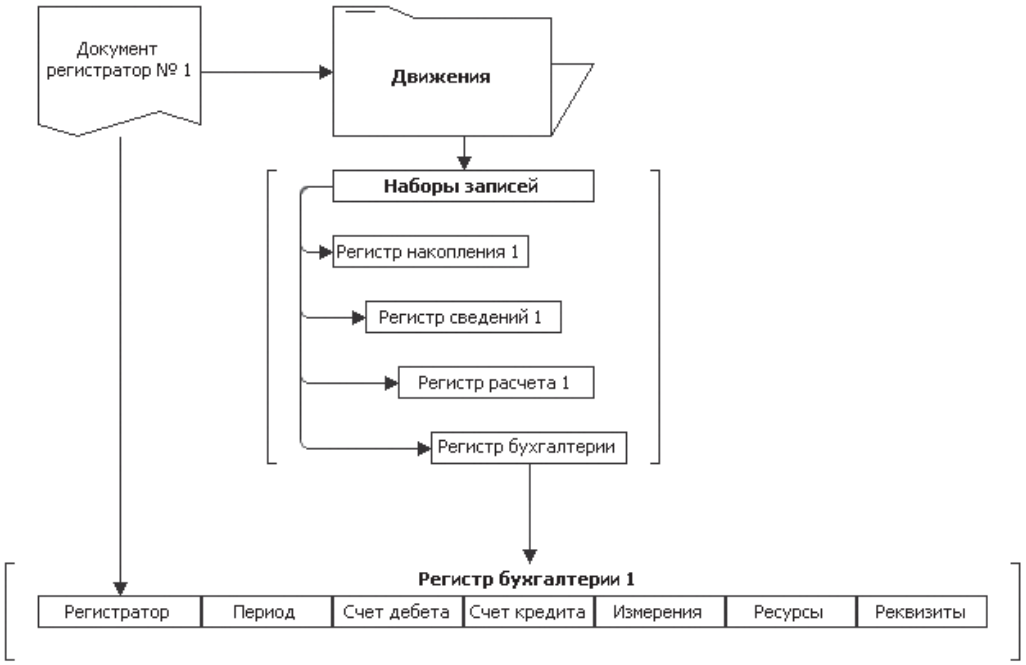


Рис. 5.6. Коллекция наборов записей Движения у документа

Подвиды и детализация учета в системе: синтетический учет, аналитический учет, консолидированный учет, количественный учет, валютный учет.

Система «1С:Предприятие» позволяет вести неограниченное число видов учета.

Синтетический учет – это учет обобщенных данных бухучета о видах имущества, обязательств и хозяйственных операций.

Единственным обязательным видом учета в бухгалтерии является синтетический учет на счетах и субсчетах в денежном выражении в валюте учета. Дополнительно к синтетическому учету может быть организовано ведение и других видов учета.

Аналитические счета открываются в развитие определенного синтетического счета в разрезе видов, частей, статей учтенных на нем объектов, где это требуется в натуральном выражении. При этом сальдо, дебетовый и кредитовый обороты одного синтетического счета должны быть равны, соответственно, сумме сальдо, сумме дебетовых и сумме кредитовых оборотов всех относящихся к нему аналитических счетов.

Аналитический учет – это учет, который ведется в лицевых, материальных и иных аналитических счетах бухучета, группирующих детальную информацию внутри каждого синтетического счета. Субсчета – подсобные счета, промежуточное звено между синтетическими и аналитическими счетами. В отличие от синтетических счетов, называемых счетами 1-го порядка, субсчета относятся к счетам 2-го порядка, а объединяемые ими аналитические счета являются счетами 3-го, 4-го и т. д. порядка. Субсчета используются для объектов учета с разнообразной номенклатурой. Сумма сальдо всех субсчетов счета равно сальдо счета.

Синтетический и аналитический учет ведутся одновременно и параллельно. Бухгалтерская проводка сначала отражается в синтетическом счете, а затем отражается в аналитическом счете, но более подробно. Записи в синтетических счетах по сумме равны записям в аналитических счетах. Аналитический учет составляет основу для формирования данных синтетического учета. Аналитические счета хранят детальные, частные сведения. Аналитический счет раскрывает синтетический в разрезе его видов, частей, статей и, где это требуется, с оценкой информации в натуральном, трудовом и денежном выражении.

Консолидированный учет – организация учета, при которой пользователь может на одном плане счетов, в одном регистре бухгалтерии вести учет нескольких самостоятельных единиц и получать по каждой из них самостоятельный баланс и суммированный баланс по всем (или по группе).

Количественный учет. Система 1С:Предприятие, помимо учета денежных средств, предоставляет пользователю возможность ведения количественного учета, т.е. это учет материальных ценностей в количественном (натуральном выражении). Количественный учет можно вести на любом счете или субсчете. Возможность ведения на счете количественного учета включается при редактировании планов счетов (признак учета «Количественный»).

Как правило, количественный учет ведется вместе с аналитическим. Например, ведение на счете учета материалов аналитического и количественного учета по материалам позволит получать сведения о наличии и движении материалов не только в денежном, но и в количественном выражении.

Тем не менее, система «1С:Предприятие» позволяет вести на любом счете или субсчете количественный учет без подключения аналитического учета. Это может потребоваться, если аналитический учет предполагается вести на субсчетах (каждый субсчет соответствует одному объекту аналитического учета), а не с использованием субконто.

Валютный учет. Система «1С:Предприятие» позволяет вести бухгалтерский учет в нескольких валютах. Счета, на которых необходимо вести валютный учет, устанавливаются в плане счетов. При вводе проводок по таким счетам система «1С:Предприятие» будет запрашивать вид используемой валюты, и суммы в рублях и в валюте. Соответственно, бухгалтерские итоги по таким счетам автоматически будут сохраняться в каждой используемой валюте и в рублевом эквиваленте по каждой валюте отдельно, и суммарно по всем валютам (рублевое покрытие). В качестве справочника валют может быть назначен любой из существующих в системе «1С:Предприятие» справочников. Как правило, для этих целей создастся специальный справочник. Один из числовых реквизитов этого справочника может использоваться для храпения курса валют относительно основной денежной единицы. Тогда при вводе проводок курс валюты будет использоваться для вычисления суммы проводки.

Тема 6. План счетов и регистр бухгалтерии: предназначение, структура и основные свойства

План счетов и его основные свойства как объекта конфигурации. Предопределенные и пользовательские счета. Регистр бухгалтерии: предназначение, структура и основные свойства. Запись движений в регистр бухгалтерии. Итоги регистра бухгалтерии: физические таблицы регистра и расчет итогов. Разделение итогов регистра. Реальные и виртуальные таблицы регистра бухгалтерии для запросов.

План счетов является одним из основных понятий бухгалтерского учета. Планом счетов называется совокупность синтетических счетов, предназначенных для группировки информации о хозяйственной деятельности предприятия. Информация, накапливаемая на таких синтетических счетах, позволяет получить полную картину состояния средств предприятия в денежном выражении. Все счета, входящие в один баланс, объединяются в один план счетов. По свой сути план счетов – это справочник счетов.

Система «1С:Предприятие» предоставляет гибкие возможности по ведению планов счетов. Собственно путем настройки плана счетов и организуется требуемая система учета.

В системе «1С:Предприятие» может быть несколько планов счетов и учет по всем планам счетов можно вести одновременно. Общее число планов счетов, которое может быть организовано в системе, с технической точки зрения неограниченно и определяется исключительно реальными потребностями учета. Например, такой «многоплановый» учет, очевидно, понадобится для совместных предприятий, которым требуется вести учет одновременно по двум или более стандартам бухгалтерского учета.

Субсчета. Планы счетов в системе «1С:Предприятие» поддерживают многоуровневую иерархию «счет - субсчета». Каждый план счетов может включать неограниченное число счетов первого уровня. К каждому счету может быть открыто также неограниченное количество субсчетов. В свою очередь, каждый субсчет может иметь свои субсчета и так далее. Количество уровней субсчетов в системе «1С:Предприятие» неограниченно [2].

Структура кода счета может быть задана при создании плана счетов в виде шаблона, состоящего из произвольной последовательности символов. Технически структура кода счета не влияет на иерархию счетов, однако при создании структуры счетов рекомендуется придерживаться структуры кодов.

Для ведения планов счетов в системе «1С:Предприятие» используются объекты конфигурации План счетов. Объектами данных этого типа являются бухгалтерские счета - учетные регистры, по которым будет выполняться группировка средств при работе с системой «1С:Предприятие». Конфигуратор позволяет создавать практически неограниченное количество планов счетов. Все созданные в конфигураторе планы счетов можно использовать одновременно.

Как описывалось выше, планы счетов в системе «1С:Предприятие» поддерживают многоуровневую иерархию «счет - субсчета».

План счетов может содержать счета, вводимые пользователем, и счета, введенные на этапе конфигурирования. Последние называются предопределенными и являются объектами конфигурации (имеют свое уникальное имя в рамках плана счетов). Нужно отметить, что для плана счетов, в отличие, например, от справочника, который обладает такой же возможностью, ввод счетов в режиме Конфигуратор является в большинстве случаев основным вариантом заполнения. План счетов, ну или по крайней мере его основа (счета первого уровня), создается еще на этапе конфигурирования и во многом определяет логику учета. Ввод новых счетов (субсчетов) пользователем является скорее исключением, чем правилом.

Регистры бухгалтерии

Для отражения в бухгалтерском учете информации о хозяйственных операциях в системе «1С:Предприятие» используются регистры бухгалтерии, описываемые в ветви дерева конфигурации Регистры бухгалтерии.

Свойства регистра бухгалтерии. По своему виду регистр бухгалтерии напоминает регистр накопления. При его редактировании разрабатывается структура регистра: создаются наборы измерений, ресурсов и реквизитов регистра; если необходимо, создаются экранные и печатные формы просмотра движений регистра.

По своей сути регистр бухгалтерии – это «регистр регистров», совокупность регистров. Один регистр бухгалтерии содержит в себе множество учетных регистров (счетов), каждый из которых может иметь независимую аналитику (субконто), в разрезе которой и будут отражаться учетные показатели. В целях контроля все учетные регистры (счета) связаны между собой правилом двойной записи, которое образует замкнутую систему показателей: невозможно изменить значение показателя одного учетного регистра, не затронув другой, причем на ту же сумму.

Счет бухгалтерского учета служит для указания, к какой области учета относится показатель, а набор субконто – измерения, в разрезе которых необходимо его учитывать.

Если рассматривать только синтетическую составляющую плана счетов и вести учет только в денежном выражении, можно интерпретировать план счетов в виде совокупности регистров учета (регистров накопления). При этом каждый счет будет представлен в этой совокупности отдельным регистром учета. Дополнительно должен быть организован еще один регистр, задача которого – хранить обороты между счетами (разделами учета). Таблица Проводки (обороты) будет создана только в том случае, если используется регистр бухгалтерии с поддержкой корреспонденций. Отличительной особенностью регистра бухгалтерии является его связь с планом счетов и поддержка механизма двойной записи. Каждая запись регистра содержит обязательные реквизиты Счет Дт (счет дебета) и Счет Кт (счет кредита) для регистров, поддерживающих корреспонденцию, и реквизит Счет для регистров, ее не поддерживающих.

Создание объекта Регистр бухгалтерии начинается с заполнения его основных свойств [2].

На закладке Основные выбирается план счетов и ***признак поддержки корреспонденций.*** Как планов счетов, так и регистров бухгалтерии может быть несколько, но один регистр бухгалтерии содержит проводки только между счетами одного плана счетов. В то же время на одном плане счетов могут основываться два и более регистра бухгалтерии.

Пример подобного использования: в разрезе счетов одного и того же плана счетов ведется учет свершившихся хозяйственных операций (один регистр бухгалтерии) и планируемых операций (бюджетных транзакций в другом регистре бухгалтерии). Наличие между ними единого связующего звена – плана счетов – позволит получить отчеты для план-фактного анализа, а различная структура регистров позволит вести каждый вид учета удобно и эффективно.

Сравнивая объекты Регистр накопления (или понятие «учетный регистр») и Регистр бухгалтерии, можно сказать, что в самом простом случае регистр бухгалтерии – это регистр, который позволяет учитывать значение какого-то (или каких-то, но как минимум одного – сумма в валюте учета) ресурса в разрезе двух «предопределенных» измерений – СчетДебета и СчетКредита (или Счет и Вид движения для регистра без поддержки корреспонденций). Это скорее образное сравнение, чем демонстрация физического строения таблиц регистра, однако, по нашему мнению, вполне имеет право на жизнь.

Для того чтобы получить самый простой бухгалтерский учет (синтетический: на счетах и субсчетах в денежном выражении в валюте учета), достаточно выбрать в регистре бухгалтерии план счетов, установить (или снять) признак поддержки корреспонденции и ввести в регистр один единственный ресурс, Сумма, например, в котором и будет храниться сумма проводки. В этом случае проводка будет выглядеть так, как показано на рис. 6.1.

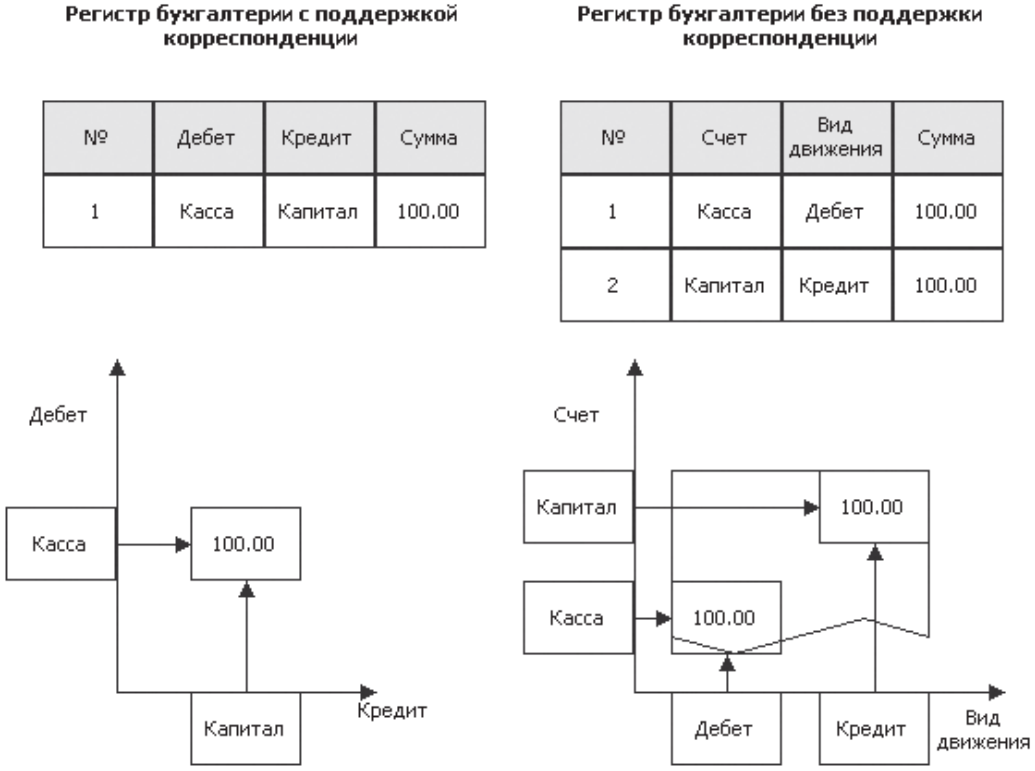


Рис. 6.1. Реализация принципа двойной записи в регистре бухгалтерии

Если регистр поддерживает корреспонденцию, т. е. реализует привычную для России схему учета, каждая проводка включает два корреспондирующих счета (счет дебета и счет кредита) и ресурс Сумма. При такой архитектуре регистра пользователю будут доступны для анализа остатки и обороты по каждому счету и обороты между счетами [2].

Учетная схема без поддержки корреспонденций может считаться более гибкой и в большей степени удовлетворяющей отражению экономической сути хозяйственных операций (хотя бы потому, что не приходится «укладывать» каждую операцию в корреспонденцию счетов, разбивая движения на корреспондирующие пары счетов). Однако отсутствие корреспонденции исключает возможность анализа оборотов между счетами.

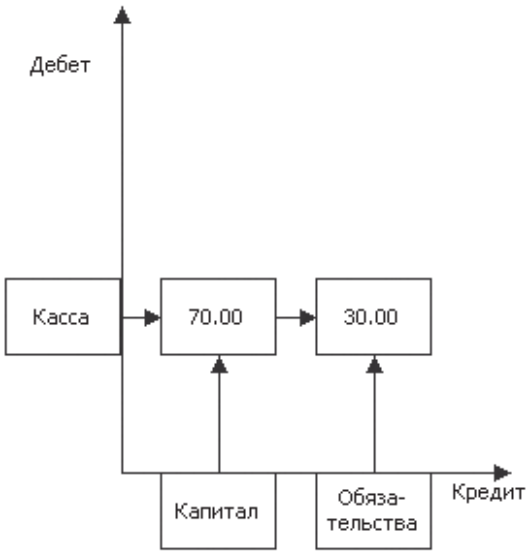
Отсутствие поддержки корреспонденций не означает отсутствие контроля двойной записи. Если корреспонденция поддерживается, контроль двойной записи осуществляется в рамках каждой проводки. Если корреспонденция не поддерживается, то контроль двойной записи осуществляется в рамках всего набора записей, принадлежащих одному регистратору (рис. 6.2).

Регистр бухгалтерии с поддержкой корреспонденции

№	Дебет	Кредит	Сумма
1	Касса	Капитал	70.00
2	Касса	Обяза-тельства	30.00

Регистр бухгалтерии без поддержки корреспонденции

№	Счет	Вид движения	Сумма
1	Касса	Дебет	100.00
2	Капитал	Кредит	70.00
3	Обяза-тельства	Кредит	30.00



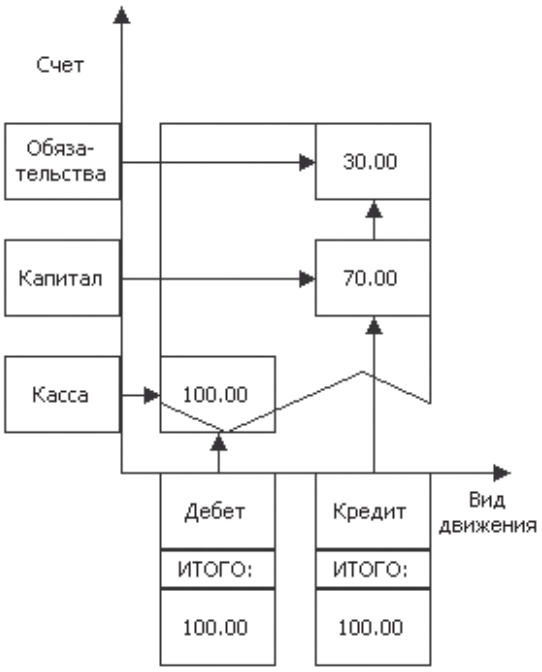


Рис. 6.2. Контроль двойной записи

Контроль двойной записи осуществляется для балансовых счетов, балансовых измерений и балансовых ресурсов. Для регистра без поддержки корреспонденции контроль двойной записи осуществляется для балансового ресурса по балансовому измерению (всем сочетаниям балансовых измерений, если их несколько) в рамках набора записей одного регистратора.

Задача любого регистра – учет значений некоторых показателей (ресурсов) в разрезе некоторых измерений (счета проводки, измерения регистра и дополнительные измерения регистра – субконто) с возможностью оставить дополнительную информацию о каждой записи регистра (реквизиты). Закладка Данные регистра бухгалтерии позволяет определить все перечисленные свойства.

Далее мы подробно рассмотрим свойства Измерения и Ресурсы регистра бухгалтерии. Реквизиты регистра, с нашей точки зрения, столь подробного описания не требуют. Реквизит характеризует лишь запись регистра бухгалтерии и, по сути, является дополнительной информацией, которую можно к этой записи прикрепить.

По значению реквизита можно получить отбор движений, но нельзя получить остаток или оборот по счету или субконто. Типичный вариант использования реквизита регистра – содержание проводки, куда пользователь сможет написать краткий комментарий к сделанной им проводке.

Нужно предостеречь от использования большого числа реквизитов или реквизитов значительной длины. Несмотря на то, что таблицы итогов регистра не включают в себя реквизиты, таблицы первичных движений будут резервировать под них место и как результат увеличивать объем базы данных. Лучше оставлять всю дополнительную информацию в документе регистраторе, который сделал это движение, чем в самом движении.

Свойство *Разрешить разделение итогов* позволяет задействовать механизм разделителя итогов, который обеспечивает более высокую параллельность работы при записи в регистр.

Ресурсы регистра бухгалтерии

Если единственным обязательным разрезом ведения бухгалтерского учета являются счета и субсчета, то единственное значение, которое в этом разрезе нужно учитывать, – это сумма в валюте учета (или валюте составления баланса, можно сказать и так). Для регламентированного учета этой валютой является национальная валюта. Другими словами, для ведения регламентированного бухгалтерского учета достаточно добавить в регистр бухгалтерии один ресурс, который обычно называют Сумма, в который будут записываться значения, выраженные в рублях.

Если речь идет об автоматизации нерегламентированного учета, то ничто не мешает в качестве валюты учеты выбрать любую другую. Например, в качестве валюты управленческого учета стараются выбрать наиболее устойчивую валюту или валюту той страны, в которой расположены основные деловые партнеры (учредители, инвесторы). В РФ управленческий учет чаще всего ведется в долларах США и с недавних пор в евро.

Рассматривая свойства первого ресурса, который был добавлен нами для ведения бухгалтерского учета, следует остановиться на его свойствах.

Ресурс регистра бухгалтерии может иметь только числовой тип, его длиной и точностью мы можем управлять.

Свойство ресурса «Балансовый»

Свойство Балансовый влияет на структуру регистра бухгалтерии. Основная цель бухгалтерского учета – баланс. Основывается баланс на методе двойной записи, в соответствии с которым при движении в учете каждое учитываемое значение (в данном случае сумма в валюте учета) должно «проходить» по двум сторонам проводки: по дебету и по кредиту. При установке свойства Балансовый платформа создаст в регистре бухгалтерии с поддержкой корреспонденции одно новое свойство – Сумма. Таким образом, заполняя сумму проводки, пользователь указывает и сумму, на которую изменяется дебетуемый счет проводки, и ту (ее же), на которую изменяется кредитуемый.

Каждый ресурс регистра бухгалтерии создает новый вид учета. Первый созданный нами ресурс «создал» в регистре «учет в денежном выражении в валюте учета». Добавляя новые ресурсы, мы можем увеличить функциональность нашей «бухгалтерии», расширив виды учета, которые можно здесь вести. Например, в случае необходимости ведения учета натуральных показателей в регистр бухгалтерии будет добавлен новый ресурс Количество.

Добавляя новый ресурс, задумаемся о смысле этого учета. На отдельных счетах материальных ценностей пользователь хочет видеть остатки и обороты по этим счетам не только в стоимостном, но и в натуральном выражении. Само собой разумеется, что контролировать баланс двойной записи по этому ресурсу смысла не имеет. Во-первых, он будет задействован не для всех счетов (какой экономический смысл ресурс Количество будет иметь для счета учета наличных денег Касса, что будет в нем учитываться – монеты, бумажки, и что будет получаться в итоге от сложения монет и бумажек?). А, во-вторых, в этом ресурсе, возможно, будут учитываться движения в различных единицах измерения (учитываться в количественном выражении будут материальные ценности, которые в программе представлены объектами аналитического учета, а они могут храниться в различных единицах измерения). Вывод: баланс не нужен. Значит, ресурс будет небалансовым. Небалансовые ресурсы предназначены для ведения вспомогательных видов учета, которые по большому счету можно было бы вообще в бухгалтерии не вести, а вести, например, в отдельных учетных регистрах оперативного учета.

Причем предоставляет возможность не только их создать, но и обеспечить пользователя возможностью добавлять новые учетные регистры, включая в них различные виды учета (универсальность регистра бухгалтерии!), например, количественный.

Экономический смысл количественного учета, рассмотренный нами выше, предполагает, что пользователь должен иметь возможность для каждого счета проводки (и дебетуемого, и кредитуемого) указать свое значение ресурса Количество. При использовании регистра бухгалтерии без поддержки корреспонденций, в котором дебетуемые и кредитуемые счета располагаются в

разных записях, в регистр будет добавлено новое поле Количество. Единственным отличием его от поля Сумма будет отсутствие контроля двойной записи. То есть при записи набора движений в базу данных платформа разрешит записать набор, если итог по ресурсу Количество дебетуемых записей не будет равен итогу по этой же графе кредитуемых.

В регистре бухгалтерии с поддержкой корреспонденции каждая запись включает сразу два счета, положение которых в проводке (дебетуется или кредитуется) и определяет вид движения по счету. Чтобы предоставить возможность независимого заполнения ресурса для каждой из сторон проводки, платформа добавляет на каждый небалансовый ресурс два поля регистра бухгалтерии. Они отличаются постфиксами (Дт и Кт), подсказывающими, к какому счету проводки поле относится. В данном случае будут добавлены поля КоличествоДт и КоличествоКт.

Следует заметить, что дополнительные виды учета можно создавать, добавляя не только небалансовые ресурсы регистра бухгалтерии, но и балансовые. Примером использования в регистре бухгалтерии более одного балансового ресурса может служить трехвалютный учет.

При таком виде учета предприятие, входя в состав холдинга, вынуждено готовить отчетность в двух валютах учета, которыми, например, могут быть национальная (для РФ – рубли) и валюта учета холдинга (если руководство холдинга европейское, видимо, это евро).

Первое решение, которое приходит в голову, – просто взять всю бухгалтерскую отчетность и пересчитать значение каждого ее показателя по курсу на сдачу отчетности. Однако на такое решение могут возникнуть замечания со стороны руководства холдинга, ведь в течение отчетного периода соотношения курсов национальной валюты и валюты холдинга менялись. И как результат необходимо пересчитывать не итоговые показатели отчетности, а сумму каждой проводки по историческому курсу (т. е. по курсу на дату проводки). Пересчитывать и где-то хранить результат, чтобы можно было быстро получить отчетность. Можно было бы пересчитывать каждый раз, а не хранить, но пересчета по историческому курсу будут требовать не только обороты за период, но и входящие остатки, т. е. считать каждый раз нужно будет с самого начала.

Поэтому проще в регистр бухгалтерии добавить еще один балансовый ресурс, который, скорее всего, не будет доступен пользователю для заполнения и изменения, а будет рассчитываться при записи набора движений в регистр. И при составлении отчетности нам останется лишь обратиться к показателям остатков и оборотов по новому виду учета.

Измерения регистра бухгалтерии

Все измерения регистра бухгалтерии можно условно разделить на собственно измерения (которым и посвящен этот раздел и которые создаются на закладке Данные свойств регистра бухгалтерии), дополнительные измерения (виды субконто) и обязательные измерения (встроенные, предопределенные, разделы учета – ни одно из этих понятий не является «штатным» описанием механизма, но позволяет понять суть явления), которыми для регистра

бухгалтерии с поддержкой корреспонденции являются СчетДебета и СчетКредита, а для регистра без поддержки корреспонденции – Счет и Вид движения.

Задачи, которые решаются с помощью измерений регистра бухгалтерии, как и сами измерения из ветки Измерения закладки Данные регистра бухгалтерии, можно разделить на две группы: ведение раздельных балансов и получение сводной отчетности (балансовые измерения) и разделение отдельных (или всех, но встречается это реже) разделов учета по какому-то признаку.

Свойство измерения «Балансовый»

Балансовые измерения делят весь учет (все разделы учета) на самостоятельные балансы. Баланс, как известно, – замкнутая система показателей. Поэтому если в процессе конфигурирования было добавлено балансовое измерение Организация, значит, по каждой организации (в данном случае список организаций ведется в справочнике Организации) программой будет составлен свой баланс. Это значит, что у каждой организации будут свои активы (касса, расчетные счета, склады и др.), свои обязательства (перед сотрудниками, перед поставщиками) и свой собственный капитал [2].

Наличие балансового измерения в регистре бухгалтерии совершенно исключает возможность, например, переместить товары с одной организации в другую. По каждой организации ведется свой баланс. А значит, для решения задачи перемещения товаров нужно в одной организации их списать (на какой-нибудь счет расчетов с той, другой, организацией), а в другой организации их оприходовать (со счета учета расчетов с первой организацией).

Эта учетная схема – всего лишь один из вариантов ведения учета, и была приведена не как руководство к действию, а как демонстрация всей серьезности шага добавления балансового измерения в регистр бухгалтерии.

Балансовое измерение добавляет новое поле во все таблицы регистра бухгалтерии. При записи набора движений в регистр осуществляется контроль двойной записи по каждому значению (сочетанию значений, если балансовых измерений несколько) балансового измерения.

Небалансовое измерение по своей сути напоминает субконто. Так же как и субконто, оно делит не все разделы учета на самостоятельные балансы, а, как правило, лишь некоторые, или все разделы учета, но баланса по нему получить нельзя. Для регистра с поддержкой корреспонденции добавляются два поля – по одному на каждую сторону проводки, например, ВалютаДт и ВалютаКт. Для регистра без поддержки поле добавляется одно, но при записи набора движений в регистр контроль двойной записи по каждому значению такого измерения не осуществляется.

Небалансовое измерение можно сравнить с субконто, которое в обязательном порядке прикрепили ко всем счетам. Хотя хранение в физических таблицах регистра у них различное, функционально механизмы небалансовых измерений и субконто похожи (табл. 6.1).

Таблица 6.1

Сравнение небалансовых измерений и субконто

Критерий	Небалансовое измерение	Субконто
Компактность хранения в базе	Поле резервируется даже для счетов, где измерение не используется	Резервируется только в том случае, если субконто на счете используется
Тип данных	Любой, чаще просто ссылка на справочник	Всегда составной, причем состав определятся для плана видов характеристик
Возможность включить/отключить для конкретного счета	Да, с помощью признаков учета	Да, при подключении вида субконто на счет
Всегда на своем месте	Да	Может быть любым по порядку на счете, и для пользовательских счетов порядок может быть задан произвольно пользователем

Таким образом, можно предложить использование небалансового измерения вместо субконто в том случае, если данный признак аналитики используется на всех (или на подавляющем большинстве) счетах.

Более того, в этом случае можно считать использование измерения более оптимальным, чем субконто, что связано с типом данных. Приведем пример: организации необходимо вести учет почти всех разделов учета в разрезе источников финансирования. Для решения этой задачи можно воспользоваться или добавлением субсчетов к нужным (почти всем) счетам, или добавлением на них нового вида субконто, или добавлением в регистр нового небалансового измерения. У каждого решения есть свои плюсы и минусы (табл. 6.2).

Таблица 6.2

Сравнение субсчетов, субконто и небалансовых измерений

Решение	Преимущества	Недостатки
1	**2**	**3**
Субсчета	Может потребовать незначительного изменения типового решения, если оно с самого начала предусматривало возможность расширения субсчетов второго или третьего уровня	Сложность получения отчетности по одному источнику финансирования и сложность ведения учета из-за значительного числа счетов в плане счетов

Продолжение табл. 6.2

1	2	3
Субконто	Может потребовать незначительного изменения типового решения, если уложиться в максимальное число субконто для плана счетов	Может потребовать значительных изменений, если все субконто для отдельных счетов уже заняты и нужно увеличивать максимальное число субконто. Увеличение максимального числа субконто может негативно повлиять на производительность
Небалансовое измерение	Позволит разделить учет более оптимально, чем в случае субконто, и удобно подготавливать отчетность, как по значению измерения, так и сводно	Потребует значительных изменений типового решения

Еще один вариант использования небалансовых измерений – добавление нового вида учета, требующего не только учета нового показателя (ресурса), но и ведения нового разреза для этого и других (например, Сумма) ресурсов. Даже если этот новый вид учета используется далеко не на всех счетах, а место под новое измерение будет все равно резервироваться для всех счетов, использование механизма небалансовых измерений оправдано необходимостью создать универсальный механизм, который сможет использовать и пользователь системы.

Примером такого использования может служить валютный (многовалютый, мультивалютный) учет. Суть учета состоит в том, что для отдельных (не всех) счетов необходимо учитывать не только показатели в валюте учета (ресурс Сумма), но и в иностранных валютах. Если бы иностранная валюта была одна, этот вид учета ничем не отличался бы от учета количественного, который мы рассмотрели в разделе, посвященном признакам учета ресурса регистра бухгалтерии. Но валют может быть много, и по каждой из них мы хотим видеть остатки и обороты, как в валюте, так и валюте учета. Другими словами, нам необходим разрез по валютам. Развивая тему количественного учета и доводя его до абсурда, можно предложить схожее решение для задачи количественного учета в разрезе единиц измерения. В этом случае к небалансовому ресурсу Количество нам пришлось бы добавить небалансовое измерение Единица измерения и связать их признаком учета, чтобы пользователь мог одним щелчком мыши включить новый вид учета на счете. Подчеркнем еще раз – это именно пример абсурдного использования регистра бухгалтерии.

Для решения задач наличия и движения средств с целью управления ими (а не предприятием в целом, как в нашем случае) существуют механизмы оперативного учета.

Возвращаясь к задаче ведения валютного учета, который, как правило, является обязанностью бухгалтерии, заметим, что кроме справочника Валюты нам потребуется еще один связанный с ним объект – периодический регистр сведений Курсы валют. Регистр будет иметь измерение Валюта типа Ссылка на справочник Валюты и ресурс Курс типа Число, а если требуется вести учет в валютах с очень маленькими значениями курса, то два ресурса – Курс и Кратность, и для получения эквивалента в валюте учета валютную сумму нужно будет сначала умножить на курс, а затем разделить на кратность.

Какие изменения коснутся механизмов бухгалтерии? Во-первых, необходимо добавить новый признак учета в плане счетов Валютный. Он позволит отметить те счета, которые требуют ведения валютного учета. Во-вторых, нужно добавить новый ресурс регистра бухгалтерии ВалютнаяСумма, где будет храниться сумма в валюте. И, в-третьих, добавить новое небалансовое измерение регистра бухгалтерии Валюта – ссылка на элемент справочника Валюты, по которому будет «разделен» учет отмеченного признаком счета.

Схему взаимодействия объектов можно изобразить графически (рис. 6.3) [2].

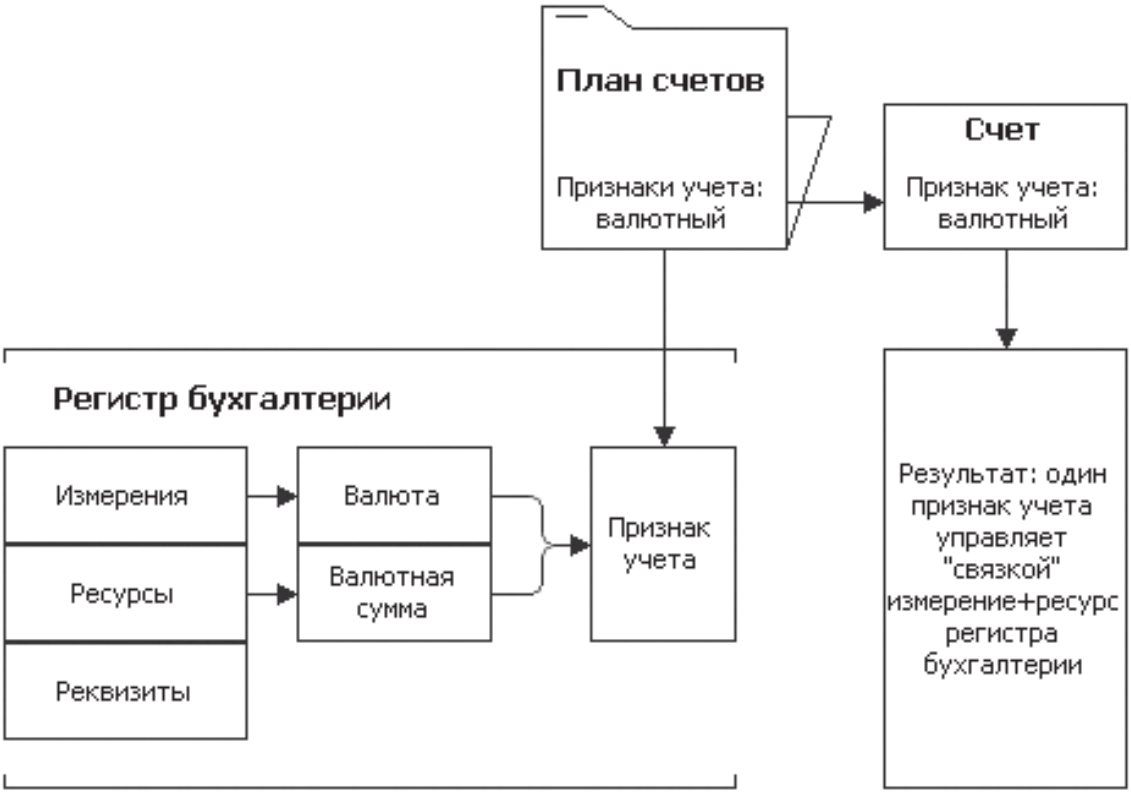


Рис. 6.3. Схема взаимодействия объектов

Результат: при установке одного флажка в настройках счета для счета становятся доступными для заполнения новый ресурс и новое измерение.

Реальные и виртуальные таблицы регистра бухгалтерии для запросов приведены на рисунке 6.4.

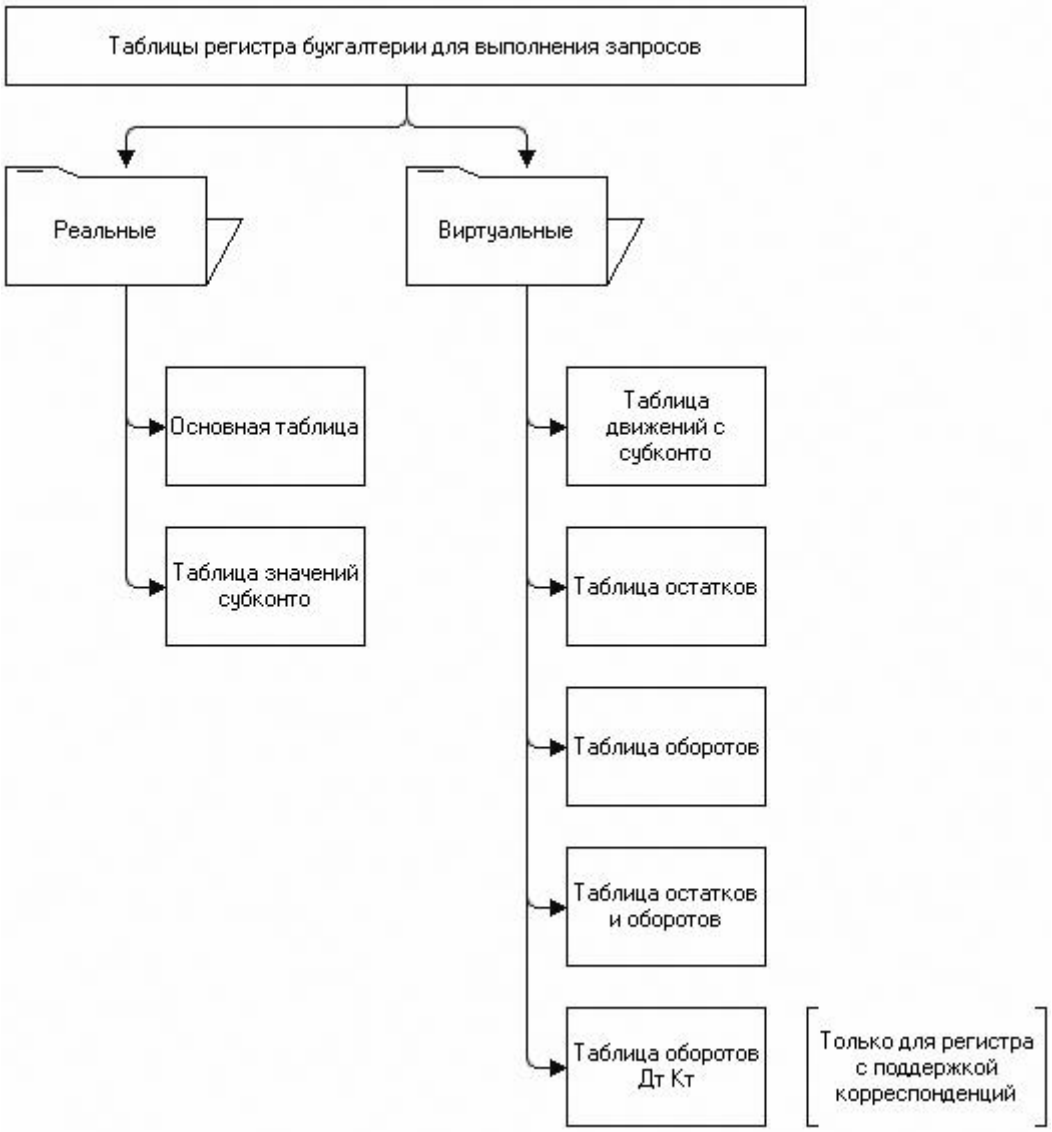


Рис. 6.4. Реальные и виртуальные таблицы регистра бухгалтерии для запросов

Две группы таблиц регистра бухгалтерии: реальные таблицы и виртуальные таблицы.

С помощью ***реальных таблиц*** доступ посредством запроса предоставляется к физически существующим в информационной базе таблицам (основная таблица и таблица значений субконто). Но они являются скорее вспомогательным механизмом, т.к. малопригодны для формирования большинства бухгалтерских отчетов.

Виртуальные таблицы создаются системой при обращении к ним и не хранятся в информационной базе (можно считать вложенными запросами, которые выполняются системой к физическим таблицам регистра бухгалтерии). Они параметризированы (можно передать в качестве параметров условия выполнения запроса). В табл. 6.3 приведены сведения о параметрах, используемых для разных виртуальных таблиц регистра бухгалтерии, а в табл. 6.4 представлены поля в виртуальных таблицах регистра бухгалтерии.

Таблица 6.3

Параметры виртуальных таблиц регистра бухгалтерии

Параметр	Таблица оборотов	Таблица оборотов Дт Кт	Таблица остатков	Таблица остатков и оборотов	Таблица движений с субконто
Период, НачалоПериода, КонецПериода	Да	Да	Да	Да	Да
Условие	Да	Да	Да	Да	Да
УсловиеСчета, УсловиеКорСчета, УсловиеСчетаДт, УсловиеСчетаКт	Да	Да	Да	Да	
Субконто, КорСубконто, СубконтоДт, СубконтоКт	Да	Да	Да	Да	
Периодичность	Да	Да		Да	
МетодДополнения				Да	

Таблица 6.4

Поля виртуальных таблиц регистра бухгалтерии

Параметр	Таблица оборотов	Таблица оборотов Дт Кт	Таблица остатков	Таблица остатков и оборотов	Таблица движений с субконто
Счет	Да		Да	Да	
КорСчет	Да				
Счет Дт/Кт		Да			Да
<Измерение>	Да	Да	Да	Да	Да
<Измерение>Кор	Да				
<Измерение>Дт/Кт		Да			Да
Субконто<N>	Да		Да	Да	
КорСубконто<N>	Да				
СубконтоДт<N>/Кт<N>		Да			Да
Период	Да*	Да*		Да*	Да
Регистратор	Да*	Да*		Да*	Да
НомерСтроки	Да*	Да*		Да*	Да
МоментВремени					Да
Активность					Да
<Реквизит>					Да
ВидСубконтоДт<N>/Кт<N>					Да
<Ресурс>					Да
<Ресурс> Дт/Кт					Да
<Ресурс>Остаток			Да	Да**	
<Ресурс>ОстатокДт/Кт			Да	Да**	
<Ресурс>РазвернутыйОстатокДт/Кт			Да	Да**	
<Ресурс>Оборот	Да	Да		Да	
<Ресурс>ОборотДт/Кт	Да	Да		Да	
<Ресурс>КорОборот	Да				
<Ресурс>КорОборотДт/Кт	Да				

** Поля присутствуют в таблице при установленном значении параметра **Периодичность**; поле **НомерСтроки** доступно, если периодичность равна **Запись**; поле **Регистратор** доступно, если периодичность равна **Запись** или **Регистратор**.*

*** В таблице присутствуют поля для получения остатков на начало и конец периода, заданного в параметрах таблицы.*

При разработке запросов к виртуальным таблицам (не только регистра бухгалтерии) при необходимости отбора информации в соответствии с какими-либо условиями следует руководствоваться следующим правилом: все условия на измерения должны формулироваться в параметрах виртуальной таблицы, а не в секции ГДЕ запроса. Связано это с тем, что виртуальные таблицы регистра бухгалтерии не хранятся и формируются при обращении к ним. Следовательно, если параметры не указаны строится сначала полная таблица, а потом из нее отбирается некоторая (возможно очень небольшая) часть информации, удовлетворяющая условиям отбора. А остальная часть сформированной виртуальной таблицы отбрасывается. А системные ресурсы на ее формирование потрачены! Поэтому гораздо более эффективным и правильным способом указания условий отбора является их указание в параметрах виртуальной таблицы, чтобы система уже при построении виртуальной таблицы сразу рассчитывала только записи, удовлетворяющие условиям.

Как видно из табл. 6.3 во всех виртуальных таблицах регистра бухгалтерии присутствуют параметры «Период» или «НачалоПериода» и «КонецПериода», параметр «Условие». Отбор итогов или на момент времени, или за период обеспечивают первые три параметра. В их можно задавать значения типа Дата, МоментВремени или Граница.

Помимо табличного способа доступа к данным регистра бухгалтерии можно получить доступ, используя методы менеджера регистра бухгалтерии. Есть три метода: Остатки(); Обороты; ОстаткиИОбороты(). Назначение этих методов – получить таблицу значений с полями, указанными в параметрах метода. Схоже с использованием виртуальных таблиц с теми же названиями.

Пример:

Результат = Регистр.Остатки(Момент, , Отбор, "Субконто1, Субконто2", "Количество, Сумма");

Строго говоря, при обращении к этим методам система все равно выполняет запрос. Поэтому разница в табличном или объектном способе доступа к регистру бухгалтерии только в синтаксисе.

Одним из факторов, влияющим на производительность запросов к виртуальным таблицам, является порядок прикрепления субконто к счетам. Субконто, по которым отбор будет производиться наиболее часто, лучше располагать в начале. Кроме того, по возможности при одинаковой аналитике на разных счетах порядок расположения тоже лучше сделать одинаковым, т.е. если он будет различным, то при обращении в запросе к разным счетам при построении виртуальной таблицы строится два подзапроса.

Тема 7. Основы организации аналитического учета в системе «1С:Предприятие»

Основы организации аналитического учета в системе "1С:Предприятие": сквозная аналитика, обычная аналитика, опционная аналитика. Организация обычной аналитики с помощью субконто. План видов характеристик и виды субконто. Настройка плана счетов для аналитического учета.

Информация о средствах предприятия, которая накапливается на счетах бухгалтерского учета, зачастую имеет обобщающий характер. Например, в «стандартном» плане счетов существует счет 10 «Материалы», который предназначен «…для обобщения информации о наличии и движении принадлежащих предприятию сырья, материалов, топлива, запасных частей, тары и т. п. ценностей». К счету 10 могут быть открыты субсчета для учета различных видов материалов. Но при такой организации учета на субсчетах будет накапливаться информация об общей стоимости материалов одного вида, а в целом на счете 10 - стоимость всех материалов.

Для получения детальной информации о наличии конкретных материалов необходима организация аналитического учета по материалам. В этом случае общие суммы на субсчетах разбиваются на более мелкие - стоимости конкретных материалов.

Под аналитическим учетом будем понимать учет более детальный, чем синтетический, т.к. аналитические отчеты, как правило, являются более подробными, чем синтетические, или, наоборот, синтетические отчеты можно считать сводными относительно отчетов аналитических.

Иерархический план счетов позволяет организовать детальный учет. И при желании можно организовать детальный аналитический учет, используя возможность организации иерархического плана счетов и ввода новых субсчетов пользователем. Однако эта детализация будет иерархической. При такой организации аналитического учета не представляется возможным получение итогов по нескольким параллельным срезам.

Рассмотрим учет товаров на складах. Предположим, в нашей организации есть несколько складов, на каждом из которых множество видов номенклатурных позиций. Перед нами стоит задача обеспечить получение остатков и оборотов по каждому складу и товару в отдельности и сводно по всем. Задачу можно решить, используя субсчета счета Товары. Изменив длину кода счета (до 10 символов) и маску кода счета @@.@@.@.@@, мы можем завести к счету учета товаров необходимое количество субсчетов [3].

Подобная организация аналитического учета возможна, но имеет ряд существенных недостатков. Во-первых, план счетов будет расти (и расти существенно) по мере изменения номенклатуры товаров (добавления новых позиций, отказа от старых и появления неиспользуемых). Во-вторых, он будет расти нелинейно, т. к. одна и та же позиция номенклатуры может храниться на разных складах; таким образом, количество субсчетов на каждую позицию будет равняться количеству складов. И самое главное – не представляется возможным

автоматически сгруппировать данные по одной позиции номенклатуры, хранимой на разных складах.

Организацию аналитического учета с использованием субсчетов нельзя назвать неправильной. Это вполне возможный вариант, который имеет свои ограничения (мы перечислили их выше) и свои достоинства. К достоинствам можно отнести, например, интуитивную прозрачность учета для любого бухгалтера, который ранее не сталкивался с учетом при помощи компьютерной программы, всю жизнь проработал в «бумажной бухгалтерии».

Однако подавляющее большинство бухгалтеров предпочитают бухгалтерские программы с большими возможностями в части ведения аналитического учета.

Можно выделить три варианта реализации аналитического учета [3]:

– *сквозная аналитика*, при которой один и тот же аналитический разрез используется на всех или почти на всех счетах;

– *обычная аналитика,* при которой на разных счетах должна присутствовать различная аналитика в разном количестве параллельных срезов;

– *опциональная аналитика,* которая является разновидностью обычной, но может добавляться или убираться со счетов пользователя без участия программиста. Используется механизм функциональных опций.

Сквозная аналитика организуется с помощью небалансовых измерений регистра бухгалтерии и признаков счета для управления доступностью измерения дебета/кредита записи регистра в зависимости от настройки выбранного счета дебета/кредита проводки.

Опциональная аналитика: если конфигурация разрабатывается с расчетом на множество организаций, которые должны ее использовать с минимальными изменениями, тогда чем больше вариантов настройки программист вынес на опции, которыми можно управлять пользователем, тем она лучше. Такими опциями, например, может управляться настройка аналитического учета на некоторых счетах.

Под *обычной аналитикой* будем понимать детализацию учета таким образом, что на разных счетах есть возможность установить разные аналитические разрезы и разное их количество (на одном может вообще не быть аналитики, другой может иметь один, два или более параллельных срезов). Для создания такого механизма необходимо:

– хранить где-то список возможных аналитических разрезов, причем каждый разрез (вид субконто, аналитический счет) должен помнить тип своих значений;

– описать максимально возможное количество срезов для каждого регистра бухгалтерии (плана счетов);

– указать, на каких счетах, какие разрезы будут использоваться и в какой последовательности.

Понятия Вид субконто и Субконто. Вид субконто – группа однородных объектов аналитического учета. Понятие «вид субконто» соответствует понятию бухгалтерского учета «аналитический счет».

Каждый элемент списка вида субконто представляет собой объект аналитического учета и называется Субконто. Ни субконто, ни вид субконто не имеют объектного представления в системе и существуют лишь как понятия, реализуемые с помощью других объектов. Виды субконто, как правило, бывают ссылочного типа и содержат в себе ссылку на справочник, документ или перечисление, хотя возможны и другие варианты. Если видом субконто является справочник, то субконто – один элемент этого справочника. Сами по себе виды субконто не хранят никакой информации об остатках и оборотах по счетам и являются, по сути, подключаемыми измерениями для счетов плана счетов. Термином «субконто» могут быть обозначены любые объекты аналитического учета: основные средства, нематериальные активы, материалы, организации, подотчетные лица, договоры, бюджеты. Видом субконто, в свою очередь, называется множество однотипных объектов аналитического учета.

Использование видов субконто для организации аналитического учета можно представить в виде схемы (рис. 7.1) [3].

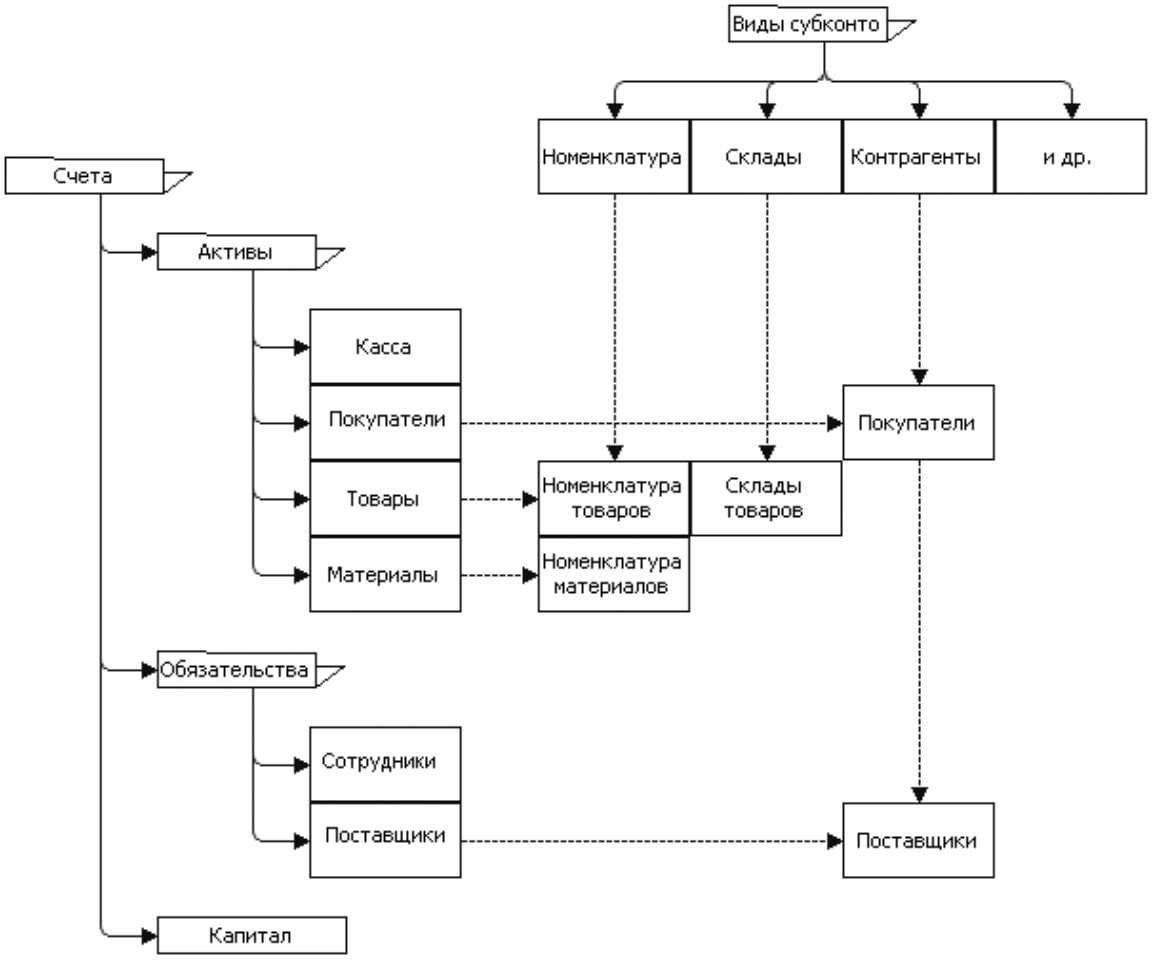


Рис. 7.1. Использование видов субконто для организации аналитического учета

Левая часть схемы описывает наш иерархический план счетов. Каждый из счетов предназначен для ввода и хранения относительно укрупненной информации – остаток всех товаров на всех складах, долг всех покупателей, задолженность перед всеми поставщиками и др. Вместо того, чтобы развивать субсчета у названных счетов, мы открываем на них аналитический учет,

используя виды субконто. Каждый счет на данной схеме можно рассматривать как учетный регистр, накапливающий остатки и обороты в денежном выражении. Виды субконто позволяют добавлять измерения в эти регистры и создают новые сущности. Так, например, один и тот же справочник Контрагенты может использоваться для организации аналитического учета как на счете Покупатели, так и на счете Поставщики. Анализируя остатки и обороты по счету Покупатели в разрезе объектов аналитического учета (контрагентов), мы получим список наших покупателей, остатки (задолженность) и обороты (увеличение, погашение) по каждому из них.

Возможность системы привязать к одному счету несколько самостоятельных видов субконто позволяет нам получить фасетное (параллельное) деление: например, на одном складе могут быть разные товары, один и тот же товар может быть на разных складах. Мы всегда можем получить остаток по складу, по товару и по товару на складе.

Для хранения списка видов субконто используется объект План видов характеристик. Получаются два уровня вложенности: план, представляющий собой совокупность видов субконто, и вид субконто. Каждый вид субконто хранит информацию о типе характеристик, являющихся объектами аналитического учета, т. е. субконто. Характеристика – это один экземпляр вида характеристик, так же как субконто – это один экземпляр вида субконто. Характеристика не является самостоятельным объектом конфигурации. Для ведения аналитического учета, как правило, используются элементы справочников, значения перечисления, ссылки на документы. Виды характеристик, как правило, имеют ссылочный тип данных (хотя возможны и другие варианты, однако, для плана видов характеристик, используемого в качестве видов субконто плана счетов, не рекомендуется использовать примитивные типы. Это может существенно сказаться на производительности при записи движений регистра бухгалтерии. Рекомендуется использовать только ссылочные типы). Один план видов характеристик может использоваться для организации аналитического учета для нескольких планов счетов. Можно провести следующие параллели между понятиями системы и понятиями предметной области (табл. 7.1).

Таблица 7.1

Соответствие понятий системы «1С:Предприятие» и понятий предметной области

Объект	Понятие
Планы видов характеристик	Все виды субконто
Вид характеристик	Вид субконто
Характеристика	Субконто

Так как в качестве объектов аналитики могут выступать данные разных объектов (разных справочников, как в нашем случае, или справочников и документов и перечислений и др.), то тип значения характеристик устанавливается составным. Фактически в диалоге редактирования типа характеристик мы выбираем объекты конфигурации, которые будут

использоваться для ведения аналитического учета. Выбираем, какого типа виды субконто мы (и пользователь) сможем создавать. Нужно отметить, что поле диалога для выбора значения субконто в документах, отчетах и других механизмах будет иметь тип Характеристика.ВидыСубконто плана видов характеристик ВидыСубконто. А тип значения поля в базе данных будет составным и будет включать все выбранные в типе значения характеристик типы. Конфигуратор системы «1С:Предприятие» позволяет создать любое количество видов субконто в соответствии с требованиями полноты аналитического учета на предприятии [3].

Для предоставления пользователю возможности создавать собственные виды субконто предусмотрен механизм дополнительных значений характеристик. Для этого создается новый справочник в конфигурации, который мы назовем, например, Субконто. Потом включим его в тип значения характеристик и получим таким образом «местечко про запас», где пользователь сможет хранить собственные объекты аналитики. Теперь пользователь сможет создать сколько угодно видов субконто, имеющих тип этого справочника, но все

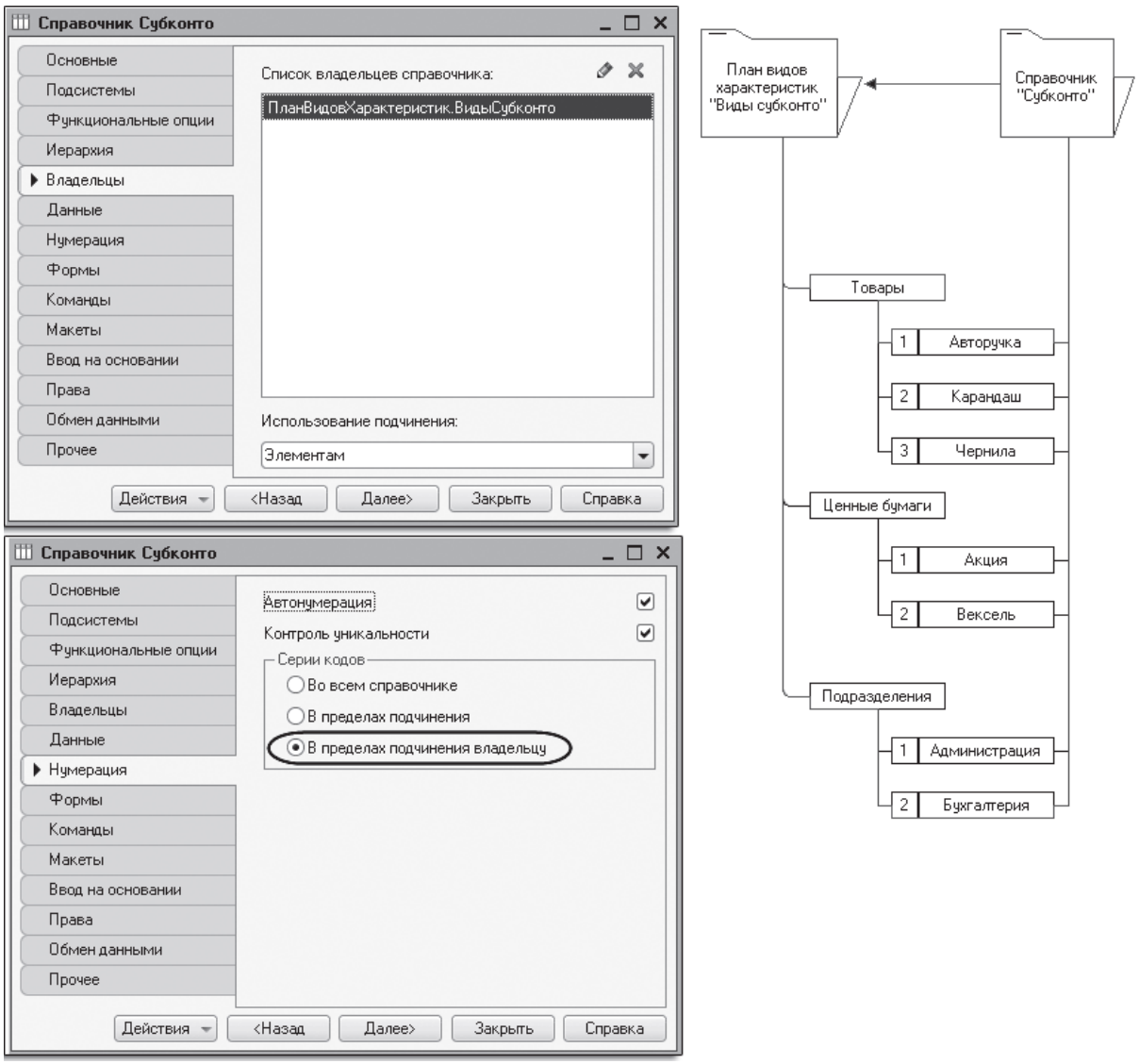


Рис. 7.2. Владельцы справочника «Субконто»

его объекты аналитики будут лежать «вперемешку». Чтобы предоставить пользователю возможность создавать отдельные подмножества объектов аналитики, справочник Субконто необходимо сделать подчиненным плану видов характеристик. Подчиненность справочника Субконто плану видов характеристик – обязательное условие функционирования механизма дополнительных характеристик (рис. 7.2).

Тем самым мы предоставили пользователю возможность создать сколько угодно новых видов субконто. Физически все они будут храниться в одном справочнике, но «перемешиваться» не будут, т. к. на каждый новый вид субконто в справочнике будет выделяться новое множество подчиненных ему элементов. А чтобы нумерация этих элементов была также независимой, имеет смысл на закладке Нумерация объекта конфигурации Справочник Субконто установить переключатель Серии кодов в значение В пределах подчинения владельцу.

Еще необходимо выбрать справочник Субконто в качестве дополнительных значений характеристик. Теперь вновь создаваемые пользователем виды субконто (виды характеристик) по умолчанию будут иметь тип значения СправочникСсылка.Субконто.

Теперь, когда созданы «аналитические счета» – виды субконто, нужно, чтобы они заработали и позволили вести аналитический учет на счетах, которые ранее были синтетическими. Для этого необходимо привязать виды субконто к счетам плана счетов.

Выполняется это в объекте План счетов. Начинаем настройку с закладки Субконто. На этой закладке в свойстве Виды субконто выбираем план видов характеристик, который хранит виды субконто для этого плана счетов.

Закладка Субконто также позволяет определить максимальное количество субконто на счете. Это число параллельных аналитических срезов (видов субконто), которое можно будет выбрать для каждого счета. И число это в большинстве случаев определяется отчетными потребностями объекта автоматизации. В типовом решении «Бухгалтерия предприятия» выбрано значение 3, максимально возможное поддерживаемое платформой значение 50. Выбор максимального количества субконто – важный этап в проектировании системы автоматизации бухгалтерского учета.

Установка этого свойства привела к тому, что для каждого счета плана счетов было добавлено новое свойство – специализированная табличная часть Виды субконто. Каждая из строк табличной части содержит ссылку на вид характеристики плана видов характеристик, выбранного в свойстве плана счетов Виды субконто, и дополнительные свойства. Каждая строка табличной части ВидыСубконто содержит следующие поля: ВидСубконто, Предопределенное, ТолькоОбороты и признаки учета субконто [3].

ВидСубконто – свойство типа ПланВидовХарактеристикСсылка.<имя>, позволяющее получить доступ к свойствам вида характеристики плана видов характеристик. Это поле задает вид субконто, в разрезе которого будет вестись аналитический учет. Также оно может быть использовано, например, при установке типов значений для реквизитов формы или для полей построителя

отчета (рис. 7.3).

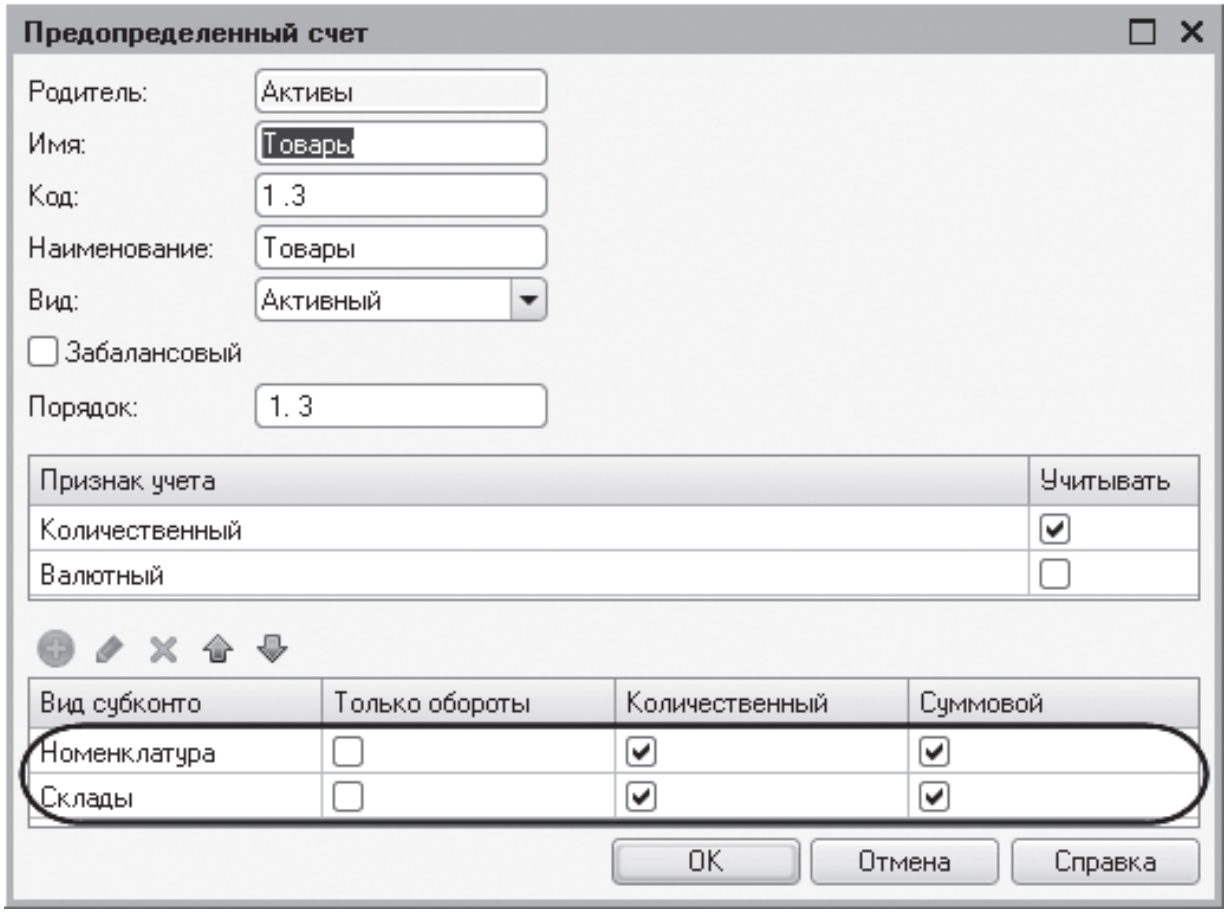


Рис. 7.3. Виды субконто

Пример использования свойства «ВидСубконто» [3]:

```
Для Каждого ВидСубконто Из Счет.ВидыСубконто Цикл
    Вид = ВидСубконто.ВидСубконто;
    ТипЗначения = Вид.ТипЗначения;
    Наименование = Вид.Наименование;
    ВведенВКонфигураторе = ВидСубконто.Предопределенное;
    Оборотный = ВидСубконто.ТолькоОбороты;
    Количественный = ВидСубконто.Количественный;
КонецЦикла;
```

В приведенном примере счет – переменная или поле ввода типа ПланСчетовСсылка.<имя>. Цикл по видам субконто счета позволит узнать их типы, наименования, были они введены в конфигураторе (предопределенный вид субконто был добавлен на счет в режиме Конфигуратор) или добавлены пользователем, узнать признаки учета субконто.

Теперь, когда созданы «аналитические счета» – виды субконто, нужно, чтобы они заработали и позволили вести аналитический учет на счетах, которые ранее были синтетическими. Для этого необходимо привязать виды субконто к счетам плана счетов.

От организации плана счетов и аналитического учета во многом зависит не только функциональность, но и производительность конфигурации.

Остановимся подробнее на наиболее важных моментах организации аналитического учета в конфигурации. Как правило, организация аналитического учета определяется отчетными потребностями руководства, организационной структурой предприятия, необходимостью выполнения стандартов учета (международных, государственных, внутрифирменных и т. д.). Рассмотрим пример организации учета товаров.

Пользователь изъявил желание вести учет товаров в разрезе номенклатуры товаров. Желание разумное, ведь подобная организация аналитического учета позволит ему анализировать остатки и обороты по каждой номенклатурной позиции. Список товаров существенный и незакрытый (не добавляются новые позиции, удаляются старые). Кроме кода и наименования номенклатуры необходимо хранить и другую дополнительную, но важную информацию. Ниже приводится таблица (табл. 7.2), позволяющая в конкретном случае принять решение об оптимальности использования механизма платформы.

Таблица 7.2

Решение по выбору субсчета или субконто

Критерий	Учет на субсчетах	Учет на субконто
Количество позиций	От единиц до десятков	От десятков до бесконечности
Список позиций учета закрыт	Как правило, да	Может и будет расширяться
Параллельные срезы возможны	Только иерархический учет	Несколько параллельных срезов
Достаточная информативность	Кода и Наименования достаточно	Есть дополнительные свойства

Принимаем решение: вести учет на счете товаров в разрезе субконто Номенклатура типа СправочникСсылка.<имя>. Следующее желание пользователя – получать дополнительные группировки и отборы в отчетах по складам, что соответствует организационной структуре предприятия, и списывать товары методом FIFO. Дополнительно потребуется группировка в отчете Остатки товаров по годам приобретения имеющихся на складе МПЗ и видам товаров. По партиям и складам, основываясь на предыдущей таблице, принимаем решение о необходимости открытия новых видов субконто Партии и Склады на счете учета товаров. По видам товаров и годам закупки МПЗ решение может быть проще (табл. 7.3).

Принимаем решение: вид товара – это новый реквизит справочника Номенклатура, год можно получить из партии товара. Стало быть, добавления новых видов аналитики эти два критерия не потребовали.

В результате пришли к выводу, что для получения затребованной информации потребуется ведение аналитического учета на счете товаров в разрезе трех независимых видов аналитики: номенклатура, склады и партии.

Таблица 7.3

Решение по выбору: новый вид субконто или новое свойство старого

Критерий	Новый вид субконто	Новое свойство субконто
Можно ли однозначно связать новый критерий отбора (группировки) с существующим объектом аналитического учета	Связать невозможно, это параллельный срез	Новый критерий вполне можно разместить как новое свойство существующего вида аналитики

Следующий вопрос: порядок следования видов аналитики. Физическая организация таблиц позволяет наиболее быстро получать развернутые отчеты по тем аналитическим срезам, которые находятся «ближе к счету» (табл. 7.4).

Таблица 7.4

Решение по выбору порядка следования видов субконто

Критерий	Первый на счете	Последний на счете
По какому аналитическому срезу отчеты чаще всего будут «разворачиваться», по какому реже	Номенклатура	Склады

Принимаем решение: первый вид субконто на счете будет Номенклатура, второй Партии, третий Склады.

Следующая наша задача – оценить возможность использования трех видов субконто на счете со стороны типового (или просто существующего) решения, которое мы адаптируем, и со стороны платформы.

Любое типовое решение ориентировано, как правило, на фиксированное заданное максимальное количество субконто на счете. Так, например, конфигурация «Бухгалтерия предприятия» содержит в себе набор документов, среди которых есть универсальные, и отчетов, предназначенных для работы максимум с тремя видами субконто на счете (три поля для выбора). Кроме того, важно оценить влияние изменившейся аналитики счета на работу существующих механизмов типового решения. Любое изменение аналитики счета, задействованного в автоматических учетных схемах, может привести к существенной переделке этих схем, даже если не превышено максимальное количество субконто, установленное в свойствах плана счетов. А если к тому же требуется увеличить максимальное количество субконто, предусмотренное в плане счетов типового решения и как следствие прописанное во всех универсальных механизмах, переделка коснется и их.

Формальное ограничение со стороны платформы на максимальное количество субконто составляет 50 субконто на счете. Во многих случаях при проектировании плана счетов имеет смысл обдумать возможность вынесения отдельных разделов учета из бухгалтерии в оперативный учет, который может быть реализован с помощью регистров накопления.

При принятии решения о необходимости вынесения раздела в оперативный учет имеет смысл учитывать критерии, перечисленные в табл.7.4.

Таблица 7.4

Решение по выбору порядка следования видов субконто

Критерий	Субконто бухгалтерии	Измерения оперативного учета
Максимальное количество субконто, выбранное в плане счетов демонстрационной конфигурации	Не более двух	Нет ограничения, может быть создан регистр с нужным количеством измерений
Максимальное количество субконто, с которыми эффективно может работать платформа	* Не представляется возможным дать этот критерий в количественном выражении, не увидев и не проанализировав задачу, которую необходимо решить. Можно сказать лишь, что при решении одной и той же задачи автоматизации одного раздела учета наличия и движения средств регистр накопления будет эффективнее регистра бухгалтерии. Это обусловлено универсальностью регистра бухгалтерии, платой за которую является эффективность.	
Количество объектов учета	Десятки, сотни, тысячи **	Десятки и сотни тысяч **
Количество документов в день	Единицы, десятки **	Десятки, сотни **
Измерение примитивного типа	Настоятельно не рекомендуется	В принципе возможно
Влияние на бухгалтерский баланс	Необходимо	Не рекомендуется

** Количественные оценки очень приблизительны и неточны, и причина этого все та же: невозможно дать универсальные рекомендации. Правильное решение должно быть результатом оценки каждой конкретной задачи, и осуществляется эта оценка на этапе технического проектирования задачи после детального изучения объекта автоматизации.

Отдельно можно выделить последний критерий: влияние на бухгалтерский баланс. Разделяя учет на бухгалтерский и оперативный, в качестве критерия можно использовать необходимость включения информации с выбранной детализацией в сводные формы бухгалтерской отчетности (в первую очередь бухгалтерский баланс). Пример: в бухгалтерском балансе товары входят в раздел «Оборотные активы» и отображаются там свернуто (без деления по номенклатуре, складам, партиям). Вывод: можно целиком вынести этот аналитический разрез в оперативный учет [3].

Тема 8. Технология реализации и основные понятия сложных периодических расчетов в корпоративных информационных системах на платформе «1С:Предприятие»

Технология реализации и основные понятия сложных периодических расчетов в корпоративных информационных системах на платформе «1С:Педприятие»: вид расчета, период регистрации, период действия, вытесняющие расчеты и фактический период действия, зависимость по базовому периоду, ведущие расчеты и перерасчет, сторнирование.

Отличительной особенностью периодических расчетов является отсутствие однозначной привязки событий к точке на оси времени. Регистрируемые события в этом виде учета имеют отношение не к моменту времени, а к периоду в целом. При этом периодом расчета может быть день, месяц, квартал или год в зависимости от специфики решаемой задачи. Так, если для регистрации факта поступления товара важен точный момент его прихода, то при начислении премии важен период расчета (например, месяц), в котором она была начислена.

Еще одним отличием периодических расчетов является протяженность некоторых регистрируемых событий во времени. Например, при регистрации отпуска сотрудника указывается дата начала и дата окончания отпуска. В прочих видах учета регистрация подобных записей невозможна.

Таким образом, помимо задач расчета зарплаты, механизм периодических расчетов может быть использован и в других областях, предполагающих периодическую регистрацию протяженных во времени событий, например, в задаче расчета аренды за помещения выставочного центра.

Периодические расчеты – это вычисления, осуществляемые с определенной периодичностью, тесно связанные друг с другом по некоторым правилам и взаимно влияющие друг на друга в пределах некоторого периода.

Механизмы периодических расчетов позволяют настроить порядок и взаимосвязь вычислений и организовать учет их результатов.

Наиболее типичным примером использования периодических расчетов является расчет заработной платы, при котором производятся расчеты начислений и удержаний. Расчеты обычно выполняются с месячной периодичностью, а результаты расчетов одного вида могут зависеть от наличия и результатов расчетов другого вида.

Ведение периодических расчетов в системе «1С:Предприятие» обеспечивают объекты конфигурации Планы видов расчета и Регистры расчета. Так как эти объекты тесно связаны между собой, в данной главе дается общая характеристика возможностей, предоставляемых этими объектами.

Основные понятия

Вид расчета

Одним из основных понятий механизма периодических расчетов является вид расчета. Любой расчет, выполняемый в системе, регистрируется с обязательным указанием вида расчета, под которым может пониматься как способ расчета данной записи, так и дополнительные свойства,

характеризующие сущность именно этого расчета. Например, для целей расчета зарплаты различными видами расчета могут быть оклад, надбавка за вредность, оплата сверхурочных. В данном случае каждая из этих записей имеет свой алгоритм расчета, поэтому для каждого вида начисления сотрудникам вводится отдельный вид расчета. Тем не менее, оплата дежурства, например, может иметь тот же алгоритм расчета, что и оклад, но при этом это будет отдельный вид расчета, так как у него другой смысл [3].

В общем случае список видов расчета, используемых в системе, зависит от специфики решаемой задачи. Расчетные инструменты платформы «1С:Предприятие» устроены таким образом, что пользователь может самостоятельно создавать и настраивать виды расчета, которые ему необходимы для решения прикладной задачи, в режиме 1С:Предприятие. Для организации системы взаимосвязанных видов расчета в платформе предусмотрен объект конфигурации План видов расчета. План видов расчета определяет структуру хранения данных о видах расчета, используемых в прикладном решении для определенных целей. Разработчик может создать в конфигурации несколько планов видов расчета для различный целей, в каждый из которых пользователь в режиме 1С:Предприятие сможет внести неограниченное количество видов расчета, а также настроить их взаимосвязь.

Все записи о расчетах регистрируются в системе в привязке к периоду расчета и конкретному виду расчета. Для хранения записей о расчетах в системе предусмотрен объект метаданных Регистр расчета. Каждый регистр расчета имеет определенную периодичность (день, месяц, квартал или год). В зависимости от периодичности записи в этом регистре имеют соответствующий период регистрации.

Все записи регистра расчета, помимо периода регистрации, содержат информацию о виде расчета, примененном в данной записи. В одном регистре расчета могут содержаться виды расчетов только из одного плана видов расчетов. При этом регистров расчета, использующих один и тот же план видов расчета, может быть несколько.

Записи в регистрах расчета могут быть определенным образом взаимосвязаны между собой. Вид этой взаимосвязи определяется настройками используемых видов расчета. Взаимосвязь, установленная для видов расчета, действует во всех регистрах, где эти виды расчета используются [3].

Период. Для расчетов важным является понятие периода. Обычно период описывается датой начала и датой окончания. Если для расчета определена периодичность (см. описание ниже), то для описания периода (действия, регистрации) данного расчета достаточно указать любую дату. По этой дате вычисляется дата начала периода, и именно эта дата будет описывать период. Такой порядок определения периода позволяет оптимизировать выполнение запросов, в которых требуется выбрать записи, относящиеся к указанному периоду.

Периодичность расчетов. Определяет, с каким периодом будут (могут) выполняться расчеты, учитываемые данным регистром. Задается в свойстве Периодичность регистра расчетов. По значению этого свойства (если регистр

периодический) определяется период действия записи регистра расчета. Например, регистр имеет периодичность Месяц, тогда при формировании записи регистра в качестве периода действия выбирается дата документа (например, некий расчет за ноябрь 2021 г.), и по ней система определяет период действия на начало 01.11.2021.

Период регистрации – дата начала периода, указанного при регистрации расчета (вычисляется по дате документа-регистратора). Например, в июне 2021 г. производится начисление оклада за май 2021 г. (расчеты с периодичностью Месяц). Май 2021 г. - период действия (в базе данных записывается дата 01.05.2021), а июнь 2021 г. - период регистрации (в базе данных записывается дата 01.06.2021).

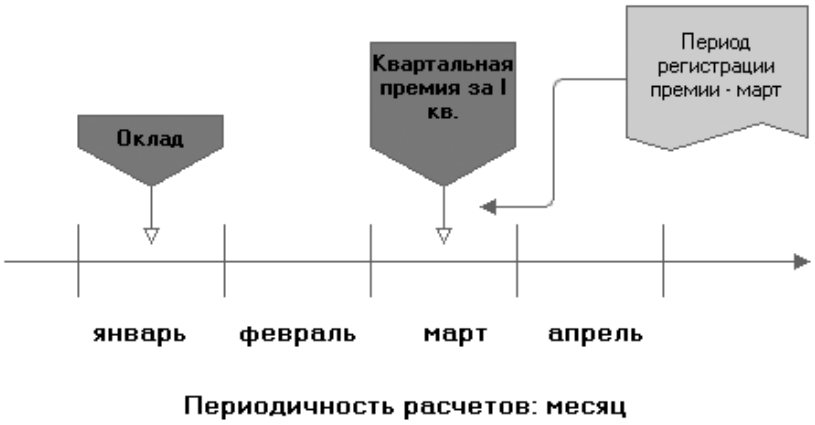


Рис. 8.1. Привязка записей расчета к периоду регистрации

Период действия - начальная дата периода, определяемая в соответствии со значением свойства Периодичность. Например, в документе указывается, что расчет производится за май 2021 г. Для значения свойства Периодичность регистра Месяц период действия определяется датой 01.05.2021; для значения Квартал - 01.04.2021.

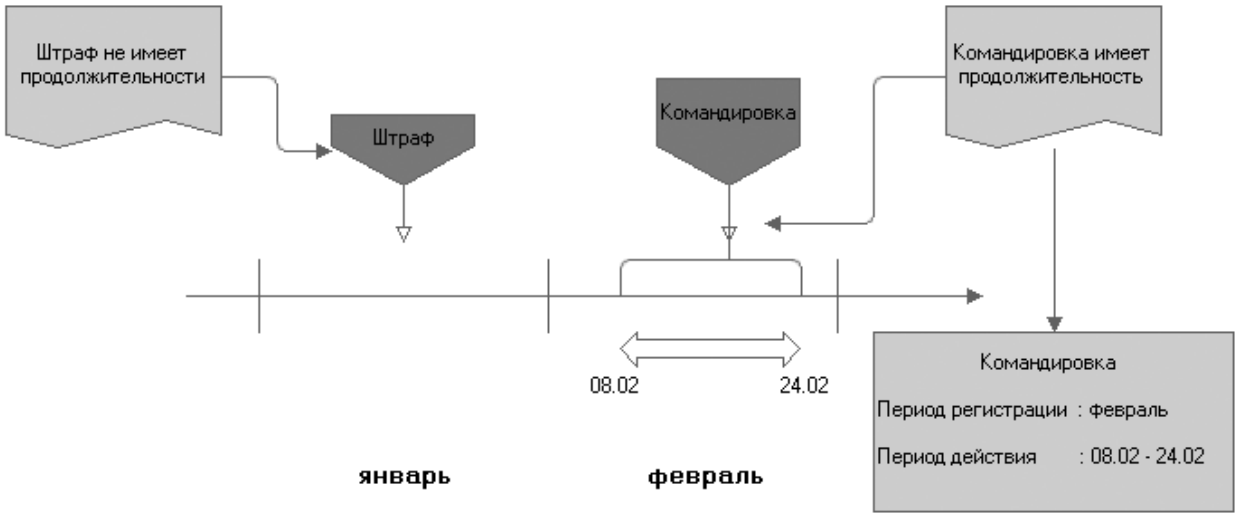


Рис. 8.2. Записи расчета, протяженные во времени

Период действия представляет собой непрерывный интервал времени, определяемый датой начала периода и датой его окончания. В приведенном примере периода действия записи о командировке – это интервал с 8 по 24 февраля. Период действия одной записи регистра расчета может лежать только

в рамках одного периода расчетов. Если необходимо ввести расчет, который длится в рамках нескольких расчетных периодов, его нужно разбить на несколько записей (рис. 8.3) [3].

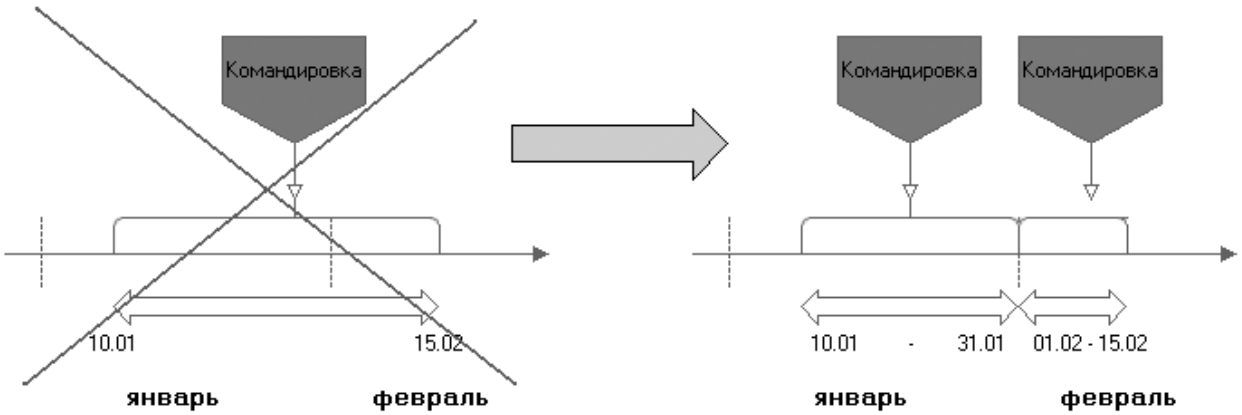


Рис. 8.3. Ввод расчета, который длится в нескольких периодах

У протяженных во времени записей период действия может не принадлежать периоду регистрации и лежать как в прошлом, так и в будущем относительно периода регистрации.

Период действия расчета - указывает период, за который производится расчет. Период определяется датой начала и датой окончания периода. Например, запись о больничном листе за май 2021 г. имеет период действия 01.05.2021, а период действия расчета определяется датой начала (например, 06.05.2021) и датой окончания (например, 15.05.2021).

Механизм вытеснения или конкуренция за период действия расчета - проявление связи видов расчетов по периоду действия расчетов. Конкуренция возникает вследствие невозможности выполнения нескольких видов расчетов одновременно. Необходимо выбрать тот расчет, который в данном периоде будет выполнен. Настройка механизма вытеснения задается в описании конкретного вида расчета. Такая настройка выполняется в разделе Вытесняющие (виды расчетов). Например, расчет Оплата по окладу не может применяться одновременно с расчетом Оплата по больничному листу. При этом говорят, что расчет больничного вытесняет расчет оклада, т. е. за период, в котором «действует» больничный, оклад не начисляется.

Вытесняющие расчеты и фактический период действия. Протяженные во времени записи регистров расчета могут конкурировать между собой за период действия. Это означает, что они не могут действовать одновременно. Такая конкуренция необходима, когда виды расчета являются по сути взаимоисключающими. Например, виды расчета Оклад и Командировка не могут действовать в один и тот же момент, так как сотрудник не может одновременно работать на основном месте и находиться в командировке (рис. 8.4). При вводе записи о командировке на определенный интервал времени система должна исключать действие оклада в этом интервале. Такие виды расчетов, которые исключают одновременное действие записей других видов расчетов, называются вытесняющими [3].

В данном случае у записей о командировке и окладе возникает конкуренция за период действия в интервале с 10.03 по 15.03 (т. е. в интервале действия командировки). Как уже было отмечено, эти записи не могут

Рис. 8.4. Вытеснение по периоду действия [3]

действовать одновременно, поэтому в указанном интервале происходит вытеснение оклада командировкой. Таким образом, вид расчета Командировка по сути является вытесняющим по отношению к виду расчета Оклад.

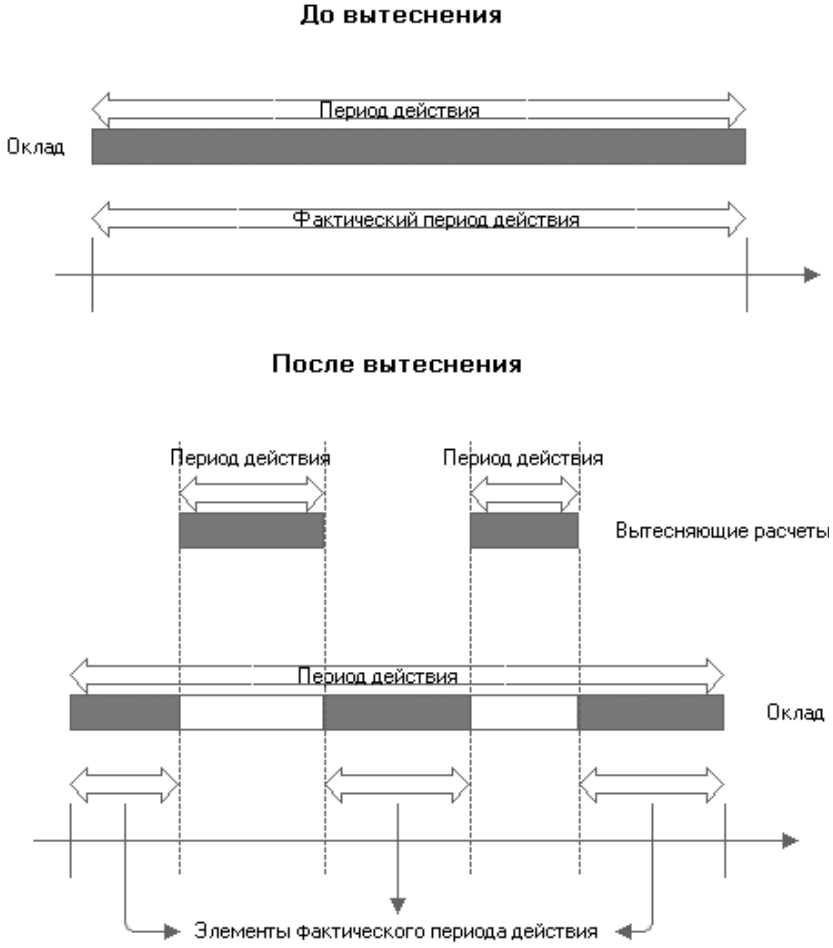


Рис. 8.5. Фактический период действия

Это означает, что записи с видом расчета Командировка будут вытеснять записи с видом расчета Оклад. При вытеснении одного расчета другим период действия вытесняемого расчета не изменяется. Результат вытеснения влияет на так называемый фактический период действия вытесняемого расчета. Под фактическим периодом действия записи понимается

совокупность интервалов времени в рамках периода действия, на протяжении которых расчет действует с учетом всех вытеснений. Если никаких вытеснений не происходило, фактический период действия совпадает с периодом действия (рис. 8.5).

Таким образом, до вытеснений фактический период действия записи об окладе содержал только один интервал, совпадающий с периодом действия. В результате вытеснения период действия оклада не изменился, а конкуренция выразилась в разбиении фактического периода действия оклада на три интервала. Важно учитывать, что вытеснение срабатывает только при вводе вытесняющих расчетов текущим или будущим периодом. При вводе расчетов за предыдущий период (задним числом) механизм вытеснения не действует. Это связано с тем, что записи, введенные в систему раньше (т. е. с меньшим периодом регистрации), имеют больший приоритет в конкуренции за период действия, чем более поздние записи (с большим периодом регистрации), причем это не зависит от настройки списка вытесняющих расчетов [3].

Фактический период действия – если расчет не вытесняется другими расчетами, то фактический период совпадает с периодом действия. Если есть вытесняющие виды расчетов, то фактический период определяется как совокупность непересекающихся периодов, в которых данный расчет не вытеснялся.

Зависимость по базовому периоду. В определенных случаях результат одного расчета может зависеть от результата других расчетов, введенных в систему. Например, квартальная премия может зависеть от суммы начисленного за квартал заработка по окладу. В этом случае говорят, что оклад входит в базу расчета премии, а вид расчета Оклад является базовым по отношению к виду расчета Квартальная премия. При этом один вид расчета может иметь несколько базовых видов расчета, то есть зависеть одновременно от нескольких других расчетов. При расчете результата расчета по базе анализируется сумма базовых видов расчета за определенный период, который называется базовым периодом. Базовый период – это произвольный непрерывный интервал дат, который может покрывать несколько расчетных периодов. Например, при расчете оплачиваемого отпуска, как правило, производится расчет среднего заработка за 3 месяца, предшествующих отпуску. Этот интервал и будет базовым периодом расчета отпуска.

Базовый период – определяет период, за который будут выбираться результаты расчетов, используемых (являющихся базовыми) для данного расчета (рис. 8.6). Например, при начислении премии за май 2021г. учитываются результаты начислений, выполненных определенными видами расчетов за некоторый период (это могут быть Оклад, Доплата, Отпуск). Этот период и будет являться базовым для расчета премии. Настройка связи видов расчетов по базовому периоду производится в описании конкретного вида расчета в разделе Базовые (виды расчетов).

В приведенном на рис. 8.6 примере виды расчетов Оклад и Надбавка являются базовыми по отношению к отпуску. Если начисляется оплата отпуска за апрель, то базовым периодом этой записи будет интервал с 1 января по 31

марта (3 целых месяца, предшествующие отпуску). В этом случае в базу расчета отпуска войдут все базовые начисления, входящие в базовый период записи об отпуске. В данном случае база расчета отпуска составляет 33 000 рублей и состоит из суммы начислений по окладу за 3 месяца и начисленной в феврале надбавки. Исходя из этой базы, будет производиться расчет начисления отпускных в соответствии с алгоритмом расчета.

То есть расчетная база – это еще не результат расчета, а лишь исходные данные, которые участвуют в алгоритме расчета. В данном случае предположим, что отпускные рассчитываются как среднее арифметическое заработков за 3 предыдущих месяца. В этом случае расчет отпуска производится следующим образом:

Результат расчета = База расчета / 3 = 33 000 / 3 = 11 000.

Расчет по базе может применяться ко всем видам расчетов, независимо от того, являются они протяженными во времени или нет. Например, квартальная премия, которая не имеет периода действия, может иметь базовый период (например, I квартал) [3].

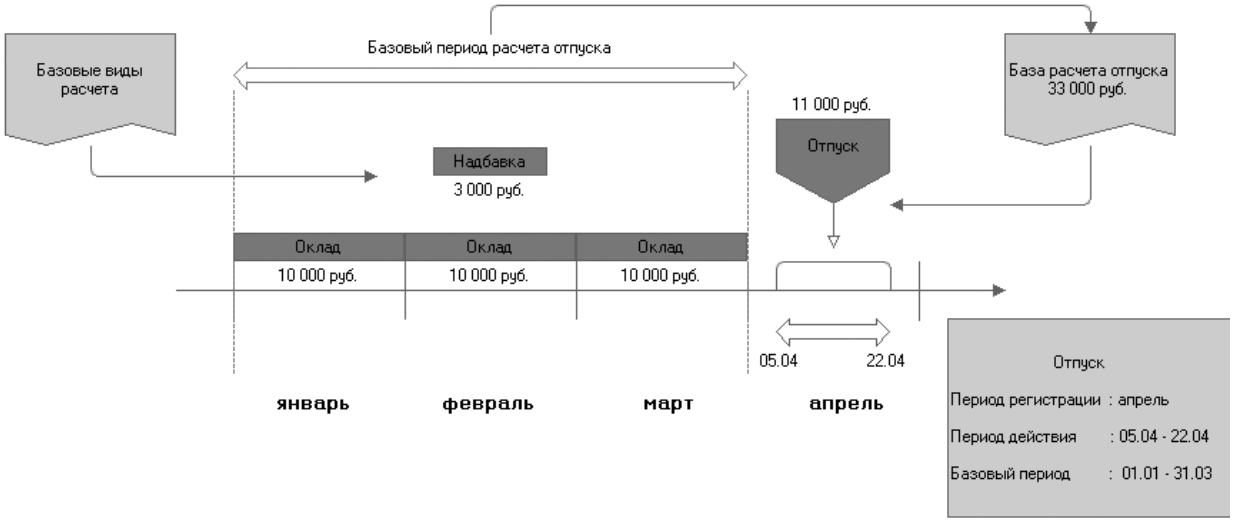


Рис. 8.6. Базовый период

Ведущие расчеты и перерасчет. Ведущими называют виды расчетов, при вводе или изменении которых необходимо перерассчитать результат уже существующих расчетов. Например, если работнику начислена премия за март в размере 10 % от заработка, то при вводе нового начисления в марте размер этой премии теряет актуальность. Премию необходимо пересчитать, чтобы учесть новое начисление [3].

Под понятием перерасчет понимается повторный расчет результата записи с учетом произошедших изменений. Система позволяет автоматически отслеживать изменения расчетов, которые могут повлечь перерасчет других записей, и формировать список расчетов, которые необходимо перерассчитать. Для этого система при вводе расчета анализирует, по отношению к каким другим видам расчета данный вид расчета является ведущим.

По сути, все виды расчета, являющиеся базовыми по отношению к данному виду расчета, должны быть одновременно и ведущими по отношению к

нему. Это связано с тем, что если данный вид расчета использует результаты базовых расчетов, то при изменении этой базы запись необходимо перерассчитать. В указанном случае оклад и надбавка составляют базу расчета премии, поэтому логично, что при их изменении нужно заново рассчитать премию.

В то же время ведущими расчетами могут быть не только базовые. Ведущими могут быть также расчеты, влияющие на результат данного вида расчета косвенно. Например, вид расчета Невыход, который вытесняет оклад, косвенно влияет на расчет премии, так как при вводе невыхода может измениться сумма оклада (рис. 8.7). В этом случае Невыход является ведущим по отношению к премии. Такая же ситуация возникает при косвенной зависимости по базе [3].

В данном случае командировка напрямую не входит в базу расчета премии, но влияет на доплату за разъезды, которая, в свою очередь, входит в базу расчета

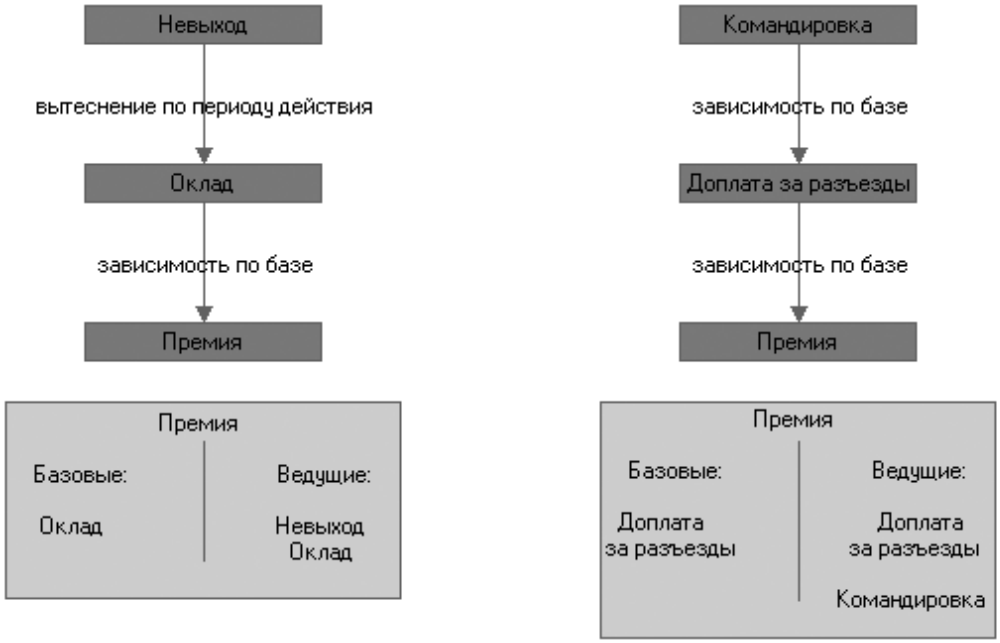


Рис. 8.7. Ведущие виды расчета

премии. Таким образом, изменение суммы за командировку косвенным образом повлияет на премию. И хотя сама премия не рассчитывается на базе командировки, Командировка будет ведущим видом расчета по отношению к премии.

Сторнирование.

Возможность указать для записи период действия, отличный от периода регистрации, существует только у регистров, поддерживающих период действия. В этом случае, если у записи период действия раньше периода регистрации (то есть запись зарегистрирована «с опозданием»), эта запись может вступать в конкуренцию за период действия с записями более раннего периода регистрации. При этом механизм вытеснения действовать не будет, так как запись имеет более поздний период регистрации, но конкуренция видов расчета останется. В результате у записи с более поздним периодом регистрации может образоваться меньший фактический период действия в силу того, что она не может действовать одновременно с конкурирующими видами расчета [3].

Как уже отмечалось, записи с более поздним периодом регистрации ни при каких обстоятельствах не могут вытеснить по периоду действия записи с более ранним периодом регистрации. Больничный за февраль, введенный в марте, не изменит фактического периода действия февральского оклада. При этом указанные записи в силу настройки вытесняющих видов расчета не могут действовать одновременно, то есть их фактические периоды действия не должны пересекаться. В результате фактический период действия больничного будет пустым. Но эта ситуация должна быть как-то учтена, т.к. работнику начислен завышенный оклад за февраль. Для решения этой задачи необходимо [3]:

– в текущем периоде регистрации отменить неправильно начисленную часть оклада;

– позволить больничному иметь непустой фактический период действия, чтобы правильное начисление могло вступить в силу.

Эта задача решается вводом в систему корректирующих записей, которые называются сторно-записями. Процесс такой корректировки называется сторнированием. Как было отмечено выше, при формировании фактического периода действия записи учитывается наличие сторно-записей по конкурирующим видам расчета. Ввод в систему таких записей позволит больничному иметь непустой фактический период действия на интервале действия сторно-записи.

В указанном примере изначально был начислен оклад в размере 10 000 рублей за февраль. В марте становится очевидным, что совершена ошибка, так как в середине месяца сотрудник болел. Если просто ввести больничный за февраль с периодом регистрации март, он будет иметь пустой фактический период действия. Чтобы этого избежать, вводится сторно-запись по окладу, которая решает две задачи:

– отменяет начисление оклада за период болезни (за этот период сотруднику было ошибочно начислено 4 000 рублей по окладу);

– позволяет больничному иметь непустой фактический период действия, в результате чего сотруднику начисляется 3 000 рублей по больничному листу.

В результате этих действий общий результат записей становится верным. Важно понимать, что периодом регистрации сторно-записи будет тот же месяц, которым введен больничный (в данном случае март). Иными словами, если в текущем периоде обнаружена ошибка и нужно внести изменения в предыдущий период, то соответствующая корректирующая запись будет зарегистрирована в текущем периоде. При этом вид расчета сторно-записи такой же, как и в ошибочно введенной записи прошлого периода (в данном случае Оклад).

В этой связи стоит еще раз остановиться на принципах формирования фактического периода действия записей. В данном случае при определении фактического периода действия больничного учитывался не только период действия оклада, но и период действия сторно-записи. Таким образом, при формировании фактического периода действия учитывается наличие сторно-записей конкурирующих видов расчета (рис. 8.8).

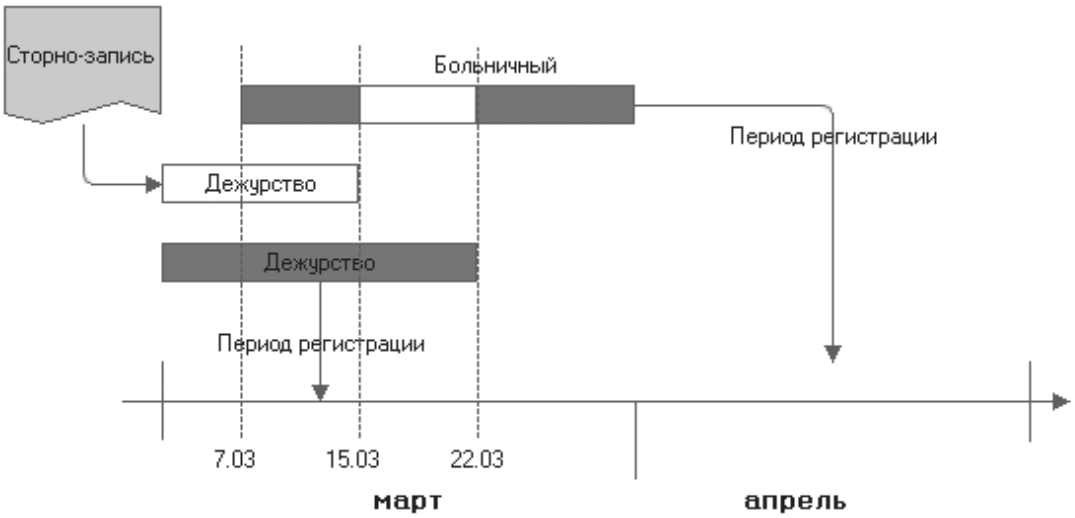


Рис. 8.8. Учет сторно-записей при формировании фактического периода действия

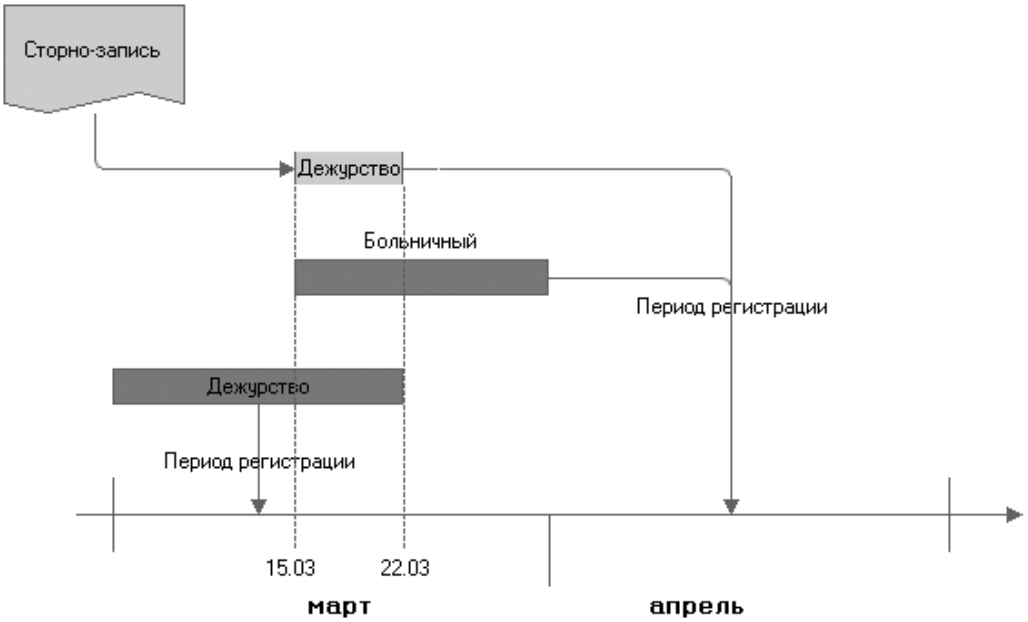


Рис. 8.9. Использование сторно-записей

В данном случае больничный, имеющий более поздний период регистрации, имел бы пустой фактический период действия на интервале пересечения с периодом действия дежурства (с 07.03 по 22.03). Однако наличие сторно-записи позволяет больничному действовать параллельно с дежурством на интервале действия сторно-записи (с 07.03 по 15.03). При этом на интервале с 16.03 по 22.03 фактический период действия больничного будет по-прежнему прерываться, так как на этом интервале не действует сторно-запись по дежурству. Поэтому для того, чтобы больничный мог занять весь свой период действия, сторно-запись должна быть введена с периодом действия 07.03–22.03, то есть на всем интервале пересечения периодов действия дежурства и больничного. Наличие сторно-записи по виду расчета Дежурство позволит записи о больничном занять весь свой период действия (рис. 8.9). При формировании фактического периода действия больничного будет учтен период действия сторно-записи по дежурству [3].

Тема 9. Планы видов расчета и регистры расчета: предназначение, структура и основные свойства. Технология формирования и расчета записей регистров расчета

Планы видов расчета и регистры расчета. Назначение, структура и свойства планов видов расчета. Назначение, структура и свойства регистра расчета. Настройка протяженных по времени расчетов: использование механизма вытеснения, использование графиков. Настройка зависимости по базовому периоду. Технология формирования и расчета записей регистров расчета.

Планы видов расчета представляют собой прикладные объекты конфигурации, предназначенные для описания структур данных, в которых хранятся однотипные виды расчета.

Каждый план видов расчета определяет отдельную структуру данных, где пользователь в режиме «1С:Предприятие» может создавать неограниченное число элементов (видов расчета). Созданные виды расчета пользователь впоследствии может изменять и удалять из базы данных. Кроме этого, в режиме Конфигуратор в плане видов расчета можно создать неограниченное количество предопределенных видов расчета.

Эти виды расчета также будут доступны для использования в режиме «1С:Предприятие», но со следующими ограничениями:

– предопределенный вид расчета не может быть удален в режиме «1С:Предприятие»;

– в режиме «1С:Предприятие» не могут быть удалены и изменены некоторые свойства предопределенного вида расчета, заданные в режиме Конфигуратор.

В конфигурации может быть создано неограниченное количество планов видов расчета. Планы видов расчета могут различаться между собой по свойствам или по назначению использования.

При создании и настройке плана видов расчета разработчик может влиять на его свойства. Для каждого плана видов расчета задается его Имя, по которому можно обращаться к этому объекту конфигурации, и Синоним, а также Представление объекта, Представление списка и т. п. для представления плана видов расчета в интерфейсе «1С:Предприятия».

При настройке необходимо указать длину кода и наименования видов расчета, содержащихся в данном плане видов расчета. Кроме этого, указывается тип кода (строка или число) и основное представление видов расчета данного плана видов расчета для пользователя: в виде кода или в виде наименования. В зависимости от этой настройки в режиме «1С:Предприятие» будет отображаться код или наименование вида расчета в полях, хранящих ссылку на этот вид расчета.

Использование указанных выше свойств аналогично их роли при настройке справочников. Кроме этого, у плана видов расчета существуют специфические свойства, определяющие расчетные свойства соответствующих видов расчета. К таким свойствам относятся [3]:

– *Использует период действия*. Если в плане видов расчета установлено свойство «Использует период действия», то все виды расчета, хранящиеся в данном плане, будут рассматриваться как протяженные во времени. Для таких видов расчета применима настройка вытеснения по периоду действия. Планы видов расчета, использующие период действия, можно использовать в регистрах расчета с периодом действия. Если свойство не установлено, все виды расчета данного плана видов расчета будут рассматриваться как непротяженные во времени, для них нет смысла настраивать вытеснение. Такие планы видов расчета не могут быть использованы в регистрах расчета с периодом действия.

– *Зависимость от базы*. Эта настройка определяет, будет ли в видах расчета данного плана видов расчета использоваться зависимость по базовому периоду. Если переключатель установлен в положение Не зависит, то виды расчета данного плана видов расчета не смогут зависеть по базовому периоду от других видов расчета. Установка переключателя в положение Зависит по периоду действия или Зависит по периоду регистрации позволит устанавливать в видах расчета зависимость по базовому периоду. При этом любой вид расчета данного плана видов расчета теоретически может зависеть по базовому периоду от любых других видов расчета, в том числе из других планов видов расчета. Поэтому при настройке зависимости от базы необходимо указать, какие виды расчета могут выступать базовыми для видов расчета данного плана. При этом базовые виды расчета могут принадлежать данному плану счетов или другому плану счетов [3].

Структура плана видов расчета может существенно меняться в зависимости от его свойств. Общая структура плана видов расчета представлена рис. 9.1.

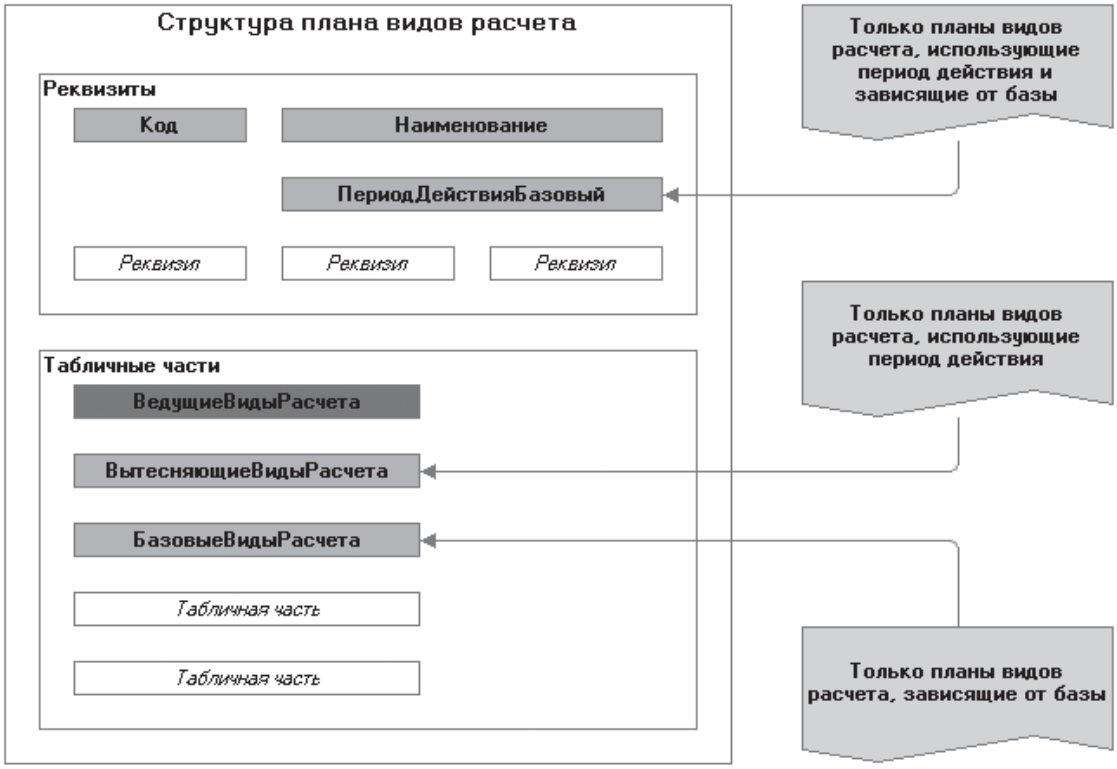


Рис. 9.1. Структура плана видов расчета

Регистры расчета. Это прикладные объекты конфигурации, которые предназначены для учета результатов вычислений, осуществляемых с некоторой периодичностью, тесно связанных друг с другом по некоторым правилам и взаимно влияющих друг на друга в пределах определенного периода (например, расчет заработной платы, расчет квартплаты, расчет арендной платы). В конфигурации их может быть неограниченное количество. Они так же как и другие регистры имеют измерения, ресурсы, реквизиты, однако, принцип действия регистров расчета абсолютно другой. Следует отметить, что взаимозависимость видов расчета настраивается в планах видов расчета, однако, расчетные механизмы платформы заложены именно в регистры расчета. Регистры расчета обеспечивают и регистрацию записей, и правильное отображение взаимосвязей расчетов (расчет записи по базе, вытеснение записей, формирование сторно-записи, учет протяженных во времени расчетов, хранение периода действия и фактического периода действия записей). Для расчетов, зависящих от базового периода, регистр расчета также хранит данные о базовом периоде.

Способность использовать описанные выше расчетные механизмы платформы напрямую зависит от настройки свойств регистра расчета: периодичность и виды расчетов, которые в нем будут регистрироваться.

Свойство Периодичность регистра расчетов определяет размерность периодов, для которых будет вестись учет в этом регистре. При настройке регистра доступны следующие виды периодичности: Год, Квартал, Месяц и День. Соответственно, записи в таком регистре будут фиксироваться в привязке к конкретному году, кварталу, месяцу или дню. Периодичность регистра определяет вид периода регистрации записей и зависит от задачи, которую он решает. Для целей расчета зарплаты, как правило, применяют регистры с периодичностью Месяц.

В одном регистре расчета могут регистрироваться записи только с видами расчета из одного плана видов расчета. Используемый в данном регистре план видов расчета указывается в свойстве регистра План видов расчета. При этом регистров, использующих один план видов расчета, может быть несколько.

Свойство *Период действия* определяет, можно ли будет в данном регистре учитывать записи, протяженные во времени. Установленное свойство Период действия позволяет для каждой записи хранить не только период регистрации, но и период действия, отражающий протяженность этой записи. Кроме этого, установка этого свойства означает, что в данном регистре будет задействован механизм вытеснения и будет формироваться фактический период действия записей. Для того чтобы учитывать протяженные во времени записи, в плане видов расчета, используемом в данном регистре, должно быть установлено свойство *Использует период действия* [3].

Если свойство регистра расчета Период действия не установлено, то регистрируемые записи не будут иметь периода действия и, соответственно, протяженности во времени. В таком регистре не будет использоваться механизм вытеснения. Тем не менее, в регистре без периода действия может быть назначен план видов расчета, использующий период действия.

Например, план видов расчета ОсновныеНачисления, использующий период действия, может регистрироваться как в регистре, где будет происходить расчет зарплаты (в этом регистре период действия необходим для расчета), так и в регистре, где начисления учитываются для целей налогообложения (в этом регистре период действия не нужен).

В данном случае один и тот же вид расчета из плана видов расчета, использующего период действия, зарегистрирован в двух разных регистрах, в одном из которых период действия учитывается, в другом – нет, так как в нем нет необходимости.

Если в регистре будут учитываться записи, протяженные во времени, то необходимо настроить для данного регистра свойство График. При расчете записей, являющихся протяженными во времени, часто значимость периода действия этих записей не является равномерной, то есть отдельные его части имеют различную значимость.

Например, если человек дежурил с 1 по 15 сентября, то этот интервал будет периодом действия этой записи. Однако в рамках этого периода не все дни могут являться одинаковыми. Например, человек работает различное количество часов в разные дни, а в какие-то дни у него выходные. Неравномерность значимости периода описывается при помощи графика.

График хранит определенное значение для каждой даты года. Соответственно, используя график, можно оценить значимость любого периода (рис. 9.2).

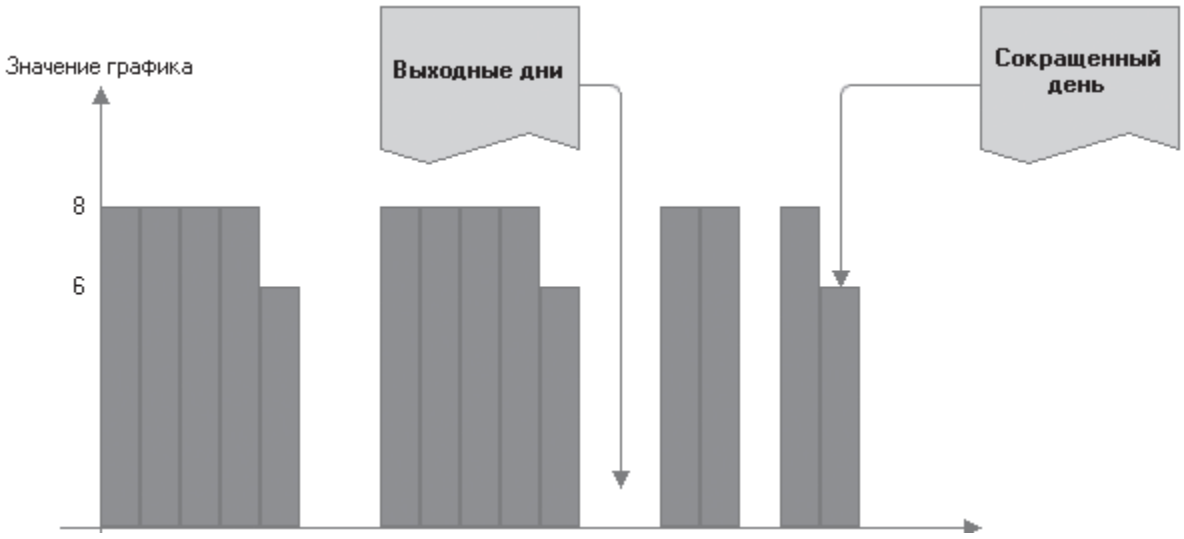


Рис. 9.2. Неоднородность периода

В приведенном примере график хранит количество рабочих часов в рамках каждого календарного дня. В данном случае количество часов и есть число, отражающее значимость каждого дня в периоде. Полные рабочие дни имеют значимость 8 (рабочих часов), сокращенные дни – 6, а выходные и праздничные дни имеют нулевую значимость.

График в системе должен быть создан разработчиком самостоятельно как непериодический регистр сведений, у которого как минимум одно измерение типа Дата и как минимум один ресурс типа Число. Разработчик может создавать

и более сложную структуру регистра, но указанные измерение и ресурс там должны присутствовать обязательно [3].

Для того чтобы привязать график к регистру расчета, необходимо в поле График свойств регистра указать нужный регистр сведений. После этого в поле Значение графика нужно указать ресурс этого регистра сведений, который является значением графика. В поле Дата графика указывается измерение регистра, которое является датой графика. Необходимость указания этих полей связана с тем, что в регистре может быть более одного измерения и более одного ресурса, поэтому поля даты и значения графика должны быть указаны явно.

Свойство регистра расчета ***Базовый период*** позволяет хранить в регистре базовый период записей и, соответственно, использовать механизм зависимости по базовому периоду. Только для записей регистров расчета с установленным свойством Базовый период возможно получение значения базы. При этом свойство Базовый период устанавливается независимо от свойства Зависимость от базы выбранного плана видов расчета. Даже если в плане видов расчета установлена зависимость от базы, в регистре свойство Базовый период может быть не установлено. В этом случае в данном регистре использование зависимости по базовому периоду будет недоступно.

В зависимости от настроек регистра расчетов его структура будет различной. Общая схема структуры регистра расчета показана на рис.9.3.

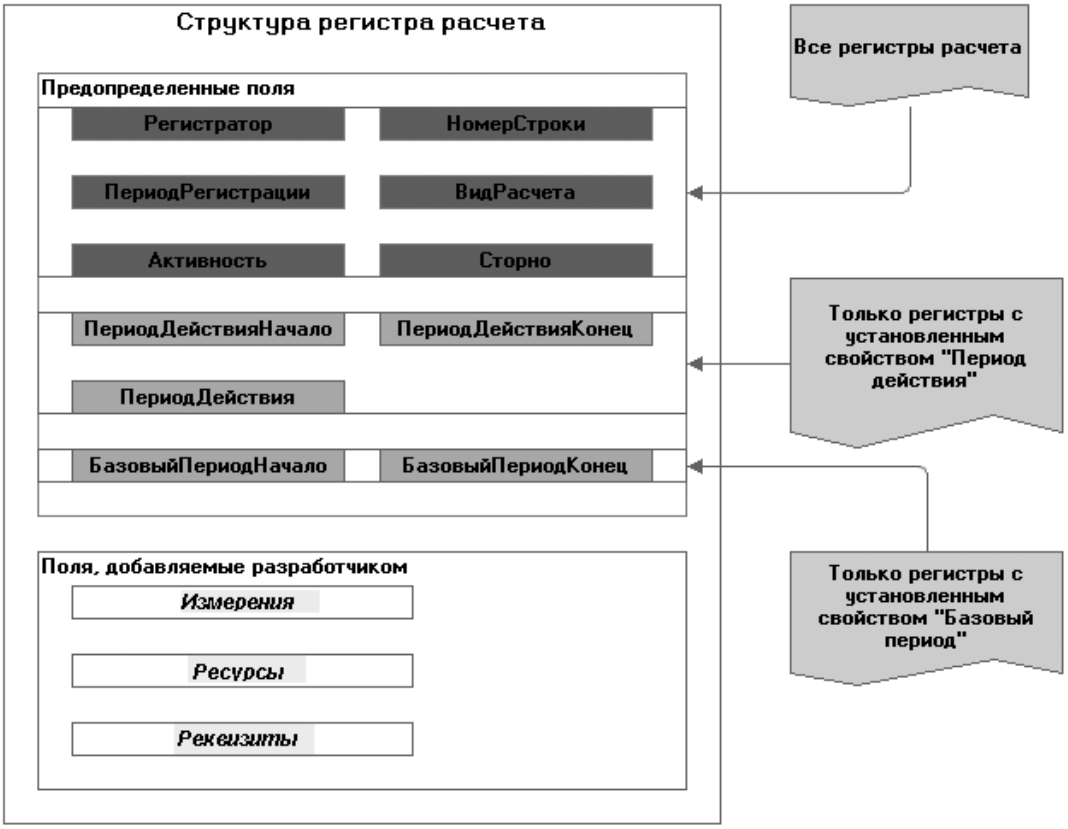


Рис. 9.3. Структура регистра расчета

В простейшем регистре расчета без периода действия и базового периода всегда будут присутствовать следующие поля [3]:

– *Регистратор* – ссылка на документ, который ввел запись в регистр.

– *Период регистрации* – период регистра расчета, к которому относится данная запись. Размер периода определяется периодичностью регистра расчета.

– *Вид расчета* – ссылка на вид расчета, используемый в данной записи.

– *Номер строки* – порядковый номер записи в рамках данного регистратора. В рамках одного документа-регистратора номера строк уникальны. Таким образом, совокупность значений Регистратор и НомерСтроки позволяет идентифицировать конкретную запись регистра расчета.

– *Активность* – признак активности данной записи. Запись влияет на учет только в том случае, если активность установлена в значение Истина. Данное поле используется в документах прямой записи в регистр. В таких документах не хранятся данные, необходимые для формирования движений по регистру. Ввод записей производится напрямую в регистр, через вывод на форму документа таблицы с привязкой к набору записей этого регистра.

Типичным примером использования этого принципа является документ Операция в решениях по бухгалтерскому учету. При пометке на удаление такого документа нельзя удалять записи из регистра, так как их нельзя будет восстановить. Вместо этого у записей регистра расчета, введенных этим документом, свойство Активность устанавливается в значение Ложь. При этом записи не удаляются из базы данных, но перестают влиять на учет. При необходимости снова задействовать эти записи достаточно установить свойство Активность в значение Истина. Именно это происходит при снятии пометки на удаление с такого документа.

– *Сторно* – это признак типа Булево, который обозначает, является ли данная запись сторно-записью. Сторно-записи записываются в регистр со значением Истина в этом поле.

Если у регистра установлено свойство Период действия, то в структуре появляются следующие предопределенные поля:

– *ПериодДействияНачало* – дата, обозначающая начало интервала периода действия записи.

– *ПериодДействияКонец* – дата, обозначающая конец интервала периода действия записи.

– *ПериодДействия* – дата, отражающая период регистра расчета, в котором действовала запись. Эта дата всегда имеет значение начала первого дня соответствующего периода (по аналогии с полем ПериодРегистрации).

Таким образом, запись с одним и тем же интервалом периода действия при записи в регистры разной периодичности будет иметь различное значение поля ПериодДействия. Это будет начало месяца в случае помесячного расчета, начало квартала при поквартальном расчете и т. д.

У регистров, для которых установлено свойство Базовый период, в структуре будут присутствовать следующие поля:

– *БазовыйПериодНачало* – дата начала интервала базового периода;

– *БазовыйПериодКонец* – дата окончания интервала базового периода.

Поля, добавляемые при разработке. Для каждого регистра расчета необходимо продумать структуру измерений, ресурсов и реквизитов, исходя из особенностей учетной задачи. Регистр расчета может и не иметь ни измерений,

ни ресурсов, ни реквизитов. В этом случае в регистре можно вести периодический учет записей с различными видами расчета без привязки к конкретным объектам (например, сотрудникам) и без сохранения числовых результатов расчетов (например, суммы начисления). При этом в регистре можно использовать период действия, будут работать механизмы вытеснения записей и перерасчетов, а также механизм зависимости по базовому периоду. При этом механизм зависимости по базовому периоду не позволит получить расчетную базу. Возможности регистра расчета в такой ситуации сильно ограничены, поэтому для решения прикладных задач, как правило, необходимо задать измерения, ресурсы и реквизиты регистра.

Ввод измерений регистров расчета позволяет обозначить объект учета и расчета. При отсутствии измерений регистра расчетные механизмы действуют в рамках всех записей. Например, в этом случае любая запись о больничном приведет к вытеснению всех записей об окладе, для которых произойдет пересечение периодов действия. В результате добавления в регистр измерения ФизЛицо работа регистра изменится следующим образом: при вводе больничного по сотруднику Иванов будут вытеснены только записи об окладе по сотруднику Иванов за соответствующий период действия.

Измерения регистров расчета могут иметь произвольный тип. К формированию структуры измерений нужно подходить аккуратно, так как они имеют специфику, отличную, например, от регистров накопления. Измерения регистров расчета не связаны с получением аналитики в определенных разрезах.

Состав ресурсов регистра расчета влияет на возможность получения расчетной базы по данному регистру для записей данного регистра или других регистров. Например, если в регистре основных начислений есть ресурс Результат, хранящий рассчитанную сумму начисления, то для записей этого или другого регистра можно получить расчетную базу по этому полю, то есть сумму начислений всех записей по базовым видам расчета за базовый период. Если числовой показатель необходимо хранить в регистре, но в расчете по базе он не участвует, то такой показатель должен быть введен в структуру регистра как реквизит. Например, поле Размер начисления, отражающее тариф, по которому был рассчитан оклад или размер процента для начисления премии, не должно вводиться в регистр как ресурс, так как получать базу по этому полю бессмысленно. Ресурсы регистров расчета всегда имеют тип Число.

Реквизиты регистра расчета – это произвольные дополнительные поля, которые необходимо хранить в регистре. Эти данные могут использоваться как при расчете записи, так и для формирования отчетов по регистру расчета. Например, реквизит РазмерНачисления участвует в формуле расчета записей, а реквизит СтатьяЗатрат может использоваться как группировка или отбор в отчете по начислениям. Добавление в регистр нового реквизита не оказывает принципиального влияния на учет и работу данного регистра (за исключением случаев, когда реквизит связан с графиком.

Настройка протяженных по времени расчетов: использование механизма вытеснения, использование графиков.

Для организации учета протяженных во времени расчетов у плана видов расчета должно быть установлено свойство *Использует период действия*. Это позволит настраивать в видах расчета взаимное вытеснение путем заполнения предопределенной табличной части *ВытесняющиеВидыРасчета*.

В используемом регистре расчета должно быть установлено свойство *ПериодДействия*, а также привязан график. При такой настройке все записи регистра помимо периода регистрации будут хранить данные о периоде действия, то есть о протяженности во времени.

При формировании записи в регистре расчета, поддерживающем период действия, происходит анализ конкуренции за период действия вводимого вида расчета с существующими в регистре записями. Результатом такой конкуренции будет изменение фактического периода действия вводимой или существующих записей.

Понятие фактического периода действия применимо только к конкретной записи регистра расчета. При этом данные о фактическом периоде действия не хранятся в основной таблице регистра расчета, но могут быть получены системными средствами.

При формировании фактического периода действия происходит анализ конкуренции записей только в рамках данного регистра расчета. Иными словами, запись регистра расчета может вытеснять только записи этого же регистра.

При анализе конкуренции видов расчета на пересечении периодов действия учитывается период регистрации записей. Если запись с вытесняющим видом расчета имеет более поздний период регистрации, чем запись с вытесняемым видом расчета, механизм вытеснения работать не будет.

При этом конкурирующие записи все равно не могут иметь пересекающийся период действия, поэтому фактический период действия записи с более поздним периодом регистрации будет пустым на интервале пересечения. При формировании фактического периода действия учитывается наличие в анализируемом периоде сторно-записей по конкурирующим видам расчета. Если на интервале пересечения конкурирующих видов расчета присутствуют сторно-записи, то фактический период действия записи с более поздним периодом регистрации будет непустым на интервале периода действия сторно-записи конкурирующего вида расчета.

Для получения фактического периода действия при помощи запроса предусмотрена виртуальная таблица фактического периода действия.

Структура полей этой таблицы полностью идентична структуре основной таблицы регистра расчета. Единственное ее отличие от основной таблицы состоит в том, что поля ПериодДействияНачало и ПериодДействияКонец обозначают не собственно период действия, а интервал фактического периода действия. Если фактический период действия записи задается несколькими интервалами, то в виртуальной таблице такая запись будет представлена несколькими строками.

Протяженные во времени расчеты тесно связаны с данными графика, используемого в регистре расчета. Как уже отмечалось, данные графика

содержат определенные значения для каждого календарного дня периода. Данные хранятся в системе в виде регистра сведений, созданного при разработке конфигурации. При этом дату графика платформа получает из измерения, заданного в свойствах регистра расчета в поле Дата графика, а значение графика – из ресурса, указанного в поле Значение графика.

Настройка зависимости по базовому периоду.

Для возможности ввода и расчета записей, зависимых по базовому периоду от других записей, необходимо настроить свойства соответствующих планов видов расчета и регистров расчета.

В плане видов расчета, виды расчета которого будут зависеть по базовому периоду от других видов расчета, необходимо настроить зависимость от базы. Для этого свойство *Зависимость от базы* плана видов расчета должно быть установлено в одно из значений *Зависит по периоду действия* или *Зависит по периоду регистрации.*

Зависимость по периоду действия предполагает анализ пересечения фактического периода действия записей базовых видов расчета с базовым периодом текущей записи. При такой зависимости возможно частичное попадание какой-либо записи в базу расчета текущей записи. Если у записи по базовому виду расчета нет периода действия, то будет проанализировано попадание периода регистрации этой записи в базовый период текущей записи.

Зависимость по периоду действия может быть установлена независимо от того, использует ли данный план видов расчета период действия или нет. В данном случае важен период действия не рассчитываемой записи, а базовых записей, на основании которых она рассчитывается. Например, премия не использует период действия, при этом она зависит по периоду действия от других записей, например, начисленных окладов.

В случае зависимости по периоду регистрации у базовой записи берется ее период, отсчитываемый от значения поля ПериодРегистрации. Например, при периодичности базового регистра Месяц будет анализироваться пересечение периода от даты ПериодРегистрации плюс месяц записей базовых видов расчета с базовым периодом текущей записи. Период действия базовых записей в данном случае значения не имеет.

Такой формат зависимости часто используется при расчете удержаний, так как удержания, как правило, рассчитываются с начисленных за период сумм, независимо от того, когда они действовали. В таких случаях важен период регистрации базовых записей, а не период действия.

Например, при расчете удержания по исполнительному листу за март будет учтена оплата больничного, начисленная в марте, даже если работник реально болел в феврале.

В данном случае в базу расчета удержания попадет оклад за март, а также больничный за февраль, зарегистрированный в марте. При этом мартовская надбавка, зарегистрированная в апреле, в базу расчета не попадет.

При настройке расчетов, зависимых от базы по периоду регистрации, необходимо помнить, что период регистрации – это всегда дата начала периода, поэтому базовый период должен заполняться с учетом этого.

После настройки зависимости от базы необходимо указать, от каких видов расчета могут зависеть по базовому периоду виды расчета текущего плана видов расчета. Для этого нужно указать список базовых планов видов расчета в свойствах текущего плана видов расчета. В результате этой настройки в любом виде расчета текущего плана видов расчета в табличной части БазовыеВидыРасчета можно будет выбрать любой вид расчета отмеченных планов видов расчета. В данном случае расчет дополнительных начислений может базироваться на данных основных и дополнительных начислений. При такой настройке для вида расчета Премия за 3 месяца в списке базовых можно будет указать оклад из основных начислений и премию за месяц из дополнительных. Указанная настройка плана видов расчета будет действовать во всех регистрах расчета, где участвует этот план видов расчета. В данном случае рассчитывать премии от оклада можно будет как в регламентированном, так и в управленческом учете. Более того, можно будет рассчитать, например, управленческую премию по окладам, начисленным в регламентированном учете.

Технология формирования и расчета записей регистров расчета.

Регистры расчета не поддерживают независимого формирования записей без использования документа-регистратора. Поэтому ввод записей в регистр расчета должен быть организован с применением документов. В документе должны быть указаны все данные, необходимые для заполнения всех полей записи регистра расчета, кроме ресурсов. Такой документ заполняется пользователем в режиме «1С:Предприятие» и формирует записи регистра расчета при проведении [3].

Рис. 9.4. Формирование записей регистра расчета

Расчет записей регистра расчета может производиться в любой момент, определенный разработчиком. Например, расчет записей может производиться при проведении документа регистрации записей. В этом случае значения ресурсов будут рассчитываться в момент проведения, и пользователь не сможет вручную изменить результат расчетов. Также можно инициировать расчет записей из формы документа. Для этого можно создать на форме кнопку Рассчитать, при нажатии которой будет производиться расчет. В этом случае результат расчета можно поместить в табличную часть документа, для того чтобы пользователь мог скорректировать его вручную. При проведении такого

документа записи регистра расчета будут сформированы с учетом ручных корректировок (рис. 9.4).

Схема расчета модулей будет схода в обоих случаях (рис. 9.5) [3].

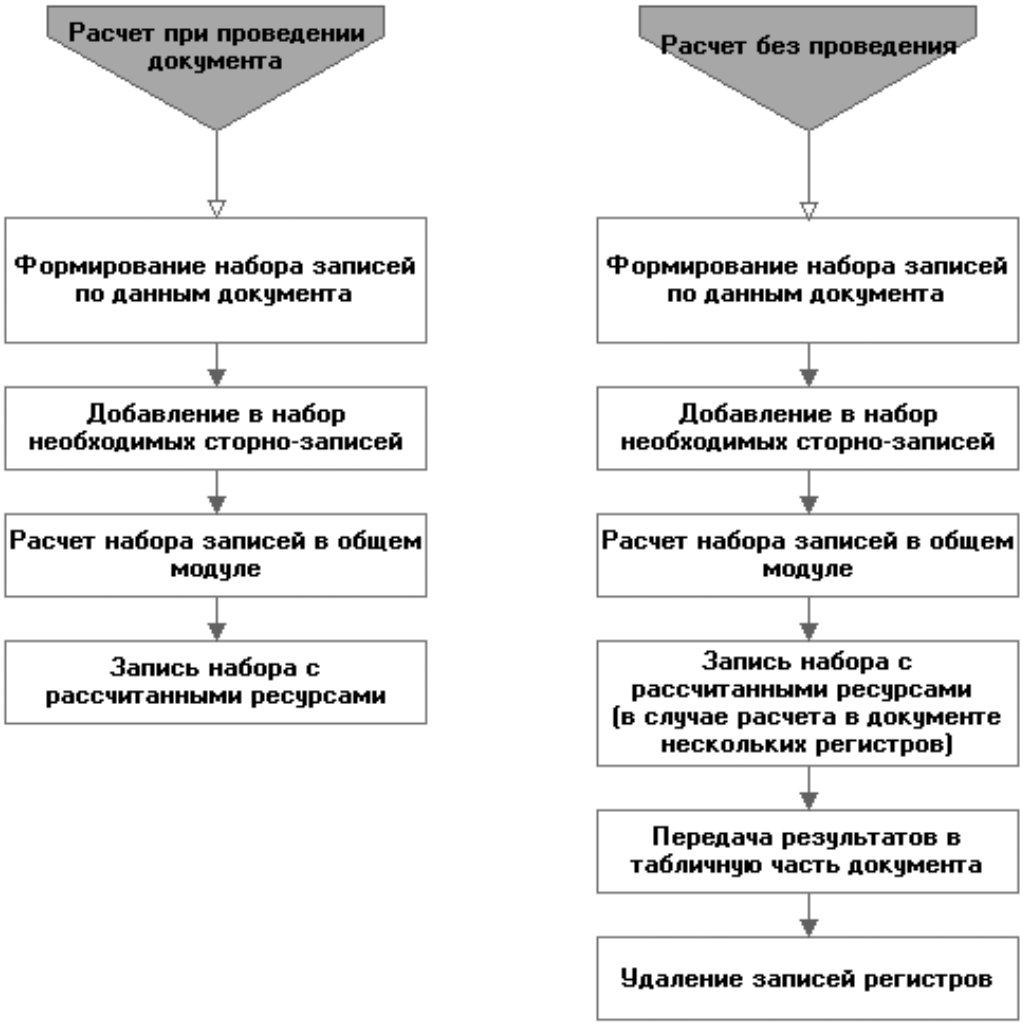


Рис. 9.5. Схема расчета

Таким образом, расчет записей без проведения документа отличается только наличием двух дополнительных этапов. Рассчитанные значения ресурсов необходимо поместить в табличную часть документа, в соответствующие поля, для того чтобы пользователь мог их изменить.

После этого необходимо удалить созданные в процессе расчета записи регистров расчета, так как пользователь в данном случае не просил проводить документ. Тем не менее, расчет не может быть выполнен без временной записи данных в регистр расчета, так как все расчетные механизмы платформы (вытеснение, зависимость по базовому периоду, расчет по графику, перерасчет) применимы только к записям регистров расчета.

Если в документе настроен расчет записей без проведения, то в момент проведения такого документа собственно расчета уже не происходит. В ресурсы регистра расчета попадают значения из табличной части документа, которые пользователь мог скорректировать после расчета [3].

Заключение

Система "1С:Предприятие" стала современным трендом цифровизации, а широкое внедрение решений "1С" – примером успешной конкурентной борьбы отечественных высокотехнологичных разработок с продукцией ведущих международных корпораций.

Более чем на 5 000 000 рабочих мест в организациях различного размера и форм собственности в России и других странах ежедневно применяются отечественные IT-решения, построенные на инновационной технологической платформе "1С:Предприятие 8". Крупнейшими пользователями корпоративных информационных систем на платформе «1С:Предприятие» являются: Почта России (более 47 000 автоматизированных рабочих мест), "Трансмашхолдинг" (более 20 000 АРМ), Правительство Москвы (единая облачная система, более 18 000 пользователей), "Башкирэнерго" (более 8 600 АРМ), КамАЗ (более 7 800 АРМ) и др.

Построение корпоративных информационно-аналитических систем на платформе «1С:Предприятие» соответствует политике импортозамещения в сфере IT-технологий. Широкая распространенность решений на этой платформе, открытость кода типовых решений и наличие большого количества специалистов фирм–франчайзи способствует эффективному внедрению и сопровождению решений на предприятиях различных отраслей, различных форм собственности и масштабов деятельности.

Материал лекционных занятий предусматривает формирование развёрнутого представления об архитектуре корпоративных информационных систем, построенных на платформе «1С:Предприятие», подходу к описанию бизнес-приложения на основе модели метаданных, вариантах работы ИС на платформе «1С:Предприятие», видах взаимодействия компонентов, видах клиентских приложений, парадигме визуального проектирования и предметной ориентированности встроенного языка системы, подходами к хранению различной информации (иерархической, имеющей привязку ко времени, объектных и необъектных сущностей и т.д.), понятием управляемого приложения и декларативным описанием пользовательского интерфейса в приложениях на платформе «1С:Предприятие».

Предусмотрена углублённая проработка отдельных вопросов двух классов задач: оперативного учета и бухгалтерского учета в рамках реализации корпоративной информационной системы предприятия, основных принципов построения модели хранения данных, механизмов платформы, заложенных для решения этих задач, основ организации аналитического учета.

А также рассмотрены аспекты технологии задач сложных периодических расчетов в рамках реализации корпоративной информационной системы предприятия, основных принципов построения модели хранения данных, механизмов платформы, заложенных для решения этих задач.

Библиографический список

1. https://its.1c.ru – сайт Информационно-технологического сопровождение пользователей системы «1С:Предприятие».

2. Профессиональная разработка в системе «1С:Предприятие»: в 2 т. / 2-е изд.– М.: 1С-Паблишинг, 2012.–Т.1.–690с.

3. Профессиональная разработка в системе «1С:Предприятие»: в 2 т. / 2-е изд.– М.: 1С-Паблишинг, 2012.–Т.2.–683с.