Лекция 8

**ТЕМА:Проектирование с использованием метода «сущность-связь».**

**Литература:**1. Карпова И.П. Проектирование реляционных баз данных //методические указания к курсовому проектированию по курсу "Базы данных"

2. http://citforum.ru/database/dblearn/dblearn08.shtml **−**Пушников А.Ю. Введение в системы управления базами данных// Учебное пособие.

В реальном проектировании структуры базы данных применяется так называемое, ***семантическое моделирование***. Семантическое моделирование представляет собой моделирование структуры данных, опираясь на смысл этих данных. В качестве инструмента семантического моделирования используются различные варианты ***диаграмм сущность-связь*** (***ER - Entity-Relationship***), которые могут быть относительно легко отображены в любую систему баз данных.

Первый вариант модели сущность-связь был предложен в 1976 г. Питером Пин-Шэн Ченом. В дальнейшем многими авторами были разработаны свои варианты подобных моделей (нотация Мартина, нотация IDEF1X, нотация Баркера и др.). Кроме того, различные CASE-средства используют несколько отличающиеся друг от друга нотации ERD. Одна из наиболее распространенных нотаций предложена Баркером и используется в Oracle Designer. В CASE-средстве SilverRun используется один из вариантов нотации Чена. CASE-средства ERwin, ER / Studio, Design / IDEF используют методологию IDEF1Х. По сути, все варианты диаграмм сущность-связь исходят из одной идеи − рисунок всегда нагляднее текстового описания. Все такие диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями.

Мы опишем работу с ER-диаграммами близко к нотации Баркера, как довольно легкой в понимании основных идей.

#### Основные понятия ER-диаграмм

**Определение 1.Сущность (Entity)** − это реальный или представляемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором должна сохраняться и быть доступна.

Каждая сущность должна иметь уникальное наименование, выраженное существительным в единственном числе. Примерами сущностей могут быть такие классы объектов как "Поставщик", "Сотрудник", "Накладная". Каждая сущность в модели изображается в виде прямоугольника с наименованием (рис.1). При этом имя сущности – это имя типа, а не некоторого конкретного экземпляра этого типа.

****

Рис. 1. Обозначение сущности.

Для сущностей различают **тип** сущности и **экземпляр**. Тип характеризуется именем и списком свойств, а экземпляр – конкретными значениями свойств.

**Определение 2. Экземпляр сущности−** это конкретный представитель данной сущности.

Например, представителем сущности "Сотрудник" может быть "Сотрудник Иванов".

Каждый экземпляр сущности должен быть отличим от любого другого экземпляра той же сущности, т.е. сущности должны иметь некоторые свойства, уникальные для каждого экземпляра этой сущности (это требование в некотором роде аналогично требованию отсутствия кортежей-дубликатов в реляционных таблицах).

Типы сущностей можно классифицировать как сильные и слабые. Сильные сущности существуют сами по себе, а существование слабых сущностей зависит от существования сильных. Например, читатель библиотеки – сильная сущность, а абонемент этого читателя – слабая, которая зависит от наличия соответствующего читателя. Слабые сущности называют подчинёнными (дочерними), а сильные – базовыми (основными, родительскими).

Также сущности разделяют на **независимые** и **зависимые.** Сущность является **независимой**, если каждый экземпляр ее может быть однозначно идентифицирован без определения его отношений с другими сущностями. Независимая сущность изображается прямоугольником с четко выраженными углами. Сущность является **зависимой**, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности. Зависимая сущность изображается прямоугольником со скругленными углами.

Сущности бывают как физически существующие (например, *СОТРУДНИК* или *АВТОМОБИЛЬ*), так и абстрактные (например, *ЭКЗАМЕН* или *ДИАГНОЗ*).

Каждая сущность может обладать любым количеством связей с другими сущностями модели.

**Определение 3. Атрибут сущности−** это именованная характеристика, являющаяся некоторым значимым для рассматриваемой предметной области свойством сущности.

Наименование атрибута должно быть выражено существительным в единственном числе (возможно, с характеризующими прилагательными).

Примерами атрибутов сущности "Сотрудник" могут быть такие атрибуты как "Табельный номер", "Фамилия", "Имя", "Отчество", "Должность", "Зарплата" и т.п.

Атрибуты изображаются в пределах прямоугольника определяющего сущность, причем каждый атрибут занимает отдельную строку, и отделяются от названия сущности линией (рис. 2).

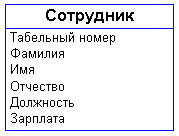
****

Рис. 2. Указание атрибутов сущности.

Рядом с именем атрибута можно приводить примеры значений данного атрибута.

Различают:

1. ***Идентифицирующие и описательные атрибуты***. Идентифицирующие атрибуты имеют уникальное значение для сущностей данного типа и являются ***потенциальными ключами****.* Они позволяют однозначно распознавать экземпляры сущности. Из потенциальных ключей выбирается один первичный ключ. В качестве первичного ключа обычно выбирается потенциальный ключ, по которому чаще происходит обращение к экземплярам сущности. Кроме того, ПК должен включать в свой состав минимально необходимое для идентификации количество атрибутов. Остальные атрибуты называются описательными и заключают в себе интересующие свойства сущности.
2. ***Составные и простые атрибуты***. Простой атрибут состоит из одного компонента, его значение неделимо. Составной атрибут является комбинацией нескольких компонентов, возможно, принадлежащих разным типам данных (например, ФИО или адрес). Решение о том, использовать составной атрибут или разбивать его на компоненты, зависит от характера его обработки и формата пользовательского представления этого атрибута.
3. ***Однозначные и многозначные атрибуты.***Могут иметь соответственно одно или много значений для каждого экземпляра сущности. На диаграмме изображаются с двойным подчеркиванием.
4. ***Основные и производные атрибуты*.** Значение основного атрибута не зависит от других атрибутов. Значение производного атрибута вычисляется на основе значений других атрибутов (например, возраст студента вычисляется на основе даты его рождения и текущей даты).

Спецификация атрибута состоит из его названия, указания типа данных и описания ограничений целостности – множества значений (или домена), которые может принимать данный атрибут.

Атрибут может быть либо обязательным, либо необязательным. Обязательность означает, что атрибут не может принимать неопределенных значений (Null). Обязательный атрибут помечается звездочкой, а необязательный − кружком (рис. 3).

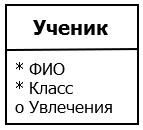
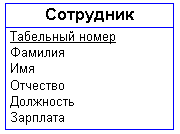
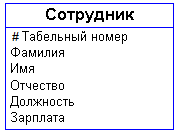


Рис. 3. Указание обязательных и необязательных атрибутов сущности.

**Определение 4.Ключ сущности−** это неизбыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. Неизбыточность заключается в том, что при удалении любого атрибута из ключа нарушается его уникальность.

Сущность может иметь несколько различных ключей. Различают **первичный, альтернативный и внешний** ключи.

**Первичный ключ (Primary Key) −**это атрибут или группа атрибутов, используемых для однозначной идентификации экземпляра сущности. На диаграмме атрибуты первичного ключа размещаются первыми в списке атрибутов и подчеркиваются или предваряются # (рис. 4):

Рис. 4. Указание ключевого атрибута.

**Альтернативный ключ (Alternate Key)** −потенциальный ключ, не ставший первичным. На диаграмме альтернативный ключ обозначается **AK n. m,** где n - порядковый номер ключа, m - порядковый номер атрибута в ключе.

**Внешние ключи (Foreign Key)** создаются, когда сущности соединяются связью при построениифизических ER-диаграмм. Происходит миграция атрибутов первичного ключа родительской сущности в дочернюю сущность. Появившийся таким образом в дочерней сущности атрибут будет являться внешним ключом. (Из родительской сущности атрибут не исчезает, а просто копируется в дочерней сущности). Внешний ключ обозначается ВК.

Первичный ключ может быть **абсолютный или относительный**. Если все атрибуты, составляющие первичный ключ, принадлежат сущности, то он является абсолютным. Если один или более атрибутов первичного ключа принадлежат другой сущности, то он является относительным. Когда первичный ключ является относительным, сущность определяется как **зависимая** сущность, поскольку ее идентификатор зависит от другой сущности. В примере на рисунке 5 первичный ключ сущности Компания является относительным. Он включает первичный ключ сущности Список компаний.

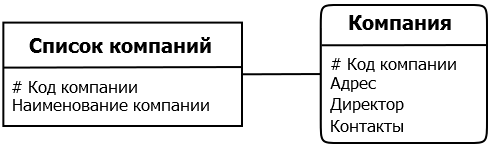


Рис. 5. Относительный первичный ключ.

**Определение 5. Связь−** это графически изображаемая ассоциация, устанавливаемая между двумя сущностями, значимая для рассматриваемой предметной области.

Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Определение связи в методе Баркера несколько отличается от данного Ченом.

**Определение 5.1. Связь−** это ассоциация между сущностями, при которой, как правило, каждый экземпляр одной сущности, называемой ***родительской сущностью***, ассоциирован с произвольным (в том числе и нулевым) количеством экземпляров второй сущности, называемой ***сущностью-потомком***, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущности-потомка может существовать только при существовании сущности-родителя.

Связи позволяют по одной сущности находить другие сущности, связанные с нею.

В любой связи выделяются два конца (в соответствии с существующей парой связываемых сущностей), на каждом из которых указывается:

* имя конца связи,
* тип конца связи (сколько экземпляров данной сущности связывается),
* обязательность связи (т.е. любой ли экземпляр данной сущности должен участвовать в данной связи).

Связь представляется в виде линии, связывающей две сущности или ведущей от сущности к ней же самой. При этом в месте "стыковки" связи с сущностью используется трехточечный вход в прямоугольник сущности, если для этой сущности в связи могут использоваться много (many) экземпляров сущности, и одноточечный вход, если в связи может участвовать только один экземпляр сущности. Обязательный конец связи изображается сплошной линией, а необязательный – прерывистой линией (рис. 6).

****

Рис. 6. Изображение связи между сущностями.

Наименование связи обычно выражается в неопределенной глагольной форме: "иметь", "принадлежать" и т.п. Каждое из наименований относится к своему концу связи. Иногда наименования не пишутся ввиду их очевидности. Имя каждой связи между двумя сущностями должно быть уникальным, но имена связей в модели не обязаны быть уникальными. Имя связи всегда формируется с точки зрения родителя, так что может быть образовано предложение соединением имени сущности-родителя, имени связи, выражения степени и имени сущности-потомка.

Каждая связь может иметь один из следующих **типов** связи (рис. 7):

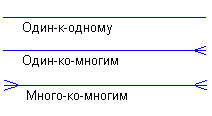
****

Рис. 7. Типы связей и их изображение на диаграмме.

Связь типа **один-к-одному** означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Связь типа **один-ко-многим** означает, что один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности (правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны "один") называется **родительской**, правая (со стороны "много") −**дочерней**. Характерный пример такой связи приведен на рис. 6.

Связь типа **много-ко-многим** означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности. Тип связи много-ко-многим является **временным** типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа **один-ко-многим** путем создания промежуточной сущности.

Каждая связь может иметь одну из двух **модальностей** связи (рис. 8):

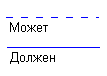
****

Рис. 8. Модальность связи и ее изображение на диаграмме.

Модальность "**может**" означает, что экземпляр одной сущности ***может быть связан*** с одним или несколькими экземплярами другой сущности, ***а может быть и не связан*** ни с одним экземпляром.

Модальность "**должен**" означает, что экземпляр одной сущности ***обязан быть связан не менее чем с одним***экземпляром другой сущности.

Связь может иметь разную модальность с разных концов (как на рис. 6).

Тип связи в совокупности с модальностью называют еще **кардинальностью** связи (табл. 1):

Таблица 1. Кардинальность связи.

|  |  |
| --- | --- |
| **Обозначение** | **Кардинальность** |
| http://unesco.kemsu.ru/study_work/method/DB/book/img/clip_image027.gif | 0,1 |
| http://unesco.kemsu.ru/study_work/method/DB/book/img/clip_image019.gif | 1,1 |
| http://unesco.kemsu.ru/study_work/method/DB/book/img/clip_image035.gif | 0,N |
| http://unesco.kemsu.ru/study_work/method/DB/book/img/clip_image036.gif | 1,N |

Описанный графический синтаксис позволяет ***однозначно*** читать диаграммы, пользуясь следующей схемой построения фраз:

<Каждый экземпляр СУЩНОСТИ 1><МОДАЛЬНОСТЬ СВЯЗИ><НАИМЕНОВАНИЕ СВЯЗИ><ТИП СВЯЗИ><экземпляр СУЩНОСТИ 2>.

Каждая связь может быть прочитана как слева направо, так и справа налево. Связь на рис. 6 читается так:

Слева направо: "каждый сотрудник может иметь несколько детей".

Справа налево: "Каждый ребенок обязан принадлежать ровно одному сотруднику".

На следующем примере (рис. 9) изображена рекурсивная связь, связывающая сущность ЧЕЛОВЕК с ней же самой. Конец связи с именем "являться сыном" определяет тот факт, что один человек может быть сыном не более чем одному отцу. Конец связи с именем "являться отцом" означает, что один человек может являться отцом для одного или более ЛЮДЕЙ (т.е не каждый человек имеет детей).

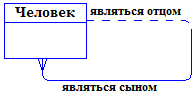


Рисунок 9. Пример рекурсивной связи.

**Более сложные элементы ER-модели**

Ранее были представлены только основные и наиболее очевидные понятия ER-модели данных. К числу более сложных элементов модели относятся следующие:

* **Подтипы и супертипы сущностей**. Как в языках программирования с развитыми типовыми системами (например, в языках объектно-ориентированного программирования), вводится возможность наследования типа сущности, исходя из одного или нескольких супертипов. Интересные нюансы связаны с необходимостью графического изображения этого механизма.
* **Уточняемые степени связи**. Иногда бывает полезно определить возможное количество экземпляров сущности, участвующих в данной связи (например, служащему разрешается участвовать не более, чем в трех проектах одновременно). Для выражения этого семантического ограничения разрешается указывать на конце связи ее максимальную или обязательную степень.
* **Каскадные удаления экземпляров сущностей**. Некоторые связи бывают настолько сильными (конечно, в случае связи "один-ко-многим"), что при удалении опорного экземпляра сущности (соответствующего концу связи "один") нужно удалить и все экземпляры сущности, соответствующие концу связи "многие". Соответствующее требование "каскадного удаления" можно сформулировать при определении сущности.
* **Домены.** Как и в случае реляционной модели данных бывает полезна возможность определения потенциально допустимого множества значений атрибута сущности (домена).

Эти и другие более сложные элементы модели данных "Сущность-Связи" делают ее существенно более мощной, но одновременно несколько усложняют ее использование.

**Разработка ER− моделей.**

Семантическое моделирование начинается с разбивки предметной области на ряд локальных областей, каждая из которых (в идеале) включает в себя информацию, достаточную для обеспечения запросов отдельной группы будущих пользователей или решения отдельной задачи (подзадачи). Каждое локальное представление моделируется отдельно, затем они объединяются.

Выбор локального представления зависит от масштабов предметной области. Обычно она разбивается на локальные области таким образом, чтобы каждая из них соответствовала отдельному внешнему приложению и содержала 6-7 сущностей.

Разработка ERD включает следующие основные этапы:

* Идентификация сущностей, их атрибутов, а также первичных и альтернативных ключей.
* Идентификация отношений между сущностями и указание типов отношений.
* Разрешение неспецифических отношений (отношений многие-ко-многим).

Процесс выделения сущностей, атрибутов и связей является итерационным. Разработав первый приближенный вариант диаграмм, мы уточняем их, опрашивая экспертов предметной области. При этом документацией, в которой фиксируются результаты бесед, являются сами ER-диаграммы.

Как и в реляционных схемах баз данных, в ER −диаграммах вводится понятие нормальных форм, причем их смысл очень близко соответствует смыслу реляционных нормальных форм. Приведем только краткие и неформальные определения трех первых нормальных форм.

В **первой нормальной форме ER-схемы** устраняются повторяющиеся атрибуты или группы атрибутов, т.е. производится выявление неявных сущностей, "замаскированных" под атрибуты.

Во **второй нормальной форме** устраняются атрибуты, зависящие только от части уникального идентификатора. Эта часть уникального идентификатора определяет отдельную сущность.

В **третьей нормальной форме** устраняются атрибуты, зависящие от атрибутов, не входящих в уникальный идентификатор. Эти атрибуты являются основой отдельной сущности.

После того, как созданы локальные представления, выполняется их объединение. При небольшом количестве локальных областей (не более пяти) они объединяются за один шаг. В противном случае обычно выполняют бинарное объединение в несколько этапов.

При объединении проектировщик может формировать конструкции, производные по отношению к тем, которые были использованы в локальных представлениях. Такой подход может преследовать следующие цели:

* объединение в единое целое фрагментарных представлений о различных свойствах одного и того же объекта;
* введение абстрактных понятий, удобных для решения задач системы, установление их связи с конкретными понятиями, использованными в модели;
* образование классов и подклассов подобных объектов (например, класс "изделие" и подклассы типов изделий, производимых на предприятии).

На этапе объединения необходимо выявить и устранить все противоречия. Например, одинаковые названия семантически различных объектов или связей или несогласованные ограничения целостности на одни и те же атрибуты в разных приложениях. Устранение противоречий вызывает необходимость возврата к этапу моделирования локальных представлений с целью внесения в них соответствующих изменений.

По завершении объединения результаты проектирования являют собой концептуальную инфологическую модель предметной области. Модели локальных представлений – это внешние инфологические модели.

**Концептуальные и физические ER-диаграммы.**

Различают **концептуальные** и **физические** ER-диаграммы. Концептуальные диаграммы не учитывают особенностей конкретных СУБД. Физические диаграммы строятся по концептуальным и представляют собой прообраз конкретной базы данных. Сущности, определенные в концептуальной диаграмме становятся таблицами, атрибуты становятся колонками таблиц (при этом учитываются допустимые для данной СУБД типы данных и наименования столбцов), связи реализуются путем **миграции** ключевых атрибутов родительских сущностей и создания внешних ключей.

При правильном определении сущностей, полученные таблицы будут сразу находиться в 3НФ.