Московский Энергетический Институт

Отчет по лабораторной работе № 4

Практическое применение методологии IDEF1X

Студент группы А-13-06 Новик Алексей
06.06.2011

Предназначение IDEF1X

IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. **Концептуальной схемой** мы называем универсальное представление структуры данных в рамках коммерческого предприятия, независимое от конечной реализации базы данных и аппаратной платформы. Будучи статическим методом разработки, IDEF1X изначально не предназначен для динамического анализа по принципу "AS IS", тем не менее, он иногда применяется в этом качестве, как альтернатива методу IDEF1. Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того, как все информационные ресурсы исследованы (скажем с помощью метода IDEF1) и решение о внедрении реляционной базы данных, как части корпоративной информационной системы, было принято. Однако не стоит забывать, что средства моделирования IDEF1X специально разработаны для построения реляционных информационных систем, и если существует необходимость проектирования другой системы, скажем объектно-ориентированной, то лучше избрать другие методы моделирования.

Существует несколько очевидных причин, по которым IDEF1X не следует применять в случае построения нереляционных систем. Во-первых, IDEF1X требует от проектировщика определить ключевые атрибуты, для того чтобы отличить одну сущность от другой, в то время как объектно-ориентированные системы не требуют задания ключевых ключей, в целях идентифицирования объектов. Во-вторых, в тех случаях, когда более чем один атрибут является однозначно идентифицирующим сущность, проектировщик должен определить один из этих атрибутов первичным ключом, а все остальные вторичными. И, таким образом, построенная проектировщиком IDEF1X-модель и переданная для окончательной реализации программисту является некорректной для применения методов объектно-ориентированной реализации, и предназначена для построения реляционной системы.

Концепция и семантика IDEF1X

Сущности в IDEF1X и их атрибуты.

Хотя терминология IDEF1X практически совпадает с терминологией IDEF1, существует ряд фундаментальных отличий в теоретических концепциях этих методологий. Сущность в IDEF1X описывает собой совокупность или набор экземпляров похожих по свойствам, но однозначно отличаемых друх от друга по одному или нескольким признакам. Каждый экземпляр является реализацией сущности. Таким образом, сущность в IDEF1X описывает конкретный набор экземпляров реального мира, в отличие от сущности в IDEF1, которая представляет собой абстрактный набор информационных отображений реального мира. Примером сущности IDEF1X может быть сущность "СОТРУДНИК", которая представляет собой всех сотрудников предприятия, а один из них, скажем, Иванов Петр Сергеевич, является конкретной реализацией этой сущности. В примере, приведенном на рис. 1, каждый экземпляр сущности СОТРУДНИК содержит следующую информацию: ID сотрудника, имя сотрудника, адрес сотрудника и

т.п. В IDEF1X модели эти свойства называются атрибутами сущности. Каждый атрибут содержит только часть информации о сущности.

Связи между сущностями

Связи в IDEF1X представляют собой ссылки, соединения и ассоциации между сущностями. Связи это суть глаголы, которые показывают, как соотносятся сущности между собой. Ниже приведен ряд примеров связи между сущностями:

Отдел <состоит из> нескольких **Сотрудников**

Самолет <перевозит> нескольких **Пассажиров.**

Сотрудник <пишет> разные **Отчеты.**

Во всех перечисленных примерах взаимосвязи между сущностями соответствуют схеме **один ко многим**. Это означает, что один экземпляр первой сущности связан с несколькими экземплярами второй сущности. Причем первая сущность называется **родительской**, а вторая - **дочерней**. В приведенных примерах глаголы заключены в угловые скобки. Связи отображаются в виде линии между двумя сущностями с точкой на одном конце и глагольной фразой, отображаемой над линией. На рис. 1 приводится диаграмма связи между Сотрудником и Отделом.

Отношения многие ко многим обычно используются на начальной стадии разработки диаграммы, например, в диаграмме зависимости сущностей и отображаются в IDEF1X в виде сплошной линии с точками на обоих концах. Так как отношения многие ко многим могут скрыть другие бизнес правила или ограничения, они должны быть полностью исследованы на одном из этапов моделирования. Например, иногда отношение многие ко многим на ранних стадиях моделирования идентифицируется неправильно, на самом деле представляя два или несколько случаев отношений один-ко-многим между связанными сущностями. Или, в случае необходимости хранения дополнительных сведений о связи многие-ко-многим, например, даты или комментария, такая связь должна быть заменена дополнительной сущностью, содержащей эти сведения. При моделировании необходимо быть уверенным в том, что все отношения многие ко многим будут подробно обсуждены на более поздних стадиях моделирования для обеспечения правильного моделирования отношений.

Идентификация сущностей. Представление о ключах.

Сущность описывается в диаграмме IDEF1X графическим объектом в виде прямоугольника. На рис.2 приведен пример IDEF1X диаграммы. Каждый прямоугольник, отображающий собой сущность, разделяется горизонтальной линией на часть, в которой расположены **ключевые поля** и часть, где расположены неключевые поля. Верхняя часть называется ключевой областью, а нижняя часть областью данных. Ключевая область объекта СОТРУДНИК содержит поле "Уникальный идентификатор сотрудника", в области данных находятся поля "Имя сотрудника", "Адрес сотрудника", "Телефон сотрудника" и т.д.

Ключевая область содержит **первичный ключ** для сущности. Первичный ключ - это набор атрибутов, выбранных для идентификации уникальных экземпляров сущности. Атрибуты первичного ключа располагаются над линией в ключевой области. Как следует из названия, неключевой атрибут - это атрибут, который не был выбран ключевым. Неключевые атрибуты располагаются под чертой, в области данных.

При создании сущности в IDEF1X модели, одним из главных вопросов, на который нужно ответить, является: "Как можно идентифицировать уникальную запись?". Для этого требуется уникальная идентификация каждой записи в сущности для того, чтобы правильно создать логическую модель данных. Напомним, что сущности в IDEF1X всегда имеют ключевую область и, поэтому в каждой сущности должны быть определены ключевые атрибуты.

Выбор первичного ключа для сущности является очень важным шагом, и требует большого внимания. В качестве первичных ключей могут быть использованы несколько атрибутов или групп атрибутов. Атрибуты, которые могут быть выбраны первичными ключами, называются кандидатами в ключевые атрибуты (потенциальные атрибуты). Кандидаты в ключи должны уникально идентифицировать каждую запись сущности. В соответствии с этим, ни одна из частей ключа не может быть NULL, не заполненной или отсутствующей.

Например, для того, чтобы корректно использовать сущность СОТРУДНИК в IDEF1X модели данных (а позже в базе данных), необходимо иметь возможность уникально идентифицировать записи. Правила, по которым вы выбираете первичный ключ из списка предполагаемых ключей, очень строги, однако могут быть применены ко всем типам баз данных и информации. Правила устанавливают, что атрибуты и группы атрибутов должны:

- Уникальным образом идентифицировать экземпляр сущности.
- Не использовать NULL значений.
- Не изменяться со временем. Экземпляр идентифицируется при помощи ключа. При изменении ключа, соответственно меняется экземпляр.
- Быть как можно более короткими для использования индексирования и получения данных. Если вам нужно использовать ключ, являющийся комбинацией ключей из других сущностей, убедитесь в том, что каждая из частей ключа соответствует правилам.

Для наглядного представления о том, как целесообразно выбирать первичные ключи, приведем следующий пример - выберем первичный ключ для знакомой нам сущности "СОТРУДНИК":

- Атрибут "ID сотрудника" является потенциальным ключом, так как он уникален для всех экземпляров сущности СОТРУДНИК.
- Атрибут "Имя сотрудника" не очень хорош для потенциального ключа, так как среди служащих на предприятии может быть, к примеру, двое Иванов Петровых.

- Атрибут "Номер страхового полиса сотрудника" является уникальным, но проблема в том, что СОТРУДНИКА может не иметь такового.
- Комбинация атрибутов "имя сотрудника" и "дата рождения сотрудника" может оказаться удачной для наших целей и стать искомым потенциальным ключом.

После проведенного анализа можно назвать два потенциальных ключа - первый "Номер сотрудника" и комбинация, включающая поля "имя сотрудника" и "Дата рождения сотрудника". Так как атрибут "Номер сотрудника" имеет самые короткие и уникальные значения, то он лучше других подходит для первичного ключа.

При выборе первичного ключа для сущности, разработчики модели часто используют дополнительный (суррогатный) ключ, т.е. произвольный номер, который уникальным образом определяет запись в сущности. Атрибут "Номер сотрудника" является примером суррогатного ключа. **Суррогатный ключ** лучше всего подходит на роль первичного ключа потому, что является коротким и быстрее всего идентифицирует экземпляры в объекте. К тому же суррогатные ключи могут автоматически генерироваться системой так, чтобы нумерация была сплошной, т.е. без пропусков.

Потенциальные ключи, которые не выбраны первичными, могут быть использованы в качестве вторичных или альтернативных ключей. С помощью альтернативных ключей часто отображают различные индексы доступа к данным в конечной реализации реляционной базы.

Если сущности в IDEF1X диаграмме связаны, связь передает ключ (или набор ключевых атрибутов) дочерней сущности. Эти атрибуты называются **внешними ключами. Внешние ключи** определяются как атрибуты первичных ключей родительского объекта, переданные дочернему объекту через их связь. Передаваемые атрибуты называются **мигрирующими.**

Классификация сущностей в IDEF1X. Зависимые и независимые сущности.

При разработке модели, зачастую, приходится сталкиваться с сущностями, уникальность которых зависит от значений атрибута внешнего ключа. Для этих сущностей (для уникального определения каждой сущности) внешний ключ должен быть частью первичного ключа дочернего объекта.

Дочерняя сущность, уникальность которой зависит от атрибута внешнего ключа, называется **зависимой сущностью**. В примере на рис.1 сущность СОТРУДНИК является зависимой сущностью потому, что его идентификация зависит от сущности ОТДЕЛ. В обозначениях IDEF1X зависимые сущности представлены в виде закругленных прямоугольников.

Зависимые сущности далее классифицируются на сущности, которые не могут существовать без родительской сущности и сущности, которые не могут быть идентифицированы без использования ключа родителя (сущности, зависящие от идентификации). Сущность СОТРУДНИК принадлежит ко второму типу зависимых сущностей, так как сотрудники могут существовать и без отдела.

Напротив, существуют ситуации в которых сущность зависит от существования другой сущности. Рассмотрим две сущности: ЗАПРОС, используемый для отслеживания запросов покупателей, и ПОЗИЦИЯ ЗАПРОСА, который отслеживает отдельные элементы в ЗАПРОСе. Связь между этими двумя сущностями может быть выражена в виде ЗАПРОС <содержит> один или несколько ПОЗИЦИЙ ЗАПРОСА. В этом случае, ПОЗИЦИЯ ЗАПРОСА зависит от существования ЗАКАЗА.

Сущности, независящие при идентификации от других объектов в модели, называются независимыми **сущностями**. В вышеописанном примере сущность ОТДЕЛ можно считать независимой. В IDEF1X независимые сущности представлены в виде прямоугольников.

Типы связей между сущностями. Идентифицирующие и неидентифицирующие связи.

В IDEF1X концепция зависимых и независимых сущностей усиливается типом взаимосвязей между двумя сущностями. Если вы хотите, чтобы внешний ключ передавался в дочернюю сущность (и, в результате, создавал зависимую сущность), то можете создать **идентифицирующую связь** между родительской и дочерней сущность.

Идентифицирующие взаимосвязи обозначаются сплошной линией между сущностями.

Неидентифицирующие связи, являющиеся уникальными для IDEF1X, также связывают родительскую сущность с дочерней. Неидентифицирующие связи используются для отображения другого типа передачи атрибутов внешних ключей - передача в область данных дочерней сущности (под линией).

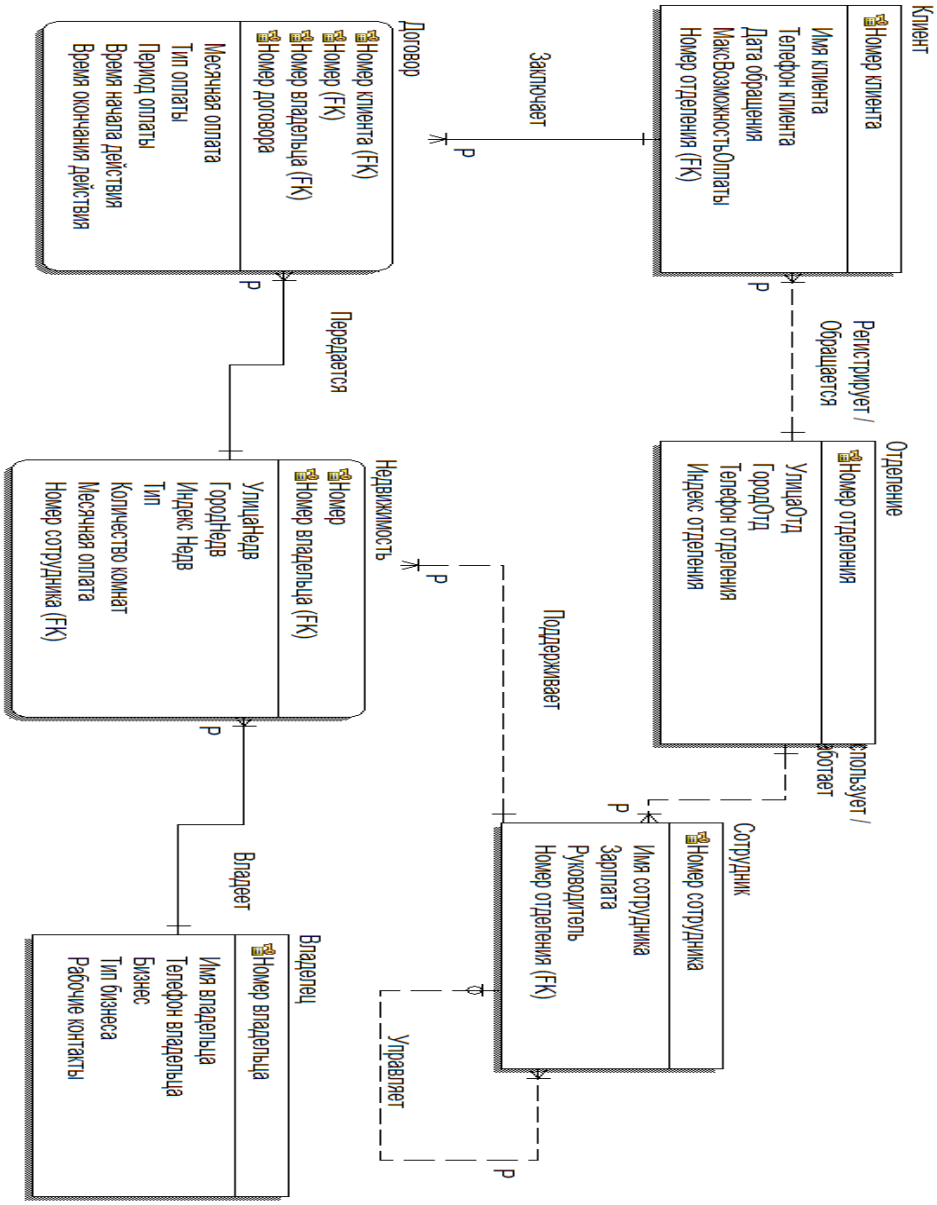
Неидентифицирующие связи отображаются пунктирной линией между объектами. Так как переданные ключи в неидентифицирующей связи не являются составной частью первичного ключа дочерней сущности, то этот вид связи не проявляется ни в одной идентифицирующей зависимости. В этом случае и ОТДЕЛ, и СОТРУДНИК рассматриваются как независимые сущности.

Тем не менее, взаимосвязь может отражать зависимость существования, если бизнес правило для взаимосвязи определяет то, что внешний ключ не может принимать значение NULL. Если внешний ключ должен существовать, то это означает, что запись в дочерней сущности может существовать только при наличии ассоциированной с ним родительской записи.

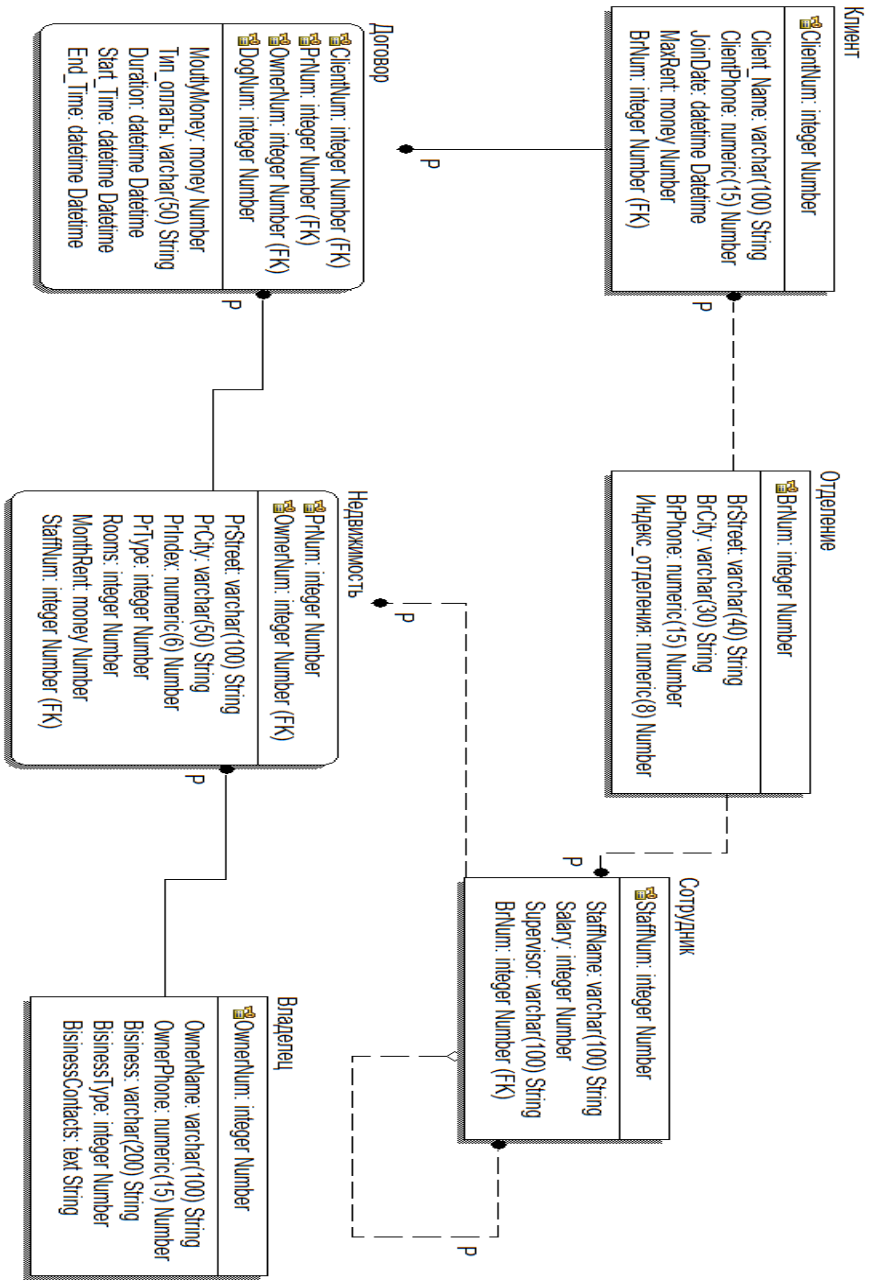
Преимущества IDEF1X

Основным преимуществом IDEF1X, по сравнению с другими многочисленными методами разработки реляционных баз данных, такими как ER и ENALIM является жесткая и строгая стандартизация моделирования. Установленные стандарты позволяют избежать различной трактовки построенной модели, которая несомненно является значительным недостатком ER.

Логическая модель:



Физическая модель:



Описание ограничений :

Validation Name	Validation Rule
FIO Mask	@col LIKE '% % %'
Property Type	@col IN (0, 1, 2)
Slave Rule	@col BETWEEN 5000 AND 100000
Type Money Rule	@col IN (0, 1)

Имя правила : FIO_Mask

Описание : Выражение задается по маске ‘% % %’, обозначает 3 слова через пробел, используется в атрибутах ФИО клиента, сотрудника, владельца.

Имя правила : Property Type

Описание : Выражение принимает значения 0,1,2 , применяется в атрибуте ‘Тип’ сущности ‘Недвижимость’.

Отображение :

Valid Value	Display Value	Definition
0	Квартира	
1	Офис	
2	Дом	

Имя правила : Slave Rule

Описание : Выражение задается в диапазоне от 5000 до 100000, применяется в атрибуте ‘Зарплата’ сущности ‘Сотрудник’.

Имя правила : Type Money Rule

Описание : Выражение принимает значения 0,1 , применяется в атрибуте ‘Тип оплаты’ сущности ‘Договор’.

Отображение :

Valid Value	Display Value	Definition
0	Безналичная	
1	Наличная	