1. ЛАБОРАТОРНАЯ РАБОТА

«ОПРЕДЕЛЕНИЕ СТРУКТУР И ЭЛЕМЕНТОВ ДАННЫХ ПРЕДМЕТНОЙ ОБЛАСТИ. СОСТАВЛЕНИЕ СЛОВАРЯ ДАННЫХ»

1.1 Цель работы

Целью настоящей работы является изучение способов описания информации об используемых в системе сущностях данных и получение практических навыков составления словаря данных.

1.2 Задание на работу

Для заданной предметной области[1] необходимо составить даталогическую схему реляционной модели данных (на основе ER-диаграммы):

1) Допускается описание *фрагмента* даталогической модели:
   - не менее 6 ключевых сущностей (объектов предметной области);
   - суммарно не менее 24 описательных (неключевых) атрибутов.
2) Набор атрибутов для каждой сущности должен быть задан в соответствии с требованиями как предметной области, так и задач, для выполнения которых предназначена система;
3) На ER-диаграмме должны быть в явном виде указаны первичные (PK) и внешние ключи (FK);
4) ER-диаграмма должна содержать одно или несколько отношений «многие ко многим» (без реструктуризации);
5) Отдельно должна быть представлена ER-диаграмма с реструктуризацией всех отношений «многие ко многим» на два отношения «один ко многим» (связующие таблицы также могут содержать дополнительные атрибуты, если в этом есть необходимость с точки зрения задач системы);
6) Представленная на ER-диаграмме информация, должна находиться в 3NF.

Определить элементы выделенных сущностей в словаре данных:

7) Словарь данных должен отражать минимальные критерии для проверки элементов данных (атрибуты сущностей) с точки зрения их отображения, хранения и выполняемых над ними операций; таким образом:
8) В словаре данных необходимо указать точную или предполагаемую *длину элементов в символах* (не в байтах) или диапазон числовых значений и соответствующий заданной длине тип данных. Если длина элемента неизвестна, то ограничение длины должно быть задано соответствующим типом данных. При этом рекомендуется использовать тип данных наименьшего размера, который гарантирует хранение всех возможных значений. Принятые в работе типы данных должны быть раскрыты в приложении к отчету;
9) Также требуется указать список разрешенных значений или значений по умолчанию, если это необходимо. В случае неочевидного использования атрибутов в системе, каждый такой атрибут должен быть сопровожден соответствующим пояснением.

[1] Предметная область (индивидуальный вариант задания) закреплена за студентом с прошлого семестра

1.3 Рекомендации и требования по содержанию отчета

При защите работы замечания преподавателя по составу атрибутов сущностей данных имеют приоритет перед требованиями задания.

Если отчет по работе загружен в личный кабинет и принят преподавателем, то дальнейшая пересдача лабораторной работы повлечет двукратное увеличение объема задания — таким образом, для ЛР 1 потребуется:

- *не менее 12 ключевых сущностей;*
- *суммарно не менее 48 описательных (неключевых) атрибутов.*

Для выполнения лабораторной работы можно воспользоваться любой средой моделирования или CASE-средством, которые поддерживают графическую нотацию диаграммы «сущность-связь».

Рекомендуемый бесплатный онлайн-сервис для построения диаграмм:
https://www.diagrams.net/ (присутствует десктопная версия)

Шаблон отчета, который также включает в себя и пример выполнения работы, прилагается к заданию.

Исходная ER-диаграмма представлена в отдельном файле: *lab1_ERD.xml* (можно открыть с помощью *diagrams.net*)

Элементы изображений в отчете должны быть визуально различимы, поэтому рекомендуемый масштаб при экспорте (для *diagrams.net*) — 150-200%

Обратите внимание, что фактический состав атрибутов сущностей данных зависит не только от того, объектом какой предметной области является сущность, но и задач самой системы. Например, если мы возьмем две системы: «Электронный тематический журнал манги» и «Сервис для чтения манги», то предметные области двух систем пересекаются в ряде сущностей, но вот состав их атрибутов может быть совершенно различным. В электронном журнале манги для сущности *Manga* может быть определен следующий набор атрибутов:

Идентификатор	int
Наименование	varchar (255)
Автор манги	int
Дата выпуска	date
Обложка	varchar (255)
История	varchar (255)
Количество страниц	smallint

В то время как для сервиса чтения манги список атрибутов данной сущности, исходя из задач системы, совершенно другой:

Идентификатор	int
Наименование	varchar (255)
Автор манги	int
Дата выпуска	date
Обложка	varchar (255)
История	varchar (255)
Количество страниц	smallint
Размер файла	int
Рейтинг просмотров	int

Оценка читателей	decimal (1,2)
Издательство	varchar (255)
Стоимость	decimal (18,2)
Возрастной рейтинг	tinyint

Отчет к лабораторной работе должен содержать:

- Титульный лист
- ВВЕДЕНИЕ (короткое описание актуальности работы, формулировка цели и постановка задачи, индивидуальный вариант задания)
- 1. Даталогическая модель базы данных
- 2. Словарь данных
- ЗАКЛЮЧЕНИЕ (выводы по работе)
- СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ
- ПРИЛОЖЕНИЕ А (принятые в работе типы данных)

Оформление отчета должно соответствовать требованиям, представленным в правилах и критериях оценки по дисциплине.

1.4 Теоретический материал

Для демонстрации процессов, которые происходят в системе, используют DFD, в свою очередь для представления данных и отношений между ними применяют модели данных. Одна из наиболее широко используемых моделей данных — диаграмма «сущность-связь» (*Entity-Relationship Diagram*, далее ERD), которую также используют в качестве инструмента для анализа требований. Основное назначение ERD заключается в анализе компонентов данных системы и их связей для определения логической или физической (реализации) структуры базы данных. Все элементы данных, которые представлены на ERD, подробно описывает словарь данных (*data dictionary*), предназначенный для сбора, организации (систематизации) и документирования конкретных фактов о системе.

1.4.1 Моделирование отношений данных

Базовыми понятиями ERD являются: сущность, атрибут и связь (отношение).

Сущность (entity) представляет собой реальный или абстрактный объект предметной области. Таким образом, это могут быть как физические элементы (включая людей), так и агрегации элементов данных, которые важных с точки зрения анализируемой системы. На ERD сущности показаны в виде прямоугольников и именуются посредством существительных в единственном числе.

Атрибут — любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Каждая сущность обладает одним или несколькими атрибутами. Атрибуты делятся на ключевые (*author_id* на рис. 1.1), являющиеся частью уникального идентификатора сущности, который называется первичным ключом, и описательные — прочие атрибуты. В словаре данных приводятся детальные определения атрибутов сущности, таким образом это выступает гарантией того, что объекты на ERD и соответствующие им хранилища данных на DFD определены одинаково. В процессе проектирования физической реляционной базы данных сущности, как правило, становятся таблицами, атрибуты сущностей — столбцами таблиц.

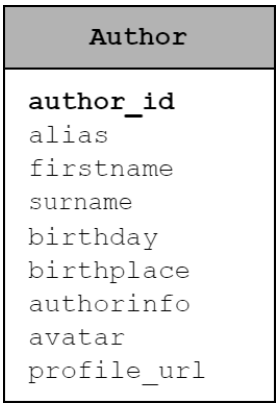


Рисунок 1.1 — Сущность *Author* с перечнем атрибутов

Первичный ключ представляет собой атрибут (или совокупность атрибутов и/или связей), который служит для уникальной идентификации каждого экземпляра сущности. В случае совокупности атрибутов говорят о составном первичном ключе, примером такого ключа для сущности *Magazine* (рис. 1.2) может служит год и номер журнала — если это периодическое издание, то год и номер журнала по отдельности могут повторяться, но вместе они образуют уникальный идентификатор сущности (иными словами, в таблице базы данных не может существовать двух записей, которые имеют одинаковое значение первичного ключа). Обратите внимание, что уникальность — не единственное требование для первичного ключа. Номер и серия

паспорта также образуют уникальное сочетание атрибутов (хотя по отдельности могут повторяться) и однозначно идентифицируют экземпляр сущности (в данном случае конкретного человека). Однако серия и номер паспорта не могут быть первичным ключом, так как подвержены изменениям (человек может заменить паспорт), а первичный ключ не должен меняться с течением времени (в противном случае возникают риски потери свойства уникальности). На ERD ключевые атрибуты помещают в начало списка и помечают символом «#» (в примерах раздела 1.4 первичные ключи выделены полужирным цветом).

Year и *Number* — пример составного естественного первичного ключа. Естественным называется первичный ключ, который содержит в себе некоторую полезную информацию об объекте предметной области. Бóльшая часть естественных ключей подвержена изменениям с течением времени (например, серия и номер паспорта или идентификационный номер налогоплательщика), что затрудняет их использование в качестве первичных ключей, поэтому вместо них, как правило, используют суррогатные первичные ключи. *Суррогатный ключ* — дополнительное служебное поле, которое не имеет никакого отношения к предметной области и выполняет единственную роль — служит первичным ключом. Обычно это генерируемый системой порядковый номер записи в базе данных по типу ID (*author_id* на рис. 1.1). Составные первичные ключи чаще всего встречаются при поддержке устаревших баз данных, поэтому от них рекомендуется отказаться в пользу суррогатных ключей. Также в качестве первичного ключа, как правило, выбирают тот, который имеет наименьший размер (физического хранения) и/или включает в себя наименьшее количество атрибутов. Естественные ключи довольно часто содержат избыточную информацию и это еще одна причина, почему вместо громоздких естественных ключей обычно используют суррогатные, так как ссылки (например, внешние ключи в связующих таблицах) удобнее хранить в виде целых чисел.

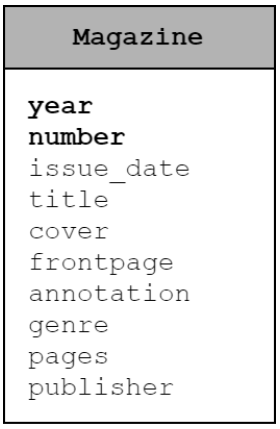


Рисунок 1.2 — Сущность *Magazine* с перечнем атрибутов

Тем не менее, в некоторых ситуациях составные ключи могут быть полезны:

- В связующих таблицах при реализации отношений «многие ко многим»;
- Когда дочерние объекты не имеют смысла без родительских. Например, если преподаватель будет заносить в базу данных ответы студентов по тестовому контролю, то в определенной ситуации (зависит от задачи) удобнее работать с ответами по идентификатору студента и номеру ответа на вопрос, а не как с 347-ым ответом в базе данных;
- Для дополнительной проверки целостности в первичный ключ могут быть добавлены дополнительные поля (этот вопрос выходит за рамки нашего курса, но возможно об этом вы узнаете в курсе баз данных).

Описательные (не ключевые) атрибуты могут быть обязательными или необязательными. Обязательные атрибуты для каждой сущности всегда имеют конкретное значение, необязательные — могут быть не определены (например, для сущности *Author* может быть не задан URL профиля в социальной сети, если у автора он отсутствует как таковой). Как правило, на ERD обязательные описательные атрибуты помечают символом «*» (в примерах раздела 1.4 не используется данное

обозначение). В словаре данных (см. подраздел 1.4.3) наименования необязательных атрибутов заключают в скобки.

Связь (или отношение[2] от relationship) — поименованная ассоциация между двумя или более сущностями, значимая для рассматриваемой предметной области. Иными словами, связь показывает, как сущности (объекты) связаны друг с другом в системе.

Мощность (или кратность) связи показывает, сколько экземпляров одной сущности ассоциировано с произвольным (в том числе и нулевым) количеством экземпляров другой сущности. Кратность обозначают цифрами или буквами на линиях, соединяющих сущности. Для ERD используются различные нотации обозначения кратности: Питера Чена, Джеймса Мартина и диаграммы классов.

В нотации Питера Чена наименования связям (фигура в форме ромба) дают в соответствие с характером соединения. При этом корректность наименования зависит от того, в какую сторону читать диаграмму (или конкретную связь). На рис. 1.3 направления чтения связей показаны стрелками (в самой нотации они отсутствуют). В данном примере, если читать диаграмму слева направо (журнал включает в себя мангу), то наименование связи точно отражает характер соединения, однако если прочитать в обратную сторону (справа налево) — наименование отношения не будет корректным, так как семантика связи подразумевает возможность включения (манга не может включать в себя журнал). Аналогично по отношению к тому, что манга или журнал могут включать в себя дополнения (*omake*) в виде цветных страниц или новых артов.

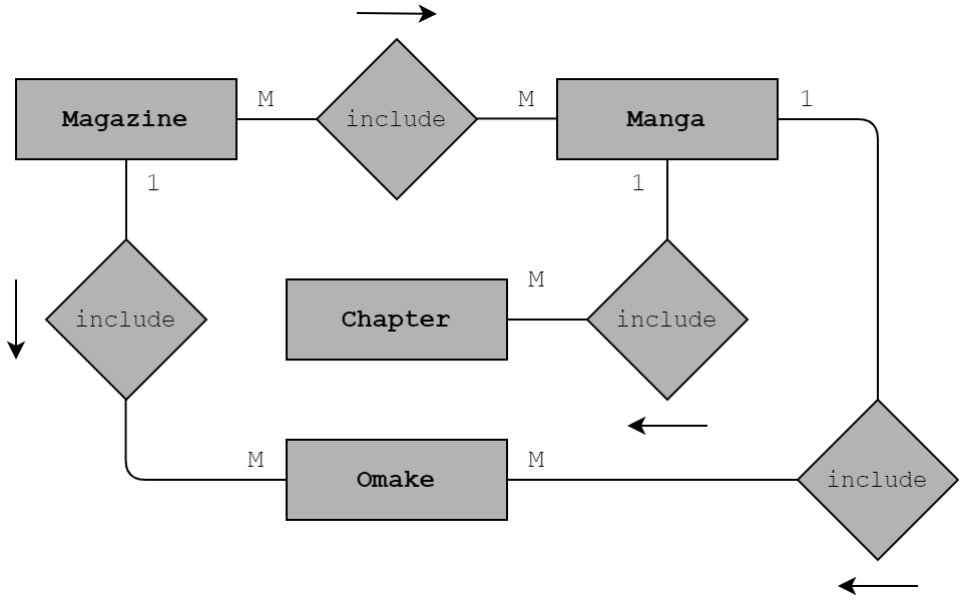


Рисунок 1.3 — Пример нотации Питера Чена

[2] Значения понятий *relationship* (в реляционной *модели данных* — связь) и *relation* (в реляционной *базе данных* — таблица) отличаются (см. подраздел 1.4.2), но в русском языке имеют единообразный перевод — отношение

В нотации Джеймса Мартина (рис. 1.4) кратность связи между сущностями обозначают на соединяющей их линии. Вертикальная черта рядом с сущностью указывает на кратность «1», «воронья лапка» — на кратность «многие». Круг рядом с «лапкой» означает, что сущность может содержать ноль или более экземпляров, аналогично для вертикальной черты с «лапкой» («один или многие») и круга с вертикальной чертой («ноль или один»).

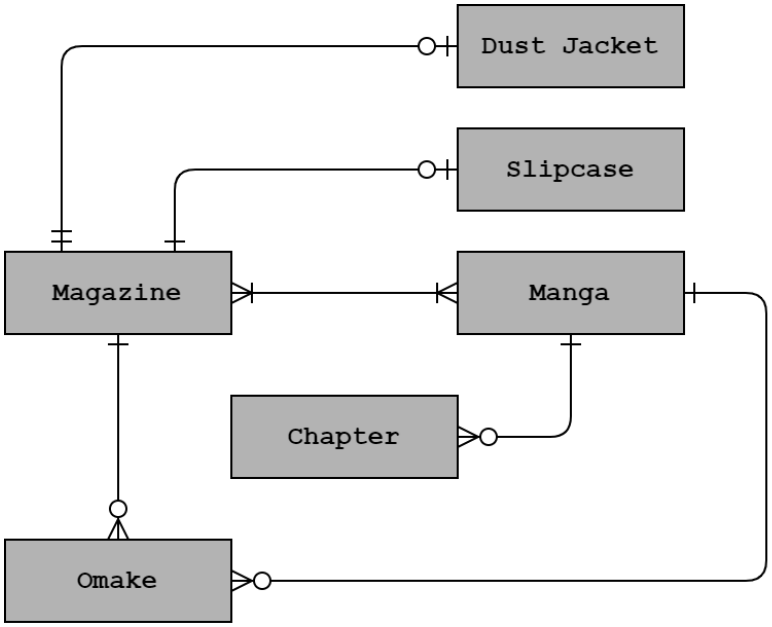


Рисунок 1.4 — Пример нотации Джеймса Мартина

Обратите внимание на следующие фрагмент диаграммы рис. 1.4:

- Журнал может как иметь специальный футляр (*Slipcase*), так и поставляться без него (информация о футляре также хранится в базе данных, так как оформление футляра представляет отдельную ценность) — отношение «один к одному»;
- Журнал или манга могут включать в себя дополнительный бонусный материал (*Omake*) — отношение «один к многим»;
- У журнала манги может быть специальная суперобложка (*Dust Jacket*) — на диаграмме это показано отношением «один и только один» (две вертикальные черты). Отношение «один и только один» графически показывается только в нотации Мартина и подразумевает неизменность ассоциированного экземпляра сущности (здесь подходит аналогия с ассоциацией экземпляров пользователя и логина — у каждого пользователя должен быть один и только один логин, который не может быть изменен).

Если нам известна более точная кратность, чем просто много, то в нотации Мартина (которую также называют Crow's Foot Notation) у нас нет возможности это указать. Поэтому иногда для моделирования отношений данных удобно использовать нотацию диаграммы классов (рис. 1.5), так как данная нотация более лаконична и допускает вместо общего «М» (много) указывать конкретное число или определенный диапазон. Например, в генеалогической системе у человека может быть 2 биологических родителя или в случае процедуры искусственного оплодотворения — 3 (таким образом, кратность в данном примере будет «2 … 3»).

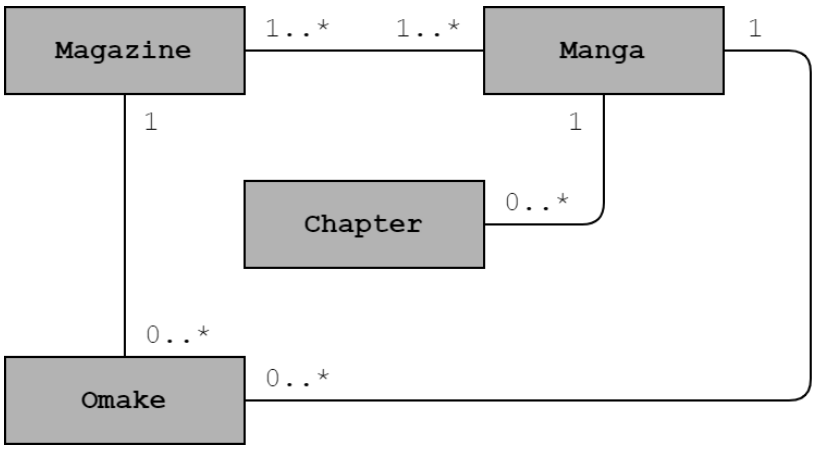


Рисунок 1.5 — Пример нотации диаграммы классов

Как правило, при построении модели базы данных, выделяют три основных типа связей между сущностями:

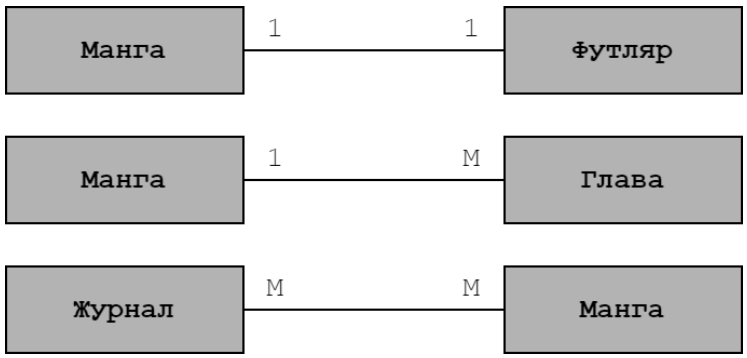


Рисунок 1.6 — Основные типы связей (отношений)

- «Один к одному» — одному экземпляру первой сущности соответствует один экземпляр второй. Данный тип связи используется достаточно редко, так как в этом случае две таблицы могут быть просто объединены в одну и в целом использование отношений «один к одному» не приветствуется. Иногда это отношение может быть необходимо, чтобы преодолеть ограничения реляционной системы управления базами данных (СУБД) или для увеличения производительности (например, можно вынести поле с типом данных BLOB[3] в отдельную таблицу для ускорения поиска по родительской таблице), также возможно изолирование части таблицы из соображений безопасности;
- «Один ко многим» — одному экземпляру первой сущности соответствуют несколько экземпляров второй (контекст связи включает возможность отсутствия экземпляров или наличие только одного из них, например, манга может иметь только одну главу или же не иметь их вовсе, если это ёнкома манга). Отношение «один ко многим» является наиболее частым видом связи между таблицами в базе данных;

[3] Binary Large Object (двоичный большой объект) — специальный тип данных для хранения изображений или скомпилированного программного кода. Изображения обычно хранятся на сервере в виде отдельных файлов, однако изображения небольшого размера по типу аватарок можно хранить и в самой базе данных

- «Многие ко многим» — каждому экземпляру первой сущности может соответствовать несколько экземпляров второй и наоборот. *Как правило, при проектировании базы данных связь «многие ко многим» рассматривают не в контексте существования в данный момент, а возможно ли оно вообще, в перспективе,* чтобы база данных была готова к этому и, таким образом, не возникло необходимости в миграции базы данных (переходу от одной структуры базы данных к другой без потери согласованности данных).

Отношение «многие ко многим» (рис. 1.6) в базе данных реализуется с помощью трех таблиц (рис. 1.7): две таблицы исходных сущностей (в данном случае *Magazine* и *Manga*) и одна связующая таблица (наименование связующей таблицы, как правило, состоит из наименований исходных — *Magazine_Manga*). Данное представление необходимо, так как отношение «многие ко многим» между двумя сущностями порождает неоднозначность — какие именно экземпляры одной сущности относятся к экземплярам другой (например, какая манга включена в какой номер журнала и наоборот).

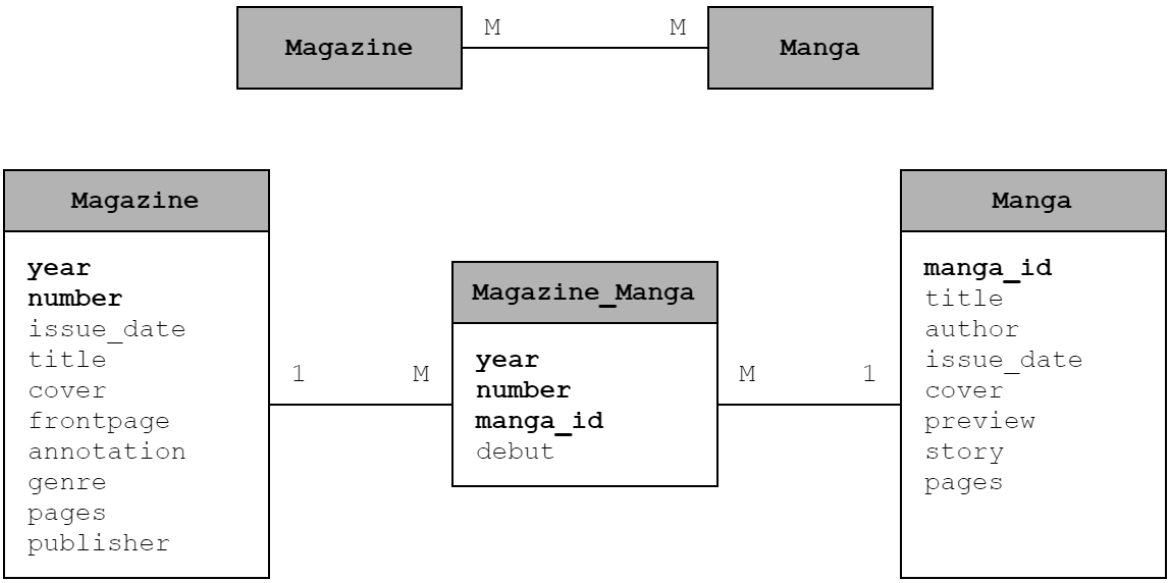


Рисунок 1.7 — Представление отношения
«многие ко многим» в базе данных

Первичный ключ связующей таблицы *всегда будет составным* — он включает в себя *внешние ключи* (это ключи, которые ссылаются на первичные ключи исходных таблиц). В данном примере первичный ключ связующей таблицы состоит из трех внешних ключей, так как первичный ключ одной из исходных таблиц также является составным. 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. В связующую таблицу можно добавлять другие атрибуты, как и в любую другую таблицу (например, в связующей таблице на рис. 1.7 присутствует атрибут *debut*, предназначенный для задания статуса манги, если она публикуется в журнале впервые).

1.4.2 Нормализация отношений

Нормальная форма — свойство отношения в реляционной модели данных[4], характеризующее его с точки зрения избыточности, которая потенциально может привести к логически ошибочным результатам выборки или изменения данных. Таким образом, нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение (рис. 1.8).

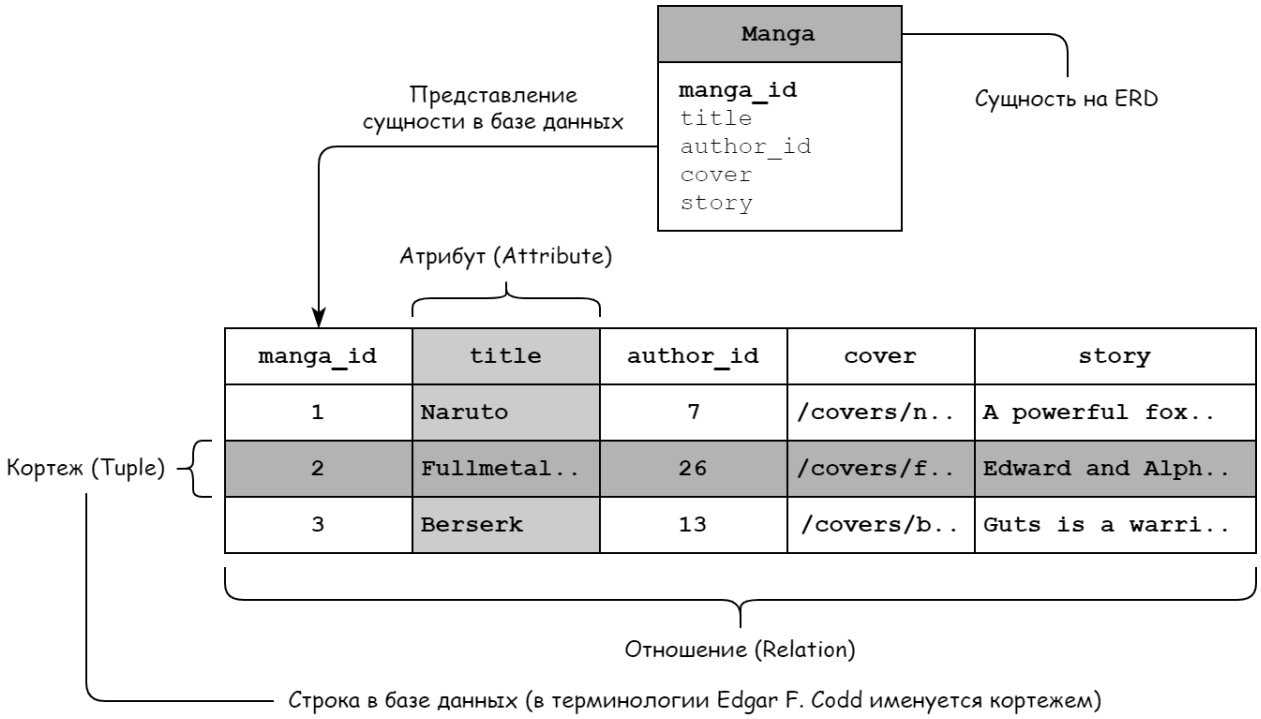


Рисунок 1.8 — Отношение, кортеж и атрибут представленные в виде таблицы, строки и столбца соответственно

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную логическую избыточность. Конечная цель нормализации — исключение некоторых типов избыточности и уменьшение потенциально противоречивой информации, хранений которой осуществляется в базе данных. Уменьшение или увеличение производительности работы, как и уменьшение или увеличение физического объёма базы данных *не являются целями нормализации*.

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). Обратите внимание, что в примере на рис. 1.2 ключевой атрибут *Year* не является частью атрибута *issue_date* (дата издания), так как номер за январь 2022 года может выйти, например, в декабре 2021 — *year* = 2022, *number* = 1, *issue_date* = 20.12.2021 — иным словами, в этом примере нет избыточности данных.

[4] Понятие «реляционный» основано на англ. *relation* («отношение, связь, зависимость»). Ключевым моментом реляционной БД являются как раз связи между таблицами. Таким образом, реляционная модель данных состоит из сущностей (таблиц баз данных) и отношений (связей) между ними

На рис. 1.8 отношение имеет простую графическую интерпретацию в виде таблицы, столбцы которой соответствуют атрибутам, а строки — кортежам, в свою очередь «ячейки» таблицы содержат значения атрибутов в кортежах. Однако в строгой реляционной модели отношение не является таблицей, кортеж — это не строка, атрибут в свою очередь — не столбец. Термины «таблица», «строка», «столбец» могут использоваться только в неформальном контексте, иными словами, используя эти термины, мы понимаем, что они являются всего лишь *приближением* обозначаемых понятий.

Таблица является прямым и верным представлением некоторого отношения, если она удовлетворяет следующим условиям:

1) Нет упорядочивания строк сверху-вниз (то есть порядок строк не несёт в себе никакой информации);
2) Нет упорядочивания столбцов слева-направо (то есть порядок столбцов не несёт в себе никакой информации);
3) Нет повторяющихся строк;
4) Каждое пересечение строки и столбца содержит ровно одно значение из соответствующего домена[5] (и больше ничего);
5) Все столбцы являются обычными. Иными словами, в таблице нет скрытых компонентов или временны́х меток, которые могут быть доступны при вызове некоторых специальных операторов.

- 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.

В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия отношения (см. пять условий выше). Однако сами таблицы могут быть неправильными представлениями отношений, и в этом случае, соответственно, они не находятся в 1NF.

В таблице ниже хранятся имена авторов манги и ссылка на профиль в соцсети. Одно из требований — возможность сохранить несколько ссылок, например, если автор поддерживает два профиля в разных соцсетях. Самое простое и неправильное решение — это хранить их в одной строке для данного атрибута, что нарушает 1NF, так как значение атрибута, согласно четвертому условию, должно быть атомарно[6] (иными словами, его нельзя разбить на два и более значений).

author_id	first_name	surname	profile_url
12	Oda	Eiichiro	https://twitter.com/zorojuro
13	Miura	Kentaro	https://twitter.com/skull_knight https://thegodhand.hell/femto

[5] Домен в реляционной модели данных представляет собой набор всех допустимых значений, которые может содержать атрибут. Например, атрибут *manga_id* принадлежит домену целых чисел, атрибуты *title* или *story* — домену строк. Иными словами, понятие домена аналогично (но не эквивалентно) понятию типа данных.

[6] Если честно, свойство атомарности достаточно неоднозначно, так как оно зависит от того, для каких целей это значение будут использоваться. Например, в данном примере полное имя автора разбито на имя и фамилию, однако в рассматриваемой системе не предполагается операций с каждым из этих атрибутов по-отдельности (то есть с точки зрения задач системы, их объединение также будет считаться атомарным)

Основная проблема данного примера — запрос на выборку всех социальных сетей для *Twitter*, так как в этом случае нам придется добавить дополнительный код (реализация парсера строки), а если структура базы данных изменится, то этот код придется либо полностью удалить, либо переписать.

Следующее неверное решение — ввести больше столбцов:

author_id	first_name	surname	profile_url_1	profile_url_2
12	Oda	Eiichiro	https://twitt..	
13	Miura	Kentaro	https://twitt..	https://thego..

Технически таблица выше не нарушает требование атомарности атрибутов. Однако два столбца для профилей соцсети повторяют то, что концептуально является одним и тем же атрибутом. Кроме того, согласно второму условию, порядок столбцов не должен иметь значения, но тогда встает вопрос — почему мы ссылку на профиль в *Twitter* определили в столбец номер 1, а не в столбец номер 2? И почему автор не может поддерживать N активных профилей?

Чтобы привести таблицу к 1NF необходимо разбить строки профиля соцсети на атомарные объекты (отдельные ссылки на профили соцсетей), таким образом, ни одна из строк не будет содержать более чем одной ссылки:

author_id	first_name	surname	profile_url
12	Oda	Eiichiro	https://twitter.com/zorojuro
13	Miura	Kentaro	https://twitter.com/skull_knight
13	Miura	Kentaro	https://thegodhand.hell/femto

В данном решении *author_id* не является уникальным, и чтобы однозначно идентифицировать строку, потребуется использовать составной ключ (*author_id* и *profile_url*). Значение составного ключа уникально, хотя каждый столбец отдельно содержит (может содержать) повторяющиеся значения. Возможность однозначно идентифицировать строку (кортеж) является требованием 1NF, но использовать составной ключ с таким длинным значением — не самое лучшее решение. И к тому же значение естественного ключа может измениться с течением времени.

Поэтому корректный альтернативный дизайн отношения, который отвечает требованиям 1NF, использует две таблицы. В примере ниже столбцы не содержат более одного профиля соцсети. Вместо этого каждая связка «идентификатор автора» и «профиль соцсети» отображается в отдельной строке.

Обратите внимание, что в данном случае между таблицами *Author* и *SN_Profile* существует связь «один ко многим» (здесь и далее цветом выделены первичные ключи таблиц). Строка в «родительской» таблице *Author* может быть связана с несколькими профилями соцсетей в «дочерней» таблице *SN_Profile*, но каждый профиль принадлежит одному и только одному автору (правда, в реальной жизни бывают редкие случаи, когда один профиль ведут несколько человек).

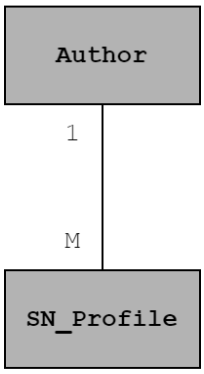
Стоит также отметить, что данное представление отвечает требованиям для 2NF и 3NF.

Таблица Author

author_id	first_name	surname
12	Oda	Eiichiro
13	Miura	Kentaro

Таблица SN_Profile

id	author_id	profile_url
1	12	https://twitter.com/zorojuro
2	13	https://twitter.com/skull_knight
3	13	https://thegodhand.hell/femto



- 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 функционально полно зависит от ее потенциального ключа.

Функционально полная зависимость означает, что неключевые атрибуты зависят от всего *потенциального ключа*, а не от его части. Если потенциальный ключ является простым (состоит из единственного атрибута), то любая функциональная зависимость от него является полной. Таким образом, вторая часть правила 2NF касается исключительно *составных потенциальных ключей*, для которых не должно быть неключевых атрибутов, зависящих только от части составного ключа.

Потенциальный ключ (candidate key) — в реляционной модели данных представляет собой подмножество атрибутов отношения, которое удовлетворяет требованию *уникальности и минимальности*.

Уникальность означает, что для данного отношения нет и не может быть двух совершенно одинаковых строк (кортежей) — в которых значения подмножества атрибутов совпадают.

Минимальность (несократимость) означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, удовлетворяющее условию уникальности. Иными словами, если из потенциального ключа убрать любой атрибут, он утратит свойство уникальности.

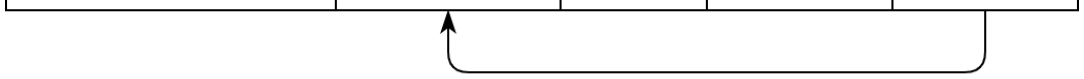
В отношении может быть одновременно несколько потенциальных ключей, один из которых можно выбрать в качестве первичного ключа отношения, тогда другие потенциальные ключи называют альтернативными ключами. На практике в качестве первичного обычно выбирается потенциальный ключ, который имеет меньший размер (физического хранения) и/или включает меньшее количество атрибутов в сравнении с альтернативными ключами.

В таблице ниже представлен пример неполной функциональной зависимости неключевых атрибутов от составного потенциального ключа. Наименование журнала (*magazine_title*) может быть не уникальным для разных издательств, поэтому

составной потенциальный ключ содержит также издателя журнала (*publisher*). Жанр (в данном случае означает также целевую аудиторию журнала) и частота выхода журнала зависят полностью от потенциального ключа, но страна зависит только от издательства.

В данном примере один и тот же журнал (*Weekly Shonen Jump*) выпускается в двух странах, но разными издательствами (оригинальную версию журнала выпускает издательство из Японии, перевод журнала — издательство из Швеции).

magazine_title	publisher	genre	frequency	country
Young Animal	Hakusensha	Seinen	Biweekly	Japan
Weekly Shonen Jump	Shueisha	Shonen	Weekly	Japan
Weekly Shonen Jump	Manga Media	Shonen	Monthly	Sweden
Be Love	Kodansha	Josei	Monthly	Japan



Функциональные зависимости для данного отношения:
magazine_title, *publisher* ⟶ *genre*, *frequency*
publisher ⟶ *country*

В качестве потенциального составного ключа можно выбрать наименование журнала и страну, но это не решает проблему — взаимозависимость издательства от страны никуда не делась.

Чтобы представление соответствовало 2NF, необходимо разделить его на два отношения — таблица журналов и таблица издательств:

magazine_title	publisher	genre	frequency
Young Animal	Hakusensha	Seinen	Biweekly
Weekly Shonen Jump	Shueisha	Shonen	Weekly
Weekly Shonen Jump	Manga Media	Shonen	Monthly
Be Love	Kodansha	Josei	Monthly

publisher_name	country
Hakusensha	Japan
Shueisha	Japan
Manga Media	Sweden
Kodansha	Japan

На практике приводить отношения к 1NF и 2NF приходится довольно редко. Обычно вполне достаточно просмотреть отношения на предмет того, не нарушают ли они первые две нормальные формы. Немного сложнее ситуация с 3NF, так как транзитивная зависимость не всегда так явно бросается в глаза. Поэтому нарушение 3NF наиболее частая проблема, с которой приходится иметь дело.

- Третья нормальная форма (3NF)

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме и ни один неключевой атрибут не находится в транзитивной функциональной зависимости от ее потенциального ключа.

Под транзитивной зависимостью понимают следующее отношение:

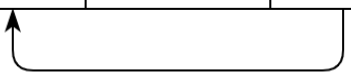
A ⟶ B (A определяет B)

B ⟶ C, следовательно, A ⟶ C (зависимость C от A является транзитивной).

Иными словами, 3NF означает, что каждый атрибут должен предоставлять информацию о потенциальном ключе и ни о чём, кроме ключа (немного упрощенная и неточная формулировка 3NF).

Например, отношение, представленное ниже, находится во 2NF (ни один из атрибутов не зависит от части составного ключа). Нарушение 3NF происходит из-за того, что неключевой атрибут *ages* (возрастная группа журнала) транзитивно зависит от потенциально ключа через неключевой атрибут *genre*, так как *genre* в данном случае определяет целевую аудиторию журнала, и одним из его критериев является возрастная направленность. Соответственно, *ages* функционально зависит от *genre*, что делает таблицу уязвимой для логических несоответствий — например, если в результате ошибки ввода появятся разные записи для одного и того же жанра, но с разным значением возрастной направленности.

magazine_title	publisher	genre	frequency	ages
Young Animal	Hakusensha	Seinen	Biweekly	18-40
Weekly Shonen Jump	Shueisha	Shonen	Weekly	12-18
Weekly Shonen Jump	Manga Media	Shonen	Monthly	12-18
Be Love	Kodansha	Josei	Monthly	18-45



Чтобы представить данное отношение, не нарушив при этом 3NF, необходимо разделить таблицу на две части. Для таблицы жанров ключевым атрибутом будет наименование жанра (*genre_name*).

magazine_title	publisher	genre	frequency
Young Animal	Hakusensha	Seinen	Biweekly
Weekly Shonen Jump	Shueisha	Shonen	Weekly
Weekly Shonen Jump	Manga Media	Shonen	Monthly
Be Love	Kodansha	Josei	Monthly

genre_name	ages
Seinen	18-40
Shonen	12-18
Josei	18-45

Обратите внимание, что приведение к 3NF не требует обязательного наличия составного потенциального ключа, в отличие от не соответствия 2NF, где это является одним из условий для возможного возникновения аномалий добавления, удаления или обновления данных.

1.4.3 Словарь данных

Словарь данных представляет собой централизованное хранилище подробной информации об используемых в приложении сущностях данных.

Информация словаря данных включает в себя определение состава структур данных предметной области, заданных или разрешенных значений, типов данных, длины и формата элементов данных, из которых состоят эти структуры. В конечном итоге словарь данных определяет имена переменных в программе, так как информация, которую он в себя включает, реализуется в форме схем и таблиц баз данных.

Поскольку определения данных часто повторно используются в других приложениях, соответственно применение единообразных определений данных снижает вероятность возникновения ошибок интеграции и интерфейса. Таким образом, словарь данных *является одним из атрибутов повышения качества разработки*, так как служит для определения критериев проверки данных, облегчает поиск необходимой информации, позволяет избежать ненужных повторов и ошибок, связанных с тем, что участники проекта по-разному понимают ключевые данные.

В настоящей работе будет рассмотрен упрощенный вариант словаря данных, состоящий только из структур (сущностей) и простейших элементов данных. Простейшим называется элемент данных, дальнейшая декомпозиция которого невозможна или не нужна (например, идентификатор пользователя, логин, et cetera). В описании простейших элементов указывают тип, длину, диапазон числовых значений, список разрешенных значений и другие уместные атрибуты. Каждый элемент структуры должен быть определен в словаре данных.

При определении такого элемента как, например, «Имя автора», возможно потребуется решить ряд вопросов (что необходимо указать в словаре данных):

- Чувствительно ли имя к регистру?
- Должна ли система приводить весь текст к верхнему или нижнему регистру, или сохранять список значений, которые запрашивали при поиске?
- Могут ли использоваться другие символы кроме 33 букв русского алфавита, такие как пробелы, дефисы, точки или апострофы?

- Разрешен ли только русский алфавит или возможно использование букв алфавитов с диакритическими знаками, такими как тильда (~), умляут (¨), ударения (` или ´), седиль (¸), et cetera.

Точные определения данных очень важны для разработчика, чтобы понимать, как проверять вводимые данные. При выполнении работы необходимо определить минимальные критерии для проверки элементов данных, такие как тип данных, длина и возможные или заданные значения.

Необязательные элементы, значение которых не должно предоставляться пользователем или системой, заключают в скобки. В примере ниже (табл. 1) такими элементами являются, например, «Страна проживания» или «Профиль соцсети» — не обязательные для заполнения.

Обозначения типов данных, принятых в SQL[7], указаны в таблице 2 (в таблицу включены типы данных, которые, скорее всего, вы используете в работе).

Обратите внимание, что в базе данных для имени пользователя фактически может быть определен тип VARCHAR (255), но в словаре данных необходимо указать *возможное максимальное значение длины*, чтобы подсказать разработчику, какого рода проверки необходимо реализовать. Например, некоторая длина атрибутов может нарушать параметры отображения их значений в интерфейсе. Иными словами, в базе данных у нас длина может быть задана 255, а для корректного отображения в интерфейсе необходимо не более 50.

Таблица 1 — Фрагмента словаря данных для системы
«Электронный тематический журнал манги»

Структура или элемент данных	Тип данных	Длина	Значение
Пользователь			
Идентификатор	INT	10	(PK) Первичный ключ — автоинкрементный номер записи, генерируемый системой, начиная с 1
Статус профиля	VARCHAR	5	ADMIN USER RCF (registration confirmation) — статус для учетной записи, которая требует подтверждения по email
Логин	VARCHAR	25	Может содержать только символы латинского алфавита, подчеркивание и цифры
E-mail	VARCHAR	50	Должен соответствовать стандарту RFC 5322
Пароль	VARCHAR	50	Может содержать символы латинского алфавита, числа и символы из следующего после двоеточия списка: ! @ # $ % ^ & ? * _ Остальные символы, включая пробел, запрещены
Имя пользователя	VARCHAR	50	Может содержать все буквенно-цифровые символы, включая символы национального алфавита
Дата рождения	DATE	—	Используется для рекомендаций
(Аватар)	VARCHAR	255	Содержит путь файла
(Фон профиля)	VARCHAR	255	Содержит путь файла

[7] Structured Query Language — язык структурированных запросов (в реляционной базе данных)

Способ авторизации	VARCHAR	20	Login Google Twitter Facebook
(Информация)	TEXT	—	Содержит дополнительную информацию, которую пользователь указывает о себе
(Страна проживания)	ENUM	—	В интерфейсе выбирается из выпадающего списка
Рассылка	BIT	1	Поле не может быть пустым 0 — Нет, спасибо 1 — Да (по умолчанию)
(Пол)	BIT	1	Используется для рекомендаций / рекламы манги 0 — Мужчина 1 — Женщина NULL (то есть отсутствие значения) соответствует: Не указан или не определен (по умолчанию)
(Контент для взрослых)	BIT	1	0 — Нет, я не хочу видеть «Контент для взрослых» 1 — Да, всегда отображать «Контент для взрослых» NULL (отсутствие значения) соответствует: Всегда спрашивать разрешение на отображение контента для взрослых (по умолчанию)

Обратите внимание на атрибут «статус профиля», для которого определен тип VARCHAR, хотя он принимает всего три значения. Для двух или трех значений, если мы знаем, что их количество будет неизменным, можно выделить тип BIT[8] (аналог типа данных BOOLEAN в SQL), который принимает одно из двух значений — 0 или 1. NULL — является отсутствием какого-либо значения, соответственно, проверку атрибута функцией IS NULL (или IS NOT NULL) можно использовать как третье «значение» (отсутствие значения использовать как значение).

Для атрибута «статус профиля» количество значений может измениться, если появится новая роль в системе со своими правами доступа, например, модератор. Поэтому тип BIT не подходит для этой задачи. Можно использовать целочисленный тип данных и это является хорошим решением, если для нас важен объем базы данных и скорость ее работы. Однако желаемый эффект будет достигнут только для по-настоящему большой базы данных ценой удобства работы с ней. Например, при написании тех же запросов намного удобнее использовать осмысленные значения[9]. Представьте, что у вас есть фронтендер (*frontend developer*) и вы ему возвращаете *userType* = 1. Что он должен думать? А если *userType* = *admin*?

Поскольку способы авторизации могут измениться в процессе эксплуатации системы, то для них ENUM не подойдет. Для страны проживания можно использовать ENUM, так как наименования стран редко меняются, однако если всё же это случится, в этом случае придется изменить структуру всей таблицы, что довольно затратно по ресурсам и времени.

[8] При BIT (n = 1), где n — количество битов

[9] Существует иная точка зрения, что осмысленные значения уязвимы, например, для SQL-инъекций и создают риски для безопасности системы. Реальность же показывает, что обфускация (запутывание кода) никогда не даёт надежной защиты и лучше сконцентрировать своё внимание на правильной валидации вводимых данных.

Стоит отдельно отметить, что некоторые веб-приложения могут использовать ненадежные способы валидации данных и фильтрации ввода, в результате чего возможны успешные атаки с помощью SQL-инъекций[10], поэтому с точки зрения безопасности необходимо хранить хэш пароля (а не сам пароль), но *в словаре данных, как правило, не указывают требования для хранения данных* (хотя и такое может быть), *в словаре необходимо указать критерии проверки данных*, в соответствии с которыми разработчик должен реализовать их валидацию, прежде, чем данные будут обработаны системой и/или сохранены в базе данных. То есть пользователь вводит пароль, далее система должна проверить, что пароль соответствует тем требованиям, которые указаны в словаре, затем хэш пароля сохраняется в базу данных. Причем типы данных также выступают критерием проверки, так как база данных может быть нереляционной (NoSQL), соответственно, определенные в словаре типы с их расшифровкой (таблица 2), помогут определить тип хранения для конкретной СУБД (иными словами, словарь не привязывается к конкретной СУБД и в нем могут быть указаны произвольные типы — например, «Целое число» вместо INT. У вас в работе тип должен соответствовать MySQL, так как с конкретными определениями типов удобнее работать, чем просто с «Целым числом».

Как было отмечено ранее, словарь данных — это инструмент для повышения качества разработки, так как единообразные наименования переменных и проверки данных облегчают разработку и сопровождение ПО (например, не нужно тратить время на обдумывание того, какие проверки данных необходимо реализовать — они определены в словаре и их можно использовать в каждом проекте).

Требования к хранению данных (например, какой алгоритм использовать для шифрования или формат и длина соли для хэш-функции) — это отдельный документ (например, это может быть указано в SRS[11]), так как если словарь данных и, представленные в нем критерии для проверки данных, можно использовать в разных проектах, то требования к хранению данных в этих проектах могут отличаться (например, если заказчиком выступает госучреждение, то в требованиях может быть указано, что для шифрования необходимо использовать не AES, а ГОСТ 34.12-2018).

Таблица 1 (продолжение)

Номер журнала			
Год выпуска	YEAR	4	(PK) Атрибут составного первичного ключа — содержит значение года, за которым закреплен выпуск данного номера журнала
Номер	TINYINT	2	(PK) Атрибут составного первичного ключа — содержит порядковый номер журнала за определенный год (не может превышать двухзначное число)
Дата выпуска	DATE	—	Содержит дату фактического выпуска журнала, которая может отличаться от года, за которым закреплен выпуск номера журнала
Обложка	VARCHAR	255	Содержит путь файла

[10] Один из самых распространенных способов взлома сайтов и программ, которые работают с базами данных. Заключается во внедрении в запрос некоторого SQL-кода, например, злоумышленник вместо данных, которые вы предлагаете ему ввести, вводит строку SQL-кода для передачи ему содержимого базы данных, и система выполнит эту строку, если в программном обеспечении системы присутствуют уязвимости безопасности, например, при неправильной фильтрации пользовательского ввода для управляющих символов, встроенных в операторы SQL, или если пользовательский ввод не является строго типизированным, в результате чего может быть выполнен как запрос (собственно, для этого и нужны критерии проверки данных, которые указывают в словаре).

[11] Software Requirements Specification — спецификация требований к программному обеспечению

Первая страница	VARCHAR	255	Содержит путь файла
Аннотация	VARCHAR	255	Содержит короткое описание содержания номера журнала
Жанр	ENUM	—	Возможные значения: Shonen Shojo Seinen Josei (используется для рекомендаций; в интерфейсе может не отображаться)
Количество страниц	SMALLINT	4	Не может превышать 620 страниц (ограничение типографского издания)
Издатель	VARCHAR	50	Значение по умолчанию: Kodansha (по умолчанию указано основное издательство)

Для жанра можно использовать ENUM, так как маловероятно, что жанры когда-либо изменятся (это связано с тем, что жанры манги/журнала представляют собой скорее целевую аудиторию, чем набор тех или иных сюжетных и стилистических признаков, как это характерно, например, для кинематографа или литературы).

Обратите внимание на атрибуты «Обложка» и «Первая страница», может показаться, что это разные значения для одного атомарного атрибута (см. стр. 11 — нарушение 1NF) , на самом деле это разные атрибуты и они по-разному используются в системе. Первая страница содержит подробную информацию о выпуске и может быть доступна только при отдельном обращении к ней, тогда как обложка доступна при каждом обращении к номеру журнала. Аналогично для обложки и превью в сущности «Манга». Превью предназначено для ознакомления со стилистикой автора манги (обложка может не содержать примеров стилистики автора или стилистически сильно отличаться от самой манги).

Таблица 1 (продолжение)

Автор			
Идентификатор	INT	10	(PK) Первичный ключ — автоинкрементный номер записи, генерируемый системой, начиная с 1
(Псевдоним)	VARCHAR	50	Может содержать все буквенно-цифровые символы
Имя автора	VARCHAR	50	Может содержать все буквенно-цифровые символы Вместо имени автора может быть указано наименование художественного коллектива
(Фамилия автора)	VARCHAR	50	Может содержать все буквенно-цифровые символы
(Дата рождения)	DATE	—	В системе можно задать отображение только числа и месяца рождения (вместо полной даты)
(Место рождения)	VARCHAR	255	Может содержать все буквенно-цифровые символы
(Информация)	TEXT	—	Краткая информация об авторе (по согласованию с автором) Подсказка при заполнении: «По умолчанию должна содержать строку следующего вида (пример): Дебютировал в 1988 году отдельным томом «It is awfully hard work doing nothing» (Kodansha)»
(Аватар)	VARCHAR	255	Содержит путь файла

Манга			
Идентификатор	INT	10	(PK) Первичный ключ — автоинкрементный номер записи, генерируемый системой, начиная с 1
Наименование	VARCHAR	255	Может содержать все буквенно-цифровые символы
Автор манги	INT	10	(FK) Внешний ключ — содержит идентификатор автора
Дата выпуска	DATE	—	В интерфейсе должно отображаться в формате YYYY/MM/DD
Обложка	VARCHAR	255	Содержит путь файла
История	VARCHAR	255	Содержит небольшое описание по типу логлайна (то есть краткая аннотация к манге, которая передает суть истории и ее основную драматическую коллизию)
Количество страниц	SMALLINT	5	Если манга состоит из глав, то здесь указывается суммарное количество страниц
Контент для взрослых	BIT	1	Поле не может быть пустым 0 — Отсутствует 1 — Могут присутствовать сексуальные сцены, эпизоды с употреблением наркотиков, нецензурной бранью, насилием, et cetera.

Обратите внимание на атрибут «Автор манги». Между таблицами «Манга» и «Автор» существует отношение «один ко многим» (а не «многие ко многим»), так как в отличие от книг, у манги всегда один автор. Если это коллектив художников, то атрибут «Автор манги» содержит идентификатор художественного коллектива, информацию о котором также включает в себя таблица «Автор», при этом состав коллектива может хранится в отдельной таблице.

Собственно, *отношение «один ко многим» означает, что в таблице со стороны «многие» должен быть первичный ключ другой таблицы*, например, «Пользователь» и «Комментарий» — в комментарии должен быть идентификатор пользователя, чтобы понять, кто написал данный комментарий (аналогично для манги и мангаки, то есть автора манги);

Также обратите внимание на атрибут «Контент для взрослых», в таблице «Пользователь» есть атрибут, который устанавливает правила отображения для данного контента (в таблице «Номер журнала» указанный атрибут не требуется, так как пользователь может ознакомиться только с отдельной мангой журнала).

Таблица 2 — Принятые типы данных в SQL (неполный список)

1	TINYINT	Целочисленный тип размером 1 байт Со знаком от -128 до 127, без знака от 0 до 255
2	SMALLINT	Целочисленный тип размером 2 байта Со знаком от -32 768 до 32 767, без знака от 0 до 65 535
3	MEDIUMINT	Целочисленный тип размером 3 байта Со знаком от -8 388 608 до 8 388 607, без знака от 0 до 16 777 215
4	INT	Целочисленный тип размером 4 байта Со знаком от -2 147 483 648 до 2 147 483 647, без знака от 0 до 4 294 967 295
5	BIGINT	Целочисленный тип размером 8 байт Со знаком от -2^{63} до $2^{63}-1$, без знака от 0 до $2^{64}-1$

6	DECIMAL	Тип с фиксированной точкой (часто используется для денежных данных) DECIMAL (size, d), где size — общее количество цифр (максимум 65), d — количество цифр после точки (максимальное значение для d — 30). Значения по умолчанию — 10 (для size) и 0 (для d). Например, DECIMAL (7, 2) может хранить значения от -99999.99 до 99999.99
7	FLOAT	Тип с плавающей точкой размером 4 байта Обратите внимание, что каждая база данных имеет разные реализации типа FLOAT. В старых версиях MySQL (до 8.0.17) можно было указать точность для числа с плавающей запятой (аналогично DECIMAL), то есть FLOAT (size, d). В текущих версиях данный тип выражается как FLOAT (n), где n определяет, будет ли значение сохранено как FLOAT или преобразовано в DOUBLE. При n от 0 до 23 значение хранится в виде 4-байтового столбца с одинарной точностью, при n от 24 до 53 в виде 8-байтового столбца с двойной точностью (тип DOUBLE). По умолчанию значение n равно 53 (двойная точность). Значения с плавающей точкой представляются в форме экспоненциальной записи: $N = M \cdot n^p$, где N — число, M — мантисса, n — основание, p — порядок. Например, число $N = 0.004137 = 4.137 \cdot 10^{-3}$ (4,137 — мантисса, 10 — основание, -3 — порядок). Эта запись для человека, а для хранения в памяти машины удобна запись в форме MEp, где M — мантисса, E — « · 10^» (умножение на 10 в степени), p — порядок. Таким образом, $4.137 \cdot 10^{-3}$ это 4.137E-3, а $4.137 \cdot 10^3$ это 4.137E+3. Диапазон значений для одинарной точности: от -3.40E+38 до -1.18E-38, 0 и от 1.18E-38 до 3.40E+38 Диапазон значений для двойной точности: от -1.79E+308 до -2.23E-308, 0 и от 2.23E-308 до 1.79E+308
8	DOUBLE	Тип с плавающей точкой размером 8 байт (двойная точностью)

9	BIT	Целочисленный тип данных, который может принимать значения 0, 1 или NULL (используется для хранение битовых значений) BIT (n), где n — количество битов (от 1 до 64), если n указано больше, чем длина хранимого значения, то значение слева заполняется нулями, например, для числа 13 — 1101, если задано BIT (7), то значение примет следующий вид — 0001101

10	DATE	Хранение даты в формате YYYY-MM-DD Поддерживает диапазон от 1000-01-01 до 9999-12-31
11	DATETIME	Хранение даты и времени в формате YYYY-MM-DD hh:mm:ss Поддерживает диапазон от 1000-01-01 00:00:00 до 9999-12-31 23:59:59
12	TIME	Хранение значения времени в формате hh:mm:ss Поддерживает диапазон от -838:59:59 до 838:59:59 Используется не только для представления времени дня (которое должно быть меньше 24 часов), но и для прошедшего времени или временного интервала между двумя событиями
13	YEAR	Хранение значения года в формате YYYY Тип YEAR занимает 1 байт, поэтому поддерживает диапазон от 1901 до 2155 и 0000 (MySQL 8.0 не поддерживает задание года в двузначном формате)

14	CHAR	Строка фиксированной длины (может содержать буквы, цифры и специальные символы). CHAR (size), где size — длина строки в символах (от 0 до 255, по умолчанию 1)

<table>
<tr><td>15</td><td>VARCHAR</td><td>Строка переменной длины (может содержать буквы, цифры и специальные символы).
VARCHAR (size), где size — максимальная длина строки в символах
(от 0 до 65535)

Разница между CHAR и VARCHAR
<table><tr><th>Значение</th><th>CHAR (4)</th><th>Байт</th><th>VARCHAR (4)</th><th>Байт</th></tr><tr><td>' '</td><td>' '</td><td>4</td><td>' '</td><td>1</td></tr><tr><td>'AB'</td><td>'AB '</td><td>4</td><td>'AB'</td><td>3</td></tr><tr><td>'ABCD'</td><td>'ABCD'</td><td>4</td><td>'ABCD'</td><td>5</td></tr><tr><td>'ABCDEFGH'</td><td>'ABCD'</td><td>4</td><td>'ABCD'</td><td>5</td></tr></table>
При использовании однобайтовых кодировок размер типов CHAR и VARCHAR при хранении равен количеству символов (VARCHAR помимо самой строки еще хранит префикс длины — количество байтов строки). Однако в случае многобайтовых кодировок, таких как UTF-8, в старших диапазонах Юникода один символ занимает два или несколько байт</td></tr>
<tr><td>16</td><td>TINYTEXT</td><td>Хранение строки максимальной длины в 255 символов</td></tr>
<tr><td>17</td><td>TEXT</td><td>Хранение строки максимальной длины в 65 535 символов</td></tr>
<tr><td>18</td><td>MEDIUMTEXT</td><td>Хранение строки максимальной длины в 16 777 215 символов</td></tr>
<tr><td>19</td><td>LONGTEXT</td><td>Хранение строки максимальной длины в 4 294 967 295 символов</td></tr>
</table>

20	BINARY	Аналог CHAR, но данные хранятся в виде бинарной строки (бинарная строка состоит только из символов 0 и 1) BINARY (size), где size — длина строки в байтах (от 0 до 255, по умолчанию 1)
21	VARBINARY	Аналог VARCHAR, но данные хранятся в виде бинарной строки VARBINARY (size), где size — максимальная длина строки в байтах (от 0 до 65535)
22	TINYBLOB	Хранение Binary Large Object (двоичный большой объект) размером до 255 байт включительно BLOB предназначен для хранения изображений или скомпилированного программного кода
23	BLOB	Хранение BLOB размером до 65 535 байт включительно
24	MEDIUMBLOB	Хранение BLOB размером до 16 777 215 байт включительно
25	LONGBLOB	Хранение BLOB размером до 4 294 967 295 байт включительно

26	ENUM	Специальный строковый тип, который принимает *только одно значение* из фиксированного списка значений. В списке ENUM, который определяется *во время создания таблицы* в базе данных, можно задать до 65 535 значений. Все недопустимые значения (которых нет в списке) при добавлении заменяются на пустые строки. Значения в списке отсортированы в порядке их ввода.

2. ЛАБОРАТОРНАЯ РАБОТА

«ОПИСАНИЕ ВЗАИМОДЕЙСТВИЯ ПОЛЬЗОВАТЕЛЯ И СИСТЕМЫ. СПЕЦИФИКАЦИЯ ВАРИАНТА ИСПОЛЬЗОВАНИЯ»

2.1 Цель работы

Целью настоящей работы является изучение одного из способов описания взаимодействия пользователя с системой и получение практических навыков составления спецификации варианта использования.

2.2 Задание на работу

Для заданной предметной области составьте спецификацию не менее четырех вариантов использования:

1) Варианты использования, контекст которых заключается в добавлении ключевых объектов в систему, имеют приоритет в плане выбора для спецификации (ключевые объекты определяются как элементы, хранение информации о которых в базе данных составляет основу деятельности в предметной области);
2) Остальные варианты использования могут быть выбраны на усмотрение студента;
3) Основная сложность вариантов использования должна заключаться во взаимодействии пользователя с системой, в противном случае вариант использования не нуждается в спецификации;
4) Обязательные элементы спецификации:
   - Уникальный идентификатор и наименование;
   - Автор спецификации и дата создания;
   - Основные действующие лица;
   - Краткое описание;
   - Приоритет;
   - Предварительные условия (ноль или больше);
   - Выходные условия (одно или больше);
   - Основные потоки;
   - Возможные альтернативные потоки;
   - Возможные наиболее вероятные исключения, которые не позволяют успешно выполнить заданные основные и альтернативные направления развития варианта использования;
   - Метки начала альтернативных потоков и вызова исключений.

Конечное содержание спецификации вариантов использования определяется во время защиты лабораторной работы. Таким образом, согласование содержания спецификации вариантов использования с преподавателем и является процедурой защиты работы.

2.3 Рекомендации и требования по содержанию отчета

Если отчет по работе загружен в личный кабинет и принят преподавателем, то дальнейшая пересдача лабораторной работы повлечет двукратное увеличение объема задания — таким образом, для ЛР 2 потребуется составить спецификацию не менее 8-ми вариантов использования.

Для выполнения лабораторной работы не потребуется специализированное программное средство. Спецификацию вариантов использования можно реализовать при помощи обычных таблиц офисного пакета.

Пример оформления отчета с частично выполненной работой прилагается к заданию.

Отчет к лабораторной работе должен содержать:

- Титульный лист
- ВВЕДЕНИЕ (короткое описание актуальности работы, формулировка цели и постановка задачи, индивидуальный вариант задания)
- Спецификация вариантов использования
- ЗАКЛЮЧЕНИЕ (выводы по работе)
- СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Оформление отчета должно соответствовать требованиям, представленным в правилах и критериях оценки по дисциплине.

2.4 Теоретический материал

Спецификация варианта использования, которая определяет порядок диалога между пользователем и системой, помогает сформулировать функциональные требования и составить варианты тестирования для определения того, насколько корректно реализован данный вариант использования.

2.4.1 Спецификация варианта использования

Вариант использования описывает последовательность взаимодействия системы и внешнего действующего лица, в результате которого действующее лицо получает полезный результат.

Имена вариантов использования всегда пишутся в формате «глагол и объект» (например: обновить профиль клиента; купить товар; отменить заказ; et cetera).

Необходимые элементы описания варианта использования (ВИ):

- Уникальный идентификатор и краткое имя (имя определяет цель пользователя);
- Краткое текстовое описание цели варианта использования;
- Условие-триггер, инициирующее выполнение ВИ;
- Список предварительных условий (*ноль или больше*), которые должны быть удовлетворены до начала выполнения ВИ;
- *Одно или больше* выходных условий, описывающих состояние системы после успешного завершения ВИ;
- Пронумерованный список действий (или операций), которые определяют последовательность этапов взаимодействия действующего лица и системы от предварительных до выходных условий.

Обратите внимание, что в задании к лабораторной работе указан несколько иной список обязательных элементов ВИ.

Типовой шаблон ВИ представлен в таблице 3. Как и другие шаблоны, его не обязательно заполнять сверху вниз, так как некоторая информация шаблона нужна далеко не в каждом ВИ. Шаблон всего лишь предоставляет структуру для хранения информации (в упорядоченном и единообразном виде), которая была выявлена в процессе обсуждения ВИ.

Таблица 3 — Шаблон описания варианта использования

Идентификатор ВИ	UC-2
Наименование	Заказ печатной версии журнала
Автор	Евгений Павлов
Дата создания	20.10.2021
Основное действующее лицо	Клиент, посетитель сайта
Дополнительное действующее лицо	Сервис электронных платежей, менеджер заказов

Описание	Клиент обращается к системе, просматривает список изданных журналов за определённый период, выбирает номер журнала и осуществляет заказ печатной версии для самовывоза непосредственно из представительства издателя или выбирает доставку в определённый пункт
Приоритет	Высокий
Условие-триггер	Клиент выражает намерение заказать печатную продукцию посредством взаимодействия с элементом интерфейса «Заказать печатную версию журнала»
Предварительные условия	PRE-1 Клиент выполнил вход в систему (в противном случае форма для заказа будет пустой) PRE-2 БД номеров журнала в данный момент доступна PRE-3 Номер журнала доступен для осуществления заказа
Выходные условия	POST-1 Заказ сохранён в БД с состоянием «Принят» POST-2 Список доступных номеров журнала обновлен с учетом элементов данного заказа POST-3 Система выводит подробную информацию о заказе на экран POST-4 Система отправляет SMS-уведомление на указанный в заказе номер телефона
Основные потоки	2.0 Заказ печатной продукции 1. Клиент просматривает список изданных номеров журнала за определённый период 2. Система отображает список доступных для заказа номеров журнала и специальные предложения (акции) 3. Клиент выбирает один или более журналов из списка и перемещает их в корзину 4. В корзине отображается количество выбранных позиций 5. Клиент переходит к оформлению заказа (см. 2.1) 6. Система просит проверить заполненную форму (регистрационные данные) и скорректировать количество печатной продукции для заказа (см. 2.2) 7. Клиент подтверждает, что оформление заказа завершено 8. Система отображает номера журналов в заказе с указанием стоимости отдельной позиции и общую сумму заказа, включая налог и стоимость доставки 9. Клиент подтверждает заказ или делает запрос на изменение заказа 10. Система выводит возможное время и пункты доставки 11. Клиент выбирает время (из списка) и указывает пункт доставки (см. 2.0 E1) 12. Клиент указывает метод оплаты 13. Система подтверждает, что заказ принят 14. Система отправляет клиенту SMS-сообщение на указанный в заказе номер с подтверждением деталей заказа, включая указания по доставке 15. Система сохраняет заказ в БД, посылает информацию о заказе менеджеру заказов
Альтернативные потоки	2.1 Заполнение формы неавторизованным пользователем (посетитель сайта) 1. Система просит заполнить форму заказа: - Имя (обязательно) - Телефон (обязательно) - Пункт доставки (обязательно) - Email

	- Время доставки (если в пункте доставки указано представительство издателя, то время доставки не является обязательным для заполнения) - Комментарий 2. Пользователь заполняет все обязательные поля (см. 2.1 E2) 3. Возврат к пункту 6 основного потока 2.2 Заказ нескольких идентичных номеров журнала 1. Клиент делает запрос на заказ определенного числа идентичных номеров журнала или удаление некоторых позиций (см. 2.2 E3) 2. Возврат к пункту 6 основного потока
Исключения	2.0 E1 Нет возможности доставки по указанному адресу 1. Система сообщает клиенту, что нет возможности доставки по указанному адресу 2.1 Если клиент отменяет оформление заказа, то система завершает вариант использования 2.2 В противном случае клиент делает запрос на самовывоз непосредственно из представительства издателя, система возвращается к пункту 12 основного потока 2.1 E2 Клиент некорректно заполнил все обязательные поля 1. Система подсвечивает некорректно заполненные поля и просит клиента проверить введенные данные 2.1 Клиент вносит необходимые корректировки и возвращается к пункту 6 основного потока 2.2 В противном случае клиент отменяет оформление заказа, система в свою очередь завершает вариант использования 2.2 E3 Невозможно выполнить заказ на указанное количество одинаковой печатной продукции (номеров журнала) 1. Система извещает клиента о максимальном числе одинаковых позиций продукции, на которые она способна принять заказ (доступное количество единиц продукции) 2.1 Клиент изменяет количество единиц продукции и возвращается к пункту 6 основного потока 2.2 В противном случае клиент отменяет оформление заказа, система в свою очередь завершает вариант использования
Бизнес-правила	(в рамках данной работы не заданы)
Другая информация	1. Клиент должен иметь возможность отменить заказ в любой момент до подтверждения заказа 2. Клиент должен иметь возможность просматривать все заказы за последние 24 месяца
Предположения	Предполагается, что 20% клиентов будут заказывать специальные предложения (номера журналов по акции)

Пользователи — это реальные люди (или системы), *действующие лица* — абстракции. Один человек может принадлежать разным классам пользователей, которые и формируют действующие лица системы (например, администратор может работать с системой как рядовой пользователь).

Обратите внимание, что при выполнении ВИ взаимодействие осуществляется также с сервисом электронных платежей (если пользователь указал безналичный способ оплаты), менеджером заказов и администратором (им приходит уведомление о новом заказе).

Короткое описание ВИ — что именно действующее лицо делает и как на это реагирует система.

Событие-триггер — условие, которое инициирует выполнение ВИ, то есть оно информирует, что пользователь хочет выполнить определенный ВИ. Не является частью предварительных условий.

Предварительные условия (предусловия) — определяют, что конкретно должно присутствовать в системе до выполнения ВИ. Система должна иметь возможность проверить все заданные предварительные условия, чтобы определить, возможно ли выполнение данного ВИ.

Обратите внимание, что результат самих предварительных условий не обязательно означает, что ВИ не будет выполнен — отсутствие соответствующей проверки может сделать невозможным выполнение ВИ, но не их результат. В примере выше, если информации о клиенте нет в базе данных, то пользователю будет предложено заполнить форму заказа (альтернативный поток 2.1) — и это не просьба авторизоваться (мы это уже проверили в предварительных условиях и вход в систему не является частью данного ВИ), это именно выполнение заказа неавторизованным пользователем.

Выходные условия (постусловия) — описывают состояние системы после успешного выполнения ВИ, например:

- Некоторое поведение системы, которое пользователь может наблюдать (информационное сообщение);
- Физические результаты (кофейный автомат выдал приготовленный кофе или терминал осуществил печать документа);
- Изменение внутреннего состояния системы (транзакции[12] в базе данных).

Основной поток — один сценарий развития ВИ, который приводит к успешному выполнению задания. С точки зрения пользователя, это то, что должно происходить по умолчанию (основных потоков может быть несколько).

Альтернативные потоки также приводят к успешному выполнению задания и удовлетворяют выходным условиям ВИ, но они представляют менее популярные или менее приоритетные вариации самой задачи или способа её выполнения.

Исключения — условия, которые препятствуют успешному выполнению ВИ. Исключения описывают не только эти условия, но и способы их обработки.

[12] Применительно к базе данных транзакция — это несколько запросов (операций чтения и записи), которые рассматриваются как единый запрос, при этом транзакция может быть выполнена либо целиком и успешно независимо от других (параллельно выполняемых) транзакций, либо не выполнена вообще, и тогда она не должна произвести никакого эффекта

3. ЛАБОРАТОРНАЯ РАБОТА

«АНАЛИЗ ПОЛЬЗОВАТЕЛЬСКИХ ТРЕБОВАНИЙ В ПРОЕКТАХ ГИБКОЙ РАЗРАБОТКИ (AGILE). ПОЛЬЗОВАТЕЛЬСКИЕ ИСТОРИИ»

3.1 Цель работы

Целью настоящей работы является изучение одного из способов анализа и документирования пользовательских требований в проектах гибкой разработки и получение соответствующих практических навыков составления пользовательских историй и критериев приёмки.

3.2 Задание на работу

Для заданной предметной области составьте не менее 6 пользовательских историй (используя шаблон Connextra):

1) Истории должны охватывать не менее 3 различных типов пользователей системы и *представлять ценность для них относительно выбранной предметной области* (иными словами, истории общего характера по типу регистрации/авторизации в системе или формы обратной связи в данном критерии не учитываются);
2) Составьте критерии приемки для всех историй;
3) Историй должны включать в себя сценарии или правила для обработки ошибочных ситуаций (наличие[13] и количество сценариев или правил зависит от их приоритета[14] — ошибочные ситуации с низким приоритетом можно не включать в критерии приемки);
4) Проверьте истории на соответствие принципу INVEST.

Конечное описание пользовательских историй определяется во время защиты лабораторной работы. Таким образом, процедура защиты работы заключается в согласовании с преподавателем содержания пользовательских историй и критериев приёмки.

3.3 Рекомендации и требования по содержанию отчета

Если отчет по работе загружен в личный кабинет и принят преподавателем, то дальнейшая пересдача лабораторной работы повлечет двукратное увеличение объема задания — таким образом, для ЛР 3 потребуется составить не менее 12 пользовательских историй.

Для выполнения лабораторной работы не потребуется система управления проектами или другое специализированное программное средство.

Пример оформления отчета прилагается к заданию.

В работе не рассматриваются критерии завершенности (Definition of Done) — глобальный контрольный список требований для всех пользовательских историй

[13] Некоторые пользовательские истории не предполагают сценарии для ошибочных ситуаций

[14] В данном случае приоритет ошибочной ситуации показывает, насколько часто пользователь может с ней столкнуться

(общие требования всех историй проекта). Критерии завершенности не стоит путать с критериями приемки (см. 3.4.2), которые составляются отдельно для каждой истории. Поскольку критерии завершенности определяются командой разработки, соответственно, они выходят за рамки данной работы.

Также стоит отметить, что написание пользовательских историй в гордом одиночестве не является хорошей практикой[15], поэтому работа предполагает некоторую имитацию обсуждения историй (согласование с преподавателем).

При выполнении работы вы можете выбрать любой из предложенных в 3.4.2 типов критериев приемки.

В задании указано, что пользовательская история должна нести ценность для пользователя с точки зрения предметной области — данное требование является попыткой ограничить использование одних и тех же историй в каждом отчете. Если история описывает взаимодействие пользователя с объектами предметной области, то она отвечает указанному требованию. Например, для электронного журнала манги такими историями могут быть:

- Поиск информации по сайту (с учетом того, что в результатах поиска могут быть разделы номеров журнала и манги, данная история взаимодействует с элементами предметной области);
- Просмотр информации о номере журнала (посетитель сайта);
- Вывод списка манги, которую включает в себя журнал (посетитель сайта);
- Просмотр информации о манге (посетитель сайта);
- Пробное чтение манги (посетитель сайта);
- Пройти опрос читателя журнала (авторизованный пользователь);
- Добавить комментарий о номере журнала (авторизованный пользователь);
- Добавить номер журнала (контент-менеджер);
- Добавить мангу (контент-менеджер);
- Добавить опрос читателей на сайт (админ);
- Et cetera…

Истории для регистрации/авторизации, обратной связи, новостной ленты, рекламных объявлений, et cetera… непосредственно (напрямую) не взаимодействуют с основными объектами предметной области, поэтому не отвечают указанному требованию задания. Тем не менее вы можете использовать их в работе, вам просто не будут начислены соответствующие баллы.

Отчет к лабораторной работе должен содержать:

- Титульный лист
- ВВЕДЕНИЕ (короткое описание актуальности работы, формулировка цели и постановка задачи, индивидуальный вариант задания)
- Пользовательские истории (шаблон Connextra)
- ЗАКЛЮЧЕНИЕ (выводы по работе)
- СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Оформление отчета должно соответствовать требованиям, представленным в правилах и критериях оценки по дисциплине.

[15] Как и поручение написание историй одному человеку (например, менеджеру проекта)

3.4 Теоретический материал

Как и варианты использования пользовательские истории предназначены для документирования пользовательских требований с точки зрения взаимодействия пользователя и системы. Но в отличие от вариантов использования, спецификация которых обеспечивает всю необходимую информацию и может быть представлена с точки зрения формальной модели[16], пользовательские истории содержат небольшое количество деталей и остаются открытыми для интерпретации. Таким образом, пользовательские истории представляют собой быстрый способ документирования требований клиента без необходимости разрабатывать и поддерживать обширные формализованные документы.

3.4.1 Пользовательские истории

Пользовательская история (user story) — это способ описания требований к разрабатываемой системе, сформулированный как одно или более предложений на повседневном языке пользователя. Иными словами, это короткое описание того, что система должна делать для пользователя. При этом история является обсуждаемым представление требования, а не детальным его описанием.

Шаблон[17] описания пользовательской истории:
Как (тип пользователя)
Я хочу (выполнить некоторое действие)
Чтобы (достигнуть определенной цели / результата / ценности)

> *Как* получатель посылки
> *Я хочу* видеть статусы всех моих посылок в одном месте
> *Чтобы* сэкономить время на запрос трекинга (отслеживания) каждой посылки

> *Как* администратор веб-сайта
> *Я хочу* иметь возможность публиковать новости
> *Таким образом*, пользователи веб-сайта будут в курсе последних изменений и важных событий

Цель пользовательских историй — объяснить *конкретные роли* пользователей в системе, их желаемые действия и то, чего они намерены достигнуть, успешно завершив историю.

Объем пользовательской истории определяют так, чтобы её реализация могла уложиться в один спринт[18] (или итерацию). Истории, которые слишком велики для реализации в одной итерации (то есть представляют собой крупный этап работы), называют эпиками (epic). Эпики необходимо разбивать на группы более мелких историй (таким образом, эпики помогают создавать иерархическую структуру). Эпики, объединенные общей целью, называют инициативами (initiative).

---

[16] То есть на математическом или любом другом формализованном языке
[17] Данный шаблон носит наименование «Connextra» (по названию компании, в которой он был придуман) — существуют и другие шаблоны истории, но они менее популярны
[18] Спринт — короткий временной интервал (обычно 1-2 недели, реже 3-4 недели), в течение которого scrum-команда выполняет заданный объем работы. Scrum — один из фреймворков гибкой разработки

Если обозначить временные границы, то история — это часть работы (одна или несколько задач[19]), которую можно выполнить за спринт продолжительностью одну или две недели. Эпик можно завершить за месяц или квартал. Для выполнения инициативы, как правило, требуется от нескольких кварталов до года. Разумеется, это немного условное представление, но как правило, эпики охватывают несколько спринтов и версий приложения. Например, реализация функций личного кабинета для электронного журнала манги является эпиком (в рамках другой предметной области это может быть инициативой). Команда разработки или владелец продукта могут структурировать стратегию выполнения данного эпика относительно версий приложения следующим образом:

- Версия 1: вход в систему, выход из системы, изменение и восстановление пароля;
- Версия 2: платная подписка, история просмотра журнала манги, добавление в избранное;
- Версия 3: изменение и сохранение настроек профиля;
- Et cetera…

Разбиение эпиков на более мелкие пользовательские истории (рис. 3.1) часто называют декомпозицией[20] истории (story decomposition).

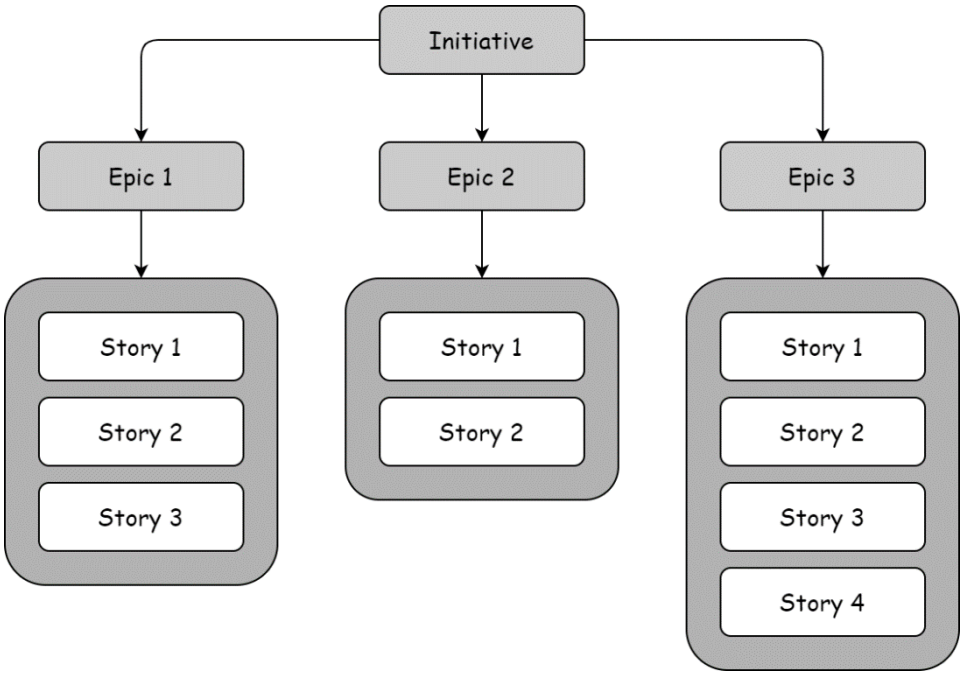


Рисунок 3.1 — Пример декомпозиции историй

В процессе декомпозиции достаточно часто получаются истории, которые больше напоминают технические задачи или архитектурные компоненты. В этом случае, как правило, требуется дополнительная декомпозиция. Бывает, что и она не помогает. Для этого был введен акроним INVEST, чтобы определить критерии хорошо декомпозированной пользовательской истории. Это не означает, что история должна строго соответствовать всем критериям (все-таки истории пишутся не ради историй), но ориентироваться на указанные критерии для улучшения историй необходимо.

[19] Некоторые команды не разбивают истории на задачи. Например, для истории поиска по веб-сайту (см. 3.4.2) отдельной задачей может быть «Добавить кнопку «Найти» для строки поиска»
[20] Аналогичная операция осуществляется при разбиении диаграммы потоков данных (DFD)

Критерии хорошей пользовательской истории:

- *I — Independent* (независимая)

  История не зависит от выполнения других историй. Вообще не все истории могут пройти проверку по данному критерию, так как иногда довольно сложно добиться полного отсутствия зависимостей. Поэтому в целом чем меньше зависимостей у истории, тем лучше. В идеале каждая история может быть реализована, протестирована и задеплоина[21] (поставлена в продукт) по отдельности;

- *N — Negotiable* (обсуждаемая)

  Как было отмечено ранее, история является обсуждаемым представлением требования, а не четкой его фиксацией, что подразумевает возможность непредвиденных изменений. Таким образом, история представляет собой некоторое компромиссное решение (команды разработки и бизнеса) между функциональностью и датой релиза. История отвечает данному критерию, если она прошла процедуру обсуждения и уточнения, при этом детали истории добавлены через сотрудничество с заказчиком;

- *V — Valuable* (несет ценность для бизнеса)

  Реализация истории принесет определенную ценность заказчику и/или пользователю. Данный критерий является наиболее важным атрибутом, именно он задает приоритет историям — *чем ценность для бизнеса выше, тем раньше история уйдет в разработку*;

- *E — Estimable* (обладает достаточной информацией для оценки)

  Можно оценить время реализации (см. 3.4.2);

- *S — Small* (обладает компактным размером)

  Можно реализовать за один спринт (итерацию). Чем меньше объем каждой истории, тем точнее можно выполнить оценку и определить приоритет;

- *T — Testable (тестируемая)*

  Существуют понятные *критерии приемки (Acceptance Criteria)* и тестовые сценарии для проверки реализации.

Стадия защиты лабораторной работы будет некоторой имитацией обсуждения историй, если у преподавателя возникнут вопросы и/или замечания.

[21] Deployment — развертывание программного обеспечения, то есть выполнение последовательности действий, которые приводят программную систему в состояние готовности для использования (для веб-приложений это, как правило, перемещение исходного кода на сервер, где он будет работать)

3.4.2 Критерии приемки

Критерии приемки представляют собой формализованный список требований, обеспечивающий завершение всех пользовательских историй и учет сценариев их выполнения. Иными словами, критерии приемки определяют условия, при которых выполняется пользовательская история. Кратко написанные критерии помогают команде разработчиков избежать двусмысленности в требованиях заказчика и предотвратить недопонимание.

Для чего необходимы критерии приемки:

- Определение границ пользовательской истории;
- Синхронизация ожиданий бизнеса и команды разработчиков;
- Служат основой для позитивного и негативного тестирования[22];
- Сценарии критериев приемки помогают правильно разделить истории на задачи, что позволяет выполнить точную оценку и планирование.

Существует несколько типов критериев приемки. Наиболее популярными являются ориентированные на правила (в виде списка) и сценарии.

Тип критериев, ориентированный на сценарии, предусматривает различные варианты использования и в дальнейшем служит основой для написания ручных и автоматизированных приемочных тестов[23].

Наиболее распространенным шаблоном для написания критериев приемки, ориентированных на сценарии, является формат Given-When-Then:

- Given (дано) — ситуация выглядит таким образом: есть некоторое состояние до того, как пользователь вошел в сценарий;
- When (когда) — затем пользователь совершает некоторые действия;
- Then (тогда) — теперь ситуация выглядит по-другому: система реагирует на действия пользователя.

Пример для функции поиска по веб-сайту:

> *Как* посетитель веб-сайта
> *Я хочу* иметь возможность выполнить поиск по строке текста
> *Чтобы* найти разделы сайта, в которых находится интересующая меня информация

> *Сценарий:* Пользователь ищет информацию на сайте по строке текста
>
> *Поскольку* я нахожусь в роли зарегистрированного пользователя или гостя, и система показывает мне строку поиска в правом верхнем углу экрана на всех страницах, кроме страниц регистрации, авторизации и подписки
>
> *Когда* я заполняю поле поиска строкой текста, нажимаю кнопку «Найти» или клавишу ENTER на клавиатуре

[22] Позитивное тестирование подразумевает применение сценариев, которые соответствуют нормальному (ожидаемому) поведению системы — в ЛР 2 это основные (или альтернативные) потоки выполнения; негативное тестирование — выполнение сценариев с неверными данными или исключениями

[23] Приемочное (acceptance) тестирование представляет собой финальный этап тестирования продукта перед его релизом (как правило, проверяются основные функции продукта, то есть приемочное тестирование не является полным и тщательным)

Тогда система открывает мода́льное окно[24], где показывает результаты поиска, в которых есть совпадения с введённой мной строкой текста. Результаты могут включать в себя номера журналов, мангу и новостные статьи. Все результаты упорядочены по важности (журналы идут в начале списка). Также система отображает количество найденных результатов поиска в левой верхней части модального окна. В правой верхней части окна система показывает кнопку «Закрыть» и ниже фильтр, который позволяет упорядочить результаты поиска по важности и по дате (в начале новые).

Сценарий для ошибочной ситуации: Пользователь ищет информацию на сайте по недействительной строке текста[25]

Поскольку я нахожусь в роли зарегистрированного пользователя или гостя

Когда я заполняю поле поиска недействительной строкой текста, нажимаю кнопку «Найти» или клавишу ENTER на клавиатуре

Тогда система открывает модальное окно, где показывает сообщение «Нет подходящих результатов».
Система предлагает выполнить поиск с помощью Google

Обратите внимание, что критерии приемки не должны описывать каждую деталь[26]. Точнее они не детализируются в самом начале проекта, избегая таким образом слишком ранней определенности, чрезмерно ограниченной формулировки решения, нагромождения требований и сопутствующих задержек в разработке. Спецификация вариантов использования (ЛР 2) предполагает высокую степень детализации требований. В свою очередь истории, как основной метод определения потребностей пользователей для команд гибкой разработки, подразумевают возможность непредвиденных изменений, соответственно, если мы имеем дело с требованиями, которые подвержены частым изменениям, то нам нет никакого смысла подробно их детализировать до непосредственного момента реализации. Как следствие, истории нуждаются в минимальном сопровождении или вообще его не требуют.

Также обратите внимание, что сценарий для критериев приемки написан от первого лица с использованием личного местоимения — это вполне допустимо (таким образом, мы говорим с точки зрения пользователя). Спецификация вариантов использования, как правило, пишется в безличной форме, поскольку является частью формализованного документа. Но и варианты использования и критерии приемки пользовательских историй пишутся на повседневном языке пользователя для облегчения взаимодействия между заинтересованными сторонами проекта. Поэтому узко специализированные термины и жаргонные слова или выражения не должны использоваться как при написании историй, так и вариантов использования. Например, если посмотреть на историю выше, словосочетание «модальное окно» может быть предметом обсуждения (стоит ли его использовать) — если в процессе

[24] Разновидность всплывающего окна, которое появляется на бóльшую часть экрана (или страницы) и блокирует работу с остальным сайтом. Как правило достаточно нажать на область вне модального окна, чтобы его закрыть (критерии приемки для данной истории предусматривают также наличие отдельной кнопки «Закрыть»)

[25] С точки зрения пользователя, отсутствие результатов поиска, является ошибкой

[26] Например, в сценарии для поиска по веб-сайту не прописано, что собой должны представлять сами результаты поиска (это просто список разделов или каждый результат должен включать фрагмент текста, где присутствует найденная строка — решение будет принято позже, возможно, непосредственно перед реализацией истории)

обсуждения мы понимаем, что с этим термином могут быть проблемы, лучше от него отказаться и заменить исходную строку на «Затем система открывает новое окно поверх страницы сайта».

Тип критериев приемки, ориентированный на правила, для истории поиска по веб-сайту выглядел бы следующим образом:

1. В правом верхнем углу страницы отображается строка поиска с кнопкой «Найти»;
2. Строка поиска есть на каждой странице, кроме страниц регистрации, авторизации и подписки;
3. Результаты поиска отображаются в модальном окне;
4. Результаты поиска могут включать в себя номера журналов, мангу и новостные статьи;
5. Все результаты поиска упорядочены по важности (журналы идут в начале списка);
6. В левой верхней части модального окна показано количество найденных результатов поиска;
7. В правой верхней части окна находится кнопка «Закрыть»;
8. В правой верхней части окна под кнопкой «Закрыть» находится фильтр, который позволяет упорядочить результаты поиска по важности и по дате (в начале новые);
9. В случае отсутствия результатов поиска в модальном окне отображается сообщение «Нет подходящих результатов» и предлагается выполнить поиск в Google.

Тип критериев, ориентированный на сценарии, удобнее в использовании, так как написан причинно-следственным образом. Тестировать сценарии также легче, чем набор правил. Однако преподаватель не будет настаивать на использовании именно сценариев (студент вправе выбрать удобный для него тип критериев).

Пример для формы обратной связи:

Как посетитель веб-сайта
Я хочу иметь возможность оставлять отзывы
Чтобы владелец системы мог учесть мои пожелания или замечания при будущих обновлениях системы

Сценарий: Пользователь отправляет отзыв о работе системы через форму обратной связи

Поскольку я нахожусь в роли зарегистрированного пользователя или гостя

Когда я открываю страницу для обратной связи

Тогда система показывает форму отправки отзыва, которая содержит «Адрес электронной почты», «Имя пользователя», «Комментарий» (все поля являются обязательными для заполнения)

Когда я заполняю указанные поля и нажимаю кнопку «Отправить отзыв»

Тогда система сохраняет мой отзыв и показывает всплывающее сообщение «Вы успешно отправили отзыв». Система очищает все поля формы обратной связи.

Обратите внимание на формулировку самой истории: если можно убрать часть истории и история при этом ничего не потеряет — значит эта часть бесполезна и от неё можно смело отказаться.

Пример для комментирования номера журнала:

> *Как* зарегистрированный пользователь
> *Я хочу* иметь возможность оставлять комментарии
> *Чтобы* высказать своё мнение относительно конкретного номера журнала

> *Сценарий:* Авторизованный пользователь оставляет комментарий к номеру журнала
>
> *Поскольку* я нахожусь в роли зарегистрированного пользователя
>
> *Когда* я открываю страницу конкретного номера журнала
>
> *Тогда* система показывает раздел «Комментарии» и список комментариев от других пользователей. Также система показывает поле «Добавить комментарий» в верхней части раздела комментариев[27]
>
> *Когда* я заполняю поле «Добавить комментарий» и нажимаю кнопку «Отправить»
>
> *Тогда* система сохраняет мой комментарий и показывает его в верхней части раздела «Комментарии».
> Система показывает мое имя и изображение профиля (аватар) слева от моего комментария.
> Также система показывает значки «Удалить» и «Изменить» справа от моего комментария.

Обратите внимание, в истории после слова «Как» мы указываем *конкретный тип пользователя*, а не просто «пользователь» — все, кто использует систему, являются её пользователями, поэтому нам необходимо точно определить тип (или категорию) пользователя, который выполняет данную историю. Поскольку только вошедшие в систему пользователи могут добавлять комментарии, то мы конкретно указываем, что это «авторизованный пользователь». Основная ошибка при работе с историями — это игнорирование роли, которую пользователь играет в системе, например, у нас может быть история «удаление устаревших или ошибочных объявлений» — необходимо учитывать, что для админа и рекламодателя это разные истории с разными ожиданиями и требованиями к системе.

Вообще есть еще один способ написания критериев приемки, который иногда встречается — в формате историй. Данный тип критериев приемки похож на список правил, но обладает более избыточной формой записи. Тем не менее, при выполнении работы, вы также можете его использовать.

[27] Обратите внимание, что здесь нет никаких деталей о том, как система отображает информацию о номере журнала, так как это другая история (каждая история по возможности должна быть независимой)

Например, для истории поиска по веб-сайту:

> Как зарегистрированный пользователь или гость я могу видеть в правом верхнем углу страницы строку поиска с кнопкой «Найти»
>
> Как … я могу видеть результаты поиска в отдельном модальном окне
>
> Как … я могу видеть в результатах поиска номера журналов, мангу и новостные статьи
>
> Как … я могу видеть в левой верхней части модального окна количество найденных результатов поиска
>
> Как … я могу видеть в правой верхней части окна кнопку «Закрыть»
>
> Как … я могу видеть в правой верхней части окна под кнопкой «Закрыть» фильтр, чтобы упорядочить результаты поиска по важности и по дате (в начале новые)
>
> Обработка ошибок:
> 1. В случае отсутствия совпадений с заданной строкой текста, система выводит сообщение «Нет подходящих результатов» и предлагает выполнить поиск с помощью Google.

> Технические заметки:
> 1. Строка поиска доступна на каждой странице, исключая страницы регистрации, авторизации и подписки;
> 2. По умолчанию результаты поиска упорядочены по важности (в начале идут журналы).

Таким образом, критерии приемки помогают команде разработки точно знать, что им нужно сделать, но при этом держат заказчика в курсе процесса разработки и позволяют ему проверять, разработанное программное обеспечение, на соответствие актуальным бизнес-требованиям. Кроме того, критерии приемки служат основой для сценариев использования и тестов, которые гарантируют достижение бизнес-целей и создание приложения без ошибок (на этих словах у преподавателя даже скупая слеза потекла по щеке).

Общие выводы к разделу:

- При написании истории необходимо концентрироваться на ценности для пользователя, а не на структурном или функциональном разбиении на задачи и технические компоненты;
- Проверяя формулировку истории, особое внимание необходимо уделить «действию» и наличию потребности;
- Истории необходимо проверять на соответствие критериям INVEST;
- Критерии приемки каждой истории необходимо отдельно прорабатывать с владельцем продукта (при выполнении работы данный пункт имитирует процедура защиты);
- Определяя содержание истории, необходимо помнить, что при реализации данной истории пользователь должен почувствовать ценность результатов разработки настолько рано, насколько это возможно.

4. ЛАБОРАТОРНАЯ РАБОТА

« SRS ? »

(появится не позднее 24 ноября)

4.1 Цель работы

todo

4.2 Задание на работу

todo

4.3 Рекомендации и требования по содержанию отчета

todo

4.4 Теоретический материал

todo