МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационных систем и технологий

УТВЕРЖДАЮ:
Проректор по учебной работе
_____ О.Г. Локтионова
«15» 12 2020 г.

СИСТЕМЫ АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ

методические указания по выполнению лабораторной работы №6
по дисциплине «Информационные технологии»
для студентов направления подготовки
10.03.01 - «Информационная безопасность»

Курск 2020

УДК 004.94

Составитель: Л.В. Стародубцева

Рецензент

Кандидат технических наук, доцент *Ю.А. Халин*

Системы автоматизации проектирования: методические указания по выполнению лабораторной работы №6 / Юго-Зап. гос. ун-т; сост. Л.В. Стародубцева. Курск, 2020. 13 с.

Содержит теоретические сведения по дисциплине «Информационные технологии». Указывается порядок выполнения лабораторной работы.

Методические указания по структуре, содержанию и стилю изложения материала соответствуют методическим и научным требованиям, предъявляемым к учебным и методическим пособиям.

Предназначены для студентов направления подготовки 10.03.01- «Информационная безопасность» очной формы обучения.

Текст печатается в авторской редакции

Подписано в печать 15.12.20. Формат 60x84 1/16.

Усл.печ.л. 0,7. Уч.-изд. л. 0,6. Тираж 40 экз. Заказ. 360 Бесплатно.

Юго-Западный государственный университет.

305040, г. Курск, ул. 50 лет Октября, 94.

Лабораторная работа № 6

Тема: Системы автоматизации проектирования

Цель работы: научиться моделировать информационное обеспечение

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*).

Теоретические сведения

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, *ER*-диаграмма 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 - 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 сущность 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, Customerjname* и *Customer'*_address.

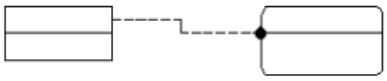
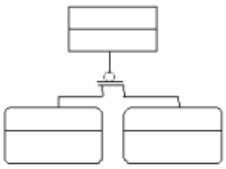
*Связи***.** В стандарте *IDE F1X* определены типы связей, показанные в табл.1.

Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой (*Relationship Verb Phrases*). Имя связи выражает некоторое ограничение или бизнес-правило и облегчает чтение диаграммы, например:

Каждый КЛИЕНТ <размещает> ЗАКАЗЫ;

Каждый ЗАКАЗ <выполняется> СОТРУДНИКОМ.

Таблица 1 - Типы связей стандарта IDEF1X

	Идентифицирующая связь
	Неидентифицирующая связь
	Рекурсивная связь
	Связь типа “многие-ко-многим”
	Связь супертипа с подтипами

Связь показывает, какие именно заказы разместил клиент и какой именно сотрудник выполняет заказ. По умолчанию имя связи на диаграмме не показывается. Для отображения имени следует в контекстном меню, которое появляется, если щелкнуть левой кнопкой мыши по любому месту диаграммы, не занятому объектами модели,

выбрать пункт *Display Options/Relationship* и затем включить опцию *Verb Phrase*.

Рисунок 1 - Имя связи - *Relationship Verb Phrases*

На логическом уровне можно установить идентифицирующую связь "один-ко-многим", связь "многие-ко-многим" и неидентифицирующую" связь "один-ко-многим" (соответственно это кнопки слева направо в палитре инструментов).

В *IDEF1X* различают зависимые и независимые сущности. Тип с ущности 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 (сущность Заказ на рис. 2). Экземпляр зависимой сущности определяется только через отношение к родительской сущности, т.е. в структуре на рис. 1 информация о заказе не может быть внесена и не имеет смысла без информации о клиенте, который его размещает. 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*).

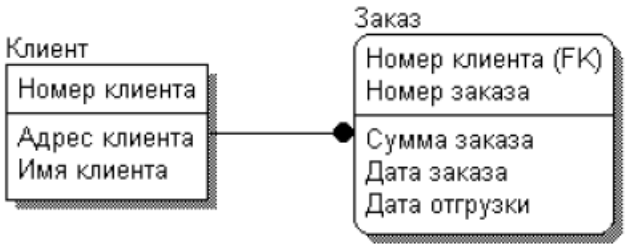


Рисунок 2 - Идентифицирующая связь между независимой и зависимой таблицей

В дальнейшем, при генерации схемы БД, атрибуты первичного ключа получат признак *NOT NULL*, что означает невозможность внесения записи в таблицу заказов без информации о номере клиента.

При установлении неидентифицирующей связи (рис. 3) дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности. Неидентифицирующая связь служит для связывания независимых сущностей.

Экземпляр сущности Сотрудник может существовать безотносительно к какому-либо экземпляру сущности Отдел, т.е. сотрудник может работать в организации, не числясь в каком-либо отделе.

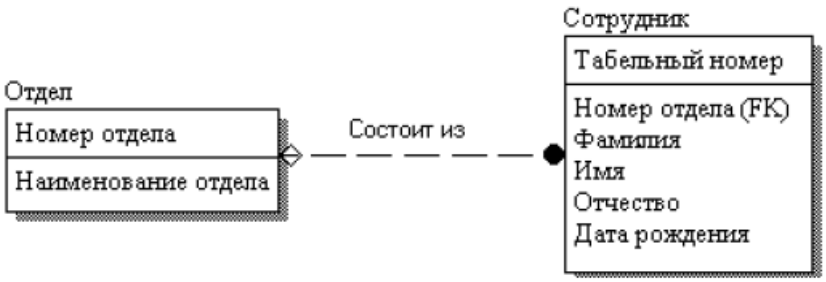


Рисунок 3 - Неидентифицирующая связь

Идентифицирующая связь показывается на диаграмме сплошной линией

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. 4).

Рисунок 4 - Пример характеристической сущности "Хобби"

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.

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. 4), 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 бы сходную по смыслу связь "работает в" с сущностью Организация), либо когда это диктуется бизнес-правилами.

Для каждой категории можно указать дискриминатор - атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой (атрибут "Тип" на рис. 5).

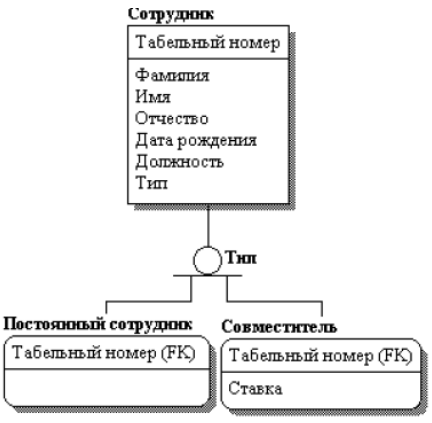


Рисунок 5 - Иерархия наследования. Неполная категория

Иерархии категорий делятся на два типа - полные и неполные. В полной категории одному экземпляру родового предка (сущность Служащий, рис. 6) обязательно соответствует экземпляр в каком-либо потомке, т. е. в примере служащий обязательно является либо совместителем, либо консультантом, либо постоянным сотрудником.

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. 5 показана неполная категория - сотрудник может быть не только постоянным или совместителем, но и консультантом, однако сущность Консультант еще не внесена в иерархию наследования.

Возможна комбинация полной и неполной категорий. На рис. 7 помимо постоянных сотрудников и совместителей могут быть и консультанты, что не отражено в иерархии (неполная категория), но каждый постоянный сотрудник

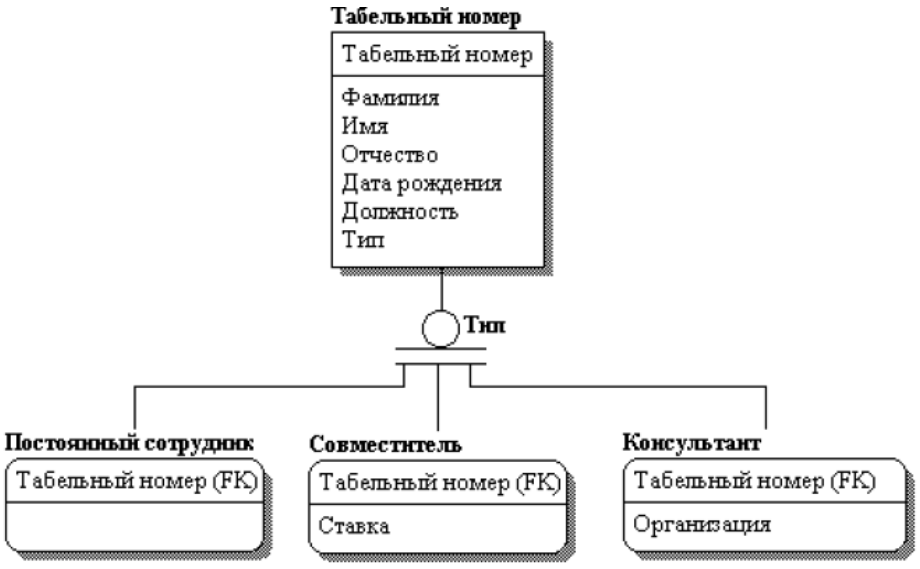


Рисунок 6 - Иерархия наследования. Полная категория

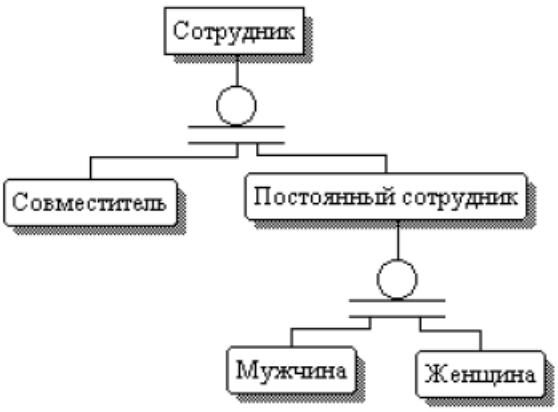


Рисунок 7 - Иерархия наследования. Комбинация полной и неполной категорий

*Ключи***.** Каждый экземпляр сущности должен быть уникален и отличаться от других атрибутов. 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. При внесении нового атрибута в диалоге *Attribute Editor* для того, чтобы сделать его атрибутом первичного ключа, нужно включить флажок *Primary Key* в нижней части закладки *General*. На диаграмме неключевой атрибут можно внести в состав первичного ключа, воспользовавшись режимом переноса атрибутов (кнопка в палитре инструментов).

Выбор первичного ключа может оказаться непростой задачей, решение

которой может повлиять на эффективность будущей ИС. 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 выше горизонтальной линии.

Атрибуты ключа не должны содержать нулевых значений. Если допускается, что сотрудник может не иметь паспорта или вместо паспорта иметь какое-либо другое удостоверение личности, то ключ № 2 не подойдет на роль первичного ключа. Если для обеспечения уникальности необходимо дополнить потенциальный ключ дополнительными атрибутами, то они не должны содержать нулевых значений. Дополняя ключ № 3 атрибутом Дата рождения, нужно убедиться в том, что даты рождения известны для всех сотрудников.

Значение атрибутов ключа не должно меняться в течение всего времени существования экземпляра сущности. 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.

Задание

Спроектируйте логическую модель БД (прямое моделирование). Задайте атрибуты для каждой определенной сущности. Введите связи между сущностями. Присвойте связям уникальные имена.

Контрольные вопросы

1. В чем состоит различие логического и физического уровней представления моделей данных с помощью ERwin?

2. В чем различие между моделями данных, представленных в форме диаграммы сущьность-связь, на основе ключей и в виде полной атрибутивной модели?

3. Какие основные компоненты содержат модели данных, представленные по методологии *IDEF1X*?

4. Какие типы связей между сущностями определены в стандарте *IDEF1X*?

5. Какие типы сущностей и иерархии наследования Вы знаете?

6. Каким требованиям должен отвечать первичный ключ?

7. Какой процесс называется прямым и обратным проектированием?