

Лекция №10

Информационное обеспечение ИС

Моделирование данных. Метод IDEF1. Отображение модели данных в инструментальном средстве ERwin. Интерфейс ERwin. Уровни отображения модели. Создание логической модели данных: уровни логической модели; сущности и атрибуты; связи; типы сущностей и иерархия наследования; ключи, нормализация данных; домены. Создание физической модели: уровни физической модели; таблицы; правила валидации и значение по умолчанию; индексы; триггеры и хранимые процедуры; проектирование хранилищ данных; вычисление размера БД; прямое и обратное проектирование. Генерация кода клиентской части с помощью ERwin: расширенные атрибуты; генерация кода в Visual Basic. Создание отчетов. Генерация словарей.

Моделирование данных

- Одной из основных частей **информационного обеспечения** является **информационная база**.
- **Информационная база (ИБ)** представляет собой совокупность данных, организованная определенным способом и хранимая в памяти вычислительной системы в виде файлов, с помощью которых удовлетворяются информационные потребности управленческих процессов и решаемых задач.
- Разработка БД выполняется с помощью моделирования данных. **Цель моделирования данных** состоит в обеспечении разработчика ИС концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных.
- Наиболее распространенным **средством моделирования** данных являются **диаграммы "сущность-связь"** (ERD). С помощью ERD осуществляется детализация накопителей данных DFD – диаграммы, а также документируются информационные аспекты бизнес-системы, включая идентификацию объектов, важных для предметной области (**сущностей**), свойств этих объектов (**атрибутов**) и их **связей** с другими объектами (отношений).

Базовые понятия ERD

- **Сущность** (Entity) — множество экземпляров реальных или абстрактных объектов, обладающих общими **атрибутами** или характеристиками. Любой объект системы может быть представлен только одной **сущностью**, которая должна быть уникально **идентифицирована**. При этом имя **сущности** должно отражать тип или класс объекта, а не его конкретный экземпляр.
- Каждая **сущность** должна обладать некоторыми свойствами:
 - иметь уникальное имя; к одному и тому же имени должна всегда применяться одна и та же интерпретация; одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
 - иметь один или несколько **атрибутов**, которые либо принадлежат **сущности**, либо наследуются через **связь**;
 - иметь один или несколько **атрибутов**, которые однозначно идентифицируют каждый экземпляр **сущности**.
- **Связь** (Relationship) — поименованная ассоциация между двумя **сущностями**, значимая для рассматриваемой предметной области. **Связь** — это ассоциация между **сущностями**, при которой каждый экземпляр одной **сущности** ассоциирован с произвольным количеством экземпляров второй **сущности**, и наоборот.
- **Атрибут** (Attribute) — любая характеристика **сущности**, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния **сущности**.
- **Экземпляр атрибута** определяется типом характеристики и ее значением, называемым **значением атрибута**.

Метод IDEF1

- Наиболее распространенными методами для построения ERD-диаграмм являются метод Баркера и метод IDEF1:
 - Метод Баркера основан на нотации, предложенной автором, и используется в case-средстве Oracle Designer.
 - **Метод IDEF1** основан на подходе Чена и позволяет построить *модель данных*, эквивалентную реляционной модели в третьей нормальной форме. На основе совершенствования метода IDEF1 создана его новая версия — метод IDEFIX, разработанный с учетом таких требований, как простота для изучения и возможность автоматизации. IDEFIX-диаграммы используются в ряде распространенных CASE-средств (в частности, ERwin, Design/IDEF).

Метод IDEFI

- В методе IDEFIX *сущность* является независимой от идентификаторов или просто независимой, если каждый экземпляр *сущности* может быть однозначно идентифицирован без определения его отношений с другими *сущностями*. *Сущность* называется зависимой от идентификаторов или просто зависимой, если однозначная идентификация экземпляра *сущности* зависит от его отношения к другой *сущности*

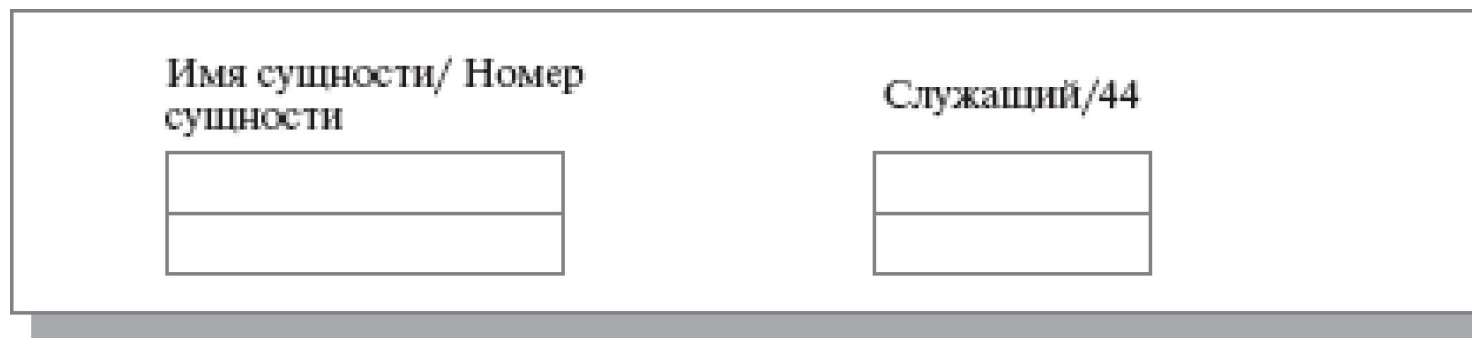


Рис. 10.1. Независимые от идентификации сущности

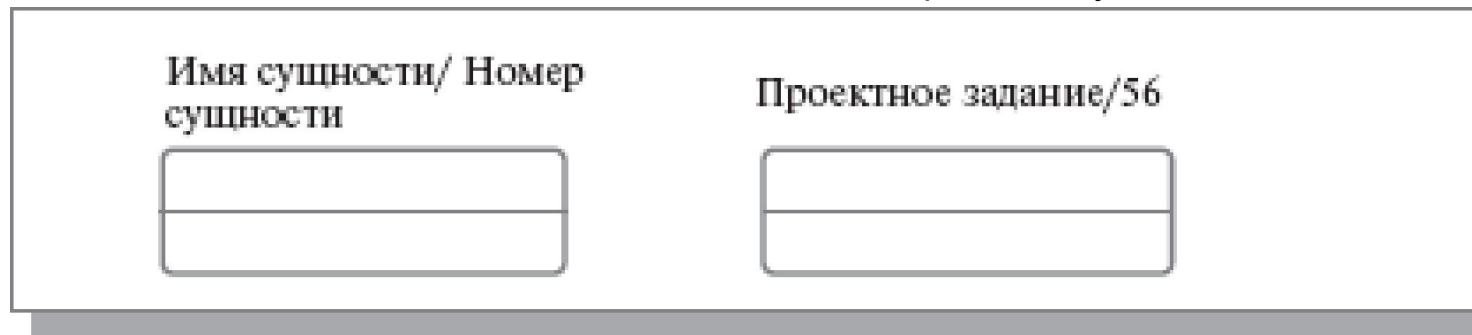


Рис. 10.2. Зависимые от идентификации сущности

Метод IDEFI

- Каждой *сущности* присваиваются уникальные имя и номер, разделяемые косой чертой "/" и помещаемые над блоком.
- *Связь* может дополнительно определяться с помощью указания степени или мощности (количества экземпляров сущности-потомка, которое может породить каждый экземпляр сущности-родителя). В IDEFIX могут быть выражены следующие *мощности связей*: каждый экземпляр сущности-родителя
 - может иметь ноль, один или более одного связанного с ним экземпляра сущности-потомка;
 - должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
 - должен иметь не более одного связанного с ним экземпляра сущности-потомка;
 - связан с некоторым фиксированным числом экземпляров сущности-потомка.
- Если экземпляр сущности-потомка однозначно определяется своей *связью* с сущностью-родителем, то *связь* называется идентифицирующей, в противном случае — не идентифицирующей.

Метод IDEF1

- *Связь* изображается линией, проводимой между сущностью-родителем и сущностью-потомком, с точкой на конце линии у сущности-потомка.
- *Мощность связей* может принимать следующие значения: N — ноль, один или более, Z — ноль или один, P — один или более. По умолчанию *мощность связей* принимается равной N.

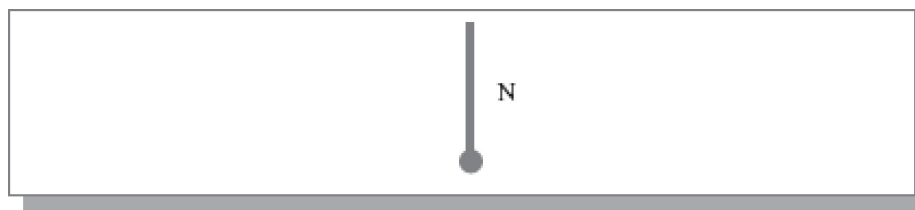


Рис. 10.3. Графическое изображение мощности связи

- Идентифицирующая *связь* между сущностью-родителем и сущностью-потомком изображается сплошной линией. Сущность-потомок в идентифицирующей *связи* является зависимой от идентификатора сущностью. Сущность-родитель в идентифицирующей *связи* может быть как независимой, так и зависимой от идентификатора сущностью (это определяется ее *связями* с другими сущностями).

Метод IDEF1

- Пунктирная линия изображает неидентифицирующую связь. Сущность-потомок в неидентифицирующей связи будет не зависимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей связи.
- *Атрибуты* изображаются в виде списка имен внутри блока *сущности*. *Атрибуты*, определяющие *первичный ключ*, размещаются наверху списка и отделяются от других *атрибутов* горизонтальной чертой.
- *Сущности* могут иметь также **внешние ключи** (Foreign Key), которые могут использоваться в качестве части или целого *первичного ключа* или неключевого *атрибута*.

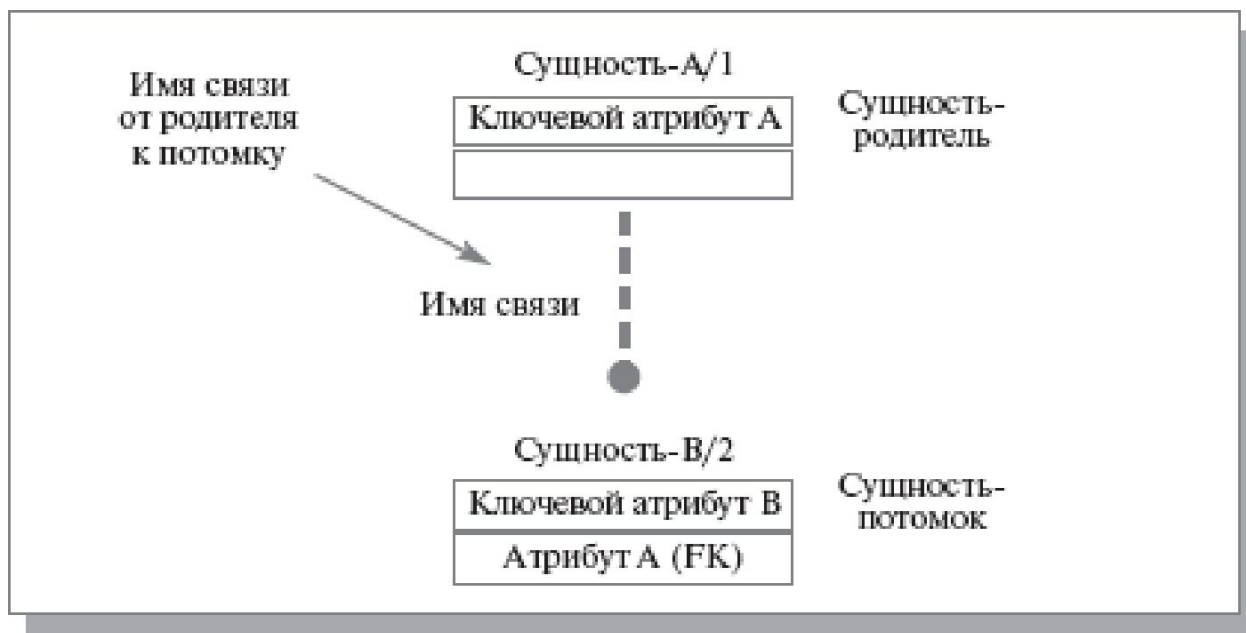


Рис. 10.4. Неидентифицирующая связь

Отображение модели данных в инструментальном средстве ERwin

- ERwin имеет два уровня *представления* модели — логический и физический.
- **Логический уровень** — это абстрактный взгляд на данные, когда данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире, например "Постоянный клиент", "Отдел" или "Фамилия сотрудника". Объекты модели, представляемые на логическом уровне, называются *сущностями* и *атрибутами*. Логическая модель данных может быть построена на основе другой логической модели, например на основе модели процессов. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.
- **Физическая модель данных**, напротив, **зависит от конкретной СУБД, фактически являясь отображением системного каталога**. В физической модели содержится информация обо всех объектах БД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же логической модели могут соответствовать несколько разных физических моделей. Если в логической модели не имеет значения, какой конкретно тип данных имеет *атрибут*, то в физической модели важно описать всю информацию о конкретных физических объектах — таблицах, колонках, *индексах*, процедурах и т.д.

Документирование модели

- Многие СУБД имеют ограничение на именование объектов (например, ограничение на длину имени таблицы или запрет использования специальных символов — пробела и т. п.). Зачастую разработчики ИС имеют дело с нелокализованными версиями СУБД. Это означает, что объекты БД могут называться короткими словами, только латинскими символами и без использования специальных символов (т. е. нельзя назвать таблицу, используя предложение — ее можно назвать только одним словом). Кроме того, проектировщики БД нередко злоупотребляют "техническими" наименованиями, в результате таблица и колонки получают наименования типа `RTD_324` или `CUST_A12` и т.д. Полученную в результате структуру могут понять только специалисты (а чаще всего — только авторы модели), ее невозможно обсуждать с экспертами предметной области.
- Разделение модели на логическую и физическую позволяет решить эту проблему. На физическом уровне объекты БД могут называться так, как того требуют ограничения СУБД. На логическом уровне можно этим объектам дать синонимы — имена более понятные неспециалистам, в том числе на кириллице и с использованием специальных символов. Например, таблице `CUST_A12` может соответствовать сущность Постоянный клиент. Такое соответствие позволяет лучше документировать модель и дает возможность обсуждать структуру данных с экспертами предметной области.

Масштабирование

- Создание *модели данных*, как правило, начинается с разработки логической модели.
- После описания логической модели проектировщик может выбрать необходимую СУБД, и ERwin автоматически создаст соответствующую физическую модель.
- На основе физической модели ERwin может сгенерировать системный каталог СУБД или соответствующий SQL-скрипт. Этот процесс называется *прямым проектированием* (Forward Engineering).
- Тем самым достигается масштабируемость — создав одну *логическую модель данных*, можно сгенерировать физические модели под любую поддерживаемую ERwin СУБД.
- С другой стороны, ERwin способен по содержимому системного каталога или SQL-скрипту воссоздать физическую и *логическую модель данных* (Reverse Engineering).
- На основе полученной *логической модели данных* можно сгенерировать физическую модель для другой СУБД и затем создать ее системный каталог. Следовательно, ERwin позволяет решить задачу по переносу структуры данных с одного сервера на другой.

- Для переключения между логической и *физической моделью данных* служит список выбора в центральной части панели инструментов ERwin
- Если при переключении физической модели еще не существует, она будет создана автоматически.
- Для переключения между логической и *физической моделью данных* служит список выбора в центральной части панели инструментов ERwin
- Если при переключении физической модели еще не существует, она будет создана автоматически.

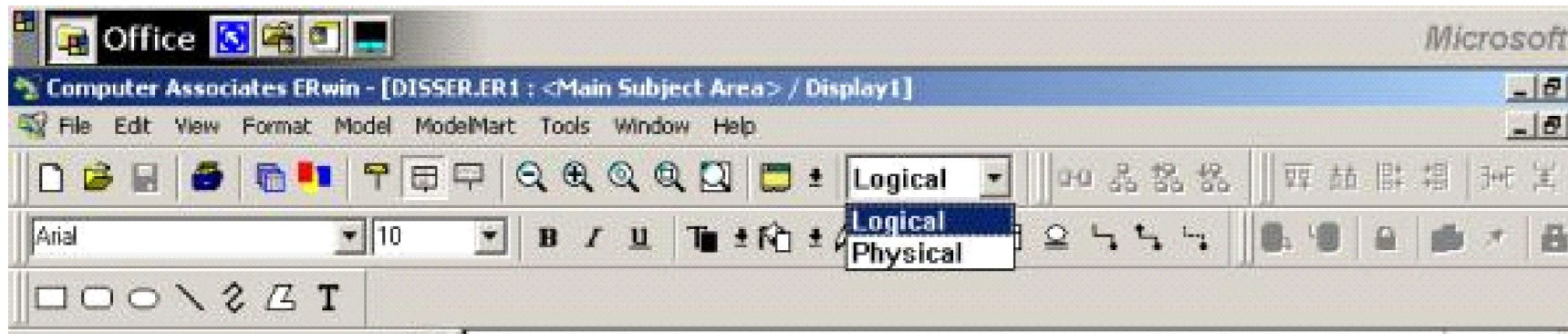
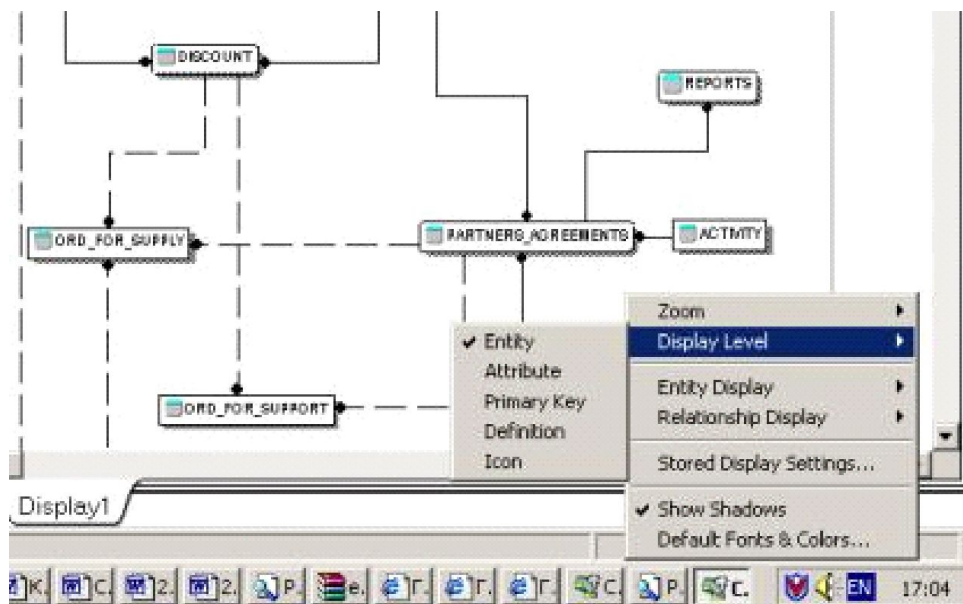


Рис. 10.5. Переключение между логической и физической моделью

Интерфейс ERwin. Уровни отображения модели

- На **логическом уровне** палитра инструментов имеет следующие кнопки:
 - кнопку указателя (режим мыши) — в этом режиме можно установить фокус на каком-либо объекте модели;
 - кнопку внесения *сущности*;
 - кнопку категории (категория, или категориальная *связь*, — специальный тип *связи* между *сущностями*, которая будет рассмотрена ниже);
 - кнопку внесения текстового блока;
 - кнопку перенесения *атрибутов* внутри *сущностей* и между ними;
 - кнопки создания *связей*: идентифицирующую, "многие-ко-многим" и неидентифицирующую.
- На **физическом уровне** палитра инструментов имеет:
 - вместо кнопки категорий — кнопку внесения *представлений* (view);
 - вместо кнопки *связи* "многие-ко-многим" — кнопку *связей представлений*.



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.
- *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.
- Каждый *атрибут* хранит информацию об **определенном свойстве сущности**, а каждый экземпляр *сущности* должен быть уникальным. *Атрибут* или группа *атрибутов*, которые идентифицируют *сущность*, называется **первичным ключем**.

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

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



- **Ассоциативная** — сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущностей.
- **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** представляет собой особый тип объединения сущностей, которые разделяют общие характеристики.
- Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи, либо когда это диктуется бизнес-правилами.
- Для каждой категории можно указать дискриминатор — атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой.



Рис. 10.8. Иерархия наследования. Неполная категория

Типы сущностей и иерархия наследования

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



Рис. 10.9. Иерархия наследования. Полная категория

Ключи

- **Первичный ключ** (primary key) — это *атрибут* или группа *атрибутов*, однозначно идентифицирующая экземпляр *сущности*. *атрибуты* *первичного ключа* на диаграмме не требуют специального обозначения — это те *атрибуты*, которые находятся в списке *атрибутов* выше горизонтальной линии.
- В одной *сущности* могут оказаться несколько *атрибутов* или наборов *атрибутов*, претендующих на роль *первичного ключа*. Такие претенденты называются **потенциальными ключами** (candidate key).
- **Альтернативный ключ** (Alternate Key) — это *потенциальный ключ*, не ставший *первичным*.



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

Нормализация данных

- Нормализация данных — процесс проверки и 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.
- ERwin не содержит полного алгоритма нормализации и не может проводить нормализацию автоматически, однако его возможности облегчают создание нормализованной *модели данных*.

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

Создание физической модели данных

- Физическая модель содержит всю информацию, необходимую для реализации конкретной БД. Различают два уровня физической модели:
 - трансформационную модель;
 - модель СУБД.
- Трансформационная модель содержит информацию для реализации отдельного проекта, который может быть частью общей ИС и описывать подмножество предметной области. Данная модель позволяет проектировщикам и администраторам БД лучше представить, какие объекты БД хранятся в словаре данных, и проверить, насколько физическая модель удовлетворяет требованиям к ИС.
- **Модель СУБД** автоматически генерируется из трансформационной модели и является точным отображением системного каталога СУБД.
- Физический уровень *представления* модели зависит от выбранного сервера. ERwin поддерживает более 20 реляционных и нереляционных БД.
- По умолчанию ERwin генерирует имена таблиц и *индексов* по шаблону на основе имен соответствующих *сущностей* и ключей логической модели, которые в дальнейшем могут быть откорректированы вручную. Имена таблиц и колонок будут сгенерированы по умолчанию на основе имен *сущностей* и *атрибутов* логической модели.

Правила валидации и значения по умолчанию

- ERwin поддерживает *правила валидации* для колонок, а также значение, присваиваемое колонкам по умолчанию.
- *Правило валидации* задает список допустимых значений для конкретной колонки и/или правила проверки допустимых значений. В список допустимых значений можно вносить новые значения. ERwin позволяет сгенерировать *правила валидации* соответственно синтаксису выбранной СУБД с учетом границ диапазона или списка значений.
- **Значение по умолчанию** – значение, которое нужно ввести в колонку, если никакое другое значение не задано явным образом во время ввода данных. С каждой колонкой или *доменом* можно связать значение по умолчанию. Список значений можно редактировать.
- После создания *правила валидации* и значения по умолчанию их можно присвоить одной или нескольким колонкам или *доменами*.

Индексы

- В БД данные обычно хранятся в том порядке, в котором их ввели в таблицу. Многие реляционные СУБД имеют страничную организацию, при которой таблица может храниться фрагментарно в разных областях диска, причем строки таблицы располагаются на страницах неупорядоченно. Такой способ позволяет быстро вводить новые данные, но затрудняет поиск данных.
- Чтобы решить проблему поиска, СУБД используют объекты, называемые *индексами*. **Индекс** содержит отсортированную по колонке или нескольким колонкам информацию и указывает на строки, в которых хранится конкретное значение колонки. Поскольку значения в *индексе* хранятся в определенном порядке, при поиске просматривать нужно значительно меньший объем данных, что существенно уменьшает время выполнения запроса. *Индекс* рекомендуется создавать для тех колонок, по которым часто производится поиск.
- При генерации схемы физической БД ERwin автоматически создает *индекс* на основе *первичного ключа* каждой таблицы, а также на основе всех *альтернативных ключей* и внешних ключей, поскольку эти колонки наиболее часто используются для поиска данных. Можно отказаться от генерации *индексов* по умолчанию и создать собственные *индексы*. Для увеличения эффективности поиска администратор БД должен анализировать часто выполняемые запросы и на основе анализа создавать собственные *индексы*.

Триггеры и хранимые процедуры

- **Триггеры и хранимые процедуры** – это именованные блоки кода SQL, которые заранее откомпилированы и хранятся на сервере для того, чтобы быстро производить обработку запросов, валидацию данных и другие часто выполняемые функции.
- Хранение и выполнение кода на сервере позволяет создавать код только один раз. *Триггеры и хранимые процедуры* не требуется пересылать по сети из клиентского приложения, что значительно снижает сетевой трафик.
- **Хранимой процедурой** называется именованный набор предварительно откомпилированных команд SQL, который может вызываться из клиентского приложения или из другой *хранимой процедуры*.
- **Триггером** называется процедура, которая выполняется автоматически как реакция на событие.
- **Триггер ссылочной целостности** – это особый вид *триггера*, используемый для поддержания целостности между двумя таблицами, которые связаны между собой.
- Для генерации *триггеров* ERwin использует механизм шаблонов – специальных скриптов, использующих макрокоманды.
- Для создания и редактирования *хранимых процедур* ERwin располагает специальными редакторами, аналогичными редакторам, используемым для создания *триггеров*.

Проектирование хранилищ данных

- В хранилища данных помещают данные, которые редко меняются. Хранилища ориентированы на выполнение аналитических запросов, обеспечивающих поддержку принятия решений для руководителей и менеджеров.
- При проектировании хранилищ данных необходимо выполнять следующие требования:
 - хранилище должно иметь понятную для пользователей структуру данных;
 - должны быть выделены статические данные, которые модифицируются по расписанию (ежедневно, еженедельно, ежеквартально);
 - должны быть упрощены требования к запросам для исключения запросов, требующих множественных утверждений SQL в традиционных реляционных СУБД;
 - должна обеспечиваться поддержка сложных запросов SQL, требующих обработки миллионов записей.

Прямое и обратное проектирование

- **Прямым проектированием** называется процесс генерации физической схемы БД из логической модели. При генерации физической схемы ERwin включает *триггеры* ссылочной целостности, *хранимые процедуры, индексы*, ограничения и другие возможности, доступные при определении таблиц в выбранной СУБД.
- **Обратным проектированием** называется процесс генерации логической модели из физической БД. *Обратное проектирование* позволяет конвертировать БД из одной СУБД в другую. После создания логической модели БД путем *обратного проектирования* можно переключиться на другой сервер и произвести *прямое проектирование*.
- Кроме режима прямого и *обратного проектирования* программа обеспечивает синхронизацию между логической моделью и системным каталогом СУБД на протяжении всего жизненного цикла создания ИС.
- **Вычисление размера БД**
ERwin позволяет рассчитать приблизительный размер БД в целом, а также таблиц, *индексов* и других объектов через определенный период времени после начала эксплуатации ИС. Расчет строится на основе следующих параметров: начальное количество строк; максимальное количество строк; прирост количества строк в месяц. Результаты расчетов сводятся в отчет.

Генерация кода клиентской части с помощью ERwin

- **Расширенные атрибуты**

ERwin поддерживает не только проектирование сервера БД, но и автоматическую генерацию клиентского приложения в средах разработки MS Visual Basic и Power Builder.

Каждой колонке в модели ERwin можно задать предварительно описанные и именованные свойства:

- *правила валидации* (проверка значений);
- начальные значения, устанавливаемые по умолчанию;
- стиль визуального объекта (например, радиокнопка, поле ввода и др.);
- формат изображения.

- **Генерация кода в Visual Basic**

ERwin поддерживает генерацию кода в Visual Basic версий 4.0 и 5.0. В качестве источника информации при генерации форм служит модель ERwin.

- **Создание отчетов**

Для генерации отчетов в ERwin имеется простой и эффективный инструмент – Report Browser.

- **Генерация словарей**

Для управления большими проектами ERwin имеет специальный инструмент – ERwin Dictionary, который обеспечивает коллективную работу над диаграммами и позволяет сохранять и документировать различные версии *моделей данных*.