СОДЕРЖАНИЕ

1. ЛАБОРАТОРНАЯ РАБОТА «СТРУКТУРНЫЙ АНАЛИЗ ИНФОРМАЦИОННОЙ СИСТЕМЫ. РАЗРАБОТКА ДИАГРАММ ПОТОКОВ ДАННЫХ»

1.1 Цель работы

Целью данной работы является изучение методологии графического структурного анализа и построение структурной модели на основе диаграмм потоков данных.

1.2 Задание на лабораторную работу

Разработать структурную модель информационной системы или ее функционально законченной части.

1. Построить контекстную диаграмму потоков данных;
2. Осуществить декомпозицию контекстной диаграммы (построить диаграмму потоков данных 1-го уровня);
3. Осуществить декомпозицию диаграммы 1-го уровня (построить диаграмму потоков данных 2-го уровня);

- для оценки «хорошо» диаграмма потоков данных 1-го уровня должна содержать не менее 10 процессов, диаграмма потоков данных 2-го уровня должна представлять собой декомпозицию любых 3 процессов диаграммы 1-го уровня;

- для оценки «отлично» диаграмма потоков данных 1-го уровня должна содержать не менее 15 процессов, диаграмма потоков данных 2-го уровня должна представлять собой декомпозицию любых 5 процессов диаграммы 1-го уровня.

1.3 Порядок выполнения работы

1. Выбрать вариант информационной системы (см. ПРИЛОЖЕНИЕ);
2. Изучить теоретический материал, изложенный в подразделе 1.4;
3. Разработать структурную модель информационной системы;
4. Ознакомиться с требованиями по содержанию отчета в подразделе 1.5;
5. Написать отчет о работе.

Для выполнения лабораторных работы можно воспользоваться любой средой моделирования или CASE-средством, которое поддерживает соответствующие структурные нотации диаграмм. Рекомендуется использовать бесплатное для некоммерческого использования CASE-средство Software Ideas Modeler [1].

На данный момент акценты в области анализа и проектирования программного обеспечения сместились от структурного к объектно-ориентированному подходу, поэтому все современные CASE-средства нацелены на поддержку унифицированного языка моделирования (Unified Modeling Language, UML), однако многие также поддерживают и структурные нотации. Ознакомиться с полным списком средств моделирования для UML можно по ссылке https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools

Убедитесь, что выбранная вами среда моделирования или CASE-средство, поддерживает структурные нотации DFD и ERD.

1.4 Теоретический материал

Диаграммы потоков данных (Data Flow Diagrams, DFD) — методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки и хранилища данных, к которым осуществляется доступ.

При использовании структурной модели для анализа и проектирования систему представляют в виде иерархии диаграмм потоков данных. На каждом следующем уровне иерархий происходит декомпозиция, то есть разбиение на структурные составляющие процессов, которые преобразуют входную информацию с заданным алгоритмом в выходную. Когда достигается требуемая глубина декомпозиции, процессы нижнего уровня сопровождаются спецификацией.

В основе структурной модели лежат понятия внешней сущности, процесса, хранилища (накопителя) данных и потока данных.

Внешняя сущность — материальный объект или физическое лицо, выступающие в качестве источников или приемников информации, например, заказчик, персонал, пользователь (тот, кто использует продукт), клиент (тот, кто приобретает продукт), склад, дата-центр etc. Определение некоторого объекта или системы в качестве внешней сущности указывает на то, что объект или система находятся за пределами границ анализируемой информационной системы. В процессе анализа некоторые внешние сущности могут быть перенесены внутрь контура (диаграммы) анализируемой системы, если это необходимо, или, наоборот, часть процессов системы может быть вынесена за пределы диаграммы и представлена как внешняя сущность.

Процесс — преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. Физически процесс может быть реализован различными способами: это может быть подразделение организации (отдел), выполняющее обработку входных документов и составление отчетов, программа, аппаратно-реализованное логическое устройство, скрипт для реализации функций взаимодействия пользователя с веб-сайтом etc. Таким образом, физически преобразование может осуществляться программными средствами, вручную или специальными устройствами. Каждый процесс в системе имеет свой номер и связан с исполнителем, который осуществляет данное преобразование.

На верхних уровнях иерархии, когда процессы еще не определены, вместо понятия «процесс» используют понятия «система» и «подсистема», которые обозначают соответственно систему в целом или ее функционально законченную часть.

Накопитель данных представляет собой абстрактное устройство для хранения информации. Тип устройства и способы помещения, извлечения и хранения для такого устройства не детализируют. Физически это может быть база данных, файл, таблица в оперативной памяти, картотека на бумаге etc.

Идентификатор накопителя данных, который может быть произвольным числом, предваряет буква «D». Имя накопителя выбирается из соображения наибольшей информативности при проектировании информационной системы.

Накопитель данных в общем случае является прообразом будущей базы данных. Описание самих данных, хранение которых обеспечивается в накопителе, должно быть отражено в семантической модели в виде диаграммы сущность-связь (ЛР №2).

Поток данных — процесс передачи некоторой информации от источника к приемнику. Физически процесс передачи информации может происходить по кабелям под управлением программы или программной системы или вручную при участии устройств или людей вне проектируемой системы.

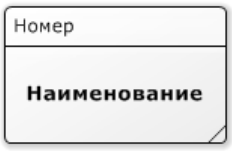
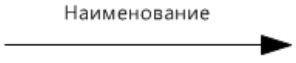
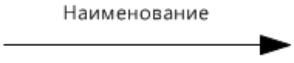
Таким образом, диаграмма иллюстрирует как потоки данных, порожденные некоторыми внешними сущностями, преобразуются соответствующими процессами (или подсистемами), сохраняются накопителями данных и передаются другим внешним сущностям — приемникам информации. В результате мы получаем структурную модель хранения (обработки) информации [2].

Построение иерархии диаграмм потоков данных начинают с диаграммы особого вида — контекстной диаграммы, которая определяет наиболее общий вид системы. На такой диаграмме показывают, как разрабатываемая система будет взаимодействовать с приемниками и источниками информации без указания исполнителей (процессов), иными словами описывают интерфейс между системой и внешним миром.

Если проектируемая система содержит большое количество внешних сущностей (более 10-ти), имеет распределенную природу или включает уже существующие подсистемы, то строят иерархии контекстных диаграмм.

Для графического описания диаграмм потоков данных в равной степени используют две нотации — Йордана (Yourdon) и Гейна-Сарсона (Gane-Sarson), которые отличаются синтаксисом. В настоящих методических указаниях используется нотация Гейна-Сарсона, однако при выполнении лабораторной работы студент вправе выбрать любую из представленных (табл.1.4).

Таблица 1.4 — Элементы диаграммы потоков данных [3]

Понятие	Нотация Йордана	Нотация Гейна-Сарсона
Внешняя сущность		
Система, подсистема или процесс		
Накопитель данных		
Поток данных		

Над линией потока, направление которого обозначают стрелкой, указывают, какая конкретно информация в данном случае передается (рис. 1.4).

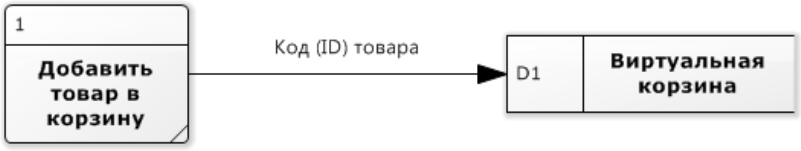


Рисунок 1.4 — Пример потока данных
(спецификация Гейна-Сарсона)

В нотации Гейна-Сарсона у процессов иногда добавляют дополнительное поле после наименования процесса — поле физической реализации, в котором указывают, какое подразделение организации, аппаратное устройство, программа или скрипт выполняют данный процесс (исполнитель). Данное поле не является обязательным для заполнения и во многих CASE-средствах в нотации Гейна-Сарсона оно отсутствует.

1.5 Содержание отчета

Отчет о работе должен содержать:

1. Титульный лист
2. Цель работы
3. Задание на лабораторную работу и вариант
4. Структурная модель информационной системы
   4.1 Контекстная диаграмма
   4.2 Диаграмма потоков данных 1-го уровня
   4.3 Диаграмма потоков данных 2-го уровня
5. Выводы по работе
6. Использованные источники (2-4 источника)

1.6 Пример выполнения работы[1]

Структурная модель информационной системы «Интернет-магазин»

Контекстная диаграмма потоков данных

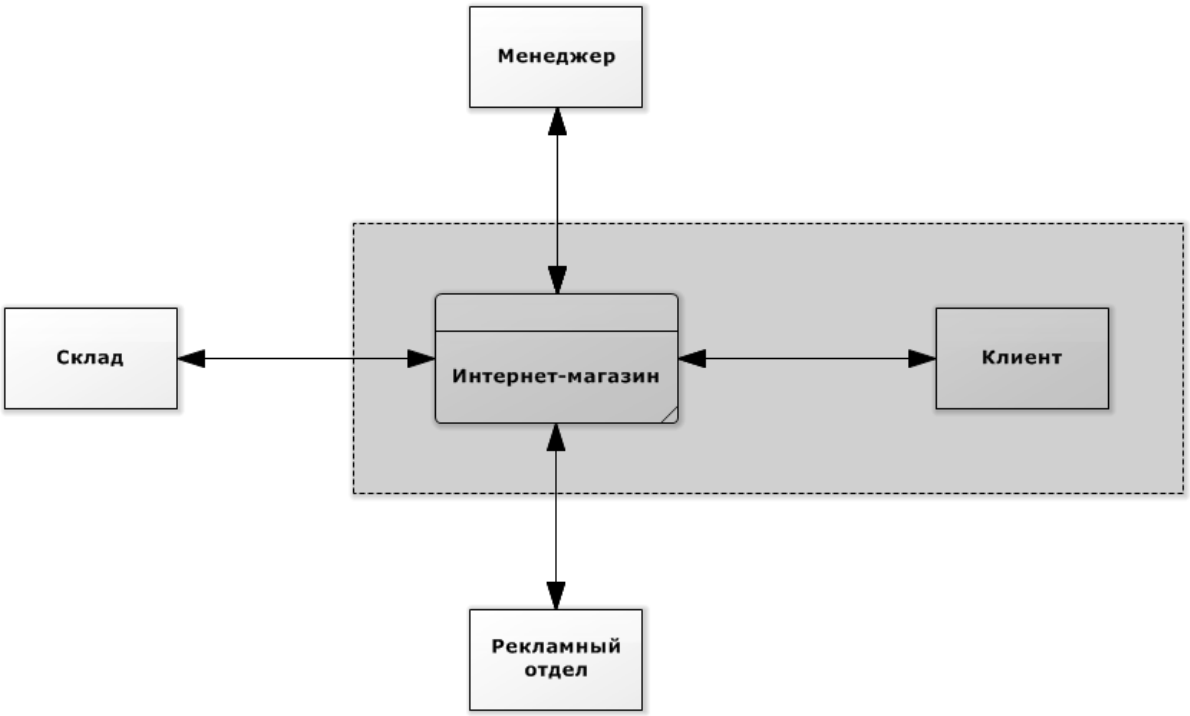


Рисунок 1.6.1 — Контекстная диаграмма системы «Интернет-магазин»

На рис. 1.6.1 серым контуром обозначен фрагмент контекстной диаграммы потоков данных, декомпозиция которого будет осуществлена. На данной контекстной диаграмме не указаны наименования потоков данных, это сделано для демонстрации общих отношений между системой и сущностями, но в целом для большей информативности контекстной диаграммы желательно выделить наименования основных потоков данных, как это сделано на рис. 1.6.2, кроме того, такой вид диаграммы используется при составлении спецификации программных требований, однако первый вариант диаграммы также допустим при выполнении ЛР №1.

1 Данный пример содержит текст пояснений и примечаний к работе, которые при подготовке отчета, должны быть опущены, однако отчет может содержать авторские замечания. В примере приведены два возможных варианта контекстной диаграммы, рекомендуется воспользоваться вариантом, представленным на рис. 1.6.2.

Рисунок 1.6.2 — Контекстная диаграмма системы «Интернет-магазин» с указанием наименований основных потоков данных

Если на контекстной диаграмме фигурирует одна система, то номер системы можно не указывать.

Диаграмма потоков данных 1-го уровня

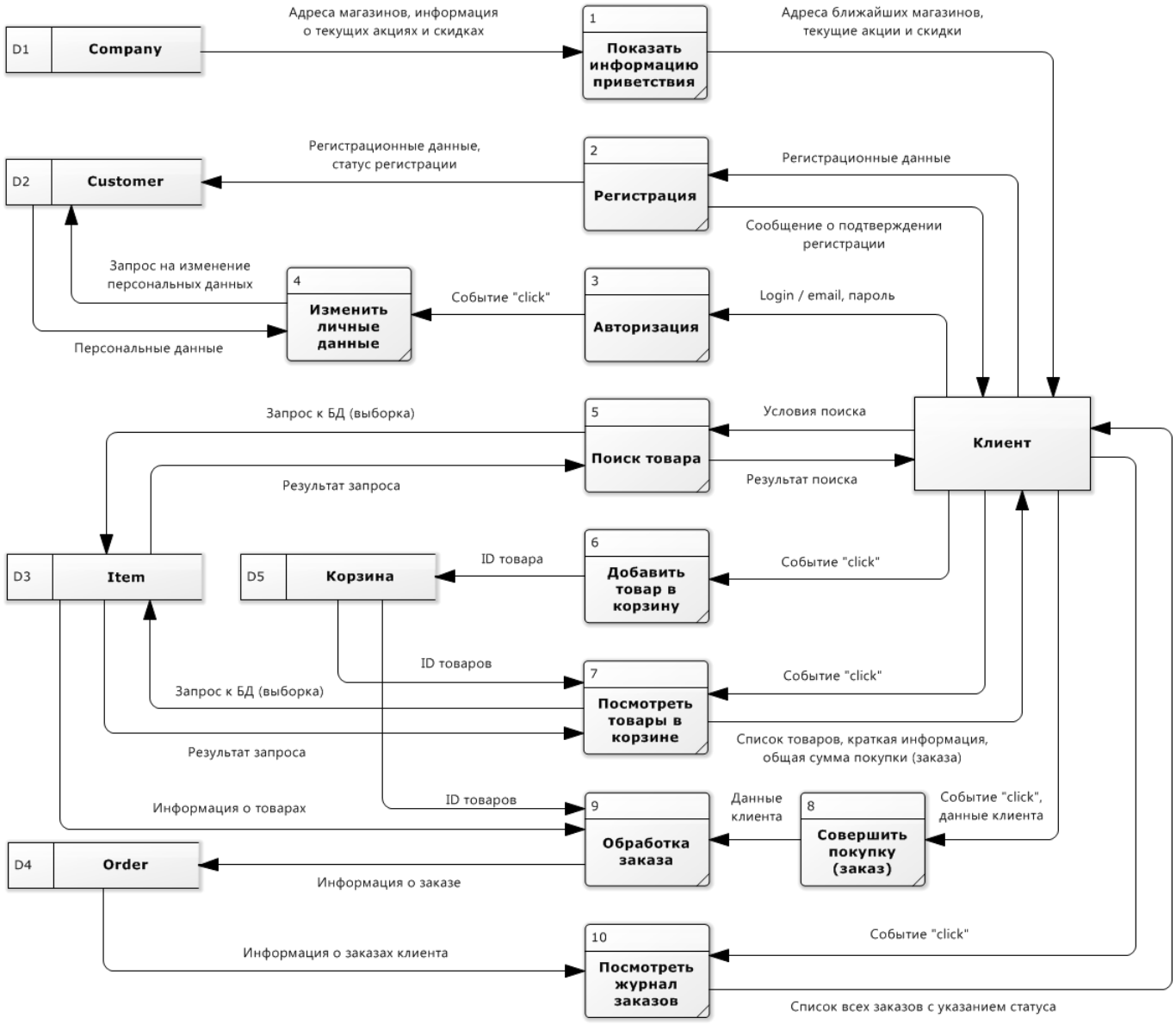


Рисунок 1.6.3 — Диаграмма потоков данных 1-го уровня
для информационной системы «Интернет-магазин»

В Software Ideas Modeler:

1. Изменение типа линии: Левой кнопкой мыши (ЛКМ) на поток → Line Style → Rectangular;

2. Изменение направления потока: ЛКМ на поток: Relationship → Reverse Relationship;

3. Экспорт диаграммы в изображение: Diagram → Export → Image.

С примерами диаграмм потоков данных [4, 5] также можно ознакомиться через СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.

Диаграмма потоков данных 2-го уровня

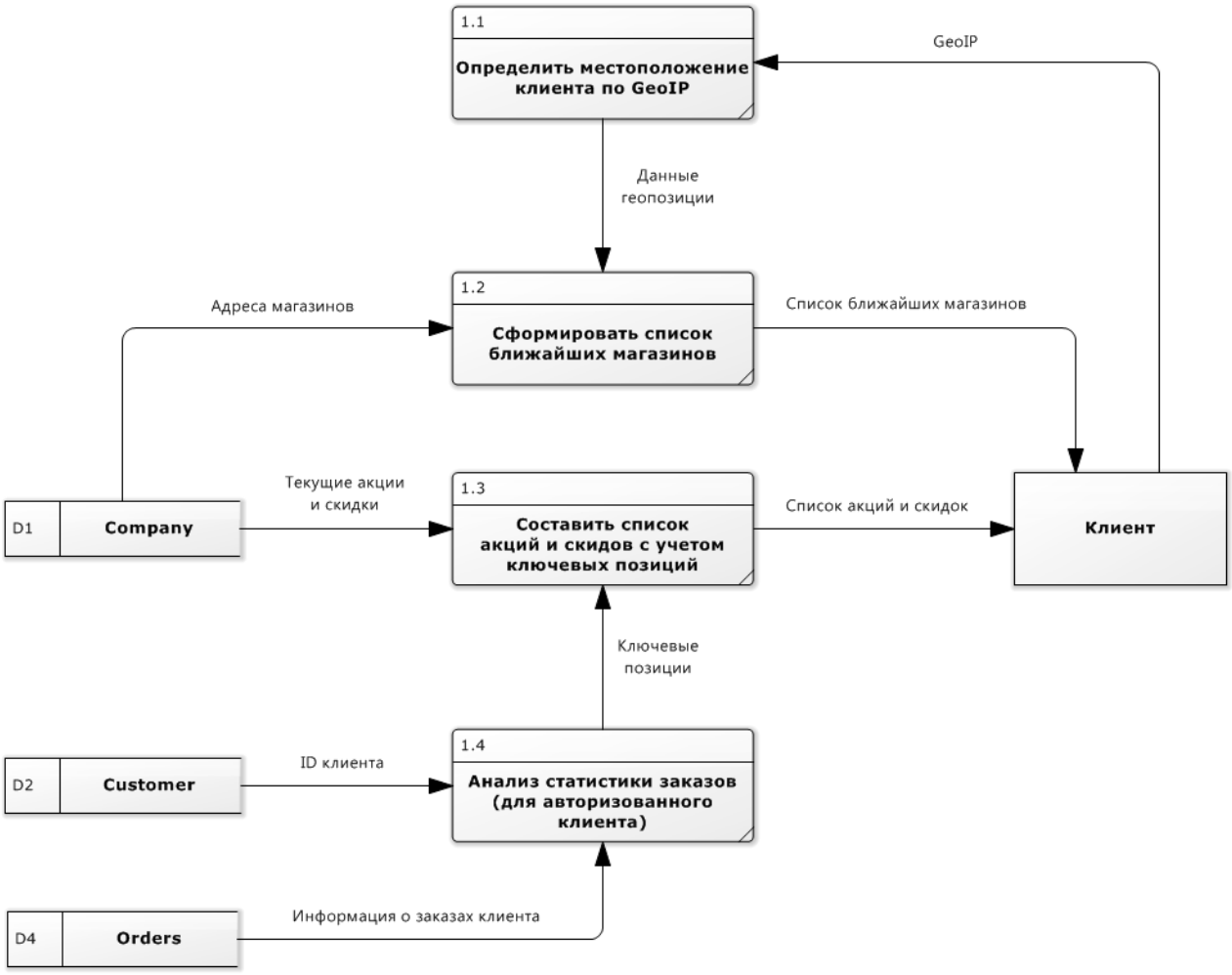


Рисунок 1.6.4 — Декомпозиция процесса №1 «Показать информацию приветствия»

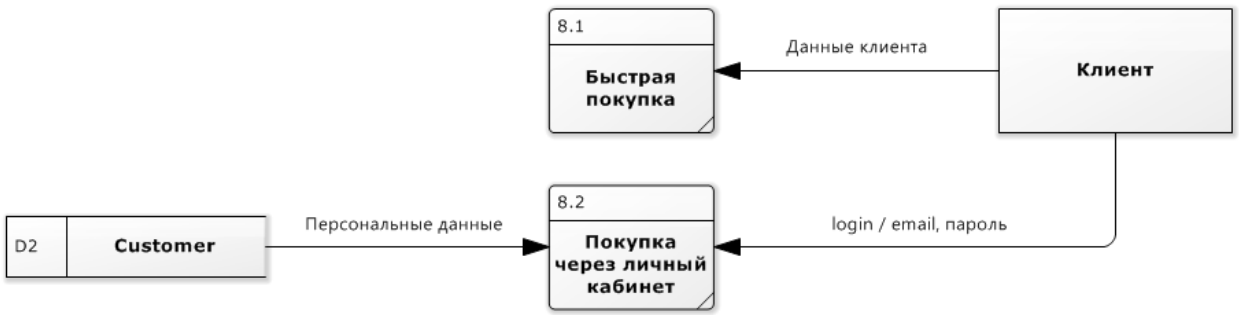


Рисунок 1.6.5 — Декомпозиция процесса №8 «Совершить покупку (заказ)»

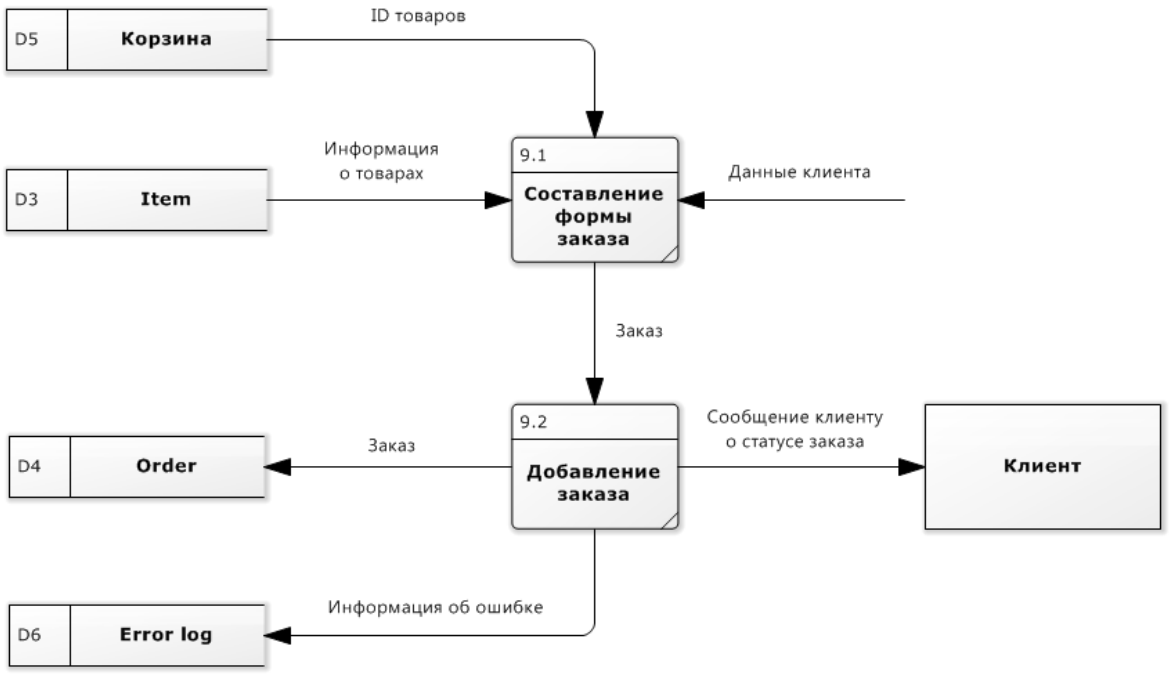


Рисунок 1.6.6 — Декомпозиция процесса №9 «Обработка заказа»

На диаграмме потоков данных 2-го уровня должны быть отражены сущности и накопители данных, с которыми осуществляется взаимодействие исходного процесса на 1-ом уровне иерархии.

Обратите внимание, что некоторые потоки данных не отражены на диаграмме 1-го уровня, например поток «Персональные данные», связывающий накопитель D2 и подпроцесс 8.2 (рис. 1.6.5) или «Сообщение клиенту о статусе заказа» (рис. 1.6.6). Диаграмма потоков данных не ставит перед собой задачу, обозначить совершенно все потоки данных, которые присутствуют в анализируемой или проектируемой системе, кроме того, это ухудшит ее читаемость, поэтому указывают только те потоки, которые существенны на данном уровне иерархии диаграммы и стадии проектирования системы.

При декомпозиции также могут появляться новые хранилища данных (рис. 1.6.6), накопитель D6 на диаграмме 1-го уровня не отражен, так как запись в журнал регистрации ошибок происходит только при возникновении ошибки во время добавления заказа и данный накопитель не является принципиальным для понимания функциональных требований к проектируемой системе.

Выводы по работе

В результате выполнения данной лабораторной работы была изучена методология и один из основных инструментов графического структурного анализа — диаграммы потоков данных.

Разработана структурная модель системы «Интернет-магазин» на основе диаграмм потоков данных. Данная модель не описывает весь функционал анализируемой системы, в модели рассматривается только клиентская часть и не затрагивается административная (панель администратора). Также помимо «менеджера (администратора сайта)» за пределы модели вынесены такие сущности, как «склад товаров» и «рекламный отдел», соответственно все процессы, связанные с данными сущностями, в модели не представлены. Для полной декомпозиции контекстной диаграммы потребуется 4 отдельные, связанные между собой диаграммы потоков данных 1-го уровня, что выходит за рамки задания.

2. ЛАБОРАТОРНАЯ РАБОТА «СЕМАНТИЧЕСКАЯ МОДЕЛЬ ИНФОРМАЦИОННОЙ СИСТЕМЫ. РАЗРАБОТКА ДИАГРАММЫ СУЩНОСТЬ-СВЯЗЬ»

2.1 Цель работы

Целью данной работы является изучение методов концептуального проектирования и построение семантической модели БД системы на основе диаграммы сущность-связь.

2.2 Задание на лабораторную работу

Разработать семантическую модель базы данных информационной системы или ее функционально законченной части с использованием диаграммы сущность-связь.

1. Исходная диаграмма должна ***НЕ ОТВЕЧАТЬ*** требованиям 1, 2 и 3NF;
2. Выполнить нормализацию ER-диаграммы (3NF);
3. Реструктурировать отношения «многие ко многим».

- для оценки «хорошо» количество сущностей на исходной диаграмме должно быть не менее 6-ти;

- для оценки «отлично» количество сущностей на исходной диаграмме должно быть не менее 8-ми.

2.3 Порядок выполнения работы

1. Изучить теоретический материал, изложенный в подразделе 2.4;
2. Проанализировать диаграмму потоков данных из ЛР №1, определить, хранение какой информации будет осуществлено в базе данных системы;
3. Построить исходную ненормализованную ER-диаграмму, содержащую не менее 6-8 сущностей. Данная диаграмма должна включать переменные отношения, не отвечающие 1, 2 и 3NF, и иметь хотя бы один тип отношения «многие ко многим»;
4. Выполнить нормализацию ER-диаграммы (3NF);
5. Реструктурировать отношения «многие ко многим», то есть продемонстрировать, как данные отношения будут представлены в БД системы;
6. Ознакомиться с требованиями по содержанию отчета в подразделе 2.5;
7. Написать отчет о работе.

2.4 Теоретический материал

Наиболее известным представителем класса семантических моделей является модель «сущность-связь» или ER-модель (Entity-Relationship Model, ERM). ER-модель используется при концептуальном проектировании базы данных. Во время проектирования базы данных происходит преобразование ER-модели в конкретную схему базы данных на основе выбранной модели (реляционной, сетевой etc.). Стандартной графической нотацией, с помощью которой можно визуализировать ER-модель, является диаграмма сущность-связь или ER-диаграмма (Entity-Relationship Diagram, ERD).

Следует отличать понятия ER-модель и ER-диаграмма, так как для визуализации ER-моделей существуют и другие графические нотации (например, нотация Питера Чена).

При выполнении лабораторной работы и построении диаграммы сущность-связь рекомендуется использовать нотацию Баркера, как самую распространенную [2]. Элементы и примеры данной нотации будут описаны ниже.

Базовыми понятиями ER-модели являются: сущность, атрибут и связь (отношение).

Сущность — реальный или абстрактный объект предметной области, имеющий атрибуты. Каждая сущность должна:
– иметь уникальное имя;
– обладать одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через связь;
– обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности.

Сущность представляет собой множество экземпляров реальных или абстрактных объектов (людей, событий, состояний, предметов etc.). Имя сущности должно отражать тип или класс объекта, а не его конкретный экземпляр. Например, «Аудитория» будет сущностью, в то время как «24-05» экземпляром данной сущности.

На ER-диаграмме в нотации Баркера сущность изображается прямоугольником, иногда с закругленными углами (рис. 4.1, a).

Атрибут — любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности (рис. 4.1, b). Каждая сущность обладает одним или несколькими атрибутами.

В ER-модели атрибуты ассоциируются с конкретными сущностями и, соответственно, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута. Данное значение будет экземпляром атрибута (определенной характеристикой конкретного экземпляра сущности). Например, если сущность представлена человеком, то одним из атрибутов данной сущности будет его имя, а конкретный экземпляр атрибута, например, имя Евгений соответственно будет характеристикой сущности.

Атрибуты делятся на ключевые, являющиеся частью уникального идентификатора, который называется первичным ключом, и описательные — прочие атрибуты.

Первичный ключ — это атрибут или совокупность атрибутов и/или связей, предназначенная для уникальной идентификации каждого экземпляра сущности. В случае совокупности атрибутов говорят о составном первичном ключе, примером такого ключа может служит серия и номер паспорта. Номер паспорта, как и серия могут повторяться, но вместе они образуют уникальный идентификатор, который однозначно идентифицирует экземпляр сущности, в качестве которой в данном случае выступает человек. Ключевые атрибуты помещают в начало списка и помечают символом «#» (рис. 4.1, c). В Software Ideas Modeler для ключевых атрибутов зарезервирован символ «+», а «#» — для внешних ключей. В настоящих методических указаниях для первичного ключа используется «#». Понятие внешнего ключа выходит за рамки данного курса, поэтому здесь оно затрагиваться не будет.

a) b) c)

Рисунок 2.1 — Обозначение сущности в нотации Баркера
a) — без атрибутов; b) — с указанием атрибутов; c) — с уточнением атрибутов и их типов;

В свою очередь описательные атрибуты бывают обязательными или необязательными. Обязательные атрибуты для каждой сущности всегда имеют конкретное значение, необязательные — могут быть не определены. Обязательные описательные атрибуты помечают символом «*». В Software Ideas Modeler используется ключевое слово «NOT NULL» для обязательных атрибутов и соответственно «NULL» для необязательных, значения которых могут быть не заданы (например, человек может не указать свой пол).

Связь — поименованная ассоциация между двумя или более сущностями, значимая для рассматриваемой предметной области. Связь, таким образом, означает, что каждый экземпляр одной сущности ассоциирован с произвольным (в том числе и нулевым) количеством экземпляров второй сущности и наоборот. Если любой экземпляр одной сущности связан хотя бы с одним экземпляром другой сущности, то связь является обязательной (рис. 4.2, a). Необязательная связь представляет собой условное отношение между сущностями (рис. 4.2, b).

a) b)

Рисунок 2.2— Обозначение связи в нотации Баркера
a) — обязательная связь b) — необязательная

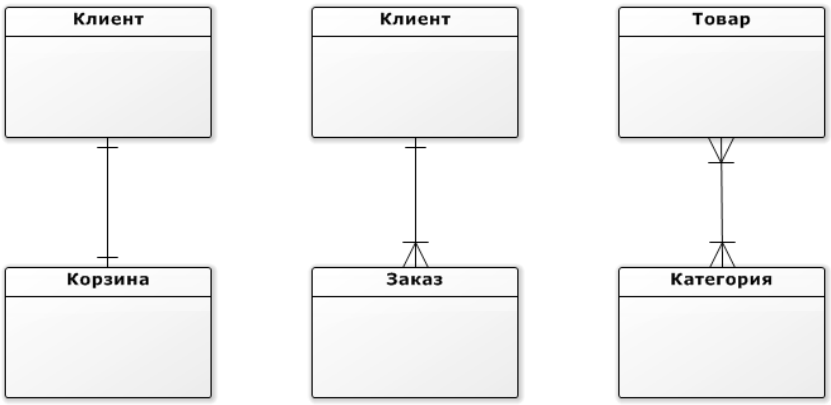
Связь предполагает некоторое отношение сущностей, которое характеризуется количеством экземпляров сущности, участвующих в связи с каждой стороны.

Различают три типа отношений (рис. 4.3):

1:1 — «один к одному» — одному экземпляру первой сущности соответствует один экземпляр второй;

1:M — «один ко многим» — одному экземпляру первой сущности соответствуют несколько экземпляров второй;

M:M — «многие ко многим» — каждому экземпляру первой сущности может соответствовать несколько экземпляров второй и, наоборот, каждому экземпляру второй сущности может соответствовать несколько экземпляров первой.



a) b) c)

Рисунок 2.3 — Обозначение отношений в нотации Баркера
a) — «один к одному» b) — «один ко многим» b) — «многие ко многим»

Отношение «многие ко многим» в базе данных реализуется с помощью трех таблиц. Две таблицы – исходных сущностей (рис. 4.4) и одна связующая таблица. Данное представление необходимо в связи с тем, что при отношении «многие ко многим» между двумя сущностями возникает неоднозначность, какие экземпляры одной сущности относятся к экземплярам другой (например, какой товар был произведен каким изготовителем, и наоборот).

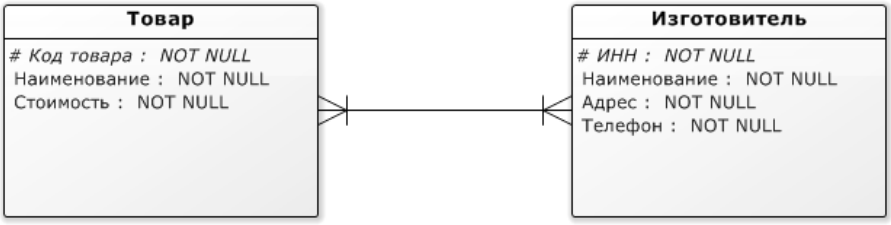


Рисунок 2.4 — Отношение «многие ко многим»

Первичный ключ связующей таблицы всегда будет составным, то есть состоять из двух внешних ключей, которые ссылаются на первичные ключи исходных таблиц (рис. 4.5).

Таким образом, при добавлении связующей таблицы, отношение «многие ко многим» будет состоять из двух отношений «один ко многим». Связующая таблица должна находиться на стороне «многих» обоих отношений [8]. Имя для связующей таблицы, как правило, состоит из имен исходных таблиц, в данном случае ТоварИзготовитель (ItemMaker).

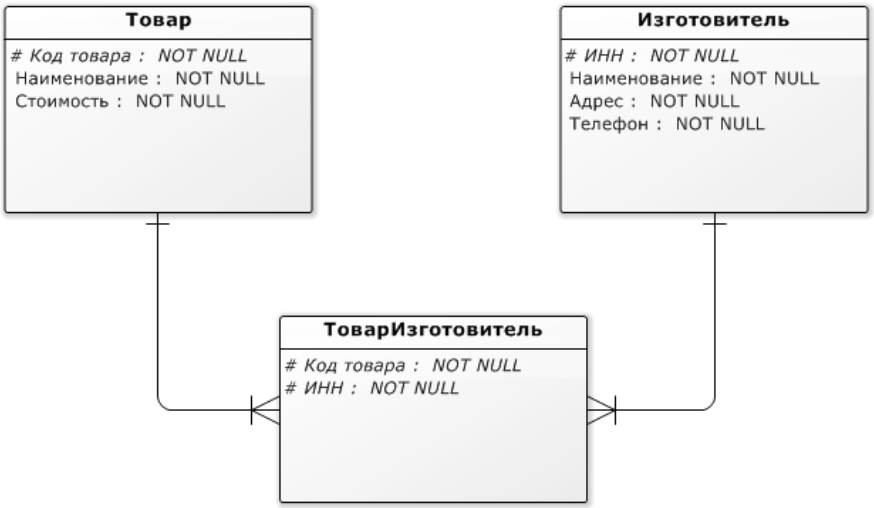


Рисунок 2.5 — Представление отношения «многие ко многим» в базе данных

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в базе данных информации [9].

Общее назначение процесса нормализации заключается в исключении некоторых типов избыточности, устранении определенных аномалий обновления и разработке проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

Первая нормальная форма (1NF).

Переменная отношения находится в первой нормальной форме тогда и только тогда, когда в любом допустимом значении отношения каждая строка содержит только одно значение для каждого из атрибутов [9].

В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия отношение.

Однако сами таблицы могут не быть правильными представлениями отношений и, соответственно, могут не находиться в 1NF.

Например, исходная ненормализованная таблица в БД:

id	customer	email
1	Архангельская Е.А.	archangel95@mail.net
2	Павлов Е.В.	evpavlov@mail.net forspamfromguap@mail.net

Таблица в БД, приведённая к 1NF, будет выглядеть следующим образом:

id	customer	email
1	Архангельская Е.А.	archangel95@mail.net
2	Павлов Е.В.	evpavlov@mail.net
3	Павлов Е.В.	forspamfromguap@mail.net

При реализации БД стараются избегать составных и многозначных атрибутов, чтобы не усложнять написание кода, не перегружать его структуру и не запутывать пользователей. Из этих соображений логически и вытекает определение первой нормальной формы.

Если у нас есть повторяющиеся атрибуты (рис. 4.6), такая сущность также не будет отвечать требованиям 1NF.

Рисунок 2.6 — Сущность с повторяющимися атрибутами

В данном случае повторяющие атрибуты выделяют в отдельную сущность:

Рисунок 2.7 — Приведение отношения к 1NF

Вторая нормальная форма (2NF).

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо зависит от её потенциального ключа [10].

Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость. Для неприводимой функциональной зависимости часто используется эквивалентное понятие «полная функциональная зависимость».

Если потенциальный ключ является простым, то есть состоит из единственного атрибута, то любая функциональная зависимость от него является неприводимой (полной). Если потенциальный ключ является составным, то согласно определению второй нормальной формы в отношении не должно быть неключевых атрибутов, зависящих от части составного потенциального ключа.

На рис. 4.8 представлен пример неполной функциональной зависимости неключевых атрибутов от составного ключа. Наименование и стоимость зависят от кода товара, но адрес изготовителя и телефон зависят только от изготовителя, что очевидно нарушает 2NF.

Рисунок 2.8 — Пример отношения не отвечающего 2NF

Для приведения ко 2NF необходимо осуществить декомпозицию на два отношения:

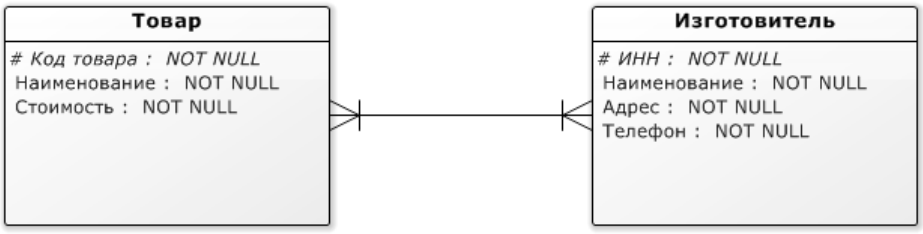


Рисунок 2.9 — Приведение отношения ко 2NF

Третья нормальная форма (3NF).

Отношение находится в 3NF тогда и только тогда, когда выполняются следующие условия:

- отношение находится во второй нормальной форме.
- ни один неключевой атрибут отношения не находится в транзитивной функциональной зависимости от потенциального ключа отношения.

Под транзитивной зависимостью понимают отношение типа: A → B (A определяет B), B → C, следовательно A → C (зависимость C от A является транзитивной).

Иными словами, 3NF можно сформулировать так: каждый атрибут должен предоставлять информацию о ключе, полном ключе и ни о чём, кроме ключа [9]. Данная формулировка проще и достаточно точно отражает условия 3NF.

Например, у нас есть отношение, представленное на рис. 4.10. Данное отношение находится во 2NF так как оно имеет простой первичный ключ (состоящий из одного атрибута). Время, статус и оператор в данном случае определяются кодом заказа, в то время, как телефон оператора полностью зависит от самого оператора. Таким образом, код заказа → оператор, оператор → телефон оператора, соответственно зависимость код заказа → телефон оператора является транзитивной, следовательно, отношение не находится в третьей нормальной форме.

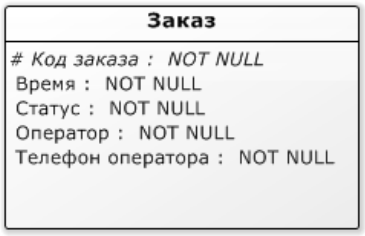


Рисунок 2.10 — Пример транзитивной зависимости атрибутов

Для приведения к 3NF необходимо осуществить декомпозицию на два отношения:

Рисунок 2.11 — Приведение отношения к 3NF

Иногда правила нормализации специально нарушают (в этом случае говорят о денормализации), это связано с двумя основными причинами — удобство и быстродействие. Меньшим число таблиц проще управлять, чем большим. Кроме того, из-за более сложного характера, нормализованные таблицы более медленные для обновления, изменения и выборки данных. С другой стороны, существует достаточно способов для улучшения производительность базы данных, но не так много способов, чтобы исправить повреждённые данные, возникшие из-за некорректно спроектированной структуры БД, именно поэтому нормализация важна с точки зрения надежности и стабильности работы БД.

Зачастую при проектировании БД вместо естественных первичных ключей (например, код заказа или серия и номер паспорта) используется суррогатный ключ — это

дополнительное служебное поле, добавленное к уже имеющимся информационным полям таблицы, единственное предназначение которого — служить первичным ключом. Значение суррогатного ключа не образуется на основе каких-либо других данных из БД, а генерируется искусственно и представляет собой числовое поле, в которое заносятся значения из возрастающей числовой последовательности. Типичным примером суррогатного ключа является автоинкрементный ключ (ID).

Суррогатный ключ используют по следующим причинам [11]:

- Неизменность. Главное достоинство суррогатного ключа состоит в том, что он практически никогда не меняется, поскольку не несёт никакой информации из предметной области и, следовательно, в минимальной степени зависит от изменений, происходящих в ней. Атрибуты естественного ключа время от времени могут меняться — например, человек может изменить имя или фамилию, получить новый паспорт взамен потерянного. В этом случае возникает необходимость так называемых «каскадных изменений» — при изменении значения естественного ключа для сохранения ссылочной целостности система должна внести соответствующие изменения во все значения внешних ключей, ссылающихся на изменяемый ключ. В больших базах данных это может приводить к существенным накладным расходам;
- Гарантированная уникальность. Далеко не всегда можно гарантировать то, что уникальность естественного ключа не будет скомпрометирована с течением времени. Различные внешние факторы могут приводить к тому, что естественный ключ, ранее уникальный, в новых обстоятельствах может утратить уникальность. Суррогатный ключ свободен от этого недостатка;
- Гибкость. Поскольку суррогатный ключ неинформативен, его можно свободно заменять. Допустим, сливаются две фирмы со сходной структурой БД; сотрудник идентифицируется сетевым логином. Чтобы в полученной БД ключ оставался уникальным, приходится добавлять в него дополнительное поле — «из какой фирмы пришёл». В случае с суррогатными ключами достаточно выдать сотрудникам одной из фирм новые ключи.
- Эффективность. Ссылки удобнее хранить в виде целых чисел, чем в виде громоздких естественных ключей (это следует из примера выше). К тому же многие запросы к БД будут компактнее и быстрее, чем при использовании естественных ключей.

Однако у суррогатного ключа есть и свои недостатки:

- Уязвимости генераторов ключей. Например, по номерам ключей можно узнать, сколько записей появилось в БД за некоторый период;
- Неинформативность. Усложняется ручная проверка БД, в запросах появляются INNER JOIN там, где без них можно обойтись. По этой причине в полях перечисляемого типа часто используют ключи в виде коротких строк.
- Склоняет администратора пропустить нормализацию. Добавить суррогатные ключи проще, чем правильно, с учётом дублирования и соотношений «один ко многим» разбить БД на таблицы и проставить уникальные индексы;
- Вопросы оптимизации. СУБД приходится поддерживать два индекса, суррогатный и естественный. Как сказано выше, могут появляться INNER JOIN там, где они не нужны;
- Невольная привязка разработчика к поведению генератора ключей в конкретной СУБД. Например, разработчик может предполагать, что сообщение с меньшим ключом появилось раньше, что в некоторых обстоятельствах может быть не так.

При всей привлекательности суррогатных ключей, лабораторную работу необходимо выполнить с упором на естественные ключи, чтобы не нарушать семантические связи между отношениями и таким образом произвести нормализацию проектируемой структуры БД.

2.5 Содержание отчета

Отчет о работе должен содержать:

1. Титульный лист
2. Цель работы
3. Задание на лабораторную работу и вариант
4. ER-модель БД информационной системы «Наименование»
   4.1 Исходная ненормализованная ER-диаграмма
   4.2 Приведенная к 3NF исходная ER-диаграмма
   4.3 ER-диаграмма с реструктурированными отношениями «многие ко многим»
5. Выводы по работе
6. Использованные источники (2-4 источника)

2.6 Пример выполнения работы

ER-модель БД информационной системы «Интернет-магазин»

Исходная ненормализованная ER-диаграмма

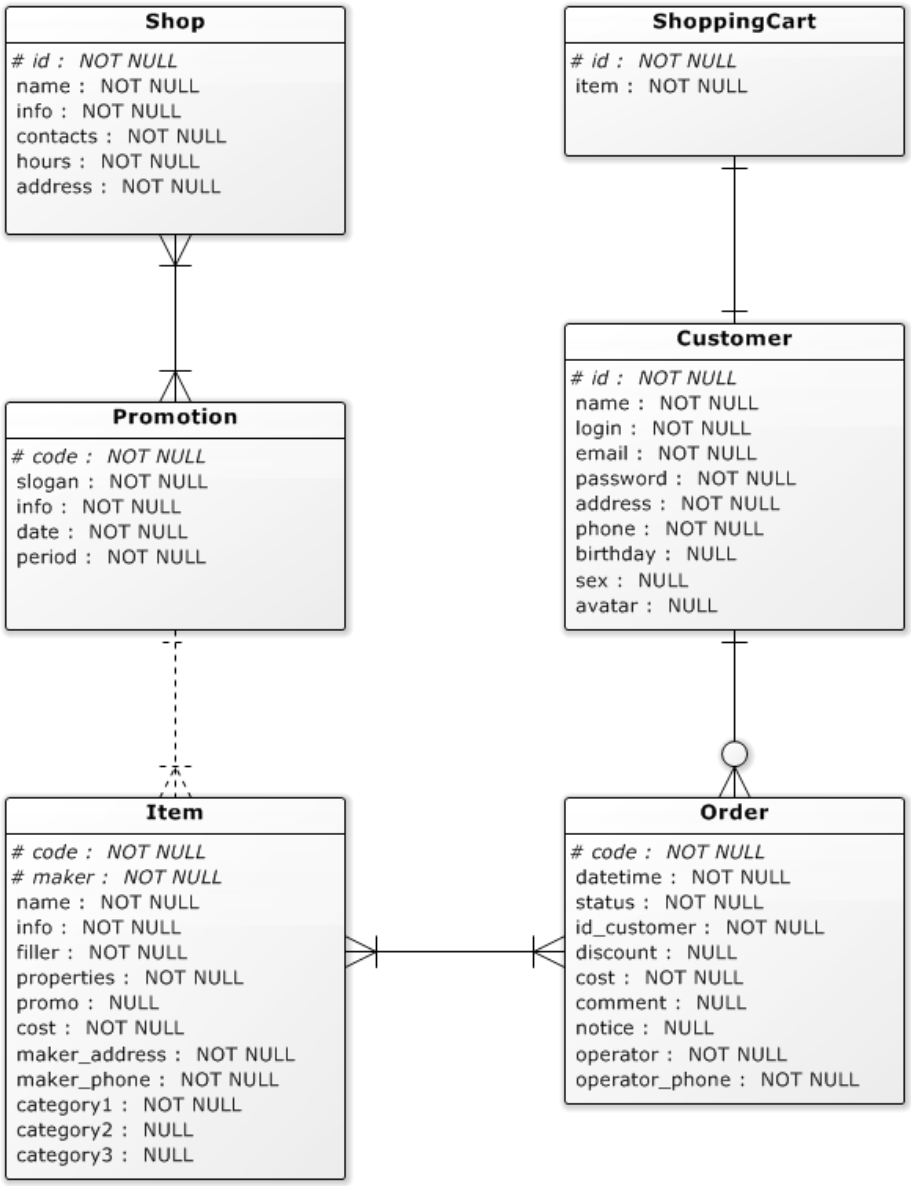


Рисунок 2.6.1 — Ненормализованная ER-диаграмма

На исходной ER-диаграмме сущность Товар (Item) не отвечает требованиям 1NF, так как содержит повторяющийся атрибут category (подразумевается, что товар может относится к различным категориям и, кроме того, это упрощает поиск товара для клиента). Также данная сущность, если привести ее к 1NF, не будет отвечать требованиям 2NF, так как в ней есть подмножество атрибутов (адрес и телефон изготовителя), которые зависят только от части составного ключа, а именно от изготовителя (maker).

Сущность Заказ (Order) не отвечает требованиями 3NF, так как содержит атрибуты, которые функционально зависимы от неключевых атрибутов. Телефон оператора (operator_phone) зависит только от оператора (operator), но не от кода заказа.

Приведение исходной ER-диаграммы к 3NF

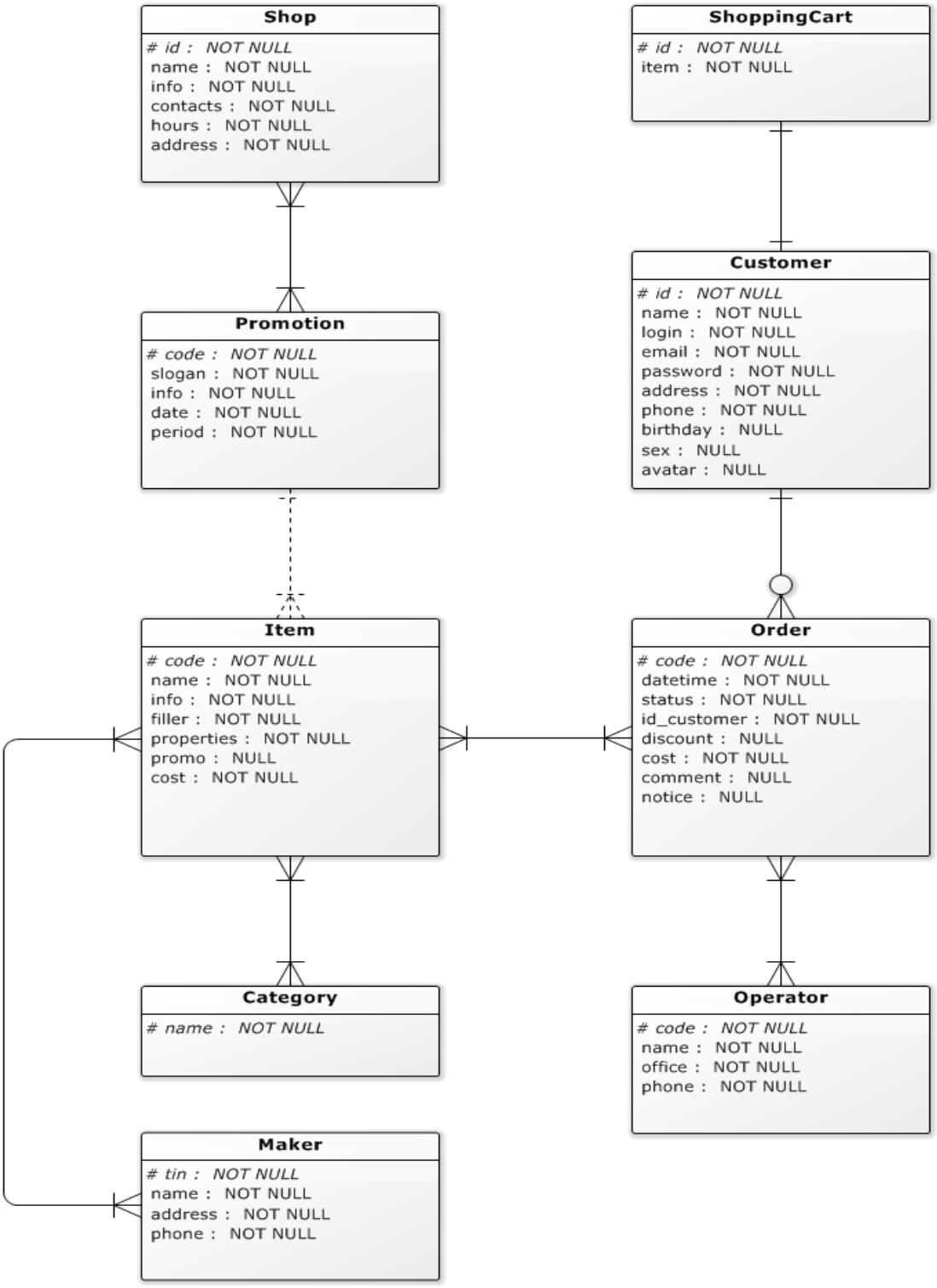


Рисунок 2.6.2 — Нормализованная ER-диаграмма

Реструктурирование отношений «многие ко многим»

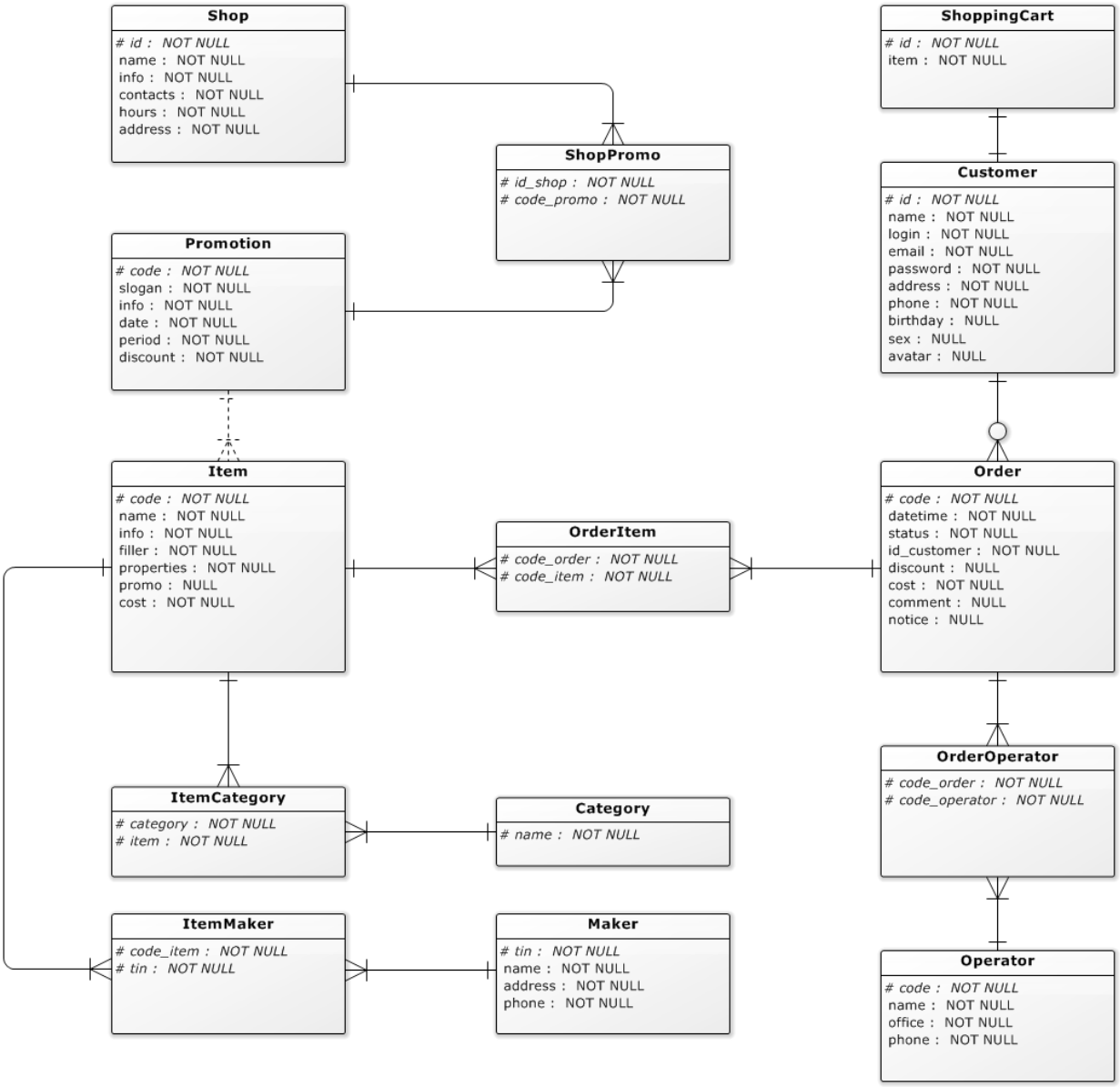


Рисунок 2.6.3 — Представление отношений «многие ко многим» в БД

Выводы по работе

В результате выполнения данной работы изучены методы построения семантической модели БД системы, разработана структура БД в виде ER-диаграммы и выполнена её нормализация.

В данной работе нормализация исходной диаграммы осуществлена по представленным правилам нормализации БД, однако приведение к нормальным формам можно выполнять различными способами, так как проектирование БД допускает личные предпочтения и интерпретации. Полученная структура БД, автору представляется избыточной и вне рамок сформулированного задания, имела бы более простую структуру и при этом отвечала главному требованию — отсутствию явных нарушений нормальных форм.