МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Шаблон «Метаморфозы»**

Выполнили:

А.Н. Ширко, Д.А. Сидоренко, С.А. Махалин

Направление подготовки 01.03.02 Прикладная математика и информатика Курс 2 в

Руководитель:

доцент кафедры математического моделирования,

доцент, кандидат технических наук Н.В. Бессарабов

Краснодар

2022

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc104723878)

[Связь методов версионирования с “метаморфозами” 4](#_Toc104723879)

[Семантика данных шаблона “Метаморфозы” 5](#_Toc104723880)

[Обучение искусственного интеллекта на машинках 9](#_Toc104723881)

[“Робот-бабочка” 12](#_Toc104723882)

[“Книгопродукция” 15](#_Toc104723883)

[Заключение 19](#_Toc104723884)

Введение

Важным обстоятельством, при проектировании систем хранения информации, является проблема правильного выбора стратегии моделирования данных, т.е. выбора основной концепции, которая позволяла бы отобразить их информационное содержание и была бы достаточно гибкой с точки зрения возможных расширений. Под концепцией будем понимать основанную на целостных и систематизированных представлениях совокупность взглядов, позволяющих выражать определенный способ понимания или трактовки каких-либо предметов, событий, процессов или явлений, представляющих ту или иную информационную ценность. Следуя этому определению можно сказать, что моделирование данных, как собственно и сам выбор модели данных, является очень важным этапом, в процессе разработки базы данных, закладывающим основы понятийного аппарата, в терминах которого будет производиться работа с системой хранения. Конкретная модель данных, при этом, является своего рода шаблоном.

В работе рассматривается шаблон проектирования реляционной базы данных, сущности которых имеют или могут иметь какое-либо развитие. Шаблон “Метаморфозы” предназначен для представления сущностей проходящих некоторую последовательность стадий развития. Каждая может описываться своим набором атрибутов и своей структурой.

Связь методов версионирования с “метаморфозами”

Если мы работаем со сложной базой данных, то таблицы, как ни странно, в ней наименее интересны (хотя тоже должны быть под контролем). Гораздо сложнее приходится с бизнес логикой, которая содержится в триггерах, представлениях, пакетах и процедурах, и тому подобных объектах. В эти элементы постоянно вносятся правки, и жизненно необходимо знать, кто внёс правки, когда, желательно знать, зачем, и как бы нам это откатить до любого нужного состояния. Для этого и используется версионирование. Версионирование - это способ группировки и маркировки изменений, вызванных эволюцией системы. Говоря проще, это использование имен, для описания некоторого состояния программы. Чаще всего, в качестве имени выступает цифра, но может так же использоваться некоторое слово или смесь обоих.

При работе с метаморфозами, происходящей в нашей базе данных, возникает необходимость в использовании систем версионирования. Такая необходимость нужна потому, что шаблон “метаморфозы” является редко изменяющимся измерением, называемым Slowly changing dimensions(SCD) – то есть измерения, не ключевые атрибуты которых имеют тенденцию изменяться.

У SCD существует множество способов реализации, которые могут быть разбиты на различные типы. Как пример, в Git хранится почти всё: коммиты, содержимое файлов и их иерархия. С помощью объектов типа BLOB сохраняется только содержимое файла и его длина(размер). В объекте типа дерева (англ. tree) хранится иерархия файловой системы. Каждая запись представляет собой следующее:

**<права файла>** **<тип объекта>** **<sha1 объекта>** **<имя файла>**

а сам коммит (англ. commit) является ссылкой на объект tree, соответствующей корневой директории, и ссылку на родительский коммит.

Говоря о Git, в нём нет отдельного хранилища. Для того, чтобы перейти к нужному изменению, необходимо пройти по ссылкам на родителя из нужного коммита. Для того, чтобы просмотреть изменения отдельного файла, нужно проделать такую же работу. К тому же Git не хранит информацию о переименовании файла. Он может лишь с некоторым допущением определить, что файл переименован, и внести нужные изменения.

Это определяет Git, как хороший пример для подражания, даже в среде баз данных.

Семантика данных шаблона “Метаморфозы”

Всякая сущность в обсуждаемой предметной области обладает своим набором атрибутов. Эти атрибуты можно разбить на 2 группы:

* Ключевые – не изменяются на протяжении своего существования.
* Атрибуты состояния – могут менять своё состояние на разных этапах.

В рамках шаблона “Метаморфозы” нас в первую очередь интересует, очевидно, 2 группа, причем те, которые в выбранной семантике способны изменить описание варианта сущности. Рассмотрим её подробнее.

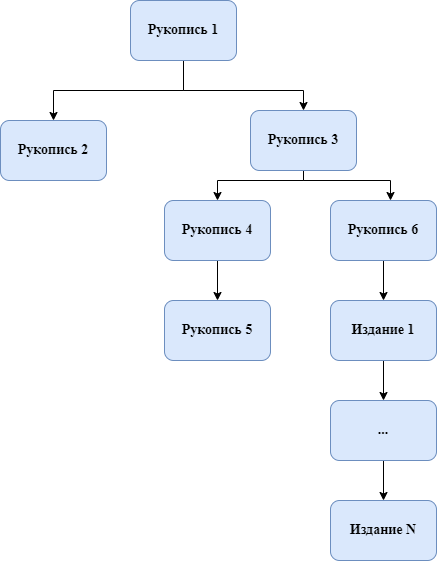
“Метаморфозы” предполагают под собой отслеживание преобразований рассматриваемой сущности. Однако не каждый её переменный атрибут отражает эти самые преобразования. В их определении и состоит главная задача архитектора БД на базе данного шаблона.

Предложим свой способ определения таких атрибутов:

1. *Выберем предметную область, в которой будем работать*. Данный шаг является основополагающим для любой БД. Определим какие сущности могут быть представлены. Какими атрибутами обладают. Выявляем взаимосвязи одних сущностей с другими.
2. *Определимся с “фазами метаморфоз”.* Приостановимся на этом этапе:

Мы определились с предметной областью. Теперь же нам необходимо выделить этапы, через которые может проходить сущность, испытывая “метаморфозы”. Эти этапы, назовем их “фазами метаморфоз”, неразрывно связаны с предметной областью, в которой находится рассматриваемая сущность. Причем нас интересуют конечные варианты сущностей на каждом из этапов, т.к. подразумевается, что изменения по сравнению с другими версиями на других этапах различны настолько, что мы можем назвать это преобразование “метаморфозой’.

К сожалению, нет способа явно выделить обобщенные “фазы” для любого рода сущностей, т.к. они связаны с особенностями предметных областей, в которых находятся. Так, если мы говорим о предметной области “книго-продукция”, то для сущности книги можно выделить следующие этапы:

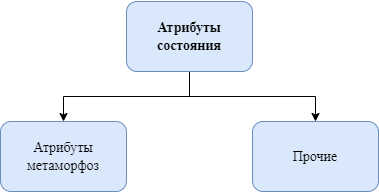


Дополнительные советы по поиску “фаз метаморфоз” выделим после 3 пункта.

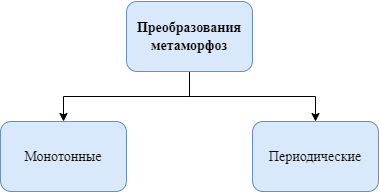
1. *Выбираем атрибуты, определяющие метаморфозы.* Разберем этот шаг подробнее:

Параметры, определяющие “метаморфозы”, обладают следующим общим признаком: они способны изменить положение сущности в предметной области, в которой эта сущность рассматривается. К примеру, *вес бойца определяет весовую категорию,* в которой он находится. Но если взять, скажем, силу удара бойца, то несмотря на то, что она может со временем меняться, её влияние не выходит за рамки *состояния бойца*. Назовём такие параметры “атрибутами метаморфоз”.

Таким образом можно разграничить атрибуты состояния на 2 группы атрибуты:



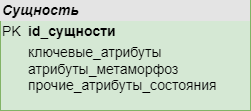
Теперь же поговорим о самих преобразованиях сущностей. Их можно разделить на 2 группы:



Под постоянными мы понимаем те преобразования, которые происходят непрерывно. В такой ситуации изменения в сущности происходят постоянно, то в качестве “системы метаморфоз” мы предлагаем взять некоторый временной интервал (зависит от предметной области с которой имеем дело), с которым мы будем фиксировать текущие состояния выбранных нами атрибутов сущности. В таких системах **обязательно** наличие темпоральных переменных. Пример таких преобразований: отнесение к категории по весу, развитие пчелки в яйце, развитие плодов овощных культур и т.д.

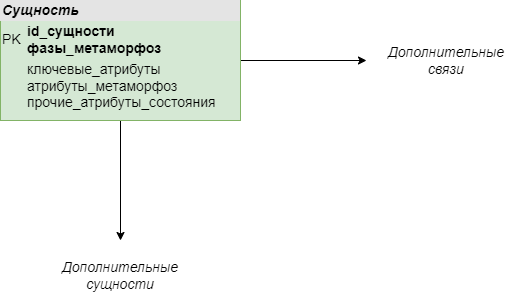
Периодические преобразования нельзя фиксировать по временным интервалам, т.к. нельзя предсказать следующее внесение изменений. Предлагается брать “систему” в виде “этапов разработки”: от прототипа1 к прототипу N, причем N – может быть неограничен. Тем самым мы фиксируем изменения прототипа, как некий законченный продукт, который, впрочем, ещё может перерабатываться. Пример таких преобразований: книго-продукция, совершенствование или разработка чего-либо, написание музыки и т.д.

Таким образом, структура данных сущностей, претерпевающих “метаморфозы”, будет выглядеть следующим образом:



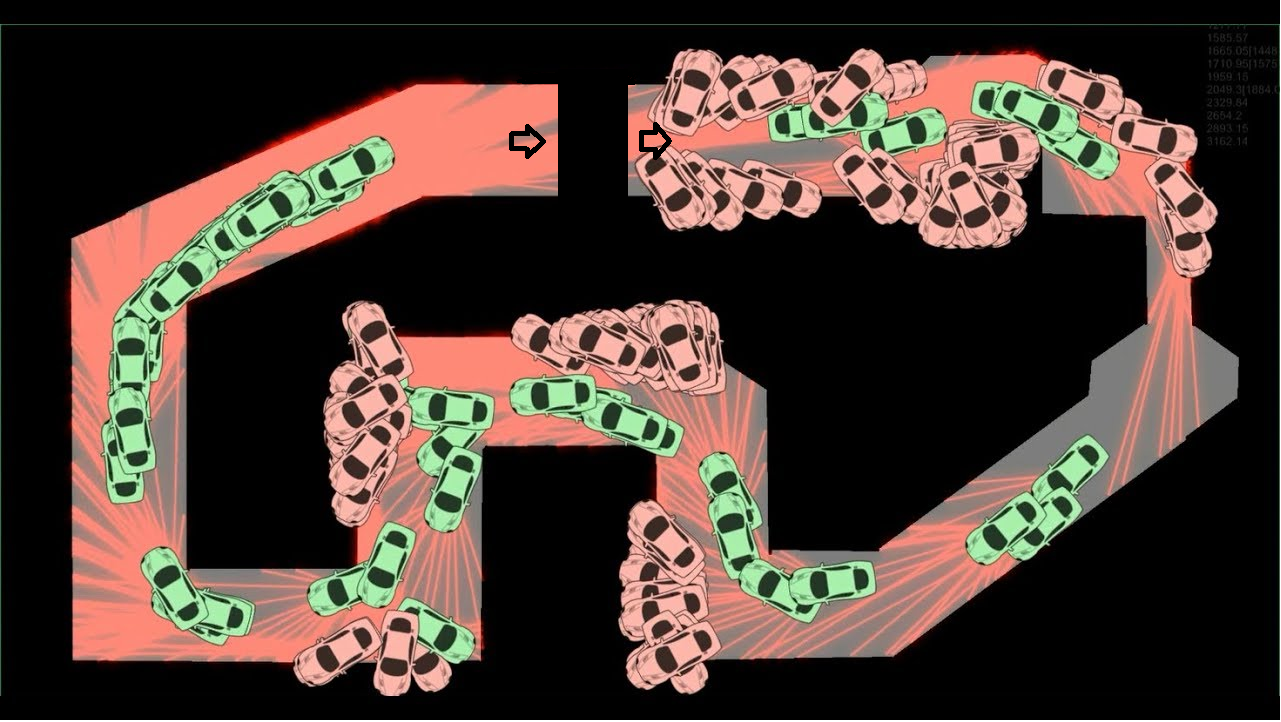
Стоит отметить, что атрибуты “метаморфоз” могут задаваться виде связей с другими сущностями. Мы увидим этот в одном из дальнейших примеров.

Тогда общий вид шаблона “метаморфоз” можно представить сделующим образом:



Обучение искусственного интеллекта на машинках

Пусть перед нами встала следующая задача: отследить “метаморфозы”, которые испытывает искусственный интеллект в ходе его обучения на базе симуляции “машинки”.



В компьютерной симуляции представлены различные варианты трасс, с препятствиями. Задача искусственного интеллекта: из начала трасы дойти до конца. Тест прекращается, когда одна из машин доедет до финиша.

Сложность тестов с плавно увеличивается, тем самым мы гарантируем, что с каждым новым тестом интеллект будет действительно развиваться, а не стагнировать (впрочем исключения возможны).

В ходе обучения интеллекта меняется код, по которому он действует. Программистами была создана только база ИИ. Дальнейшее вмешательство в код недопустимо.

Задача ясна. Приступим к созданию базы данных, отражающую “метаморфозы”:

1. *Выявим семантику*: существует некоторая главная сущность AI\_car (искусственный интеллект машины). В ней будем хранить успешный код, полученный на каждом из тестов.

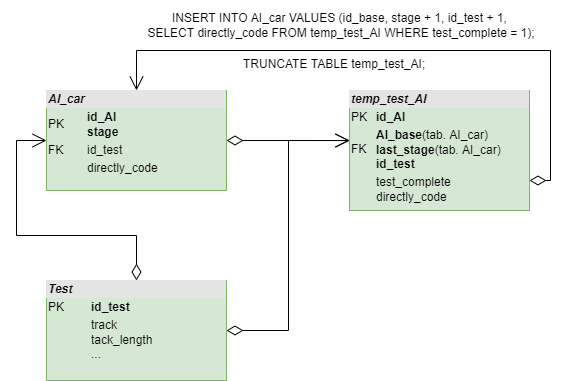
Также присутствует сущность temp\_test\_AI (текущая вариация AI), которая содержит все вариации AI на текущем тесте. После окончания

теста, вариация кода первого AI, который смог преодолеть трассу преодолеть трассу будет записана как новая “метаморфоза” в AI\_car, после чего temp\_test\_AI очищается.

Также добавим сущность Test(тест), которая хранит некоторые характеристики теста.

1. *Определимся с “фазами метаморфоз”.* В данном примере, очевидно, в качестве “фаз” можно взять множество пройденных тестов. По окончанию каждого из них будем получать более совершенный код ИИ, т.к. сложность плавно возрастает с каждым новым тестом.
2. *Выделим “атрибут метаморфоз”.* С точки зрения предметной области нас интересуют изменения в нашем искусственном интеллекте, следовательно, изменения эти видны в коде, который меняется от теста к тесту. Следовательно, сам код AI\_car и будет являться “атрибутом метаморфоз”. Действительно ведь именно преобразования кода дает возможность интеллекту переходить к новой “фазе”.

Таким образом строим БД:



Семантика данных:

Таблица AI\_car, как было сказано ранее, отражает изменения, происходящие в ИИ. Храним: id\_AI (идентификатор ИИ), stage (стадию / “фазу”), на которой находится наш ИИ, id\_test (внешний ключ) идентификатор теста, и directly\_code (собственно код). На каждой стадии изменения существенны настолько, что мы можем назвать это “метаморфозой”.

Таблица temp\_test\_AI хранит всевозможные вариации (id\_AI) преобразованных экземпляров нашего ИИ, созданных на базе AI\_base стадии last\_stage (внешние ключи от AI\_car). Атрибут id\_test (внешний ключ) идентификатор теста. Также содержит маркер test\_complete (закончен ли тест), который принимает значения:1 – если текущий экземпляр дошел до конца (такой экземпляр всего один), 0 – иначе. И наконец directly\_code (код). После того как тест объявляется законченным (т.е. один из экземпляров достиг финиша) создается новый запрос INSERT INTO AI\_car (…), в которой среди прочих атрибутов особую ценность представляет directly\_code, полученный от того экземпляра temp\_test\_AI, у которого test\_complete = 1. После этого таблица temp\_test\_AI очищается, т.к. прочие варианты нас не интересуют. Т.е. таблица постоянно перезаполняется.

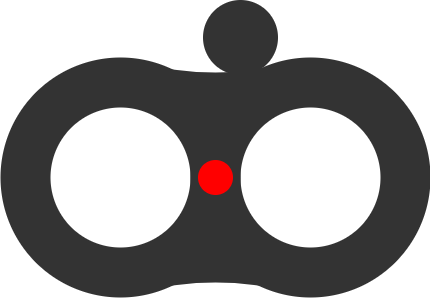
Таблица Test содержит некоторую информацию о тесте.

Все таблицы находятся как минимум в 3НФ.

Построенная БД отражает “метаморфозы”, происходящие с ИИ в ходе обучения на базе симуляции “машинки”.

“Робот-бабочка”

Рассмотрим модель задачи “Робота-бабочки”, симулируемой искусственным интеллектом.



Перед нами стоит задача научить искусственный интеллект перемещать мячик по траектории, задаваемой внешней стороной основного инструмента искусственного интеллекта, учитывая большое количество фактов. Для условности назовём этот инструмент “основным телом робота бабочки”.

Итак, искусственный интеллект имеет возможность крутить основное тело вокруг оси относительно центра в виде красного круга в середине. При этом перед ним не стоит задача лишь пропустить шарик по нужной траектории, а после уронить его. Она состоит в том, чтобы удерживать шарик как можно дольше на основном теле робота бабочки.

Создадим базу данных:

1. *Выявим семантику*: существует некоторая главная сущность ButterflyRB (искусственный интеллект). В ней будем хранить успешный код, полученный с каждой итерации.   
   Присутствует time\_retention (время, которое мячик держится на траектории). Он будет определятся по данным дополнительной сущности Ball (мяча), в которой будут хранится время начала его движения (time\_start) и время конца (time\_end). Из них и будет определять общее время в движении (time\_retention).
2. *Определимся с “фазами метаморфоз”.* С каждой итерацией, при том, что код (directly\_code) меняется, мы получаем новую фазу. Это и отметим.
3. *Выделим “атрибут метаморфоз”.* С точки зрения предметной области нас интересуют изменения в нашем искусственном интеллекте, следовательно, изменения эти видны в коде, который меняется от итерации к итерации. Тогда атрибутом будет являться код (directly\_code) нашей главной сущности ButterflyRB.

Рассмотрим схему данной модели.



Семантика данных:

Имеется главная сущность ButterflyRB, которая описывает некоторые характеристики: время удержания мячика до падения (time\_retention), код (directly\_code), а также id набора движений (id\_moveset – внешний ключ), id мяча (id\_ball – внешний ключ) и прочие.

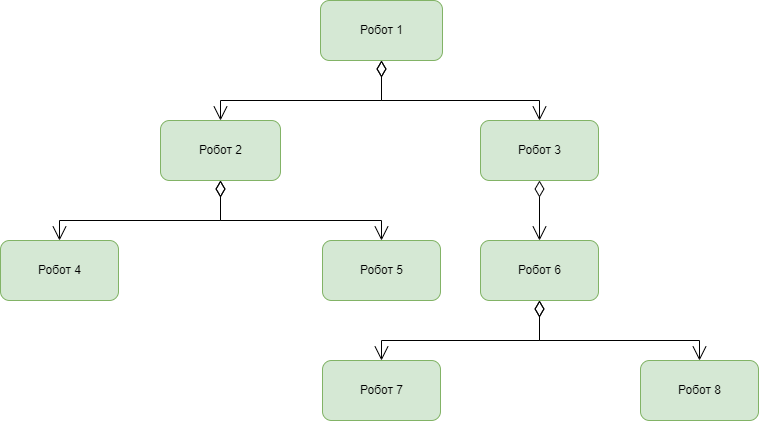
Таблица Moveset хранит информацию вариантов движения основного инструмента робота бабочки.

Таблица Ball хранит данные мяча: время начала движения (time\_start), время окончания движения (time\_end), пропорции (proportions) и т. д.

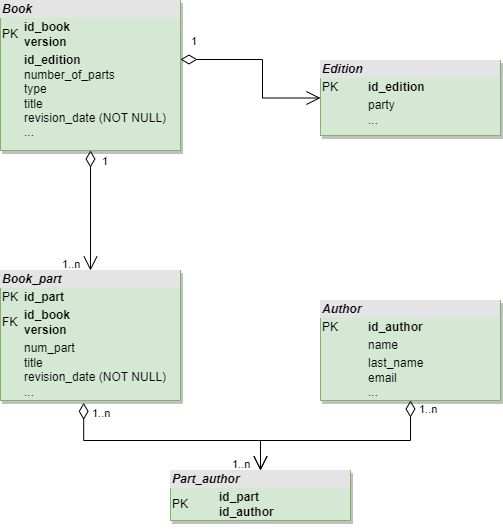
Рассмотрим получше stage, а также изменения, происходящие в сущности относительно рассматриваемой предметной области:

Искусственный интеллект, начиная свою деятельность, стремиться научиться удерживать шарик (time\_retention) как можно дольше на траектории, при этом не уронив его. Если шарик всё же упал, то он делает выводы на собственных параметрах, а далее меняет код (directly\_code) в соответствии с собранными данными. Определим stage как вариант сущности, которая прошла это изменение.

В таком случае можем получить дерево вида “родитель-ребёнок”, в котором получим множество связанных вариантов моделей.



“Книгопродукция”



Семантика данных:

Схема описывает этапы “развития” книги.

В таблице Book хранится информация о конкретной версии(version) книги(id\_book) на момент revision\_date. Количество томов (частей - number\_of\_parts) из которых состоит, тип(type) – наши “фазы”, например, “Рукопись”, название и остальные локальные атрибуты. Подразумевается, что если id\_edition не пуст (т.е. книги имеют издательство), то книга однажды выпускалась в продажу, поэтому такая версия книги конечная и её изменение приведет к созданию новой версии.

Таблица Book\_part – отдельные части книги (будем подразумевать тома). Тома могут изменяться от версии к версии книги, поэтому храним ключевые атрибуты таблицы Book. Тома могут редактироваться, учитываем это параметром revision\_date (ограничение целостности: revision\_date (Book\_part) <= revision\_date (Book), т.к. эта книга уже издавалась). И остальные локальные атрибуты. Отдельные изменения частей книги влияет на её законченность type, в которой она сейчас находится, следовательно, мы можем их назвать “подсущностями метаморфоз главной сущности Book”.

Таблица Edition – сведения об издательстве. Таблица является постоянной и необходима лишь для полноты картины, поэтому хранит для примера только партию (party).

Author – личная информация об авторе. Так же важна для примера.

Таблица Part\_author – определяет причастность автора к какой-либо главе. Так как каждая глава имеет свой уникальный номер, то легко определить причастность автора и к определённой книге.

Каждая из таблиц находится как минимум в 3НФ.

О преобразованиях в приведенных схемах:

В таблице Book в процессе “развития” книги может меняться количество томов(частей), тип сущности (например, при переходе от черновика к рукописи), название (на разных этапах может меняться).

Book\_part – во многом похожа на Book, за исключением собственной даты последнего изменения(revision\_date). Однако у каждой главы есть свой автор (авторы), поэтому в отдельной таблице Part\_author также возможны преобразования.

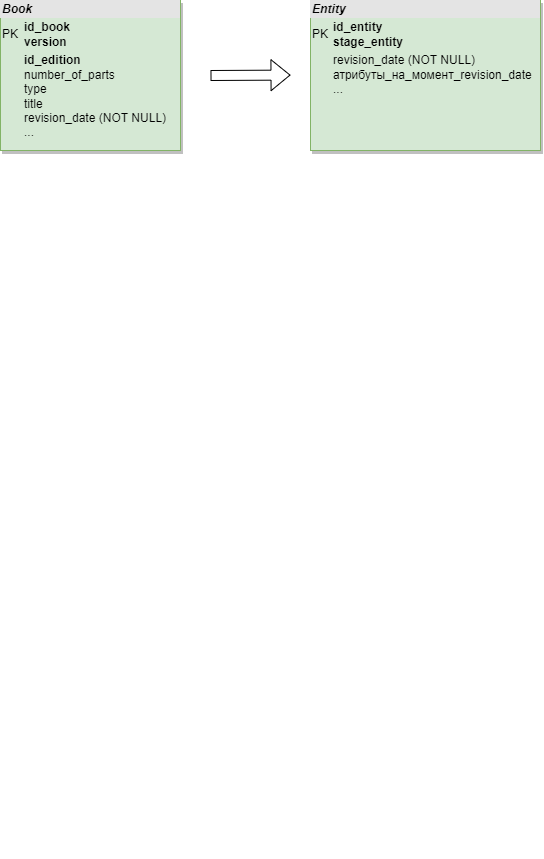
Условия метаморфоз выполняются. Данная схема отражает изменения, которые происходят в ходе написания, редакции или переиздания конкретной книги.

Пойдём по пути “от примера к шаблону”.

Будем обобщать схему.

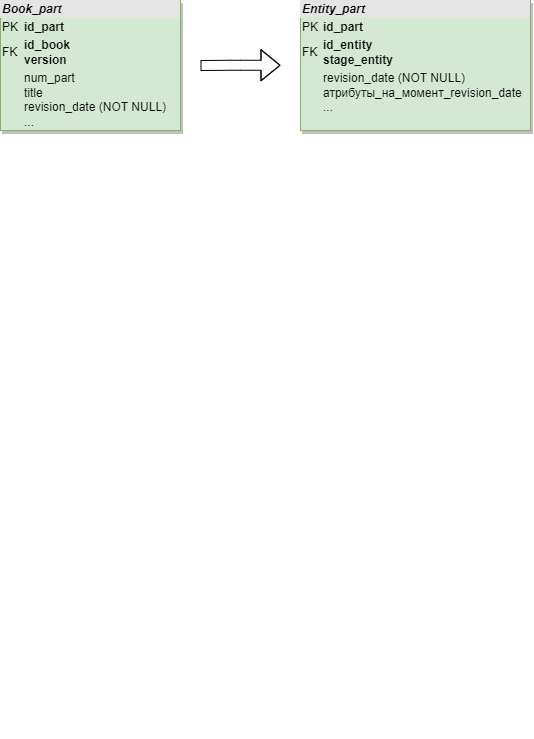
1. Для начала выделим главную сущность. В вышеприведенном примере это – Book. На её основе строится вся схема. Обобщим имя до Entity.
2. Её главным атрибутам дадим более обобщенные имена: id\_book станет id\_entity, version станет stage\_entity.
3. Важным параметром любой схемы подразумевающей “метаморфозы” является момент, на который данные конкретного экземпляра актуальны. Темпоральный атрибут revision\_date – это ни что иное как такой момент, он обязателен и не может быть пустым!
4. Остальные же параметры можно объединить в одну категорию: атрибуты актуальные на момент update.

Итого имеем:



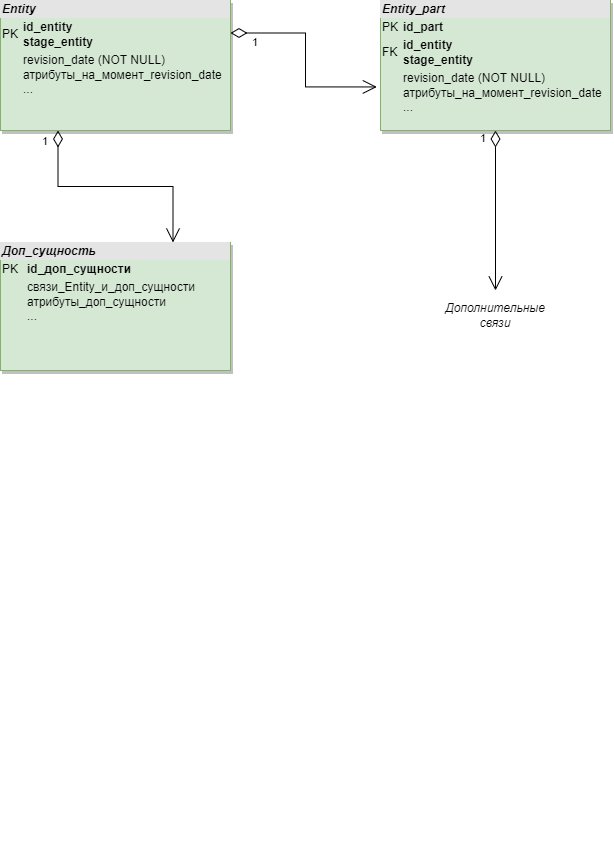
1. Процесс “метаморфоз” протекает планомерно, от одной стадии части сущности к другой. Поэтому выделим второстепенную главную сущность, зависящую от Book. Из схемы очевидно, что это Book\_part. Обобщим имя до Entity\_part.
2. Главный ключ id\_part сохранит своё название.
3. Так как Entity\_part является лишь частью от Entity, то необходимо ссылаться на эту главную часть. Т.е. в дополнительные атрибуты перейдут внешние ключи id\_entity и stage\_entity.
4. Аналогично пункту 3 остается параметр revision\_date. Замечание: для данного параметра возможны наложения дополнительных ограничений целостности в зависимости от требований бизнеса.
5. Аналогично пункту 4.

Итого имеем:



1. Остальные взаимосвязи и дополнительные таблицы могут добавляться по мере необходимости.

Таким образом получили прототип шаблона “Метаморфозы”:



Заметим, что данном примере мы “атрибуты метаморфоз” определили через связи между частями сущности и самой сущностью.

Заключение

Выбор соответствующего шаблона модели данных – это по сути процесс выбора парадигмы работы с данными на основе модели предметной области и используемого уровня абстракции. Ценность применения любого из шаблонов состоит в том, что, освоившись с подходом, мы получаем в свое распоряжение множество приемов, в соответствии уровню сложности шаблона, позволяющих контролировать как сложность предметной области, так и сложность системы хранения, а также программного обеспечения для работы с данными. Разумеется, каким бы ни был выбранный подход он не заменит наличие опыта и стиля мышления, необходимого для правильного выбора стратегии работы над соответствующим проектом обработки и анализа данных.