Доклад на тему:
«Создание логической модели данных»

по дисциплине: «Архитектура информационных систем».

Уровни логической модели

Различают три уровня логической модели, отличающихся по глубине представления информации о данных:

- диаграмма сущность-связь (Entity Relationship Diagram, ERD);
- модель данных, основанная на ключах (Key Based model, KB);
- полная атрибутивная модель (Fully Attributed model, FA).

Диаграмма сущность-связь представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям, предъявляемым к ИС. Диаграмма сущность-связь может включать связи "многие-ко-многим" и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области.

Модель данных, основанная на ключах, — более подробное представление данных. Она включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

Полная атрибутивная модель — наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

Сущности и атрибуты

Основные компоненты диаграммы ERwin — это сущности, атрибуты и связи. Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр индивидуален и должен отличаться от всех остальных экземпляров. Атрибут выражает определенное свойство объекта. С точки зрения БД (физическая модель) сущности соответствует таблица, экземпляру сущности — строка в таблице, а атрибуту — колонка таблицы.

Построение модели данных предполагает определение сущностей и атрибутов, т. е. необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте. Сущность можно определить как объект, событие или концепцию, информация о которых должна сохраняться. сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном числе, не носить «технических» наименований и быть достаточно важными для того, чтобы их моделировать. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически имя сущности дается по имени ее экземпляра. Примером может быть сущности Заказчик (но не Заказчики!) с атрибутами Номер заказчика, Фамилия заказчика и Адрес заказчика. На уровне физической модели ей может соответствовать таблица Customer с колонками Customer_number, Customer_name и Customer_address. Каждая сущность должна быть полностью определена с помощью текстового описания. Для внесения дополнительных комментариев и определений к сущности служат свойства, определенные пользователем (UDP). Использование (UDP) аналогично их использованию в BPwin.

Сущности и атрибуты

Как было указано выше, каждый атрибут хранит информацию об определенном свойстве сущности, а каждый экземпляр сущности должен быть уникальным. Атрибут или группа атрибутов, которые идентифицируют сущность, называется первичным ключом .

Очень важно дать атрибуту правильное имя. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Соблюдение этого правила позволяет частично решить проблему нормализации данных уже на этапе определения атрибутов. Например, создание в сущности Сотрудник атрибута Телефоны сотрудника противоречит требованиям нормализации, поскольку атрибут должен быть атомарным, т. е. не содержать множественных значений. Согласно синтаксису IDEFIX имя атрибута должно быть уникально в рамках модели (а не только в рамках сущности!). По умолчанию при попытке внесения уже существующего имени атрибута ERwin переименовывает его.

Каждый атрибут должен быть определен, при этом следует избегать циклических определений, например, когда термин 1 определяется через термин 2, термин 2 — через термин 3, а термин 3 в свою очередь — через термин 1. Часто приходится создавать производные атрибуты, т. е. атрибуты, значение которых можно вычислить из других атрибутов. Примером производного атрибута может служить Возраст сотрудника, который может быть вычислен из атрибута Дата рождения сотрудника. Такой атрибут может привести к конфликтам; действительно, если вовремя не обновить значение атрибута Возраст сотрудника, он может противоречить значению атрибута Дата рождения сотрудника. Производные атрибуты — ошибка нормализации, однако их вводят для повышения производительности системы, чтобы не проводить вычисления, которые на практике могут быть сложными.

Связи

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой. Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы. По умолчанию имя связи на диаграмме не показывается. На логическом уровне можно установить идентифицирующую связь «один-ко-многим», связь «многие-ко-многим» и неидентифицирующую связь «один-ко-многим».

В IDEFIX различают зависимые и независимые сущности. Тип сущности определяется ее связью с другими сущностями. Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Когда рисуется идентифицирующая связь, ERwin автоматически преобразует дочернюю сущность в зависимую. Зависимая сущность изображается прямоугольником со скругленными углами. Экземпляр зависимой сущности определяется только через отношение к родительской сущности. При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ — FK.

При установлении неидентифицирующей связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей.

Идентифицирующая связь показывается на диаграмме сплошной линией с жирной точкой на дочернем конце связи, неидентифицирующая – пунктирной (см. рис. 10.6).

Мощность связей (Cardinality) — служит для обозначения отношения числа экземпляров родительской сущности к числу экземпляров дочерней.

Связи

Различают четыре типа сущности:

- общий случай, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности ; не помечается каким-либо символом;
- символом P помечается случай, когда одному экземпляру родительской сущности соответствуют 1 или много экземпляров дочерней сущности (исключено нулевое значение);
- символом Z помечается случай, когда одному экземпляру родительской сущности соответствуют 0 или 1 экземпляр дочерней сущности (исключены множественные значения);
- цифрой помечается случай точного соответствия, когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.

Имя связи (Verb Phrase) — фраза, характеризующая отношение между родительской и дочерней сущностями . Для связи "один-ко-многим", идентифицирующей или неидентифицирующей, достаточно указать имя, характеризующее отношение от родительской к дочерней сущности (Parent-to-Child). Для связи многие-ко-многим следует указывать имена как Parent-to-Child, так и Child-to-Parent.

Типы сущностей и иерархия наследования

Как было указано выше, связи определяют, является ли сущность независимой или зависимой. Различают несколько типов зависимых сущностей.

- Характеристическая — зависимая дочерняя сущность, которая связана только с одной родительской и по смыслу хранит информацию о характеристиках родительской сущности (рис. 10.7).

Рис. 10.7. Пример характеристической сущности "Хобби"

- Ассоциативная — сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связяхсущностей.
- Именующая — частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрировавших в качестве внешнего ключа).
- Категориальная — дочерняя сущность в иерархии наследования.

Типы сущностей и иерархия наследования

Иерархия наследования (или иерархия категорий) представляет собой особый тип объединения *сущностей*, которые разделяют общие характеристики. Например, в организации работают служащие, занятые полный рабочий день (постоянные служащие), и совместители. Из их общих свойств можно сформировать обобщенную *сущность* (родовой предок) Сотрудник (рис. 10.8), чтобы представить информацию, общую для всех типов служащих. Специфическая для каждого типа информация может быть расположена в категориальных *сущностях* (потомках) Постоянный сотрудник и Совместитель.

Обычно *иерархию наследования* создают, когда несколько *сущностей* имеют общие по смыслу *атрибуты*, либо когда *сущности* имеют общие по смыслу *связи* (например, если бы Постоянный сотрудник и Совместитель имели сходную по смыслу *связь* "работает в" с *сущностью* Организация), либо когда это диктуется бизнес-правилами.

Типы сущностей и иерархия наследования

Для каждой категории можно указать дискриминатор — *атрибут* родового предка, который показывает, как отличить одну категориальную *сущность* от другой (*атрибут* Тип на рис. 10.8).

Рис. 10.8. Иерархия наследования. Неполная категория

Иерархии категорий делятся на два типа — полные и неполные. В полной категории одному экземпляру родового предка (*сущность*Cjn, рис. 10.9) обязательно соответствует экземпляр в каком-либо потомке, т. е. в примере сотрудник обязательно является либо совместителем, либо консультантом, либо постоянным сотрудником.

Типы сущностей и иерархия наследования

Рис. 10.9. Иерархия наследования. Полная категория

Если категория еще не выстроена полностью и в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках, то такая категория будет неполной. На рис. 10.8 показана неполная категория — сотрудник может быть не только постоянным или совместителем, но и консультантом, однако *сущность* Консультант еще не внесена в *иерархию наследования*.

Ключи

Как было сказано выше, каждый экземпляр *сущности* должен быть уникален и должен отличаться от других *атрибутов*.

Первичный ключ (primary key) — это *атрибут* или группа *атрибутов*, однозначно идентифицирующая экземпляр *сущности* . *атрибутыпервичного ключа* на диаграмме не требуют специального обозначения — это те *атрибуты*, которые находятся в списке *атрибутов* выше горизонтальной линии (см., например, рис. 10.9).

В одной *сущности* могут оказаться несколько *атрибутов* или наборов *атрибутов*, претендующих на роль *первичного ключа*. Такие претенденты называются *потенциальными ключами* (candidate key).

Ключи могут быть сложными, т. е. содержащими несколько *атрибутов*. Сложные *первичные ключи* не требуют специального обозначения — это список *атрибутов*, расположенных выше горизонтальной линии.

Ключи

Рассмотрим кандидатов на роль *первичного ключа сущности* Сотрудник (рис. 10.10).

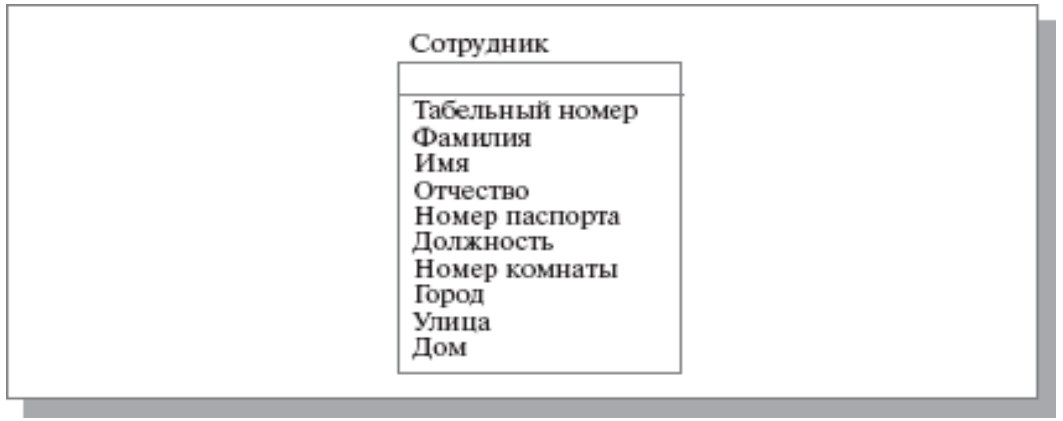

Рис. 10.10. Определение первичного ключа для сущности "Сотрудник"

Здесь можно выделить следующие *потенциальные ключи*:

- Табельный номер ;
- Номер паспорта ;
- Фамилия + Имя + Отчество.

Ключи

Для того чтобы стать первичным, *потенциальный ключ* должен удовлетворять ряду требований:

Уникальность. Два экземпляра не должны иметь одинаковых значений *возможного ключа*. *потенциальный ключ* № 3 (Фамилия + Имя + Отчество) является плохим кандидатом, поскольку в организации могут работать полные тезки.

Компактность. Сложный возможный ключ не должен содержать ни одного *атрибута*, удаление которого не приводило бы к утрате уникальности. Для обеспечения *уникальности ключа* № 3 дополним его *атрибутами* Дата рождения и Цвет волос. Если бизнес-правила говорят, что сочетания *атрибутов* Фамилия + Имя + Отчество + Дата рождения достаточно для однозначной идентификации сотрудника, то Цвет волос оказывается лишним, т. е. ключ Фамилия + Имя + Отчество + Дата рождения + Цвет волос не является компактным.

При выборе *первичного ключа* предпочтение должно отдаваться более простым ключам, т. е. ключам, содержащим меньшее количество *атрибутов*. В приведенном примере ключи № 1 и 2 предпочтительней ключа № 3.

Атрибуты ключа не должны содержать нулевых значений. Значение *атрибутов* ключа не должно меняться в течение всего времени существования экземпляра *сущности*. Сотрудница организации может выйти замуж и сменить как фамилию, так и паспорт. Поэтому ключи № 2 и 3 не подходят на роль *первичного ключа*.

Каждая *сущность* должна иметь по крайней мере один *потенциальный ключ*. Многие *сущности* имеют только один *потенциальный ключ*. Такой ключ становится первичным. Некоторые *сущности* могут иметь более одного *возможного ключа*. Тогда один из них становится первичным, а остальные — *альтернативными ключами*.

*Альтернативный ключ (*Alternate Key) — это *потенциальный ключ*, не ставший первичным.

Нормализация данных

Нормализация данных - процесс проверки реорганизации *сущностей* и *атрибутов* с целью удовлетворения требований к реляционной *модели данных*. Нормализация позволяет быть уверенным, что каждый *атрибут* определен для своей *сущности*, а также значительно сократить объем памяти для хранения информации и устранить аномалии в организации хранения данных. В результате проведения нормализации должна быть создана структура данных, при которой информация о каждом факте хранится только в одном месте. *Процесс нормализации* сводится к последовательному приведению структуры данных к нормальным формам — формализованным требованиям к организации данных. Известны шесть нормальных форм.

На практике обычно ограничиваются приведением данных к *третьей нормальной форме*. Для углубленного изучения нормализации рекомендуется книга К. Дж. Дейта "Введение в системы баз данных" (Киев; М.: *Диалектика*, 1998).

ERwin не содержит полного алгоритма нормализации и не может проводить нормализацию автоматически, однако его возможности облегчают создание нормализованной *модели данных*. Запрет на присвоение неуникальных имен *атрибутов* в рамках модели (при соответствующей установке опции Unique Name) облегчает соблюдение правила "один факт — в одном месте". Имена ролей *атрибутов*внешних ключей и унификация *атрибутов* также облегчают построение нормализованной модели.

Домены

Домен можно определить как совокупность значений, из которых берутся значения *атрибутов* . Каждый *атрибут* может быть определен только на одном *домене*, но на каждом *домене* может быть определено множество *атрибутов*. В понятие *домена* входит не только тип данных, но и область значений данных. Например, можно определить *домен* "Возраст" как положительное целое число и определить *атрибут* Возраст сотрудника как принадлежащий этому *домену*.

В ERwin *домен* может быть определен только один раз и использоваться как в логической, так и в *физической модели*.

Домены позволяют облегчить работу с данными как разработчикам на этапе проектирования, так и администраторам БД на этапе эксплуатации системы. На логическом уровне *домены* можно описать без конкретных физических свойств. На физическом уровне они автоматически получают специфические свойства, которые можно изменить вручную. Так, *домен* "Возраст" может иметь на логическом уровне тип Number, на физическом уровне колонкам *домена* будет присвоен тип INTEGER.

Каждый *домен* может быть описан, снабжен комментарием или свойством, определенным пользователем (UDP).

Тест

1. Какой уровень логической модели не существует?

a) диаграмма сущность-связь (Entity Relationship Diagram, ERD);
b) диаграмма сущность-сущность (Relation-Relation model)
c) модель данных, основанная на ключах (Key Based model, KB);
d) полная атрибутивная модель (Fully Attributed model, FA).

Тест

2. Какой тип сущность не существует?

a) Категориальная
b) Ассоциативная
c) Именная
d) Именующая

Тест

3. Нормализация данных это -

a) процесс проверки реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных
b) последовательное приведение структуры данных к нормальным формам - формализованным требованиям к организации данных
c) процесс удаления неиспользованных данных
d) все вышеперечисленное

Тест

4. Какие варианты ответов не относятся к логической модели?

a) Сущность и атрибуты
b) Ключи и связи
c) Домены
d) Нормальные данные

Тест

5. Что предполагает основу построения логической модели?

a) Определение сущностей и атрибутов
b) Нормализация данных
c) Создание физической модели
d) Определение доменов