ЧЕРЕПОВЕЦКИЙ
ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

Лекция 10. Проектирование ИС

Часть 1. Этапы разработки проекта: стратегия и анализ

Проектирование информационных систем всегда начинается с определения цели проекта. Основная задача любого успешного проекта заключается в том, чтобы на момент запуска системы и в течение всего времени ее эксплуатации можно было обеспечить:

- требуемую функциональность системы и степень адаптации к изменяющимся условиям ее функционирования;
- требуемую пропускную способность системы;
- требуемое время реакции системы на запрос;
- безотказную работу системы в требуемом режиме, иными словами - готовность и доступность системы для обработки запросов пользователей;
- простоту эксплуатации и поддержки системы;
- необходимую безопасность.

Производительность является главным фактором, определяющим эффективность системы. Хорошее проектное решение служит основой высокопроизводительной системы.

Проектирование информационных систем охватывает три основные области:

- проектирование объектов данных, которые будут реализованы в базе данных;
- проектирование программ, экранных форм, отчетов, которые будут обеспечивать выполнение запросов к данным;
- учет конкретной среды или технологии, а именно: топологии сети, конфигурации аппаратных средств, используемой архитектуры (файл-сервер или клиент-сервер), параллельной обработки, распределенной обработки данных и т.п.

Вопрос 1. Модели жизненного цикла ПО .

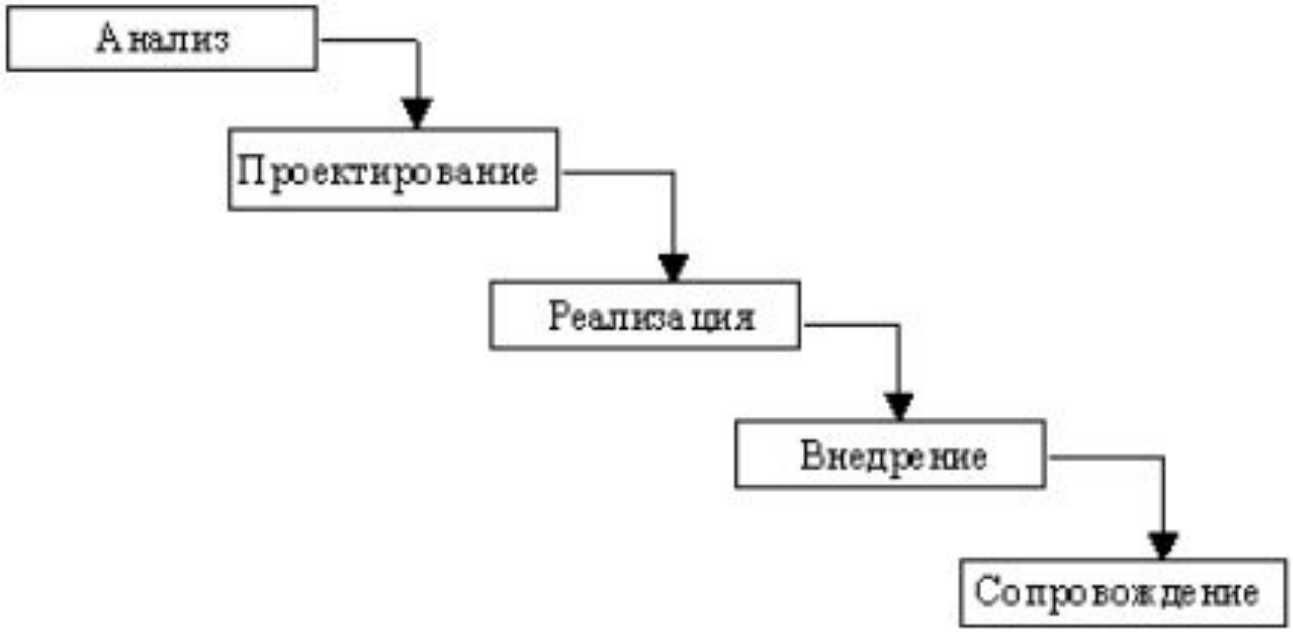
Под моделью ЖЦ понимается структура, определяющая последовательность выполнения и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ. Модель ЖЦ зависит от специфики ИС и специфики условий, в которых последняя создается и функционирует

Стандарт ISO/IEC 12207 не предлагает конкретную модель ЖЦ и методы разработки ПО. Описывает структуру процессов ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, включенные в эти процессы.

Известны следующие модели жизненного цикла:

- ***Каскадная модель***.
  Переход на следующий этап означает полное завершение работ на предыдущем этапе.
- ***Поэтапная модель с промежуточным контролем.***
  Разработка ПО ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют уменьшить трудоемкость процесса разработки по сравнению с каскадной моделью; время жизни каждого из этапов растягивается на весь период разработки.
- ***Спиральная модель***.
  Особое внимание уделяется начальным этапам разработки - выработке стратегии, анализу и проектированию, где реализуемость тех или иных технических решений проверяется и обосновывается посредством создания прототипов (макетирования). Каждый виток спирали предполагает создание некой версии продукта или какого-либо его компонента, при этом уточняются характеристики и цели проекта, определяется его качество и планируются работы следующего витка спирали.

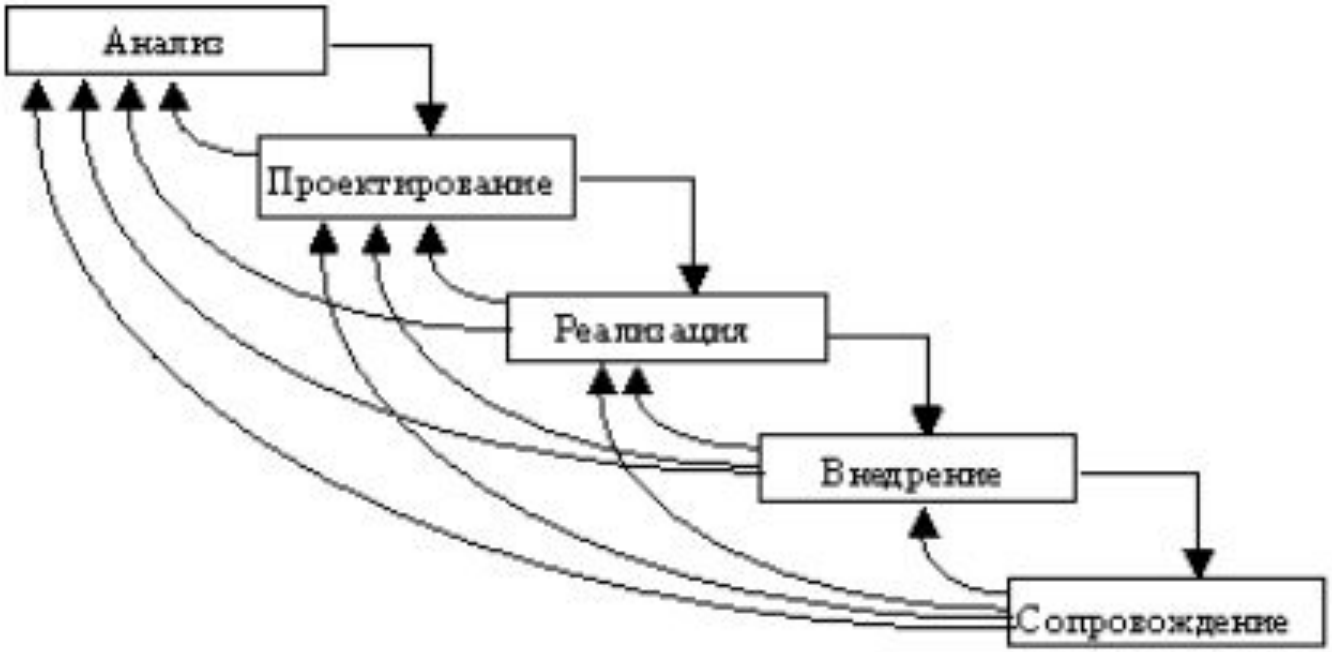
Каскадная схема разработки проекта



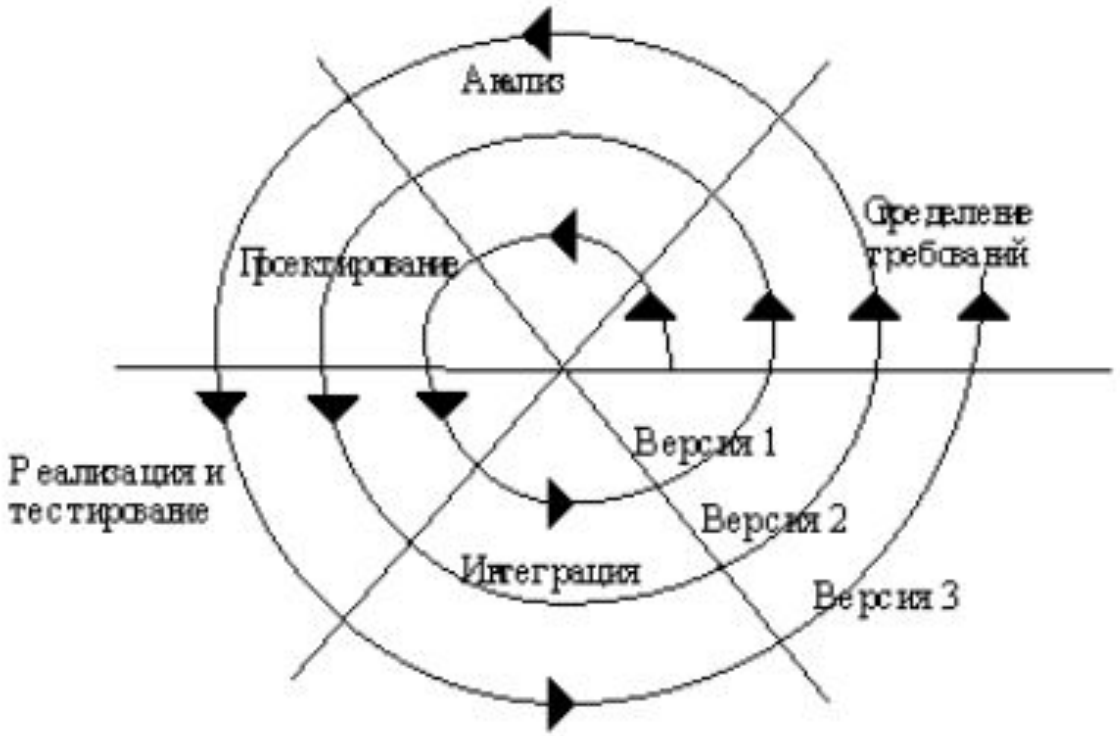
Положительные стороны применения каскадного подхода заключаются в следующем:

- на каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности;
- выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

"Водопад" - схема разработки проекта



Спиральная модель ЖЦ ПО



Вопрос 2. Стратегия

Определение стратегии предполагает обследование системы. Основная задача обследования - оценка реального объема проекта, его целей и задач, а также получение определений сущностей и функций на высоком уровне.
Этап предполагает тесное взаимодействие с основными пользователями системы и бизнес-экспертами.
Основная задача взаимодействия - получить как можно более полную информацию о системе (полное и однозначное понимание требований заказчика) и передать данную информацию в формализованном виде системным аналитикам для последующего проведения этапа анализа

Результатом этапа определения стратегии является документ, в котором :

- ограничения, риски, критические факторы, влияющие на успешность проекта, например время реакции системы на запрос является заданным ограничением, а не желательным фактором;
- совокупность условий, при которых предполагается эксплуатировать будущую систему: архитектура системы, аппаратные и программные ресурсы, предоставляемые системе, внешние условия ее функционирования, состав людей и работ, которые обеспечивают бесперебойное функционирование системы;
- сроки завершения отдельных этапов, форма сдачи работ, ресурсы, привлекаемые в процессе разработки проекта, меры по защите информации;
- описание выполняемых системой функций;

- будущие требования к системе в случае ее развития, например возможность работы пользователя с системой с помощью Интернета и т.п.;
- сущности, необходимые для выполнения функций системы;
- интерфейсы и распределение функций между человеком и системой;
- требования к программным и информационным компонентам ПО, требования к СУБД (если проект предполагается реализовывать для нескольких СУБД, то требования к каждой из них, или общие требования к абстрактной СУБД и список рекомендуемых для данного проекта СУБД, которые удовлетворяют заданным условиям);
- что не будет реализовано в рамках проекта.

Выполненная на данном этапе работа позволяет ответить на вопрос, стоит ли продолжать данный проект и какие требования заказчика могут быть удовлетворены при тех или иных условиях.
Может оказаться, что проект продолжать не имеет смысла, например из-за того, что те или иные требования не могут быть удовлетворены по каким-то объективным причинам.

Если принимается решение о продолжении проекта, то для проведения следующего этапа анализа уже имеются представление об объеме проекта и смета затрат.

Следует классифицировать планируемые функции системы по степени важности. Один из возможных форматов представления такой классификации - MoSCoW - предложен в Clegg, Dai and Richard Barker, Case Method Fast-track: A RAD Approach, Adison-Wesley, 1994.

Аббревиатура расшифровывается так:
Must have - необходимые функции;
Should have - желательные функции;
Could have - возможные функции;
Won't have - отсутствующие функции.

Функции первой категории обеспечивают критичные для успешной работы системы возможности.
Реализация функций второй и третьей категорий ограничивается временными и финансовыми рамками: разрабатываем то, что необходимо, а также максимально возможное в порядке приоритета число функций второй и третьей категорий.
Последняя категория функций особенно важна, поскольку необходимо четко представлять границы проекта и набор функций, которые будут отсутствовать в системе.

Вопрос 3. Этап анализа

Этап предполагает подробное исследование бизнес-процессов (функций, определенных на этапе выбора стратегии) и информации, необходимой для их выполнения (сущностей, их атрибутов и связей (отношений)).

На этом этапе создается ***информационная модель***, а на следующем за ним этапе проектирования - ***модель данных***.

Вся информация о системе, собранная на этапе определения стратегии, формализуется и уточняется на этапе анализа.

Особое внимание следует уделить:

- полноте переданной информации;
- анализу информации на предмет отсутствия противоречий;
- поиску неиспользуемой вообще или дублирующейся информации.

Как правило, заказчик не сразу формирует требования к системе в целом, а формулирует требования к отдельным ее компонентам. Уделите внимание согласованности этих компонентов.

Аналитики собирают и фиксируют информацию в двух взаимосвязанных формах:

- функции - информация о событиях и процессах, которые происходят в бизнесе;
- сущности - информация о вещах, имеющих значение для организации и о которых что-то известно.

Двумя классическими результатами анализа являются:

- иерархия функций, которая разбивает процесс обработки на составные части (что делается и из чего это состоит);
- модель "сущность-связь" (Entry Relationship model, ER-модель), которая описывает сущности, их атрибуты и связи (отношения) между ними.

Эти результаты являются необходимыми, но не достаточными.

К достаточным результатам следует отнести диаграммы потоков данных и диаграммы жизненных циклов сущностей. Довольно часто ошибки анализа возникают при попытке показать жизненный цикл сущности на диаграмме ER.

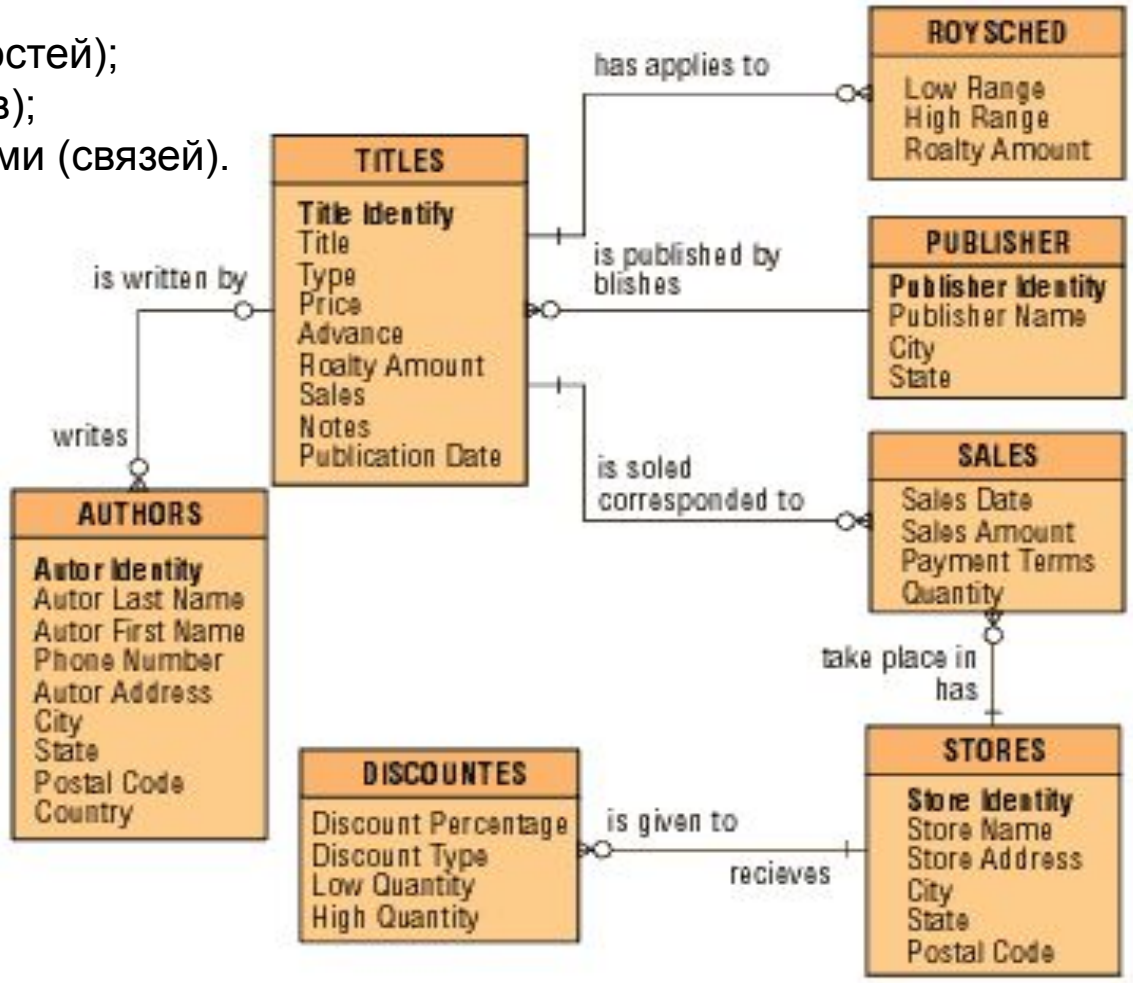
Наиболее часто применяются следующие методологии структурного анализа:

- диаграммы "сущность-связь" (Entity-Relationship Diagrams, ERD), которые служат для формализации информации о сущностях и их отношениях;
- диаграммы потоков данных (Data Flow Diagrams, DFD), которые служат для формализации представления функций системы;
- диаграммы переходов состояний (State Transition Diagrams, STD), которые отражают поведение системы, зависящее от времени; диаграммы жизненных циклов сущностей относятся именно к этому классу диаграмм.

3.1. ER-диаграммы

Представляют собой стандартный способ определения данных и отношений между ними. ER-диаграмма содержит информацию о сущностях системы и способах их взаимодействия, включает:

- идентификацию объектов (сущностей);
- свойств этих объектов (атрибутов);
- их отношений с другими объектами (связей).



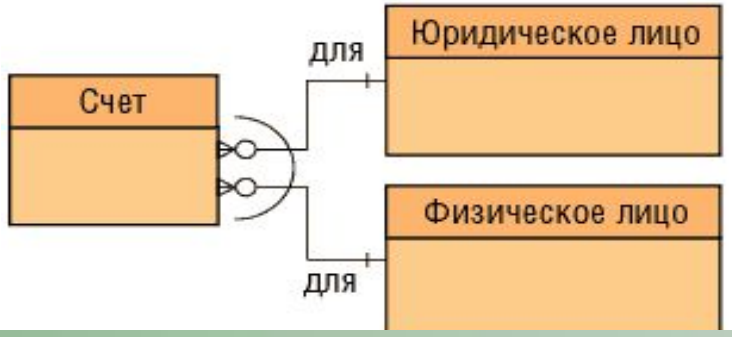
Отношение изображается линией между двумя сущностями.
Одиночная линия справа означает "один", "птичья лапка" слева - "многие",

Пример изображения рефлексивного отношения "сотрудник", где один сотрудник может руководить несколькими подчиненными и так далее вниз по иерархии должностей

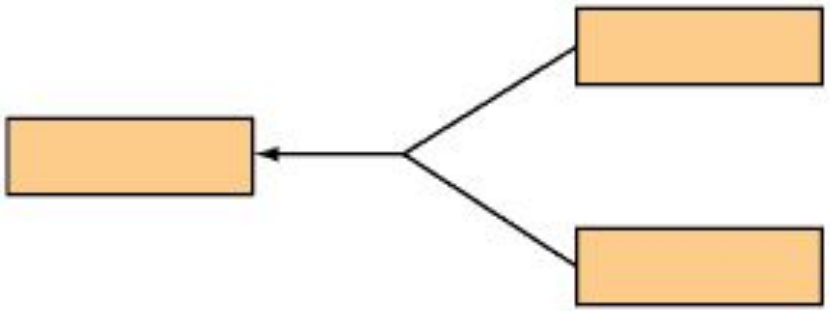
3.2. Дуги

Если сущность имеет набор взаимоисключающих отношений с другими сущностями, то говорят, что такие отношения находятся в дуге. Например, банковский счет может быть оформлен или для юридического лица, или для физического лица. Фрагмент ER-диаграммы для такого типа отношений приведен на рисунке.

Полученные в результате сущности называют подтипами, а исходная сущность становится супертипом. Чтобы понять, нужен супертип или нет, надо установить, сколько одинаковых свойств имеют различные подтипы.



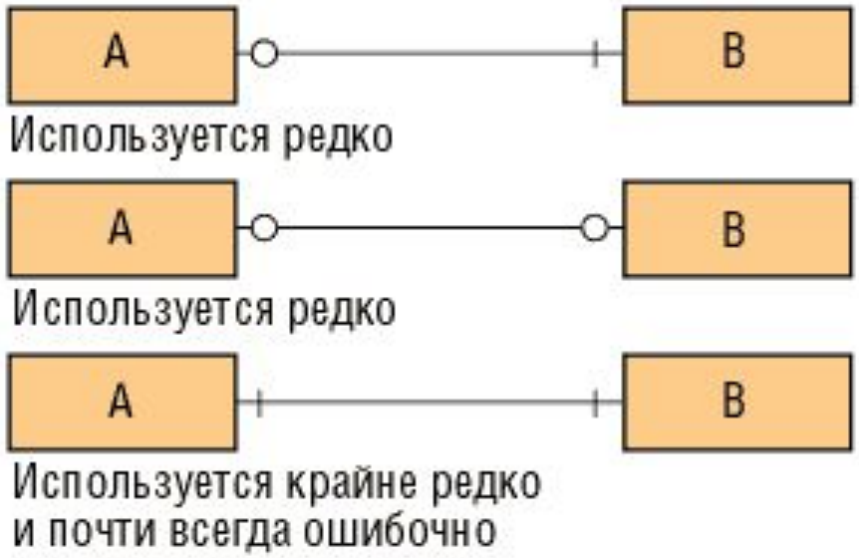
Следует отметить, что злоупотребление подтипами и супертипами является довольно распространенной ошибкой.
Изображают их так, как показано на рисунке



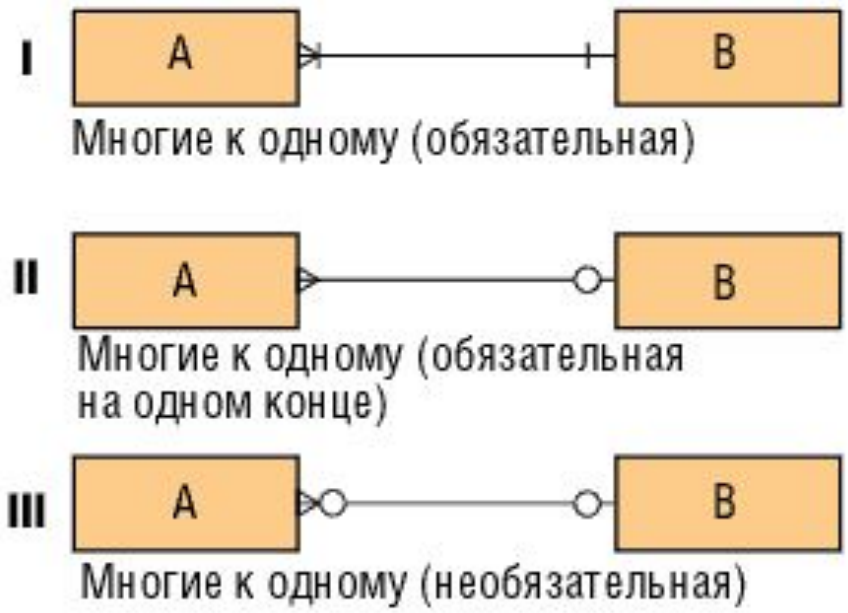
3.3. Допустимые типы связей.

Связи типа "один к одному"

почти всегда оказывается, что A и B представляют собой в действительности разные подмножества одного и того же предмета или разные точки зрения на него, просто имеющие отличные имена и по-разному описанные связи и атрибуты.



Связи "многие к одному"



I - предполагает, что вхождение сущности B не может быть создано без одновременного создания, по меньшей мере, одного связанного с ним вхождения сущности A.

II - наиболее часто встречающаяся форма связи. Предполагает, что каждое и любое вхождение сущности A может существовать только в контексте одного (и только одного) вхождения сущности B. В свою очередь, вхождения B могут существовать как в связи с вхождениями A, так и без нее.

III - применяется редко. Как A, так и B могут существовать без связи между ними.

Связи "многие ко многим"

I

Многие ко многим (необязательная)

II

Многие ко многим (обязательная на одном конце)

I - такая конструкция часто имеет место в начале этапа анализа и означает связь - либо понятую не до конца и требующую дополнительного разрешения, либо отражающую простое коллективное отношение - двунаправленный список.

II - применяется редко. Такие связи всегда подлежат дальнейшей детализации.

Недопустимые типы связей.

К недопустимым типам связей относятся следующие:

3.4. Диаграммы потоков данных

Логическая DFD показывает внешние по отношению к системе источники и стоки (адресаты) данных, идентифицирует логические функции (процессы) и группы элементов данных, связывающие одну функцию с другой (потоки), а также идентифицирует хранилища (накопители) данных, к которым осуществляется доступ.

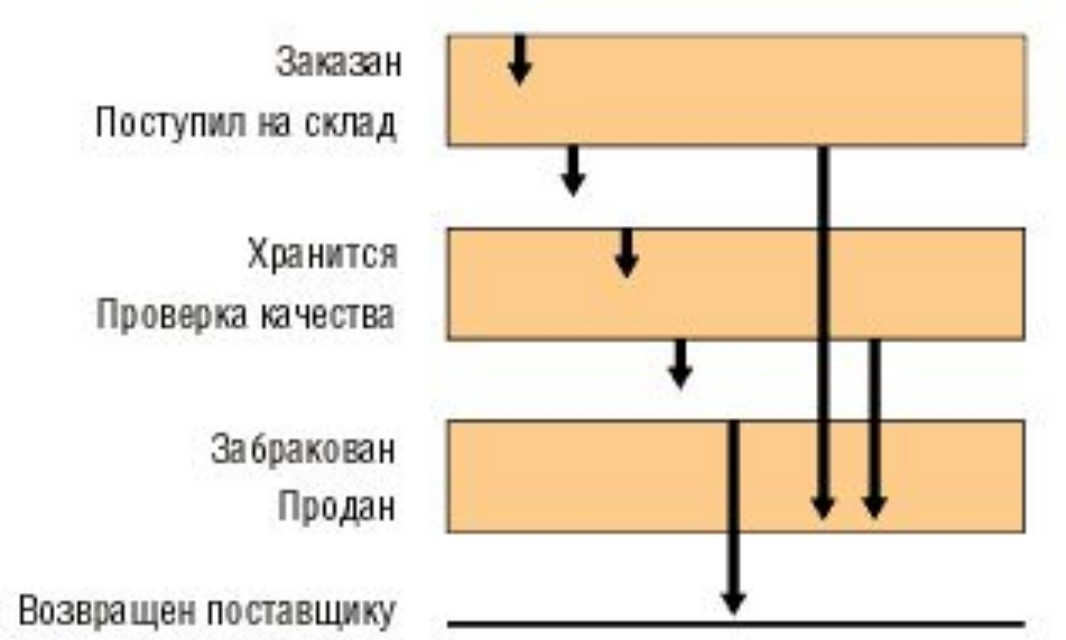
Каждая логическая функция (процесс) может быть детализирована с помощью DFD нижнего уровня; когда дальнейшая детализация перестает быть полезной, переходят к выражению логики функции при помощи спецификации процесса (мини-спецификации). Модель данных хранилища раскрывается с помощью ER-диаграмм.

DFD :

- позволяют представить систему с точки зрения данных;
- иллюстрируют внешние механизмы подачи данных, требующие наличия специальных интерфейсов;
- позволяют представить как автоматизированные, так и ручные процессы системы;
- выполняют ориентированное на данные секционирование всей системы.

3.5. Диаграммы изменения состояний STD

Жизненный цикл сущности относится к классу STD-диаграмм



Отражает изменение состояния объекта с течением времени.
Например, рассмотрим состояние товара на складе: товар может быть заказан у поставщика, поступить на склад, храниться на складе, проходить контроль качества, может быть продан, забракован, возвращен поставщику.
Стрелки на диаграмме показывают допустимые изменения состояний.

3.6. Качество сущностей

Список проверочных вопросов для сущности, ответ на которые даст ответ, действительно ли объект является сущностью, то есть важным объектом или явлением, информация о котором должна храниться в базе данных:

- отражает ли имя сущности суть данного объекта?
- нет ли пересечения с другими сущностями?
- имеются ли хотя бы два атрибута?
- есть ли синонимы/омонимы данной сущности?
- сущность определена полностью?
- есть ли уникальный идентификатор?
- имеется ли хотя бы одна связь?
- существует ли хотя бы одна функция по созданию, поиску, корректировке, удалению, архивированию и использованию значения сущности?
- ведется ли история изменений?
- имеет ли место соответствие принципам нормализации данных?
- нет ли такой же сущности в другой прикладной системе, возможно, под другим именем?
- не имеет ли сущность слишком общий смысл?
- достаточен ли уровень обобщения, воплощенный в ней?

3.7. Качество атрибутов

Следует выяснить, а действительно ли это атрибуты, то есть, описывают ли они тем или иным образом данную сущность.
Список проверочных вопросов для атрибута:

- является ли наименование атрибута существительным единственного числа, отражающим суть обозначаемого атрибутом свойства?
- не включает ли в себя наименование атрибута имя сущности (этого быть не должно)?
- имеет ли атрибут только одно значение в каждый момент времени?
- отсутствуют ли повторяющиеся значения (или группы)?
- описаны ли формат, длина, допустимые значения, алгоритм получения и т.п.?
- не может ли этот атрибут быть пропущенной сущностью, которая пригодилась бы для другой прикладной системы?
- не может ли он быть пропущенной связью?
- есть ли необходимость в истории изменений?
- зависит ли его значение только от данной сущности?
- если значение атрибута является обязательным, всегда ли оно известно?
- есть ли необходимость в создании домена для этого и ему подобных атрибутов?
- зависит ли его значение только от какой-то части уникального идентификатора?
- зависит ли его значение от значений некоторых атрибутов, не включенных в уникальный идентификатор?

3.8. Качество связей

Список проверочных вопросов для связи:

- имеется ли ее описание для каждой участвующей стороны, точно ли оно отражает содержание связи и вписывается ли в принятый синтаксис?
- участвуют ли в ней только две стороны?

Не относится ли конструкция связи к редко используемым?

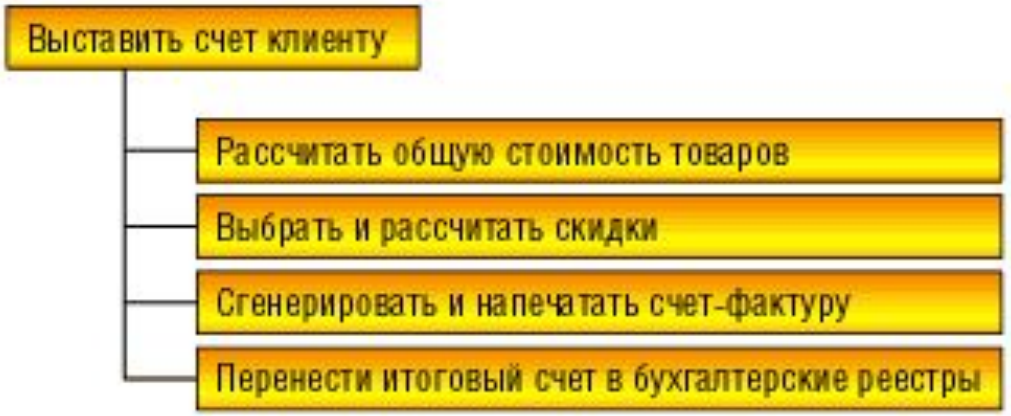
- не является ли она избыточной?
- не изменяется ли она с течением времени?
- если связь обязательная, всегда ли она отражает отношение к сущности, представляющей противоположную сторону?

Для исключающей связи:

- все ли концы связей, покрываемые исключающей дугой, имеют один и тот же тип обязательности?
- все ли из них относятся к одной и той же сущности?
- обычно дуги пересекают разветвляющиеся концы - что вы можете сказать о данном случае?
- связь может покрываться только одной дугой. Так ли это?
- все ли концы связей, покрываемые дугой, входят в уникальный идентификатор?

3.9. Функции системы

При необходимости описывать достаточно сложные бизнес-процессы прибегают к функциональной декомпозиции, которая показывает разбиение одного процесса на ряд более мелких функций до тех пор, пока каждую из них уже нельзя будет разбить без ущерба для смысла.
Конечный продукт декомпозиции представляет собой иерархию функций, на самом нижнем уровне которой находятся атомарные с точки зрения смысловой нагрузки функции.
Простой пример такой декомпозиции.
Рассмотрим простейшую задачу выписки счета клиенту при отпуске товара со склада при условии, что набор товаров, которые хочет приобрести клиент, уже известен



Атомарные функции описываются подробно, например, с помощью DFD и STD.

На этапе анализа следует уделить внимание *функциям анализа и обработки возможных ошибок и отклонений* от предполагаемого эталона работы системы.

Следует выделить *наиболее критичные для работы системы процессы* и обеспечить для них особенно строгий анализ ошибок.
Обработка ошибок СУБД (коды возврата), как правило, представляет собой обособленный набор функций или одну-единственную функцию.

3.10 Уточнение стратегии

На этапе анализа происходит уточнение выбранных для конечной реализации аппаратных и программных средств.

Уточнение требований к СУБД.
При проектировании информационной системы важно учесть и дальнейшее развитие системы, например рост объемов обрабатываемых данных, увеличение интенсивности потока запросов, изменение требований надежности информационной системы.

Уточнение требований к средствам разработки.
Если выбранное на предыдущем этапе средство разработки не позволяет выполнить ту или иную часть работ в заданный срок, то принимается решение об изменении сроков (как правило, это увеличение срока разработки) или о смене средства разработки.

*Уточняются также ограничения, риски, критические факторы***.**