Лекция 3. Этап концептуального проектирования. Основные понятия концептуального проектирования: объекты, свойства, связи. Их типы.

На прошлой лекции мы вкратце определили порядок выполнения работ по созданию базы данных.

На первом этапе необходимо описать предметную область, причем сделать это так, чтобы добиться необходимой степени детализации. Описание предметной области выполняется на основании информационных потребностей разных групп пользователей будущей системы баз данных. На этом этапе (предварительном) информация, собранная о предметной области предстает в неформальном, описательном виде.

На этапе концептуального проектирования (2 этап) внешние представления пользователей описываются в виде внешних схем, и на основании этих внешних схем строится концептуальная схема всей базы данных.

Цель данной лекции - разобраться с одной из методик формализации описания предметной области. Эта методика называется: "Объект-Свойство-Связь".

3.1 Концептуальное проектирование

Любая база данных - некоторая модель предметной области, т.е. в БД сохраняются только те факты реального мира, которые необходимы в конкретной задаче. Следовательно, при проектировании нужно выделить факты, интересующие пользователей, и отсечь ненужные, а затем формально описать нужные факты.

Семантическое моделирование - наиболее популярный подход к формальному описанию предметной области.

Этот подход основан на признании факта существования в реальном мире **объектов**. Объекты имеют наборы характеристик (или **свойств**) и взаимодействуют между собой с помощью **связей**.

Преимущества подхода “Объект - Свойство - Связь” — как самого популярного из подходов семантического моделирования — таковы:

- Независимость от дальнейшей реализации;
- Интуитивные основные понятия;
- Отражение семантики предметной области (смысла каждого объекта, связи, свойства).

Особенно важно то, что использование подхода “ Объект - Свойство - Связь” позволяет сохранить не только данные, но и частично смысл (семантику) этих данных.

Методологии проектирования, основанные на идеях семантического моделирования, часто называют **нисходящими** методологиями, т.к. они начинают с высшего уровня абстракции – конструкции реального мира, и заканчивают на уровне, когда создается конкретная схема БД.

3.1.1 Объекты и их свойства

Дадим определения основных понятий семантического моделирования:

> ***Сущность (entity***) – собирательное понятие, некоторая **абстракция** реально существующего **объекта**, **процесса** или **явления**, о котором необходимо хранить информацию в базе данных.

В семантическом моделировании применяют не просто понятие "сущность", а говорят "тип сущностей":

> ***Тип сущностей*** - определяет набор объектов с одним и тем же набором свойств.
> ***Экземпляр сущности*** - конкретный объект в наборе.

Пример 1:
Если мы хотим описать всех сотрудников предприятия, то все те общие свойства, которые присущи всем сотрудникам (это может быть "имя", "фамилия", "должность", "зарплата"), сформируют ***тип сущности СОТРУДНИК*** *(EMPLOYEE). Тогда все отдельные сотрудники являются* ***экземплярами сущности СОТРУДНИК****.*

Выбор сущностей зависит от цели создания БД. Если в базе данных отдела кадров СОТРУДНИК - отдельный тип сущностей со своими характеристиками, то в базе данных проектного отдела сотрудник - всего лишь фамилия ответственного за проект, т.е. характеристика ПРОЕКТА как типа сущностей.

Каждый тип сущности имеет набор свойств (характеристик), присущих всем экземплярам данного типа.

> ***Свойство (attribute, атрибут)*** - поименованная характеристика сущности, которая принимает значения из некоторого множества значений (домена).

Пример 2.
Все сотрудники должны иметь: имя, фамилию, дату_рождения, специальность, дату_поступления_на_работу, табельный_номер. Эти характеристики являются ***атрибутами*** *типа сущностей СОТРУДНИК.*

Каждый тип сущностей должен отличаться от всех остальных. **Уникальность типа сущностей** определяется наличием уникальных, идентифицирующих только этот тип, свойств. Каждый тип сущности должен иметь уникальные свойства, присущие только некоторым объектам предметной области.

Пример 3.
Рассмотрим всех сотрудников учебного заведения. Очевидно, что есть разные группы сотрудников - преподаватели, программисты, и т.д.
Для того чтобы этим группам соответствовали свои типы сущностей, необходимо выделить уникальные для каждой группы сотрудников свойства.
ПРЕПОДАВАТЕЛИ отличаются от всех других сотрудников тем, что имеют такие уникальные свойства: "ученая_степень", "звание".
ПРОГРАММИСТЫ отличаются такими свойствами: "специализация" (базы данных, системное программирование, компьютерная графика и т.д.), "языки_программирования", которыми они владеют.

Для идентификации **экземпляра типа сущности** используются специальные свойства. Это может быть одно или несколько свойств, значения которых позволяют **однозначно** отличать один экземпляр сущности от другого. Этот набор специальных свойств называется **первичным ключом**.

Пример 4.
*Любой сотрудник имеет уникальный "идентификационный код налоговой инспекции". Это свойство может быть **первичным ключом**, поскольку значения такого кода никогда не повторяются, и у одного человека может быть только один такой код.*

Пример 5.
Предположим, в предметной области "Аэропорт" есть тип сущностей РЕЙС. Его характеристики таковы:"номер_рейса", "день_вылета", "тип_самолета","пилот".
Рейс 555 "Запорожье - Киев" выполняется каждый день, но по средам и пятницам этот рейс выполняется на самолете типа "Як-40" под управлением пилота Иванова, а во все остальные дни - на "ТУ-104" под управлением пилота Боброва. Для идентификации записи о рейсах самолетов необходим первичный ключ типа ("номер_рейса" + "день_вылета").

Описанный в примере 5 первичный ключ называется **составным.**

Анализ свойств различных объектов показал, что свойства бывают:

а) единичные и множественные

Единичное свойство — каждый экземпляр типа сущности имеет единственное значение этого свойства.

Множественное свойство — каждый экземпляр типа сущности имеет более одного значения этого свойства.

Пример 6.
Тип сущностей “ДЕТАЛЬ”: свойство “Название” - единичное: Гайка, Болт, Винт, Шуруп. Свойство “Диаметр” - множественное: деталь Гайка (один из экземпляров ДЕТАЛИ) может быть разных диаметров.

б) статичные и динамичные

Статичное свойство не меняется с течением времени, в отличие **от динамичного свойства**.

Пример 7.
Тип сущностей “ЛИЧНОСТЬ”: свойство ”Год_рождения” – статичное, свойство “Ученая_степень” – динамичное.

в) обязательные и условные свойства

Свойство обязательно, если оно должно присутствовать у всех экземпляров данного типа сущностей.

Свойство условно, если оно может отсутствовать у некоторых экземпляров данного типа сущностей.

Пример 8.
Тип сущностей “ЛИЧНОСТЬ”: свойство “Ученая степень” – условное, так как его присутствие необязательно у экземпляров ЛИЧНОСТЕЙ (таких как секретарь-референт, ассистент), а свойство “Фамилия” – обязательное.

г) составные свойства

Пример 9.
Тип сущностей “ЛИЧНОСТЬ”: свойство "Адрес" составное:
"адрес" = "город" + "улица" + "дом" + "квартира", где символ "+" означает "конкатенация строк"
Свойство "дата_рождения" = "год" + "месяц" + "день" - тоже составное.

3.1.2 Взаимоотношения объектов: связи

Реальный мир, который нас окружает, состоит из объектов ("Мебель", "Сотрудник", "Самолет"), явлений ("Осадки" из предметной области "Прогноз погоды", и т.п.), процессов ("Поставка" из предметной области "Складской учет", "Выступление" из предметной области "Концерты артистов эстрады"). Как уже было показано выше, *сущности* предметной области обладают наборами свойств. Однако, объекты, процессы, явления существуют не просто сами по себе, а находятся во взаимосвязи друг с другом.

Пример 10.
В предметной области "Концерты артистов эстрады" можно найти такие взаимодействия:
Есть сущность "СПОНСОР" (со свойствами: "фамилия", "адрес", "телефон") и сущность "КОНЦЕРТ" (со свойствами: "название_концерта", "исполнитель", "количество_номеров_в программе", "место_проведения").
Каждый СПОНСОР вполне может финансировать несколько КОНЦЕРТОВ, в то же время каждый КОНЦЕРТ может быть организован одновременно несколькими СПОНСОРАМИ. ***Взаимодействия*** *между двумя сущностями представлены фразами "может финансировать несколько" и "может быть организован несколькими".*

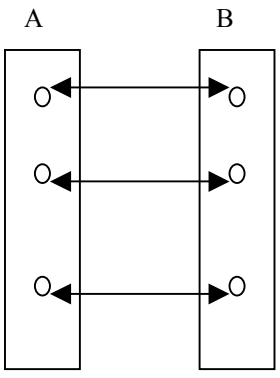
Говорят, что

> В предметной области объекты **взаимодействуют** друг с другом посредством ***связей (relationships)***.

В связи может участвовать два и более объектов. Связи, в которых участвуют два объекта, называются **бинарными**. Связи, в которых участвуют три объекта - **тернарные**, и т.д.
Различают такие типы связей:

"один к одному" (1:1), "один ко многим" (1:m), "многие ко многим" (m:n)

> Связь "**один к одному**" означает, что каждому экземпляру одного типа сущностей **обязательно** соответствует 1 экземпляр другого типа сущностей, и наоборот (см. рис.1)



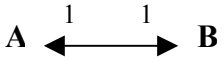


Рисунок 1 - Связь "1:1"

Идентификация экземпляров сущностей уникальна в обоих направлениях.

Пример 11.
“ЛИЧНОСТЬ”↔ “ДНК
У каждой личности есть уникальный код ДНК. Тип сущностей "ДНК" в таком случае может иметь единственное свойство: "код_ДНК"

> Связь "**один ко многим**" означает, что каждому экземпляру одного типа сущностей (А) **обязательно соответствует** 1 или более экземпляров другого типа сущностей(В), однако каждому экземпляру типа В соответствует только один экземпляр типа А.(см. рис.2)

Идентификация экземпляров сущностей уникальна только от В к А.

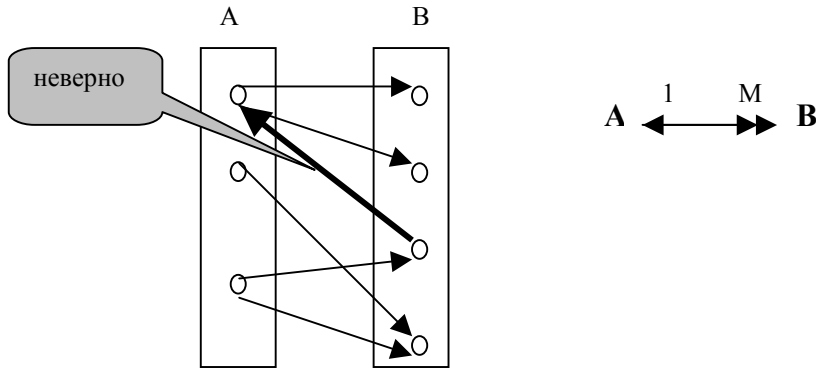


Рисунок 2 - Связь "1:M"

Пример 12.
Связь "1:M":
"РУКОВОДИТЕЛЬ" "отвечает за несколько" “ПРОЕКТОВ”, в то время как у каждого "ПРОЕКТА" только один ответственный "РУКОВОДИТЕЛЬ".
"ВЛАДЕЛЕЦ" "имеет несколько" "АВТОМОБИЛЕЙ", но у каждого "АВТОМОБИЛЯ" только один "ВЛАДЕЛЕЦ"
"ГОРОД" "состоит из нескольких" "РАЙОНОВ", но каждый "РАЙОН" находится только в одном "ГОРОДЕ".

> Связь "**многие ко многим**" означает, что каждому экземпляру одного типа сущностей (А) **обязательно соответствует** 1 или более экземпляров другого типа сущностей(В), и наоборот (см. рис.3)

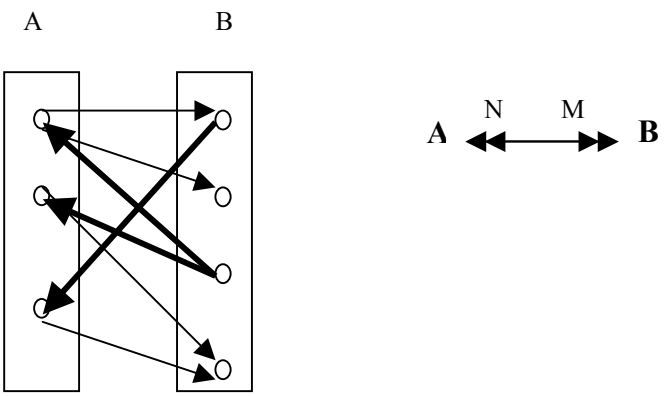


Рисунок 3 - Связь "N: M"

Идентификация экземпляра сущностей неуникальна в обоих направлениях.

Пример 13.
Связь "N:M":
"СТУДЕНТ" "изучает несколько" "ДИСЦИПЛИН", и наоборот "ДИСЦИПЛИНУ" "изучают много" " СТУДЕНТОВ"
"ДЕТАЛЬ" "поставляют несколько" "ПОСТАВЩИКОВ", и наоборот каждый "ПОСТАВЩИК" "осуществляет поставки нескольких" "ДЕТАЛЕЙ".

Важной характеристикой любой связи является ее **класс принадлежности**.
Класс принадлежности показывает, **обязательна** ли связь между экземплярами этих сущностей или нет.

Пример 14:
*Связь обязательна: каждый "СТУДЕНТ " **должен** прослушать несколько "ДИСЦИПЛИН" (связь* 1:M*), обязательная со стороны "СТУДЕНТА" и необязательна со стороны "ДИСЦИПЛИНЫ".*
*Связь необязательна со стороны "ЛИЧНОСТИ": "ЛИЧНОСТЬ" "**может** иметь максимум один" "АВТОМОБИЛЬ" (связь 0:1)*
*Связь обязательна с двух сторон: "ЛИЧНОСТЬ" "**должна** иметь" "ДНК" (связь* 1:1*).*

Таким образом, всего различают **6** типов связей:
Обязательные связи "1:1", "1:M", "N:M"
Необязательные связи "0:1","0:N" и " N:M , необязательная"

Для связей определяются **минимальные** и **максимальные мощности** каждого типа связи.

> ***Мощность*** типа связи - количество экземпляров одного объекта, связанных с одним экземпляром другого объекта.

Если A и B - типы сущностей, и между ними существует связь AB, то
формат спецификации связи такой:

(мин. количество экземпляров B, связанных с одним экземпляром A
,
макс. значение экземпляров B, связанных с одним экземпляром A**)**
:
(мин. количество экземпляров A, связанных с одним экземпляром B
,
макс. значение экземпляров A, связанных с одним экземпляром B**)**

Пример 15.
Если в фирме есть правило: "каждый сотрудник может участвовать по крайней мере в одном проекте, но максимум в 5 разных проектах", то связь между "СОТРУДНИКОМ" и

*"ПРОЕКТОМ" будет выражена как **"(1,5) : (1,N)"**, поскольку над каждым "ПРОЕКТОМ" может работать от 1 до N "СОТРУДНИКОВ".*

Связи, так же как и сущности, могут иметь свойства.

Пример 16.
Связь между "СПОНСОРОМ" и "КОНЦЕРТОМ" может иметь свойства "объем финансирования" (сколько денег вкладывает спонсор в конкретный концерт) и "тип поддержки" (информационная, транспортная, общая).

3.1.3 «Слабые» сущности

Иногда возникает ситуация, когда первичный ключ некоторого типа сущностей состоит из свойств, которые принадлежат другому типу сущностей.

> ***Слабая сущность (weak entity)*** – такой тип сущностей, первичный ключ которого состоит (полностью или частично) из свойств другого типа сущностей.
> Иначе, слабая сущность называется ***зависимой от других***.

Замечание: возможна ситуация, когда слабая сущность может зависеть от нескольких типов сущностей.

Пример 17.
Рассмотрим предметную область «Пользователи электронной почты». Каждый пользователь описывается "именем" и "паролем", имя и пароль выбираются для каждого почтового сервера, на котором этот пользователь держит почтовый ящик.
vasya@zsu.zp.ua, пароль gfhjkm, и vasya@mail.ru, пароль gfccdjhl
В свою очередь, каждый почтовый сервер характеризуется свойствами "адрес почтового сервера", "объем почтового ящика", и т д. Это позволяет выделить два типа сущностей, ПОЛЬЗОВАТЕЛЬ и СЕРВЕР. Однако всех свойств типа сущностей ПОЛЬЗОВАТЕЛЬ будет недостаточно, и необходимо использовать свойство "адрес почтового сервера", которое является ключом типа сущностей СЕРВЕР. Таким образом, ПОЛЬЗОВАТЕЛЬ будет слабой сущностью.

Причиной появления слабых сущностей может быть то, что типы сущностей в предметной области образуют иерархию, в которой каждый подтип характеризуется как своими собственными свойствами, так и ключевыми свойствами своего супертипа(супертипов).

Пример 18.(продолжение примера 17)
Рассмотрим на примере vasya@zsu.zp.ua адреса почтовых серверов:
ua – домен верхнего уровня, zp –домен второго уровня, zsu – имя хоста. Очевидно, что если считать ДОМЕНЫ ВЕРХНЕГО УРОВНЯ, ДОМЕНЫ ВТОРОГО УРОВНЯ и ХОСТЫ как отдельные типы сущностей, обладающие своими наборами свойств, то окажется, что только ДОМЕНЫ ВЕРХНЕГО УРОВНЯ будет сильной сущностью, а ДОМЕНЫ ВТОРОГО УРОВНЯ и ХОСТЫ будут слабыми сущностями, поскольку хост zsu может быть как в домене zp (Запорожье), так и в домене dn (Днепропетровск), и т.д.

Существует ряд ограничений на выбор ключа для слабой сущности.
1. Если W - слабая сущность, тогда должны существовать явные связи R_i с каждой сущностью E_i, чьи свойства определяют ключ W.
2. Связь R_i – обязательная, бинарная, типа W <<-> E_i (многие к одному со стороны W).

3. Те атрибуты ключа W, которые предоставляет сущность E_i, должны быть ключевыми в E_i.
4. Если, в свою очередь, сущность E_i - тоже слабая, зависящая от SE_i, тогда те свойства, которые E_i поставляет в W для создания ключа, могут быть свойствами, позаимствованными сущностью E_i у SE_i.

3.1.4 Сложные сущности

Наконец, различают такие типы сущностей: **простые**, т.е. неделимые сущности, и **сложные** сущности.

Сложные сущности бывают:

- **составные** - соответствуют отображению “целое - часть”;
- **обобщенные** - соответствует отображению ”род-вид” или “супертип-подтип”
- **агрегированные** – соответствует обычно какому-либо **процессу**, в который вовлечены другие объекты.

Пример 19
Составной объект “КОМПЬЮТЕР”: его части “ПРОЦЕССОР”, ”ОЗУ”, ”ЖЕСТКИЙ_ДИСК”, ”ВИДЕОКАРТА”, и т.д.
Обобщенный объект “СОТРУДНИК”: его подтипы - ”МЕНЕДЖЕР”, ”ПРОГРАММИСТ”, ”ДИСПЕТЧЕР”.
Агрегированный объект “ПОСТАВКА”: в поставке участвуют ”ПОКУПАТЕЛЬ”, ”ПОСТАВЩИК”, ”ТОВАР”.

3.2 Проведение этапа концептуального проектирования системы баз данных

Опишем теперь сам процесс концептуального проектирования.

3.2.1 Моделирование внешних представлений пользователей о предметной области

Каждая группа пользователей предоставляет свои сведения о предметной области, такие как: входные данные и выходные данные, их форматы, вспомогательные данные, необходимые в процессе работы, и другие сведения (например, требования безопасности).

Разработчик системы базы данных на основании этих сведений выполняет такие шаги:

1. **Неформальное описание предметной области с точки зрения конкретной группы пользователей.**
2. **Разбор описания с целью вычленения типов сущностей и их свойств.**

Для любого внешнего представления нужно выделить типы сущностей, необходимые для его описания. В отдельных случаях это сделать сложно, так как информация о предметной области может быть представлена либо как сущность, либо как атрибут, либо связь. Здесь на помощь может прийти интуиция и опыт разработчика.

Пример 20.
"ДЕТАЛЬ" и "ИЗДЕЛИЕ" можно представить как отдельные типы сущностей, или как сущность "ИЗДЕЛИЕ" с атрибутом "ДЕТАЛЬ".

Для каждого выбранного типа сущностей определяются наборы атрибутов, и выбирается первичный ключ (или несколько альтернативных ключей).

В процессе определения типов сущностей необходимо получить ответ на следующие вопросы:

1. Каково имя (имена) каждого типа сущности?
2. Каково значение (смысл) каждого типа сущности?
3. Какие атрибуты каждого типа сущности представляют интерес?
4. Каково имя каждого атрибута?
5. Каково значение (смысл) каждого атрибута?

Для каждого выделенного типа создается его описание. Описание состоит из: типа сущности, описания, что этот тип представляет, перечисляет все его атрибуты. Для типа сущности и атрибутов указываются синонимы, а также подтипы типа сущностей. Это описание вводится в словарь данных.
Выделение типов сущностей – итеративный процесс. Описание типа может многократно изменяться до тех пор, пока не будет достигнуто всеобщее согласие по поводу его правильности.

3. Разбор описания с целью фиксации связей, существующих между объектами в предметной области и важных для данной информационной системы. Определение типов связей.

Выявляются зависимости между сущностями, определяются свойства зависимостей. Каждый тип связи “Сущность - Сущность” именуется. Кроме спецификации связей между сущностями, определяются также связи типа “Сущность - Атрибут” (атрибут обязательный/необязательный, множественный/единичный, статичный/динамичный и т.д.), и связи “Атрибут - Атрибут” между атрибутами одного типа сущностей (т.н. *функциональные зависимости*, о них позже, в лекции 9).

4. Определение ограничений на значения свойств, на типы связей.

Для атрибутов определяется область их значений, например, "фамилия" сотрудника есть текстовая строка длиной не больше 50 символов.

При определении ограничений необходимо получить ответ на такие вопросы:
1. Какова область значений каждого атрибута?
2. Каковы известные функциональные зависимости между атрибутами каждого типа сущности?
3. Каковы ключи (если таковые существуют) каждого типа сущности?
4. Какой тип отображения соответствует каждому типу связи (например, один к одному, один ко многим, многие ко многим)?
5. Какие ограничения, выражаемые предикатами, накладываются на данные?

5. Определение тех основных операций над данными, которые будет выполнять конкретная группа пользователей.

Как правило, для каждой группы пользователей существует несколько типичных операций, которые будут выполняться в процессе работы с системой баз данных.

Пример 21.

Для предметной области о сотрудниках и проектах, такими типичными операциями могут быть:

- *назначение сотрудника на проект;*
- *получение данных о проектах, в которых участвует конкретный сотрудник;*
- *снятие сотрудника с проекта;*
- *создание /закрытие проекта.*

Для каждой операции указываются вид (выборка или обновление данных), частота реализации, внешний источник, получатель результатов, часть (части) схемы, затрагиваемые операцией.

Определяя требования к реализации операций, целесообразно задаться такими вопросами:

1. Какие операции необходимо реализовывать для каждой группы пользователей?
2. Какие типы сущностей, атрибуты и типы связей затрагиваются каждой операцией?
3. Как в общих чертах охарактеризовать каждую операцию в терминах описания предметной области — на естественном языке или проблемно-ориентированном спецификационном языке?
4. Какого рода доступ к данным связан с каждой операцией (выборка, обновление)?
5. Критично ли выполнение операции ко времени (пакетный, интерактивный)?
6. Как часто будет вызываться операция (ежедневно, еженедельно, каждый час)
7. Каков приоритет обработки, присваиваемый операции?
8. Каковы требования к процессам параллельной обработки?
9. Какова ожидаемая схема использования базы данных (например, смесь операций, расписания выполнения и т.д.)?
10. Какие отчеты необходимы?
11. Каковы форматы каждого отчета?
12. Каковы приемлемые временные рамки получения каждого отчета?
13. Какие части базы данных наиболее существенны с точки зрения обеспечения функционирования базы данных?

6. Формализация результатов шагов 1-5 и создание *внешней схемы* БД для этой группы пользователей.

На этом шаге создаются формальные спецификации внешнего представления. Подробнее об этом - см. лекцию 4.

3.2.2 Объединение внешних представлений в единое концептуальное представление

При объединении внешних схем могут создаваться конструкции, являющиеся производными по отношению к конструкциям во внешних схемах. Такие конструкции называются **абстракциями**.

> **Абстракция** – модель системы, в которой намеренно опущены некоторые детали.

Цели введения абстракций чаще всего таковы:

- Объединить разрозненные по разным внешним представлениям пользователей свойства одного типа сущностей в один тип;
- Устранить различия в представлении подобных объектов.

Остается отметить, что этап концептуального проектирования выполняется не один раз, а несколько раз, с постепенным уточнением всех характеристик будущей системы баз данных.

Этот этап можно считать самым важным, поскольку ошибки, допущенные на этом этапе, проявляются в дальнейшем в некорректной или неудобной логической структуре базы данных, в избыточности данных, и как следствие, в плохой физической организации хранения данных, в потере производительности, в затратах на перепроектирование структур данных и прикладных программ в составе информационной системы.

Пример этапа концептуального проектирования внешнего представления будет рассмотрен в лекции 5.