Муниципальное бюджетное образовательное учреждение  
высшего образования  
«Самарская академия государственного и муниципального управления»  
  
Юридический факультет  
Кафедра Общеобразовательных дисциплин и права

Пояснительная записка  
к курсовому проекту  
по дисциплине «Корпоративные информационные системы в структуре архитектуры предприятий и бизнеса»

Выполнил Иванов И.И.,  
 студент группы 14.21-З  
  
Проверила Коробецкая А.А.  
  
Оценка дата

Самара  
2016

Реферат

Пояснительная записка к курсовому проекту.

Содержит: 30 стр., 10 рис., 2 табл., 10 источников, 2 приложения.

АРХИТЕКТУРА ПРЕДПРИЯТИЯ, СТРУКТУРА ЗАХТМАНА, ИНФОРМАЦИОННАЯ СИСТЕМА, УЧЕТ ЗАКАЗОВ, БАЗА ДАННЫХ, UML, ACCESS

Рассмотрена структура Захтмана для архитектуры предприятий, ее назначение, особенности, преимущества и недостатки. Выполнено проектирование подсистемы ИС предприятия для учета заказов товаров. Сформированы модели системы в соответствии с методологией UML. Реализована структура БД и основные запросы к системе.

Содержание

[Перечень используемых сокращений 4](#_Toc451568145)

[Введение 5](#_Toc451568146)

[1 Структура Захтмана для архитектуры предприятий 6](#_Toc451568147)

[1.1 Основные понятия и задачи архитектуры предприятия 6](#_Toc451568148)

[1.2 Сущность модели Дж. А. Захмана 7](#_Toc451568149)

[1.3 Преимущества и недостатки применения модели Захмана 13](#_Toc451568150)

[2 Проектирование информационной системы учета заказов товаров 15](#_Toc451568151)

[2.1 Разработка требований к системе 15](#_Toc451568152)

[2.2 Моделирование системы по методологии UML 17](#_Toc451568153)

[2.2.1 Описание бизнес-логики процесса заказа товара 19](#_Toc451568154)

[2.2.2 Диаграмма вариантов использования 21](#_Toc451568155)

[2.2.3 Диаграмма классов 26](#_Toc451568156)

[2.3 Реализация структуры базы данных 28](#_Toc451568157)

[2.3.1 Разработка структуры таблиц 28](#_Toc451568158)

[2.3.2 Реализация схемы данных 30](#_Toc451568159)

[2.3.3 Составление основных запросов к базе данных 30](#_Toc451568160)

[Заключение 34](#_Toc451568161)

[Список использованных источников 35](#_Toc451568162)

# Перечень используемых сокращений

БД – база данных

ИС – информационная система

ОЗУ – оперативное запоминающее устройство

СУБД – система управления базами данных

ФИО – фамилия, имя, отчество

ЭВМ – электронная вычислительная машина

TOGAF – The Open Group Architecture Framework

UML – Unified Modeling Language

# Введение

Проектирование архитектуры предприятий и бизнеса – важный аспект работы современного бизнес-аналитика. Как правило, разработка и анализ архитектуры предприятия связана с автоматизацией его работы путем создания информационной системы.

Необходимость проектирования архитектуры связана с наличием двух проблем. Первая – это возрастающая сложность и стоимость ИС. Вторая – это неэффективная организация бизнеса и его автоматизация, т.е., несмотря на широкое внедрение ИС, они зачастую не позволяют повысить эффективность работы предприятия.

В результате предприятия получают высокие затраты и низкую эффективность. Эти проблемы, впервые выявленные уже почти 30 лет назад, сегодня достигли критической точки. Стоимость и сложность ИТ-систем выросли экспоненциально, а реальная польза от них резко снизилась [3].

В своей статье 1987 г. Дж. Захман пишет: «Для снижения затрат и обеспечения успеха бизнеса, все больше зависящего от информационных систем, необходим строгий подход к управлению такими системами» [3].

Сегодня существует несколько десятков методик проектирования архитектуры, однако, первой из них, заложившей основные принципы, была методика Захмана. Она была развита по сравнению с первоначальным вариантом и сегодня является одной из наиболее популярных.

Для практической реализации методик проектирования архитектуры предприятий необходимо владеть методами моделирования ИС и баз данных.

Одним из наиболее популярных на сегодняшний день средств моделирования ИС является UML – унифицированный язык моделирования. Из более чем 20 диаграмм UML чаще всего применяются две – диаграмма вариантов использования и диаграмма классов. На основе диаграммы сущностных классов строится структура таблиц и схема данных БД. Описанные на диаграмме вариантов использования функции реализуются на основе запросов к БД.

Таким образом, в данной курсовой работе ставится две задачи: изучить теоретические аспекты проектирования архитектуры предприятия на примере методики Захмана и освоить практические аспекты проектирования ИС средствами UML на примере системы учета заказов товаров.

# Структура Захтмана для архитектуры предприятий

## Основные понятия и задачи архитектуры предприятия

При разработке информационных систем на начальном этапе требуется проанализировать объект автоматизации (предприятие) и спроектировать архитектуру, т.е. каркас будущей системы.

Основная задача архитектуры предприятия – снижение сложности ИС и затрат на их сопровождение при увеличении ценности бизнеса и росте эффективности, а в конечном итоге – повышение конкурентоспособности предприятия в условиях все более жесткой конкуренции.

Отметим, что существует два основных подхода к определению архитектуры.

Первый подразумевает, что архитектура – это описание структуры некоторой сложной системы в определенный момент времени. Это статичный подход, аналогичный схемам и чертежам здания, города, сада или компьютерной сети.

Второй подход рассматривает архитектуру как процесс, т.е. набор руководств, правил и/или стандартов, которые применяются в процессе построения новых систем. Иначе говоря, второй, динамический, смысл архитектуры заключается в создании системы правил, обеспечивающих направленный переход из текущего состояния ИС предприятия в некоторое будущее (желательно, лучшее) состояние.

Обобщающее эти две точки зрения определение приводится в спецификации IEEE-1471-2000 [4]: *архитектура* – фундаментальная организация системы, реализованная в ее компонентах, связях компонентов друг с другом и окружающей средой и принципах, определяющих ее проектирование и развитие

В свою очередь, *архитектура предприятия* — архитектура, в которой системой является целое предприятие, в частности, бизнес-процессы, технологии и информационные системы

Модель архитектуры предприятия аккумулирует знания о его процессах, поведении, информационных и материальных потоках, ресурсах и организационных единицах, инфраструктуре и архитектуре систем.

Существуют различные подходы или рамочные модели, **методики** (по-английски *frameworks*) к описанию архитектуры предприятия. Эти методики задают классификацию основных элементов архитектуры, их взаимосвязей и единые принципы для их описания, а также описание используемых правил (политик), стандартов, процессов, моделей, которые используются для определения различных элементов архитектуры на разных уровнях абстракции.

Каждая методика является инструментом для создания не одной конкретной, а целого спектра различных архитектур. Она, как правило, включает в себя:

* описание методов проектирования архитектуры в терминах использования определенных «строительных блоков»;
* описание того, как эти «строительные блоки» связаны между собой;
* набор инструментов для описания элементов архитектуры;
* общий словарь используемых терминов.

Методики также могут содержать список рекомендуемых стандартов и совместимых продуктов, которые могут использоваться для реализации различных элементов архитектуры. Важно понимать, что методики не только задают набор документов и планов, необходимых для описания предприятия, но и определяют, как все эти элементы описания связаны между собой.

Одними из наиболее распространенных являются следующие методики:

* модель Захмана;
* методика TOGAF;
* методики, опубликованные аналитическими компаниями, такими как Gartner, Giga Group, META Group и другими;
* методика POSIX 1003.23, которая основывается на разработках компании Cap Gemini, переданных для публичного использования в 1996 году.

Для государственных организаций существуют специальные методики, такие как разрабатываемая при поддержке правительства США Федеральная Архитектура Госорганизаций (FEAF – Federal Enterprise Architecture Framework) или используемая в Министерстве Обороны США DoDAF (Department of Defence Architecture Framework).

## Сущность модели Дж. А. Захмана

Модель (методика) Захмана является одной из ранних попыток связать характеристики информационной системы с бизнес-задачами предприятия. Джон А. Захман (John A. Zachman) предложил простую, но концептуально мощную схему, показывающую различные уровни представления архитектуры ИС, различные виды ее обеспечения, а также их основные взаимосвязи.

Этот метод хорошо известен в мировой практике. Суть его сводится к формализованному представлению модели предприятия в виде матрицы. Модель Захмана основана на дисциплине классической архитектуры и обеспечивает общий словарь и набор перспектив, или структур (framework), для описания современных сложных корпоративных систем.

Само понятие «архитектура предприятия» впервые появилось в 1987 г. в статье Дж.А. Захмана «Структура архитектуры информационных систем», опубликованной в журнале IBM Systems Journal [3]. В этой статье автор изложил свое видение архитектур предприятий и связанных с ними проблем, которое задало направление развития на последующие десятилетия.

Видение Захмана заключалось в том, что для обеспечения высокой ценности и гибкости бизнеса необходим целостный подход к архитектуре систем, в рамках которого каждая существенная проблема рассматривается с разных точек зрения. Такой подход к созданию архитектуры систем представляет собой то, что Захман изначально называл «архитектурной структурой ИС», а впоследствии – «структурой архитектуры предприятия».

«Структура» Захмана, скорее, представляет собой классификацию для упорядочения элементов архитектуры (проектной документации, спецификации и моделей). Причем классификация ведется в двух направлениях: с одной стороны, учитываются заинтересованные лица (например, владелец бизнеса и строитель), с другой – конкретная проблема, которую необходимо устранить (например, проблема с данными и функциональностью).

Захман изначально проводил аналогию между построением ИС и строительной отраслью. В строительстве элементы архитектуры (архитектурные артефакты) неявным образом организованы в двумерную структуру.

Первым измерением являются различные «*игроки*». В случае строительства здания такими игроками являются владелец (оплачивает проект), строитель (координирует процесс постройки) и архитектор-планировщик (обеспечивает соблюдение строительных норм и правил).

Каждому игроку необходима полная информация о будущем здании и процессе его постройки, однако понятие полноты для каждого игрока свое. Владелец заинтересован в полном описании функциональности и эстетики здания. Строитель заинтересован в полном описании строительных материалов и этапов строительства. При этом владельца не интересуют гвозди в стенах, а строителя не интересует, виден ли из окон спальни восход солнца.

Вторым измерением классификации являются *описательные аспекты*: кто, что, где, когда, как и почему. Второе измерение не зависит от первого. И строитель, и владелец должны знать «что», но это «что» для владельца отличается от «что» для строителя.

В первой статье и последующей работе в 1992 г. [9] Захман предложил шесть описательных аспектов (данные, функция, сеть, люди, время и мотивация) и шесть игроков (планировщик, владелец, проектировщик, строитель, субподрядчик и предприятие). Эти два измерения можно представить в виде таблицы или матрицы, как показано на рис. 1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **ЧТО** | **КАК** | **ГДЕ** | **КТО** | **КОГДА** | **ЗАЧЕМ** |  |
| **Планировщик** | Перечень важных понятий и объектов | Перечень основных бизнес-процессов | Террито­риальное распо­ложение | Ключевые организации | Важнейшие события | Бизнес-цели и стратегии | *Сфера действия (контекст)* |
| **Владелец, менеджер** | Концепту­альная модель данных | Модель бизнес-процессов | Схема логистики | Модель потока работ (workflow) | Мастер-план реализации | Бизнес-план | *Модель предпри­ятия* |
| **Конструктор, архитектор** | Логическая модель данных | Архитектура приложения | Модель распре­деленной архитектуры | Архитектура интерфейса пользо­вателя | Структура процессов | Роли и модели бищнес-правил | *Модель ИС* |
| **Проекти­ровщик** | Физическая модель данных | Системный проект | Техноло­гическая архитектура | Архитектура презентации | Структуры управления | Описание бизнес-правил | *Физическая модель ИС* |
| **Разработчик** | Описание структуры данных | Программный код | Сетевая архитектура | Архитектура безопас­ности | Определение временны́х привязок | Реализация бизнес-логики | *Детали реализации* |
|  | Данные | Работающие программы | Сеть | Реальные люди, организации | Бизнес-события | Работающие бизнес-стратегии | *Работа­ющее пред­приятие* |
|  | Данные | Функции | Дислокация, сеть | Люди | Время | Мотивация |  |

Рис. 1. Модель Захмана в виде матрицы

Например, с точки зрения владельца бизнеса, «данные» – это объекты, важные для бизнеса. Они могут включать сведения как о самих объектах, например, о клиентах и продукции, так и об отношениях между ними, например, о демографических группах и складских запасах. В разговоре о данных с владельцем бизнеса следует использовать именно этот язык.

С точки зрения разработчика БД, «данные» – это связанные друг с другом таблицы, их строки и столбцы. В разговоре о данных с разработчиком БД следует говорить не о демографических группах клиентов, а о реляционных таблицах, запросах и связях «один-ко-многим».

Ни одна из этих точек зрения не является более важной или детализированной, чем другая. Они обе важны для целостного понимания архитектуры системы.

Основные правила заполнения таблицы следующие:

1. каждая клетка таблицы независима от других, вместе они образуют функционально полное пространство для описания системы ("базис");
2. порядок следования колонок несущественен;
3. каждая клетка содержит соответствующее описание аспекта реализации системы в виде определенной модели (диаграммы) или простого текста;
4. базовые модели для каждой из колонок являются уникальными;
5. все модели в каждой строке в совокупности образуют полное описание системы с выбранной перспективы;
6. заполнение клеток должно проводиться последовательно сверху вниз, пропускать строки нельзя.

Первая строка соответствует уровню планирования бизнеса в целом (**бизнес-модель**). На этом уровне вводятся достаточно общие основные понятия, определяющие бизнес – например, продукты и услуги, клиенты, расположение объектов бизнеса, а также формулируется бизнес-стратегия (колонка 6 – мотивация). Фактически, данная строка определяет контекст всех последующих строк.

Вторая строка (**концептуальная модель**) предназначена для определения в терминах бизнеса структуры организации, ключевых и вспомогательных бизнес-процессов.

Третий уровень (**логическая модель**) соответствует рассмотрению с точки зрения Системного Архитектора. Здесь бизнес-процессы описываются уже в терминах информационных систем, включая различные типы данных, правила их преобразования и обработки для выполнения определенных на уровне 2 бизнес-функций.

На четвертом уровне – **технологической или физической модели** – осуществляется привязка данных и операций над ними к выбранным технологиям реализации. Например, здесь может быть определен выбор реляционной СУБД, или средств работы с неструктурированными данными, или объектно-ориентированной среды.

Пятый уровень соответствует **детальной реализации системы**, включая конкретные модели оборудования, топологию сети, производителя и версию СУБД, средства разработки и собственно готовый программный код. Многие из работ на данном уровне часто выполняются субподрядчиками.

Последний, шестой уровень описывает **работающую систему**. На этом уровне могут быть введены, в том числе, такие объекты, как инструкции для работы с системой, фактические базы данных, работа службы HelpDesk. Необхиодимо заметить, что в исходной работе Захмана содержание этого уровня не детализируется.

Каждый из столбцов матрицы отвечает на один из поставленных выше вопросов.

ЧТО? – **данные** (data), любые формы представления информации, критически важной для бизнеса, в данном случае это основной субъект таблицы. Важно, что данные описываются во взаимосвязи между собой.

КАК? – **функции** (function), операции, выполняемые над субъектом, выделение и передача знаний. Это функциональное описание системы, здесь представлено то, как организация работает, как устроен поток работ, как используются данные.

ГДЕ? – сеть (network), расположение субъекта. Это, в конечном итоге, описание информационных потоков предприятия.

КТО? – люди (people), распределение ответственности и функции работников. Человек рассматривается в роли носителя и распорядителя знаний.

КОГДА? – время (time), может быть абсолютным или относительным, отражающим взаимосвязанность процессов.

ЗАЧЕМ – мотивация (motivation), ключевой вопрос бизнес цели и стратегия.

Первая колонка отвечает на вопрос "ЧТО?" и определяет используемые в системе данные. На верхнем уровне достаточным будет простое перечисление основных объектов, используемых в бизнесе. На втором уровне данные объекты объединяются в семантическую модель высокого уровня и обычно описываются в виде диаграммы «сущность-связь» (ER-модель) с отражением основных связей и наиболее существенных бизнес-ограничений. На третьем уровне эта модель приводится к нормализованной форме, определяются все атрибуты и ключи. Четвертый уровень представляет собой физическую модель данных в системе (в объектно-ориентированном подходе – иерархию классов). Следующий уровень содержит описание модели на языке управления данными для формирования таблиц, готовые библиотеки классов, табличные пространства СУБД. Наконец, последний уровень может описывать фактические наборы данных, в том числе такие характеристики, как журналы доступа, размеры реально занимаемого дискового пространства, статистику обращений и т.п.

Колонка функций (ответ на вопрос "КАК?") предназначена для последовательной детализации описания того, как миссия предприятия реализуется на уровне отдельных операций. В частности, на первом уровне достаточным будет простое перечисление бизнес-процессов. Второй уровень будет содержать модель бизнес-процессов, которая впоследствии детализируется в операции над данными и архитектуру приложений (уровень 3), методы классов (уровень 4), программный код (уровень 5) и, наконец, исполняемые модули. При этом, начиная с 4-го уровня, рассмотрение ведется уже не в рамках предприятия в целом, а по отдельным подсистемам или приложениям.

Следующая колонка (вопрос "ГДЕ?") определяет пространственное распределение компонентов системы и сетевую организацию. На уровне планирования бизнеса здесь достаточно определить расположение всех производственных объектов. На следующем уровне эти объекты объединяются в модель со связями, характеризующими взаимодействие между собой, – будь то обмен информацией или поставки товаров. На третьем уровне системной архитектуры осуществляется привязка компонент информационной системы к узлам сети. Четвертый уровень служит для определения физической реализации в терминах аппаратных платформ, системного программного обеспечения, а также средств промежуточного уровня (так называемое "middleware"), используемых для интеграции различных компонент информационной системы между собой. Типичным примером могут являться брокеры запросов или средства обмена сообщениями. На пятом уровне определяются используемые протоколы и спецификации каналов связи. Последний уровень описывает функционирование реализованной сети.

Колонка таблицы, отвечающая на вопрос "КТО?", определяет участников процесса. На уровне планирования бизнеса здесь представлен список подразделений предприятия и выполняемые ими функции. На следующем уровне приводится полная организационная диаграмма, а также могут быть определены общие требования к информационной безопасности. Далее последовательно определяются участники бизнес-процессов и их роли (описание вариантов использования), требования к интерфейсам пользователя и правила доступа к отдельным объектам, физическая их реализация на уровне кода или операторов определения доступа к таблицам в СУБД. Последний уровень описывает обученных пользователей системы.

Пятая колонка отвечает на вопрос "КОГДА?" и определяет временные характеристики бизнес-процессов и работы системы. Опять-таки детализация осуществляется сверху вниз, начиная от календарного плана (уровень 1) и основных параметров, характеризующих выполнение бизнес-процессов, – например, требование ко времени оформления сделки (уровень 2). На третьем уровне определяются события, вызывающие изменение состояния информационных объектов и инициацию операций над ними. На следующем уровне эти события транслируются в программные вызовы (триггеры) или передаваемые сообщения. Пятый уровень определяет физическую реализацию обработки таких событий. Наконец, на 6-м уровне – фактическая история функционирования системы.

Последняя колонка ("ПОЧЕМУ?" или "ЗАЧЕМ?") служит для определения мотивации и задает порядок перехода от задач бизнеса к требованиям и элементам информационных систем. Исходной точкой является бизнес-стратегия, которая затем последовательно транслируется в бизнес-план, затем в правила и ограничения для реализации бизнес-процессов, а на уровне 4 – в соответствующие приложения, необходимые для включения в состав информационных систем и, в дальнейшем, в их физическую реализацию.

Такая модель описания в целом полезна для идентификации возможных ограничений. Эти ограничения могут распространяться как от верхних уровней к нижним (например, указание руководства компании о выборе тех или иных средств, продуктов или принципов работы), так и в обратном направлении – например, возможности существующих технологий беспроводной связи в значительной степени определяют спектр предлагаемых услуг и организацию бизнес-процессов у провайдеров этих услуг.

Важным принципом предложенной модели является необходимость последовательного перехода при углублении детализации рассмотрения. Пропуск отдельных элементов, например, прямой переход от описания модели бизнес-процесса к физической реализации системы требует «телепатии» и почти всегда приводит к неудаче. На практике это часто случается при попытке разработки программы на основании только устного описания требований пользователя.

## Преимущества и недостатки применения модели Захмана

Исходя из сказанного, таблица Захмана позволяет согласовать требования основных лиц, участвующих и заинтересованных в разработке ИС предприятия:

* убедиться в том, что точка зрения каждого заинтересованного лица была рассмотрена в каждом описательном аспекте;
* улучшить элементы архитектуры путем точной подгонки описательных аспектов под конкретную аудиторию;
* убедиться в том, что все бизнес-требования руководства предприятия технически реализуемы;
* убедить руководство предприятия в том, что ИТ-специалисты не планируют реализовывать бесполезные функции;
* убедить ИТ-разработчиков в том, что при планировании учитываются их интересы и возможности.

В качестве преимуществ данной модели можно указать следующие.

Во-первых, данную модель удобно применять для классификации всей информации, описывающей предприятие и информационные системы этого предприятия, выявления "белых пятен" и координации работ.

Во-вторых, данную модель можно использовать на метауровне – для сравнения различных реализаций создания архитектур предприятия.

Наконец, она может являться удобным средством для использования в отдельных проектах. Например, в проекте по созданию корпоративного информационного портала необходимо определить элементы в строках 3-5 колонки 4 – соответственно, требования пользователей к представлению данных, интерфейсы и спецификацию по разграничению доступа с учетом существующих "унаследованных" компонент информационной системы. Эта существующая технологическая архитектура, в свою очередь, рассматривается в ячейке на пересечении четвертой строки и третьего столбца таблицы.

Нельзя, конечно, считать, что данная модель лишена недостатков. Один из них заключается в том, что при применении ее на практике возникают определенные трудности, связанные с отсутствием механизма распространения изменений между элементами таблицы. Действительно, предположим, что изменилась организация процесса поставок в компании (схема логистики). Это потребует отслеживания всех взаимосвязей, проверки актуальности и внесения изменений в модели и другие артефакты во всех потенциально затрагиваемых ячейках.

Другим ограничением модели является отсутствие рассмотрения системы в динамике. Действительно, каждый элемент таблицы может содержать как описание существующего состояния («как есть»), так и целевого, а также всех промежуточных состояний. При этом сама модель не содержит средств для четкого разделения этих различных временных срезов.

Методология Захмана не дает пошаговых инструкций по созданию архитектуры. Методология Захмана даже не позволяет определить, является ли создаваемая архитектура лучшей из возможных. Кроме того, методология Захмана не позволяет определить, необходимо ли вообще создавать новую архитектуру. Для решения этих и других проблем необходимо обратиться к другим методологиям.

Тем не менее, существуют специализированные продукты, такие как Popkin Software System Architect (в настоящее время не развивается, последняя актуальная версия – System Architect 2001 [10]), которые фактически основаны на модели Захмана и позволяют достаточно эффективно управлять созданием моделей и артефактов описания архитектуры предприятия [5].

Многие сторонники структуры Захмана рассматривают ее как междисциплинарную, распространяющуюся далеко за пределы ИТ. Например, в одной популярной книге, посвященной методологии Захмана, говорится следующее: «...рано или поздно вы обнаружите, что структура Захмана присутствует во всем, чем вы занимаетесь, а не только в ИТ-проектах. Когда вы поймете структуру Захмана, вы станете более эффективным во всем. Во всем, не больше и не меньше. Это утверждение абсолютно серьезно». [6]

Сам Захман в своем интервью электронному журналу «Enterprise Architect Online» без ложной скромности сравнил свою модель с периодической таблицей элементов Менделеева и назвал ее «периодической таблицей описательных представлений предприятия или двумерной системой, схемой классификации» [7]. Это действительно ко многому обязывающее сравнение, но Захман привел следующие аргументы в его пользу: «Вопросы что, как, где, кто, когда и зачем существуют тысячи лет. И они будут существовать еще тысячи лет. Требования к системам, процесс проектирования, производства или концептуальный, логический и физический уровень описания также существуют в течение тысяч лет и будут существовать еще тысячи лет. Логическая структура классификации, схема – неизменны».

В конце концов, как отмечает Захман, коммерческие предприятия и государственные организации должны понимать, что путь к эффективным информационным системам требует систематических подходов в проектировании (engineering). Если вы, например, занимаетесь большими проектами, связанными с интеграцией государственных информационных систем, то «...назначить одного ответственного человека – это еще не решение проблемы. Никакого чуда просто так не произойдет. В какой-то момент вы поймете, что настоящий процесс проектирования должен быть реализован для того, чтобы интегрировать эти объекты».

# Проектирование информационной системы учета заказов товаров

## Разработка требований к системе

В качестве **объекта автоматизации** выступает процесс учета заказов товаров торговой фирмы от покупателей – физических лиц, включая:

* прием заказа от покупателя;
* обработку заказа менеджером;
* оплату заказа покупателем;
* отгрузку заказа со склада;
* получение заказа покупателем.

**Требования к информационному и методическому обеспечению**: система должна вести учет в соответствии со следующими документами:

* публичная оферта, размещенная на сайте компании;
* прайс-лист компании со списком имеющихся товаров;
* журнал учета движения товаров на складе (ТОРГ-18);
* список менеджеров, осуществляющих обработку заказов;
* бланки заказа покупателей в бумажном или электронном виде;
* документы об оплате товара;
* накладные и иные документы от транспортных компаний, осуществляющих доставку товаров.

**Требования к техническому обеспечению**:

* IBM PC совместимый персональный компьютер;
* процессор – 500 МГц или более;
* не менее 256 МБ ОЗУ;
* 2 ГБ свободного дискового пространства;
* разрешение экрана не менее 800х600 точек;
* доступ к сети Интернет на скорости 128 Кбит/с или выше.

**Требования к программному обеспечению**:

* операционная система: Windows XP/Vista/7/8/10;
* СУБД: MS Access 2007.

**Общие требования к проектируемой системе**:

1. функциональные требования:
   * ведение списка клиентов с указанием номера клиента, ФИО, контактных данных (телефон, e-mail, адрес доставки);
   * регистрация заказов от клиентов (заказ может содержать несколько товарных позиций с указанием количества), с указанием способа доставки, способа оплаты, с вычислением общей стоимости;
   * формирование запроса для проверки наличия и резервирования товара на складе через систему складского учета;
   * согласование заказа с клиентом с возможностью редактирования заказа;
   * подтверждение оплаты заказа клиентом;
   * отмена резервирования в случае отказа клиента от заказа или отсутствия оплаты в установленный срок;
   * отметка с указанием даты и времени об отгрузке оплаченного заказа со склада и передачи в центр выдачи либо в службу доставки;
   * отметка с указанием даты и времени о получении товара покупателем;
   * возврат товара до получения покупателем с указанием причины;
   * информирование клиента о состоянии заказа через email или по телефону;
   * формирование итогового отчета по сумме оплаченных и отгруженных товаров;
   * формирование отчета о неполученных заказах;
   * поиск неоформленных (несогласованных) и неоплаченных заказов;
   * поиск постоянных клиентов (сделавших более трех заказов, причем последний из них в течение последнего года).
2. требования к надежности:
   * доступ к системе должен осуществляться только зарегистрированными пользователями с помощью личного логина и пароля;
   * система должна поддерживать резервное копирование и восстановление БД.
3. условия эксплуатации:
   * максимальное время отклика системы – 10с;
   * система должна соответствовать санитарным правилам и нормам СанПин 2.2.2/2.4 1340-03;
   * условия работы средств вычислительной техники должны соответствовать группе 1 пункта 1.3.1 ГОСТ 21552-84.

**Требования к документации**:

* проектирование системы осуществляется в соответствии с методологией UML;
* модель системы должно включать диаграмму вариантов использования и диаграмму сущностных классов.

## Моделирование системы по методологии UML

Унифицированный язык моделирования (Unified Modeling Language, UML) является графическим языком для визуализации, специфицирования, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению [2]. С помощью UML можно разработать детальный план создаваемой системы, содержащий не только ее концептуальные элементы, но и конкретные особенности, например классы, написанные на специальных языках программирования, схемы баз данных и программные компоненты многократного использования.

UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений и даже встроенных систем реального времени. Это очень выразительный язык, позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию. Несмотря на обилие выразительных возможностей, этот язык прост для понимания и использования.

Язык состоит из словаря и правил, позволяющих комбинировать входящие в него слова и получать осмысленные конструкции. В языке моделирования словарь и правила ориентированы на концептуальное и физическое представление системы.

Моделирование необходимо для понимания системы. При этом единственной модели никогда не бывает достаточно. Напротив, для понимания любой нетривиальной системы приходится разрабатывать большое количество взаимосвязанных моделей. В применении к программным системам это означает, что необходим язык, с помощью которого можно с различных точек зрения описать представления архитектуры системы на протяжении цикла ее разработки.

Словарь и правила такого языка, как UML, объясняют, как создавать и читать хорошо определенные модели, но ничего не сообщают о том, какие модели и в каких случаях нужно создавать. Это задача всего процесса разработки программного обеспечения.

Некоторые особенности системы лучше всего моделировать в виде текста, другие – графически. На самом деле, во всех сложных системах существуют структуры, которые невозможно представить с помощью одного лишь языка программирования. UML – графический язык, что позволяет понимать его людям, не являющимся специалистами в области ИТ и программирования.

UML позволяет специфицировать все существенные решения, касающиеся анализа, проектирования и реализации, которые должны приниматься в процессе разработки и развертывания системы программного обеспечения. В данном контексте специфицирование означает построение точных, недвусмысленных и полных моделей.

UML не является языком визуального программирования, но модели, созданные с его помощью, могут быть непосредственно переведены на различные языки программирования. Иными словами, UML-модель можно отобразить на такие языки, как Java, C++, Visual Basic, и даже на таблицы реляционной базы данных или устойчивые объекты объектно-ориентированной базы данных. Те понятия, которые предпочтительно передавать графически, так и представляются в UML; те же, которые лучше описывать в текстовом виде, выражаются с помощью языка программирования.

Помимо прямого отображения в языки программирования UML в силу своей выразительности и однозначности позволяет непосредственно исполнять модели, имитировать поведение систем и контролировать действующие системы.

### Описание бизнес-логики процесса заказа товара

Прежде, чем приступить к моделированию системы на языке UML, необходимо понять, что именно она должна автоматизировать, и убедиться, что разработчик БД получил полную информацию о соответствующей бизнес-логике.

Процесс заказа товара включает следующие этапы:

1. Размещение заказа клиентом. Заказ может включать несколько товарных позиций. Система принимает заказ через веб-интерфейс. Необходима поддержка функций добавления и удаления товаров в заказ, изменения количества, выбора способа получения (доставка или самовывоз), способа оплаты (наличными при получении или безналичный перевод), подсчета общей стоимости товара. При регистрации заказа клиент указывает необходимые контактные данные: ФИО, телефон, email, адрес.
2. Заказ регистрируется в системе и передается менеджеру по продажам. Менеджер проверяет наличие товара на складе через ИС и согласовывает заказ с клиентом. Клиент может в любой момент отменить заказ, связавшись с менеджером по телефону или email.
3. После успешного согласования заказа товар резервируется на складе. Клиенту отправляется уведомление о готовности заказа к отгрузке с напоминанием об оплате.
4. Клиент оплачивает товар:
   1. В кассе при получении в пункте выдачи.
   2. Наличными курьеру при доставке.
   3. Безналичным переводом.
5. После подтверждения оплаты (автоматически при получении денежных средств или после предъявления клиентом документа об оплате) товар отгружается со склада:
   1. В пункт выдачи товара.
   2. Службе доставки.
6. Клиенту направляется сообщение о готовности товара к получению.
7. Клиент получает товар:
   1. В пункте выдачи товара.
   2. На руки от курьера.
8. Если клиент по каким-то причинам не получил товар или отказался от покупки, товар возвращается на склад, а заказ отменяется с указанием причины. Клиенту направляется уведомление об аннулировании заказа. Возврат полученного товара и денежных средств выходит за рамки разрабатываемой системы.

Таким образом, в процессе заказа товара можно выделить следующие основные сущности:

*субъекты взаимодействия*:

* клиент;
* менеджер;
* склад, включая ИС складского учета;
* пункт выдачи;
* служба доставки;
* система безналичной оплаты;

*объекты*:

* заказ;
* товар;
* платеж.

Основные операции, совершаемые в процессе обработки заказа товара:

* регистрация заказа;
* согласование заказа менеджером;
* резервирование товара на складе;
* оплата заказа клиентом;
* отгрузка товара складом;
* передача товара в пункт выдачи;
* доставка товара клиенту;
* получение заказа клиентом;
* отмена заказа;
* возврат товара на склад.

Основные действия, выполняемые непосредственно в системе:

* регистрация заказа;
* проверка наличия товара;
* резервирование товара;
* отметка об оплате заказа;
* отметка о получении товара;
* отмена (аннулирование) заказа;
* отправка клиенту уведомлений о состоянии заказа.

Таким образом, часть операций, связанных с заказом товара, выходит за рамки разрабатываемой ИС. Это операции, связанные с обработкой платежей, логистикой и складским учетом.

### Диаграмма вариантов использования

*Диаграмма вариантов использования* (Use Case) применяется для описания функциональных требований к системе, ролей пользователей, а также определения границ системы. Диаграмма вариантов использования составляется, как правило, в ходе этапов анализа предметной области и постановки задачи. Это удобный инструмент описания бизнес-логики системы, ее основного назначения и взаимодействия с внешними сущностями.

Основными объектами на диаграмме являются действующие лица (актеры, actors), прецеденты (варианты использования, use case) и связи между ними.

*Актер* представляет собой человека, внешнюю систему или любого другого активного субъекта, который взаимодействуют с системой и использует ее. На диаграмме актер изображается в виде человечка. Примеры актеров:

* пользователь системы;
* системный администратор;
* клиент;
* сотрудник;
* отдел кадров;
* время (системный таймер);
* операционная система;
* внешняя база данных.

*Прецедент* описывает некоторое действие, которое актер (или несколько актеров) может выполнить в системе. Прецеденты обозначаются на схеме овалами и описываются глаголами либо отглагольными существительными с зависимыми словами. Например, «Внести сведения о сотруднике организации», «Покупка товара», «Ввод пароля», «Печать годового отчета».

На диаграмме Use Case используются следующие виды связей: ассоциация, направленная ассоциация, обобщение, зависимость, включение, расширение.

*Ассоциация* (assotiation) изображается сплошной линией без стрелки и чаще всего используется для связей между актером с одной стороны и прецедентом с другой. Она не имеет направления и показывает, что данный прецедент выполняется данным актером.

*Направленная ассоциация* (directed assotiation) показывается сплошной линией со стрелкой и является частным случаем обычной ассоциации. Она применяется, когда нужно указать направление связи. Например, при вводе данных можно поставить направление ассоциации от пользователя к системе, а при выводе – наоборот.

*Обобщение* (generalization) показывается сплошной стрелкой в виде пустого треугольника. Обобщение может связывать актера с актером или вариант использования с вариантом использования и показывает, что одно является более общим случаем другого. Стрелка ведется от частного к общему. Например, актер «Пользователь» является обобщением для актеров «Администратор» и «Гость». Вариант использования «Оплата» обобщает «Оплату в кассе» и «Безналичный перевод», а «Оплата в кассе», в свою очередь, обобщает «Оплату наличными» и «Оплату банковской картой».

*Зависимость* (dependency) изображается пунктирной линией со стрелкой. Она показывает, что один вариант использования зависит от другого. Например, «Выдача товара» зависит от «Выбора товара» и «Оплаты». Однако, зависимости на схеме обычно не показывают, если они достаточно очевидны.

*Включение* (include) и *расширение* (дополнение, extend) близки по смыслу и показываются пунктирной линией со стрелкой и соответствующей надписью в угловых скобках. Включение означает, что один вариант использования обязательно содержит в себе другой, как один из этапов выполнения, и связь ведется от целого к части. Например, «Авторизация пользователя» включает «Ввод пароля» и «Ввод имени пользователя» (пропустить их нельзя).

Расширение показывает, что один вариант использования дает дополнительные возможности другому, связь ведется от дополнения к основному. Например, «Геопозиционирование» расширяет вариант использования «Ввод адреса» (но не обязательно используется).

Для описания сложных проектов, как правило, недостаточно одной диаграммы вариантов использования (или она будет слишком большой). Разные диаграммы могут показывать систему на разных уровнях (вся система целиком и ее отдельные части) или с разных точек зрения (бизнес-логика, автоматизация, программное обеспечение, аппаратное обеспечение и др.).

Диаграмма, показывающая бизнес-логику процесса заказа товара показана на рисунке . Основные актеры – менеджер и клиент. На диаграмме показаны внешние актеры, с которыми взаимодействует система, и ее границы. Варианты использования, которые находятся за границами системы, закрашены серым.

На рисунке показана диаграмма вариантов использования с точки зрения работы информационной системы. Здесь представлена только внутренняя логика работы системы и ее пользователи.

System

**Менеджер**

**Регистрация**

**заказа**

**Провека наличия**

**товара на складе**

**Клиент**

**Согласование**

**заказа**

**Склад**

**Подтверждение**

**оплаты**

**Резервирование**

**товара**

**Отмена**

**резервирования**

**Отметка об**

**отгрузке товара**

**Отметка о**

**получении**

**Получение**

**товара**

**Служба доставки**

**Доставка товара**

**Передача товара в**

**службу довтавки**

**Оплата товара**

**Отгрузка товара**

**Указание причины**

**возврата**

**Получение в**

**пункте выдачи**

**Пункт выдачи**

**Передача товара**

**в пункт выдачи**

**Отмена заказа**

**Наличными при**

**получении**

**Безналичный**

**перевод**

<<include>>

Рис. 2. Диаграмма вариантов использования, описывающая бизнес-логику процесса заказа товара

**Пользователь**

**Менеджер**

**Администратор**

**Авторизация**

**Управление**

**пользователями**

**Архивация и**

**восстановление**

**Ведение**

**справочников**

**Клиент**

**Создание заказа**

**Выбор товара**

**Расчет суммы**

**заказа**

**Редактирование**

**заказа**

<<include>>

<<include>>

<<include>>

<<include>>

**Система**

**складского учета**

**Резервирование**

**товара**

**Отметка об**

**отгрузке товара**

**Отметка о**

**получении**

**Провека наличия**

**товара на складе**

**Отмена**

**резервирования**

**Ввод сведений**

**о клиенте**

<<include>>

<<extend>>

**Отмена заказа**

**Указание причины**

**отмены**

<<include>>

**Выбор формы**

**оплаты**

<<include>>

<<extend>>

**Выбор способа**

**получения**

<<include>>

<<extend>>

**Отправка сообщения**

**о состоянии заказа**

<<include>>

<<include>>

<<include>>

**Утверждение**

**заказа**

<<include>>

Рис. 3. Диаграмма вариантов использования, описывающая бизнес-логику процесса заказа товара

### Диаграмма классов

Диаграммы классов при моделировании объектно-ориентированных систем встречаются чаще других. На таких диаграммах показывается множество классов, интерфейсов, коопераций и отношений между ними.

*Класс* – это множество однотипных объектов, обладающих общим набором свойств (атрибутов) и методов поведения. Описание класса содержит стереотип, название, перечень атрибутов и перечень методов (операций). Любая часть описания (кроме названия) может быть пропущена. На схеме классы показываются в виде прямоугольников, разделенных на три части: заголовок, атрибуты и методы.

Между классами могут присутствовать следующие виды связей:

* ассоциация;
* обобщение;
* агрегация;
* композиция;
* зависимость.

*Ассоциация,* *обобщение* и *зависимость* имеют тот же смысл, что и на диаграмме вариантов использования.

*Агрегация* означает, что один класс является составной частью другого. Пример: двигатель является частью автомобиля, студент является частью группы.

*Композиция* – частный случай агрегации. В этом случае часть не может существовать без целого. Пример: факультет является частью вуза и не может существовать без него.

Классы по своей роли в системе делятся на группы. Сам по себе язык UML жестко не оговаривает эти группы, оставляя группировку на усмотрение разработчиков. На основе опыта, накопленного при создании автоматизированных систем, целесообразно выделить следующие группы (категории, стереотипы) классов:

* граничные (boundary) классы;
* сущностные (entity) классы;
* классы управления (control);
* классы прикладной логики (logic).

Объекты *сущностных классов* представляют собой блоки длительно хранимой информации, используемые для организации баз данных и баз знаний, файловых систем хранения данных различной логической структуры. В основном в этих классах развит атрибутный раздел, однако, имеется небольшое число операций контроля ограничений целостности как стандартных, так и специфичных для данной предметной области.

На диаграмме сущностных классов чаще всего используются связи-ассоциации с указанием кратности на каждой стороне связи:

* 1 – один;
* 0..1 – ноль или один (необязательная связь);
* \* – много;
* 0..\* – ноль или много (необязательная связь);
* 1..\* – один или много (как минимум один);
* любое конкретное число или диапазон значений.

На рисунке 4 представлена диаграмма сущностных классов разрабатываемой ИС.



Рис. 4. Диаграмма сущностных классов

На диаграмме присутствует семь основных классов, а также один ассоциативный класс «Получение статуса». Классы «Форма оплаты» и «Способ получения» соответствуют справочникам системы (содержат редко изменяемые данные).

## Реализация структуры базы данных

Для разработки БД была выбрана СУБД Microsoft Access 2007, входящая в состав Microsoft Office. Данная СУБД предназначена для работы с относительно небольшими локальными базами данных. Ее достоинствами является простой и удобный пользовательский интерфейс, а также возможность разработки собственного интерфейса БД в виде форм и отчетов непосредственно в рамках СУБД, без необходимости разработки отдельного клиентского приложения.

Отметим, что для рассматриваемой темы более обоснованным был бы выбор клиент-серверной архитектуры и СУБД MySQL, чаще всего применяющаяся в сочетании с web-интерфейсом, но в учебных целях рассматривается создание БД именно в Access.

### Разработка структуры таблиц

Разработанная диаграмма сущностных классов описывает концептуальную модель БД. Для перехода к схеме БД необходимо:

1. каждую сущность преобразовать в таблицу, каждый атрибут – в поле (столбец);
2. если присутствуют неатомарные атрибуты, то разбить их на несколько атрибутов или выделить в отдельную таблицу;
3. ввести промежуточные таблицы для связей типа «многие-ко-многим» и разрешить эти связи к типу «один-ко-многим»;
4. при необходимости добавить справочные таблицы;
5. выбрать первичные ключи (primary key) для каждой сущности, при необходимости ввести атрибут «Код» уникальным числовым номером ;
6. добавить внешние ключи (foreign key):
   1. для связей «один-ко-многим» на стороне много;
   2. для связей «один-к-одному» с необязательным классом принадлежности – на стороне 0..1.

Таким образом, для рассматриваемой схемы получим следующий перечень таблиц и полей (первичные ключи подчеркнуты, внешние ключи имеют метку FK):

**Клиент** (Код, ФИО, Телефон, email, Адрес)

**Товар** (Артикул, Наименование, Цена)

**Заказ** (Номер, *Клиент* *(FK)*, Комментарий, *Способ получения (FK)*, *Форма оплаты (FK)*)

**Товарная позиция** (Код, *Заказ (FK)*, *Товар (FK)*, Количество, Цена)

**Способ получения** (Код, Название)

**Форма оплаты** (Код, Название)

**Статус заказа** (Код, Название, Текст сообщения)

**Получение статуса** (Код, *Заказ (FK)*, *Статус (FK)*, Дата/время)

Создадим соответствующие таблицы в MS Access. Все имена таблиц и полей являются допустимыми в данной СУБД, переименование не требуется. Выберем подходящий тип данных и размер для каждого поля.

Полученные таблицы и их поля представлены ниже. Названия ключевых полей подчеркнуты.

Таблица: Заказ

|  |  |  |
| --- | --- | --- |
| *Имя* | *Тип* | *Размер* |
| Номер | Длинное целое | 4 |
| Клиент | Длинное целое | 4 |
| Комментарий | Текстовый | 255 |
| Способ получения | Длинное целое | 4 |
| Форма оплаты | Длинное целое | 4 |

Таблица: Клиент

|  |  |  |
| --- | --- | --- |
| *Имя* | *Тип* | *Размер* |
| Код | Длинное целое | 4 |
| ФИО | Текстовый | 255 |
| Телефон | Текстовый | 10 |
| email | Текстовый | 30 |
| Адрес | Текстовый | 255 |

Таблица: Получение статуса

|  |  |  |
| --- | --- | --- |
| *Имя* | *Тип* | *Размер* |
| Код | Длинное целое | 4 |
| Заказ | Длинное целое | 4 |
| Статус | Длинное целое | 4 |
| Дата/время | Дата/время | 8 |

Таблица: Способ получения

|  |  |  |
| --- | --- | --- |
| *Имя* | *Тип* | *Размер* |
| Код | Длинное целое | 4 |
| Название | Текстовый | 255 |

Таблица: Статус заказа

|  |  |  |
| --- | --- | --- |
| *Имя* | *Тип* | *Размер* |
| Код | Длинное целое | 4 |
| Название | Текстовый | 255 |
| Текст сообщения | Текстовый | 255 |

Таблица: Товар

|  |  |  |
| --- | --- | --- |
| *Имя* | *Тип* | *Размер* |
| Артикул | Текстовый | 15 |
| Наименование | Текстовый | 100 |
| Цена | Денежный | 8 |

Таблица: Товарная позиция

|  |  |  |
| --- | --- | --- |
| *Имя* | *Тип* | *Размер* |
| Код | Длинное целое | 4 |
| Заказ | Длинное целое | 4 |
| Товар | Текстовый | 15 |
| Количество | Длинное целое | 4 |
| Цена | Денежный | 8 |

Таблица: Форма оплаты

|  |  |  |
| --- | --- | --- |
| *Имя* | *Тип* | *Размер* |
| Код | Длинное целое | 4 |
| Название | Текстовый | 255 |

### Реализация схемы данных

Реализуем схему данных, соответствующую разработанной модели данных (рисунок 5).

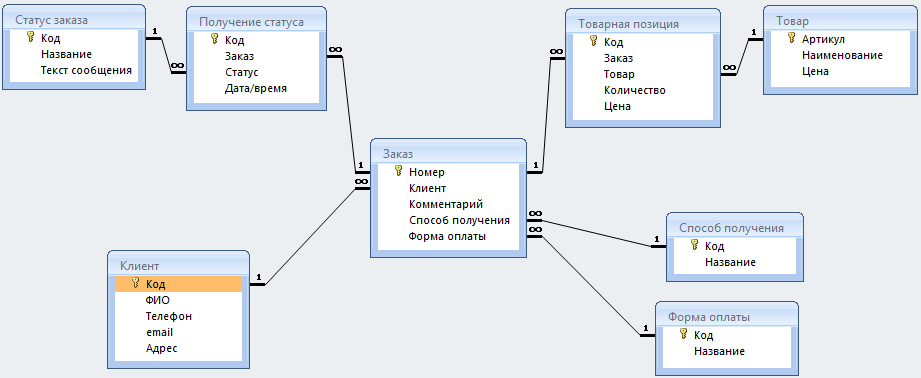


Рис. 5. Схема данных

Для всех связей установлено свойство «Проверка целостности данных». Для связи между полем Товар!Артикул и [Товарная позиция]!Товар установлено свойство «Каскадное обновление данных», чтобы при изменении артикула автоматически обновились данные в связанных записях. В других связях на стороне «один» участвуют поля типа «Счетчик», которые не изменяют своих значений, поэтому каскадное обновление не требуется.

### Составление основных запросов к базе данных

В требованиях к ИС указаны три основных запроса:

1. Вычисление общей стоимости заказа и каждой товарной позиции в нем.
2. Поиск постоянных клиентов, совершивших не менее 3 заказов, причем последний из них в течение последнего года.
3. Поиск неоформленных и неоплаченных заказов.

Реализация необходимых запросов в Конструкторе представлена на рисунках 6-11ю

Первый запрос, фактически, состоит из двух запросов: первый вычисляет стоимость каждой товарной позиции (Цена \* Количество), а второй – суммирует стоимость товарных позиций по каждому заказу (запрос с группировкой и итогами).

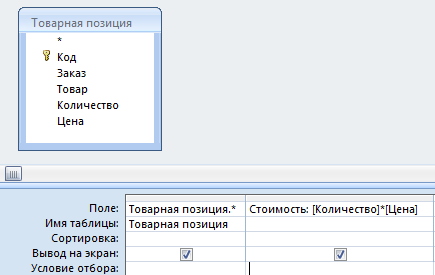


Рис. . Запрос на вычисление стоимости каждой товарной позиции

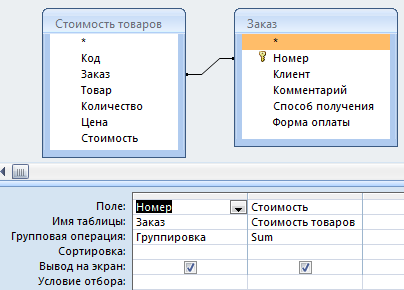


Рис. . Запрос на вычисление суммарной стоимости заказа

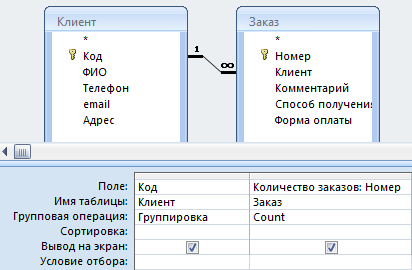


Рис. . Запрос на вычисление количества заказов у каждого клиента

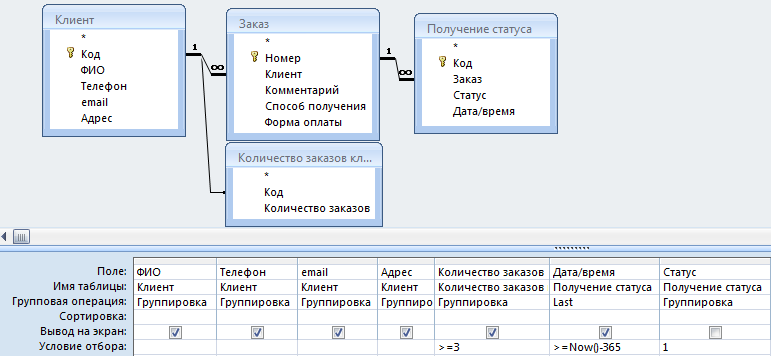


Рис. . Запрос на поиск постоянных клиентов

Второй запрос – поиск постоянных клиентов – требует предварительного вычисления количества заказов у каждого клиента. Затем клиенты отбираются по дате регистрации последнего заказа.

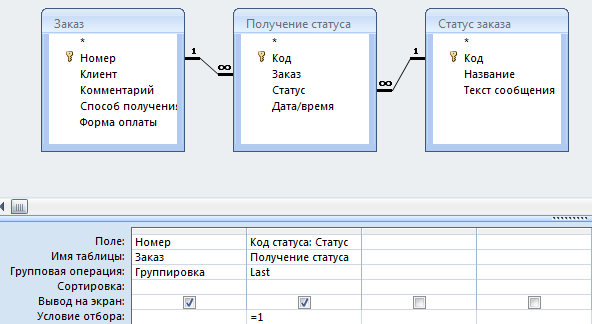


Рис. . Запрос на поиск неоформленных заказов

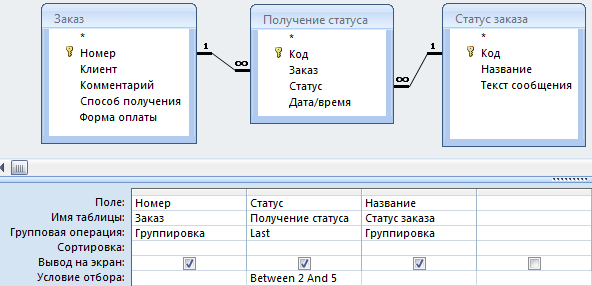


Рис. . Запрос на поиск неоплаченных заказов

Третий запрос также состоит из двух независимых запросов: первый ищет все неоформленные заказы (последний статус = 1 (зарегистрирован)), второй – все неоплаченные (последний статус между 2 (подтвержден) и 5 (оплачен)).

# Заключение

В курсовой работе рассмотрена самая первая методика проектирования архитектуры предприятия – методика Захмана, предложенная им в 1987 г. Методика основана на заполнении матрицы, в которой строки представляют собой различные точки зрения на систему, а столбцы – проблемы разработки системы.

Показано, что данная методика весьма универсальна и полезна для согласования интересов сторон, заинтересованных в разработке ИС предприятия (владельцев, менеджеров, бизнес-аналитиков, ИТ-специалистов). Однако она имеет и недостатки – не рассматривает развитие системы и не дает конкретных рекомендаций по практическому применению.

Во второй части работы выполнен анализ и проектирование ИС учета заказов товаров.

Сформулированы требования к системе, ее информационному, техническому, программному обеспечению, функционалу и условиям эксплуатации.

Выполнено описание бизнес-логики автоматизируемого процесса. Выделены основные этапы процесса заказа товара, определены субъекты и объекты взаимодействия.

Для проектирования ИС применен язык UML. Разработаны две диаграммы вариантов использования: одна с точки зрения бизнес-логики процесса, а вторая – с точки зрения вариантов использования ИС и пользователей. Это позволило более четко определить границы системы и ее функции.

Проектирование концептуальной схемы данных использовалась диаграмма сущностных классов. Были выделены основные сущности, их атрибуты и связи между ними. Концептуальная схема соответствует сформулированным требованиям к ИС.

Затем схема данных была перенесена в СУБД MS Access 2007. Созданы необходимые таблицы, определены типы данных полей. Созданы связи между таблицами. В завершение реализованы три запроса к БД, связанные с основным функционалом БД.

Таким образом, в курсовой работе были рассмотрены теоретические и практические аспекты проектирования архитектуры предприятия и информационных систем.

# Список использованных источников

1. Баронов, В.В. Информационные технологии и управление предприятием / В.В. Баронов, Г.Н.  Калянов, Ю.Н. Попов, И.Н. Титовский. – М.: Компания АйТи, 2009. – 328 с.
2. Буч, Г. Язык UML. Руководство пользователя / Гради Буч, Джеймс Рамбо, Ивар Якобсон. – М.: ДМК Пресс, 2007. – 496 с.
3. Сешнс, Р. Сравнение четырех ведущих методологий построения архитектуры предприятия [Электронный ресурс] / Роджер Сешнс. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/ee914379.aspx (дата обращения 19.05.2016).
4. Стандарт IEEE 1471-2000. Рекомендации IEEE по архитектурному описанию преимущественно программных систем.
5. Федоров, А. Средства проектирования данных // КомпьютерПресс / Алексей Федоров, Наталия Елманова. – №1, 2001. – Режим доступа: http://www.interface.ru/fset.asp?Url=/misc/sredprdn2.htm
6. Carol O'Rourke, Neal Fishman, and Warren Selkow «Enterprise Architecture Using the Zachman Framework». ISBN 0-619-06446-3 Published by Course Technology, 2003.
7. D. Rudy, Erecting the Framework [Электронный ресурс]. – 2004. – Режим доступа: http://www.ftponline.com/ea/magazine/spring/online/druby
8. John A. Zachman. A Framework for Information System Architecture. *IBM System Journal*, vol. 26, no. 3, 1987.
9. John A. Zachman, John F. Sowa Extending and Formalizing the Framework for Information Systems Architecture. *IBM Systems Journal*, Vol. 31, no.3, 1992. p. 590-616.
10. Popkin Software & Systems Inc. – Режим доступа: http://popkin-software-systems-inc.software.informer.com/